A discovery of neural network architectures for context-dependent computations

Doris Voina

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2022

Reading Committee:

Eric Shea-Brown, Chair

Stefan Mihalas

Jose Kutz

Program Authorized to Offer Degree:
Applied Mathematics

University of Washington

Abstract

A discovery of neural network architectures for context-dependent computations

Doris Voina

Chair of the Supervisory Committee:
Professor Eric Shea-Brown
Applied Mathematics

All human and animal behavior from seeing, hearing, running, and falling in love, is the result of complex dynamics in a web of intricate networks in the brain. The human brain, in particular, contains close to 100 billion brain cells (or neurons) of different types connected through more than 100 trillion connections (or synapses), often in complicated patterns (or motifs) that depend on the brain area and function of the network. How these neurons and synapses are organized into specific network architectures so that neuronal activity and dynamics can give rise to behavior is still a mystery. A similar problem exists in the case of artificial neural networks: there is no systematic approach to designing artificial network architectures that generalize well across tasks, conditions, and contexts. For artificial and biological networks alike, we are interested in understanding the building blocks that permit a broad array of neural network functionality to emerge. We approach this problem from several perspectives: 1) we show how a biologically inspired microcircuit with several specific features (multiple inhibitory cell types, a comparatively smaller neuron population recurrently connected to the network that acts in a switch-like manner, and a disinhibitory network motif) is a minimally complex architecture that can switch between visual processing of the static context and the moving context; 2) we find a fast and flexible artificial network with a biologically-inspired network motif that generalizes across context when classifying visual stimuli shown sequentially and with different background contexts; 3) we begin the process of identifying new, bio-inspired network motifs via methods that identify

network motifs that inform neuron type classification. Our work clarifies the set of network connection structures that are both necessary and sufficient to achieve more flexible computational capability in both biological and artificial neural networks.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS

My gratitude goes first and foremost to my mentors, Stefan Mihalas and Eric Shea-Brown, who have been incredibly supportive and encouraging throughout my Ph.D. They were always open to my ideas and provided me with an intellectually free and stimulating environment to grow as a scientist. I have been lucky to take advantage of Eric's infectious positivity and Stefan's creative suggestions.

The Computational Neuroscience Center is an excellent community and I have taken advantage of the many (formal and informal) talks, lectures, discussions, and journal clubs. Adrienne Fairhall is an integral part of this community and I am grateful for the inspiring role model she continues to be as a woman in science. Her thoughtful observations are always a reminder of the high caliber of people we have the immense privilege to be around. I also thank Jessica Huszar for running the center and ensuring all goes smoothly.

I'm thankful for all the Ph.D. and Masters's students in the Applied Math and in the Physiology and Biophysics departments. What a joy and inspiration it has been to work with you! It has been an intellectually stimulating few years, indeed. I am grateful I was able to forge close friendships, like with Jithin and Mathias, while busying myself with academic work. I have a few great role models as well, like Kameron Harris and Alison Weber, who are continuing to inspire me as scientists and just overall great people.

The Applied Math department has been supportive as well during all stages of my Ph.D, from coming to Seattle as a Master's student, to finally defending my thesis this year. I'm grateful to Bernard Deconinck for being understanding and encouraging when I took time off from the program. It felt stressful, but Bernard helped me put things in perspective. I

am forever grateful to the staff in Applied Math for their tireless work. Lauren Lederer has been an amazing resource throughout these years and I am incredibly lucky to have her as a colleague as I move to a different department. Tony Garcia and Katherine McDermott have helped me a great deal as well. Katherine has been invaluable in ensuring I was able to do an internship during the last Spring quarter of my program.

Members of my committee – Amy Orsborn, Sasha Aravkin, Adrienne Fairhall, Nathan Kutz, along with my advisers – have been invaluable in challenging me and asking me all the right questions. I am grateful for their time, interest, and curiosity.

Outside of school, I am thankful to be part of a few wonderful communities. One is the hiking community I joined in 2019. Here in the Pacific Northwest, I have been blessed with beautiful nature and the possibility to go on many unforgettable hiking and camping trips. The other is the community I put together centered on mindfulness meditation and the study of non-dualistic traditions. My interest in understanding the brain can be traced back to a beautiful non-dual experience I had while in college, when I realized how limited our normal, everyday conscious experience was and what immense potential we all have within.

I'm also grateful for the wonderful friendships that I've forged throughout the years. Braiden, thank you for caring and being by my side during my qualifying exams. Lauren, I'm so glad we're keeping in touch!

I am thankful to the people who have walked with me every step of the way, even when it was hard and sometimes chaotic. Andrey, thank you from the bottom of my heart for being there and supporting me.

Last but not least, my family has been ever-present and always loving and supportive.

Daisy has been precious in these past few months, offering me the unconditional love I was longing for. All this work would have been impossible without my parents, Lili and Mircea. Despite our differences, I have always felt loved and cared for. Lili has always been by my side, even in very difficult times when I struggled. My grandparents have been there for me through their kindness and care. I know I was always on my grandmother Ana's mind. I dearly miss my grandfather who sadly passed away at the age of 87, just a few months short of seeing me graduate. How proud he would have been!

I am proud to be an Applied Mathematician and Computational Neuroscientist, but my accomplishments would have been impossible without my mentors, my friends, my family, and the incredible community at the University of Washington and in Seattle!

DEDICATION

to Daisy

Chapter 1

INTRODUCTION

Understanding the brain has fascinated humans for millennia, from the Roman physician Galen, who in 170 BC suggested that the brain was the seat of complex thought and determined personality and bodily functions, to today's neuroscientists who rigorously and systematically study the nervous system at different scales to uncover the biological basis of learning, memory, behavior, perception, and consciousness. The effort to explain how the brain is capable of a broad range of behaviors and unparalleled feats of cognition has spawned a revolution of sophisticated techniques for visualization and measurement: imaging tools like magnetic resonance imaging and magnetoencephalography,and optical, electrical, and chemical methods to explore brain functionality by allowing both measurement and manipulation of thousands or even millions of brain cells [24]. Simultaneous with these experimental achievements are advances in network-based approaches, statistics, models, and theories that can shed light on hidden mechanisms employed by the brain. My particular interest is in computational neuroscience, where mathematical tools and intuitions are developed along with computer simulations to understand the principles that govern the development, structure, physiology, and cognitive abilities of the nervous system. Neuroscientist Eric Kandel has described these concerted efforts to understand the brain as the "epic challenge" of the biological sciences: "The last frontier of the biological sciences – their ultimate challenge – is to understand the biological basis of consciousness and the mental processes by which we perceive, act, learn, and remember." [17].

While many important scientific inquiries regarding the brain's complex mechanisms are currently underway, a distinct but related field – deep learning – has made extraordinary advances in the past decade. Deep learning is part of a larger family of machine learning methods based on artificial neural networks. The number of current applications for

artificial neural networks is extensive and ever-expanding, from processing and identifying images of objects, scenes, and people; to parsing, recognizing, and responding to written or spoken language; to planning and making decisions in complex settings.

There is a natural relationship between deep learning and neuroscience: deep learning's artificial neural networks are loosely inspired by the brain. To see why taking inspiration from the brain's mechanisms could be advantageous, suppose one attempted to automate the process of recognizing handwritten digits (Figure 1.1a). Such an algorithm could be used for instance by banks to identify handwritten digits on checks. Designing an algorithm to solve this problem is surprisingly difficult. Applying simple intuitions like "a 9 has a loop at the top, and a vertical stroke in the bottom right" seems simple enough, but when attempting to make such rules precise, one quickly runs into a morass of exceptions and special cases. Expressing our own intuitions about what constitutes a 9 algorithmically appears to be an inefficient solution, as exemplified by the logical processing systems of the 1970s and 1980s based on serial computation, an approach inspired in part by the notion that human intelligence involves the manipulation of symbolic representations [12]. However, these purely symbolic approaches appeared to be too brittle and inflexible to solve complex real-world problems of the kind that humans routinely handle. Artificial neural networks succeed where symbolic approaches fail, and the crucial insight of ANNs is to learn the connections between units in the neural network in a way that roughly mimics the brain.

Indeed, recognizing handwritten digits like the ones in Figure 1.1a (up), or the natural images from the ImageNet dataset shown below is something humans are very adept at. This ease is misleading. To recognize the categories in Figure 1.1a, hundreds of millions of neurons in the visual cortex with billions of connections between them process the information relayed through the retina and, through a complex transformation, yield the correct identity of what is shown in the image. These neurons and their connections have been tuned by evolution over millions of years, in order to seamlessly recognize objects, from simple handwritten digits to more complex stimuli like faces. This suggests that the brain employs a type of *computation*, in that it is able to perform information processing on dif-

ferent variables or entities following specific rules, in order to produce a specified output. The brain can be thought of as performing computations in as much as it is a system whose components must be functionally organized to process the information in accordance with an established set of rules. While still under debate, the notion that, through neuromodulation and the concerted activity of populations of neurons, the brain *computes* on its inputs has been more widely accepted.

Therefore, studying the computational mechanisms of the brain becomes of paramount importance not just to further our scientific knowledge, but also to inform future computational systems and guide the design of Artificial Intelligence (AI) architectures.

1.1 Neuroscience 101

According to our current understanding, the primary processing unit of the brain is the neuron. Neurons generate characteristic electrical pulses called action potentials, or more simply, spikes which occur when they receive a strong enough signal from their incoming connections, and the membrane potential increases to a precisely defined threshold voltage. Importantly, incoming signals are not simply summed, but combined in a complex and non-linear fashion.

Neurons are highly heterogeneous and have been so far classified into distinct cell types according to their location in the nervous system and their distinct shape (e.g., basket cells, Purkinje cells, pyramidal cells, etc.). A beautiful illustration of these cells can be seen in Figure 1.1b, which depicts the neurons in the chick cerebellum – one of the many drawings by Santiago Ramon y Cajal, a pioneer of modern neuroscience. While the exact computational role of these multiple cell types is currently under debate, we do know that neurons are highly interconnected to each other, sending signal pulses along outgoing connections (or synapses) to other neurons. Moreover, neurons form neuronal populations that are the main channels for information propagation across the brain. This passing of information signals between neurons and populations of neurons is presumed to be the brain's main means for computation. More precisely, it is assumed that by firing sequences of spikes in diverse temporal patterns, neurons represent and transmit information. On a broader

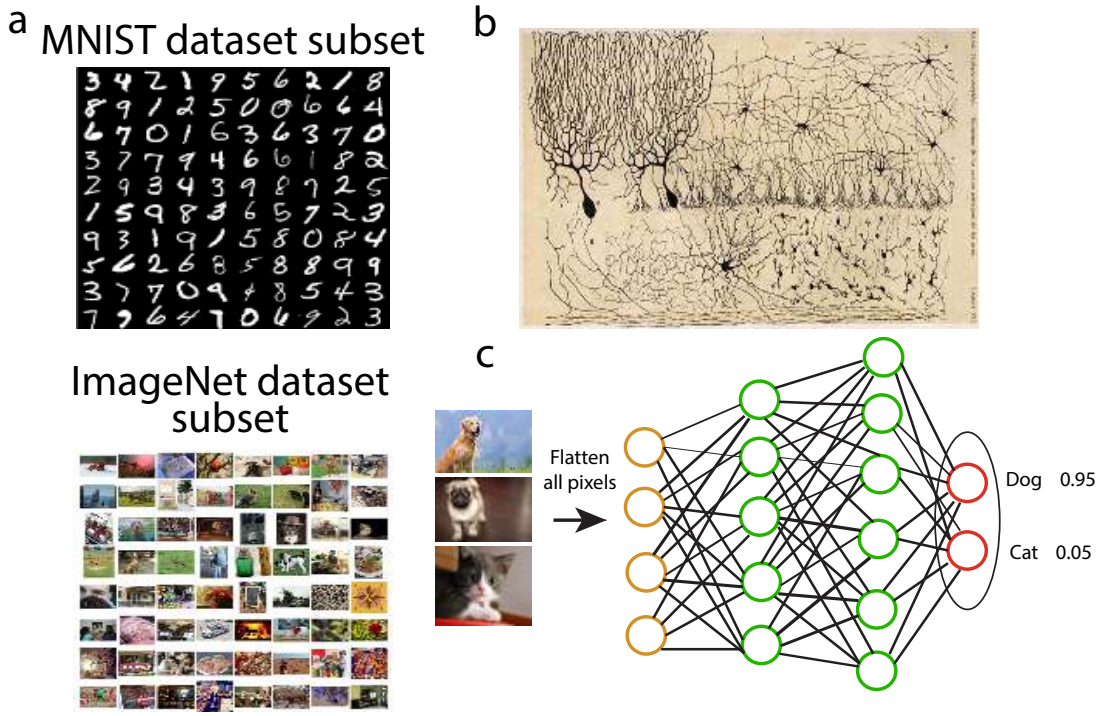


Figure 1.1: (a) Top: MNIST (Modified National Institute of Standards and Technology, from https://theanets.readthedocs.io/en/stable/examples/mnist-classifier.html) dataset and bottom: ImageNet dataset used in visual object recognition (from [34]). MNIST is a dataset of digits used especially in the early days of research on neural networks. ImageNet is a large visual database, with more than 14 million images that have been hand-annotated with appropriate labels (i.e. dog, cat, human, car, etc.). Although ImageNet contains more than $20,000$ categories, most research uses a trimmed version of 1000 categories. (b) Illustration of the cells of the chick cerebellum by Santiago Ramón y Cajal, the Spanish neuroscientist, from the book "Estructura de los centros nerviosos de las aves," Madrid, circa 1905.(c) Artificial neural network with one hidden layer that can classify images depending on whether a cat or a dog appears in the image.

scale, the vast interconnectedness of neurons, along with their signal transmission through spikes, is thought to generate the mind's functional states. The study of neural encoding describes how diverse stimulus attributes such as light, sound intensity, or motor actions are represented through sequences of spikes.

When an animal learns any particular behavior, essentially what happens is that the strength of connections between neurons is modified. Stronger connections represent larger signals transmitted during spiking. The most well-known rule by which connections are modified is the basic Hebbian learning rule [13], where the inter-neuron connection is strengthened when both neurons spike sufficiently close together in time. Many other rules have been discovered [2, 32, 45], but it is still an open problem as to how the brain adjusts connections in order to learn.

Brain network structure operates at different scales, starting from proteins and protein interaction networks within neurons, to individual neuron biophysics and the wiring of synaptic connectivity between neurons, and finally to brain regions and the mesoscale networks connecting these. Using hierarchical network or other theoretical models, the hope is that through this cross-scale integration we can understand how the architecture of connectivity at each scale emerges from the scale below [24]. In addition, the physical circuitry at all these scales supports complex dynamics given by the inter-neuron signaling, so it is not enough to have a complete description of the brain's structural wiring, it must be coupled with an understanding of how neuronal activity and connectivity evolves over time. Discovering fundamental principles of computation using knowledge and/or models of network architecture and dynamics will further our understanding of this fascinating organ and contribute to therapies for neurological diseases and psychiatric disorders.

1.2 Models of the brain

New data from human and animal brains is accrued every year. Crucially, the empirical data now available has shifted from largely single-neuron or small ensemble recordings to high-precision and large-scale recordings of hundreds to thousands of neurons. This is

often conducted in vivo, and uses a variety of optical imaging (Calcium imaging, wide-field microscopy, 2 photon microscopy) and electrophysiological approaches (silicon-based electrodes, including next-generation high-density probes like Neuropixels) [33]. Discovering the brain's organizing principles from this vast array of data is no easy task because of the number of neurons active and the ultra-high complexity of structural connectivity, both of which change in response to external stimuli and internal processes. Furthermore, the brain's dynamics happen at multiple time scales and in a hierarchical fashion, with the bidirectional connectivity between different hierarchical levels being continuously reconfigured due to behavioral demands and neuromodulation. In physics terms, the brain is an *adaptive complex dynamic system* [41]. In such systems, the behavior of the whole cannot be predicted from the dynamics of its components, and they are characterized by the self-organization of a rich repertoire of dynamical states and the transitions between them. These complexities have motivated rigorous theory, modeling, and statistics (all part of "Computational" and "Theoretical" Neuroscience) to advance our understanding of the highly non-linear processes happening in the brain whenever our intuition fails [41].

There are a few ways to classify recent computational models. First, we have descriptive models, normative theories, and mechanistic (or biologically realistic) models. Descriptive models quantitatively match neuronal data (e.g., signal processing algorithms, stochastic process models, neural spike trains, linear filter models of sensory neurons, population coding, and decoding algorithms). Second, there are normative models attempting to answer *the why*, explaining neural activity and connectivity in terms of their high-level functional role for the organism. For instance, neural coding of sensory information is made more efficient by using decorrelation, a computation that reduces redundancy in neural activity due to stimulus inputs and explains aspects of adaptation in early sensory systems. Another example involves changes in neuronal firing when animals switch environments, to allow adaptation to new stimuli. This change occurs because sensory systems aim to encode maximal information about environments with different statistics, an instance of the "efficient coding hypothesis" [1]. In this line of research, how neurons' mean activity and the cross-correlation between these activities change as a result of switching the visual environment

from static to moving states – in a model of a primary visual cortex (V1) microcircuit – is the subject of Chapter 2. Allied normative theories may also involve statistical Bayesian inference theory, arguing that neural coding and sensory stimuli processing depend on the organism's prior information about the environment. Thus, the prior probability distribution of the sensory environment can be fixed as the model is optimized to correspond to empirical data [41]. Several other studies on Bayesian inference focus on optimal integration of information from different sensory modalities about one feature [25]. In Chapter 2, we propose that V1 lateral connections (connections between neurons within a layer) serve to optimally integrate information from context – the surrounding visual features to a particular feature in the visual space. Contextual information is thus stored in the lateral connections between neurons [16, 40]. These connections are dependent on the statistics of the environment, more precisely on the frequency of co-occurrence in the environment of the features the neurons are tuned to. In sum, we create a normative model where the circuit performs optimal inference and is adapted to the statistical regularities of the context through lateral connections.

Lastly, a type of model which we only briefly discuss here is the biophysically realistic model based on neuroanatomy (cell types, connectivity, and morphology) and neurophysiology (from the biophysics of neurons and synapses to neuronal population activity during behavior). One of the first, and certainly the most well-known. biophysically realistic neuron models of electrophysiology was achieved by physiologists Alan Lloyd Hodgkin and Andrew Fielding Huxley [14], who jointly won the Nobel Prize in Physiology or Medicine in 1963. While subsequent chapters of this dissertation do not include such a level of biologically realistic detail in mathematical models of action potential initiation and propagation, we do focus on including cell types in our models and describing their computational roles. In Chapter 2, we study a biologically inspired microcircuit model with multiple inhibitory cell types can switch between visual processing of the static context and the moving context; in Chapter 3, we introduce a new artificial neural network architecture, the bottleneck-switching network, which includes switching units that enable the network to adapt to different changing backgrounds while maintaining good classification performance on cur-

rent tasks, without forgetting previous tasks; finally, in Chapter 4, we classify cell types using neuronal activity and features of proximal neurons in the connectivity graph, while showing which network motifs are more closely associated with each cell type. Such a focus on including heterogeneous cells and connections has been a significant step forward in efforts to design more biologically realistic models, in contrast to typical artificial neural network models where there is no such distinction between individual neural "units.". This heterogeneity of units and interaction patterns in neural systems in principle limits the scope of basic continuum models or mean-field theories, instead leading computational scientists to focus on tools from network science and other fields with the goal of developing statistical mechanics of complex networks [24]. One interesting recent study [23] describes a computational role for cell types in synaptic learning as part of the model, where modulatory terms respecting neuronal types provide nonlocal information in the form of diffusive signaling. Other efforts to introduce cell-type information in models of biological and artificial neural networks have gained traction in recent years [42, 7, 6, 3].

From a different perspective, computational scientists have approached modeling through two other categories of approach: modeling the brain network structure and modeling the brain network function. The former refers to modeling the connectivity patterns of brain networks, the physical "structure" on which neuronal dynamics occurs. Using tools and intuitions primarily from network science, one important goal has been to uncover the organizing principles of structural networks of brains that have been mapped extensively (C. elegans, the mouse, cat, macaque, human, etc.) An approach is to introduce generative network models that explain how the network's properties emerge from fundamental biological mechanisms [24]. Examples include random networks (through Erdos-Renyi models); networks with community structures where densely connected communities are separated by sparse inter-community connectivity (through stochastic block models); small-world structures with short average path lengths between all nodes (through Watts-Strogatz model); hub structure with degree "hubs" that form a densely inter-connected core (through the Barabasi-Albert model); spatial structure, where the brain's wiring is subject to physical and metabolic constraints.

Such models are however insufficient. Understanding the computation in of brain networks entails also modeling their functional dynamics. The most basic example of this is Hebbian plasticity [13] by which the strength of a synaptic connection increases with the persistent synchronized firing of its pre- and postsynaptic neurons. As indicated by Lynn and Bassett [24], Hebbian learning and other biological mechanisms highlight the fact that in order to have a complete understanding of the brain, a characterization of its structural wiring is not enough; instead, we must also describe the dynamics supported by this physical circuitry. Studying the functional brain network involves understanding how the neural wiring of the brain enables neural dynamics. Whereas structural brain networks define connectivity between nodes (neurons, brain regions, etc.) based on physical measurements of wiring (e.g. synapse strength, number of white matter tracts between brain regions), functional brain networks describe connectivity based on the similarity between two elements' dynamics. Functional and structural organization bear striking similarities, therefore it is easy to assume that functional brain networks approximately resemble their corresponding physical wiring. However, the relationship between structure and function is highly non-linear [31], therefore understanding how a functional brain network arises from its underlying structural connectivity, and describing the statistical, communication, and biophysical models of higher-order interactions that can be used to translate brain structure to brain function remain topics of intense study [24, 26].

Finally, dynamical modeling occurs at different scales, from the smaller scale modeling of individual neuronal dynamics, then increasing in scale to neuronal mass models of distinct brain regions, and finally to models of entire networks of neurons and brain regions. There are two classes of models here, the biophysically realistic ones, and the artificial neural networks. We already mentioned the former, and we discuss the latter class in more detail below, in Section 1.3. Importantly, there is a lot of potential for cross-pollination between computational neuroscience and the field of deep learning whose main focus is artificial neural networks. A better understanding of the neural architectures and motifs that biological brains employ, as well as the algorithms, functions, and representations, utilized through their dynamics, could play a vital role in inspiring next-generation intelligent machines.

1.3 Artificial Neural Networks

Artificial Neural Networks (ANNs) are computing systems inspired by biological neural networks. The main idea, pioneered by in McCulloch and Pitts in the mid 1940s, and then advanced by Rosenblatt in 1958, is that artificial neurons typically take in a weighted combination of inputs and pass the inputs through a nonlinear threshold function to generate an output. Typical nonlinear functions are the sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$ or ReLU:

$$\sigma(x) = \begin{cases} \text{x} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{1.1}$$

in addition to the Heaviside (binarizing) function used below.

The perception is a simple ANN with one layer. The output of a perceptron can be expressed in terms of its inputs via:

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j \geq \text{threshold} \end{cases} \tag{1.2}$$

where x_j are the perceptron's inputs, w_j are *weights* that are perceptron parameters forming the weighted combination of inputs that gets thresholded.

Scaling this to many such neurons and doing it sequentially across a number of layers is what gives rise to an artificial neural network. This is shown for a simple neural network (Figure 1.1c): the input is all the pixels of the image, and the goal of the ANN is to classify whether the image represents a cat or a dog. Each unit in the first layer takes a linear combination of input pixels and then applies a non-linear function. The result is called *activation* of the unit. This calculation is done for every unit in the first layer, and then these activations are used as the inputs for the next layer, and so on. Activations at layer l are generally described by equations such as the following:

$$a_j^l = \sigma(\sum_k w_{jk} a_k^{l-1} + b_j^l) \tag{1.3}$$

where a_j^l is the jth activation unit at layer l, w_{jk} are weights between the jth and kth units, and b_j^l are the corresponding biases. The following is a vectorized equivalent version of equation 1.3:

$$a^l = \sigma(w^l a^{l-1} + b^l) \tag{1.4}$$

where w^l are now matrices of weights, with entries $\{w_{jk}\}_{j,k}$.

The output after L layers is a^L, which is usually of dimension equal to the number of classes and is interpreted as the probability corresponding to a certain class. For instance, if the output is 2-dimensional, e.g. $[0.95, 0.05]$, we think of the probability that the image is a dog being 95%. The output is then compared with the actual label of the input image, and a particular learning rule adjusts the weights of the network according to the error between the labels found by the ANN and the true labels. The error between output labels of ANNs and true labels is quantified by the cost function, which can take many forms (e.g. mean squared error, cross-entropy loss, negative log-likelihood loss, etc.). An easy-to-conceptualize cost function is the mean squared error cost:

$$C = \frac{1}{2n} \sum_x \|y(x) - a^L(x)\|^2 \tag{1.5}$$

where C is the cost, $y(x)$ is the desired output corresponding to the true label, and $a^L(x)$ is the actual output of the ANN. In the case of classification problems, the preferred outputs y are vectors of probabilities with entries of one corresponding to the true class, and entries of zero everywhere else. The cost function is used in tandem with a learning rule (often, through an algorithm called backpropagation) that is typically based on gradient descent, a first-order iterative optimization algorithm that takes repeated steps in the opposite direction of the gradient of the cost function at the current point because this is the direction of steepest descent. Updating the weights of the ANN according to gradient descent can be mathematically described via the following formula:

$$\Delta w_{jk} = w_{jk} - \eta \frac{\partial C}{\partial w_{jk}} \tag{1.6}$$

where η is the learning rate, another hyperparameter of the training process. Upon certain choice of hyperparameters for the network (learning rate, number of layers of the network, etc.) the learning rule eventually results in the network modifying its weights to produce outputs that closely match the true labels.

These ANNs ignore details about the biophysics of real neurons and the transmission of signals. For instance, chemical and electrical signaling between neurons (through the use of neurotransmitters and through gap junctions) are different ways by which neurons communicate with each other, and recent work has revealed further roles for local neuropeptide signaling in the regulation of cortical synaptic plasticity [38]. Therefore, biological neurons can interconnect in complex ways, through many types of connections, whereas units of classical ANNs are simplified variants that use a single type of weight. The computational power and richness of the ANN models come instead from the large number of connections (or "weights") between the units and the depth of the network.

Because of their ability to reproduce and model nonlinear processes, ANNs are currently successfully applied in diverse disciplines. In particular, the last decade – from 2010 until the present – has seen an explosion in ANN applications, spanning diverse fields such as pattern recognition (object recognition, face identification, signal classification), machine translation and natural language processing, system identification, and control (vehicle control, trajectory prediction), sequence recognition (speech, handwritten and printed text), medical diagnosis, finance, data visualization, and many others. There have been a few important precursors leading up to this latest surge in deep learning innovation that are worth highlighting here. First, Yann LeCun proposed the first convolutional neural network (CNNs) in 1994. These networks' main insight is that image features are distributed across the entire image, and convolutions with learnable parameters are an effective way to extract similar features at multiple locations with few parameters [22, 10]. Given the computational

power at that time, the ability to limit the number of parameters and save time was key. The way CNNs process their input is in contrast to using each pixel as a separate input in a multi-layer ANN as described above. LeCun's CNN, LeNet5, made use of the fact that images are highly spatially correlated, therefore using individual pixels of the image as separate input features would not take advantage of these correlations. Intriguingly, CNNs are inspired by biology, in that the network's units resemble neurons from the visual cortex responding only to a restricted region of the visual field known as the receptive field. In this way, the connectivity pattern between neurons resembles the organization of the visual cortex.

Another breakthrough in the field was the creation of ImageNet by Stanford's FeiFei Li. ImageNet is a massive training dataset of more than 14 million natural images [34] that made it easier for researchers to develop computer vision algorithms, which in turn paved the way for a series of difficult image competitions that saw teams around the globe competing to see which could train the most accurate ANNs. Finally, another essential ingredient came when Ciresan and colleagues [4] made training ANNs using the backpropagation algorithm faster using graphics processing units (GPUs), which are specialized electronic circuits whose parallel structure makes them more efficient than general-purpose central processing units (CPUs) for algorithms that process large blocks of data in parallel. These developments made possible the next generation of very powerful ANNs, starting with Alex Krizhevsky's AlexNet [21] which was a deeper and much wider version of the LeNet and won by a large margin the difficult ImageNet competition. Other sophisticated networks in the field of Computer Vision followed, with impressive results: VGG and its iterations [37], Network-in-Network [27], GoogLeNet and its Inception iterations [39], etc. In turn, these computer vision algorithms have led to similar paradigms for natural language processing and other bedrock AI applications. These novel and highly effective artificial networks are now further removed from biology and specifically designed towards improving empirical results.

Despite the fact that the deep learning community has largely abandoned attempts to

remain true to the ANN's biological precursors, Neuroscience has and will hopefully continue to play a vital role in building intelligent machines. Hassabis et. al. (2017) [11] and other recent publications argue about the critical and ongoing importance of neuroscience in generating ideas that will accelerate and guide AI research. Neuroscience has provided a rich source of inspiration for new directions in deep learning, specifically at the computational and algorithmic level [1]. As mentioned above, CNNs are inspired by the visual cortex through every unit's focus on a patch of the visual space at a time, akin to neurons' receptive fields; they also use several canonical hallmarks of neural computation, including nonlinear transduction, divisive normalization, and maximum-based pooling of inputs [11]. These operations are directly inspired by recordings of single cells from the mammalian visual cortex that demonstrated how visual input is filtered and pooled in simple and complex cells. Another example of neuro-inspired deep learning is a regularization scheme called dropout that supports generalization beyond training data. When dropout is applied to a network layer, only a subset of units participate in the processing of a given training example, similarly to the stochasticity that is inherent in biological systems populated by neurons that fire with Poisson-like statistics. Neuroscience was also instrumental in stimulating the emergence of the field of reinforcement learning (RL), and in particular, the temporal difference (TD) method, which was inspired by research into animal learning.

While these are classic examples that are used in standard ANNs, there are more recent examples where neuroscience has inspired current developments in deep learning research. For instance, attentional mechanisms have been a source of inspiration for AI architectures that take in sections of the input at each step, rather than the whole input at once. One such network [28] was able to use the selective attentional mechanism to ignore a cluttered background with many irrelevant objects and perform a difficult object classification task. Another feature of ANNs inspired by neuroscience is the use of episodic memory. Through the use of "experience replay", the network stores a subset of the training data, then re-

[1] we omit discussing the implementation level, where significant progress has been made using spiking neural networks in neuromorphic architectures, and focus on building intelligent architectures in silico, using the simplified ANN framework described above.

plays it offline, learning from the successes and failures that happened in the past [29]. Such a network is the deep Q-network (DQN), a network that learns to play Atari 2600 video games at an expert level and is capable of achieving "one-shot learning" (learning after just one exposure) by using experience replay. AI research has also drawn inspiration from working memory models by building architectures such as recurrent neural networks (RNNs) displaying attractor dynamics and rich sequential behavior [9, 15]. Finally, a key challenge for ANNs is continual learning, the ability to master a new task without forgetting how to perform prior tasks. An elegant solution called elastic weight consolidation [20] suggests that memories can be protected from interference through a subset of synapses important for the initial task that are less plastic and thus slower to change their value. A number of other bio-inspired architectures have been proposed to address continual learning [44, 30, 8, 18, 36]. We discuss continual learning at length in Chapter 3 and find a bio-inspired network motif that can solve the continual learning problem when the background statistics of images changes.

There are a few other areas where ANNs will likely make significant contributions, and we expect this to be in part guided by neuroscience in establishing a roadmap for the research agenda. ANNs will likely improve their intuitive understanding of the physical world, constructing mental compositional models that can guide inference and prediction. Recent work has developed deep network able to mimic the process by which children gain a commonsense understanding of the world through interactive experiments [5]. ANNs should also be able to learn rapidly from just a few examples, "learn to learn" by generalizing on prior tasks, achieve transfer learning by which generalized knowledge in one context is transferred to novel domains, and create "mental models" by which ANNs can flexibly select actions based on future predictions through simulation-based planning that is generated from an internal model of the environment learned through experience [11]. All these areas of research would benefit from insights into the brain's unique ability to learn to perform complex computations underlying perception, cognition, and motor control – defining features of intelligent behavior.

In computational neuroscience, ANNs have a dual role: as machine learning tools to aid in data mining efforts (e.g., to classify cell types based on a number of features from connectivity and neural recording data, see Chapter 4), but distinctly also as models capable of reproducing important aspects of neural activity and system function that have been observed in a range of physiological and behavioral experiments. Unlike biophysical models which provide a more detailed account of realistic neuronal dynamics, the unit of an ANN is a purely conceptual neuron model. Despite this, important insights can be gained from state-of-the-art neural networks used as plausible reproductions of biological brains, potentially providing explanations of the computations occurring therein. Relevant work here focuses on CNNs and has insights on the neural representations in high-level visual areas. From a set of more than 30 ANNs, deep supervised networks were able to explain the structure of neural representations in the ventral visual stream of humans and monkeys [19, 43]. Deep learning is also providing novel insights on how the brain might implement an algorithmic parallel to backpropagation, the key mechanism that allows weights within multiple layers of a hierarchical network to be optimized toward an objective function [35]. Other areas of AI contributions to neuroscience include insights from Recurrent Neural Networks (RNNs) and Long-Short term memory networks (LSTMs), external memory architectures and their ability to model the hippocampus [11].

These interactions point to fundamental contributions to advancing AI research from the field of neuroscience and to the potential of these two fields to interact synergistically. Ovearall, some insights from neuroscience may not be directly transferable to ANNs, rather they can stimulate algorithmic-level questions about facets of intelligence and learning of interest to AI researchers, providing intuitions towards relevant mechanisms [11].

Bibliography

[1] H Barlow. Possible principles underlying the transformation of sensory messages. Sensory Communication, pages 217–234, 1961.

[2] CC Bell. An efference copy which is modified by reafferent input. Science, 214:450–452, 1981.

[3] SR Bittner, RC Williamson, AC Snyder, A Litwin-Kumar, B Doiron, SM Chase, MA Smith, and BM Yu. Population activity structure of excitatory and inhibitory neurons. PLoS One, 12(8), 2017.

[4] DC Cireşan, U Meier, LM Gambardella, and J Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. Neural Computation, 22(12):3207–3220, 2010.

[5] M Denil, P Agrawal, TD Kulkarni, T Erez, P Battaglia, and N de Freitas. Learning to perform physics experiments via deep reinforcement learning. 2016.

[6] M Dipoppa, A Ranson, M Krumin, M Pachitariu, M Carandini, and KD Harris. Vision and locomotion shape the interactions between neuron types in mouse visual cortex. Neuron, 98(3):602–615, 2018.

[7] B Doty, S Mihalas, A Arkhipov, and A Piet. Heterogeneous "cell types" can improve performance of deep neural networks. bioRxiv, 2021.

[8] Timothy J. Draelos, Nadine E. Miner, Christopher C. Lamb, Jonathan A. Cox, Craig M. Vineyard, Kristofor D. Carlson, William M. Severa, Conrad D. James, and James B. Aimone. Neurogenesis deep learning: Extending deep networks to accommodate new classes. 2017 International Joint Conference on Neural Networks (IJCNN), pages 526–533, 2017.

[9] JL Elman. Finding structure in time. Cogn. Sci., 14:179–211, 1990.

[10] K Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol. Cybernetics, 36:193–202, 1980.

[11] D Hassabis, D Kumaran, C Summerfield, and M Botvinick. Neuroscience-inspired artificial intelligence. Neuron, 95(2):245–258, 2017.

[12] J Haugeland. Artificial Intelligence: The Very Idea. MIT Press, 1985.

[13] DO Hebb. Organization of Behavior. Psychology Press, 1949.

[14] AL Hodgkin and AF Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. The Journal of Physiology, 117(4):500–44, 1952.

[15] JJ Hopfield and DW Tank. Computing with neural circuits: a model. Science, 233:625–633, 1986.

[16] R Iyer, B Hu, and S Mihalas. Contextual integration in cortical and convolutional neural networks. Frontiers Computational Neuroscience, 2020.

[17] ER Kandel. Principles of Neural Science. Number Fifth Edition. McGraw-Hill Education, 2012.

[18] Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. ArXiv, abs/1711.10563, 2018.

[19] SM Khaligh-Razavi and N Kriegeskorte. Deep supervised, but not unsupervised, models may explain it cortical representation. PLoS Comput. Biol., 10, 2014.

[20] J Kirkpatrick, R Pascanu, N Rabinowitz, J Veness, G Desjardins, AA Rusu, and et al. Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences, 114(13):3521–3526, 2017.

[21] A Krizhevsky, I Sutskever, and GE Hinton. Imagenet classification with deep convolutional neural networks. Communications of the ACM, 60(6):84–90, 2017.

[22] Y LeCun, B Boser, JS Denker, D Henderson, RE Howard, W Hubbard, and LD Jackel. Backpropagation applied to handwritten zip code recognition. Neural Computation, 4(541-551), 1989.

[23] YH Liu, S Smith, S Mihalas, E Shea-Brown, and U Sümbül. Cell-type specific neuromodulation guides synaptic credit assignment in a spiking neural network. Proceedings of the National Academy of Sciences, 118(51):e2111821118, 2021.

[24] CW Lynn and DS Bassett. The physics of brain network structure, function, and control. Nature Review Physics, 1:318–332, 2019.

[25] WJ Ma, JM Beck, PE Latham PE, and A Pouget. Bayesian inference with probabilistic population codes. Nature Neuroscience, 9(11)::1432–8, 2006.

[26] JD Medaglia. Functional alignment with anatomical networks is associated with cognitive flexibility. Nature Human Behaviour, 2:156–164, 2018.

[27] L Min, Q Chen, and S Yan. Network in network. 2013.

[28] V Mnih, N Heess, A Graves, and K Kavukcuoglu. Recurrent models of visual attention. 2014.

[29] V Mnih, K Kavukcuoglu, D Silver, AA Rusu, J Veness ans MG Bellemare, A Graves, M Riedmiller, AK Fidjeland, G Ostrovski, et al. Human- level control through deep reinforcement learning. Nature, 518:529–533, 2015.

[30] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. Neural Networks, 113:54–71, 2019.

[31] HJ Park and K Friston. Structural and functional brain networks: from connections to cognition. Science, 342(1238411), 2013.

[32] PD Roberts and CC Bell. Active control of spike-timing dependent synaptic plasticity in an electrosensory system. Journal of Physiology, 96:445–449, 2002.

[33] RH Roth and JB Ding. From neurons to cognition: Technologies for precise recording of neural activity underlying behavior. BME Frontiers, 2020:7190517, 2020.

[34] O Russakovsky, J Deng, H Su, J Krause, S Satheesh, S Ma, Z Huang, A Karpathy, A Khosla, M Bernstein, AC Berg, and L Fei-Fei. Imagenet large scale visual recognition challenge. IJCV, 2015.

[35] B Scellier and Y Bengio. Equilibrium propagation: bridging the gap between energy-based models and backpropagation. 2016.

[36] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In NIPS, 2017.

[37] K Simonyan and A ZissermanVery deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. Very deep convolutional networks for large-scale image recognition. 2014.

[38] SJ Smith, M Hawrylycz, J Rossier, and U Sümbül. New light on cortical neuropeptides and synaptic network plasticity. Current Opinion in Neurobiology, 63:176–188, 2020. Cellular Neuroscience.

[39] C Szegedy, W Liu, Y Jia, P Sermanet, S Reed, ... D Anguelov, and A Rabinovich. Going deeper with convolutions. Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1–9, 2015.

[40] D Voina, S Recanatesi, B Hu, E Shea-Brown, and S Mihalas. Single circuit in v1 capable of switching contexts during movement using an inhibitory population as a switch. Neural Computation, 34(3):541–594, 2022.

[41] XJ Wang, H Hu, C Huang, H Kennedy, CT Li, N Logothetis, ZL Lu, Q Luo, M Poo, D Tsao, S Wu, Z Wu, X Zhang, and D Zhou. Computational neuroscience: a frontier of the 21st century. National Science Review, 7(9):1418–1422, 2020.

[42] CN Winston, D Mastrovito, E Shea-Brown, and S Mihalas. Heterogeneity in neuronal dynamics is learned by gradient descent for temporal processing tasks. bioRxiv, 2022.

[43] DL Yamins and 356-365. JJ DiCarloUsing goal-driven deep learning models to understand sensory cortex. Nat. Neurosci. 19. Using goal-driven deep learning models to understand sensory cortex. Nat. Neurosci., 19:356–365, 2016.

[44] F Zenke, B Poole, and S Ganguli. Continual learning through synaptic intelligence. International Conference on Machine Learning, pages 3987–3995, 2017.

[45] GKH Zupanc. Behavioral Neurobiology: An Integrative Approach. Oxford University Press, Oxford, UK, 2004.

Chapter 2

SINGLE CIRCUIT IN V1 CAPABLE OF SWITCHING CONTEXTS DURING MOVEMENT USING VIP POPULATION AS A SWITCH

The first neural network architecture we will study is a microcircuit in the visual cortex whose activity is modulated differently by visual input during movement and static conditions. We propose a normative, computational model with cell types that specifies the circuit activity and connectivity such that optimal context integration is achieved. We find that this optimality introduces requirements on the architecture and the units of the neural network. This is an instance of neural network architecture discovery where empirical evidence and our modeling constrain the network's structure.
Work in this chapter has appeared in the Neural Computation journal [68].

2.1 Summary

As animals adapt to their environments, their brains are tasked with processing stimuli in different sensory contexts. Whether these computations are context dependent or independent, they are all implemented in the same neural tissue. A crucial question is what neural architectures can respond flexibly to a range of stimulus conditions and switch between them. This is a particular case of flexible architecture that permits multiple related computations within a single circuit.

Here, we address this question in the specific case of the visual system circuitry, focusing on context integration, defined as the integration of feedforward and surround information across visual space. We show that a biologically inspired microcircuit with multiple inhibitory cell types can switch between visual processing of the static context and the moving context. In our model, the VIP population acts as the switch and modulates the visual circuit through a disinhibitory motif. Moreover, the VIP population is efficient, requiring only a relatively small number of neurons to switch contexts. This circuit eliminates noise in videos by using appropriate lateral connections for contextual spatio-temporal surround

modulation, having superior denoising performance compared to circuits where only one context is learned. Our findings shed light on a minimally complex architecture that is capable of switching between two naturalistic contexts using few switching units.

2.2 Introduction

Our brains are unique in their ability to adapt to the context in which stimuli appear. Animals face the problem of processing visual stimuli rapidly and efficiently while adapting to different contexts every time they transition to a new environment (e.g. from jungle to savanna, from the shores of a river to underwater). A classic example of adaptation to different contexts is discussed in Barlow's "efficient coding hypothesis" (Barlow, 1961), which proposes that sensory systems encode maximal information about environments with different statistics (Olshausen and Field 1996a,b). In this and other cases, when context changes, neural circuits switch from previous strategies of feature representation to new ones that are better adapted to the statistical properties of the new context. How the neuronal circuitry of the brain is organized to account for the multitude of contexts animals may encounter has not yet been established (Yang et. al., 2019). In particular, when do we need separate circuits for different contexts, and when can single circuits be modulated to switch among multiple contexts? (Gozzi et. al., 2010; Koganezawa et. al., 2016; Zhou et. al., 2017; Cardin, 2019; Mante et. al., 2013; Cohen et. al., 1990; Yang et. al., 2019). Our aim is to identify a biologically constrained network that is capable of switching contexts, and to infer the building blocks required for such switching. In constructing such a network we will only discuss and include the structural and functional detail needed for the switching of contexts.

We focus on a concrete setting in which rapid context switching is apparent. This is mouse V1, which responds differently to inputs when the animal is running (moving condition), compared to when it is stationary (static condition) (Niell and Stryker, 2010; Fu et. al., 2014). When the animal transitions from standing still to running, visually-evoked firing rates significantly increase. For example, in one experimental setting, the firing rate of neurons in layers II/III of area V1 more than double (Niell and Stryker, 2010), while in layer V of V1, noise correlations between pairs of neurons are substantially reduced (Dadarlat and

Stryker, 2017).

While an enormous diversity of cell types has been characterized (Tasic et. al., 2018), in this work we focus on the three primary classes of inhibitory interneurons: vasoactive intestinal peptide (VIP), somatostatin (SST), parvalbumin (PV), and one class of long range projecting excitatory neurons: the pyramidal neurons (PYR) as shown in Figure 4.3a (Fu et. al., 2014; Cardin, 2018; Rudy, 2011; Pfeffer et. al., 2013). VIP is an inhibitory population of neurons which is very strongly modulated by running (Fu et. al., 2014). In our simplified model of the circuit, VIP neurons act in a switch-like manner: they are silent when animals are static, but start firing when animals are running, inhibiting SST cells and hence releasing PYR cells from SST inhibition. The disinhibition of PYR cells is not uniform, but rather a complex pattern which is dependent on the particular PYR cell response. We will show that the switch can only be effective if PYR cells provide input information to the VIP cells. Although this simple model does not capture all the physiological responses of VIP neurons, we believe the model captures the crux of the disinhibitory switching computation at the expense of biological realism.

We study this circuit using a model in which the contextual information is stored in the lateral connections between neurons (Iyer et. al., 2020). Each neuron receives information about the visual scene from feedforward connections (which can be arbitrary in this model), and complements this with surround information provided by nearby neurons. The connections are dependent on the statistics of the environment; more precisely they depend on the frequency of co-occurrence in the environment of the features which the neurons represent. These connections are most useful if the information from the feedforward connections is corrupted (e.g. by occlusions).

Importantly, the contextual information via lateral connections comes not only from the spatial surround, but also from the past. Synaptic delays introduce a constraint on the available information each neuron gets. During the static condition, past surround information matches present information, and thus there is no temporal variability of the context. During movement, this no longer holds; neighboring features now also vary temporally, which changes the co-occurrence frequency, and hence the statistics of the moving context

is different. We aim to find connection strengths from the switching VIP units that, during movement, modulate firing rates and neuronal correlation structure to adapt and enhance the encoding of visual stimuli when the moving context is turned on. Although throughout the paper we focus on the visual circuit and the switching role of the VIP neural population, these results can be generalized to circuits processing multiple contexts, and thus their applicability has broader scope. In the discussion section, we list several other biological examples of circuits processing multiple contexts.

Understanding switching circuits may also further aid efforts to design both flexible and efficient artificial neural architectures. This research area has benefited from bio-inspired architectures and algorithms like elastic weight consolidation (Kirkpatrick et. al., 2017), intelligent synapses (Zenke et. al., 2017), iterative pruning (Mallya and Lazebnik, 2018), leveraging prior knowledge through lateral connections (Rusu et. al., 2016), task-based hard attention mechanism (Serra et. al., 2018), block-modular architecture (Terekhov et. al., 2015), etc. to enable sequential learning by eliminating "catastrophic forgetting" (where previously acquired memories are overwritten once new tasks are learned). We hypothesize that a few switching units akin to VIP can be incorporated as part of the hidden layers to enable context modulation. This makes such a switching circuit architecture (Figure 4.3c) more efficient than employing separate circuits for the different contexts (Figure 4.3b) because switching circuits have fewer connections to learn [1]. We hope such a circuit architecture will inspire next-generation flexible artificial nets that can process stimuli in changing contexts.

Outline of chapter In section 2.3.1, we first detail a model introduced in Iyer et. al. (2020) that describes neuronal connections and firing rates of a circuit adapted to static visual scenes (images). We next extend this model to the case of circuits adapted to moving visual scenes (videos). These circuits are attuned to the statistical regularities of

[1] In general, if N is the number of neurons per location, L is the number of locations, and C is the number of connections per neuron, then the total number of connections in a circuit is NLC. Two identical circuits have $2NLC$ connectivities, while a switching circuit has $NLC + LM(c_{in} + c_{out})$, where M is the number of switching (VIP) units, and c_{in}, c_{out} are the number of connections to and from the switching units, respectively. When $M << N$, then $c_{in}, c_{out} < C$ and thus $2NLC > NLC + LM(c_{in} + c_{out}) \Leftrightarrow NC > M(c_{in} + c_{out})$ which is true for circuits with small M, c_{in}, c_{out}.

movement and take into account constraints of biological networks, like synaptic delay. We are able to map these two circuit models to the V1 circuit, consisting of PYR, SST, and PV neuron populations. We thus obtain two different networks with full cell-type specifications achieving optimal context integration for static and moving contexts, respectively. In section 2.3.2 we detail the datasets and procedures used to quantify connectivities and firing rates in these two circuits. In section 2.3.3, we go on to describe a circuit that can switch between neuronal activity in static circuit and neuronal activity in the moving circuit, by virtue of adding a single population, the VIP. We find that VIP projections to SST and PYR are not enough to shift activity during movement, but that we need a feedback connection from the PYR to the VIP (sec. 2.3.4). The resulting circuit is the minimally complex circuit resembling V1 we have found to switch contexts. In section 2.3.5, we describe how this circuit switches using only a small number of VIP units. We follow up on these results in section 2.3.6, where we utilize this switching circuit to obtain better reconstructions of videos in conditions of high noise. Finally, we evaluate the new switching circuit architecture with data from V1 that confirms some of the model's predictions (sec. 2.3.7).

2.3 Results

2.3.1 Theoretical models of processing visual information in static and moving contexts

We first introduce two models of visual processing in the V1 in the static and moving contexts where the circuits implementing the computations perform optimal inference and are adapted to the statistical regularities of the contexts through the lateral connections between neurons.

Model of visual processing in the static context. To study optimal context integration in the static condition (where the visual input is static images), we take as a starting point a model proposed by Iyer et al. (2020) where model neurons respond to a patch in the visual space — the classical receptive field — but this response is modulated by a larger region of space — the extra-classical receptive field. The extra-classical receptive field contribution is determined by nearby local receptive fields providing indirect input from a larger

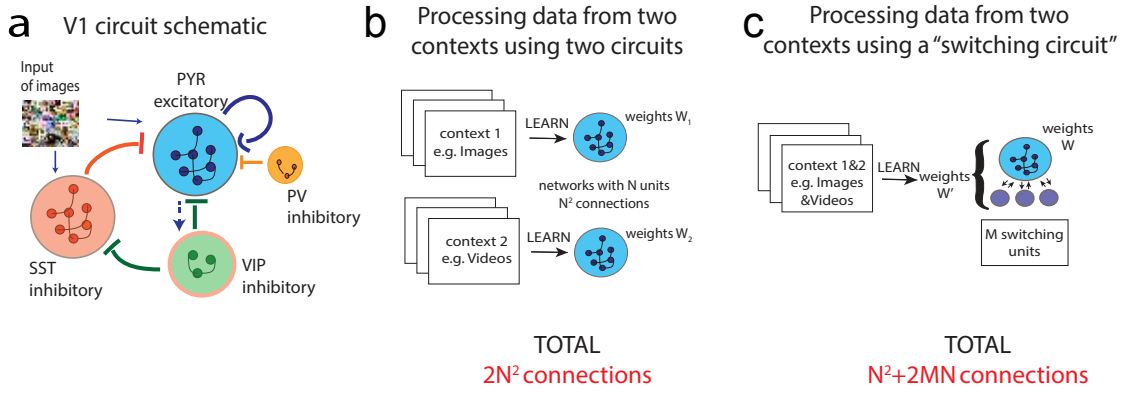


Figure 2.1: (a). Schematic of a circuit involving VIP, SST, PV, and PYR groups of neurons. When VIPs are silent, PYR are self-excitatory, while SST and PV inhibit PYR. When VIP are active, they inhibit the PYR, while also creating a disinhibitory motif given by VIP-SST-PYR. The potential connection from PYR to VIP explored in this paper is marked with a dotted arrow. (b). Processing of two input types (e.g. images, videos) happens using two separate networks for each type of input, each having N units with $2N^2$ weights in total to learn. (c). Processing of two input types can be done with one circuit — a switching circuit with N units adapted to one of the contexts, and M switching units that turn on when the other context is presented. We may want $M << N$, with $N^2 + 2NM$ connections to learn (assuming switching units are not inter-connected). When the number of switching units required in a switching circuit is small, there are fewer connections that need to be learned; more specifically, if $M < N/2 \Rightarrow N^2 + 2MN < 2N^2$. This generalizes well to a range of circuits, including in the case of sparse connectivities, as often presented throughout the paper.

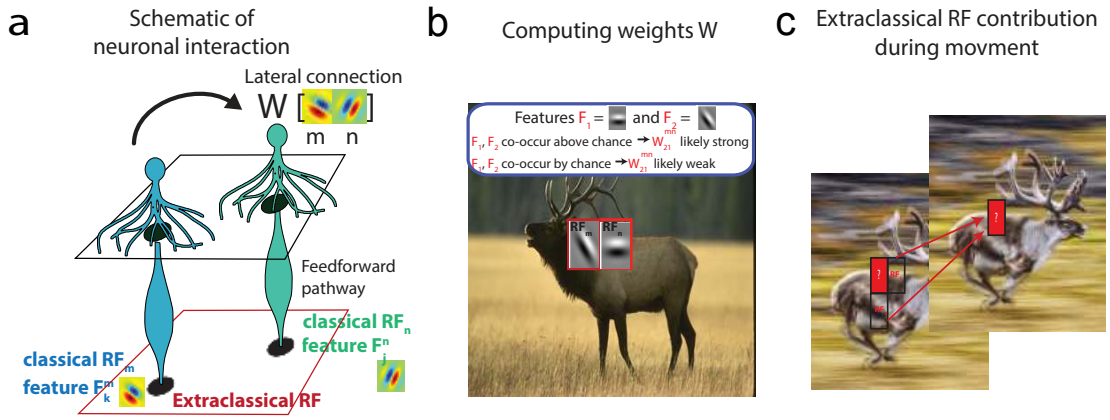


Figure 2.2: (a). Neurons receive stimulus input from a patch in space at position n, their classical receptive field (RF_n), but also from surrounding patches in space (for e.g. the patch at position m) through interactions with other neurons. These neurons are connected by weights $\mathbf{W}_{jk}^{mn}$ that depend on the statistical regularities of natural scenes. (b). When features $\mathbf{F}_1$ and $\mathbf{F}_2$ at positions m, n occur together often in natural scenes, then $\mathbf{W}_{21}^{mn}$ is strong; when $\mathbf{F}_1$ and $\mathbf{F}_2$ occur together by chance, without significant correlation, $\mathbf{W}_{21}^{mn}$ is close to 0. (c). Spatio-temporal surround for motion processing. Due to synaptic delay, context integration uses surrounding patches that are also Δt ms in the past to assess the features in the present frame.

area of visual space (Figure 4.4a). Specifically, inter-neuron interactions providing extra-classical information from the surround via lateral connections (cfr. Methods sec. 2.5.1) complement intrinsic neuronal responses to classical receptive fields to determine firing rates.

Starting from the assumption that firing rates of a population of neurons encode the probability of specific features being present in a given location of the image, we consider a probabilistic framework that includes probability of feature occurrence and feature co-occurrence, that we can then map to an equation involving firing rates of neurons and weights (cfr. Methods sec. 2.5.1). In general, a feature j, denoted by $\mathbf{F}_j$, describes a specific pattern that neurons are most attuned to, that can vary from simplistic, like Gabor filters, to complex, like faces or objects that are robust to stimulus transformations such as scale and position changes. In more detail, for neurons responding to $\mathbf{F}_j^n$ (feature j at patch n in visual space), we define $\mathbf{f}_j^n$ to be the steady-state firing rate due to the classical receptive field, and $\mathbf{r}_j^n$ to be the (overall) steady-state firing rate taking into account the extra-classical receptive field contribution. The probabilistic assumption stated above is such that $\mathbf{f}_j^n$ relates to the probability $p(\mathbf{F}_j^n|i^n)$ by the following relation:

$$\mathbf{f}_j^n = g(p(\mathbf{F}_j^n|i^n)) \tag{2.1}$$

where g is a monotonically increasing function, i^n is a patch n in visual space, and $\sum_j p(\mathbf{F}_j^n|i^n) = 1$. For simplicity, we fix g to be the identity, leaving the relaxation of this linear assumption for future work. With $\mathbf{f}_j^n = p(\mathbf{F}_j^n|i^n)$, neurons tuned for distinct features respond differently to the same patch i^n in visual space depending on how well its corresponding feature is represented. Operationally, to compute $\mathbf{f}_j^n$ in response to an image, we first chose a basis of features, for e.g. features obtained by approximating spatial receptive fields from recorded neurons in V1. We then pre-processed the image (cfr. Methods 4.2), convolved the image with feature j and normalized the result such that the sum over all features is 1 at each spatial position, and finally considered the patch i^n of the normalized convolution.

Once $\mathbf{f}_j^n$ is computed, we can continue assuming that neuronal firing rates contain information about feature occurrence in the surround, so that $\mathbf{r}_j^n = p(\mathbf{F}_j^n) = p(\mathbf{F}_j^n | i^1, i^2, ..., i^n, ...)$, where $i^1, i^2, ..., i^m$ are surrounding patches of i^n. We can then use Bayes rule to express this probability in terms of feature probability at patch i^n and at surrounding locations i^m (see Methods sec. 2.5.1 for a detailed calculation), and finally map the resulting equations to neurobiological quantities (Methods sec. 2.5.1). In summary, these operations yield that the firing rates $\mathbf{r}_j^n$ of neurons are the result of modulating the classical receptive field firing rate $\mathbf{f}_j^n$ by extra-classical receptive field information from the surround which is a linear function of other neurons' classical receptive field firing rates, $\mathbf{f}_k^m$. These firing rates are weighed by the lateral connections $\mathbf{W}^{\text{static}}$, representing the prior information about the statistical regularities of natural images. After ignoring terms which are due to higher order modulation of the surround (cfr. Methods sec. 2.5.1), specifically neurons from the surround having surround modulation of their own, we obtain the following firing rates (Figure 4.4a) as explained in detail in Methods sec. 2.5.1:

$$\mathbf{r}_j^n \approx \mathbf{f}_j^n \circ (1 + \sum_{m,k} \mathbf{W}_{kj}^{mn} \mathbf{f}_k^m) \tag{2.2}$$

with the weights expressed as:

$$\mathbf{W}_{kj}^{mn} = \frac{p(\mathbf{F}_k^m \cap \mathbf{F}_j^n)}{p(\mathbf{F}_k^m) p(\mathbf{F}_j^n)} - 1 = \frac{\langle \mathbf{f}_k^m, \mathbf{f}_j^n \rangle_{\text{all images}}}{\langle \mathbf{f}_k^m \rangle_{\text{all images}} \langle \mathbf{f}_j^n \rangle_{\text{all images}}} - 1 \tag{2.3}$$

where $\mathbf{F}_j^n$ is a Gabor-like feature n at location j that we will illustrate shortly, the symbol $\cap$ denotes the co-occurrence of two features, and $\circ$ is the Hadamard product, the element-wise multiplication between tensors $\mathbf{f}_j^n$ and $1 + \sum_{m,k} \mathbf{W}_{kj}^{mn} \mathbf{f}_k^m$. Further, $\mathbf{f}_j^n$ is the evoked firing rate due to the classical receptive field of neurons firing for feature $\mathbf{F}_j^n$, and $\mathbf{r}_j^n$ is the firing rate of neurons firing for feature $\mathbf{F}_j^n$ using information from classical and extra-classical receptive fields. The sum $\sum_{m,k} \mathbf{W}_{kj}^{mn} \mathbf{f}_k^m$ is over neurons with receptive fields at different locations m, responsive to features k. Finally, $\mathbf{W}_{kj}^{mn}$ is the connectivity in the static context between neurons responsive to features $\mathbf{F}_k^m$ and $\mathbf{F}_j^n$. We define $\mathbf{W}^{\text{static}} \equiv \{\mathbf{W}_{kj}^{mn}\}_{m,n,k,j}$ as the connectivity applied to static visual scenes. Assuming that weights only connect neurons with non-overlapping receptive fields, the resulting weights are sparse (see Methods

sec. 2.5.2).

From a computational perspective, the organism cannot measure the feature probabilities and joint probabilities in (2.1) and (2.3) directly, but these can be estimated given our defined neural code as the convolutions between image and feature, i.e. $p(\mathbf{F}_j^n|i^n) = \mathbf{f}_j^n = i^n * \mathbf{F}_j$, and as the cross-correlations between classical receptive field firing rates, i.e. $p(\mathbf{F}_k \cap \mathbf{F}_j) = \mathbf{f}_k * \mathbf{f}_j$. By mapping these probabilistic statements on feature occurrence to neurobiological quantities that capture firing rates and weights, we have obtained a circuit that does approximate context integration, extracting information through priors embedded in the neural connectivities. While the start of the model is Bayes-optimal via Equations (2.37) - (2.39), a set of approximations are needed to keep the circuit simple.

There are multiple possible mappings from the probabilistic framework to the neurobiological circuit (Iyer et. al., 2020), but the current correspondence is straightforward and yields successful predictions from data, such as like-to like connectivity, as detailed below. When a pair of features is frequently co-occurring, weights between neurons preferential for these features are strong and positive (Figure 4.4b). In contrast, when two features are unlikely to co-occur in the same image the connectivity is strong and negative. Overall occurrence probabilities of individual features normalize the co-occurrence probabilities so that the weights express the co-occurrence of features over and above chance. Co-occurrence probabilities of features are then averaged over many natural scenes so that the corresponding weights $\mathbf{W}^{\text{static}}$ capture the statistical regularities of natural environments.

Model of visual processing in the moving context. We next show how the framework above can be applied to the moving context. While Equations (2.2) - (2.3) show how connectivity and firing rates can be optimized to account for spatially co-occurring features — features that appear at the same moment in time but in different locations of the visual field — we now extend these equations to account for temporal co-occurring features — features which occur at nearby moments in time at different locations of the visual field.

In more detail, context is generally integrated from Δt in the past due to synaptic delay

(Figure 2.2c), and weights are proportional to co-occurrence probabilities of neighboring features that are also separated by a time window Δt. This is a direct generalization of the model in Iyer et. al. (2020) to the time domain, and includes synaptic delay as a biologically motivated constraint. The extended model can capture how local circuit connectivity is shaped by spatio-temporal correlations across receptive fields and across time windows characteristic of biological processes like synaptic delay. The firing rate during the moving context is (cfr. Methods sec. 2.5.1, 2.5.2):

$$\mathbf{r}_j^{n,t} \approx \mathbf{f}_j^{n,t} \circ (1 + \sum_{m,k} \mathbf{W}_{kj}^{mn,\Delta t} \mathbf{f}_j^{n,t-\Delta t}) \tag{2.4}$$

with the weights expressed as:

$$\mathbf{W}_{kj}^{mn,\Delta t} = \frac{p(\mathbf{F}_k^{m,t} \cap \mathbf{F}_j^{n,t-\Delta t})}{p(\mathbf{F}_k^{m,t}) p(\mathbf{F}_j^{n,t-\Delta t})} - 1 = \frac{< \mathbf{f}_k^{m,t}, \mathbf{f}_j^{n,t-\Delta t} >_{\text{all videos}}}{< \mathbf{f}_k^{m,t} >_{\text{all videos}} < \mathbf{f}_j^{n,t-\Delta t} >_{\text{all videos}}} - 1 \tag{2.5}$$

where we apply an analogous notation as for Equation (2.2) and Equation (2.3), the only difference being the additional $t, \Delta t, t - \Delta t$ superscripts that denote the time coordinate for the features, firing rates, and weights. $\mathbf{W}^{\text{moving}} \equiv \mathbf{W}_{kj}^{nm,\Delta t}$ is the connectivity in the moving context between neurons responsive to features $\mathbf{F}_k^{m,t}$ and $\mathbf{F}_j^{n,t-\Delta t}$ whose activation is separated by a time delay Δt. Note that the expression for $\mathbf{W}_{kj}^{nm,\Delta t}$ as shown in (2.5) also holds for the static context when we use static visual input to compute the weights, such that $\mathbf{f}^t = \mathbf{f}^{t-\Delta t}$, for all $t, \Delta t$.

We have introduced a model of visual processing where feedforward and lateral connections between neurons serve different roles. The lateral connections between neurons perform unsupervised learning of the probability of co-occurrence of visual features which the neurons represent. For the purpose of this study, the feedforward connections can be arbitrary, and the microcircuit described here can be at any level of processing. This separation of the roles for the feedforward and lateral connections allows for an easy implementation of both supervised and unsupervised learning in deep networks (Hu and Mihalas, 2018).

Here, we show how this model can integrate information from the surround using these within-layer connectivities in both static and moving states. However, integration of these two contexts results in two distinct circuits needed to perform visual processing under different conditions (static vs moving). The model optimally integrates context in the Bayes sense, meaning it uses priors on the co-occurrence of features in natural scenes when integrating information from the surround. These priors reflect the known statistical regularities of the environment (Simoncelli, 2003; Barlow, 1961; Marr, 1982) and weigh the surround contributions appropriately. We are then able to map this model formalism to the circuit architecture in V1 described above while specifying steady state network weights and activations, as well as cell type functionality. This model emphasises robust coding, and applies best in conditions of high noise, where parts of the visual scene are missing due to occlusions or are corrupted, and thus where context information may play a critical role. We next describe our model of visual processing in detail.

2.3.2 Modeling firing rates and weights in networks responding to images and videos

We next describe two separate circuits capable of doing optimal context integration in each of the moving and static contexts. We characterize these two circuits through the connectivities $\mathbf{W}^{\text{static}}$ and $\mathbf{W}^{\text{moving}}$, computed by using images and videos in training datasets and applying formulas (2.3) and (2.5). Once the corresponding connectivities are specified, we can further characterize the static and moving circuits by their neural activations. In the following, we elaborate, section by section, on the algorithm we implemented to compute the static and the moving weights.

Dataset and feature preparation. We applied our framework for processing static images and videos to different benchmark datasets, chosen to address differences in the statistics of visual features across conditions: during viewing of static images (static condition) and during viewing of videos which contain motion (moving condition). For the static condition, we used 300 selected grayscale images of the BSDS dataset (Martin et. al., 2001)

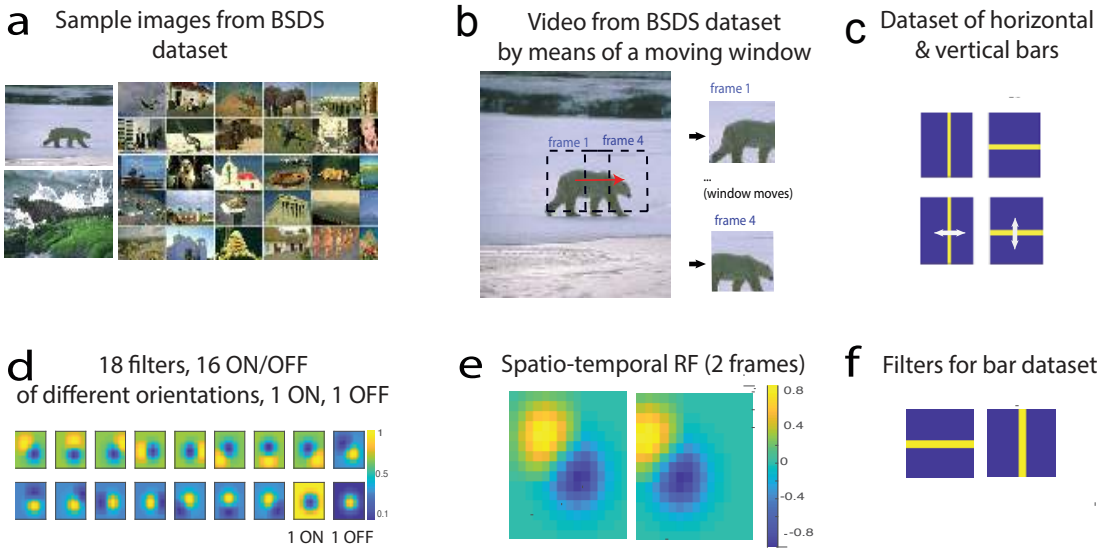


Figure 2.3: (a). Sample images from the BSDS dataset. Images of animals, human faces, landscapes, buildings, etc. are used. (b). Sliding window on images from the BSDS dataset so that the appearance of movement is achieved. Shown by the red arrow is how much the window has moved from frame 1 to frame 4. In general, movement of sliding window is random and in any direction, but we focus on horizontal movement in the case of natural videos. (c). Images of horizontal and vertical bars (above) and how the bars move in videos (below). (d). 18 filters: ON, OFF, ON/OFF with 2 Gaussian subfields, different subfields dominating, at different intensities and orientations. Colorbars show the different intensities of pixels. (e). Example of a spatio-temporal filter comprising of 2 frames. Spatio-temporal filters are added to the 18 original filters, to make up a total of 34 filters. The filter shown here over 2 frames captures a 45 deg bar moving to the left and is obtained by translating the original filter by 3 pixels. Colorbars show the different intensities of pixels to the left. (f). 2 filters for the simplistic "bar world" comprising of a horizontal and a vertical bar, respectively.

(Figure 4.5a) while for videos, the BSDS dataset is pre-processed through a smaller sliding window that travels along the image to reproduce motion (Figure 4.5b, cfr. Methods sec. 2.5.4). Although in general the sliding window can move in any direction (see Figure 2.10, Figure 2.11 for results in this case), here we constrained it to move solely in the horizontal direction to roughly approximate flow of images across the (sideways-facing) eyes of mice during forward movement. We have not used a generic dataset of natural videos since most videos in such datasets contain limited movement of objects, humans, or animals, rather than movement of sections of an environment that would mimic the visual experience of a running animal.

We generated a dictionary of features (filters) based on a parametrized set of models derived from recordings in V1 (Durand et. al., 2016). This contains 18 filters with Gaussian subfields (Figure 4.5d) at different relative intensities and orientations. We added filters containing a temporal dimension — *spatio-temporal filters* — to obtain a set of 34 filters. Our spatio-temporal filters consist of 2 frames (Figure 2.3e) and represent a temporal shift by several pixels in the horizontal direction, corresponding to the direction of movement and amount of displacement of the sliding window in the videos described above.

To more easily illustrate and interpret our model, we first tested our framework on a different, synthetic context. We analyzed a simplified 9×9 world of horizontal and vertical bars moving up-and-down as well as left-and-right (Figure 4.5c). This simple dataset has only two features, horizontal bars and vertical bars (Figure 2.3f), but movement can be in any of the four orthogonal directions.

Computing the weights $\mathbf{W}^{\mathbf{static}}$, $\mathbf{W}^{\mathbf{moving}}$. The firing rates $\mathbf{f}$ due to the classical receptive field represent feature probabilities (Equation (2.1) with $g(x) = x$) and were computed by the following sequence of operations: pre-processing inputs and filters (cfr. Methods sec. 2.5.2), convolving the image or video frames with the respective sets of filters, rectifying, and then normalizing so that all firing rates $\mathbf{f}_k^m$ lie in the interval between 0 and 1 and sum up to 1 across all features k. To find the weights for static and moving

contexts, $\mathbf{W}^{\text{static}}$ and $\mathbf{W}^{\text{moving}}$, we fixed Δt. After convolving $\mathbf{f}_k^t$ and $\mathbf{f}_j^{t-\Delta t}$ in accordance with Equations (2.3), (2.5), and following the procedure outlined in figs. 2.4a to 2.4b, we obtained a high dimensional tensor that characterizes the connections between each pair of cell types (k, j) at each position in the image. Using the feature $\mathbf{F}_j^k$ as a proxy for a cell "type," the resulting tensor is 4 dimensional, with dimensions: cell type of the source, cell type of the target, and relative spatial position of the source and target in x and y directions.

Simplifications to weights. We make three simplifications to reduce the number of parameters in this tensor (cfr. Methods 4.2): (1) we assume translational invariance so that only the relative position of two filters is relevant ($\mathbf{W}_{j_1,j_2}^{n_1,n_2} = \mathbf{W}_{j_1,j_2}^{n_3,n_4}$ when $\vec{n_1}-\vec{n_2} = \vec{n_3}-\vec{n_4}$); (2) the model is designed to compute connections to neurons which receive independent observations, thus we only consider connections between neurons whose receptive fields are sufficiently far apart (i.e. at least half a receptive field apart), (3) as statistical dependencies in natural images decay with distance, we limit the spatial extent of connectivity to three times the size of the classical receptive field. Figure 2.4c and Figure 2.4d show several 2D slices through this tensor, corresponding to a specific cell source and target, as well as the full static and moving weights (Figure 2.4f) ordered by spatial position and feature type (see also Figure 2.10a). Figure 2.4c serves to provide some intuition as to what these weights represent and how they are structured: in the dataset of bars, horizontal feature $\mathbf{F}_1$ frequently occurs or is absent together with other horizontal features $\mathbf{F}_1$ at neighboring locations, which leads $\mathbf{W}_{11}^{\text{static}}$ to have positive values. Conversely, horizontal feature $\mathbf{F}_1$ occurs always when vertical feature $\mathbf{F}_2$ is absent, and vice versa, leading to negative weights $\mathbf{W}_{12}^{\text{static}}$,$\mathbf{W}_{21}^{\text{static}}$ (Figure 2.4c).

Characterizing $\mathbf{W}^{\text{moving}}$ in the case of two different video statistics. In the generation of the video dataset we use a sliding window to enforce controlled and comparable statistics between the moving and static contexts. When the sliding window is free to move in all directions, the moving weights tend to be weaker in absolute value, which holds for the simple dataset of bars (Figure 2.4c), and the weights generated from the dataset of natural images and videos (figs. 2.10a to 2.10b). This effect is due to the weaker sta-

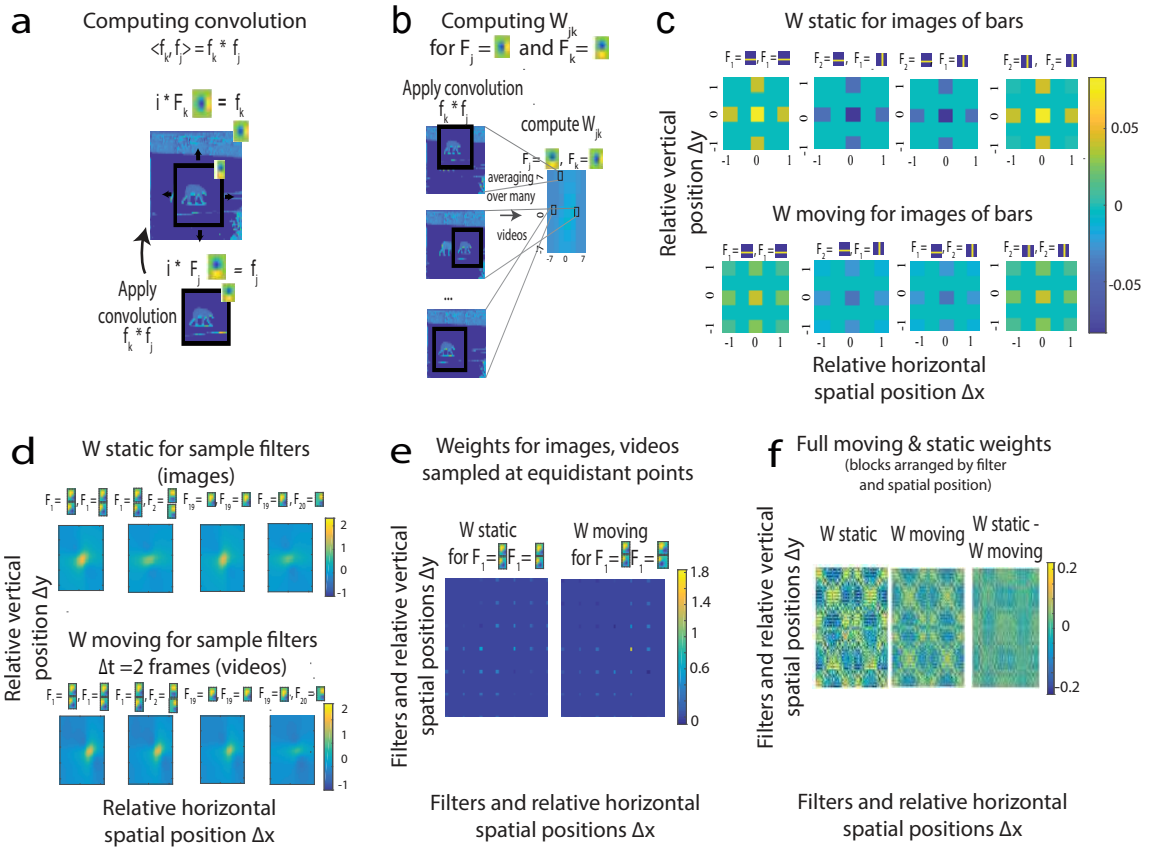
tistical dependence of features separated by the time window Δt. Feature co-occurrence, and thus connectivity, is affected by the distortions during movements, like a change of orientation of objects, or the appearance or disappearance of objects in the visual scene. Moving weights in this case are approximately smoothed-out versions of the static weights (figs. 2.10a to 2.10b). In these conditions, as the information from surround is less reliable, the feedforward input plays a more important role during movement.

In the case when the sliding window moves s pixels horizontally in Δt time steps, $\mathbf{F}_k^{n,t}$ and $\mathbf{F}_k^{n+(s,0),t-\Delta t}$ actually coincide so that their probability of co-occurrence is maximized. This means that for horizontal movement, $\mathbf{W}_{kk}^{\text{moving}}$ peaks s pixels from the center for any feature $\mathbf{F}_k$ and $\mathbf{W}_{kk}^{n,n+(s,0),\Delta t}$ is strong (figs. 2.4c to 2.4e). Results for natural videos below are for horizontal movement, although the same general conclusions hold when movement is allowed in any direction (see 2.11).

Finally, using $\mathbf{W}^{\text{static}}, \mathbf{W}^{\text{moving}}$ and applying Equations (2.2), (2.4), we obtain the corresponding firing rates $\mathbf{r}$ in both static and moving contexts.

2.3.3 Implementing a switching circuit

Having two just defined optimal connectivities, $\mathbf{W}^{\text{static}}$ and $\mathbf{W}^{\text{moving}}$, for the static and moving contexts, we next consider whether a single circuit involving the cell types described above (VIP, PYR, SST, and PV) can respond optimally in these two contexts and switch between them. We additionally seek the computational principles behind the minimally complex circuit (i.e. the circuit with fewest connections) for such a switching circuit. Specifically, we ask whether a circuit with optimal weights for the static context can switch to produce nearly optimal activities in the moving context, via projections from a set of switching units. In such a circuit, every PYR neuron approximates Bayesian inference, combining classical receptive field information with information from the surround to estimate feature probability.



,

Figure 2.4: (a). To obtain the weight matrix, we first take the convolution of video frames with features from the feature basis (e.g., $i * F_k, i * F_j$). We then consider the convolution of these convolved image frames to detect feature co-occurrence (e.g. $f_k * f_j$). (b). Schematic of how weights are represented. Normalized convolutions between patches separated by the same spatial and temporal distances are averaged and stored in the corresponding entry of the weight matrix. (c). Above: static weights for the dataset of images of bars; below: moving weights for the dataset of videos of bars. (d). Static weights (above) and moving weights (below) for the dataset of natural images/videos during horizontal motion only. (e). Sparse versions of slices from the static and moving weights for the datasets of natural images/videos during horizontal motion. Weights between neurons whose receptive fields are not at certain pre-selected, sufficiently far apart locations in the visual space were discarded to satisfy the constraint that patches are independent. (f). The full (non-sparse) tensors $\mathbf{W}^{\text{static}}$, $\mathbf{W}^{\text{moving}}$, and $\mathbf{W}^{\text{moving}} - \mathbf{W}^{\text{static}}$, ordered first by spatial position, then by filter.

We start by rewriting the model described by Equations (2.3)- (2.4) in vector form to obtain the following firing rates:

$$\mathbf{r}^{t,\text{static}} = \mathbf{f}^t \circ (1 + \mathbf{W}^{\text{static}}\mathbf{f}^t) \tag{2.6}$$

$$\mathbf{r}^{t,\text{moving}} = \mathbf{f}^t \circ (1 + \mathbf{W}^{\text{moving}}\mathbf{f}^t) \tag{2.7}$$

Assuming, as discussed above, that the activation of the VIP neural population implements the switch between contexts, we want the switching circuit to reproduce the firing rates given by (2.6) when the VIP neurons are silent in the static context, and the firing rates given by (2.7) when the VIP neurons are active in the moving context (Figure 2.5a, Figure 2.5b). We next explain how $\mathbf{r}^{\text{static}}, \mathbf{r}^{\text{moving}}$ above can be modeled as the firing rates of the PYR neurons.

When the VIP are silent, the only groups of neurons active are PV, SST, and PYR. This circuit is equivalent to one without any VIP connections, reproducing firing rates of PYR given by (2.6) when the animal is static. PYR neurons contribute to integrating surround information through excitatory projections, and receive inhibitory feedback from SST interneurons (Braitenberg and Schuz, 1991). PV implements a normalization of the PYR population in our model, consistent with data on their connectivity (Jiang et. al., 2015; Pfeffer et. al., 2013). Empirically it has been shown these neurons receive the average inputs of the PYR neurons whose receptive fields overlap with their classical receptive fields, and project back equally (Pfeffer et. al., 2013). In our model, this normalization applies to the classical receptive field $\mathbf{f}$, as described in Methods sec. 2.5.1. As for the role of PYR and SST, given that PYR are excitatory and SST are inhibitory, and that $\mathbf{W}^{\text{static}} = \mathbf{W}_+^{\text{static}} + \mathbf{W}_-^{\text{static}}$, it is natural to map the positive component of the static weights, $\mathbf{W}_+^{\text{static}}$, to the connections within the PYR population, and the negative component of the static weights, $\mathbf{W}_-^{\text{static}}$, to the inhibitory connections from SST to PYR. Hence, we obtain the following:

$$\mathbf{r}^{t,\text{static}} = \mathbf{f}^t \circ (1 + \mathbf{W}^{\text{static}}\mathbf{f}^t) = \mathbf{f}^t \circ (1 + \mathbf{W}_+^{\text{static}}\mathbf{f}^t + \mathbf{W}_-^{\text{static}}\mathbf{f}^t) \tag{2.8}$$

can be mapped to

$$\mathbf{r}^{t,\text{static}} = \mathbf{f}^t \circ (1 + \mathbf{W}^{\text{static}}\mathbf{f}^t) = \mathbf{f}^t \circ (1 + \mathbf{W}^{\text{PYR}\to\text{PYR}}\mathbf{f}^t + \mathbf{W}^{\text{SST}\to\text{PYR}}\mathbf{f}^t) \tag{2.9}$$

where $\mathbf{W}^{X\to Y}$ denotes the weights that connect neuronal populations $\mathbf{X}$ (the source) and $\mathbf{Y}$ (the target).

On the other hand, when VIP are active, PYR firing rates ought to reproduce the activity given by (2.7). We make the simplifying assumptions that the switch from static to moving can happen instantaneously, and that the VIP switch is binary. When the animal initiates movement and the VIP turns on, the model circuit should approximate the optimal response of PYR neurons resulting from the $\mathbf{W}^{\text{moving}}$ connectivities, within a circuit where the 4 neuronal populations interact (Figure 2.5b). For VIP modulation of PYR (which is either direct or through the SST) that gives rise to the optimal firing rates in the moving context, we have that:

$$\mathbf{r}^{t,\text{moving}} = \mathbf{f}^t \circ (1 + \mathbf{W}^{\text{moving}}\mathbf{f}^{t-\Delta t}) \tag{2.10}$$

is mapped to

$$\mathbf{r}^{t,\text{moving}} = \mathbf{f}^t \circ (1 + \mathbf{W}^{\text{static}}\mathbf{f}^{t-\Delta t} + \text{VIP contribution}) \tag{2.11}$$

Thus, the switch in the circuit occurs as VIP neurons modulate SST and PYR neurons and make PYR switch firing rates from $\mathbf{r}^{\text{static}}$ to $\mathbf{r}^{\text{moving}}$. We now proceed to find the unknown connectivities from VIP to PYR and from VIP to SST, that causes this to occur within the circuit (Figure 2.5b, Figure 2.5c).

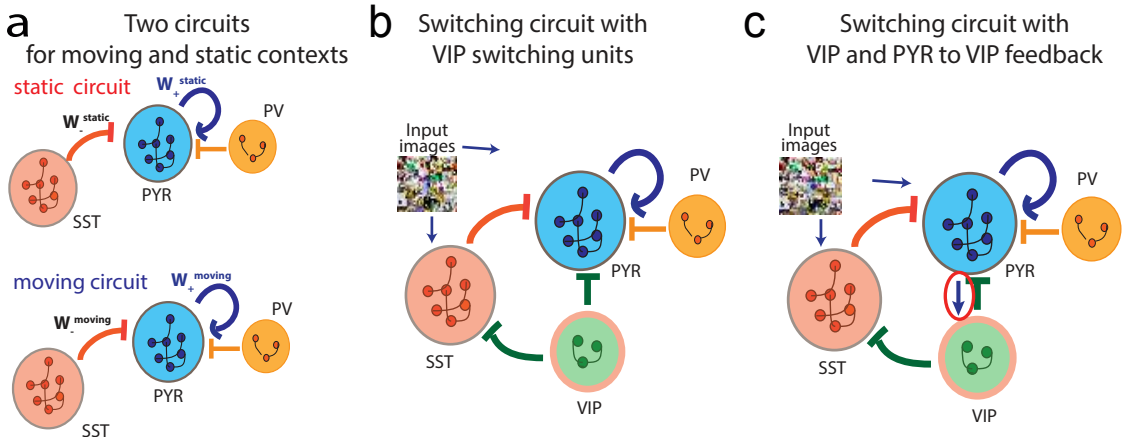


Figure 2.5: (a). Two separate circuits for optimal visual processing of static (top) and moving contexts (bottom), respectively. (b). The proposed switching circuit with the VIP population approximates the static circuit when the VIP are silent and the animal is static, and approximates the moving circuit when the VIP are active and the animal is moving. (c). Previous circuit, but with a feedback connection added from the PYR population to the VIP.

2.3.4 In the absence of feedback to VIP neurons, the circuit is unable to switch from static to moving conditions

We attempt to describe the computational principles of the minimal switching circuit inspired by the V1 circuitry whose main structure and logic was described in Fu et. al., 2014. After adding the switching population VIP, the goal is to find connectivities from VIP to the other two neuronal populations (PYR, SST) that would account for the PYR firing rates that yield optimal representation in the moving context. With the VIP contribution, the firing rate of PYR neurons can be expressed as (cfr. Methods sec. 2.5.5):

$$\mathbf{r}^{t,\text{moving}} = \mathbf{f}^{t} \circ (1 + \mathbf{W}^{\text{static}} \mathbf{f}^{t-\Delta t} + \mathbf{W}^{\text{SST} \rightarrow \text{PYR}} \mathbf{W}^{\text{VIP} \rightarrow \text{SST}} \mathbf{f}^{t-\Delta t, VIP} + \mathbf{W}^{\text{VIP} \rightarrow \text{PYR}} \mathbf{f}^{t-\Delta t, VIP}), \tag{2.12}$$

where $\mathbf{f}^t, \mathbf{f}^{t-\Delta t}$ are firing rates due to the classical receptive field at times t and $t-\Delta t$ and inferred from the dataset of natural videos as outlined in sec. 2.3.1 and Methods sec. 2.5.2, $\mathbf{f}^{\mathrm{t},VIP}$ are the intrinsic firing rates of the VIP at time t, and $\mathbf{r}^{t,\mathrm{moving}}$ is the firing rate during the moving context with the extra-classical receptive field contribution. Here, $\mathbf{W}^{\mathrm{SST}\rightarrow\ \mathrm{PYR}}$ are weights from SST to PYR, $\mathbf{W}^{\mathrm{VIP}\rightarrow\ \mathrm{SST}}$ are weights from VIP to SST, and $\mathbf{W}^{\mathrm{VIP}\ \rightarrow\ \mathrm{PYR}}$ are weights from VIP to PYR. VIP neurons project to PYR neurons directly via weights $\mathbf{W}^{VIP\rightarrow PYR}$ and indirectly via the SST population. The effects of the indirect pathway VIP-SST-PYR can be captured by taking the product of connectivities, yielding $\mathbf{W}^{SST\rightarrow PYR}\mathbf{W}^{VIP\rightarrow SST}$. The three unknown variables are then $\mathbf{f}^{\mathrm{t,VIP}}$, $\mathbf{W}^{\mathrm{VIP}\rightarrow\ \mathrm{SST}}$, and $\mathbf{W}^{\mathrm{VIP}\ \rightarrow\ \mathrm{PYR}}$, but since we assume $\mathbf{f}^{t,VIP}$ is constant in time t, this tensor can be combined with the connectivities to form the effective parameters

$$\mathbf{f}^{\alpha} = \mathbf{W}^{VIP\rightarrow SST}\mathbf{f}^{t-\Delta t,VIP} \tag{2.13}$$

and

$$\mathbf{f}^{\beta} = \mathbf{W}^{VIP\rightarrow PYR}\mathbf{f}^{t-\Delta t,VIP} \tag{2.14}$$

and hence reduce the number of unknowns and simplify notation. Our objective is to have firing rates in the switching circuit be as closely matched as possible to the firing rates in the separate moving circuit with $\mathbf{W}^{\mathrm{moving}}$:

$$\begin{aligned}\mathbf{r}^{\mathrm{moving},t} &= \mathbf{f}^t \circ (1 + \mathbf{W}^{\mathrm{moving}}\mathbf{f}^{t-\Delta t}) \\ &\approx \mathbf{f}^t \circ (1 + \mathbf{W}^{\mathrm{static}}\mathbf{f}^{t-\Delta t} + \mathbf{W}^{\mathrm{SST}\rightarrow\ \mathrm{PYR}}\mathbf{f}^{\alpha} + \mathbf{f}^{\beta})\end{aligned} \tag{2.15}$$

This amounts to minimizing the loss function defined by the approximation error $E_{\mathrm{switch},1}$ over the variables $\mathbf{f}^{\alpha}, \mathbf{f}^{\beta}$:

$$\min_{\mathbf{f}^{\alpha},\mathbf{f}^{\beta}} E_{\mathrm{switch},1} = \min_{\mathbf{f}^{\alpha},\mathbf{f}^{\beta}} \frac{1}{N}\sum_{f} ||(\mathbf{W}^{\mathrm{moving}} - \mathbf{W}^{\mathrm{static}})\mathbf{f} - \mathbf{W}^{\mathrm{SST}\ \rightarrow\ \mathrm{PYR}}\mathbf{f}^{\alpha} - \mathbf{f}^{\beta}||_F, \tag{2.16}$$

where $||\cdot||_F$ is the Frobenius norm of a tensor, for all $\mathbf{f}$ (firing rates due to classical receptive fields) corresponding to video frames, and N is a normalization factor, the number

of video frames in our dataset. $\mathbf{f}$ is inferred through our model from the datasets of video frames and features using $\mathbf{f}_j^n = p(\mathbf{F}_j^n|i^n) = i^n * \mathbf{F}_j$ and thus is a known quantity throughout the optimization. Importantly, since $\mathbf{f}^{t,VIP}$ are firing rates and hence $\mathbf{f}^{t,VIP} \geq 0$, while $\mathbf{W}^{\text{SST}\to\text{PYR}} \leq 0, \mathbf{W}^{\text{VIP}\to\text{SST}} \leq 0$, and $\mathbf{W}^{\text{VIP}\to\text{PYR}} \leq 0$, we have that $\mathbf{f}^\alpha, \mathbf{f}^\beta \leq 0$, and $\mathbf{W}^{\text{SST}\to\text{PYR}}\mathbf{f}^\alpha \geq 0$.

This is a high dimensional constrained optimization problem with the loss function defined as in (2.16), which we solved by means of a gradient descent method using the gradient-based Adam optimizer, implemented in pytorch [2]. The weights $\mathbf{f}^\alpha$ and $\mathbf{f}^\beta$ as defined in Equations (2.13) and (2.14) are unknown and learned by Stochastic Gradient Descent (SGD), while $\mathbf{W}^{\text{moving}}, \mathbf{W}^{\text{static}}$, $\mathbf{W}^{SST\to PYR} \equiv [\mathbf{W}^{\text{static}}]_-$ are fixed. Finding the global minimum of the loss function is difficult, but the main goal is to find weights that give a small enough error $E_{switch,1}$ instead and later test these on a specific task to demonstrate that the optimal moving circuit can be approximated successfully (sec. 2.3.6). We assessed the stability of our optimization by modifying several learning hyperparameters: learning rate (ranging from 0.001 to 0.1), optimization algorithm (SGD, AdaGrad, RMSProp, Adam), etc. and checking the generalization error on a small number of frames (50) that were not used during training.

Regardless of hyperparameters, our optimization procedure did not find weights that together approximate the moving circuit significantly better than the static circuit. In other words, adding VIP neurons in an attempt to switch contexts does not lead to a significantly better approximation of the moving circuit than having no VIPs. This result holds for both the simple dataset of horizontal and vertical bars, and for the more complex dataset of natural images and videos (figs. 2.6b to 2.6c).

In order to understand the origin of this failure, we mathematically analyzed the circuit at hand. Analytically, if the loss is small $\text{E}_{\text{switch},1} \approx 0$, then $(\mathbf{W}^{\text{moving}} - \mathbf{W}^{\text{static}})\mathbf{f} \approx \mathbf{W}^{\text{SST}\to\text{PYR}}\mathbf{f}^\alpha + \mathbf{f}^\beta$, where $\mathbf{f}$ is unique to each image in the data. The left hand side be-

[2] The tensor weights are very high-dimensional so that the least-squares method and variations thereof have failed due to the high memory requirements.

comes a term that varies across a wide range of video frames, while the right hand side is a constant term incorporating the weights we are solving for: $\mathbf{f}^{\alpha}, \mathbf{f}^{\beta}$. This suggests that the failure of our optimization procedure to yield weights that approximate the moving circuit results from the VIP having no stimulus dependence.

We conclude that the circuit switching between static and moving contexts must be more complex than the simple circuit here, which has only outgoing projections from VIP. Below, we introduce recurrent connections which make the VIP input-dependent, and overcome the limitations above.

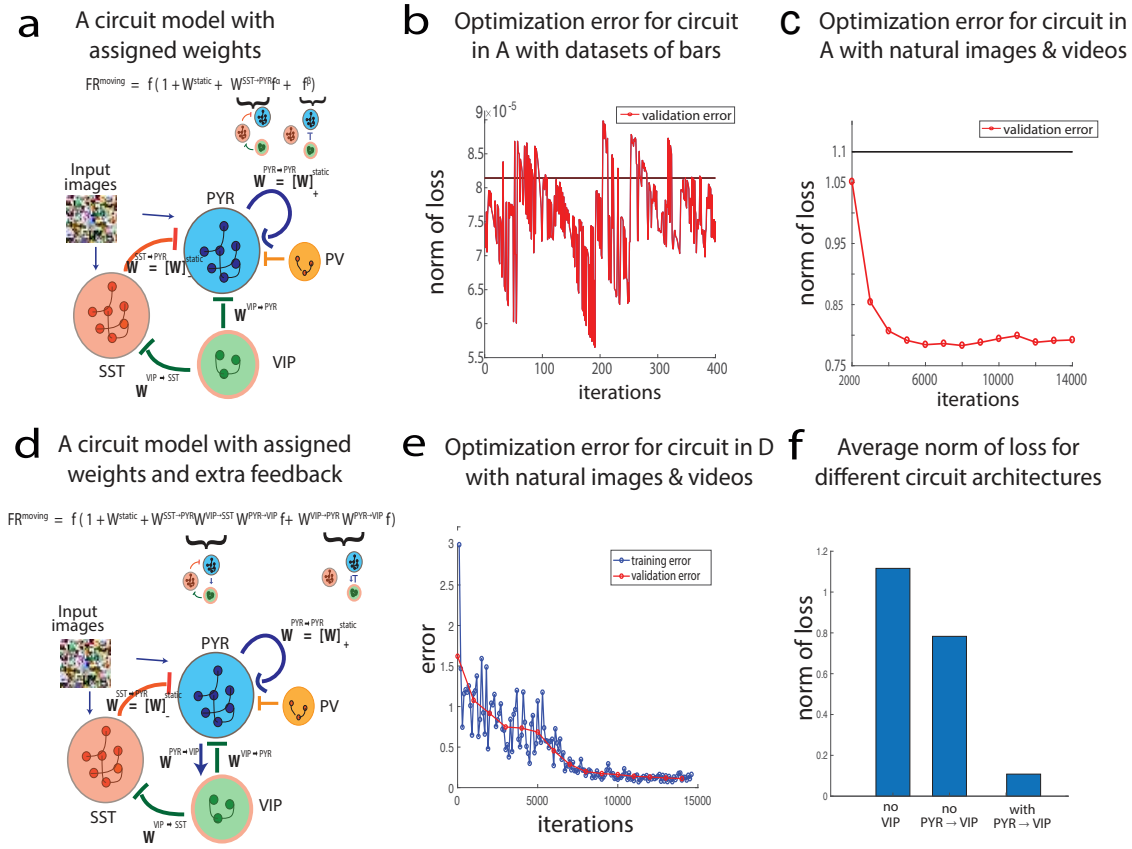


Figure 2.6: (a). Goal: instead of two separate circuits for visual processing of static and moving contexts, the proposed circuit approximates the static circuit when the VIP are silent and the animal is static, and the moving circuit when the VIP are active and the animal is moving. (b). Generalization/validation error found during the optimization to minimize the functional $E_{switch,1}$ for the datasets of static and moving bars does not converge. (c). Generalization/validation error found during the optimization to minimize the functional $E_{switch,1}$ for the datasets of natural images and videos converges, but the norm of the loss function decreases by only $\approx 25\%$. (d). Circuit as in (a), but with a feedback connection added from the PYR population to the VIP. (e). Training error (blue) and generalization/validation error (red) found during the optimization to minimize the functional $E_{switch,2}$ (movement approximation error) for the datasets of natural images and videos converges to yield a relatively small error. (f). The movement approximation error for various circuit architectures: the static circuit with no VIP switching units, the circuit depicted in (a) without PYR to VIP feedback, the circuit depicted in (d).

2.3.5 *VIP circuit with feedback from the PYR cells can switch context integration from static to moving conditions*

Above we showed that a minimal switching circuit with only outgoing projections from the VIP units is insufficient to switch between the two contexts. Hence, we added an additional connection between PYR and VIP, such that the VIP group of neurons has access to information about the visual input through PYR (Figure 2.5c). In this case we can approximate the firing rate of PYR during movement as follows, using the same conventions and assumptions as before (cfr. Methods sec. 2.5.5):

$$\begin{aligned}\mathbf{r}^{\text{moving},t} = \mathbf{f}^t \circ (1 + \mathbf{W}^{\text{static}}\mathbf{f}^{t-\Delta t} + \mathbf{W}^{\text{SST}\rightarrow\text{PYR}}\mathbf{W}^{\text{VIP}\rightarrow\text{SST}}\mathbf{W}^{\text{PYR}\rightarrow\text{VIP}}\mathbf{f}^{t-\Delta t} + \\ + \mathbf{W}^{\text{VIP}\rightarrow\text{PYR}}\mathbf{W}^{\text{PYR}\rightarrow\text{VIP}}\mathbf{f}^{t-\Delta t})\end{aligned} \tag{2.17}$$

We remind the reader that $\mathbf{f}$ is the contribution to the firing rate of the classical receptive field, $\mathbf{W}^{\text{X}\rightarrow\text{Y}}$ are the weights from population X of neurons to population Y of neurons, where X, Y are the PYR, SST, VIP neurons. In addition to the fixed $\mathbf{W}^{\text{static}}$ and $\mathbf{W}^{\text{moving}}$, we also fix $\mathbf{W}^{\text{SST}\rightarrow\text{PYR}} = [\mathbf{W}^{\text{static}}]_-$. A schematic of the underlying circuit model, along with the corresponding formula for the firing rate of PYR, is shown in Figure 2.6d.

We would like to find the three unknown weights $\mathbf{W}^{VIP\rightarrow PYR}$, $\mathbf{W}^{VIP\rightarrow SST}$, and $\mathbf{W}^{PYR\rightarrow VIP}$, to best achieve the approximation:

$$\begin{aligned}\mathbf{r}^{\text{moving},t} &= \mathbf{f}^t \circ (1 + \mathbf{W}^{\text{moving}}\mathbf{f}^{t-\Delta t}) \\ &\approx \mathbf{f}^{\mathbf{t}} \circ (1 + \mathbf{W}^{\text{static}}\mathbf{f}^{t-\Delta t} + \mathbf{W}^{\text{SST}\rightarrow\text{PYR}}\mathbf{W}^{\text{VIP}\rightarrow\text{SST}}\mathbf{W}^{\text{PYR}\rightarrow\text{VIP}}\mathbf{f}^{t-\Delta t} + \\ &+ \mathbf{W}^{\text{VIP}\rightarrow\text{PYR}}\mathbf{W}^{\text{PYR}\rightarrow\text{VIP}}\mathbf{f}^{t-\Delta t})\end{aligned} \tag{2.18}$$

We denote the approximated expression of (2.18) by $\mathbf{r}^{\text{approx}}$. This approximation $\mathbf{r}^{\text{approx}} \approx \mathbf{r}^{\text{moving}}$ amounts to minimizing the loss function defining the *movement approximation error* $E_{\text{switch},2}$:

$$E_{\text{switch},2} = \frac{1}{N}\sum_{f} ||(\mathbf{W}^{\text{moving}} - \mathbf{W}^{\text{static}})\mathbf{f} - \mathbf{W}^{\text{SST}\rightarrow\text{ PYR}}\mathbf{W}^{\text{VIP}\rightarrow\text{ SST}}\mathbf{W}^{\text{PYR}\rightarrow\text{VIP}}\mathbf{f} - \\ - \mathbf{W}^{\text{VIP}\rightarrow\text{PYR}}\mathbf{W}^{\text{PYR}\rightarrow\text{VIP}}\mathbf{f}||_F, \tag{2.19}$$

for all N frames whose corresponding classical receptive field firing rate is $\mathbf{f}$. In the case of simple images and videos of bars we consider $\mathbf{W} \cdot \mathbf{f}$ to be the regular matrix vector multiplication, while in the case of natural scenes we perform the convolution operation $\mathbf{W} * \mathbf{f}$. Applying convolution for natural images and videos fits with the assumption we have applied for the PYR, SST populations, that weights between neurons are translationally invariant, and further reduces the number of parameters.

To solve this high dimensional optimization problem, we set up, as in sec. 2.3.4, an optimization problem with the loss function being the average Frobenius norm as defined in (2.19). Weights to and from VIP are unknown ($\mathbf{W}^{\text{VIP}\rightarrow\text{SST}}$, $\mathbf{W}^{\text{VIP}\rightarrow\text{PYR}}$, and $\mathbf{W}^{\text{PYR}\rightarrow\text{VIP}}$) and learned by SGD, while $\mathbf{W}^{\text{moving}} - \mathbf{W}^{\text{static}}$, $\mathbf{W}^{\text{SST}\rightarrow\text{ PYR}}$ are fixed. Importantly, Dale's law is enforced ($\mathbf{W}^{\text{VIP}\rightarrow\text{SST}}$, $\mathbf{W}^{\text{VIP}\rightarrow\text{PYR}} \leq 0$, $\mathbf{W}^{\text{PYR}\rightarrow\text{VIP}} \geq 0$) for biological realism.

To find how many switching units are needed, we varied the number of VIP neurons, which was equivalent to varying the dimensionality of tensors $\mathbf{W}^{\text{VIP}\rightarrow\text{SST}}$, $\mathbf{W}^{\text{VIP}\rightarrow\text{PYR}}$, and $\mathbf{W}^{\text{PYR}\rightarrow\text{VIP}}$. We found the smallest number of switching neurons VIP that enabled the loss (2.19) to be minimized. We first considered the simple image/video dataset which was 9×9 with horizontal and vertical bars. In this case, the loss was minimized with at least 20 VIP neurons (Figure 2.7a). For comparison, there are 162 PYR and SST neurons, one for each filter and pixel in the image or frame. As increasing the number of VIP units further does not decrease the loss function, we conclude that, for the case of bar-like images, having 20 switching units is enough.

Second, in the distinct case of more complex stimuli like images and videos of natural scenes, the *movement approximation error* in (2.19) was minimized when the number of

VIP units is 34 per unit space, which matches the number of units in the PYR and SST population. However, the approximation error was already significantly minimized with only 5 VIP units per unit space, without any significant improvement after adding more units (Figure 2.7b). Varying the dimensionality of spatial components of the tensors (Figure 2.13) we were solving for ($\mathbf{W}^{VIP\to SST}$, $\mathbf{W}^{VIP\to PYR}$, $\mathbf{W}^{PYR\to VIP}$) and the synaptic delay Δt for sparse weights $\mathbf{W}$ that account for patch independence, we obtained the same qualitative results. Our results also hold for non-sparse weights, as shown in Figure 2.14. Fixing the number of VIP units to 5 per unit space, we find that the approximated firing rate of (2.18) matches $\mathbf{r}^{\text{moving}}$ compared to the $\mathbf{r}^{\text{static}}$ firing rates of a circuit without VIP units (Figure 2.7c). We conclude that for the specific parameters chosen in Figure 2.7b, the ratio of PYR to switching VIP units is $34/5 = 6.9$, so that the switching operation requires relatively few units, a fact we return to in the context of the underlying biology below.

All in all, we have shown that a switching circuit with relatively few numbers of switching VIP units and appropriate feedback connections can be implemented to achieve visual processing during the static and moving contexts, and for both a simple synthetic dataset of bars, and a biologically relevant dataset of natural images and videos.

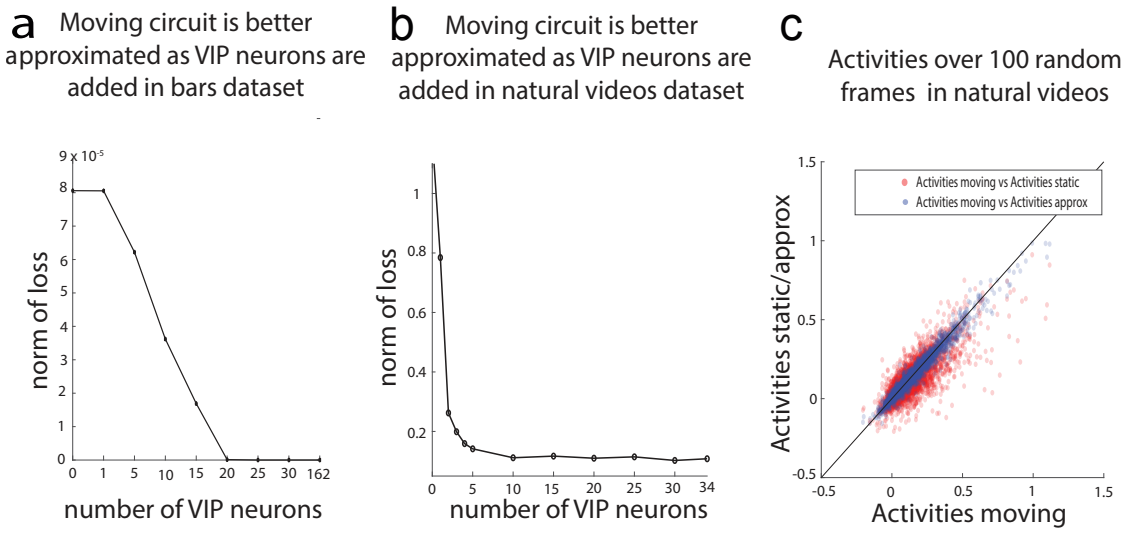


Figure 2.7: (a). Adding VIP switching units to the circuit processing videos of bars approximates the activity to that of the optimal circuit for moving context for this simple dataset. However, no more than 20 VIPs are needed in practice, compared to the 162 PYR and SST cells. (b). Adding VIP switching units to the circuit processing natural videos approximates the activity to that of the optimal circuit for moving context for the naturalistic dataset. However, no more than 5 VIPs per unit space are needed in practice, compared to the 34 PYR and SST cells per unit space. The parameters chosen for this optimization are $\Delta t = 2$ and $\dim(\mathbf{W}^{VIP\to SST}) = \dim(\mathbf{W}^{VIP\to PYR}) = 34 \times Nf_2 \times 3 \times 3$, $\dim(\mathbf{W}^{PYR\to VIP}) = Nf_2 \times 34 \times 3 \times 3$, where Nf_2 is the variable number of VIP units. (c). A random subset of activities corresponding to different video frames, filters, spatial positions for the static, moving, and approximated moving circuit. Red dots for activities for moving circuit ($\mathbf{r}^{\text{moving}}$) vs activities for static circuit ($\mathbf{r}^{\text{static}}$); blue dots for activities for moving circuit vs activities for approximated switching circuit ($\mathbf{r}^{\text{approx}}$). Activities are computed using weights with 5 VIP units/unit space. Activities chosen for the approximated switching circuit are able to better estimate the activities in the moving circuit in comparison to the ability of the activities in the static circuit to estimate the activities in the moving circuit.

2.3.6 Context-dependent visual processing with extra-classical receptive fields leads to denoising

According to our theory (Methods, sec. 2.5.1), the moving circuit achieves optimality of visual processing for videos, the static circuit achieves optimality of processing for static images, and we have found appropriate connectivities to and from a population of switching units — VIP — that can approximate either circuit in a model of V1, the *switching circuit*. We have however not yet assessed the performance of these circuits on specific visual processing tasks. We pursue this here for the task of denoising. Specifically, we ask how well (a) extra-classical receptive field contributions from the static or moving circuits (Figure 2.5a) can improve reconstructions of noisy images and videos and (b) whether the switching circuit can achieve the same level of performance as the separately optimized moving circuit when processing videos. We focus on reconstructions of video frames and the superior performance of the moving and switching circuits for processing moving contexts, although we also mention the comparably high performance of the static circuit, and implicitly that of the switching circuit responding to static scenes, for processing static contexts.

To reconstruct a visual scene during movement, our brain uses information from the present, but also time-delayed surround information, both of which can be inaccurate or incomplete. We use $\mathbf{W}^{\text{moving}}$ to weigh the past surround information, as these weights encapsulate the cross-correlational structure between features of the past and the present, thereby informing which features are more or less likely. We note that, during motion, using $\mathbf{W}^{\text{static}}$ to weigh surround information may still be better that using no surround information at all: if movement in the videos is slow enough, or Δt is small, features are smooth and $\mathbf{W}^{\text{static}}$ and $\mathbf{W}^{\text{moving}}$ are highly correlated.

To apply our models to the task of denoising, we apply Gaussian white noise or salt and pepper noise ξ to the original frames X of the videos (Figure 2.8a), and compute firing rates in the circuits in response to the noisy frames $X + \xi$. The firing rates are expressed as:

$$\mathbf{r}^{\text{no EXC}}(t) = \mathbf{f}^t \tag{2.20}$$

$$\mathbf{r}^{\text{static}}(t) = \mathbf{f}^t \circ (1 + \mathbf{W}^{\text{static}}\mathbf{f}^{t-\Delta t}) \tag{2.21}$$

$$\mathbf{r}^{\text{moving}}(t) = \mathbf{f}^t \circ (1 + \mathbf{W}^{\text{moving}}\mathbf{f}^{t-\Delta t}) \tag{2.22}$$

$$\mathbf{r}^{\text{approx}}(t) = \mathbf{f}^t \circ (1 + \mathbf{W}^{\text{static}}\mathbf{f}^{t-\Delta t} + \mathbf{W}^{SST\to PYR}\mathbf{W}^{VIP\to SST}\mathbf{W}^{PYR\to VIP}\mathbf{f}^{t-\Delta t} + \\ + \mathbf{W}^{VIP\to PYR}\mathbf{W}^{PYR\to VIP}\mathbf{f}^{t-\Delta t}) \tag{2.23}$$

We denote "EXC" throughout the figures and text to represent the extra-classical receptive field contribution. Hence, $\mathbf{r}^{\text{no EXC}}$ is the firing rate due to only the feedforward pathway, with no lateral connections, and thus without any extra-classical, surround modulation. In the case of $\mathbf{r}^{\text{static}}$ ($\mathbf{r}^{\text{moving}}$), $\mathbf{W}^{\text{static}}$ ($\mathbf{W}^{\text{moving}}$) weights are the lateral connections applied that weigh the extra-classical receptive field information from the past surround. While $\mathbf{W}^{\text{static}}$ are non-optimal weights to compute the firing rate, $\mathbf{W}^{\text{moving}}$ are optimal for inferring features in noisy conditions as described below (cfr. Methods sec. 2.5.1). Finally, $\mathbf{r}^{\text{approx}}$ results from lateral connections from our switching circuit with connections to and from VIP.

For each image frame X we computed the corresponding firing rate $\mathbf{r}$ via equations (2.20) - (2.23), to obtain a tensor with entries for every filter and spatial position of X. We then deconvolved $\mathbf{r}$ for each filter $\mathbf{F}_j$ (Methods sec. 2.5.6) along its corresponding dimension to obtain the "reconstructed" frame X':

$$X + \xi \to \mathbf{r} \to X' \tag{2.24}$$

Although there are ways for a biological circuit to do more accurate reconstructions (e.g. via learning weights), we have chosen a simple reconstruction approach that does not require additional assumptions here (e.g. the circuit does not know the structure of the noise or the input), as described in Methods sec. 2.5.6.

We compare the quality of reconstructions from the four circuit models above. The baseline

for these comparisons is the reconstruction of a noiseless image frame ($\xi = 0$), where the extra-classical contribution does not provide any additional information. (Note that this reconstruction X' is not the same as the original frame X, as all feature information not included in the filters is lost in the initial convolution of the image frame to get $\mathbf{r}$). We denote by $\rho(\cdot)$ a metric of the quality of the reconstruction. This takes the firing rate $\mathbf{r}$ as input, and generates the Pearson correlation coefficient between the reconstruction X' and the baseline reconstruction described above as output. The metric ρ for a video frame with noise ξ is

$$\rho(\mathbf{r}) = Corr(X'_\xi, X'_{\xi=0}) = \frac{(X'_\xi - \bar{X}'_\xi) \cdot (X'_{\xi=0} - \bar{X}'_{\xi=0})}{||X_\xi - \bar{X}_\xi||_2 ||X_{\xi=0} - \bar{X}_{\xi=0}||_2} \tag{2.25}$$

where $\cdot$ is the dot product, and $\bar{X}, \bar{X}'$ are the means of the image and reconstruction, respectively. The upper limit for correlation coefficient is 1 when there is no noise in the image or frame (Figure 2.8f).

Thus equipped, we ask which circuit architecture gives rise to neural activity best suited for decoding visual scenes in noisy conditions. Figure 2.8a shows reconstructions of a video frame using different such circuit architectures. We expect $\rho(\mathbf{r}^{\text{no EXC}}), \rho(\mathbf{r}^{\text{static}}) < \rho(\mathbf{r}^{\text{moving}}), \rho(\mathbf{r}^{\text{approx}})$ on average, as $\mathbf{W}^{\text{moving}}$ are the optimal lateral connections as defined above. However, the exact relationship between $\rho(\mathbf{r}^{\text{no EXC}})$, $\rho(\mathbf{r}^{\text{static}})$, $\rho(\mathbf{r}^{\text{moving}})$, $\rho(\mathbf{r}^{\text{approx}})$ depends on the exact correlational structure of the frames for each video. Some videos match our prediction that $\rho(\mathbf{r}^{\text{moving}})$ is maximized (Figure 2.8b), while other videos do not (Figure 2.8c). Specifically, there are videos where surround modulation is not effective, which appears to be due to the presence of independent features where the information in the extra-classical receptive field does not aid image reconstruction.

On average throughout the videos, $\mathbf{r}^{\text{moving}}$ and $\mathbf{r}^{\text{approx}}$ yield the best reconstructions (dark and light green bars), displaying the highest cross-correlation coefficients ρ between the noiseless reconstruction (the baseline) and the reconstructed frames (Figure 2.8d). Figures 2.8d and 2.8e show this holds true when adding to the original frames either salt and pepper noise, when we varied the proportion of pixels occluded, or Gaussian white noise,

when we varied the standard deviation of the normal distribution of noise. The relation $\rho(\mathbf{r}^{\text{no EXC}}), \rho(\mathbf{r}^{\text{static}}) < \rho(\mathbf{r}^{\text{moving}}) \approx \rho(\mathbf{r}^{\text{approx}})$ is robust to the amount of noise added to the frames (Figure 2.8f), whether for salt and pepper noise or Gaussian noise. This holds true both when the complete set of 34 spatio-temporal filters is used (Figure 2.19a), and when only the set of 18 filters with no temporal component is used (Figure 2.19b). As expected, the addition of filters with a temporal component improves the reconstruction performance in all the four circuit architectures presented (Figure 2.19c). Furthermore, reconstruction performance for images in the static condition is maximized on average using $\mathbf{W}^{\text{static}}$ to weigh the surround so that $\rho(\mathbf{r}^{\text{no EXC}}), \rho(\mathbf{r}^{\text{moving}}), \rho(\mathbf{r}^{\text{approx}}) < \rho(\mathbf{r}^{\text{static}})$ on average (Figure 2.18). This shows that the moving circuit is best used for processing noisy video frames and that the static circuit (or switching circuit with VIP silent) is ideally used for processing images at the highest performance.

Thus the switching circuit provides reconstruction performance comparable to a dedicated moving circuit for videos and comparable to a dedicated static circuit for images. In the case of videos, this is because the switching circuit reproduces firing rates that are close enough to $\mathbf{r}^{\text{moving}}$ to improve reconstruction fidelity. The correlation coefficients found between noiseless baseline reconstructions and reconstructions due to the moving and switching circuits, respectively, present almost perfect overlap (light and dark green curves in figs. 2.19a to 2.19b). In sum, we conclude that the extra-classical receptive field contribution in the moving circuit and approximated switching circuit generates neural activity that can be decoded to produce more accurate frame reconstructions in videos. To produce the most accurate **image** reconstructions, the VIP neurons in the switching circuit must be silent so that the network implements the static circuit.

2.3.7 Experimental evidence of VIP role in movement-related visual coding

When we examine the weights W to and from the VIP we have inferred in our model, we find that there are a few, equally correct solutions for the optimization problem (2.19) due to the multiple local minima of the movement approximation error. One of the pos-

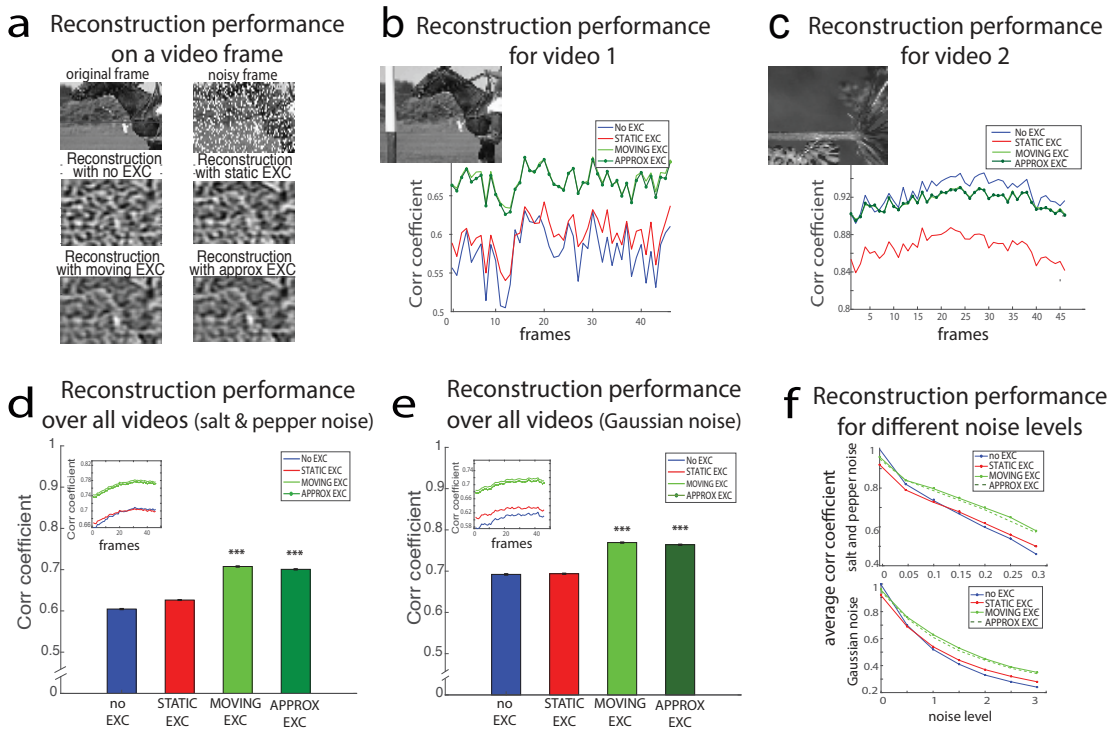


Figure 2.8: (a). Example of a reconstructed frame for each condition/circuit architecture: no EXC, static EXC, moving EXC, approximated EXC. (b). Average correlation coefficients between reconstructed noisy frames and reconstructed noiseless frames for one video in our dataset. Here reconstruction benefits from surround contextual information. (c). Same as (a), but in this case the general inequality that holds on average $\rho(\mathbf{r}^{\text{no EXC}}), \rho(\mathbf{r}^{\text{static}}) < \rho(\mathbf{r}^{\text{moving}}) \approx \rho(\mathbf{r}^{\text{approx}})$ breaks down and $r(\mathbf{r}^{\text{no EXC}}) \approx r(\mathbf{r}^{\text{moving}})$. (d). Average correlation coefficient over all frames and all videos after salt and pepper noise was added to the video frames. The probability is 0.2 each pixel is changed to white and 0.2 each pixel is changed to black, and $\Delta t = 2$ (frames). The moving and approximated EXC average correlation coefficients are higher than for static EXC or no EXC (p-value < 0.05 using the Wilcoxon rank-sum test for all relevant comparisons). Inset: Correlation coefficients in time, averaged across videos. (e). Same as (d), for Gaussian white noise with 0.5 standard deviation. $\Delta t = 2$ (frames). $p < 0.05$ for all relevant comparisons, Wilcoxon rank-sum test. (f). Average correlation coefficient over frames and videos as noise level is varied. Top: Salt and pepper noise is varied; Down: Gaussian white noise std is varied.

sible solutions we found matched experimental data showing that in various layers of V1, the VIP to SST connection is strong compared to other connections, specifically the VIP to PYR connection (Figure 2.15). Interestingly, this property arose only when including weights from SST to VIP in the circuit, consistent with experiments (Pfeffer et. al., 2013 found the connection probability/strength from SST to VIP to be strong). Including this connection in our circuit and re-writing the circuit equations as in (2.49), we obtain a new set of connectivity patterns and activities so that we can now compare predictions of our model switching circuit to the extensive empirical evidence from the literature.

Importantly, we have not meticulously explored the set of all possible solutions from the optimization problem (2.19), and further the optimization may allow additional constraints to the switching circuit while still admitting solutions. Acknowledging this, we now study both the connectivity and activity of the switching circuit with an additional SST to VIP connection.

Connectivity. We find that our model produces connectivity patterns that are largely consistent with empirical findings, as we describe next. Connection weights in the model can be interpreted as corresponding to a combination of connection probabilities and connection strengths in the data, as these have been shown to correlate well (Cossell et. al., 2015). Regarding the aforementioned connection from the VIP to SST, experimental data on connectivity in the visual cortex from Pfeffer et. al. has shown that in layer 4 of V1, the average connection probability from VIP to SST is double the connection probability from VIP to PYR (0.625 compared to 0.351), while in layer 5, VIP to SST is 5 times more probable (0.625 compared to 0.125) (Pfeffer et. al., 2013). A recent study by Campagnola et al. (2021) has confirmed the relative paucity of VIP to PYR connections as compared to VIP to SST connections throughout all layers, for example finding 3 out of 52 VIP to PYR versus 5 out of 33 VIP to SST inter-area L2/3 connections (Campagnola et al., 2021). VIP to SST connections are also stronger than VIP to PYR throughout all the layers: 0.32 compared to 0.28 as found by Jiang et. al., 2015 and 0.3 compared to 0.21 as found by Campagnola et. al., 2021.

We next examine the distribution of connectivity patterns in our computational model, as displayed in Figures 2.9a to 2.9b, and compare these model findings with experimental results. As found empirically, VIP to PYR connections in our model are sparser than VIP to SST connections – with a large peak at 0 in the connectivity histogram – in addition to being on average weaker (0.38 vs. 0.47 for average weights in our model). Despite their sparsity, our model predicts a long tail to the distribution of VIP to PYR connection strengths. In addition, our model also predicts very high variability of $\mathbf{W}^{VIP\to PYR}$ connection strengths averaged with respect to the filter (which represents the post-synaptic cell type) as shown in Figure 2.9c. The strong connections correspond to the vertically oriented filters, as detailed below. We conclude that our model agrees with previous measurements and makes further predictions on the V1 microcircuit connectivity when including weights from SST to VIP.

We next inquire whether the synapses encode the contextual statistics by probing like-to-like connectivity both between PYR neurons, and between VIP and PYR populations of neurons (cfr. Methods sec. 2.5.9). We find that while there is like-to-like connectivity between PYR neurons as found by Iyer et al. (2020), this effect is largely absent between the VIP and PYR. To further examine the pattern of connectivity from the VIP, we correlate both $\mathbf{W}^{VIP\to PYR}$ and $\mathbf{W}^{VIP\to SST}$ to $\mathbf{W}^{\text{moving/static}}$ and $\mathbf{W}^{\text{moving}} - \mathbf{W}^{\text{static}}$, because these later weights reflect the statistical regularities of the static and moving contexts. We obtain that, after averaging over pre-synaptic filters (Nf_2) and the spatial receptive fields, $\mathbf{W}^{VIP\to PYR}$ correlates positively with $\mathbf{W}^{\text{moving}} - \mathbf{W}^{\text{static}}$ (0.41, pval < 0.02, two-sided ttest); while $\mathbf{W}^{VIP\to SST}$ also correlates positively, the correlation coefficient is weaker and not statistically significant. Similarly, the convolution $\mathbf{W}^{PYR\to VIP\to PYR} \equiv \mathbf{W}^{VIP\to PYR} * \mathbf{W}^{PYR\to VIP}$ also correlates positively with $\mathbf{W}^{\text{moving}} - \mathbf{W}^{\text{static}}$ (0.15, pval < 0.01, two-sided ttest).

Analysing the average post-synaptic weights $\mathbf{W}^{\text{moving}} - \mathbf{W}^{\text{static}}, \mathbf{W}^{VIP\to PYR}$, $\mathbf{W}^{PYR\to VIP\to PYR}$ more specifically, we find that the strongest connections are for inhibited post-synaptic units corresponding to vertical or diagonal filters. Looking at the strongest inhibitory weights for $\mathbf{W}^{VIP\to PYR}$ for example (Figure 2.9c), we find that 6/10 correspond to post-synaptic vertical filters and 9/10 to either vertical or diagonal post-synaptic filters.

For $\mathbf{W}^{\text{moving}} - \mathbf{W}^{\text{static}}$ and $\mathbf{W}^{PYR \to VIP \to PYR}$, 7 out of 10 such filters are vertical or diagonal (Figure 2.20). We note also that the average connection strength of $\mathbf{W}^{\text{moving}} - \mathbf{W}^{\text{static}}$ for post-synaptic vertical filters is negative and stronger ($-5.4 \cdot 10^{-5}$) compared to that for horizontal filters ($2 \cdot 10^{-6}$). This can be interpreted as follows: our videos feature horizontal movement hence the spatio-temporal co-occurrence for vertical features in particular will be distorted during the moving context; this results in weaker $\mathbf{W}^{\text{moving}}$ weights overall when the post-synaptic cell responds to vertical filters and thus $\mathbf{W}^{\text{moving}} - \mathbf{W}^{\text{static}}$ weights are strongly negative on average for such filters. The overall positive correlation of $\mathbf{W}^{\text{moving}} - \mathbf{W}^{\text{static}}$ with $\mathbf{W}^{VIP \to PYR}, \mathbf{W}^{PYR \to VIP \to PYR}$ determines that post-synaptic units tuned for vertical features be more strongly inhibited through these connections when switching from static to moving contexts. This phenomenon is more prevalent for $\mathbf{W}^{VIP \to PYR}$ where more of the strongest connections (9/10) are driven at least in part by inhibition of vertically tuned units, and in contrast with $\mathbf{W}^{\text{static}}$ and even $\mathbf{W}^{\text{moving}}$, where the strongest inhibitory connections are mostly for horizontally tuned units (5/5 and 5/5 respectively of top inhibitory filters are either horizontal or diagonal) while $\mathbf{W}^{\text{static}}$, $\mathbf{W}^{\text{moving}}$ connections for vertically tuned units are mostly excitatory (on average $1.4 \cdot 10^{-4}$ and $8.7 \cdot 10^{-5}$, respectively).

Activity. We next study the consistency of activity patterns produced by our model with respect to empirical data. Published experimental findings provide strong evidence that the VIP inhibitory population acts to modulate the visual circuitry in a movement dependent manner (Niell and Stryker, 2010; Pfeffer et. al., 2013; Fu et. al., 2014). Very recent results show that VIP neurons respond synergistically to stimuli moving front to back during locomotion, a conjunction expected during locomotion in a natural environment for mice, with a preference for low but non-zero contrasts (Millman et. al., 2020). Such movement-modulated activity matches the one required in our models, although we have not endowed the VIP units with specific feature selectivity. Additionally, we perform a set of new analyses of experimental data in the context of our model. These draw both on the literature and on the Allen Brain Observatory (2015), which contains in vivo physiological activity in the mouse visual cortex, featuring representations of visually evoked Calcium responses from GCaMP6-expressing neurons in selected cortical layers, visual areas, and Cre lines.

The dataset contains calcium activations across multiple experimental conditions, and here we focus on periods of spontaneous activity, natural images, and drifting gratings.

Our model of the switching circuit shows that the relative number of VIP neurons required to switch between moving and static contexts is relatively low when compared the number of PYR or SST neurons (Figures 2.7a to 2.7b). This number qualitatively matches the relative abundance of neurons in the three populations. Excitatory neurons PYR are more abundant than inhibitory ones (roughly 80% to 20%), and VIP are a minority of inhibitory cells. Moreover, the existing VIP cells recorded in the Allen Observatory do not appear to exploit substantially more degrees of freedom (as measured by their relative dimensionality) than other cell populations (Figure 2.19a), consistent with a small number of effective VIP "units."

We now highlight two aspects of VIP neural activity which are directly related to our model and which justify the choice of VIP as switching units whose activities are modulated by the locomotion state of the animal. First, VIP activity dimensionality is significantly modulated across the moving and static conditions during periods of spontaneous activity, as shown in Figure 2.9d and Figure 2.9e. To extract such dimensionality modulation, we considered periods of spontaneous activity in the recordings and divided the statistical distribution of the animal's speed, for each experimental session, into 4 quartiles. We then computed the average *dimensionality*, or Participation Ratio (PR, cf. Methods sec. 2.5.8) for each recording in each quartile, which we define here as the (lower) dimension of a subspace where the data of activations can be represented while retaining some meaningful properties of the original data. We define the "dimensionality modulation" to be the ratio between the average dimensionality distribution within the highest quartile (movement condition) and the average within the first quartile (static condition). Such ratio is displayed in Figure 2.9e. The dimensionality of the VIP population is significantly modulated by movement, while in other populations the same quantity was not significantly different across moving and static conditions (Figure 2.9d). The histogram of such statistics is shown in Figure 2.9e.

Second, we analyzed evoked activity during the animals' viewing of natural scenes. We performed a Calcium signal modulation analysis and found that, for this stimulus set, the activity was strongly modulated for the VIP population and less so for other neural populations (Figure 2.9f) across moving and static conditions assessed via the quartile method just described. This further confirms the stronger VIP modulation across the moving-static conditions. Further pieces of experimental evidence are presented in Figure 2.21.

Finally, we analyzed the activities of VIP and PYR neuron populations. Similarly to Niell and Stryker (2010), we find the activity of the PYR during the moving condition to be higher than the stationary condition on average (0.066 vs. 0.074, pval$<$ 0.01). However, our PYR population activity does not double during locomotion compared to periods of stationarity, as in Niell and Stryker (2010). More recent studies however have reproduced the relation between excitatory neuronal activity in mouse visual cortex and running, but have observed a much weaker relation (Millman et al. 2020, Figure 5e).

We conducted further analysis to infer the tuning properties of the PYR and the VIP. This was achieved by considering a wavelet family (e.g., Daubechies), taking the 2-dimensional discrete wavelet transforms of the video frames in our data, regarding the corresponding average wavelet transforms as features, and finally performing a linear regression or GLM against VIP or PYR activities with the average wavelet transforms as the independent variables (cfr. Methods sec. 2.5.9). We find that most PYR neurons are tuned to horizontal features, and much less to vertical features. Because VIP neurons in our model only get input from the PYR, while the top-down input activating VIP is described simply by the binary term s_t, VIP acquires the same preferential selectivity to horizontal features over and above that to vertical features (Figure 2.22, Methods sec. 2.5.9 for details). This is counter to what we would expect if the VIP were capable of detecting the horizontal movement in our dataset by exhibiting preferential selectivity towards vertical features within their receptive fields instead of through the ad-hoc bulit-in switch term s_t. We conclude the simplification used by employing a binary term s_t in Eq. (2.52) prevents us from observing a more realistic VIP activation pattern that would deviate from the PYR pattern

and provide further insights. This points to an important direction for future, and to more detailed modeling expanding on our current simplified model.

Altogether these comparisons provide further support for our modeling assumptions, and for the role of VIP neurons in visual coding across static and moving conditions. We conclude that our switching circuit model reproduces the global pattern of interactions via VIP that we expect, approximating the static and moving circuits, synchronal with VIP activation. Further analysis of future datasets, as examined in the Discussion section, will guide next steps of circuit modeling.

2.4 Discussion

We have introduced a computational model for V1 circuitry that uses multiple cell types to integrate contextual information into local visual processing, during two different — static and moving — contexts. We have identified a need for recurrence, leading to the architecture of a *switching circuit* with bidirectional, learned connections to a switching population (here, the VIP cell class). Beyond V1 and biological circuit modeling, this circuit may be useful in searching for artificial neural network (ANN) architectures that can operate in different contexts and switch effectively between them.

Our model connects to a body of recent empirical studies elucidating V1 neural cell types and network logic. First, Niell and Stryker (2010) have established that as the speed of mice increases, the circuit increases spiking overall and changes the frequency content of local field potentials (Niell and Stryker, 2010). Potentially, distinct activity patterns during locomotion could be attributed to effects from eye movements, however Niell and Stryker provide evidence against this hypothesis. These findings prompt us to model the network as a switching circuit that adapts its activity as the state of the animal changes from static to moving. Later studies have focused on the connection strengths for excitatory and inhibitory neurons: neurons display "like-to-like" connectivity (Cossell et. al., 2015; Ko et. al., 2011), whereby neurons with similar orientation tuning have a higher probability of connecting and display stronger connections on average. Pfeffer et al. describe the V1 circuit

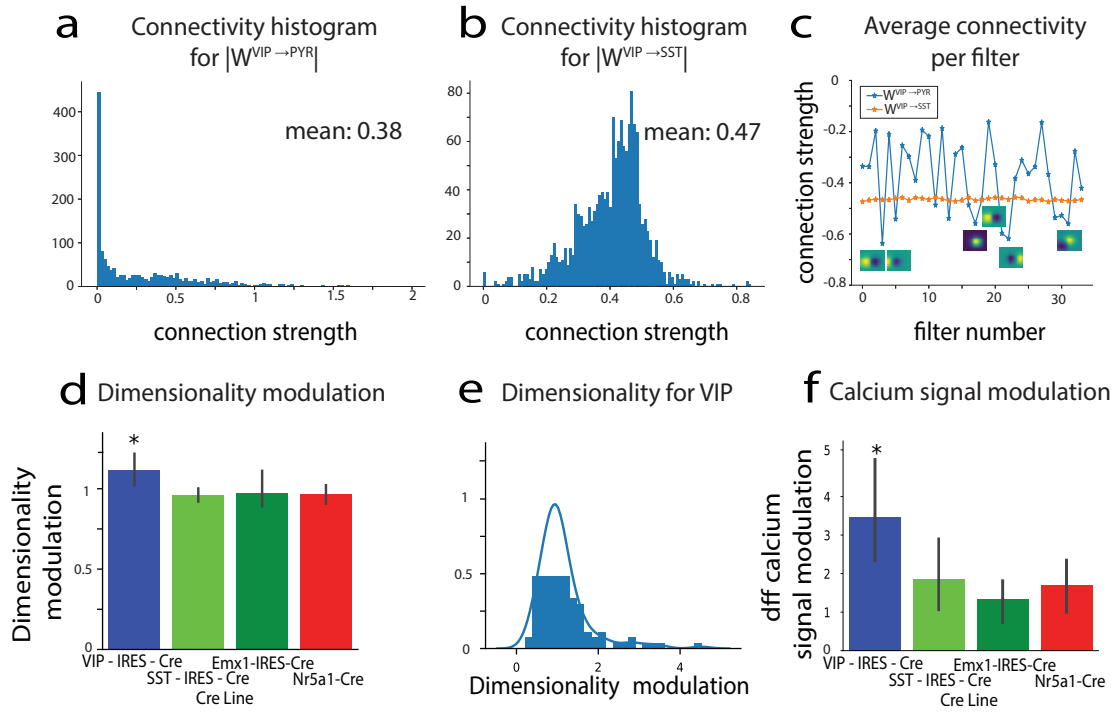


Figure 2.9: (a)) - (c)) Analysis of model connectivities $\mathbf{W}^{VIP\to SST}$, $\mathbf{W}^{VIP\to PYR}$. (a)) Histogram of the absolute value of connectivities for $\mathbf{W}^{VIP\to PYR}$ showing a mean of 0.31. (b)) Histogram of the absolute value of connectivities for $\mathbf{W}^{VIP\to SST}$ showing a mean of 0.4. (c)) Average connectivity per filter – corresponding to the post-synaptic cell type – for $\mathbf{W}^{VIP\to PYR}$ (blue) and $\mathbf{W}^{VIP\to SST}$ (orange). Filters for post-synaptic units corresponding to the strongest conectivities are displayed to show what units are strongly inhibited during movement. (d)) - (f)) Data analysis of VIP population activity in calcium imaging data. (d)) Dimensionality ratio (Participation Ratio measure) during periods of spontaneous activity between movement and static conditions across CRE lines. (e)) Histogram of the modulation of dimensionality (statistics relative to the blue bar in panel ((d))). (f)) Activity (dff signal) ratio during periods of natural images viewing between movement and static conditions across CRE lines.

logic by using transgenic mouse lines expressing fluorescent proteins or Cre-recombinase, providing a consistent classification of cell-populations across experiments (Pfeffer et. al.,

2013). Three large non-overlapping classes of molecularly distinct interneurons that interact via a simple connectivity scheme were identified: PV, SST, and VIP inhibitory neurons. In particular, PV inhibit one another, SST avoid one another and inhibit all other types of interneurons, and VIP preferentially inhibit SST cells.

Another important development made by Fu et al. (2014) has established that locomotion activates VIP neurons independent of visual stimulation and predominantly through nicotinic inputs from basal forebrain. This study was the first to propose the existence of a cortical circuit for the enhancement of visual response by locomotion, describing a modulation of sensory processing by behavioral state. These studies motivate us to choose VIP as switching units and to map the positive and negative weights of our model to connectivities between different neuronal populations. Finally, another study suggests that differentiated network response during locomotion can be advantageous for visual processing (Dadarlat and Stryker, 2017): an increase in firing rates can enhance the mutual information between visual stimuli and single neuron responses over a fixed window of time, while noise correlations decrease across the population which further improves stimulus discrimination. The authors hypothesize that cortical state modulation due to locomotion likely increases visually pertinent information encoded in the V1 population during times when visual information changes rapidly, such as during movement.

At least one study (Dipoppa et. al., 2018) has disputed the findings of Neill and Stryker and of Fu et. al., finding contrary evidence to the disinhibitory model. Experiments with the light on and visual stimuli present showed that locomotion increased both SST responses to large stimuli and VIP responses to small stimuli. However, the authors note that re-running the measurements in darkness reproduced results from Fu et. al., reinforcing the assumption that our model operates in conditions of poor visibility and high noise.

There is a vast literature on models of efficient coding starting with Barlow (1961), Attneave (1954) (for a great description of this literature see Chalk, Marre, and Tkačik, 2018). On one extreme, if the signal to noise ratio is high and additional constraints (e.g. sparsity)

are introduced, such models emphasize redundancy reduction (Olshausen and Field, 1996 a; Rao and Ballard, 1999; Harper and Prager, 1996; Comon, 1994; Bell and Sejnowski, 1995; Zemel, 1993; Dayan et. al., 1995). At the other extreme, if the signal to noise ratio is low, such models emphasize robust coding (Karklin and Simoncelli, 2011; Doi and Lewicki, 2014). We use a theoretical framework that emphasises robust coding and that we have selected because of its generality. It starts with an assumption on neuronal activation functionality (i.e. firing rates of neurons encode the probability of specific features being present in a given location of the image). This model describes local circuit interactions needed for integration of information from surrounding visual stimuli in noisy conditions for an arbitrary representation. The model matches multiple empirical findings, for example that statistical regularities of natural images give rise to "like-to-like" local circuit connectivities, as observed experimentally (Cossell et. al., 2015; Ko et. al., 2011). However, in different contexts the model predicts different functional lateral interactions. Therefore, we looked at circuits which can implement multiple functional interactions in one circuit.

Our model also relates to other switching circuits reported in the experimental literature. For example, selective inhibition of a subset of neurons in the central nucleus of the amygdala (CeA) led to decreased conditioned freezing behavior and increased cortical arousal as visualized by fMRI (Gozzi et. al., 2010). This, therefore, identifies a circuit that can shift fear reactions from passive to active. Another study has unraveled the cellular identity of the neural switch that governs the alternative activation of aggression and courtship in Drosophila fruit flies (Koganezawa et. al., 2016). While these studies detail circuits responsible for switching behaviors, there are circuits switching between contexts: from *detection* of weak visual stimuli to *discrimination* after adaptation in mice (Ollerenshaw et. al., 2014); from high response firing during active whisker movement, to low response when no tactile processing is initiated (Zhou et. al., 2017); from odor attraction in food-deprived larva switching to odor aversion in the well-fed larva (Vogt et. al., 2020).

In contrast to this rich body of experimental studies, there are relatively few computational models proposed so far that explain the switching of circuits (Yang et. al., 2019). We

may compare our V1 circuit to the recurrent circuits utilizing FORCE learning, where a single unit or few units project their feedback onto a recurrent neural net and momentarily disrupt chaotic activity to enable training. VIP units in our model precisely resemble such output units providing feedback in the FORCE framework, but it is unclear how far this analogy goes and to what extent the framework in Sussillo and Abbot (2009) is helpful in understanding V1 circuitry.

Another interesting example of a circuit with flexible, context-dependent behavior has been proposed by Mante et. al. (2013), where pre-frontal cortex (PFC) activity is modulated by the presence of a visual cue signaling which feature (color vs direction) the animals must integrate in a random-dots decision task. PFC functionality in this task has been modeled using a recurrent neural network (RNN) that takes the direction of motion, color of random dots, and visual cue as input, and outputs the appropriate, reward-generating direction to saccade. This suggests the RNN enacts a potentially new mechanism for selection and integration of context-dependent inputs, with gating possible because the representations of the inputs and the upcoming choice are separable at the population level, even though they are deeply entangled at the single neuron level. The architecture of the model RNN proposed in this study is simpler than what we have laid out, while also attaining high flexibility. There are important differences between the framework outlined in this paper and our work: first, it is unclear what the number of weights in the network might be for the circuit in Mante et. al. (2013) to be multitasking. One of our main motivations has been to achieve a switching circuit with few added units and weights so that the circuit has fewer weights to learn than two separate circuits processing the two contexts independently. It is unclear if this potential advantage holds in the case of Mante et al. Second, our circuit adapts to the statistics of both static and moving scenes and yields firing rates that are optimal for visual processing in either context. In the case of Mante et al., the circuit does not change momentary input processing when the context changes, it simply adapts its dynamics to integrate the appropriate feature and initiate the action that will be rewarded. Context takes on different meanings in these two instances: in our model, context is given by the statistical regularities of a certain environment, static or moving; in Mante et al. context refers to an input cue that changes the goals and reward dependencies of actions within the

task. Importantly, we have focused on switching circuits that modulate their responses to different sensory contexts, as opposed to different input cues and behaviors. It is unclear whether identical or different mechanisms for switching apply in the case of sensory processing or action selection, when the animal changes scene statistics or behaviors, respectively.

Although our model is faithful to some aspects of the biology of V1 circuits, it has several limitations. First, it has been reported that during animal locomotion, firing rates of neurons more than double, at least in layers II/III of V1. Our firing rates are normalized to sum to one across features and cannot reproduce a doubling occurring uniformly over features. Second, the model does not reproduce a few experimental findings as reported in Ayaz et. al. (2013) and Keller et. al. (2020). For instance, locomotion does not increase spontaneous activity as found by sequentially showing, to the static and moving circuits, images where every pixel takes on a constant value or images with Gaussian white noise (0.13 vs. 0.11 mean static, moving activity for constant pixel images; 0.063 vs. 0.53 mean static, moving activity for Gaussian white noise images/videos). Similar to Keller et. al., the firing rate due to the cross-oriented surround is only slightly higher than the firing rate due to the iso-oriented surround (0.087 vs. 0.085, pval$<$ 0.015, see Figure 2.23 for stimuli shown to the circuits). However, locomotion does weaken signals conveying surround suppression as reported in Ayaz et. al. through the inhibition of the SST population by the VIP.

Moreover, another study (Dadarlat and Stryker, 2017) reported that noise correlations are reduced during motion, but this does not occur in our model. Further, we model VIP as a switch which is off during the static condition and has an activation during locomotion dependent on input images, whereas data shows VIP activity is modulated at a finer scale and correlates strongly with speed (Fu et. al., 2014). In addition, VIP switching units in our model turn on based on perfect knowledge of whether the animal is static or moving, rather than based on more subtle time-varying visual or motor features. Furthermore, data from Ko et. al. (2011), Pfeffer et. al. (2013), Jiang et. al. (2015), Hofer et. al. (2011), Lefort et. al. (2009), Thomson et. al. (2002), Cauli et. al. (1997) on connection probabilities and strengths between neuron populations presents a richer, more complex picture

than our simplified circuit. There is wide-ranging connectivity to and from PV, there are strong connections from PYR to SST in most layers, and the weights from SST to VIP are strong (in terms of both connection probability and strength across layers), details that our simplified model cannot describe. Enabling weights from SST to VIP showed that we can similarly infer weights to and from VIP so that we are able to approximate the circuit during the moving condition (figs. 2.15a to 2.15b). However, there are still many more potential connectivity structures between neuron populations our model does not describe.

From a computational perspective, our model makes several simplifications in describing context integration in circuits tuned to the statistical regularities of natural scenes. These include approximating a product with a sum in Equation (2.37) in Methods and ignoring higher order surround modulation going from Equation (2.31) to (2.33) in Methods. Furthermore, our equations have omitted terms explicitly describing feedback from higher-order areas. Top-down input to the VIP that mediates an increase of local PYR activity has been reported for example in Zhang et. al. (2014), Wilmes et. al. (2019), Hertäg et. al. (2019), Batista-Brito et. al. (2018), Wall et. al. (2016). In our model, terms modulating the VIP firing rate causing the neuronal population to have a switch-like behavior have been essentially encapsulated into the binary s_t variable in Equation (2.52). Despite the fact that incorporating cell type-specific contributions of top-down feedback in our model is an avenue of clear importance to relate to recent experimental findings, we leave this to future work. For simplicity, we have also limited the basis set of filters to one that extracts information about oriented edges in natural scenes. However, the computation of the extra-classical receptive fields need not be intrinsically limited to simple cells responding to Gabor-like filters, but can be extended to encompass neurons responding to more complex features in areas beyond V1. Switching circuits can occur more generally, including in somatosensory and auditory cortices, where some of the same neuronal populations interact using similar circuit logic (Niell and Stryker, 2010; Bigelow et. al., 2019). Populations of neurons in general switching circuits can respond to diverse stimuli (e.g. the VIP in auditory cortex are activated by punishment in Pi et. al., 2013).

The theoretical framework here did not make assumptions regarding the completeness of the basis. Instead, it focuses specifically on interactions outside the classical receptive field. Prior work of Olshausen and Field (1996, 1997, 2013) and that of Lewicki and Sejnowski (2000) have discussed extensively the benefits of overcomplete bases. The key feature in our model is the normalization of the activity of the neurons in patch, and not the orthogonality or completeness of the basis (indeed, 34 filters used here are not orthogonal). In our model, the interactions outside the classical receptive field of a cell are expressed exclusively on the representations by the cells with classical receptive fields in that location. As such, features not represented in an incomplete basis will be ignored in the context calculations. We use a relatively simple, linear model for the classical receptive field formation. If there are nonlinear interactions in the classical receptive field, the model can be expanded to represent covariance of neuronal activities rather than covariance of projections on a linear filter, however, the analysis of such an extension is beyond the scope of the current study.

Here, we showed how a biologically inspired switching mechanism can enable a network to efficiently process stimuli in two different conditions. Most artificial neural networks (ANNs) suffer from what has been termed "catastrophic forgetting", by which previously acquired memories are overwritten once new tasks are learned. Conversely, humans and other animals are capable of "transfer learning", the ability to use past information without overwriting previous knowledge. Proposed solutions to this problem, like elastic weight consolidation or intelligent synapses, are discussed in Kirkpatrick et. al. (2017), Zenke et. al. (2017), and Mallya et. al. (2018). When applied to a narrow condition of learning new contexts, our work adds a switching mechanism based on the connections among different cell types in V1. This may open new doors to artificial neural networks with analogous switching architectures.

2.5 Methods

2.5.1 A theory of optimal integration of static context in images

A theory of optimal context integration was first outlined in Iyer et. al. (2020) and describes a probabilistic framework for inferring features at particular locations of an image given the features at surrounding locations. The probabilities of these feature occurring and co-occurring are then mapped to elements of a biological circuit (firing rates, weights).

Neuronal code We assume the firing rate of neurons to be a function of the probability of a feature being present at a specific location of the image:

$$\mathbf{f}^m_{k,X} = g(p(\mathbf{F}^m_k|i_X)) \tag{2.26}$$

where $\mathbf{f}^m_{k,X}$ represents the firing rate due to the classical receptive field of a neuron coding for feature $\mathbf{F}_k$ at location m in response to image i_X, and g is a monotonic function. For every image and every location we impose a normalization over features:

$$\sum_k p(\mathbf{F}^m_k|i_X) = \sum_k g^{-1}(\mathbf{f}^m_{k,X}) = 1 \tag{2.27}$$

Thus, the sum over probabilities of features adds up to 1. Throughout the paper, we assume $g(y) = y$, although the model may be applied with other monotonic functions as well.

Probabilistic framework We subdivide the image X into N patches that correspond to the classical receptive fields of neurons. Thus, we have:

$$p(\mathbf{F}^m_k|i_X) = p(\mathbf{F}^m_k|i^1_X, i^2_X, ..., i^N_X) \tag{2.28}$$

We will assume from this point forward that the firing rates are in response to an image X (i_X), but omit the subscript X to simplify the notation.

We first look at the simple case where there are only 2 patches: the classical receptive

field (patch i^m) and the surround, which is part of the extra-classical receptive field (patch i^n). We will take into account other surrounding patches later, when we perform an order expansion from $p(\mathbf{F}_k^m|i^m, i^n)$ to $p(\mathbf{F}_k^m|i^1, i^2, ..., i^N)$. The aim in the simple case with two patches is to infer to what extent feature $\mathbf{F}_k$ at patch i^m, denoted by $\mathbf{F}_k^m$, is present given information from both the classical receptive field and the surrounding extra-classical receptive field. Using Bayes rule and simple probabilistic relations, we sum over all possible features $\mathbf{F}_j^m$ in patch i^m to get:

$$p(\mathbf{F}_k^m|i^m, i^n) = \sum_j p(\mathbf{F}_k^m|i^m, i^n, \mathbf{F}_j^n) p(\mathbf{F}_j^n|i^m, i^n) \tag{2.29}$$

We can simplify the above relation by assuming the surround contribution from i^n does not contain higher order surround information, instead it includes only data from the classical receptive field: $p(\mathbf{F}_k^m|i^m, i^n, \mathbf{F}_j^n) \approx p(\mathbf{F}_k^m|i^m, \mathbf{F}_j^n)$. Our previous probabilistic statement (2.29) thus becomes

$$p(\mathbf{F}_k^m|i^m, i^n) = \sum_j p(\mathbf{F}_k^m|i^m, \mathbf{F}_j^n) p(\mathbf{F}_j^n|i^m, i^n). \tag{2.30}$$

Using Bayes rule for the first term,

$$p(\mathbf{F}_k^m|i^m, \mathbf{F}_j^n) = \frac{p(\mathbf{F}_j^n|\mathbf{F}_k^m, i^m) p(\mathbf{F}_k^m|i^m)}{p(\mathbf{F}_j^n|i^m)}, \tag{2.31}$$

Equation (2.30) becomes

$$p(\mathbf{F}_k^m|i^m, i^n) = p(\mathbf{F}_k^m|i^m) \sum_j \frac{p(\mathbf{F}_j^n|i^m, \mathbf{F}_k^m)}{p(\mathbf{F}_j^n|i^m)} p(\mathbf{F}_j^n|i^m, i^n). \tag{2.32}$$

Assuming that we can ignore higher order contributions due to surround modulation, i.e. the surround modulation of the surround, we can make the following simplifications: $p(\mathbf{F}_j^n|i^m, \mathbf{F}_k^m) \approx p(\mathbf{F}_j^n|\mathbf{F}_k^m)$, $p(\mathbf{F}_j^n|i^m) \approx p(\mathbf{F}_j^n)$, and $p(\mathbf{F}_j^n|i^m, i^n) \approx p(\mathbf{F}_j^n|i^n)$. This way, patch i^n is in the surround of patch i^m and modulates the firing rate due to i^m, but we are not concerned about the further effect i^m has on i^n. Then equation (2.31) thus becomes

$$p(\mathbf{F}_k^m|i^m, \mathbf{F}_j^n) = \frac{p(\mathbf{F}_j^n \cap \mathbf{F}_k^m) p(\mathbf{F}_k^m|i^m)}{p(\mathbf{F}_j^n) p(\mathbf{F}_k^m)}. \tag{2.33}$$

The original equation (2.29) becomes:

$$p(\mathbf{F}_k^m|i^m,i^n) = p(\mathbf{F}_k^m|i^m)\sum_j(1+\frac{p(\mathbf{F}_j^n\cap\mathbf{F}_k^m)-p(\mathbf{F}_j^n)p(\mathbf{F}_k^m)}{p(\mathbf{F}_j^n)p(\mathbf{F}_k^m)})p(\mathbf{F}_j^n|i^n) \Leftrightarrow \tag{2.34}$$

$$p(\mathbf{F}_k^m|i^m,i^n) = p(\mathbf{F}_k^m|i^m)(1+\sum_j\frac{p(\mathbf{F}_j^n\cap\mathbf{F}_k^m)-p(\mathbf{F}_j^n)p(\mathbf{F}_k^m)}{p(\mathbf{F}_j^n)p(\mathbf{F}_k^m)}p(\mathbf{F}_j^n|i^n)) \tag{2.35}$$

The last equivalence holds because we have assumed in (2.27) that all probabilities sum to 1.

We can now go from two patches to N patches that cover the entire image: $i^1, i^2, ..., i^N$. We further assume that each patch provides independent information to a neuron coding for $\mathbf{F}_k^m$ so that we obtain:

$$\begin{aligned} p(\mathbf{F}_k^m|i) &= p(\mathbf{F}_k^m|i^1,i^2,...,i^N) \\ &= p(\mathbf{F}_k^m|i^m)\cdot\prod_{n\neq m}^{N}(1+\sum_j\frac{p(\mathbf{F}_j^n\cap\mathbf{F}_k^m)-p(\mathbf{F}_j^n)p(\mathbf{F}_k^m)}{p(\mathbf{F}_j^n)p(\mathbf{F}_k^m)}p(\mathbf{F}_j^n|i^n)) \end{aligned} \tag{2.36}$$

If the contribution from each patch is very small, we can ignore the higher order terms in (2.39) and apply the approximation $\prod_i(1+x_i)\approx 1+\sum_i x_i$ for $x_i \ll 1$:

$$\begin{aligned} p(\mathbf{F}_k^m|i) &= p(\mathbf{F}_k^m|i^1,i^2,...,i^N) \\ &= p(\mathbf{F}_k^m|i^m)\cdot(1+\sum_{n,n\neq m}\sum_j\frac{p(\mathbf{F}_j^n\cap\mathbf{F}_k^m)-p(\mathbf{F}_j^n)p(\mathbf{F}_k^m)}{p(\mathbf{F}_j^n)p(\mathbf{F}_k^m)}p(\mathbf{F}_j^n|i^n)) \end{aligned} \tag{2.37}$$

Mapping from the probabilistic framework to a neural network Using a simple neural code with $g(x) = x$, so that the firing rate represents the probability of feature presence, we obtain a simple mapping to a network of neurons. We denote

$$\mathbf{W}_{kj}^{mn} = \frac{p(\mathbf{F}_k^m\cap\mathbf{F}_j^n)-p(\mathbf{F}_k^m)p(\mathbf{F}_j^n)}{p(\mathbf{F}_k^m)p(\mathbf{F}_j^n)} = \frac{p(\mathbf{F}_k^m\cap\mathbf{F}_j^n)}{p(\mathbf{F}_k^m)p(\mathbf{F}_j^n)} - 1 \tag{2.38}$$

and map $\mathbf{W}_{kj}^{mn}$ to the synaptic weight between neurons responding preferentially to features $\mathbf{F}_k^m$ and $\mathbf{F}_j^n$, respectively. Then equation (2.37) becomes,

$$p(\mathbf{F}_k^m|i) = p(\mathbf{F}_k^m|i^m) \cdot (1 + \sum_{n,n\neq m} \sum_j \mathbf{W}_{kj}^{mn} p(\mathbf{F}_j^n|i^n)). \tag{2.39}$$

We can also map firing rates to probabilities: $\mathbf{r}_k^m = p(\mathbf{F}_k^m|i)$ and $\mathbf{f}_k^m = p(\mathbf{F}_k^m|i^m)$, where $\mathbf{r}_k^m$ is the firing of the neuron with receptive field at patch m and most responsive to feature $\mathbf{F}_k$, and $\mathbf{f}_k^m$ is the firing rate of the same neuron due to just the classical receptive field i^m. As we recognize below, inferring these firing rates from our image and video datasets requires rectification and normalization so that $\mathbf{f}$ and $\mathbf{r}$ can be interpreted as probabilities.

The formula for synaptic weight can be expressed based on average activities of cells, when X spans a comprehensive set of natural images:

$$\mathbf{W}_{kj}^{mn} = \frac{\langle \mathbf{r}_k^m \mathbf{r}_j^n \rangle_X}{\langle \mathbf{r}_k^m \rangle_X \langle \mathbf{r}_j^n \rangle_X} - 1 \tag{2.40}$$

These weights can be achieved using Hebbian learning in an unsupervised manner. To avoid writing implicit equations for the firing rates which are difficult to solve, and to make the computation tractable in practice without requiring learning, we use an approximation that requires only $\mathbf{f}$, the firing rates due to the classical receptive fields:

$$\mathbf{W}_{kj}^{mn} \approx \frac{\langle \mathbf{f}_k^m \mathbf{f}_j^n \rangle_X}{\langle \mathbf{f}_k^m \rangle_X \langle \mathbf{f}_j^n \rangle_X} - 1 \tag{2.41}$$

Finally, the probabilistic equations (2.37)-(2.39) outlined above can be re-written in terms of biologically-relevant quantities like firing rates and synaptic weights by applying the appropriate mappings:

$$\mathbf{r}_k^m = \frac{1}{\mathbf{L}^m} \mathbf{f}_k^m \prod_{n,n\neq 1} (1 + \sum_j \mathbf{W}_{kj}^{mn} \mathbf{f}_j^n), \tag{2.42}$$

or, more simply,

$$\mathbf{r}_k^m \approx \frac{1}{\mathbf{L}^m} \mathbf{f}_k^m (1 + \sum_{n,n\neq m} \sum_j \mathbf{W}_{kj}^{mn} \mathbf{f}_j^n). \tag{2.43}$$

when lateral connections given by $\mathbf{W}_{kj}^{mn}$ all sum up together to have a multiplicative effect. Here $\mathbf{L}^m$ is a normalization coefficient for patch i^m, since we require

$$\sum_k \mathbf{r}_k^m = 1 \tag{2.44}$$

and thus denote

$$\mathbf{L}^m = \sum_k \mathbf{f}_k^m \cdot \prod_{n \neq m}^{N} (1 + \sum_j \mathbf{W}_{kj}^{mn} \mathbf{f}_j^n) \tag{2.45}$$

As outlined in Iyer et. al. (2020), this can be implemented in a network in which a set of neurons responsible for normalization have a divisive effect on the neurons, are patch-specific (have a classical receptive field of similar size to the neurons), inhibit equally all the neurons in their image patch, are untuned to features in the visual space, and receive inputs equal to the average of the inputs of the neurons in the patch.

2.5.2 Computing the synaptic weights

To compute weights according to (2.41), we first compute $\mathbf{f}_k^n$, the firing rates due to the classical receptive field for every image X in a large dataset. Initially, we pre-process the image: we convert the image to grayscale, subtract the mean, and normalize the image to have a maximum value of 1. Similarly, we pre-process the filters so the mean of each is 0. $\mathbf{f}_k$ is the result of convolving X with feature k, rectifying and then normalizing so that at each location n the sum over features k of firing rates $\mathbf{f}_k^n$ is equal to 1. Rectification ensures that firing rates are non-negative, while normalization further ensures we can interpret $\mathbf{f}$ as probabilities. We average these firing rates over all images X in the dataset to obtain $< \mathbf{f}_k^n >_X$ for each feature k. The feature co-occurrence probability is given by $< \mathbf{f}_k^m, \mathbf{f}_j^n >_X$ in the numerator for the synaptic weight formula is then computed by further pairwise convolution of firing rates due to the classical receptive field for each possible pair of filters in the basis set and each image in the dataset, and then averaged over all images.

For a dataset of videos, formula (2.41) becomes

$$\mathbf{W}_{k_1 k_2}^{n_1 n_2, \Delta t} = \frac{\langle \mathbf{f}_k^{m,t}, \mathbf{f}_j^{n,t-\Delta t} \rangle_{\text{frames}}}{\langle \mathbf{f}_k^{m,t} \rangle_{\text{frames}} \langle \mathbf{f}_j^{n,t-\Delta t} \rangle_{\text{frames}}} - 1 \tag{2.46}$$

The feature co-occurrence probability given by $\langle \mathbf{f}_k^{m,t}, \mathbf{f}_j^{n,t-\Delta t} \rangle_{\text{frames}}$ is computed by convolution of firing rates due to the classical receptive field at different frames (t and $t - \Delta t$) for each video and averaged over all videos and video frames. The assumption here is that extra-classical effects are delayed by a time Δt that corresponds with the time between movie frames or, biologically, corresponds to the synaptic delay.

We first assume translational invariance so that only the relative position of two filters is relevant: $\mathbf{W}_{j_1,j_2}^{n_1,n_2} = \mathbf{W}_{j_1,j_2}^{n_3,n_4}$ when $\vec{n_1} - \vec{n_2} = \vec{n_3} - \vec{n_4}$. The assumption that weights act with translational invariance allows to rewrite the connectivities as simply a function of the distance, in image space, between the receptive field centers of the two neurons. Second, the mathematical validity of our probabilistic framework relies on the assumption that patches in the visual space, representing receptive fields of neurons, contain independent information. To reconcile this assumption with our empirically derived weights, we only consider connections between neurons whose receptive fields are sufficiently far apart, regardless of their corresponding feature identity. This leads to the usage of sparse weights for moving and static contexts (Figure 2.4e), where the only non-zero weights we allow in W are spatially half of receptive field apart. More precisely, for every feature k, synaptic weights from target filters were sampled in steps of $0.5\times$ the receptive field size at 3 distances in each direction around $(0, 0)$, so that we have synaptic weights on a (7×7) grid (3 connections to the left/up + 3 connections to the right/down + self-connection = 7). Instead of using these sparse weights after sampling, we could have also re-scaled the original, non-sparse weights by a scalar α so that $||\mathbf{W}^{\text{static/moving (sparse)}} - \alpha \mathbf{W}^{\text{static/moving}}|| \approx 0$. Searching over possible values of α, we find $\alpha \approx 1/50$. We choose however to work with sparse weights or test our results on the original, non-sparse weights without worrying about the re-scaling by α. Although results presented in this study are largely for sparse weights, we have checked that the main results also hold when using full connectivity, at least for small $\Delta t \in \{1, 2\}$ (Figure 2.15a). Further, assuming that the contribution due to context integration decays as the filters are spatially further and further apart, we can limit the weights in space to three times the size of the classical receptive field. Sample synaptic weights obtained using this procedure are shown in Figure 2.4e (and Figures Figures 2.4d and 2.4f without the

sampling of weights).

2.5.3 Constructing the feature space for natural images and videos

We chose a basis of spatial filters that was constructed as outlined in Iyer et. al. (2020). This is done by averaging approximations of spatial receptive field sizes from 212 recorded neurons in V1 (Durand et. al., 2016). This set of filters is our first feature space and consists of four classes of spatial RFs observed experimentally: ON (1 feature), OFF (1 feature), and two versions of ON/OFF neurons (8 features each, for a total of 16), with the first version having a stronger ON subfield, and the second a stronger OFF subfield. Each subfield was modeled as a 2D Gaussian with a standard deviation of $\sigma = 0.5\times$ average subfield size, which was measured to be 4.8 degrees for the OFF subfield, and 4.2 degrees for the ON subfield. The relative orientation between two subfields for each ON/OFF class was varied uniformly in steps of 45 degrees, from 0 to 315 degrees. Also for the ON/OFF class, the relative distance between the centers of the ON and OFF subfields was chosen to be 5 degrees, which equates to roughly 2σ. According to the data, the amplitude of the weaker subfield is chosen to be half that of the stronger subfield, whose highest amplitude was chosen to be unity. These two subfields are then combined additively to form a receptive field whose size is 7 degrees (the distance between the two subfields plus σ). The set of 18 features is shown in Figure 4.5d.

We then added 16 more filters with a temporal component, for a total of 34 filters. These filters have 2 frames with the first frame being one of the ON/OFF filters. The second frame is the ON/OFF filter in the previous frame shifted 3 pixels to the left, which matches the distance the sliding window moves every frame to generate the video. Such a spatio-temporal filter is shown in Figure 2.3e.

2.5.4 Datasets of natural and synthetic images and videos

Natural images and videos For the dataset of images, we used the Berkeley Segmentation Dataset (BSDS) training and test datasets (Martin et. al., 2001). The training dataset

consists of 200 images of animals, human faces, landscapes, buildings, etc., and is used to compute the weights $\mathbf{W}^{\text{static}}$. This same training set is then employed to construct the dataset of 200 videos where a sliding window moves across the image for each frame of the video. In the simple case, the sliding window (167×167) moves 3 pixels per frame in the horizontal direction across the image (321×481 or 481×321), from left to right for 50 frames (Figure 4.5b). The sliding window may also move in any random direction, resulting in different statistics of the video dataset and hence different $\mathbf{W}^{\text{moving}}$. This different dataset of videos is generated by choosing any pixel in the image and moving the sliding window toward it in smaller increments until that pixel is reached; a new pixel is then chosen from the image until there are a maximum limit of frames in the video (50 frames). Results from this different dataset are shown in Figure 2.10 and Figure 2.11. We further get 100 images from the BSDS test set to generate the corresponding 100 videos and use in the optimization problem. These video frames are provided as input to the optimizer that minimizes the loss functions $E_{switch,1}$ and $E_{switch,2}$ to find $\mathbf{f}^{\alpha}$, $\mathbf{f}^{\beta}$ for $E_{switch,1}$ and $\mathbf{W}^{VIP \to SST}$, $\mathbf{W}^{VIP \to PYR}$, and $\mathbf{W}^{PYR \to VIP}$ for $E_{switch,2}$. For both optimization problems we set 50 frames aside from these 100 videos to compute the generalization error during the minimization procedure.

In order to generate the figures in Figure 2.8, another set of 100 videos generated from BSDS testing dataset is altered by adding Gaussian and salt-and-pepper noise of different parameters to each frame. The resulting noisy video frames are used to establish the ability of the switching circuit to do visual processing of stimuli with better reconstruction capability than the circuit implementing the static extra-classical receptive field or without extra-classical receptive field (sec. 2.3.6). Gaussian white noise has standard deviation $\sigma = 0.5$ for reconstructions in Figure 2.8e, while salt-and-pepper noise turns pixels black or white with probability $p = 0.2$ each, for reconstructions in Figure 2.8d, figs. 2.8g to 2.8i. Parameters σ and p are varied ($\sigma \in [0.5, 3]$, $p \in [0.05, 0.3]$) in Figure 2.8f.

Synthetic datasets of images and videos of horizontal and vertical bars This simple synthetic dataset consists of 18 images of horizontal and vertical bars (9 horizontal, 9 vertical). Images are 9×9, each image having a bar at a different location. Videos consist

of bars moving in any direction 1 pixel at a time: left or right (for horizontal bars), and up or down (for vertical bars).

2.5.5 Deriving an equation for PYR firing rate consistent with V1 circuit architecture

Let $\mathbf{f}$ be the firing rate due to the classical receptive field, $\mathbf{r}$ the firing rate incorporating extra-classical receptive field information, and $W^{X\to Y}$ the weights between neuronal populations X, Y. We can write *approximated* expressions for firing rates of PYR, SST, VIP neurons at time t:

a) When there is no feedback connection from PYR to VIP

$$\mathbf{r}^t_{PYR} = \mathbf{f}^t_{PYR} \circ (1 + \mathbf{W}^{PYR\to PYR}\mathbf{r}^{t-1}_{PYR} + \mathbf{W}^{SST\to PYR}\mathbf{r}^{t-1}_{SST} + \mathbf{W}^{VIP\to PYR}\mathbf{r}^{t-1}_{VIP}) \tag{2.47}$$

$$\mathbf{r}^t_{SST} = \mathbf{f}^t_{SST} + \mathbf{W}^{VIP\to SST}\mathbf{r}^t_{VIP} \tag{2.48}$$

$$\mathbf{r}^t_{VIP} = s_t \cdot \mathbf{f}^t_{VIP}. \tag{2.49}$$

b) When there is feedback from PYR to VIP

$$\mathbf{r}^t_{PYR} = \mathbf{f}^t_{PYR} \cdot (1 + \mathbf{W}^{PYR\to PYR}\mathbf{r}^{t-1}_{PYR} + \mathbf{W}^{SST\to PYR}\mathbf{r}^{t-1}_{SST} + \mathbf{W}^{VIP\to PYR}\mathbf{r}^{t-1}_{VIP}) \tag{2.50}$$

$$\mathbf{r}^t_{SST} = \mathbf{f}^t_{SST} + \mathbf{W}^{VIP\to SST}\mathbf{r}^t_{VIP} \tag{2.51}$$

$$\mathbf{r}^t_{VIP} = s_t \cdot \mathbf{W}^{PYR\to VIP}\mathbf{r}^t_{PYR}, \tag{2.52}$$

where $\mathbf{f}^t_{VIP}$ of Equation (2.49) is the intrinsic firing rate of VIP and s_t is a binary variable that takes the value 1 during the moving condition and 0 during the static condition. For the analysis of the firing rate during movement we assume $s_t = 1$. Equations (2.47) and (2.50), expressing the firing rate $\mathbf{r}^t_{PYR}$ of the PYR population, assume the extra-classical receptive field contribution given by lateral connections has a multiplicative effect on the feedforward activities $\mathbf{f}_{PYR}$. This multiplicative gain is the result of mapping from the probabilistic framework of Equations (2.39) to (2.43) and their analogs for the moving circuit activities and weights. This results in the network doing optimal inference of visual features via PYR firing rates as expressed in (2.47) and (2.50), and as detailed in sec. 2.3.1. The VIP firing

rate $\mathbf{r}_{VIP}$ expression involves a binary gating term that switches based on state (static or moving), a simplification of what has been found empirically. The model could incorporate a term $\mathbf{f}_{VIP}$ into the expression (2.52) describing VIP firing rates driven independently from PYR such that $\mathbf{r}^t_{VIP} = s_t \cdot \mathbf{W}^{PYR\to VIP}\mathbf{r}^t_{PYR} + \mathbf{f}_{VIP}$, but this change would not alter our main results. Finally, only the inter-neuron connections with the longest synaptic delay are assumed to be non-instantaneous (connections to and from PYR), while other connections are presumed to occur at a much faster time-scale (connections between inhibitor neurons). Biologically, PYR are assumed to carry out computations by using dendritic trees, as outlined in Poirazi et. al. (2003), while SST and VIP are more spatially compact than PYR (Gouwens et. al., 2019). Hence, synaptic delays between PYR and other neuron populations are longer than between other populations.

Making the appropriate substitutions in (2.47) and in (2.50), we get the PYR firing rates:

for case **a)**,

$$\mathbf{r}^t_{PYR} = \mathbf{f}^t_{PYR} \circ [1 + \mathbf{W}^{PYR\to PYR}\mathbf{r}^{t-1}_{PYR} + \mathbf{W}^{SST\to PYR}(\mathbf{f}^{t-1}_{SST} + \mathbf{W}^{VIP\to SST}\mathbf{f}^{t-1}_{VIP}) + \\ + \mathbf{W}^{VIP\to PYR}\mathbf{f}^{t-1}_{VIP}] \tag{2.53}$$

for case **b)**,

$$\mathbf{r}^t_{PYR} = \mathbf{f}^t_{PYR} \circ [1 + \mathbf{W}^{PYR\to PYR}\mathbf{r}^{t-1}_{PYR} + \mathbf{W}^{SST\to PYR}(\mathbf{f}^{t-1}_{SST} + \mathbf{W}^{VIP\to SST}\mathbf{W}^{PYR\to VIP}\mathbf{r}^{t-1}_{PYR}) + \\ + \mathbf{W}^{VIP\to PYR}\mathbf{W}^{PYR\to VIP}\mathbf{r}^{t-1}_{PYR}] \tag{2.54}$$

We can ignore further recurrence due to additional extra-classical receptive field contributions by making the approximation $\mathbf{r}^{t-1}_{PYR} = \mathbf{f}^{t-1}_{PYR}$. We are thus ignoring contextual surround modulation that is itself subject to surround influence — a "higher order" surround modulation — and instead consider only the classical receptive field response from surround neurons. These terms are small since this additional contribution is a linear combination of $\mathbf{f}_i\mathbf{f}_j$, $\mathbf{f}_i\mathbf{f}_j\mathbf{f}_k$, etc, where $\mathbf{f}_i$ are classical receptive field firing rates of neuron i and $0 \leq \mathbf{f}_i \leq 1$.

Additionally, we assume PYR and SST receive the same input so that $\mathbf{f}^t_{PYR} = \mathbf{f}^t_{SST}$. With these simplifications and dropping the subscript PYR for clarity, the equations for $\mathbf{r}^t_{PYR}$ become:

for case **a)**,

$$\begin{aligned}\mathbf{r}^t = \mathbf{f}^t \circ (1 + \mathbf{W}^{PYR\to PYR}\mathbf{f}^{t-1} + \mathbf{W}^{SST\to PYR}\mathbf{f}^{t-1} \\ + \mathbf{W}^{SST\to PYR}\mathbf{W}^{VIP\to SST}\mathbf{f}_{VIP} + \mathbf{W}^{VIP\to PYR}\mathbf{f}_{VIP})\end{aligned} \tag{2.55}$$

which leads to

$$\begin{aligned}\mathbf{r}^t = \mathbf{f}^t \circ (1 + \mathbf{W}^{PYR\to PYR}\mathbf{f}^{t-1} + \mathbf{W}^{SST\to PYR}\mathbf{f}^{t-1} \\ + \mathbf{W}^{SST\to PYR}\mathbf{f}^{\alpha} + \mathbf{f}^{\beta})\end{aligned} \tag{2.56}$$

where

$$\mathbf{f}^{\alpha} \equiv \mathbf{W}^{VIP\to SST}\mathbf{f}_{VIP} \tag{2.57}$$

and

$$\mathbf{f}^{\beta} \equiv \mathbf{W}^{VIP\to PYR}\mathbf{f}_{VIP}, \tag{2.58}$$

while for case **b)**,

$$\begin{aligned}\mathbf{r}^t = \mathbf{f}^t \circ (1 + \mathbf{W}^{PYR\to PYR}\mathbf{f}^{t-1} + \mathbf{W}^{SST\to PYR}\mathbf{f}^{t-1} \\ + \mathbf{W}^{SST\to PYR}\mathbf{W}^{VIP\to SST}\mathbf{W}^{PYR\to VIP}\mathbf{f}^{t-1} + \mathbf{W}^{VIP\to PYR}\mathbf{W}^{PYR\to VIP}\mathbf{f}^{t-1}) \ .\end{aligned} \tag{2.59}$$

During the static condition, there is no contribution from the VIP and $\mathbf{f}^t = \mathbf{f}^{t-1}$ so the firing rate becomes

$$\mathbf{r}^{\text{static}} = \mathbf{f} \circ (1 + \mathbf{W}^{PYR\to PYR}\mathbf{f} + \mathbf{W}^{SST\to PYR}\mathbf{f}) \ . \tag{2.60}$$

However, we know from our theoretical framework that the firing rate during the static context can be written as:

$$\mathbf{r}^{\text{static}} = \mathbf{f} \circ (1 + \mathbf{W}^{\text{static}}\mathbf{f}) \tag{2.61}$$

where $\mathbf{W}^{\text{static}}$ has been computed from the dataset(s) of images and is a function of the average feature co-occurrence probability for pairs of spatial features. Therefore, we can consider a simple mapping that assigns $\mathbf{W}^{PYR\to PYR}$ and $\mathbf{W}^{SST\to PYR}$ to known weights: $\mathbf{W}^{PYR\to PYR} = \mathbf{W}_{+}^{\text{static}}$ and $\mathbf{W}^{SST\to PYR} = \mathbf{W}_{-}^{\text{static}}$, where $\mathbf{W}_{+}^{\text{static}}$ is the positive and $\mathbf{W}_{-}^{\text{static}}$ is the negative component of $\mathbf{W}^{\text{static}}$. The unknowns of equation (2.62) corresponding to the V1 circuit model with PYR to VIP connections, are thus only three sets of weights to and from VIP: $\mathbf{W}^{VIP\to SST}$, $\mathbf{W}^{VIP\to PYR}$, $\mathbf{W}^{PYR\to VIP}$.

Finally, the equation for the firing rate of PYR neurons during the moving condition that we focus on throughout the paper (with PYR projecting to VIP) becomes:

$$\begin{aligned}\mathbf{r}^t &= \mathbf{f}^t \circ (1 + \mathbf{W}_{+}^{\text{static}}\mathbf{f}^{t-1} + \mathbf{W}_{-}^{\text{static}}\mathbf{f}^{t-1} \\ &+ \mathbf{W}_{-}^{\text{static}}\mathbf{W}^{VIP\to SST}\mathbf{W}^{PYR\to VIP}\mathbf{f}^{t-1} + \mathbf{W}^{VIP\to PYR}\mathbf{W}^{PYR\to VIP}\mathbf{f}^{t-1}) \\ &= \mathbf{f}^t \circ (1 + \mathbf{W}^{\text{static}}\mathbf{f}^{t-1} + \\ &+ \mathbf{W}^{SST\to PYR}\mathbf{W}^{VIP\to SST}\mathbf{W}^{PYR\to VIP}\mathbf{f}^{t-1} + \mathbf{W}^{VIP\to PYR}\mathbf{W}^{PYR\to VIP}\mathbf{f}^{t-1}).\end{aligned} \tag{2.62}$$

2.5.6 Reconstructions from noisy videos using firing rates and optimal synaptic weights of different circuit architectures

To gain insight into how optimal synaptic weights can facilitate decoding of information present in the neuronal activity, we reconstructed natural image frames from videos using 4 distinct circuits. The firing rates in these circuits are described by the following equations:

$$\mathbf{r}^{\text{no EXC}}(t) = \mathbf{f}^t \tag{2.63}$$

$$\mathbf{r}^{\text{static}}(t) = \mathbf{f}^t \circ (1 + \mathbf{W}^{\text{static}}\mathbf{f}^{t-\Delta t}) \tag{2.64}$$

$$\mathbf{r}^{\text{moving}}(t) = \mathbf{f}^t \circ (1 + \mathbf{W}^{\text{moving}}\mathbf{f}^{t-\Delta t}) \tag{2.65}$$

$$\mathbf{r}^{\text{approx}}(t) = \mathbf{f}^t \circ (1 + \mathbf{W}^{\text{static}}\mathbf{f}^{t-\Delta t} + \mathbf{W}^{SST\to PYR}\mathbf{W}^{VIP\to SST}\mathbf{W}^{PYR\to VIP}\mathbf{f}^{t-\Delta t} + \tag{2.66}$$

$$\mathbf{W}^{VIP\to PYR}\mathbf{W}^{PYR\to VIP}\mathbf{f}^{t-\Delta t}) \tag{2.67}$$

The first equation above describing $\mathbf{r}^{\text{no EXC}}$ relies solely on the feedforward information

where no extra-classical receptive field contribution is included. The next two expressions re-state how the firing rates for the static and moving circuits require contributions from the extra-classical receptive fields through lateral connections $\mathbf{W}^{\text{static}}, \mathbf{W}^{\text{moving}}$, reflective of the statistical regularities of images/videos. Equation (2.67) describes the switching circuit we have implemented and characterized above and should approximate the firing rate in the moving circuit when VIP are active: $\mathbf{r}^{\text{moving}} \approx \mathbf{r}^{\text{approx}}$.

The reconstruction was performed as follows. For any noisy input image $X + \xi$, where ξ is some random variable representing a noisy process, we calculated the effective firing rate (activity) $\mathbf{r}$ of neuron/feature k at location n using the eqs. (2.63) to (2.67) above. To reconstruct image frames from firing rates, we convolved the firing rates computed with the inverses of the filters in our basis set. More specifically, the activity $\mathbf{r}_k$ corresponding to filter k was convolved with the inverse of k, which was obtained by flipping k about the horizontal and vertical axes. These convolutions for all filters were then averaged to obtain the final reconstruction.

We then performed the reconstruction for the same image frame X without any noise added. We assessed the de-noising capability of our circuits by computing the Pearson correlation coefficient ρ between the reconstruction of $X + \xi$ and the reconstruction of X. The latter is a baseline for our comparisons, as there is no noise to remove from the image frame through extra-classical surround modulation. The Pearson correlation coefficient ρ is a function of the activity $\mathbf{r}$ of different circuit architectures and is discussed and compared across circuits in sec. 2.3.6.

There are two further issues that merit further discussion. First, if the spectral content of the noise and image frame is known, a Wiener de-convolution can be applied which minimizes the mean square error between the estimated reconstruction and the original frame. Such a Wiener de-convolution would minimize the impact of de-convolved noise at frequencies with poor signal-to-noise ratio. However, we assume here that interpretation of signals is done without access to knowledge of this spectral content, but rather implementing a

naive reconstruction as would be optimal in the noise-free limit. Second, given the presence of extra-classical surround contribution, the de-convolution operation may be more complex than the simple, filter by filter, convolution with the inverse filter $\mathbf{F}^T$. Specifically, the inverse may contain information about the cross-correlation of features. Again we work in the simplifying limit in which this is not the case. We do not exclude however the possibility that the biological circuit may apply a more complex reconstruction (e.g. via learning weights), an interesting avenue to explore in future work.

2.5.7 Like-to-like connectivity for PYR and VIP populations

In addition to inter-neuron connectivity discussed in sec. 2.3.7, PYR connection probability as a function of the difference in orientation tuning (figs. 2.11c to 2.11d) qualitatively matches the same graph reported experimentally (Ko et. al., 2011). This like-to-like connectivity, with neurons responding to similar features (orientations) more strongly connected, holds true for both static (shown in Iyer et. al., 2020) and moving weights (shown in figs. 2.11c to 2.11d and Figure 2.12). Another feature concerns the amplitude of static and moving weights which decreases with distance from the classical receptive field, with lower weights on average between neurons whose classical receptive fields are far away. Figure 2.11 shows the dependence of the maximum, minimum, and average positive and negative synaptic weights, on the distance between neuronal receptive fields. Assuming an exponential spatial decay of weights with distance and using the first two points in the plot displaying decreasing distance dependence in the mean positive static weights curve (Figure 2.11a), we computed the spatial constants $D_{\text{static/moving}} = 0.8\times$ the classical receptive field size. This is in accordance with past findings (Angelucci et. al., 2006; Iyer et. al., 2020), suggesting that the near surround extends over a range which is similar in size to the classical receptive field.

We further study the inferred connections to and from the VIP to establish whether these weights reflect the contextual statistics of static and moving states. We first inferred whether there is like-to-like connectivity between VIP and PYR populations by building a similarity matrix of dimension number of VIP neurons $\times$ number of PYR neurons that measures re-

sponse similarity between VIP and PYR neuronal populations. Each entry of this response similarity matrix is computed by taking the Pearson correlation between the GLM coefficients found above (sec. 2.3.7) for each VIP neuron and each PYR neuron, respectively. We next built, from our $Nf_2 \times 34 \times 3 \times 3$ tensor $\mathbf{W}^{VIP \to PYR}$ used for convolution, a matrix of connectivities of dimension number of VIP neurons $\times$ number of PYR neurons. Finally, taking the Pearson correlation coefficient between the response similarity matrix and the matrix of connectivities yields a statistically significant but very low correlation coefficient (-0.01, pval$<$ 0.01, Kolmogorov-Smirnov test). We conclude that, while like-to-like connectivity is present between PYR neurons, this phenomenon is not prevalent between VIP and PYR populations.

2.5.8 Measuring dimensionality with the participation ratio

We aim to characterize the dimensionality of the distribution of population vector responses representing neural activity. Across many trials, these population vectors populate a cloud of points. The dimensionality is a weighted measure of the number of axes explored by that cloud:

$$\mathrm{Dim}(C) = \frac{(\mathrm{Tr}C)^2}{\mathrm{Tr}C^2} = \frac{(\sum_i \lambda_i)^2}{\sum_i \lambda_i^2} \tag{2.68}$$

where C is the covariance matrix of the matrix of neural activations, and λ_i is the i^{th} eigenvalue of the covariance matrix C. $\mathrm{Dim}(C)$ measures the dimensionality of neural activity of our network and is termed the *participation ratio*. The eigenvectors of the covariance matrix C are the axes of our cloud of points representing activity in neural space. If the neural activities are independent and all have equal variance, all the eigenvalues of the covariance matrix have the same value and $\mathrm{Dim}(C) = N$. Alternatively, if the components are correlated so that the variance is evenly spread across M dimensions, only M eigenvalues would be nonzero and $\mathrm{Dim}(C) = M$. For other correlation structures, this measure interpolates between these two regimes and, as a rule of thumb, the dimensionality can be thought as corresponding to the number of dimensions required to explain about 80% of the total population variance in many settings (Mazzucato et. al., 2016; Gao et. al.,

2020; Litwin-Kumar et. al., 2017).

2.5.9 Inferring the tuning properties of VIP and PYR neurons

We further study the activation patterns of units in our switching circuit model by inferring the tuning properties of VIP and PYR units. To achieve this, we first choose a wavelet family that will determine our features, and which differs from the basis approximating spatial receptive fields in V1 from Methods sec. 2.5.3. We chose the Daubechies 4 wavelet family with a mother wavelet of length 15 pixels, as shown in 2.22a. We then consider the 2D discrete wavelet transforms of our video frames to obtain the approximation, horizontal detail, vertical detail and diagonal detail coefficients (wavelet transforms) respectively for each frame. The goal is to use the averages of these coefficients as the independent variables of a linear regression or GLM that models VIP or PYR activations.

To achieve this, we first reduce the dimensionality of the wavelet transforms obtained above by considering $100 = 10 \times 10$ patches of size 5×5 that tile wavelet transforms of each video frame. Averaging over the spatial component of these patches, we obtain three sets of 10×10 coefficients (for the horizontal, vertical, and diagonal detail respectively) that will be the independent variables of the linear regression or GLM. For each PYR/VIP neuron, we can regress its activity for every video frame against the 300 ($= 3 \times 10 \times 10$) coefficients we have inferred:

$$a = C \cdot x \tag{2.69}$$

where $a \in \mathbb{R}^{\text{no. frames}=4700}$ is the activity of a neuron (for every 4700 frames), $C \in \mathbb{R}^{\text{no. frames} \times \text{no. regressors}(300)}$ is the matrix of regressors, and $x \in \mathbb{R}^{\text{no. coefficients}=\text{no. regressors}}$ contains the unknowns that will determine the tuning of each neuron.

We obtain that most PYR neurons are tuned to horizontal features, and much less so to vertical features (data not shown). Using either a linear regression or a GLM with a Poisson distribution yields qualitatively similar results. Because VIP neurons in our model

only get input from the PYR, while the top-down input activating VIP is described simply by the binary term s_t, we obtain that VIP acquires the same preferential selectivity to horizontal features to the detriment of vertical features (Figure 2.22d). VIP neurons are tuned to horizontal features with an average regression coefficient of 0.65, while they are tuned to vertical features with an average regression coefficient of 0.015 (using the results from the linear regression). This runs counter to our expectation that VIP is capable of detecting horizontal movement in our dataset by exhibiting preferential selectivity towards vertical features within their receptive fields, analogously to empirical results in Millman et. al. (2020). Clearly, the simplification we have made by employing a binary term s_t in Eq. (2.52) prevents us from observing a more realistic VIP activation pattern that would deviate from the PYR pattern and provide further insights. We leave the more detailed modeling expanding our current simplified model in this direction to future work.

2.5.10 Code

Source code is available in ModelDB (McDougal et al. 2017) at http://modeldb.yale.edu/267120

Supplemental figures

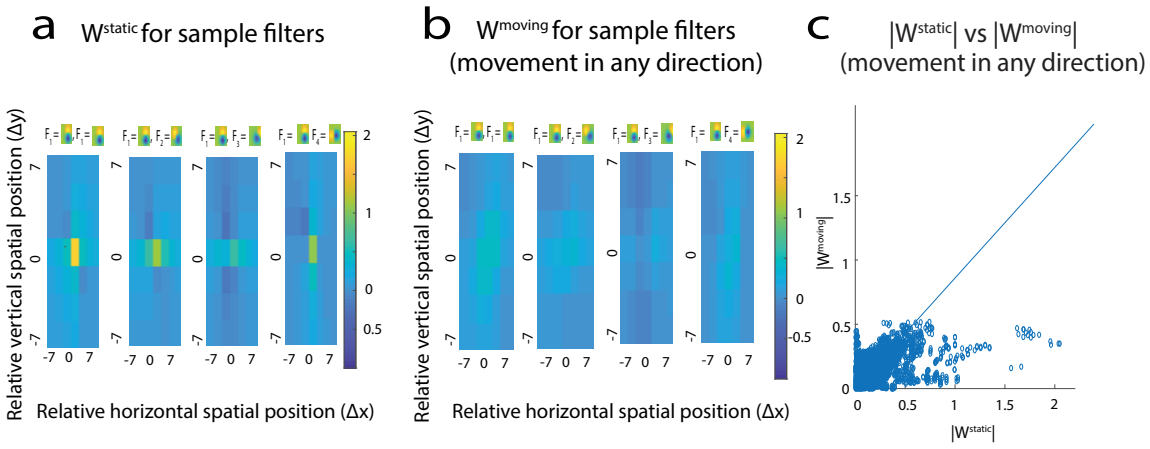


Figure 2.10: (a). Slices of $\mathbf{W}^{\text{static}}$ corresponding to different pairs of filters (feature $\mathbf{F}_1$ paired with features $\mathbf{F}_1$ - $\mathbf{F}_4$). (b). Slices of $\mathbf{W}^{\text{moving}}$ computed for the dataset of videos where movement is in any direction. Slices shown correspond to different pairs of filters (feature $\mathbf{F}_1$ paired with features $\mathbf{F}_1$ - $\mathbf{F}_4$). (c). Scatter plot of $|\mathbf{W}^{\text{static}}|$ vs $|\mathbf{W}^{\text{moving}}|$. This reveals that on average, $\|\mathbf{W}^{\text{static}}\| > \|\mathbf{W}^{\text{moving}}\|$ for the dataset of natural images and videos where movement can be in any direction.

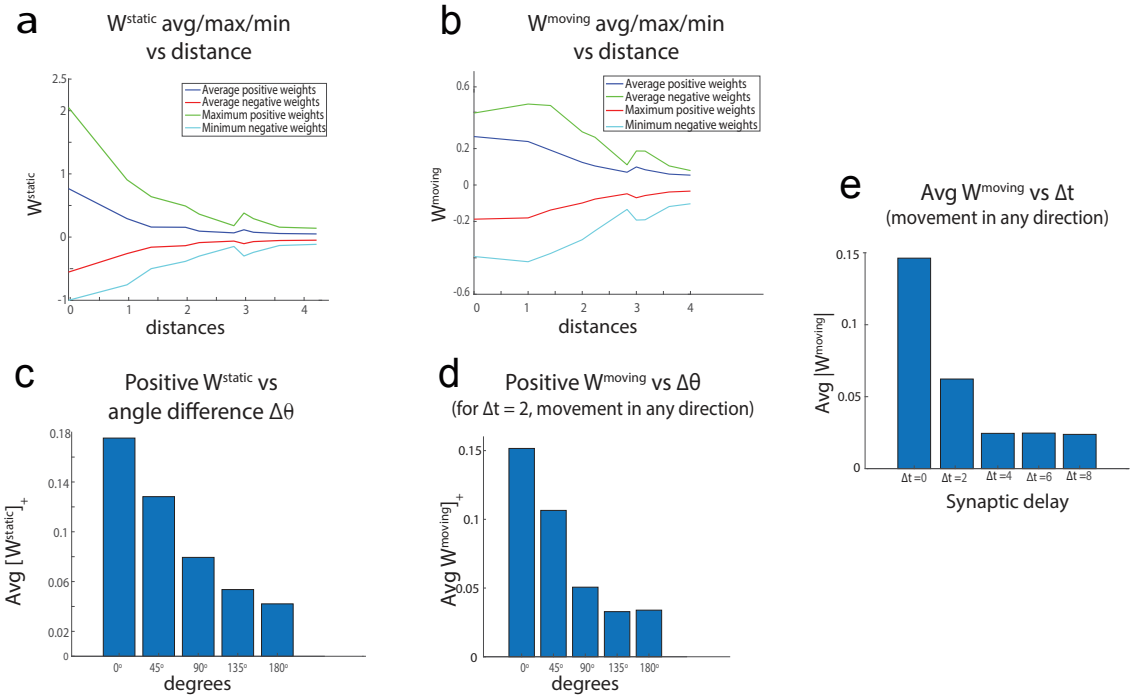


Figure 2.11: (a). Dependence of the maximum, minimum, average positive and negative synaptic weights for the *static* context ($\mathbf{W}^{\text{static}}$) onto a target neuron k from all neurons on the distance measured in terms of receptive field size (1 unit = 1/2 RF size = 7 pixels). This distance dependence enables us to compute the spatial constant in terms of the classical receptive field size and compare it to data. (b). Dependence of the maximum, minimum, average positive and negative synaptic weights for the *moving* context ($\mathbf{W}^{\text{moving}}$ with $\Delta t = 2$) onto a target neuron k from all neurons on the distance measured in terms of receptive field size (1 unit = 1/2 RF size = 7 pixels). The dataset of videos used to compute the weights here and in (d), (e) is the one where the movement can be in any direction. (c). Predicted average positive $\mathbf{W}^{\text{static}}$ as a function of the difference in orientation of features. This predicts that excitatory weights between neurons responsive to more similar features (similar in orientation) are stronger than those between neurons responsive to different features. The trend matches data in [17]. (d). Predicted average positive $\mathbf{W}^{\text{moving}}$ ($\Delta t = 2$) as a function of difference in orientation of features. (e). Average strength of $\mathbf{W}^{\text{moving}}$ as a function of Δt, a parameter describing synaptic delay. The higher the synaptic delay, the closer to chance the co-occurrence probability is, and thus the lower the absolute values of the synaptic weights are.

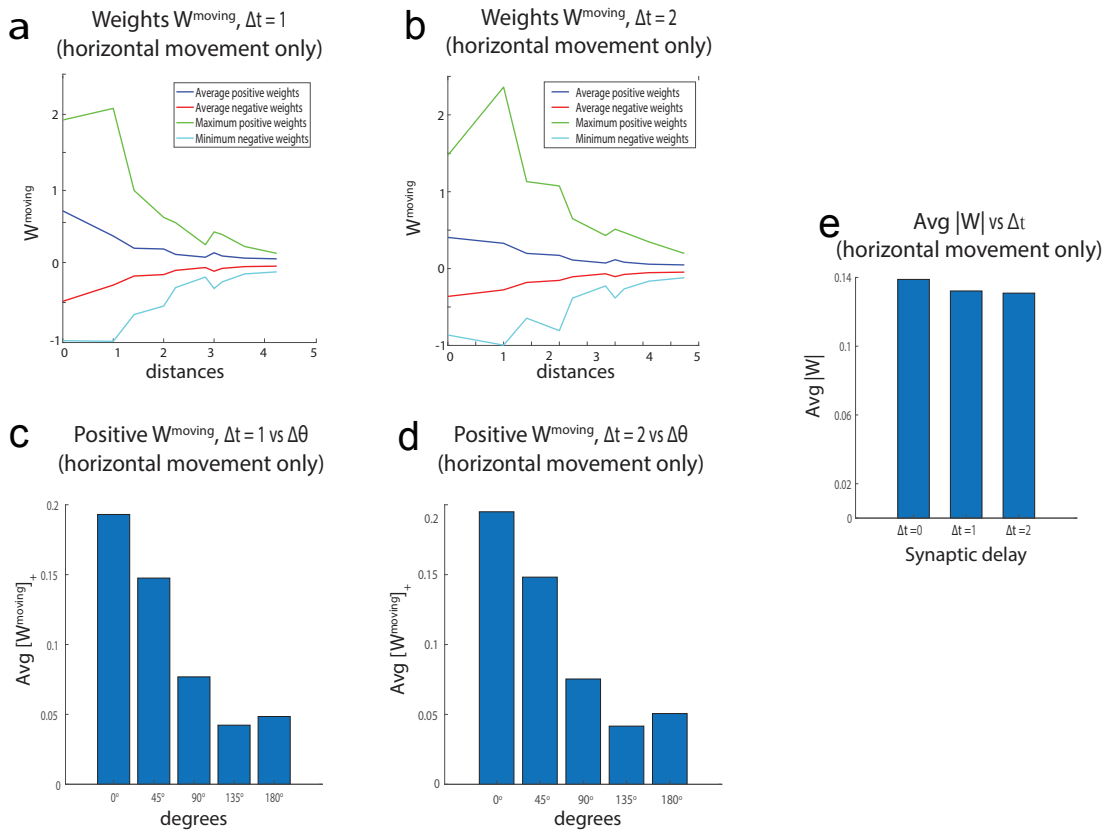


Figure 2.12: (a). Dependence of the maximum, minimum, average positive and negative synaptic weights for the *moving* context ($\mathbf{W}^{\text{moving}}$) with $\Delta t = 1$ onto a target neuron k from all neurons on the distance measured in terms of receptive field size (1 unit = 1/2 RF size = 7 pixels). (b). Dependence of the maximum, minimum, average positive and negative $\mathbf{W}^{\text{moving}}$ with $\Delta t = 2$ onto a target neuron k from all neurons on the distance measured in terms of receptive field size (1 unit = 1/2 RF size = 7 pixels). (c). Predicted average positive $\mathbf{W}^{\text{moving}}$ ($\Delta t = 1$) as a function of difference in orientation of features. (d). Same as (c), but with $\Delta t = 2$. (e). Average weight strength in terms of synaptic delay Δt, where $\Delta t = 0$ corresponds to $\mathbf{W}^{\text{static}}$. Unlike the weights in Figure 2.11, which correspond to movement in any direction, the average weight strength does not decrease significantly with Δt. Indeed, the peak of the tensor simply shifts at different spatial positions depending on how large the synaptic delay is, but otherwise, the tensor remains (mostly) unchanged.

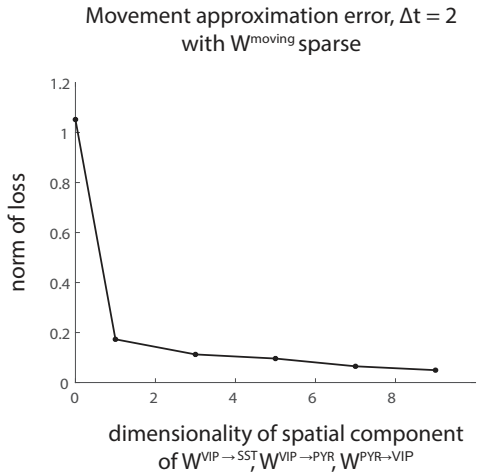


Figure 2.13: Varying the dimensionality of the tensors $\mathbf{W}^{\mathrm{VIP}\to\mathrm{SST}}$, $\mathbf{W}^{\mathrm{VIP}\to\mathrm{PYR}}$, $\mathbf{W}^{\mathrm{PYR}\to\mathrm{VIP}}$ can lower the movement approximation error as defined in (19). These tensors have dimension $Nf_1 \times Nf_2 \times c \times c$, where Nf_1, Nf_2 represent the number of VIP, SST, or PYR neurons, and c represents the dimensionality corresponding to the spatial component (shown on x-axis). We set $\Delta t = 2$, $Nf_1 = 5, Nf_2 = 34$ for $\mathbf{W}^{\mathrm{VIP}\to\mathrm{SST}}, \mathbf{W}^{\mathrm{VIP}\to\mathrm{PYR}}$, $Nf_1 = 34, Nf_2 = 5$ for $\mathbf{W}^{\mathrm{PYR}\to\mathrm{VIP}}$, and use sparse weights for the optimization procedure.

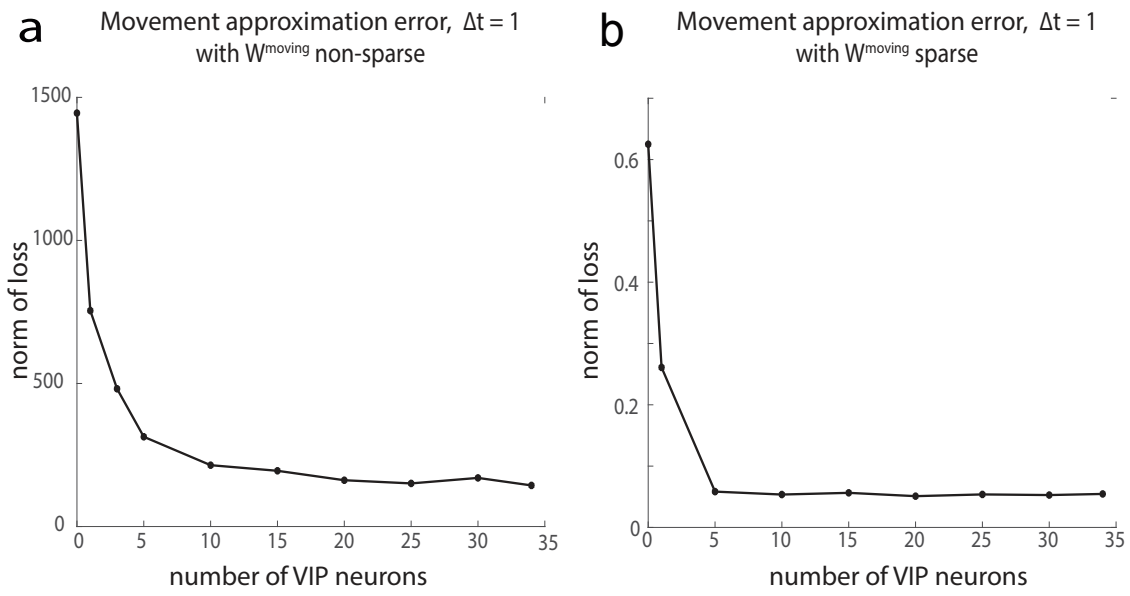


Figure 2.14: (a). Movement approximation error (defined as in (19)) decreases with increasing number of VIP neurons for synaptic delay $\Delta t = 1$ and using the full $\mathbf{W}^{\text{moving}}$ (non-sparse). (b). Movement approximation error decreases with increasing number of VIP neurons for synaptic delay $\Delta t = 1$ and using the sparse sampled $\mathbf{W}^{\text{moving}}$.

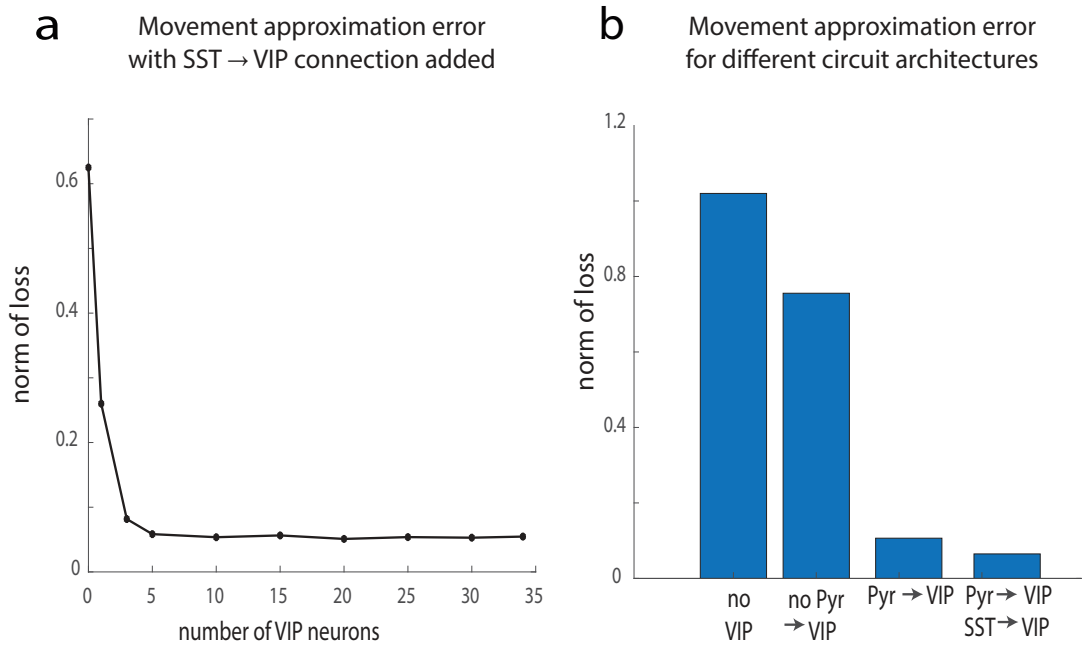


Figure 2.15: (a). Movement approximation error (defined as in (19)) decreases with increasing number of VIP neurons, after an additional connection from SST to VIP is added. We set synaptic delay to $\Delta t = 1$ and use the sparse sampled $\mathbf{W}^{\text{moving}}$. (b). Movement approximation error for different circuits: a circuit with no VIP units (leftmost bar), a circuit with VIP and connections from VIP to PYR and SST (middle left bar), a circuit with an additional connection from PYR to VIP added (middle right bar), a circuit with an additional connection from SST to VIP added (rightmost bar).

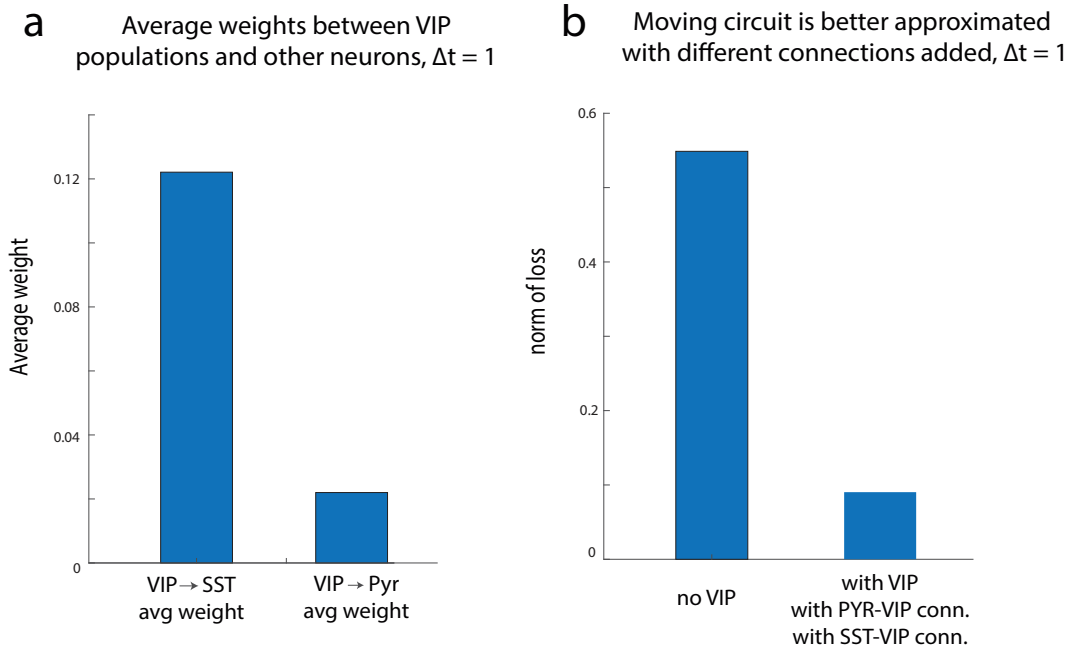


Figure 2.16: (a). Comparison of $\mathbf{W}^{VIP\to SST}$ average weights to $\mathbf{W}^{VIP\to PYR}$ average weights (0.12 compared to 0.022). The ratio between these average weights is invariant to re-scaling due to patch independence that results in sparse weights $\mathbf{W}^{VIP\to SST}, \mathbf{W}^{VIP\to PYR}$. These weights have been computed by optimizing (19) for $\mathbf{W}^{\text{moving}}$ with $\Delta t = 1$ (although a similar result holds for $\Delta t = 2$) (b). Verifying that using the solutions $\mathbf{W}^{VIP\to SST}, \mathbf{W}^{VIP\to PYR}$ to the optimization problem (19) yields a small movement approximation error (right bar) compared to the same error $E_{switch,2}$ when no VIP units are considered (left bar). The movement approximation error when VIP units are added (right bar) is for the circuit that includes SST to VIP and PYR to VIP connections.

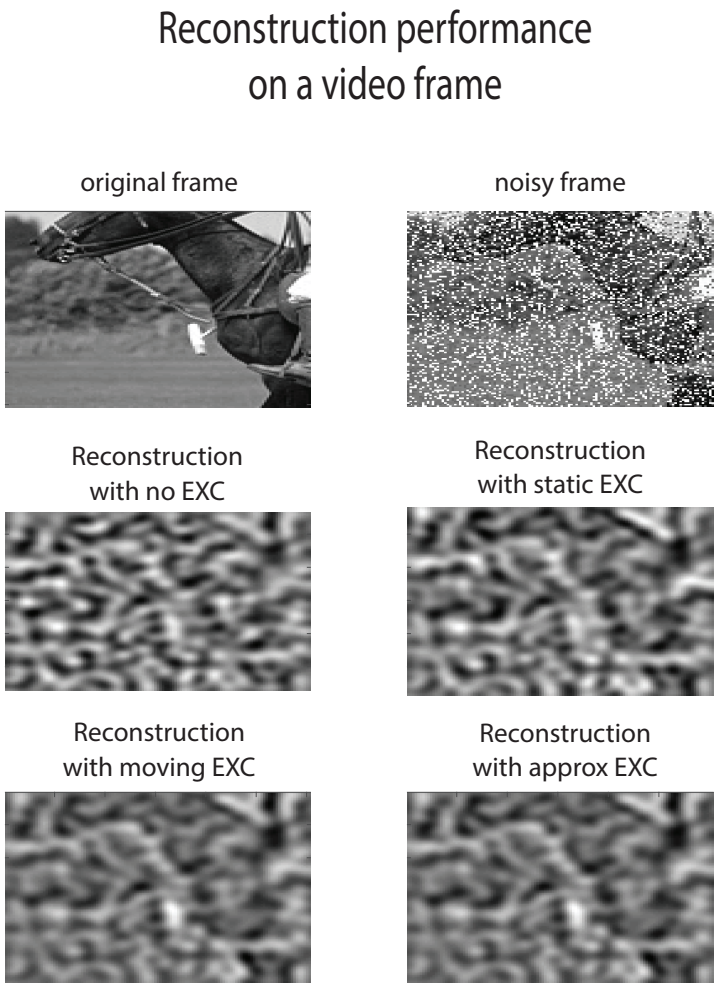


Figure 2.17: Example of a reconstructed video frame for each condition/circuit architecture: no EXC, static EXC, moving EXC, approximated EXC. Enlarged Fig. 7A.

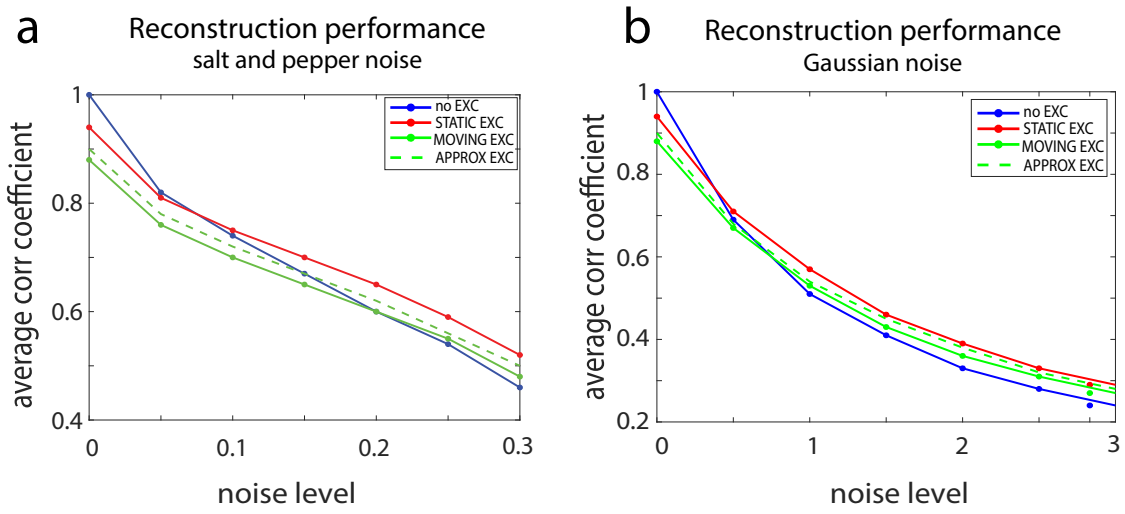


Figure 2.18: Average correlation coefficient over *images* as noise level is varied. (a). x-axis: noise level as salt and pepper noise is varied; (b). x-axis: noise level as Gaussian white noise std is varied.

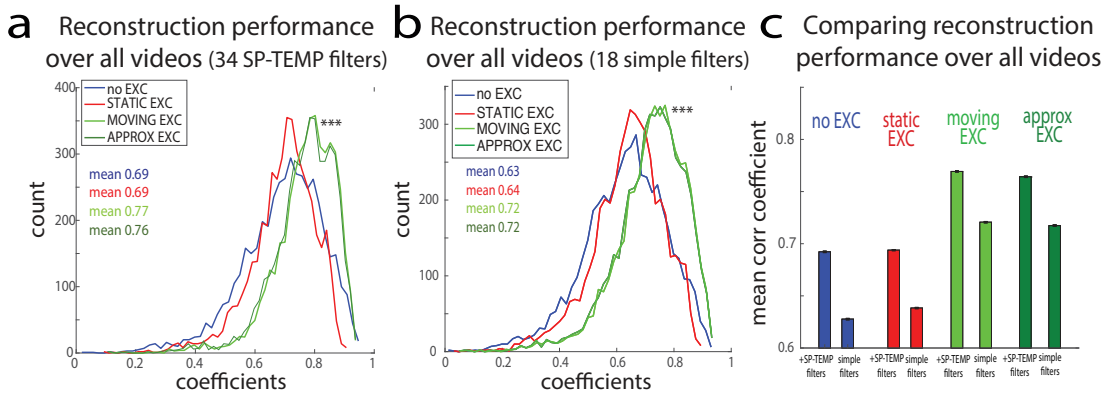


Figure 2.19: (a). Correlation coefficients over frames and videos for different conditions/circuit architectures when all 34 spatio-temporal filters are used. (b). Correlation coefficients over frames and videos for different conditions/circuit architectures when only 18 filters are used. The filters used are ones without the temporal component. (c). Comparison of average correlation coefficients across conditions/circuit architectures for the 34 spatio-temporal filters and the 18 "simple" spatial filters.

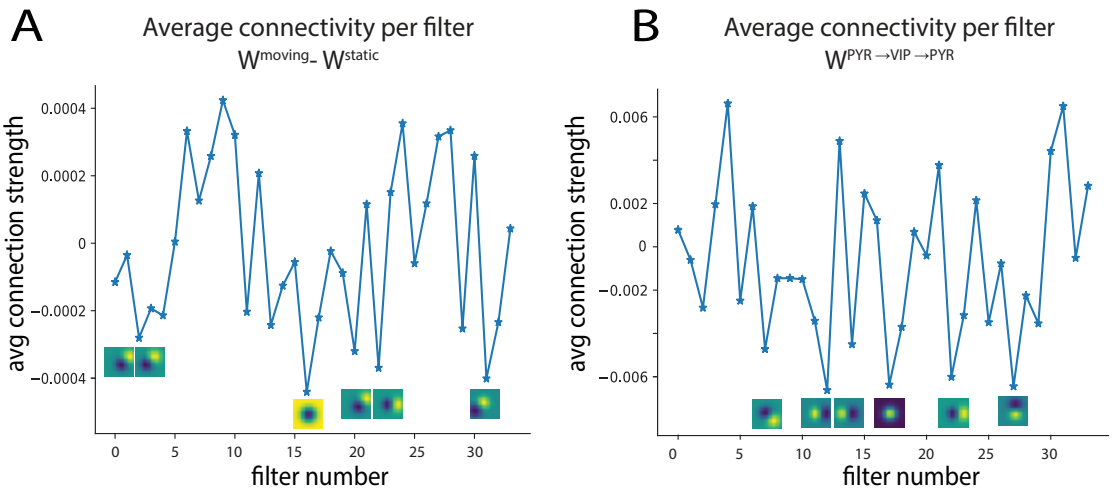


Figure 2.20: (a). Connectivity across post-synaptic filters, averaged over the spatial and pre-synaptic filter dimensions for $\mathbf{W}^{\text{moving}} - \mathbf{W}^{\text{static}}$. (b). Connectivity across post-synaptic filters, averaged over the spatial and pre-synaptic filter dimensions for $\mathbf{W}^{PYR \to VIP \to PYR} \equiv \mathbf{W}^{VIP \to PYR} * \mathbf{W}^{PYR \to VIP}$. Shown in both subfigures are the filters corresponding to the strongest inhibitory connections.

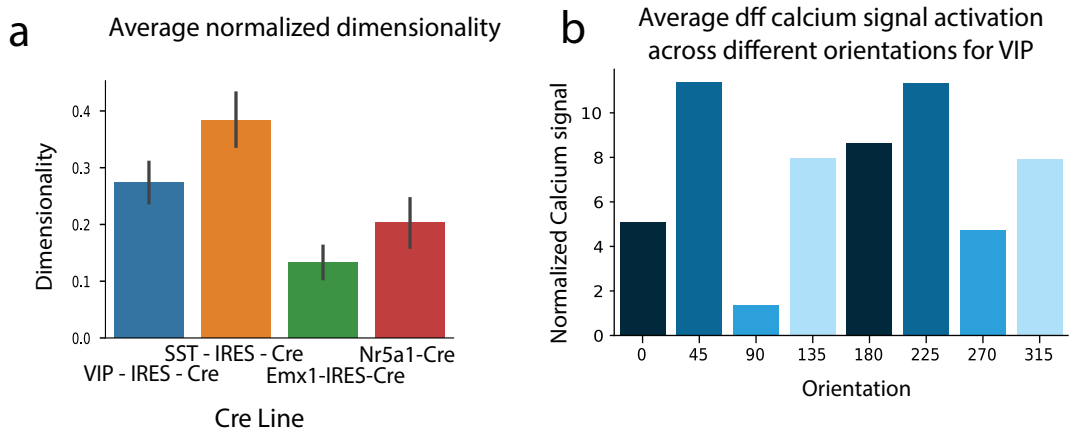


Figure 2.21: (a). Average dimensionality across sessions normalized to the number of neurons in each session for multiple neural populations. Dimensionality is assessed by means of the Participation Ratio measure (Methods, sec. 4.8) during epochs of spontaneous activity for the Calcium dff signal. While the average dimensionality of the activity of the PYR population is lower, this is partially due to the number of PYR units recorded being higher. (b). Average dff Calcium signal activation across different orientations for the VIP population during drifting gratings stimuli. Despite the trend appearing across orientations this is not significant as the Standard Error (not shown) is high due to the high variability across recordings.

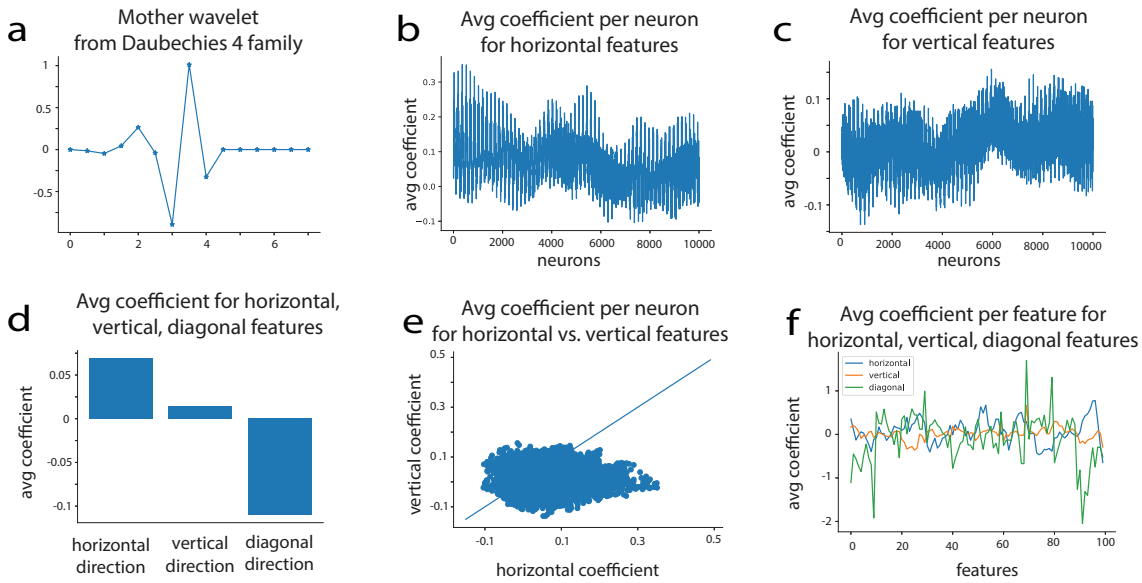


Figure 2.22: (a). Mother wavelet from the Daubechies wavelet family used to compute the wavelet transforms. (b)-(c). Average regression coefficients for 10,000 VIP neurons after performing a linear regression against VIP unit activations using patch averages of wavelet transforms computed from video frames as regressors. (b). Coefficients corresponding to horizontal features. (c). Coefficients corresponding to vertical features. (d). Regression coefficients for horizontal, vertical, and diagonal features averaged over neurons and over subsets of features. (e). Scatter plot of regression coefficients for different neurons, averaged over subsets of features, corresponding to the horizontal and vertical directions. (f). Regression coefficients averaged over neurons for different features which are the patch averages of wavelet transforms corresponding to video frames from our dataset.

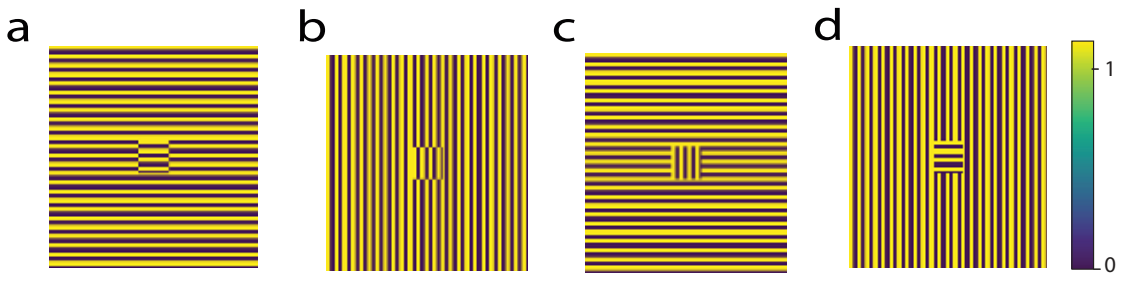


Figure 2.23: Stimuli shown to the static and moving circuits to probe whether neurons respond to iso-oriented surround differently than to cross-oriented surround as found by Keller et. al. (2020). (a)-(b) iso-oriented surround; (c)-(d) cross-oriented surround.

Bibliography

[1] © 2015 Allen Institute for Brain Science, Allen Brain Observatory, Available from: http://observatory.brain-map.org/visualcoding, 2016.

[2] Angelucci A. and Bressloff P. C. (2006). Contribution of feedforward, lateral and feedback connections to the classical receptive field center and extra-classical receptive field surround of primate V1 neurons. in Progress in Brain Research, volume 2006; 154, pages 93–120.

[3] Ayaz A and Saleem A.B. and Scholvinc M.L. and Carandini M., Locomotion Controls Spatial Integration in Mouse Visual Cortex (2013), Current Biology 23, 890–894, May 20, 2013

[4] Attneave F. (1954). Some informational aspects of visual perception. Psychological Review; Vol.61, No 3.

[5] Barlow H. (1961). Possible principles underlying the transformation of sensory messages. Sensory Communication; p217–234.

[6] Batista-Brito R. and Zagha E. and Ratliff J.M. and Vinck M., (2018). Modulation of cortical circuits by top-down processing and arousal state in health and disease, Current Opinion in Neurobiology 2018, 52:172–181.

[7] Bell A. J. and Sejnowski T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. Neural Computation, 1995; 7, 1129-1159.

[8] Bigelow J. and Morrill R. J. and Dekloe J. and Hasenstaub A. R. (2019). Movement and VIP Interneuron Activation Differentially Modulate Encoding in Mouse Auditory Cortex. eNeuro, 6(5) ENEURO September 2019; 0164-19.2019.

[9] Braitenberg V. and Schüz A. (1991). Anatomy of the Cortex: Statistics and Geometry. Springer-Verlag, Berlin.

[10] Campagnola L. and Seeman S.C. and Chartrand T. and Kim L. and Hoggarth A. and Gamlin C. and et. al. (2021), Local Connectivity and Synaptic Dynamics in Mouse and Human Neocortex. https://www.biorxiv.org/content/10.1101/2021.03.31.437553v2 [Preprint]. 2021 [cited 27 July 2021]

[11] Cardin J. (2019). Functional flexibility in cortical circuits. Current Opinion in Neurobiology, 2019, 58:175–180.

[12] Cardin J. (2018). Inhibitory interneurons regulate temporal precision and correlations in cortical circuits. Trends Neurosci; 41:689-700.

[13] Cauli B. and Audinat E. and Lambolez B. and Angulo M. C. and Ropert N. and Tsuzuki M. and Hestrin S. and et. al. (1997). Molecular and physiological diversity of cortical nonpyramidal cells. J Neurosci, May 1997; 17(10):3894-906.

[14] Chalk M. and Marre O. and Tkačik G. (2018). Toward a unified theory of efficient, predictive, and sparse coding. Proceedings National Academy of Science USA, Jan 2; 115(1):186-191. doi: 10.1073/pnas.1711114115

[15] Cohen J. D. and Dunbar K. and McClelland J. L. (1990). On the Control of Automatic Processes: A Parallel Distributed Processing Account of the Stroop Effect. Psychological Review Vol 97; No. 3, 332-361.

[16] Comon P. (1994). Independent component analysis, a new concept? Signal Processing, 1994; 36, 287-314.

[17] Cossell L. and Iacaruso M. F. and Muir D. R. and Houlton R. and Sader E. N. and Ko H. and et. al. (2015). Functional organization of excitatory synaptic strength in primary visual cortex. Nature; 518(7539):399–403, 2.

[18] Dadarlat M. C. and Stryker M. P. (2017). Locomotion Enhances Neural Encoding of Visual Stimuli in Mouse V1. The Journal of Neuroscience; 37(14):3764 –3775.

[19] Dayan P. and Hinton G. E.and Neal R. M. and Zemel R. S. (1995). The Helmholtz machine. Neural Computation, 1995; 7, 889-904.

[20] Dipoppa M. and Ranson A. and Krumin M. and Pachitariu M. and Carandini M. and Harris K.D., (2018). Vision and Locomotion Shape the Interactions between Neuron Types in Mouse Visual Cortex, Neuron 98, 602–615, May 2, 2018

[21] Doi E. and Lewicki M. S. (2014). A simple model of optimal population coding for sensory systems. PLoS Comput Biol. 2014 Aug; 10(8): e1003761.

[22] Durand S. and Iyer R. and Mizuseki K. and de Vries S. and Mihalas S. and Reid R. C. (2016). A Comparison of Visual Response Properties in the Lateral Geniculate Nucleus and Primary Visual Cortex of Awake and Anesthetized Mice. J Neurosci.; 36(48):12144-12156.

[23] Fu Y. and Tucciarone J. M. and Espinosa J. S. and Sheng N. and Darcy D. P. and Nicoll R. A. and et al. (2014). A Cortical Circuit for Gain Control by Behavioral State. Cell, Vol. 156; Issue 6, p1139-1152.

[24] Gao P. and Trautmann E. and Yu B. and Santhanam G. and Ryu S. and Shenoy K. and et. al. (2020). A theory of multineuronal dimensionality, dynamics and measurement. https://www.biorxiv.org/content/early/2017/11/05/214262 [Preprint]. 2017 5 Nov [cited 27 July 2021].

[25] Gouwens N. W. and Sorensen S. A. and Berg J. and Lee C. and Jarsky T. and Ting J. and et. al. (2019). Classification of Electrophysiological and Morphological Neuron Types in the Mouse Visual Cortex. Nat Neurosci 2019 Jul; 22(7):1182-1195. doi: 10.1038/s41593-019-0417-0

[26] Gozzi A. and Jain A. and Giovanelli A. and Bertollini C. and Crestan V. and Schwarz A. J. and et al. (2010). A Neural Switch for Active and Passive Fear. Neuron 67; 656–666.

[27] Harpur G. F. and Prager R. W. (1996). Development of low entropy coding in a recurrent network. Network; 7, 277-284.

[28] Hertäg L. and Sprekeler H. (2019) Amplifying the redistribution of somato-dendritic inhibition by the interplay of three interneuron types. PLoS Comput Biol, 2019 May 16;15(5):e1006999. doi: 10.1371/journal.pcbi.1006999.

[29] Hofer S. B. and Ko H. and Pichler B. and Vogelstein J. and Ros H. and Zeng H. and et. al. (2011). Differential connectivity and response dynamics of excitatory and inhibitory neurons in visual cortex. Nature Neuroscience 2011; volume 14, pages1045–1052.

[30] Hu B. and Mihalas S. (2018). Convolutional neural networks with extra-classical receptive fields. https://arxiv.org/abs/1810.11594v1 [Preprint]. 2018 [cited 27 July 2021]

[31] Iyer R. and Hu B. and Mihalas S. (2020). Contextual Integration in Cortical and Convolutional Neural Networks. Front. Comput. Neurosci., 2020; p31.

[32] Jiang X. and Shen S. and Cadwell C. R. and Berens P. and Sinz F. and Ecker A. S. and et. al. (2015). Principles of connectivity among morphologically defined cell types in adult neocortex, Science. Nov 27; 350(6264): aac9462. doi: 10.1126/science.aac9462

[33] Keller A.J. and Dipoppa M. and Roth M.M. and Caudill M.S. and Ingrosso A. and K.D. Miller and Scanziani M., A Disinhibitory Circuit for Contextual Modulation in Primary Visual Cortex (2020), https://www.biorxiv.org/content/10.1101/2020.01.31.929166v2, [Preprint]. 2021 [cited 27 July 2021]

[34] Karklin Y. and Simoncelli E. P. (2011). Efficient coding of natural images with a population of noisy Linear-Nonlinear neurons. Adv Neural Inf Process Syst. 2011 Dec; 24():999-1007.

[35] Kirkpatrick J. and Pascanu R. and Hadsel R. (2017). Overcoming catastrophic forgetting in neural networks. PNAS; 114(13) 3521-3526.

[36] Ko H. and Hofer S. B. and Pichler B. and Buchanan K. A. and Sjostro P. J. and Mrsic-Flogel T. D. (2011). Functional specificity of local synaptic connections in neocortical networks. Nature; 473(7345):87–91, 5.

[37] Koganezawa M. and Kimura K. and Yamamoto D. (2016). The Neural Circuitry that Functions as a Switch for Courtship versus Aggression in Drosophila Males. Current Biology 26; 1395–1403.

[38] Lefort S. and Tomm C. and Floyd Sarria J. C. and Petersen C. C. H. (2009). The Excitatory Neuronal Network of the C2 Barrel Column in Mouse Primary Somatosensory Cortex. Jan 2009; Volume 61, Issue 2, Pages 301-316.

[39] Lewicki M.S. and Sejnowski T.J. (2000). Learning overcomplete representations, Neural Computation, 12, pages 337-365, 2000.

[40] Litwin-Kumar A. and Harris K. D. and Axel R. and Sompolinsky H. and Abbott L. F. (2017). Optimal Degrees of Synaptic Connectivity. Neuron; 93(5):1153–1164.e7. doi:10.1016/j.neuron.2017.01.030

[41] McDougal R.A. and Morse T.M. and Carnevale T., Marenco L. and Wang R. and Migliore M. and Miller P.L. and Shepherd G.M. and Hines M.L. (2017). Twenty years of ModelDB and beyond: building essential modeling tools for the future of neuroscience. J Comput Neurosci. 2017; 42(1):1-10.

[42] Mallya A. and Davis D. and Lazebnik S. (2018). Piggyback: Adapting a Single Network to Multiple Tasks by Learning to Mask Weights. European Conference on Computer Vision (ECCV).

[43] Mallya A. and Lazebnik S. (2018). PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. Computer Vision and Pattern Recognition (CVPR).

[44] Mante V. and Sussillo D. and Shenoy K. V. and Newsome W. T. (2013). Context-dependent computation by recurrent dynamics in prefrontal cortex. Nature; volume 503, 78–84.

[45] Marr D. (1982). Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. San Francisco: WH Freeman and Company.

[46] Martin D. and Fowlkes C. and Tal D. and Malik J. (2001). A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics, ICCV.

[47] Mazzucato L. and Fontanini A. and La Camera G. (2016). Stimuli Reduce the Dimensionality of Cortical Activity. Front Syst Neurosci. 2016; 10: 11. doi:10.3389/fnsys.2016.00011

[48] Millman D. J. and Ocker G. K. and Caldejon S., Kato I. and Larkin J. D. and Lee E. K. and et. al. (2020). VIP interneurons in mouse primary visual cortex selectively enhance responses to weak but specific stimuli Elife. 2020;9:e55130. doi:10.7554/eLife.55130.

[49] Niell C. M. and Stryker M. P. (2010). Modulation of visual responses by behavioral state in mouse visual cortex. Neuron; 65(4): 472–479.

[50] Ollerenshaw D. R. and Zheng H. J. V. and Millard D. C. and Wang Q. and Stanley G. B. (2014). The Adaptive Trade-Off between Detection and Discrimination in Cortical Representations and Behavior. Neuron 81, 2014 March 5; 1152–1164.

[51] Olshausen B. A. and Field D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature; 381(6583):607–9.

[52] Olshausen B. A. and Field D. J. (1996). Natural image statistics and efficient coding. Network (Bristol, England); 7(2):333–9, 5.

[53] Olshausen B. A. and Field D. J. (1997). Sparse coding with an overcomplete basis set: a strategy employed by V1? Vision Res., Vol. 37, No. 23, pp. 3311-3325, 1997

[54] Olshausen B.A., Highly overcomplete sparse coding, (2013). Proc. of SPIE Vol. 8651 86510S-1

[55] Pfeffer C. K. and Xue M. and He M. and Huang Z. J. and Scanziani M. (2013). Inhibition of Inhibition in Visual Cortex: The Logic of Connections Between Molecularly Distinct Interneurons. Nat Neurosci. Aug; 16(8): 1068–1076. doi: 10.1038/nn.3446

[56] Pi H. J. and Hangya B. and Kvitsiani D. and Sanders J. and Huang Z. J. and Kepecs A. (2013). Cortical interneurons that specialize in disinhibitory control. Nature; 503:521-524.

[57] Poirazi P. and Brannon T. and Mel B. W. (2003). Pyramidal Neuron as Two-Layer Neural Network. Neuron, Mar 27; 37(6):989-99. doi: 10.1016/s0896-6273(03)00149-1

[58] Rao R. P. N. and Ballard D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. Nature Neuroscience, January 1999. volume 2, no 1.

[59] Rudy B. (2011). Three groups of interneurons account for nearly 100% of neocortical GABAergic neurons. Dev Neurobiol 71 (1); 45–61.

[60] Rusu A. A. and Rabinowitz N. C. and Desjardins G. and Soyer H. and Kirkpatrick J. and Kavukcuoglu K. and et. al. (2016). Progressive Neural Networks. arXiv:1606.04671 [Preprint]. 2016 [cited 27 July 2021].

[61] Serra J. and Sur is D. and Miron M. and Karatzoglou A. (2018). Overcoming Catastrophic Forgetting with Hard Attention to the Task. Proceedings of the 35th International Conference on Machine Learning, in PMLR; 80:4548-4557

[62] Simoncelli E. (2003). Vision and the statistics of the visual environment. Current Opinion in Neurobiology, Volume 13, Issue 2, April 2003, Pages 144-149

[63] Sussillo D. and Abbot L. F. (2009) Generating Coherent Patterns of Activity from Chaotic Neural Networks. Neuron 63; 544–557.

[64] Tasic B. and Yao Z. and Graybuck L. T. and Smith K. A and Nguyen T. N. and Bertagnolli D. (2018). Shared and distinct transcriptomic cell types across neocortical areas. Nature; volume 563, p72–78.

[65] Terekhov A. V. and Montone G. and O'Regan J. K. (2015). Knowledge Transfer in Deep Block-Modular Neural Networks. In: Wilson S., Verschure P., Mura A., Prescott T. (eds) Biomimetic and Biohybrid Systems. Living Machines. Lecture Notes in Computer Science, vol 9222. Springer, Cham. https://doi.org/10.1007/978-3-319-22979-9 27

[66] Thomson A. M. and West D. C. and Wang Y. and Bannister A. P. (2002). Synaptic connections and small circuits involving excitatory and inhibitory neurons in layers 2–5 of Adult Rar and cat neocortex: Triple intracellular recordings and biocytin labelling in vitro. Cerebral Cortex Oct 2002; 12(9):936-53.

[67] Vogt K. and Zimmerman D. M. and Schlichting M. and Hernandez-Nuñez L. and Qin S. and Malacon K. and et. al. (2020). Internal state configures olfactory behavior and early sensory processing in Drosophila larva. https://doi.org/10.1101/2020.03.02.973941 [Preprint]. 2020 [cited 27 July 2021]

[68] Voina D., Recanatesi S., Hu B., Shea-Brown E., Mihalas S. (2020). Single Circuit in V1 Capable of Switching Contexts During Movement Using an Inhibitory Population as a Switch. Neural Comput. 2022 Feb, 17;34(3):541-594.

[69] Wall N.R. and De La Parra M. and Sorokin J.M. and Taniguchi H. and Huang Z.J. and Callaway E.M., (2016). Brain-Wide Maps of Synaptic Input to Cortical Interneurons, Journal of Neuroscience April 2016, 36 (14) 4000-4009; DOI:https://doi.org/10.1523/JNEUROSCI.3967-15.

[70] Wilmes KA and Clopath C, (2019). Inhibitory microcircuits for top-down plasticity of sensory representations. Nature Communications volume 10, article no. 5055

[71] Yang G. R. and Cole M. W. and Rajan K. (2019) How to study the neural mechanisms of multiple tasks. Current Opinion in Behavioral Sciences; 29:134–143.

[72] Zemel R. S. (1993). A minimum description length framework for unsupervised learning. Ph.D. Thesis, University of Toronto, Department of Computer Science.

[73] Zenke F. and Poole B. and Ganguli S. (2017). Continual Learning Through Synaptic Intelligence. Proc Mach Learn Res. 2017; 70: 3987–3995.

[74] Zhou T. and Zhu H. and Fan Z. and Wang F. and Chen Y. and Liang H. and et al. (2017). History of winning remodels thalamo-PFC circuit to reinforce social dominance. Science 357; 162–168.

Chapter 3

A BIOLOGICALLY INSPIRED ARCHITECTURE WITH SWITCHING UNITS CAN LEARN TO GENERALIZE ACROSS BACKGROUNDS

In the previous chapter, we studied a biological circuit model where a neuronal population (the VIP) played a specific functional role and mediated switching between visual processing of static and moving states. Specific circuit components such as the VIP neurons, which we have referred to as *switching units*, and the recurrent connections between these units and another excitatory population have proved to be necessary ingredients for our network architecture. This chapter continues our exploration of the necessary and sufficient components of architectures in the context of artificial networks. Specifically, it searches for simple architectures that can perform continual learning: solving classification tasks while these are presented sequentially, without returning to previous tasks (the *sequential context switching problem*). Specifically, in our case, we focus on classification tasks where the background statistics change sequentially. A primary issue in this setting is that even deep feedforward networks "forget" older tasks, i.e. these networks have a diminished accuracy when performing older tasks. In the simplified setting of the ANNs where units are not strictly excitatory or inhibitory, switching units can be added in a similar way as for the V1 circuit in order to solve the sequential context switching problem. We analyze the network architecture required for continual learning, taking inspiration from the biological V1 circuit. We find that a few switching units turning ON and OFF to different context statistics and interacting with the main network through recurrent connections are sufficient to solve the sequential context switching problem. This sheds light on the significance of a bio-inspired network motif augmented to ANNs that enhances the network architecture to perform in more difficult settings, where classical deep architectures fail.

Work in this chapter has appeared in a manuscript on the bioarxiv [13].

3.1 Summary

Humans and other animals navigate different environments effortlessly, their brains rapidly and accurately generalizing across contexts. Despite recent progress in deep learning, this flexibility remains a challenge for many artificial systems. Here, we show how a bio-inspired network motif can explicitly address this issue. We do this using a dataset of MNIST digits of varying transparency, set on one of two backgrounds of different statistics that define two contexts: a pixel-wise noise or a more naturalistic background from the CIFAR-10 dataset. After learning digit classification when both contexts are shown sequentially, we find that both shallow and deep networks have sharply decreased performance when returning to the first background – an instance of the catastrophic forgetting phenomenon known from continual learning. To overcome this, we propose the bottleneck-switching network or switching network for short. This is a bio-inspired architecture analogous to a well-studied network motif in the visual cortex, with additional "switching" units that are activated in the presence of a new background. Intriguingly, only a few of these switching units are sufficient to enable the network to learn the new context without catastrophic forgetting through inhibition of redundant background features. Further, the bottleneck-switching network can generalize to novel contexts similar to contexts it has learned. Importantly, we find that – again as in the underlying biological network motif, the switching units must be recurrently connected to network layers for context generalization to succeed.

3.2 Introduction

The ability to adapt to changes in context while preserving relevant information that is contextually invariant is one of the traits that make biological brains so effective and robust in complex, natural environments. In the visual domain, for example, humans are particularly adept at generalization across contexts and will react similarly when seeing a familiar face during a remote video call or during an interaction in person. Translating this capacity for contextual adaptation to artificially intelligent agents is essential if, for example, we are to upgrade current visual learning algorithms to generalize across new environments and abstract visual concepts [4]. A central motivation of this study is to find the relevant network

mechanisms that allow artificial neural networks to accomplish adaptation and switching to different contexts. One difficulty in this setting is that most current deep feedforward architectures learn the background and foreground in tandem, without abstracting objects of interest from their surround. For example, recent studies show networks that repeatedly misclassify familiar objects set on new backgrounds [4], while select adversarial backgrounds may fool even deep state-of-the-networks up to 87% of the time [69]. As we will show, simply relying on network depth is insufficient. Instead, novel architectures supporting context-dependent computations are required.

In general, datasets used to train current state-of-the-art neural networks are shown to be biased [65, 2, 15, 10], with object class correlating with background, or the background statistics being constrained to belong to a particular distribution. For example, certain datasets may predominantly contain street or nature scenes, or have a preferred viewing angle [62]; in the case of ImageNet, the dataset predominantly contains centered objects with limited background clutter, while the PascalVOC dataset depicts more complex scenes with multiple objects and a significant amount of background clutter [47]. Such biases prevent even the most sophisticated deep architectures from generalizing across contexts so that when the network is tested on a dataset with different background statistics the accuracy can be significantly affected [3]. Few studies so far, such as [4, 3], have disentangled different dataset biases to separately address how tasks like object classification are affected by variations of context.

To address these challenges systematically, we first construct a simple dataset where context is clearly defined and without other complicating biases. The dataset consists of MNIST digits set on either pixel-wise noisy backgrounds or more naturalistic backgrounds of images from the CIFAR-10 dataset (Figure 4.3A). To parametrically vary the difficulty of this task, we control the transparency of the MNIST digits. We use this dataset in a biologically realistic setting where agents learn to classify digits set on different backgrounds but are exposed to these contexts sequentially and without having access to previous data points. This is the continual learning framework, where a major challenge is that old tasks are

forgotten when network weights are overwritten to solve new tasks, a phenomenon called *catastrophic forgetting* [45, 50]. We show that both deep and shallow networks trained to classify MNIST digits set on the two contexts fail to sequentially learn without catastrophic forgetting. We refer to this classification task with varying context as *sequential context switching*.

We introduce a new network architecture for solving sequential context switching that is roughly inspired by a recently characterized local circuit in the mouse visual cortex. This circuit in the mouse primary visual cortex (V1) modulates its activity whenever an animal shifts from stationary to moving behavior via a neuronal population expressing Vasoactive Intestinal Peptide – the VIP population. It has been hypothesized that this neuron population, which is recurrently connected to other neurons in V1 (e.g., Pyramidal and Somatostatin neurons), turns ON like a switch and reconfigures the circuit dynamics to efficiently process the corresponding static vs. moving visual scenes [63, 74]. Inspired by this biological circuit motif and without emulating other features of the V1 circuit, we propose a network architecture – the *bottleneck-switching network* or *switching network*, for short – where, using a feedforward neural network to start (Figure 4.3D), we add units that are OFF when the network is presented the first context but turn ON for the second context, similarly to the VIP (Figure 4.3C-D). We refer to these added units as *switching units*. Due to this network's architecture and the training method implemented (Sec. 3.4), *catastrophic forgetting is guaranteed not to occur*.

Our paper makes the following main points: (1) We introduce and share a simple dataset for the classification of MNIST images set on contexts of different (parametric and non-parametric) statistics. (Sec. 3.1, Figure 4.3A). (2) Using this dataset, we demonstrate systematically how catastrophic forgetting occurs in sequential context switching for both deep and shallow neural networks (Sec. 3.2, Figure 4.4). (3) We then test a set of basic network architectures that incorporate context in different ways, and find that they fail to achieve sequential context switching (Sec. 3.3, Figure 3.3). (4) We propose a bio-inspired architecture, the *bottleneck-switching network*, that succeeds in sequential context switching

while also using a few additional units (Sec. 3.4, Figure 3.4). (5) We show how the *switching network* can improve performance relative to other established methods by assessing the performance of EWC [29] and ProgNet, [54] in the sequential context switching task (Sec. 3.5, Figure 3.5). (6) We propose a mechanism behind switching networks' high performance, via a sparsification of the initial network activities, specifically through inhibition of background features (Sec. 3.6, Figure 3.6). (7) Testing digit-background pairs that have not been trained on the switching network, we find that our network has superior performance to a feedforward network trained on one context (Sec. 3.7, Figure 3.7A-B). (8) Further, we show that switching networks can generalize well to distinct contexts whose image statistics are similar to the ones it has trained on (Sec. 3.8, Figure 3.7C-E).

3.3 Related Work

Transfer learning (TL), domain adaptation (DA), multi-task learning (MTL), and continual learning (CL) are allied fields that have the broad goal of training networks on two or more datasets or tasks, with the common objective of using the knowledge learned from one task (source task) to efficiently learn a different but related task (target task). General strategies [66] applied to this end include trying to correct for the input marginal distribution difference [34, 18, 57, 21, 48, 47] or the conditional distribution difference [14, 61, 72, 9, 19, 39, 68] between the source and target tasks. In this paper, we focus on context switching, when only the features and statistics of the background change. This scenario is most similar to domain adaptation (DA), a particular case of TL (specifically, heterogeneous, transductive TL, [12, 67]) when the source and target tasks are identical but the source and target features, as well as their distribution are different due to selection bias or distribution mismatch.

A rich literature spanning decades describes *shallow DA* methods aiming to solve the domain shift between the source and target domains by broadly using two strategies [67]: (1) train models on reweighed source samples to reduce the discrepancy [8, 11]; and (2) find a common shared space where the two domain distributions are matched [22, 48]. Current state-of-the-art DA methods use deep networks – *deep DA* – which have reliably outperformed previous strategies. Various approaches are based on the fine-tuning of deep

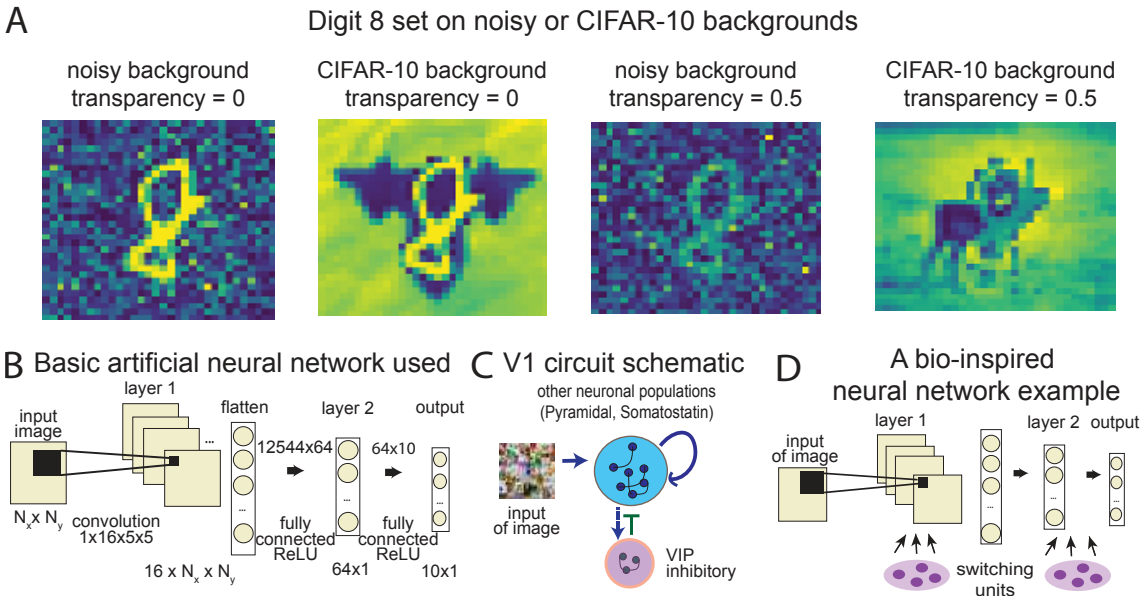


Figure 3.1: Contexts and networks used for sequential context switching. A) Digit "8" set on two different backgrounds (noisy and CIFAR-10 background) with different levels of transparency T. B) Basic artificial neural network architecture without switching units used for the toy examples presented below. C) Schematic of the biological circuit in V1 with the VIP population acting as binary switch that turns ON/OFF due to the stationary and moving states of the animal [74]. G) A bio-inspired neural network with switching units.

networks, either by aligning the statistical distribution shift [40, 60], or by adjusting the architectures of deep networks [35, 70]. Another popular method is the adversarial deep DA approach, where a domain discriminator that decides whether the data point belongs to the source or the target domains is used to encourage domain confusion through minimizing the distance between the source and target input distributions [38, 6]. Other well-known methods use autoencoders to reconstruct the inputs, focusing on creating a shared representation between the two domains while maintaining the individual characteristics of each [7]. However, these methods do not specifically address generalization across background and are not tested against a common benchmark. Furthermore, while these network architectures have performed impressively in Machine Learning tasks, to the best of our knowledge they have not been related to specific biological circuits.

Here we focus on the case when the data is observed incrementally as a continuous stream – the CL setting. Unlike learning in TL and MTL, the network in this case does not have access to old data that can be interleaved with the new data, but rather agents continually learn new knowledge across time while retaining previously learned information [49]. While the dataset of different contexts we have chosen distinctly requires us to consider DA as an appropriate framework, we have chosen a task (sequential context switching) whose structure promotes the formation of context-independent features through the CL framework, assuming that classification without catastrophic forgetting is conducive to context invariance and generalization. Datasets/contexts and tasks requiring DA are not always studied in the CL setting, however our study of sequential context switching implies that our work lies at the intersection of the two fields.

Models in CL have taken inspiration from neurophysiological principles (e.g. Hebbian plasticity, compensatory homeostatic plasticity, [76, 1], and from computational learning models (complementary learning systems theory, [44, 30]). There are several categories of computational approaches to CL that permit good generalization and avoid catastrophic forgetting [49]: (i) learning models that regulate levels of plasticity to protect consolidated knowledge through regularization [36, 51, 16]; (ii) adding additional neural resources such as neurons

to learn new information [78, 71, 17, 73, 52]; (iii) using complementary learning systems for memory consolidation and experience replay [23, 20, 58, 28]. In the case of regularization methods, a successful strategy has been applied in [29], describing the EWC, and in [77] by which more influential parameters from previous tasks are pulled back towards a reference weight with good performance on previous tasks. An example of the second approach where the network dynamically adds neuronal resources is the ProgNet architecture [54]. This architecture expands through the allocation of new "column" networks, trained on novel information, and receiving lateral connections from the other columns. We will describe ProgNet as presented in [54], and EWC as introduced in [29] in more detail below.

Direct comparison between these methods has been difficult because of a lack of established benchmark datasets and metrics. Many training and evaluation protocols have shifted from MNIST or CIFAR-10 datasets [29, 77, 26, 75], to more challenging datasets (ImageNet, [53]; MS COCO, [37]; OpenImages, [31]), similar in complexity to realistic settings [29, 36, 46, 42, 41, 27]. However, large datasets also contain substantial contextual biases, in background/context, rotation, viewpoint, etc., and have insufficient controls to ensure networks do not exploit trivial correlations in the data [3, 69, 79, 56, 55]. For instance, in [69] authors find that changing backgrounds in ImageNet significantly decreases average performance, and that choosing backgrounds in an adversarial manner can lead to misclassifying 87.5% of the images.

Therefore, an important undertaking is to carefully analyze generalization (or lack thereof) in detection and classification tasks as discussed in [4], dissecting the biases neural networks can abuse or misuse (variations in lightning, viewpoints, context/background etc.). To that end, our work distinctively addresses variations of context for a classification task, when the network has to generalize by ignoring contexts and abstracting MNIST digits. More precisely, this work sets out to (1) systematically address context generalization for parametric and non-parametric backgrounds, and specifically when the statistics of the background is varied in a controlled way (see Sec. 3.8); (2) do a thorough comparison between our bio-inspired architecture, the bottleneck-switching network, and relevant prior methods like

EWC and Prognet, or other simple architectures.

3.4 Results

3.4.1 A simple dataset for sequential context switching

To address the problem of sequential context switching, we create a dataset whose only confounding feature is context. We make this dataset publicly available to test generalization across contexts. The dataset consists of MNIST digits with varying degrees of transparency set on either noisy backgrounds, with noise being the absolute value of a Gaussian random variable normalized appropriately, or MNIST digits set on a more naturalistic background from the CIFAR-10 dataset (Figure 4.3A, Methods Sec. 1.1). We refer to the subset of MNIST digits set on noisy backgrounds as "MNIST+noise" and to the subset of MNIST digits set on CIFAR-10 backgrounds as "MNIST+ cifar". The goal is to perform image classification so that neural networks (NN) correctly identify the MNIST digit despite the different backgrounds. A parameter that makes the task more difficult is digit transparency; as we increase transparency, the background interferes with the digit and the identity of the digit becomes more ambiguous (Figure 4.3A, right).

3.4.2 Catastrophic forgetting in the MNIST+noise and MNIST+cifar datasets

To systematically address the problem of sequential context switching and infer suitable network architectures, we start by first choosing a basic NN (Figure 4.3B) that solves the digit classification task with a performance above 90% on either MNIST+noise (96% accuracy) or MNIST+cifar (93% accuracy) datasets with no digit transparency (Figure 4.4B,Figure 3.8B). The classic MNIST classification task can be performed at high accuracies using even simple linear classifiers, with accuracies as high as 92.4% reported in [32]. However, the dataset we focus on here contains backgrounds and additional transparency T causing the backgrounds to obscure the digits, making this a more difficult task. For instance, for intermediate levels of transparency ($T = 0.5$) accuracies are 94% and 85% for MNIST+noise and MNIST+cifar, respectively. For high levels of transparency ($T = 0.85$), the same accuracies are 67% and 25%, respectively.

For details on how the training of the basic NN and other networks is run, see Supplementary Material, Methods Sec. 1.2.2.

Turning to the problem of continual learning applied to context switching, we ask whether this NN learns digit classification on MNIST+noise and MNIST+cifar, sequentially and in either order. To achieve continual learning, we implement the following steps (Figure 4.4A, Figure 3.8A):

- at **step 1**, we train the network on context 1, then test on context 1 – this is the *matched condition*;
  - we test context 2 on the network trained on context 1 – this is the *unmatched* condition;
- at **step 2**, we retrain the NN on context 2 and then test on context 2 – this is the *re-matched condition*;
- at **step 3**, we re-test context 1 – this is the test catastrophic forgetting or *test CF condition*, where context $1, 2$ are either MNIST+noise or MNIST+cifar.

Our initial findings are as follows:

- at **step 1**, networks learn digit classification on context 1 with high accuracies (Figure 4.4B, Suppl. Figure 3.8B), in this case up to 96% on the test set (Figure 4.4B), with reduced accuracy at increased transparency.
  - Testing the NN in the *unmatched condition* using context 2, the accuracy is comparatively poor (orange vs blue lines in Figure 4.4C). This is the effect of changing context: weights from the prior context can no longer be used for accurate classification in the novel context.
- At **step 2**, for the *re-matched condition*, we obtain a performance (green line, Figure 4.4C) comparable to the *matched condition*, as if we trained on context 2 from scratch (blue line).

- At **step 3**, in the *test CF condition*, the network substantially fails to remember the original dataset MNIST+noise after being retrained on MNIST+cifar (Figure 4.4D) and vice versa (Figure 3.8D). This is especially true in cases of increased transparency.

Moreover, increasing either the depth or the width of the network does not alleviate this forgetting. For depth, we use the well-known VGG-16 [59]. For width, we increase twofold the number of filters or hidden units in the convolutional and linear layers, respectively. Figure 4.4E-G, Figure 3.8E-G, and Figure 3.9 all show that the accuracy of the network in the *test CF condition* (orange line) remains lower than in the *matched condition* (blue line). In summary, Figure 4.4B-G demonstrates that CF occurs when context changes in a step-wise manner, indicating a failure of standard feedforward NN's in the present continual learning setting.

3.4.3 Simple networks with contextual input, output, or feedforward switching units fail to perform sequential context switching

We next test three simple modifications to the networks that could avoid catastrophic forgetting.

The first strategy is to add a binary contextual input to the NN, similar to the input received in some widely used recurrent neural network models (e.g., see [43]). The input is added to the hidden layers (Figure 3.3A) and is set to 0 for context 1 (e.g., MNIST+noise) and 1 for context 2 (e.g., MNIST+cifar). We implement the following steps to test this architecture:

- at **step 1**, we test context 1 on a network trained on context 1, without any contributions from binary units, which are set to 0 – this is the *matched condition*;

- at **step 2**, we train the NN on context 2 while setting the binary units to 1 and learning weights from these units in tandem with other network weights;

- at **step 3**, the binary units are set to 0 once again and we test the performance of the network on context 1 – this is the *test CF condition*.

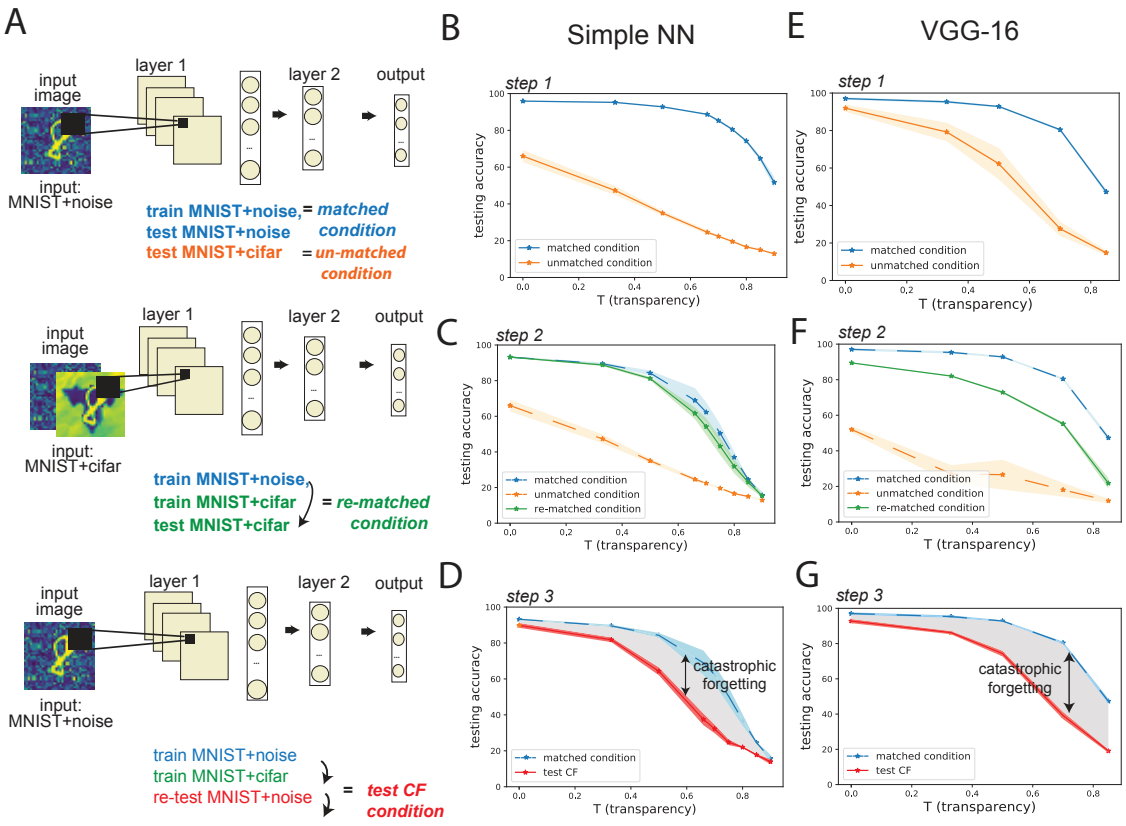


Figure 3.2: The catastrophic forgetting phenomenon occurs for sequential context switching in both deep and shallow neural networks. A) Schematic of training and testing for sequential context switching to test catastrophic forgetting. B) Average performance on the *matched condition* for MNIST+noise (blue) and on the *unmatched condition* for MNIST+cifar (orange) as transparency T increases. C) The MNIST+cifar dataset is less accurately classified by a network trained on MNIST+noise (*unmatched condition*, orange line) than one trained on MNIST+cifar (*matched condition*, blue line). However, once we re-train the MNIST+noise-trained NN on MNIST+cifar (Step 2), the accuracy approaches that for the *matched condition* (*re-matched condition*, green line). D) The performance when re-testing MNIST+cifar on a network that was first trained on MNIST+cifar, then on MNIST+noise, is reduced (*test CF*, red line) compared to the *matched condition*, therefore catastrophic forgetting occurs. E)-G) same as B)-D) using the VGG-16 network instead of the basic network.

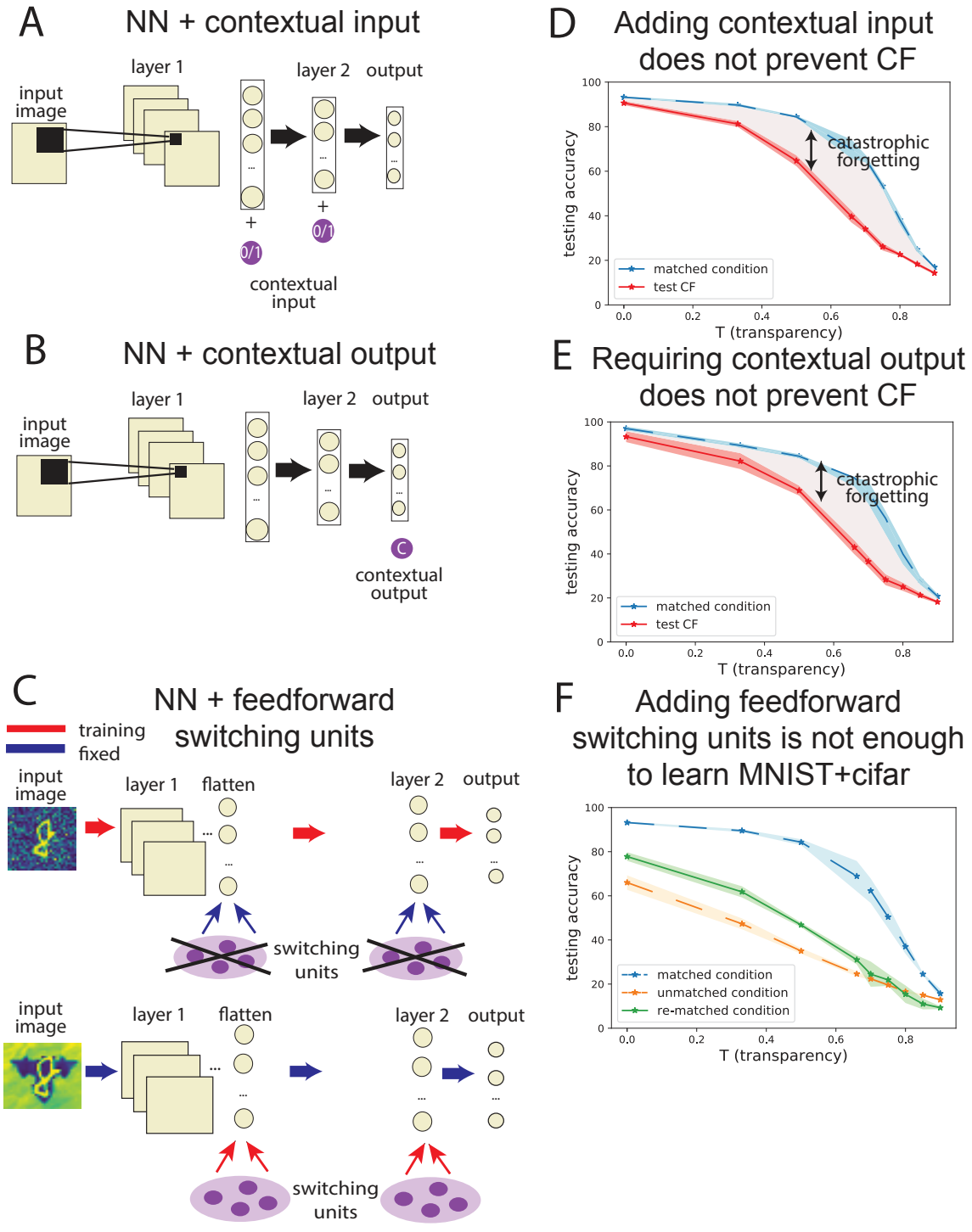


Figure 3.3: Simple network architectures that fail to perform sequential context switching. Schematics for A) the neural network with contextual input; B) the neural network with contextual output; C) the neural network with feedforward switching units. Testing CF for MNIST+cifar: D) adding contextual input information; E) requiring contextual output. F) Adding switching units with feedforward connections to the hidden layers.

We find that catastrophic forgetting still occurs (red vs blue lines in Figure 3.3D, Figure 3.10C).

The second strategy was to require a separate binary output for context: 0 for context 1 and 1 for context 2 (Figure 3.3B). The steps we go through here are similar, except for replacing the added binary units with binary outputs which are required to be 0 when training/testing context 1 (steps 1,3) and 1 for context 2 (step 2). Again, this fails to reduce catastrophic forgetting (red vs blue lines in Figure 3.3E, Figure 3.10D). The idea underpinning these strategies is that by either directly providing knowledge about the identity of the context (noisy or CIFAR-10 context) – or explicitly requiring the network to represent this context in its output – the NN would be able to separate contextual information from digit information, and would learn to use only the latter in the classification task.

The third strategy takes inspiration from the local circuit in the mouse visual cortex discussed above [63, 74], where the circuit modulates its activity whenever an animal shifts from stationary to moving behavior via the "VIP" neuronal population. It has been hypothesized that this neuron population, which is recurrently connected to other neurons in V1 (e.g., Pyramidal and Somatostatin neurons), turns ON like a switch and reconfigures the circuit dynamics to efficiently process the corresponding static vs. moving visual scenes [63, 74].

In our first, simplest model inspired by this circuit, we add *switching units* akin to the VIP neurons: they are OFF for the first context and ON for the second context, while abstracting away other VIP properties (e.g. their inhibitory effect and their recurrent connectivities). This architecture is the "feedforward switching network", with the NN from Figure 4.3B as the main network, while the switching units added are auxiliary units (Figure 3.3C). We describe the step-by-step training applied to the feedforward switching network in the following:

- at **step 1**, we train then test context 1 with the switching units OFF – this is *matched condition*.

- at **step 2**, we hold the weights learned at step 1, when the switching units were OFF, fixed. We then train the NN on context 2 while setting the switching units to 1 and learning weights only from these units. The contribution of the switching units gets added to the activations of the main network hidden layer before it gets through a ReLU non-linearity. For the convolutional layers, we add the switching units' contributions to the flattened hidden activations of the network. The resulting performance is the *re-matched condition*.

- we no longer require a separate **step 3**, as catastrophic forgetting is guaranteed not to happen. To re-test digit classification on context 1, we simply turn the switching units OFF, returning the network (and therefore its performance) to an identical state to step 1's *matched condition*.

We note that this is a similar network to the one with binary inputs described above, with one important difference: after training for context 1 with the switching units OFF, we *freeze* the weights already learned and only changing the weights from the switching units.

For this switching network we emphasize a key fact: we are guaranteed to return to original performance on the first learned context by simply turning the switching units OFF. Thus, such a network will automatically overcome catastrophic forgetting – if it can solve the task on context 2 by only learning the weights from the switching units. However, we find that this does not occur. The feedforward switching network achieves accuracies on context 2 (Figure 3.3F, green line) only slightly higher than the *unmatched condition* when using a network trained on context 1 alone (orange line), and far below the reference *matched* condition (blue line). (This is consistent with allied findings in [63]). As before, the conclusions hold when context 1 is for either MNIST+noise or MNIST+cifar (Figure 3.10D), and we assume this is the case throughout the paper unless otherwise noted. In sum, while the feedforward switching network does not, by construction, show catastrophic forgetting, it has inferior performance on the second context.

We conclude that none of the three strategies considered above is successful in sequential context switching. In the next section, we introduce an improvement to the feedforward switching network that attains success.

3.4.4 Networks with recurrent switching units succeed in performing sequential context switching

We next show that we can improve the performance of the feedforward switching network by adding recurrent connections to the switching units (blue or red arrows from the switching units, Figure 3.4A, left). These connections are directly inspired by biological data and confirmed by a prior computational model of the V1 circuit described in [63], which suggests recurrence from switching units is necessary and sufficient for context-dependent computations. Our goal is to study a bio-inspired circuit motif that incorporates into a feedforward artificial NN architecture a specific cell type reproducing the simplified activity of the VIP neural population. We then assess whether this motif enables sequential context switching between backgrounds of different statistics, analogously to the V1 circuit switching between static and moving contexts.

Inspired by the overall structure of the V1 circuit motif we propose a network architecture – the *bottleneck-switching network* (or *switching network*). This is the same as the feedforward switching network above (Figure 3.3C) but allows recurrent connections back to switching units. In detail, using a feedforward NN to start (Figure 4.3B), we add *switching units* akin to the feedforward switching units from the previous section (Sec. 3.3), only that they are recurrently connected to the main network units. As before, the switching units are OFF when the network is presented the first context, but turn ON for the second context, similarly to the VIP (Figure 4.3C,D). Due to this network's architecture and the training method implemented, *catastrophic forgetting is guaranteed not to occur.*

As for the feedforward switching network, we follow the same sequence of steps:

- at **step 1**, we train then test context 1 with the switching units OFF – this is *matched*

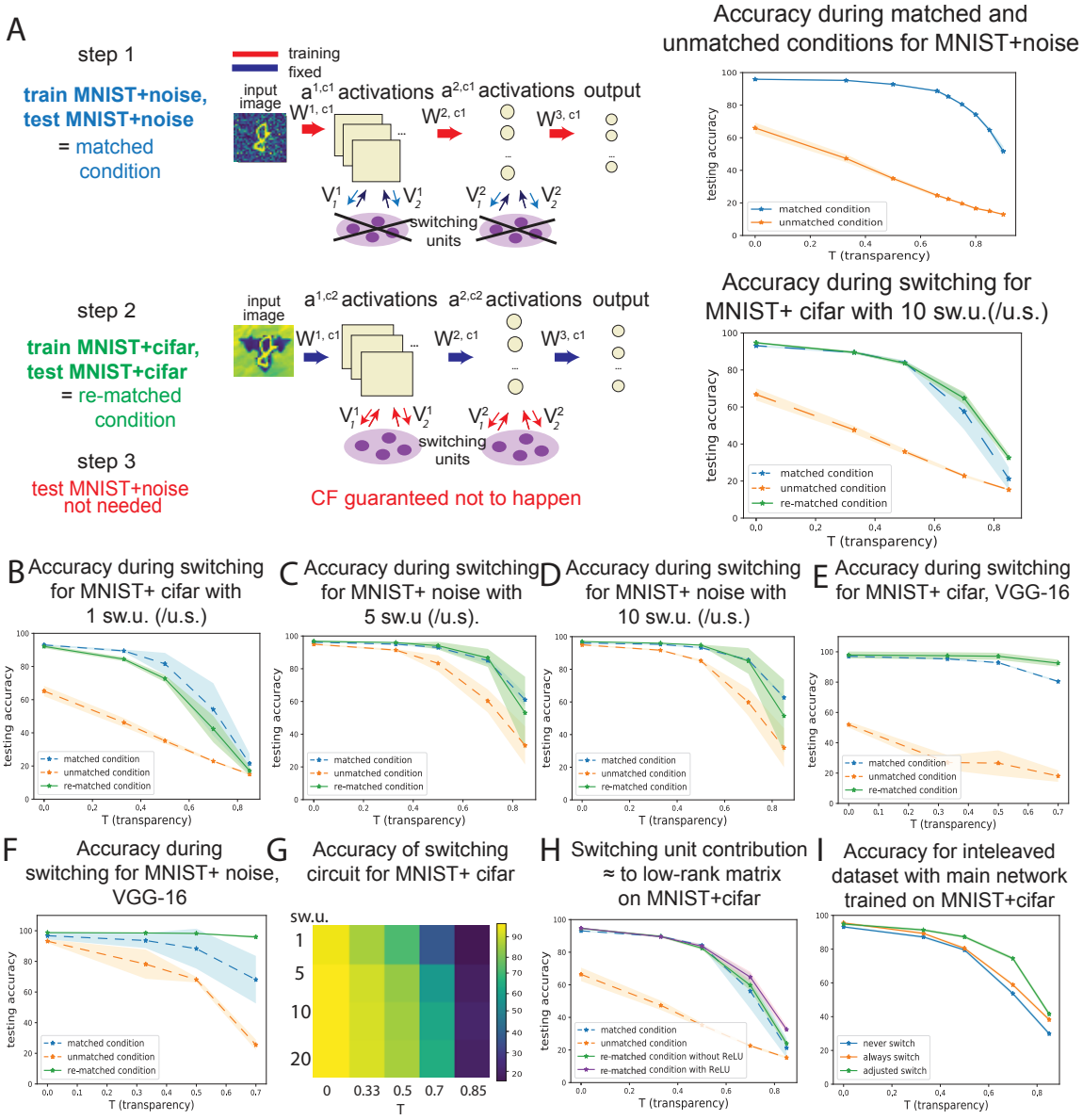


Figure 3.4: A) Summary and schematic of the training procedure for the bottleneck-switching network. B-F) Accuracy on MNIST+noise/cifar vs transparency T: after training in the *matched condition* (step 1, blue line); after training in the *unmatched condition* (orange line); after training in the *re-matched condition* with switching units. E-F) Accuracy on the VGG-16 with switching units. G) Heat map of accuracy on the basic network across a number of switching units and transparency T. H) Accuracy of a linearized version of the switching network that is equivalent to a low-rank perturbation on weights. I) Accuracy using the network with a built-in downstream module that classifies contexts and turns the switching units ON or OFF automatically.

condition.

- at **step 2**, we hold the weights learned at step 1, when the switching units were OFF, fixed. We then train the NN on context 2 while setting the switching units ON and learning weights to and from these units. The contribution of the switching units gets added to the activations of the main network hidden layer before it gets through a ReLU non-linearity. The resulting performance is the *re-matched condition.*

- we no longer require a separate **step 3**, as catastrophic forgetting is guaranteed not to happen.

When adding switching units to convolutional hidden layers, we aim to perform convolutional operations from the hidden layers to the switching units and back to the main network, so we essentially add kernels. Each kernel is a square matrix that extracts spatial features from the images, so we refer to the number of kernels added as switching units per unit space (sw.u/u.s.).

The activities of this improved network at layer l for contexts c_1, c_2 can be expressed as:

$$\begin{aligned} a^{l,c_1} &= \sigma(W^{l,c_1} a^{l-1,c_1}) \\ a^{l,c_2} &= \sigma(W^{l,c_1} a^{l-1,c_2} + V_2^l \sigma(V_1^l W^{l,c_1} a^{l-1,c_2})). \end{aligned} \tag{3.1}$$

where W^{l,c_1} are the weights between layers $l-1$ and l of the main network for context 1 (c_1), σ is the non-linearity (ReLU), a^{l,c_i} are the activations at layer l for context c_i, a^{0,c_i} is the input image, and V_1^l, V_2^l are the weights to and from the switching units at layer l, as shown in Figure 3.4A. For classifying digits set on the first context c_1, we simply use the basic NN, a feedforward neural network whose activities at layer l are computed via the first equation by applying a linear operation to the activities at the previous layer/input using the weights W^{l,c_1}, then implementing a non-linear function σ (Figure 3.4A (step 1)). For classifying digits set on c_2, we apply a similar equation to compute the activities in the network layers, except we add an additional term from the switching units, $V_2^l \sigma(V_1^l W^{l,c_1} a^{l-1,c_2})$ that essentially projects the activations in the current layer, $W^{l,c_1} a^{l-1,c_2}$ onto a lower dimensional

space (the switching units), applies a non-linear function, then finally projects back onto the layer l activities via the matrix V_2^l (Figure 3.4A (step 2)).

When the switching units are ON, we learn connections to and from these units, intending to classify classifying context 2. This strategy succeeds: we find that the accuracy on context 2 for different values of T is much improved when adding even one switching unit per unit space at the first layer (green line in Figure 3.4B). We can compare this performance (*re-matched condition*) to the *unmatched* and *matched conditions*: the *unmatched condition* entails testing context 2 on the NN trained on context 1 (orange lines) and represents a lower bound, while the *matched condition* implies testing context 2 on the NN trained on context 2 (blue lines) and is the accuracy we want to reach or exceed. As we add more switching units, the performance of the bottleneck-switching network on context 2 approaches or even surpasses the *matched condition*. Using 5 switching units (5 sw.u./u.s. for convolutional layers and 5 simple units for regular hidden layers), the performance reaches that of the *matched condition* (Figure 3.4C), with an even more pronounced boost in performance as we increase the number of switching units to 10 (Figure 3.4A,D p-value $< 8.6 \cdot 10^{-5}$ on MNIST+cifar). When adding 10 switching units we use small 3×3 kernels, which causes the total number of parameters to be comparatively reduced. Switching units applied to the convolutional layers of the much deeper VGG-16 network also achieve sequential context switching, with superior accuracy in the *re-matched condition* using switching units to classify MNIST+cifar (Figure 3.4E-F).

Thus, we see that generally few switching units are required for the highest accuracy to be achieved. Using the basic NN as the main network, the accuracy either plateaus or increases slowly, such that with $5-10$ switching units we are close to peak performance (Figure 3.11). Using the same basic network, a summary of how testing performance varies with the number of switching units and transparency shows the same tendency of the accuracy to plateau as switching units increase (Figure 3.4G). For the VGG-16, we use approximately a tenth of the number sw.u/u.s. present in the VGG main network, and solely for a subset of the convolutional layers (see Methods Sec 1.2.1,), hence comparatively fewer switching units are

used for the deep network as well.

To further analyze the switching network motif, we next study a version of this network simplified by eliminating the ReLU non-linearity σ after the switching contributions are computed. In other words, we study a linearized version of the recurrent switching component of the network. The activities of the NN after this simplification can now be expressed as:

$$\begin{aligned} a^{l,c_2} &= \sigma(W^{l,c_1}a^{l-1,c_2} + V_2^l V_1^l W^{l,c_1} a^{l-1,c_2}) \\ &= \sigma((W^{l,c_1} + V_2^l V_1^l W^{l,c_1})a^{l-1,c_2}). \end{aligned} \tag{3.2}$$

The contribution from the switching units is $V_2^l V_1^l W^{l,c_1} a^{l-1,c_1}$. This contribution becomes a low-rank addition to the activities of the NN if few switching units are required, i.e. V_1^l, V_2^l are lower dimensional in one of the two dimensions of the matrix. We find that this low-rank contribution successfully performs sequential context switching. Figure 3.4H shows the performance of the bottleneck-switching network on context 2 (*re-matched condition*) approaching and surpassing the *matched condition*, just as before, even as accuracies are reduced compared to when the ReLU non-linearity is used. This suggests that varied network mechanisms that implement low-rank updates to weight matrices may be sufficient to support contextual switches. Further, we can link our work to a rich body of literature that emphasizes the effectiveness of low-rank weights in NNs trained for a variety of tasks (Bau et. al. (2020), Swaminathan et.al. (2020)).

We then ran a series of controls to further test the effectiveness of the bottleneck-switching network.

First, we shuffled labels for images set on context 2, while images set on context 1 had typically assigned labels ("1" for a written digit of 1, etc.). We find that switching units do not work as well when the context is unchanged, but the task has different input-output dependencies (Figure 3.12A,B). This suggests that for sufficiently different datasets – differ-

ing through more than the change of context features, the switching network fails to reach similarly high performance.

Second, we assessed a network where only weights to and from the switching units are learned, while the weights in the main network are fixed and randomly initialized (Figure 3.12C,D). The goal is to establish baseline performance when the main network does not contribute features from context 1 to the task. To infer this baseline performance, we train weights to and from the switching units (turned ON) on context 2, without modifying weights in the main network. We then test this network on context 2 to find that the accuracy on context 2 is low, barely surpassing the *unmatched condition*, especially when the switching network with switching units ON learns MNIST+cifar (red line, Figure 3.12D). We conclude that main network weights and activations from training on a similar dataset are necessary for sequential context switching.

Finally, we address a simplification we made in the framework presented above: that switching units turn ON/OFF based on perfect knowledge of the context presented. For a more realistic case, in which contexts are detected from images, the network can include a downstream module consisting of a one-layer network to identify contexts through a binary classification using supervised learning, with *a priori* training on the noisy and CIFAR-10 backgrounds. In this setting, the switching network includes binary multiplicative inputs from the one-layer network that enable the switching units to turn ON and OFF automatically based on context. Augmenting such a module to classify contexts to the bottleneck-switching network does not deteriorate performance (Figure 3.4I, Figure 3.21), as can be seen by comparing the accuracy with a trained module augmented (green line), compared to the accuracies when the network never turns ON the switching units (orange line), or when the switching units are always OFF (blue line).

We conclude that relatively few switching units that are recurrently connected to the main network layers can provide substantial performance improvement in the sequential context switching task and that the switching unit contribution can be approximated to a low-rank perturbation to the weights.

3.4.5 A comparison between the bottleneck-switching network and two established continual learning methods

We next compare the bottleneck-switching network with two other learning methods that could achieve generalization across contexts: Progressive Networks and Elastic Weight Consolidation. Whereas many studies in the literature describe these methods' ability to perform continual learning by switching between *tasks*, we are focused on switching between *contexts*, and will test the noisy and CIFAR-10 contexts in our framework.

Progressive Network (ProgNet). ProgNets [54] maintain a set of pre-trained networks ("columns") for each task (or context) and learn lateral connections between these columns to extract useful features from related tasks (contexts) (Figure 3.5A, Methods sec. 1.2.3). During learning, the parameters of columns trained on previous contexts are kept constant, so weights are not overwritten and the network is immune to catastrophic forgetting by design. Considering only two columns corresponding to classification of different contexts (noise/CIFAR-10), the activations in column 2 of the ProgNet can be expressed as:

$$a^{l,c_2} = \sigma(W^{l,c_2} a^{l-1,c_2} + U^{l,c_{12}} a^{l-1,c_1}), \tag{3.3}$$

where a^{l,c_k} are the activations of layer l of column c_k, $c_k \in \{1, 2\}$, $W^{l,c_k} \in \mathbb{R}^{n_l \times n_{l-1}}$ is the weight matrix of layer l and column c_k, n_l are the number of units in these hidden layers, $U^{l,c_{12}} \in \mathbb{R}^{n_{l-1} \times n_l}$ are the lateral connections from layer $l-1$ of column 1 to layer l of column 2, a^{0,c_k} is the network input for column c_k, and σ is an element-wise non-linearity (ReLU).

Additionally, we introduce another version of ProgNet, which we call "ProgNet2" where the lateral connections from previous column layers are initialized to be the same as the feedforward connections between the corresponding column layers ($U^{l,c_{12}} = W^{l,c_1}$).
A drawback of these approaches is the growth in the number of parameters with the number of tasks, as we detail below.

Elastic Weight Consolidation (EWC). EWC [29] averts forgetting of old tasks by constraining learning on the weights important for those tasks (or contexts). The importance

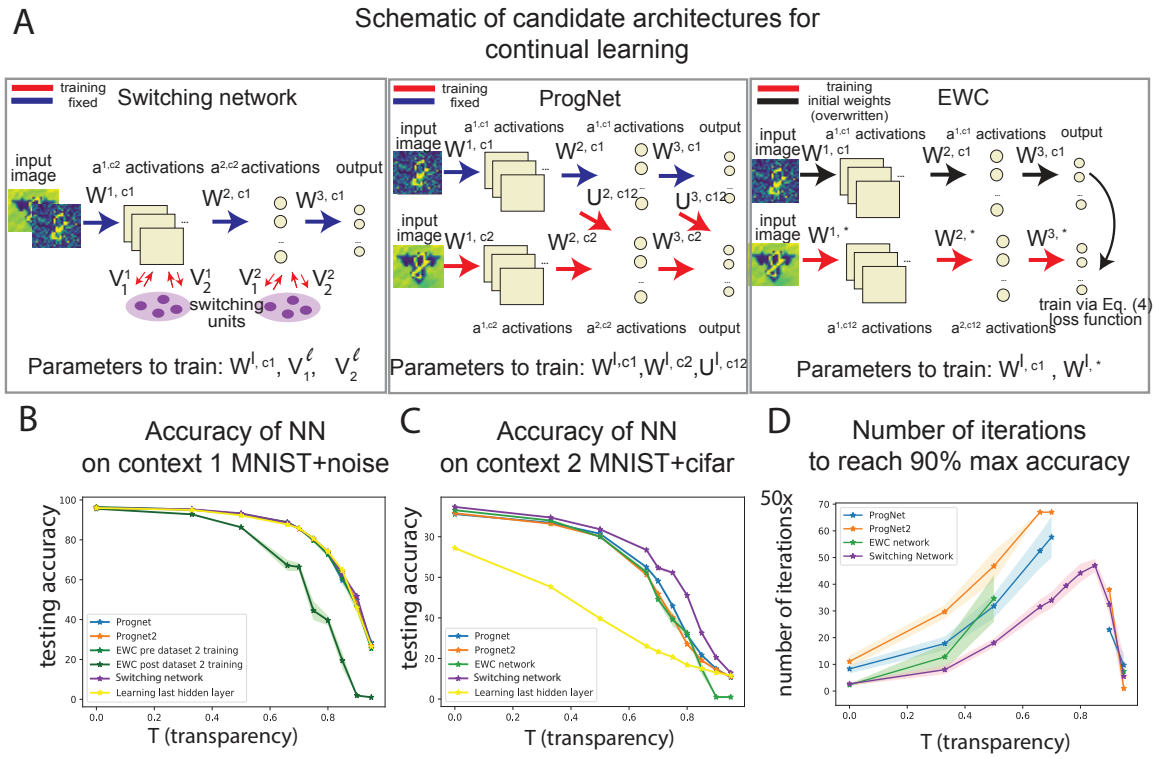


Figure 3.5: The bottleneck-switching network compares favorably to other continual learning solutions. A) Schematic of network architectures: bottleneck-switching network, ProgNet, and EWC. B) Testing accuracy for context 1 (MNIST+noise) for the five NNs. Only the EWC (dark green) on context 1 after training on context 2 (MNIST+cifar) shows a poorer performance because it cannot overcome catastrophic forgetting. C). Testing accuracies on context 2 (MNIST + cifar) for various NNs. We note that the switching network has a slight edge across all transparencies. D). Number of iterations (in steps of 50) for each NN, during training on MNIST+cifar, to reach 90% of the maximum peak accuracy over all the methods. Several data points are not shown because the respective network does not reach 90% of this peak accuracy.

of parameters for a particular task (context) is quantified by the diagonal of the Fisher information matrix. The important weights stay close to their old values, keeping the parameters in a region of low error for context 1, centered around W^{c_1} (the weights for context 1) while learning context 2. The crux of EWC lies in the loss function used for training, which contains an additional regularization term keeping weights close to their old values:

$$L(W^*) = L_2(W^*) + \lambda/2 \sum_i F_i (W_i^* - W_i^{c_1})^2 \tag{3.4}$$

where L is the loss function that prevents catastrophic forgetting (CF), L_2 is the loss functions for context 2 without considering CF, λ sets how important the old task is compared with the new one, i labels each parameter, F_i is the i-th entry of the Fisher information matrix diagonal corresponding to parameter W_i, $W_i^{c_1}$ is the solution found for the i-th parameter when learning context 1, W_i^* is the solution for the i-th parameter found by optimizing Eq. 3.7 as in [29].

Architecture comparison. Our network architecture, the switching network, is most similar to ProgNet, an architecture that similarly adjoins units with each context learned. A few important differences between these architectures stand out. First, the operation we implement in the switching network constrains it to use the input representation pre-nonlinearity at that layer and transform it via the addition of a low-rank modification of this representation (Eq. 3.2). Second, our findings with respect to the small number of switching units required suggest that there are fewer weights to learn, i.e. fewer parameters to learn for $V_1^{l_1}, V_2^{l_1}, V_1^{l_2}, V_2^{l_2}$ ($V_{1,2}^{l_{1,2}}$ for short), than for $W^{l,c_2}, U^{l,c_{12}}$, which can imply fewer training iterations required to train our switching network. An explicit parameter count is presented below. The EWC, in contrast does not add any extra units to the network, instead relying on a clever regularization strategy as shown in (Eq. 3.7). EWC has an intermediate number of parameters to learn, provided there are indeed a very few number of switching units to be added for classification on context 2 (i.e. fewer weights to learn for $V_{1,2}^{l_1}$ than for $W^{l,*}$).

Performance comparison. We compare the performance of the bottleneck-switching

network with the performance of ProgNet and EWC. All networks have the same architecture in common as the main network (Figure 4.3B) of our switching network. The networks we compare against are: the main network implementing EWC during learning ("EWC network"); a ProgNet with two columns like the main network and lateral connections ("ProgNet"); a network like ProgNet, save for how lateral connections are initialized ("ProgNet2"); a main network with two separate last hidden layers for each context.

We first apply training as before, by first training/testing the basic NN on context 1, the *matched condition* (Figure 3.5B). For EWC, we train on this network using the cross-entropy loss function L_1 that does not include the regularization term. When networks are trained on the *matched condition* for MNIST+noise, they perform equally well since initially all of them use the same architecture, the basic NN. At the next step, we train on context 2 by activating the switching units (red line for the bottleneck-switching network); implementing a new column and learning weights from the column trained on context 1 (blue, orange lines for the ProgNets); learning using the basic NN as before using the loss function L_2 as shown in Eq. (3.7) (green line for EWC); training separate connections from the last hidden layer (yellow line). We show the corresponding accuracies on context 2 in Figure 3.5C.

The switching network and both ProgNets avoid catastrophic forgetting by design, but that is not the case for the EWC network. Varying λ, the trade-off hyperparameter that balances how well the network performs on context 2 versus how close the weights for context 1 and 2 are, we were unable to find a regime for EWC where both accuracies for the *test CF condition* and the *re-matched condition* were high. We conclude that, at least for the specific architecture and contexts studied here, EWC cannot perform sequential context switching, as Figure 3.5B shows catastrophic forgetting for EWC (dark green line). A possible explanation for this problem is the high complexity of the task compared to the complexity of the architecture used, which leads to a large fraction of the weights being essential for the first context, leaving few unimportant weights to learn the second context.

When testing on context 2 (e.g., MNIST+cifar), we find that the switching network per-

forms slightly better than the ProgNets across all transparencies (Figure 3.5C). This is unexpected, as Prognets could achieve *matched condition* performance by training the second column on context 2 and setting all lateral connections to 0. However, the switching network evaluated in Figure 3.5C surpasses *matched condition* performance. It is possible that the ProgNets are over-parametrized and suffer from overfitting, or that training settles into a local minimum. The switching network also performs better compared to a network where only the last layer has been trained on context 2, while the previous layer weights are constant, set to the weights for an NN trained on context 1 (yellow line, Figure 3.5C).

Furthermore, switching networks are learning faster than the ProgNets on transparencies ≤ 0.7 (Figure 3.5D), when considering the number of iterations to reach 90% of the peak accuracy for that transparency, maximized between the switching network, ProgNet, ProgNet2, and EWC. Several data points are not shown for ProgNet, ProgNet2, and EWC because the respective network does not reach 90% of this peak accuracy. Alternatively, the switching network is the fastest on average for all $T \leq 0.66$, if we consider the number of iterations to reach 90% converging accuracy for each method independently (Figure 3.14D-E). The comparatively rapid increase to peak accuracy could be explained because the switching network has fewer weights to learn than the ProgNet, in addition to taking advantage of the features learned from context 1. For learning two tasks, we are using 10 switching units for each layer (sw.u/u.s., with 3×3 kernels for the convolutional layers), and so we add an additional $10 \times 3 \times 3 \times 2 + 10 \times 2 = 200$ parameters to learn, compared to $O(100,000)$ for ProgNets. This over-parametrization might be the reason that for higher transparencies T these networks never reach the accuracy levels of the switching network, as the learning procedure could be stuck in a local minima. As the number of contexts (tasks) increases, the number of parameters for ProgNet remains much higher than for the switching network, even if we add switching units for every new context (task) (in some cases, adding new switching units is not necessary – see Sec. 3.8, Bottleneck-switching network for contexts of different statistics – however a careful analysis of the generality of the switching units is beyond the scope of this study). In conclusion, for most transparencies, our switching network shows enhanced performance at sequential context switching in terms of combined

accuracy and learning speed (Figure 3.5, Figure 3.14).

3.4.6 Analysis of a switching network mechanism for sequential context switching

Having established the bottleneck-switching network as a suitable architecture for sequential context switching, we ask what mechanisms underlie its performance. We first observe that switching units in the first layer have a predominantly negative effect – that is, reducing the activations of the main network, as seen from the histograms of contributions for switching to MNIST+noise and MNIST+cifar (Figure 3.6A). These predominantly negative weight contributions from switching units were not built into the network but emerged during learning of the second context.

We next make an allied observation about network representations. The negative contributions from switching units favor a sparsification of activations (after applying the ReLU non-linearity). Specifically, the activations after switching are much more sparse than those without switching (Figure 3.6B), and we hypothesize that, at least for lower transparencies, this allows the network to highlight features useful for digit classification while inhibiting redundant features from the background (Figure 3.6C, Figure 3.16B). The validity of this hypothesis for lower transparencies is reinforced by evidence that the switching units inhibit a larger percentage of the background than of the digit (Figure 3.6D, Figure 3.16C), indicating that the background features are suppressed preferentially.

Furthermore, when we shift MNIST digits several pixels to the right without changing the background, we find an analogous result. Within the basic NN with two hidden layers, we train weights to the last layers (second hidden layer and output layer), while keeping weights to the first hidden layer and the corresponding weights to and from the switching units constant. We find that an analogous sparsification of digits appears in the first hidden layer (Suppl Figure 10A-C), driven by the switching units that inhibit the background despite the shift in the position of the digits. Accuracy in this input regime without training the first hidden layer and the switching unit connectivities is high (Figure 3.17D).

We next test alternative (simpler) mechanisms that could enable switching between contexts. For instance, a sparsification of the activities by using L1 regularization during training may be sufficient for sequential context switching. Using L1 regularization, the testing accuracy for context 1 is close to the *matched condition* accuracy (Figure 3.18A). We hypothesize that L1 regularization may enable background inhibition, analogous to the behavior of the switching units. However, testing on context 2, we see that context 2 has substantially decreased performance compared to the *re-matched condition* using the bottleneck-switching network (Figure 3.18B). This conclusion still holds as we vary the hyperparameter λ that controls how much the L1 norm is weighed. Alternatively, after training on both context 1 (e.g., MNIST+noise) and on context 2 (MNIST+cifar) using L1 regularization, we may test context 1 for the catastrophic forgetting phenomenon (*test CF condition*). As shown in Figure 3.18C, this strategy is not effective in preventing catastrophic forgetting, given lower accuracies for the *test CF condition* (green line). Additionally, accuracies for the *re-matched condition* testing MNIST+cifar after L1 regularization (green line, Figure 3.18D) are also lower than accuracies for the *re-matched condition* using the bottleneck-switching network.

We further examine the switching unit contributions from the simpler, feedforward switching units (Sec. 3.3). Interestingly, we find that the distribution of switching unit contributions is similarly left skewed, with weights on average inhibitory (Figure 3.18E). However, upon closer examination, we find that the first hidden layer activations for the simple network from Section 3.3 are not sparsified, with the background inhibited similarly as the bottleneck-switching network that has recurrent switching units (Figure 3.18F). This reinforces our finding that only by using appropriately, recurrently connected switching units, can switching networks be capable of sequential context switching by inhibiting the redundant features from the background.

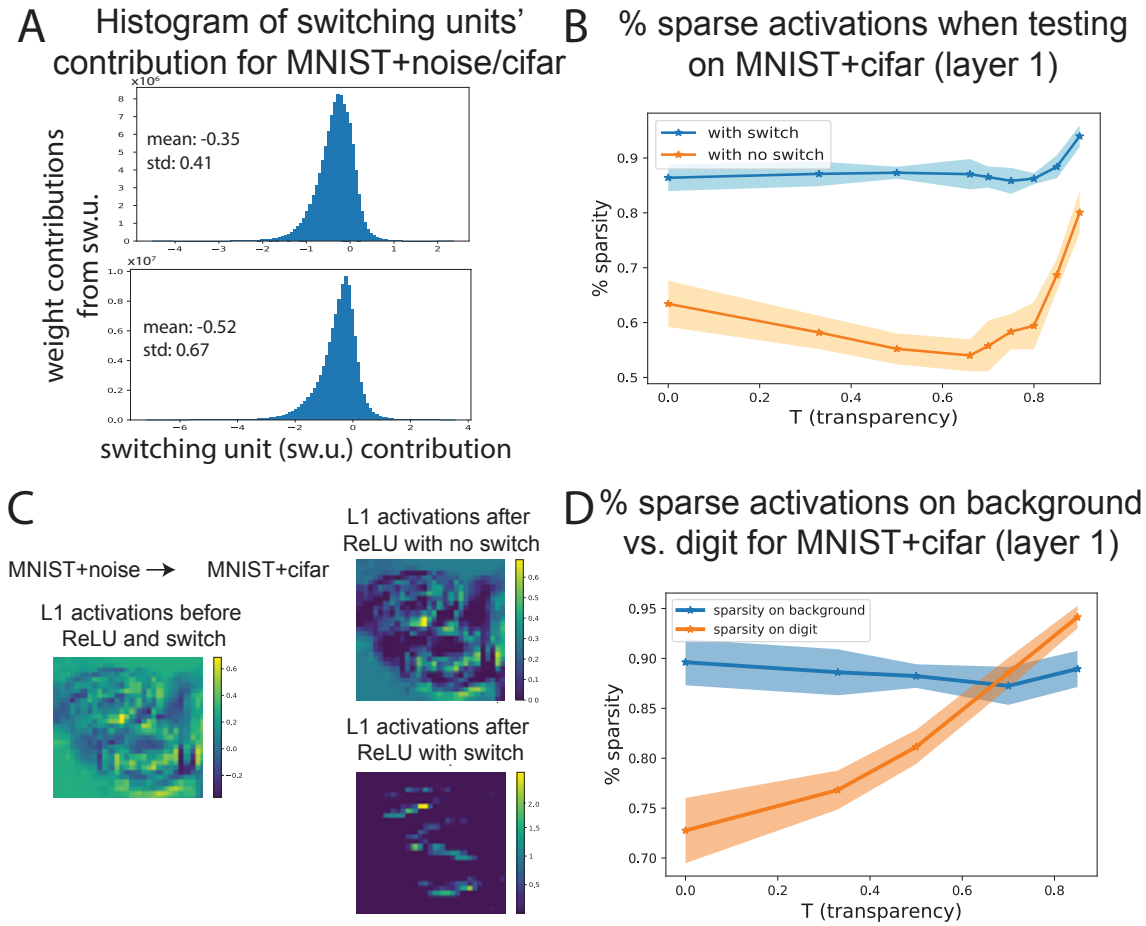


Figure 3.6: The switching mechanism leads to sparser activation patterns, specifically an inhibition of redundant background features. A) Histograms of switching unit contributions to the main network (for switching to MNIST+noise, up; for switching to MNIST+cifar, down). B) Percentage layer 1 sparse activations with or without adding the switching contributions (blue vs orange line), and after applying ReLU. C) Examples of (convolutional) layer 1 activation patterns with or without the switching contribution. D) Percentage layer 1 sparsity (zero activations) within the background (blue) and within the digit (orange) when switching to MNIST+cifar after adding the switching contributions and the ReLU non-linearity.

3.4.7 Bottleneck-switching network for digit class and background pairs not used in training

As described above, switching unit contributions inhibit the background whenever digits set in context 2 are shown to the switching network, boosting the accuracy of the network in the new context. Can switching boost accuracy when the weights to and from the switching units have not been trained on specific digit-background pairs?

To investigate this problem, we choose a random pair of digits (e.g., 8,9) that will not be part of the training of the switching unit weights, then train the switching network in the manner shown in Figure 3.7A. First, we train the main network (with the switching units OFF) in context 1 (e.g., MNIST+noise), using all digits $0-9$. Then, we train the switching network (with switching units ON) on the second context (e.g., MNIST+cifar), using the digits not included in the pre-selected pair (e.g., 0-7). Finally, we employ binary classification on the chosen pair of digits set on context 2 and compare the performance of the switching network with the switching units turned ON, with the performance when switching units are OFF (Figure 3.7B, Figure 3.19). Presumably, the switching unit contributions will inhibit the background, but is training on digit identity necessary when learning weights to and from the switching units?

We find that, on average, switching units' contribution increases accuracy even for digit-background pairs never-before seen by the switching network, i.e. digit-background pairs for which the weights to and from the switching units have not been trained (Figure 3.7B, Figure 3.19). This is shown for select pairs of digits (Figure 3.19), and when averaging over ten randomly chosen pairs of digits (Figure 3.7B). We conclude that the bottleneck-switching network is capable of generalizing to digit-background pairs it has not trained on, given that the main network and the switching units have been separately trained on the digits and the background, respectively. This is a compelling generalization property of the switching network, showing how classification in different contexts can arise synergistically. More work is needed to infer what the minimal training set for the switching unit weights is that still improves performance for sequential context switching.

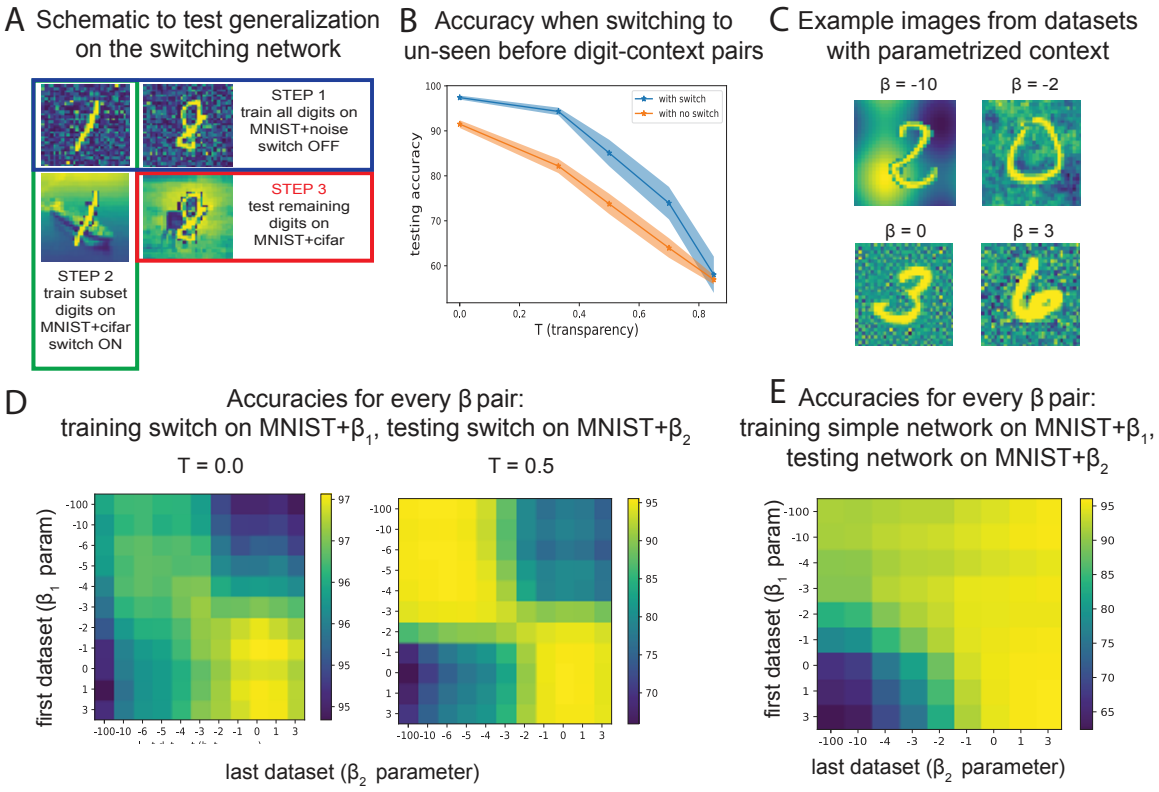


Figure 3.7: The switching network improves accuracy for digit-background pairs that the network has not been trained on, compared to simple feedforward networks trained on one context. A) Schematic of training the switching network to test generalization performance for digit-background pairs that switching units' weights have not been trained on. B) Overall accuracy, averaged over ten randomly chosen digit pairs, of switching network (with switch ON, blue line; with switch OFF, orange line) when tested on digit-context pairs that the switch has not been trained on. C) Example images from MNIST + β parametrized backgrounds. D) Test accuracies for different β pairs and two values of T ($T = 0.0, 0.5$): training the main network on MNIST+cifar, training the weights to and from the switching units on MNIST+β_1, testing the switching network (with switch ON) on MNIST+β_2. D) Test accuracies for different β pairs: train a simple network (the base network) on MNIST+β_1, test the network on MNIST+β_2.

3.4.8 Bottleneck-switching network for contexts of different statistics

More generally, we would like to investigate switching between contexts of any statistics. So far, we have seen how a switching network can be trained to switch between MNIST+noise (a pixel-wise noisy context) and MNIST+cifar (a realistic and non-parametrized context). To study the more general problem, we parametrize contexts using β, the spectral distribution, such that the spectral density is $S(f) = Nf^{\beta}$, where f is the frequency, and N is the normalization coefficient. With this parametrization, $\beta = 0$ denotes a white noise context; $\beta = -1$ denotes a pink noise context; $\beta = -2$ denotes a Brownian noise context; $\beta = 1$ represents a blue noise context; $\beta = 2$ denotes a violet noise context. More details on how these contexts are generated can be found in Supplementary Materials, Sec. 1.2.4.

We super-impose MNIST digits on backgrounds of different statistics to obtain different datasets representing different contexts (Figure 3.7C). We refer to these datasets as "MNIST + β" (or "MNIST + β" - parametrized contexts) for different values of β. Using these datasets, we seek to investigate two main questions. First, given contexts of different statistics, how well can the switching units that are trained on a particular statistics help – without further training – with the classification of a context with different statistics? Second, are some contexts "closer" in some sense to others (with respect to a certain distance or metric), hence enabling the switch to readily generalize to different statistics? If so, is this distance symmetric?[1]

To answer these questions, we first train the bottleneck-switching network using a familiar approach: the main network is trained on MNIST + cifar with the switching units OFF; we then turn the switching units ON, and train on MNIST+noise while keeping the weights of the main network constant. Using this trained network, we can now test its performance on

[1]Viewing the switching unit contribution as essentially a low-rank perturbation to the weights (at least to a linear approximation), an interesting problem is whether there is a metric in the space of contexts, that can determine whether low-rank perturbations to the weights of the main network enable switching contexts with different statistics while maintaining the task. Additionally, if two contextual statistics are "close enough", we hypothesize that the low-rank perturbation is capable of generalizing to enable classification in a similar context.

the MNIST+β-parametrized contexts. With the switching units ON, we assess the accuracy of the network to identify which dataset of parametrized contexts can utilize the switching units best to achieve digit classification. We find there is maximum accuracy when $\beta = 0$, with generally high accuracies for $\beta \geq 0$ (Figure 3.20A). This occurs because MNIST+noise corresponds to an MNIST+β-parametrized context with $\beta = 0$. While the peak at $\beta = 0$ is explainable, the high performance for $\beta > 0$ demonstrates an anti-symmetry between the capacity of the switching network to generalize well to particular contextual statistics, while not to others. We conclude that switching networks whose weights have been trained on MNIST+noise $\equiv$ MNIST + $\beta = 0$ (images with pixel-wise noisy contexts) generalize well to higher frequency contexts ($\beta > 0$).

The same results hold when training the main network on MNIST+cifar, training the switching units on MNIST+β, and then testing the switching network on MNIST+noise. As before, we find there is maximum accuracy when $\beta = 0$. (Figure 3.20B). Accuracies for contexts with $\beta \geq 0$ are high, with the accuracy plateauing for higher frequency contexts.

Finally, we would like to gain insight into the issue of distance in the space of contexts with different statistics, where contexts that exhibit the closest statistics may benefit most from the same switching units (or from the same low-rank perturbation to the weights). To investigate this problem, we use the following training procedure: we train the main network on MNIST+cifar (with switching units OFF); we then turn the switching units ON and train the switching weights on a context with some fixed β, keeping weights in the main network fixed; lastly, we test different MNIST+β-parametrized contexts by varying β. We can find generalization accuracies for different pairs of β corresponding to pairs of contexts used for training and testing the switch, respectively. The matrix obtained (Figure 3.7D) shows how efficient the switch is across different contexts (for $T = 0.0,\ 0.5$) and can be interpreted as the inverse of the distance between two contexts: the higher the accuracy is, the better we can classify utilizing the same set of switching weights, suggesting that these contexts are "close" in some sense.

We find a bimodal behavior of the switching network: when contexts corresponding to

(β_1, β_2)-pairs are trained on the network, we obtain high testing accuracies when both β_1, β_2 have higher or lower frequencies, i.e. $\beta_1, \beta_2 \leq -3$ or $\beta_1, \beta_2 \geq -2$. When $\beta_1 \geq -2$ and $\beta_2 \leq -3$ (or vice versa), the generalization accuracy is poor. This is particularly evident for higher transparencies (Figure 3.7D, T = 0.5).

Different results hold when training a feedforward network without switching units in the following way: we train the basic NN we have so far employed on an MNIST+ β-parametrized context, and then test on a different MNIST+β-parametrized context. We found distinct conditions for satisfactory generalization: it is best, whichever context the network was trained on, to test on contexts with higher-frequency statistics. It is also satisfactory to first train in low-frequency contexts. The instance most prone to (generalization) error is for training on higher frequency contexts (positive β) and testing on low-frequency contexts (negative β), as seen in Figure 3.7E.

3.5 Conclusion

We study a biologically inspired switching network that is capable of domain adaptation between backgrounds (contexts) with different statistical properties, while the basic task structure is maintained. We construct and share a parameterized dataset to test this adaptation, by overlaying MNIST digits with different transparencies on noisy or CIFAR-10 backgrounds. The switching network can leverage features learned in one context and use them for digit classification in a second context, without forgetting classification in the first context. We refer to this network as the bottleneck-switching network because it has only relatively few switching units that are recurrently connected to the main layers. This structure compresses network activations through a bottleneck, then relays this compressed information back to the main network.

While we do not claim to have identified a precise mechanism for how the switching allows the network to perform well under both contexts with so few neurons, a set of analyses allows us to speculate. We observe a sparsification of the NN representation with the switch ON (Figure 3.6B,C). We believe this allows the network to select the features relevant for

the task; these features are different from the features in the new background, which serve as distractors. We show that this can be achieved with relatively few switching neurons. We also show that a low-rank change in the connectivity matrix can result in a similar performance (Figure 3.4H, Figure 3.15). We connect these findings by noting that a small number of neurons recurrently connected can provide the mechanism for a low-rank matrix change in network connectivity.

We also note similarities and differences to the biology of the VIP circuitry in the visual cortex. We find that the small number of switching units in the first NN layer are inhibitory towards the hidden layer activations of the main network. While this invites direct comparison to the underlying biology, as VIP neurons are inhibitory, our framework omits the disinhibitory motif, and specific inhibitory/excitatory neuron populations. Specifically, while the switching units roughly reproduce the behavior of the VIP neurons by turning ON/OFF contingent on the context, and by recurrently interacting with other network units to modulate network activity, these added units are not constrained to have any sign in particular. Moreover, the inhibitory dominance is not apparent in the second layer, although we note that the switching units at the first convolutional layer are the main drivers of high performance (Figure 3.13D-E). In sum, we do not claim to have implemented in this paper a precise model of the VIP circuitry, but rather to have used a bio-inspired circuit motif with VIP-like, bottlenecked, and recurrent connectivity.

Our findings have several limitations. First, the dataset we used is a simple benchmark with few categories (MNIST digits). As we describe above, we opted for this dataset to remove other potential confounds, but future work should concentrate on more complex datasets. Second, while we found high performance with relatively low numbers of switching units, especially for the VGG network, we do not currently have theoretical guarantees on the number of switching units sufficient for sequential context switching. Moreover, beyond our main studies with a pair of contexts (noise and CIFAR-10), we have studied the transition between multiple contexts using only a simple continuously parametrized noise

setting (the MNIST+β contexts). More work is needed to address the problem of continuous contexts in general, although the suppression of the contextual features shown above opens the possibility that multiple context switches can be incorporated. Finally, we showed that a switching network can readily include a downstream module to perform binary classification of contexts using supervised learning, with *a priori* training to different backgrounds (Figure 3.4I, Figure 3.21). The binary output is then used by the VIP neurons to turn them ON and OFF in a switch-like manner, automating the sequential context switching task. However, a more general framework would enable the module to detect a novel context in an unsupervised way, and this is another important avenue to explore in future work.

In principle, bottleneck-switching networks employ a general circuit motif that implements a context-dependent computation, therefore switching units could be integrated into any neural network in order to address the problem of adaptation to context. A possible application area of high social importance is in addressing biases of background, in which networks can learn unwanted correlations between context and objects of interest [15].

Finally, we note that this is one of few studies to address generalization across context in a bio-inspired architecture while focusing specifically on background variation and removing other confounds [4]. In future years, we look forward to progress in understanding the context-dependent computations that enable the extraordinary versatility of biological agents to adapt and switch environments so effortlessly. Deciphering the general principles behind contextual generalization will enable our current networks to develop from being sophisticated "pattern-matching machines" to more intelligent, human-like learners capable of abstracting visual concepts [4].

3.6 Methods

3.6.1 Dataset

To address the problem of image classification with context switching, we created a dataset whose only confounding feature is context. We make this dataset publicly available under the Creative Commons v 4.0 license as a benchmark to test generalization across context

[64]: https://figshare.com/s/e5807249c28ec5bcff94.

The dataset consists of 32×32 sized images of MNIST digits set on either noisy backgrounds, with background pixel intensity being the absolute value of random variables drawn independently from a Gaussian distribution, or MNIST digits set on a more naturalistic background from the CIFAR-10 dataset. We refer to the subset of MNIST digits set on noisy backgrounds as the "MNIST+noise" dataset and to the subset of MNIST digits set on CIFAR-10 backgrounds as "MNIST+ cifar" dataset.

To construct these two sets of images, we first consider the normalized 28×28 pixel images from the MNIST dataset with maximum pixel value of 1, and refer to this normalized set unless otherwise specified. We then identify the pixels in these images that belong to the digit. The background in the MNIST dataset images is uniformly 0 therefore we consider all non-zero pixels to belong to the digit as opposed to the background. The pixels that constitute the digits are retained in a mask image to be used on the background, as described below.

For MNIST+noise, we create 32×32 pixel images where the pixels are the absolute value of Gaussian random variables of mean 0 and standard deviation 1, then normalize appropriately so that the maximum value is 1. The mask described above is super-imposed on this noisy background, after being properly shifted so that the digit pixels represented by the mask are in the middle of the noisy image (given that MNIST images are 28×28 and noisy image backgrounds are 32×32, there will be a 2 pixel shift). While pixels outside the mask are not modified, pixels from the noisy background that correspond to the mask (and hence digit) are linearly combined with MNIST digit pixels.

$$p_i = T \cdot p_i^b + (1 - T) \cdot p_i^{digit} \tag{3.5}$$

where p_i is pixel i from the MNIST+noise image, p_i^b is pixel i from the noisy background and p_i^{digit} is pixel i from the MNIST image. T is a parameter that makes the task more difficult, and we refer to T as digit *transparency*. As we increase the transparency, the back-

ground interferes with the digit more, and the identity of the digit becomes more ambiguous.

For MNIST+cifar, we consider the $3 \times 32 \times 32$ images from CIFAR-10 and average across the color dimension, then normalize as before, so that the maximum pixel intensity is 1. We then proceed analogously as for MNIST+noise to create MNIST+cifar, a dataset of MNIST digits on more naturalistic CIFAR-10 backgrounds. Each MNIST digit is super-imposed on a unique CIFAR-10 background, so that there is a one-to-one correspondence between the image digits and backgrounds (although this is not important). The correspondence is random, ignoring the CIFAR-10 class corresponding to the background image. The number of samples in the CIFAR-10 dataset, for both training and testing, is smaller than in the MNIST dataset, therefore we limit the number of samples in our MNIST+cifar dataset to that of the CIFAR-10. We limit the number of samples in the MNIST+noise data in a similar way so that the number of samples in the two datasets is identical.

The goal of these two datasets is to perform image classification and correctly identify the MNIST digit despite the different interfering backgrounds.

3.6.2 Experiments

We carried out all experiments using the Torch neural network framework on two machines as described below in the *Computing resources* subsection. We used the dataset with 32×32 MNIST digit images set on different backgrounds described above, except for the experiments on the deeper network VGG-16 [59], when we re-sized these images to be $224 \times 224 \times 3$ by first using the resize method with inter-area interpolation from the cv2 python library, and then concatenating two other 224×224 identically 0 matrices.

We split both the MNIST+noise and the MNIST+cifar data into 80% training and 20% testing samples. When using a validation dataset (either for determining the number of switching units to be chosen or to search for the hyperparameter λ in the case of the EWC algorithm), the split is 64% training, 16% validation, and 20% testing data from the total samples. This partition of training and testing data does not correspond to the classic

MNIST data training and testing split.
Throughout all experiments we considered both the case when the order of training was MNIST+noise, then MNIST+cifar and vice versa.

Network architectures

We tested both simple (toy) neural network (NN) examples and a more complex, deeper architecture, the VGG-16 [59]. The *basic* network architecture used throughout the paper is shown in Figure 4.3B. Aside from the input and output layers, the network has 2 hidden layers, where the first is a $16 \times 28 \times 28$ convolutional layer and the second layer is a 64 unit fully connected hidden layer. The first convolution has a kernel size of 5, stride 1, 16 filters, no padding, and is followed by a ReLU non-linearity. The second operation is linear, between the flattened 12544 unit convolutional layer and the 64 unit hidden layer. After another ReLU non-linearity, the final operation is a 64×10 linear operation. All operations have biases in addition to weights.

There are two more networks we use to test the 32×32 MNIST+noise and MNIST+cifar datasets (Figure 3.13). The second network we employ is a 3 hidden layer network with two convolutional layers and a fully connected layer. The network has the same architecture as before (kernel size 5, stride 1, 16 filters for the first convolution; 64 units for the fully connected layer), except for the second convolution which now employs 8 filters, with the same kernel size and stride. As before, these layers apply a ReLU non-linearity.
Another architecture is a 4-layer network with 2 convolutional hidden layers and 2 fully connected hidden layers. This network has a $16 \times 28 \times 28$ convolutional first layer, a $32 \times 14 \times 14$ convolutional second layer (using a 5×5 kernel and stride 1 as before), an 128 unit fully connected layer, and a 64 unit fully connected layer. In addition to the ReLU non-linearity, the network uses max-pooling after every convolution and dropout with $p = 0.2$ for all except the last hidden layer.
These two additional networks are tested to assess the effectiveness of switching units on various architectures, and at different layers. We find that switching units are most useful

when added at the first convolutional layer, although there is some benefit, albeit smaller, to adding switching units at later hidden layers (Figure 3.13). These findings lead us to apply switching units only at the first convolutional layers of VGG-16.

In the case of VGG-16, we replaced the final layer with two fully connected hidden layers with 256 and 10 units respectively, the latter corresponding to the number of classes in our dataset. Regarding switching units, we added about a tenth of the number of units to the corresponding hidden layer: 6 switching units per unit space (sw.u./u.s.) for the 64 filter hidden layer; 12 sw.u./u.s. for the 128 filter hidden layer; 50 sw.u./u.s. for the 512 filter hidden layer. We only added switching units to six out of the thirteen convolutional layers, and specifically to the first three layers. No switching units were added to change activations of the latter linear layers.

Training the networks

For the basic network (Figure 4.3B) and for the other 3− and 4− hidden layer networks, we used the ADAM optimizer, with variable learning rate that was not a priori fixed, and cross-entropy loss (Table 4.2 below). For the simple network enforcing a contextual output (Figure 3.10A, middle) we used a cross-entropy loss for the output layer units doing digit classification and a mean squared error (MSE) loss for the contextual output. We use the MSE loss for the simple one-layer network that forms the downstream module classifying context, where the binary output indicates whether the background is noisy or CIFAR-10 (see Sec. 3.4, Figure 3.21).

For VGG-16, we use the network pre-trained on ImageNet. During training of the main network, we keep the first eleven (out of thirteen) convolutional weights fixed while varying the other weights in the network. We tried a variety of optimizers with different hyperparameters, with distinct results depending on whether dataset 1 was MNIST+noise or MNIST+cifar. After a thorough search, see Table 4.2 below for a summary of all hyperparameters used during training.

hyperparameters	basic network/ switching network	VGG-16, MNIST+noise	VGG-16, MNIST+cifar
optimizer	ADAM	RMSProp	SGD
learning rate	–	0.0001	0.01
other hyperparameters	–	–	momentum= 0.9 weight decay=$5 \cdot 10^{-4}$
loss function	cross entropy	cross entropy	cross entropy
epochs	5	5	5
batch size	50	32	32

Table 3.1: Hyperparameters and optimizers used for training MNIST+noise and MNIST+cifar on the basic network, switching network, and VGG-16 with switching units.

These hyperparameters were chosen to generate Figure 3.9, Figure 3.12, and the Supplementary Figures.

We mention that for VGG-16 the weights to and from the switching units were learned with RMSProp using a learning rate of 0.0001, momentum 0.9, and weight decay $5 \cdot 10^{-4}$, whether learning MNIST+noise or MNIST+cifar. These were chosen after careful experiments testing different learning methods and hyperparameters.

We attempt to use a validation dataset in order to choose an optimal number of switching units. Predictably, as the number of switching units increases, the testing accuracy increases as well, however the increase is negligible and effectively the accuracy plateaus as seen in Figure 3.11A-C. For a majority of the experiments, when not specifically indicated, we use 10 switching units (per unit space with a kernel size of 3 for convolutional layers) for the bottleneck-switching network, as this adds a limited number of parameters while maintaining high accuracy. We add a fixed number of switching units to the VGG-16 as described above in the *Network architectures* section (about a tenth of the corresponding

hidden layer units).

The bottleneck-switching network may be equipped with a "context detector" – the downstream module classifying context. We trained a simple one layer neural network to classify the backgrounds of images, whether it was noisy of CIFAR-10 backgrounds. The output of this network then determined whether the switch was ON or OFF during testing of a dataset with randomly interleaved MNIST+noise and MNIST+cifar digits (Figure 3.21). We find that having the switching units turn ON or OFF depending on context (*adjusted switch*) gives higher performance than having the switch always OFF (*never switch* or *unmatched condition*) or having the switch always ON (*always switch*). The design of a network that detects new contexts in an unsupervised way is left for future work.

The results of experiments performed on VGG-16 using switching units are shown in Figure 3.4E,F. It is worth highlighting the superior performance of the *re-matched condition* as compared to the *matched* and *unmatched conditions*: the bottleneck-switching network leverages the features learned by VGG-16 on context 1 to increase performance on context 2, with performance surpassing that of a network explicitly trained for context 2.

We run each experiment 10 times with different seeds for the basic network (and also for the 3- and 4- hidden layer networks), but only 5 times for VGG-16 because of the prohibitively long training time for this deep network. Error bars in the plots quantify the uncertainty in the test set and represent the standard deviation of the samples obtained from these experiments. We mention that uncertainty in Figures and Supplementary Figures is only w.r.t. network initialization, as there is no shuffling of the training data for each epoch/experiment; this can cause error bars to be comparatively small.

Training using other continual learning methods

We compared our bottleneck-switching network with two prominent continual learning methods: elastic weight consolidation (EWC, [29]) and progressive networks (ProgNets,

[54]). To implement these methods, we use two Github repositories: [5] for EWC, and [25] for ProgNet. Both repositories are under the MIT license, so we were able to appropriately change the code to suit our sequential context switching problem. Scripts in [25] underwent only minor modifications, whereas scripts in [5] had to be more substantially modified to be operational.

ProgNet. The main idea of ProgNets is that CF is prevented by instantiating a new neural network – "a column" – for each task being solved, while transfer is enabled via lateral connections from features of previously learned columns. The parameters for all previous columns (and tasks/contexts) are held fixed while training the new column so that there is no interference between tasks and hence the ProgNets are immune to CF by design. A schematic of the ProgNet is shown in Figure 3.5A, middle.

A ProgNet starts with a single column: in our case, the main network of 4.3B used for EWC and for the switching network. We can denote the hidden activations of this column as $h_i^{(1)} \in \mathbb{R}^{n_i}$, where n_i is the number of units at layer $i \leq L$ and parameters $\theta^{(1)}$ are trained for convergence. When learning on the second task, the parameters $\theta^{(1)}$ are "frozen" and a new column with parameters $\theta^{(2)}$ is instantiated with random initializations. Layer $h_i^{(2)}$ receives input from both $h_{i-1}^{(2)}$ and $h_{i-1}^{(1)}$ via the lateral connections. We can generalize to K tasks so that the activations at column k can be expressed as:

$$h_i^{(k)} = f(W_i^{(k)} h_{i-1}^{(k)} + \sum_{j<k} U_i^{(k:j)} h_{i-1}^{(j)}), \tag{3.6}$$

where $W_i^{(k)} \in \mathbb{R}^{n_i \times n_{i-1}}$ is the weight matrix of layer i and column k, $U_i^{(k:j)} \in \mathbb{R}^{n_{i-1} \times n_i}$ are the lateral connections from layer $i-1$ of column j to layer i of column k, h_0 is the network input, and f is an element-wise non-linearity.

An advantage of this approach is that ProgNets make no assumption about the relationship between tasks. Hence, columns in ProgNets are free to re-use, modify, or ignore previously learned features from columns via lateral connections. A downside of this approach however

is the growth in the number of parameters with the number of tasks/contexts. Even for our two columns corresponding to MNIST+noise and MNIST+cifar, there are $O(100,000)$ additional parameters added for the lateral connections. As suggested in [54], only a fraction of the new capacity is probably utilized, which indicates that pruning or online compression during learning could alleviate this overparametrization.

For our training we use the ADAM optimizer with variable learning rate and the cross entropy loss. Our ProgNet has two columns, one trained on the MNIST+noise dataset, the other trained on the MNIST+cifar dataset. We probe sequential context switching in either order, as we did for EWC and the bottleneck-switching network. A version of our network which we call "ProgNet2" initializes the lateral connections between $h_{i-1}^{(1)}$ and $h_i^{(2)}$ before training with the feedforward connections between $h_{i-1}^{(1)}$ and $h_i^{(1)}$. Otherwise, these weights are randomly initialized for the standard ProgNet.

EWC. EWC is a continual learning method that protects consolidated knowledge through regularization. It takes inspiration from synaptic consolidation in the neocortex, where a proportion of synapses becomes less plastic and hence more stable over longer timescales in order to durably encode information. In EWC, each weight is pulled back toward its old values by an amount proportional to its importance in previously learned tasks, which slows down learning on these important weights. Effectively, EWC constrains important parameters to stay close to their old values and this constraint is implemented as a quadratic penalty (Equation (3.7)).
The posterior $p(\theta|D_A)$ is the probability of parameters θ given task data A (D_A) and contains information about which parameters are important for the task. We can approximate this posterior as a Gaussian distribution with mean given by the old values of θ on task A, θ_A^*, and a a diagonal precision given by the diagonal of the Fisher information matrix F.

Given that we consider the diagonal of the Fisher information matrix to be a good estimate of how important each parameter is, we can express the loss function at the second task B (after first task A) as:

$$L(\theta) = L_B(\theta) + \lambda/2 \sum_i \cdot F_i(\theta_i - \theta^*_{A,i})^2 \tag{3.7}$$

where L is the loss function that prevents catastrophic forgetting (CF), L_B is a loss function on task B without considering CF, λ sets how important the old task is compared with the new one, i labels each parameter, F_i is the i-th entry of the Fisher information matrix diagonal corresponding to parameter θ_i, and $\theta^*_{A,i}$ is the solution found for the i-th parameter when learning task A.

We applied EWC on the main network (Figure 4.3B) used for the bottleneck-switching network. We used cross entropy loss, the ADAM optimizer with variable learning rate, and fixed hyperparameters like the Fisher estimation sample size (the number of samples used to compute the diagonal of the Fisher information matrix) to 1024. Using the validation set, we do a systematic search over λ, from values of 100 up to 10^{10} in steps of $10^i/2$, $i = 2, ..., 10$. We find that for high values of λ ($\approx 10^8$), CF is indeed averted, but at the expense of lower accuracy for context 2. For lower values of λ, the accuracy on context 2 improves and is competitive with the bottleneck-switching network accuracy, but at the expense of forgetting context 1. Allowing different values of λ for different transparencies did not overcome this issue. We conclude that, barring attempts to change the optimization hyperparameters, loss function, or Fisher estimation sample size, EWC is unsuccessful at sequential context switching.

Generating datasets of images with parametrized background

To generate an MNIST+β-parametrized backgrounds of varying transparencies T, we superimpose MNIST digits weighed by $1 - T$ onto backgrounds weighed by T. To generate the β-parametrized backgrounds given a fixed β, we generate a grid of frequencies in x and y directions:

$$S_f = (u^2 + v^2)^{\beta/2}, \tag{3.8}$$

where u is the set of frequencies along the first dimension, with positive frequencies in

the first quadrant; v is the set of frequencies along the second dimension. We also generate a grid of random phase shifts, ϕ.
Next, we can take the inverse fast Fourier transform of $S_f^{1/2} \cdot (\cos(2\pi\phi) + i\sin(2\pi\phi))$.
Finally, we take the real part of the inverse Fourier transform to obtain the β-parametrized background.

This procedure generates $1/f$ spatial noise, with a normal error distribution. $1/f$ noise is scale-invariant, there is no spatial scale for which the variance plateaus out, so the process is non-stationary.

β defines the spectral distribution. The spectral density is $S(f) = Nf^{\beta}$, where f is the frequency and N is normalization coefficient. Given this formulation, $\beta = 0$ denotes random white noise, $\beta = -1$ denotes pink noise, $\beta = -2$ denotes Brownian noise. The fractal dimension is related to β by $D = (6 + \beta)/2$. Note that the spatial pattern is periodic.

The errors are normally distributed because of the central limit theorem. The phases of each frequency component are randomly assigned with a uniform distribution from 0 to 2π. By summing up the frequency components the error distribution approaches a normal distribution.

The method is briefly described in [33]. The code was implemented in MATLAB, was written by Jon Yearsley, and retrieved from the MATLAB Central File Exchange [24].

3.6.3 Computing resources

For our experiments, we used two machines. One had two NVIDIA TITAN X (Pascal) 12G GPUs and a 20-core Intel(R) Xeon(R) CPU E5-2698 v4 with 512 GB RAM. The operating system used was Linux Ubuntu 18.04 LTS.
The second machine had four GF RTX 2080 Ti GPUs and a 40-core Intel Gold 6230 CPU with 512 GB RAM.

Our (toy) switching network using the basic NN takes about 4 hours to train for a particular combination of switching units (e.g. 10 sw.u.(/u.s.)), for $T \in \{0.0, 0.33, 0.5, 0.66, 0.7, 0.75, 0.8, 0.85, 0.9\}$, and for 10 different seeds. Plotting Suppl. Figure 3.11C takes about 3 days. For VGG-16 it takes about 10 hours to train (for one seed), for a particular number of switching units chosen and for $T \in \{0.0, 0.33, 0.5, 0.7, 0.85\}$.

3.6.4 Code and datasets

A repository with the code to generate the figures in the main paper can be found at
https://github.com/dvoina13/switching_network_ANN
The MNIST+noise/cifar datasets can be found in [64].

3.7 *Supplementary Figures*

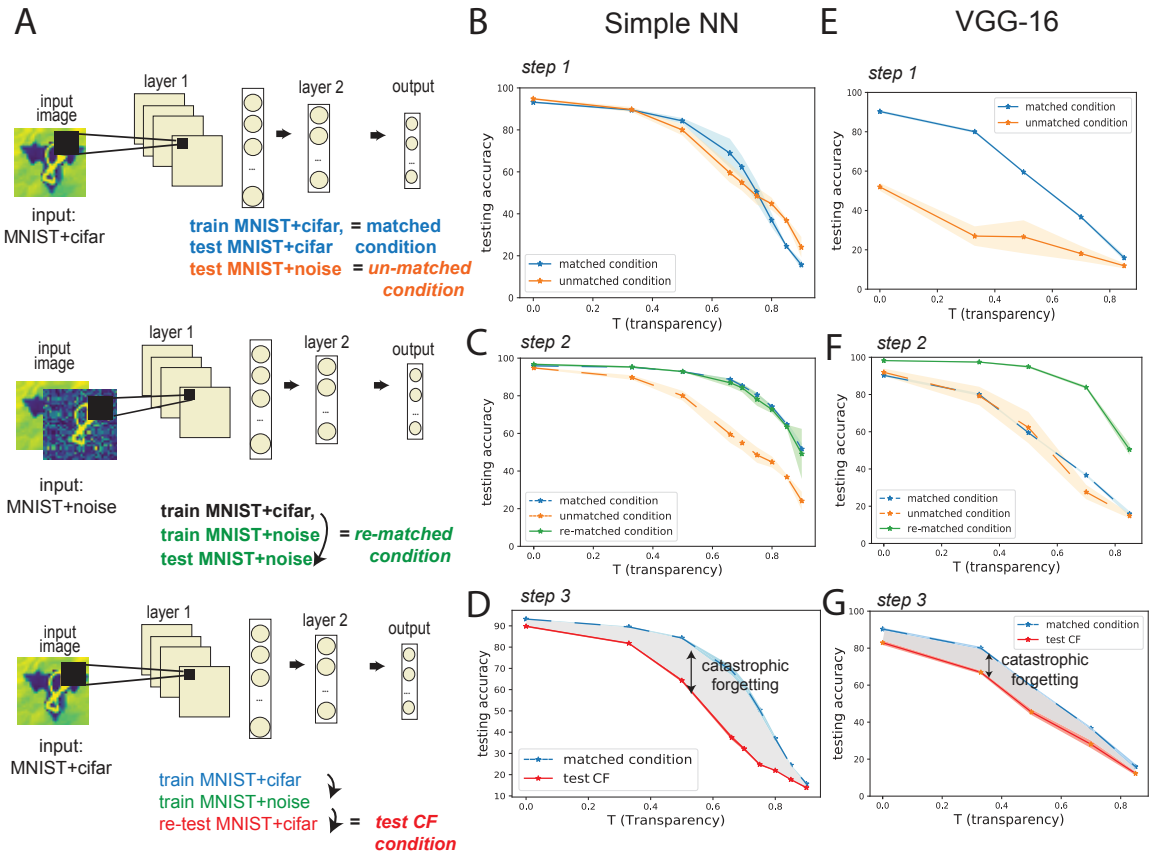


Figure 3.8: A) Schematic of training and testing for sequential context switching to test catastrophic forgetting. B) Average performance on the *matched condition* for MNIST+cifar (blue) and on the *unmatched condition* for MNIST+noise (orange) as transparency T increases. C) The MNIST+noise dataset is less accurately classified by a network trained on MNIST+cifar (*unmatched condition*, orange line) than one trained on MNIST+noise (*matched condition*, blue line). However, once we re-train the MNIST+cifar-trained NN on MNIST+noise (Step 2), the accuracy approaches that for the *matched condition* (*re-matched condition*, green line). D) The performance when re-testing MNIST+cifar on a network that was first trained on MNIST+cifar, then on MNIST+noise, is reduced (*test CF*, red line) compared to the *matched condition*, therefore catastrophic forgetting occurs. E)-G) same as B)-D) using the VGG-16 network instead of the basic network.

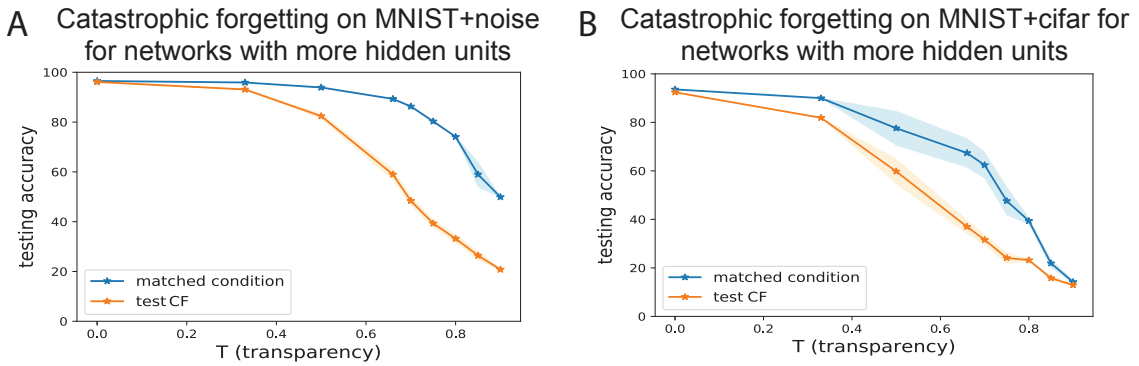


Figure 3.9: Increasing the width of the network does not alleviate forgetting. To probe CF on wider networks, we increased twofold the number of filters or hidden units in the convolutional and linear layers, respectively. A). Accuracy on the MNIST+noise dataset for the *test CF condition*, after training on MNIST+noise, then on MNIST+cifar is lower particularly for higher T than for the *matched condition*. This shows CF occurs. B). Same as in A), for the MNIST+cifar dataset.

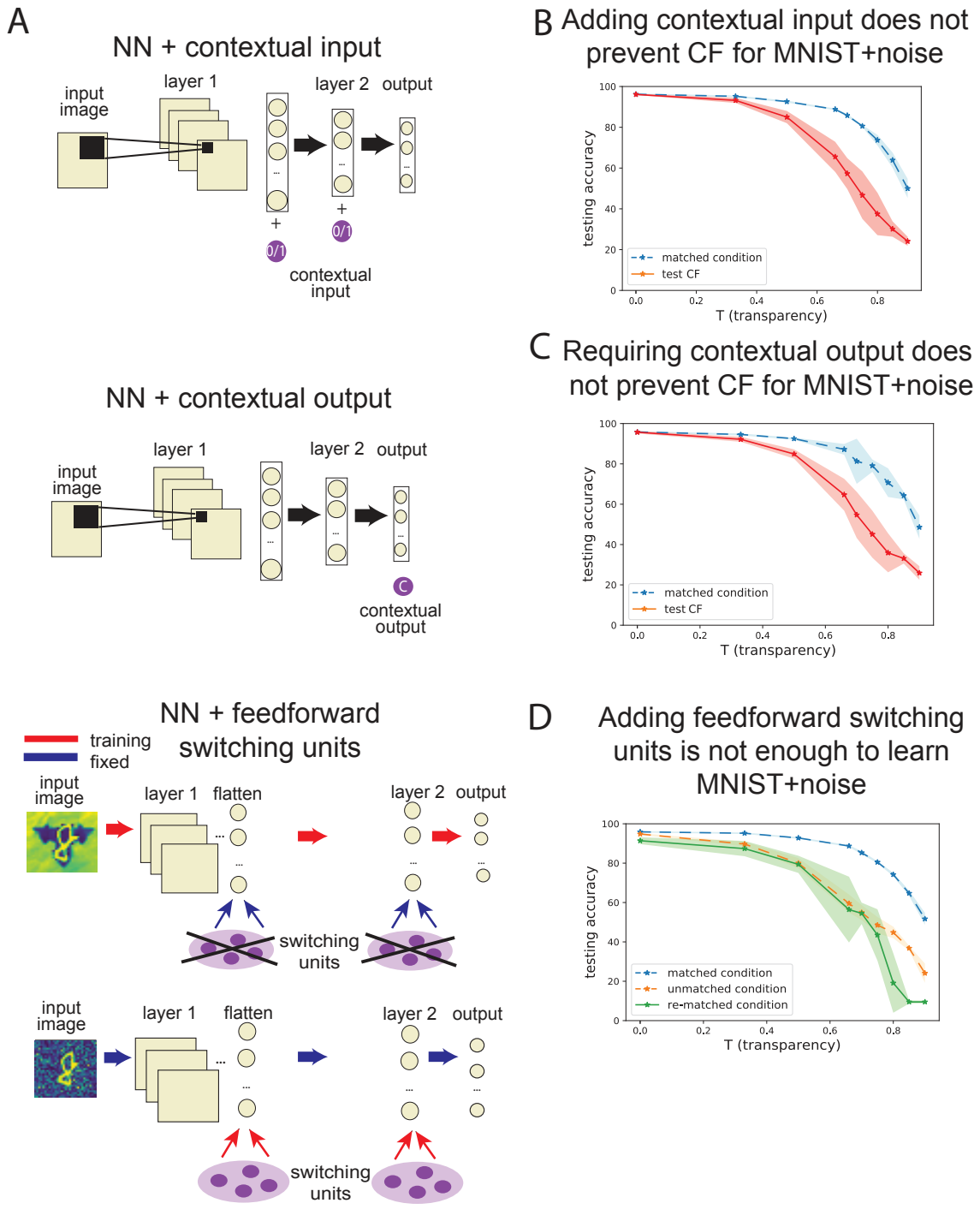


Figure 3.10: Simple network architectures that fail to perform sequential context switching. Schematics for A) the neural network with contextual input; the neural network with contextual output; the neural network with feedforward switching units. Testing CF for MNIST+noise: D) adding contextual input information; E) requiring contextual output. F) Adding switching units with feedforward connections to the hidden layers.

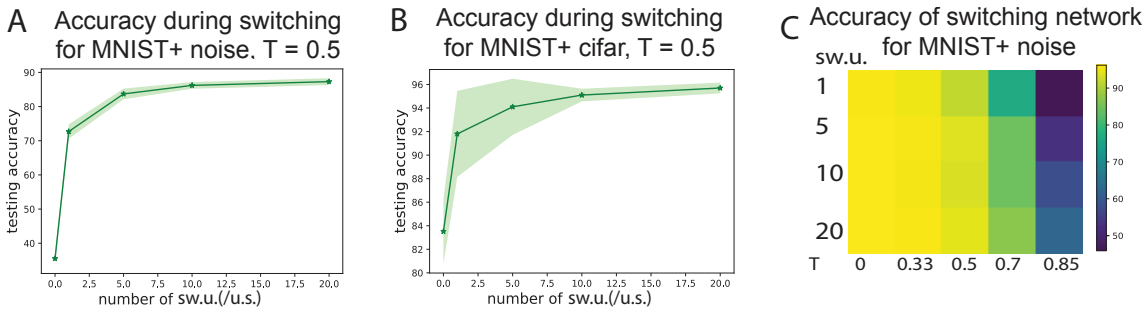


Figure 3.11: A). Accuracy for the bottleneck-switching network on MNIST+noise with $T = 0.5$, while varying the number of switching units (per unit space – sw.u/u.s. – for convolutional layers). B). Accuracy for the bottleneck-switching network on MNIST+cifar with $T = 0.5$, while varying the number of sw.u.(/u.s.). C). Heat map of accuracy on MNIST+noise for the bottleneck-switching network across number of switching units and transparency T.

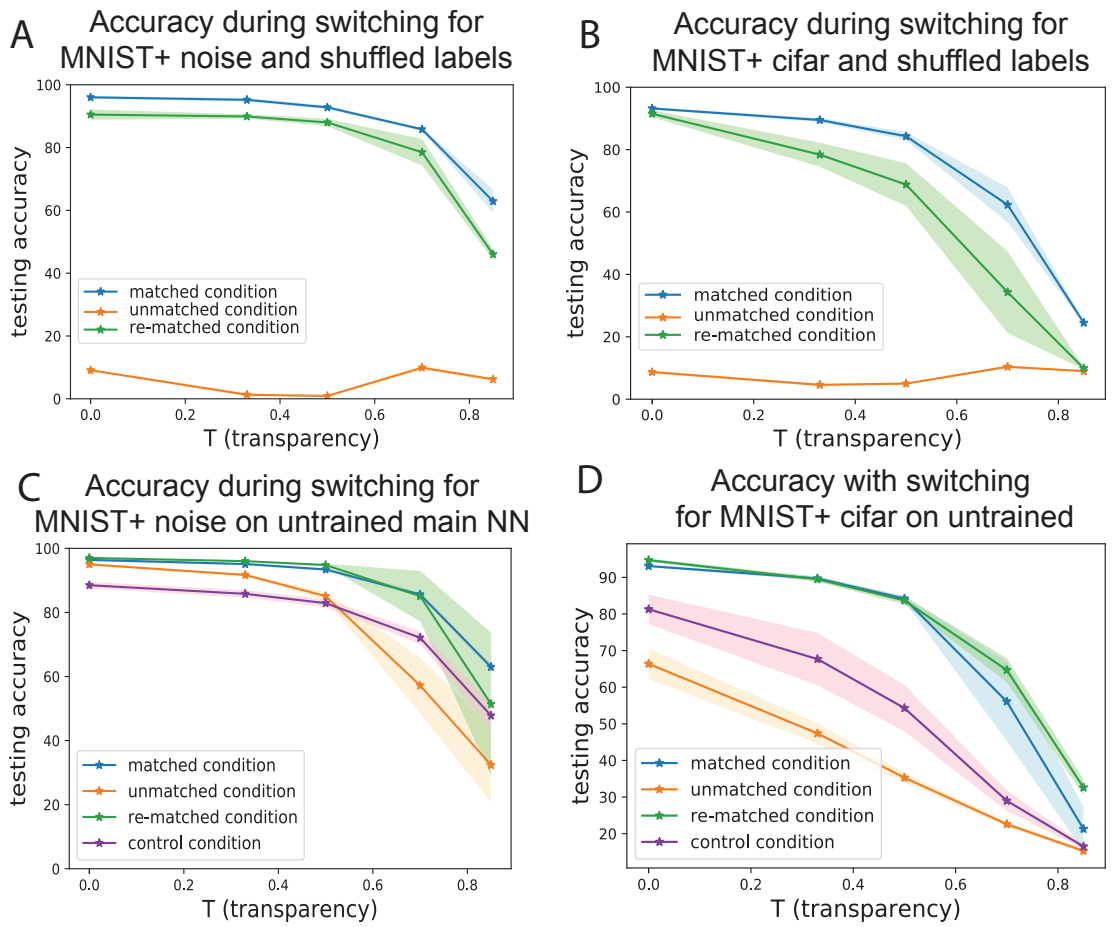


Figure 3.12: A) Accuracy for the bottleneck-switching network on MNIST+noise as labels are shuffled and the network must adapt to a new context as well as different input-output dependencies. This is a control experiment to show the bottleneck-switching network becomes less effective as tasks/contexts become more distinct. B) Same as A), for MNIST+cifar. C) Accuracy for the bottleneck-switching network on MNIST+noise (red line), when training weights to and from the switching units, but keeping weights in the main network randomly initialized. D) Same as A), for MNIST+cifar.

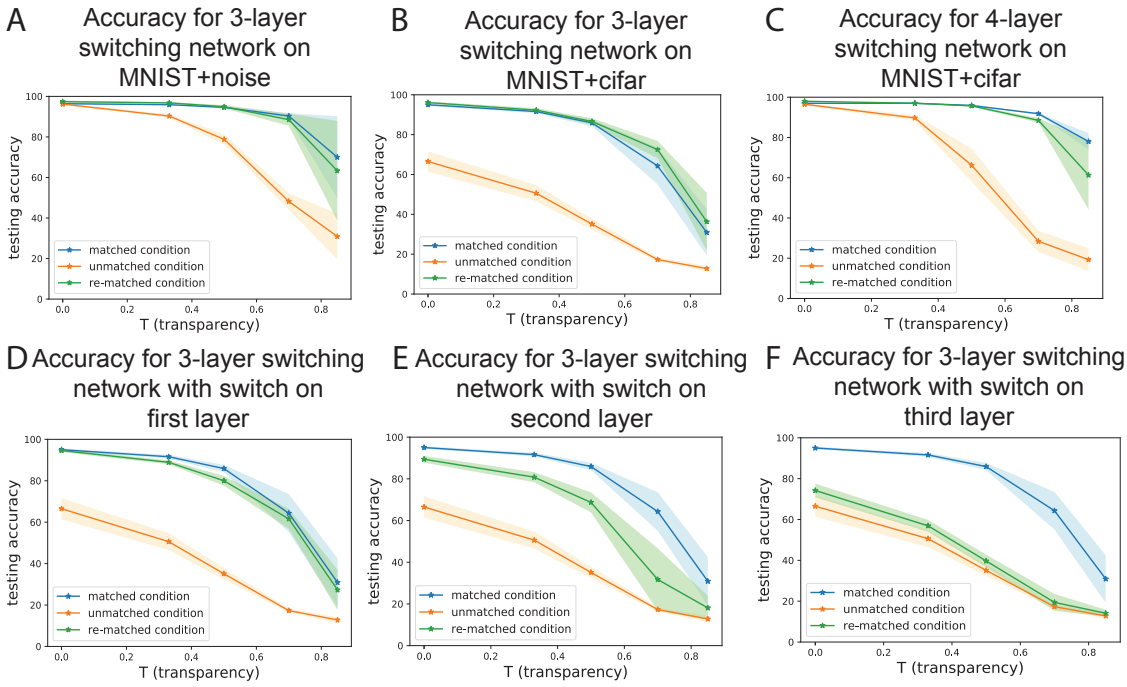


Figure 3.13: Bottleneck-switching networks with 3 or 4 hidden layers are effective. A). Accuracies on MNIST+noise for a 3 hidden layer NN as described in Section 1.2.1, *Network architectures*. B). Same as A), on MNIST+cifar. C). Accuracies on MNIST+noise for a 4 hidden layer NN as described in Section 1.2.1, *Network architectures*. D). Accuracies on MNIST+cifar for a 3 hidden layer NN when switching units are added only to the first hidden layer. We note that the switching units added at this layer (as opposed to deeper layers) are the most effective. E). Accuracies on MNIST+cifar for a 3 hidden layer NN when switching units are added only to the second hidden layer. F). Same as D-E), but with switching units added only to the third hidden layer.

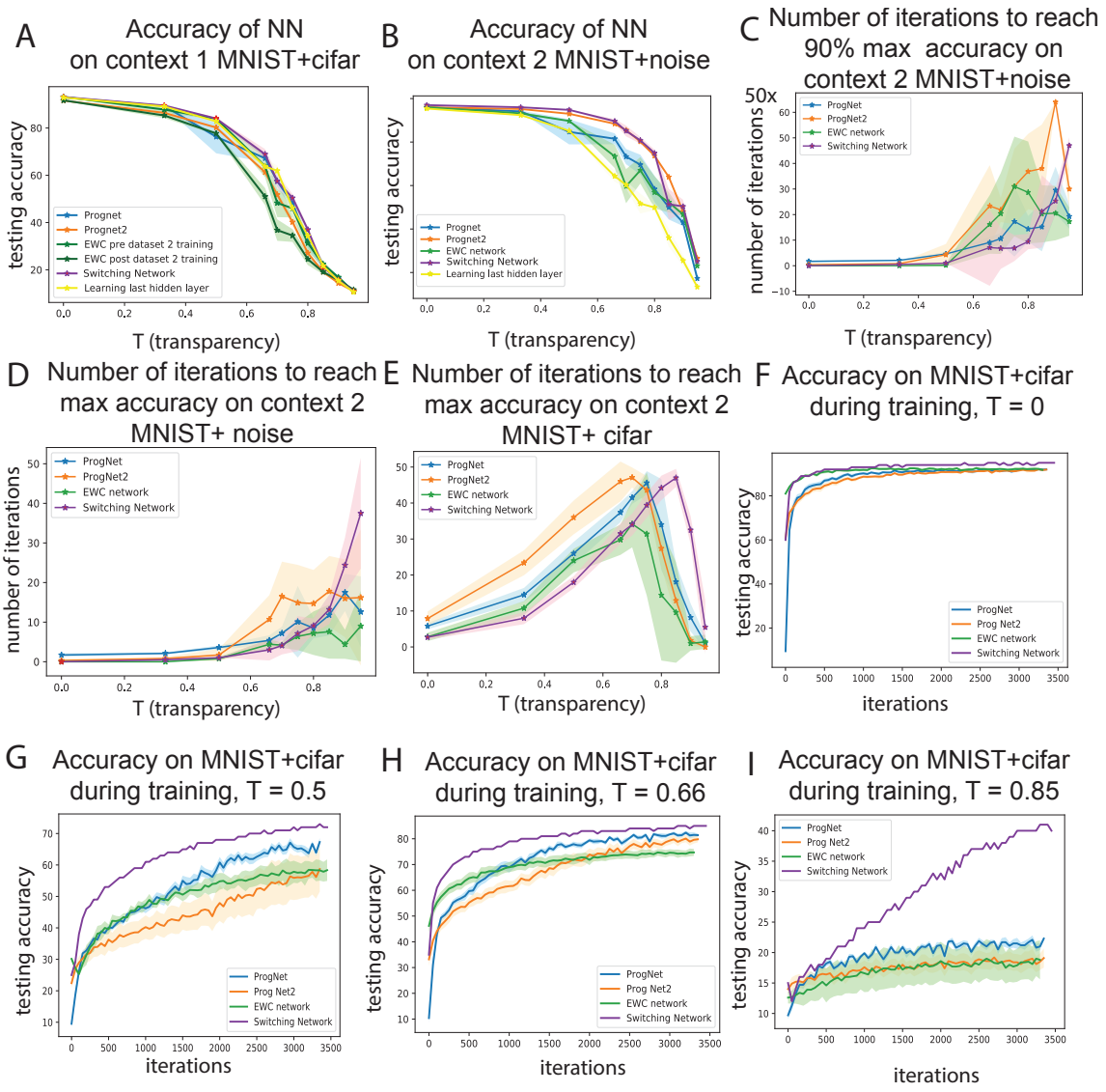


Figure 3.14: A). Accuracy on context 1 (MNIST+cifar) for the five NNs we are testing. B). Accuracies on context 2 (MNIST + noise) for the NNs we are testing. We note that the bottleneck-switching network has a slight edge across all transparencies. C). Number of iterations (in steps of 50) for each NN, during training on MNIST+noise, to reach 90% of the max. peak accuracy (where the maximum is considered over all NNs). D) Number of iterations (in steps of 50) for each NN, during training on MNIST+noise, to reach 90% converging accuracy (where the maximum is considered for each method in part). E) Same as D), for MNIST+cifar. F)-I) Accuracy during training on MNIST+cifar with $T = 0, 0.5, 0.66, 0.85$. A)-C) differs from Fig. 5 in that here context 1 is MNIST+cifar and context 2 is MNIST+noise.

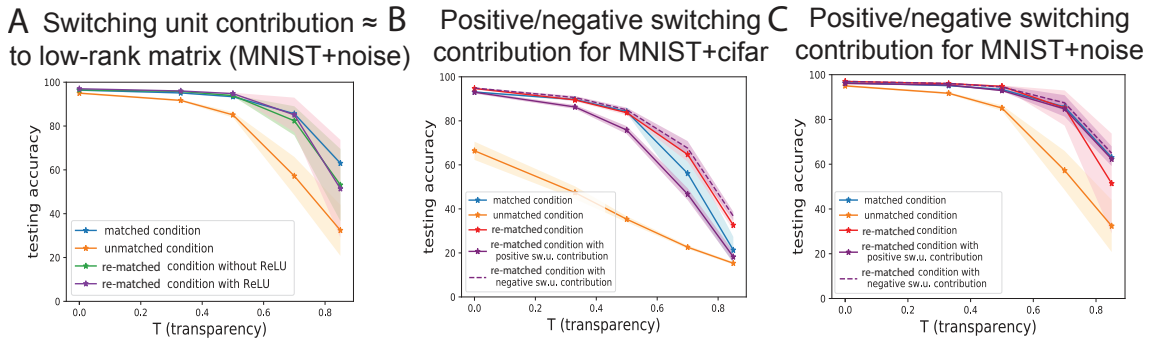


Figure 3.15: A) A bottleneck-switching network without applying the ReLU nonlinearity to the switching unit activations for the MNIST+noise dataset (green line) performs comparably well to the matched condition (blue line) and to the bottleneck-switching network with ReLUs described in main text in Eq. (1) (red line). Without the ReLU non-linearity applied to the switching units, their contribution is low rank. Note that the basic network still has nonlinearities, these are removed here only from the switching units. B). Enforcing a strictly positive switching unit contribution for MNIST+cifar (solid purple line) decreases performance more than enforcing a strictly negative contribution (dashed purple line), p-value$< 2.23 \cdot 10^{-6}$ using the t-test. C). Same as B), but for MNIST+noise.

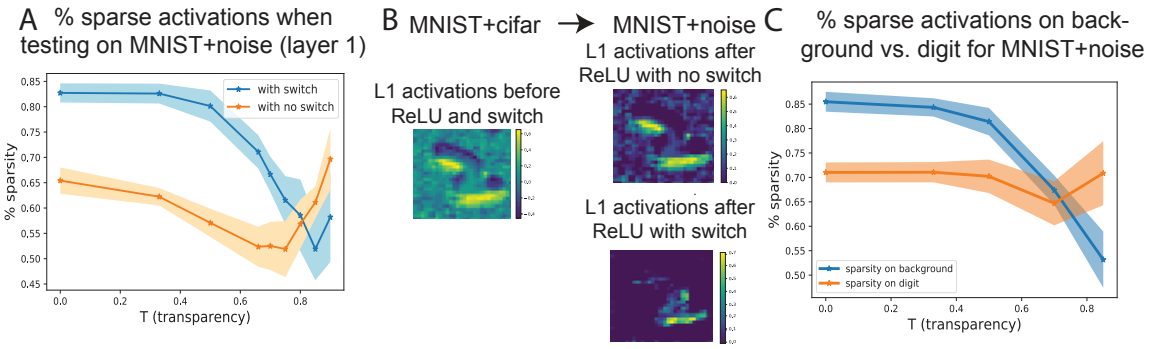


Figure 3.16: A). Sparsity (percentage layer 1 activations which are zero) when switching to MNIST+noise, with or without adding the switching contributions (blue vs orange line), and after applying ReLU. The mechanism described in Section 3.6 of the main text does not seem to apply for high transparencies when switching to MNIST+noise (sparsity % with switching is lower than the sparsity % without switching). B). Examples of (convolutional) layer 1 activation patterns with or without the switching contribution for $T = 0$, when switching to MNIST+noise. Figure 3.13B-C has shown these results for switching to MNIST+cifar. C). Sparsity (percentage layer 1 activations which are 0) within the background (blue) and within the digit (orange) when switching to MNIST+noise after adding the switching contributions and the ReLU non-linearity.

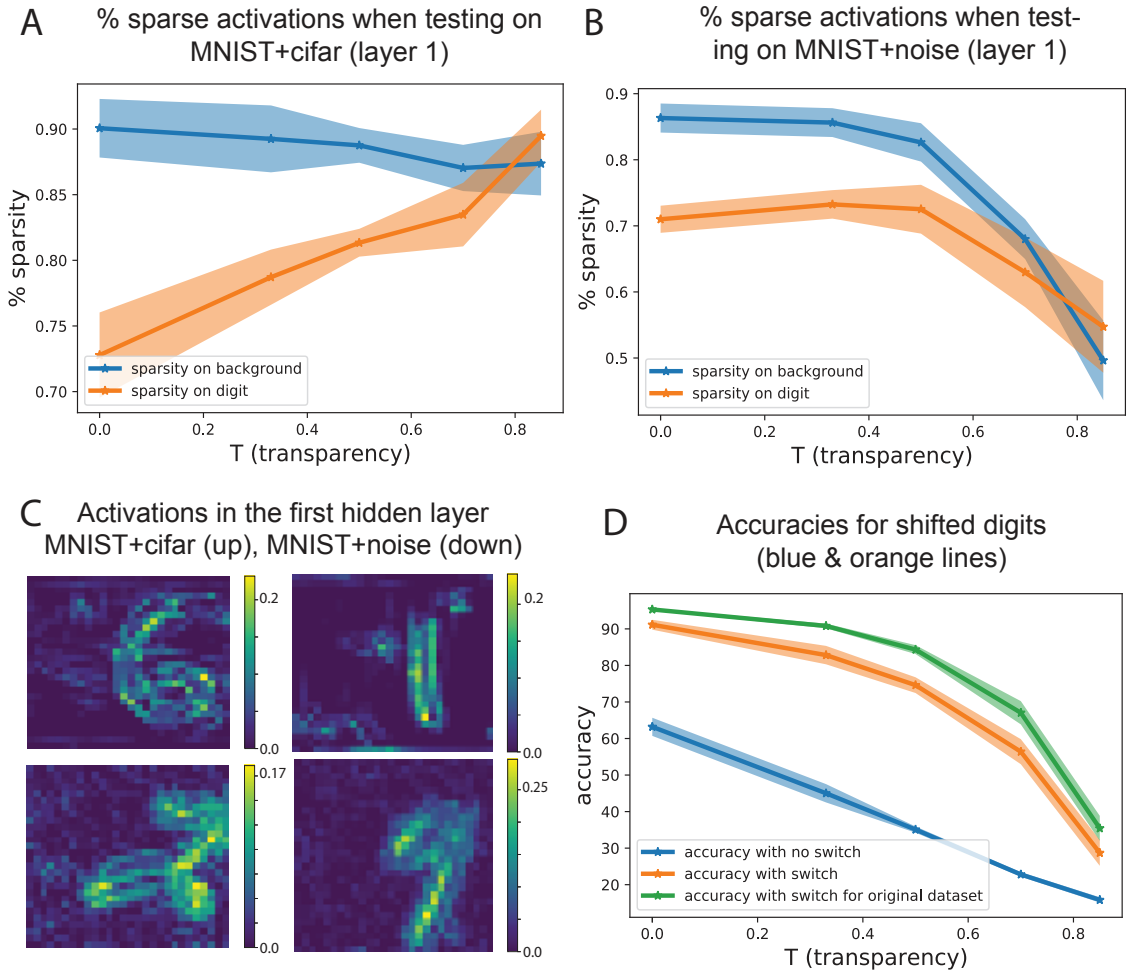


Figure 3.17: A) Sparsity (percentage layer 1 zero activations) when switching to MNIST+cifar from MNIST+noise when the digit is displaced by 5 pixels. The sparsity is represented with or without the switching contributions (blue vs orange line), and after applying ReLU. B) Same as A), but tested on MNIST+noise after training the main network on MNIST+cifar. C) Activations in the first hidden layer when switching to MNIST+cifar (up) and when switching to MNIST+noise (down). The background is consistently inhibited, albeit noisy. D) Accuracies when digits are shifted by 5 pixels (blue and orange lines): accuracy without switching (blue line), but training the last two layers of the basic network (no training of the convolutional layer); accuracy when switching is turned ON at the first hidden layer, using the same training procedure as before, but without training of the weights to and from the switching units corresponding to the first layer (orange line); accuracy of the bottleneck-switching network on the original datasetof centered digits, where training is done on all weights of the network with switching units, (green line).

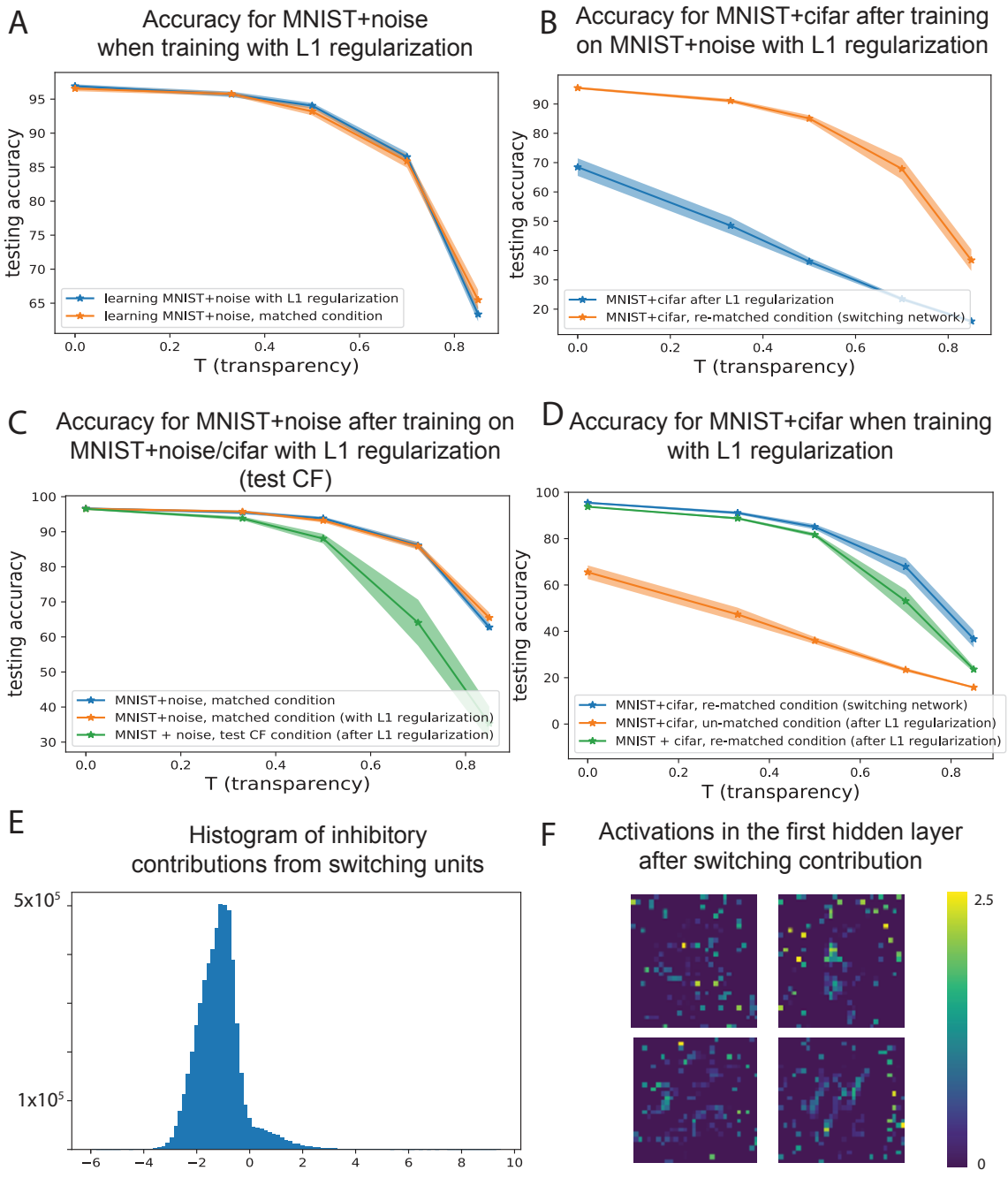


Figure 3.18: A-B) Testing accuracy after training with L1 regularization on the layer 1 activations. C) Testing accuracy for MNIST+noise (green line) after training on MNIST+noise, then training on MNIST+cifar with L1 regularization, both on layer 1 activities. D) Testing accuracy for MNIST+cifar (green line) after training on MNIST+noise with L1 regularization, then training on MNIST+cifar with L1 regularization. E) Histogram of switching unit contributions for the feedforward switching network. F) First hidden layer activations after adding the negative switching unit contribution for the feedforward switching network.

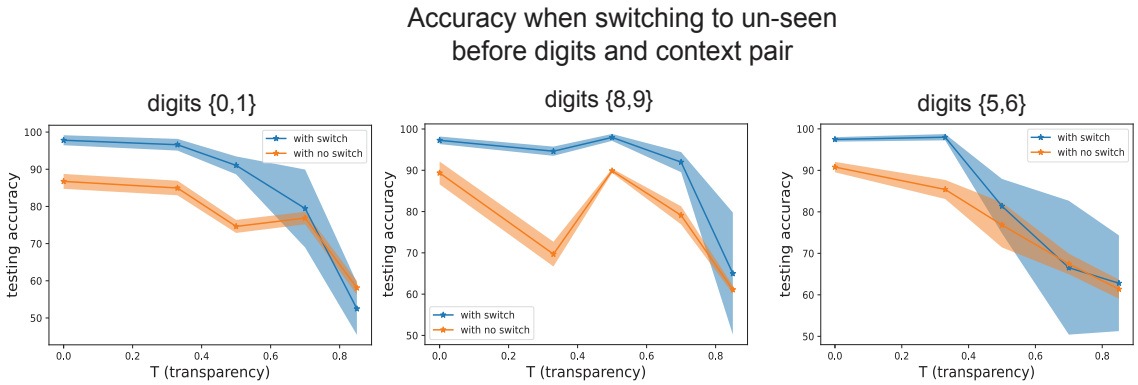


Figure 3.19: A) Testing accuracy on pair $\{0, 1\}$ of digits that the switching units have not been trained on. B) Like A), but using $\{8, 9\}$ as pairs of digits. C) Like A) and B), but using $\{5, 6\}$ as pairs of digits. For A)-C) using mean $\pm$ sem (standard error of the mean) to show average performance.

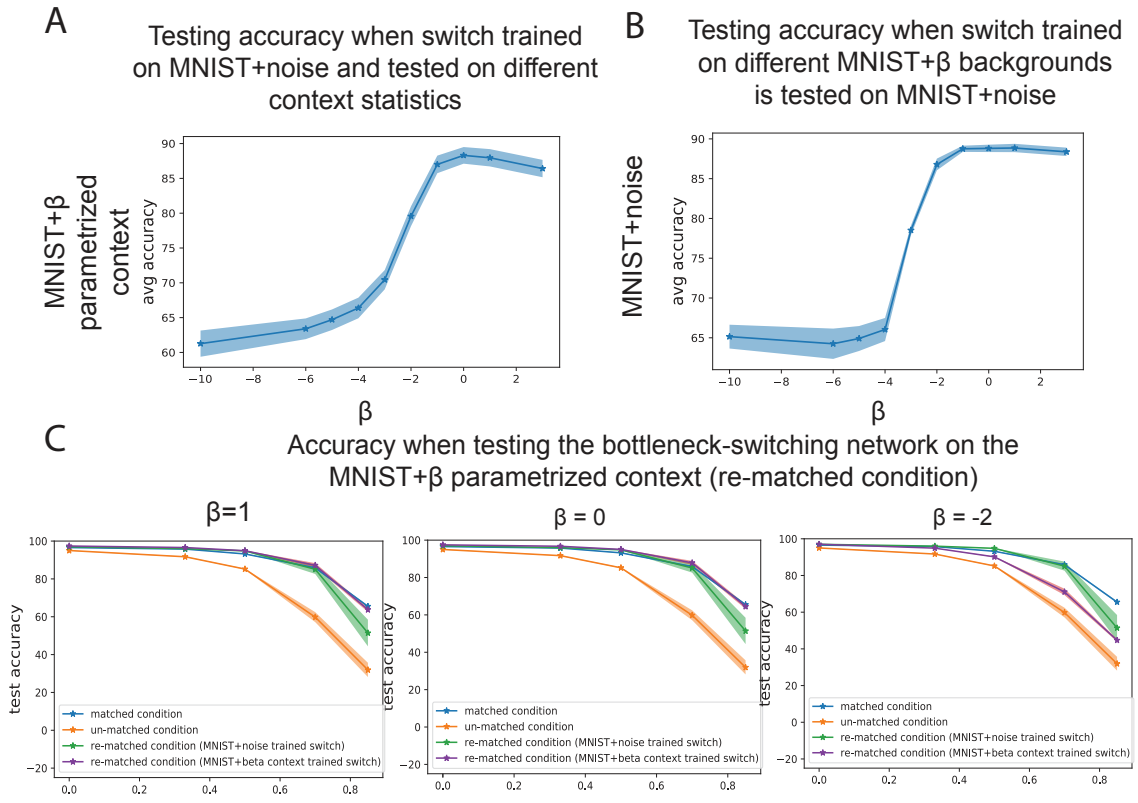


Figure 3.20: A) Testing accuracy on MNIST+β parametrized context datasets using the bottleneck-switching network with switching units ON, after the switching network's main network is trained on MNIST+cifar, and the switching units are trained on MNIST+noise. Peak is for $\beta = 0$, as the MNIST+noise dataset corresponds to the MNIST+β parametrized context dataset with $\beta = 0$. B) Testing accuracy on the MNIST+noise dataset, after training the switching network's main network on MNIST+cifar, and the switching units are trained on MNIST+β. Peak is for $\beta = 0$; C) Testing accuracy on the MNIST+β dataset over different transparencies T, and for different values of β. Bottleneck-switching networks are used where the main network is trained on MNIST+cifar and the switching units are trained on MNIST+noise. Highest accuracy with perfect overlap to matched condition occurs for $\beta = 0$, but equally high accuracy values are attained for $\beta = 1$, $\beta = 3$ (not shown). Matched, un-matched, and re-matched conditions correspond here to testing MNIST+noise.

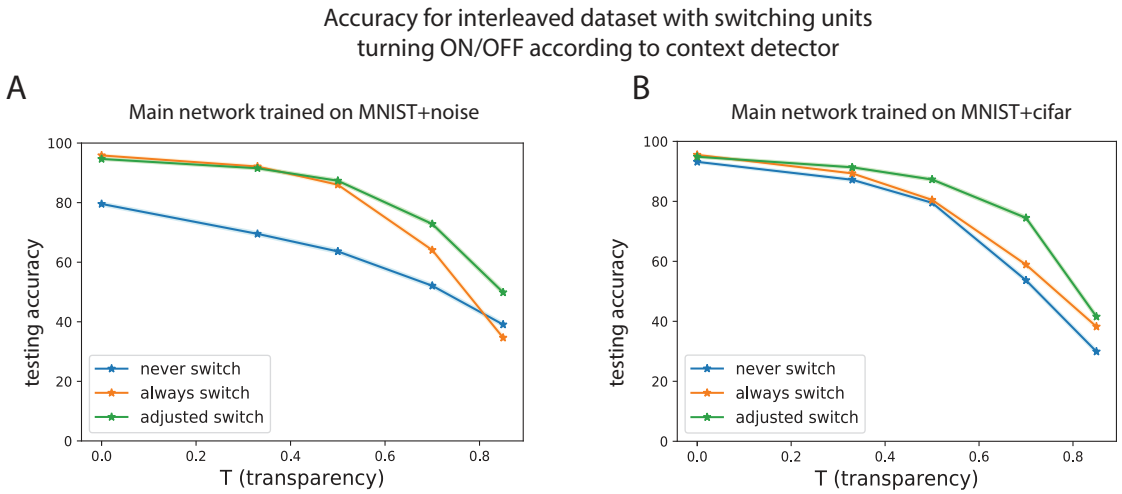


Figure 3.21: Accuracies for a network with a built-in "context detector", a simple one layer classifier trained on the data and used to decide whether the digit is set on noisy or CIFAR-10 context. This classifier outputs 0 for the first context (context 1) and 1 for the second context (context 2). The data to test this network contains interleaved MNIST+noise and MNIST+cifar images. A). Accuracy using the network with a built-in context detector, with the main network trained on MNIST+noise and switching units trained on MNIST+cifar. B). Same as A), but the main network is trained on MNIST+cifar and the switching units are trained on MNIST+noise.

Bibliography

[1] WC Abraham and A Robins. Memory retention and weight plasticity in ann simulations. Trends in Neurosciences, 28(2):73–78, 2005.

[2] R Amir, R Zemel, and J Tsotsos. The elephant in the room, 2018.

[3] A Barbu, D Mayo, J Alverio, W Luo, C Wang, D Gutfreund, J Tenenbaum, and B Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In Advances in Neural Information Processing Systems, volume 32, pages 9448–9458, 2019.

[4] S Beery, G Van Horn, and P Perona. Recognition in terra incognita. In European Conference on Computer Vision (ECCV), pages 472–489, 2018.

[5] Marco Birck. Catastrophicforgetting-ewc [github repository]. https://github.com/mabirck/CatastrophicForgetting-EWC, 2018, Last Retrieved January 2022.

[6] K Bousmalis, N Silberman, D Dohan, D Erhan, and D Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks, 2016.

[7] K Bousmalis, G Trigeorgis, N Silberman, D Krishnan, and D Erhan. Domain separation networks. In Advances in Neural Information Processing Systems, pages 343–351, 2016.

[8] L Bruzzone and M Marconcini. Domain adaptation problems: A dasvm classification technique and a circular validation strategy. IEEE transactions on pattern analysis and machine intelligence, 32(5):770–787, 2010.

[9] R Chattopadhyay, J Ye, S Panchanathan, W Fan, and I Davidson. Multi-source domain adaptation and its application to early detection of fatigue. In In Proc. KDD, volume 6, pages 717–725, 2011.

[10] MJ Choi, A Torralba, and AS Willsky. Context models and out-of-context objects. Pattern Recognition Letters, 33(7):853–862, 2012.

[11] WS Chu, F de la Torre, and JF Cohn. Selective transfer machine for personalized facial action unit detection. In Proceedings of the IEEE Conference on cOmputer Vision and Pattern Recognition, pages 3515–3522, 2013.

[12] G Csurka. Domain adaptation for visual applications: A comprehensive survey. In G Csurka, editor, Domain Adaptation in Computer Vision Applications. Advances in Computer Vision and Pattern Recognition. Springer, Cham, 2017.

[13] D Voina D, E Shea-Brown E, and S Mihalas. A biologically inspired architecture with switching units can learn to generalize across backgrounds. bioRxiv, 2021.

[14] HIII Daume. Frustratingly easy domain adaptation. In Proceedings of ACL., pages 256–263, 2007.

[15] T de Vries, I Misra, C Wang, and L van der Maaten. Does object recognition work for everyone? In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, pages 52–59, 2019.

[16] J Donahue, Y Jia, O Vinyals, J Hoffman, N Zhang, E Tzeng, and T Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In ICML: Proceedings of the 31st International Conference on International Conference on Machine Learning, volume 32, 2014.

[17] TJ Draelos, NE Miner, CC Lamb, CM Vineyard, KD Carlson, CD James, and JB Aimone. Neurogenesis deep learning. In International Joint Conference on Neural Networks (IJCNN), pages 526–533, 2017.

[18] L Duan, IW Tsang, and D Xu. Domain transfer multiple kernel learning. In IEEE Trans Pattern Anal Mach Intell., volume 34, pages 465–479, 2012.

[19] L Duan, D Xu, and SF Chang. Exploiting web images for event recognition in consumer videos: a multiple source domain adaptation approach. In IEEE 2012 conference on computer vision and pattern recognition, pages 1338–1345, 2012.

[20] RM French. Pseudo-recurrent connectionist networks: An approach to the sensitivity-stability dilemma. In Connection Science, volume 9, pages 353–380, 1997.

[21] X Glorot, A Bordes, and Y Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In Proceedings of the 28th international conference on machine learning, volume 27, pages 97–110, 2011.

[22] B Gong, K Grauman, and F Sha. Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In International Conference on Machine Learning, pages 222–230, 2013.

[23] GE Hinton and DC Plaut. Using fast weights to deblur old memories. In Proceedings of the Annual Conference of the Cognitive Science Society, pages 177–186, 1987.

[24] Yearsley J. Generate spatial data. (https://www.mathworks.com/matlabcentral/fileexchange/5091-generate-spatial-data), MATLAB Central File Exchange, May 2004, Last Retrieved May 2022.

[25] Maxwell Joseph Jacobson. Doric [github repository]. https://github.com/arcosin/Doric, 2020, Last Retrieved January 2022.

[26] H Jung, J Ju, M Jung, and J Kim. Less-forgetting learning in deep neural networks, 2016.

[27] L Kaiser, AN Gomez, N Shazeer, A Vaswani, N Parmar, L Jones, and J Uszkoreit. One model to learn them all, 2017.

[28] R Kemker and C Kanan. Fearnet: Brain-inspired model for incremental learning. In International Conference on Learning Representations (ICLR), 2018.

[29] J Kirkpatrick, R Pascanu, N Rabinowitz, J Veness, G Desjardins, AA Rusu, and et al. Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences, 114(13):3521–3526, 2017.

[30] D Kumaran, D Hassabis, and JL McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. Trends in Cognitive Sciences, 20(7):512–534, 2016.

[31] A Kuznetsova, H Rom, N Alldrin, J Uijlings, I Krasin, J Pont-Tuset, S Kamali, S Popov, M Malloci, T Duerig, and et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale, 2018.

[32] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 2013.

[33] J.L. Lennon. Red-shifts and red herrings in geographical ecology. Ecography, 23:101–113, 2000.

[34] F Li, SJ Pan, O Jin, Q Yang, and X Zhu. Cross-domain co-extraction of sentiment and topic lexicons. In Proceedings of the 50th annual meeting of the association for computational linguistics long papers, pages 410–419, 2012.

[35] Y Li, N Wang, J Shi, J Liu, and X Hou. Revisiting batch normalization for practical domain adaptation, 2016.

[36] Z Li and D Hoiem. Learning without forgetting. In ECCV 2016: Computer Vision, pages 614–629, 2016.

[37] T Lin, M Maire, S Belongie, J Hays, P Perona, D Ramanan, P Dollár, and CL Zitnic. Microsoft coco: Common objects in context. In European Conference on Computer Vision (ECCV), 2014.

[38] MY Liu and O Tuzel. Coupled generative adversarial networks. pages 469–477, 2016.

[39] M Long, J Wang, G Ding, J Sun, and PS Yu. Transfer feature learning with joint distribution adaptation. In Proceedings of the 2013 IEEE international conference on computer vision, pages 2200–2207, 2013.

[40] M Long, J Wang, and MI Jordan. Deep transfer learning with joint adaptation networks, 2016.

[41] A Mallya, D Davis, and S Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. ECCV 2018: Computer Vision, pages 72–88, 2018.

[42] A Mallya and S Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7765–7773, 2018.

[43] Valerio Mante, David Sussillo, Krishna Shenoy, and William Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. Nature, 503:78–84, 2013.

[44] JL McClelland, BL McNaughton, and RC O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. Psychological Review, 102(419-457), 1995.

[45] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. Psychology of Learning and Motivation, 24(C):109–165, 1989.

[46] I Misra, A Shrivastava, A Gupta, and M Hebert. Cross-stitch networks for multi-task learning. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[47] M Oquab, L Bottou, I Laptev, and J Sivic. Learning and transferring mid-level image representations using convolutional neural networks. pages 1717–1724, 2014.

[48] SJ Pan, IW Tsang, JT Kwok, and Q Yang. Domain adaptation via transfer component analysis. In IEEE Trans Neural Netw., volume 22, pages 199–210, 2009.

[49] GI Parisi, R Kemker, JL Part, C Kanan, and S Wermter. Continual lifelong learning with neural networks: A review. Neural Networks, 113:57–71, 2019.

[50] Roger Ratcliff. Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. Psychological Review, 97(2):285–308, 1990.

[51] AS Razavian, H Azizpour, J Sullivan, and S Carlsson. Cnn features off-the-shelf: An astounding baseline for recognition,. In IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 512–519, 2014.

[52] SA Rebuffi, A Kolesnikov, G Sperl, and CH Lampert. icarl: Incremental classifier and representation learning, 2016.

[53] O Russakovsky, J Deng, H Su, J Krause, S Satheesh, S Ma, Z Huang, A Karpathy, A Khosla, M Bernstein, and et al. Imagenet large scale visual recognition challenge. International Journal of Computer Vision, 115(3):211–252, 2015.

[54] AA Rusu, NC Rabinowitz, G Desjardins, H Soyer, J Kirkpatrick, K Kavukcuoglu, R Pascanu, and R Hadsell. Progressive neural networks, 2016.

[55] S Sagawa and L Percy WK Pang, HB Tatsunori. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In International Conference on Learning Representations, 2020.

[56] R Shetty, M Fritz, and B Schiele. Adversarial scene editing: Automatic object removal from weak supervision. In Neural Information Processing Systems (NeurIPS), 2018.

[57] Y Shi and F Sha. Information-theoretical learning of discriminative clusters for unsupervised domain adaptation. In Proceedings of the 29th international conference on machine learning, pages 1–8, 2012.

[58] H Shin, JK Lee, J Kim, and J Kim. Continual learning with deep generative replay. In NIPS: Proceedings of the 31st International Conference on Neural Information Processing Systems, pages 2994–3003, 2017.

[59] K Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2015.

[60] B Sun and K Saenko. Deep coral: Correlation alignment for deep domain adaptation. In Computer Vision–ECCV 2016 Workshops, pages 443–450. Springer, 2016.

[61] T Tommasi, F Orabona, and B Caputo. Safety in numbers: learning categories from few examples with multi model knowledge transfer. In IEEE Conf Comput Vision Pattern Recog., pages 3081–3088, 2010.

[62] A Torralba and AA Efros. Unbiased look at dataset bias. CVPR, pages 1521–1528, 2011.

[63] D Voina, S Recanatesi, B Hu, E Shea-Brown, and S Mihalas. Single circuit in v1 capable of switching contexts during movement using vip population as a switch. Neural Computation, 34 (3):pp. 541–594, 2022.

[64] D Voina, E Shea-Brown, and S Mihalas. mnist+context, 2021.

[65] A Wang, A Narayanan, and O Russakovsky. Revise: A tool for measuring and mitigating bias in visual datasets. In European Conference on Computer Vision (ECCV), 2020.

[66] Karl Weiss, Taghi M Khoshgoftaar, and Ding Ding Wang. A survey of transfer learning. Journal of Big Data, 3(1):9, 2016.

[67] M Weng and W Deng. Deep visual domain adaptation: A survey. Neurocomputing, 312:135–153, 2018.

[68] R Xia, C Zong, X Hu, and E Cambria. Feature ensemble plus sample selection: domain adaptation for sentiment classification. In IEEE Intell Syst., volume 28, pages 10–18, 2013.

[69] K Xiao, L Engstrom, A Ilyas, and A Madry. Noise or signal: The role of image backgrounds in object recognition, 2020.

[70] T Xiao, H Li, W Ouyang, and X Wang. Learning deep feature representations with domain guided dropout for person re-identification. In In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1249–1258, 2016.

[71] T Xiao, J Zhang, K Yang, Y Peng, and Z Zhang. Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In Proceedings of the ACM International Conference on Multimedia, pages 177–186, 2014.

[72] Y Yao and G Doretto. Boosting for transfer learning with multiple sources. In Proceedings of the IEEE computer society conference on computer vision and pattern recognition, pages 1855–1862, 2010.

[73] J Yoon, E Yang, J Lee, and SJ Hwang. Lifelong learning with dynamically expandable networks. In International Conference on Learning Representations (ICLR), 2018.

[74] F Yu, JM Tucciarone, JS Espinosa, N Sheng, DP Darcy, RA Nicoll, ZJ Huang, and MP Stryker. A cortical circuit for gain control by behavioral state. Cell, 156(6):1139–1152, 2014.

[75] G Zeng, Y Chen, B Cui, and S Yu. Continuous learning of context-dependent processing in neural networks. Nature Machine Intelligence, 1:364–372, 2019.

[76] F Zenke, W Gerstner, and S Ganguli. The temporal paradox of hebbian learning and homeostatic plasticity. In Neurobiology, volume 43, pages 166–176, 2017.

[77] F Zenke, B Poole, and S Ganguli. Continual learning through synaptic intelligence. In ICML: Proceedings of the 34th International Conference on Machine Learning, volume 70, 2017.

[78] G Zhou, K Sohn, and H Lee. Online incremental feature learning with denoising autoencoders. In International Conference on Artificial Intelligence and Statistics, pages 1453–1461, 2012.

[79] Z Zhu, L Xie, and A Yuille. Object recognition without and without objects. In International Joint Conference on Artificial Intelligence, 2017.

Chapter 4

CELL TYPE CLASSIFICATION FROM CONNECTIVITY INFORMATION USING GRAPH NEURAL NETWORKS

4.1 Introduction and Background

A hallmark of the monumental task facing scientists seeking to untangle the brain's complex dynamical repertoire is its high heterogeneity, with many different types of neurons and interaction patterns across a range of spatial and temporal scales. This is in contrast with physics applications which often assume simple lattice or random network architectures [27], or with ANN architectures where all units and connections are indistinguishable from each other. For instance, while ANN units treat their inputs in a stereotyped way by applying a linear mapping to an input vector, followed by the same non-linear function across all units to obtain a static state, biological neurons have a wide range of internal dynamics and intrinsic neuronal mechanisms modulating their activities. By utilizing an array of nonlinear transformations, biological neural activity gives rise to a highly heterogeneous dynamics.

There are at least two, often overlapping, ways in which heterogeneity is apparent in the brain: through cell types and through connectivity patterns. The precise computational role of this heterogeneity is still a subject of intense research, with a range of promising approaches have already underway (e.g. [25, 9]).

Neurons have been found to differ from each other by morphology, gene transcription, electrical properties, and location in the brain [41, 42, 20, 19] (Figure 4.1A). Classifying neural cell types dates back to the pioneering work of Ramon y Cajal, who described in fine detail elaborate neural morphology [26]. More recently, a large-scale effort at the Allen Institute for Brain Science has characterized the diversity of cell types in the primary visual cortex (V1) of the adult mouse using electrophysiology, morphological reconstructions, connectiv-

ity, and modeling [42]. Genetically identified neurons have each been mapped to a common coordinate framework and distinguished through their morphological and electrophysiological properties. This has resulted in the Allen Cell Types Database, a dataset publicly and freely available at http://celltypes.brain-map.org. While unsupervised machine learning methods like clustering have used quantitative features to classify cell types [20], an interesting study has developed a different (computational) approach [42]. Here, simplified point neuron models called GLIF (Generalized Leaky Integrate and Fire) models are found to have a striking ability to fit neuronal spiking data, reproducing spiking behavior with high fidelity. Parameters obtained from the fitting (e.g., membrane resistance, capacitance, resting potential, firing threshold, etc.) can be used to successfully classify cells into "types" associated with specific transgenic lines, effectively reducing the biological space to a set of useful parameters for cell type classification. While Teeter et. al. [42] associates cell types to specific parameters responsible for neurons' internal dynamics, our study aims to draw a correspondence between neuron types their and connectivity patterns, or motifs.

4.1.1 Inferring cell types and their computational roles

Characterizing cell types through empirical measurements (as in [20]), through their specific intrinsic dynamics (as in [42]), as well as through their connections (as in [19]) and higher-order connectivity patterns (as shown in this study), could pave to way to better understanding the biological functions these units have and the specific computational roles they play. So far, many possible roles for neural cell types have been suggested, including the balancing of excitation and inhibition and synaptic normalization [31]. More recently, the diversity of cell types and the associated cell-type–specific neuromodulators have been proposed as possible solutions to the credit-assignment problem, which tries to infer the amount by which network weights should be adjusted to drive learning of a task [25]. In addition to their role in learning, diverse cell types have been found to improve processing of temporal data [45, 7], via more biophysically realistic models with after-spike currents (currents induced or modulated by neuronal spikes), as in [45], or with heterogeneous synaptic timescales, as in [7]. A similar study [35] finds that heterogeneity, for instance in membrane

and synaptic time constants, causes networks to be more stable and robust in tasks with a rich temporal structure. Similar results are found in Zeldenrust et. al., 2021 [49] where a class of recurrent filter networks with heterogeneous spiking neurons are both more efficient and more robust against correlated noise than their homogeneous counterparts. The improved performance of these networks on standardized sequential datasets point to an active and important role of cell type heterogeneity in allowing animals to learn in changing environments. Another study investigated cell heterogeneity through the lens of connectivity. Embedding cell type specific connectivity rules into network architectures in addition to adjusting the connections through learning can improve performance with a reduced number of neurons [40]. The resulting neural networks with architectural features at the statistical level are likely to provide a powerful platform for subsequent rapid learning. Additionally, cell type diversity has been shown to reduce pair-wise output spike correlations, enhance information carrying capacity, and increase functional specialization, thus playing an important role for neural coding [17, 16, 34].

Specifying cell types can enhance the biological realism in computational models and provide further insights, as described in Chapter 2, where our microcircuit model with PYR, SST, VIP and PV neuron populations was able to switch between optimized processing of two different contexts. A similar study [14] employing a recurrent network model with cell types highlights a different approach to explain the same biological phenomenon. Furthermore, cell types can be incorporated into ANNs as well. For instance, neural cell types can be instantiated as diverse activation functions in CNNs by mixing multiple activation functions within each activation layer [15]. These heterogeneous cell type networks can have higher image classification performance than that of homogeneous control networks with only one activation function per network. Hence diversifying activation functions can assist the computational abilities of neural circuits. Another application to ANNs incorporates Dale's law, which ensures that biological neurons are either exclusively excitatory or inhibitory, into the artificial network architecture [10]. These ANNs with excitatory and inhibitory units are shown to learn just as well as standard ANNs by implementing normalization schemes and specific update rules to inhibitory units. In Chapter 3, we showed

that by including specific types of units – the switching units – our network was capable of performing tasks in a continual learning setting, where it was prevented from forgetting to do classification on background statistics learned previously. Lastly, cell type information can improve our data analysis, specifically population analysis approaches where dimensionality reduction methods are applied without taking into account neuronal heterogeneity [5]. Knowledge of neuron type is shown to be important, allowing for stronger statistical tests when interpreting population activity structure.

4.1.2 Inferring circuit structure through different approaches

These neurons of different types are all inter-connected through a web of networks, forming patterns of connectivity at different scales. To model the network structure at either the neuron, population, or brain area levels, tools from network science are useful [27]. Several generative network models help explain important properties of brain networks and how these properties have emerged from underlying biological mechanisms. As mentioned in Chapter 1, the simplest and most common model for generating random networks is the Erdos–Renyi model, which produces purely random networks wherein each pair of nodes is connected independently with a fixed probability P. While these capture some aspects of connectivity (e.g., the network variability among members of a species and certain neuronal population connectivity patterns, e.g. in the cerebellum), these models are insufficient to capture realistic properties of the network and the mechanisms by which they grow. Other types of well-known models are the stochastic-block model generating a community structure of densely-connected communities separated by sparse inter-community connectivity; the Watts-Strogatz model generating small-world structures, where average path lengths between all nodes are significantly shorter than in a random network; the Barabasi-Albert model generating hub structure, an architecture where high degree "hubs" form a densely inter-connected structural core [27].

While these generative models provide descriptive statistics that shape the neuronal architecture, on the opposite end we also have fairly specific cell-type connectivity information.

For instance, several studies detail the logic of connections between molecularly distinct interneurons (SST, VIP, PV) and excitatory neurons (PYR) in mouse V1 [36], [8] (Figure 4.1B). Interneurons are a variety of molecularly distinct types of GABAergic neurons responsible for cortical inhibition that can also inhibit one another. According to Pfeffer et. al., PV neurons strongly inhibit one another but provide little inhibition to other populations; SST interneurons avoid inhibiting one another yet strongly inhibit all other populations; VIP interneurons preferentially inhibit SST interneurons. In another example study, Campagnola et. al. [8] examines the principles that relate cell-type to circuit organization in the mouse and human cortex. In the stereotyped network architecture of the cortex in which synapse properties and connectivity are strongly influenced by cell type, the authors compare connection probabilities across layer, cell subclass, and species. For instance, connectivity between excitatory cells and VIP inhibitory cells was present in layer 2/3 and absent in layer 5/6 of mouse cortex. Likewise, connection probability among layer 4 excitatory cells was high in mouse cortex and nearly absent in human cortex. In a similar study, Tasic et. al [41], provides a transcriptomic and projectional taxonomy of cortical cell types. While nearly all GABAergic interneurons form local connections, the authors find that they can match transcriptomic types of glutamatergic neurons to long-range projections.

In addition to characterizing network structure through generative models that specify connection statistics, and defining a more precise logic of inter-connectivity between diverse cell types, a different approach goes beyond global statistical features of (first-order) connections. Namely, networks can be decomposed into *network motifs*, patterns of interconnections between nodes of a network, often focusing on the ones that recur at frequencies much higher than those found in randomized networks (Figure 4.1C). Network motifs are considered the building blocks of directed networks and have been investigated in biological, technological, social, and even word-adjacency networks [33] showing that these networks can be grouped into clusters that share similar motif abundance profiles [38]. A plausible hypothesis is that significant motifs may possess properties important enough to become overrepresented or are conducive to the function of the organism within its environment,

and therefore favored by evolutionary forces [38, 18].

Including information about node identity has been shown to be important in motif discovery. In a study of the network patterns in C. elegans [38] for instance, both the wiring diagram and functional information about neurons (i.e. whether neurons were sensory, motor, or interneurons) was used in order to reveal the most abundant motifs and their computational properties. Listing motifs solely based on topology, and thus treating each neuron and edge on the same footing, proved to be counterproductive. This is because two topologically identical motifs whose nodes perform very different functions will play very different roles in the network. Particular motifs are indeed significantly more abundant in the worm than expected by chance, and have computational functions related to the feed-forward structure of information processing in the network, while avoiding feedback loops. Interneurons are strongly over-represented among the common motifs, processing information from the sensor neurons towards the muscles. Some of the most common motifs identified play a crucial role in the system of neurons controlling the worm's locomotion. Another study on C elegans have focused on motif detection using software developed by U.Alon [22] to find significant 3- and 4- node motifs and produce simulations of their dynamics. Further findings on the structure-function relationship also in the nematode focuses on how the anatomical connectome and neuronal dynamics relate to each other [44]. Surprisingly, few local connectivity motifs (i.e. pair-wise connections between neurons), and mostly other non-local features such as triplet motifs and input similarities can predict functional relationships between neurons. Network topology is found to be sufficient to predict correlations in nervous system- wide neuronal dynamics, and triplet motifs play an important role.

The functional role of network motifs could explain their relative abundance throughout networks. The impact of motifs has been recently documented in network fragility analysis and classification in the context of power grid networks, though the authors have emphasized the generality of their findings [12]. Network resilience and reliability is assessed under various types of intentional attacks, and is found to depend on the presence or absence of

more local connectivity patterns such as motifs, as opposed to more global characterstics of the network. Hence, motif characteristics, such as motif concentrations, can be potentially used as alternative local metrics of network robustness, in power grids and more generally in complex networks, as well as early warning indicators of system degradation and failure. In the context of biology and neuroscience, a computational analysis demonstrates that stability or robustness to small perturbations is highly correlated with the relative abundance of small network motifs in several previously determined biological networks. Robust dynamical stability is proposed as an influential property that can determine the non-random structure of biological networks [37]. Other work has shown that motifs play an important role in gene regulation [2, 29], accelerated response times [30], and dynamic stability [28].

The architecture of network connectivity, specifically higher-order network motifs encompassing neurons in larger neighborhoods, along with the dynamical properties of single cells, has been shown to shape the magnitude and timescale of correlations [43]. In Trousdale et. al. [43], explicit expressions for the approximate cross-correlation between neurons is derived, and expanded in terms of paths through the network. Similar results are shown through simulations of the C. elegans neural network, where graph features in the connectome are compared to correlations in the neuronal dynamics [44]. Few local connectivity motifs, and mostly non-local connection patterns such as triplet motifs and input similarities, are responsible for functional relationship between neurons. Moreover, hub neurons in the network are found to be key to these correlations: by inhibiting multiple hub neurons brain-wide correlations are specifically disrupted.

Motifs and a diversity of connections between them have been found to form large statistically over-represented structures which are an additional type of building block in complex networks [18]. The study of motif aggregation has focused on the well-documented feed-forward loop (FFL) motif, and shows how structures of FFLs are organized in a range of natural and engineered networks (Figure 4.1C). Highly distinctive types of FFL cluster for different types of network, including in biology. More research is needed to detect, categorize, and quantify motif clusters.

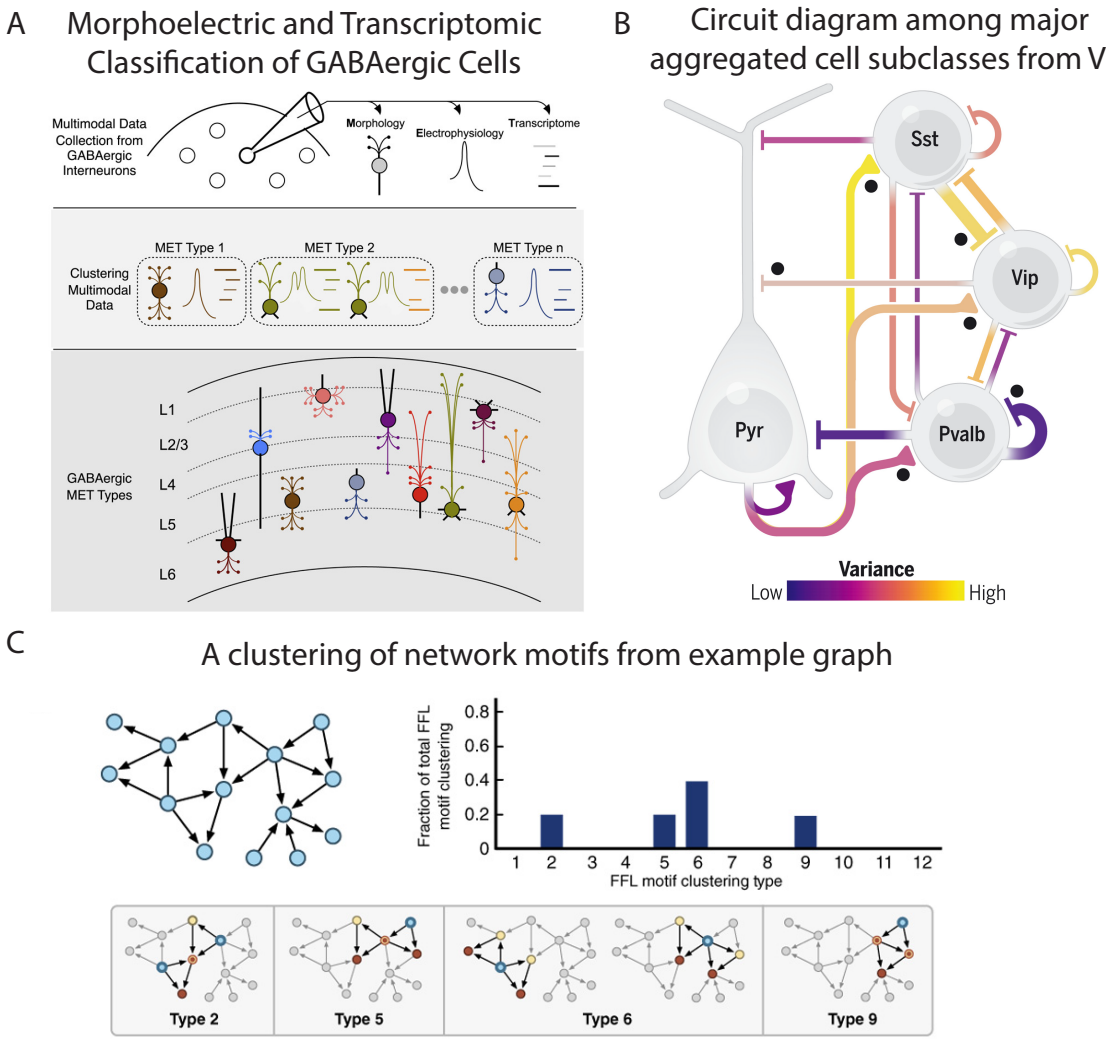


Figure 4.1: (a) Schematic from [19], where the transcriptomes and physiological properties of over 4,200 mouse V1 GABAergic interneurons are characterized and the local morphologies of 517 of those neurons are reconstructed. Most cells mapping to a transcriptomic type exhibit consistent electrophysiological and morphological properties. Through multimodal integrated analysis, 28 *met-types* are defined with congruent morphological, electrophysiological, and transcriptomic properties and robust mutual predictability. (b) Intralaminar circuit diagram among major excitatory (PYR) and inhibitory (PV, SST, and VIP) cell subclasses aggregated from all layers of mouse primary visual cortex. From [8]. (c) Example network (left) with the associated motif clustering type distribution (right), and the motif pairs and their classifications (below). From [18].

4.1.3 Neural networks as directed, heterogeneous graphs

Graphs are a kind of data structure which models a set of objects (nodes) and their relationships (edges). Biological and artificial neural networks are naturally modeled as graphs, with nodes represented by neurons and edges by connections between neurons. Specifically, graphs representing neural networks are directed graphs because connectivity between neurons is directional, so that there are pairs of neurons where only one neuron projects upon another; neural network graphs are also heterogeneous, if we want to add details such as cell-type specificity and the connection type between neurons. Graphs may specify for example if an edge represents an inhibitory or an excitatory connection, even if this information is already redundant when specifying cell-type (whether a connection is inhibitory or excitatory depends only on whether the neuron that provides the input is inhibitory or excitatory). To capture the complexity of bio-realistic neural networks, multiple edges may connect the same neurons. For example, cell-type–specific, diffuse modulatory signals may be additional ways to achieve inter-neuron communication, forming a multidigraph as in [25]. The cell-type–specific neuromodulators improve the efficiency of synaptic weight adjustments for task learning in neuronal networks, but such complexity is beyond the scope of this study. In the following, we assume simple directed, heterogeneous graphs.

Nodes of the graph are classified according to cell-type, which is often determined via electrophysiological, morphological, and transcriptomic measurements [20]. Our goal is to understand if there are topological properties, local or non-local, arising through connectivity patterns that are correlated with cell type. While certain simple local properties of diverse neural types are known, in terms of neighboring neurons [19], we are also interested in further network motifs. In this chapter, we show that higher-order connections are useful for classifying cell type, over and above using only knowledge of neighboring neurons that project onto the cell. Thus, knowledge from larger neighborhoods is informative of cell type, establishing a role for the corresponding network motifs. The network motifs closely associated with cell types may be further used to inform extensions of circuit models (as in Chapter 2) or ANN architectures (as in Chapter 3).

Recent efforts have produced vast amounts of data, describing both a variety of cell types and individual synapses. An ambitious future goal is to map the whole mouse brain connectome, a transformative project "to apply new and emerging tools to revolutionize our understanding of brain circuits" [1]. The motivation is that discovering the connectome, a complete matrix of structural connections between the nodes of a nervous system, would enable us to further link mind, brain, and behavior. It is not yet clear how to properly make use of the massive datasets we already have at our disposal, and fully take advantage of connectome data in the future. New computational tools are needed that can operate on networks to enhance our understanding of the brain's complexity. Reconsidering our neural network interpretation that models neural architectures as graphs suggests a novel and widely applied graph analysis method: Graph Neural Networks (GNNs) [52]. These are deep learning based methods that operate on graph domain. In this study, we exemplify this powerful computational tool on a graph that combines cell type and connectivity data to gain insights on network motifs associated with diverse cell types. Our analysis paves the way for future studies linking cell types, network motifs, and their functionality.

Our work is structured as follows: in section 2, we describe Graph Neural Networks, relatively novel neural models that capture node inter-dependence through connection data; in section 3, we describe a synthetic, but bio-realistic, dataset of neuronal activity and connectivity, simulated through a Generalized Leaky Integrate and Fire network model; in section 4, we classify cell-types using only connectivity data and show that introducing higher-order connectivity information through GNNs significantly improves cell type classification; in section 5, we redo the classification using neural activity data, in addition to connectivity data, and show once more how knowledge of second-order network motif can aid classification; in section 6, we use an explainability algorithm to associate cell types with pertinent network motifs. Finally, we draw conclusions in section 7.

4.2 Graph Neural Networks

A Graph Neural Network (GNN) is a class of artificial neural networks for processing data that can be represented as graphs. GNNs' use of pairwise message passing is a key design element, such that graph nodes iteratively update their state by exchanging information with their neighbors. By exploiting graph data containing rich relational information among elements, GNNs achieve great expressive power and have so far been successfully used for problems such as node classification, link prediction, and clustering. Inspired initially by CNNs and graph embedding algorithms, variants of GNNs are proposed to collectively aggregate information from graph structure, and thus model input consisting of features and their dependencies. Graphs are ubiquitous and thus can be used to model a large number of systems across diverse applications and fields of study. A large number of areas including physics, biology, social science, finance, and others use GNNs for, e.g., modeling physics systems, learning molecular fingerprints, predicting protein interface, classifying diseases, etc. GNNs can be used even in other domains where learning occurs on non-structural data like texts and images, where graphs are not explicit, though they have to rely on user-defined extracted structures (like the dependency trees of sentences and the scene graphs of images). Variants of GNNs such as graph convolutional networks (GCN), graph attention networks (GAT), and graph recurrent networks (GRN) have demonstrated ground-breaking performances on many deep learning tasks. This chapter will focus on GCN and GraphSAGE, two GNNs that we use to classify cell types. Multiple survey papers provide a detailed review of existing GNNs, present a general design pipeline, describe their applications, and propose problems for future research [53, 6, 51].

Let's denote a graph as $G = (V, E)$, where $|V| = N$ is the number of nodes in the graph and $|E| = N^e$ is the number of edges. Additionally, $\mathbf{A} \in R^{N \times N}$ is the adjacency matrix. We use $\mathbf{h}_v$ and $\mathbf{o}_v$ as the hidden state and output vector of node v. Further, let x_v be the input features of node $v \in V$. N_v denotes the neighborhood of some node $v \in V$, and e_{uv} are the features of edge $(u, v) \in E$.

A GNN's main computational building block is the message passing layer, which maps a

graph into an updated representation of the same graph. This module is used to propagate information between nodes so that the aggregated information could capture both feature and topological information. Using the notation above, the computation that occurs in the message passing layer can be expressed as:

$$\mathbf{h}_v = \phi(\mathbf{x}_v, \oplus_{u \in N_v} \psi(\mathbf{x}_v, \mathbf{x}_u, \mathbf{e}_{vu})) \tag{4.1}$$

where ϕ, ψ are differentiable functions and $\oplus$ is a permutation invariant aggregation operator (e.g., element-wise sum, mean, or max) that can accept an arbitrary number of inputs. In particular, ϕ and ψ are referred to as update and message functions, respectively. Intuitively, in a message passing computational block, graph nodes update their representations by aggregating the messages received from their neighbors.

Multiple message passing layers may be stacked, and the outputs of message passing are node representations h_v for each node $v \in V$ in the graph. These representations can then be connected downstream through a fully connected layer for any task of choice, such as node/graph classification or edge prediction. Graph nodes in a GNN update their representation by aggregating information from their immediate neighbors. Therefore stacking n message passing layers means that one node will be able to communicate with nodes that are at most n "hops" away. If our goal is to have every node receive information from every other node, the solution is to stack a number of layers equal to the graph diameter. However, stacking many message passing layers may cause issues such as every node representation becoming indistinguishable. Another issue refers to the bottleneck created by squeezing long-range dependencies into fixed-size representations. A wide range of solutions have been designed to address these problems [47, 24, 46]. We next present two "flavors" of message passing computational blocks that have been developed, such as graph convolutional networks (GCN) [23] and GraphSAGE [21], whose definitions can all be expressed in terms of the above formalism. Rough schematics of the ideas presented here are shown in Figure 4.2, both from the graph neural network literature (see [48, 21]).

GCN. GCNs are a generalization of convolutional neural networks to graph-structured data first introduced by Kipling and Welling in 2017 [23]. The formal (vectorized) expression of a GCN computational block is the following:

$$\mathbf{H} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\mathbf{W}) \tag{4.2}$$

where $\mathbf{H}$ is the matrix of node representations h_v, $\mathbf{X}$ is the matrix of node features x_v, $\sigma(\cdot)$ is an activation function (e.g., ReLU), $\tilde{\mathbf{A}}$ is the graph adjacency matrix with the addition of self-loops, $\tilde{\mathbf{D}}$ is the graph degree matrix with the addition of self-loops, and $\mathbf{W}$ is a matrix of trainable parameters. In particular, if $\mathbf{A}$ is the adjacency matrix, adding self loops results in $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$. Multiplying by $\tilde{\mathbf{D}}^{-\frac{1}{2}}$ to the left and right normalizes the adjacency matrix values and ensures that the eigenvalues of the matrix are between $[0, 1]$ in order to avoid numerical instabilities and exploding/vanishing gradients.

GraphSAGE. GraphSAGE [21] is a general inductive method that generates graph embeddings by sampling and aggregating features from a node's local neighborhood. A general formulation can be expressed as:

$$\mathbf{h}_{N_v}^{t+1} = AGG_{t+1}(\{\mathbf{h}_u^t, \forall u \in N_v\}) \tag{4.3}$$

$$\mathbf{h}_v^{t+1} = \sigma(\mathbf{W}^{t+1} \cdot [\mathbf{h}_v^t || \mathbf{h}_{N_v}^{t+1}]) \tag{4.4}$$

where AGG is an aggregation function, and the aggregators suggested are the mean aggregator, the LSTM aggregator, and the pooling operator.

A distinct property of GraphSAGE is that it uniformly samples a fixed-size set of neighbors to aggregate, instead of using all the neighbors. Sampling is ideal particularly if the graph is very large (for instance, too large to store in memory). This also alleviates the "neighbor explosion issue", by which if we apply multiple GNN layers, the size of the supporting neighbors grows exponentially with the depth of the network.

We use both GCN and GraphSAGE as ANNs. Given our directed, heterogeneous graph of neuronal connectivities (with cell and edge types), the type of problem we attempt to

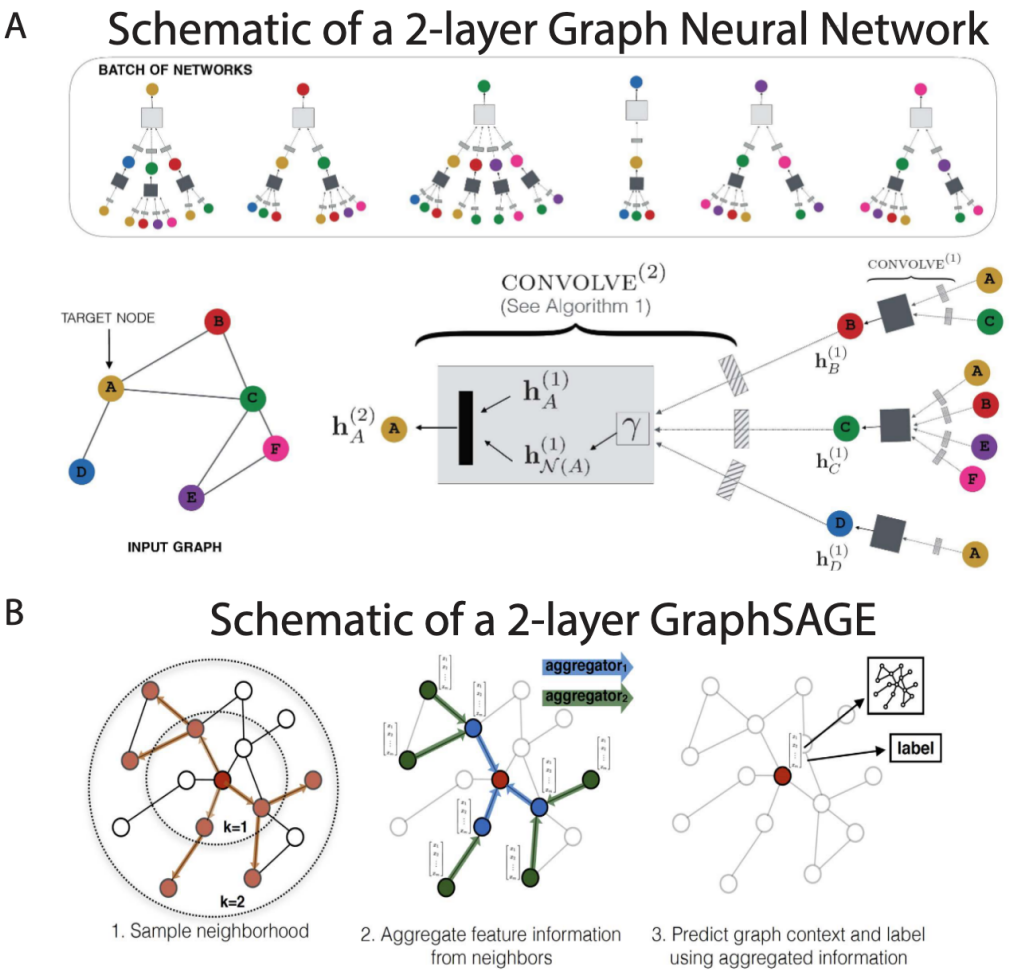


Figure 4.2: (a) Schematic of graph neural network from [48]. Top: These are neural networks that compute embeddings of each node of the input graph. While neural networks differ from node to node they all share the same set of parameters (i.e., the parameters of the convolve$^{(1)}$ and convolve$^{(2)}$ functions). Bottom: The 2-layer neural network that computes the embedding $h_A^{(2)}$ of node A using the previous-layer representation, $h_A^{(1)}$ of node A and that of its neighborhood $N(A)$ (nodes B, C, D). Boxes with the same shading patterns share parameters; γ denotes an importance pooling function; and thin rectangular boxes denote densely-connected multi-layer neural networks. (b) Schematic of GraphSAGE from [21]. First step is to sample nodes (first and second-order neighbors). Second is aggregating information from neighbors, usually by computing the average of the neighbor embeddings. Third, use the aggregated information to predict label (and graph context, not applicable for the problem we study here).

solve is performing node classification given only information about edges and neighbor cell types (sec. 4.4), or information about edges and neuron activity from all nodes (sec. 4.5). We do this in a supervised setting by providing labeled data for training. In the following, we describe the bio-realistic dataset we will use in more detail.

4.3 A synthetic dataset of neuronal activity and connectivity

We use a bio-realistic synthetic dataset that simulates the awake mouse V1. We use synthetic rather than "real" neural data because (1) we would like to first test our method in a setting where we have ground truth labels for the neuron cell types; (2) there is not currently enough detailed connectivity data to fully specify our graph. Given this, the mouse V1 simulation from Billeh et. al. [4] where neurons respond for $3s$ to drifting gratings is ideal for our GNN classification study.

To simulate realistic neuronal activities, the authors in [4] systematically integrated multimodal data about neuron types, connectivity, and sensory innervations to create the biologically realistic mouse V1 model. Mouse V1 specifically has substantial amounts of high-quality data, especially from standardized pipelines at the Allen Institute for Brain Science. The aim is to provide a computational platform to study bio-realistic cortical structure and computation. While two models with different levels of granularity were presented – one a more detailed, biophysically realistic model, and the other a point GLIF neuron model – they perform similarly at the level of firing rate distributions. We have used the point GLIF models for simplicity. All models, code, and meta-data are publicly available via the Allen Institute web portal at https://portal.brain-map.org/explore/models/mv1-all-layers. Models use the Brain Modeling Toolkit (BMTK) https://alleninstitute.github.io/bmtk which facilitates simulations with both NEURON and NEST, while supporting Python 2.7 and 3.6. Tutorials for BMTK and the models implemented in [4] are all available online at the links provided. The resulting simulations match in vivo extracellular recordings recorded from a standardized pipeline that is also freely available ([39], https://portal.brain-map.org/explore/circuits/visual-coding-neuropixels).

The simulation contains $\approx 230,000$ neurons from the V1 area of the mouse across a $845\mu m$-radius of the cortex. Although recent studies describe $50-100$ cell types in the V1, available neuronal models, in vivo recordings, and connectivity data prescribe only 17 classes. These classes are represented by 111 unique dynamical models for the GLIF network, i.e. 111 sets of parameters for the dynamics.

Synaptic connecitvity was determined using three design iterations: (1) constructing the feedforward geniculate input into the cortex; (2) introducing high synaptic recurrence, depending on the tuning stimulus of the cells; (3) refining the recurrent connectivity with respect to the stimulus tuning properties. We briefly describe all three model contributions below.

The V1 neurons receive inputs from the lateral geniculate nucleus (LGN) which is composed of spatio-temporal separable filters fitted to electrophysiology recordings. LGN input was carefully tuned such that physiological levels of direction selectivity were reproduced in V1 cells. Using a multi-step procedure, 17,400 LGN filters were instantiated in visual space and LGN-to-V1 connections were established. The resulting activity is compared to data from in vivo extracellular Neuropixels recordings from awake mice. A second step is to create the recurrent connectivity in the V1 network in a data-driven manner via extensive curation of the literature and Allen Institute data. Data used was on (1) connection probability (2) synaptic strengths (3) axonal delays (4) dendritic targeting of synapses. Connection probabilities took into account studies showing that excitatory-to-excitatory connections exhibit "like-to-like" preferences, i.e. cells preferring similar stimuli are preferentially connected. Besides connection probability, synaptic strength has also been shown to display like-to-like dependence. Axonal delay and dendritic targeting data was sparse, so it was used sparingly.

A heuristic optimization technique was used to fit synaptic weights using grid searches such that spontaneous and peak firing rates in response to a single trial of a drifting grating match experimental data and such that simulated activities are not epileptic. Readjustments of the synaptic connections were made by taking into account the functional rules of individual

recurrent connectivity to amplify direction selectivity. For excitatory-to-excitatory classes, a decrease of synaptic strength with distance in retinotopic space was also considered. Upon further refinement, the new connectivity rules enabled direction selectivity in a variety of neuronal populations while still obeying several empirical constraints. Using dynamics-based metrics, the simulated neural data maintains a strong match with experiments. After tuning of the model, authors are able to simulate stable responses to drastically different stimuli (flashes, natural movies, a looming disk).

The stimuli used for the simulation we analyze here are drifting gratings of eight different orientations $(0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°)$. We only utilize data on the cell type, connection type (excitatory and inhibitory), and activity. We choose a time window $\Delta t = 100$ms and average spikes in this window for every neuron in the dataset. This leads to a reduction in the number of features used for classification (from 3000-dimensional vector, where each datapoint is the presence or absence of a spike, to 30-dimensional vectors of real numbers where each datapoint is the average number of spikes in the Δt time window).

In the following, we present in detail the results from solving a supervised learning problem to classify cell types given different input features.

4.4 Cell-type classification using only connectivity data

The first problem we tackle is inferring cell type for each neuron in the graph given information about all other cell types and given connectivity information. If there is structurally-rich information in the graph in terms of an over-abundance of network motifs that include a particular cell type, our cell type classification should be able to detect these connectivity patterns and improve accuracy.

4.4.1 Methods

The features we select for classification with GNNs are 21-dimensional one-hot encodings of the cell type, corresponding to the 21 cell types in the dataset. We also provide the edges which are connectivities between cell types. We use a GCN with two layers for this

problem, with ReLU activation functions and a small amount of dropout to prevent overfitting (see table for hyperparameters chosen in table 4.2). In order to make this problem non-trivial, we remove the self-loops in the GCN, therefore the equation corresponding to the feedforward pass is given by:

$$\mathbf{H} = \sigma(\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\mathbf{X}\mathbf{W}) \tag{4.5}$$

where notation remains the same as that in sec. 4.2, Eq. 4.2. After the first iteration, the hidden units approximately represent a linear transformation to a vector containing the average number of neighbors of each cell type followed by a non-linearity (i.e. a linear transformation to a 21-dimensional vector where every entry represents how many cells of a particular cell type there are compared to the total number of neighbors followed by the application of σ). The adjacency matrix is such that the first order *neighbors* are the neurons that have inputs to the neuron we want to classify. This is in contrast to having as neighbors the neurons which the unit to be classified projects towards.

The first issue we encounter is that there is considerable class imbalance. For instance, the excitatory populations e23Cux2 and e6Ntsr1 are over-represented, each making out 25% of the population. We address this by adding weights for each cross-entropy term in our loss function. The weight is inversely proportional to the class frequency and multiplies the cross-entropy term corresponding to that class. There are three ways we considered to weigh the cross-entropy loss: (1) inverse of number of samples (INS) (2) inverse of Square Root of Number of Samples (ISNS) and (3) effective number of samples (ENS).

INS weighs the samples as the inverse of the class frequency for the class they belong to:

$$w_{n,c} = \frac{1}{\text{number of samples in class c}} \tag{4.6}$$

ISNS weighs the samples as the inverse of the Square Root of class frequency for the class they belong to:

$$w_{n,c} = \frac{1}{\sqrt{\text{number of samples in class c}}} \tag{4.7}$$

The ENS weighting scheme was introduced in the CVPR'19 paper by Google: Class-Balanced Loss Based on Effective Number of Samples [11]. Unlike INS and ISNS, this weighting scheme relies on the "Effective Number of Samples":

$$w_{n,c} = \frac{1}{E_{n,c}} \tag{4.8}$$

$$E_{n,c} = \frac{1-\beta^{n_c}}{1-\beta} \tag{4.9}$$

where $E_{n,c}$ represents the effective number of samples. The authors suggest trying different values for β: $0.9, 0.99, 0.999$.

We experimented with different weighing schemes and settled for ISNS. We also scaled the ISNS vector by multiplying it with a scalar; after a careful search for the best scalar, we chose to weigh the cross-entropy loss via $10 \cdot \text{ISNS}$.

Because of class imbalance, we decided to report F1 scores in addition to accuracy. F1 scores are a more reliable metric to decide how well our network classifies cell type. The F1 score is given by $\frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision}+\text{recall}}$, where precision is defined by $\frac{\text{true positive}}{\text{true positive}+\text{false positive}}$ and recall by $\frac{\text{true positive}}{\text{true positive}+\text{false negative}}$.

A thorough grid search was performed before deciding on a proper hyperparameter set (results not shown). The hyperparameters were chosen after training on a subset of our graph that was obtained by sampling 10% of the nodes and considering all the connections between these sub-sampled nodes. Once hyperparameters were fixed, we trained on sampled subgraphs that were 10% of the entire graph. We then tested our model on a test set that was 10% the graph size for the results presented below. This procedure was necessary because of the large size of the graph which made training very slow. Sampling sub-graph batches as in [21] was also unsuccessful, as the learning failed to converge. Our solution

proved to be fast and easy to implement.

We compared learning using the ADAM optimizer with different learning rates and weight decay. We found that setting the learning rate at its default, 0.001, and weight decay at 0, we obtained the best results.

We compare the results for training this network with the accuracies and F1 scores for training a feedforward neural network. The network's hyperparameters were chosen to match that of the GCN described above for an objective comparison. However, we also searched for other hyperparameters, without superior results (not shown). The ANN is trained using 21-dimensional features representing the average number of neighbors of a particular cell type (neurons projecting *towards* the neuron we are trying to classify, as before). This is similar to training with the GCN, except at the second iteration the GCN also considers neighbors that are 2 hops away. GCN takes into account second-order neighbors, unlike the ANN which simply applies linear and non-linear transformations to a vector corresponding to a node's first-order connections.

A summary of the hyperparameters chosen for the network architectures and the training is shown in the table below:

hyperparameters	**GCN**	feedforward **ANN**
hidden layers	2	2
hidden dimensions	$[64, 64]$	$[64, 64]$
learning algorithm	ADAM	ADAM
learning rate	0.001	0.001
weight decay	0	0
dropout	0.1	0.1
batch size	–	512
epochs	2000	2000

Table 4.1: Hyperparameters and optimization algorithms used for cell type classification using a graph neural network (GCN) and a feedforward artificial neural network. Training/testing uses only edge and cell type information (no activity). The same hyperparameters are used.

4.4.2 Results

After training the GCN and the feedforward ANN (see schematics of how this is done in 4.3) on 10 different sub-graphs and testing, we find that GCN has superior accuracies and F1 scores when classifying cell types. The mean F1 score for the GCN is 0.57, while for the ANN the mean F1 score is 0.28 (pvalue=0.008 $<$ 0.05, Kolmogorov-Smirnov test). The mean accuracy for the GCN is 70%, while the mean accuracy for the ANN is 64% (pvalue $=$ 0.079, Kolmogorov-Smirnov test). Histograms of the F1 score and accuracy distributions are shown in Figure 4.4. While the difference in testing accuracy is small and not statistically significant, the difference in F1 scores is large and highlights the GCN as a preferable method for classifying cell types.

4.5 Cell-type classification using activity and connectivity data

The second problem we tackle is inferring cell type for each neuron in the graph given activity and connectivity information. In this case, we exclude all cell-type information

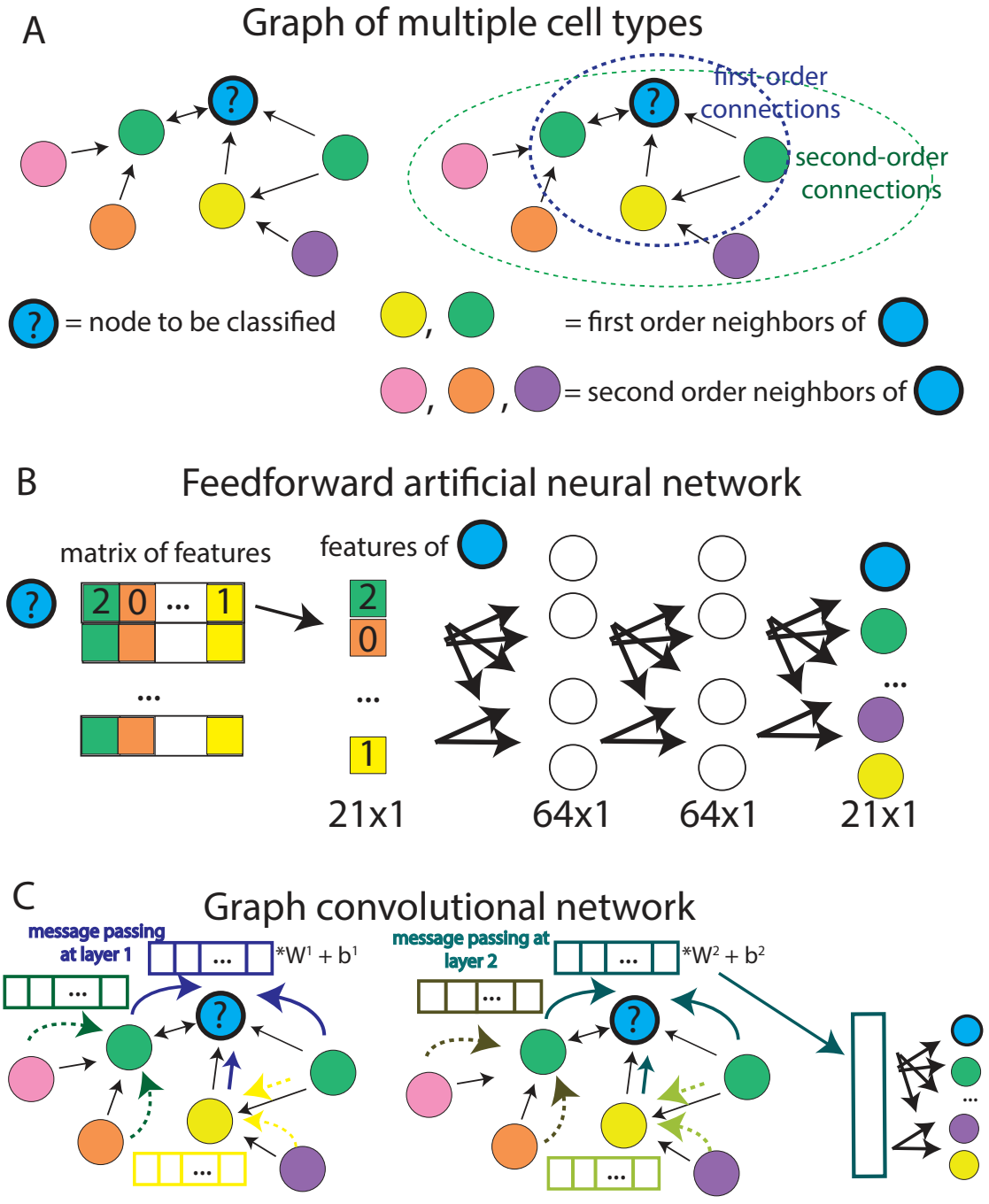


Figure 4.3: (a). Schematic graph with blue node to be classified and its first and second-degree connections. (b). ANN used for node classification. Input is a vector of features indicating the number of neighbors of a certain cell type. (c). Schematic of graph convolutional network.

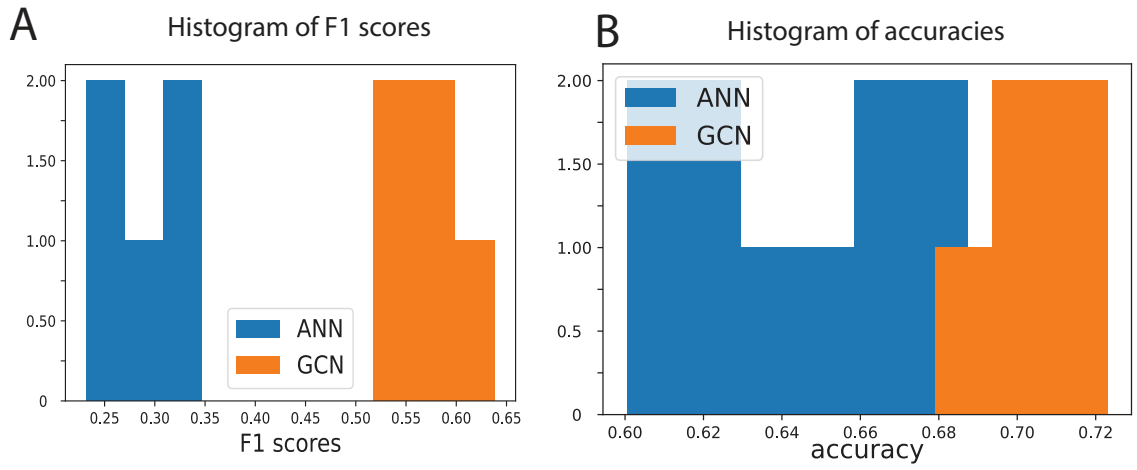


Figure 4.4: (a) Histogram of F1 scores using ANN (blue), using the GCN (orange). (b) Histogram of accuracies using ANN (blue), using the GCN (orange).

from training.

4.5.1 Methods

The features we select for each neuron are the activities corresponding to presentations (or viewings) of drifting gratings. There are 3 viewings for each drifting grating, and 8 drifting gratings corresponding to grating orientations in 45° increments. Each activity vector is a 30×1-dimensional vector where each entry is the average number of spikes over a 100 ms time window, reduced from a 3000×1-dimensional vector representing the activity over 3s. We test a few possible feature vectors: (1) concatenating the activity vectors corresponding to each viewing of a drifting grating (720 features); (2) averaging the activity vectors over repeated viewings of the same grating (i.e. the 3 viewings), then concatenating the resulting vectors for each different grating orientation (240 features); (3) summing the spikes to obtain the total number of spikes for each grating (8 features), then adding a computed orientation selectivity index as an additional feature as per [32]. We tested how a feedforward ANN can be trained to classify using each of these possible feature vectors and decided to use

the concatenated vector with all the activity data (720 features) because the classification accuracy was the highest.

Instead of a GCN as in the previous classification, we use GraphSAGE, another graph neural network described in detail in [21]. The GraphSAGE network we use has 4 hidden layers and the number of hidden dimensions is 100, with a dropout set to 0.1. We use the default activation function, ReLU. GraphSAGE implements the following operation at the first hidden layer:

$$\mathbf{h}_i = \sigma(\mathbf{W}_1 x_i + \mathbf{W}_2 \text{mean}_{j \in N(i)} x_j) \tag{4.10}$$

where x_i is a node (neuron) representing activities, x_j are neighboring nodes (neurons) that x_i receives input from, $\mathbf{W}_1$ and $\mathbf{W}_2$ are weight matrices to be learned, and σ is the ReLU non-linearity. Operations at the next layer are precisely analogous: to compute the activation of a hidden layer unit $\mathbf{h}_i$, a linear transformation is applied to the corresponding hidden unit from the previous layer. This is added to a linear transformation of the mean of neighboring hidden layer activations from the previous layer, followed by a non-linearity σ.

We largely follow the Methods outlined in 4.4.1. As before, we solve the problem of class imbalance by weighing the terms in the cross-entropy loss, using ISNS (see 4.4.1). In this setting, no scaling of the ISNS was necessary. F1 scores and accuracies have been computed, with F1 scores the more reliable metric because of class imbalance. We followed the same procedure for training, testing, and choosing the hyperparameters for our problem.

We compared our results for training this network to the F1 scores and accuracies obtained from training a feedforward ANN. We tried multiple hyperparameters for training this ANN and ultimately found that a 5-layer ANN with $2000, 1000, 500, 100$, and 50 hidden units was the most accurate and provided the highest F1 scores. The ANN is trained using only activity data while omitting the connectivity information, as shown in (Figure 4.5A). Meanwhile, the GraphSAGE has 4 hidden layers and so it takes into account the first, sec-

ond, third, and fourth-order neighbors.

We opted to use GraphSAGE because GCN did not outperform the ANN. This might be because GCN roughly averages the neighboring features along with features of the unit to be classified. In the case of activity data, this can be detrimental as we may be weakening important features from the neuron we are classifying while equally considering activities that may be inversely correlated with these features (so far we are not making any distinction between excitatory and inhibitory connections).

A summary of hyperparameters we are using is shown below:

hyperparameters	**GraphSAGE**	feedforward **ANN**
hidden layers	4	5
hidden dimensions	$[100, 100, 100, 100]$	$[2000, 1000, 500, 100, 50]$
learning algorithm	ADAM	ADAM
learning rate	0.001	0.001
weight decay	0	0
dropout	0.1	0.1
batch size	–	512
epochs	2000	2000

Table 4.2: Hyperparameters and optimization algorithms used for cell type classification using a graph neural network (GraphSAGE) and a feedforward artificial neural network (ANN). Training/testing uses both connectivity and activity information, and no cell type information. Almost the same hyperparameters are used.

4.5.2 Results

GraphSAGE is more successful at classifying cell types than the feedforward ANN (see schematics of these two methods in Figures 4.5A and B). After training on 10 different

samples from the graph and verifying our results on the test set, we obtained that the mean F1 score for GraphSAGE is 0.82, while for the ANN the mean F1 score is 0.47 (pvalue = $4e-05$, Kolmogorov-Smirnov test). The mean accuracy for GraphSAGE is 70%, while the mean accuracy for the ANN is 34% (pvalue = $4e-05$, Kolmogorov-Smirnov test). Histograms of the F1 score and accuracy distributions are shown in Figures 4.5 C and D.

GraphSAGE has access to more information, given that it receives connectivity data as input. The ANN has an order of magnitude more parameters than GraphSAGE ($3,996,050$ versus $188,200$), but cannot be competitive as connectivity information fails to be used in a very efficient way. More controls need to be tested to establish GraphSAGE as a generally superior method, a topic for future experiments.

4.6 Conclusions

While ambitious new programs (e.g. the BRAIN Initiative) target different species and brain areas with the goal of imaging, recording, and amassing new connectivity information at different levels of resolution, it is still unclear which tools and intuitions from data and network science can be successfully applied to make sense of all this data. A significant difficulty is understanding how the heterogeneity we see through numerous cell types and the structural wiring present through connectivity patterns support the computations necessary to enable the brain's cognitive processes.

In this chapter we study how higher-order connectivity information can be exploited to classify cell types. We use this connectivity information in a very specific way by employing graph neural networks which preserve graph topological properties while learning to perform a given task. GNNs are part of a nascent sub-field of deep learning dealing with various graph-related tasks such as graph classification and graph representation learning. It is straightforward to view neuronal networks with varying cell types and inter-connectivity patterns as heterogeneous, directed graphs; this makes GNNs well-suited machine learning tools for node or graph classification. Applying deep learning methods that are not graph-based (e.g. CNNs) directly to neuronal networks overlooks the relationship between

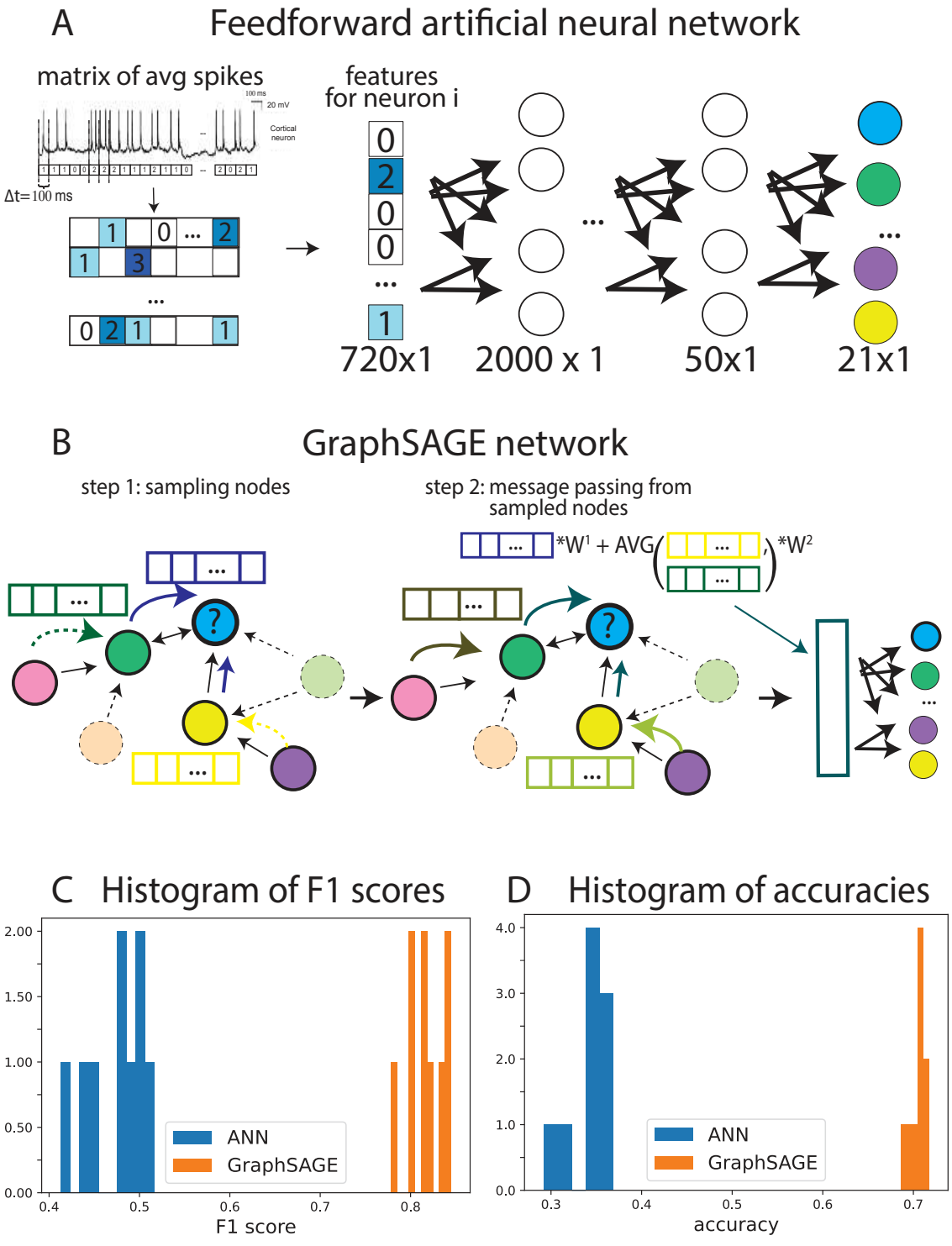


Figure 4.5: (a) Schematic of feedforward neural network. (b) Schematic of GraphSAGE. (c) Histogram of F1 scores using ANN (blue), using the GraphSAGE (orange). (d) Histogram of accuracies using ANN (blue), using the GraphSAGE (orange).

neurons and their connections, which causes an important loss of topological properties inherently encoded in a graph representation [3]. So far, GNNs have been successfully applied in a small set of problems pertaining to network neuroscience and medical image processing where functional MRI and diffusion MRI datasets were employed, often for disease or biomarker prediction. For instance, graph generative models are capable of filling in missing observations given that real-world connectomic datasets are usually incomplete. Similar to this problem, several studies have used generative adversarial networks (GANs) – a machine learning method that learns to generate new data with the same statistics as the training set – to perform graph prediction, e.g. by predicting a structural brain graph from a functional one [50]. Other graph generative models are able to find a representative template of a population of brain multigraphs with shared neurological state (e.g., brains with Alzheimer's). This generated template encodes a holistic mapping of common characteristics shared within populations of brain multigraphs [13]. A careful review of the 30 studies found between 2017-2020 that apply GNNs to brain data is presented in [3].

To the best of our knowledge, GNNs have not been applied at the level of resolution of single neurons and synapses as here. Our eventual goal is to relate cell types to different network motifs, to then be used across computational models and artificial architectures. We choose to classify cell types given higher order connectivity information and determine that this is significant for increased classification effectiveness. Our next steps will attempt to leverage GNN explainability methods in order to find connections that are most significant for the cell type classification. This points to a method that maps cell types to different connectivity patterns. The association between cell types and network motifs can be further used in computational models and in designing artificial network architectures. These designed architectures may be endowed with the computational properties of their biological counterparts. This is akin to how our bottleneck-switching network (see Chapter 3) was effective in solving a continual learning problem after adding switching units to the network architecture. An allied insight is that connecting these units with the same (recurrent) network motif as that for the VIP neuron population (Chapter 2) was necessary to make the network successful at the task. Our proposed framework may provide us with

helpful insights to construct next-generation architectures based on cell types.

Overall, especially as the computational neuroscience field has been relatively slow to pay attention to GNNs, this study raises attention to this evolving field of deep learning, with promising applications that may enhance our understanding of structural properties of artificial and biological neural networks.

Bibliography

[1] LF Abbott, DD Bock, EM Callaway, W Denk, C Dulac, AL Fairhall, I Fiete, KM Harris, M Helmstaedter, V Jain, N Kasthuri, Y LeCun, JW Lichtman, PB Littlewood, L Luo, JHR Maunsell, RC Reid, BR Rosen, GM Rubin, TJ Sejnowski, HS Seung, K Svoboda, DW Tank, D Tsao, and DC Van Essen. The mind of a mouse. Cell, 182(6):1372–1376, 2020.

[2] U Alon. Network motifs: theory and experimental approaches. Nature Reviews Genetics, 8(6):450–461, 2007.

[3] Alaa Bessadok, Mohamed Ali Mahjoub, and Islem Rekik. Graph neural networks in network neuroscience. ArXiv, abs/2106.03535, 2021.

[4] YN Billeh, B Cai, SL Gratiy, K Dai, R Iyer, NW Gouwens, R Abbasi-Asl, X Jia, JH Siegle, SR Olsen, C Koch, S Mihalas, and A Arkhipov. Systematic integration of structural and functional data into multi-scale models of mouse primary visual cortex. Neuron, 106(3):388–403, 2020.

[5] SR Bittner, RC Williamson, AC Snyder, A Litwin-Kumar A, B Doiron B, SM Chase, MA Smith, and BM Yu. Population activity structure of excitatory and inhibitory neurons. PLoS One, 12(8), 2017.

[6] MM Bronstein, J Bruna, Y LeCun, A Szlam, and P Vandergheynst. Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine, 34(4):18–42, 2017.

[7] D Burnham, E Shea-Brown, and S Mihalas. Learning to predict in networks with heterogeneous and dynamic synapses. bioRxiv, 2021.

[8] L Campagnola, SC Seeman, T Chartrand, L Kim, A Hoggarth, C Gamlin, S Ito, J Trinh, P Davoudian, C Radaelli, MH Kim, T Hage, T Braun, L Alfiler, J Andrade, P Bohn, R Dalley, A Henry, and S Kebede S. Local connectivity and synaptic dynamics in mouse and human neocortex. Science, 375(6585), 2022.

[9] H Choi and S Mihalas. Synchronization dependent on spatial structures of a mesoscopic whole-brain network. PLoS computational biology, 15(4), 2019.

[10] J Cornford, D Kalajdzievski, M Leite, A Lamarquette, DM Kullmann, and B Richards. Learning to live with dale's principle: Anns with separate excitatory and inhibitory units. bioRxiv, 2021.

[11] Y Cui, M Jia, TY Lin, Y Song, and S. Belongie. Class-balanced loss based on effective number of samples. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), (9260-9269), 2019.

[12] AK Dey, YR Gel, and HV Poor. What network motifs tell us about resilience and reliability of complex networks. Proceedings of the National Academy of Sciences, 116(39):19368–19373, 2019.

[13] S Dhifallah, I Rekik, and Alzheimer's Disease Neuroimaging Initiative. Estimation of connectional brain templates using selective multi-view network normalization. Medical image analysis, 59, 59, 2020.

[14] M Dipoppa, A Ranson, M Krumin, M Pachitariu, M Carandini, and KD Harris. Vision and locomotion shape the interactions between neuron types in mouse visual cortex. Neuron, 98(3):602–615, 2018.

[15] B Doty, S Mihalas, A Arkhipov, and A Piet. Heterogeneous "cell types" can improve performance of deep neural networks can improve performance of deep neural networks. bioRxiv, 2021.

[16] R Duarte and A Morrison. Leveraging heterogeneity for neural computation with fading memory in layer 2/3 cortical microcircuits. PLOS Computational Biology, 15(4):1–43, 04 2019.

[17] J Gjorgjieva, G Drion, and E Marder. Computational implications of biophysical diversity and multiple timescales in neurons and synapses for circuit performance. Current Opinion in Neurobiology, 37:44–52, 2016. Neurobiology of cognitive behavior.

[18] TE Gorochowski, CS Grierson, and M di Bernardo. Organization of feed-forward loop motifs reveals architectural principles in natural and engineered networks. Science Advances, 4(3):eaap9751, 2018.

[19] NW Gouwens, SA Sorensen, F Baftizadeh, A Budzillo, BR Lee, T Jarsky, L Alfiler, K Baker, E Barkan, K Berry, D Bertagnolli, K Bickley, J Bomben, T Braun, K Brouner, T Casper, K Crichton, TL Daigle, R Dalley ..., and H Zeng. Integrated morphoelectric and transcriptomic classification of cortical gabaergic cells. Cell, 183(4):935–953, 2020.

[20] NW Gouwens, SA Sorensen, J Berg, C Lee, T Jarsky, J Ting, SM Sunkin, D Feng, CA Anastassiou, E Barkan, K Bickley, N Blesie, T Braun, K Brouner, A Budzillo, S Caldejon, T Casper, D Castelli, P Chong ..., and C Koch. Classification of electrophysiological and morphological types in mouse visual cortex. Nature Neuroscience, 22(7):1182–1195, 2019.

[21] W Hamilton, Z Ying, and J Leskovec. Inductive representation learning on large graphs. Advances in neural information processing systems, 30, 2017.

[22] N Kashtan, S Itzkovitz, R Milo, and U Alon. mfinder, a software tool for network motifs detection.

[23] TN Kipf and M Welling. Semi-supervised classification with graph convolutional networks. arXiv preprint, 2016.

[24] Y Li, D Tarlow, M Brockschmidt, and R Zemel. Gated graph sequence neural networks. arxiv, 2016.

[25] YH Liu, S Smith, S Mihalas, E Shea-Brown, and U Sümbül. Cell-type specific neuromodulation guides synaptic credit assignment in a spiking neural network. Proceedings of the National Academy of Sciences, 118(51):e2111821118, 2021.

[26] RR Llinás. The contribution of santiago ramón y cajal to functional neuroscience. Nat Rev Neuroscience, 4(1):77–80, 2003.

[27] CW Lynn and DS Bassett. The physics of brain network structure, function, and control. Nature Review Physics, 1:318–332, 2019.

[28] A Ma'ayan, GA Cecchi, J Wagner, AR Rao, R Iyengar, and G Stolovitzky. Ordered cyclic motifs contribute to dynamic stability in biological and engineered networks. Proceedings of the National Academy of Sciences, 105(49):19235–19240, 2008.

[29] S Mangan and U Alon. Structure and function of the feed-forward loop network motif. Proceedings of the National Academy of Sciences, 100(21):11980–11985, 2003.

[30] S Mangan, S Itzkovitz, A Zaslaver, and U Alon. The incoherent feed-forward loop accelerates the response-time of the gal system of escherichia coli. Journal of Molecular Biology, 356(5):1073–1081, 2006.

[31] AH Marblestone, G Wayne, and KP Kording. Toward an integration of deep learning and neuroscience. Frontiers in Computational Neuroscience, 10, 2016.

[32] M Mazurek, M Kager, and SD Van Hooser. Robust quantification of orientation selectivity and direction selectivity. Frontiers in Neural Circuits, 8, 2014.

[33] R Milo, S Itzkovitz, N Kashtan, R Levitt, Shen-Orr, I Ayzenshtat, M Sheffer, and U Alon. Superfamilies of evolved and designed networks. Science, 303(5663):1538–1542, 2004.

[34] K Padmanabhan and NN Urban. Intrinsic biophysical diversity decorrelates neuronal firing while increasing information content. Nature Neuroscience, 13(10):1276–82, 2010.

[35] N Perez-Nieves, VCH Leung, PL Dragotti, and DFM Goodman. Neural heterogeneity promotes robust learning. Nature Communications, 12(1):5791, 2021.

[36] CK Pfeffer, M Xue, M He, ZJ Huang, and M Scanziani. Inhibition of inhibition in visual cortex: the logic of connections between molecularly distinct interneurons. Nature Neuroscience, 16(8):1068–1076, 2013.

[37] RJ Prill, PA Iglesias, and A Levchenko. Dynamic properties of network motifs contribute to biological network organization. PLOS Biology, 3(11):null, 10 2005.

[38] J Qian, A Hintze, and C Adami. Colored motifs reveal computational building blocks in the c. elegans brain. PLOS ONE, 6(3):1–8, 03 2011.

[39] JH Siegle, X Jia, S Durand, S Gale, C Bennett, N Graddis, G Heller, TK Ramirez, H Choi, JA Luviano ..., and C Koch. Survey of spiking in the mouse visual system reveals functional hierarchy. Nature, 592:86–92, 2021.

[40] C Stöck, D Lang, and W Maass. Probabilistic skeletons endow brain-like neural networks with innate computing capabilities. bioarxiv, 2021.

[41] B Tasic, Z Yao, LT Graybuck, KA Smith, TN Nguyen, D Bertagnolli, J Goldy, E Garren, MN Economo, S Viswanathan, O Penn, T Bakken, V Menon, J Miller, O Fong, KE Hirokawa, K Lathia, C Rimorin, M Tieu, ..., and H Zeng. Shared and distinct transcriptomic cell types across neocortical areas. Nature, 563:72–78, 2018.

[42] C Teeter, R Iyer, V Menon, N Gouwens, D Feng, J Berg, A Szafer, N Cain, H Zeng, M Hawrylycz, C Koch, and S Mihalas. Generalized leaky integrate-and-fire models classify multiple neuron types. Nature Communications, 9(709), 2018.

[43] J Trousdale, Y Hu, E Shea-Brown, and K Josić. Impact of network structure and cellular response on spike time correlations. PLOS Computational Biology, 8(3):1–15, 03 2012.

[44] K Uzel, S Kato, and M Zimmer. A set of hub neurons and non-local connectivity features support global brain dynamics in c. elegans. Current Biology, 2022.

[45] CN Winston, D Mastrovito, E Shea-Brown, and S Mihalas. Heterogeneity in neuronal dynamics is learned by gradient descent for temporal processing tasks. bioRxiv, 2022.

[46] K Xu, C Li, Y Tian, T Sonobe, K Kawarabayashi, and S Jegelka. Representation learning on graphs with jumping knowledge networks. arxiv, 2018.

[47] K Xu, M Zhang, S Jegelka, and K Kawaguchi. Optimization of graph neural networks: Implicit acceleration by skip connections and more depth. arxiv, 2021.

[48] R Ying, R He, K Chen, P Eksombatchai, WL Hamilton, and J Leskovec. Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining, KDD '18, pages 974–983, New York, NY, USA, 2018. Association for Computing Machinery.

[49] F Zeldenrust, B Gutkin, and S Denéve. Efficient and robust coding in heterogeneous recurrent networks. PLOS Computational Biology, 17(4):1–27, 04 2021.

[50] L Zhang, L Wang, and D Zhu. Recovering brain structural connectivity from functional connectivity via multi-gcn based generative adversarial network. International Conference on Medical Image Computing and Computer-Assisted Intervention, pages 53–61, 2020.

[51] S Zhang, H Tong, J Xu, and R Maciejewski. Graph convolutional networks: a comprehensive review. Computational Social Networks, 6:1–23, 2019.

[52] J Zhou, G Cui, S Hu, Z Zhang, C Yang, Z Liu, L Wang, C Li, and M Sun. Graph neural networks: A review of methods and applications. AI Open, 1:57–81, 2020.

[53] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. AI Open, 1:57–81, 2020.

Chapter 5

CONCLUSION

Our work brings together three approaches to neural network architecture discovery, starting from the premise that using different building blocks like cell types and structural patterns (network motifs) can enhance the range of computations a circuit can implement. While the precise relationship between structure and function in biological and artificial networks still eludes neuroscientists and machine learning experts alike, a number of prior studies link heterogeneous motifs with different node types to particular computational functions (e.g., [4, 2, 1, 3]). For biological networks, these findings may be tied to the brain area where the network motif is embedded; furthermore, it is now clear that both topological and functional information – related to the neuronal dynamics that the physical circuitry supports – is crucial to understanding the computational role of specific motifs. Despite these challenges, here we successfully introduced cell types and searched for the appropriate architectures to enable specific computations, such as context adaptation and switching. Specifically, we identified structural motifs with the desired properties both in a biologically-based circuit model (Chapter 2) and a more abstracted artificial neural network model (ANN, Chapter 3).

In **Chapter 2** we study how different but distinct computations can be implemented in the same neural tissue. We show that a biologically inspired microcircuit with multiple inhibitory cell types can switch between visual processing of data in two contexts: static and moving. Specifically, the VIP cell type population in our model turns ON whenever the animal initiates moving and is OFF otherwise. The VIP neurons modulate the visual circuit through a recurrent motif, involving the PYR population and (through a disinhibitory motif) the SST population. We constructed this circuit in a step-by-step manner by first considering the network performing visual processing in the static context; we then added

the minimal number of components (cells, connections) to ensure the circuit can switch to process the moving context.

In **Chapter 3**, we take inspiration from the V1 microcircuit to implement an abstracted version of the underlying network motif in an ANN. The goal is to enable ANNs to be more flexible, as tested by the ability to sequentially switch from classifying digits set on backgrounds (or contexts) of different statistics. In particular, sequential context switching should be possible without forgetting to classify digits set on the first context, after experience learning the second context. When we augment the ANN layers with switching units inspired by the VIP cell type, the network can indeed reliably perform the sequential task without forgetting. Moreover, the same minimally recurrent motif that was implemented in V1 is also necessary in the ANN. We called the resulting ANN with recurrently connected switching units the "bottleneck-switching network."

Chapter 4 takes a step back from networks' computational properties. Our main result here confirms that higher-order connectivity patterns inform cell identity. We find this by using a graph neural network (GNN) to classify cell types based on their connectivity. Studies so far classifying neurons into their different types have found that cell type is determined by morphological, electrophysiological, and transcriptomic properties. We find that connectivity patterns surrounding the cell are also highly relevant and correlate highly with type. The resulting tool enables future work that can link cell type with relevant network motifs by employing GNN explainability methods. Our ultimate goal is to implement these cell types and network motifs as part of further ANN structures, taking the bottleneck-switching network study of Chapter 3 as a blueprint.

Our outlook toward future work begins by noting that, despite impressive and ongoing advances, current artificial intelligent systems still do not display many of the sophisticated capabilities our human minds possess. For instance, the remarkable flexibility displayed by the brain to switch tasks and contexts is absent or limited in current artificial systems, which frequently require large amounts of data to learn very specific tasks. Reverse engineering the brain's computational capabilities can shed light on principles of network architecture design and provide us with inspiration for next-generation artificial intelligent machines.

Including cell type information into our models and network architectures adds biological realism, but it also enables biological and artificial networks alike to acquire novel functionalities. Our hope is that a diverse set of computations can emerge in circuits whose building blocks and logic follow quantifiable, biologically inspired rules.

In this light, Chapter 4 shows that higher-order connectivity motifs are linked to cell type, because knowledge of these motifs improves classification of cell types. This said, more work is needed to identify the specific higher-order connectivity patterns that are helpful for cell type classification. We offer here a step-by-step outline of future directions to extract such important network motifs.

1. We have already used multi-layer GNNs to classify cell types given connectivity data from a larger well-studied network. Following this,

2. We will use GNNExplainer [5], or some other GNN explainability method from [6] to find the connections important for the classification. GNNExplainer is an explainability method that learns soft masks for edges and node features in order to rank the most important edges and features for solving a prediction problem. In more detail: First, the masks are randomly initialized and treated as trainable variables. Then, masks are used with the original feature and adjacency matrices via element-wise multiplication. Next, the masks are optimized so that the GNN with masked features/edges has the same performance as before the mask was applied. A regularization term is added to the optimization to maximize the sparsity of the masks, which makes the solution non-trivial (otherwise the trivial solution is not masking any edge or feature). After the soft masks are learned, a threshold can be applied to find the most important edges/features. If y is the output of the GNN Φ and Y is the output random variable, c are possible classes, A is the adjacency matrix, X is the feature matrix, and $G = (A, X)$ is the graph, then the simplified optimization loss (without regularization) applied to learn the mask M can be expressed as:

$$\text{loss} = \min_{M} - \sum_{c=1}^{C} 1[y = c] P_{\Phi}(Y = y | G = (A \odot \sigma(M), X)) \tag{5.1}$$

GNNExplainer provides a mask for every node in the graph and this mask shows which ones are the most important edges for *that* prediction. **3.** Following the methods outlined in [5], we consider explanations for each class separately and find the most important sub-graphs (i.e. network motifs) that explain each cell type. It is likely that domain knowledge of important connectivity patterns and cell types will be critical to filter and select relevant network motifs at this step.
4. These network motifs can then be tested in a model or ANN. This application could follow the framework of the bottleneck-switching network from Chapter 3.

Above, we have outlined a framework that closes the gap between our exploratory work in Chapter 4 – that aims to do network motif discovery – and our more applied work in Chapters 2 and 3 that seeks to find minimally-complex circuit architectures to switch contexts in two different tasks. This step-by-step procedure could guide future artificial neural network architectures, helping to prescribe their building blocks. In particular, our framework seeks to include structures that are minimally complex while being able to capture important computational mechanisms, through cell types and network motifs. Insights from neuroscience experiments will remain crucial because these guide our intuition regarding which structures are essential for certain computational functions – insight we relied on significantly in Chapters 2 and 3. New data, including connectivity data from the mouse and fruit fly, will serves as valuable resources, via detailed information at the single-neuron resolution, including knowledge of cell types. These datasets are complemented by a rich and evolving literature that describes the neuronal circuits' functional properties and mechanisms, sharpening our intuition about the role of different connectivity patterns. Applying graph neural network explainability methods can point to important first and higher-order connections for cell type classification and thus link certain cell types to motifs. Introducing cell types in our architecture designs without also considering the connectivity patterns associated with these types may lead to circuits that cannot reproduce the computational capabilities that their biological counterparts can. In sum, our work can be viewed as a first step to automate the process of network motif discovery in order to embed these structural

patterns in our models and artificial networks, leading to a better understanding of the computational principles these architectural units facilitate and to leading to new horizons in network computation.

Bibliography

[1] AK Dey, YR Gel, and HV Poor. What network motifs tell us about resilience and reliability of complex networks. Proceedings of the National Academy of Sciences, 116(39):19368–19373, 2019.

[2] TE Gorochowski, CS Grierson, and M di Bernardo. Organization of feed-forward loop motifs reveals architectural principles in natural and engineered networks. Science Advances, 4(3):eaap9751, 2018.

[3] RJ Prill, PA Iglesias, and A Levchenko. Dynamic properties of network motifs contribute to biological network organization. PLOS Biology, 3(11):null, 10 2005.

[4] J Qian, A Hintze, and C Adami. Colored motifs reveal computational building blocks in the c. elegans brain. PLOS ONE, 6(3):1–8, 03 2011.

[5] R Ying, D Bourgeois, J You, M Zitnik, and J Leskovec. Gnnexplainer: Generating explanations for graph neural networks. Advances in neural information processing systems, 32:9240–9251, 2019.

[6] H Yuan, H Yu, S Gui, and S Ji. Explainability in graph neural networks: A taxonomic survey. ArXiv, abs/2012.15445, 2020.