Learning Neural Representations Compatible with Behavior Interpretation

Jingyuan Li

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

Doctor of Philosophy

University of Washington

2025

Reading Committee:
Eli Shlizerman, Chair
Amy Orsborn
Rajesh Rao

Program Authorized to Offer Degree:
Electrical and Computer Engineering

University of Washington

Abstract

Learning Neural Representations Compatible with Behavior Interpretation

Jingyuan Li

Chair of the Supervisory Committee:
Eli Shlizerman
Electrical and Computer Engineering

Recent advances in neurotechnology have enabled the simultaneous recording of large-scale neural activity and behavioral data, opening opportunities to elucidate the neural mechanisms underlying behavior and to improve brain-computer interfaces (BCIs). Establishing association between neural and behavioral recordings plays a pivotal role in accomplishing these opportunities. However, both neural and behavioral recordings are inherently high-dimensional, posing challenges to establish the association in a direct way. Alternatively, projecting these high-dimensional recordings onto structured latent spaces can be effective in identifying underlying neural and behavioral relationships and bridging the gap between the two.

Algorithms that project neural and behavioral recordings onto lower-dimensional representations have been proposed. For instance, latent variable models (LVMs) have been introduced to transform neural data into interpretable low-dimensional representations for analysis. While effective, these methods often overlook the sequential and causal properties inherent in neural activity. Meanwhile, approaches for automatic behavior understanding typically depend on extensive human annotation to achieve high precision. The annotation procedure is often labor-intensive and subjective, limiting scalability and consistency. To overcome these limitations, this thesis proposes approaches for a more biologically realistic and efficient analysis of neural and behavioral recordings.

Specifically, we introduce a neural representation learning approach that explicitly incorporates temporal causality. In this framework, representation learning is formulated to estimate future neural activity solely from its past. We additionally included a graphical prior to model pairwise spatial interactions among neural recording channels. Experiments on synthetic and actual neural datasets demonstrate that this method can enhance the estimation of future neural dynamics, recover the underlying spatial interactions, and align neural trajectories with behavioral states. In parallel, for efficient behavioral state discovery, we develop an active learning-based, semi-supervised approach for behavioral state discovery and classification, significantly reducing the annotation burden while maintaining high accuracy. As a result, we introduce OpenLabCluster, an open-source toolkit designed to identify behavior categories across diverse species to democratize these behavioral techniques as a convenient user interface. Finally, we propose algorithms to bridge neural representations and behavioral states. With an application to brain-to-text decoding, we show enhanced accuracy in decoding neural signals into phonemic and textual outputs by leveraging refined behavioral states. This algorithm underscores the importance of using relevant and precise behavioral states as additional guidance to enhance brain decoding fidelity.

Overall, these contributions facilitate the understanding of neural-behavior relationships and improve the precision of brain-computer interfaces, potentially paving the way for neural computation discovery and the development of high-performance BCIs.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I wish to express my sincere appreciation to my advisor, Dr. Eli Shlizerman. It has been an honor to be under his guidance. His insightfulness regarding research directions and unwavering support have profoundly influenced my academic journey. Besides, his thoughtful consideration beyond the realm of academia has inspired me to face life's challenges with courage.

I am also deeply grateful to my supervisory committee—Dr. Amy Orsborn, Dr. Rajesh Rao, and Dr. Vikram Iyer—for their dedicated support and valuable feedback.

I extend my gratitude to my colleagues and friends—Trung Le, Ying Yu, Kun Su, Yang Zheng, Xiulong Liu, Mingfei Chen, Jinlin Xiang, Rahul Biswas, Jimin Kim, Moishe Keselman, Yan Jiang, Wenqi Cui, Lin Zhang, Feifei Yang, Jing Wang, Tianyan Zhou, Yuanyuan Wei, Junyou Dong, Jianwei Liu, and Wenxiao Zhang, among others—whose camaraderie and encouragement have greatly enriched my PhD journey.

I am especially grateful to my external research collaborators—Dr. Carlos Campos from the UW Medicine Division of Metabolism, Endocrinology, and Nutrition; Dr. Anna Bowen from the Steinmetz Lab; Dr. Leo Scholl; and Pavithra Rajeswaran from aoLab—as well as to my mentor, Dr. Pan Li from Georgia Tech, whose expertise and shared insights have significantly contributed to the success of my PhD study.

I am thankful for the experiences and knowledge shared by my mentors, Chunfeng Weng and Yijie Hong, during my internships at Google X. My time at Microsoft Research was similarly enriched by the creative insights provided by Dr. Yansen Wang, Dr. Dongsheng Li, and Dr. Dongqi Han. I also extend my sincere thanks to my collaborators at Google X and Microsoft Research—Xiaoyuan Guo, Matthias

Minderer, Alexey A. Gritsenko, Nie Lin, Yimu Zhang, Zhihe Yang, Qijun Luo, and Taiting Lu—for their support and encouragement.

Over the past five years, amidst the challenges of both life and research, I discovered paths to persevere while forging meaningful connections with those who embody kindness and support. This PhD journey has not only enriched my research and learning abilities but has also taught me how to adapt in the face of adversity. These invaluable experiences have shaped my life and laid a strong foundation for my future success. I am deeply grateful to everyone who has supported and inspired me along the way.

DEDICATION

This thesis is dedicated to my parents, Fengling Da and Yi Li, and all my family members, whose unwavering love, guidance, and sacrifices have made this journey possible.

Chapter 1
INTRODUCTION

1.1 Background

Understanding the neural mechanisms underlying behavior has long been a central goal in neuroscience. Building on achievements in single-neuron recordings—exemplified by Adrian's pioneering work in 1928 [R 1]—researchers have progressively advanced techniques that enable large-scale neural recordings alongside simultaneous behavioral recordings to infer neural mechanisms underlying behavior [R 2, 3].

To reach such goal, it is indispensable to identify relationships between neural and behavioral recordings. This process typically involves (1) identifying the characteristic of neural activity associated with specific behavioral states and (2) establishing the transformation between neural activity and behavior. Despite availability of large-scale neural and behavioral recordings, establishing their relationships remains a challenge [R 3]. This is primarily due to the high dimensionality of both modalities. For example, neural activity recorded via high-density electronic arrays is typically represented as multivariate time series as shown in Fig. 1.1 (top left). While, behavioral data is typically captured as continuous video streams (Fig. 1.1, top right). To address these challenges, researchers have been working on three key aspects (Fig. 1.1): Neural Encoders, which transform neural recordings into low-dimensional neural representations; Behavioral Encoders, which map behavioral data into discrete behavioral states; and Neural Decoders, which translate neural activity into behavioral states.

A widely adopted approach for Neural Encoders involves Latent Variable Models (LVMs) [R 4, 5]. LVMs apply nonlinear transformations to neural signals, producing low-dimensional latent representations. Original neural signals can be reconstructed from these representations

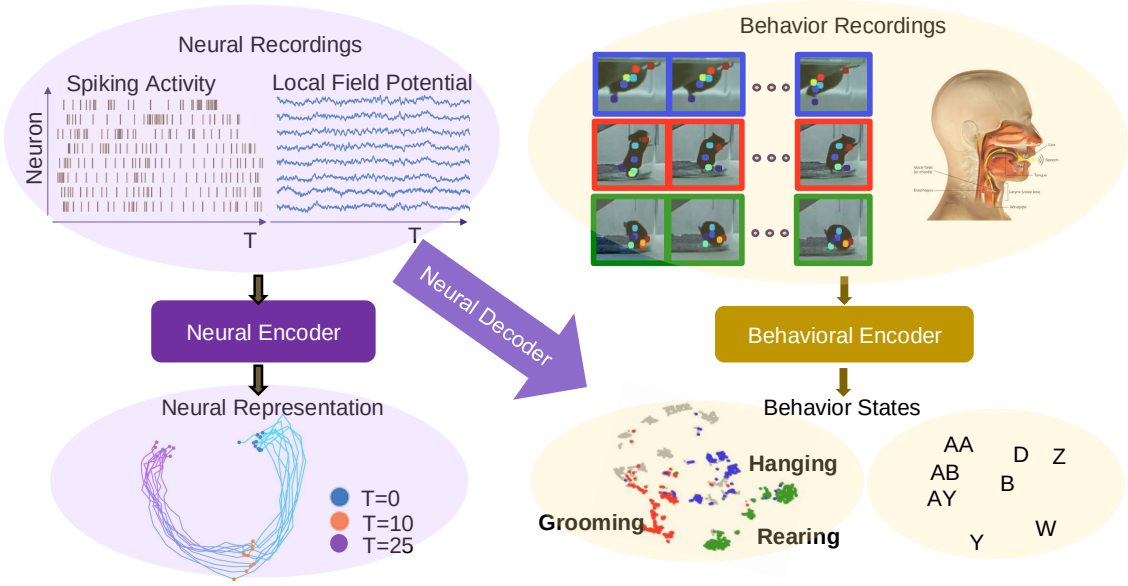

Figure 1.1: High-level overview of learning neural representations (left) and behavior representations (right). Encoders transform recordings into corresponding latent representations. The neural decoder decodes the information from the neural space into behaviors. The choices of neural encoder, behavior encoder, and neural decoder are vital and could lead to different neural representations, behavior representations, and behavior decoding performance.

[R 6, 7, 8]. Notably, reconstruction-based LVMs are typically non-causal; that is, the latent representation at the time t is derived from both preceding ($t_p < t$) and subsequent ($t_f > t$) neural signals. In contrast, biological neural systems operate under strict temporal constraints. Neural activity at any given time t is determined exclusively by past and present neural states. Therefore, utilizing only historical data for neural representation learning, such as forecasting, is more aligned with the brain's inherent temporal causality and may better reveal underlying neural computations. While Neural Encoders are not usually consistent with biological realism, Behavioral Encoders face challenges in recovering discrete behavioral states efficiently.

Most Behavioral Encoders rely on supervised learning, where classifiers categorize behavior sequences into predefined states [R 9, 10, 11, 12]. However, achieving high precision requires large volumes of annotated training data, making the process labor-intensive, time-consuming, and subjective. Algorithms that can efficiently learn a behavior classifier with only sparse annotated samples are required to perform large-scale neural-behavioral analysis effectively.

Additionally, the precision of identified behavioral states can further affect accuracy of Neural Decoders. A key application of these decoders is brain-to-text decoding. In this case, neural activity recorded from the motor cortex is transformed into discrete behavioral states, represented as phonetic units [R 13, 14]. Such states are not guaranteed to capture the complexities of the neural computation underlying behavior, such as the coarticulation effect in neural representation [R 15, 16]. The misalignment in behavioral states and neural representations limits the performance of these decoders. A more promising approach would involve incorporating behavioral states at a finer temporal resolution to account for coarticulation dynamics, potentially leading to improved decoding accuracy.

To overcome these limitations, this thesis introduces biologically plausible Neural Encoders and efficient Behavioral Encoders designed to improve the representations of neural and behavioral data. Additionally, it provides evidence demonstrating how high-temporal-resolution discrete behavioral states enhance the accuracy of Neural Decoders, improving brain-to-behavior mapping precision.

For **Neural Encoding**, we present a forecasting-oriented framework for neural representation learning that leverages spatiotemporal graphs to capture both spatial organization and temporal evolution of neural recordings. In our framework, each recording channel is modeled as a node in graphs, and inter-channel interactions are characterized by additive and multiplicative adjacency matrices emulating established mechanisms of neural interaction [R 17, 18, 19, 20]. We validate the effectiveness of our approach using both synthetic datasets and actual neural recordings from non-human primates. Our results demonstrate that the proposed framework can (1) recover underlying neural interaction patterns, (2) generate accurate predictions of future neural activities, and (3) align neural trajectories in latent

space that correspond to behavioral states.

To have an efficient and accurate **Behavioral Encoding**, this work proposes a semi-supervised, skeleton-based action recognition method (AL-SAR) that incorporates active learning principles. The approach synergistically combines the efficiency of unsupervised representation learning with the precision of supervised classification trained on selectively annotated samples. In particular, AL-SAR initially performs unsupervised representation without requiring human input, learning mapping time-series keypoint data into a structured latent space where behaviorally similar sequences naturally cluster together. Representative sequences are then strategically selected for human annotation, maximizing information gain while minimizing manual labeling. Extensive experiments on human datasets demonstrate that the proposed method achieves promising behavior classification accuracy using only sparsely annotated samples, yielding performance comparable to classifiers trained on fully annotated datasets.

Extending the method to general animal species, we developed OpenLabCluster, an open-source graphical interface in which behavioral clustering and classification stages are embedded together to study animal behavior across species. Results on animal datasets, including Home-Cage Mouse [R 9], Zebrafish [R 21], and C. elegans [R 22], indicate that OpenLabCluster is generalizable across species. Moreover, the open-source graphical interface enables behavior annotation and classification without requiring programming expertise.

In order to improve **Neural Decoding** fidelity for brain-to-text decoding, we introduce a novel framework that leverages diphone representations to integrate contextual information. In this framework, diphones (adjacent phoneme pairs capturing coarticulation dynamics) serve as the decoding targets, providing more affluent and precise contextual cues. Directly decoding diphones can introduce computational challenges, as the number of potential class combinations increases quadratically when moving from phonemes to diphones. To mitigate this complexity, we implemented a divide-and-conquer strategy that efficiently decodes diphone activities even with limited training data. When experimenting with Brain-to-Text 2024 Benchmark [R 13], a superior brain-to-phoneme decoding accuracy is observed leveraging

diphone and divide-and-conquer strategy, demonstrating the significance of incorporating behavioral definitions aligned with neural representations for improved brain decoding accuracy. Furthermore, we enhance the decoding pipeline with a three-stage language model integration. Combining the divide-and-conquer strategy and the LLM enhancements, we demonstrate significant improvement in brain-to-phoneme and brain-to-text decoding accuracy through experiments.

1.2 Summary of Contributions

My doctoral research focuses on developing methodologies facilitating the study of neural mechanisms that lead to complex overt behavior. The key contributions of this work span three interconnected areas:

- We introduce a forecasting-oriented task that reimagines the modeling of neural dynamics together with Additive, Multiplicative, and Adaptive Graph Neural Networks (AMAG) to capture both the temporal evolution and spatial organization of neural activity. This approach respects neural causality by forecasting future neural activities solely from the history while explicitly modeling inter-neuronal interactions through learnable graph structures [Jingyuan 1].

- We devise efficient methodologies for behavioral pattern discovery. By integrating active learning strategies with unsupervised representation techniques [Jingyuan 2][1], this work significantly reduces the manual effort required for annotation, which is essential for learning a behavior classification model, while maintaining high accuracy.

[1]In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of University of Washington's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink. If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

- We further extend the work with the development of OpenLabCluster [Jingyuan 3], a graphical user interface that enables annotation, classification, and detection of behavioral patterns across diverse species [Jingyuan 4].

- A direct mapping from neural signals to observed behavior is established with a focus on brain-to-speech decoding [Jingyuan 5]. By introducing diphone-based decoding targets and integrating them with large language model pipelines, this approach addresses the context dependency of articulatory neural representations, significantly improving brain-to-text translation accuracy

These contributions collectively aim to approach neural–behavioral relationships and advance practical brain–computer interface technologies byintroducing novel algorithms for large-scale neural and behavior recordings analysis and enhancing neural-to-behavior decoding accuracy through behaviorally aligned decoding targets.

1.3 Thesis Outline

The thesis includes seven chapters:

Chapter 2: Fundamental Concepts

This chapter introduces the essential background for the work, including neural data recording techniques, behavioral recording methodologies, and the key machine learning approaches employed throughout the thesis.

Chapter 3: Neural Representation Learning

Here, we outline the motivation for learning neural representations through a self-supervised, forecasting-oriented approach. This chapter emphasizes the use of Graph Neural Networks to capture the spatial relationships among neurons, reflecting the causal dynamics inherent in neural activity.

Chapter 4: Behavioral Representation Learning

This chapter presents efficient methods for discovering and classifying behavioral patterns. By integrating active learning with semi-supervised techniques, the proposed framework

significantly reduces the manual annotation burden while ensuring behavior recognition accuracy.

Chapter 5: Graphical User Interface for Behavior Recognition

Extending the work of Chapter 4, this chapter details the development of an interactive graphical user interface specifically designed for behavior recognition, with a particular focus on animal behavior. The system democratizes advanced behavioral analysis by enabling users to annotate, classify, and detect behavioral patterns.

Chapter 6: Neural Decoding and Brain-to-Text Conversion

In this chapter, we describe methods that bridge neural signals with behavioral outputs. Focusing on speech production, We demonstrate how neural signals can be decoded into representations relevant to articulatory activity and subsequently translated into text with high accuracy. This work addresses challenges such as coarticulation and context dependency in speech decoding.

Chapter 7: Conclusions and Contributions

The final chapter summarizes the key findings and contributions of the thesis, discussing the implications of the developed methods for understanding neural computation and advancing brain–computer interface technologies.

Chapter 2
FUNDAMENTAL CONCEPTS AND RELATED WORKS

This section provides an overview of the background and prior work on neural and behavioral recordings for understanding the rationale behind the analytical methods developed in this field. Additionally, we provide an overview of the approaches targeting neural representation learning and behavior recognition. Since many of these methodologies are grounded in machine learning (ML) and deep learning (DL) techniques, key ML and DL basics are also reviewed at the end of the chapter.

2.1 Neural Recordings

A wide range of techniques is available for collecting signals from the brain to infer neuronal activity. Each recording modality offers distinct advantages and limitations that affect subsequent analysis and applications. In this section, we highlight two groups of recording techniques: electrode-based and imaging-based.

Electrode Based Recordings. Electrode-based neural recording techniques can be categorized based on the number and configuration of electrodes. Single-electrode approaches, such as depth electrodes [R 23], are designed to record neuronal activity in deep brain regions. In most clinical and research applications, multiple depth electrodes are implanted parallelly to form an array, allowing for broader spatial coverage. However, the spacing between these electrodes is typically larger than that of high-density electrode arrays, such as laminar electrodes [R 24, 25], the Utah array [R 26, 27], electrocorticography (ECoG) arrays [R 28], and Neuropixels probes [R 29].

Laminar electrodes are designed to record neural activity across different cortical layers

within a single column, typically comprising 16–32 recording channels. Similarly, Neuropixels probes capture layer-specific neural activity but at a significantly higher spatial resolution, featuring approximately 1,000 recording sites, thereby enabling the simultaneous recording of large-scale neuronal populations.

In contrast, the Utah array is a 3D grid of microelectrodes that penetrate the cortical surface. However, unlike laminar electrodes or Neuropixels, it records activity primarily within the same cortical layer rather than across layers. The Utah array, typically containing 96 electrodes, is optimized for capturing single-unit activity (SUA), multi-unit activity (MUA), and local field potentials (LFPs) [R 30].

The electrocorticography (ECoG) array differs from both laminar and Utah arrays regarding invasiveness and signal resolution. Unlike the Utah array, ECoG electrodes do not penetrate the cortex but are placed directly onto the brain surface, making them less invasive. Consequently, while ECoG provides higher spatial resolution than non-invasive techniques, it lacks the single-neuron resolution compare to Neuropixels, Utah arrays, and laminar electrodes, as it records aggregated electrical activity from neuronal ensembles rather than individual neurons.

When electrodes are placed outside the skull, the technique is known as electroencephalography (EEG) [R 31]. EEG employs a multi-electrode array distributed across multiple brain regions to capture whole-brain electrical activity. However, due to the extracranial placement of electrodes, EEG signals represent the summed activity of large neural populations and are susceptible to artifacts from muscle activity and environmental noise. Additionally, because signals must pass through the skull, the spatial resolution of EEG is significantly lower than that of invasive recording techniques, limiting its ability to localize neural events precisely.

The signals recorded using these recording techniques are continuous spatiotemporal waveforms. A typical pre-processing step is to decompose these waveforms into distinct frequency bands, including Delta (0.5–4 Hz), Theta (4–8 Hz), Gamma (8–12 Hz), Beta (13–30 Hz), and High Gamma (70–150 Hz), as distinct bands could correspond to different functions. To study neural recordings at the single neuronal level resolution, researchers propose to

examine neurons' spike activity using spike sorting and thresholding techniques. As shown in Fig.1.1 (top left). The neural spike activity is a discrete representation of the initially collected signal, which is less noisy with a sacrifice of some information contained in the original signals.

Neural Imaging Techniques. In contrast to electrode arrays, calcium imaging captures neural activity in the form of images. This technique leverages the pivotal role of calcium ions in cellular processes—such as neurotransmitter release, muscle contraction, and gene expression—to visualize neuronal activity in real-time. By using fluorescent indicators that bind to calcium ions and emit light in response to changes in calcium concentration, calcium imaging enables direct monitoring of neuronal activity [R 32, 33]. It can record activity across multiple scales, from subcellular components to large neuronal populations. Calcium image can achieve high spatial and temporal resolution [R 34, 35]. This section focuses on neural imaging techniques and the corresponding modeling approaches.

Functional Magnetic Resonance Imaging (fMRI) is a non-invasive method for recording brain activity with high spatial resolution. It indirectly infers neural activity by capturing hemodynamic responses, specifically changes in blood flow and oxygenation [R 36]. This approach relies on the premise that increased neural activity elicits a localized increase in blood flow, which is then reflected in the fMRI signal. The resulting data are presented in an image format with a typical spatial resolution of 3 to 4 millimeters, allowing precise localization of brain activity to specific anatomical structures. However, due to its relatively low temporal resolution—on the order of seconds—fMRI is less effective at capturing rapid neuronal events [R 37].

2.2 Learning Neural Representation

The preceding section reviewed various techniques for collecting neural signals, which serve as quantitative measures of brain state. However, deciphering neural mechanisms from these recordings is challenging due to their high-dimensional spatiotemporal nature, complicating

human interpretation. As a result, numerous methods have been developed to embed neural activity into more tractable, low-dimensional latent representations. In this section, we focus on modeling approaches extracting latent representation from electrode-based neural recordings.

Neural Representation Learning. Studying the latent representation of neural signals provides insights into the computation performed in the brain [R 38]. The classical works were proposed to recover latent representation with Gaussian Process [R 4, 39, 5] and linear dynamical systems [R 40, 41, 42, 43, 44]. More recently, a series of nonlinear models have been proposed, including Multi-Layer Perceptron (MLP) [R 45, 46, 47], RNNs [R 6, 48, 49, 50, 51], Neural ODEs [R 52], and Transformers [R 53, 54, 55, 56]. Latent representations of population dynamics are often extracted while these models optimize a reconstruction objective, upon which interpretable features emerge naturally [R 57, 58, 59]. Different optimization tasks could lead to different embedding spaces. Forecasting task, which aims to predict future activity based on past data, adhering to temporal causality constraints, could provide further insight into neural representations that are closer to representations employed by the brain [R 60, 61, 62, 63].

2.3 Behavior Recordings

Behavior, as the substrate of neural population activity, can be recorded in various formats—such as video recordings or anatomical keypoint sequence. This section describes how these recordings are commonly obtained.

Source of Behavior Video Recordings. Video recordings provide a powerful means to capture free-moving individuals in both indoor and outdoor settings. Widely used behavioral benchmark datasets, such as Kinetics 400 [R 64], UCF101 [R 65], and CMKT101 [R 64], are typically provided in video format. However, these recordings often include significant background noise, which can interfere with behavior analysis by introducing irrelevant information.

Source of Anatomical Keypoint Recordings. Anatomical keypoint recordings are commonly acquired using marker-based systems that affix lightweight, non-intrusive markers to specific anatomical landmarks. This approach minimizes interference with natural behavior, provides precise estimates of body positions, and effectively filters out extraneous background information.

An alternative to obtaining body keypoints via physical markers is to extract them directly from raw video using markerless pose estimation algorithms. This automatic detection method is advantageous because it eliminates the need for physical markers. Several open-source packages offer markerless pose estimation with sufficient accuracy from general video recordings. For instance, OpenPose [R 66] performs human pose estimation, while DeepLabCut [R 67, 68] is designed for animal keypoint estimation. DeepLabCut includes an adaption of a pre-trained model to the particular video and keypoints which allows it to perform animal pose estimation across various species. Extensions and improvements of DeepLabCut, such as DeepLabCut 2+, enabling 3D pose estimation, better accuracy, and multi-animal pose estimation, have been recently made available [R 69, 70]. Furthermore, additional approaches which address animal pose estimation have been further developed [R 71, 72, 73, 74, 75, 76, 77].

Behavior Recording During Speech Production. Speech production is a complex behavior that requires the coordinated movement of the lips, tongue, laryngeal muscles, and pharyngeal muscles to generate audible sound [R 78]. To capture these muscle movements, researchers have devised various recording techniques, such as Electromagnetic Articulography (EMA) which tracks tongue movements using marker coils attached to the tongue.[R 79, 80], and magnetoencephalographic-compatible speech tracking systems [R 81, 82]. When these orchestrated movements generatestructured, intelligible speech, the resulting acoustic signal can be decomposed into fundamental phonetic units known as phonemes. For example, spoken English comprises approximately 40 distinct phonemes. Consequently, for applications such as brain-to-speech decoding, analyzing these high-level phonemic representations is often more efficient than directly recording the precise muscular movements.

2.4 Learning Behavior Semantics

This section introduces methodologies to extract behavioral semantics from video and anatomical keypoints recordings.

Extracting Behavior Semantics from Video Recordings. Various research efforts have been made toward raw video based automatic understanding of behavior. Particularly with machine learning-based methods such as Multi-Fiber Network, Selfee, BehaveNet, and uBAM [R 83, 84, 85, 86] which leverage Convolution Neural Networks (CNN) architecture to extract frame-wise features. Temporal structure could be included with Recurrent Neural Networks (RNNs) [R 87, 88], Temporal Gaussian Mixture [R 89] or temporal CNN as in SIPEC [R 90].

Extracting Behavior Semantics from body Skeleton Keypoints . Various skeleton-based behavior recognition approaches have been proposed. These include supervised methods which analyze behavior related physical statistics [R 91, 92, 93], deep learning with CNNs [R 94, 95, 96], RNNs and their variants [R 97, 98, 99], and Graph Convolutional Networks [R 100, 101, 102, 103, 104]. Unsupervised methods have been introduced as well. Such approaches build a latent representation through learning to reconstruct input sequences with an encoder-decoder network structure. The latent space of such networks is shown to self-organize into clusters enabling a simple classifier such as K-Nearest Neighbour (KNN) to identify action types [R 105, 106]. However, the KNN component still remains supervised and annotations are necessary to identify actions. To avoid supervision and improve action recognition accuracy, semi-supervised approaches have been proposed. These approaches learn the recognition task by leveraging annotations from a randomly selected subset of samples. Examples include methods such as ASSL [R 107], MS^{2}L [R 108], SC3D [R 109]. These methods do not deal with the selection of sequences for annotation and instead, assume a given random annotated set. In applications, it is critical to optimize such a selection and to seek samples that are more informative for learning actively. Such a selection would need to be achieved with AL

methods.

2.5 Behavior Relevant Neural Representation Extraction and Decoding

Behavior-relevant neural representations can be extracted either through aligning neural and behavioral representations in a shared latent space or decoding neural signals to behavior, with decoding offering an efficient strategy for developing brain–computer interfaces.

Prototyped Behavior Decoding from Neural Signals. Neural signals have long been used for behavior decoding. Traditional methods, such as linear regression and Kalman filters, achieve reasonable accuracy when behavioral dynamics are confined to low-dimensional spaces [R 110, 111, 112]. More recently, deep learning models have demonstrated superior performance in this domain [R 113, 114]. For example, POYO leverages transformer architectures to integrate neural signals collected over multiple days, sessions, and subjects, enabling large-scale behavior decoding with promising accuracy and generalizability [R 114]. To learn both behavior-relevant and behavior-irrelevant representations, the Preferential Subspace Identification (PSID) method introduces a two-stage training paradigm that disentangles neural representations into a behavior-relevant subspace, which governs behavior generation, and a behavior-irrelevant subspace that captures internal neural dynamics [R 62, 61].

Spoken Behavior Decoding from Neural Signals. Extracting speech-relevant neural signals offers a promising avenue for patients with movement impairments who cannot produce expected speech. Early work in speech decoding focused on limited vocabulary sizes [R 115, 116], while subsequent efforts improved performance by decoding letters [R 117]. Other studies have explored phonemes as decoding targets [R 118, 16, 119, 13, 14]. However, direct phoneme decoding is not optimal, as neural representations of phonemes can vary with context [R 16].

Neural and Behavioral Representations Alignment Neural representation can be learned jointly with behavior representation learning. For example, CEBRA employs nonlinear

techniques to simultaneously extract and align neural and behavioral representations within a shared latent space [R 58]. This joint learning paradigm not only deepens our understanding of the neural-behavioral relationship but also enhances behavior decoding performance. When these representations are well organized in latent space, neural activity can be used to retrieve corresponding behavioral information—enabling applications such as generating behavior videos from neural signals [R 120] or mapping neural trajectories to behavioral trajectories [R 121]. Continued advances in machine learning and large-scale neural and behavioral recording techniques promise to further elucidate the link between neural and behavioral phenomena [R 3, 122].

2.6 Relevant Machine Learning Concepts

This section introduces basic machine learning concepts discussed in the following Chapters as well as more advanced ones including Active Learning, Time-Series Forecasting, Graph Neural Networks (GNNs) and Speech-to-Text Decoding.

Supervised Learning. Supervised learning entails training a model on human-annotated labels—such as image or behavior class labels. The accuracy of the resulting classifier depends significantly on factors including the model architecture, as well as the volume and quality of the annotations provided.

Unsupervised Learning. Unsupervised learning, in contrast, trains models on data without human-provided annotations. Typically, these models learn by reconstructing the original input, enabling them to discover underlying structures. For time series data, unsupervised models may be trained to predict future inputs—a task commonly referred to as forecasting. Because these models operate without supervision, they learn to represent the input in a compressed, low-dimensional space. Although these latent features are useful for reconstruction or forecasting, these latent features are not always optimal for downstream tasks such as classification, which generally require supervised learning to achieve higher performance.

Few-shots Learning. Few-shot learning methods propose to "metatrain" a base model on auxiliary action classes, leveraging a large amount of annotated samples for these classes. This training process aims to learn a set of general model parameters, enabling the base model's efficient adaptation to unseen classes not included in the auxiliary action classes. This adaptation is achieved through further training on just a few examples of these unseen classes, as discussed in previous works [R 123, 124, 125, 126]. For example, in the context of action recognition, few-shot learning techniques for skeleton-based action recognition aim to diminish the required number of annotations without compromising the accuracy of action recognition.

Semi-supervised learning. The availability of auxiliary classes with abundant annotated samples is not always guaranteed for supervised training. Additionally, the expectation that auxiliary and unseen classes follow a similar data distribution poses a limitation on the practicality of few-shot learning. Particularly for skeleton based action classification, variability in data quality, style, the number of skeleton keypoints, and even subjects (e.g., human subjects vs. animal subjects) across different datasets can be significant. To address scenarios where well-annotated auxiliary classes are unavailable, semi-supervised methods have been proposed. In such cases, the model is trained by combining an unsupervised target and a supervised target using partially annotated samples. This involves learning informative representations for both labeled and unlabeled data. Notable examples of this approach in skeleton-based action recognition include methods such as ASSL [R 107], MS^{2}L [R 108], and SC3D [R 109].

In-Context Learning. In-Context learning is often discussed in context of Large Language Models (LLMs). LLMs pretrained on large corpora of texts exhibit the ability to learn new tasks from context [R 127]. That is, conditioning on a few demonstrations of input-target pairs, LLMs can generalize to unseen cases without updating their weights. This ICL ability has proven useful across a wide range of tasks [R 128, 129]. While ICL typically underperforms a

specialized LLM finetuned for a specific downstream task, it still surpasses zero-shot inference, and is particularly valuable when finetuning is not feasible due to resource constraints such as time or computational power, or the inacessibility of proprietary LLMs [R 130].

Active Learning. Semi-supervised methods do not consider that not all annotated samples contribute equally to the training of a classifier, and therefore, it is advantageous to select for annotation the samples that dominantly represent their classes. Such selection could improve the effectiveness of the classifier and at the same time minimize the number of annotations needed for training. Active Learning (AL) algorithms [R 131, 132, 133] have been established based on this principle and showed promising results when applied to image classification tasks [R 134, 135, 136]. Indeed, a variety of AL selection strategies have been developed. These include approaches based on principles of diversity [R 134], uncertainty [R 136, 137, 138], and model decision [R 139]. While AL is widely adopted for image data, only a handful of studies have investigated the applicability of AL to sequential data that is spatio-temporal [R 140, 141, 142], and direct application of these methods show no superiority over the random selection of samples. AL methods typically belong to three categories: (i) sample synthesis, (ii) stream-based selective sampling, and (iii) pool-based sampling [R 131, 132, 143].

Sample Synthesis Based AL. Sample synthesis is based on generating additional samples of action sequences. These generated sequences are typically of lower quality, making them challenging candidates for annotation [R 144].

Stream Based AL. Stream-based AL methods work with real data. The stream-based selection considers one sample at a time, decides whether to annotate the sample at that time, and is applicable in online learning scenarios [R 132]. **Pool Based AL**. Pool-based methods select a set of samples at each stage and are therefore expected to be more efficient for applications with a dataset already prepared.

Pool Based AL Criterion. Pool based AL methods are widely used in classical machine learning techniques such as support vector machine and logistic regression. and take into

consideration aspects such as diversity [R 145, 146, 147], and uncertainty (based on entropy [R 148], confidence estimation [R 149], and margin estimation [R 150, 151]). The idea of enforcing diversity or uncertainty for sample selection has been adapted to deep learning as well. For example, diversity is incorporated by selecting a sample batch that covers the whole space with a minimum covering radius for each sample [R 134]. Uncertainty metrics with deep learning models are computed with techniques such as dropout [R 138, 148, 152], the ensemble of models [R 135], and image augmentation [R 136]. These techniques compute the prediction entropy or the variance among 'multiple-outputs' for a sample. In many scenarios, it is advantageous to consider both diversity and uncertainty [R 153, 154, 155], since diversity reduces redundancy of selected samples and uncertainty focuses on samples where the model is less confident. In addition to these methods, a new branch of deep learning pool-based AL methods has been introduced, solving AL from a different aspect by learning a Discriminator (DIS) where samples which DIS is least confident in are selected for annotation [R 139, 141]. These methods rely on a network to learn the characteristics of unlabeled samples instead of measuring the uncertainty or the diversity with hand-designed features as the aforementioned approaches do.

Time-Series Forecasting. Time series forecasting, i.e., the prediction of the next values in a time series, is a well-studied topic that encompasses various applications, such as traffic flow forecasting, motion prediction, and electric load forecasting. Examples of proposed methods include methods that are CNN-based [R 156, 157], RNN based [R 158, 159, 160], Multi-layer perceptron [R 161] and so on. Transformers with additional components for capturing long-term dependencies have also been proposed for time series forecasting [R 162, 163, 72]. Also, models such as Graph Neural Networks (GNNs) have been proposed for time series forecasting focusing on datasets with inherent structural organization, e.g., traffic flow [R 164, 53, 165]. For example, these methods have been applied to functional Magnetic Resonance Imaging (fMRI) forecasting [R 166] leveraging Diffusion Convolutional Recurrent Neural Network (DCRNN) and Graph WaveNet [R 167, 168]. In these applications, GNN has been embedded

in GRU gating mechanism to learn the spatial-temporal relationship (DCRNN) or CNN and GNN have been proposed to be intertwined to perform spatial-temporal modeling (Graph WaveNet). GraphS4mer, combining graph neural networks and structured state space models, instead performs temporal embedding followed by spatial message-passing in a sequential manner.[R 169]. These methods perform only the additive message-passing, which may not be sufficient for a complex system such as the brain.

Graph Neural Networks (GNN). In recent years, GNNs have been proposed as plausible models for analyzing brain activity across different modalities [R 170, 171, 172, 173]. In non-invasive neural recordings, GNNs have been applied to EEG and fMRI data to capture the relationships between different brain regions or features. For example, LGGNet learns local-global-graph representations of EEG for various cognitive classification tasks [R 174], and BrainGNN has been proposed to identify neurological biomarkers with GNN, leveraging the topological and functional information of fMRI [R 175]. Recently, the method has been extended to analyze ECoG recordings as a graph diffusion process [R 176]. A few applications of GNNs to invasive neural measurements have been introduced, although these are less ubiquitous than GNNs for other recording types. The Two-Stream GNN model, which has been designed to classify anesthetized states based on ECoG recording by building two fixed graphs, is a notable example of such a model [R 177]. The first graph in this model is the functional connectivity matrix computed from the phase lag index, and the second is the dual version of the first one. The Two-Stream GNN is not suitable for forecasting since it combines temporal signals into a unified vector as the model input.

Speech-to-Text Decoding. While brain-to-text and speech-to-text decoding share certain similarities, decoding text from neural signals is a significantly more challenging task. A key difference is that speech signals are univariate, while neural activity is multivariate as it is recorded by multi-channel electrodes. Furthermore, neural signals are far more intricate. Less is known about how neurons encode speech within their spiking activity, as well as the degree

to which speech-relevant components can be extracted from the complex interaction of neural population. However, brain-to-text decoding methods have drawn inspiration from speech-to-text decoding research, commonly referred to as Automatic Speech Recognition (ASR). Earlier studies [R 178, 179, 180] use Hidden Markov Models and Gaussian Mixture Models to decode recorded speech signals into phonemes before translating into words. [R 181, 182] suggest that using diphone or triphone could enhance the accuracy of ASR systems. Modern ASR systems have transitioned to end-to-end learning approaches, directly decoding speech signals into words [R 183, 184, 185, 186, 187]. These systems typically utilize Transformers [R 188, 189] or hybrid architectures that combine Transformers with Convolutional Neural Networks [R 185]. A common attribute to end-to-end learning with Transformers based architectures is that this type of learning requires large, multiple and diverse input-to-text datasets. Such datasets are generally not available in the neuroscience domain. Recurrent Neural Networks (RNNs) [R 190] decoders thereby have been traditionally more commonly implemented in existing works that address decoding tasks that involve neural activity. We therefore, as in prior works of brain-to-text decoding, adopt the two-stage system for brain-to-text decoding, where phonemes serve as the intermediate decoding targets.

Chapter 3

GRAPH NEURAL NETWORK FOR FORECASTING NEURON ACTIVITY

This chapter is a reproduction of published work in [1] with adaptations to fit the thesis structure and formatting.

3.1 Motivation

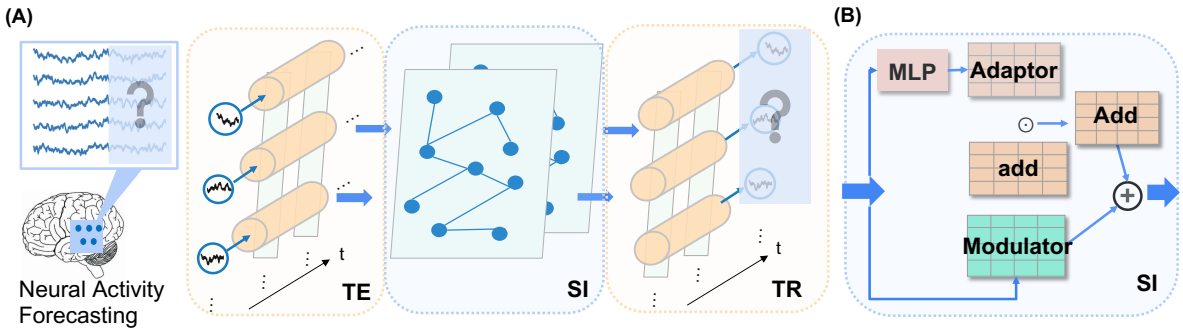

Figure 3.1: AMAG Overview. (A) The forecasting task and AMAG architecture with temporal Encoding (TE), Spatial Interaction (SI), and Temporal Readout (TR) modules to address the task. (B) Components of SI module.

Most deep neural network (DNN) architectures for neural data modeling—such as Latent Factor Analysis via Dynamical Systems (LFADS) and related methods—rely on a reconstruction-based paradigm: they encode single-trial population activity into a low-dimensional representation from which the original time series can be reconstructed [R 191, 192, 193]. Although these approaches capture latent manifolds effectively, they do not explicitly account for causal aspects of neural dynamics.

In contrast, forecasting future neural activity from past observations can provide insights more closely aligned with the brain's intrinsic temporal constraints [R 60, 61, 62, 63]. Forecasting also offers practical benefits for real-time neural decoding in Brain–Computer Interface (BCI) applications—by predicting upcoming signals, systems can reduce latency and potentially detect deviations from expected patterns (e.g., indicating equipment malfunctions or changes in neuronal states) [R 194]. However, forecasting is inherently more demanding than reconstruction, as it often requires larger datasets and is more sensitive to noise in the recorded signals.

To address these challenges, we propose incorporating domain-specific priors into the model design so that the learned representations emphasize generalizable features rather than overfitting to noisy or limited training data. One key prior is the spatial interaction among neurons, which can be modeled through additive processes [R 17, 18] and multiplicative processes [R 19, 20], as in Hodgkin–Huxley or Leaky Integrate-and-Fire frameworks. Other network-level modeling efforts have explored probabilistic graphical models to learn neuron interactions, though they often focus on capturing these interactions alone rather than using them for forecasting tasks [R 195, 196, 197, 198, 199]. More recent approaches, including Spatial Temporal Neural Data Transformer (STNDT) and Embedded Interaction Transformer (EIT), use attention mechanisms to represent spatial relationships [R 54, 55, 162], while Graph Neural Networks (GNNs) offer an alternative that is well suited for graph-structured data [R 200].

Following this rationale, we develop a GNN-based method called AMAG (Additive, Multiplicative, and Adaptive Graph Neural Network) to forecast neuron signals using sample-dependent additive and multiplicative message-passing operations. As shown in Fig. 3.1(A), AMAG consists of three modules: (1) Temporal Encoding (TE), which processes time series from each electrode channel, (2) Spatial Interaction (SI), which captures inter-channel relationships via additive and multiplicative updates, and (3) Temporal Readout (TR), which outputs forecasts for future time steps. We validate AMAG on synthetic datasets, where it accurately recovers ground-truth channel interactions, and on four real motor cortex recordings

from rhesus macaques—two publicly available Utah array datasets and two large-scale micro-electrocorticography (μECoG) recordings. We compare AMAG to existing methods and to ablated variants of our own approach. Overall, AMAG consistently outperforms other models in forecasting accuracy while uncovering meaningful spatial relationships across electrode channels.

Below is a summary of our main contributions:

- *Modeling Spatiotemporal Relationships with AMAG.* We introduce a GNN architecture (Section 3.2) that forecasts neural activity through additive and multiplicative message passing, guided by a learnable adjacency matrix.

- *Revealing Underlying Spatial Interactions.* We demonstrate on synthetic data (Section 3.3.1) that AMAG not only achieves accurate forecasts but also uncovers the true interaction patterns between channels.

- *Predicting Future Neural Signals.* We apply AMAG to four real neural datasets (Section 3.3.2) and show that it reliably generates accurate predictions of future activity, recovers spatial proximity in the adjacency matrix, and aligns neural trajectories in the latent space.

3.2 Method

Our goal is to forecast future neural dynamics based on previously observed activity. Formally, we seek a function f_θ such that $\hat{\boldsymbol{X}}_{t+1:t+\tau} = f_\theta(\boldsymbol{X}_{0:t})$, where $\boldsymbol{X} \in \mathbb{R}^{T\times N\times D}$ represents neural signals recorded from N electrode channels over a time window of size T, with each channel having D dimensional features. Here, $\boldsymbol{X}_{0:t}$ indicates the observed neural signals from 0 to t, and $\hat{\boldsymbol{X}}_{t+1:t+\tau}$ represents model's predictions for the interval $t+1$ to $t+\tau$.

To learn $f_\theta(\cdot)$, we formulate an optimization problem minimizing the loss $\mathcal{L}$ between predicted future signals $\hat{\boldsymbol{X}}_{t+1:t+\tau}$ and ground truth future signals $\boldsymbol{X}_{t+1:t+\tau}$ as follows

$$\min_{f_\theta} \mathbb{E}_{\boldsymbol{X}}\Big[\mathcal{L}(f_\theta\big(\boldsymbol{X}_{0:t}\big), \boldsymbol{X}_{t+1:t+\tau}\big)\Big]. \tag{3.1}$$

In particular, $f_\theta(\cdot)$ is constructed from three sub-modules as shown in Fig. 3.1: Temporal Encoding (**TE**) Module, Spatial Interaction (**SI**) Module, and Temporal Readout (**TR**) Module. The TE module captures temporal features for each channel individually. Then SI facilitates information exchange between channels, followed by TR to generate future neural activities. Below, we describe the SI module in detail, followed by an overview of TE and TR.

3.2.1 SI with Add and Modulator

We frame the SI module as a Graph Neural Network (GNN) operating on Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{A})$, where $\mathcal{V}$ denotes a set of nodes $|\mathcal{V}| = N$, $\mathcal{E}$ is a set of edges, and $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ is the adjacency matrix containing the weights of edges. The neighborhood of a node v is written as $\boldsymbol{N}(v)$, where $\boldsymbol{N}(v) = \{u|u \in V, (u, v) \in \mathcal{E}\}$. We denote neural signal from channel v at time t as $X_t^{(v)}$. In the notation that follows, FC is a single layer neural network, MLP refers to multilayer perceptron, and $\sigma(\cdot)$ is the sigmoid function.

Motivated by known additive [R 17] and multiplicative [R 19, 20] neuronal interactions, SI consists of two subgraphs—Add and Modulator—with distinct message-passing functions. We also introduce a sample-dependent Adaptor to capture dynamic variability across different input samples.

Add Module. The Add Module performs the additive interaction between channels with the message-passing function $\mathcal{M}_a(\cdot)$, depending on the adjacency matrix $\boldsymbol{A}_a \in \mathbb{R}^{N\times N}$. Assuming we have d dimensional channel features $\boldsymbol{h}_t^{(v)}$ for node v at timestep t ($\boldsymbol{h}_t^{(v)} \in \mathbb{R}^d$), $\mathcal{M}_a(\cdot)$ updates $\boldsymbol{h}_t^{(v)}$ with the additive message from neighbor node u as $\mathcal{M}_a(\boldsymbol{h}_t^{(u)}, \boldsymbol{h}_t^{(v)}, \boldsymbol{A}_a) = A_a^{(u,v)}\boldsymbol{h}_t^{(u)}$, where $u \in \boldsymbol{N}_a(v)$. Such that the updated channel feature of Add Module is the weighted feature of neighborhood,

$$\boldsymbol{a}_t^{(v)} = \sum_{u\in\boldsymbol{N}_a(v)} A_a^{(u,v)}\boldsymbol{h}_t^{(u)}. \tag{3.2}$$

The element (u, v) in the adjacency matrix $\boldsymbol{A}_a$ indicates the influence of channel u on v. Since $\boldsymbol{A}_a$ is shared for all the input sequences, while channel interaction can change across inputs, we introduce a sample-dependent *Adaptor* Module. We treat $\boldsymbol{A}_a$ as a fundamental adjacency

matrix, and then we further learn a sample-dependent Adaptor as a matrix $\boldsymbol{S} \in \mathbb{R}^{N \times N}$ to uniquely modulate $\boldsymbol{A}_a$ depending on each sample. Then, the shared message-passing function Eq. 3.2 becomes a sample-dependent message-passing function Eq. 3.3,

$$\boldsymbol{a}_t^{(v)} = \sum_{u \in \boldsymbol{N}_a(v)} S^{(u,v)} A_a^{(u,v)} \boldsymbol{h}_t^{(u)}. \tag{3.3}$$

$S^{(u,v)}$ represents interaction strength between channel u and v relying on temporal embeddings for each channel, e.g., $\boldsymbol{H}^{(v)} = [\boldsymbol{h}_1^{(v)}, \ldots, \boldsymbol{h}_t^{(v)}]$ where t is the context window for future neural signal generation. Specifically, we compute $S^{(u,v)}$ as $\sigma(MLP([\boldsymbol{H}^{(u)}, \boldsymbol{H}^{(v)}]))$, such that $S^{(u,v)}$ ranges from 0 to 1, similar to matrix construction for learning interpretable GNNs [R 201].

Since the fundamental adjacency matrix for the Add Module is unknown, $\boldsymbol{A}_a$ needs to be learned during the optimization of the model parameters. We observe that direct learning of $\boldsymbol{A}_a$ from random initialization could make the training process unstable and thus instead, we initialize the $\boldsymbol{A}_a$ with the precomputed correlation matrix from the neural recordings to stabilize the learning process.

Modulator Module. The Modulator Module is the multiplicative message-passing operator, with function $\mathcal{M}_m(\cdot)$. We incorporate the adjacency matrix $\boldsymbol{A}_m \in \mathbf{R}^{N \times N}$ to encode the neighborhood information that performs a multiplicative operation. $\mathcal{M}_m(\cdot)$ indicates how the feature of the neighborhood modulates the target node feature, which can be expressed as $\mathcal{M}_m(\boldsymbol{h}_t^{(u)}, \boldsymbol{h}_t^{(v)}, \boldsymbol{A}_m) = A_m^{(u,v)} \boldsymbol{h}_t^{(u)} \odot \boldsymbol{h}_t^{(v)}$, where $\odot$ represents Hadamard product. The Modulator Module includes all the multiplicative information from the neighbors to update feature for target channel u at time t as $\mathbf{m}_t^{(u)}$

$$\boldsymbol{m}_t^{(v)} = \sum_{u \in \boldsymbol{N}_m(v)} \mathcal{M}_m(\boldsymbol{h}_t^{(u)}, \boldsymbol{h}_t^{(v)}, \boldsymbol{A}_m) \tag{3.4}$$

Similarly to the adjacency matrix for the Add Module, the adjacency matrix in the Modulator Module is also trainable and is initialized from the correlation matrix.

In summary, at a specific timestep t, the output of SI is

$$\boldsymbol{z}_t^{(v)} = \beta_1 \boldsymbol{h}_t^{(v)} + \beta_2 FC(\boldsymbol{a}_t^{(v)}) + \beta_3 FC(\boldsymbol{m}_t^{(v)}), \tag{3.5}$$

with β_1, β_2, and β_3 controlling the contributions of the self-connection, Add Module, and Modulator Module respectively.

3.2.2 Temporal Processing with TE and TR Modules

TE and TR perform temporal encoding and decoding. In particular, TE embeds input neural signals $\boldsymbol{X}^{(v)} \in \mathbb{R}^{T\times D}$ from channel v into the embedding space $\boldsymbol{H}^{(v)} \in \mathbb{R}^{T\times d}$. For multi-step forecasting, we mask neural signals after the context window t with constant, i.e., $\boldsymbol{X}^{(v)}_{t:T} = constant$. Then SI updates representation of channel v as $\boldsymbol{Z}^{(v)} \in \mathbb{R}^{T\times d}$, followed by TR generating future neuron activity $\hat{\boldsymbol{X}}^{(v)}_{t:T}$ taking $\boldsymbol{Z}^{(v)}$ as inputs. TE and TR can be Transformer or GRU. Since Recurrent Neural Networks (RNN) are common in modeling continuous time-series signals, we employ a variant of RNN, Gated Recurrent Unit (GRU) [R 159], to facilitate the temporal encoding (TE) and readout (TR) modules. The TE module consists of a single-layer GRU that takes input from channel v and computes the corresponding temporal embedding features $\boldsymbol{h}^{(v)}_t \in \mathbb{R}^d$ for each step t. These embedding features capture the underlying temporal dynamics of neural activity. To enable the exchange of information between the target channel v and its neighboring channels $\boldsymbol{N}(v)$, specified by the adjacency matrices $\boldsymbol{A}_a$ and $\boldsymbol{A}_m$, the spatial interaction (SI) module updates the embedding features. This interaction among channels allows for modeling of spatial dependencies.

The TR module consists of a single-layer GRU and a fully connected (FC) layer. The module generates future neural signals. This pipeline of TE, SI and TR modules is shown below

$$
\begin{aligned}
\boldsymbol{h}^{(v)}_t &= GRU_{TE}(X^{(v)}_t, \boldsymbol{h}^{(v)}_{t-1}), \quad \boldsymbol{h}^{(v)}_0 = \boldsymbol{0} \\
\boldsymbol{z}^{(v)}_t &= SI(\boldsymbol{h}^{(v)}_t, \{\boldsymbol{h}^{(u)}_t | u \in \boldsymbol{N}(v)\}, \boldsymbol{A}_a, \boldsymbol{A}_m) \\
\boldsymbol{r}^{(v)}_t &= GRU_{TR}(\boldsymbol{z}^{(v)}_t, \boldsymbol{r}^{(v)}_{t-1}) \\
X^{(v)}_{t+1} &= FC(\boldsymbol{r}^{(v)}_t)
\end{aligned}
$$

Parameter Learning. In summary, AMAG implents f_θ by subsequently applying TE, SI, and TR on the past neural signals. The optimal parameters θ are obtained through

backpropagation minimizing the Mean Squared Error (MSE) loss,

$$\mathcal{L} = \mathbb{E}_{\boldsymbol{X}}\big(\big\|\hat{\boldsymbol{X}}_{t:T} - \boldsymbol{X}_{t:T}\big\|_2\big). \tag{3.6}$$

3.3 Results

3.3.1 Learning The Adjacency Matrix of a Synthetic Dataset

Synthetic Dataset. T o explore the ability of AMAG to capture channel interactions, we generate synthetic datasets using linear non-Gaussian dynamic systems similar to previous work [R 202, 203, 204]. Specifically, with adjacency matrix $\boldsymbol{A}$ describing the spatial interactions among variables (channels), datasets $\boldsymbol{X}$ are generated recursively at each time step as $\boldsymbol{X}_t = \boldsymbol{X}_{t-1} + \mathcal{G}(\boldsymbol{X}_{t-1}, \boldsymbol{A}) + \mu$, where μ is uniformly distributed noise term. We consider two scenarios where the connection ratio of the adjacency matrix is 0.2 and 0.5, respectively. The resulting datasets consist of multichannel time series, where the value of each channel at a given time step depends on its past and its neighboring channels, as defined by the adjacency matrix.

Adjacency Matrix Recovery with AMAG. We investigate AMAG's ability to recover the ground-truth adjacency matrix under different training tasks: one-step forecasting, multi-step forecasting, reconstruction, and masked input reconstruction. Specifically, we incorporate these supplementary tasks alongside forecasting to examine how they affect adjacency recovery. . For a synthetic dataset with 20 channels, we visualize both the learned and ground-truth adjacency matrices in Fig.3.2.

To quantify adjacency recovery accuracy, we employ the F1 Score as a quantitative metric measuring the harmonic mean between precision and recall. Our results show that the one-step forecasting task achieves the highest recovery of ground truth edges ($F1 = \mathbf{0.88}$ at 0.5 connection ratio), followed by the multi-step forecasting task ($F1 = \mathbf{0.82}$). This is compared to the reconstruction task which yields a lower F1 score of **0.63**. The lower performance of reconstruction tasks could be due to the model's access to future signals, which potentially alleviates the need to learn the interactions with neighbors that future

signals rely on. While multi-step forecasting generates multiple future signals, some adjacency matrices may generate more accurate signals than others at a given moment, but considering all future moments collectively, different adjacency matrices may achieve similar performance, resulting in learned adjacency matrices being less accurate than one-step forecasting.

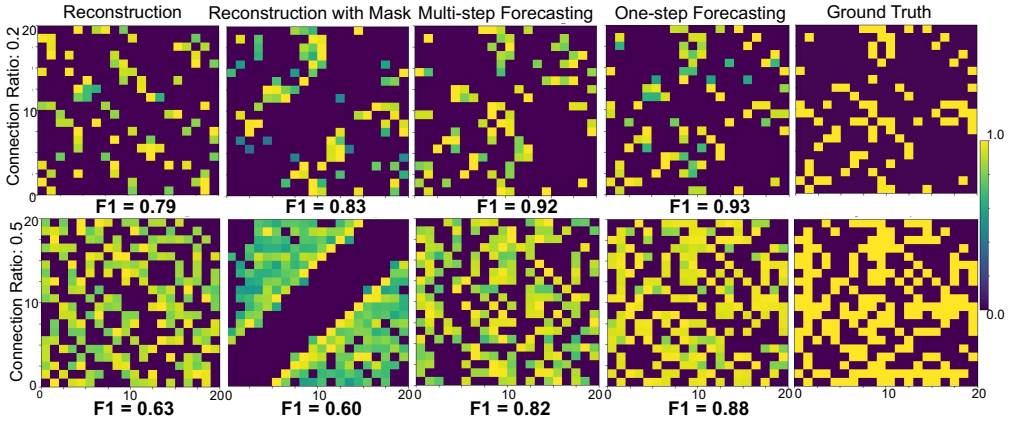

Figure 3.2: Colormap visualization of AMAG learned adjacency matrices arranged by F1 score (right better), with yellow indicating Ground Truth recovery while gree-blue indicating lower accuracy, as reconstruction task (first column), reconstruction of masked input task (second column), multi-step forecasting (third column) and one-step forecasting task (fourth column), alongside ground truth adjacency matrices generating synthetic data (last column), demonstrating AMAG's ability to recover the underlying spatial interactions doing forecasting tasks.

3.3.2 Evaluation on Forecasting with Neural Signals

Dataset and Preprocessing In this study, we recorded neural signals from two monkeys—Affogato (A) and Beignet (B)—each implanted with a μECoG array in their motor cortices. These arrays spanned five subregions (Primary Motor Cortex (M1), Premotor

Cortex, Frontal Eye Field (FEF), Supplementary Motor Area (SMA), and Dorsolateral Prefrontal Cortex (DLPFC)) and included 239 functioning electrodes, as described in [R 205]. During data collection, the monkeys performed a center-out reaching task toward one of eight possible directions. All procedures were approved by the University of Washington Institutional Animal Care and Use Committee. For the experiments described below, we used all electrode channels from Monkey A and only the M1 channels from Monkey B. During data collection, the subjects performed a reaching task towards one of eight different directions. The subject's hand position was tracked in real-time using a marker-based camera system (Optitrak) and mapped to control the position of a cursor with which they performed the point-to-point movements. The collected neural signals can be segmented into trials based on the stage of the behavioral task. Within each trial, the subjects were trained first to reach the center target, followed by the appearance of the surrounding target. Subsequently, a delay period ranging from approximately 50ms to 150ms was introduced. After the delay period, the subjects moved from the center target to the surrounding target. A trial is considered successful if the subjects successfully reach the surrounding target, resulting in a reward. For our study, we include the neural signals recorded during the 600ms following the appearance of the surrounding target as our samples. Specifically, we treat the first 150ms of neural signals as a preparatory context for forecasting the future 450ms of neural signals. Consequently, trials with a duration shorter than 600ms after the appearance of the surrounding target are excluded from the study. In our study, we use data collected from monkeys A and B across 13 different sessions. These sessions are combined to form a comprehensive dataset for analysis. The dataset is split into 80% for training, 10% for validation, and 10% for testing. Specifically, dataset A contains 985 training samples, 122 validation, and 122 testing samples, while dataset B contains 700 training samples, 87 validation samples, and 87 testing samples.

We further evaluated AMAG on two publicly available datasets recorded with Utah arrays in the M1 region of Monkey Mihili (M) and Monkey Chewie (C), both performing a similar reaching task [R 206, 207]. These M and C datasets provide Local Motor Potential (LMP) and power band activity covering eight frequency ranges (0.5–4 Hz, 4–8 Hz, 8–12 Hz, 12–25

Hz, 25–50 Hz, 50–100 Hz, 100–200 Hz, and 200–400 Hz), downsampled to 30 ms intervals. In contrast, the A and B signals were originally sampled at 25 kHz, so we applied a preprocessing pipeline to extract LMP and identical power band features [R 208, 209], thereby expanding the dimensionality of the recorded signals. For all four datasets, each reaching attempt was treated as an individual trial. In the M and C datasets, we aligned trials to movement onset, incorporating 300 ms before and 450 ms after onset. For A and B, we aligned trials to the appearance of the peripheral target and included 600 ms after that event. Similarly, we split the datasets into 80% training data, 10% validation data, and 10% test data, collected in 6 sessions for Monkey C and 7 sessions for Monkey M. The number of training, validation and testing samples for Monkey C is 1285, 160, and 160, respectively. For dataset M, we have 1023 training samples, 127 validation, and 127 testing samples. All the trials have at least 450ms after the target onset.

Benchmark Methods We compare AMAG with existing approaches, including variants of RNNs, and Transformers. Specifically, for one-step forecasting, we evaluate the following methods:

- *LRNN*, RNN model without a non-linear activation function and gating mechanism [R 210];
- *RNNf*, RNN model with non-linear activation and gating mechanism [R 50, 56];
- *LFADS*, RNN based model with pre-computed initial states [R 192].
- *NDT*, a decoder based transformer in temporal domain to estimate neural firing rate [R 8];
- *STNDT*, a model that is similar to NDT with additional attention based spatial interaction between neurons [R 54];
- *TERN*, a model that includes the GRU for encoding and temporal Transformer for decoding [R 50, 56].

- *RNN PSID*, RNN based model with 2-stage training for learning behavior-related neural space and behavior-unrelated neural space combined to ensure accurate future forecasting accuracy [R 62]; Notably, we exclude RNN PSID [R 62] from multi-step forecasting comparisons since learning behavior-related neural space in the first stage of RNN PSID requires access to all the previous steps at a given moment, which is not available in a multi-step forecasting scenario. In addition, we compare AMAG with GNN-based models. In particular, *GWNet*: A model which intertwines CNN and GNN for spatial interaction with a learnable adjacency matrix [R 167], *DCRNN*: embedded GNN in each GRU processing step using the predefined adjacency matrix [R 168], *GraphS4mer*: Sequential learning of temporal embedding and spatial interaction with time-dependent adjacency matrices [R 169]. For GraphS4mer we consider two variants, GRU based temporal learning (GS4-G) and state space model based temporal learning (GS4-S). Both GraphS4mer variants are excluded from one-step forecasting comparisons due to their dynamic graph learning design, which increases computational complexity and is unnecessary for single-step predictions.

Experiment Setup. For all four datasets, we generate future neural signals using a minimum of five context steps, corresponding to a 150ms time window. For one-step prediction, we employ a GRU architecture for both TE and TR. For multi-step prediction, we utilize Transformers for TE and TR. All models are trained using the Adam optimizer on a Titan X GPU. The initial learning rates are set to $1e-4$ for non-GNN methods and $5e-4$ for GNN-based methods. We evaluate performance using three key metrics: (1) R-squared (R^2) measures the proportion of variance in the future neuron recordings that the forecasted signal can explain. (2) Correlation ($Corr$) focuses on the matching of the trend of the ground truth signal and the forecasted signal. (3) Mean squared error (MSE) measures the L2 distance of the forecasted signal to the ground truth neuron signal. In detail, to optimize the performance of models, we conducted a hyperparameter search by exploring different values for the learning rate, hidden size, and weight decay. Specifically, we considered learning

rates of 10^{-3}, 5×10^{-4}, and 10^{-4}, hidden sizes of 64, 128, 512, 1024, and 2048, and weight decay values of 0, 10^{-4}, 10^{-5}, 10^{-6}, and 10^{-7}. We used the Adam optimizer to update the model parameters during training. We evaluate the model performance in terms of the R^2, correlation coefficient ($Corr$), and mean squared error (MSE) metrics on the validation set every 10 training epochs. The reported values in the tables of the main paper represent the testing set performance over the course of 500 epochs for most cases when the validation set achieves best performance. However, for the one-step forecasting task using the AMAG model and the STNDT-based models for both one-step and multi-step forecasting, we trained the models for 1000 epochs to ensure sufficient convergence and performance evaluation.

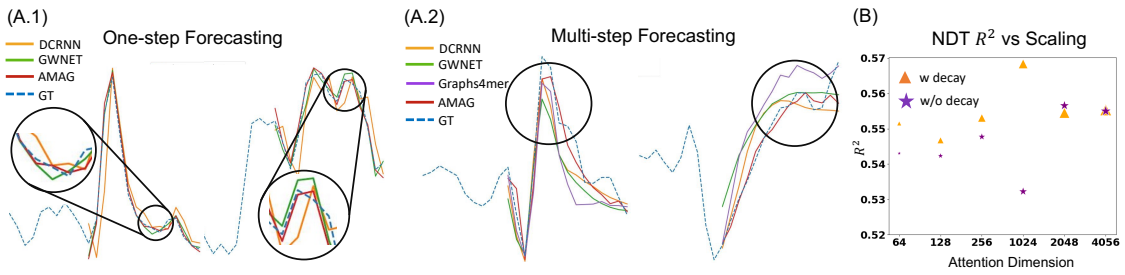

Figure 3.3: Panel A: Examples of predicted neuron trajectory for one-step (A.1) and multi-step (A.2) with representative methods DCRNN (orange), GWNet (green), Graphs4mer (purple), and AMAG (red). Ground Truth (GT) trajectories are shown in dashed blue. Panel B: Evaluation of the effect of attention dimension on NDT performance, with weight decay (w decay) in orange triangles, without weight decay (w/o decay) in purple circles. The larger size of the marker reflects the larger size of the model.

Forecasting Performance. We evaluate the performance AMAG to forecast future neural activities and present the results in Table 3.1 and Table 3.2, for one-step and multi-step forecasting, respectively. The standard deviation is obtained from three runs. Compared with the best non-GNN methods, AMAG improves the R^2 score by 8.2%, 5.0%, 5.7%, and 5.0% on Monkey M, C, B, and A datasets, respectively in Table 3.1. Transformer only

Table 3.1: ***One-step forecasting*** *results on Monkey M, C, B, and A with benchmark methods and* ***AMAG****.*

	Monkey M			**Monkey C**			**Monkey B**			**Monkey A**		
	$R^2\uparrow$	$Corr\uparrow$	$MSE\downarrow$	$R^2\uparrow$	$Corr\uparrow$	$MSE\downarrow$	$R^2\uparrow$	$Corr\uparrow$	$MSE\downarrow$	$R^2\uparrow$	$Corr\uparrow$	$MSE\downarrow$
STNDT [R 54]	0.778±4e-4	0.883±4e-4	0.0333±1e-4	0.860±3e-3	0.928±1e-3	0.0111±2e-4	0.857±2e-3	0.932±8e-4	0.0089±2e-4	0.879±1e-3	0.939±7e-4	0.0074±7e-5
NDT [R 8]	0.837±2e-3	0.915±2e-3	0.0292±2e-3	0.908±4e-4	0.953±1e-4	0.0074±3e-5	0.897±3e-4	0.950±1e-4	0.0058±2e-5	0.924±6e-4	0.962±2e-4	0.0046±4e-5
LFADS [R 192]	0.847±3e-4	0.922±2e-4	0.0274±8e-5	0.905±7e-4	0.953±2e-4	0.0074±3e-5	0.846±1e-4	0.923±1e-4	0.0088±9e-5	0.903±7e-4	0.951±2e-4	0.0060±5e-5
RNNf [R 56]	0.823±2e-4	0.911±1e-4	0.0250±4e-5	0.886±7e-4	0.943±3e-4	0.0091±5e-5	0.909±6e-4	0.954±2e-4	0.0052±4e-5	0.926±7e-4	0.963±2e-4	0.0045±4e-5
RNN PSID [R 62]	0.883±1e-4	0.940±1e-4	0.0108±2e-5	0.915±3e-4	0.957±6e-5	0.0071±2e-5	0.915±7e-4	0.957±3e-4	0.0048±5e-5	0.908±3e-4	0.953±4e-4	0.0049±1e-5
LRNN [R 210]	0.879±7e-4	0.937±4e-4	0.0137±7e-5	0.916±2e-4	0.957±7e-5	0.0067±1e-5	0.916±8e-4	0.957±4e-4	0.0047±4e-5	0.927±3e-4	0.963±8e-5	0.0045±2e-5
TERN [R 56]	0.866±7e-4	0.932±4e-4	0.0247±3e-4	0.920±9e-4	0.960±2e-4	0.0062±4e-5	0.888±1e-3	0.945±7e-4	0.0067±2e-4	0.929±4e-4	0.964±2e-4	0.0043±2e-5
DCRNN [R 168]	0.956±3e-3	0.978±1e-3	0.0190±3e-4	0.965±3e-3	0.983±2e-3	0.0026±3e-4	0.964±1e-4	0.982±8e-5	0.0020±4e-6	0.977±2e-3	0.988±8e-4	0.0014±1e-4
GWNet [R 167]	**0.971±5e-4**	**0.986±4e-4**	**0.0176±6e-4**	**0.985±3e-4**	**0.992±1e-4**	**0.0012±2e-5**	0.942±2e-3	0.971±1e-3	0.0033±1e-4	0.949±2e-3	0.974±1e-3	0.0031±1e-4
AMAG	0.965±1e-3	0.982±1e-3	0.0209±1e-3	0.972±1e-3	0.986±6e-4	0.0021±8e-5	**0.973±2e-3**	**0.986±1e-3**	**0.0015±1e-4**	**0.979±7e-4**	**0.990±4e-4**	**0.0013±4e-5**

Table 3.2: ***Multi-step forecasting*** *results on Monkey M, C, B, and A with benchmark methods and* ***AMAG****.*

	Monkey M			**Monkey C**			**Monkey B**			**Monkey A**		
	$R^2\uparrow$	$Corr\uparrow$	$MSE\downarrow$	$R^2\uparrow$	$Corr\uparrow$	$MSE\downarrow$	$R^2\uparrow$	$Corr\uparrow$	$MSE\downarrow$	$R^2\uparrow$	$Corr\uparrow$	$MSE\downarrow$
LFADS [R 192]	0.233±4e-3	0.485±3e-3	0.0780±4e-4	0.514±3e-3	0.725±2e-3	0.0387±2e-4	0.427±4e-3	0.672±6e-3	0.0338±4e-4	0.731±2e-3	0.855±1e-3	0.0164±1e-4
STNDT [R 54]	0.227±3e-3	0.480±4e-3	0.0780±4e-4	0.518±1e-3	0.720±5e-4	0.0387±2e-4	0.525±4e-3	0.725±2e-3	0.0288±2e-4	0.726±9e-3	0.852±6e-3	0.0167±6e-4
LRNN [R 210]	0.250±3e-3	0.500±2e-3	0.0757±1e-4	0.483±4e-3	0.697±2e-3	0.0416±2e-4	0.507±6e-3	0.725±1e-3	0.0286±4e-4	0.696±6e-4	0.833±1e-3	0.0187±3e-5
RNNf [R 56]	0.269±6e-3	0.515±4e-3	0.0746±4e-4	0.517±9e-3	0.725±4e-3	0.0382±7e-4	0.472±9e-3	0.694±6e-3	0.0314±5e-4	0.733±2e-3	0.856±8e-4	0.0163±1e-4
TERN [R 56]	0.257±6e-3	0.510±4e-3	0.0729±6e-4	0.548±5e-3	0.746±6e-3	0.0358±5e-4	0.559±3e-3	0.752±2e-3	0.0265±2e-4	0.746±1e-3	0.865±4e-4	0.0154±8e-5
NDT [R 8]	0.287±3e-3	0.528±3e-3	0.0723±3e-4	0.565±2e-3	0.752±1e-3	0.0348±1e-4	0.575±4e-3	0.773±3e-3	0.0250±2e-4	0.756±3e-3	0.873±1e-3	0.0149±2e-4
GWNet [R 167]	0.272±8e-3	0.524±1e-2	0.0721±8e-4	0.606±4e-3	0.779±3e-3	0.0309±4e-4	0.588±2e-3	0.769±2e-3	0.0242±4e-4	0.724±1e-3	0.851±2e-4	0.0168±9e-5
GS4-S [R 169]	0.181±7e-3	0.463±2e-3	0.0792±7e-4	0.553±3e-3	0.749±3e-3	0.0350±3e-4	0.600±7e-3	0.782±1e-3	0.0236±3e-4	0.740±5e-3	0.861±3e-3	0.0158±3e-4
GS4-G [R 169]	0.268±3e-3	0.530±3e-3	0.0730±1e-4	0.586±7e-3	0.769±4e-3	0.0322±6e-4	0.659±3e-3	0.812±3e-3	0.0194±1e-4	0.753±8e-4	0.869±6e-4	0.0149±5e-5
DCRNN [R 168]	0.288±3e-3	0.545±4e-3	0.0707±7e-4	0.606±2e-3	0.782±2e-3	0.0302±2e-4	0.635±4e-3	0.797±2e-3	0.0208±3e-4	0.756±2e-3	0.870±9e-4	0.0148±1e-4
AMAG	**0.331±4e-3**	**0.575±8e-4**	**0.0694±4e-4**	**0.657±2e-3**	**0.811±2e-3**	**0.0266±2e-4**	**0.665±2e-3**	**0.817±1e-3**	**0.0192±3e-4**	**0.763±4e-3**	**0.874±2e-3**	**0.0144±2e-4**

methods, STNDT [R 54] and NDT [R 8] achieve less optimal performance on four datasets. For instance, on Monkey M, the R^2 score is 0.883 for the best RNN based methods (RNN PSID) and 0.837 for the best Transformer based method (NDT). This could be due to the fact that the neural signals are continuous and future signals are primarily dependent on nearby precedents, which fits the RNN use cases better. Consequently, the advantage of the Transformer encoding long-term dependency may not be as prominent in this context. While Graph methods achieve comparable performance as shown in Table 3.1, AMAG shows consistent performance over four datasets, compared to GWNet which has variable accuracy

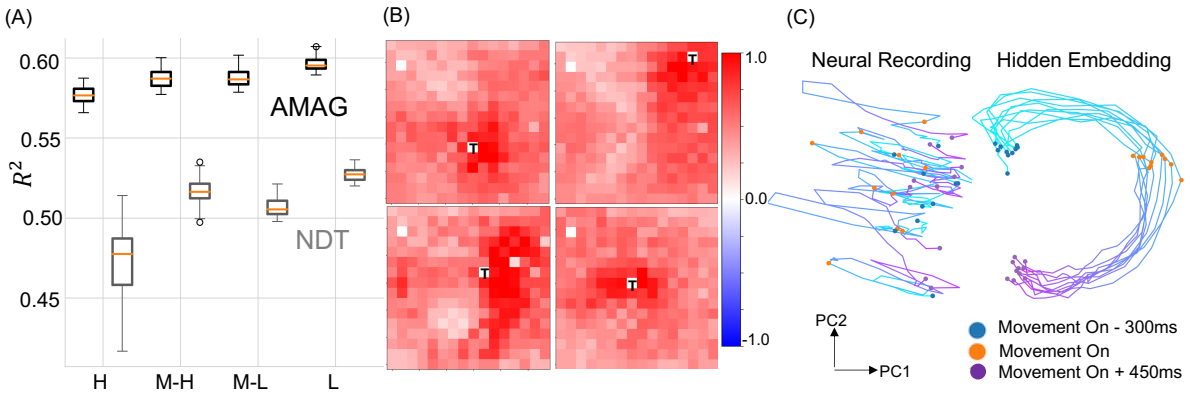

Figure 3.4: (A): Performance evaluation of masking channels from High (H), Mid-high (M-H), Mid-low (M-L), and Low (L) groups categorized by weight of learned adjacency matrix, showing the importance of high weight channel for forecasting performance. (B): Visualization of interaction strength of target channel to the reaming channels arranged according to spatial proximity in micro-ECoG array. (C): Examples of neural population trajectories in 2-dimensional PCA space projected from original neuron recording (left) and hidden embedding of AMAG (right).

on different datasets. Similar trends can be observed in multi-step forecasting results. As in Table 3.2, Graph-based methods, including AMAG, demonstrate better learning of future neural dynamics, highlighting the potential of GNNs in this task. GS4-G performs better than GS4-S on all four datasets, indicating that GRU based temporal encoding could be more suitable for neural signal forecasting. We find that AMAG is more optimal by almost 10% in R^2 on Monkey C and B when compared with non-GNN methods. Other GNN based methods, such as DCRNN, perform well, but AMAG exhibits additional improvement. For example, on Monkey C, AMAG achieves R^2 score of 0.657, whereas DCRNN achieves around 0.606. Another advantage of AMAG is that it dynamically learns the pruning of unnecessary edges in the adjacency matrix, while DCRNN and variants of GraphS4mer rely on a predetermined correlation matrix to define the connected edges which is typically

required to be sparse for optimal performance and handcrafted sparsification is needed. Thus, we prune the DCRNN, and GraphS4mer variants with K-Nearest Neighbour (KNN) pruning as mentioned in GraphS4mer such that each neuron is connected to half of the other neurons. Threshold pruning can be an alternative method for DCRNN, but we find that it is not optimal as the R^2 on Monkey C is equal to 0.573 and 0.445 for similarity thresholds of 0.5 and 0.8, respectively. These are lower than R^2 of 0.606 obtained with the KNN pruning.

In Fig. 3.3, we visualize the forecasted trajectory of the strongest baselines: DCRNN, GWNET, GS4-G, and AMAG. AMAG exhibits a higher closeness to the ground-truth signal (highlighted by black circles). This is consistent with behavior decoding from forecasted signals. Specifically, a linear behavior decoder is trained on ground truth neural and behavior signals (Monkey C). Then, the same decoder is used to decode behavior from AMAG and other GNN based baselines forecasted signals. Specifically, with the forecasted signals from the strongest baseline, we get decoding performance in terms of R^2 as follows: 0.350 (GS4-G), 0.350 (DCRNN), 0.440 (GWNet), 0.555 (AMAG), compared to 0.656 when using ground truth neural signals, demonstrating that AMAG achieves the best behavior decoding performance (0.555) compared to other GNN based methods regardless of a gap compared to using ground truth signal. Overall, the results demonstrate the advantage of AMAG over other methods and highlight the significance of utilizing GNN based methods for forecasting neuronal signals.

Investigating Adjacency Matrix Learned by AMAG. We hypothesize that the learned adjacency matrix in AMAG reflects interaction strength between channels, with channels exhibiting higher interaction strength playing a crucial role in forecasting. To test this hypothesis, we group channels into four categories—High (H), Mid-High (M-H), Mid-Low (M-L), and Low (L)—based on connection strength in A_a. The importance of each group is evaluated by measuring the performance drop in AMAG when channels in that group are masked. To assess the generality of channel importance, we apply the same procedure to NDT. The forecasting performance (R^2) of NDT and AMAG with masked channels as input are shown in Fig. 3.4 (A). In both models, masking channels consistently degrades performance, with the most significant drop occurring when channels in the H group are

Table 3.3: *Behavior Performance Comparison of Ground Truth (GT) and Graph-Based Methods.*

	GT	GraphS4mer	DCRNN	GWNet	AMAG
$R^2 \uparrow$	0.656	0.350	0.440	0.453	0.555

masked. This confirms our assumption that the weights learned in adjacency matrices provide meaningful insights into channel importance. Additionally, we investigate the relationship between learned weights and channel proximity. Figure 3.4 (B) illustrates the projection of the interaction strength of a target channel (T) onto other channels in the space organized by the μECoG arrangement. We observe that target channels exhibit stronger interactions with surrounding channels, which aligns with the properties of μECoG arrays. These findings collectively demonstrate that the learned adjacency matrix effectively identifies important channels, and the interaction strengths captured by AMAG align with the spatial organization of ECoG arrays.

Neuron Trajectory Alignment. Beyond learning meaningful adjacency matrices, it is also valuable to explore the neural trajectories captured by AMAG during the forecasting task. To investigate this, we project the original neural signals and their hidden embeddings from AMAG into 2D space using PCA, as shown in Fig. 3.4(C). We observe that while neural trajectories projected from the original signals appear tangled, with random start and endpoints, the trajectories projected from AMAG's hidden embeddings are disentangled and exhibit a circular structure, resembling patterns observed in prior studies [R 191]. This illustration indicates that AMAG learns informative features that can be visualized as coherent and interpretable neural trajectories in 2D space.

Computation Complexity. We additionally compare in Table 3.4 the number of parameters #P(M), training time per epoch T(S), and maximum memory cost during training M(GB). We observe that GNN based methods use much fewer parameters than non-GNN methods, with AMAG consistently using a minimal or comparable (DCRNN

Table 3.4: ***Computation complexity estimation*** *for multi-step and one-step forecasting task. Bold marks AMAG estimates and those methods whose estimates are better than AMAG.*

	Mutli-step Forecasting									
	AMAG	LRNN	LFADS	STNDT	NDT	RNNf	TERN	DCRNN	GWNet	GS4-G
#P(M)	**0.27**	4.59	5.53	2.38	1.06	3.55	9.93	0.60	1.12	1.52
T(S)	**9.74**	**2.18**	**3.44**	**4.70**	**5.96**	**7.22**	**8.48**	11.01	12.27	13.53
M(GB)	**5.74**	**0.12**	**0.15**	**0.19**	**0.04**	**0.11**	**0.34**	**1.69**	**1.53**	**4.48**
	One-step Forecasting									
#P(M)	**0.22**	1.25	5.84	8.31	3.70	3.55	9.93	**0.15**	1.12	N/A
T(S)	**9.90**	**1.97**	**3.29**	**4.61**	**5.94**	**7.26**	**8.58**	11.22	12.54	N/A
M(GB)	**5.36**	**0.04**	**0.15**	**0.43**	**0.10**	**0.11**	**0.34**	**1.05**	**1.53**	N/A

for one-step forecasting) number of parameters when compared to graph model baselines (DCRNN, GWNet, GS4-G). This demonstrates that the source of the contribution to AMAG performance is its architecture rather than the number of parameters. The graph architecture of AMAG provides the topological constraint across channels. The absence of the topological constraint in non-GNN based methods could result in overfitting with limited training samples. Adding regularization terms, e.g., weight decay could help, but will also limit models' capacity. We demonstrate it in Fig. 3.3 (B). Adding weight decay to NDT improves performance (orange triangle vs purple circle), however, as the attention dimension increases, the effect of regularization diminishes, and for dim¿1024, weight decay does not contribute to further improvement.

3.3.3 Ablation Study

Here we examine how each component in AMAG contributes to forecasting. Specifically, we consider the following ablated versions of AMAG: (1) removing self-connection in AMAG (amAG), (2) removing both the "Add" and "Modulator" module (–AG), (3) Only sample

Table 3.5: *Comparison between the Ablated versions of AMAG.*

	$R^2 \uparrow$	$CORR \uparrow$	$MSE \downarrow$
–AG	0.424±1e-3	0.650±1e-3	0.0456±1e-4
amAG	0.427±6e-3	0.652±4e-3	0.0453±5e-4
-MAG	0.647±2e-3	0.805±6e-4	0.0274±2e-4
$\tilde{a}$MAG	0.648±1e-3	0.806±4e-4	0.0273±6e-5
A-AG	0.652±9e-4	0.807±3e-4	0.0268±1e-4
aMAG	0.655±2e-3	0.810±1e-3	0.0269±2e-4
AM-G (Rand Init)	0.575±2e-2	0.767±7e-3	0.0329±1e-3
AM-G (Corr Init)	0.617±2e-4	0.786±7e-4	0.0296±2e-5
AMAG (Rand Init)	0.652±1e-3	0.807±8e-4	0.0270±1e-4
AMAG (Corr Init)	**0.657±2e-3**	**0.811±2e-3**	**0.0266±2e-4**

dependent "Adaptor" is kept in "Add" Module ($\tilde{a}$MAG), (4) Adjacency matrix is not learnable (AM-G), (5) The "Add" is ablated from the graph operation (-MAG), (6) "Modulator" is removed (A-AG), (7) The "Adaptor" in excluded from the "Add" (aMAG). The results are shown in Table.3.5. We find that removing *Add* or *Modulator* modules can cause the most significant forecasting accuracy drop (–AG) as well as the self-connection part in the graph (amAG). When keeping sample-dependent Adaptor term only ($\tilde{a}$MAG) in the Add module, the performance is very close to ablating the Add module (-MAG). The sample-dependent Adaptor term appears to be less important compared to other components but still reduces the accuracy if being ablated from AMAG (aMAG).

In addition, we investigate the effect of initialization (random versus correlation) when the adjacency matrices are not adaptable (AM-G), specifically, AM-G (Rand Init 1), AM-G (Rand Init 2), AM-G (Corr Init). The forecasting accuracy of AM-G compared to the other

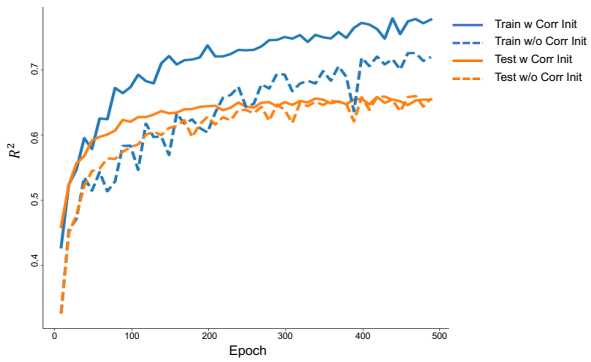

Figure 3.5: Visualization of R^2 score during the training and testing with the initialization of correlation matrix (solid curves) and random initialization (dashed curves). The correlation matrix initialized adjacency matrix stabilizes the training process.

two variants demonstrates the importance of correlation initialization. In particular, random and correlation initialization achieve similar performance when the adjacency matrices are adaptable, while random initialization leads to a less stable training process. In particular, both initialization methods result in a similar performance in terms of the final R^2 scores, i.e., 0.659 (randomly initialized) *vs.* 0.658 (correlation matrix initialized). However, their learning curves exhibit distinct characteristics, as depicted in Fig. 3.5. When the adjacency matrix is initialized randomly (dashed curves), both training and testing curves display more fluctuations. In contrast, when the adjacency matrix is initialized with the correlation matrix (solid lines), the learning curves become smoother. This indicates that the correlation matrix initialization promotes a more stable and consistent learning process. Furthermore, examining the R^2 values in the initial epochs, we observe that the model initialized with the correlation matrix achieved higher performance compared to the random initialization. This finding suggests that careful initialization with the correlation matrix can accelerate the training process, leading to improved performance during the early stages of training. In Fig. 3.6, we visualized the learned adjacency matrix with correlation initialization (top) and random initialization (bottom) for one-step (left) and multi-step forecasting (right). Globally, the

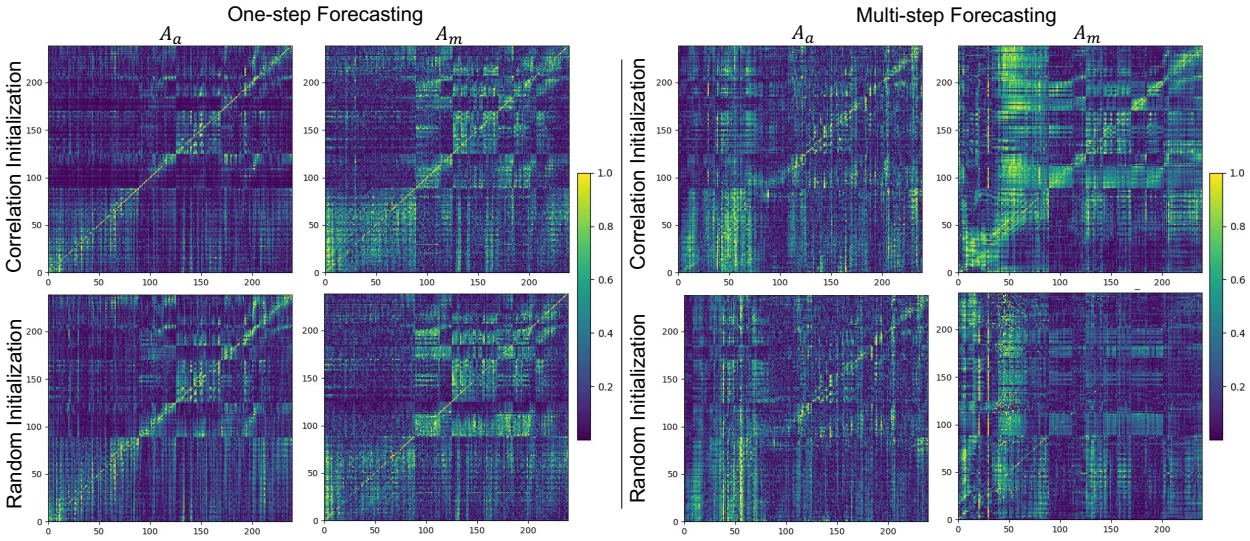

Figure 3.6: Visualization of the adjacency matrix of Add module (A_a) and Mudulator module (A_m) learned by AMAG performing one-step forecasting (left) and multi-step forecasting (right) using with correlation initialization (top) and random initialization (bottom).

learned adjacency matrices show similar connection patterns, while locally, the connection can be different with different initialization matrices. This indicates that regardless of the initialization method, AMAG can converge to graph patterns with globally similar connections while multiple local connections are available to achieve similar forecasting performance.

Chapter 4

ACTIVE LEARNING FOR EFFICIENT BEHAVIOR RECOGNITION

This chapter is a reproduction of published work in [2][1] with adaptations to fit the thesis structure and formatting.

4.1 Motivation

Recognizing actions from spatiotemporal data is crucial for a wide range of applications, including human activity analysis, human–robot interaction, and control systems. Unlike conventional video-based approaches that classify actions using raw image frames, skeleton-based methods focus on pose estimation features (e.g., joint positions or contours). This produces a compact yet expressive representation by filtering out background details and other extraneous information.

Numerous frameworks have been proposed to infer actions from these body-skeleton keypoints, typically by modeling the spatial and temporal dependencies within a sequence before assigning an action label [R 97, 211, 212, 100]. Despite often achieving high recognition accuracy, fully supervised approaches rely heavily on large annotated datasets—obtaining ground-truth labels for each sequence is both labor-intensive and dependent on expert input. This labeling bottleneck hampers scalability and limits deployment to new actions and subject populations. To address these challenges, unsupervised methods have been proposed. They

[1]In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of University of Washington's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink. If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

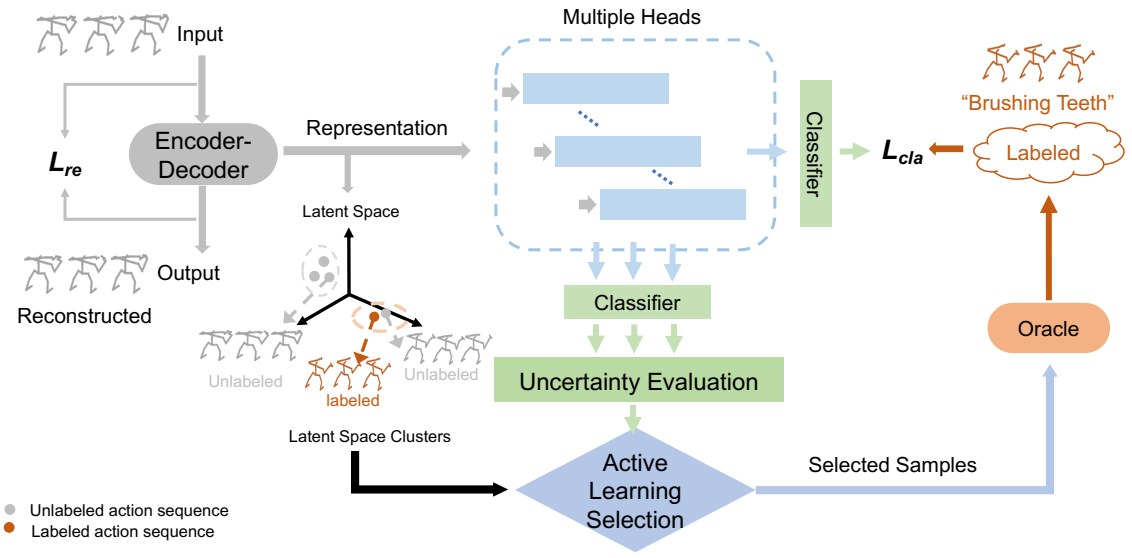

Figure 4.1: AL-SAR system overview. A multi-head mechanism (middle) operates on a learned hidden representation of an encoder-decoder (left) network, which extracts the latent representation using the reconstruction task. Consequently, these latent representations are grouped into Latent Space Clusters. The multi-head mechanism (middle) computes the uncertainty of samples through the estimation of the collective confidence of the heads on the prediction output. By incorporating the latent space information and the uncertainty evaluation, the active learning algorithm ensures that the selected samples are diversified and informative.

demonstrate the potential of a framework that includes two network components, an encoder, and a decoder, cooperating to reconstruct spatio-temporal sequences of keypoints [R 213, 105, 106]. By jointly training the encoder and decoder, these networks were found to self-organize the latent space shared between the encoder and the decoder to form clusters that correspond to actions. While these methods appear to be promising, accurately translating these clusters into action labels still demands extensive annotation efforts, ultimately constraining the benefits of an unsupervised workflow.

To propose an effective selection paradigm, we introduce Active Learning for Skeleton-based Action Recognition, termed AL-SAR, which actively selects key sequences for annotation based on both latent-space clustering and robust uncertainty estimation. AL-SAR selects sequences for annotation according to the clustering information in the latent space along with robust estimation of uncertainty. Our approach is a novel extension of the margin-based uncertainty selection strategy with a multi-head mechanism. Although we present AL-SAR in the context of skeleton-based action recognition, its principles generalize to other unsupervised sequence reconstruction and classification settings where annotations are costly. We evaluate AL-SAR on three widely used skeleton-based action recognition benchmarks—UWA3D [R 214], NW-UCLA [R 215], and NTU-RGB+D [R 98]. Our method demonstrates significant improvements over state-of-the-art unsupervised and semi-supervised baselines, especially under conditions where only a sparse set of labeled samples is available.

4.2 Method

AL-SAR works with datasets of multi-dimensional time series specifying the coordinates of body keypoints at each given time. We denote the times-series as $\mathcal{X} = \{\mathcal{X}_u \bigcup \mathcal{X}_l\}$, with $\mathcal{X}_u$ representing the sequences in the unlabeled set and $\mathcal{X}_l$ in the labeled set. At first, $\mathcal{X}$ has only unlabeled samples ($\mathcal{X} = \mathcal{X}_u$). A sample $\boldsymbol{X}_i \in \mathcal{X}$ is represented as a sequence $\boldsymbol{X}_i = [\mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_t, ...\mathbf{x}_T]$, where $\mathbf{x}_t$ is the vector of coordinates of the keypoints at time t, $\mathbf{x}_t \in \mathbb{R}^{N \times D}$. Here N corresponds to the number of keypoints, and D is the dimension of the keypoints (typically $D = 3$). N is expected to vary across datasets. For datasets with keypoints obtained from video frames recorded from multiple views (e.g., NW-UCLA, UWA3D), we follow the procedure of transforming them to a view-invariant representation [R 105, 216].

The proposed AL-SAR system includes three main components: (i) Learning the latent representation of skeleton sequences, (ii) The multi-head mechanism for robust computation of the margin-based metric, (iii) AL selection which integrates location information in the latent space and the margin-based metric to select samples for annotation. In this paper, we

focus on extending components (ii), and (iii) since, for (i), there are powerful pre-existing methods available. An overview of AL-SAR system architecture is depicted in Figure 4.1.

(i) Preliminary: Learning Meaningful Latent Representations. In AL-SAR, we embed the spatial-temporal body keypoints into latent representation space with the encoder-decoder framework introduced in [R 105, 106], which has been shown to achieve meaningful latent representation. The encoder uses bidirectional Gated Recurrent Units (GRU) and receives $\mathbf{X}_i$ as input. The vector $\mathbf{h}_i^T$ is the latent code transferred from the encoder at the last time step T to the decoder. It encodes the dynamic properties of the whole sequence $\mathbf{X}_i$ and lies in the latent space V, where $V = \{\mathbf{h}_i^T | \mathbf{h}_i^T = encoder(\mathbf{X}_i), \mathbf{X}_i \in \mathcal{X}\}$, i.e., the space spanned by the latent codes of all sequences. The unidirectional GRU-based decoder receives $\mathbf{h}_i^T$ and reconstructs the original input sequences by minimizing the reconstruction loss

$$\mathcal{L}_{re} = |\hat{\mathbf{X}}_i - \mathbf{X}_i|. \tag{4.1}$$

Notably, AL-SAR is not limited to the encoder-decoder framework and training strategy of P&C [R 105], and as we demonstrate in Experiments & Results, AL-SAR could achieve successful sample selection and classification with latent representation learned by other encoders.

(ii) Multi-head Mechanism. Pool-based active learning (AL) methods that rely on uncertainty measures can be susceptible to biased classification predictions, particularly when few labeled samples are available. Classifier outputs are influenced by factors such as network initialization and training regimen, and prior research indicates that models can exhibit overconfidence on certain instances [R 217]. Consequently, directly relying on classification outputs for uncertainty estimation may lead to misguided sample selection. The multi-head mechanism introduced here aims to mitigate this issue and yield more reliable measures of uncertainty.

Unlike the multi-head configurations employed in Transformer networks for attention weighting [R 162] or in CNN-based spectrogram inversion [R 218], our approach differs in

its structure, training paradigm, and intended purpose, though it likewise involves parallel components. Specifically, we randomly initialize several fully connected layers, leave their parameters fixed, and insert them in parallel immediately prior to the final classification layer. The classifier is then trained to correctly classify the labeled samples from the output of any head. During the selection phase, we combine the outputs of all heads to produce a robust uncertainty estimate.

The multi-head network is structured as follows. We consider classifier $\mathcal{C}$ as a single fully connected layer with weights W_θ. The multi-head mechanism is constructed as multiple additional heads receiving $\mathbf{h}_i^T$ as input and sending outputs to the classifier. Each head is a single fully connected layer with weights W_δ initialized according to the uniform distribution, i.e., $W_\delta \sim \mathcal{U}\left(-\frac{1}{\dim(\mathbf{h}_i^T)}, \frac{1}{\dim(\mathbf{h}_i^T)}\right)$, and kept fixed throughout the training process. Here $\dim(\mathbf{h}_i^T)$ denotes the dimension of the latent code. During training and testing, a single head is randomly chosen to be activated at each time. During operation, i.e., at the sampling phase, all heads are activated and average predictions are used to evaluate the uncertainty of the input samples. We show in Results (Section 4.3) that by generating multiple transformations of $\mathbf{h}_i^T$ and collectively contributing their different confidence to the knowledge of the unlabeled samples, the heads effectively reduce artifacts in estimation of uncertainty and improve the performance of the overall system. We use the margin-based uncertainty metric [R 150, 151], i.e., **marginal index** (***MI***), which evaluates the probability prediction p_z from each head j

$$p_j^l(\mathbf{X}_i) = p_j^l(\hat{y}_i = l | W_\theta, W_\delta) = \mathcal{C}(\mathcal{H}_z(\mathbf{X}_i))$$
$$j \in [1, N_h]; \text{and } l \in [1, C],$$

where p_z^l denotes the probability of a sample which belongs to class l predicted by the classifier $\mathcal{C}$ when the head z is activated. For C possible classes, $\mathcal{C}$ and $\mathcal{H}_z$ indicate the transformation with the classifier and the z^{th} head respectively. N_h denotes the number of heads. Given probability predictions, MI is computed as the measure of the confidence difference between the most probable class and the second most probable class, using the average probability

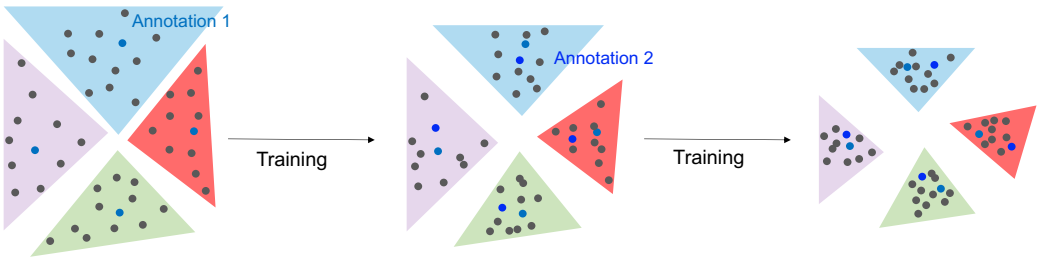

Figure 4.2: Illustration of latent space organization when the encoder-decoder network is trained with AL. The illustration of three training iterations depict that clusters in the latent space self-organize and co-adapt during training and annotation process. Gray points are unlabeled points, blue points are samples selected for annotation. In each iteration, clusters are formed in the latent space. The samples closest to the center of each cluster are annotated in the Initial Selection (Annotation 1). In subsequent iterations, samples for annotation are chosen from each cluster according to AL-SAR strategy (Annotation 2). The process is repeated for multiple iterations until it reaches the maximal annotation budget.

prediction of the heads, i.e.,

$$MI = \max_{l \in [1:C]} \left(\frac{1}{N_h} \sum_{j=1}^{N_h} \mathbf{p}_j \right) - \max_{l \in ([1:C] \setminus l^*)} \left(\frac{1}{N_h} \sum_{j=1}^{N_h} \mathbf{p}_j \right), \tag{4.2}$$

$$\text{Where, } \mathbf{p}_j = [p_j^1, .., p_j^l, .., p_j^C],$$
$$l^* = \underset{l \in [1:C]}{\text{argmax}} \left(\frac{1}{N_h} \sum_{j=1}^{N_h} \mathbf{p}_j \right).$$

Given the classifier output, the classification loss is computed as

$$\mathcal{L}_{cla}^i = \sum_{l=1}^{C} -y_i^l \log(p_j^l(\mathbf{X}_i)), \;\; j = \text{rand}(1 : N_h), \tag{4.3}$$

where $y_i^l = 1$ if $\mathbf{X}_i$ belongs to class l, and $y_i^l = 0$ otherwise. The loss, incurred for each sample $\mathbf{X}_i$, is composed from the reconstruction loss $\mathcal{L}_{re}^i$ and the classification loss $\mathcal{L}_{cla}^i$ for the labeled

samples. The full model is trained according to the total loss

$$\mathcal{L} = \sum_{\mathbf{x}_i \in \mathcal{X}_l} \mathcal{L}_{cla}^i + \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x}_i \in \mathcal{X}} \mathcal{L}_{re}^i, \tag{4.4}$$

where $|\mathcal{X}|$ is the total number of samples in the dataset.

(iii) Active Selection. In addition to the margin-based uncertainty measure (MI), we incorporate diversity filtering by leveraging clustering information in the latent space to enhance coverage and effectiveness of selected samples. We indeed observe that the inclusion of clustering information in AL boosts the overall classification. It presumably brings closer samples in the latent space that belong to the same class, while increasing the distances between distinct classes, as shown in Fig. 4.2. In each iteration, a new set of unlabeled samples is selected for annotation and then all samples (labeled and unlabeled) are used to refine the latent code and enhance the classifier. The selection can be subdivided into two scenarios: i) Initial Selection, ii) Subsequent Selection.

Initial Selection is regarded as the 'cold-start' problem, where neither ground truth annotation nor the predictions of the classifier are available [R 219, 220]. For effective initial selection, we form samples into clusters according to latent representation $\mathbf{h}_i^T$ generated by the encoder and then the samples in each cluster center are selected for annotation based on the assumption that these samples represent the clusters. Specifically, we use *K-Means* clustering to transform the latent representation into a *collection of clusters* $\mathcal{K}$. The *number of clusters* k

$$k = \frac{1}{N_{iter}} \times \ percentage \ \times |\mathcal{X}|, \tag{4.5}$$

is chosen based on the total number of selection iterations N_{iter} and the selection budget (the *percentage* of data we want to annotate). k is fixed across selection stages and is the budget for each selection. With the labeled samples, the classifier is trained until the classification accuracy on these labeled samples converges.

For Subsequent Selections, which will occur multiple times, we combine cluster information and MI computed by the multi-head structure. We choose the samples (s) with the minimum MI within every cluster K_i ($i \in [1, k]$). The selected samples are then passed to the 'Oracle'

Algorithm 1 AL-SAR Iterative Sample Selection Procedure

procedure AL-SAR

Number of training epochs N_{ep}, $\mathbf{h}_u^T$ latent representation for all unlabeled samples. Standard Deviationn (*std*).

Inputs: unlabeled samples $\mathcal{X}_u$, labeled samples $\mathcal{X}_l = \emptyset$, $n_{iter} = 0$

$\mathbf{h}_u^T \leftarrow Encoder(\mathbf{X}_i),\ \mathbf{X}_i \in \mathcal{X}_u$

$\mathcal{K} \leftarrow K\text{-}Means(\mathbf{h}_u^T, k)$

$\mathbf{s} \leftarrow Oracle([center(K_i) \text{ for } K_i \text{ in } \mathcal{K}])$

$\mathcal{X}_l \leftarrow \mathcal{X}_l \cup \mathbf{s}$

$\mathcal{X}_u \leftarrow \mathcal{X}_u \backslash \mathbf{s}$

for $\tau = 1{:}N_{ep}$ **do**

$\mathbf{ACC}_\tau \leftarrow$ *Classification Accuracy*

if $std(\mathbf{ACC}_{\tau-1}, \mathbf{ACC}_{\tau-2}, \mathbf{ACC}_{\tau-3}) < 0.01$, $\tau \geq 3$ and $n_{iter} < N_{iter}$ **then**

$\mathbf{h}_u^T \leftarrow Encoder(\mathbf{X}_i),\ \mathbf{X}_i \in \mathcal{X}_u$

$\mathcal{K} \leftarrow K\text{-}Means(\mathbf{h}_u^T, k)$

$\mathbf{s} \leftarrow Oracle([\text{argmin}(MI(K_i)),$

for K_i in $\mathcal{K}])$

$\mathcal{X}_l \leftarrow \mathcal{X}_l \cup \mathbf{s}$

$\mathcal{X}_u \leftarrow \mathcal{X}_u \backslash \mathbf{s}$

$n_{iter} \leftarrow n_{iter} + 1$

for annotation. A new set of k annotated samples is subsequently added to the labeled set and the model is continually trained with the loss of Eq. 4.4. We summarize the complete procedure in Algorithm 1.

4.3 Results

Datasets. We evaluate the performance of AL-SAR on three skeleton based action recognition benchmarks, *UWA3D Multiview Activity II (UWA3D)* [R 214], *North-Western UCLA (NW-*

Table 4.1: Comparison of fully supervised (FS), semi-supervised (SS) in top section, SOTA AL (mid section) accuracy with AL-SAR (bottom section) on three benchmarks of action recognition.

		UWA3D VIEW3				NW-UCLA				NTU RGB+D 60 CS			
	%Labels	5%	10%	20%	50%	5%	15%	30%	40%	1%	2%	5%	10%
	#Labels	25	50	100	250	50	150	300	400	400	800	$2K$	$4K$
Full & Semi Supervised	C	20.0	23.3	37.3	50.0	42.7	57.9	70.9	68.5	21.8	37.2	49.6	56.7
	RC	20.9	32.2	38.3	48.8	55.1	50.9	72.1	77.0	33.8	41.6	47.8	60.0
	IRC	21.7	29.6	41.5	55.4	50.7	59.3	78.6	78.0	36.7	42.7	53.9	61.2
	ASSL [R 107]	–	–	–	–	52.6	74.8	78.0	78.4	–	–	57.3	64.3
	MS^{2}L [R 108]	–	–	–	–	–	–	–	–	33.1	–	–	65.2
	SC3D [R 109]	–	–	–	–	–	–	–	–	35.7	–	**59.6**	**65.9**
AL	DIS	19.3	28.7	40.2	53.8	47.7	71.8	76.6	80.5	34.9	39.5	53.8	60.4
	CS	21.5	29.9	40.3	52.6	57.3	69.4	77.3	80.6	17.6	23.1	37.0	49.6
	AUG	21.9	30.2	40.8	**56.2**	48.2	65.8	77.0	81.7	20.7	33.5	50.4	60.2
	U	23.4	30.5	42.1	53.4	52.3	70.4	78.4	80.8	34.6	43.8	56.3	61.2
Ours	AL-SAR	**25.7**	**35.3**	**45.7**	53.9	**61.0**	**75.9**	**82.5**	**84.1**	**38.8**	**47.6**	57.9	63.7
	↑vs U	↑2.3	↑4.8	↑3.6	↑0.5	↑8.7	↑5.5	↑4.1	↑3.3	↑4.2	↑3.8	↑1.7	↑2.5

UCLA) [R 215], *NTU RGB+D 60* [R 98]. These datasets feature varying numbers of actions, subjects, and viewpoints, thus enabling a comprehensive assessment of our method under both cross-view (CV) and cross-subject (CS) conditions. ***UWA3D*** This dataset contains 30 action categories, each performed 4 times by 10 subjects, captured from frontal, left, right, and top viewpoints. Following standard practice [R 105, 221], we use the first two views for training and the third view for testing, as it represents a more challenging scenario.

NW-UCLA Recorded with three Kinect V1 cameras, NW-UCLA comprises 10 action classes, each performed by 10 different subjects 1–10 times. We follow the protocol of [R

Table 4.2: Comparing AL-SAR with its ablation versions: Uniform with multi-head (UH), no cluster with multi-head (NKH), average *MI* computed with Dropout (Drop), Using cluster without head (KNH).

			UWA3D VIEW3		**NW-UCLA**		**NTU 60 CS**	
% Labels	Cluster-	Multi-	5%	20%	5%	30%	1%	5%
# Labels	ing (K)	Head (H)	25	100	50	300	400	$2K$
Random Selection								
U	✗	✗	23.4	42.1	52.3	78.4	34.6	56.3
UH	✗	✓	25.1	41.3	54.9	76.9	33.8	54.9
Margin Based Selection								
MK	✓	✗	23.8	**49.7**	55.7	81.9	38.2	57.4
MKD	✓	Dropout	23.2	45.2	57.6	81.3	37.1	57.2
MH	✗	✓	23.2	42.5	57.2	81.0	34.9	56.6
AL-SAR	✓	✓	**25.9**	45.4	**59.4**	**82.2**	**38.9**	**57.9**

105, 94, 215], using the first two views as the training set and the remaining view as the test set.

NTU RGB+D 60 Collected from 40 subjects using 3 different cameras, this dataset includes 60 action classes. We evaluate AL-SAR in both cross-subject (CS) and cross-view (CV) settings. In the CV scenario, samples from cameras 2 and 3 are used for training, while camera 1 is used for testing. The CS setup splits the 40 subjects into two groups of 20 for training and testing, respectively—a more demanding task for unsupervised and semi-supervised methods [R 105, 107].

Implementation Details. In our experiments, the encoder is composed of three bi-directional GRU layers, each with 1024 hidden units per direction. The two directions' hidden states are concatenated, yielding a 2048-dimensional vector $\mathbf{h}_i^T$, which serves as input to the

decoder. The decoder is chosen to be a uni-GRU with the hidden size of 2048. Each head is implemented by a fully connected (FC) layer of size 2048x1024. A single-layer FC classifier $\mathcal{C}$ receives the output from these heads and produces class probabilities for each sample. Model parameters are optimized using Adam . The learning rate is set to 10^{-4} and then is multiplied by 0.95 to decay every 10 epochs on UWA3D and NW-UCLA, and every 3 epochs on NTU RGB+D. We employ 5 heads for UAW3D and NW-UCLA. For NTU RGB+D, we use 3 heads at 1% and 2% annotation budgets, and 5 heads at 5% and 10% budgets.

Comparison with SOTA AL Methods. Since AL methods specifically designed for action recognition are not available as far as we know, we compared AL-SAR with SOTA AL generic techniques as well as methods developed for other applications. In particular, we implement and examine Uniform sampling (U), Core-Set (CS) [R 134], Discriminator based selection (DIS) [R 139], and Consistency-based AL under Augmentation (AUG) [R 136]. We benchmarked these AL approaches against the margin-based selection strategy enhanced with a multi-head mechanism in AL-SAR. In Uniform Sampling (U), samples are randomly chosen for annotation from the entire dataset. Core-Set (CS) aims to comprehensively cover the feature space by selecting samples that minimize the overall coverage range [R 134]. Discriminator-based Selection (DIS) employs a discriminator to differentiate between labeled and unlabeled samples, as elaborated in the Related Work section [R 139, 141]. For Consistency-based AL under Augmentation (AUG), we applied skeleton sequence augmentation techniques introduced in [R 222] in conjunction with the consistency-based AL method proposed in [R 136].

The performance of AL-SAR is summarized in Table 4.1 for the UWA3D VIEW3, NW-UCLA, and NTU RGB+D 60 CS datasets. Our results demonstrate that AL-SAR consistently outperforms existing active learning (AL) methods, including Uniform Sampling (U). Specifically, with annotation budgets of 5% and 10% on the UWA3D View3 dataset, AL-SAR achieves accuracy improvements of 2.3 and 4.8 percentage points over the best baseline method, U. On the NW-UCLA dataset, AL-SAR exceeds the top-performing methods—Core-Set (CS) and Discriminator-based Selection (DIS)—by 3.7 and 4.1 percentage points at 5%

and 15% labeled samples, respectively. A similar pattern is observed in the NTU RGB+D 60 CS dataset, where AL-SAR outperforms DIS by 3.9 percentage points when only 1% of samples are annotated. Additional results on UWA3D V4 is shown in the Table 4.3. We can see that our proposed active learning method consistently outperforming other AL methods.

Table 4.3: Performance of different semi-supervised approaches (top), AL-SAR with SOTA AL methods (middle) AL-SAR on UWA3D View4.

		UWA3D VIEW4			
	% Labels	5%	10%	20%	50%
	# Labels	25	50	100	250
Full&Semi Supervised	C	22.4	25.5	42.5	54.4
	RC	22.0	30.5	38.7	54.4
	IRC	24.1	32.0	43.0	57.2
AL	CS	18.1	29.6	41.1	56.7
	DIS	20.0	32.3	42.9	56.9
	AUG	23.3	30.4	42.5	55.7
	U	24.8	33.2	43.5	56.9
		Ours			
	AL-SAR	**25.6**	**36.4**	**45.4**	**58.0**
	↑vs U	↑0.8	↑3.2	↑1.9	↑1.1

More results on NTU RGB+D CV is shown in the Table 4.4. Similar as NTU RGB+D 60 CS, when 1% and 2% of samples are labeled, AL-SAR outperforms the best AL methods by average 2.25 in terms of accuracy.

Table 4.4: Performance of different semi-supervised approaches (top), AL-SAR with SOTA AL methods (middle) AL-SAR on NTU RGB+D 60 Cross View (CV) dataset.

		NTU RGB+D 60 CV			
	% Labels	1%	2%	5%	10%
	# Labels	400	800	$2K$	$4K$
Full&Semi Supervised	C	28.7	38.7	53.7	61.2
	RC	35.1	45.0	55.7	66.1
	IRC	39.0	48.1	60.5	67.2
	ASSL [R 107]	–	–	63.6	69.8
	SC3D [R 109]	38.1	–	**65.7**	**72.5**
AL	CS	16.0	22.2	36.2	45.4
	DIS	31.2	46.3	57.6	65.2
	AUG	27.4	33.7	54.8	64.5
	U	39.3	46.5	59.0	65.5
		Ours			
	AL-SAR	**41.5**	**50.4**	61.7	69.4
	↑vs U	↑2.2	↑3.9	↑2.7	↑3.9

As the annotation budget increases, the relative improvement of AL-SAR compared to Uniform Sampling (U) generally diminishes, yet remains noteworthy. For example, with a 10% annotation budget (400K samples) on the NTU RGB+D 60 CS dataset, AL-SAR outperforms U by 2.5 percentage points in accuracy. Overall, AL-SAR consistently exhibits high efficiency across all benchmark datasets. This conclusion is further supported by Figure 4.3, which illustrates the number of labels required to achieve 80% accuracy for AL-SAR and other AL methods (U, CS, DIS), as well as baseline non-AL methods including C (encoder with

classifier trained solely on classification loss), RC (C enhanced with a decoder and trained with reconstruction loss), and IRC (a variation of RC initialized with a pre-trained encoder-decoder model) [R 105]. As shown in Figure 4.3 (left), AL-SAR only needs 20% of annotated samples to reach 80% accuracy, significantly reducing the labeling burden compared to C, which requires 70% of labels. While other baselines like RC and IRC perform better than C, they still necessitate 40%-50% of labeled data. Additionally, Figure 4.3 (right) demonstrates that when using a sparse set of 5% annotated samples, AL-SAR continues to improve its performance through successive learning iterations. In contrast, methods such as C and U quickly plateau after a few learning epochs, limiting their effectiveness with limited annotations. These analyses confirm that AL-SAR is highly efficient in utilizing sparsely annotated data, significantly enhancing overall action classification performance while minimizing the need for extensive labeling.

Comparison with SOTA Semi-Supervised Methods. In addition to comparing AL-SAR with other AL methods, it is of interest to compare AL-SAR with semi-supervised approaches for skeleton-based action recognition, such as ASSL [R 107], MS^{2}L [R 108], and SC3D [R 109], even though these methods assume *a given set of annotated samples* and do not deal with active sample selection. As presented in Table 4.1, AL-SAR outperforms these semi-supervised techniques when the number of annotated samples is limited (# Labels $\leq$ 800). However, with a larger set of annotated samples—such as 4000 samples (10% labels) on NTU RGB+D—ASSL and MS^{2}L outperform AL-SAR. This is expected, as these methods are specifically designed to capitalize on additional annotated data to enhance their performance. Similarly, SC3D achieves superior results with 2000 annotated samples, attributable to its more complex and expansive network architecture, which includes two GRU-based sub-networks and an additional graph convolutional neural network. Nonetheless, as detailed in the Ablation Study (subsection that follows), AL-SAR can be effectively integrated with SC3D to bolster its active sample selection strategy, thereby further improving classification accuracy. This synergy highlights the versatility and complementary strengths of AL-SAR when combined with existing semi-supervised frameworks. This comparison elucidates that

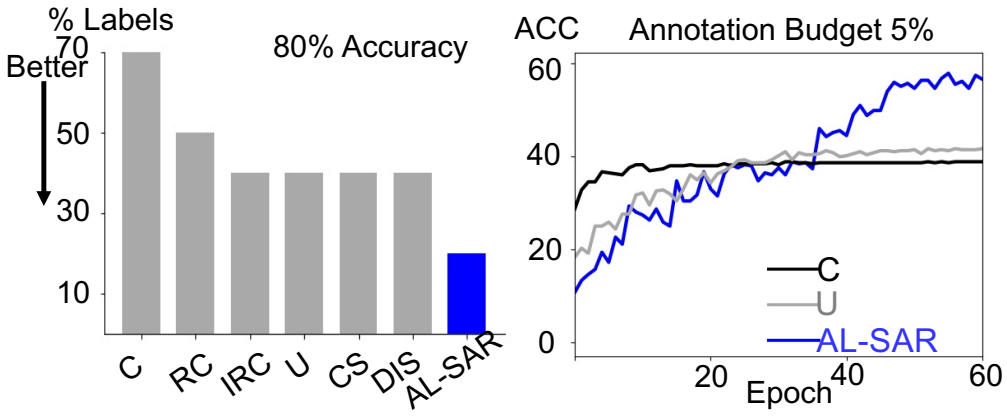

Figure 4.3: Left: Annotation (% of labeled samples) required to achieve 80% percent accuracy on NW-UCLA. Comparisons are made for C, RC, IRC, U, DIS, AL-SAR. Right: Training trajectory with 5% annotated samples for C, U, and AL-SAR on NW-UCLA showing that AL-SAR continues to improve with each new set of samples being selected for annotation. Selections are performed at epoch 1, 22, 29, 36, 41.

.

the main application of AL-SAR is when the annotation budget is small and when the annotation is performed iteratively along with inspection of the action clusters.

Ablation Study. We test how aspects of AL-SAR, such as latent space clustering with *k-Means* (K), multi-head structure (H), and margin-based selection (M) contribute to its overall performance. First, we test the influence of multi-head mechanism as an additional component to U, in abbreviation UH. Results in Table 4.2 show that UH is comparable to or underperforms U in many cases, e.g., UH accuracy is 1.5 and 1.4 lower than U on 30% NW-UCLA and 5% NTU RGB+D CS, respectively. We thus observe that integration of multi-head structure without marginal selection is not optimal. We, therefore, examine variants of marginal selection and test the effectiveness of multi-head components and clustering. In AL-SAR, *MI* is used as the uncertainty measure defined in Eq. 4.2. When the multi-head

Table 4.5: Evaluation of Original Space (OS), beta-variational autoencoder (β-VAE), vanilla Encoder-Decoder (ED) and Predict&Cluster (P&C) in terms of ability to produce meaningful latent states for classification.

	β-VAE	**OS**	**ED**	**P&C**
KNN Accuracy	41.5	66.5	82.9	83.6

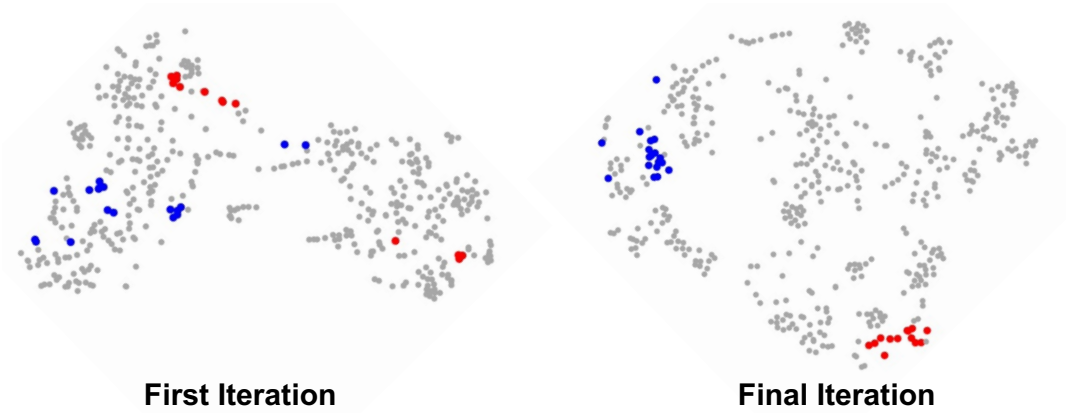

Figure 4.4: T-SNE embeddings of the latent space for the first iteration (left) and the final iteration (right) on UCLA dataset. Two representative classes belonging to actions of 'two hand punching' (red) and 'squatting' (blue) are marked. The first iteration is initialized with 10% annotated samples, where in each iteration additional 10% of samples are annotated. At the final iteration, the same class samples are gathered into more enhanced clusters

mechanism is removed, MK variant), the equation for MI becomes

$$MI = \max_{l \in [1:C]} (\mathbf{p}) - \max_{l \in ([1:C] \setminus l^*)} (\mathbf{p}) \,, \tag{4.6}$$

where $\mathbf{p}$ is the probability output of the classifier taking as input the latent representation $\mathbf{h}_i^T$. As presented in Table 4.2, AL-SAR outperforms MK across all evaluated datasets and annotation budgets, with the exception of the UWA3D View3 dataset at a 20% annotation

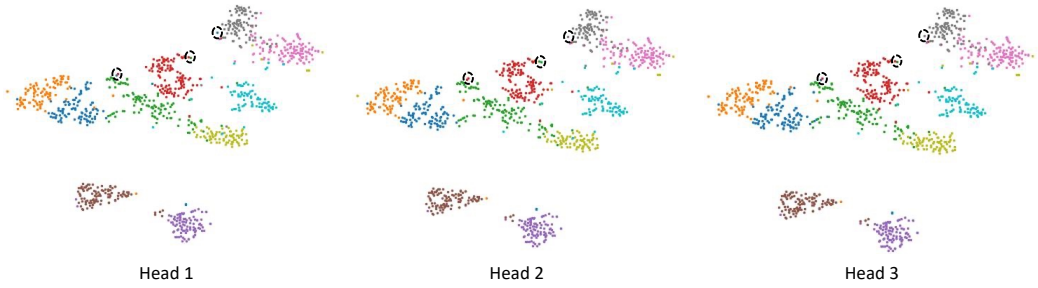

Figure 4.5: Visualization of the classifier predictions in t-SNE embedded latent space. The predictions are obtained from the well-trained classifier as it is connected to different heads, and colors indicate class labels. The classifier makes consistent predictions across three heads except for a few examples (denoted by black dashed circles).

budget. Detailed results from the ablation study examining the impact of varying the number of heads in AL-SAR are provided in Table 4.6 where the accuracy AL-SAR achieved with 3, 5, or 10 heads are close. However, adding more heads does not constantly improve performance, e.g., the performance of AL-SAR using 20 heads (AL-SAR 20) is lower than other AL-SAR variants, which is consistent with the comparison between U and UH in Table II of the main paper, i.e., without active selection in AL-SAR (U), simply adding the fixed head to U, as UH, is not beneficial. This is probably because, as more heads are included in the multi-head mechanism, the classifier may not be able to generalize to all heads, especially when the action recognition task is hard (UWA3D VIEW3).

Another approach to implementing a multi-head mechanism involves the use of Dropout [R 223], which introduces stochastic noise into the network and estimates probability outputs through multiple forward passes. Our comparative analysis indicates that MKD does not achieve the same level of classification accuracy as AL-SAR. This discrepancy may be attributed to the fact that Dropout-induced noise is not as systematic or structured as the predefined transformations executed by the multi-head mechanism in AL-SAR. Unlike Dropout, which randomly deactivates neurons to prevent overfitting, the multi-head approach

Table 4.6: Comparison of MK (no head) and AL-SAR variants with different number of heads: AL-SAR-3 (3 heads), AL-SAR-5 (5 heads), AL-SAR-10 (10 heads), AL-SAR-20 (20 heads).

	UWA3D	**VIEW3**	**NW-UCLA**	
% Labels	5%	20%	5%	30%
# Labels	25	100	50	300
MK	23.8	**49.7**	55.7	81.9
AL-SAR-3	**25.9**	45.4	59.4	82.2
AL-SAR-5	25.7	45.7	**61.0**	82.5
AL-SAR-10	25.4	45.8	57.8	**83.6**
AL-SAR-20	22.7	40.3	59.0	81.8

in AL-SAR employs parallel, fixed transformations that provide a more consistent and reliable estimation of uncertainty, thereby enhancing classification performance.

We study the effectiveness of clustering in latent space with MH variant in Table 4.2, where clustering is ablated from AL-SAR. MH underperforms AL-SAR by an average of 2.7. Indeed, clustering in the latent space is considered for diversity selection since the space has been shown to self-organize into meaningful structures [R 105]. As we show in Fig. 4.4, t-SNE representation of the latent space in the first and final iteration of AL indicates that training that leverages clustering succeeds to form clusters of samples that are clearly identified with action, and thus ensures selection diversity.

Different methods to construct latent space can potentially influence the final accuracy. We assess how well the latent space obtained by beta-variational autoencoder (β-VAE) [R 224], vanilla encoder-decoder (ED), and Predict&Cluster (P&C) [R 105] construct discriminative clusters. In all three variants, we evaluate the accuracy of KNN (k=1) on NW-UCLA and use the same encoder and decoder for comparison. We report the performance in Table 4.5 along with the baseline of KNN accuracy (k=1) on the original input space (OS). P&C architecture

yields the highest accuracy and thus we choose this architecture for AL-SAR experiment.

Table 4.7: Comparison of vanilla SC3D, AL-SAR with the encoder-decoder of P&C (AL-SAR-PC), and AL-SAR with the encoder in SC3D (AL-SAR-SC3D) on NTU RGB+D 60 CS.

% Labels	1%	5%	10%
SC3D	35.7	59.6	65.9
AL-SAR-PC	38.8	57.9	63.7
AL-SAR-SC3D	**40.0**	**63.8**	**72.7**

In addition, we are interested in whether a powerful semi-supervised learning method could be combined with AL-SAR. One such method is SC3D [R 109], which shows strong action recognition accuracy on NTU RGB+D dataset under 5% and 10% annotation budget (Table 4.1). SC3D's primary relevance to our work is in learning the data representation for AL-SAR as an alternative to P&C [R 105]. To test the combination, we substitute encoder-decoder of P&C with the encoder of SC3D to test whether this substitution will enhance the results of AL-SAR with P&C and/or the results of SC3D alone. The results are shown in Table 4.7. We denote the original AL-SAR as AL-SAR-PC to distinguish with SC3D substituted AL-SAR (AL-SAR-SC3D). AL-SAR-SC3D surpasses the performance of both SC3D and AL-SAR-PC on three annotation budgets 1%, 5%, and 10%. Specifically, for 10%, the improvement is 6.8 and 9.0 with respect to SC3D and AL-SAR-PC, respectively. The results indicate that incorporating a more complex model in AL-SAR, such as SC3D, could improve the accuracy and demonstrates the versatility of AL-SAR in discovering informative samples from latent representations embedded with different encoders.

The Multi-head Mechanism. As described previously, the multi-head module consists of a set of randomly initialized, fixed fully connected layers inserted immediately before the final classification layer. Evaluating the impact of these heads on classifier predictions is of

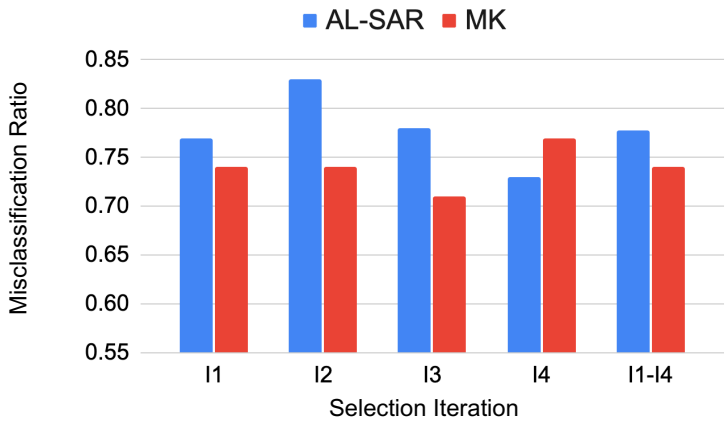

Figure 4.6: Misclassification ratio within selected samples. Comparisons are made between the AL-SAR (blue bar) and MK (red bar) in four iterations: iteration 1 (I1), iteration 2 (I2), iteration 3 (I3), iteration 4 (I4), and integration of the four iterations (I1-I4). In most cases, AL-SAR has a higher misclassification ratio indicating the preference of AL-SAR for selecting more informative misclassified samples.

particular interest. Figure 4.5 illustrates the classifier's predictions in the latent representation space when interfaced with different heads, revealing that the outputs remain nearly identical regardless of the head used. This consistency indicates that the classifier robustly generalizes across the outputs of various heads.

To further examine the utility of the multi-head mechanism in the context of active sample selection, we conducted a quantitative comparison of AL-SAR (Eq. 4.2) and MK (Eq. 4.6) in their ability to identify misclassified samples. Specifically, we calculated the misclassification ratio for the samples selected over four iterations, where MI is computed in Fig 4.6. AL-SAR generally achieved a higher misclassification ratio than MK in most iterations, with the exception of iteration 4 (I4). When aggregated across all iterations, AL-SAR consistently prioritized the selection of misclassified samples, which are presumed to be more valuable for improving classifier performance than correctly classified ones. Additional visualizations

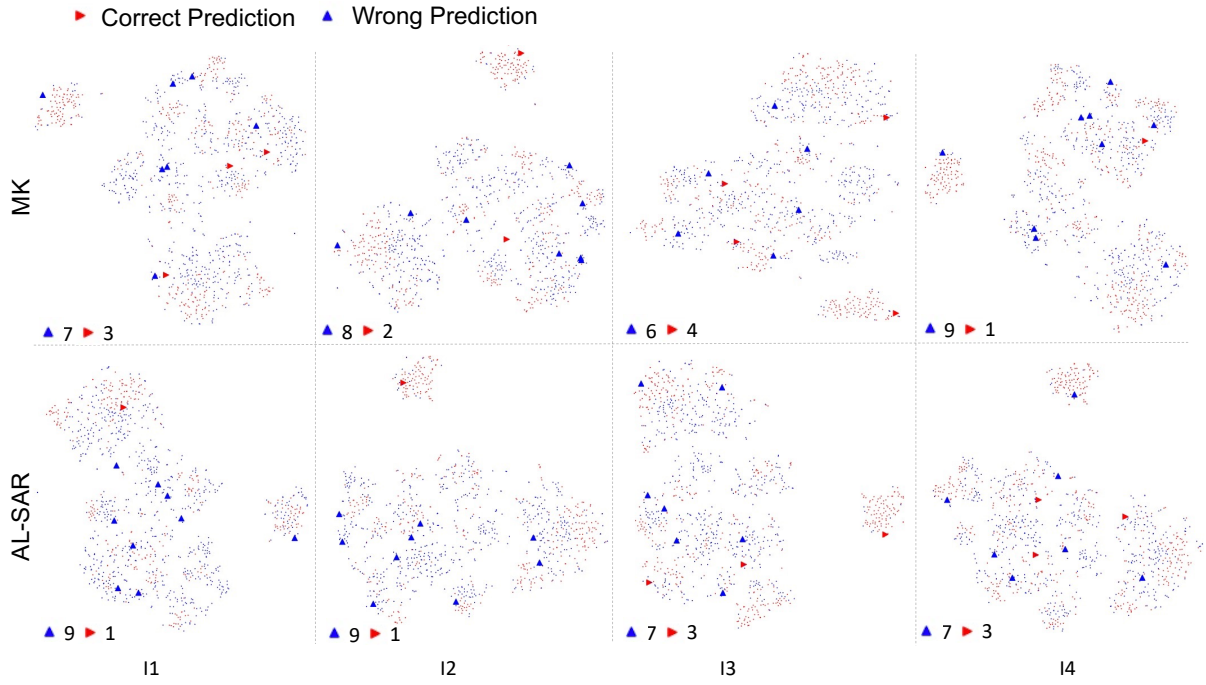

Figure 4.7: Visualization of t-SEN projected latent representation of selected samples with MK (no head) and AL-SAR in four selection iterations: iteration 1 (I1), iteration 2 (I2), iteration 3 (I3), iteration 4 (I4). Large triangles are selected points in the current iteration; small points indicate other samples. Samples that are correctly classified are in red, otherwise in blue. The number of wrong predictions (blue triangles) and correct predictions (red triangles) in selected samples are shown on the bottom left in each subplot.

detailing the spatial distribution and significance of these samples are provided in Fig. 4.7. Specifically, we visualize the localization and the correctness of samples selected either by MK without multi-head mechanism (top row) and AL-SAR (bottom row) in 2D space on NW-UCLA dataset under 5% selection budget. Both methods select samples scattered in the latent space, ensuring the diversity of the selection, but compared to MK, AL-SAR is less likely to select samples that are already been correctly recognized by the classifier, which may not be informative to the classifier.

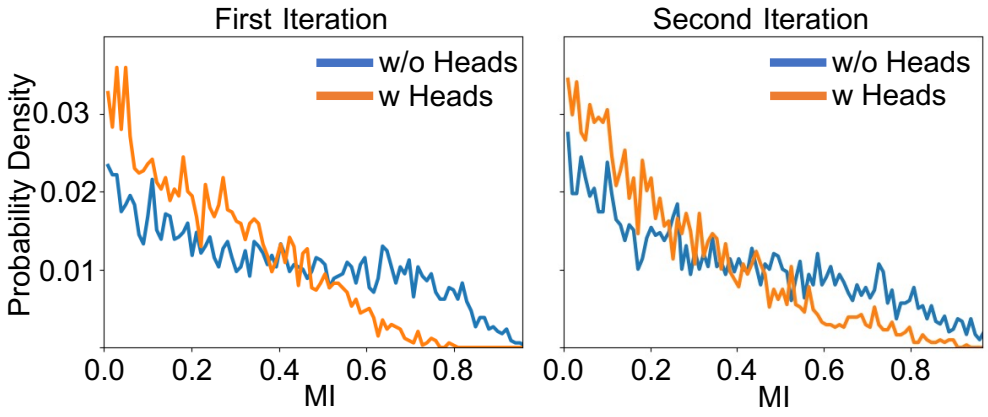

Figure 4.8: *MI* distribution among the mis-classified samples in the presence (w) and absence (w/o) of multi-head structure at two stages: first iteration (left) and second iteration (right). Compared to w/o Heads, *MI* computed with heads is less likely to be larger than 0.6.

Previous studies have shown that classifiers may exhibit overconfidence on misclassified samples, potentially resulting in misleadingly high MI values (indicating low uncertainty) [R 217, 225]. To address this issue, we investigate whether the multi-head mechanism computed *MI* can alleviate the artifact associated with overconfident predictions. We compare the *MI* distributions of misclassified samples with multi-head mechanism and without the multi-head mechanism over two selection iterations (Fig 4.8). The density at the high *MI* regime could indicate the extent of over-confidence since, presumably, the *MI* of misclassified samples should be low. As shown in Fig 4.8, in the high MI regime, the density of *MI* generated by the multi-head mechanism is lower (orange curve) than the one without the multi-head mechanism (blue curve). This indicates that, with the multi-head mechanism that integrates the classifier outputs from all the heads to guide the sample selection, the estimated *MI* is less likely to be affected by the overconfident classifier, thus constituting a more robust uncertainty estimation.

Chapter 5

ANIMAL BEHAVIOR RECOGNITION WITH OPEN USER INTERFACE

5.1 Motivation

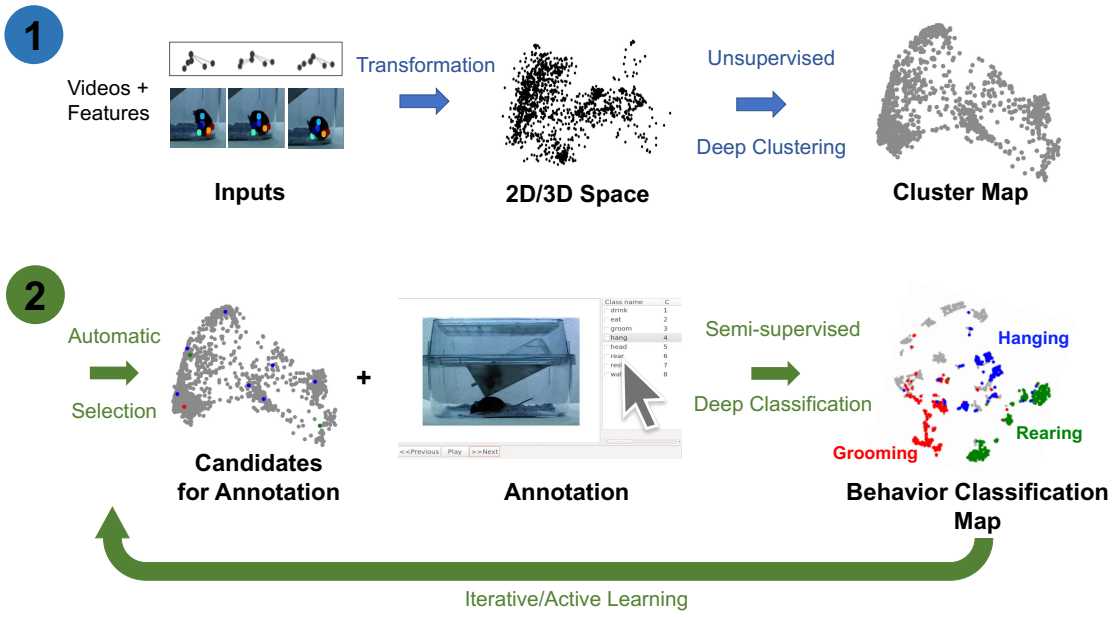

Figure 5.1: OpenLabCluster overview. (1) Clustering: Input of body keypoints segments is mapped to low dimensional space. Unsupervised encoder-decoder maps them to a Cluster Map. (2) Classification: Active Learning algorithms automatically select candidates for annotation after which Cluster Map is reshaped into the Behavior Classification Map where each point is associated with a behavioral state (Grooming (red), Hanging (blue), Rearing (blue)). Mouse images are reproduced from frames of videos provided in the dataset of [9].

Understanding and interpreting animal behavior is critical to numerous biological investigations, ranging from ethological research to behavioral assays aimed at elucidating biological mechanisms [R 226, 21, 227, 228, 229, 230, 231, 232, 71, 233]. Robust, uninterrupted, and high-resolution behavioral observations are essential, and over decades, researchers have employed diverse recording modalities—such as video, audio, and physical markers—to capture animal behavior [R 234, 235, 236, 237, 238, 239, 240]. Recent advancements in recording technologies have enabled prolonged observations in varied environments and with multiple modalities. This progress necessitates effective organization, interpretation, and classification of recordings into distinct behavioral states—a task that is time-consuming and requires specialized expertise when performed manually. Consequently, the development of automated methodologies to accelerate behavioral classification, while minimizing human involvement, is of paramount importance [R 241, 242, 243].

Early efforts in automatic behavior classification focused on raw video analysis using machine learning techniques such as Convolutional Neural Networks (CNNs) [R 83, 84, 85, 86], Recurrent Neural Networks (RNNs) [R 87, 88], Temporal Gaussian Mixture models [R 89], and temporal CNNs [R 90]. Although effective in specific scenarios, video-based methods often incorporate extraneous background information and noise (e.g., camera artifacts), which can undermine reliability and require considerable computational resources due to the high-dimensional nature of video data [R 90]. In contrast, approaches that concentrate on movement by utilizing body keypoints or kinematics—extracted from video frames—can circumvent these limitations [R 10, 244, 245, 11, 246].

Markerless pose estimation techniques, such as OpenPose [R 66], DeepLabCut [R 67, 68] and their subsequent improvements [R 69, 70], enable accurate keypoint detection without the need for physical markers. Numerous other tools and approaches have also been developed to advance animal pose estimation [R 71, 72, 73, 74, 75, 76, 77, 247]. Once body keypoints are estimated, behavioral segmentation can be achieved using unsupervised clustering methods—such as HBDSCAN [R 248, 249], hierarchical clustering [R 250], and the Watershed algorithm [R 251, 252, 253]—which group similar postural states and dif-

ferentiate distinct behaviors [R 21, 249]. Dimensionality reduction techniques, including Principal Component Analysis (PCA) and Uniform Manifold Approximation and Projection (UMAP), further enhance the representation of body keypoints for effective clustering [R 254, 250, 255, 249, 256].

Recent deep learning methods have advanced latent keypoint representation learning through task-specific optimization, as demonstrated by TREBA [R 257] and its automated extension, AutoSWAP [R 258]. Contrastive learning approaches have also been explored to refine the latent space by drawing together similar behavioral samples and separating dissimilar ones [R 259, 58], although the selection of appropriate positive and negative samples remains challenging without expert guidance.

General methods such as Predict&Cluster [R 105] and VAME [R 260] address these challenges by focusing on sequence reconstruction and future prediction, enabling the unsupervised clustering of behavioral patterns [R 105, 260, 261]. However, while unsupervised clustering effectively identifies similar behavioral patterns, it may not isolate behaviors of specific interest. Supervised classification methods overcome this limitation by mapping behavioral segments to predefined categories using annotated training data [R 10, 11, 12]. The accuracy of these classifiers depends critically on both the classifier design and the quality and quantity of the annotations. Early successes using classical machine learning [R 262, 9, 12, 263, 264, 265, 266, 249] have recently been supplemented by deep learning approaches [R 267, 268, 259, 247, 11]. Nevertheless, manual annotation remains labor-intensive and subject to inter-annotator variability.

To mitigate the burden of manual annotation, methods such as SaLSa—which assigns uniform labels to pre-computed unsupervised clusters—and JAABA—which provides an interactive framework for correcting misclassifications—have been developed [R 268, 269]. Active learning (AL) techniques further streamline the process by automatically selecting samples for annotation, balancing annotation effort with classification accuracy [R 131, 132, 133, 270]. For instance, A-SOiD employs AL to prioritize samples with high prediction uncertainty [R 270]; however, uncertainty-based selection may inadvertently target redundant

samples. Integrating clustering information with classifier uncertainty could improve the efficiency of sample selection.

In this work, we extend prior methods by jointly learning representations for active learning and classifier training using pose estimates derived from DeepLabCut [R 67, 69, 68] applied to animal videos. We introduce the OpenLabCluster toolset—an active learning-based, semi-supervised behavior classification platform with a graphical interface for classifying animal behavior from body keypoints. The system implements multiple semi-supervised active learning strategies in an iterative framework. In each iteration, a subset of candidate segments is automatically selected for annotation, thereby progressively enhancing the accuracy of clustering and classification. OpenLabCluster comprises two main components (Fig. 3.1A): (1) an unsupervised deep encoder-decoder that generates Cluster Maps to visualize the representation space and its groupings, and (2) an iterative mechanism for the automatic selection of representations for annotation, which subsequently produces Behavior Classification Maps where each point is re-positioned and associated with a behavioral class. These operations are performed through the simultaneous training of a clustering encoder-decoder and a deep classifier. OpenLabCluster is released as an open-source graphical user interface (GUI), designed to empower scientists—regardless of their expertise in deep learning—to perform animal behavior classification, while also offering advanced options for expert users.

5.2 Experiments and Results

Datasets. Behavioral states and their dynamics vary from species to species and from recordings to recordings. We use four different datasets to demonstrate OpenLabCluster applicability to various settings. The datasets include videos of behaviors of four different animal species (Mouse [R 9], Zebrafish [R 21], *C. elegans* [R 22], Monkey [R 73]) with three types of motion features (body keypoints, kinematics, segments), as depicted in Fig. 5.2. Two of the datasets include apriori annotated behavioral states (ground truth) (Mouse, *C. elegans*), while the Zebrafish dataset includes ground truth a priori predicted by another

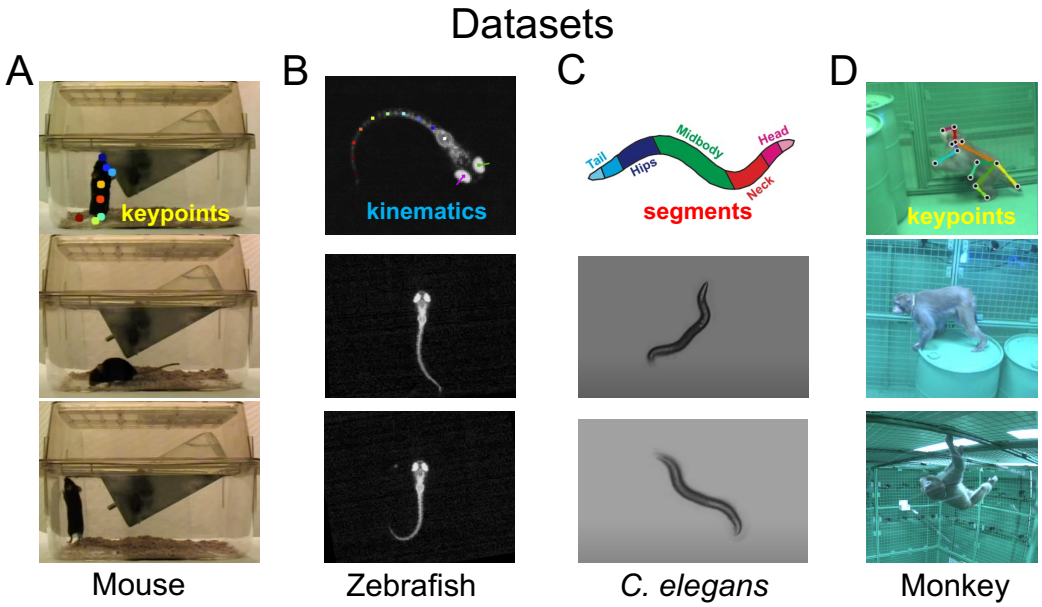

Figure 5.2: Visualization of four animal behavior datasets. (A) Home-Cage mouse dataset (Mouse) (B) *C. elegans* Movement Dataset (*C. elegans*) (C) Zebrafish free swimming dataset (Zebrafish) (D) OpenMonkeyStudio Macaque behaviors dataset (Monkey). The top row shows positions of extracted keypoints for each dataset. Images sources: A. Images are reproduced from frames of videos in the dataset of [9]. B. Images are reproduced from Figure 1 and frames of videos of [21]. C. Images are reproduced from Supplementary Figure 1 and images of [22]. D. Images are reproduced from Figure 8 and images of [73].

method, and the Monkey dataset does not include ground truth annotations. Three of the datasets have been temporally segmented into *single-action clips* (Mouse, Zebrafish, *C. elegans*), i.e., temporal segments while the Monkey dataset is a continuous recording that requires segmentation into clips. We describe further details about each dataset below.

Home-Cage Mouse. The dataset includes video segments of 8 identified behavioral states [R 9]. In particular, it contains videos recorded by front cage cameras when the mouse is moving freely and exhibits natural behaviors, such as drinking, eating, grooming, hanging,

micromovement, rearing, walking, and resting. Since keypoints have not been provided in this dataset, we use DeepLabcut [R 67, 69, 68] to automatically mark and track eight body joint keypoints (snout, left-forelimb, right-forelimb, left-hindlimb, right-hindlimb, fore-body, hind-body, and tail) in all recorded videos frames. An example of estimated keypoints overlaid on top of the corresponding video frame is shown in Fig. 5.2A (top). To reduce the noise that could be induced by the pose estimation procedure, we only use the segments for which DeepLabCut estimation confidence is high enough. We use 8 sessions for training the models of clustering and classification (2856 segments) and test classification accuracy on 4 other sessions (393 segments).

Zebrafish. The dataset includes video footage of zebrafish movements and was utilized in [R 21] for unsupervised behavior clustering using 101 precomputed kinematic features, a procedure that identified 13 clusters which were manually related to 13 behavior prototypes including: approach swims (ASs), slow types 1 (S1), slow types 2 (S2), short and long capture swim (SCS and LCS), burst type forward swim (BS), J-turns, high-angle turn (HAT), C-start escape swims (SLC), long latency C-starts (LLC), O-bends, and routine turns (RTs), spot avoidance turn (SAT). In the application of OpenLabCluster to this dataset, we utilize only a small subset of these features (16 features) and examine whether OpenLabCluster is able to generate classes aligned with the unsupervised clustering results obtained on full 101 features (as the ground truth). We use 5294 segments for training and 2781 segments for testing.

C. elegans. The dataset is recorded with Worm Tracker 2.0 when the worm is freely moving. The body contour is identified automatically using contrast to background from which kinematic features are calculated and constitute 98 features that correspond to body segments from head to tail in 2D coordinates, see [R 22] and Fig. 5.2C. Behavioral states are divided into three classes: moving forward, moving backward, and staying stationary. We use ten sessions (a subset) to investigate the application of OpenLabCluster to this dataset, where the first 7 sessions (543 segments) are used for training and the remaining 3 sessions (196 segments) are used for testing.

Monkey. This dataset is from OpenMonkeyStudio repository [R 73] and captures freely

moving macaques in a large unconstrained environment using 64 cameras encircling an open enclosure. 3D keypoints positions are reconstructed from 2D images by applying deep neural network reconstruction algorithms on the multi-view images. Among the movements, 6 behavioral classes have been identified. In contrast to other datasets, this dataset consists of continuous recordings without segmentation into action clips. We thereby segment the videos by clipping them into fixed duration clips (10 frames with 30 fps rate) which results in 919 segments, where each segment is ≈ 0.33 seconds long. OpenLabCluster receives the 3D body key points of each segment as inputs. Notably, a more advanced technology could be implemented to segment the videos as described in [R 271]. Here, we focused on examining the ability of OpenLabCluster to work with segments that have not been pre-analyzed and thus used the simplest and most direct segmentation method.

Evaluation Metrics. We evaluate the accuracy of OpenLabCluster by computing the percentage of temporal segments in the test set that OpenLabCluster correctly associated with the states given as ground truth, such that 100% accuracy will indicate that OpenLabCluster correctly classified all temporal segments in the test set. Since OpenLabCluster implements the semi-supervised approach to minimize the number of annotations for segments, we compute the accuracy given annotation budgets of overall 5%, 10%, 20% labels to be used over the possible iterations in conjunction with active learning. In particular, we test the accuracy when the Top, CS, and MI active learning methods implemented in OpenLabCluster are used for the selection of temporal segments to annotate.

Benchmark Comparison. We evaluated our method against established animal behavior classification approaches, including both traditional and active learning methods. For traditional approaches, we compared against K-Nearest Neighbour (KNN) [R 272], support vector machine (SVM) [R 9], SimBA's Random Forest Classifier (RFC) [R 266], and VAME with an additional classifier (VAME+C) [R 260].For active learning, we compared our method to A-SOiD [R 270], which employs RFC with selective sampling. Furthermore, we conducted ablation studies by evaluating a version of OpenLabCluster without the decoder and explored alternative architectures by integrating VAME's encoder-decoder (OpenLabCluster-V) with

	Mouse (8 classes; Keypoints)			
Labels (%)	5	10	20	100
Labels (#)	143	286	571	2856
KNN [272]	43.5±3.9	53.1±1.7	51.5±2.9	60.8±0.0
SVM[9]	50.6±6.0	60.3±2.6	64.6±1.6	72.3±0.0
C	55.2±3.6	60.7±1.7	64.5±2.1	71.5±1.0
SimBA[266]	62.2±3.8	66.5±2.2	69.8±1.2	79.8±0.9
A-SOiD[270]	63.7±3.9	60.1±3.4	65.5±1.7	70.9±0.9
VAME+C[260]	67.4±4.6	75.1±1.8	77.8±1.3	85.2±0.7
OpenLabCluster Top	58.6±2.6	69.2±1.8	76.7±1.2	83.8±0.4
OpenLabCluster MI	65.8±2.8	76.6±0.9	79.1±0.7	82.0±0.7
OpenLabCluster CS	66.2±3.1	74.5±1.8	**81.5**±1.0	83.9±0.3
OpenLabCluster-V TOP	68.3±1.8	75.3±1.8	77.5±1.2	85.6±0.8
OpenLabCluster-V CS	74.4±1.4	**77.7±1.8**	81.4±0.9	**85.8±0.7**
OpenLabCluster-V MI	**74.7±1.2**	76.1±1.5	80.1±0.6	85.3±0.3

Table 5.1: Classification accuracy of Home-Cage Mouse behaviors for increasing number of annotated segments (reported as percentage (%)). Top: Classification accuracy of standard methods: KNN, SVM, C, SimBA, A-SOiD, and VAME+C. Middle: Accuracy of OpenLabCluster using active learning strategies: CS, Top, and MI. Bottom: Accuracy of OpenLabCluster with the VAME encoder-decoder (OpenLabCluster-V). Boldface indicates the best accuracy.

various active learning strategies (CS, TOP, and MI).

Outcomes. The results of evaluation in 5 runs are shown in Tables 5.1 and 5.2 and further analysis in Fig. 5.3 and Fig. 5.4. We summarize the main outcomes of the evaluation

	Zebrafish (13 classes; Kinematics)				*C. elegans* (3 classes; Segments)			
Labels (%)	5	10	20	100	5	10	20	100
Labels (#)	265	530	1059	5294	27	55	109	543
SVM[9]	55.8±1.5	63.9±1.2	68.6±0.5	74.0±0.0	76.3±0.8	76.4±0.3	74.5±0.6	76.0±0.0
KNN [272]	57.2±1.0	60.3±1.1	63.0±0.3	69.2±0.0	61.0±11.3	71.5±3.4	73.3±2.6	77.0±0.0
VAME+C[260]	62.1±1.1	67.7±0.4	70.9±0.5	75.2±0.3	73.8±3.8	75.7±1.5	76.6±0.2	76.5±0.0
C	65.7±1.6	70.0±0.7	75.8±0.9	82.8±0.1	71.3±4.7	75.2±3.4	77.5±0.9	**94.8**±0.7
A-SOiD[270]	68.7±0.7	72.3±0.5	74.5±0.4	77.8±0.1	73.4±5.6	74.4±2.5	76.6±1.0	80.1±1.0
SimBA[266]	69.7±1.0	73.0±0.8	75.9±0.7	80.3±0.3	68.9±12.7	74.1±4.1	77.2±2.1	83.5±0.6
OpenLabCluster CS	71.9±1.0	76.6±1.0	79.8±0.6	83.2±0.1	73.5±1.0	64.3±5.1	75.1±3.1	92.8±0.4
OpenLabCluster Top	72.0±1.2	77.1±0.7	80.1±0.7	**83.5±0.4**	76.5±0.0	76.5±3.1	**77.9**±1.1	93.0±0.5
OpenLabCluster MI	**74.7±0.7**	**79.2±0.3**	**81.1±0.4**	83.4±0.2	76.6±0.2	74.7±3.0	76.9±0.2	93.6±0.8
OpenLabCluster-V CS	44.4±1.6	53.5±2.6	64.3±1.3	75.1±0.2	65.6±12.6	76.5±0.3	75.4±2.4	76.6±0.2
OpenLabCluster-V TOP	63.0±1.0	67.8±0.8	71.6±0.9	75.1±0.2	**75.8±2.2**	**76.8±0.2**	76.7±0.2	76.5±0.0
OpenLabCluster-V MI	65.9±0.7	69.4±0.7	71.9±0.4	75.1±0.3	75.7±1.2	76.8±0.4	76.8±0.4	76.5±0.0

Table 5.2: Classification accuracy of Zebrafish and *C. elegans* behaviors for increasing the number of annotated segments (reported as percentage (%)). Top: benchmark methods KNN, SVM, and C, SimBA, A-SOiD, VAME+C, C. Middle: Accuracy of OpenLabCluster using various active learning approaches: CS, Top, and MI. Bottom: OpenLabCluster with VAME encoder-decoder (OpenLabCluster-V). Best accuracy is highlighted in boldface.

and their interpretation below.

Accuracy of Classification. We observe that *the accuracy of classification of OpenLabCluster across datasets is consistently higher than standard supervised classification methods (e.g., C, SVM, SimBA) for almost any budget of annotation.* Specifically, for the Home-Cage Mouse Behavior dataset Table 5.1, OpenLabCluster achieves an accuracy of 66.2% when just 143 (5% of 2856) segments have been annotated. Accuracy improves along with the increase in the number of annotated segments, i.e., accuracy is 76.6% when 10% of segments are annotated, and 81.5% when 20% of segments are annotated. Compared to C, the encoder-only version of

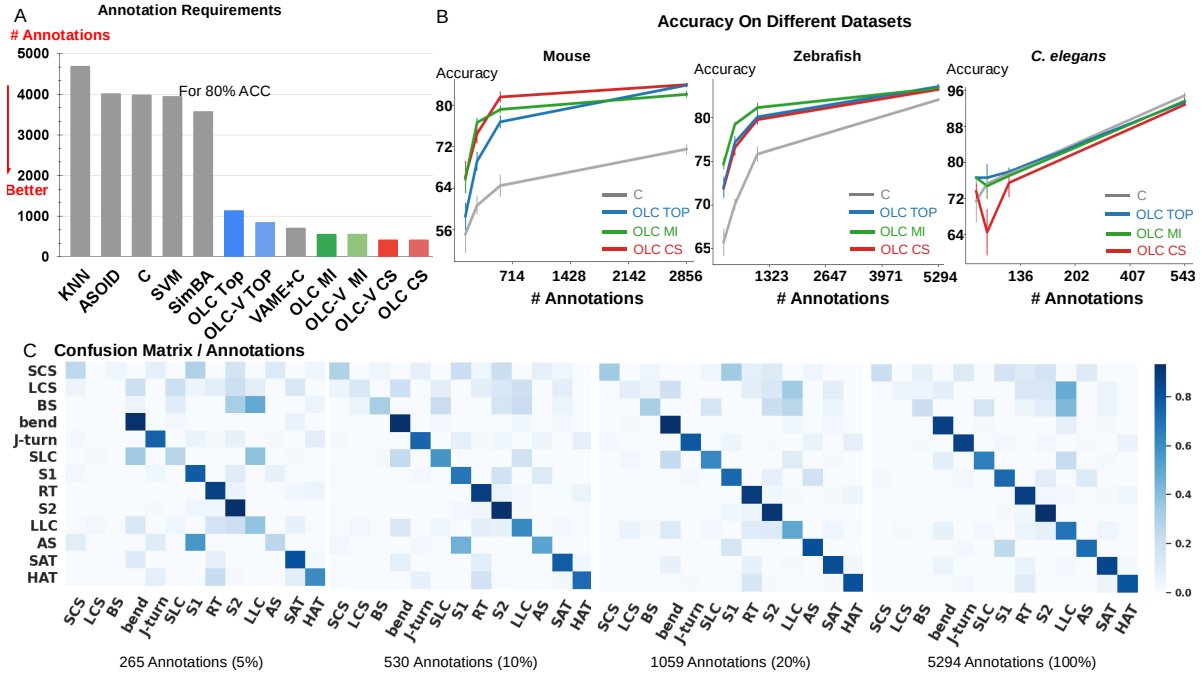

Figure 5.3: Relation Between Accuracy and Annotation. A: The amount of annotations required to achieve 80% accuracy for classification of Home-Cage Mouse behaviors. Computed for benchmark methods (KNN, SVM, and C, SimBA, A-SOiD, VAME+C), and variants of OpenLabCluster with three AL methods (Top, MI, CS). B: Prediction accuracy with increasing annotation budget on three datasets of Mouse, *C. elegans* and Zebrafish. C: Confusion matrix for zebrafish dataset for increasing annotation budget (5%, 10%, 20%, 100%).

OpenLabCluster, the increase in accuracy is of $\approx 12\%$ on average, highlighting the importance of the encoder-decoder structure and active sample selection. While VAME+C which includes the encoder-decoder and the classifier also shows promise, achieving an accuracy of 67.4% with 5% annotated samples. It is still inferior to OpenLabCLuster-V which attains 74.7% under the same annotation budgets. This further illustrates the effectiveness of active learning for accurate behavior classification with limited labels. To be noticed, although A-SOiD also

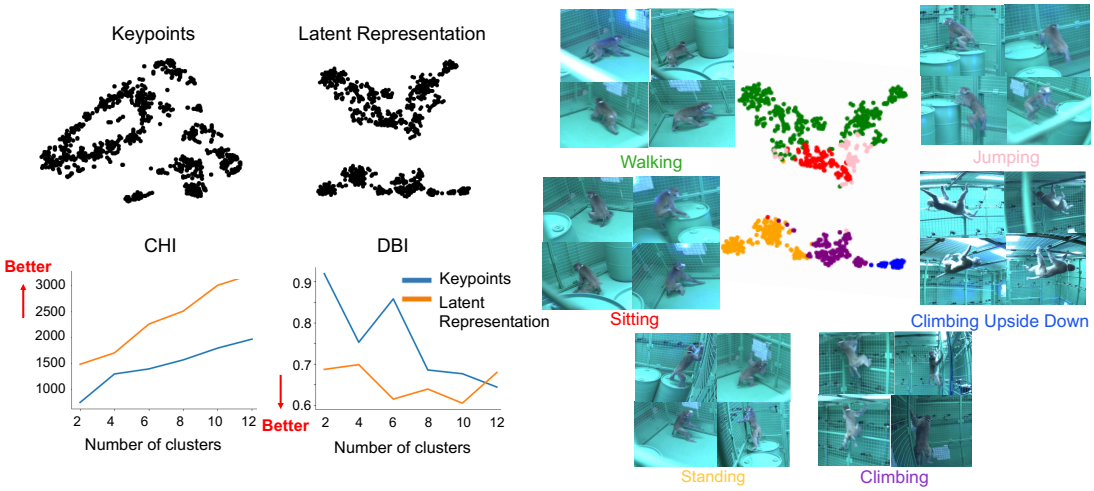

Figure 5.4: 2D tSNE projection of behavioral segments. Left, Top: 2D tSNE projection of keypoints compared with 2D tSNE projection of Latent Representation (Cluster Map). Left, Bottom: CHI and DBI metrics computed for each projection for increasing number of clusters. Right: Behavior Classification Map (2D tSNE projection of Latent Representation with associated behavioral states colors) along with example video frames from segments associated with each class. Images are reproduced from images in the datset of [73].

integrates active learning for sample selection, its classifier design and the AL method—relying solely on uncertainty-based selection—limits its performance relative to OpenLabCluster. Among the active learning (AL) strategies TOP, CS, and MI, both CS and MI outperform TOP on the Home-Cage Mouse dataset. Notably, while active learning is expected to be especially effective in sparse annotation scenarios when all segments are annotated (fully supervised scenario) the accuracy of the OpenLabCluster MI approach exceeds supervised classification approaches (C) by 12.4% (rightmost column in Table 5.1). This reflects the effectiveness of the targeted selection of candidates for annotation and the use of clustering latent representation to enhance the overall organization of the segments.

For Zebrafish and *C. elegans* datasets, OpenLabCluster consistently achieves higher

accuracy, except when 100% of the *C. elegans* dataset are annotated, illustrating its generalizability across various animal behavior datasets. Compared to its encoder-only variant (C), OpenLabCluster shows an accuracy improvement of around 7.8% on the zebrafish dataset, and around 2.2% on *C. elegans* dataset. These gains are less pronounced than on the Home-Cage Mouse dataset. These could be associated with not having manually identified ground truth behavior states for Zebrafish and having only 3 classes for the *C. elegans* dataset which is a simpler semantic task that does not challenge classifiers. We can indeed observe that when all annotations are considered in *C. elegans* dataset, all approaches perform well (above 92%) and a standard classifier achieves the best accuracy. Unlike the Home-Cage Mouse dataset results, OpenLabCluster-V generally underperforms OpenLabCluster on the Zebrafish dataset, while exhibiting comparable performance on *C. elegans*. Moreover, the CS strategy appears unsuitable for OpenLabCluster-V in the Zebrafish setting, leading to poorer outcomes.

The Amount of Required Annotations. Since accuracy varies across datasets and depends on the number of classes and other aspects, we examine the relationship between accuracy and the number of required annotations. In Fig. 5.3A, we compute the necessary number of annotations required to achieve 80% of classification accuracy with benchmark methods, OpenLabCluster and OpenLabCluster-V on the Home-Cage Mouse dataset. We observe that AL methods require only 15-20% of annotated segments while methods without pre-learned sequence representation (KNN, SVM, C, A-SOiD, SimBA) require nine times more annotated segments than the most optimal active learning approach. Meanwhile, VAME+C, OpenLabCluster, and OpenLabCluster-V reach 80% accuracy with far fewer annotations overall. Among AL methods, MI and CS allow OpenLabCluster TOP and OpenLabCluster-V to achieve 80% accuracy with roughly 400 annotated samples. In contrast, TOP selection alone is less effective, requiring approximately 700 annotations.

We further visualize the effectiveness of pertaining (C vs. OpenLabCluster) under varying annotation budgets in Fig. 5.3B. We observe that for most cases, OpenLabCluster methods lead to higher accuracy for a given number of annotations than the counterpart, encoder-only classifier. The curves indicating the accuracy of various OpenLabCluster AL methods (red,

green, blue) have a clear gap between them and C curve (darkgray), especially in the mid-range of the number of annotations. However, the improvement of OpenLabCluster over C is less pronounced on the *C. elegans* dataset, likely due to the dataset's relative simplicity. In Fig. 5.3C, we further examine class-wise confusion matrices for the Zebrafish dataset on 4 annotation budgets (5%, 10%, 20%, 100%). From visual inspection, it appears that the matrix that corresponds to 20% annotations is close to the matrix that corresponds to 100% annotations. This proximity suggests that the annotation of the full dataset might be redundant. Indeed, further inspection of Fig. 5.3C, indicates that samples annotated as LCS and BS classes (y-axis) by the unsupervised learning method are likely to be predicted as the LLC (x-axis) by OpenLabCluster. One possibility for the discrepancy could be annotation errors of the prior clustering method, which are taken as the ground truth. Re-examination of the dynamics of some of the features (e.g. tail angle) further supports this hypothesis and demonstrates that the methods in OpenLabCluster can potentially identify the outlier segments whose annotation settles the organization of the Behavior Classification Map.

Organization of the Latent Representation. Our results indicate that the Latent Representation captured by the OpenLabCluster encoder-decoder and the classifier are able to better organize behavioral segments in comparison to direct embeddings of body keypoints. We quantitatively investigate such an organization with the Monkey dataset, for which ground truth annotations and segmentation are unavailable. Specifically, we obtain the Cluster Map of the segments with OpenLabCluster and then annotate 5% of segments through active learning methods which further train the encoder-decoder and the classifier, generating a revised Cluster Map. We then depict the 2D tSNE projection of the Latent Representation and compare it with the 2D tSNE projection of body keypoints in Fig. 5.4. Indeed, it can be observed that within the Cluster Map, segments are grouped into more distinct and enhanced clusters. To measure the clustering properties of each embedding, we apply clustering metrics of Calinski-Harabasz (CHI) [R 273] and Davies-Bouldin (DBI) [R 274]. CHI measures the ratio of inter- and intra-cluster dispersion, with larger values indicating better clustering. DBI measures the ratio of inter-cluster distance to intra-cluster distance, with lower values

indicating better clustering. CHI and DBI are shown in the bottom left of Fig. 5.4 considering the various number of clusters (2, 4, 6, 8, 10, 12). The comparison shows that the CHI index is higher for the Cluster Map than the embedding of the keypoints regardless of the number of clusters being considered and is monotonically increasing with the number of clusters. The DBI index for the Cluster Map is significantly lower than the index of the embedding of the keypoints for up to 10 clusters with the minimum at 6 and 10 clusters. This is consistent with the expectation that clustering quality will be consistent with the number of behavioral types. Indeed in these behaviors, there are 6 major behavioral classes that can be identified and possibly several transitional ones. We show in Fig. 5.4 (right) the Behavioral Classification Map generated by OpenLabCluster along with frames from representative segments of each class.

5.3 Materials and Methods

Existing approaches for behavior classification from keypoints are supervised and require annotation of extensive datasets before training [R 10, 11]. The requirement limits the generalization of classification from one subject to another, from animal to animal, from a set of keypoints to another, and from one set of behaviors to another due to the need for re-annotation when such variations are introduced.

In contrast, grouping behavioral segments into similarity groups (clustering) typically does not require annotation and could be done by finding an alternative representation of behavioral segments reflecting the differences and the similarities among segments. Both classical and deep-learning approaches address such groupings [R 21, 249]. Notably, clustering is a 'weaker' task than classification since does not provide the semantic association of groups with behavioral classes, however, could be used as a preliminary stage for classification. If leveraged effectively, as a preliminary stage, clustering can direct annotation to minimize the number of segments that need to be annotated and at the same time to boost classification accuracy.

OpenLabCluster, that is primarily based on this concept, first infers a *Cluster Map* and

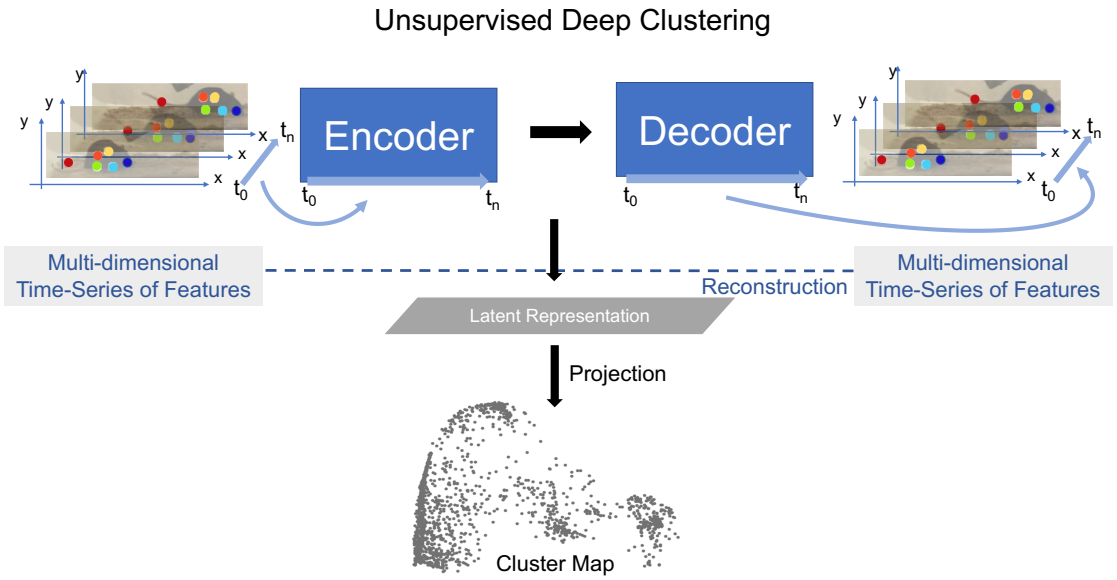

Figure 5.5: Latent Representation is learned by performing the reconstruction task using an encoder-decoder structure. Latent vectors (last state of the encoder) are projected onto low dimensional space with various dimension reduction techniques to visualize the Latent Representation which constitute the Cluster Map. Mouse images are reproduced from frames of videos in the dataset of [9].

then leverages it for automatic selection of sparse segments for annotation (active learning) that will both inform behavior classification and enhance clustering. It iteratively converges to a detailed *Behavior Classification Map* where segments are grouped into similarity classes and each class is homogeneously representing a behavioral state. Below we describe the components.

Clustering The inputs into OpenLabCluster denoted as $\mathcal{X}$, are sets of keypoints coordinates (2D or 3D) or kinematics features for each time segment along with the video footage (image frames that correspond to these keypoints). Effectively, each input segment of keypoints is a matrix with the row dimension indicating the keypoints coordinate, e.g. the first row will indicate the x-coordinate of the first keypoint and the second row will indicate

the y-coordinate of the first keypoint and so on.

The first stage of OpenLabCluster is to employ a Recurrent Neural Network (RNN) encoder-decoder architecture that will learn a Latent Representation for the segments as shown in Fig. 5.5. The encoder is composed of m bi-directional gated recurrent units (bi-GRU) [R 159] sequentially encoding time-series input into a Latent Representation (latent vector in $\mathbb{R}^m$ space). Thus each segment is represented as a point in the Latent Representation $\mathbb{R}^m$ space. The decoder is composed of uni-directional GRUs that receive as input the latent vector and decode (*reconstruct*) the same keypoints from the latent vector. Training optimizes encoder-decoder connectivity weights such that the reconstruction loss, the distance between the segment keypoints reconstructed by the decoder and the input segment, is minimized. This process reshapes the latent vector points in the Latent Representation space to better represent the segments similarities and distinctions. In great details, We adopt the Predict & Cluster encoder-decoder framework which has been shown to achieve self-organizing latent representations [R 105, 106]. The encoder uses bidirectional Gated Recurrent Units (GRU) and receives $\mathbf{X}_i \in \mathcal{X}$ as input. The vector $\mathbf{h}_i^T$ is the latent representation which is the hidden state of the encoder GRU at the last time step T. It encodes the dynamic properties of the whole sequence $\mathbf{x_i}$ and lies in the latent space V, where $V = \{\mathbf{h}_i^T | \mathbf{h}_i^T = encoder(\mathbf{X}_i), \mathbf{X}_i \in \mathcal{X}\}$, i.e., the space spanned by the latent codes of all sequences. The unidirectional GRU-based decoder receives $\mathbf{h_i^T}$ and generates $\hat{\mathbf{X}}_i$ - the reconstruction of the original input sequences. The encoder-decoder network is trained by minimizing the reconstruction loss

$$\mathcal{L}_{re} = |\hat{\mathbf{X}}_i - \mathbf{X}_i|. \tag{5.1}$$

To visualize the relative locations of segments in the Latent Representation, OpenLab-Cluster implements various dimension reductions (from $\mathbb{R}^m \rightarrow \mathbb{R}^2$ or $\mathbb{R}^m \rightarrow \mathbb{R}^3$), such as PCA, tSNE, UMAP, to obtain Cluster Maps, see Fig. 5.5-bottom. Thus each point in the Cluster Map is a reduced-dimensional Latent Representation of an input segment. From inspection of the Cluster Map on multiple examples and benchmarks, it can be observed that the Latent Representation clusters segments that represent similar movements into the

same clusters, to a certain extent, typically more effectively than an application of dimension reduction techniques directly to the keypoints segments [R 105, 106].

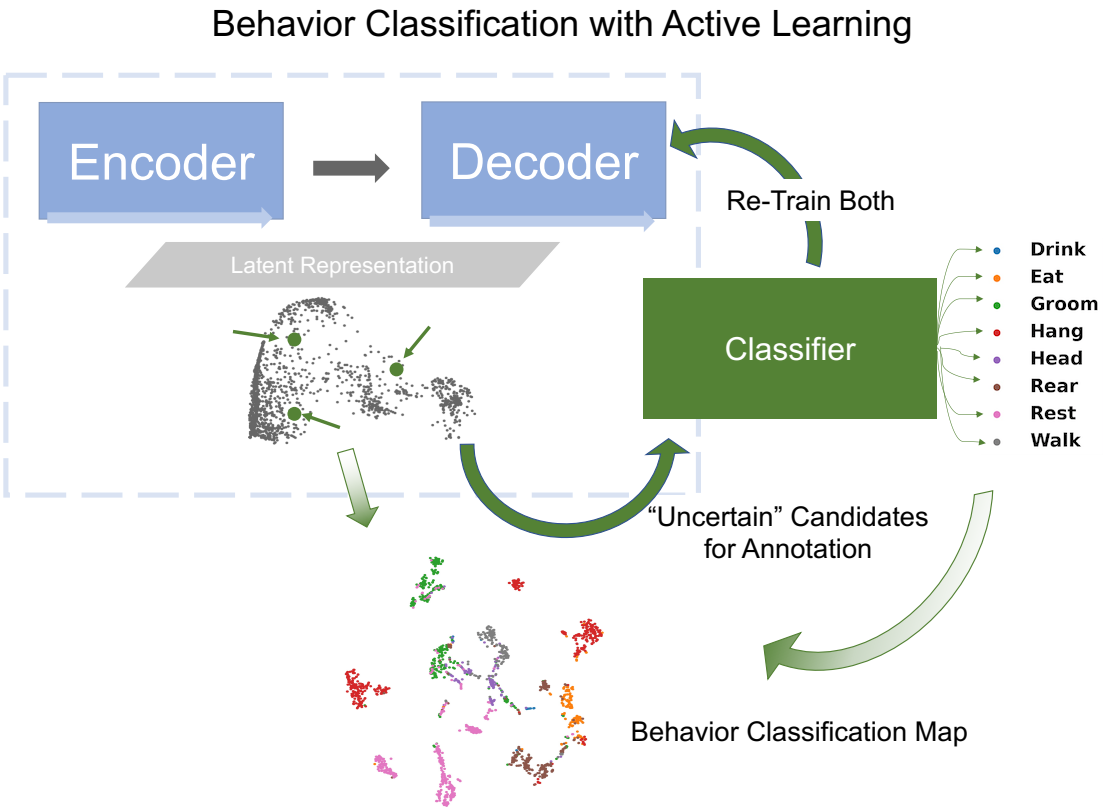

Figure 5.6: Behavior Classification Map is generated by a fully connected classifier network (green rectangle) which receives the latent vector transformed by the encoder-decoder as input and classifies them into behavior classes (example shown: 8 classes in Home-Cage Mouse Behavior dataset). Behavioral Classification Map is generated from the Cluster Map and indicates the predicted classes of all segments.

Classification. To classify behavioral segments that have been clustered, we append a classifier, a fully connected network, to the encoder.

The training of the classifier is based on segments that have been annotated and minimizes the error between the predicted behavioral states and the behavioral states given by the

annotation (cross-entropy loss). It takes Latent Representation as input and generates the probabilities that a sample belongs to each behavioral states. During training, the classifier is learned to maximize the probability of the annotated behavior state. In other words, with the annotated samples given the classifier output, the classification loss is computed as

$$\mathcal{L}_{cla}^{i} = \sum_{l=1}^{C} -y_i^l \log(p^l(\mathbf{X}_i)), \tag{5.2}$$

where $y_i^l = 1$ if $\mathbf{X}_i$ belongs to class l, and $y_i^l = 0$ otherwise. The complete loss for each sample $\mathbf{X}_i$, is then composed from the reconstruction loss $\mathcal{L}_{re}^{i}$ and the classification loss $\mathcal{L}_{cla}^{i}$ for the annotated samples.

$$\mathcal{L} = \sum_{\mathbf{X}_i \in \mathcal{X}_l} \mathcal{L}_{cla}^{i} + \frac{1}{|\mathcal{X}|} \sum_{\mathbf{X}_i \in \mathcal{X}} \mathcal{L}_{re}^{i}, \tag{5.3}$$

where $|\mathcal{X}|$ is the total number of samples in the dataset. This includes all labeled samples annotated in current and earlier iteration. When the annotated segments well represent the states and the clusters, the learned knowledge is transferable to other unlabeled segments.

Active Learning methods such as Cluster Center (*Top*), Core-Set (*CS*) and Marginal Index (*MI*) aim to select such representative segments by analyzing the Latent Representation. *Top* leverages clusters information in the latent space to enhance coverage and effectiveness of selected segments (samples). Specifically, *K-Means* clustering is used to transform the latent representation into a collection of clusters $\mathcal{K}$. The number of clusters k

$$k = \frac{1}{N_{iter}} \times \ percentage \ \times |\mathcal{X}|, \tag{5.4}$$

is chosen based on the total number of selection iterations N_{iter} and the *percentage* of data would be annotated in total. k is fixed across selection iterations such that k is the number of samples to be annotated and each is located in a different cluster. This approach is effective at the initial stage. *MI* is an uncertainty-based selection method, selecting samples that the network is most uncertain about. The uncertainty is measured based on the classifier outputs and measures the top two difference between the classifier outputs. The probability

prediction p of each class l is

$$p^l(\mathbf{X}_i) = p^l(\hat{y}_i = l|W_\theta, W_\delta) = \mathcal{C}^l(\mathbf{X}_i),$$

where p^l denotes the probability of a sample to belong to a class l among C classes ($l \in [1, C]$) predicted by the classifier $\mathcal{C}$. $\mathcal{C}$ indicates the transformation with the classifier. MI is computed as the measure of the confidence, difference between the most probable class and the second most probable class [R 150, 151], i.e.,

$$MI = \max_{l \in [1:C]} \left(p^l\right) - \max_{l \in ([1:C] \setminus l^*)} \left(p^l\right), \tag{5.5}$$

where $l^* = \arg\max_{l \in [1:C]} \left(p^l\right)$ CS aims to discover a set of samples that can cover unlabeled samples with a certain radius [R 134]. The algorithm finds a set of samples such that the radius is minimal. Once segments for annotation are chosen by the active learning method, OpenLabCluster highlights the points in the Cluster Map that represent them and their associated video segments, such that they can be annotated within the graphic interface of OpenLabCluster (choosing the most related behavioral class). When the annotations are set, the full network of encoder-decoder with appended classifier is re-trained to perform classification and predict the labels of all segments. The outcome of this process is the Behavior Classification Map which depicts both the points representing segments in clusters and associated states labels with each point (color) as illustrated in Fig. 5.6. In this process, each time that a new set of samples is selected for annotation, the parameters of the encoder-decoder and the classifier are being tuned to generate more distinctive clusters and more accurate behavioral states classification. The process of annotation and tuning is repeated, typically until the number of annotations reaches the maximum amount of the annotation budget, or when clustering and classification appear to converge to a steady state.

Implementation Details. OpenLabCluster code [Jingyuan 4] was developed in University of Washington UW NeuroAI Lab by Jingyuan Li and Moishe Keselman. OpenLabCluster interface is inspired by DeepLabCut [R 68], which code is used as a backbone for user interface panels, interaction with the back-end, logging, and visualization. OpenLabCluster also uses

Google Active Learning Playground code [R 275] for the implementation of the K-center selection method in the Core-Set active learning option. For specific usage please see the third_party folder within the OpenLabCluster code repository [R 4]. OpenLabCluster is available as a GitHub Repository (OpenLabCluster) `https://github.com/shlizee/OpenLabCluster` and also can be installed with Package Installer for Python (PIP) *pip install openlabcluster* [6]. The repository includes a manual, instructions, and examples.

Benchmark Details. As described earlier, OpenLabCluster summarizes keypoints or kinematic features of a temporal segment into a latent representation and then classifies the behavior using this summarized representation. This approach captures the intrinsic dynamics of short behavior prototypes, in contrast to benchmark methods that compute movement features at each timestep via predefined protocols [R 12, 270] and classify behavior on a per-timestep basis. To ensure fair comparisons, we concatenated frame-wise features within each segment and applied each frame-wise classification method to the concatenated representation. Specifically, for KNN [R 272], SVM [R 9], and A-SOiD [R 270], we concatenated the frame-wise features of each action segment and then used each method's proposed classifier for behavior recognition. For SimBA [R 266], we extracted movement features from each frame and integrated them with pose-based features. The final representation is formed by concatenating these integrated features across all timesteps within the segment. VAME [R 260] most closely resembles OpenLabCluster by learning the unified representation of entire sequences. In VAME+C, we pre-train VAME, add a classifier to its latent feature (which encodes the temporal segment's dynamics), and fine-tune it using a classification loss. For the Home-Cage Mouse, Zebrafish, and C. elegans datasets, sequences are pre-segmented so that each segment contains a single behavior prototype. For the OpenStudio Monkey dataset, which is continuously recorded, we split the videos into fixed temporal windows. More advanced approaches, such as change-point detection algorithms [R 276, 277], could also be applied to segment the video.

Chapter 6

BRAIN DECODING WITH CONTEXT-AWARE NEURAL REPRESENTATIONS AND LARGE LANGUAGE MODELS

The following Chapter is adapted from work [5], which is under review in Journal of Neural Engineering to align with the structure of the thesis.

6.1 Motivation

Verbal communication is a unique feature of human social interaction. Loss of ability to articulate speech as a result of neurological pathologies such as stroke and Amyotrophic Lateral Sclerosis (ALS) can significantly reduce the quality of life for affected individuals. Recent advancements in Brain-Computer Interfaces (BCI) offer promising pathways toward restoring communication ability in these patients by translating neural activity into communicative messages. These messages can be conveyed through various modalities, including typed characters [R 6], handwriting [R 278], text [R 119, 13, 14], and synthesized speech [R 14].

Among existing speech BCI systems, the methods with highest decoding accuracy and throughput are those that translate neural signals associated with orofacial movements during attempted speech into fundamental acoustic units (phonemes), which are then decoded into words and sentences [R 13, 14, 279]. This two-staged approach typically involves (1) neural signal to phonemes: using a temporal deep network to decode a binned multi-channel neural time series into probability of phonemes being spoken at each time step, and (2) phonemes to text: employing a language model (LM) to infer the most probable sequence of words given the phoneme probabilities.

Prior work shows that decoding phonemes as an intermediate representation rather than directly decoding words, provides the system the flexibility to decode phrases from extensive

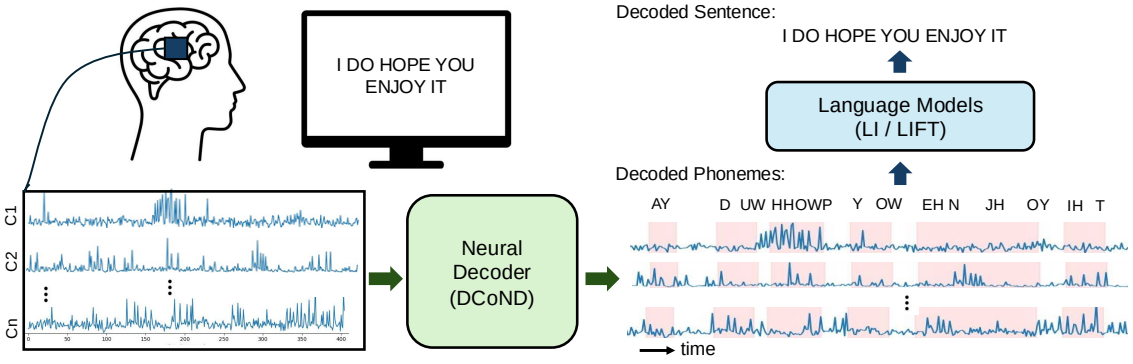

Figure 6.1: Overview of the Brain-to-Text decoding pipeline. The Neural Decoder with Divide-and-Conquer Strategy (DCoND) decodes multi-channel neural activity into phonemes. The phonemes are subsequently converted into words by LLMs using either ICL or fine-tuning techniques.

vocabularies a limited set of training examples [R 14], since from a fixed set of 40 phonemes, one can practically construct any word of any arbitrary length. This scalability is especially advantageous given the limited availability of neural recordings in clinical settings. Indeed, prior works and state of the art methods show that such an intermediate step is advantageous [R 13, 14, 279].

While decoding single phonemes from neural activity may offer more scalability than direct decoding of words, it remains a challenging task. Given the instability of neural recordings, the mapping from neural recordings to phonemes is many-to-one and highly nonlinear. Furthermore, evidence suggests that cortical activation patterns producing a particular phoneme are not static, but can vary depending on the context of surrounding phonemes, a phenomenon known as *coarticulation* [R 15, 16]. In other words, cortical neurons at any given time during speech production are likely encoding a phoneme along with its context, rather than a phoneme in isolation. Given this observation, diphone [R 280] - a sequence of two adjacent phonemes - is a more suitable representation for capturing this context dependency in neural signals and potentially reducing the nonlinearity in phoneme

decoding. Hence we propose to decompose the phoneme classification task into the subtasks of 1) diphone classification, after which 2) diphone probabilities are summed up to obtain the phoneme prediction, i.e. predicting phoneme distribution by marginalizing over the diphone distribution. We show that this divide-and-conquer strategy significantly enhances phoneme decoding performance.

Recently introduced approaches leverage language models, such as n-gram model, to translate phoneme probabilities into words [R 13, 14, 281]. Notably, [R 281] further uses GPT3.5 [R 127] after a 5-gram model to refine the resulting word sequences into coherent sentences by ensembling multiple 5-gram transcription candidates. However, the transcription candidates generated by the n-gram model can significantly deviate from the ground truth phoneme sequence. To address this issue, we propose to augment the ensembling method in [R 281] to include decoded phonemes alongside transcription candidates, which proves to provide extra information for GPT3.5 to infer the correct transcription. In addition, we propose an In-Context Learning (ICL) paradigm for LLMs, enabling them to adapt quickly to newly decoded inputs in a gradient-free manner without the need for the computationally expensive finetuning process. This approach offers a more efficient alternative for improving transcription accuracy in resource-constrained settings.

In summary, our contributions in this work are as follows:

- We propose DCoND (**D**ivide-and-**Co**nquer **N**eural **D**ecoder), a novel framework for decoding phonemes from neural activity during attempted speech. Backed by neuroscientific insights, DCoND infers the temporal phoneme distribution by marginalizing over the diphone distribution, leveraging the context-dependent nature of phonemes in neural representation. The divide-and-conquer strategy can serve as an adaptable component for various neural decoders in brain-to-text decoding.

- We propose incorporating decoded phonemes alongside decoded words in an **L**LM-based ensembling strategy to enhance the speech decoding performance. We also propose the use of (**I**CL) paradigm (DCoND-LI) as an alternative to **F**ine**T**uning LLMs

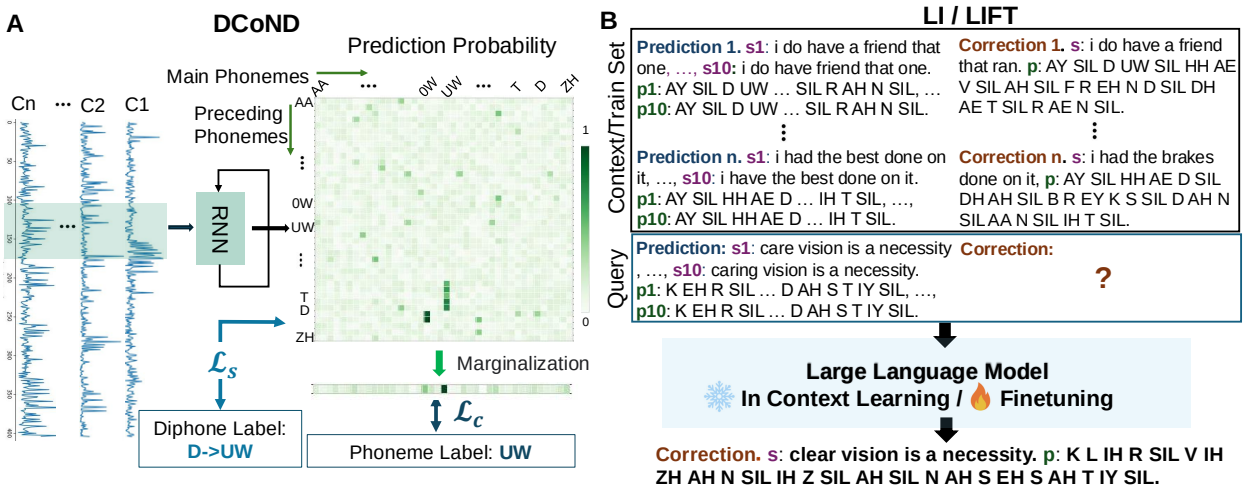

Figure 6.2: **A**: Illustration of the brain-to-phoneme decoding pipeline (DCoND). A Neural-Decoder in DCoND takes multi-channel neural signals as inputs and generates diphone probabilities, which are then marginalized into single phoneme probabilities. **B**: Illustration of the ensembling method for refining transcription predictions (LI/LIFT). Given an ensemble of phoneme and transcription candidates as a query, GPT3.5 produces the most sensible transcription composed from these inputs. To do this, the LLM leverages examples of prediction-correction pairs provided either in-context at inference time (LI) or as training data during the finetuning process (LIFT).

(DCoND-LIFT), offering a more efficient solution for resource-constrained brain-to-text systems.

- We demonstrate the effectiveness of our approaches on the Brain-to-Text 2024 benchmark, where our approach outperforms existing state-of-the-art (SOTA) approaches and achieves PER of 15.34% and WER of 5.77%, a significant improvement compared to 8.93% WER of the leading existing SOTA method.

6.2 Methods

Problem formulation The problem of decoding phonemes from neural activity can be formulated as follows. Let $f : \boldsymbol{X} \rightarrow \mathbf{z}$ be the mapping from neural activity $\boldsymbol{X} \in \mathbb{R}^{T \times N \times D}$ to phoneme sequence $\mathbf{z} \in \mathbb{Z}^{T'}$, where D is the number of neural features, T is the number of neural time bins, and T' is the number of ground truth phonemes in a sentence. We note that $T > T'$ in general, i.e. the articulation of one phoneme may span multiple timesteps. We also emphasize that there is no ground truth temporal alignment between $\boldsymbol{X}$ and $\mathbf{z}$ due to the nature of the silent speech task. Both T and T' vary across trials depending on the length of the sentence in that trial. We aim to learn a model $f_\theta : \boldsymbol{X} \rightarrow \mathbf{z}$ to approximate f with a set of parameters θ, through a deep Neural-Decoder model.

For the Neural-Decoder, we employ an GRU recurrent neural network model together with Connectionist Temporal Classification (CTC) loss as the optimization objective. Previous work investigated various architectures for the Neural-Decoder and demonstrated that GRU shows superior performance [R 13, 282]. In particular, we reconfirm the previous results of effectiveness of GRU vs. other architectures such as Transformers. Notably, the proposed divide-and-conquer strategy is adaptable to various decoder architecture and is designed to enhance the decoding performance of the any decoder that has been chosen. Decoded phonemes $\mathbf{z}$ can be subsequently translated to sentences $\mathbf{y}$ with the help of a language model $h_\phi : \mathbf{z} \rightarrow \mathbf{y}$, where h_ϕ can be a pre-built statistical language model, e.g. 3-gram, 5-gram, or an LLM, e.g. GPT3 [R 127]. The overall pipeline is depicted in Figure 6.1.

A Divide-and-Conquer strategy for phoneme decoding Decoding phonemes from neural activity is a nontrivial task given the highly nonlinear nature of f and the variability of the neural population dynamics. Evidence exists that the neural representations for phonemes vary depending on the surrounding contexts [R 15, 16]. We illustrate this observation in Figure 6.3 where segments of phoneme-aligned neural activity form clusters in the neural space based on the context they are in. It can be seen that there is no single cluster representing

each phoneme, but rather each phoneme is represented by multiple subclusters. We further show that the subclusters are identifiable by the phoneme preceding the phoneme of interest. For instance, the phoneme AH is represented by subclusters DH $\rightarrow$ AH and SIL $\rightarrow$ AH (see further discussion in Section 6.3.4). Learning to model these context-aware sub-units of speech instead of single phonemes directly could facilitate the phoneme decoding task. Concretely,

$$f(\boldsymbol{X}) := p(z|\boldsymbol{X}) = \sum_{s \in S} p(z, s|\boldsymbol{X}) = \sum_{s \in S} g_s^z(\boldsymbol{X}) \tag{6.1}$$

where s is a random variable denoting the context surrounding the phoneme z. For simplicity of notation, here we consider the prediction of $\mathbf{z}$ at a single time step, i.e. $T' = 1$. z takes discrete values from phoneme classes, i.e. $z \in [1, C]$. The problem of learning single phoneme classes (f) now reduces to the problem of learning the phoneme context-dependent subclasses (g_s^Z), which is more manageable and in-line with the context-dependent nature of the data. We refer to our phoneme decoder with this divide-and-conquer strategy as **DCoND**.

Diphone as a context-dependent representation of phonemes The context-dependent subclasses could be defined in multiple ways. In this work, we adopt diphone, a context-dependent representation for phoneme sequences where transitions between phonemes are the subject of interest. For example, the single phoneme representation of "hope", $H, \quad OW, \quad P$, will have a diphone representation:

$$SIL \rightarrow H, \quad H \rightarrow H, \quad H \rightarrow OW, \quad OW \rightarrow OW, \quad OW \rightarrow P, \quad P \rightarrow P, \quad P \rightarrow SIL.$$

where 'SIL' indicates the silence between the words. Diphone expands the length of phoneme sequence to $T'' = 2T'$, where $T\prime$ and $T\prime\prime$ indicate the lengths of single phonemes and the diphone respectively, and increases the number of decoding classes to C^2, where the number of classes $C = 40$ for the English language[1].

[1]the phonemes are defined as per CMU Pronouncing Dictionary: `http://www.speech.cs.cmu.edu/cgi-bin/cmudict/`

Formally, we reformulate the problem of *decoding a phoneme* from neural activity as the *marginalization over the distribution of diphones*,conditioning on the observed neural activity

$$p(z = c_i|X) = \sum_{c_j \in S} p(c_j, c_i|X),$$

where $p(c_j, c_i|X)$ is the probability of neural activity X encoding the diphone $c_j \rightarrow c_i$. A visualization of the marginalization process is shown in Figure 6.2A. Neural activity is processed by a Neural Decoder to predict the probability of 40^2 diphones being spoken at each timestep. The diphone probability is depicted by a 40×40 matrix where columns correspond to the main phonemes and rows correspond to the preceding phonemes. The single phoneme probability is then obtained by summing the joint probabilities column-wise.

Parameter Optimization for Phoneme Decoding As mentioned above, since there is no temporal alignment between T timesteps of neural activity and T' ground truth phonemes in each trial, we therefore use the Connectionist Temporal Classification (CTC) loss as proposed in [R 283] to resolve the unaligned sequences comparison. Specifically, for neural activity X and phoneme sequence Z, CTC aims to maximize the probability of Z given X

$$p(\mathbf{z}|\boldsymbol{X}) = \sum_{\mathbf{a} \in \mathcal{A}_{(\boldsymbol{X},\mathbf{z})}} \prod_{t=1}^{T'} p(a_t|\boldsymbol{X}), \tag{6.2}$$

where $\mathcal{A}_{(\boldsymbol{X},\mathbf{z})}$ is the set of valid alignments between $\boldsymbol{X}$ and $\mathbf{z}$.

Given that we have the diphone representation for each ground truth sentence, we consider the CTC losses over both the diphone and single phoneme representations:

$$\mathcal{L} = \alpha \mathcal{L}_c + (1 - \alpha)\mathcal{L}_s \tag{6.3}$$

where $\mathcal{L}_c = -\log(\sum_{\mathbf{a} \in \mathcal{A}_{(\boldsymbol{X},\mathbf{z})}} \prod_{t=1}^{T'} p_m(a_t|\boldsymbol{X}))$ is the loss for single phoneme decoding, $\mathcal{L}_s = -\log(\sum_{\mathbf{a} \in \mathcal{A}_{(\boldsymbol{X},\mathbf{S})}} \prod_{t=1}^{T''} p(a_t|\boldsymbol{X}))$ defines the loss over subclasses (diphone) decoding.

Coefficient α controls the balance of the single phoneme decoding and diphone decoding. α is designed to be small at the beginning and gradually increase over the course of training.

Word Decoding with Language Models The predicted phoneme probabilities are further transformed into high-quality text through (i) *Generation* of transcription candidates from phonemes, (ii) *Re-scoring* of transcription candidates, and (iii) *Error correction* using an ensemble of selected candidates.

Transcription Generation. During the phase of candidate sentence generation, we convert the predicted phoneme probabilities into words using a 5-gram model. Based on the predicted phoneme probability distribution, the 5-gram model leverages its internal word and sentence distributions to generate the most likely sentence candidates [R 178, 13]. Each candidate is associated with a likelihood score provided by the 5-gram model.

Transcription Re-scoring LLMs trained on large corpora of texts, such as the Open Pre-trained Transformer (OPT) [R 284], could provide more accurate likelihood of the generated transcriptions. Hence, we use OPT to re-score the 5-gram likelihood outputs. The transcription candidates with the highest likelihoods are selected[R 13].

Transcription Error Correction with Ensemble Method. While the 5-gram and OPT models can correct some phoneme errors made by the phoneme decoder to produce more contextually sound sentences (transcriptions), these sentences are not always perfect. Variations of the phoneme decoding model could result in changes of generated and selected sentence candidates. Ensembles of phoneme decoding models, with each model being an expert in different situations, could mitigate the errors made by another model.

In [R 281] GPT3.5 is finetuned to evaluate an ensemble of 10 transcription candidates and generate the most sensible sentence from the 10 candidates. However, providing GPT3.5 only the candidate transcriptions hinders the LLM's ability to understand the underlying phoneme sequences, which are the generating source of the transcriptions and might have been incorrectly converted by the 5-gram model. We therefore propose to include both the transcription candidates and the corresponding phoneme sequences as inputs to GPT3.5, tasking the model with generating both the correct transcription and phoneme sequence. An illustration of such task is shown in Fig.6.2. By finetuning the LLM in this manner, we train it to infer the relationship between predicted phonemes and the predicted transcriptions, as

well as identifying common model-specific mistakes made by the phoneme decoders across their predictions. We show in Section 6.3.3 that this strategy further boosts the WER from 8.06% to 5.77%.

In addition, since finetuning LLM is a resource-intensive process, we also propose to leverage ICL as an alternative learning paradigm for refining predicted transcriptions. Instead of finetuning GPT3.5 over multiple batches of (10× predictions, 1× ground truth) pairs, we directly include N examples of these pairs as context in each prompt, along with a query input to be refined. The LLM then leverages its ICL ability to quickly refine the query transcriptions without updating its weights.

6.3 Experiments

6.3.1 Dataset

We demonstrate the effectiveness of DCoND-LIFT in decoding attempted speech using the Brain-to-Text Benchmark 2024 [R 13, 285]. The dataset was collected from a human subject with ALS who had lost the ability to produce intelligible speech. In the experiments, the subject attempts to silently speak sentences displayed on a screen. These sentences are composed from a vocabulary set of 125,000 words. In each trial, one sentence is shown followed by an auditory 'Go' cue, after which the subject attempts to speak at their own pace. Neural activity (multiunit threshold crossings and spike band power) is recorded from the ventral premotor cortex (6V) while the subject attempted speaking. Due to the nature of the silent speech task, the correspondence between neural activity and the produced speech is unknown. The dataset is split into training, validation, and competition sets with 8800, 600, and 1200 sentences, respectively.

6.3.2 Evaluation Metrics

PER Phoneme Error Rate (PER) is calculated by comparing the decoded phoneme sequence with the ground truth phoneme sequence. After aligning the recognized phoneme sequence

with the reference phoneme sequence, the number of insertions, deletions, and substitutions required to match the sequences are counted. The sum of these operations is divided by the total number of phonemes in the ground truth sequence to compute the PER. This metric reflects how accurately neural signals can be recognized into phonetic units.

WER Similarly to PER, word error rate (WER) is computed by aligning the sequence of recognized words with the ground truth sentence first and then counting the number of insertions, deletions, and substitutions of words needed to reconcile any discrepancies between the two sequences. The total number of these operations is divided by the total number of words in the reference sequence to obtain WER. As neural activity is translated into phonemes before converted into words, WER reflects the performance of both the neural decoder and the language model.

P-WER We adapt Perceptual Word Error Rate (P-WER) [R 14] to measure the quality of phoneme decoding at the word perception level. Specifically, we use eSpeak-NG [R 286] to synthesize speech from the decoded phoneme sequences. Then the synthesized speech is translated into sentences by Whisper [R 287] from which the WER is estimated. Considering the systematic errors introduced by the eSpeak-NG synthesizer and the Whisper ASR system, we define P-WER as follows

$$\text{P-WER} = (1 - \frac{1 - \text{WER}_{Whisper\text{-}P}}{1 - \text{WER}_{Whisper\text{-}GT}}),$$

where $\text{WER}_{Whisper-GT}$ and $\text{WER}_{Whisper-P}$ are the WER measured on Whisper's decoded transcriptions when audio is synthesized with ground truth phoneme sequences (GT) and predicted phoneme sequences (P), respectively.

6.3.3 Comparison with SOTA Methods

Comparison of DCoND-LIFT performance with existing methods shows that DCoND-LIFT achieves state-of-the-art performance on the Brain-to-Text Benchmark 2024, where WER is the primary evaluation metric (see Table 6.1). Specifically, we compared DCoND-LIFT with the leading methods NPTL [R 13] and LISA [R 281]. NPTL uses a 5-layer RNN to decode

Table 6.1: Performance comparison on Brain-to-Text 2024 Benchmark

	PER×100 ↓	WER×100 ↓	P-WER×100 ↓
NPTL [R 13]	16.62	9.46	11.33
LISA [R 281]	–	8.93	–
DCoND-L (Ours)	**15.34**	8.06	**8.02**
DCoND-LI (Ours)	–	7.29	–
DCoND-LIFT (Ours)	–	**5.77**	–

neural activity to phonemes, followed by a combination of 5-gram and OPT language models [R 178, 284] to translate decoded phonemes to texts. LISA also uses RNN as phonemes decoder from neural activity, however it leverages GPT3.5 to further improve transcriptions given by the 5-gram model.

As seen in Table 6.1, DCoND model variants outperform the competing methods. DCoND combined with 5-gram LM and OPT (DCoND-L) yields WER of 8.06%, compared to 9.46% WER of NPTL and 8.93% of LISA. Given that DCoND-L uses the same backbone, RNN decoder and LMs, as NPTL, we posit that the improvements in WER are due to the effectiveness of our proposed divide-and-conquer phoneme decoding strategy. Indeed, DCoND-L achieves a better PER and P-WER (15.34% and 8.02% compared to 16.62% and 11.33% of NPTL).

Further improvement in WER is achieved when we equip DCoND-L with the more powerful language model GPT3.5, to evaluate an ensemble of predicted transcriptions and their associated phoneme representations. When ensemble examplars are shown to GPT3.5 in-context (DCoND-LI), WER improves from 8.06% to 7.29%. This performance is achieved with 25 ICL examplars, the largest number of ICL examplars GPT3.5 can afford due to its prompt length constraint. When we finetune GPT3.5 with all available training examplars (DCoND-

LIFT), WER is further boosted to 5.77%, a significant improvement from 8.93% WER of LISA. These results support our proposal of including both transcriptions and phoneme representations in the demonstrations to GPT3.5 so that it can leverage the relationship between phonemes and words to refine the transcriptions.

6.3.4 Phoneme Decoding Analyses

Neural activity represents phonemes in context-dependent clusters Previous works demonstrate that the decoding accuracy of phonemes from neural activity could be reduced when phonemes are pronounced in the context of other phonemes as opposed to being pronounced individually [R 16]. To get a glimpse of how the brain encodes phonemes, in Fig. 6.3A we visualize phoneme-aligned segments of neural activity in the 2D t-SNE space [R 288]. Since the dataset does not have the exact temporal correspondence between neural activity and phonemes, we leverage Dynamic Time Warping (DTW) to align the ground truth phonemes to neural activity segments according to the timestamps obtained from the decoded phonemes [R 289]. We annotate the neural activity segments based on the resulting phoneme alignment. The visualization reveals that neural activity segments form distinct clusters in the t-SNE space. Notably, these clusters are organized based not only on single phonemes but also on the context in which they are spoken. For instance, during periods where 'T' is the main phoneme being spoken, corresponding neural activity is organized into subclusters of AE→T (orange) and SIL→T (pink), depending on whether phoneme 'AE' or 'SIL' is spoken before 'T'. Similar observations hold for subclusters DH→AH (green) and SIL→AH (red) for the phoneme 'AH'. We note that further subclusters could exist within each subcluster, suggesting a continuum of finer contexts beyond the preceding phoneme.

Diphone decoding leads to enhanced clusters in latent space We visualize in Figure 6.3C and 6.3D the latent space at the last layer of the neural decoder when trained to decode single phonemes (monophones) vs. diphones. In Figure 6.3C, each color represents a single decoded phoneme label. For clarity of the visualization, we selected five single phoneme classes with the most samples. The clusters that correspond to single phonemes appear to

spread out over the whole space, and overlap with each other. In Figure 6.3D, each color represents a decoded diphone. Since there are fewer samples for each diphone, we visualize 16 diphone classes with the highest occurrence. It can be observed that the neural decoder represents diphones in the latent space by clusters that are significantly more condensed and well-separated. Such clear structure facilitates the subsequent classification of single phonemes and demonstrates the effectiveness of our divide-and-conquer phoneme decoding method.

Phoneme prediction error analysis In Figure 6.3B, we show the confusion matrix of the predicted phonemes and the ground truth phonemes. It can be seen from the figure that most phonemes are correctly classified with accuracy greater than 80%. The mistakes that the model typically makes, if any, are on phonemes that are pronounced similarly. For example, the model usually confuses 'SH' with 'S', and 'CH' with 'TH'. Since the articulation of these phonemes is very similar, the neural activity generating them is likely to be similar. Such confusion is expected to some extent, given the ALS condition hindering the ability of the subject to articulate the desired words clearly.

6.3.5 Ablation Study

Trade-off Between Diphone Loss and Monophone Loss We systematically investigate the trade-off between diphone loss $\mathcal{L}_c$ and monophone loss $\mathcal{L}_s$, controlled by the parameter α in Equation 6.2. The impact of varying α on model performance is shown in Table 6.2. We find that a balance between these two losses, with $\alpha = 0.6$, yields the most optimal results. Consequently, we adopt $\alpha = 0.6$ for all DCoND models used in this paper.

Alternatives for context-dependent phoneme representations Besides diphone, triphone is another way to define context-dependent representations for phonemes. Each triphone class consists of three consecutive phonemes, e.g. $H \rightarrow OW \rightarrow P$, providing a finer granularity of context dependency with 40^3 possible classes. Such a large number of classes can be overwhelming for the model to learn. Given that many of them have few to no

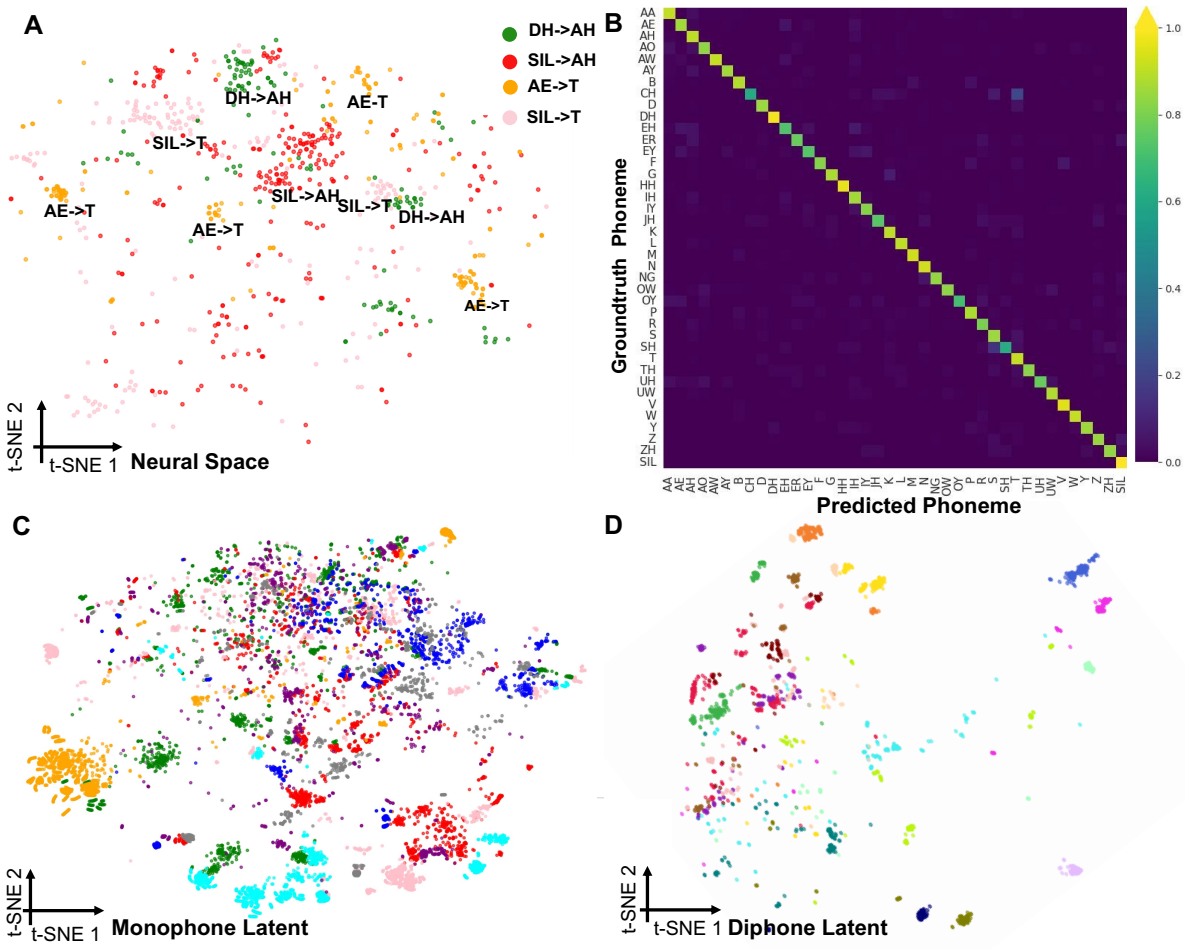

Figure 6.3: **A**: 2D t-SNE visualization of neural signal projections illustrating the context-dependent nature of phonemes in neural reprentations. Different colors indicate different diphone classes. **B**: Confusion matrix of ground truth phonemes vs. DCoND's predicted phonemes. **C**: 2D t-SNE visualization for the latent space of the neural decoder trained with single phoneme decoding objective (Monophone). Different colors indicate different phoneme classes. **D**: 2D t-SNE visualization for the latent space of the neural decoder trained with diphone decoding objective. Different colors indicate different diphone classes.

Table 6.2: Trade-offs between diphone loss and monophone loss.

	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 0.6$ (DCoND-L)	$\alpha = 0.8$	$\alpha = 1.0$ (NPTL)
PER×100 ↓	15.64	15.26	15.34	15.49	16.62
WER×100 ↓	8.47	8.70	8.06	8.64	9.46

Table 6.3: Ablation study on alternative definitions of context-aware phoneme representations.

		Triphone			
	DCoND-L	K=50	K=100	K=200	Grouping
PER×100 ↓	15.34	16.01	**15.02**	15.11	28.55
WER×100 ↓	**8.06**	9.69	9.67	9.81	13.98

presence in the data, to efficiently maintain a manageable size of decoding classes we select the top K combinations of preceding and succeeding phonemes for each main phoneme, e.g. $* \rightarrow OW \rightarrow *$, based on their frequency of occurrence in the data, where $K \in [50, 100, 200]$. Alternatively, the preceding and succeeding phonemes could be grouped based on their articulatory similarity. Specifically, triphones expand upon diphones by incorporating a larger context. Specifically, a triphone considers one phoneme before and one phoneme after the current main phoneme. Consequently, when a neural signal segment is decoded into acoustic units based on the continuity of three phonemes, it reflects a triphone structure. For example, the single phoneme sequence

$$H, \quad OW, \quad P$$

for "hope", can be transferred to triphone

$$\text{“}SIL \rightarrow H \rightarrow OW, \quad H \rightarrow OW \rightarrow P, \quad OW \rightarrow P \rightarrow SIL\text{”}.$$

In this scenario, the time steps required for decoding single phonemes and triphones remain the same. However, triphones introduce a substantial increase in the number of classes, scaling as N^3, which can be prohibitively large (e.g., 64000 when $N = 40$). The divide and conquer idea in this case could be expressed as:

$$f(x) = p(Z = c_i|X) = \sum_{c_j \in C, c_q \in C} p(c_j, c_i, c_q|X)$$

Similar to the diphone probability matrix, these triphone classes are then mapped into a triphone matrix, where each element represents the probability of the current neural signal encoding the phoneme transition from phoneme c_j to phoneme c_i and concluding at phoneme c_q. By summing over the first and last dimensions, we obtain $p(Z = c_i|X)$. Given the potential sparsity of triphone combinations, certain triphone subclasses may not occur frequently in a given language. To mitigate this, we select the top K subclasses for each triphone sample, based on occurrence counts within the current vocabulary. Specifically, for a main phoneme c_i, we rank all possible combinations of $*- > c_i- > *$ and retain the top K as subclasses for the phoneme class c_i.

Additionally, aside from selecting the top K subclasses, an alternative approach involves grouping phones according to articulation similarity [R 119]. This categorization leads to subclasses of the phoneme c_i as $group_j- > c_i- > group_q$. We categorize phonemes into 14 groups, encompassing Bilabial Sounds, Labiodental Sounds, Dental Sounds, Alveolar Sounds, Palatal Sounds, Velar Sounds, Glottal Sounds, Front Vowels, Central Vowels, Back Vowels, and SIL. In this context, the number of subclasses amounts to $14 * 40 * 14$, which is comparable to the number of classes when $K = 200$ (resulting in a total of 200*40 subclasses).

Results in Table 6.3 suggest that triphone with appropriate class size achieves comparable PER as the diphone counterpart (DCoND-L). However, triphone modeling underperforms diphone modeling in terms of WER, possibly due to the reduction in the triphone class size that is potentially skewing the phoneme distribution output of the neural decoder, making it incompatible with the distribution the subsequent 5-gram model was originally trained on single phoneme. Notably, while the grouping method despite yields a class size

similar to that of $K = 200$, it performs signficantly worse in both PER and WER. This implies that neural encoding of phonemes is more intricate, and grouping phonemes based on pronunciation similarity may not be optimal. Overall, we empirically find that diphone, with its context-dependent nature and manageable class size, as the most suitable modeling choice for this task and dataset.

Contribution of LLMs LLMs play an important role in translating phonemes into sentences. As detailed in Section 6.2, our LLM-based phoneme-to-text pipeline consists of three steps: (i) transcription generation (*5-gram*), (ii) transcription re-scoring (*OPT*), (iii) error correction via ensemble with *ICL GPT3.5* or *finetuned GPT3.5*. We show in Figure 6.4 how each step of the LLM pipeline contributes to the overall WER. In particular, we consider the following variants of LLMs, as an added component to DCoND: *5-gram+OPT* as used in NPTL (DCoND-L), *5-gram+OPT+ICL GPT3.5* with context length of 5 (DCoND-LI5), context length of 15 (DCoND-LI15) and context length of 25 (DCoND-LI25), *5-gram+OPT+finetuned GPT3.5* without phoneme inputs (DCoND-LIFT w/o P), and our most performative model – *5-gram+OPT+finetuned GPT3.5* with phoneme inputs (DCoND-LIFT). We show that the use of GPT3.5 to refine the transcriptions from an ensemble of candidates, selected from the highest re-scored likelihood given by the *5-gram+OPT* step, leads to an improvement in WER. The most optimal WER is achieved when GPT3.5 fully leverages the predicted phonemes to refine query transcriptions (DCoND-LIFT). In an ablated version, where limited ICL exemplars are given to GPT3.5 without fine-tuning (DCoND-LIxx), the WER score is enhanced when compared to the baseline (DCoND-L), but does not fully reach the accuracy of DCoND-LIFT. Such disparity between the two modes

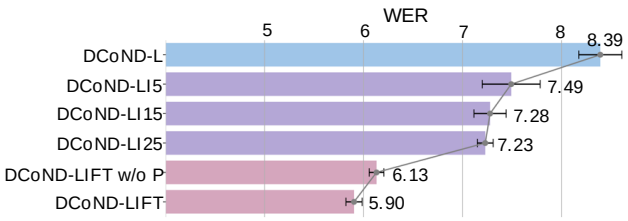

Figure 6.4: Ablation study on the contribution of LLMs.

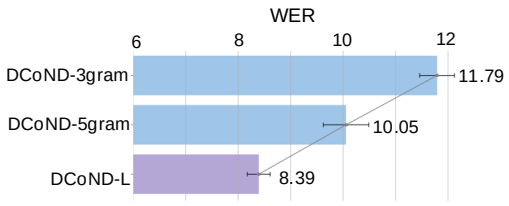

Figure 6.5: Ablation study on the contribution of re-scoring step in the phoneme-to-transcription pipeline.

of LLM effectiveness is expected since the two modes are designed for different operational regimes: DCoND-LI is applicable in scenarios with fast inference and constraints on response time, while DCoND-LIFT is applicable in scenarios in which such constraints do not exist.

Additional Ablation Study on the Contribution of LMs We conduct additional study to assess the role of phoneme-to-transcription generation and re-scoring methods (Figure 6.5). We show that removing the re-scoring step performed by the OPT model in DCoND-L significantly degrades WER (DCoND-3gram and DCoND-5gram), highlighting the importance of the transcription re-scoring step. In addition, the 5-gram model with longer phoneme dependency generates more accurate transcription candidates compared to the 3-gram model.

Open-Source LLMs for DCoND-LI & DCoND-LIFT In addition to the closed-source GPT-3.5, we explore the use of the open-source Llama-3.1-70B for refining transcription predictions. We evaluated Llama-3.1-70B in both in-context learning (DCoND-LI) and fine-tuning (DCoND-LIFT) scenarios and compare it against GPT3.5 (Table 6.4). Llama-3.1-70B performs on par with GPT3.5 in ICL setting, while closely trail behind in finetuning setting, all the while outperforming NPTL and LISA baselines. These results demonstrate our method's robustness and generalizability to other LLMs besides GPT3.5, and warrant the accessibility of our methods to the broad community.

Table 6.4: GPT-3.5 vs Llama-3.1-70B for error correction from ensemble of transcriptions

	Llama-3.1-70B WER	GPT 3.5 WER
DCoND-LI	7.38	7.29
DCoND-LIFT	6.85	5.77

Table 6.5: Comparison of different model architectures on phoneme decoding performance

	PER		
	Transformer	LSTM	GRU
NPTL	39.58 ± 0.15	17.49 ± 0.32	16.63 ± 0.19
DCoND	38.88 ± 0.17	16.08 ± 0.23	15.44 ± 0.46

Investigation on Architecture Choices for Neural Decoders We study the effects of different model architectures on the phoneme decoding performance (PER) (Table 6.5). We observe a significant performance degradation in PER when using Transformer as the neural decoder. On the other hand, RNN counterparts (LSTM and GRU) perform decently well, with GRU being the most performant model for both single phoneme decoding (NPTL) and diphone decoding (DCoND).

6.3.6 Implementation Details

We preprocess the neural signal and construct an RNN neural encoder following the methodology outlined in [R 13]. The raw neural signal $X \in \mathbb{R}^{T \times D}$ is initially partitioned into smaller patches with a window size of W, resulting in a patched neural signal of shape $X \in \mathbb{R}^{T' \times (DW)}$. Overlapping between patches is permitted and determined by the stride size. $W = 14$ for diphone experiments and 32 for the triphone experiemnts. The bidirectional RNN processes

these patched neural signals as inputs, which are subsequently transformed into the neural representation space $H = [h_1, h_2, \cdots, h_{T'}] \in \mathbb{R}^{T' \times d}$. A fully connected layer then maps the hidden representations to diphone or triphone subclasses, denoted as $P(S = s_i|X)$. The outputs of the fully connected layer are used to compute $\mathcal{L}_s$. The computation of single phoneme probabilities is detailed in Equation 6.2. We merge the probability computed from diphone or triphone.

During the RNN training, we utilize a batch size of 32, a learning rate of 0.02, and the Adam optimizer across various experiments the same set of parameters as used in NPTL baseline [R 13]. To facilitate diphone and triphone learning, we initially train the subclasses for 10 epochs and then gradually increase the ratio of the single phoneme loss by 0.1 every 10 epochs until it reaches 0.6. The number of training epochs varies for single phoneme learning, diphone learning, and triphone learning. Specifically, we conduct experiments for up to 100 epochs for single phoneme learning (NPTL baseline), 120 epochs for diphone learning, and 140 epochs for triphone learning since the diphone and triphone required additional subclass training procedures. Increasing the number of training epochs can often lead the model to overfit the training data. Training was done on 2 GeForce RTX 2080 Ti with around 12GB memory. The training take around 6-8 hours.

The 5-gram model takes the predicted phoneme logits as inputs, which can be scaled by a temperature factor denoted as t using the formula $logits := logits/t$. Through experimentation, we have found that setting $t = 1.2$ generally improves the decoding performance. Therefore, we use $t = 1.2$ for our experiments, including the implementation of NPTL, which has resulted in improved baseline results. Specifically, the leaderboard score has improved from 9.76 to 9.46.

Chapter 7
CONCLUSION

This thesis investigates methodologies aiming to facilitate the deciphering of neural mechanisms underlying behavior by examining neural activity and behavioral observations. Considering these two modalities together is beneficial since neural activity reflects brain states and behavioral metrics. Leveraging large-scale recordings of neural and behavioral data, we introduce novel methodologies for analyzing these recordings and examine key factors that facilitate a robust mapping between neural signals and behavioral observations.

For neural recording analysis, we present an adaptive, additive, and multiplicative graph neural network (AMAG) to forecast future neural signals from historical data (Chapter 3). The proposed method incorporates two key innovations: (1) a forecasting task that learns neural dynamics in a biologically plausible, causal manner—each representation relies solely on past and current signals—and (2) an effective modeling of across-neuron interactions using two learnable adjacency matrices.

The success of AMAG stems from integrating the forecasting task with a graph neural network. This forecasting task utilizes the learnable adjacency matrices to discover the optimal organization for controlling information flow between recording channels, thereby enhancing the prediction of future neural activity. Moreover, the resemblance between GNN architecture and network-level neural coding facilitates the recovery of underlying neural dynamics. Validation on both synthetic datasets—with known ground-truth channel connections—and neural recordings from macaque monkeys during hand-reaching movement (via microelectrode arrays and μECoG arrays) demonstrate that our approach not only predicts future activity and identifies critical recording channels for prediction but also yields adjacency matrices that reflect the spatial organization of the recording array. Additionally,

AMAG provides additional insight into the alignment between neural trajectories and their corresponding behavioral stage in latent space.

While AMAG effectively illustrates aligned neural trajectories in latent space during structured hand-reaching behavior, analyzing neural representations in less structured behaviors will require further development of efficient behavior analysis approaches. We have addressed the efficient identification of behavioral states in Chapter 4. In this chapter, we introduce an active-learning-based, semi-supervised action recognition (AL-SAR) approach that classifies behavioral states from skeleton keypoint sequences with minimal human annotation while maintaining high accuracy. The AL-SAR system operates in two stages. In the first stage, unsupervised behavioral representation learning constructs a structured latent space. In the second stage, an active-learning-enhanced classification process selects informative samples for annotation and subsequent classifier training. Specifically, the AL selection process accounts for both samples' position within the latent space and the classifier's uncertainty, as quantified by a marginal index (MI). To enhance the reliability of MI estimation, we introduce a multi-head mechanism that leverages multiple fixed random initializations to augment the latent space and mitigate classifier overconfidence. Throughout iterative training, this mechanism consistently identifies samples that differ from those previously annotated and for which the model exhibits high uncertainty, thereby maximizing the information gained from each annotation. Evaluations of three human benchmark datasets demonstrate that our active-learning strategy outperforms existing methods. Furthermore, AL-SAR's plug-and-play nature enables its integration with advanced unsupervised techniques, such as SC3D, which further enhances the behavior identification accuracy on benchmark datasets.

Building on the behavior analysis framework from Chapter 4, Chapter 5 extends these methodologies to develop OpenLabCluster, a novel toolset for the quantitative analysis of animal behavior from video recordings. OpenLabCluster automatically clusters behavioral segments and assigns them to classes by analyzing body keypoints that capture pose and movement across frames. Its efficacy arises from an unsupervised pre-training phase using an encoder-decoder network, combined with actively selected annotated samples, and is further

enhanced by a graphical user interface that enables scientists without deep learning expertise to annotate samples, train models, and inspect results. We evaluate OpenLabCluster on datasets from freely behaving species, including Home-Cage Mouse, Zebrafish, *C. elegans*, and Monkeys, and demonstrate that, for datasets with ground-truth labels, its classification accuracy exceeds that of direct deep classifiers even under constrained annotation budgets. Further study on the Monkey dataset provided a significant demonstration of OpenLabCluster identifying behavioral representations. These results underscore the generalizability of OpenLabCluster. The methodologies discussed in Chapters 3–5 have thus far addressed neural and behavioral recordings independently.

In Chapter 6, we bridge these modalities by investigating the extraction of behavior-relevant neural representations from brain recordings, thereby establishing an accurate mapping from neural signals to behavioral states. We demonstrate that fine-grained behavior categorization, when precisely aligned with neural representations, enhances neural-to-behavior mapping—particularly in brain-to-text decoding. Specifically, we introduce a Divide-and-Conquer approach for Neural Decoders (DCoND), integrated with an LLM-enhanced ensemble method (comprising LI and LIFT), to transform neural signals into phonetic units and ultimately into text. Unlike conventional neural decoders that decode single phonemes, our method accounts for the co-articulation effect by treating diphones (transitions between consecutive phonemes) as decoding targets, thereby introducing a more detailed behavior-neural correspondence. To mitigate the complexity arising from a larger number of diphone classes, the marginalization step is incorporated into the divide-and-conque algorithm. Subsequent refinement is achieved through n-gram models, rescoring techniques, and error correction using large language models. Experimental results on the Brain-to-Text 2024 Benchmark indicate that our best model, DCoND-LIFT, significantly outperforms existing baselines, underscoring the potential of LLM-based enhancements for more accurate brain-computer interfaces.

Throughout this thesis, the primary contributions lie in the development of biologically realistic and efficient methodologies, including the Neural Encoder, Behavioral Encoder, and

Neural Decoder. These approaches establish a robust connection between neural and behavioral signals, enhancing the performance of brain-computer interfaces (BCIs) for movement restoration. Future work could extend these methods to model discrete behavioral states with high temporal resolution, enabling a more precise understanding of neural representations associated with these states. This, in turn, would facilitate the discovery of collective neural dynamics underlying complex behaviors at a fine-grained temporal scale.

JINGYUAN LI'S PUBLICATIONS

[Jingyuan1] Jingyuan Li, Leo Scholl, Trung Le, Pavithra Rajeswaran, Amy L Orsborn, and Eli Shlizerman. AMAG: Additive, multiplicative and adaptive graph neural network for forecasting neuron activity. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[Jingyuan2] Jingyuan Li, Trung Le, and Eli Shlizerman. AL-SAR: Active learning for skeleton-based action recognition. *IEEE Trans Neural Netw Learn Syst*, 2023.

[Jingyuan3] Jingyuan Li, Moishe Keselman, and Eli Shlizerman. *OpenLabCluster: Active learning based clustering and classification of animal behaviors in videos based on automatically extracted kinematic body keypoints*, 2022.

[Jingyuan4] Jingyuan Li, Moishe Keselman, and Eli Shlizerman. *OpenLabCluster GitHub repository.*

[Jingyuan5] Jingyuan Li, Trung Le, Chaofei Fan, Mingfei Chen, and Eli Shlizerman. Brain-to-text decoding with context-aware neural representations and large language models. *In submission to Journal of Neural Engineering*, 2024.

[Jingyuan6] Jingyuan Li, Moishe Keselman, and Eli Shlizerman. *OpenLabCluster Package.* PyPI, 2022.

RELATED PUBLICATIONS

[R1] Edgar Douglas Adrian. The basis of sensation. *WW Norton & Co*, 1928.

[R2] Antal Berényi, Zoltán Somogyvári, Anett J Nagy, Lisa Roux, John D Long, Shigeyoshi Fujisawa, Eran Stark, Anthony Leonardo, Timothy D Harris, and György Buzsáki. Large-scale, high-density (up to 512 channels) recording of local circuits in behaving animals. *Journal of neurophysiology*, 111(5):1132–1149, 2014.

[R3] Anne E Urai, Brent Doiron, Andrew M Leifer, and Anne K Churchland. Large-scale neural recordings call for new insights to link brain and behavior. *Nature neuroscience*, 25(1):11–19, 2022.

[R4] Anqi Wu, Nicholas A Roy, Stephen Keeley, and Jonathan W Pillow. Gaussian process based nonlinear latent structure discovery in multivariate spike train data. *Advances in neural information processing systems*, 30, 2017.

[R5] Yuan Zhao and Il Memming Park. Variational latent gaussian process for recovering single-trial dynamics from population spike trains. *Neural computation*, 29(5):1293–1316, 2017.

[R6] Chethan Pandarinath, Paul Nuyujukian, Christine H Blabe, Brittany L Sorice, Jad Saab, Francis R Willett, Leigh R Hochberg, Krishna V Shenoy, and Jaimie M Henderson. High performance communication by people with paralysis using an intracortical brain-computer interface. *Elife*, 6:e18554, 2017.

[R7] Mohammad Reza Keshtkaran and Chethan Pandarinath. Enabling hyperparameter optimization in sequential autoencoders for spiking neural data. *Advances in neural information processing systems*, 32, 2019.

[R8] Joel Ye and Chethan Pandarinath. Representation learning for neural population activity with Neural Data Transformers. *Neurons, Behavior, Data analysis, and Theory*, August 2021.

[R9] Hueihan Jhuang, Estibaliz Garrote, Xinlin Yu, Vinita Khilnani, Tomaso Poggio, Andrew D Steele, and Thomas Serre. Automated home-cage behavioural phenotyping of mice. *Nature communications*, 1(1):1–10, 2010.

[R10] Chi Xu, Lakshmi Narasimhan Govindarajan, Yu Zhang, and Li Cheng. Lie-x: Depth image based articulated object pose estimation, tracking, and action recognition on lie groups. *International Journal of Computer Vision*, 123(3):454–478, 2017.

[R11] Oliver Sturman, Lukas von Ziegler, Christa Schläppi, Furkan Akyol, Mattia Privitera, Daria Slominski, Christina Grimm, Laetitia Thieren, Valerio Zerbi, Benjamin Grewe, et al. Deep learning-based behavioral analysis reaches human accuracy and is capable of outperforming commercial solutions. *Neuropsychopharmacology*, 45(11):1942–1952, 2020.

[R12] Cristina Segalin, Jalani Williams, Tomomi Karigo, May Hui, Moriel Zelikowsky, Jennifer J Sun, Pietro Perona, David J Anderson, and Ann Kennedy. The mouse action recognition system (mars) software pipeline for automated analysis of social behaviors in mice. *Elife*, 10:e63720, 2021.

[R13] Francis R Willett, Erin M Kunz, Chaofei Fan, Donald T Avansino, Guy H Wilson, Eun Young Choi, Foram Kamdar, Matthew F Glasser, Leigh R Hochberg, Shaul Druckmann, et al. A high-performance speech neuroprosthesis. *Nature*, pages 1–6, 2023.

[R14] Sean L Metzger, Kaylo T Littlejohn, Alexander B Silva, David A Moses, Margaret P Seaton, Ran Wang, Maximilian E Dougherty, Jessie R Liu, Peter Wu, Michael A Berger, et al. A high-performance neuroprosthesis for speech decoding and avatar control. *Nature*, pages 1–10, 2023.

[R15] Kristofer E Bouchard and Edward F Chang. Neural decoding of spoken vowels from human sensory-motor cortex with high-density electrocorticography. In *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 6782–6785. IEEE, 2014.

[R16] Emily M Mugler, James L Patton, Robert D Flint, Zachary A Wright, Stephan U Schuele, Joshua Rosenow, Jerry J Shih, Dean J Krusienski, and Marc W Slutzky. Direct classification of all american english phonemes using signals from functional speech motor cortex. *Journal of neural engineering*, 11(3):035015, 2014.

[R17] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500, 1952.

[R18] Nicolas Brunel. Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *Journal of computational neuroscience*, 8:183–208, 2000.

[R19] Katie A Ferguson and Jessica A Cardin. Mechanisms underlying gain modulation in the cortex. *Nature Reviews Neuroscience*, 21(2):80–92, 2020.

[R20] Lukas N Groschner, Jonatan G Malis, Birte Zuidinga, and Alexander Borst. A biophysical account of multiplication by a single neuron. *Nature*, 603(7899):119–123, 2022.

[R21] João C Marques, Simone Lackner, Rita Félix, and Michael B Orger. Structure of the zebrafish locomotor repertoire revealed with unsupervised behavioral clustering. *Current Biology*, 28(2):181–195, 2018.

[R22] Eviatar Yemini, Tadas Jucikas, Laura J Grundy, André EX Brown, and William R Schafer. A database of caenorhabditis elegans behavioral phenotypes. *Nature methods*, 10(9):877–879, 2013.

[R23] Alejandro O Blenkmann, Holly N Phillips, Juan P Princich, James B Rowe, Tristan A Bekinschtein, Carlos H Muravchik, and Silvia Kochen. ielectrodes: a comprehensive open-source toolbox for depth and subdural grid electrode localization. *Frontiers in neuroinformatics*, 11:14, 2017.

[R24] A Zátonyi, Gábor Orbán, R Modi, Gergely Márton, Domokos Meszéna, István Ulbert, A Pongrácz, Melanie Ecker, WE Voit, A Joshi-Imre, et al. A softening laminar electrode for recording single unit activity from the rat hippocampus. *Scientific reports*, 9(1):2321, 2019.

[R25] Jacob A Westerberg, Michelle S Schall, Alexander Maier, Geoffrey F Woodman, and Jeffrey D Schall. Laminar microcircuitry of visual cortex producing attention-associated electric fields. *Elife*, 11:e72139, 2022.

[R26] Edwin M Maynard, Craig T Nordhausen, and Richard A Normann. The utah intracortical electrode array: a recording structure for potential brain-computer interfaces. *Electroencephalography and clinical neurophysiology*, 102(3):228–239, 1997.

[R27] Karen C Cheung. Implantable microscale neural interfaces. *Biomedical microdevices*, 9:923–938, 2007.

[R28] Marwan I Hariz, Patric Blomstedt, and Ludvic Zrinzo. Deep brain stimulation between 1947 and 1987: the untold story. *Neurosurgical focus*, 29(2):E1, 2010.

[R29] James J Jun, Nicholas A Steinmetz, Joshua H Siegle, Daniel J Denman, Marius Bauza, Brian Barbarits, Albert K Lee, Costas A Anastassiou, Alexandru Andrei, Çağatay Aydın, et al. Fully integrated silicon probes for high-density recording of neural activity. *Nature*, 551(7679):232–236, 2017.

[R30] Santosh Chandrasekaran, Matthew Fifer, Stephan Bickel, Luke Osborn, Jose Herrero, Breanne Christie, Junqian Xu, Rory KJ Murphy, Sandeep Singh, Matthew F Glasser,

et al. Historical perspectives, challenges, and future directions of implantable brain-computer interfaces for sensorimotor applications. *Bioelectronic medicine*, 7:1–11, 2021.

[R31] Jeffrey W Britton, Lauren C Frey, Jennifer L Hopp, Pearce Korb, Mohamad Z Koubeissi, William E Lievens, Elia M Pestana-Knight, and Erk K St Louis. Electroencephalography (eeg): An introductory text and atlas of normal and abnormal findings in adults, children, and infants. 2016.

[R32] MB Cannell, JR Berlin, and WJ Lederer. Intracellular calcium in cardiac myocytes: calcium transients measured using fluorescence imaging. *Soc Gen Physiol Ser*, 42:201–214, 1987.

[R33] Marsilius Mues, Ingo Bartholomäus, Thomas Thestrup, Oliver Griesbeck, Hartmut Wekerle, Naoto Kawakami, and Gurumoorthy Krishnamoorthy. Real-time in vivo analysis of t cell activation in the central nervous system using a genetically encoded calcium indicator. *Nature medicine*, 19(6):778–783, 2013.

[R34] F Fröhlich. Chapter 11–optical measurements and perturbations. *Network Neuroscience. San Diego, CA: Academic Press*, pages 145–159, 2016.

[R35] Cody A Siciliano and Kay M Tye. Leveraging calcium imaging to illuminate circuit dysfunction in addiction. *Alcohol*, 74:47–63, 2019.

[R36] Nikos K Logothetis, Jon Pauls, Mark Augath, Torsten Trinath, and Axel Oeltermann. Neurophysiological investigation of the basis of the fmri signal. *Nature*, 412:150–157, 2001.

[R37] Richard B Buxton. *Introduction to functional magnetic resonance imaging: principles and techniques*. Cambridge university press, 2009.

[R38] Saurabh Vyas, Matthew D Golub, David Sussillo, and Krishna V Shenoy. Computation through neural population dynamics. *Annual review of neuroscience*, 43:249–275, 2020.

[R39] Byron M Yu, John P Cunningham, Gopal Santhanam, Stephen Ryu, Krishna V Shenoy, and Maneesh Sahani. Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. *Advances in neural information processing systems*, 21, 2008.

[R40] Jakob H Macke, Lars Buesing, John P Cunningham, Byron M Yu, Krishna V Shenoy, and Maneesh Sahani. Empirical models of spiking in neural populations. *Advances in neural information processing systems*, 24, 2011.

[R41] Jakob H Macke, Lars Buesing, and Maneesh Sahani. Estimating state and parameters in state space models of spike trains. *Advanced state space methods for neural and clinical data*, 137, 2015.

[R42] Yuanjun Gao, Evan W Archer, Liam Paninski, and John P Cunningham. Linear dynamical neural population models through nonlinear embeddings. *Advances in neural information processing systems*, 29, 2016.

[R43] Scott W Linderman, Andrew C Miller, Ryan P Adams, David M Blei, Liam Paninski, and Matthew J Johnson. Recurrent switching linear dynamical systems. *arXiv preprint arXiv:1610.08466*, 2016.

[R44] Evan Archer, Il Memming Park, Lars Buesing, John Cunningham, and Liam Paninski. Black box variational inference for state space models. *arXiv preprint arXiv:1511.07367*, 2015.

[R45] Ran Liu, Mehdi Azabou, Max Dabagia, Chi-Heng Lin, Mohammad Gheshlaghi Azar, Keith Hengen, Michal Valko, and Eva Dyer. Drop, swap, and generate: A self-supervised approach for generating neural activity. *Advances in Neural Information Processing Systems*, 34, 2021.

[R46] Ding Zhou and Xue-Xin Wei. Learning identifiable and interpretable latent models of

high-dimensional neural activity using pi-vae. *Advances in Neural Information Processing Systems*, 33:7234–7247, 2020.

[R47] Martin Bjerke, Lukas Schott, Kristopher T Jensen, Claudia Battistin, David A Klindt, and Benjamin A Dunn. Understanding neural coding on latent manifolds by sharing features and dividing ensembles. *arXiv preprint arXiv:2210.03155*, 2022.

[R48] Mohammad Reza Keshtkaran, Andrew R Sedler, Raeed H Chowdhury, Raghav Tandon, Diya Basrai, Sarah L Nguyen, Hansem Sohn, Mehrdad Jazayeri, Lee E Miller, and Chethan Pandarinath. A large-scale neural network training framework for generalized estimation of single-trial population dynamics. *bioRxiv*, 2021.

[R49] Marine Schimel, Ta-Chu Kao, Kristopher T Jensen, and Guillaume Hennequin. ilqr-vae: control-based learning of input-driven dynamics with applications to neural data. *bioRxiv*, 2021.

[R50] Bryan Jimenez. *Neural Network Models For Neurophysiology Data*. PhD thesis, Purdue University Graduate School, 2022.

[R51] Qi She and Anqi Wu. Neural dynamics discovery via gaussian process recurrent neural networks. In *Uncertainty in Artificial Intelligence*, pages 454–464. PMLR, 2020.

[R52] Timothy D Kim, Thomas Z Luo, Jonathan W Pillow, and Carlos Brody. Inferring latent dynamics underlying neural population activity via neural differential equations. In *International Conference on Machine Learning*, pages 5551–5561. PMLR, 2021.

[R53] Jiexia Ye, Juanjuan Zhao, Kejiang Ye, and Chengzhong Xu. How to build a graph-based deep learning architecture in traffic domain: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(5):3904–3924, 2020.

[R54] Trung Le and Eli Shlizerman. Stndt: Modeling neural population activity with a spatiotemporal transformer. *arXiv preprint arXiv:2206.04727*, 2022.

[R55] Ran Liu, Mehdi Azabou, Max Dabagia, Jingyun Xiao, and Eva Dyer. Seeing the forest and the tree: Building representations of both individual and collective dynamics with transformers. *Advances in Neural Information Processing Systems*, 35:2377–2391, 2022.

[R56] Ganga Meghanath, Bryan Jimenez, and Joseph G Makin. Inferring population dynamics in macaque cortex. *arXiv preprint arXiv:2304.06040*, 2023.

[R57] Felix Pei, Joel Ye, David Zoltowski, Anqi Wu, Raeed H Chowdhury, Hansem Sohn, Joseph E O'Doherty, Krishna V Shenoy, Matthew T Kaufman, Mark Churchland, et al. Neural latents benchmark'21: evaluating latent variable models of neural population activity. *arXiv preprint arXiv:2109.04463*, 2021.

[R58] Steffen Schneider, Jin Hwa Lee, and Mackenzie Weygandt Mathis. Learnable latent embeddings for joint behavioural and neural analysis. *Nature*, pages 1–9, 2023.

[R59] Edoardo Balzani, Jean Paul Noel, Pedro Herrero-Vidal, Dora E Angelaki, and Cristina Savin. A probabilistic framework for task-aligned intra-and inter-area neural manifold estimation. *arXiv preprint arXiv:2209.02816*, 2022.

[R60] Yuan Zhao and Il Memming Park. Variational online learning of neural dynamics. *Frontiers in computational neuroscience*, 14:71, 2020.

[R61] Omid G Sani, Hamidreza Abbaspourazad, Yan T Wong, Bijan Pesaran, and Maryam M Shanechi. Modeling behaviorally relevant neural dynamics enabled by preferential subspace identification. *Nature Neuroscience*, 24(1):140–149, 2021.

[R62] Omid G Sani, Bijan Pesaran, and Maryam M Shanechi. Where is all the nonlinearity: flexible nonlinear modeling of behaviorally relevant neural dynamics using recurrent neural networks. *bioRxiv*, pages 2021–09, 2021.

[R63] Matthew Dowling, Yuan Zhao, and Il Memming Park. Real-time variational method for learning neural trajectory and its dynamics. In *The Eleventh International Conference on Learning Representations*, 2023.

[R64] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[R65] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[R66] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *IEEE transactions on pattern analysis and machine intelligence*, 43(1):172–186, 2019.

[R67] Alexander Mathis, Pranav Mamidanna, Kevin M Cury, Taiga Abe, Venkatesh N Murthy, Mackenzie Weygandt Mathis, and Matthias Bethge. Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. *Nature neuroscience*, 21(9):1281, 2018.

[R68] Deeplabcut: a software package for animal pose estimation github repository. `https://github.com/DeepLabCut/DeepLabCut`.

[R69] Tanmay Nath, Alexander Mathis, An Chi Chen, Amir Patel, Matthias Bethge, and Mackenzie Weygandt Mathis. Using deeplabcut for 3d markerless pose estimation across species and behaviors. *Nature protocols*, 14(7):2152–2176, 2019.

[R70] Jessy Lauer, Mu Zhou, Shaokai Ye, William Menegas, Steffen Schneider, Tanmay Nath, Mohammed Mostafizur Rahman, Valentina Di Santo, Daniel Soberanes, Guoping Feng, Venkatesh N. Murthy, George Lauder, Catherine Dulac, M. Mathis, and Alexander Mathis. Multi-animal pose estimation, identification and tracking with deeplabcut. *Nature Methods*, 19:496 – 504, 2022.

[R71] Talmo D Pereira, Diego E Aldarondo, Lindsay Willmore, Mikhail Kislin, Samuel S-H

Wang, Mala Murthy, and Joshua W Shaevitz. Fast animal pose estimation using deep neural networks. *Nature methods*, 16(1):117–125, 2019.

[R72] Anqi Wu, E Kelly Buchanan, Matthew Whiteway, Michael Schartner, Guido Meijer, Jean-Paul Noel, Erica Rodriguez, Claire Everett, Amy Norovich, Evan Schaffer, et al. Deep graph pose: a semi-supervised deep graphical model for improved animal pose tracking. *bioRxiv*, 2020.

[R73] Praneet C Bala, Benjamin R Eisenreich, Seng Bum Michael Yoo, Benjamin Y Hayden, Hyun Soo Park, and Jan Zimmermann. Automated markerless pose estimation in freely moving macaques with openmonkeystudio. *Nature communications*, 11(1):1–12, 2020.

[R74] Talmo D Pereira, Nathaniel Tabris, Junyu Li, Shruthi Ravindranath, Eleni S Papadoyannis, Z Yan Wang, David M Turner, Grace McKenzie-Smith, Sarah D Kocher, Annegret L Falkner, et al. Sleap: Multi-animal pose tracking. *BioRxiv*, 2020.

[R75] Pierre Karashchuk, Katie L Rupp, Evyn S Dickinson, Sarah Walling-Bell, Elischa Sanders, Eiman Azim, Bingni W Brunton, and John C Tuthill. Anipose: a toolkit for robust markerless 3d pose estimation. *Cell reports*, 36(13):109730, 2021.

[R76] Libby Zhang, Tim Dunn, Jesse Marshall, Bence Olveczky, and Scott Linderman. Animal pose estimation from video data with a hierarchical von mises-fisher-gaussian model. In *International conference on artificial intelligence and statistics*, pages 2800–2808. PMLR, 2021.

[R77] Ben Usman, Andrea Tagliasacchi, Kate Saenko, and Avneesh Sud. Metapose: Fast 3d pose from multiple views without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6759–6770, 2022.

[R78] Kristofer E Bouchard, Nima Mesgarani, Keith Johnson, and Edward F Chang. Functional organization of human sensorimotor cortex for speech articulation. *Nature*, 495(7441):327–332, 2013.

[R79] Paul W Schönle, Klaus Gräbe, Peter Wenig, Jörg Höhne, Jörg Schrader, and Bastian Conrad. Electromagnetic articulography: Use of alternating magnetic fields for tracking movements of multiple points inside and outside the vocal tract. *Brain and Language*, 31(1):26–35, 1987.

[R80] Jose A Gonzalez, Lam A Cheah, James M Gilbert, Jie Bai, Stephen R Ell, Phil D Green, and Roger K Moore. A silent speech system based on permanent magnet articulography and direct synthesis. *Computer Speech & Language*, 39:67–87, 2016.

[R81] Ioanna Anastasopoulou, Pascal van Lieshout, Douglas O Cheyne, and Blake W Johnson. Speech kinematics and coordination measured with an meg-compatible speech tracking system. *Frontiers in Neurology*, 13:828237, 2022.

[R82] Natasha Alves, Cecilia Jobst, Fanny Hotze, Paul Ferrari, Marc Lalancette, Tom Chau, Pascal van Lieshout, and Douglas Cheyne. An meg-compatible electromagnetic-tracking system for monitoring orofacial kinematics. *IEEE Transactions on Biomedical Engineering*, 63(8):1709–1717, 2015.

[R83] Elsbeth A van Dam, Lucas PJJ Noldus, and Marcel AJ van Gerven. Deep learning improves automated rodent behavior recognition within a specific experimental setup. *Journal of neuroscience methods*, 332:108536, 2020.

[R84] Yinjun Jia, Shuaishuai Li, Xuan Guo, Bo Lei, Junqiang Hu, Xiao-Hong Xu, and Wei Zhang. Selfee, self-supervised features extraction of animal behaviors. *Elife*, 11:e76218, 2022.

[R85] Eleanor Batty, Matthew Whiteway, Shreya Saxena, Dan Biderman, Taiga Abe, Simon Musall, Winthrop Gillis, Jeffrey Markowitz, Anne Churchland, John P Cunningham, et al. Behavenet: nonlinear embedding and bayesian neural decoding of behavioral videos. *Advances in Neural Information Processing Systems*, 32, 2019.

[R86] Biagio Brattoli, Uta Büchler, Michael Dorkenwald, Philipp Reiser, Linard Filli, Fritjof Helmchen, Anna-Sophia Wahl, and Björn Ommer. Unsupervised behaviour analysis and magnification (ubam) using deep learning. *Nature Machine Intelligence*, 3(6):495–506, 2021.

[R87] Ulrich Stern, Ruo He, and Chung-Hui Yang. Analyzing animal behavior via classifying each video frame using convolutional neural networks. *Scientific reports*, 5(1):1–13, 2015.

[R88] Kartikeya Murari et al. Recurrent 3d convolutional network for rodent behavior recognition. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1174–1178. IEEE, 2019.

[R89] James P Bohnslav, Nivanthika K Wimalasena, Kelsey J Clausing, Yu Y Dai, David A Yarmolinsky, Tomás Cruz, Adam D Kashlan, M Eugenia Chiappe, Lauren L Orefice, Clifford J Woolf, et al. Deepethogram, a machine learning pipeline for supervised behavior classification from raw pixels. *Elife*, 10:e63377, 2021.

[R90] Markus Marks, Qiuhan Jin, Oliver Sturman, Lukas von Ziegler, Sepp Kollmorgen, Wolfger von der Behrens, Valerio Mante, Johannes Bohacek, and Mehmet Fatih Yanik. Deep-learning-based identification, tracking, pose estimation and behaviour classification of interacting primates and mice in complex environments. *Nature machine intelligence*, 4(4):331–340, 2022.

[R91] Lu Xia, Chia-Chih Chen, and Jake K Aggarwal. View invariant human action recognition using histograms of 3d joints. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 20–27. IEEE, 2012.

[R92] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. Learning actionlet ensemble for 3d human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 36(5):914–927, 2013.

[R93] Raviteja Vemulapalli, Felipe Arrate, and Rama Chellappa. Human action recognition by representing 3d skeletons as points in a lie group. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 588–595, 2014.

[R94] Mengyuan Liu, Hong Liu, and Chen Chen. Enhanced skeleton visualization for view invariant human action recognition. *Pattern Recognition*, 68:346–362, 2017.

[R95] Tae Soo Kim and Austin Reiter. Interpretable 3d human action analysis with temporal convolutional networks. In *2017 IEEE conference on computer vision and pattern recognition workshops (CVPRW)*, pages 1623–1631. IEEE, 2017.

[R96] Qiuhong Ke, Mohammed Bennamoun, Senjian An, Ferdous Sohel, and Farid Boussaid. A new representation of skeleton sequences for 3d action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3288–3297, 2017.

[R97] Yong Du, Wei Wang, and Liang Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1110–1118, 2015.

[R98] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+ d: A large scale dataset for 3d human activity analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1010–1019, 2016.

[R99] Wentao Zhu, Cuiling Lan, Junliang Xing, Wenjun Zeng, Yanghao Li, Li Shen, and Xiaohui Xie. Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks. *arXiv preprint arXiv:1603.07772*, 2016.

[R100] Bo Fu, Shilin Fu, Liyan Wang, Yuhan Dong, and Yonggong Ren. Deep residual split directed graph convolutional neural networks for action recognition. *IEEE MultiMedia*, 27(4):9–17, 2020.

[R101] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[R102] Xikun Zhang, Chang Xu, and Dacheng Tao. Context aware graph convolution for skeleton-based action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14333–14342, 2020.

[R103] Ke Cheng, Yifan Zhang, Xiangyu He, Weihan Chen, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with shift graph convolutional network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 183–192, 2020.

[R104] Maosen Li, Siheng Chen, Yangheng Zhao, Ya Zhang, Yanfeng Wang, and Qi Tian. Dynamic multiscale graph neural networks for 3d skeleton based human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 214–223, 2020.

[R105] Kun Su, Xiulong Liu, and Eli Shlizerman. Predict & cluster: Unsupervised skeleton based action recognition. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[R106] Kun Su and Eli Shlizerman. Clustering and recognition of spatiotemporal features through interpretable embedding of sequence to sequence recurrent neural networks. *arXiv preprint arXiv 1905.12176*, 2020.

[R107] Chenyang Si, Xuecheng Nie, Wei Wang, Liang Wang, Tieniu Tan, and Jiashi Feng. Adversarial self-supervised learning for semi-supervised 3d action recognition. *arXiv preprint arXiv:2007.05934*, 2020.

[R108] Lilang Lin, Sijie Song, Wenhan Yang, and Jiaying Liu. Ms2l: Multi-task self-supervised

learning for skeleton based action recognition. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 2490–2498, 2020.

[R109] Fida Mohammad Thoker, Hazel Doughty, and Cees GM Snoek. Skeleton-contrastive 3d action representation learning. *arXiv preprint arXiv:2108.03656*, 2021.

[R110] W Wu, M. Black, Y. Gao, M. Serruya, A. Shaikhouni, J. Donoghue, and Elie Bienenstock. Neural decoding of cursor motion using a kalman filter. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2002.

[R111] Amy M Ni, Chengcheng Huang, Brent Doiron, and Marlene R Cohen. A general decoding strategy explains the relationship between behavior and correlated variability. *eLife*, 11:e67258, jun 2022.

[R112] Wei Wu, Yun Gao, Elie Bienenstock, John P. Donoghue, and Michael J. Black. Bayesian Population Decoding of Motor Cortical Activity Using a Kalman Filter. *Neural Computation*, 18(1):80–118, 2006.

[R113] Ziqian Xie, Odelia Schwartz, and Abhishek Prasad. Decoding of finger trajectory from ECoG using deep learning. *Journal of Neural Engineering*, 15(3):036009, 2018.

[R114] Mehdi Azabou, Vinam Arora, Venkataramana Ganesh, Ximeng Mao, Santosh Nachimuthu, Michael Mendelson, Blake Richards, Matthew Perich, Guillaume Lajoie, and Eva L. Dyer. A unified, scalable framework for neural population decoding. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[R115] David A Moses, Sean L Metzger, Jessie R Liu, Gopala K Anumanchipalli, Joseph G Makin, Pengfei F Sun, Josh Chartier, Maximilian E Dougherty, Patricia M Liu, Gary M Abrams, et al. Neuroprosthesis for decoding speech in a paralyzed person with anarthria. *New England Journal of Medicine*, 385(3):217–227, 2021.

[R116] Spencer Kellis, Kai Miller, Kyle Thomson, Richard Brown, Paul House, and Bradley Greger. Decoding spoken words using local field potentials recorded from the cortical surface. *Journal of neural engineering*, 7(5):056007, 2010.

[R117] Sean L Metzger, Jessie R Liu, David A Moses, Maximilian E Dougherty, Margaret P Seaton, Kaylo T Littlejohn, Josh Chartier, Gopala K Anumanchipalli, Adelyn Tu-Chan, Karunesh Ganguly, et al. Generalizable spelling using a speech neuroprosthesis in an individual with severe limb and vocal paralysis. *Nature Communications*, 13(1):6510, 2022.

[R118] Xiaomei Pei, Dennis L Barbour, Eric C Leuthardt, and Gerwin Schalk. Decoding vowels and consonants in spoken and imagined words using electrocorticographic signals in humans. *Journal of neural engineering*, 8(4):046028, 2011.

[R119] Christian Herff, Dominic Heger, Adriana De Pesters, Dominic Telaar, Peter Brunner, Gerwin Schalk, and Tanja Schultz. Brain-to-text: decoding spoken phrases from phone representations in the brain. *Frontiers in neuroscience*, 9:217, 2015.

[R120] Eleanor Batty, Matthew Whiteway, Shreya Saxena, Dan Biderman, Taiga Abe, Simon Musall, Winthrop Gillis, Jeffrey Markowitz, Anne Churchland, John P Cunningham, Sandeep R Datta, Scott Linderman, and Liam Paninski. Behavenet: nonlinear embedding and bayesian neural decoding of behavioral videos. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[R121] Sean M. Perkins, John P. Cunningham, Qi Wang, and Mark M. Churchland. Simple decoding of behavior from a complicated neural manifold. *bioRxiv*, 2023.

[R122] Nikolaus Kriegeskorte and Pamela K Douglas. Interpreting encoding and decoding models. *Current Opinion in Neurobiology*, 55:167–179, 2019.

[R123] Jiahao Wang, Yunhong Wang, Sheng Liu, and Annan Li. Few-shot fine-grained action recognition via bidirectional attention and contrastive meta-learning. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 582–591, 2021.

[R124] Jay Patravali, Gaurav Mittal, Ye Yu, Fuxin Li, and Mei Chen. Unsupervised few-shot action recognition via action-appearance aligned meta-adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8484–8494, 2021.

[R125] Lei Wang, Jun Liu, and Piotr Koniusz. 3d skeleton-based few-shot action recognition with jeanie is not so na\" ive. *arXiv preprint arXiv:2112.12668*, 2021.

[R126] Anqi Zhu, Qiuhong Ke, Mingming Gong, and James Bailey. Adaptive local-component-aware graph convolutional network for one-shot skeleton-based action recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6038–6047, 2023.

[R127] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[R128] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[R129] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal

Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[R130] Marius Mosbach, Tiago Pimentel, Shauli Ravfogel, Dietrich Klakow, and Yanai Elazar. Few-shot fine-tuning vs. in-context learning: A fair comparison and evaluation. *arXiv preprint arXiv:2305.16938*, 2023.

[R131] David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996.

[R132] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[R133] Burr Settles. Active learning. *Synthesis lectures on artificial intelligence and machine learning*, 6(1):1–114, 2012.

[R134] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

[R135] William H Beluch, Tim Genewein, Andreas Nürnberger, and Jan M Köhler. The power of ensembles for active learning in image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9368–9377, 2018.

[R136] Mingfei Gao, Zizhao Zhang, Guo Yu, Sercan Ö Arık, Larry S Davis, and Tomas Pfister. Consistency-based semi-supervised active learning: Towards minimizing labeling cost. In *European Conference on Computer Vision*, pages 510–526. Springer, 2020.

[R137] Dan Roth and Kevin Small. Margin-based active learning for structured output spaces. In *European Conference on Machine Learning*, pages 413–424. Springer, 2006.

[R138] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning

with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR, 2017.

[R139] Samarth Sinha, Sayna Ebrahimi, and Trevor Darrell. Variational adversarial active learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5972–5981, 2019.

[R140] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079, 2008.

[R141] Yue Deng, KaWai Chen, Yilin Shen, and Hongxia Jin. Adversarial active learning for sequences labeling and generation. In *IJCAI*, pages 4012–4018, 2018.

[R142] Artem Shelmanov, Dmitri Puzyrev, Lyubov Kupriyanova, Denis Belyakov, Daniil Larionov, Nikita Khromov, Olga Kozlova, Ekaterina Artemova, Dmitry V Dylov, and Alexander Panchenko. Active learning for sequence tagging with deep pre-trained models and bayesian uncertainty estimates. *arXiv preprint arXiv:2101.08133*, 2021.

[R143] Steve Hanneke et al. Theory of disagreement-based active learning. *Foundations and Trends® in Machine Learning*, 7(2-3):131–309, 2014.

[R144] Kevin Lang and Eric Baum. Query learning can work poorly when a human oracle is used. *IEEE Intl. JointConference on Neural Networks*, 1992.

[R145] Klaus Brinker. Incorporating diversity in active learning with support vector machines. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 59–66, 2003.

[R146] Zuobing Xu, Ram Akella, and Yi Zhang. Incorporating diversity and density in active learning for relevance feedback. In *European Conference on Information Retrieval*, pages 246–257. Springer, 2007.

[R147] Yang Wang, Jinjia Peng, Huibing Wang, and Meng Wang. Progressive learning with multi-scale attention network for cross-domain vehicle re-identification. *Science China Information Sciences*, 65(6):1–15, 2022.

[R148] Rebecca Hwa. Sample selection for statistical parsing. *Computational linguistics*, 30(3):253–276, 2004.

[R149] Aron Culotta and Andrew McCallum. Reducing labeling effort for structured prediction tasks. In *AAAI*, volume 5, pages 746–751, 2005.

[R150] Maria-Florina Balcan, Andrei Broder, and Tong Zhang. Margin based active learning. In *International Conference on Computational Learning Theory*, pages 35–50. Springer, 2007.

[R151] Ajay J Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. Multi-class active learning for image classification. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2372–2379. IEEE, 2009.

[R152] Evgenii Tsymbalov, Maxim Panov, and Alexander Shapeev. Dropout-based active learning for regression. In *International conference on analysis of images, social networks and texts*, pages 247–258. Springer, 2018.

[R153] Ehsan Elhamifar, Guillermo Sapiro, Allen Yang, and S Shankar Sasrty. A convex optimization framework for active learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 209–216, 2013.

[R154] Lin Yang, Yizhe Zhang, Jianxu Chen, Siyuan Zhang, and Danny Z Chen. Suggestive annotation: A deep active learning framework for biomedical image segmentation. In *International conference on medical image computing and computer-assisted intervention*, pages 399–407. Springer, 2017.

[R155] Luiz FS Coletta, Moacir Ponti, Eduardo R Hruschka, Ayan Acharya, and Joydeep Ghosh. Combining clustering and active learning for the detection and learning of new image classes. *Neurocomputing*, 358:150–165, 2019.

[R156] Weiwei Jiang and Lin Zhang. Geospatial data to images: A deep-learning framework for traffic forecasting. *Tsinghua Science and Technology*, 24(1):52–64, 2018.

[R157] Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*, 2017.

[R158] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug):115–143, 2002.

[R159] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[R160] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[R161] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? *arXiv preprint arXiv:2205.13504*, 2022.

[R162] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[R163] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[R164] Weiwei Jiang, Jiayun Luo, Miao He, and Weixi Gu. Graph neural network for traffic forecasting: The research progress. *ISPRS International Journal of Geo-Information*, 12(3):100, 2023.

[R165] Maosen Li, Siheng Chen, Yangheng Zhao, Ya Zhang, Yanfeng Wang, and Qi Tian. Multiscale spatio-temporal graph neural networks for 3d skeleton-based motion prediction. *IEEE Transactions on Image Processing*, 30:7760–7775, 2021.

[R166] Simon Wein, Alina Schüller, Ana Maria Tomé, Wilhelm M Malloni, Mark W Greenlee, and Elmar W Lang. Forecasting brain activity based on models of spatiotemporal brain dynamics: A comparison of graph neural network architectures. *Network Neuroscience*, 6(3):665–701, 2022.

[R167] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. *arXiv preprint arXiv:1906.00121*, 2019.

[R168] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

[R169] Siyi Tang, Jared A Dunnmon, Liangqiong Qu, Khaled K Saab, Christopher Lee-Messer, and Daniel L Rubin. Spatiotemporal modeling of multivariate signals with graph neural networks and structured state space models. *arXiv preprint arXiv:2211.11176*, 2022.

[R170] Tengfei Song, Wenming Zheng, Peng Song, and Zhen Cui. Eeg emotion recognition using dynamical graph convolutional neural networks. *IEEE Transactions on Affective Computing*, 11(3):532–541, 2018.

[R171] Ming Jin, Enwei Zhu, Changde Du, Huiguang He, and Jinpeng Li. Pgcn: Pyramidal graph convolutional network for eeg emotion recognition. *arXiv preprint arXiv:2302.02520*, 2023.

[R172] Yuqian Chen, Fan Zhang, Leo R Zekelman, Tengfei Xue, Chaoyi Zhang, Yang Song, Nikos Makris, Yogesh Rathi, Weidong Cai, and Lauren J O'Donnell. Tractgraphcnn: anatomically informed graph cnn for classification using diffusion mri tractography. *arXiv preprint arXiv:2301.01911*, 2023.

[R173] Yicong Huang and Zhuliang Yu. Representation learning for dynamic functional connectivities via variational dynamic graph latent variable models. *Entropy*, 24(2):152, 2022.

[R174] Yi Ding, Neethu Robinson, Chengxuan Tong, Qiuhao Zeng, and Cuntai Guan. Lggnet: Learning from local-global-graph representations for brain–computer interface. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[R175] Xiaoxiao Li, Yuan Zhou, Nicha Dvornek, Muhan Zhang, Siyuan Gao, Juntang Zhuang, Dustin Scheinost, Lawrence H Staib, Pamela Ventola, and James S Duncan. Braingnn: Interpretable brain graph neural network for fmri analysis. *Medical Image Analysis*, 74:102233, 2021.

[R176] Felix Schwock, Julien Bloch, Les Atlas, Shima Abadi, and Azadeh Yazdan-Shahmorad. Estimating and analyzing neural information flow using signal processing on graphs. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[R177] Kun Chen, Tianli Xie, Li Ma, Andrew E Hudson, Qingsong Ai, and Quan Liu. A two-stream graph convolutional network based on brain connectivity for anesthetized states analysis. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 30:2077–2087, 2022.

[R178] Yajie Miao, Mohammad Gowayyed, and Florian Metze. Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding. In *2015 IEEE workshop on automatic speech recognition and understanding (ASRU)*, pages 167–174. IEEE, 2015.

[R179] Rajesh Kumar Aggarwal and Mayank Dave. Acoustic modeling problem for automatic speech recognition system: conventional methods (part i). *International Journal of Speech Technology*, 14:297–308, 2011.

[R180] Xuedong Huang, James Baker, and Raj Reddy. A historical perspective of speech recognition. *Communications of the ACM*, 57(1):94–103, 2014.

[R181] Sakhia Darjaa, Miloš Cerňak, Štefan Beňuš, Milan Rusko, Róbert Sabo, and Marián Trnka. Rule-based triphone mapping for acoustic modeling in automatic speech recognition. In *Text, Speech and Dialogue: 14th International Conference, TSD 2011, Pilsen, Czech Republic, September 1-5, 2011. Proceedings 14*, pages 268–275. Springer, 2011.

[R182] Fréjus AA LAleye, Laurent Besacier, Eugène C Ezin, and Cina Motamed. First automatic fongbe continuous speech recognition system: Development of acoustic models and language models. In *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 477–482. IEEE, 2016.

[R183] Rohit Prabhavalkar, Takaaki Hori, Tara N Sainath, Ralf Schlüter, and Shinji Watanabe. End-to-end speech recognition: A survey. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.

[R184] Alex Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012.

[R185] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020.

[R186] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: Self-supervised speech representa-

tion learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3451–3460, 2021.

[R187] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*, 2019.

[R188] Linhao Dong, Shuang Xu, and Bo Xu. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5884–5888. IEEE, 2018.

[R189] Qian Zhang, Han Lu, Hasim Sak, Anshuman Tripathi, Erik McDermott, Stephen Koo, and Shankar Kumar. Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7829–7833. IEEE, 2020.

[R190] Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, 2015.

[R191] Chethan Pandarinath, Daniel J O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D Stavisky, Jonathan C Kao, Eric M Trautmann, Matthew T Kaufman, Stephen I Ryu, Leigh R Hochberg, et al. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature methods*, 15(10):805–815, 2018.

[R192] David Sussillo, Rafal Jozefowicz, LF Abbott, and Chethan Pandarinath. Lfads-latent factor analysis via dynamical systems. *arXiv preprint arXiv:1608.06315*, 2016.

[R193] Michael Nolan, Bijan Pesaran, Eli Shlizerman, and Amy L Orsborn. Multi-block rnn autoencoders enable broadband ecog signal reconstruction. *bioRxiv*, pages 2022–09, 2022.

[R194] Navid Ayoobi and Elnaz Banan Sadeghian. A self-paced bci system with low latency for motor imagery onset detection based on time series prediction paradigm. *arXiv preprint arXiv:2204.05450*, 2022.

[R195] Rainer Dahlhaus. Graphical interaction models for multivariate time series. *Metrika*, 51:157–172, 2000.

[R196] Hexuan Liu, Jimin Kim, and Eli Shlizerman. Functional connectomics from neural dynamics: probabilistic graphical models for neuronal network of caenorhabditis elegans. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 373(1758):20170377, 2018.

[R197] Giuseppe Vinci, Valérie Ventura, Matthew A Smith, and Robert E Kass. Adjusted regularization in latent graphical models: Application to multiple-neuron spike count data. *The annals of applied statistics*, 12(2):1068, 2018.

[R198] Natalie Klein, Josue Orellana, Scott L Brincat, Earl K Miller, and Robert E Kass. Torus graphs for multivariate phase coupling analysis. 2020.

[R199] Luis Carrillo-Reid, Shuting Han, Darik O'Neil, Ekaterina Taralova, Tony Jebara, and Rafael Yuste. Identification of pattern completion neurons in neuronal ensembles using probabilistic graphical models. *Journal of Neuroscience*, 41(41):8577–8588, 2021.

[R200] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[R201] Siqi Miao, Mia Liu, and Pan Li. Interpretable and generalizable graph learning via stochastic attention mechanism. *International Conference on Machine Learning*, 2022.

[R202] Le Song, Mladen Kolar, and Eric Xing. Time-varying dynamic bayesian networks. *Advances in neural information processing systems*, 22, 2009.

[R203] Rahul Biswas and Eli Shlizerman. Statistical perspective on functional and causal neural connectomics: A comparative study. *Frontiers in Systems Neuroscience*, 16:817962, 2022.

[R204] Rahul Biswas and Eli Shlizerman. Statistical perspective on functional and causal neural connectomics: The time-aware pc algorithm. *PLOS Computational Biology*, 18(11):e1010653, 2022.

[R205] Michael Trumpis, Chia-Han Chiang, Amy L Orsborn, Brinnae Bent, Jinghua Li, John A Rogers, Bijan Pesaran, Gregory Cogan, and Jonathan Viventi. Sufficient sampling for kriging prediction of cortical potential in rat, monkey, and human μecog. *Journal of neural engineering*, 18(3):036011, 2021.

[R206] Cecilia Gallego-Carracedo, Matthew G Perich, Raeed H Chowdhury, Lee E Miller, and Juan Álvaro Gallego. Local field potentials reflect cortical population dynamics in a region-specific and frequency-dependent manner. *Elife*, 11:e73155, 2022.

[R207] Cecilia Gallego-Carracedo, Matthew G Perich, Raeed H Chowdhury, Lee E Miller, and Juan Álvaro Gallego. Dataset from "local field potentials reflect cortical population dynamics in a region-specific and frequency-dependent manner". 2022.

[R208] Robert D Flint, Christian Ethier, Emily R Oby, Lee E Miller, and Marc W Slutzky. Local field potentials allow accurate decoding of muscle activity. *Journal of neurophysiology*, 108(1):18–24, 2012.

[R209] David Sussillo, Mark M Churchland, Matthew T Kaufman, and Krishna V Shenoy. A neural network that finds a naturalistic solution for the production of muscle activity. *Nature neuroscience*, 18(7):1025–1033, 2015.

[R210] Frieder Stolzenburg, Sandra Litz, Olivia Michael, and Oliver Obst. The power of linear recurrent neural networks. *arXiv preprint arXiv:1802.03308*, 2018.

[R211] Chao Li, Qiaoyong Zhong, Di Xie, and Shiliang Pu. Co-occurrence feature learning from skeleton data for action recognition and detection with hierarchical aggregation. *arXiv preprint arXiv:1804.06055*, 2018.

[R212] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with multi-stream adaptive graph convolutional networks. *IEEE Transactions on Image Processing*, 29:9532–9545, 2020.

[R213] Nenggan Zheng, Jun Wen, Risheng Liu, Liangqu Long, Jianhua Dai, and Zhefeng Gong. Unsupervised representation learning with long-term dynamics for skeleton based action recognition. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[R214] Hossein Rahmani, Arif Mahmood, Du Q Huynh, and Ajmal Mian. Hopc: Histogram of oriented principal components of 3d pointclouds for action recognition. In *European conference on computer vision*, pages 742–757. Springer, 2014.

[R215] Jiang Wang, Xiaohan Nie, Yin Xia, Ying Wu, and Song-Chun Zhu. Cross-view action modeling, learning and recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2649–2656, 2014.

[R216] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *CVPR 2011*, pages 1297–1304. Ieee, 2011.

[R217] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[R218] Sercan Ö Arık, Heewoo Jun, and Gregory Diamos. Fast spectrogram inversion using multi-head convolutional neural networks. *IEEE Signal Processing Letters*, 26(1):94–98, 2018.

[R219] David Maltz and Kate Ehrlich. Pointing the way: Active collaborative filtering. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 202–209, 1995.

[R220] Neil Houlsby, José Miguel Hernández-Lobato, and Zoubin Ghahramani. Cold-start active learning with robust ordinal matrix factorization. In *International Conference on Machine Learning*, pages 766–774. PMLR, 2014.

[R221] Pengfei Zhang, Cuiling Lan, Junliang Xing, Wenjun Zeng, Jianru Xue, and Nanning Zheng. View adaptive recurrent neural networks for high performance human action recognition from skeleton data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2117–2126, 2017.

[R222] Haocong Rao, Shihao Xu, Xiping Hu, Jun Cheng, and Bin Hu. Augmented skeleton based contrastive action learning with momentum lstm for unsupervised action recognition. *Information Sciences*, 569:90–109, 2021.

[R223] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[R224] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

[R225] Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[R226] Valentin Sturm, Dmitry Efrosinin, Natalia Efrosinina, Leonie Roland, Michael Iwersen, Marc Drillich, and Wolfgang Auer. A chaos theoretic approach to animal activity recognition. *Journal of Mathematical Sciences*, 237(5):730–743, 2019.

[R227] Robert Evan Johnson, Scott Linderman, Thomas Panier, Caroline Lei Wee, Erin Song, Kristian Joseph Herrera, Andrew Miller, and Florian Engert. Probabilistic models of larval zebrafish behavior reveal structure on many scales. *Current Biology*, 30(1):70–82, 2020.

[R228] Gordon J Berman, William Bialek, and Joshua W Shaevitz. Predictability and hierarchy in drosophila behavior. *Proceedings of the National Academy of Sciences*, 113(42):11943–11948, 2016.

[R229] Mehrdad Jazayeri and Arash Afraz. Navigating the neural space in search of the neural code. *Neuron*, 93(5):1003–1014, 2017.

[R230] Sandeep Robert Datta, David J Anderson, Kristin Branson, Pietro Perona, and Andrew Leifer. Computational neuroethology: a call to action. *Neuron*, 104(1):11–24, 2019.

[R231] Rebecca Zoe Weber, Geertje Mulders, Julia Kaiser, Christian Tackenberg, and Ruslan Rust. Deep learning based behavioral profiling of rodent stroke recovery. *BioRxiv*, 2021.

[R232] Michael H McCullough and Geoffrey J Goodhill. Unsupervised quantification of naturalistic animal behaviors for gaining insight into the brain. *Current Opinion in Neurobiology*, 70:89–100, 2021.

[R233] Ilya E Monosov, Jan Zimmermann, Michael J Frank, Mackenzie W Mathis, and Justin T Baker. Ethological computational psychiatry: Challenges and opportunities. *Current Opinion in Neurobiology*, 86:102881, 2024.

[R234] Z Taiwanica. Ethom: event-recording computer software for the study of animal behavior. *Acta Zool. Taiwanica*, 11:47–61, 2000.

[R235] J Morrow-Tesch, JW Dailey, and H Jiang. A video data base system for studying animal behavior. *Journal of animal science*, 76(10):2605–2608, 1998.

[R236] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[R237] Emma Lynch, Lisa Angeloni, Kurt Fristrup, Damon Joyce, and George Wittemyer. The use of on-animal acoustical recording devices for studying animal behavior. *Ecology and Evolution*, 3(7):2030–2037, 2013.

[R238] Tomoya Nakamura, Jumpei Matsumoto, Hiroshi Nishimaru, Rafael Vieira Bretas, Yusaku Takamura, Etsuro Hori, Taketoshi Ono, and Hisao Nishijo. A markerless 3d computerized motion capture system incorporating a skeleton model for monkeys. *PloS one*, 11(11):e0166154, 2016.

[R239] Alessio Paolo Buccino, Mikkel Elle Lepperød, Svenn-Arne Dragly, Philipp Häfliger, Marianne Fyhn, and Torkel Hafting. Open source modules for tracking animal behavior and closed-loop stimulation based on open ephys and bonsai. *Journal of neural engineering*, 15(5):055002, 2018.

[R240] Max Bain, Arsha Nagrani, Daniel Schofield, Sophie Berdugo, Joana Bessa, Jake Owen, Kimberley J Hockings, Tetsuro Matsuzawa, Misato Hayashi, Dora Biro, et al. Automated audiovisual behavior recognition in wild primates. *Science advances*, 7(46):eabi4883, 2021.

[R241] David J Anderson and Pietro Perona. Toward a science of computational ethology. *Neuron*, 84(1):18–31, 2014.

[R242] Anthony I Dell, John A Bender, Kristin Branson, Iain D Couzin, Gonzalo G de Polavieja, Lucas PJJ Noldus, Alfonso Pérez-Escudero, Pietro Perona, Andrew D Straw, Martin Wikelski, et al. Automated image-based tracking and its application in ecology. *Trends in ecology & evolution*, 29(7):417–428, 2014.

[R243] Jens Krause, Stefan Krause, Robert Arlinghaus, Ioannis Psorakis, Stephen Roberts,

and Christian Rutz. Reality mining of animal social systems. *Trends in ecology & evolution*, 28(9):541–551, 2013.

[R244] Eldar Insafutdinov, Mykhaylo Andriluka, Leonid Pishchulin, Siyu Tang, Evgeny Levinkov, Bjoern Andres, and Bernt Schiele. Arttrack: Articulated multi-person tracking in the wild. In *CVPR'17*, 2017.

[R245] Eldar Insafutdinov, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. Deepercut: A deeper, stronger, and faster multi-person pose estimation model. 2016.

[R246] Sena Isik and Gunes Unal. Open-source software for automated rodent behavioral analysis. *Frontiers in Neuroscience*, 17:1149027, 2023.

[R247] Shaokai Ye, Anastasiia Filippova, Jessy Lauer, Steffen Schneider, Maxime Vidal, Tian Qiu, Alexander Mathis, and Mackenzie Weygandt Mathis. Superanimal pretrained pose estimation models for behavioral analysis. *Nature Communications*, 15(1):5165, 2024.

[R248] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[R249] Alexander I Hsu and Eric A Yttri. B-soid, an open-source unsupervised algorithm for identification and fast prediction of behaviors. *Nature communications*, 12(1):5188, 2021.

[R250] Kang Huang, Yaning Han, Ke Chen, Hongli Pan, Gaoyang Zhao, Wenling Yi, Xiaoxi Li, Siyuan Liu, Pengfei Wei, and Liping Wang. A hierarchical 3d-motion learning framework for animal spontaneous behavior mapping. *Nature communications*, 12(1):2784, 2021.

[R251] Fernand Meyer. Topographic distance and watershed lines. *Signal processing*, 38(1):113–125, 1994.

[R252] Gordon J Berman, Daniel M Choi, William Bialek, and Joshua W Shaevitz. Mapping the stereotyped behaviour of freely moving fruit flies. *Journal of The Royal Society Interface*, 11(99):20140672, 2014.

[R253] Timothy W Dunn, Jesse D Marshall, Kyle S Severson, Diego E Aldarondo, David GC Hildebrand, Selmaan N Chettih, William L Wang, Amanda J Gellis, David E Carlson, Dmitriy Aronov, et al. Geometric deep learning enables 3d kinematic profiling across species and environments. *Nature methods*, 18(5):564–573, 2021.

[R254] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[R255] Jea Kwon, Sunpil Kim, Dong-Kyum Kim, Jinhyeong Joo, SoHyung Kim, Meeyoung Cha, and C Justin Lee. Subtle: An unsupervised platform with temporal link embedding that maps animal behavior. *bioRxiv*, pages 2023–04, 2023.

[R256] Alexander B Wiltschko, Tatsuya Tsukahara, Ayman Zeine, Rockwell Anyoha, Winthrop F Gillis, Jeffrey E Markowitz, Ralph E Peterson, Jesse Katon, Matthew J Johnson, and Sandeep Robert Datta. Revealing the structure of pharmacobehavioral space through motion sequencing. *Nature neuroscience*, 23(11):1433–1443, 2020.

[R257] Jennifer J Sun, Ann Kennedy, Eric Zhan, David J Anderson, Yisong Yue, and Pietro Perona. Task programming: Learning data efficient behavior representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2876–2885, 2021.

[R258] Albert Tseng, Jennifer J Sun, and Yisong Yue. Automatic synthesis of diverse weak supervision sources for behavior analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2211–2220, 2022.

[R259] Tianxun Zhou, Calvin Chee Hoe Cheah, Eunice Wei Mun Chin, Jie Chen, Hui Jia Farm, Eyleen Lay Keow Goh, and Keng-Hwee Chiam. Constrastivepose: A contrastive learning approach for self-supervised feature engineering for pose estimation and behavorial classification of interacting animals. *bioRxiv*, pages 2022–11, 2022.

[R260] Kevin Luxem, Petra Mocellin, Falko Fuhrmann, Johannes Kürsch, Stefan Remy, and Pavol Bauer. Identifying behavioral structure from deep variational embeddings of animal motion. *BioRxiv*, pages 2020–05, 2022.

[R261] Caleb Weinreb, Jonah E Pearl, Sherry Lin, Mohammed Abdal Monium Osman, Libby Zhang, Sidharth Annapragada, Eli Conlin, Red Hoffmann, Sofia Makowska, Winthrop F Gillis, et al. Keypoint-moseq: parsing behavior by linking point tracking to pose dynamics. *Nature methods*, 21(7):1329–1339, 2024.

[R262] Heiko Dankert, Liming Wang, Eric D Hoopfer, David J Anderson, and Pietro Perona. Automated monitoring and analysis of social behavior in drosophila. *Nature methods*, 6(4):297–303, 2009.

[R263] Xavier P Burgos-Artizzu, Piotr Dollár, Dayu Lin, David J Anderson, and Pietro Perona. Social behavior recognition in continuous video. In *2012 IEEE conference on computer vision and pattern recognition*, pages 1322–1329. IEEE, 2012.

[R264] Weizhe Hong, Ann Kennedy, Xavier P Burgos-Artizzu, Moriel Zelikowsky, Santiago G Navonne, Pietro Perona, and David J Anderson. Automated measurement of mouse social behaviors using depth sensing, video tracking, and machine learning. *Proceedings of the National Academy of Sciences*, 112(38):E5351–E5360, 2015.

[R265] Fabrice De Chaumont, Elodie Ey, Nicolas Torquet, Thibault Lagache, Stéphane Dallongeville, Albane Imbert, Thierry Legou, Anne-Marie Le Sourd, Philippe Faure, Thomas Bourgeron, et al. Real-time analysis of the behaviour of groups of mice via a

depth-sensing camera and machine learning. *Nature biomedical engineering*, 3(11):930–942, 2019.

[R266] Simon RO Nilsson, Nastacia L Goodwin, Jia Jie Choong, Sophia Hwang, Hayden R Wright, Zane C Norville, Xiaoyu Tong, Dayu Lin, Brandon S Bentzley, Neir Eshel, et al. Simple behavioral analysis (simba)–an open source toolkit for computer classification of complex social behaviors in experimental animals. *BioRxiv*, pages 2020–04, 2020.

[R267] JBI Rousseau, PBA Van Lochem, WH Gispen, and BM Spruijt. Classification of rat behavior with an image-processing method and a neural network. *Behavior Research Methods, Instruments, & Computers*, 32:63–71, 2000.

[R268] Shuzo Sakata. Salsa: a combinatory approach of semi-automatic labeling and long short-term memory to classify behavioral syllables. *bioRxiv*, pages 2023–04, 2023.

[R269] Mayank Kabra, Alice A Robie, Marta Rivera-Alba, Steven Branson, and Kristin Branson. Jaaba: interactive machine learning for automatic annotation of animal behavior. *Nature methods*, 10(1):64–67, 2013.

[R270] Jens F. Schweihoff, Alexander I. Hsu, Martin K. Schwarz, and Eric A Yttri. A-soid, an active learning platform for expert-guided, data efficient discovery of behavior. *bioRxiv*, 2022.

[R271] Saquib Sarfraz, Naila Murray, Vivek Sharma, Ali Diba, Luc Van Gool, and Rainer Stiefelhagen. Temporally-weighted hierarchical clustering for unsupervised action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11225–11234, 2021.

[R272] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[R273] Tadeusz Caliński and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27, 1974.

[R274] David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.

[R275] Google active learning playground github repository. `https://github.com/google/active-learning`, 2017.

[R276] Hendrik Edelhoff, Johannes Signer, and Niko Balkenhol. Path segmentation for beginners: an overview of current methods for detecting changes in animal movement patterns. *Movement ecology*, 4(1):1–21, 2016.

[R277] Mohammad Etemad, Amilcar Soares, Elham Etemad, Jordan Rose, Luis Torgo, and Stan Matwin. Sws: an unsupervised trajectory segmentation algorithm based on change detection with interpolation kernels. *GeoInformatica*, 25:269–289, 2021.

[R278] Francis R Willett, Donald T Avansino, Leigh R Hochberg, Jaimie M Henderson, and Krishna V Shenoy. High-performance brain-to-text communication via handwriting. *Nature*, 593(7858):249–254, 2021.

[R279] Nicholas S. Card, Maitreyee Wairagkar, Carrina Iacobacci, Xianda Hou, Tyler Singer-Clark, Francis R. Willett, Erin M. Kunz, Chaofei Fan, Maryam Vahdati Nia, Darrel R. Deo, Aparna Srinivasan, Eun Young Choi, Matthew F. Glasser, Leigh R. Hochberg, Jaimie M. Henderson, Kiarash Shahlaie, Sergey D. Stavisky, and David M. Brandman. An accurate and rapidly calibrating speech neuroprosthesis. *New England Journal of Medicine*, 391(7):609–618, 2024.

[R280] Jon P Nedel, Rita Singh, and Richard M Stern. Phone transition acoustic modeling: application to speaker independent and spontaneous speech systems. In *INTERSPEECH*, pages 572–575, 2000.

[R281] Tyler Benster, Guy Wilson, Reshef Elisha, Francis R Willett, and Shaul Druckmann. A cross-modal approach to silent speech with llm-enhanced recognition. *arXiv preprint arXiv:2403.05583*, 2024.

[R282] Yohann Benchetrit, Hubert Banville, and Jean-Rémi King. Brain decoding: toward real-time reconstruction of visual perception. *arXiv preprint arXiv:2310.19812*, 2023.

[R283] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

[R284] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[R285] Francis R Willett, Erin M Kunz, Chaofei Fan, Donald T Avansino, Guy H Wilson, Eun Young Choi, Foram Kamdar, Matthew F Glasser, Leigh R Hochberg, Shaul Druckmann, Krishna Shenoy, and Jaimie M. Henderson. Data for: A high-performance speech neuroprosthesis [dataset]. *Dryad*, pages 1–6, 2023.

[R286] Alexander Epaneshnikov Reece H. Dunn, Valdis Vitolins. espeak ng text-to-speech. *GitHub*, 2015.

[R287] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. arxiv (2022). *arXiv preprint arXiv:2212.04356*, 2022.

[R288] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[R289] Meinard Müller. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84, 2007.