Learning Structured Representations of the Visual World

Matthew Wallingford

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington
2025

Reading Committee:
Ali Farhadi, Chair
Ranjay Krishna
Roozbeh Mottaghi

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Learning Structured Representations of the Visual World

Matthew Wallingford

Chair of the Supervisory Committee:

Ali Farhadi
Paul G. Allen School of Computer Science and Engineering

Humans develop complex internal models of the world which allow us to generalize remarkably well to new scenarios and tasks. While deep learning has steadily improved in performance through data and scale, it conspicuously lags behind in its generalization to changing data distributions and transfer across tasks when compared to biological intelligence. We argue that one key element absent from current deep learning systems is this internal model of the world to enable efficient transfer of knowledge to new settings and data. In this work, we investigate how aspects of world models such as compositionality and 3D spatial understanding can be learned from visual data and be used to improve the efficiency and robustness of current machine learning systems. We develop new methods and loss objectives for learning structured representations. We demonstrate how learning from more complex visual data such as video, embodied exploration, and 360° video enables learning more structured world models which improves sample efficiency and spatial understanding. In addition, we explore other directions and develop methods to improve the transfer of knowledge between tasks and robustness to shifting data distributions.

Acknowledgements

I've been fortunate to have had a great advisor, friends, and mentors throughout my PhD which has made the experience far more enjoyable. I'd like to thank Ali for being the best possible advisor. His wisdom about both research and life has been invaluable. I'd like to thank Ani for being a supportive mentor and I admire his dedication to fostering young researchers. I'd like to thank Roozbeh for his mentorship and I'm grateful for his valuable advice and feedback throughout my PhD. Last, I'd like to thank Ranjay for his support and willingness to be part of my committee.

I am lucky to have had amazing lab mates (Aditya, James, Mitchell, Gabe, Vivek, Sarah, Reza, Ainaz, Nabil, Yashas, "young" Matt, Sebastin, Yashas, Etash, Thao, Wei-Chiu, and more), many who I can call close friends and collaborators. Working remotely during COVID gave me greater appreciation for being around other talented researchers with whom I could share and discuss ideas. I've learned a wide range of skills from my lab mates from niche latex tricks to best practices for empirical deep learning. Apart from research, I appreciate the positive and collaborative culture that everyone brought to the lab. Our annual hikes to Rainier, playing beach volleyball (poorly) after work, and spontaneous dinner trips to Biryani Pizza House are just some of the memorable lab activities that I'll cherish. In particular, I'd like to thank Aditya for always believing in me and my research as well as being a great friend.

I'd like to thank my partner Clare who was always supportive and generous with her time, even while completing her own PhD. Clare's unwavering encouragement and love gave me strength and motivation when facing difficult challenges throughout the PhD. Finally, I'd like to thank my parents and sister for their love and support throughout my life. They've always encouraged me to pursue my passions and fostered my intellectual curiosity.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The ability to learn from vast collections of data and transfer that knowledge to other tasks is a hallmark of deep learning that has led to remarkable advancements. Learning transferrable priors and adapting them to different data is key to much of the progress in large language modeling and modern vision we see today. While this phenomena has been studied over time across many sub-fields such as representation learning, transfer learning, pretraining/post training, domain adaptation, and meta-learning, the goal of learning good representations and efficiently transferring them has remained the same.

Though deep learning's ability to transfer knowledge was a tremendous departure from that of previous machine learning methods, it is conspicuously brittle and inefficient in this regard compared to biological intelligence. We see this manifest frequently in various forms across different mediums of data. In computer vision, one notable instance would be the brittleness to distribution shift. For example, a model that is trained on ImageNet and evaluated on ImageNet-v2 (nearly identical datasets with imperceptible differences) performs significantly worse on the latter. In contrast, a human that has learned to classify images from ImageNet has no performance drop when evaluated on ImageNet-v2. In language we observe surprising brittleness in the inability of the model to count the number of r's in the word strawberry. Though the model has clearly trained on data concerning the english alphabet and numerical counting, it cannot generalize to count letters which would be trivial for humans despite no explicit practice in such an activity.

We hypothesize that one source of the gap in transfer efficiency between deep learning and biological intelligence is in how the representations are learned and structured. The prevailing paradigm in computer

vision has been to pretrain on large-scale internet images such as [213, 201] and to learn a feature extractor which outputs a flat vector of scalars. This representation looks quite different from the internal models of the world that humans develop which are composed of objects, have spatial and dynamic understanding, and contain hierarchical semantics [99, 174]. For example, a person learning to drive starts with vast prior knowledge that helps them learn the task. Not only do they start with a nuanced understanding of how cars and pedestrians act, but they also have broad world knowledge such as intuition for physical concepts like inertia and friction and social intuition that a pedestrian on their phone might be distracted. We often see this lack of broad world knowledge in the mistakes that self-driving systems make such as perceiving shadows as solid objects or perceiving overhead bridges as large trucks. It's unreasonable to expect a vision model trained to classify static images from the internet to learn such intricate models of the world.

With this motivation in mind, we work to develop methods and learning objectives that lead to more structured models of the world and investigate what advantages they afford such as sample-efficiency and robustness. Computer vision has historically focused on narrow and controlled benchmarks which are less likely to require models of the world such as image classification, object detection, and semantic segmentation. We start by investigating whether a structured representation learned from exploring a simulated environment is effective for the task of embodied navigation. We design a learning objective which decomposes the environments into a finite set of discrete objects and reconstruct the 3D environment from these assets. We show that the resulting 3D representation leads to improved performance, sample-efficiency, and spatial understanding.

We then investigate how this general approach can be extended from simulation to the real-world which presents new challenges particularly from the perspective of data. We develop a methodology for structuring 360° video data to train a model with a novel view synthesis objective. We collect over a million 360° videos and train one of the first models capable of synthesizing real-world scenes from a single image. We then show the potential for this model to be used for trajectory planning and other downstream applications. Chapter 3 concludes the main efforts at understanding and developing methods for learning world models. The following chapters focus on understanding and improving transfer efficiency and robustness through other research directions.

In chapter 4 we explore a tangential direction for improving the sample-efficiency for image classification.

We look at a retrieval-based approach where the model recalls training data that closely matches the task at test time. The method is inspired by the concept of cognitive priming in humans where exposure to a specific stimulus activates neurons associated with a task resulting in better performance or recall. We find that retrieving previously observed data at test time improves robustness to distribution shift and improves accuracy.

In chapter 5 we design a method for learning which parameters to transfer across tasks. Often tasks have overlapping objectives and thus benefit from sharing some, but not all parameters. Previous approaches typically used fixed architectures for multi-task learning, but this was suboptimal as which parameters to share depends on the relationship between the tasks. We show that a desirable configuration can be found quickly for minimal overhead by differentiably searching over a combinatorial number of possibilities.

Finally we conclude with chapter 6 where we design an evaluation framework for understanding the performance of a learned representation. Often representations are evaluated on the same set of static closed-world benchmarks. We find that representation learning methods have over-optimized to this narrow set of evaluations and that a more holistic evaluation shows that many of the state-of-the-art methods are quite brittle to the experimental setting they were designed for.

Chapter 2

Neural Radiance Field Codebooks

2.1 Overview

Compositional representations of the world are a promising step towards enabling high-level scene understanding and efficient transfer to downstream tasks. Learning such representations for complex scenes and tasks remains an open challenge. Towards this goal, we introduce Neural Radiance Field Codebooks (NRC), a scalable method for learning object-centric representations through novel view reconstruction. NRC learns to reconstruct scenes from novel views using a dictionary of object codes which are decoded through a volumetric renderer. This enables the discovery of reoccurring visual and geometric patterns across scenes which are transferable to downstream tasks. We show that NRC representations transfer well to object navigation in THOR, outperforming 2D and 3D representation learning methods by 3.1% success rate. We demonstrate that our approach is able to perform unsupervised segmentation for more complex synthetic (THOR) and real scenes (NYU Depth) better than prior methods (29% relative improvement). Finally, we show that NRC improves on the task of depth ordering by 5.5% accuracy in THOR. Project Website and Code can be found here.

2.2 Introduction

Parsing the world at the abstraction of objects is a key characteristic of human perception and reasoning [198, 98]. Such object-centric representations enable us to infer attributes such as geometry, affordances, and

physical properties of objects solely from perception [224]. For example, upon perceiving a cup for the first time one can easily infer how to grasp it, know that it is designed for holding liquid, and estimate the force needed to lift it. Learning such models of the world without explicit supervision remains an open challenge.

Unsupervised decomposition of the visual world into objects has been a long-standing challenge [214]. More recent work focuses on reconstructing images from sparse encodings as an objective for learning object-centric representations [22, 66, 137, 131, 157, 220]. The intuition is that object encodings which map closely to the underlying structure of the data should provide the most accurate reconstruction given a limited encoding size. Such methods have shown to be effective at decomposing 2D games and simple synthetic scenes into their parts. However, they rely solely on color cues and do not scale to more complex datasets [101, 170].

Advances in neural rendering [153, 274] have enabled learning geometric representations of objects from 2D images. Recent work has leveraged scene reconstruction from different views as a source of supervision for learning object-centric representations [228, 282, 204, 221]. However, such methods have a few key limitations. The computational cost of rendering scenes grows linearly with the number of objects which inhibits scaling to more complex datasets. Additionally, the number of objects per scene is fixed and fails to consider variable scene complexity. Finally, objects are decomposed on a per scene basis, therefore semantic and geometric information is not shared across object categories.

With this in consideration we introduce, Neural Radiance Codebooks (NRC). NRC learns a codebook of object categories which are composed to explain the appearance of 3D scenes from multiple views. By reconstructing scenes from different views NRC captures reoccurring geometric and visual patterns to form object categories. This learned representation can be used for segmentation as well as geometry-based tasks such as object navigation and depth ordering. Furthermore, NRC resolves the limitations of current 3D object-centric methods. First, NRC's method for assigning object codes to regions of the image enables *constant rendering compute* whereas that of other methods scales with number of objects. Second, we introduce a novel mechanism for differentiably adding new categories which allows the codebook to scale with the complexity of the data. Last, modeling intra-category variation in conjunction with the codebook enables sharing of semantic and geometric object information across scenes.

We evaluate NRC on unsupervised segmentation, object navigation, and depth ordering. For segmentation

on indoor scenes from ProcTHOR [40] we show 29.4% relative ARI improvement compared to current 3D object-centric methods [228, 282]. On real-world images (NYU Depth [218]) we show promising qualitative results (Figure 2.3) and 29% relative improvement. For object navigation and depth ordering, where geometric understanding is relevant, we observe 3.1% improvement in navigation success rate 5.5% improvement in depth ordering accuracy over comparable self-supervised and object-centric methods. Interestingly, we find qualitative evidence that the learned codes categorize objects by both visual appearance and geometric structure (Figure 2.1).

2.3 Related Work

Object-Centric Learning Object-centric learning aims to build compositional models of the world from building blocks which share meaningful properties and regularities across scenes. Prior works such as MONet [22], IODINE [66], Slot Attention [137], and [157] have demonstrated the potential for disentangling objects from images. Other work has shown the ability to decompose videos [100, 105]. In particular, Marionette [220] learns a shared dictionary for decomposing scenes of 2D sprites. We draw inspiration from MarioNette for learning codebooks, but differ in that we model the image formation process and intra-code variation, and dynamically add codes to our dictionary.

3D Object-Centric Learning Recent work has shown novel view reconstruction to be a promising approach for disentangling object representations. uORF [282] and ObSuRF [228] combine Slot Attention with Neural Radiance Fields [153] to decompose scenes. COLF [221] replaces the volumetric renderer with light fields to improve computational efficiency. NeRF-SOS [51] uses contrastive loss for both geometry and appearance to perform object segmentation. SRT [205] encodes scenes into a set of latent vectors which are used to condition a light field. OSRT [204] extends SRT by explicitly assigning regions of the image to latent vectors. NeSF [250] learns to perform 3D object segmentation using NeRF with 2D supervision. Although great progress has been made, these methods are limited to synthetic and relatively simple scenes. Our work differs from previous 3D object-centric works in that we learn reoccurring object codes across scenes and explicitly localize the learned codes. Additionally, our method can model an unbounded number of objects per scene compared to prior work which fixes this hyper-parameter a priori. We show that our approach generalizes to

Figure 2.1: Visualization of learned codes. The NRC codebook encodes reoccurring geometric and visual patterns. In the top row, couches of differing appearance are grouped by geometric structure. In the middle row, different textured floors are categorized based on their shared planar geometry. In the bottom row, NRC learns correspondences between fridges from different views and scenes.

more complex synthetic and real-world scenes.

Neural Rendering Advances in neural rendering, in particular Neural Radiance Fields (NeRF) [153], have enabled a host of new applications [93, 154, 179, 171, 122, 165, 287]. NeRF differentiably renders novel views of a scene by optimizing a continuous volumetric scene function given a sparse set of input views. Original formulation of NeRF learned one representation for each scene; other works [281, 89, 112] showed

conditioning NeRFs on images enables generation of novel views of new scenes.

Dictionary/Codebook Learning Dictionary (codebook) learning [168] involves learning of a specific set of atoms or codes that potentially form a basis and span the input space through sparse combinations. Codebooks have been widely used for generative and discriminative tasks across vision [49, 141], NLP [149] and signal processing [85]. Learning sparse representations based on codes enables large-scale methods which rely on latent representations. More recently, codebooks have been shown to be crucial in scaling discrete representation learning [245, 118]. Marionette [220] is an object-centric representation learning method that relies on codebooks, unlike most other methods that are developed around set latent representations [204, 205, 137, 282]. Object-centric codebooks help in semantic grounding for transfer between category instances and are important for large-scale representation learning across diverse scenes and objects.

2.4 Method

Our goal is to discover object categories without supervision, learn priors over their geometry and visual appearance, and model the variation between instances belonging to each group. Given multiple views of a scene, the objective is to explain all views of the scene given a set of object-codes. This learned decomposition can be used for segmentation and other downstream tasks that require semantic and geometric understanding such as depth ordering and object-navigation.

Figure 6.1 illustrates the training pipeline. We begin by processing the image through a convolutional network to obtain a spatial feature map. The feature map is then projected to a novel view using the relative camera matrix. Feature vectors from each respective region of the image are assigned to categorical latent codes from the finite-size codebook. The object codes and feature vectors are passed to a convolutional network which transform the categorical codes to fine-grained instance codes. A volumetric renderer is then conditioned on the instance code, view direction, and positional encodings to render each region of the scene from the novel view. The rendered image from the novel view is compared to the ground truth using L_2 pixel loss. The categorical codes, assignment mechanism, and volumetric renderer are learned jointly in an end-to-end fashion.

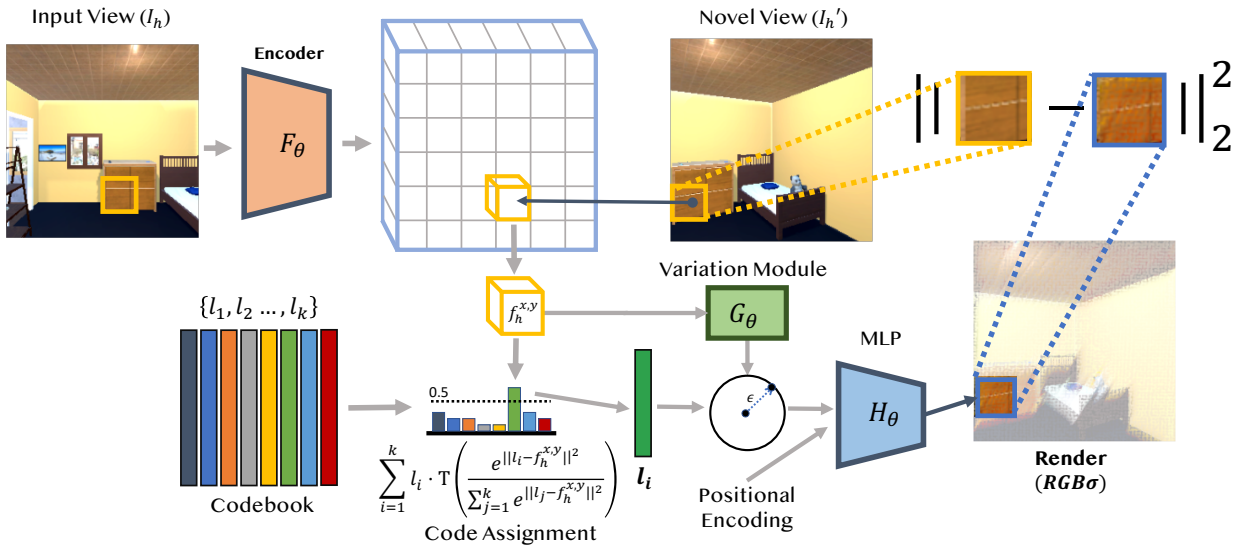


Figure 2.2: An overview of NRC. We learn a set of shared codes for decomposing scenes into objects. Each point in the scene is assigned one of n latent codes from the codebook. The variation module models the intra-code variation between objects by perturbing the code in latent space. A conditional NERF model renders the scene and is compared to the ground truth novel view for supervision.

Image Encoding and Camera Projection Given an input frame and novel image, $I_h, I'_h \in \mathcal{R}^{3\times H\times W}$ respectively from scene S_h, we first encode I_h into a spatial feature map, $f_h \in R^{d\times H/k\times W/k}$, using a convolutional network, F_θ. We project each point, (x, y, z), in world coordinates of the novel view to camera coordinates in the input frame, (x, y), using the relative camera pose. Given (x, y), we select the spatial feature $f_h^{x,y} \in \mathcal{R}^d$ from the patch that contains the projected coordinates. The spatial features vectors are then passed to the next stage where they are assigned to categorical object-codes.

Assigning + Learning Codes Our goal is to jointly learn a shared set of object categories and priors about their appearance and geometry. By mapping the spatial features from a continuous vector space to a discrete, finite set of codes the model is incentivized to find reoccurring patterns in the images.

Given the features for a point in the novel view, $f_h^{x,y}$, we assign a code, l^*, chosen from the shared codebook, $\mathbb{L}$. We do so with an $\arg\max$ 1-nearest-neighbors during inference:

$$l^*(x,y) \leftarrow \underset{l_i;\, i\in[k]}{\overset{\text{(STE)}}{\arg\max}} \frac{e^{-\|l_i - f_h^{x,y}\|_2}}{\sum_{j=1}^{k} e^{-\|l_j - f_h^{x,y}\|_2}} \tag{2.1}$$

The nearest-neighbor assignment used during inference is a non-differentiable operation therefore

propagating gradients to the encoder and codebook would not be possible. To enable learning of the codebook elements we use a softmax relaxation of nearest-neighbors during the backward pass in conjunction with the straight-through-estimator (STE) [14]:

$$l^*_{back}(x,y) \leftarrow \frac{e^{-\|l_i - f_h^{x,y}\|_2}}{\sum_{j=1}^{k} e^{-\|l_j - f_h^{x,y}\|_2}} \tag{2.2}$$

Adding Categorical Codes The number of codes should depend on the complexity of the scenes they model. Learning when to add new codes is non-trivial because the number and selection of codes is discrete and non-differentiable. To circumvent this problem, we use a series of step functions with a straight-through-estimator (STE) to sequentially add elements to the codebook. Each code $l_i \in \mathbb{L}$ is gated according to the following:

$$l_i \leftarrow \mathcal{T}\left(\boldsymbol{\sigma}(s - {}^{i^2}/_{\lambda}), \frac{1}{2}\right) \cdot l_i; \;\; \mathcal{T}(a,t) := \begin{cases} 1, & a > t \\ 0, & a \leq t \end{cases} \tag{2.3}$$

$\mathcal{T}(.)$ is a binarization function in the forward pass and lets the gradients pass through using STE in the back pass. $\boldsymbol{\sigma}(\cdot)$ is the sigmoid function, λ is a scaling hyperparameter, and s is a learnable scoring parameter whose magnitude is correlated with the overall capacity (number of codes) required to model the scenes accurately. A new code l_i is added when s exceeds the threshold i^2/λ. New codes are initialized using a standard normal distribution. Throughout training we keep $k+1$ total codes where k is the current number of learnable codes. The extra code is used by the straight-through-estimator to optimize for s on the backward pass. This formulation can be viewed as the discrete analog of a gaussian prior over the number of elements, k in the codebook: $P(k) = e^{-k^2/\lambda}$.

Modeling Intra-Code Variation Once a categorical code has been assigned to a region in the novel frame, the model must account for variation across instances. We model this variation in latent space using an encoder that takes in both the spatial feature, $f_h^{x,y}$, and the categorical code $l^*(x,y)$. We rescale the norm of the variation vector by ϵ, a hyperparameter, to ensure the instance and categorical codes are close in latent

space. The instance code is formulated as the following:

$$l^*_{\text{instance}}(x,y) = l^*(x,y) + \epsilon \cdot \frac{G_{\theta'}([l^*(x,y), f_h^{x,y}])}{||G_{\theta'}([l^*(x,y), f_h^{x,y}])||_2}. \tag{2.4}$$

We concatenate $f_h^{x,y}$ and $l^*(x,y)$ as input to the variation module, $G_{\theta'}$, which we model as a 3-layer convolutional network. $G_{\theta'}$ provides a d dimensional perturbation vector which models the intra-category variation and transforms the categorical code to an instance level code.

Decoding and Rendering Given the localized instance codes for a scene, we render it in the novel view and compare with the ground truth using $L2$ pixel loss. Intuitively, object categories which encode geometric and visual patterns should render the scene more accurately from novel views. Each region of the scene is rendered using an MLP conditioned on the instance codes and the volumetric rendering equation following the convention of NeRF [153]:

$$H_{\hat{\theta}}(l^*_{\text{instance}}(x,y), \mathbf{p}, \mathbf{d}) = (\mathbf{c}, \sigma), \tag{2.5}$$

Here $\mathbf{p} = (x,y,z)$ is a coordinate in the scene, $\mathbf{d} \in \mathcal{R}^3$ is a view direction, $\mathbf{c}$ is the RGB value at $\mathbf{p}$ in the direction of $\mathbf{d}$ and σ is the volume density at that point. Recall that (x,y,z) corresponds to (x,y) in the input frame I_h. We can project (x,y,z) into the camera coordinates of the novel view I'_h to get (x',y'). This pixel (x',y') in the novel view corresponds to (x,y) in the input frame, meaning they represent the same point in world coordinates. To get an RGB value for (x,y), we use volume rendering along the ray from camera view I_h into the scene, given by

$$\hat{\mathbf{C}}(\mathbf{r}) = \int_{t_n}^{t_f} T(t) \cdot \sigma(t) \cdot \mathbf{c}(t) \cdot dt, \tag{2.6}$$

where $T(t) = \exp\left(-\int_{t_n}^{t} \sigma(s) \cdot ds\right)$ models absorbance and t_n and t_f are the near and far field. Given a target view with pose $\mathbf{P}$, the ray to the target camera is given by $\mathbf{r}(t) = \mathbf{o} + t \cdot \mathbf{d}$ where $\mathbf{d}$ is a unit direction vector which passes through (x,y). The volume rendering for a particular pixel occurs along this ray. Let $\mathbf{d}'$ be the direction associated with the novel view (x',y') and $\mathbf{r}'(t) = \mathbf{o} + t \cdot \mathbf{d}'$. The pixel intensity at (x',y') is

given by $\hat{\mathbf{C}}' = \hat{\mathbf{C}}(\mathbf{r}')$ and our final loss is

$$\mathcal{L}(I_h, I'_h, x, y) = \|\hat{\mathbf{C}}' - I'_h(x', y')\|_2 + s, \tag{2.7}$$

where $I'_h(x', y')$ is the ground-truth pixel value at (x', y'). We penalize the scoring parameter, s, from section 2.4 in the loss to encourage learning a minimal number of codes.

NRC **for Downstream Tasks** Once the encoder, codebook, and MLP have been trained, we evaluate the learned representation on various downstream tasks. To perform segmentation, we process each image through the trained encoder, F_θ, to obtain the spatial feature map. Each feature vector, $f_{x,y}$ in the spatial map is assigned to the nearest categorical code, $l^\star$ in the learned codebook. The categorical codes are then designated to the corresponding pixel to obtain a segmentation mask.

Traditionally in object navigation, frames from the embodied agent are processed by a frozen, pretrained network. The resulting feature vector is then passed to a policy network which chooses an action. To assess the utility of the NRC representation, we replace the pretrained network with the NRC encoder and codebook. We process each frame to obtain instance codes for each region of the image which are then fed to the policy network.

Depth ordering task consists of predicting which of two objects is closer to the camera. To perform depth ordering with NRC we predict a segmentation mask and depth map. To predict the depth map we condition the trained MLP on the instance codes & predict the density, σ, along a given ray. We estimate the transmittance to predict the depth following the method of Yu et al. [281]. Depth map and segmentation mask are combined to predict the average distance of each object from the camera.

2.5 Experiments

We evaluate our decomposition and representations on several downstream tasks: unsupervised segmentation (real and synthetic), object navigation, and depth ordering. NRC shows improvement over baseline methods on all three tasks. Prior works in object-centric learning have focused on unsupervised segmentation for measuring the quality of their decomposition. We show that NRC representations are also effective

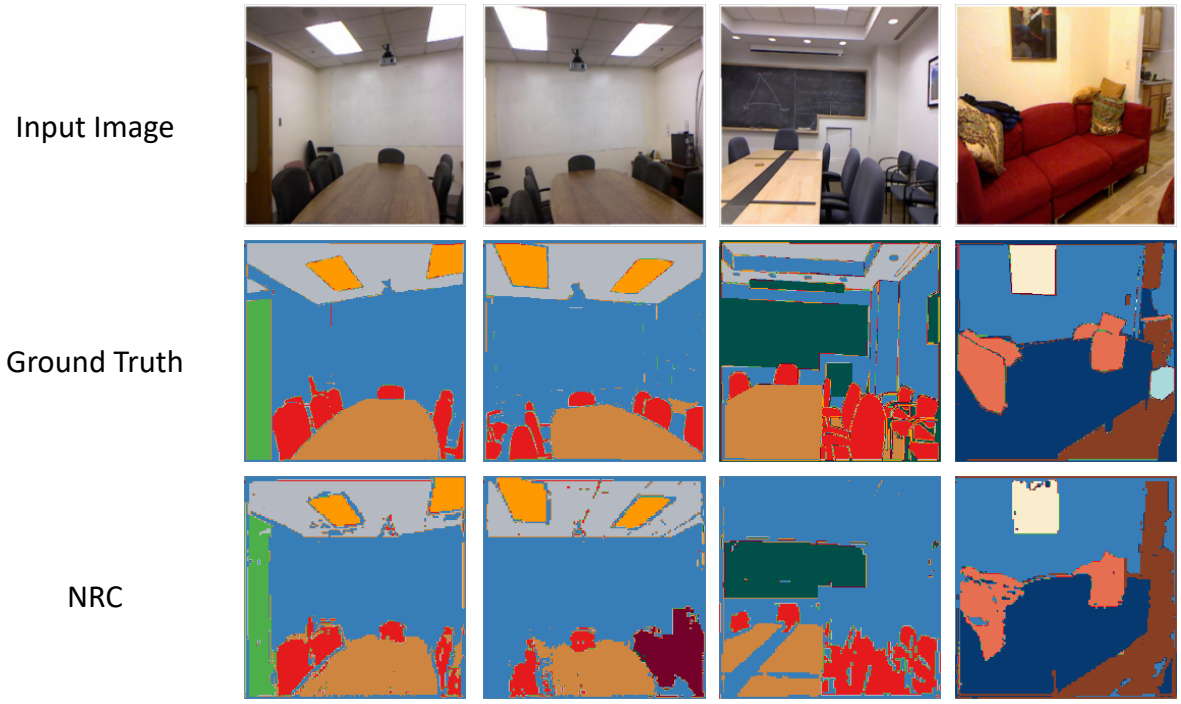


Figure 2.3: Unsupervised segmentation of real-world images. NRC segments scenes that have significant object category overlap with ProcTHOR. We show the first results for object-centric unsupervised segmentation of real-world scenes.

for downstream applications that require geometric and semantic understanding of scenes such as object navigation and depth ordering.

2.5.1 Datasets

ProcTHOR & RoboTHOR THOR [110] consists of interactive home environments built in the Unity game engine. We benchmark on the task of object navigation in RoboTHOR [38], a variant of the THOR environment aimed at sim2real benchmarking. Object navigation consists of an agent moving through different scenes to locate specified objects. RoboTHOR consists of 89 indoor scenes split between train, validation, and test. ProcTHOR [40] consists of procedurally generated indoor scenes similar to RoboTHOR.

CLEVR-3D CLEVR-3D [96] is a synthetic dataset consisting of geometric primitives from multiple views and is used for unsupervised segmentation. Following the convention of Stelzner et al. [228], we test on the first 500 scenes of the validation set and report foreground-adjusted random index (FG-ARI). Adjusted random index (ARI) [278] measures the agreement between two clusterings and is a standard metric for

unsupervised segmentation. In our case the two clusterings are the predicted and ground truth segmentations. Foreground adjusted random index only measures the ARI for pixels belonging to foreground objects. For comparison to prior works, we consider segmentations at both the class and instance level to be correct for CLEVR-3D, ProcTHOR, and NYU Depth.

NYU Depth The NYU Depth Dataset [218] consists of images from real-world indoor scenes accompanied by depth and segmentation maps. Methods are trained on the ProcThor dataset then evaluated on NYU Depth for segmentation. We chose NYU Depth because it has object categories and scene layouts that are similar to THOR. We report the adjusted random index (ARI).

Table 2.1: Segmentation results (ARI) for NRC and comparable methods. We find that for more complex datasets, ProcTHOR and NYU Depth, NRC outperforms other methods.

Method	ProcTHOR (ARI)	NYU Depth (ARI)	CLEVR-3D (FG-ARI)
MarioNette	.127	.035	-
uORF	.193	.115	.962
ObSuRF	.228	.141	**.978**
NRC	**.295**	**.182**	.977

2.5.2 Unsupervised Segmentation

Experimental Setup We evaluate NRC, ObSuRF, uORF, and MarioNette for unsupervised segmentation on ProcTHOR, CLEVR-3D, and NYU Depth. We compare with MarioNette because it uses a similar code mechanism for reconstruction. We report FG-ARI on CLEVR-3D for comparison to prior works and ARI on the other datasets. For NYU Depth evaluation we use the representations trained on ProcTHOR and only consider classes that are seen in the training dataset.

Results We find that for NYU Depth and ProcTHOR which have more complex layouts and object diversity, NRC significantly outperforms other methods (Table 2.1). Figure 2.1 shows examples of the object codes learned by ProcTHOR and Figure 2.3 shows segmentation examples of real-world images. To our knowledge, this is the first object-centric method which has shown unsupervised segmentation results for complex real-world images. We find that NRC categorizes similar objects across scenes based on both geometry and visual appearance. In the top row of Figure 2.1, we find that couches of similar shape are assigned to the

same code despite differing visual appearance which indicates that NRC codes capture geometric similarity. In the same figure, we show further examples where floors of different texture are categorized by the same code. The improved segmentation performance and geometric categorization of objects indicates that NRC can leverage more than simple color cues which has been significant limitation for object-centric learning.

2.5.3 Object Navigation

Experimental Setup We design the object navigation experiments in THOR to understand how well the learned representations transfer from observational data to embodied navigation [6, 11]. Object navigation consists of an embodied agent with the goal of moving through indoor scenes to specified objects. The agent can rotate its camera and move in discrete directions. At each step, agent is fed with the current RGB frame relayed by the camera.

For the representation learning component of the experiment we collect observational video data from a heuristic planner, which walks through procedurally generated ProcTHOR scenes. In total, the dataset consists of 1.5 million video frames from 500 indoor scenes.

After training on ProcThor videos, we freeze the visual representations following standard practice [103]. We train a policy using DD-PPO [262] for 200M steps on the training set of RoboTHOR then evaluate on the test set. We report success rate (SR) and success weighted by path length (SPL). Success is defined as the agent signaling the stop action within 1 meter of the goal object with it in the view. SPL is defined as $\frac{1}{N}\sum_{i=1}^{N} S_i \frac{\ell_i}{\max(p_i,\ell_i)}$, where l_i is the shortest possible path, p_i is the taken path, & S_i is the binary indicator of success for episode i.

We compare with the following baselines: ObSuRF, uORF, Video MoCo [54], and EmbedCLIP [103]. ObSuRF and uORF are 3D, object-centric methods, and Video MoCo is a contrastive video representation learning method. We include Video MoCo for comparison as it was designed for large-scale, discriminative tasks, while ObSuRF and uORF were primarily intended for segmentation.

Results Table 2.2 shows the performance of NRC and baselines on RoboTHOR object navigation. NRC outperforms the best baseline by 3% in success rate (SR) and by a 20% relative improvement in SPL. These performance gains indicate that the geometrically-aware NRC representation provide an advantage over traditional representations such as EmbCLIP and Video MoCo. Another key observation is that NRC has

Table 2.2: Results for object navigation on RoboTHOR object navigation. Visual representations are trained on observations from 500 scenes of ProcThor. A policy is learned on top of the frozen visual representations by training on the object navigation task in RoboTHOR training scenes. The results are obtained by evaluating on RoboTHOR test scenes.

Method	Success Rate (%)	SPL
uORF [282]	31.3	.146
ImageNet Pretraining	33.4	.150
ObSuRF [228]	38.9	.167
Video MoCo [54]	43.9	.184
EmbCLIP [103]	47.0	.200
NRC (Ours)	**50.1**	**.239**

at least 18.8% improvement in SR and relative SPL over the recent object-centric learning methods, uORF and ObSuRF. In particular, NRC performs better than ObSuRF and uORF when navigating near furniture and other immovable objects (see supplemental for object navigation videos). We hypothesize this is due to NRC's more precise localization of objects.

2.5.4 Depth Ordering

Experimental Setup We evaluate NRC on the task of ordering objects based on their depth from the camera. Understanding the relative depth of objects requires both geometric and semantic understanding of a scene. For this task we evaluate on the ProcTHOR test dataset which provides dense depth and segmentation maps. Following the convention of [47], we determine ground truth depth of each object by computing the mean depth of all pixels associated with its ground truth segmentation mask.

For evaluation, we select pairs of objects in a scene and the goal is to predict which object is closer. We take the segmentation that has the largest IoU with the ground truth mask as the predicted object mask. To determine the predicted object depth, we compute the mean predicted depth of each pixel associated with the predicted object mask. All representations are trained on the ProcTHOR dataset and evaluated on the ProcTHOR test set. In total we evaluate 2,000 object pairs.

Results Depth ordering requires accurate segmentation and depth estimation. Due to NRC's stronger segmentation performance and better depth estimation, we see 5.5% and 10.3% depth ordering accuracy compared to ObSuRF and uORF respectively. The fine-grained localization of categorical latent codes in

Table 2.3: We compare depth ordering results on RoboTHOR with other geometrically-aware representations. Given pairs of objects in the scene, the model must infer which object is closer. We report the accuracy as the number of correct orderings over the total number of object pairs.

Method	Depth Order Acc. (%)
uORF [282]	13.5
ObSuRF [228]	18.3
NRC (Ours)	**23.8**

NRC allows for better depth ordering over existing object-centric methods. In particular, the other object-centric methods tend to assign object instances to the background which leads to large errors in estimating the depth of objects.

2.5.5 Ablation Study

Table 2.4: Ablation study for modeling the intra-code variation and learning the number of codes evaluated on unsupervised segmentation in ProcTHOR. Default fixed number of codes is set to 25.

Method + (Ablation)	ProcTHOR (ARI)
NRC	.182
NRC + Learned # of Codes	.197
NRC + Variation Module	.284
NRC + Variation Module + Learned # of Codes	**.295**

We present an ablation study on the ProcTHOR dataset to determine the effect of the variation module and learnable codebook size on unsupervised segmentation performance. Quantitative results can be found in Table 2.4. We observe that performance improves by $\sim 9\%$ when intra-class variation is explicitly modeled. Intuitively, allowing for small variation between instances of the same category should lead to better representations and allow for greater expressiveness.

We also find that learning the number of codes moderately improves performance. However, if number of codes is found via hyper-parameter tuning, performance is matched. Nonetheless, differentiably learning the codebook size avoids computationally expensive hyper-parameter tuning.

2.6 Limitations

Novel view reconstruction requires camera pose, which is not available for most images and videos. Some datasets such as Ego4D [65] provide data from inertial measurement units that can be used to approximate

camera pose, although this approach is prone to drift.

An incorrect assumption that NRC and most object-centric prior works make is that scenes are static. However, it rarely is the case that scenes are free of movement due to the physical dynamics of our world. Recently, Kipf et al. [105], Pumarola et al. [179] made strides in learning representations from dynamic scenes.

Although NRC is relatively efficient compared to the other NeRF based methods, the NeRF sampling procedure is compute and memory intensive. [204] and [221] leverage object-centric light fields to reduce memory and compute costs. The efficiency improvement from modeling scenes as light fields is orthogonal to NRC and can be combined.

A final challenge inherent to novel view reconstruction is finding appropriate corresponding frames of videos. For example, if two subsequent frames differ by a 60° rotation of the camera, then most of the scene in the subsequent frame will be completely new. Therefore, constructing the content in the novel view is ill-posed. Pairing frames with overlapping frustums is a potential solution, although the content of the scene may not be contained in the intersecting volume of the frustums.

2.7 Conclusion

Compositional, object-centric understanding of the world is a fundamental characteristic of human vision and such representations have the potential to enable high-level reasoning and efficient transfer to downstream tasks. Towards this goal, we presented Neural Radiance Field Codebooks (NRC), a new approach for learning geometry-aware, object-centric representations through novel view reconstruction. By jointly learning a shared dictionary of object codes through a differentiable renderer and explicitly localizing object codes within the scene, NRC finds reoccurring geometric and visual similarities to form objects categories. Through experiments, we show that NRC representations improve performance on object navigation and depth ordering compared to strong baselines by 3.1% success rate and 5.5% accuracy respectively. Additionally, we find our method is capable of scaling to complex scenes with more objects and greater diversity. NRC shows relative ARI improvement over baselines for unsupervised segmentation by 29.4% on ProcTHOR and 29.0% on NYU Depth. Qualitatively, NRC representations trained on synthetic data from ProcTHOR show reasonable transfer to real-world scenes from NYU Depth.

Chapter 3

Learning to Imagine the World from a Million 360° Videos

3.1 Overview

Three-dimensional (3D) understanding of objects and scenes play a key role in humans' ability to interact with the world and has been an active area of research in computer vision, graphics, and robotics. Large scale synthetic and object-centric 3D datasets have shown to be effective in training models that have 3D understanding of objects. However, applying a similar approach to real-world objects and scenes is difficult due to a lack of large-scale data. Videos are a potential source for real-world 3D data, but finding diverse yet corresponding views of the same content has shown to be difficult at scale. Furthermore, standard videos come with fixed viewpoints, determined at the time of capture. This restricts the ability to access scenes from a variety of more diverse and potentially useful perspectives. We argue that large scale 360° videos can address these limitations to provide: *scalable corresponding frames from diverse views*. In this paper, we introduce 360-1M, a 360° video dataset, and a process for efficiently finding corresponding frames from diverse viewpoints at scale. We train our diffusion-based model, ODIN, on 360-1M. Empowered by the largest real-world, multi-view dataset to date, ODIN is able to freely generate novel views of real-world scenes. Unlike previous methods, ODIN can move the camera through the environment, enabling the model to infer the geometry and layout of the scene. Additionally, we show improved performance on standard

novel view synthesis and 3D reconstruction benchmarks. See webpage and code.

3.2 Introduction

Humans have the ability to understand and reason about the 3D geometry of the world, which is key for everyday tasks such as navigation and object manipulation [56, 288, 277, 238]. In machine learning, 3D perception and reasoning has been a long-standing goal for researchers with broad applications in robotics [270, 110, 276], vision [159, 139], and graphics [129, 269]. Fueled by large-scale datasets of synthetic objects [39, 41], recent generative models have shown impressive understanding of 3D objects [135, 215, 176]. While these models' ability to generate synthetic objects is impressive, enabling 3D generative models for real world scenes and objects remains an open challenge.

One intuitive source for scalable data has been video as it implicitly contains rich information about the 3D world. However, learning 3D modeling from video has been elusive despite impressive effort [286, 204, 105, 205, 206]. The key problem has been how to consistently transform video into a form amenable to learning about the 3D world. Existing 3D models learn from multi-view data, a collection of images of scenes or objects and their respective camera pose. Creating such multi-view datasets from video requires finding sets of corresponding frames that capture similar parts of the scene but from different locations (Figure 3.2).

This search for corresponding frames in video has proven difficult at scale for a few reasons. First, correspondences are sparsely distributed throughout the video because the trajectory of the camera is fixed at the time of capture. Ideally, the camera operator would focus on a specific object or portion of the scene while moving around it. However, in-the-wild videos are far from this ideal. For example, if a person records themselves walking in the park, it is rare that they consistently focus the camera on the same object such as a bench as they walk towards, past, and away from it. Second, the computational cost of checking whether frames form a correspondence is expensive [211, 209, 18], therefore searching extensively is infeasible. Given these limitations, the largest real-world multi-view datasets to date [194, 283] utilize Amazon Mechanical Turkers to manually record video clips of objects, and are limited to 50 and 238 object categories respectively.

To address these limitations, we collect one million 360° videos from YouTube, introduce a process to efficiently transform 360° video into multi-view data, and train a diffusion-based novel-view synthesis (NVS) model on the dataset. Our model named ODIN, is the first to reasonably synthesize real-world 3D

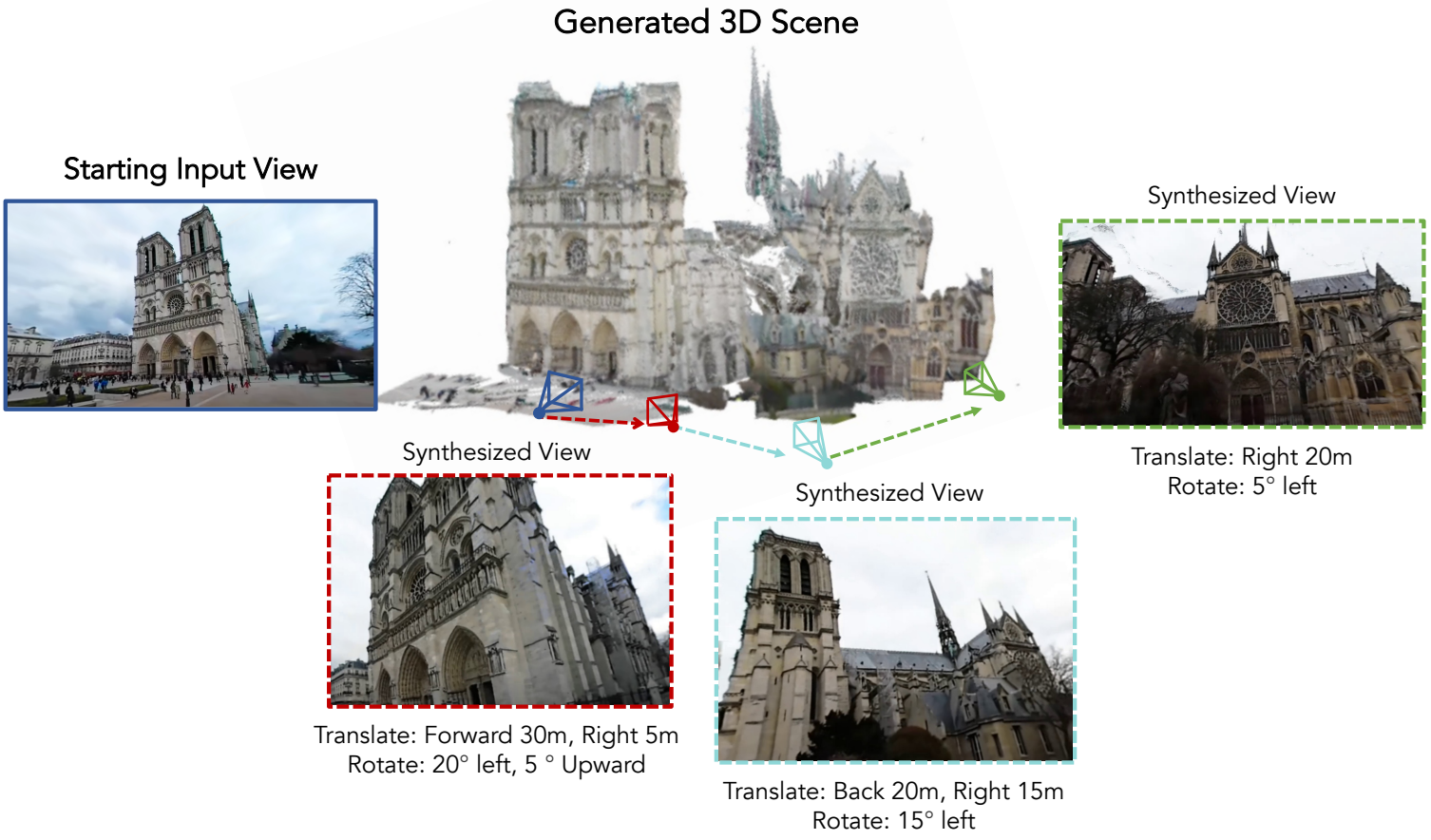


Figure 3.1: By learning from the largest real-world, multi-view dataset to date, our model ODIN, can synthesize novel views of rich scenes from a single input image with free camera movement throughout the scene. We can then reconstruct the 3D scene geometry from these geometrically consistent generations.

scenes and reconstruct their geometry *conditioned on a single image*. Quantitatively we evaluate our method on standard novel view synthesis benchmarks (DTU and MipNeRF360) and find improved performance compared to existing models without fine-tuning our own. Additionally, we compare ODIN to existing methods for 3D reconstruction on Google Scanned Objects as well as a held-out set of 360-1M and show significantly improved performance, especially on complex real-world scenes. We will open-source our model and dataset.

3.3 Related Work

Novel View Synthesis. NeRF [152] optimizes a volumetric scene function using sparse 2D images, representing the scene as a continuous 5D function. MipNeRF [8] extends NeRF with a multi-scale representation to enhance detail and reduce aliasing. Plenoctree [279] combines NeRF principles with an octree structure for efficient rendering. DIVeR [267] proposes a deterministic volumetric rendering for NeRF. Gaussian Splatting [102] uses Gaussian functions and splatting techniques for detailed scene representation and rendering. Unlike these methods which rely on densely sampled multi-view images and known camera poses, our approach captures extensive real-world scenes from widely varying camera views. PixelNeRF [280] and DietNeRF [88] extend NeRF to handle sparse input views but only for controlled settings.

Recent works leverage powerful generative (diffusion) models [197] for novel view synthesis of objects [176, 128, 257, 26], and more recently for scenes [26, 208, 34]. ZeroNVS uses a 3D-aware diffusion model with novel camera conditioning to generate 360-degree views from a single image, focusing on depth-scale ambiguity and background diversity with synthetic and real-world datasets. Diffusion with Forward Models [235] integrates a forward model into the diffusion process for unsupervised training on partial observations, solving inverse problems like view synthesis without direct signal supervision. ReconFusion [268] combines NeRFs with diffusion priors to enhance 3D reconstruction from limited views, improving geometry and texture plausibility with real and synthetic multi-view datasets. LucidDreamer [34] and RealmDreamer [216] use a multi-step pipeline involving point cloud guidance and Gaussian splats to generate detailed 3D scenes from text or image prompts but lacks physical realism and has limited control over viewpoint changes. In contrast, our method leverages a large-scale collection of 360-degree YouTube videos to train a diffusion-based model, enabling the synthesis of diverse real-world 3D scenes and reconstruction from a single image, thus accommodating significant camera view changes and a broader range of scenarios.

Camera Pose Estimation and Structure from Motion. Estimating camera pose and structure-from-motion (SfM) have a rich history in computer vision [3, 212, 243]. Camera pose estimation consists of estimating the 6 degrees of freedom of cameras from which images were taken and the camera intrinsics. The process typically involves finding corresponding key-points between multiple images of a scene, and using their apparent motion within the images to infer the 3D geometry of the scene and relative location of each camera.

For multi-view datasets and novel view synthesis, works typically use COLMAP [211, 209] or a SLAM variant [234, 161]. We choose to use the recent method Dust3R [255] as it is computationally faster and allows for as few as 2 images whereas most SfM methods require dozens. This enables us to scan much more quickly through videos for frame correspondences and create the large-scale dataset from 360° video.

Multi-View Datasets. Existing multi-view datasets such as MVImageNet [283], CO3D [194], RealEstate10K [290], ACID [133], Epic-Kitchens [36], MipNeRF-360 [9], and Epic-Fields [241] provide valuable multi-view sequences of real-world scenes and objects but are often constrained by the specific environments or objects they capture. MVImageNet is the largest multi-view dataset to date with over 200,000 video clips captured of 238 object categories. Though this effort is impressive, using Mechanical Turkers to manually capture videos of objects is difficult to scale further and limits content diversity. Large-scale 3D object datasets like Objaverse [39], Objaverse-XL [41], and infinigen [183] focus on detailed 3D object assets to generate synthetic objects and scenes. Autonomous driving and 3D reconstruction datasets such as Kitti [61], DTU [2], ShapeNet [27], and Google Scanned Objects [45] offer multi-view data for specific tasks like driving scenarios, 3D modeling, and object classification.

In contrast, our dataset leverages a large-scale collection of 360-degree YouTube videos, providing a vastly more diverse and extensive source of real-world data. Our dataset accommodates significant camera view changes and broader real-world applications, going beyond the constraints of controlled multi-view datasets and specific domain focuses.

3.4 Multi-View Data from 360° Video

There are two key elements missing from current multi-view datasets: *scale* and *real-world* data. Various datasets and works have managed [194, 283, 39, 41] to make progress along these dimensions individually, however, no current datasets afford both aspects.

The key challenge in collecting large-scale, multi-view datasets derives from the difficulty of finding high-quality frame correspondences, and estimating their relative poses. Existing structure-from-motion algorithms, such as COLMAP [210] and HLOC [209], are slow and require many images of the same scene. In this section we detail our process for efficiently transforming 360° video into high-quality multi-view data.

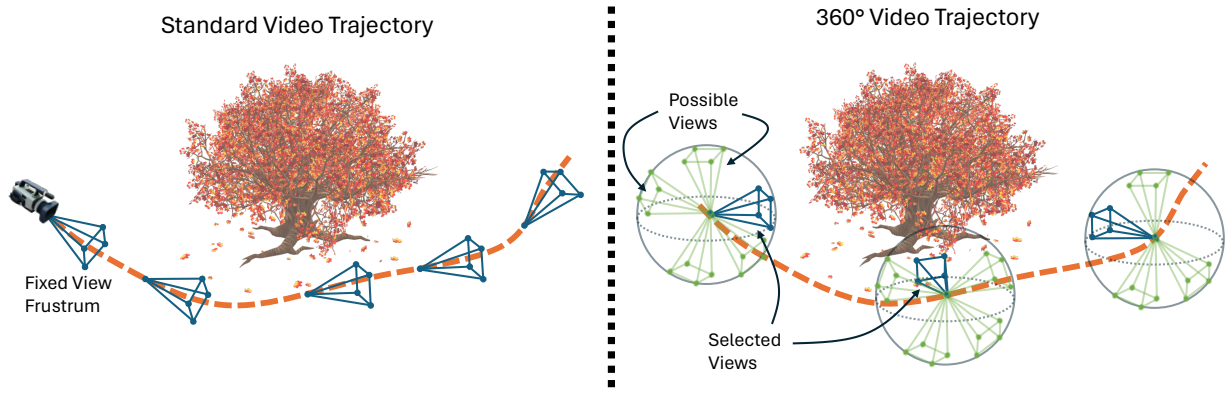


Figure 3.2: Left: An illustrative trajectory of standard video with the view point fixed at the time of capture. The fixed view point makes finding corresponding frames challenging. Right: The trajectory of a 360° video through the scene. The controllable camera enables alignment of views at different frames of the video.

3.4.1 Scalable Correspondence Search

There are two properties of corresponding video frames that are necessary for training novel view synthesis models: *sufficiently differing viewpoints* and *overlapping content*. In manually collected novel view synthesis (NVS) datasets this is accomplished by taking a video while circling the object. Finding frames that fit these criteria from in-the-wild video is much more difficult.

A major reason is that high-quality correspondences are sparsely distributed in standard videos. For example, someone taking a video while walking down the street often keeps their camera view facing their direction of travel. So while they may capture a parked car on the side of the road while walking towards it, they likely will not pan their camera to capture it from many angles while walking past or away from it. Therefore, it is difficult to obtain paired images of the scenes or objects from distant locations and diverse views at scale. One solution to this problem is to leverage 360° videos. The 360° nature allows the views of frames to be rotated such that they contain overlapping content. Therefore given two frames that are close enough in spatial location, in theory we can align the views to look at the various regions of the scene to form multiple view correspondences.

Now we describe how we operationalize this approach. We begin by sub-sampling frames of the 360° video at $r = 1$ frame-per-second. We find empirically this to be a sufficiently fast frame rate given the movement speed of the camera. The computation of the correspondence search scales with r^2, therefore we judicially select the frame rate. Next we perform pairwise comparison between frames within a frame

window of length, $L = 20$. We map the 360 panoramic frames using an equirectangular projection $E(I, \theta, \phi)$ where θ is the pitch, ϕ is the yaw, and I is the image. We map the panoramic image to four different views $E(I, j * \pi/2, 0)$ for $j \in \{1, \ldots, 4\}$. Thus a panoramic frame, F_t at time t produces four frames $\{F_{t,0}, F_{t,\pi/2}, \ldots, F_{t,3*\pi/2}\}$. We then pass all pairs within the time window to the Dust3r model [255] which outputs relative pose estimate, P and confidence map, C. We take the mean confidence over the spatial components of the confidence map with height h and width w, $\mu_c = \frac{1}{hw} \sum_{c \in C} c$, and filter out frames below threshold, $\tau = 4$. A higher mean confidence means that the frames must be overlapping as the model can accurately estimate the pose.

Once the correspondences have been found we refine the relative pose between them by performing gradient descent on the pitch and yaw of both equirectangular projections with respect to μ_c. Intuitively, we can think of this as rotating the cameras to maximize the overlap (Figure 3.2). After all correspondences have been found, we discard pairs with relative translation less than .25 m because they provide minimal information for training the model.

3.4.2 Correspondence Propagation

Computing relative pose between frames, especially for video, has been computationally prohibitive and a major bottle-neck for large-scale multi-view datasets [36, 194, 283]. An exhaustive search between all frames of a video would incur a cost of s^2r^2 where s is the number of seconds, and r is the frame rate. A common approach is to limit the search with a window of size L to reduce the cost to L^2, however this limits the pairs to short-range correspondences.

We propose a hybrid approach that enables finding long-range correspondences with limited additional compute. After the initial frames have been found as detailed in section 3.4.1, we create a graph in which the nodes are frames and an edge exists if two frames have correspondence. We then perform the same procedure outlined in section 3.1 for all connected frames in each sub-graph. Intuitively, if two frames share a corresponding third frame (connected in the graph)then the two are also likely to share a correspondence. This approach allows us to maintain a small search window, while still finding long-range correspondences.

3.4.3 Resolving Scale Ambiguity

Dust3R and other structure-from-motion methods [162, 210, 184] output relative camera poses in dimensionless quantities, therefore we need to calibrate them to a universal scale. We do so by fusing the depth map estimates, $\hat{D}$, from an off-the-shelf depth estimator [275], with the point map, $X \in \mathbb{R}^{h \times w \times 3}$, predicted by Dust3R. A pointmap is a correspondence between each pixel (i, j) and the point in 3D where the ray from pixel (i, j) intersects the scene. We anchor the dimensionless pointmap to the depth math D by optimizing for a scale factor, σ, in the following equation:

$$\arg\min_{\sigma} \sum_{i=1}^{h} \sum_{j=1}^{w} |C_{ij}(\sigma z_{ij} - D_{ij})|, \tag{3.1}$$

where z_{ij} is the depth component of X_{ij} and C_{ij} is the confidence map output by Dust3R. C_{ij} is close to 0 for points which the model has high uncertainty and acts as a filter for points with poor estimates. We choose L1 distance to limit the effect of outliers. Once we recover this scale factor, we multiply the translation of the estimated camera pose (R, t) by σ to obtain the metric pose estimate.

3.5 Dataset Collection and Statistics

To leverage the proposed scalable correspondence search (Section 3.4) for generating a large-scale multi-view dataset, we collect the largest 360° video dataset to date, 360-1M, consisting of over 1 million 360° videos. In this section, we describe the collection process and statistics for the dataset.

3.5.1 Collecting 360° Video

We collect all meta-data from YouTube in order to filter 360° videos. The meta-data provides information on duration, view count, format, and subject category among other fields. We filter for the equirectangular format which indicates 360° video and results in 1,076,592 total videos. We then download the videos in the equirectangular format at the best quality available. We will release the meta-data for the 360° videos alongside the dataset.

We filter the downloaded videos for empty, and duplicate videos. We remove duplicated videos with a deduplication model [90] run on the thumbnails of the videos. This does not guarantee the contents of the

video are unique, however running over all frames is computationally infeasible.

3.5.2 Dataset Statistics

360-1M consists of 80,567,325 unique frames extracted from 1,076,592 videos with an average of 74.83 unique frames per video. The average video length is 6.3 minutes and is distributed in a long-tail fashion. When searching for correspondences, we sample the videos at 1 FPS. The videos are distributed evenly across 15 subject categories, with the most popular category being Travel and Events (149,534 videos) and the least popular being Pets and Animals (8802 videos). We find 363,417,730 total frame correspondences along with their relative camera poses.

3.6 Method

Our final goal is to generate images along a viewpoint trajectory conditioned on a single image of a scene – a task known as novel view synthesis (NVS). Note that our task differs from tradition novel view synthesis work such as [152] which aim to generate novel views after training on many images of a single scene. Similar to prior works [208, 135], we leverage a diffusion-based model. This class of models have shown impressive capabilities in learning priors from large-scale data. An alternative is a NeRF based approach which is mainly effective in small-scale settings.

3.6.1 Viewpoint Conditioned Diffusion

Given a single image, $x \in \mathbb{R}^{h \times w \times 3}$, of a scene, our objective is to generate a sequence of images, $\hat{x_i}$ from different viewpoints, (R, t) where R is the relative rotation and t is the translation between views. Following [135], we use a latent diffusion architecture which consists of an encoder $\mathcal{E}$, a denoiser U-net f_θ, and decoder $\mathcal{D}$. The standard diffusion training objective is:

$$\min_{\theta} \mathbb{E}_{z \sim \mathcal{E}(x), t, \epsilon \sim \mathcal{N}(0,1)} \|(\epsilon - f_\theta(z_t, t, f(x, R, t)))\|_2^2$$

Our modeling objective differs from previous work in that we condition on both rotation, R, and translation t. The long-range correspondences in our training data afford much freer camera movement

Figure 3.3: Qualitative comparison of novel view synthesis on real-world scenes. The left and right images are conditioned on camera views from the left and right respectively. In the middle scene of the kitchen, ODIN accurately models the geometry of the table counter and chairs as well as unseen parts of the scene such as the living room.

throughout the scene compared to previous works. Due to the limitations of previous training data, other methods can only rotate about a center point of the object or scene.

3.6.2 Motion Masking

Learning how to perform novel view synthesis from videos poses a challenge as it assumes the scene itself does not vary with time when generating images from novel viewpoints. Previous approaches have addressed this challenge by training solely on videos of static scenes such as only indoor houses [290] or manually filtering videos [194, 283]. However, such approaches limit the diversity and scale of the data. Therefore, to learn from in-the-wild videos, we propose *motion masking*, an approach for handling dynamic objects.

Motion masking consists of predicting a dense mask of values between 0 and 1, which we apply to the output by the U-net f_θ through elementwise multiplication. This soft mask allows the model to filter out portions of the scene which may be difficult to predict due to object movement. To produce the motion mask we add an additional channel to the U-Net denoiser, which outputs a dense mask with values which we clamp between 0 and 1. During training, this mask filters dynamic elements from the loss function.

Formally, let $M \in \mathbb{R}^{h \times w}$ denote the dense mask generated by the decoder. The modified loss function,

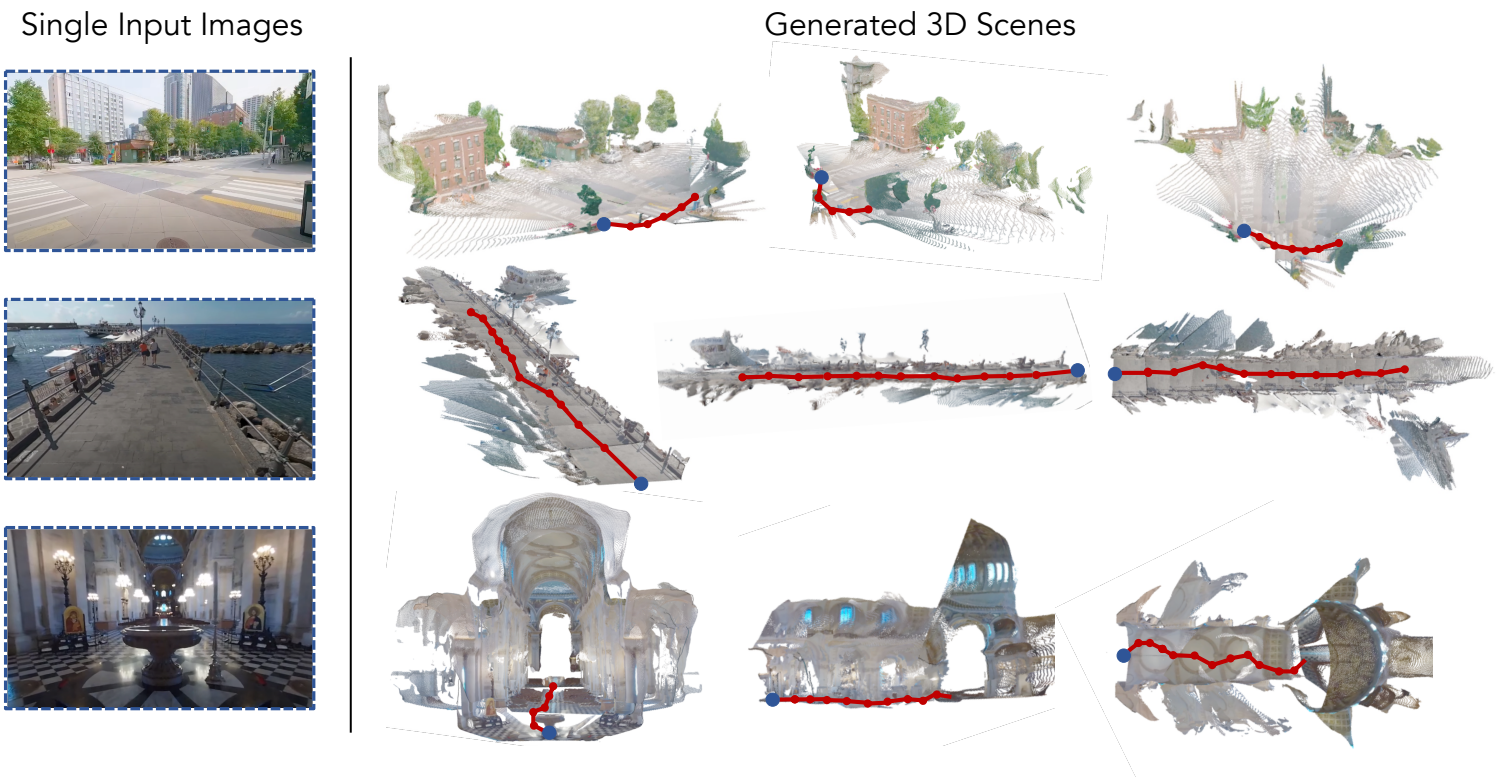


Figure 3.4: Examples of generated 3D scenes using ODIN. The blue dot indicates the location of the input image and the red lines indicate the trajectory of the camera which generated the images. ODIN is capable of long-range generation of geometrically consistent images. In the bottom scene, we see the model accurately infers the geometry of the unseen cathedral ceiling and the long hallway.

incorporating temporal masking, is given by:

$$\mathcal{L} = \|(\epsilon - \epsilon_\theta(z_t, t, f_\theta(x, R, t))) \cdot M\|_2^2 \tag{3.2}$$

However, directly optimizing this loss leads to a degenerate solution where all elements of the scene are filtered from the loss. To address this, we introduce an auxiliary loss term that incentivizes the mask to be non-zero:

$$\mathcal{L}_{\text{auxiliary}} = -\lambda \sum_{i,j} M_{ij} \tag{3.3}$$

Incorporating motion masking and the auxiliary loss enables the model to focus on static elements in dynamic scenes while training for novel view synthesis.

3.6.3 3D Reconstruction

Our model is trained to output a single image given an input image and target view, a popular approach which provides flexibility in the type of data that can be trained on, while still allowing for 3D reconstruction and multi-view generation. This flexibility is particularly crucial for training from video data, where obtaining a full collection of frames for a given scene from in-the-wild videos may not be possible.

Naively, the image to image paradigm has the drawback that generating multiple views does not guarantee consistency across views. To address this, we follow the approach of previous works [208, 176, 135] which employ various techniques to induce consistency across multiple generations. We adopt a trajectory-based sampling approach similar to [208] where images are sampled along a smooth trajectory, though in our case we are not restricted to simple rotations. While sampling, subsequent generations are conditioned on the previous generation, $\epsilon_{i,t} = f_\theta(x_i, R, t)$. Once multiple views are generated we reconstruct the scene using Dust3r [255].

3.7 Experiments

In this section we benchmark our model, ODIN, against existing methods for novel view synthesis and 3D reconstruction. We improve performance on standard benchmarks which consist of relatively simple scenes with minimal camera translation, all without fine-tuning on the target task. Qualitatively we find that ODIN has new capabilities in generating real-world scenes from long-range novel views.

3.7.1 Experimental Setup

We evaluate our model on the standard novel view synthesis (NVS) benchmarks, DTU [2], and Mip-NeRF 360 [9]. DTU consists of table-top items and Mip-NeRF 360 consists of scenes with views rotated 360° around a point. We report the standard NVS metrics, PSNR, LPIPS, and SSIM. As noted by previous literature, PSNR and SSIM are not well correlated with human evaluation so we primarily focus on LPIPS and qualitative comparison. Furthermore, to showcase the novel capabilities of our model, we evaluate our method on a held-out set of 360-1M constructed from one-thousand 360° videos.

For 3D reconstruction we compare with Zero1-to-3 [135], MCC, SJC-I, and Point-E on Google Scanned

Objects (GSO) and ZeroNVS on our held-out set of 360-1M. For 360-1M we derive the pseudo-ground truth from a Dust3R model which is trained on all ground truth views of the scene given by the video. We report Chamfer-Distance for 360-1M in addition to volumetric IoU for GSO. The 3D reconstructions for our model are created by generating images along trajectories then using Dust3r to reconstruct the scene.

Table 3.1: Comparison with other novel view synthesis models on the DTU benchmark which consists of single objects placed on table tops.

NVS	LPIPS ↓	PSNR ↑	SSIM ↑
PixelNeRF [280]	0.535	15.55	0.537
SinNeRF [272]	0.525	16.52	**0.560**
DietNeRF [88]	0.487	14.24	0.481
NeRDi [42]	0.421	14.47	0.465
ZeroNVS [208]	0.380	13.55	0.469
ODIN (Ours)	**0.378**	**16.67**	0.525

Table 3.2: Comparison of various novel view synthesis models on the MipNeRF 360 benchmark [208, 9]. As noted by previous work [208], PSNR and SSIM are unreliable metrics for novel view synthesis so we focus on LPIPS.

NVS	LPIPS ↓	PSNR ↑	SSIM ↑
PixelNeRF [280]	0.718	16.50	**0.556**
Zero-1-to-3 [135]	0.667	11.70	0.196
ZeroNVS [208]	0.625	13.20	0.240
ODIN (Ours)	**0.587**	**16.84**	0.537

The models we benchmark against are trained on a variety of 2D and multi-view data sources. The diffusion-based methods, Zero1-to-3 [135] and ZeroNVS [208] start from a StableDiffusion pretrained model. Zero1-to-3 [135] fine-tunes on Objaverse [39], while ZeroNVS [208] fine-tunes on Co3D [194], ACID [133], and Real-Estate10k [290]. When possible we evaluate the models provided by the original works. Most closely related to our work in architecture is Zero1-to-3 [135] with the key difference being our addition of motion masking for training on video.

3.7.2 Novel View Synthesis

We observe improved performance on DTU and Mip-NeRF 360 on the standard NVS metrics (Tables 3.1 and 3.2). Our improvement on DTU is relatively small which is to be expected as the dataset consists of simple objects, with black backgrounds. On Mip-NeRF 360, which consists of real-world scenes, we see significant improvement. In particular, the other methods struggle to generate reasonable images from views that differ significantly from the input view. In Figure 3.3 we compare qualitatively to other recent works. We observe that Zero1-to-3 cannot generate full scenes and struggles to generate real objects as expected due its training data. ZeroNVS generates more plausible views, but is still considerably worse for more complex scenes.

3.7.3 3D Reconstruction

We present 3D scenes reconstructed from ODIN generated images along a trajectory (Figure 3.4). Quantitative comparison can be found in Table 3.3. For Google Scanned Objects [45] our method is comparable to Zero-1-to-3 [135] and outperforms other methods. Comparable performance to Zero-1-to-3 is expected as it was designed for synthetic objects. We compare with ZeroNVS for scene reconstruction on a held-out set of 360-1M. Other methods are not capable of generating scenes therefore we only benchmark against this method.

Table 3.3: 3D reconstruction results on Google Scanned Objects [45].

Method	MCC [266]	SJC-I [254]	Point-E [164]	Zero-1-to-3 [135]	ODIN (Ours)
Chamfer Distance ↓	0.1230	0.2245	0.0804	0.0717	**0.0697**
IoU ↑	0.2343	0.1332	0.2944	0.5052	**0.5328**

3.8 Limitations and Broader Impact

The framework and method we presented in this work are a promising step towards large-scale 3D models, however there are some limitations to our approach. From a modeling perspective, the motion mask allows us to filter portions of scenes which have dynamic elements, however ideally we would like to learn to model the dynamic elements as well. Some progress has been made on 4D NeRF models which can move the camera view in both time and space, however generalized 4D models are largely unexplored.

Our work may have positive societal impact in the creation of 3D assets for AR and VR or downstream applications such as robotic navigation. From a negative perspective our work could be used to create fake images or inappropriate scenes.

3.9 Conclusion

In this work, we propose a scalable approach to constructing real-world multi-view data and show the merits of our model, ODIN, trained on the largest multi-view dataset, 360-1M, to date. Enabled by the scale, diversity, and long-range correspondences in 360-1M, ODIN demonstrates capabilities beyond those of

previous methods in generating 3D-consistent novel views of real-world scenes with free camera movement. On novel view synthesis and 3D reconstruction benchmarks ODIN outperforms existing methods without fine-tuning to the target data. While ODIN shows impressive results, we believe that there is further potential in the use of 360-1M and 360° video for novel view synthesis as well as other domains such as video generation. For novel view synthesis an exciting next step would be to model dynamics to generate 4D scenes. We will open-source our code, models, and dataset.

Chapter 4

Neural Priming for Sample-Efficient Adaptation

4.1 Overview

We propose Neural Priming, a technique for adapting large pretrained models to distribution shifts and downstream tasks given few or no labeled examples. Presented with class names or unlabeled test samples, Neural Priming enables the model to recall and conditions its parameters on relevant data seen throughout pretraining, thereby priming it for the test distribution. Neural Priming can be performed at inference, even for pretraining datasets as large as LAION-2B. Performing lightweight updates on the recalled data significantly improves accuracy across a variety of distribution shift and transfer learning benchmarks. Concretely, in the zero-shot setting, we see a 2.45% improvement in accuracy on ImageNet and 3.81% accuracy improvement on average across standard transfer learning benchmarks. Further, using Neural Priming at inference to adapt to distribution shift, we see a 1.41% accuracy improvement on ImageNetV2. These results demonstrate the effectiveness of Neural Priming in addressing the challenge of limited labeled data and changing distributions. Code is available at https://github.com/RAIVNLab/neural-priming.

4.2 Introduction

Humans have a vast store of prior experience which we draw on to flexibly perform a diverse range of tasks [81, 20, 19, 57]. While engaging in an activity, we naturally retrieve relevant information or schema in a cognitive phenomena known as Priming [151]. This process ensures that necessary knowledge is readily accessible in memory, leading to enhanced performance for the task at hand [196]. Pre-trained, general-purpose models such as CLIP [182] and ALIGN [94] have extensive prior knowledge learned from large-scale, diverse datasets. These datasets seek to capture all natural variation in real data within their distribution. Can these models also benefit from something like priming? We observe that models trained even on the largest of such datasets often substantially improve in performance when fine-tuned on task-specific data. This begs the question of what the model learns from fine-tuning on the target dataset, if it already trained on many similar examples during pre-training.

We speculate that the effect of fine-tuning a pre-trained model on task-specific data is similar to that of priming. Given the sheer size and diversity of the pre-training dataset it becomes challenging for the model to find a consistent solution that is optimal for all subsets of the data. This becomes particularly evident for open-vocabulary models such as CLIP, where multiple natural language descriptions can correspond to a single image, highlighting the challenge of accommodating diverse interpretations. We hypothesize that training on the downstream dataset re-aligns the model to the specific objective.

With this in consideration, we propose Neural Priming. Specifically, Neural Priming recalls a subset of the pre-training data similar to the target distribution, re-aligns the natural language descriptions to the downstream task, and quickly adapts the model to the subset. We perform extensive experiments on 7 transfer learning and 4 distribution shift datasets to validate our method. We use the OpenCLIP [182, 265] ViT [44] set of models pre-trained on the LAION-2B and 400M [213]. We find Neural Priming leads to significant accuracy improvements, particularly when labeled data is scarce and in specialized domains. Concretely, Neural Priming improves accuracy by 2.45% on ImageNet and 4.25% on average across the other 6 datasets over the base CLIP model. In the few-shot setting, Neural Priming improves accuracy by 3.81% on average over recent methods on standard transfer learning benchmarks. We show Neural Priming is efficient and can be performed on-the-fly. For datasets containing more than 2 billion images, we can prime our model to ImageNet in less than 2 minutes with a single commercial GPU.

Neural Priming is flexible and can be used with variable degrees of information about the downstream distribution. When the model has language-only task descriptions, our approach can efficiently retrieve a *priming pool* of relevant examples from the pre-training set and attune the model to this data. At inference time, given a set of test images to classify, Neural Priming is able to use test examples to adapt the priming pool to distribution shifts. When we have access to training examples in the few-shot setting, Neural Priming can filter the priming pool to align with the training distribution.

We make the following contributions:

- We introduce Neural Priming, a novel method that leverages retrieval from the pre-training dataset for efficient and accurate adaptation of large, pre-trained models to downstream tasks.
- Neural Priming achieves state-of-the-art zero-shot and few-shot accuracy on the standard transfer learning datasets – up to 4.25% and 3.81% improvements respectively over baselines (Section 4.5.2).
- Neural Priming also enables transductive learning and improves performance on standard distribution shift datasets by 2.51% on average, all without using any additional data (Section 4.5.3).
- Our approach generalizes to various architectures and pre-training datasets while being complementary to techniques [150, 178] that improve zero-shot performance of open-vocabulary models.

4.3 Related Work

4.3.1 Open-Vocabulary Models and Zero-shot Inference

Open-vocabulary models have proven to be an effective approach for transfer learning. Such models enable training on vast amounts of web-scale images without the need for labor-intensive human labeling by leveraging pre-existing natural language descriptions [182]. Open-vocabulary models have set state-of-the-art on ImageNet [43] as well other transfer learning benchmarks [271, 7, 192].

Open-vocabulary models offer additional capabilities beyond standard pre-trained models. They can perform zero-shot inference, where predictions are made without training on target data. Additionally, they are robust to distribution shifts [182], enable prompt-tuning methods [178, 150], and can be used for text-based retrieval [53]. Zero-shot can have different meanings in the literature. In the context of this paper,

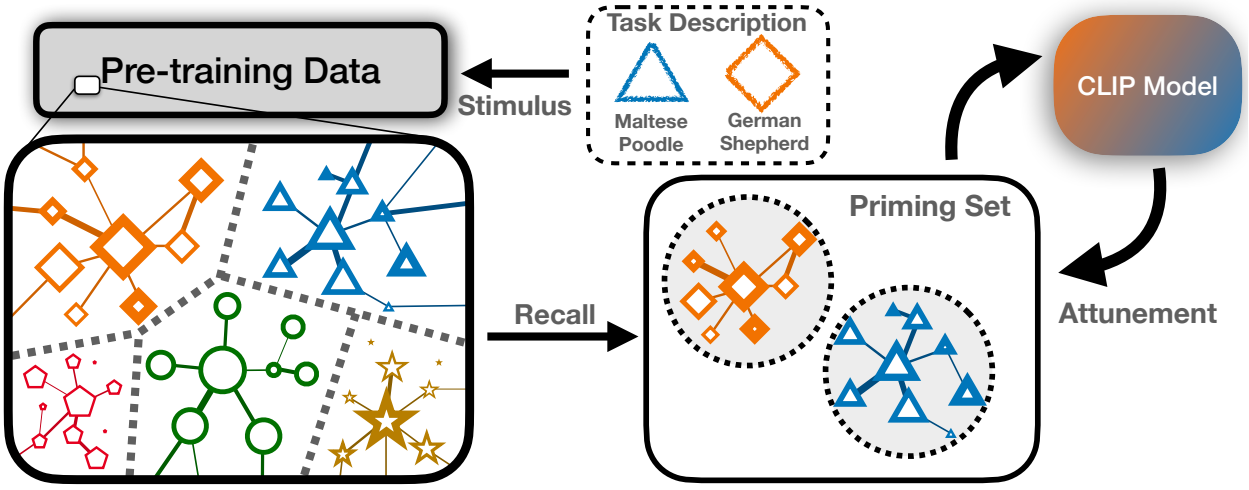


Figure 4.1: A diagram of Neural Priming**, our proposed method.** Neural Priming is a framework for leveraging an open-vocabulary model's *own pre-training data* to improve performance on downstream tasks. Neural Priming encompasses two processes: **1.** Collecting a *priming pool* of relevant examples from the pre-training set to prime with and **2.** using these examples to attune our model to a given task. We show performance improvements across a wide range of transfer learning and robustness benchmarks.

we consider zero-shot as the experimental setting in which the model receives no training examples drawn from the training distribution.

Prompt-tuning has emerged as a popular research direction in the domains of large language and open-vocabulary models. In the context of open-vocabulary models, prompt-tuning can involve modifying the textual prompts or queries used during the training or inference of the model to improve its understanding of visual content or achieve specific goals. In the original CLIP paper, Radford et al. [182] design hand-crafted prompt templates for ImageNet and other transfer learning datasets and show that this leads to substantial accuracy improvements. More recently, other work [150, 289] has used machine learning approaches to learn the prompts rather than hand-crafting them.

4.3.2 Distribution Shifts

Robustness to distribution shift is a key property of good machine learning models as it represents a notion of reliability. In particular, studies on natural distribution shifts, including ImageNet-V2 [192], ImageNet-Sketch [253], ImageNet-R [79], and ImageNet-A [80], find that models have a consistent performance drop when exposed to a distribution at inference time not seen during train time [233]. In order to focus on

robustness and eliminate the confounder of better models being generally better, this performance gap is measured through effective robustness, which is the robustness improvement over ImageNet trained models. Prior work has shown that the performance of models on in distribution and out of distribution is highly correlated across many algorithmic training interventions, except for cases where training on larger and more diverse datasets increases robustness [155].

The most significant recent improvement in effective robustness [192] is the introduction of open-vocabulary models. At its time of release, CLIP [182] achieved unprecedented effective robustness on a variety of distribution shifts. Studies have suggested that these models achieve high effective robustness through their data distribution [52], a result of training on large amounts of web-scraped data. However, these models are still worse at downstream tasks than models fine-tuned on in-distribution data. Moreover, fine-tuning on downstream data causes robustness on other data distributions to deteriorate [182, 265]. Many mitigation methods have been proposed to such as Wise-FT, FLYP, LP-FT, and model surgery [265, 64, 116, 125]. Our paper differs from these methods in goal: whereas they seek to keep model robustness while gaining the benefits of fine-tuning on task-specific data, we seek the benefits of fine-tuning while *not collecting any in-distribution data*. Hence these methods are complementary to Neural Priming, and we employ Wise-FT in our model attunement procedure.

4.3.3 Transductive Learning

Transductive learning [58, 29] focuses on leveraging unlabeled data during inference. It differs from traditional supervised learning, which solely relies on labeled data at train time. Related to transductive learning is test-time training [231, 59, 217]. Test-time training involves adapting and refining the model's predictions based on the specific testing examples encountered. Transductive learning differs from test-time training in that test-time training only considers one test sample at a time, whereas transductive aims to learn from the entire test set.

4.3.4 Few-Shot Learning

Few-shot learning research aims to addresses the challenge of learning from a limited number of labeled examples. In many real-world scenarios, acquiring large labeled datasets is impractical or costly. Older

lines of work have focused on meta-training small models [222, 55, 169, 91] on small-scale datasets. More recently, the approach for few-shot learning has shifted towards training large, general-purpose models such as CLIP [182] and ALIGN [94] on web-scale datasets.

4.3.5 Retrieval-Augmented Models

In language, works have demonstrated the effectiveness of retrieval from text corpora or structured data for tasks such as question answering [17, 69, 104]. In general, these methods seek to recover facts either from a large corpus or knowledge graph, then use those to complete tasks. This differs from our scenario, where exact examples at inference time do not necessarily exist in the pre-training corpus. REACT [134] and SuS-X [242] are retrieval-augmented methods for open-vocabulary models which use search to fine-tune with relevant examples [134]. We differ from Liu et al. [134] in that they add a substantial number of new parameters whereas we do not. Additionally, our approach is significantly more efficient, both computationally and in terms of number of samples, enabling use at inference for additional improvement (Section 4.4.1). We differ from [242] in that their work uses semantic retrieval whereas Neural Priming leverages language for fast initial filtering and image search for accurate retrieval. Further, Neural Priming shows that models can improve by revisiting examples seen throughout pretraining whereas other works retrieve new examples from external datasets.

4.4 Method

Neural Priming is the process of retrieving relevant information from the pre-training dataset and leveraging it for a specific task. We study it in the context of vision-language contrastive pre-training, so the form our task description takes is a set of class names, $\mathcal{C}$, already in natural language. A CLIP model [182] consists of a vision embedding model, V, and a language embedding model, L, each producing a vector representation in $\mathbb{R}^d$. The pre-training dataset, $\mathcal{D}$, consists of a large number of image-text pairs collected from the web. The text component can be noisy, potentially containing irrelevant or inaccurate information about the image content.

We break our method down into two main steps: **1.** Collecting the priming pool, where we gather data from our pre-training dataset relevant to a particular task and **2.** model attunement, where we leverage this

data to improve our model.

4.4.1 Collecting the Priming Pool

Leveraging Natural Language Task Information

The goal of this step is to collect an initial pool of images relevant to the task at hand given the previously defined natural language description $\mathcal{C}$. For example, if our task is a set of dog breeds, ideally we would collect sets of images belonging to those breeds and label them accordingly. A simple way to prime is by using retrieval to gather relevant data points from our pre-training dataset. An existing method for language-based retrieval involves using the CLIP text embedding of a class description $c \in \mathcal{C}$ for retrieval using semantic similarity scores on the pre-training set [12, 193]. However, with neural priming, prioritizing precision over recall is crucial, considering the size, diversity, and noise of the pre-training dataset. This form of semantic retrieval has a major downside: it is not clear where to threshold similarity scores to retrieve the most relevant images. Threshold too late and we allow unrelated images to be included in our pool. Further, this threshold is often specific to a category, making it infeasible to search at scale.

Our approach to language-based priming is to search for the existence of the class name, $c \in \mathcal{C}$, in the captions of our pre-training dataset to retrieve images relevant to a particular category. We organize these image-text pairs into separate categorical clusters $\{B_c\}$ according to the class name c mentioned in their captions. This approach has a few advantages over semantic retrieval: **1.** After setting up an inverted index search structure for text retrieval [108, 219], exact string matching is far faster than semantic retrieval, even when approximate nearest neighbor strategies are employed [97], **2.** with exact string search, the category boundary is clear and therefore does not require per category tuning, **3.** the retrieval results are overall qualitatively more relevant. Finally, to leverage the semantic understanding of our CLIP model, we filter the priming pool using CLIP similarity score. We do this by constructing a "zero-shot" CLIP classifier, as defined by Radford et al. [182], and removing examples from categorical clusters that do not align with their label according to the CLIP model.

Leveraging Image Information at Test Time

At inference time, the model can narrow the relevant priming pool even further by utilizing information about the test distribution. To do this, given an image x in our test set, we compute the cosine similarity using our CLIP image encoder V, $\cos(V(x), V(y))$ for every $y \in P$, our priming pool. We retrieve examples with the top-k cosine similarity scores ($k = 10$ in most of our experiments). We do this collectively for every image in the test set and collect the retrievals to form a filtered priming pool. If an example is retrieved twice, we de-duplicate them in the final priming pool. Since we do this for all images in the test set, we consider this the *transductive setting* to align with prior work [58, 29].

4.4.2 Attuning CLIP to the Priming Pool

The goal of this step is to modify our CLIP model to take advantage of the data in the priming pool P. We first construct the task-specific zero-shot linear head $W_z \in \mathbb{R}^{d \times n}$ using the text encoder and the natural language names of each class, where d is the feature dimension and n is the number of classes. To get logits for a particular example, x, we compute $W_z \cdot L(x)$, so our prediction is $\arg\max_c W_z \cdot L(x)$.

To attune our CLIP model to the priming pool, we perform nearest-class mean (NCM) [147] on all retrieved examples to obtain a classification head from the priming pool. Namely for a given class c, we compute a centroid $\tilde{y}_c = \frac{1}{|B_c|} \sum_{x \in B_c} L(x)$ and then normalize this centroid to produce a class embedding $y_c = \tilde{y}_c / \|y_c\|$. We define the collection of centroids as matrix $W_{ft} = [y_c]_c \in \mathbb{R}^{d \times n}$. To expand this to few-shot scenarios, we mix the labeled data into the corresponding categorical clusters before performing NCM. Finally, we ensemble W_z and W_{ft} using a mixing coefficient $\alpha \in [0, 1]$ as $W_\alpha = (1-\alpha) \cdot W_{ft} + \alpha \cdot W_z$, which is our final classification head. We choose alpha according to a heuristic $\alpha = e^{-|P|/\sigma^2}$ which can be derived from a Bayesian prior over the text features. We also find that α can be effectively chosen through cross-validation on a held-out portion of the retrieval set. Intuitively, if we do not have much data in our priming pool, we want it to influence our model less. We use NCM as the classifier as it has shown to be sample-efficient [222].

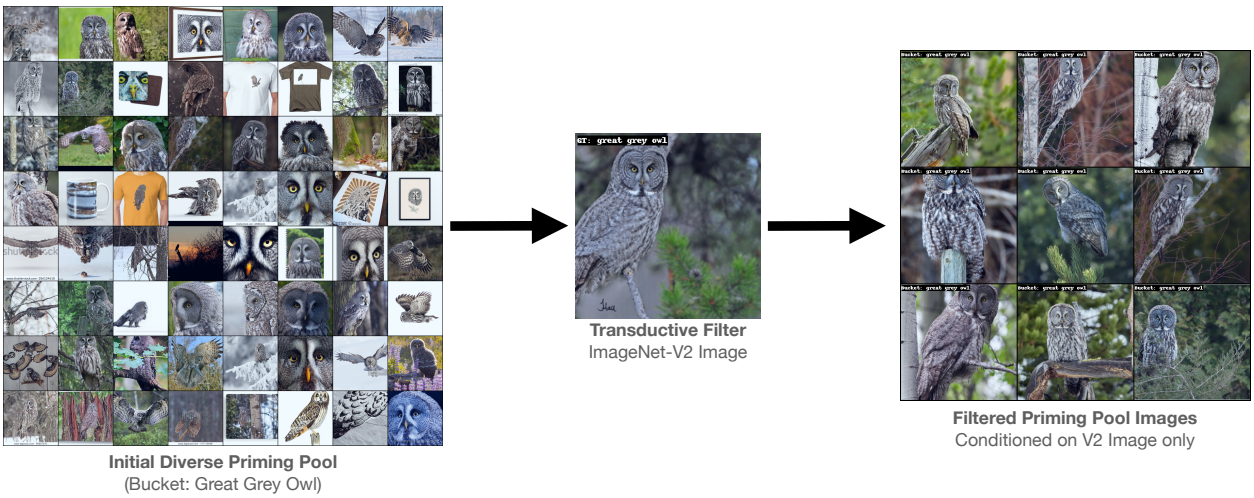


Figure 4.2: A qualitative example of our approach for transductive image filtering. Given an initial *priming pool*, acquired through natural-language text search on the captions of our pre-training dataset (Section 4.4.1), we filter out irrelevant examples using images from our test set. **(left)** we show examples from the great owl categorical cluster of our priming pool before filtering, **(center)** we show an example image from the same category of ImageNet-V2, **(right)** example retrievals using image embedding similarity from the entire priming pool. The visual similarity of the retrievals are apparent, and they are generally from the appropriate categorical cluster. Doing this filtering results in a significantly more relevant priming pool.

4.5 Experiments

Our key results include: **1.** Priming improves performance over baselines in the few-shot setting by 3.81% on average across all datasets and 2.4% on ImageNet in the zero-shot setting **2.** Priming in the transductive, or on-the-fly, setting further improves performance over baselines by 2.51% accuracy and 1.09% over standard Neural Priming **3.** Priming is complementary to existing prompt-tuning methods. Our finding indicates that images in the priming set impart distinct information to the model compared to textual class descriptions. We include full details of hyperparameter choices and error bars included in the appendix.

4.5.1 Datasets and Architectures

We evaluate on standard transfer learning and distribution shift benchmarks. ImageNet [43] is a large-scale, general classification dataset that has been well-studied in both transfer learning and distribution shift. ImageNetV2 [192] is one of its natural distribution shift test sets, made by reproducing the original data collection procedure of ImageNet, but even modern large-scale pre-trained models have performance drops on it. ImageNet Sketch [253] and ImageNet-R [79] are natural distribution shifts created by assembling

Table 4.1: Performance of Neural Priming and comparable methods in the zero-shot setting. Priming consistently improves top-1 accuracy across standard transfer learning data sets. Performance reported for the OpenCLIP ViT-B-16 model pretrained on LAION-2B.

	ImageNet	Stanford Cars	FGVC Aircraft	Flowers102	Food101	Oxford Pets	SUN397
CLIP [182, 87]	68.30	87.40	25.86	71.65	86.58	90.21	67.35
Retrieval + Finetuning	70.28	87.95	26.22	72.15	86.63	90.35	68.01
VLM [150]	69.35	87.88	28.54	72.11	86.31	90.24	67.73
CuPL [178]	70.25	88.63	29.64	72.32	86.20	91.16	70.80
Priming (Ours)	70.75	89.30	33.03	79.81	86.66	**91.87**	71.21
Priming + CuPL (Ours)	**71.38**	**90.23**	**36.00**	**80.04**	**86.86**	91.85	**72.35**

sketches and various renditions of the ImageNet classes. ImageNet-A [80] is a natural adversarial distribution shift of ImageNet, created by collecting images that are misclassified by ResNets. StanfordCars [114], FGVCAircraft [143], Flowers102 [167], and OxfordPets [172] are fine-grained classification datasets which require understanding subtle visual differences between classes and are commonly used for transfer learning benchmarks [76, 87, 182, 289]. SUN397 [271] is a large-scale scene recognition dataset with 397 scene categories.

We perform our experiments with OpenCLIP models [265] trained on LAION-2B and 400M [213]. We choose OpenCLIP because their pretrain datasets are publicly available, therefore we can control what data is introduced to the model. The model architecture reported in the main paper is the B-16 variant trained on LAION-2B unless otherwise stated.

4.5.2 Zero-shot Results

In this setting, our model only has access to data it has seen during pre-training, in this case LAION-2B. Neural Priming improves top-1 accuracy by 2.45% on ImageNet and 4.25% on average across 6 other diverse downstream datasets compared to the CLIP baseline. In the zero-shot setting, Neural Priming outperforms the 3-shot CLIP model on StanfordCars and FGVCAircraft. This result is particularly noteworthy since traditionally training on in-distribution data generally outperforms zero-shot techniques [285]. Note that we do not present error-bars for the zero-shot experiments as the process is deterministic.

We also compare with VLM [150] and CuPL [178], two zero-shot prompt-tuning methods which obtain

natural language descriptions of each class using language models, and a retrieval with fine-tuning baseline. Interestingly, we find that Neural Priming is complementary to existing prompt-tuning methods. The accuracy improvements from CuPL and VLM are additive with Neural Priming. For example, CuPL and Neural Priming each improve performance by **3.78%** and **7.17%** respectively on FGVCAircraft. Ensembling the methods results in **10.74%** improvement over the baseline (Table 4.2). This surprising result suggests that the textual class descriptions in CuPL and VLM provide unique information to the model that differ from the information obtained from the images in the priming set.

Another observation is that Neural Priming is especially effective for specialized domains such as StanfordCars, Flowers102, and FGVCAircraft. We speculate this is due to the fact that the label space and image content differs from the majority of the pre-training set. For example, although airplanes occur frequently in LAION-2B, they are rarely described according to their specific model such as *Boeing 737-200*. Therefore, recalling and priming the model on pre-train images with such fine-grained classes significantly improves the model.

In contrast, for datasets which are more aligned with LAION-2B and the distribution of internet images, such as ImageNet and SUN397, the accuracy gain provided from Neural Priming is smaller in comparison, albeit still significant. In the limit of this trend, Food101 sees almost no improvement across all methods, and even training on in-distribution data for the few-shot case barely improves the accuracy. We speculate that this is because images similar to those in Food101 are already well-represented in LAION-2B, rendering additional food images of marginal informational value.

To be precise, when we refer to term "shot number" throughout the experiments section, we mean the number of labeled examples from the target training set. We do not consider images retrieved from LAION-2B as shots in this setting because they are obtained from the pre-training set.

4.5.3 Few-Shot Results

Neural Priming improves performance for all datasets and shots in the few-shot setting. We compare with CoOp, a recent method for few-shot prompt-tuning, and a Nearest-class-Mean (NCM) baseline. On average across all shots and datasets Neural Priming improves by 3.81% in accuracy over the closest baseline. Results can be found in Figure 4.3.

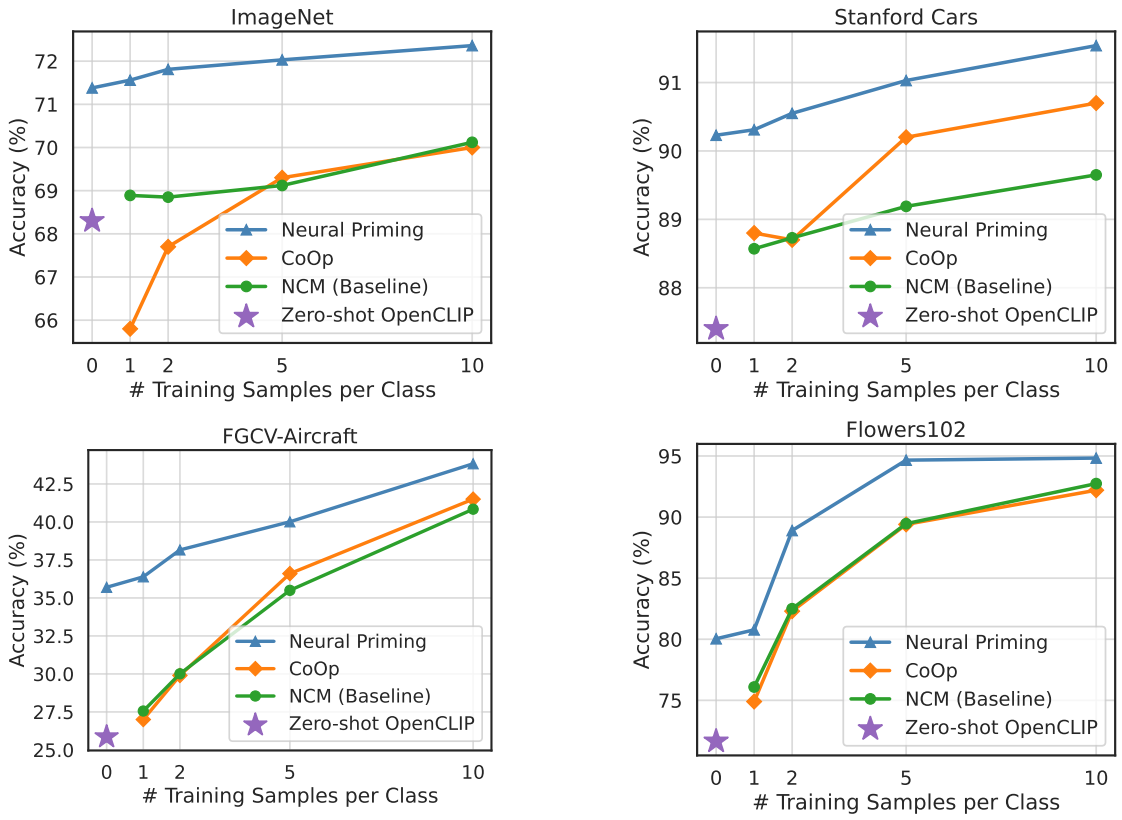


Figure 4.3: Performance of Neural Priming and comparable methods in the few-shot setting. We find consistent improvement across shot numbers and datasets. In particular, Neural Priming especially excels for fine-grained datasets such as FGVCAircraft and Flowers102. We hypothesize that such fine-grained captioned images are not well represented in LAION-2B, therefore revisting this subset of data improves the model more.

Notably, we find that Neural Priming can match the accuracy of models trained with a substantial number of training examples *without using any of the labeled training data* for all of the evaluated datasets (Figure 4.3). Additionally, we observe that as the shot number increases, improvement over the baseline decreases. At 1-shot the improvement in accuracy over the baselines is 5.63% on average, while at 10-shot the improvement is 2.04%. Intuitively, as the model receives more target training data, obtaining additional examples from the pretrain set becomes less necessary.

4.5.4 Transductive Results

We compare Neural Priming in the transductive setting on 4 standard distribution shift datasets, ImageNet-V2, ImageNet Sketch, ImageNet-R and ImageNet-A. Distribution shift datasets are a natural application of

Table 4.2: Performance of Neural Priming and relevant methods for the transductive setting. Neural Priming finds examples similar to the test image at inference to optimize the model. Models are evaluated zero-shot on 4 distribution shift datasets. Neural Priming excels on distribution shifts which differ significantly from the natural language description of the class names. Performance reported for the OpenCLIP ViT-B-16 model pretrained on LAION-2B.

	ImageNet-V2	ImageNet-R	ImageNet Sketch	ImageNet-A
CLIP [87, 182]	59.35	64.57	57.05	35.95
TPT [217]	59.84	78.74	52.75	36.92
Priming (Ours)	60.12	77.98	58.29	37.56
Transduct. Priming (Ours)	**60.76**	**79.37**	**59.97**	**38.20**

adaptation at test-time. Often real-world datasets differ from the training data, therefore models should be able to adapt on-the-fly. In this setting, the model can learn from the test images without labels before making predictions. We compare with Test-Time Prompt-Tuning (TPT), a state-of-the-art method which uses a self-supervised objective to learn from test data.

We find that Neural Priming with images in the test set improves performance over standard Neural Priming by 1.09% as well as 2.51% over TPT across the 4 distribution shifts (Table 4.2). Looking at Figure 4.2, we qualitatively see that the priming pool more closely matches the test images after filtering for the closest images in the initial priming pool. Though the distribution shift can often be imperceptible such as between ImageNet and ImageNetv2, quantitatively we see that the transductive filtering step finds images in the pretraining close to the test distribution.

The transductive retrieval for 50,000 images in the test set on average takes 96 seconds for a priming pool of 1 million images, while retraining the classifier takes on average 11.5 seconds for a priming pool of size 10,000 on standard hardware.

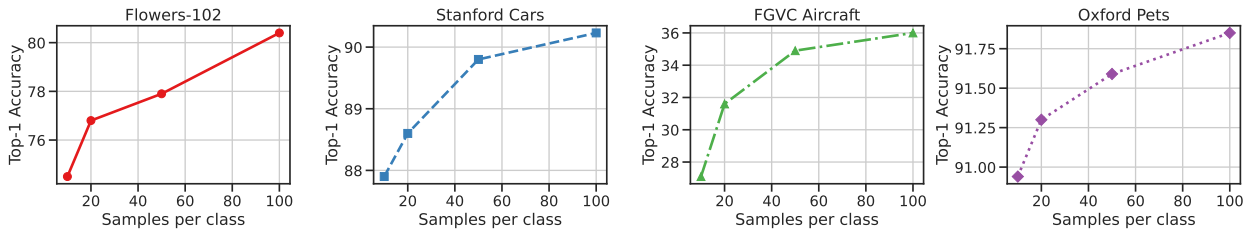


Figure 4.4: Ablation over the number of samples per class in the priming pool. We observe a consistent zero-shot accuracy improvement as the number of samples drawn from our pool increases.

4.5.5 Ablations

We investigate the impact of the priming pool size on the zero-shot accuracy of downstream tasks (Figure 4.4). Our analysis reveals that as the size of the priming pool increases, there is a general improvement in accuracy. However, there are certain limitations associated with enlarging the pool. The majority of classes in the downstream task have a limited number of available images. Consequently, when we retrieve a larger number of images for the priming pool, they tend to contain more noise and mislabeled samples. Furthermore, for rare classes, the number of images obtained through exact string search is often less than 100. To address this, a potential extension could involve utilizing a language model to generate alias names for classes, which could then be used to perform additional string searches, thereby expanding the initial priming pool size.

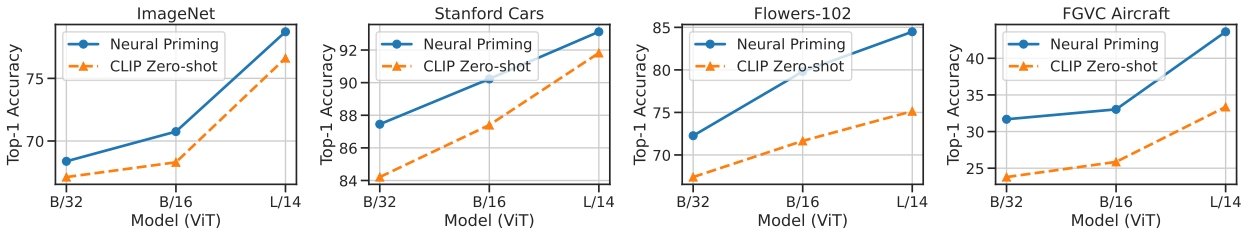


Figure 4.5: Analyzing the effect of model capacity on Neural Priming. We find the relative error reduction stays consistent even as the scale of the model increases.

We also analyze the impact of the architecture on the accuracy improvement achieved by Neural Priming in the zero-shot setting (Figure 4.5). To examine this, we conduct experiments using models of varying capacities, namely ViT B-32, B-16, and L-14. We observe that the gains remains consistent across the models. This finding suggests that even as we scale the architecture's capacity, our method will continue to yield significant and consistent relative error reduction.

4.6 Limitations

Neural Priming has a few potential limitations. Firstly, it requires that the pre-train dataset contains images similar to those in the downstream task. Though all of the datasets we benchmark have abundant relevant data, it is possible for more out-of-distribution datasets that LAION-2B simply does not contain related or queryable images. Secondly, accurate class names are required for retrieval. Meaningful class names for some datasets can be difficult to obtain. For example, in the Flowers102 dataset, some flower species are

given by their latin names, which leads to poor retrieval. This issue generally affects open-vocabulary models which require accurate class names to initialize the zero-shot classifier. This limitation may be resolved by using language models to replace class names with their more commonly known synonyms. Lastly, Neural Priming requires access to the pre-training data set which is not always possible such as in the case of OpenAI variant of CLIP. In this case a surrogate dataset would likely suffice, such as using LAION-2B.

4.7 Discussion & Conclusion

We present Neural Priming, a method to improve the performance of open-vocabulary models by leveraging their own large-scale, diverse pre-training data with no additional data required. With Neural Priming, we demonstrate how to construct a high quality priming pool of examples from the pre-training dataset relevant to a particular task and how to utilize this pool to improve our model. We further show that our method is effective across a variety of downstream tasks and settings. In particular, our method can be used in situations where only natural language descriptions of relevant classes are given, when we have the ability to adapt at inference time, and when we are provided with few labeled in-distribution examples. In all settings, our framework demonstrates a substantial improvement in performance over existing interventions, and is in fact complementary with current prompt-tuning and robustness methods. Our method is also computationally cheap, not requiring any modification of model backbone weights and only a fast text search on the pre-training corpus.

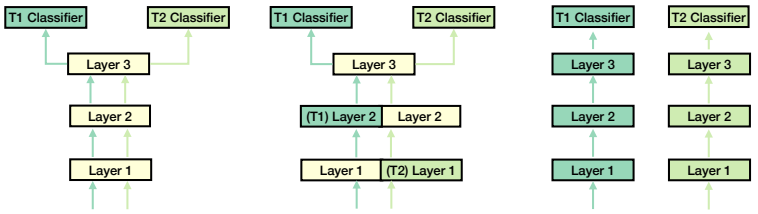
The efficacy of Neural Priming leads to some interesting questions for future work. For example, if the model has seen this data before, why does it help to recall them? We hypothesize that this is due to the fact that the diversity of these datasets introduces competing objectives, which are difficult for the model to optimize directly. For example, the same kind of image could appear with multiple captions and vice-versa, making it difficult to prompt a CLIP model trained on such data at inference time for a particular task. A systematic study of this could elucidate important limitations of current large-scale training paradigms.

Chapter 5

Task adaptive Parameter Sharing for Multi-Task Learning

5.1 Overview

Adapting pre-trained models with broad capabilities has become standard practice for learning a wide range of downstream tasks. The typical approach of fine-tuning different models for each task is performant, but incurs a substantial memory cost. To efficiently learn multiple downstream tasks we introduce Task Adaptive Parameter Sharing (TAPS), a general method for tuning a base model to a new task by adaptively modifying a small, task-specific subset of layers. This enables multi-task learning while minimizing resources used and competition between tasks. TAPS solves a joint optimization problem which determines which layers to share with the base model and the value of the task-specific weights. Further, a sparsity penalty on the number of active layers encourages weight sharing with the base model. Compared to other methods, TAPS retains high accuracy on downstream tasks while introducing few task-specific parameters. Moreover, TAPS is agnostic to the model architecture and requires only minor changes to the training scheme. We evaluate our method on a suite of fine-tuning tasks and architectures (ResNet, DenseNet, ViT) and show that it achieves state-of-the-art performance while being simple to implement.



(a) Feature Extractor (b) Our Method (c) Fine-Tuning

Figure 5.1: Overview of our approach. Difference between (b) our approach, (a) feature extractor as well as (c) finetuning. Here we have two tasks T_1 and T_2. T_1 shown by turquoise and T_2 by green. Yellow boxes denote the base network layer. Our approach adds task specific parameters to different layers based on the target tasks. Notice that this is in contrast to (a) no task specific layers are used and (c) where every layer is tasks specific. Also (c) suffers from catastrophic forgetting, and lose the base network's parameters unlike (b) and (a).

5.2 Introduction

Real-world applications of deep learning frequently require performing multiple tasks (Multi-Task Learning/MTL). To avoid competition between tasks, a simple solution is to train separate models starting from a common pre-trained model. Although this approach results in capable task-specific models, the training, inference, and memory cost associated grows quickly with the number of tasks. Further, tasks are learned independently, missing the opportunity to share when tasks are related.

Ideally, one would train a single model to solve all tasks simultaneously. A common approach is to fix a base model and add task-specific parameters (e.g., adding branches, classifiers) which are trained separately for each task. However, deciding where to branch or add parameters is non-trivial since the optimal choice depends on both the initial model and the downstream task, to the point that some methods train a secondary network to make these decisions.

Moreover, adding weights (layers, parameters, etc.) to a network independent of the task is also not ideal: some methods [145, 188, 126] add a small, fixed number of learnable task-specific parameters, however they sacrifice performance when the downstream task is dissimilar from the pre-training task. Other methods perform well on more complex tasks, but add an excessive number of task-specific parameters even when the downstream task is simple [203, 67, 284]

In this work, we overcome these issues by introducing Task Adaptive Parameter Sharing (TAPS). Rather than modifying the architecture of the network or adding a fixed set of parameters, TAPS adaptively selects a

minimal subset of the existing layers and retrains them. At first sight, selecting the best subset of layers to adapt is a complex combinatorial problem which requires an extensive search among 2^L different configurations, where L is the number of layers. The key idea of TAPS is to relax the layer selection to a continuous problem, so that deciding which layers of the base model to specialize into task-specific layers can be done *during training* by solving a joint optimization using stochastic gradient descent.

The final result is a smaller subset of task-specific parameters (the selected layers) which replace the base layers. Our approach has several advantages: (i) It can be applied to any architecture and does not need to modify it by introducing task-specific branches; (ii) TAPS does not reduce the accuracy of the target task (compared to the paragon of full fine-tuning) while introducing fewer task-specific parameters; (iii) The decision of which layers to specialize is interpretable, done with a simple optimization procedure, and does not require learning a policy network; (iv) It can be implemented in a few lines, and requires minimal change to the training scheme.

Our method finds both intuitive sharing strategies, and other less intuitive but effective ones. For example, TAPS tends to modify the last few layers for ResNet models while for Vision Transformers TAPS discovers a significantly different sharing pattern, learning to adapt only the self-attention layers while sharing the feed-forward layers.

We test our method on standard benchmarks and show that it outperforms several alternative methods. Moreover, we show that the results of our method are in-line with the standard fine-tuning practices used in the community. The contributions of the paper can be summarized as follows:

1. We propose TAPS, a method for differentiably learning which layers to tune when adapting a pre-trained network to a downstream task. This could range from adapting or specializing an entire model, to only changing 0.1% of the pre-trained model, depending on the complexity/similarity of the new task.

2. We show that TAPS can optimize for accuracy or efficiency and performs on par with other methods. Moreover, it automatically discovers effective architecture-specific sharing patterns as opposed to hand-crafted weight or layer sharing schemes.

3. TAPS enables efficient incremental and joint multi-task learning without competition or forgetting.

5.3 Related Work

Multi-Domain and Incremental Multi-Task Learning In many applications, it is desirable to adapt one network to multiple visual classification tasks or domains (Multi-Domain Learning, or MDL). Unlike Multi-Task Learning (MTL) where the tasks are learned simultaneously, in MDL the focus is to learn the domains incrementally, as often not all data is available at once. Accordingly, in this work, we also refer to MDL as incremental MTL. The standard approach for adapting a network to a single downstream task is fine-tuning. However, adapting to multiple domains incrementally poses the challenge of catastrophically forgetting previously learned tasks. To foster research in the area, Rebuffi et al. [188] introduced the Visual Decathlon challenge and proposed residual adapters. Residual adapters fix most of the network while training small residual modules that adapt to new domains. This architecture was modified to a parallel adapter architecture in [191]. A controller based method called Deep Adaptation (DA) was introduced in [199] to modify the learning algorithm using existing parameters. A simpler approach of using binary masks was proposed in Piggyback [145]. Task specific masks are learned then applied to the weights of the original network. This approach was further extended in Weight Transformations using Binary Masks (WTPB) [146] by modifying how the masks are applied. These methods focus on adding a small number of new parameters per task and underperform on more complex tasks as they have fixed capacity. Other solutions such as SpotTune [67] focus on performance without consideration for parameter efficiency. It trains an auxiliary policy network that decides whether to route each sample through a shared layer or task-specific layer. In contrast, TAPS does not require modification of the network architecture via adapters or training an auxiliary policy network. TAPS trains in one run with the same base architecture and requires no additional inference compute.

Parameter Efficient Multi-Domain Learning (MDL) Another line of work in MDL is that of parameter sharing [158, 144]. These approaches typically perform multi-stage training. NetTailor [158] leverages the intuition that simple tasks require smaller networks than more complex tasks. They train teacher and student networks using knowledge distillation and a three-stage training scheme. PackNet [144] adds multiple tasks to a single network by iterative pruning. However, pruning weights generally causes performance degradation. More recently, Berriel et al. proposed Budget-Aware Adapters (BA^2) [15]. This method selects and uses

feature channels that are relevant for a task. Using a budget constraint, a network with the desired complexity can be obtained. In summary, most efficient parameter methods obtain efficiency at the loss of performance. Even with the largest budget in BA^2, the performance is substantially lower compared to TAPS. Unlike existing methods, TAPS does not need to choose a high accuracy or a high efficiency regime. As shown in fig. 5.2, for a given task we can obtain models anywhere along the accuracy-efficiency frontier.

Multi-Task Learning (MTL) MTL focuses on learning a diverse set of tasks in the same visual domain simultaneously by sharing information and computation, usually in the form of layers shared across all tasks and specialized branches for specific tasks [109, 185]. A few methods have attempted to learn multi-branch network architectures [138, 247] and some methods have sought to find sharing parameters among task-specific models [156, 200, 60]. A closely related work is AdaShare [230], which learns a task-specific policy that selectively chooses which layers to execute for a given task in the multitask network. They use Gumbel Softmax Sampling [140, 92] to learn the layer sharing policy jointly with the network parameters through standard back-propagation. Since this approach skips layers based on the task, it can only be applied to a subset of existing architectures where the input and output dimension of each layer is constant. Additionally because the sharing policy is trained jointly across all tasks, AdaShare cannot learn in the incremental setting.

Incremental Learning Related to MDL is the problem of incremental learning. Here, the goal is to start with a few classes and incrementally learn more classes as more data becomes available. There are two approaches in this regard, methods that add extra capacity [203], [284] (layer, filter, etc.) and methods that do not[30, 106, 189, 127]. Methods that do not add extra capacity mitigate catastrophic forgetting by either using a replay buffer [30, 189] or minimizing changes to the weights [106]. Similar to our method, [203, 284] add capacity to the network to accommodate new tasks and prevent catastrophic forgetting. Progressive network [203] adds an entire network of parameters while Side-Tune [284] adds a smaller fixed-size network. Adding fixed capacity independent of the downstream task is sub-optimal since less complex tasks require fewer added parameters. In the extreme case where the pretrain task matches the downstream task no additional parameters are needed. In contrast to previous methods, TAPS adaptively adds capacity based on the downstream task and base network. Further, the goal of TAPS differs from incremental learning in that it starts with a pre-trained base model and learns new tasks or domains as opposed to adding new classes from

a similar domain.

5.4 Approach

Given a pre-trained deep neural network with L layers of weights and a set of K target tasks $\mathcal{T} = \{T_1, T_2, ..., T_K\}$, for each task we want to select the minimal necessary subset of layers that needs to be tuned to achieve the best (or close to the best) performance. This allows us to learn new tasks incrementally while adding the fewest new parameters. In principle, this requires a combinatorial search over 2^L possible subsets. The idea of Task Adaptive Parameter Sharing (TAPS) is to relax the combinatorial problem into a continuous one, which will ultimately give us a simple joint loss function to find both the optimal task-specific layers to tune and optimize parameters of those layers. An overview of our approach is shown in fig. 6.1.

Weight parametrization We first introduce a scoring parameter s_i for each shared layer, where $i = 1, \ldots, L$. We then reparametrize the weights of each layer as:

$$w_i = \bar{w}_i + I_\tau(s_i)\, \delta w_i, \tag{5.1}$$

where $\bar{w}_i$ are the (shared) weights of the pre-trained network and δw_i is a trainable parameter which describes a task-specific perturbation of the base network. The crucial component is the indicator function $I_\tau(s_i)$ defined by

$$I_\tau(s_i) = \begin{cases} 1 & \text{if } s_i \geq \tau, \\ 0 & \text{otherwise.} \end{cases} \tag{5.2}$$

for some threshold τ. Hence, when $s_i \geq \tau$, the layer is transformed to a task-specific layer, and consequently will make its parameters task-specific. On the other hand, for $s_i < \tau$ the layer is the same as the base network and no new parameters are introduced.

The same approach can be used for different architecture and layer types, whether linear, convolutional, or attention. In the latter case, we add task-specific parameters to the query-key-value matrices as well as the projection layers.

Joint optimization Given the parametization in eq. (5.1), we can recast the initial combinatorial problem as a joint optimization problem over weight deltas and scores:

$$(\delta\mathbf{w}, \mathbf{s}) = \arg\min_{\mathbf{w},\mathbf{s}} \mathcal{L}_D(\mathbf{w}) + \frac{\lambda}{L}\sum_{i=1}^{L} |s_i|, \tag{5.3}$$

where $\mathbf{s} = (s_1, \ldots, s_L)$, $\delta\mathbf{w} = (\delta w_1, \ldots, \delta w_L)$, $\mathbf{w} = (w_1, \ldots, w_L)$ and we denote with $L_D(\mathbf{w})$ the loss of the model on the dataset D. The first term of eq. (5.3) tries to optimize the task specific parameters to achieve the best performance on the task, while the second term is a sparsity inducing regularizer on $\mathbf{s}$ which penalizes large values of s_i, encouraging layer sharing rather than introducing task-specific parameters.

Straight-through gradient estimation While the optimization problem in eq. (5.3) captures the original problem, it cannot be directly optimized with stochastic gradient descent since the gradient of the indicator function $I_\tau(s)$ is zero at all points except τ. To make the problem learnable, we utilize a straight-through gradient estimator [14]. That is, we modify the backward pass and redefine the gradient update as:

$$\nabla_{s_i} w_i = \delta w_i,$$

in place of the true gradient $\nabla_{s_i} w_i = 0$. This corresponds to computing the derivative of the function $w_i = \bar{w}_i + s_i \delta w_i$ rather than $w_i = \bar{w}_i + I_\tau(s_i)\delta w_i$. The inspiration to utilize the straight-through estimator for gating task-specific weight perturbations, δw_i, comes from sparsity literature [263, 186] where it has been used to prune network weights.

Joint MTL A natural question is if, rather than using a generic pretrained model, we can learn a base network optimized for multi-task learning. In particular, is there a pretrained representation that reduces the number of task-specific layers that need to be learned to obtain optimal performance? To answer the question, we note that if data from multiple tasks is available simultaneously at training time, we can optimize eq. (5.1) jointly across all downstream tasks for both the base weights $\bar{w}_i$ (which will be shared between all tasks) and

the task specific δw_i. The loss function becomes:

$$(\bar{\mathbf{w}}, \delta\mathbf{w}_1, \ldots, \delta\mathbf{w}_K, \mathbf{s}_1, \ldots, \mathbf{s}_K) = \\ \arg\min_{\bar{\mathbf{w}}, \delta\mathbf{w}^k, \mathbf{s}} \sum_{k=1}^{K} \Big(\mathcal{L}_{D_k}(\bar{\mathbf{w}}, \delta\mathbf{w}^k) + \frac{\lambda}{L} \sum_{i=1}^{L} |s_i^k| \Big), \tag{5.4}$$

where K is the total number of tasks, $\bar{\mathbf{w}}$ is shared between all tasks and $\delta\mathbf{w}_k$ and $\mathbf{s}_k$ are task specific parameters. This loss encourages learning common weights $\bar{\mathbf{w}}$ in such a way that the number of task specific parameters is minimized, due to the L_1 penalty on $\mathbf{s}$. In Sec. 5.5.2 we show that the joint multi-task variant of TAPS does increase weight sharing without loss in accuracy. In particular, the number of task-specific parameters is significantly reduced in the joint multi-task training setting with respect to incremental multi-task training.

A limitation of the joint multi-task variant of TAPS and other joint MTL methods [230] is that the memory footprint during training increases linearly with the number of tasks. Our solution is to learn a single network which is trained jointly on all tasks, with task specific classifiers. Then train with the incremental variant of TAPS (Eq. 5.3) to adapt the jointly trained base network to each task. This approach achieves comparable accuracy and parameter sharing with the joint variant, while requiring constant memory during training.

Batch Normalization Learning task specific batch normalization layers improves accuracy on average by $2-3\%$, while adding only a small amount of parameters ($.06\%$ for ResNet-34 model). For this reason, we follow the same setup as most methods and learn task-specific batch-norm parameters.

5.5 Experiments

In this section, we compare TAPS with existing methods in two settings: incremental MTL (§5.5.1) and joint MTL (§5.5.2). The details are as follows.

5.5.1 Incremental MTL

In this scenario, methods adapt the pre-trained model for each task individually and combine them into a single model that works for every domain. This approach is efficient during training in terms of both speed

and memory as it can be parallelized and requires at most $2\times$ the parameters of the base model. Alternatively, all tasks could interact and learn common weights, which is the joint multi-task scenario described in §5.5.2.

Datasets We show results on two benchmarks. One is the standard benchmark used in [67, 145, 144, 146] which consists of 5 datasets: Flowers [166], Cars [113], Sketch [48], CUB [251] and WikiArt [207]. Following [15], we refer to this benchmark as ImageNet-to-Sketch. For dataset splits, augmentation, crops, and other aspects, we use the same setting as [145]. Our second benchmark is the Visual Decathlon Challenge [188]. This challenge consists of 10 tasks which include the following datasets: ImageNet [202], Aircraft [142], CIFAR-100 [115], Describable textures [35], Daimler pedestrian classification [160], German traffic signs [225], UCF-101 Dynamic Images [223], SVHN [163], Omniglot [121], and Oxford Flowers [166].

Methods of Comparison Our paragon is fine-tuning the entire network separately for each task, resulting in the best performance at the cost of no weight sharing. Our baseline is the fixed feature extractor, which typically gives the worst performance and shares all layers. In the incremental multi-task setting we compare our method with Piggyback [145], SpotTune [67], PackNet [144], and Residual Adapters [191].

Metrics We report the top-1 accuracy on each task and the S-score for the Visual Decathlon challenge as proposed in [188]. In addition, we report the total percentage of additional parameters and task specific layers needed for all tasks. Methods [145, 144, 146] that use a binary mask for their algorithm report the theoretical total number of bits (e.g., 32 for floats, 1 for boolean) required for storage rather than reporting the total number of parameters. However, as [145] notes, depending on the hardware, the actual storage cost in memory may vary (e.g., booleans are usually encoded as 8-bits). To establish parity between different reporting structures, we report the total number of parameters used without normalization and the normalized count (assuming that a boolean parameter can be stored as 8-bits).

Training details We use an ImageNet pre-trained ResNet-50 [74] model as our base model for ImageNet-to-Sketch. We train TAPS for 30 epochs with batch size 32 on a single GPU. For the fine-tuning paragon and fixed feature extractor we report the best performance for the learning rates $lr \in \{0.001, 0.005\}$.

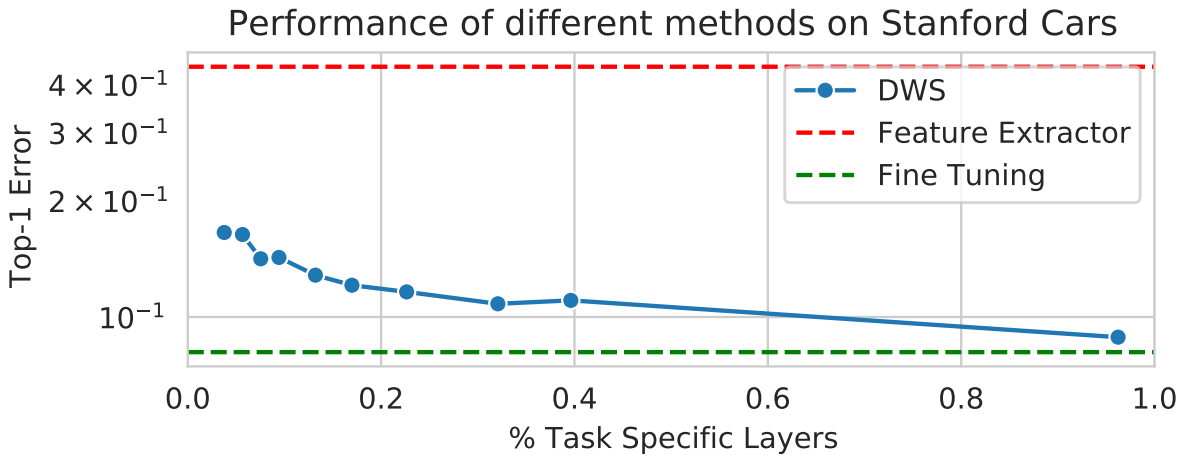


Figure 5.2: Accuracy vs Task Specific Layers: Accuracy vs percentage of task specific layers for the Cars dataset is shown. Varying λ gives a variety of configuration. With even a very few task specific layers (high λ), we perform significantly better than the feature extraction baseline. Our method also reaches the fine-tune performance but needs a significant number of task specific layers.

For TAPS use the SGD optimizer with no weight decay. We sweep over $\lambda \in \{0.25, 0.5, 0.75\}$ and $lr \in \{0.001, 0.005\}$. We fix the threshold, $\tau = 0.1$, for all datasets and use the cosine annealing learning rate scheduler.

In addition to ResNet-50, we also apply TAPS to DenseNet-121 [84] and Vision Transformers [44]. To the best of our knowledge, we are the first to provide results for a transformer based architecture.

For the Visual Decathlon challenge, we use the WideResNet-28 as in [188], which is also referred to as ResNet-26 in [67]. Following existing work, we use a learning rate of 0.1 with weight decay of 0.0005 and train the network for 120 epochs. We report our results for $\lambda \in \{0.25, 0.5, 1.0\}$. Like existing methods [67, 188, 191], we report our accuracy on the test set while training on the training and validation dataset. We also calculate the S-Score to make a consolidated ranking of our method.

Results on ImageNet-to-Sketch table 5.1 shows the comparison of our method with existing approaches on ImageNet-to-Sketch. Average accuracy over 3 runs of our method and fine-tuning are reported. TAPS outperforms Piggyback and Packnet across all 5 datasets, Spot-Tune for 3 out of the 5 datasets, WTPB and BA^2 for 4 out of 5 datasets. We also note that TAPS uses, on average, only 57% of the parameters that Spot-Tune does. We do not outperform existing methods on the Cars dataset. Indeed, for this dataset the best

results are obtained with $\lambda = 0.0$ (see fig. 5.2), suggesting that most layers needs to be adapted for optimal performance. In general, we perform significantly better in terms of accuracy compared to methods that are parameter efficient, while we achieve the same performance as methods that are designed for accuracy, but at a fraction of the parameter cost.

Table 5.1: Performance of various method using a ResNet-50 model on ImageNet-to-Sketch benchmark. Accuracy for different methods are shown for the different datasets. For TAPS and fine-tuning, we report the average accuracy over three runs. We report the total number of paramters (when available) and in parenthesis the data-type normalized parameter count. Numbers in bold denote the best performing method (other than fine-tuning) for each dataset. For Packnet, the arrow indicates the order of adding tasks.

	Param Count	Flowers	WikiArt	Sketch	Cars	CUB
Fine-Tuning	6×	95.73	78.02	81.83	91.89	83.61
Feature Extractor	1×	89.14	61.74	65.90	55.52	63.46
Piggyback [145]	6× (2.25×)	94.76	71.33	79.91	89.62	81.59
Packnet [144] →	(1.60×)	93.00	69.40	76.20	86.10	80.40
Packnet [144] ←	(1.60×)	90.60	70.3	78.7	80.0	71.4
Spot-tune [67]	7× (7×)	96.34	75.77	80.2	**92.4**	**84.03**
WTPB [146]	6× (2.25×)	96.50	74.8	80.2	91.5	82.6
BA^2 [15]	3.8× (1.71×)	95.74	72.32	79.28	92.14	81.19
TAPS	4.12×	**96.68**	**76.94**	**80.74**	89.76	82.65

Task-specific layers In fig. 5.3, we show the layers that are task-specific for the different datasets. As expected, the final convolutional layers are always adapted. This corresponds to the common practice of freezing the initial 3 out of the 4 blocks of the ResNet-50 model and fine-tuning the last block. But, interestingly, we see that some of the middle layers are also always active. For instance, layer 26 is often

Table 5.2: Percentage of additional parameters and layers. The percentage of task specific parameters and layers needed for each dataset across network architectures is shown. Numbers in bold denote the lowest across architectures. ViT-S/16 uses the least number of extra parameters, while ResNet-50 adds the least number of task specific layers.

	Flowers	WikiArt	Sketch	Cars	CUB
Percentage of Additional Parameters					
DenseNet-121	80.2	41.2	58.5	50.4	43.8
ViT-S/16	**41.3**	**30.4**	**24.1**	**26.1**	**41.3**
ResNet-50	65.5	52.8	75.9	41.9	70.6
Percentage of Task Specific Layers					
DenseNet-121	69.4	22.5	41.1	28.3	23.9
ViT-S/16	54.2	20.8	22.9	37.5	54.2
ResNet-50	**22.6**	**20.8**	**43.4**	**14.5**	**28.3**

adapted as a task-specific layer. Specifically for the Sketch task which has differing low-level features compared to ImageNet, the first convolution layer is consistently considered as task-specific. We see this is the case across varying values of λ, which aligns with the intuition that initial layers of ResNet should be retrained when transferring to a domain with different low-level features.

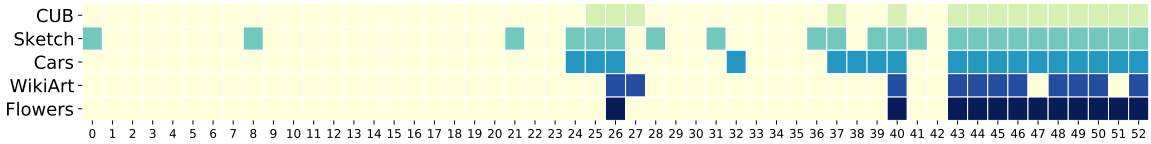


Figure 5.3: Task-specific layers for different datasets. Each row shows the 53 convolution layers of ResNet-50 layers for different datasets. Layer 0 is closest to the input, while layer 52 is closest to the classifier. Layers shown in yellow are shared with the ImageNet pre-trained model while layers shown in color are task-specific weights. We see that most tasks specific layers are toward the classifier.

Effect of choice of pre-trained model To analyze the effect of using different pre-trained models, we replaced the base ImageNet model with a Places-365 model and applied TAPS on the datasets listed in table 5.1. We notice changes both in task-specific layers and in the performance. The number of task-specific layers increases for every task, in particular the average percentage of task-specific layers grows from 25.91% to 36.60%. We hypothesize that the Places-365 pre-training may not be well suited for object classification, so more layers need to be tuned. Supporting this, we also see a drop 2.77% in average accuracy across the datasets. These observations are consistent with the findings in [145].

Table 5.3: Accuracy of various methods across architectures and datasets. Classification accuracy for various methods is shown across different datasets and architectures. The ViT-S/16 model has the highest accuracy, across datasets. TAPS is able to match the fine-tuning performance for ViT-S/16 and is about 1-2% away for DenseNet-121.

	Flowers	WikiArt	Sketch	Cars	CUB
DenseNet-121					
Fine-Tuning	95.6	77.0	81.1	89.5	82.6
Piggyback	94.7	70.4	79.7	89.1	80.5
TAPS	95.8	73.6	80.2	88.0	80.9
ViT-S/16					
Fine-Tuning	99.3	82.6	81.9	89.2	88.9
TAPS	99.1	82.3	82.2	88.7	88.4

Effect of Architecture Choice To demonstrate that TAPS is agnostic to architecture, we evaluate it on DenseNet-121 [84]. We report the performance of our method compared to fine-tuning and Piggyback in table 5.3 and parameters in table 5.2. The number of task-specific layers are high in DenseNet-121 compared to ResNet-50. We conjecture that because of the extra skip connections, changing a single layer has more impact on the output compared to the ResNet model.

Transformers We show results of our method on the transformer architecture. Here, we use the ViT-S/16 model [227]. In table 5.3 we report the performance of our method and parameters in table 5.2. For the transformer architecture, the performance is better than CNNs as expected. We also notice that fewer parameters are made task-specific compared to CNNs. Although TAPS modifies more task specific layers, the adapted layers have fewer parameters. We show the layers that are task-specific in fig. 5.4. From this figure we see that the layers that are adapted to be task-specific for transformers follow a very different pattern from those of CNNs. While in the latter, lower layers tend to be task agnostic, and final layers task-specific, this is not the case for transformers. Here, layers throughout the whole network tend to be adapted to the task. Moreover, attention and projection layers tend to be adapted, whereas MLP layers are fixed. This shows that TAPS can adapt in nontrivial ways to different architectures without any hand-crafted prior.

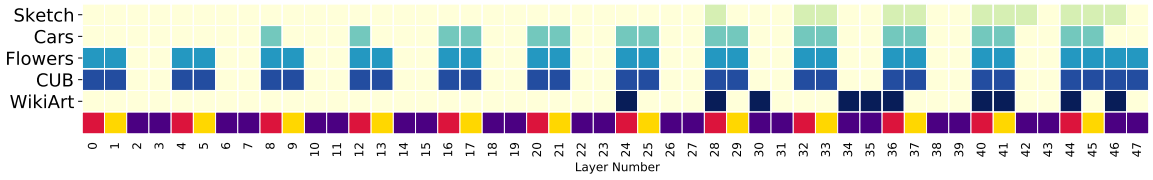


Figure 5.4: Task-specific layers for different datasets for the ViT model. The figure shows the task-specific layers that are active for different datasets. Each rows shows the different layers that are present in a ViT model. Layer 0 is closest to the input. Layers shown in yellow are shared with the ImageNet pre-trained model. The last row shows the type of layer and is denoted in color. Here crimson denotes the Query-Key-Value in the attention layer, gold denotes the projection layer and purple denotes the MLP layer. Unlike CNNs, we see that the fine-tuning strategy is very different. Instead of freezing blocks, we need to freeze the MLP layers.

Visual Decathlon table 5.4 shows that for $\lambda = 0.25$, our method achieves the second highest S-score, without any dataset-specific hyper-parameter tuning. For this λ, we use half the number of parameters as Spot-Tune while performing better in 6/10 datasets and also have a higher mean score. All variants of our

method outperform Res. Adapt., Deep Adapt., Piggyback. Moving from $\lambda = 0.25$ to $\lambda = 1.0$, we further reduce the number of task-specific layers by half while increasing the mean error only by 1%. At $\lambda = 1$, we we outperform Piggyback while using lesser total number of parameters and storage space of the models, even considering that Piggyback uses Boolean parameters.

To analyze what layers are being used we plot the active layers at the highest compression ($\lambda = 1$) in fig. 5.5. For all datasets, the number of task-specific layers is small, with Omniglot requiring the most layers, while DPed and GTSR require the least. In fact, for the latter datasets, no task-specific layers are required outside of updating the batch norm layers (which leads to a significant boost in performance compared to fixed feature extraction). Again, we note that TAPS can easily find complex nonstandard sharing schemes for each dataset, which would otherwise have required an expensive combinatorial search.

Table 5.4: Accuracy of various methods on the Visual Decathlon Challenge datasets. Accuracy for each dataset, the mean accuracy across all datasets and the S-Score [188] is shown. TAPS has the second best S-Score at almost half the parameters of the best method. Our method can tradeoff accuracy vs additional parameters. We report the total number of parameters and in parenthesis the data-type normalized parameter count.

Method	Params	Airc.	C100	DPed	DTD	GTSR	Flwr.	Oglt.	SVHN	UCF	Mean	S-Score
Fixed Feature [191]	1×	23.3	63.1	80.3	45.4	68.2	73.7	58.5	43.5	26.8	54.3	544
FineTuning [191]	10×	60.3	82.1	92.8	55.5	97.5	81.4	87.7	96.6	51.2	76.5	2500
Res. Adapt.[188]	2×	56.7	81.2.	93.9	50.9	97.1	66.2	89.6	96.1	47.5	73.9	2118
DAM [199]	2.17×	64.1	80.1	91.3	56.5	98.5	86.1	**89.7**	96.8	49.4	77.0	2851
PA [191]	2×	64.2	81.9	94.7	58.8	99.4	84.7	89.2	96.5	50.9	78.1	3412
Piggyback[145]	10×(3.25×)	65.3	79.9	97.0	57.5	97.3	79.1	87.6	**97.2**	47.5	76.6	2838
WTPB [146]	10×(3.25×)	52.8	**82.0**	96.2	58.7	99.2	**88.2**	89.2	96.8	48.6	77.2	3497
BA^2 [15]	6.13× (2.28×)	49.9	78.1	95.5	55.1	99.4	86.1	88.7	96.9	50.2	75.7	3199
Spot-tune [67]	11×	63.9	80.5	96.5	57.1	**99.5**	85.2	88.8	96.7	**52.3**	78.1	3612
TAPS (λ=0.25)	5.24×	**66.58**	81.76	**97.07**	**58.83**	99.07	86.99	88.79	95.72	51.92	**78.70**	3533
TAPS (λ=0.50)	3.88×	62.05	81.74	97.13	57.02	98.40	85.80	88.96	95.62	49.06	77.61	3180
TAPS (λ=0.75)	3.43×	62.62	81.07	95.77	57.34	98.61	85.67	89.00	95.65	49.56	77.56	3096
TAPS (λ=1.0)	3.13×	63.43	81.04	96.99	58.19	98.38	84.08	89.16	94.99	51.10	77.77	3088

5.5.2 Joint Multi-Task Learning

Setting We compare TAPS with AdaShare [230] in the joint MTL setting, where multiple tasks are learned together with (selectively) shared backbones and independent task heads. We follow the same setting as AdaShare, .i.e, datasets and network for a fair comparison. Specifically, we compare performance on the DomainNet datasets [173] with ResNet-34. This dataset contains the same labels across 6 domains and is an

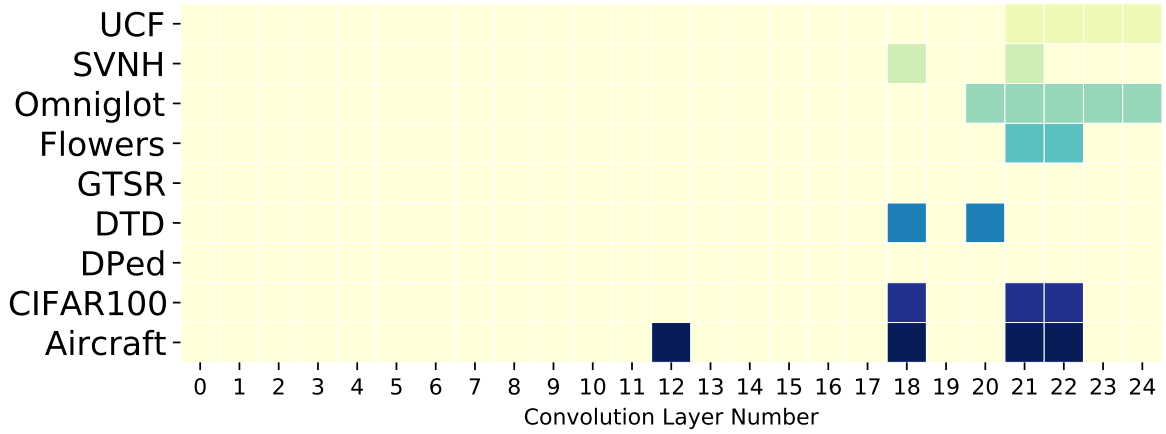


Figure 5.5: Task-specific layers for Visual Decathalon. Each row shows the task-specific layers for different datasets ($\lambda = 1.0$). For two datasets: DPed and GTSR, no task-specific parameters are needed. The performance improves solely due to updating the batch norm parameters.

Table 5.5: TAPS vs AdaShare. The accuracy of methods on the DomainNet dataset in both joint and incremental MTL settings is shown. All results are obtained with ResNet-34 unless stated otherwise. Bold numbers represent the higher accuracy between TAPS and AdaShare. Numbers with underline denote the best performing method in each setting. TAPS outperforms AdaShare in both settings. The Params column measures the total parameters for supporting all tasks in comparison with the single base model.

MTL Setting	Method	Params	Real	Painting	Quickdraw	Clipart	Infograph	Sketch	Mean
Joint	Fine-tuning	1×	75.01	66.13	54.72	75.00	36.35	65.55	62.12
	AdaShare	1×	76.90	67.90	61.17	75.88	31.52	63.96	62.88
	TAPS	1.46×	**78.91**	**67.91**	**70.18**	**76.98**	**39.30**	**67.81**	**66.84**
Incremental	Fine-tuning	6×	81.51	69.90	73.17	74.08	40.38	67.39	67.73
	AdaShare	5.73×	79.39	65.74	68.15	74.45	34.11	64.15	64.33
	AdaShare ResNet-50	4.99×	78.71	64.01	67.00	73.07	31.19	63.40	62.90
	TAPS	4.90×	**80.28**	**67.28**	**71.79**	**74.85**	**38.21**	**66.66**	**66.51**

excellent candidate for MTL learning as there are opportunities for sharing as well as task competition.

To analyze the difference between TAPS and AdaShare, we further compare them in the incremental MTL setting, where each task is learned independently ($K = 1$). We also include full fine-tuning as a baseline.

Select-or-Skip vs Add-or-Not As shown in Table 5.5, TAPS outperforms AdaShare in both the incremental and joint multi-task settings across all six domains. This may partly be due to AdaShare's select-or-skip strategy, which effectively reduces the number of residual blocks of the network. TAPS, on the other hand, replaces layers with task-specific versions without altering the model capacity, which results in performance closer to full fine-tuning. To verify whether a larger network will improve AdaShare's performance, we

perform the same incremental experiment with ResNet-50. Surprisingly, AdaShare with ResNet-50 has a slightly lower performance than its ResNet-34 version, suggesting that capacity might not be a limiting issue for the method.

Incremental Training vs Joint Training As shown in Table 5.5, full fine-tuning for each task often yields the best performance and significantly outperforms the joint fine-tuning version. This suggests that *task competition* exists among the domains and only one domain (ClipArt) benefits from the joint training. When Adashare is trained with the incremental setting, we also see an increase in performance compared to the joint fine-tuning version. However, both methods come at the cost of weights sharing among tasks and a linear increase of the total model size. For TAPS, we see the performance is relatively stable when switching from joint training to incremental training and is close to the performance of fine-tuning.

Parameter and Training Efficiency In the incremental setting, TAPS is 18% more parameter efficient compared to Adashare while performs 2.18% better on average. In this setting, AdaShare modifies the weights of existing layers and *no* layers are shared across tasks. Parameter savings here come from skipped blocks. Conversely, in the joint MTL setting, AdaShare is more parameter efficient, as no new parameters are introduced. However, this leads to performance degradation. TAPS performs uniformly better while introducing 0.46$\times$ more task-specific parameters.

As for training efficiency, AdaShare learns the select-or-skip policy first, which optimizes both weights and policy scores alternatively. After the policy learning phase, multiple architectures are sampled and retrained to get the best performance. This two-phase learning process increases the training cost significantly in comparison with TAPS, which has very little overhead compared to standard fine-tuning.

5.6 Limitations

A limitation of TAPS is that tasks do not share task-specific layers with each other, .i.e., new tasks either learn their own task-specific parameters or share with the pre-trained model. The joint training approach we propose mitigates this issue by learning parameters common to all tasks. However, tasks added incrementally still cannot share parameters with others. Retraining the network on all tasks becomes infeasible as the

number of tasks grow. We leave this aspect of parameter sharing as future work.

5.7 Conclusion

We have presented Task Adaptive Parameter Sharing, a simple method to adapt a base model to a new task by modifying a small task-specific subset of layers. We show that we are able to learn which layers to share differentiably using a straight-through estimator with gating over task-specific weight deltas. Our experimental results show that, TAPS retains high accuracy on target tasks using task-specific parameters. TAPS is agnostic to the particular architecture used, as seen in our results with ResNet-50, ResNet-34, DenseNet-121 and ViT models. We are able to discover standard and unique fine-tuning schemes. Furthermore, in the MTL setting we are able to avoid task competition by using task-specific weights.

Chapter 6

Fluid: A Unified Evaluation Framework for Flexible Sequential Data

6.1 Overview

Modern machine learning methods excel when training data is large-scale, well labeled, and matches the test distribution. Learning in less ideal conditions remains an open challenge. The sub-fields of few-shot, continual, transfer, and representation learning have made substantial strides in learning under adverse conditions, each affording distinct advantages through methods and insights. These methods address different challenges such as data arriving sequentially or scarce training examples, however often the difficult conditions an ML system will face over its lifetime cannot be anticipated prior to deployment. Therefore, general ML systems which can handle the many challenges of learning in practical settings are needed. To foster research towards the goal of general ML methods, we introduce a new unified evaluation framework – FLUID (Flexible Sequential Data). FLUID integrates the objectives of few-shot, continual, transfer, and representation learning while enabling comparison and integration of techniques across these subfields. In FLUID, a learner faces a stream of data and must make sequential predictions while choosing how to update itself, adapt quickly to novel classes, and deal with changing data distributions; while accounting for the total amount of compute. We conduct experiments on a broad set of methods which shed new insight on the advantages and limitations of current techniques and indicate new research problems to solve. As a starting point towards more general

methods, we present two new baselines which outperform other evaluated methods on FLUID. Code can be found at https://github.com/RAIVNLab/FLUID.

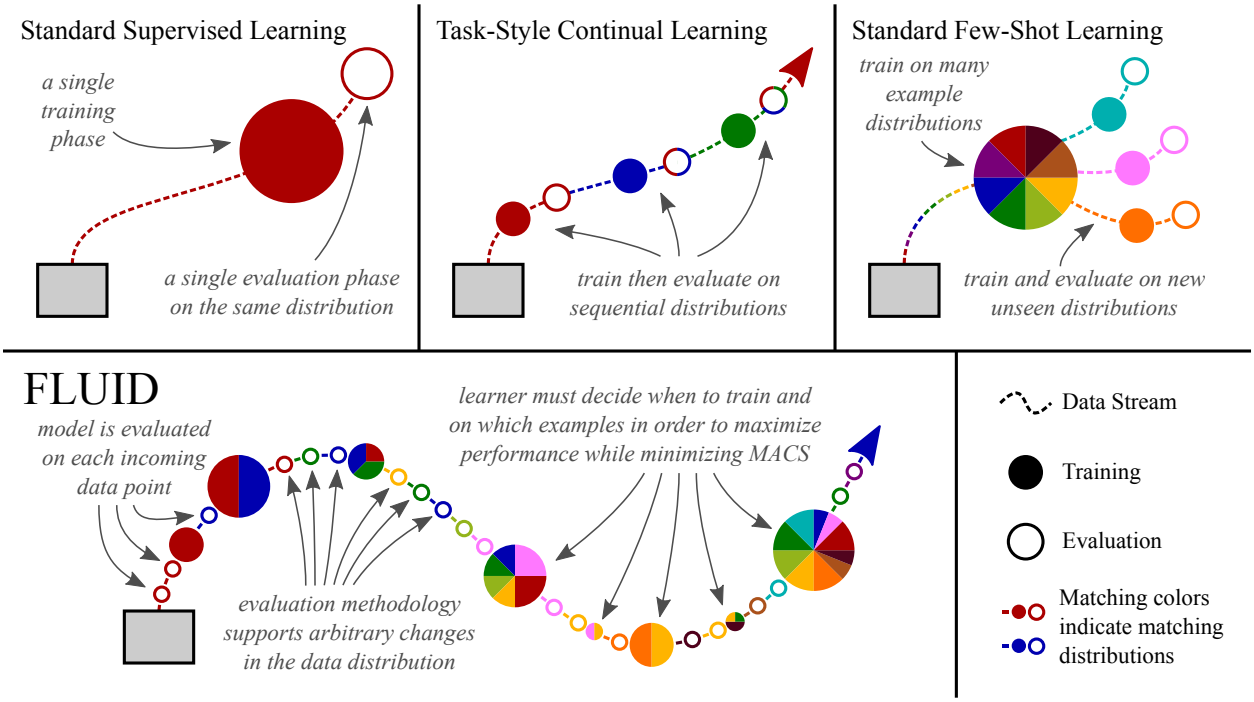


Figure 6.1: Comparison of supervised (top-left), continual (top-middle), and few-shot learning (top-right) with FLUID (bottom). The learner (grey box) accumulates data (dotted path), trains on given data (filled nodes), then is evaluated (empty nodes). The size of the node indicates the scale of the training or evaluation. Each color represents a different set of classes.

6.2 Introduction

Modern ML methods have demonstrated remarkable capabilities, particularly in settings with large-scale labeled training data drawn IID. However, in practice the learning conditions are often not so ideal. Consider a general recognition system, a key component in many computer vision applications. One would expect such a system to learn from new data distributions, recognize classes with few and many examples, revise the set of known classes as novel ones are seen, and update itself over time using new data.

Various subfields such as few-shot, continual, transfer, and representation learning have made substantial progress on the challenges associated with learning in non-ideal settings. The methods from these fields excel when the deployment conditions can be anticipated and align with specific scenarios. For example, few-shot

methods perform well when the number of new classes and examples per class are few and known in advance. Similarly, continual learning techniques improve performance when data from new distributions arrive in fixed-size batches at predictable intervals. However, in many applications the exact conditions cannot be known a priori and are likely to change over time. For example, computer vision systems for autonomous self-checkout systems must account for changing inventories, distribution shifts in background and lighting between stores, and a long-tailed distribution of products, among other real-world challenges [175, 259]. This calls for general methods that can handle a plethora of scenarios during deployment.

To make progress towards such general methods, new evaluations which reflect the key aspects of learning in practical settings are essential. But, *what are these aspects?* We posit the following as some of the necessary elements: (1) *Sequential Data* - In many application domains the data streams in. ML methods must be capable of learning from sequential data and new distributions. (2) *X-shot* - Data often has a different number of examples for each class (few for some and many for others). Current evaluations assume prior knowledge of which data regime (few-shot, many-shot, etc.) new classes will be from, but often this cannot be known in advance. (3) *Flexible Training Phases* – Practical scenarios rarely delineate when and how a system should train. ML systems should be capable of making decisions such as when to train based on incoming data, which data to train with, and whether to update all parameters or just the classifier. (4) *Compute Aware* – Real-world systems often have computational constraints not only for inference but also for training. ML systems should account for the total compute used throughout their lifetime. (5) *Open-world* - As new data is encountered the set of known classes may change over time. Learning in practical settings often entails recognizing known classes while detecting when data comes from new classes.

With these elements in consideration we introduce FLUID (**f**lexib**l**e seq**u**enti**a**l **d**ata), a unified evaluation framework. FLUID integrates the objectives of few-shot, continual, transfer, and representation learning into a simple and realizable formulation along with a benchmarkable implementation. In FLUID, a learner is deployed on a stream of data from an unknown distribution and must classify incoming samples one at a time while deciding when and how to update based on newly received data.

We conduct extensive experiments with FLUID on a broad set of methods across subfields. The experimental results quantitatively demonstrate the current limitations and capabilities of various ML approaches. For example, we find that current few-shot methods do not scale well to more classes and a varying number

of examples. Similarly, we observe that catastrophic forgetting is a significant challenge in the FLUID setting which is not solved by existing continual learning approaches. Finally, we briefly investigate the unexplored problem of update strategies for deciding when and how to train efficient on incoming data. The framework, data and models will be open-sourced.

We make the following contributions:

1. We propose a new evaluation framework, FLUID, which unifies the objectives of few-shot, continual, transfer, and self-supervised learning into a simple benchmark that enables comparison and integration of methods across related subfields and presents new research challenges.
2. We present empirical findings from experiments with FLUID which demonstrate the utility of more general evaluations. Specifically, we find that existing few-shot methods do not scale well to the FLUID setting which has more classes and varying number of examples per class. Larger networks perform better for few-shot classes, contrary to prevailing thought in few-shot works which train light-weight models to avoid overfitting [222, 55, 229, 273]. We find this discrepancy to be caused by meta-training decreasing performance for larger networks whereas supervised pretraining does not. We observe in the FLUID setting that freezing network parameters prevents catastrophic forgetting and learns novel classes better than existing continual learning methods and suggests significant room for improvement.
3. We introduce two baselines, Exemplar Tuning & Minimum Distance Thresholding, which outperform evaluated methods in FLUID while matching performance in supervised and few-shot settings.

6.3 Related Work

We discuss the key aspects of FLUID in the context of other works and evaluations. We compare existing frameworks in Table 6.1.

Sequential Data and Continual Learning New data is an inevitable consequence of our dynamic world and learning over time is a long-standing challenge [236]. In recent years, continual learning (CL) has made notable progress on the problem of learning in a sequential fashion [127, 107, 190, 4, 5, 195]. Several setups have been proposed in order to evaluate systems' abilities to learn continuously and primarily focus on *catastrophic forgetting*, a phenomenon where models drastically lose accuracy on old tasks when trained

on new tasks. The typical CL setup sequentially presents data from each task then evaluates on current and previous tasks [127, 107, 190]. Recent variants have proposed a task-free setting where the data distribution changes without the model's knowledge [71, 195, 73, 264, 232].

Recently several works have empirically investigated the disconnect between existing continual learning evaluations and real-world applications [177, 86, 1]. Such works show that implicit details in the experimental setups can lead to significantly different methods and empirical conclusions. For example, [177] show that not accounting for compute allows for an unrealistic method which retrains from scratch to drastically outperform sophisticated CL methods. In FLUID, we aim to address unrealistic assumptions and carefully consider each detail of the experimental design.

There are two assumptions in existing CL evaluations which we remove in FLUID. The first assumption is that data will be received in large batches with ample data for every class in the task. This circumvents a fundamental challenge of sequential data which is learning new classes from only a few examples. Consider the common scenario in which a learner encounters an instance from a novel class. The system must determine that it belongs to a new class with no previous examples (zero-shot learning and out-of-distribution detection). The next time an instance from the category appears, the system must be capable of one-shot learning, and so forth. In other words, few-shot learning is an emergent requirement of learning from sequential data. The second assumption is that the training and testing phases will be delineated to the system. Deciding when to train and which data to train on is an intrinsic challenge of learning continuously.

Some CL evaluations include a memory cache to store images, typically between 0 - 1000 examples from previous tasks. We argue that setting a specific memory constraint, particularly this small, is too constraining. Methods should account for memory, but the FLUID framework does not explicitly restrict memory during streaming. Note that all methods use memory caching during the sequential phase except for the Nearest Class Mean (NCM) baseline.

Data stream classification [62, 258, 226, 16] has worked on the problem of learning from sequential data. This line of work primarily focuses on traditional classification and not image recognition. At a high level FLUID has similar goals as data stream classification. FLUID differs in its implementation while integrating the preexisting fields of few-shot, transfer, representation, and continual learning.

Most closely related to FLUID is the OSAKA benchmark [25]. Caccia et al. [25] propose a more

Table 6.1: We categorize existing evaluations frameworks aimed at learning in practical settings. ✓: presence; ✗: absence & –: not applicable.

FrameworkProperty	Open-world	Sequential	Variable Batch-size	Few-shot	Many-shot	Compute Aware	Memory Constrained	Flexible Training	Distribution Shift	Non-Stationary
Representation Learning	✗	✗	–	✗	✓	–	–	–	✓	–
Transfer Learning	✗	✗	–	✗	✓	–	–	–	✓	–
Task-based Continual Learning	✗	✓	✗	✗	✓	–	✓	✗	✓	✓
Task-free Continual Learning	✗	✓	✗	✗	✓	–	✓	✗	✓	✓
Few-shot Learning	✗	✗	–	✓	✗	–	–	–	✓	–
Generalized Few-shot Learning	✗	✗	–	✓	✓	–	–	–	✓	–
Streaming Perception	✗	✓	–	–	–	✓	–	–	✗	✗
Open Long-Tailed Recognition	✓	✗	–	✓	✓	–	–	–	✗	✗
Data Stream Classification	–	✓	✓	–	✓	–	✓	–	✓	–
Test Time Training	✗	✓	✓	–	–	–	–	✗	✓	✗
OSAKA	✓	✓	✓	✓	✓	–	–	✓	✓	✓
FLUID **(Ours)**	✓	✓	✓	✓	✓	✓	–	✓	✓	✗

general scenario which unifies meta-learning, meta-continual learning, and continual-meta learning and addresses limitations of the previous evaluations. FLUID builds upon this direction of research with a few key differences. First, FLUID accounts for compute consumed throughout training, which is important metric to consider when flexible training phases are allowed. Second, FLUID samples from a long-tail distribution and evaluates accuracy with respect to class frequency (head and tail class accuracy). This allows us to study the trade-off between meta-learning methods which excel in the few-shot regime and methods which are better with large-scale data. Last, we conduct the experiments at the ImageNet scale. We find that the larger-scale setting leads to new empirical findings such as meta-training not scaling to larger networks and more data.

One key distinction that OSAKA and other CL frameworks incorporate that is not included in FLUID is a non-stationary distribution. In general, FLUID differs from traditional continual learning formulations [244, 37] in that the goal is to perform well on the deployment distribution, rather than current and previously seen distributions. In FLUID the system adapts to one unknown distribution shift, whereas traditional CL frameworks change distributions over multiple episodes.

Few-shot and X-shot Learning Learning from few examples for some classes is an inherent aspect of the real-world. Learning from large, uniform datasets [201, 130] has been the primary focus of supervised learning while few-shot learning has gained traction as a subfield [187, 70, 169, 229].

While few-shot learning is a step towards more generally applicable ML methods, the framework has assumptions that are unlikely to hold in practical settings. The experimental setup for few-shot is typically the n-shot k-way evaluation. Models are trained on base classes during *meta-training* and then tested on novel classes during *meta-testing*. The n-shot k-way experimental setup is limited in two respects. n-shot k-way

assumes that a model will always be given exactly n examples for k classes at test time which is unrealistic. Second, most works only evaluate 5-way scenarios with 1, 5, and 10 shots. Realistic settings often have a mix of classes from both the high and low data regime. Recently, more general variants of the few-shot benchmark have been proposed which introduce variable shot numbers and greater domain difference between datasets [28, 239, 46].

FLUID naturally integrates the few-shot problem into its framework by sequentially presenting data from a long tail distribution and evaluates systems across a spectrum of shots and ways. Our experimental results on canonical few-shot methods indicate that methods are overly tuned to the specific conditions of the few-shot evaluation which indicates the need for more general experimental frameworks such as FLUID.

Flexible Training Phases Current experimental setups dictate when models will be trained and tested. Ideally, an ML system should decide when to train itself, what data to train on, and what to optimize for [33]. By removing the assumption that training and testing phases are fixed and known in advance, FLUID provides a benchmark for tackling the unexplored challenge of learning when to train.

Compute Aware ML systems capable of adapting to their environment over time must account for the computational costs of their learning strategies as well as of inference. Prabhu et al. [177] showed that current CL frameworks do not measure total compute and therefore a naive, but compute-hungry strategy can drastically outperform state of the art methods. Previous works have focused on efficient inference [186, 83, 117, 118, 119, 132, 252] and some on training costs [50]. In FLUID we measure the total compute for both learning and inference over the sequence.

Open-world Practical scenarios entail inferring in an open world - where the classes and number of classes are unknown to the learner. Few-shot, continual, and traditional supervised learning setups assume that test samples can only come from known classes. Previous works explored static open-world recognition [136, 13, 111, 248, 182] and the related problem of out-of-distribution detection [78, 148, 124]. FLUID is a natural integration of sequential and open-world learning where the learner must identify new classes and update its known class set throughout the stream.

6.4 FLUID Evaluation Details

FLUID evaluation is designed to be simple and general while integrating the key aspects outlined in section 6.3.

Formulation Let learning system S be composed of a model, $f_\theta : \mathbf{x} \mapsto \mathbf{y}$, and update strategy, $U : f_\theta \times \bigcup_{t=1}^{T}(x_t, y_t) \mapsto f_{\theta'}$, where $\bigcup_{t=1}^{T}(x_t, y_t)$ is the training data collected up to time T. Model, f_θ, may be initialized using pretraining data $D = \{x_i, y_i\}_{i=1}^{n}$.

At each new time step, $t+1$, the model is given a sample, x_t, and provides a class label, $f_\theta(x_{t+1}) \in \{1...K+1\}$, for K known classes. In other words, the sample may belong to one of K previously seen classes, or a new class. The model output is evaluated with respect to the true label, $1\{f_\theta(x_{t+1}) = y_{t+1}\}$, and (x_{t+1}, y_{t+1}) is added to the training set. If y_{t+1} is from a new class, the set of known classes is updated accordingly. Next the model, f_θ, may be updated according to U using all previously observed data. This process is repeated for some total number of time steps.

Systems are evaluated on a suite of metrics including the overall and mean class accuracy throughout the stream along with the total compute required for updates and inference.

Data In this paper, we evaluate methods with FLUID using a subset of ImageNet-22K [43]. Traditionally, few-shot learning used datasets like Omniglot [120] & MiniImagenet [249] and continual learning focused on MNIST [123] & CIFAR [115]. Some recent continual learning works have used Split-ImageNet [260]. The aforementioned datasets are mostly small-scale and have very few classes. We evaluate on the ImageNet-22K

Table 6.2: The evaluation metrics used in the FLUID framework to capture the performance and capabilities of various algorithms.

Metric	Description
Overall Accuracy	Accuracy over the sequence.
Mean Per-Class Accuracy	Accuracy averaged over all classes in the sequence.
Total Compute	MAC operations for all compute over the sequence.
Unseen Class Detection	AUROC for the detection of OOD samples.
Cross-Sectional Accuracies	Classes in the sequence belong fall into 4 subcategories: 1) *Pretraining-Head*: > 50 samples & in pretraining. 2) *Pretraining-Tail*: ≤ 50 samples & in pretraining. 3) *Novel-Head*: > 50 samples & not in pretraining. 4) *Novel-Tail*: ≤ 50 samples & not in pretraining.

dataset to present new challenges to existing models. Recently, the INaturalist [246, 261] and LVIS [68] datasets have advocated for heavy-tailed distributions. We follow suit and draw our sequences from a heavy-tailed distribution.

The dataset consists of a pretraining dataset and 5 different sequences of images for streaming (3 test and 2 validation sequences). For pretraining we use the standard ImageNet-1K [201]. This allows us to leverage existing models built by the community as pre-trained checkpoints. Sequence images come from ImageNet-22K after removing ImageNet-1K's images. Each test sequence contains images from 1000 different classes, 750 of which do not appear in ImageNet-1K. We refer to the overlapping 250 classes as Pretrain classes and the remaining 750 as Novel classes. Each sequence is constructed by randomly sampling images from a heavy-tailed distribution of these 1000 classes. Each sequence contains ~ 90000 samples, where head classes contain > 50 and tail classes contain ≤ 50 samples. The sequence allows us to study how methods perform on combinations of pretrain vs novel, and head vs tail classes. **Pretraining** Supervised pretraining [75] on large annotated datasets like ImageNet facilitates the transfer of learnt representations to help data-scarce downstream tasks. Unsupervised learning methods like autoencoders [240] and more recent self-supervised methods [95, 180, 63] like Momentum Contrast (MoCo) [77] and SimCLR [31] have begun to produce representations as rich as that of supervised learning and achieve similar accuracy on various downstream tasks.

Before the sequential phase, we pretrain our model on ImageNet-1K. In our experiments, we compare how different pretraining strategies (contrastive learning, meta-training, & supervised training) perform under more adverse conditions. We find new insights such as contrastive representations perform significantly worse on few-shot classes compared to their supervised counterparts in the FLUID evaluation.

Evaluation metrics Table 6.2 defines the evaluation metrics in FLUID to gauge the performance of the algorithms.

6.5 Baselines and Methods

In this section, we summarize the baselines, other methods, and our proposed baselines, Exemplar Tuning and MDT.

Standard Training and Fine-Tuning We evaluate standard model training (update all parameters in the

network) and fine-tuning (update only the final linear classifier) with offline batch training.

Nearest Class Mean (NCM) Recently, multiple works [237, 256] have found that Nearest Class Mean (NCM) is comparable to state-of-the-art few-shot methods [229, 169]. NCM in the context of deep learning performs a 1-nearest neighbor search in feature space with the centroid of each class as a neighbor. We pretrain a neural network with a linear classifier using softmax cross-entropy loss, then freeze the parameters to obtain features.

Few-shot Methods We evaluate the following methods: MAML [55], Prototypical Networks (PTN) [222], Weight Imprinting [181], ProtoMAML [239], SimpleCNAPS [10], New Meta-Baseline [32] and ConstellationNet [273].

PTN trains a deep feature embedding using 1-nearest neighbor with class centroids and soft nearest neighbor loss. Parameters are trained with meta-training and backprop.

MAML is a gradient-based approach which uses second-order optimization to learn parameters that can be quickly adapted to a given task.

Weight Imprinting initializes the weights of a cosine classifier as the class centroids, then fine-tunes with a learnable temperature. Meta-Baseline is same in implementation as NCM except a phase of meta-training is done after standard batch training.

Continual Learning (CL) Methods We evaluate Learning without Forgetting (LwF) [127], Elastic Weight Consolidation (EWC) [107], Dark Experience Replay(DER) [23], & Experience Replay with Asymmetric Cross-Entropy (ER-ACE) [24] to observe whether continual learning techniques improve performance in FLUID.

LwF leverages knowledge distillation [21] to retain accuracy on previous training data without storing it. EWC enables CL in a supervised learning context by penalizing the total distance moved by the parameters from the optimal model of previous tasks weighted by the corresponding Fisher information. Unlike LwF, EWC requires stored data, typically the validation set, from the previous tasks. In FLUID, we use LwF and EWC to retain performance on pretrain classes. For the continual learning methods we train all network parameters according to standard training procedures.

Out-of-Distribution (OOD) Methods We evaluate two methods proposed by Hendrycks & Gimpel [78] (HG) and OLTR [136] along with our proposed OOD baseline. The HG baseline thresholds the maximum

probability output of the softmax classifier to determine whether a sample is OOD.

We propose the baseline, **Minimum Distance Thresholding (MDT)**, which utilizes the minimum distance from the sample to all class representations, c_i. In the case of NCM the class representation is the class mean and for a linear layer it is the ith column vector. For distance function d and a threshold τ, a sample is out of distribution if: $\mathbf{I}\left(\min_i d\left(c_i, \mathbf{x}\right) < t\right)$. MDT with a Nearest Class Mean classifier can be derived from a Dirichlet process mixture model [82], where a sample is considered to be out of distribution if it is assigned to a new cluster. The concentration parameter for the Chinese Restaurant process can be related to the out of distribution threshold τ as:

$$\tau = 2\sigma \log\left(\frac{\alpha}{\left(1+\frac{\rho}{\sigma}\right)^{d/2}}\right)$$

ρ is the covariance scaling of the gaussian prior over the cluster means, $\boldsymbol{\mu} \sim \mathcal{N}(\mathbf{0}, \rho I)$ and σ is the scaling for the isotropic covariance of each gaussian cluster, $\mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{z}_i}, \sigma I\right)$. Similar to the DP-Means derivation, as σ goes to 0, the probability of a sample x being assigned to a new cluster goes to 1 when the distance of x to the closest cluster exceeds τ.

Other metric learning techniques have proposed using distance to detect out of distribution examples [124, 148]. MDT primarily differs from these works in that it can be used with a standard classification network and can be performed in a single forward pass with negligible extra compute.

Exemplar Tuning (ET) We present a new baseline that leverages the inductive biases of instance-based methods and parametric deep learning. The traditional classification layer is effective when given a large number of examples but performs poorly when only a few examples are present. On the other hand, NCM and other few-shot methods are accurate in the low data regime but do not significantly improve when more data is added. Exemplar Tuning (ET) synthesizes these methods in order to initialize class representations accurately when learning new classes and to have the capacity to improve when presented with more data. We formulate each class representation (classifier), C_i, and class probability as the following:

$$C_i = \frac{1}{n}\sum_{x \in D_i} \frac{f(x;\theta)}{\|f(x;\theta)\|} + \mathbf{r}_i; \quad p(y=i|x) = \frac{e^{C_i \cdot f(x;\theta)}}{\sum_{i \neq j} e^{C_j \cdot f(x;\theta)}} \tag{6.1}$$

where $f(x;\theta)$ is a parametrized neural network, $\mathbf{r}_i$ is a learnable residual, n is the number of class examples, and D_i are all examples in class i. C_i is analogous to the i-th column vector in a linear classification layer.

The class centroid (the first term of C_i in Eq 6.1) provides an accurate initialization from which the residual term $\mathbf{r}_i$ can continue to learn. Thus ET is accurate for classes with few examples (where deep parametric models are inaccurate) and continues to improve for classes with more examples (where few-shot methods are lacking). In our experiments, the centroid is updated after each sample for minimal compute and batch train the residual vector with cross-entropy loss according to the same schedule as fine-tuning.

We compare ET to initializing a cosine classifier with class centroids and fine-tuning (Weight Imprinting). Exemplar Tuning outperforms Weight Imprinting and affords two significant advantages besides better accuracy. 1) ET has two frequencies of updates (fast instance-based and slow gradient-based) which allows the method to quickly adapt to distribution shifts while providing the capacity to improve over a long time horizon. 2) ET automatically balances between few-shot & many-shot performance, unlike Weight Imprinting which requires apriori knowledge of when to switch from centroid-based classification to fine-tuning.

6.6 Experiments and Analysis

We evaluate an array of methods from few-shot, continual, self-supervised learning, and out-of-distribution detection in the proposed FLUID framework. We present a broad set of empirical findings which validate the need for more general evaluations such as FLUID and suggest future research directions. Table 6.3 displays a comprehensive set of metrics for the set of methods (outlined in Sec 6.5). Throughout this section, we will refer to rows of the table for specific analysis. For a summary of the main insights see Sec 6.6.7.

6.6.1 Few-shot Analysis

We evaluate three groups of methods to understand the effects of meta-training on generalization to novel classes and to gauge the overall utility of current few-shot methods in FLUID. The first group consists of methods which are purely meta-trained (Prototypical Networks and MAML). The second group consists of methods that do not utilize meta-training (Weight Imprinting, NCM, and fine-tuning). Finally, we evaluate methods which use both meta-training and standard batch training (ProtoMAML, SimpleCNAPS,

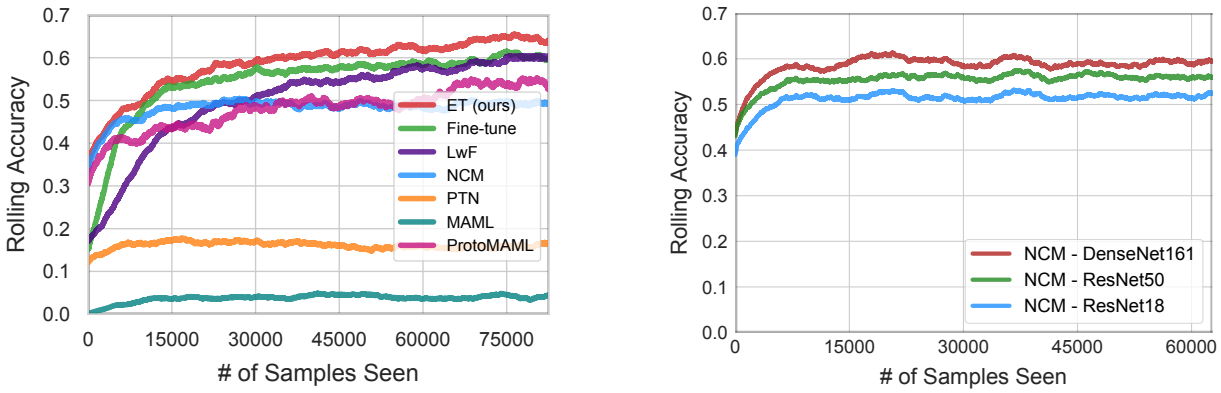
Table 6.3: Performance of the suite of methods (outlined in Section 6.5). We present several accuracy metrics - Overall, Mean-per-class as well as accuracy bucketed into 4 categories: Novel-Head, Novel-Tail, Pretrain-Head and Pretrain-Tail (Pretrain refers to classes present in the ImageNet-1K dataset).

	Method	Pretrain Strategy	Novel - Head (>50)	Pretrain - Head (>50)	Novel - Tail (<50)	Pretrain - Tail (<50)	**Mean Per-Class**	**Overall**	GMACs↓ ($\times 10^6$)
	Backbone - Conv-4								
FSL	(a) Prototypical Networks	Meta	11.63	22.03	6.90	13.26	11.13	15.98	**0.06**
FSL	(b) MAML	Meta	2.86	2.02	0.15	0.10	1.10	3.64	2.20
	Backbone - ResNet18								
FSL	(c) Prototypical Networks	Meta	8.64	16.98	6.79	12.74	9.50	11.14	0.15
FSL	(d) ConstellationNet	Sup+Meta.	39.26	65.35	26.28	52.91	40.21	46.13	0.16
FSL	(e) Meta-Baseline	Sup.+Meta	40.47	67.03	27.53	53.87	40.23	47.62	0.16 / 5.73
FSL	(f) ProtoMAML	Sup.+Meta	41.68	69.48	28.91	54.16	42.94	49.25	5.73
FSL	(g) SimpleCNAPS	Sup.+Meta	41.59	68.79	25.79	52.16	41.82	48.23	0.16
FSL	(h) Weight Imprinting	Sup.	40.32	67.46	15.35	34.18	32.69	48.51	0.16 / 5.73
FSL	(i) OLTR	Sup.	40.83	40.00	17.27	13.85	27.77	45.06	0.16 / 6.39
CL	(j) LwF	Sup.	30.07	67.50	7.23	56.96	31.02	48.76	22.58 / 45.16
CL	(k) EWC	Sup.	39.03	70.84	16.59	47.18	34.89	50.39	12.29
CL	(l) DER	Sup.	35.34	74.41	10.64	52.05	32.30	49.07	11.29
CL	(m) ER-ACE	Sup.	33.13	69.61	16.59	49.00	31.34	44.50	12.29
Baselines	(n) Fine-tune	Sup.	43.41	**77.29**	23.56	**58.77**	41.54	53.80	0.16 / 5.73
Baselines	(o) Standard Training	Sup.	38.51	68.14	16.90	43.25	33.99	49.46	11.29
Baselines	(p) NCM	Sup.	42.35	72.69	**31.72**	56.17	43.44	50.62	**0.15**
Baselines	(q) Exemplar Tuning **(Ours)**	Sup.	**48.85**	75.70	27.93	45.73	**43.61**	**58.16**	0.16 / 5.73
MoCo	(r) Weight Imprinting	MoCo	16.77	26.98	6.19	8.69	12.60	22.90	0.16 / 5.73
MoCo	(s) OLTR	MoCo	34.60	33.74	13.38	9.38	22.68	39.92	0.16 / 6.39
MoCo	(t) Fine-tune	MoCo	14.49	27.59	0.10	4.96	8.91	26.86	0.16 / 5.73
MoCo	(u) Standard Training	MoCo	26.63	45.02	9.63	20.54	21.12	35.60	11.29
MoCo	(v) NCM	MoCo	19.24	31.12	14.40	21.95	18.99	22.90	0.15
MoCo	(w) Exemplar Tuning **(Ours)**	MoCo	31.50	46.21	12.90	21.10	24.36	39.61	0.16 / 5.73

ConstellationNet, Meta-Baseline).

We observe the methods that purely meta-train (PTN and MAML) do not perform well in the large-scale FLUID setting with over 30% lower overall accuracy than the NCM baseline (Table 6.3 and Figure 6.2-a). One might argue that PTN and MAML could simply scale to the larger setting by increasing model capacity. However, few-shot works indicate that training with deeper and overparameterized networks decrease performance [229, 169, 222, 55]. We verify this observation, noting that the 4-layer convnet PTN (Table 6.3-a) outperforms the ResNet18 PTN in overall and novel class accuracy.

The prevailing thought in few-shot literature has been that smaller networks overfit less to base classes, and therefore methods use shallow networks or develop techniques to constrain deeper ones. We find evidence to the contrary, that deeper networks generalize better to novel classes when using standard batch sampling



(a) (b)

Figure 6.2: (a) Compares the accuracy of various methods over the stream of data. (b) Compares the accuracy of NCM on novel classes across network architectures. Contrary to prevailing thought, we find that deeper networks generalize better to novel few-shot classes.

(see Figure 6.2-b). Given that NCM and PTN differ only in the use of meta-training, the experiments indicate that meta-training is responsible for the lower performance of deeper networks. This evidence is further reinforced by the fact that Meta-Baseline performs worse than NCM with the inclusion of meta-training (Table 6.3-e,p).

For the more recent few-shot methods (ProtoMAML, Meta-Baseline, ConstellationNet) that utilize a combination of meta-training and mini-batch training we observe significantly better results compared to purely meta-trained methods. However, we find that these recent few-shot methods are outperformed by the NCM baseline for novel, pretrain, tail and head classes (Table 6.3) indicating that meta-training in its current form reduces performance in settings with varying number of examples.

Typically meta-training is performed with a fixed number of classes (way) and number of examples (shot). In FLUID the number of examples and classes are changing, therefore standard meta-training may not be suitable to match the test conditions. We conduct experiments to observe whether changing the meta-training procedure to better reflect the test conditions improves performance of the model. We ablate over the way and shot number as well as randomly sample the shot and way throughout training. We find that changing the shot hyper-parameter changes which part of the distribution the model performs well on (tail and head), but the overall accuracy does not significantly improve.

6.6.2 Continual Learning Analysis

We evaluate Learning without Forgetting (LwF), Elastic Weight Consolidation (EWC), Dark Experience Replay (DER), and Experience Replay with Asymmetric Cross Entropy (ER-ACE). For analysis, we compare to the baselines NCM, standard training, and fine-tuning. LwF and EWC are prominent CL methods while DER and ER-ACE are recent methods with state-of-the-art performance.

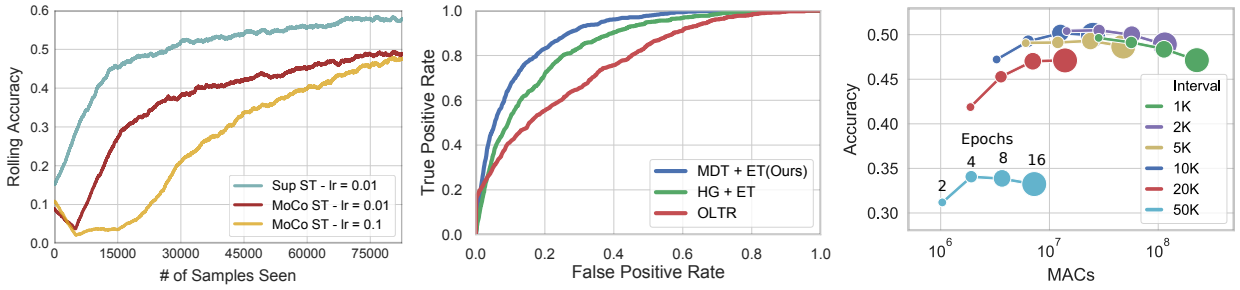
We find that catastrophic forgetting of the pretrain parameters is a significant challenge in the FLUID evaluation. Standard training of the parameters on the sequential data degrades not only the accuracy for pretrain classes, but also for novel classes compared to CL methods and freezing network parameters (table 3). We hypothesize that large-scale pretraining provides better features even for novel classes compared to training on the smaller sequential data set. A similar observation was made by [72] in a CL setting. Further evidence for this can be found in the update strategies section where we observe that standard training on sequential data for too many epochs reduces overall accuracy (Section 6.6.6 - Figure 6.3).

Our experiments show that freezing the feature extractor (NCM and fine-tuning) is more effective in preventing catastrophic forgetting than existing CL methods. Specifically, NCM obtains $\sim 8\%$ (Table 6.3 j-m) higher mean-per-class accuracy compared to existing CL methods and fine-tuning obtains $\sim 4\%$ higher overall accuracy. This result indicates that there is significant room for progress in reducing forgetting in the FLUID setting and motivates the need for new evaluations which more closely model the challenges faced by real-world ML systems. For scenarios in which the pretraining data is radically different from the target distribution the above conclusion may not hold, such as for permuted MNIST.

The notable differences between FLUID and standard CL formulations are the inclusion of pretraining, flexible training, one distribution shift rather than multiple, and measuring performance only on the deployment distribution. We contend pretraining is a reasonable inclusion as real-world vision systems have access to large datasets such as ImageNet, though some scenarios, especially those outside the domain of computer vision, may not afford pretraining.

6.6.3 Exemplar Tuning

We find that ET (Table 6.3-w) has significantly higher overall and mean-class accuracy than other evaluated methods and uses similar compute as fine-tuning. Figure 6.2-a shows how ET quickly adapts to new classes



(a) (b) (c)

Figure 6.3: (a) Accuracy of standard training with MoCo & supervised pretraining. Unexpectedly, MoCo accuracy falls during the initial streaming phase. (b) ROC curves for unseen class detection. MDT outperforms all OOD baselines evaluated in FLUID. (c) Standard training accuracy curve for a range of training frequencies & epochs showing that over training can lead to lower accuracy. MACs $\propto$ total gradient updates.

and continues to learn in the standard data regime (high accuracy at the start and end of the stream). Finally, we show that ET outperforms simple NCM + fine-tuning (Weight Imprinting) by $\sim$10%, in addition to the practical advantages outlined in section 6.5.

6.6.4 Novel Class Detection and MDT

We measure AUROC for detecting new classes throughout the sequence and present in Figure 6.3-b. HG baseline + ET, OLTR, and MDT + ET achieve 0.84, 0.78 and 0.92 AUROC scores respectively. The performance of Minimum Distance Thresholding (MDT) indicates that standard recognition networks are well suited for detecting out-of-distribution classes and can be done simultaneously with classification.

6.6.5 Representation Learning Analysis

We observe unexpected behavior from contrastive MoCo [77] and VINCE [63] representations in the FLUID setting. For fine-tuning the classifier of a MoCo representation, we find that accuracy is less than 1% and 4.96% on novel and pretrain tail classes respectively (Table 6.3-t). In comparison the supervised counterpart (Table 6.3-n) obtains 23.56% and 58.77% accuracy respectively. We conclude that this difficulty is due to learning the linear classifier because NCM with MoCo (Table 6.3-v) does not exhibit the same drop in performance. Figure 6.3-a shows other unexpected behavior in which MoCo accuracy drastically decreases initially when standard training, then begins improving after 10K samples. This behavior is not

observed for supervised pretraining and occurs for a range of learning rates. We argue that this is related to learning a mixture of pretrain and novel classes which is the primary difference between FLUID and previous downstream tasks. The significantly lower accuracy of MoCo representations on novel-tail classes while fine-tuning (Table 6.3-t) further reinforces this hypothesis. These observations and insights are also observed for VINCE [63], a similar contrastive method. These results validate the utility of an evaluation such as FLUID which assess the capabilities of methods more generally.

6.6.6 Update Strategies

We investigate trade-offs between compute cost and accuracy of simple update strategies and leave the challenges of learning adaptive update strategies in FLUID as future work.

For update strategies, we ablate over the frequencies and number of training epochs per update (Figure 6.3-c) and measure the trade-off in accuracy and total compute cost. We conduct these experiments for fine-tuning and for standard training (Figure 6.3-c) on ResNet18 model with supervised pretraining.

We observe that training for too many total epochs (training frequency $\times$ epochs) with standard training (Figure 6.3-c) decreases overall accuracy, though for fine-tuning accuracy asymptotically improves. We hypothesize that the optimal amount of training balances the features learned from ImageNet-1K with those from the smaller, imbalanced streaming data. This aligns with our continual learning experiments that indicate large-scale pretrained features trained on more data outperform specialized features. These initial experiments are intended to illustrate the new problems that FLUID presents for future research. The results indicate that there is significant room for improvement in both efficiency and accuracy with new strategies for training networks under streaming conditions which we leave for future work.

6.6.7 Summary of Insights

1. Representative few-shot methods do not scale well to FLUID with a varying number of examples per class and more classes. **Supporting Evidence:** Prototypical Networks (PTN) and MAML perform 30% worse in overall accuracy compared to baselines of fine-tuning and nearest class mean (NCM) (Table 6.3 a-b, n-p). Recent state-of-the-art methods ProtoMAML, SimpleCNAPS, and ConstellationNet are outperformed by the baseline NCM for both novel and pretrain classes.

2. The current formulation of meta-training inhibits few-shot methods from scaling to more data and larger architectures in the FLUID setting. **Supporting Evidence:** PTN decreases in all accuracy metrics when increasing architecture size from Conv-4 to ResNet18 (Table 6.3 a, c) while NCM increases in all accuracy metrics with larger models (Fig 2b). PTN & NCM differ only in that PTN uses meta-training while NCM uses standard batch training.

3. Catastrophic forgetting is a significant challenge in the FLUID setting which is not solved by existing continual learning approaches and large-scale pretraining changes the types of methods which are effective for preventing forgetting. **Supporting Evidence:** Freezing the network parameters (Fine-tuning and NCM) obtains higher accuracy on novel and pretrain classes compared to all evaluated CL methods (Table 6.3).

4. Contrastive self-supervised representations perform significantly worse on novel classes compared to those of supervised when learning from a mix of novel and pretrain classes in FLUID. **Supporting Evidence:** Fine-tuning from MoCo (Table 6.3) obtain 0.1% and 1.69% accuracy on novel tail classes. Supervised fine-tuning obtains 23.56% on the same cross-section of data. This drastic gap between supervised & MoCo representation is absent in the original work by He et al. [77] when fine-tuning to COCO and other downstream tasks.

6.7 Limitations and Future Work

Throughout this paper, we studied various methods and settings in the context of supervised image classification, a highly explored problem in ML. While we do not make design decisions specific to image classification, incorporating other mainstream tasks into FLUID is a potential next step. Also while the FLUID framework is agnostic to any particular data set, our experiments and conclusions are anchored in the computer vision domain. Across the experiments in this paper, we impose some assumptions about the learning conditions, albeit only a few, on FLUID. For example, we currently assume that FLUID has access to labels as the data streams in. One exciting future direction is to add the semi- or un-supervised aspects to FLUID. Relaxing these remaining assumptions to bring FLUID closer to real world conditions is an interesting future direction.

6.8 Conclusions

We introduce FLUID, a unified evaluation framework designed to facilitate research towards more general methods capable of handling the challenges of learning in deployment settings. FLUID enables comparison and integration of solutions across few-shot, transfer, continual and representation learning, & out-of-distribution detection while introducing new research challenges like how and when to update model parameters based on incoming data. Through our experiments with FLUID on a wide-range of methods we show the limitations and merits of existing solutions. For example, few-shot methods do not scale well to settings with more classes and varying number of examples and freezing network parameters prevents catastrophic forgetting better than representative continual learning methods when in the FLUID setting. As a starting point for solving the new challenges, we present two baselines, Exemplar Tuning & Minimum Distance Thresholding, which outperform existing methods on FLUID.

Bibliography

[1] Online continual learning for progressive distribution shift (ocl-pds): A practitioner's perspective. 2022.

[2] H. Aanæs, R. R. Jensen, G. Vogiatzis, E. Tola, and A. B. Dahl. Large-scale data for multiple-view stereopsis. *International Journal of Computer Vision*, pages 1–16, 2016.

[3] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *Communications of the ACM*, 54:105–112, 2011.

[4] R. Aljundi, M. Rohrbach, and T. Tuytelaars. Selfless sequential learning. *arXiv preprint arXiv:1806.05421*, 2018.

[5] R. Aljundi, K. Kelchtermans, and T. Tuytelaars. Task-free continual learning. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019.

[6] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.

[7] A. Barbu, D. Mayo, J. Alverio, W. Luo, C. Wang, D. Gutfreund, J. Tenenbaum, and B. Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. *Advances in neural information processing systems*, 32, 2019.

[8] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021.

[9] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022.

[10] P. Bateni, R. Goyal, V. Masrani, F. Wood, and L. Sigal. Improved few-shot visual classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14493–14502, 2020.

[11] D. Batra, A. Gokaslan, A. Kembhavi, O. Maksymets, R. Mottaghi, M. Savva, A. Toshev, and E. Wijmans. Objectnav revisited: On evaluation of embodied agents navigating to objects. *arXiv preprint arXiv:2006.13171*, 2020.

[12] R. Beaumont. *Clip Retrieval*, 2021. URL https://github.com/rom1504/clip-retrieval.

[13] A. Bendale and T. Boult. Towards open world recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1893–1902, 2015.

[14] Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[15] R. Berriel, S. Lathuillere, M. Nabi, T. Klein, T. Oliveira-Santos, N. Sebe, and E. Ricci. Budget-aware adapters for multi-domain learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[16] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. Moa: Massive online analysis. *Journal of Machine Learning Research*, 11:1601–1604, 2010.

[17] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. B. Van Den Driessche, J.-B. Lespiau, B. Damoc, A. Clark, et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR, 2022.

[18] E. Brachmann, J. Wynn, S. Chen, T. Cavallari, Á. Monszpart, D. Turmukhambetov, and V. A. Prisacariu. Scene coordinate reconstruction: Posing of image collections via incremental learning of a relocalizer. *arXiv preprint arXiv:2404.14351*, 2024.

[19] G. Brod, M. Werkle-Bergner, and Y. L. Shing. The influence of prior knowledge on memory: a developmental cognitive neuroscience perspective. *Frontiers in behavioral neuroscience*, 7:139, 2013.

[20] A. L. Brown and M. J. Kane. Preschool children can learn to transfer: Learning to learn and learning from example. *Cognitive psychology*, 20(4):493–523, 1988.

[21] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.

[22] C. P. Burgess, L. Matthey, N. Watters, R. Kabra, I. Higgins, M. Botvinick, and A. Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

[23] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33: 15920–15930, 2020.

[24] L. Caccia, R. Aljundi, N. Asadi, T. Tuytelaars, J. Pineau, and E. Belilovsky. New insights on reducing abrupt representation change in online continual learning. In *International Conference on Learning Representations*, 2022.

[25] M. Caccia, P. Rodriguez, O. Ostapenko, F. Normandin, M. Lin, L. Caccia, I. Laradji, I. Rish, A. Lacoste, D. Vazquez, et al. Online fast adaptation and knowledge accumulation: a new approach to continual learning. *arXiv preprint arXiv:2003.05856*, 2020.

[26] E. R. Chan, K. Nagano, M. A. Chan, A. W. Bergman, J. J. Park, A. Levy, M. Aittala, S. D. Mello, T. Karras, and G. Wetzstein. GeNVS: Generative novel view synthesis with 3D-aware diffusion models. In *ICCV*, 2023.

[27] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. ShapeNet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[28] W.-L. Chao, S. Changpinyo, B. Gong, and F. Sha. An empirical study and analysis of generalized

zero-shot learning for object recognition in the wild. In *European conference on computer vision*, pages 52–68. Springer, 2016.

[29] O. Chapelle, B. Schölkopf, and A. Zien. A discussion of semi-supervised learning and transduction. In *Semi-supervised learning*, pages 473–478. MIT Press, 2006.

[30] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny. Efficient lifelong learning with a-gem. In *ICLR*, 2019.

[31] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[32] Y. Chen, X. Wang, Z. Liu, H. Xu, and T. Darrell. A new meta-baseline for few-shot learning. *arXiv preprint arXiv:2003.04390*, 2020.

[33] J. Cho, T. Shon, K. Choi, and J. Moon. Dynamic learning model update of hybrid-classifiers for intrusion detection. *The Journal of Supercomputing*, 64(2):522–526, 2013.

[34] J. Chung, S. Lee, H. Nam, J. Lee, and K. M. Lee. Luciddreamer: Domain-free generation of 3d gaussian splatting scenes. *arXiv preprint arXiv:2311.13384*, 2023.

[35] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[36] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, et al. The epic-kitchens dataset: Collection, challenges and baselines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):4125–4141, 2020.

[37] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

[38] M. Deitke, W. Han, A. Herrasti, A. Kembhavi, E. Kolve, R. Mottaghi, J. Salvador, D. Schwenk, E. VanderBilt, M. Wallingford, et al. Robothor: An open simulation-to-real embodied ai platform.

In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3164–3174, 2020.

[39] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi. Objaverse: A universe of annotated 3D objects. *arXiv preprint arXiv:2212.08051*, 2022.

[40] M. Deitke, E. VanderBilt, A. Herrasti, L. Weihs, J. Salvador, K. Ehsani, W. Han, E. Kolve, A. Farhadi, A. Kembhavi, et al. Procthor: Large-scale embodied ai using procedural generation. *arXiv preprint arXiv:2206.06994*, 2022.

[41] M. Deitke, R. Liu, M. Wallingford, H. Ngo, O. Michel, A. Kusupati, A. Fan, C. Laforte, V. Voleti, S. Y. Gadre, E. VanderBilt, A. Kembhavi, C. Vondrick, G. Gkioxari, K. Ehsani, L. Schmidt, and A. Farhadi. Objaverse-XL: A universe of 10M+ 3D objects. *arXiv preprint arXiv:2307.05663*, 2023.

[42] C. Deng, C. Jiang, C. R. Qi, X. Yan, Y. Zhou, L. Guibas, D. Anguelov, et al. NeRDi: Single-view NeRF synthesis with language-guided diffusion as general image priors. In *CVPR*, 2022.

[43] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[44] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[45] L. Downs, A. Francis, N. Koenig, B. Kinman, R. Hickman, K. Reymann, T. B. McHugh, and V. Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2553–2560. IEEE, 2022.

[46] V. Dumoulin, N. Houlsby, U. Evci, X. Zhai, R. Goroshin, S. Gelly, and H. Larochelle. Comparing transfer and meta learning approaches on a unified few-shot classification benchmark. *arXiv preprint arXiv:2104.02638*, 2021.

[47] K. Ehsani, R. Mottaghi, and A. Farhadi. Segan: Segmenting and generating the invisible. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6144–6153, 2018.

[48] M. Eitz, J. Hays, and M. Alexa. How do humans sketch objects? *ACM Transactions on Graphics (TOG)*, 31:1 – 10, 2012.

[49] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image processing*, 15(12):3736–3745, 2006.

[50] U. Evci, T. Gale, J. Menick, P. S. Castro, and E. Elsen. Rigging the lottery: Making all tickets winners. In *Proceedings of the International Conference on Machine Learning*, 2020.

[51] Z. Fan, P. Wang, Y. Jiang, X. Gong, D. Xu, and Z. Wang. Nerf-sos: Any-view self-supervised object segmentation on complex scenes. *arXiv preprint arXiv:2209.08776*, 2022.

[52] A. Fang, G. Ilharco, M. Wortsman, Y. Wan, V. Shankar, A. Dave, and L. Schmidt. Data determines distributional robustness in contrastive language image pre-training (CLIP). In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 6216–6234. PMLR, 2022. URL https://proceedings.mlr.press/v162/fang22a.html.

[53] H. Fang, P. Xiong, L. Xu, and Y. Chen. Clip2video: Mastering video-text retrieval via image clip. *arXiv preprint arXiv:2106.11097*, 2021.

[54] C. Feichtenhofer, H. Fan, B. Xiong, R. Girshick, and K. He. A large-scale study on unsupervised spatiotemporal representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3299–3309, 2021.

[55] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[56] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. *AAAI*, 1999.

[57] N. T. Franklin and M. J. Frank. Generalizing to generalize: Humans flexibly switch between compositional and conjunctive structures during reinforcement learning. *PLoS computational biology*, 16(4): e1007720, 2020.

[58] A. Gammerman, V. Vovk, and V. Vapnik. Learning by transduction. *arXiv preprint arXiv:1301.7375*, 2013.

[59] Y. Gandelsman, Y. Sun, X. Chen, and A. Efros. Test-time training with masked autoencoders. *Advances in Neural Information Processing Systems*, 35:29374–29385, 2022.

[60] Y. Gao, J. Ma, M. Zhao, W. Liu, and A. L. Yuille. Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3205–3214, 2019.

[61] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[62] H. M. Gomes, J. Read, A. Bifet, J. P. Barddal, and J. Gama. Machine learning for streaming data: state of the art, challenges, and opportunities. *ACM SIGKDD Explorations Newsletter*, 2019.

[63] D. Gordon, K. Ehsani, D. Fox, and A. Farhadi. Watching the world go by: Representation learning from unlabeled videos. *arXiv preprint arXiv:2003.07990*, 2020.

[64] S. Goyal, A. Kumar, S. Garg, Z. Kolter, and A. Raghunathan. Finetune like you pretrain: Improved finetuning of zero-shot vision models. *arXiv preprint arXiv:2212.00638*, 2022.

[65] K. Grauman, A. Westbury, E. Byrne, Z. Chavis, A. Furnari, R. Girdhar, J. Hamburger, H. Jiang, M. Liu, X. Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18995–19012, 2022.

[66] K. Greff, R. L. Kaufman, R. Kabra, N. Watters, C. Burgess, D. Zoran, L. Matthey, M. Botvinick, and

A. Lerchner. Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning*, pages 2424–2433. PMLR, 2019.

[67] Y. Guo, H. Shi, A. Kumar, K. Grauman, T. Rosing, and R. Feris. Spottune: transfer learning through adaptive fine-tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4805–4814, 2019.

[68] A. Gupta, P. Dollar, and R. Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5356–5364, 2019.

[69] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR, 2020.

[70] B. Hariharan and R. Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3018–3027, 2017.

[71] J. Harrison, A. Sharma, C. Finn, and M. Pavone. Continuous meta-learning without tasks. *Advances in neural information processing systems*, 2019.

[72] T. L. Hayes and C. Kanan. Lifelong machine learning with deep streaming linear discriminant analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 220–221, 2020.

[73] J. He, R. Mao, Z. Shao, and F. Zhu. Incremental learning in online scenario. *arXiv preprint arXiv:2003.13191*, 2020.

[74] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[75] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[76] K. He, R. Girshick, and P. Dollár. Rethinking imagenet pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4918–4927, 2019.

[77] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

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

[79] D. Hendrycks, S. Basart, N. Mu, S. Kadavath, F. Wang, E. Dorundo, R. Desai, T. Zhu, S. Parajuli, M. Guo, D. Song, J. Steinhardt, and J. Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *ICCV*, 2021.

[80] D. Hendrycks, K. Zhao, S. Basart, J. Steinhardt, and D. Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15262–15271, 2021.

[81] L. Hermer-Vazquez, A. Moffet, and P. Munkholm. Language, space, and the development of cognitive flexibility in humans: The case of two spatial memory tasks. *Cognition*, 79(3):263–299, 2001.

[82] N. L. Hjort, C. Holmes, P. Müller, and S. G. Walker. *Bayesian nonparametrics*, volume 28. Cambridge University Press, 2010.

[83] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[84] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[85] K. Huang and S. Aviyente. Sparse representation for signal classification. *Advances in neural information processing systems*, 19, 2006.

[86] A. Hussain, N. Holla, P. Mishra, H. Yannakoudakis, and E. Shutova. Towards a robust experimental framework and benchmark for lifelong language learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

[87] G. Ilharco, M. Wortsman, R. Wightman, C. Gordon, N. Carlini, R. Taori, A. Dave, V. Shankar, H. Namkoong, J. Miller, H. Hajishirzi, A. Farhadi, and L. Schmidt. Openclip, July 2021. URL https://doi.org/10.5281/zenodo.5143773.

[88] A. Jain, M. Tancik, and P. Abbeel. Putting NeRF on a diet: Semantically consistent few-shot view synthesis. In *ICCV*, 2021.

[89] A. Jain, M. Tancik, and P. Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5885–5894, 2021.

[90] T. Jain, C. Lennan, Z. John, and D. Tran. Imagededup. https://github.com/idealo/imagededup, 2019.

[91] M. A. Jamal and G.-J. Qi. Task agnostic meta-learning for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11719–11727, 2019.

[92] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[93] W. Jang and L. Agapito. Codenerf: Disentangled neural radiance fields for object categories. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12949–12958, 2021.

[94] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. V. Le, Y. Sung, Z. Li, and T. Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. *arXiv preprint arXiv:2102.05918*, 2021.

[95] L. Jing and Y. Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[96] J. Johnson, B. Hariharan, L. Van Der Maaten, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910, 2017.

[97] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[98] S. P. Johnson, D. Amso, and J. A. Slemmer. Development of object concepts in infancy: Evidence for early learning in an eye-tracking paradigm. *Proceedings of the National Academy of Sciences*, 100 (18):10568–10573, 2003.

[99] P. N. Johnson-Laird. Mental models and human reasoning. *Proceedings of the National Academy of Sciences of the United States of America*, 107(43):18243–18250, 2010. doi: 10.1073/pnas.1012933107. URL https://doi.org/10.1073/pnas.1012933107.

[100] R. Kabra, D. Zoran, G. Erdogan, L. Matthey, A. Creswell, M. Botvinick, A. Lerchner, and C. Burgess. Simone: View-invariant, temporally-abstracted object representations via unsupervised video decomposition. *Advances in Neural Information Processing Systems*, 34:20146–20159, 2021.

[101] L. Karazija, I. Laina, and C. Rupprecht. Clevrtex: A texture-rich benchmark for unsupervised multi-object segmentation. *arXiv preprint arXiv:2111.10265*, 2021.

[102] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):1–14, 2023.

[103] A. Khandelwal, L. Weihs, R. Mottaghi, and A. Kembhavi. Simple but effective: Clip embeddings for embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14829–14838, 2022.

[104] U. Khandelwal, O. Levy, D. Jurafsky, L. Zettlemoyer, and M. Lewis. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019.

[105] T. Kipf, G. F. Elsayed, A. Mahendran, A. Stone, S. Sabour, G. Heigold, R. Jonschkowski, A. Dosovitskiy, and K. Greff. Conditional object-centric learning from video. *arXiv preprint arXiv:2111.12594*, 2021.

[106] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Over-

coming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. ISSN 0027-8424. doi: 10.1073/pnas.1611835114. URL https://www.pnas.org/content/114/13/3521.

[107] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 2017.

[108] D. E. Knuth. Retrieval on secondary keys. *The art of computer programming: Sorting and Searching*, 3:550–567, 1997.

[109] I. Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *CVPR*, pages 6129–6138, 2017.

[110] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

[111] S. Kong and D. Ramanan. Opengan: Open-set recognition via open data generation. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021.

[112] A. R. Kosiorek, H. Strathmann, D. Zoran, P. Moreno, R. Schneider, S. Mokrá, and D. J. Rezende. Nerf-vae: A geometry aware 3d scene generative model. In *International Conference on Machine Learning*, pages 5742–5752. PMLR, 2021.

[113] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. *2013 IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013.

[114] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013.

[115] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[116] A. Kumar, A. Raghunathan, R. M. Jones, T. Ma, and P. Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=UYneFzXSJWh.

[117] A. Kusupati, V. Ramanujan, R. Somani, M. Wortsman, P. Jain, S. Kakade, and A. Farhadi. Soft threshold weight reparameterization for learnable sparsity. In *Proceedings of the International Conference on Machine Learning*, 2020.

[118] A. Kusupati, M. Wallingford, V. Ramanujan, R. Somani, J. S. Park, K. Pillutla, P. Jain, S. Kakade, and A. Farhadi. Llc: Accurate, multi-purpose learnt low-dimensional binary codes. *Advances in Neural Information Processing Systems*, 34:23900–23913, 2021.

[119] A. Kusupati, G. Bhatt, A. Rege, M. Wallingford, A. Sinha, V. Ramanujan, W. Howard-Snyder, K. Chen, S. Kakade, P. Jain, et al. Matryoshka representation learning. *arXiv preprint arXiv:2205.13147*, 2022.

[120] B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011.

[121] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[122] V. Lazova, V. Guzov, K. Olszewski, S. Tulyakov, and G. Pons-Moll. Control-nerf: Editable feature volumes for scene rendering and manipulation. *arXiv preprint arXiv:2204.10850*, 2022.

[123] Y. LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[124] K. Lee. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in neural information processing systems workshop*.

[125] Y. Lee, A. S. Chen, F. Tajwar, A. Kumar, H. Yao, P. Liang, and C. Finn. Surgical fine-tuning improves adaptation to distribution shifts. *CoRR*, abs/2210.11466, 2022. doi: 10.48550/arXiv.2210.11466. URL https://doi.org/10.48550/arXiv.2210.11466.

[126] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016.

[127] Z. Li and D. Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[128] C.-H. Lin, J. Gao, L. Tang, T. Takikawa, X. Zeng, X. Huang, K. Kreis, S. Fidler, M.-Y. Liu, and T.-Y. Lin. Magic3D: High-resolution text-to-3D content creation. In *CVPR*, 2023.

[129] H. Lin. Robotic manipulation based on 3d vision: A survey. *Proceedings of the 2020 International Conference on Pattern Recognition and Intelligent Systems*, 2020. URL https://api.semanticscholar.org/CorpusID:221498989.

[130] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[131] Z. Lin, Y.-F. Wu, S. V. Peri, W. Sun, G. Singh, F. Deng, J. Jiang, and S. Ahn. Space: Unsupervised object-oriented scene representation via spatial attention and decomposition. *arXiv preprint arXiv:2001.02407*, 2020.

[132] Z. Lin, D. Ramanan, and A. Bansal. Streaming self-training via domain-agnostic unlabeled images. *arXiv preprint arXiv:2104.03309*, 2021.

[133] A. Liu, R. Tucker, V. Jampani, A. Makadia, N. Snavely, and A. Kanazawa. Infinite nature: Perpetual view generation of natural scenes from a single image. In *ICCV*, 2021.

[134] H. Liu, K. Son, J. Yang, C. Liu, J. Gao, Y. J. Lee, and C. Li. Learning customized visual models with retrieval-augmented knowledge. *arXiv preprint arXiv:2301.07094*, 2023.

[135] R. Liu, R. Wu, B. V. Hoorick, P. Tokmakov, S. Zakharov, and C. Vondrick. Zero-1-to-3: Zero-shot one image to 3D object. In *CVPR*, 2023.

[136] Z. Liu, Z. Miao, X. Zhan, J. Wang, B. Gong, and S. X. Yu. Large-scale long-tailed recognition in an

open world. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2537–2546, 2019.

[137] F. Locatello, D. Weissenborn, T. Unterthiner, A. Mahendran, G. Heigold, J. Uszkoreit, A. Dosovitskiy, and T. Kipf. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 33:11525–11538, 2020.

[138] Y. Lu, A. Kumar, S. Zhai, Y. Cheng, T. Javidi, and R. Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5334–5343, 2017.

[139] W.-C. Ma, A. J. Yang, S. Wang, R. Urtasun, and A. Torralba. Virtual correspondence: Humans as a cue for extreme-view geometry. In *CVPR*, 2022.

[140] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[141] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Learning discriminative dictionaries for local image analysis. In *26th IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[142] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013.

[143] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[144] A. Mallya and S. Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[145] A. Mallya, D. Davis, and S. Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82, 2018.

[146] M. Mancini, E. Ricci, B. Caputo, and S. Rota Bulò. Adding new tasks to a single network with weight transformations using binary masks. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[147] C. Manning, P. Raghavan, and H. Schütze. Vector space classification. *Introduction to Information Retrieval*, pages 289–317, 2008.

[148] M. Masana, I. Ruiz, J. Serrat, J. van de Weijer, and A. M. Lopez. Metric learning for novelty and anomaly detection. In *Proceedings of the British Machine Vision Conference*, 2018.

[149] J. Mcauliffe and D. Blei. Supervised topic models. *Advances in neural information processing systems*, 20, 2007.

[150] S. Menon and C. Vondrick. Visual classification via description from large language models. *arXiv preprint arXiv:2210.07183*, 2022.

[151] D. E. Meyer and R. W. Schvaneveldt. Facilitation in recognizing pairs of words: evidence of a dependence between retrieval operations. *Journal of experimental psychology*, 90(2):227, 1971.

[152] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[153] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[154] B. Mildenhall, P. Hedman, R. Martin-Brualla, P. P. Srinivasan, and J. T. Barron. Nerf in the dark: High dynamic range view synthesis from noisy raw images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16190–16199, 2022.

[155] J. Miller, R. Taori, A. Raghunathan, S. Sagawa, P. W. Koh, V. Shankar, P. Liang, Y. Carmon, and L. Schmidt. Accuracy on the line: on the strong correlation between out-of-distribution and in-distribution generalization. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, vol-

ume 139 of *Proceedings of Machine Learning Research*, pages 7721–7735. PMLR, 2021. URL http://proceedings.mlr.press/v139/miller21b.html.

[156] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3994–4003, 2016.

[157] T. Monnier, E. Vincent, J. Ponce, and M. Aubry. Unsupervised layered image decomposition into object prototypes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8640–8650, 2021.

[158] P. Morgado and N. Vasconcelos. Nettailor: Tuning the architecture, not just the weights. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3044–3054, 2019.

[159] R. Mottaghi, C. Schenck, D. Fox, and A. Farhadi. See the glass half full: Reasoning about liquid containers, their volume and content. *ICCV*, 2017.

[160] S. Munder and D. M. Gavrila. An experimental study on pedestrian classification. *IEEE transactions on pattern analysis and machine intelligence*, 28(11):1863–1868, 2006.

[161] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[162] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[163] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[164] A. Nichol, H. Jun, P. Dhariwal, P. Mishkin, and M. Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022.

[165] M. Niemeyer and A. Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11453–11464, 2021.

[166] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729, 2008.

[167] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008.

[168] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.

[169] B. Oreshkin, P. Rodríguez López, and A. Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. *Advances in neural information processing systems*, 31, 2018.

[170] S. Papa, O. Winther, and A. Dittadi. Inductive biases for object-centric representations in the presence of complex textures. In *UAI 2022 Workshop on Causal Representation Learning*, 2022.

[171] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2021.

[172] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE, 2012.

[173] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1406–1415, 2019.

[174] J. Piaget. *Child's Conception of Space: Selected Works vol 4*. Routledge, 2013.

[175] A. Polacco and K. Backes. The amazon go concept: Implications, applications, and sustainability. *Journal of Business and Management*, 24(1):79–92, 2018.

[176] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall. DreamFusion: Text-to-3D using 2D diffusion. In *ICLR*, 2022.

[177] A. Prabhu, P. H. Torr, and P. K. Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *Proceedings of the European Conference on Computer Vision*, 2020.

[178] S. Pratt, R. Liu, and A. Farhadi. What does a platypus look like? generating customized prompts for zero-shot image classification. *arXiv preprint arXiv:2209.03320*, 2022.

[179] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021.

[180] S. Purushwalkam and A. Gupta. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. *arXiv preprint arXiv:2007.13916*, 2020.

[181] H. Qi, M. Brown, and D. G. Lowe. Low-shot learning with imprinted weights. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5822–5830, 2018.

[182] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[183] A. Raistrick, L. Lipson, Z. Ma, L. Mei, M. Wang, Y. Zuo, K. Kayan, H. Wen, B. Han, Y. Wang, et al. Infinite photorealistic worlds using procedural generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12630–12641, 2023.

[184] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with CLIP latents. *arXiv preprint arXiv:2204.06125*, 2022.

[185] R. Ranjan, V. M. Patel, and R. Chellappa. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *IEEE transactions on pattern analysis and machine intelligence*, 41(1):121–135, 2017.

[186] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

[187] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. *International Conference on Learning Representations*, 2017.

[188] S.-A. Rebuffi, H. Bilen, and A. Vedaldi. Learning multiple visual domains with residual adapters. *arXiv preprint arXiv:1705.08045*, 2017.

[189] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5533–5542, 2017. doi: 10.1109/CVPR.2017.587.

[190] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[191] S.-A. Rebuffi, H. Bilen, and A. Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8119–8127, 2018.

[192] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar. Do imagenet classifiers generalize to imagenet? In *International conference on machine learning*, pages 5389–5400. PMLR, 2019.

[193] A. Rege, A. Kusupati, A. Fan, Q. Cao, S. Kakade, P. Jain, A. Farhadi, et al. Adanns: A framework for adaptive semantic search. *arXiv preprint arXiv:2305.19435*, 2023.

[194] J. Reizenstein, R. Shapovalov, P. Henzler, L. Sbordone, P. Labatut, and D. Novotny. Common objects in 3D: Large-scale learning and evaluation of real-life 3D category reconstruction. In *ICCV*, 2021.

[195] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing inference. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019.

[196] H. L. Roediger and K. B. McDermott. Creating false memories: Remembering words not presented in lists. *Journal of experimental psychology: Learning, Memory, and Cognition*, 21(4):803, 1995.

[197] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. *arXiv*, 2021.

[198] E. Rosch, C. B. Mervis, W. D. Gray, D. M. Johnson, and P. Boyes-Braem. Basic objects in natural categories. *Cognitive psychology*, 8(3):382–439, 1976.

[199] A. Rosenfeld and J. K. Tsotsos. Incremental learning through deep adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 42(3):651–663, 2018.

[200] S. Ruder, J. Bingel, I. Augenstein, and A. Søgaard. Latent multi-task architecture learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4822–4829, 2019.

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

[202] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015.

[203] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *ArXiv*, abs/1606.04671, 2016.

[204] M. S. Sajjadi, D. Duckworth, A. Mahendran, S. van Steenkiste, F. Pavetić, M. Lučić, L. J. Guibas, K. Greff, and T. Kipf. Object scene representation transformer. *arXiv preprint arXiv:2206.06922*, 2022.

[205] M. S. Sajjadi, H. Meyer, E. Pot, U. Bergmann, K. Greff, N. Radwan, S. Vora, M. Lučić, D. Duckworth, A. Dosovitskiy, et al. Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6229–6238, 2022.

[206] M. S. Sajjadi, A. Mahendran, T. Kipf, E. Pot, D. Duckworth, M. Lučić, and K. Greff. Rust: Latent

neural scene representations from unposed imagery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17297–17306, 2023.

[207] B. Saleh and A. Elgammal. Large-scale classification of fine-art paintings: Learning the right metric on the right feature. *ArXiv*, abs/1505.00855, 2015.

[208] K. Sargent, Z. Li, T. Shah, C. Herrmann, H.-X. Yu, Y. Zhang, E. R. Chan, D. Lagun, L. Fei-Fei, D. Sun, et al. Zeronvs: Zero-shot 360-degree view synthesis from a single real image. *arXiv preprint arXiv:2310.17994*, 2023.

[209] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12716–12725, 2019.

[210] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *CVPR*, 2016.

[211] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[212] J. L. Schonberger and J.-M. Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.

[213] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022.

[214] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[215] Y. Shi, P. Wang, J. Ye, M. Long, K. Li, and X. Yang. Mvdream: Multi-view diffusion for 3d generation. *arXiv preprint arXiv:2308.16512*, 2023.

[216] J. Shriram, A. Trevithick, L. Liu, and R. Ramamoorthi. Realmdreamer: Text-driven 3d scene generation with inpainting and depth diffusion. *arXiv preprint arXiv:2404.07199*, 2024.

[217] M. Shu, W. Nie, D.-A. Huang, Z. Yu, T. Goldstein, A. Anandkumar, and C. Xiao. Test-time prompt tuning for zero-shot generalization in vision-language models. *arXiv preprint arXiv:2209.07511*, 2022.

[218] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pages 746–760. Springer, 2012.

[219] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *Computer Vision, IEEE International Conference on*, volume 3, pages 1470–1470. IEEE Computer Society, 2003.

[220] D. Smirnov, M. Gharbi, M. Fisher, V. Guizilini, A. A. Efros, and J. Solomon. MarioNette: Self-supervised sprite learning. In *Advances in Neural Information Processing Systems*, 2021.

[221] C. Smith, H.-X. Yu, S. Zakharov, F. Durand, J. B. Tenenbaum, J. Wu, and V. Sitzmann. Unsupervised discovery and composition of object light fields. *arXiv preprint arXiv:2205.03923*, 2022.

[222] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

[223] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[224] E. S. Spelke. Principles of object perception. *Cognitive science*, 14(1):29–56, 1990.

[225] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32:323–332, 2012.

[226] J. Stefanowski and D. Brzezinski. Stream classification., 2017.

[227] A. Steiner, A. Kolesnikov, X. Zhai, R. Wightman, J. Uszkoreit, and L. Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv preprint arXiv:2106.10270*, 2021.

[228] K. Stelzner, K. Kersting, and A. R. Kosiorek. Decomposing 3d scenes into objects via unsupervised volume segmentation. *arXiv preprint arXiv:2104.01148*, 2021.

[229] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele. Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 403–412, 2019.

[230] X. Sun, R. Panda, R. Feris, and K. Saenko. Adashare: Learning what to share for efficient deep multi-task learning. *Neural Information Processing Systems*, 2020.

[231] Y. Sun, X. Wang, Z. Liu, J. Miller, A. Efros, and M. Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pages 9229–9248. PMLR, 2020.

[232] Y. Sun, X. Wang, L. Zhuang, J. Miller, M. Hardt, and A. A. Efros. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, 2020.

[233] R. Taori, A. Dave, V. Shankar, N. Carlini, B. Recht, and L. Schmidt. Measuring robustness to natural distribution shifts in image classification. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/d8330f857a17c53d217014ee776bfd50-Abstract.html.

[234] Z. Teed and J. Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *Advances in neural information processing systems*, 34:16558–16569, 2021.

[235] A. Tewari, T. Yin, G. Cazenavette, S. Rezchikov, J. Tenenbaum, F. Durand, B. Freeman, and V. Sitzmann. Diffusion with forward models: Solving stochastic inverse problems without direct supervision. *Advances in Neural Information Processing Systems*, 36, 2024.

[236] S. Thrun. Is learning the n-th thing any easier than learning the first? *Advances in neural information processing systems*, 1996.

[237] Y. Tian, Y. Wang, D. Krishnan, J. B. Tenenbaum, and P. Isola. Rethinking few-shot image classification: a good embedding is all you need? *arXiv preprint arXiv:2003.11539*, 2020.

[238] J. T. Todd. The visual perception of 3d shape. *Trends in cognitive sciences*, 8(3):115–121, 2004.

[239] E. Triantafillou, T. Zhu, V. Dumoulin, P. Lamblin, U. Evci, K. Xu, R. Goroshin, C. Gelada, K. Swersky, P.-A. Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. *International Conference on Learning Representations*, 2020.

[240] M. Tschannen, O. Bachem, and M. Lucic. Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*, 2018.

[241] V. Tschernezki, A. Darkhalil, Z. Zhu, D. Fouhey, I. Laina, D. Larlus, D. Damen, and A. Vedaldi. Epic fields: Marrying 3d geometry and video understanding. *Advances in Neural Information Processing Systems*, 36, 2024.

[242] V. Udandarao, A. Gupta, and S. Albanie. Sus-x: Training-free name-only transfer of vision-language models. *arXiv preprint arXiv:2211.16198*, 2022.

[243] S. Ullman. The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 203(1153):405–426, 1979.

[244] G. M. van de Ven, T. Tuytelaars, and A. S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, pages 1–13, 2022.

[245] A. Van Den Oord, O. Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[246] G. Van Horn, O. Mac Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778, 2018.

[247] S. Vandenhende, S. Georgoulis, B. De Brabandere, and L. Van Gool. Branched multi-task networks: deciding what layers to share. *arXiv preprint arXiv:1904.02920*, 2019.

[248] S. Vaze, K. Han, A. Vedaldi, and A. Zisserman. Open-set recognition: A good closed-set classifier is all you need. *arXiv preprint arXiv:2110.06207*, 2021.

[249] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[250] S. Vora, N. Radwan, K. Greff, H. Meyer, K. Genova, M. S. Sajjadi, E. Pot, A. Tagliasacchi, and D. Duckworth. Nesf: Neural semantic fields for generalizable semantic segmentation of 3d scenes. *arXiv preprint arXiv:2111.13260*, 2021.

[251] C. Wah, S. Branson, P. Welinder, P. Perona, and S. J. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[252] M. Wallingford, H. Li, A. Achille, A. Ravichandran, C. Fowlkes, R. Bhotika, and S. Soatto. Task adaptive parameter sharing for multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7561–7570, 2022.

[253] H. Wang, S. Ge, Z. C. Lipton, and E. P. Xing. Learning robust global representations by penalizing local predictive power. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 10506–10518, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/3eefceb8087e964f89c2d59e8a249915-Abstract.html.

[254] H. Wang, X. Du, J. Li, R. A. Yeh, and G. Shakhnarovich. Score Jacobian chaining: Lifting pretrained 2D diffusion models for 3D generation. *arXiv preprint arXiv:2212.00774*, 2022.

[255] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud. Dust3r: Geometric 3d vision made easy. *arXiv preprint arXiv:2312.14132*, 2023.

[256] Y. Wang, W.-L. Chao, K. Q. Weinberger, and L. van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv preprint arXiv:1911.04623*, 2019.

[257] Z. Wang, C. Lu, Y. Wang, F. Bao, C. Li, H. Su, and J. Zhu. ProlificDreamer: High-fidelity and diverse text-to-3D generation with variational score distillation. *arXiv preprint arXiv:2305.16213*, 2023.

[258] K. K. Wankhade, S. S. Dongre, and K. C. Jondhale. Data stream classification: a review.

[259] K. Wankhede, B. Wukkadada, and V. Nadar. Just walk-out technology and its challenges: A case

of amazon go. In *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, pages 254–257. IEEE, 2018.

[260] Y. Wen, D. Tran, and J. Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *International Conference on Learning Representations*, 2020.

[261] D. Wertheimer and B. Hariharan. Few-shot learning with localization in realistic settings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6558–6567, 2019.

[262] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *arXiv preprint arXiv:1911.00357*, 2019.

[263] M. Wortsman, A. Farhadi, and M. Rastegari. Discovering neural wirings. *Advances in Neural Information Processing Systems*, 32, 2019.

[264] M. Wortsman, V. Ramanujan, R. Liu, A. Kembhavi, M. Rastegari, J. Yosinski, and A. Farhadi. Supermasks in superposition. *arXiv preprint arXiv:2006.14769*, 2020.

[265] M. Wortsman, G. Ilharco, J. W. Kim, M. Li, S. Kornblith, R. Roelofs, R. G. Lopes, H. Hajishirzi, A. Farhadi, H. Namkoong, et al. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7959–7971, 2022.

[266] C.-Y. Wu, J. Johnson, J. Malik, C. Feichtenhofer, and G. Gkioxari. Multiview compressive coding for 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9065–9075, 2023.

[267] L. Wu, J. Y. Lee, A. Bhattad, Y.-X. Wang, and D. Forsyth. Diver: Real-time and accurate neural radiance fields with deterministic integration for volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16200–16209, 2022.

[268] R. Wu, B. Mildenhall, P. Henzler, K. Park, R. Gao, D. Watson, P. P. Srinivasan, D. Verbin, J. T. Barron, B. Poole, and A. Holynski. Reconfusion: 3d reconstruction with diffusion priors. *arXiv*, 2023.

[269] H. Xia, Z.-H. Lin, W.-C. Ma, and S. Wang. Video2game: Real-time, interactive, realistic and browser-compatible environment from a single video, 2024.

[270] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv*, 2017.

[271] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3485–3492. IEEE, 2010.

[272] D. Xu, Y. Jiang, P. Wang, Z. Fan, H. Shi, and Z. Wang. SinNeRF: Training neural radiance fields on complex scenes from a single image. In *ECCV*, 2022.

[273] W. Xu, H. Wang, Z. Tu, et al. Attentional constellation nets for few-shot learning. In *International Conference on Learning Representations*, 2020.

[274] B. Yang, Y. Zhang, Y. Xu, Y. Li, H. Zhou, H. Bao, G. Zhang, and Z. Cui. Learning object-compositional neural radiance field for editable scene rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13779–13788, 2021.

[275] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. *arXiv preprint arXiv:2401.10891*, 2024.

[276] Z. Yang, Y. Chen, J. Wang, S. Manivasagam, W.-C. Ma, A. J. Yang, and R. Urtasun. Unisim: A neural closed-loop sensor simulator. *CVPR*, 2023.

[277] L. Yen-Chen, P. Florence, A. Zeng, J. T. Barron, Y. Du, W.-C. Ma, A. Simeonov, A. R. Garcia, and P. Isola. Mira: Mental imagery for robotic affordances, 2022.

[278] K. Y. Yeung and W. L. Ruzzo. Details of the adjusted rand index and clustering algorithms supplement to the paper " an empirical study on principal component analysis for clustering gene expression data " (to appear in bioinformatics). 2001.

[279] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa. Plenoctrees for real-time rendering of neural

radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5761, 2021.

[280] A. Yu, V. Ye, M. Tancik, and A. Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021.

[281] A. Yu, V. Ye, M. Tancik, and A. Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021.

[282] H.-X. Yu, L. J. Guibas, and J. Wu. Unsupervised discovery of object radiance fields. *arXiv preprint arXiv:2107.07905*, 2021.

[283] X. Yu, M. Xu, Y. Zhang, H. Liu, C. Ye, Y. Wu, Z. Yan, C. Zhu, Z. Xiong, T. Liang, et al. Mvimgnet: A large-scale dataset of multi-view images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9150–9161, 2023.

[284] J. O. Zhang, A. Sax, A. Zamir, L. Guibas, and J. Malik. Side-tuning: A baseline for network adaptation via additive side networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 698–714. Springer, 2020.

[285] R. Zhang, W. Zhang, R. Fang, P. Gao, K. Li, J. Dai, Y. Qiao, and H. Li. Tip-adapter: Training-free adaption of clip for few-shot classification. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXV*, pages 493–510. Springer, 2022.

[286] X. Zhao, A. Colburn, F. Ma, M. A. Bautista, J. M. Susskind, and A. G. Schwing. Is generalized dynamic novel view synthesis from monocular videos possible today? *arXiv preprint arXiv:2310.08587*, 2023.

[287] S. Zhi, T. Laidlow, S. Leutenegger, and A. J. Davison. In-place scene labelling and understanding with implicit scene representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15838–15847, 2021.

[288] B. Zhou, P. Krähenbühl, and V. Koltun. Does computer vision matter for action? *Science Robotics*, 2019.

[289] K. Zhou, J. Yang, C. C. Loy, and Z. Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348, 2022.

[290] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 37, 2018.