Scalable Binding

Allan Jabri

Scalable Binding

by

Allan Anwar Jabri

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Summer 2023

The dissertation of Allan Anwar Jabri, titled **Scalable Binding**, is approved:

Chair		Date	
		Date	
		Date	
		Date	

University of California, Berkeley

Scalable Binding

Abstract

Scalable Binding

by

Allan Anwar Jabri

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Alexei A. Efros, Chair

Any useful agent will face many tasks and must rely on transfer of prior knowledge acquired in a scalable manner. This thesis explores inductive biases that enable scalable pre-training of representations – and algorithms that *bind* them – from the design of architectures capable of adaptive computation for scalable generative modeling, to self-supervised objectives that prepare embodied agents with mechanisms for state representation and reward maximization.

First, I consider the challenge of gracefully scaling generative models to high-dimensional data, motivating the importance of adaptive computation, a property missing from predominant architectures. This leads to a simple attention-based architecture for diffusion models capable of dedicating computation adaptively across its input and output, attaining superior performance in image and video generation despite being more domain-agnostic and efficient. Attention visualizations demonstrate how the model learns to allocate computation to more complex parts of samples, and in cases of high redundancy such as video prediction, can even copy information when needed.

Next, I show how self-supervised objectives that exploit more domain knowledge can be used to efficiently solve related downstream tasks. In the domain of perception, I show how a simple self-supervised objective for space-time attention can be used to solve a range of tasks involving temporal correspondence, a central challenge in state representation for embodied agents. In the domain of reinforcement learning, I motivate the importance of scalable construction of task distributions and demonstrate how meta-reinforcement learners can be pre-trained with self-supervised reward models.

Finally, I conclude with a perspective on open problems in scalable pre-training, with a focus on the interplay between transfer across modalities, universal generative modeling objectives for discrete and continuous data, and adaptive computation.

To my family

Contents

List of Figures

List of Tables

Acknowledgments

Chasing the dragon in graduate school is much more fun with the support of mentors and the love of friends and family. As cat-like as I may be, I had the fortune of always finding support. I am indebted to the brilliant minds and kind souls that have helped me grow along my journey:

To my advisor, Alyosha Efros, for his enduring trust in me. It is one thing to wax poetic about the pursuit of knowledge and another to forge a path with equal romance and skepticism. Sometimes graduate school is like backpacking alone in your mind – I am grateful to Alyosha for always letting me hike freely to find my way, while sharing his beacon in times of need.

To members of my committee, Jitendra Malik, Pieter Abbeel, and Rob Fergus, for their guidance and battle-testing my interests along the way. To gracious collaborators, Andrew Owens, Sergey Levine, Chelsea Finn, Pulkit Agrawal, Pavel Tokmakov, waypoints along a winding path.

To Berkeley f(r)iends, Kelvin Xu for being just as much of a wildcard, Andreea Bobu for candid honesty, Aravind Srinivas for living on the edge, Jasmine Collins for vegan hotpot, Ilija Radosavovic for gentle insight, Vickie Ye for praising my tang yuan, Aleksander Holynski for grizzly peak cycles in the rain, Max Simchowitz for authentic stir-fry recipes, Dave Epstein for expert arabic, Yeshwanth Cherapanamjeri for humble wisdom, Armin Askari for fatherly instincts, Yu Sun for a need for speed, Ashish Kumar for the bantz, Parsa Mahmoudieh for an open heart, Michael Chang for indulgent debates, Nilesh Tripuraneni for hopeful sarcasm, Lerrel Pinto for merciful slams, Taesung Park for runs in Port Meadow, Angjoo Kanazawa for a killer ollie. To Ashvin Nair, Chandan Singh, Zihao Chen, Thanard Kurutach for putting up with my soccer skills. To fjords with Avi Singh, Erin Grant, and Vitchyr Pong. To Paula Gradu, for clairvoyance, candlelight, and black sesame.

To kind labmates, Zhe Cao, Sasha Sax, Shubham Goel, Richard Zhang, Tinghui Zhou, Junyan Zhu, Bill Peebles, Tim Brooks, Toru Lin, Yossi Gandelsman, Shiry Ginosar, Evan Shelhamer, David Fouhey, Medhini Naramsiham, Evonne Ng, Kartikeya Mangalam, Haozhi Qi, Jacob Huh, Devon Guillory. To Angie, for making valentine cards a lab tradition. To Jean Nguyen and Shirley Salanio, for helping me graduate.

To the friends with open arms throughout an autumn at Oxford, Lili Momeni, Daffy Afouras, Tengda Han, Chuhan Zhang, Shangzhe Wu, Andrew Brown, and Kai Han. To Andrew Zisserman and Andrea Vedaldi, who welcomed us to VGG. To the pool on Iffley Road, for allowing me to swim in it.

To my hosts at Google Brain, Ting Chen for his patience and appetite for hard problems, and David Fleet for his sense for science. To Thomas Kipf, Sara Sabour, and Geoff Hinton for generosity in time and feedback.

To Upper Street, and Brendan O'Donoghue, Sander Dieleman, Kelsey Allen, and Will Whitney, who welcomed me at DeepMind. To the busker at Angel Station who sounds just like Jimmy Hendrix, Afghan Kitchen for deeming me a regular, and all the runners along Regents Canal for inadvertently racing me.

To mentors at FAIR NY, Laurens van der Maaten and Armand Joulin, who drove me early on to hone a balance between skepticism and optimism. To Marco Baroni, Tomas Mikolov, and Arthur Szlam, who followed convictions with grace. To Rob Fergus and Yann LeCun, for giving me a chance. To Elizabeth Street Garden, the best park in New York.

To friends, Tanner Smith, Adlan Jackson, Alan Hatfield, Christian Puhrsch, June Chang, Green Choi, Brian Choi, Gavin Cook, Luis Stubbe, William Lee, William Palmer, Levi Stanton. To Taylor Chin, for happiness and growth. To Chet Kristy, Zander Slavitz, Daniel Thompson, Seth Andrews, Trevor Burke, Nick Baker, Connor Hansen, Sam Pannepacker.

Finally, to my mother, for showing me how to care. To my father, for the importance of reason. To my grandmother, for the art of war. To my sister, for being my closest confidant and likely the main reason for any redeeming qualities I may have.

Chapter 1

Introduction

Any useful agent will face many tasks and must rely on transfer of prior knowledge in order to learn new tasks efficiently. More than ever before, it is clear that the most effective approaches for acquiring prior knowledge are learning systems that scale with computation and data [224, 86]. The extent to which these systems scale with computation is contingent on the cost of the inductive bias in their underlying architectures, objectives, and training data (Figure 1.1). While early applications of machine learning resorted to manual design of task-specific representations, since the seminal work of Krizhevsky et al [132], progress in supervised deep learning has shown how shifting the burden of inductive bias from representation design to architecture and training data design can allow for learning stronger representations in a manner that scales more gracefully for learning many tasks.

The underlying lesson is that inductive bias is useful insofar as it constrains the space of possible solutions, but can be detrimental if its underlying assumptions are limiting and should scale across tasks gracefully: despite being more expensive to design, manual representations based on human understanding of complex structure eventually lose out to data-driven representations learned end-to-end with architectures laced with more general inductive bias, since they more readily benefit from scaling computation to eventually surpass human insight.

1.1 Scalable Pre-training

While supervised deep learning allows for end-to-end learning representations, the cost of task-specific data annotation still limits scaling across tasks. This motivates approaches for *pre-training* with objectives that do not require such detailed annotation and thus scale even better across domains by leveraging abundant unlabeled data.

Two key approaches for scalable pre-training are self-supervised learning and generative modeling. Self-supervised learning shifts the burden from data annotation to the design of annotation-free objectives that are hypothesized to induce learning of representations similar to those needed for downstream tasks. The design of self-supervised objectives thus

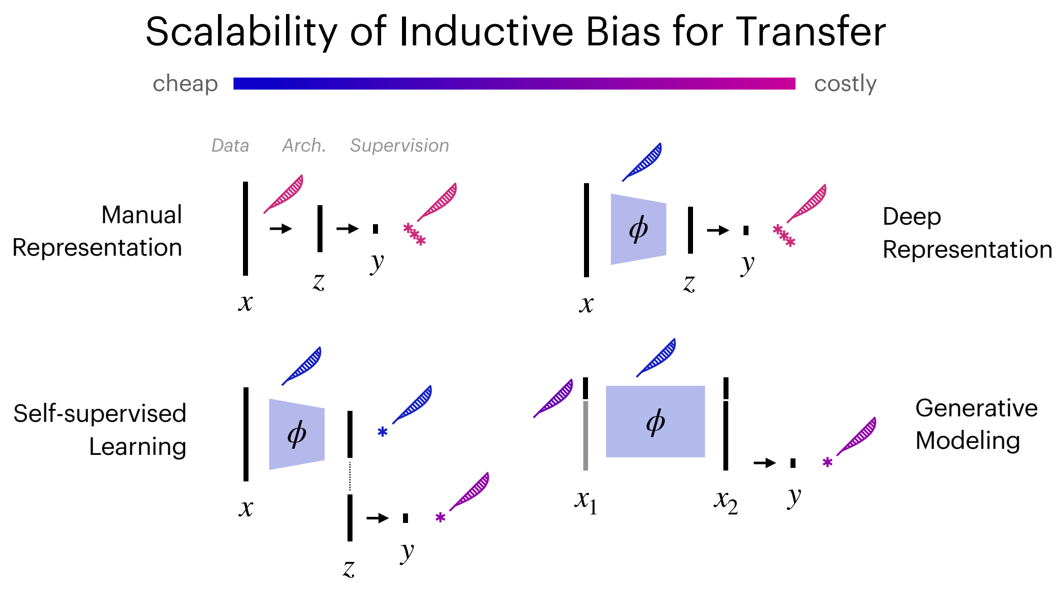


Figure 1.1: **Scalability of Inductive Bias for Transfer.** The scalability of a modern learning system for acquiring prior knowledge is contingent on the cost of the inductive biase laced in its underlying data, architecture, and objectives. Self-supervised and generative learning – i.e. pre-training – allow for acquiring transferrable biases in a scalable manner by shifting the burden from representation design and human data annotation to label-free objectives and unlabeled data curation.

hinges on an understanding of tasks in domains of interest, and assumptions that capture factors of variation in input data that should be modeled. While this improves the sample efficiency of supervised learning of downstream tasks, self-supervised objectives are not entirely task-agnostic and must be designed for groups of tasks.

On the other hand, generative modeling considers more universal objectives based on compression of the input data distribution, such as maximizing data likelihood and proxies thereof. Because these objectives tend to minimally rely on domain-specific assumptions, these techniques tend to be viewed as *unsupervised* learning and are more generally applicable across domains. While this comes at the cost of having to optimize the more demanding objective of high-dimensional stochastic data prediction, progress in large-scale neural network training has unlocked the ability to fit models with orders of magnitude more parameters than ever before. This has ushered in an era wherein the complexity of generative modeling can be counteracted by sufficiently expressive function approximators, such that the overhead of generative pre-training has begun to bear fruit, leading to breakthroughs in language modeling[184, 21], image and video generation[44, 188, 104], and geometric graph applications such as protein design[141]. While self-supervised learning treats representation as the end itself, generative pre-training learns representations – and as we will next discuss, algorithmic subroutines that bind representations – as a means to the end of data prediction.

1.2 Mechanisms of Transfer: Binding

While pre-training techniques aim for transferable knowledge, it is worth asking what mechanisms underlie transfer. The traditional view of pre-training and unsupervised learning centers around the learning of *representations* capturing useful patterns or associations in input data[14]. Recent developments in large-scale generative pre-training have shown that pre-training large neural networks to model conditional distributions of diverse data is not only useful for learning representations, but also for learning algorithms (such as in-context learning [21, 31]) that manipulate sequences of representations in a manner that transfers across domains[119, 21]. For example, the Induction Head hypothesis of Olsson et al [167] puts forth evidence that generative pre-training of sequence models on text data (e.g. language modeling) using architectures with relational inductive biases (e.g. attention-based architectures such as Transformers[235]) can lead to the emergence of attention-heads that implement operations for inductive learning that point to, copy, and transform representations of the input in order to produce the output. Surprisingly, these neural algorithms seem generalize to inputs beyond the training data, such as random sequences of tokens, and seem to be a consequence of the burstiness of rare tokens in training data (a description which fits many kinds of natural data, such as natural language)[31]. Thus, pre-training can induce not only useful representations but also more general purpose algorithms that route, relate, and transduce – more generally, *bind* – said representations for improved generalization.

While these emergent properties are exciting from the perspective of machine learning, it is interesting to note their relevance to long-standing problems in neuroscience and cognitive science, such as the binding problem [191, 6, 230, 56], which asks the question of how humans consolidate important attributes embedded in the flurry of high-dimensional observations drawn from disparate sensors to enable goal-directed behaviour. While on the surface this seems to correspond most directly to the machine learning problem of representation learning, predominant mechanistic theories indulge the importance of subroutines for domain-agnostic routing of information between representations. For example, feature integration theory (FIT) of Treisman et al [230] proposes subroutines of sequential attention operating on top of representations produced by early cortex. Global Workspace Theory[6] purports that domain-specific processors operate in parallel to produce representations connected to a central *global workspace* that integrates and broadcasts information. Similar links are explicitly established in the more recent position paper by Greff, Steenkiste, and Schmidhuber [76], which draws inspiration from the binding problem to motivate inductive biases in neural networks that lead to emergent representations and operators for relational reasoning.

And yet, the Transformer[235] – with an underlying relational inductive bias that supports many emergent properties evocative of binding (such as the Induction Head hypothesis) – was motivated by a need for more parallelizable approaches for training large-scale auto-regressive models. Machine learning systems need not imitate systems hypothesized to underlie human intelligence – notably, since human understanding of human intelligence is fraught with issues of reproducibility and verifiability, and since machines are not subject to the same constraints[1] – but it is no coincidence that mechanistic accounts of emergent solutions in machine and human intelligence sometimes converge despite arising from divergent paradigms. Though these fields operate on different computational substrates, the constraints faced by their systems are much more similar than different, such that their inductive biases are bound to be shared.

1.3 Contributions

This thesis explores inductive biases that enable scalable pre-training, from the design of architectures capable of adaptive computation for scalable generative modeling, to self-supervised objectives that prepare embodied agents with mechanisms for state representation and reward maximization.

In **Chapter 2**, I focus on the role of neural network architecture in scaling generative modeling to high-dimensional data. In particular, motivated by the ubiquity of redundancy in natural data, I argue for the importance of adaptively allocating computation to subspaces of the input and output that carry more information – a property missing from predominant architectures such as Transformers and convolutional networks, which tile the input and output with computation uniformly. This leads to an attention-based architecture in which most computation is applied on auxiliary memory vectors which read from and write to the set of input tokens with attention. Despite being domain-agnostic and more efficient, it allows for superior image and video generation. Moreover, we will see that visualizing the *read* attention allows for identifying input tokens favoured by the auxiliary memory and that thus dominate computation. Throughout generation, read attention (and thus computation) is sparse and biased towards parts of images with more information; for video generation, the network learns to copy information from conditioning frames and focuses computation on regions with more complex dynamics. This behaviour emerges even though position representations are fully learned.

Shifting focus, I show how leveraging more domain-specific structure can lead to self-supervised objectives for more targeted pre-training for downstream tasks.

In **Chapter 3**, I show how a simple self-supervised objective can allow for solving a range of perception tasks involving space-time correspondence, a challenge in state representation for embodied agents which typically requires expensive human annotation. The approach allows for learning a representation for space-time attention that transfers to optical flow, space-time segmentation, and tracking, with a unified self-supervised loss function. Moreover, optimizing the objective with a sequence model leads to object permanence.

In **Chapter 4**, I motivate the importance of scalable task (e.g. reward function) distributions for pre-training in-context reinforcement learners. This leads to a technique for constructing self-supervised reward models which can be used to derive training objectives

[1]For instance, backpropagation and weight sharing are hard to implement in biological neural networks but easy for artificial neural networks.

for meta-reinforcement learning agents such as RL^2[49] without human annotation. We will see that this allows for accelerated reinforcement learning learning of related downstream reward functions specified by humans.

Finally, I conclude with a perspective on open problems in scalable pre-training, with a focus on the interplay between transfer across modalities, universal generative modeling objectives for discrete and continuous data, and adaptive computation.

Chapter 2

A Scalable Architecture for Generative Modeling

Generative models learn to predict conditional distributions of high-dimensional data. In natural data, conditional entropy is never distributed evenly across subspaces of each data point. In this chapter, we argue for the importance of adaptive computation in generative modeling. This leads us to develop the *Recurrent Interface Network* (RIN), a neural net architecture that learns to allocate computation adaptively to parts of the input and output, allowing it to scale to generate high-dimensional data. The hidden units of RINs are partitioned into the *interface*, which is locally connected to inputs, and *latents*, which are decoupled from inputs and can exchange information globally. The RIN block selectively reads from the interface into latents for high-capacity processing, with updates written back to the interface. Stacking multiple blocks allows for more effective routing across local and global levels. While routing adds overhead, the cost can be amortized in recurrent computation settings where inputs change gradually while more global context persists, such as iterative generation using diffusion models. To leverage recurrence, we propose a latent self-conditioning technique that "warm-starts" the latents at each iteration of the generation process. When applied to diffusion models operating directly on pixels, RINs yield state-of-the-art image and video generation without cascades or guidance, while being domain-agnostic and up to $10\times$ more efficient compared to specialized 2D and 3D U-Nets.[1]

2.1 Adaptive Computation

The design of effective neural network architectures has been crucial to the success of deep learning [132, 93, 235]. Influenced by modern accelerator hardware, predominant architectures, such as convolutional neural networks [66, 135, 93] and Transformers [235], allocate computation in a fixed, uniform manner over the input data (e.g., over image pixels,

[1]This work was published as *Scalable Adaptive Computation for Iterative Generation*, Jabri et al, ICML 2023 [110].

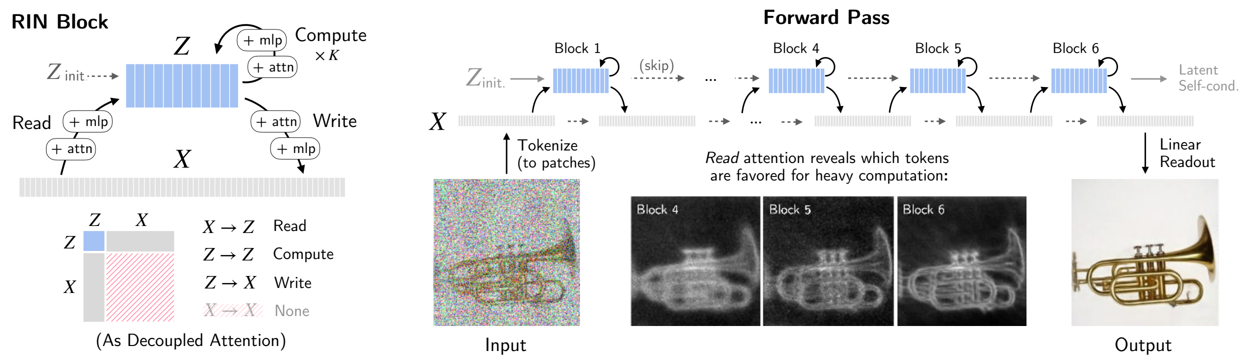


Figure 2.1: Overview of Recurrent Interface Networks. The input is tokenized to form the interface X. A stack of blocks route information between X and latents Z, avoiding quadratic pairwise interactions between tokens in X (bottom left). Note that $|Z| < |X|$, and most computation is applied to Z, which allows for scaling to large X. The network's read attention maps reveals how tokens are favored for latent computation (right), when trained for a task like diffusion generative modeling. See Figure 2.6 for more visualizations.

image patches, or token sequences). Information in natural data is often distributed unevenly, or exhibits redundancy, so it is important to ask how to allocate computation in an adaptive manner to improve scalability. While prior work has explored more dynamic and input-decoupled computation, e.g., networks with auxiliary memory [41, 185] and global units [260, 23, 114, 113], general architectures that leverage adaptive computation to effectively scale to tasks with large input and output spaces remain elusive.

In this chapter, we consider this issue as it manifests in high-dimensional generative modeling tasks, such as image and video generation. When generating an image with a simple background, for instance, an adaptive architecture should ideally be able to allocate computation to regions with complex objects and textures, rather than regions with little or no structure (e.g., the sky). When generating video, one should exploit frame to frame redundancy, allocating less computation to static regions. While such non-uniform allocation of computation becomes more crucial in higher-dimensional data, achieving it efficiently is challenging on modern hardware, given the preference for fixed computation graphs with dense matrix multiplication.

To address this challenge, we introduce a new architecture, dubbed Recurrent Interface Networks (RINs). In RINs (Fig. 2.1), hidden units are partitioned into the *interface* and *latents*. Interface units are locally connected to the input and grow linearly with input size. In contrast, latents are decoupled from the input space, forming a more compact representation. Computation proceeds in a stack of "read-process-write" blocks: in each block, information is selectively read from the interface into the latents for high-capacity global processing, after which incremental updates are sent back to the interface. Alternating computation between latents and the interface allows information to be processed at local and global levels, accumulating context for better routing. As such, RINs allocate computation more effectively

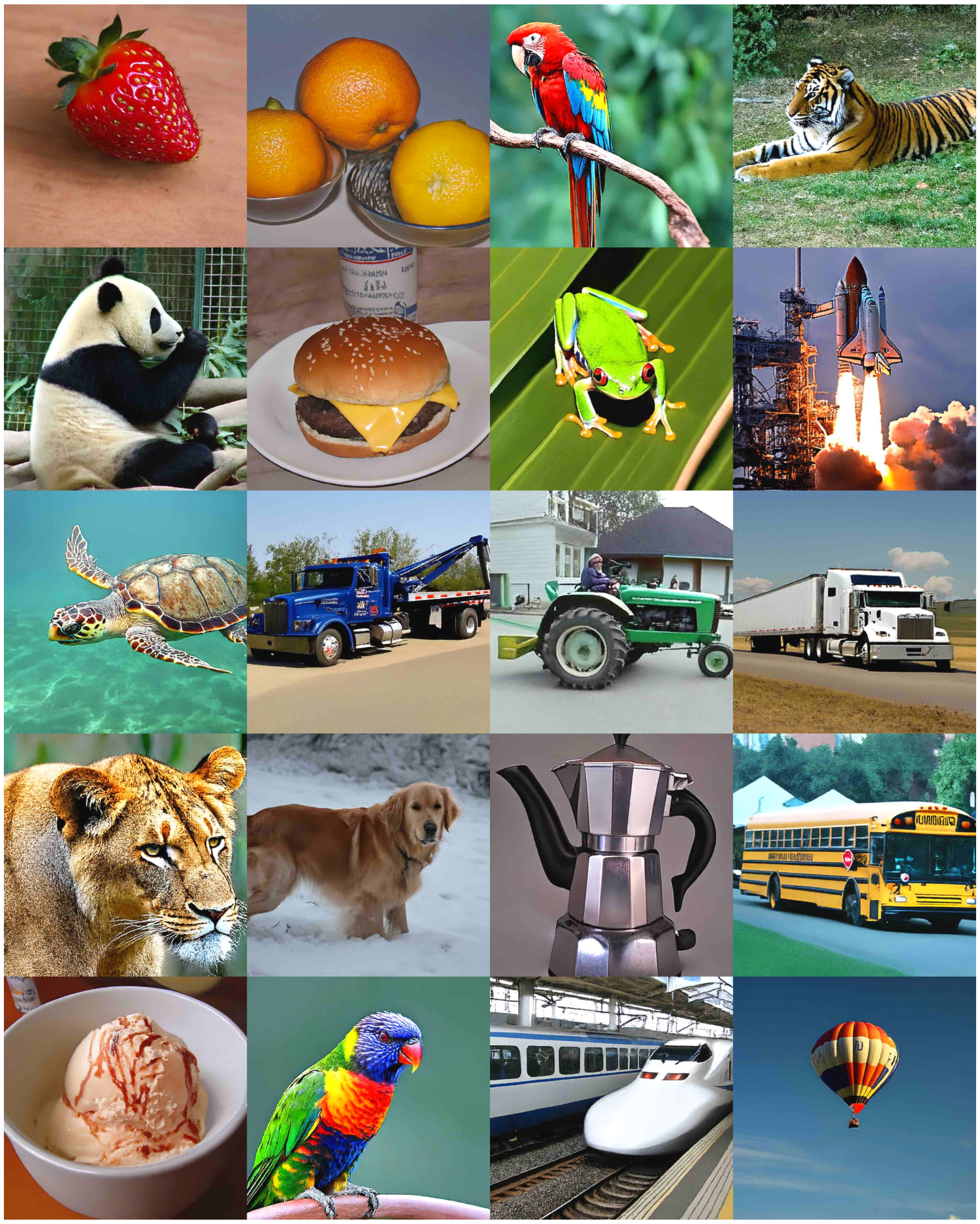


Figure 2.2: Class-conditional ImageNet 1024×1024 samples generated by RINs for diffusion modeling of pixels. While inputs of this size lead to 16384 data tokens, RINs can learn to route the bulk of computation through just 256 latent vectors, using only domain-agnostic attention blocks.

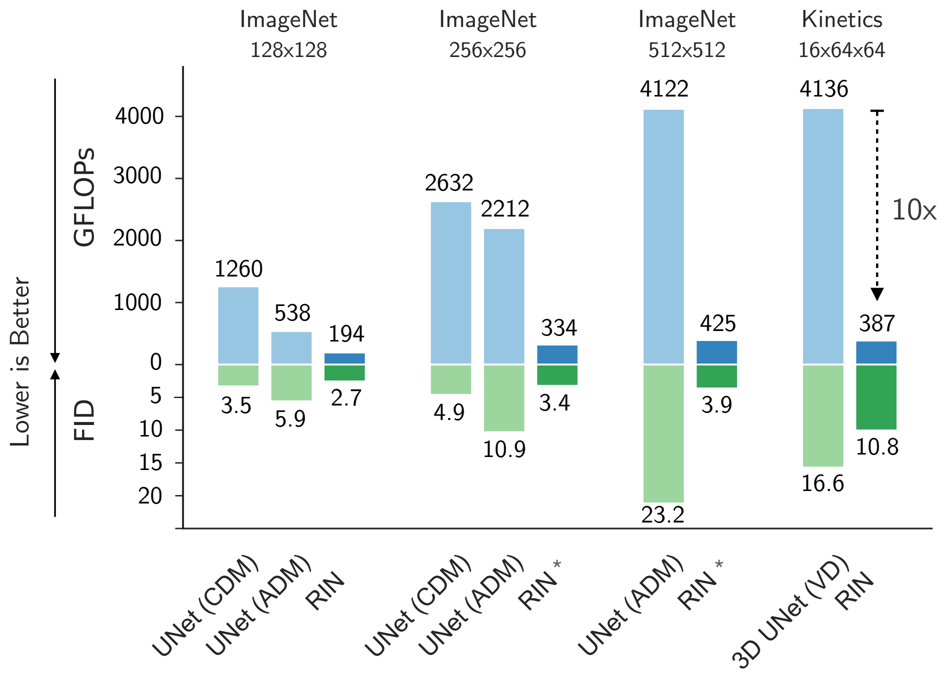


Figure 2.3: RINs outperform U-Nets widely used in recent state-of-the-art image and video diffusion models, while being significantly more efficient and domain-agnostic. Our models are simple pixel-level denoising diffusion models without cascades as in (CDM [103]) or guidance (as in ADM [44] and VD [105]). $*$: from [34] with input scaling.

than uniform models, scaling particularly well when information is unevenly distributed or redundant across the input, as is common in natural data.

Iterative routing of information does require some overhead, which can overshadow potential efficiency gains if the latents are initialized without context, especially for shallow networks. This cost can be reduced in scenarios involving recurrent computation, where network inputs change gradually and persistent context can be leveraged across iterations, in effect forming a deeper network. As a concrete application, we consider iterative generation of images and video with denoising diffusion models [213, 101, 214, 215]. To leverage recurrence, we propose latent self-conditioning as a "warm-start" mechanism for latents to amortize the cost of effective information routing. Instead of reinitializing latents at each iteration, we use latents from previous iterations as additional context, similar to a recurrent network but without requiring backpropagation through time.

Our experiments with diffusion models show that RINs outperform U-Net architectures for image and video generation (see Fig. 2.3). For class-conditional ImageNet models, from 64×64 up to 1024×1024, RINs outperform leading diffusion models that use cascades [103] or guidance [44, 102], while consuming up to 10× fewer FLOPs per inference step. For video prediction, RINs surpass leading approaches [105] on the Kinetics600 benchmark while reducing the FLOPs of each step by 10×.

Our contributions are summarized as follows:

- We propose RINs, a domain-agnostic architecture for input-dependent allocation of computation that scales well to high dimensional data.
- We identify recurrent computation settings in which RINs thrive and advocate latent self-conditioning to amortize the cost of routing.
- Despite reduced inductive bias, this leads to significant performance and efficiency gains over 2D and 3D U-Nets in diffusion models for image and video generation.

2.2 Recurrent Interface Networks

In RINs, the interface is locally connected to the input space and initialized via a form of tokenization (e.g., patch embeddings), while the latents are decoupled from data and initialized as learnable embeddings. The basic RIN block allocates computation by *routing* information between the interface and the latents. By stacking multiple blocks, we can update the interface and latents incrementally, such that bottom-up and top-down context can inform routing in the next block (see Fig. 2.4). Finally, a readout function (e.g. a linear projection) produces the network's output from the final interface representation.

Since the interface is tied to data, it grows linearly with input size and may be large (e.g., thousands of vectors), while the number of latent units can be much smaller (e.g., hundreds of vectors). The computation operating directly on the interface (e.g. tokenization, read, write) is uniform across the input space, but is designed to be relatively light-weight, to limit the amount of uniform computation. The high-capacity processing is reserved for the latents, formed by reading information from the interface selectively, such that the bulk of the computation can be adapted to the structure and content of the input. When there is redundancy in the input, the latents can further compress the input for more efficient processing.

Compared to convolutional nets such as U-Nets [190, 101], RINs do not rely on fixed downsampling or upsampling for global computation. Compared to Transformers [235], RINs operate on sets of tokens with positional encoding for similar flexibility across input domains, but avoid pairwise attention across tokens to reduce compute and memory requirements per token. Compared to other decoupled architectures such as Perceiver [114, 113], alternating computation between interface and latents enables more expressive routing without a prohibitively large set of latents.

While RINs are versatile, their advantages are more pronounced in recurrent settings, where inputs may change gradually over time such that it is possible to propagate persistent context to further prime the routing of information. Therefore, here we focus on the application of RINs to iterative generation with diffusion models.

2.2.1 Iterative Generation with Diffusion Models

We first provide a brief overview of diffusion models [213, 101, 214, 215, 123, 35]. Diffusion models learn a series of state transitions to map noise $\boldsymbol{\epsilon}$ from a known prior distribution to $\boldsymbol{x}_0$ from the data distribution. To learn this (reverse) transition from noise to data, a forward transition from $\boldsymbol{x}_0$ to $\boldsymbol{x}_t$ is first defined:

$$\boldsymbol{x}_t = \sqrt{\gamma(t)}\ \boldsymbol{x}_0 + \sqrt{1-\gamma(t)}\ \boldsymbol{\epsilon},$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, $t \sim \mathcal{U}(0,1)$, and $\gamma(t)$ is a monotonically decreasing function from 1 to 0. Instead of directly learning a neural net to model the transition from $\boldsymbol{x}_t$ to $\boldsymbol{x}_{t-\Delta}$, one can learn a neural net $f(\boldsymbol{x}_t, t)$ to predict $\boldsymbol{\epsilon}$ from $\boldsymbol{x}_t$, and then estimate $\boldsymbol{x}_{t-\Delta}$ from the estimated $\tilde{\boldsymbol{\epsilon}}$ and $\boldsymbol{x}_t$. The objective for $f(\boldsymbol{x}_t, t)$ is thus the ℓ_2 regression loss:

$$\mathbb{E}_{t\sim\mathcal{U}(0,1),\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{1})}\|f(\sqrt{\gamma(t)}\ \boldsymbol{x}_0 + \sqrt{1-\gamma(t)}\ \boldsymbol{\epsilon}, t) - \boldsymbol{\epsilon}\|^2.$$

To generate samples from a learned model, we follow a series of (reverse) state transition $\boldsymbol{x}_1 \rightarrow \boldsymbol{x}_{1-\Delta} \rightarrow \cdots \rightarrow \boldsymbol{x}_0$. This is done by iteratively applying the denoising function f on each state $\boldsymbol{x}_t$ to estimate $\boldsymbol{\epsilon}$, and hence $\boldsymbol{x}_{t-\Delta}$, using transition rules as in DDPM [101] or DDIM [214]. As we will see, the gradual refinement of $\boldsymbol{x}$ through repeated application of the denoising function is a natural fit for RINs. The network takes as input a noisy image $\boldsymbol{x}_t$, a time step t, and an optional conditioning variable e.g. a class label y, and then outputs the estimated noise $\tilde{\boldsymbol{\epsilon}}$.

2.2.2 Elements of Recurrent Interface Networks

We next describe the major components of RINs, illustrated in Figure 2.4.

Interface Initialization. The interface is initialized from an input $\boldsymbol{x}$, such as an image $\boldsymbol{x}_{\text{image}} \in \mathbb{R}^{h\times w\times 3}$, or video $\boldsymbol{x}_{\text{video}} \in \mathbb{R}^{h\times w\times l\times 3}$ by tokenizing x into a set of n vectors $X \in \mathbb{R}^{n\times d}$. For example, we use a linear patch embedding similar to [47] to convert an image into a set of patch tokens; for video, we use 3-D patches. To indicate their location, patch embeddings are summed with (learnable) positional encodings. Beyond tokenization, the model is domain-agnostic, as X is simply a set of vectors.

Latent Initialization. The latents $Z \in \mathbb{R}^{m\times d'}$ are (for now) initialized as learned embeddings, independent of the input. Conditioning variables, such as class labels and time step t of diffusion models, are mapped to embeddings; in our experiments, we simply concatenate them to the set of latents, since they only account for two tokens. While n is linear in size of x, m is decoupled from the input size and can remain small even for large inputs, since routing adaptively selects information from the interface for processing.

RIN Block. The RIN block routes information by reading from X into Z, computing on Z, and writing updates back to X. We implement these operations with key components of

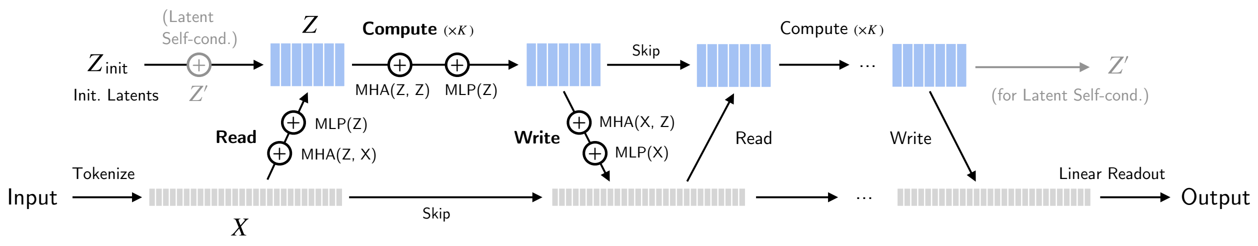


Figure 2.4: The RIN computation graph. RINs stack blocks that read, compute, and write. *Read* operations load information into latents with cross-attention. *Compute* operations exchange information across latent tokens with self-attention and across channels with token-wise MLPs. *Write* operations update the interface with information from the latents with cross-attention, and mix information across channels with token-wise MLPs. Latent self-conditioning (gray lines) allows for propagation of latent context between iterations.

Transformers:

$$\begin{aligned}
\textbf{Read:}\quad & Z = Z + \mathrm{MHA}(Z, X) \\
& Z = Z + \mathrm{MLP}(Z) \\
\textbf{Compute}:\quad & Z = Z + \mathrm{MHA}(Z, Z) \\
[\times K]\quad & Z = Z + \mathrm{MLP}(Z) \\
\textbf{Write:}\quad & X = X + \mathrm{MHA}(X, Z) \\
& X = X + \mathrm{MLP}(X)
\end{aligned}$$

MLP denotes a multi-layer perceptron, and MHA(Q, K) denotes multi-head attention with queries Q, and K being the keys and values.[2] The depth of latent computation layers K allows for controlling the ratio of computation occurring on the interface and latents. From the perspective of information exchange among hidden units, MHA propagates information across vectors (i.e. between latents, or between latents and interface), while the MLP (applied vector-wise, with shared weights) mixes information across their channels. Note that here computation on the interface is folded into the write operation, as MHA followed by an MLP.

RIN blocks can be stacked to allow latents to accumulate context and write incremental updates to the interface. To produce output predictions, we apply a readout layer (e.g. a linear projection) to the corresponding interface tokens to predict local outputs (such as patches of images or videos). The local outputs are then combined to form the desired output (e.g., patches are simply reshaped into an image). A detailed implementation is given in Appendix 2.6.1 (Alg 3).

[2]See [235] for details about multi-head attention, which extends single-head attention defined as $\mathrm{Attention}(Z, X) = \mathrm{softmax}(ZW_Q W_K^\top X^\top) X W_V$. $\mathrm{MLP}(Z) = \sigma(ZW_1 + b_1)W_2 + b_2$ where σ is the GELU activation function [96]. W are learned linear projections.

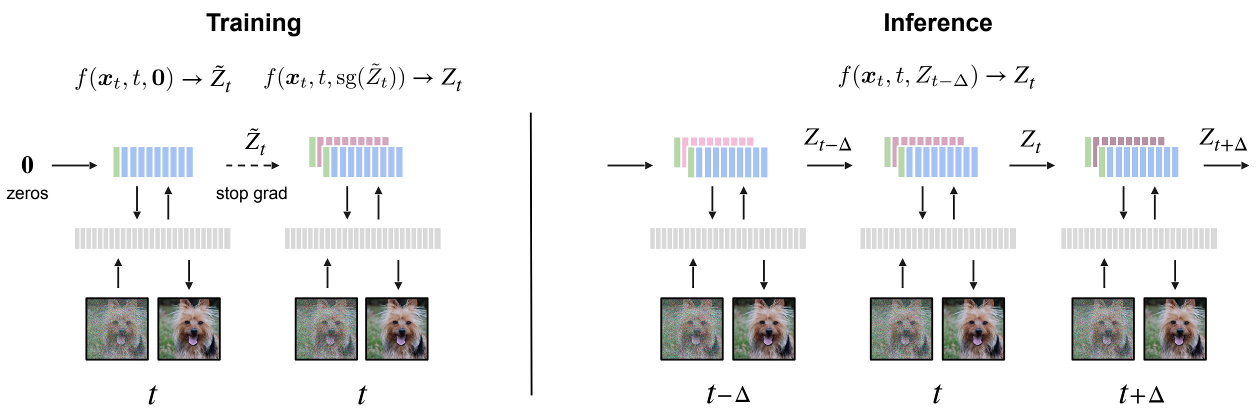

Training
$f(\boldsymbol{x}_t, t, \mathbf{0}) \to \tilde{Z}_t$
$f(\boldsymbol{x}_t, t, \mathrm{sg}(\tilde{Z}_t)) \to Z_t$
$\mathbf{0}$
zeros
$\tilde{Z}_t$
stop grad
t
t
Inference
$f(\boldsymbol{x}_t, t, Z_{t-\Delta}) \to Z_t$
$Z_{t-\Delta}$
Z_t
$Z_{t+\Delta}$
$t-\Delta$
t
$t+\Delta$

Figure 2.5: Latent Self-Conditioning for Diffusion Models with RINs. (Left) During training, latents for self-conditioning are first estimated with a forward pass of the denoising network (with zeros as previous latents); we then condition the denoising network with these estimated latents by treating them as latents of the previous iteration (without back-propagating through the estimated latents). (Right) During sampling, we start with zero latents, and use computed latents at each time-step to condition the next time-step.

2.2.3 Latent Self-Conditioning

RINs rely on routing information to dynamically allocate compute to parts of the input. Effective routing relies on latents that are specific to the input, and input-specific latents are built by reading interface information. This iterative process can incur additional cost that may overshadow the benefits of adaptive computation, especially if the network begins without context, i.e. from a “cold-start”. Intuitively, as humans, we face a similar “cold-start” problem under changes in the environment, requiring gradual familiarization of new state to enhance our ability to infer relevant information. If contexts switch rapidly without sufficient time for “warm-up”, we repeatedly face costly adaptation. The “warm-up” cost in RINs can be similarly amortized in sequential computation settings where inputs gradually change while global context persists. We posit that in such settings, there exists useful context in the latents accumulated in each forward pass.

Warm-starting Latents. With this in mind, we propose to “warm-start” the latents using latents computed at a previous step. Concretely, the initial latents at current time step t are the sum of the learnable embeddings Z_{emb} (independent of the input), and a transformation of previous latents from t' which are a function of correlated input at time step t':

$$Z_t = Z_{emb} + \mathrm{LayerNorm}(Z_{t'} + \mathrm{MLP}(Z_{t'}))\ ,$$

where LayerNorm [5] is initialized with zero scaling and bias, so $Z_t = Z_{emb}$ early in training.

In principle, this relies on the existence of latents from a previous time step, $Z_{t'}$, and requires unrolling iterations and learning with backpropagation through time, which can hamper scalability. A key advantage of diffusion models is that the chain of transitions

Algorithm 1 Training RINs with Latent Self-Cond.

```
def train_loss(x, self_cond_rate, latent_shape):
  # Add noise.
  t = uniform(0, 1)
  eps = normal(mean=0, std=1)
  x_t = sqrt(gamma(t)) * x + sqrt(1-gamma(t)) * eps

  # Compute latent self-cond estimate.
  latents = zeros(latent_shape)
  if uniform(0, 1) < self_cond_rate:
    _, latents = rin((x_t, latents), t)
    latents = stop_gradient(latents)

  # Predict and compute loss.
  eps_pred, _ = rin((x_t, latents), t)
  loss = (eps_pred - eps)**2
  return loss.mean()
```

Algorithm 2 Sampling with Latent Self-Cond.

```
def generate(steps):
  x_t = normal(mean=0, std=1)
  latents = zeros(latent_shape)

  for step in range(steps):
    # Get time for current and next states.
    t = 1 - step / steps
    t_m1 = max(1 - (step + 1) / steps, 0)

    # Predict eps.
    eps_pred, latents = rin((x_t, latents), t)

    # Estimate x at t_m1.
    x_t = ddim_or_ddpm_step(x_t, eps_pred, t, t_m1)

  return x_t
```

decomposes into conditionally independent steps allowing for highly parallelizable training, an effect we would like to preserve. To this end, we draw inspiration from the self-conditioning technique of [35], which conditions a denoising network at time t with its own unconditional prediction for time t.

Concretely, consider the conditional denoising network $f(\boldsymbol{x}_t, t, Z_{t'})$ that takes as input $\boldsymbol{x}_t$ and t, as well as context latents $Z_{t'}$. During training, with some probability, we use $f(\boldsymbol{x}_t, t, \mathbf{0})$ to directly compute the prediction $\tilde{\epsilon}_t$. Otherwise, we first apply $f(\boldsymbol{x}_t, t, \mathbf{0})$ to obtain latents $\tilde{Z}_t$ as an estimate of $Z_{t'}$, and compute the prediction with $f(\boldsymbol{x}_t, t, \mathrm{sg}(\tilde{Z}_t))$. Here, sg is the stop-gradient operation, used to avoid back-propagating through the latent estimates. At inference time, we directly use latents from previous time step t' to initialize the latents at current time step t, i.e., $f(\boldsymbol{x}_t, t, Z_{t'})$, in a recurrent fashion. This bootstrapping procedure marginally increases the training time ($< 25\%$ in practice, due to the stop-gradient), but has a negligible cost at inference time. In contrast to self-conditioning at the data level [35], here we condition on the latent activations of the neural network, so we call it *latent self-conditioning*.

Figure 2.5 illustrates the training and sampling process with the proposed latent self-conditioning. Algorithms 1 and 2 give the proposed modifications to training and sampling of the standard diffusion process. Details of common functions used in the algorithms can be found in Appendix 2.6.2.

Table 2.1: RIN configurations for each task.

	128px	256px	512px	1024px	Kinetics
$\lvert Z\rvert$	128	256	256	256	256
dim(Z)	1024	1024	768	768	1024
$\lvert X\rvert$	1024	1024	4096	16384	2048
dim(X)	512	512	512	512	512
Blocks	6	6	6	6	6
Depth K	4	4	6	8	4
Tokenization	4×4	8×8	8×8	8×8	2×4×4

2.3 Experiments

We demonstrate that RINs improve state-of-the-art performance on benchmarks for image generation and video prediction with pixel-space diffusion models. In all experiments, we do not use guidance. For each benchmark, we also compare the number of floating point operations (GFLOPs) across methods; as we will see, RINs are also more efficient. Samples and further visualizations are provided in Appendix 2.3.5 and the supplementary material.

2.3.1 Implementation Details

Noise Schedule. Similar to [123, 35], we use a continuous-time noise schedule function $\gamma(t)$. By default we use a cosine schedule, as in previous work [163] but find it is sometimes unstable for higher resolution images; we conjecture this is due to a relatively high probability of sampling extremely high or low noise levels, which contribute noise in gradients. We therefore explore schedules based the sigmoid function with different temperature, which shift weight away from the tails of the noise schedule. Detailed implementation of noise schedules and ablations are provided in Appendix 2.6.2. For large images, we report results obtained with models trained using input scaling[34, 36].

Tokenization and Readout. For image generation, we tokenize images by extracting non-overlapping patches followed by a linear projection. We use a patch size of 4 for 64×64 and 128×128 images, and 8 for larger images. To produce the output, we apply a linear projection to interface tokens and unfold each projected token to obtain predicted patches, which we reshape to form an image.

For video, we tokenize and produce predictions in the same manner as images; for 16×64×64 inputs, we use 2×4×4 patches, resulting in 2048 tokens. For conditional generation, during training, the context frames are provided as part of the input, without noise added. During sampling, the context frames are held fixed.

Table 2.1 compares model configuration across tasks. See Appendix 2.6.3 for detailed model and training hyper-parameters, and Appendix 2.6.1 for detailed pseudo-code of the full model.

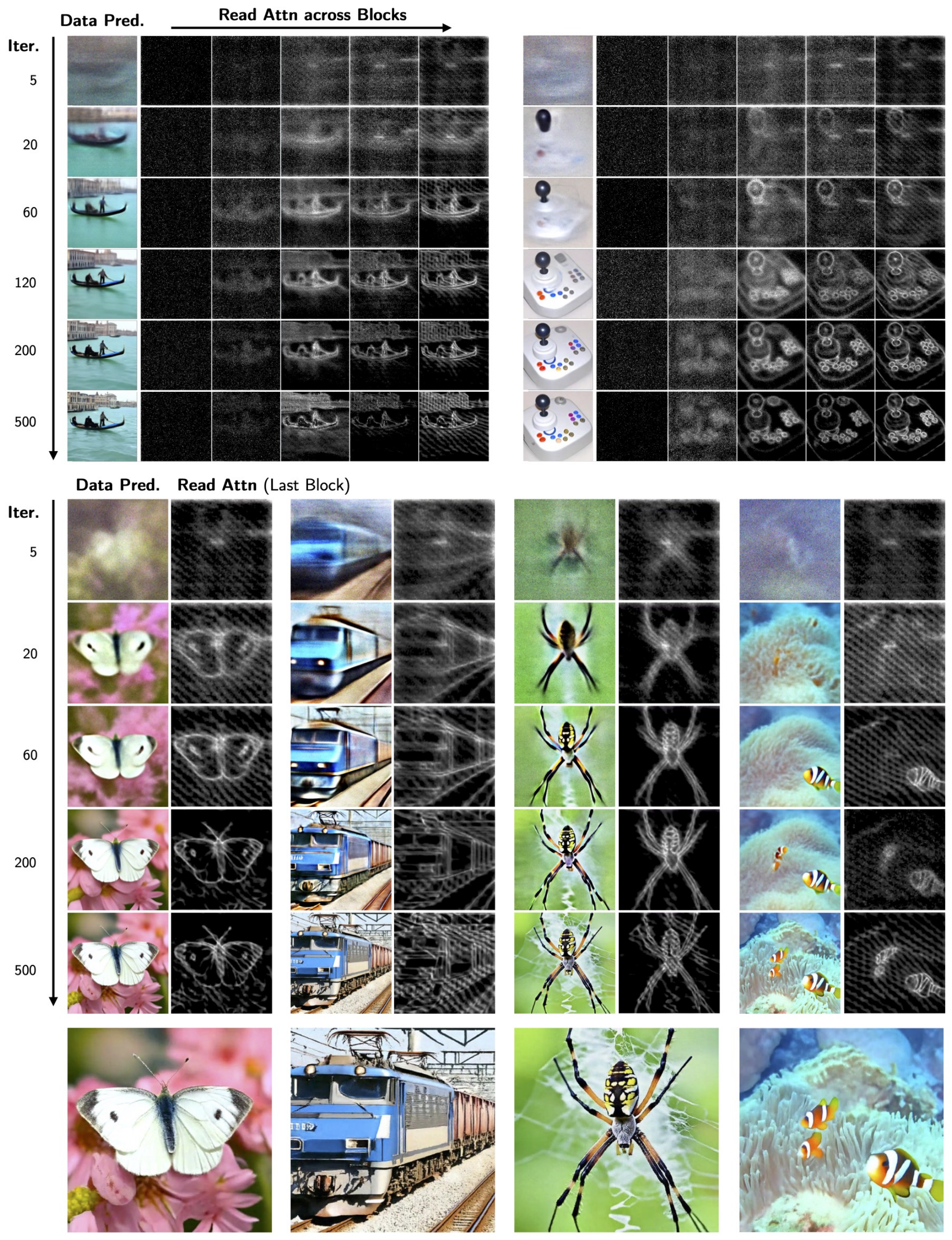


Figure 2.6: Visualizing Adaptive Computation. The read attention reveals which information is routed into latents for heavy computation. We visualize the read attention (averaged across latents) at each block (top) or the last block (bottom), at selected steps of the reverse process when generating ImageNet 512×512 samples. While it is similar across samples in early iterations, it becomes more sparse and data-specific, focusing latent computation on more complex regions.

2.3.2 Experimental Setup

For image generation, we mainly use the ImageNet dataset [193]. We only use center crops and random left-right flipping. We also use CIFAR-10 [131] to show the model can be trained with small datasets. For evaluation, we follow common practice, using FID [98] and Inception Score [196] as metrics computed on 50K samples, generated by 1000 steps of DDPM.

For video prediction, we use Kinetics-600 dataset [29] at 16×64×64 resolution. For evaluation, we follow common practice [105] and use FVD [233] and Inception Scores computed on 50K samples, generated by 400 or 1000 steps of DDPM.

2.3.3 Comparison to SOTA

Image Generation. Table 2.2 compares our architectures against existing state-of-the-art pixel-space diffusion models on ImageNet. Despite being fully attention-based and single-scale, our model attains superior generation quality (in both FID and IS) compared to existing models that rely on specialized convolutional architectures, cascaded generation, and/or class-guidance. Both the parameter count and FLOPs are significantly reduced in our model compared to baselines, which is useful for training performant models at higher resolutions without relying on cascades. For large images (512 and 1024), we report performance of RINs trained with input scaling [34]. We find that 256 latents are sufficient for strong performance even for 1024×1024 images, which produce 16384 tokens; this model are 2× more efficient than the 256×256 ADM UNet, despite operating at 4× higher resolution. Samples are shown in Fig. 2.2, 2.7 & 2.9).

Despite the lack of inductive bias, the model also works well with small datasets such as CIFAR-10. Compared to state-of-the-art FID of 1.79 EDM [120], we obtain 1.81 FID without using their improved sampling procedure. While our model has 31M parameters and trains in 3 hours on 8 TPUv3 chips, their model has 67M parameters and trains in 2 days on 8 A100 GPUs (faster accelerators).

Video Generation. Table 2.3 compares our model to existing methods on the Kinetics-600 Video Prediction benchmark. We follow common practice and use 5 conditioning frames. Despite the architecture's simplicity, RINs attain superior quality and are more efficient (up to 10× per step), without using guidance. Beyond using 3D patches instead of 2D patches, the architecture is identical to that used in 256×256 image generation; while the number of tokens is 2048, the model can attain strong performance with 256 latents. The model is especially suitable for video given the intense temporal redundancy, and learns to copy information and dedicate computation to regions of change, as discussed in Section 2.3.5. Samples are shown in Fig. 2.10.

2.3.4 Ablations

For efficiency, we ablate using smaller architectures (latent dimension of 768 instead of 1024) on the ImageNet 64×64 and 128×128 tasks with higher learning rate (2×10^{-3}) and fewer

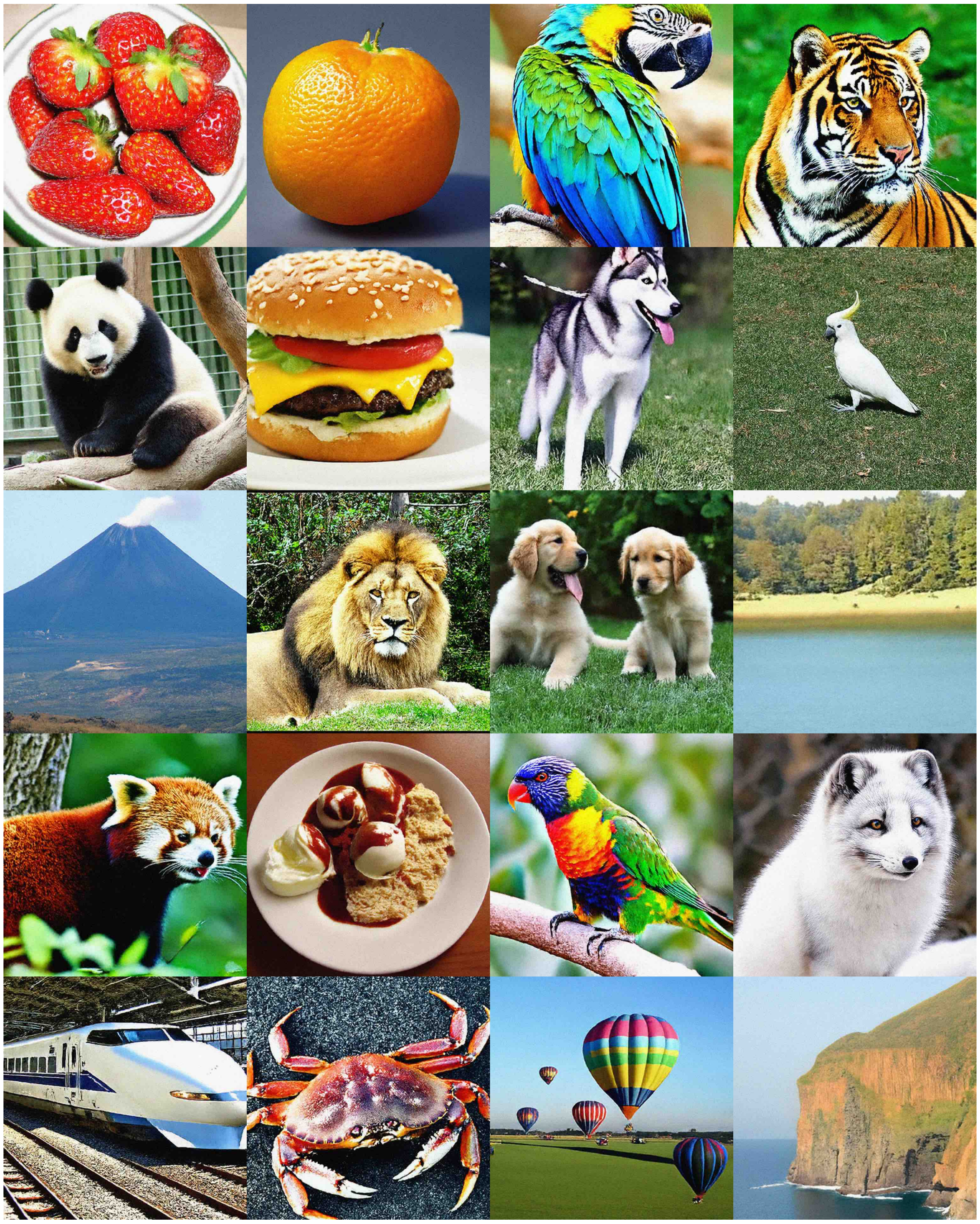


Figure 2.7: Class-conditional ImageNet 768×768 samples generated by RINs with 256 latents and 4096 tokens.

Table 2.2: Comparison to leading approaches for Class-Conditional Generation on ImageNet. †: use of class guidance, 1: [44], 2: [102], 3: [103].

Method	FID ↓	IS ↑	GFLOPs	Param (M)
IN 64×64				
ADM [1]	–	2.07	210	297
CF-guidance [2†]	1.55	66.0	–	–
CDM [3]	1.48	66.0	–	–
RIN	**1.23**	**66.5**	106	281
IN 128×128				
ADM [1]	5.91	–	538	386
ADM + guid. [1†]	2.97	–	>538	>386
CF-guidance [2†]	**2.43**	**156.0**	–	–
CDM [3]	3.51	128.0	1268	1058
RIN	2.75	144.1	194	410
IN 256×256				
ADM [1]	10.94	100.9	2212	553
ADM + guid.[1†]	4.59	186.7	>2212	>553
CDM[3]	4.88	158.7	2620	1953
RIN	4.51	161.0	334	410
RIN + input scaling	**3.42**	**182.0**	334	410
IN 512×512				
ADM [1]	23.2	58.1	4122	559
ADM + guid.[1†]	7.72	172.7	>4122	>559
RIN + input scaling	**3.95**	**216.0**	415	320
IN 1024×1024				
RIN + input scaling	**8.72**	**163.9**	1120	412

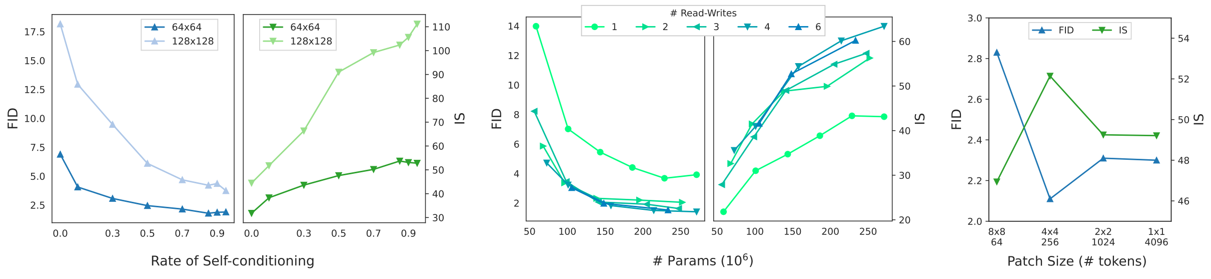
updates (150k and 220k, respectively). While this performs worse than our best models, it is sufficient for demonstrating the effect of different design choices.

Importance of Latent Self-conditioning. We study the effect of the rate of self-conditioning at training time. A rate of 0 denotes the special case where no self-conditioning is used (for training nor inference), while a rate > 0 e.g. 0.9 means that self-conditioning is used for 90% of each batch of training tasks (and always used at inference). As demonstrated in Figure 2.8a, there is a clear correlation between self-conditioning rate and sample quality (i.e., FID/IS), validating the importance using latent self-conditioning to provide context for enhanced routing. We use a rate of 0.9 for our best results reported.

Stacking Blocks. An important design choice in our architecture is the stacking of read-

Table 2.3: Video Prediction on Kinetics. †: reconstruction guidance. 1: [39], 2: [240], 3: [147], 4: [162], 5: [105].

Method	FVD	IS	GFLOPs	Param (M)
DVD-GAN-FP[1]	69.1	–	–	–
Video VQ-VAE[2]	64.3	–	–	–
TrIVD-GAN-FP[3]	25.7	12.54	–	–
Transframer[4]	25.4	–	–	–
Video Diffusion[5]†	16.6	15.64	4136	1100
RIN – 400 steps	11.5	17.7	386	411
RIN – 1000 steps	**10.8**	**17.7**	386	411



(a) Latent self-condition (b) Number of Blocks (c) Tokenization

Figure 2.8: Ablations. (a) Effect of the self-conditioning rate for training: self-conditioning is crucial; a rate of 0 is the special case of no self-conditioning. (b) Effect of the read-write/routing frequency: multiple rounds of read-writes are important to obtain the best result. (c) Effect of tokenization: the model can handle a large number (4096, with 1×1 patches in this case) of tokens on the inferface.

process-write blocks to enhance global and local processing. For a fair comparison, we analyze the effect of model size on generation quality for a variety of read-write frequencies (Fig. 2.8b) obtained by stacking blocks with varying number of processing layers per block. Note that a *single* read-write operation *without latent self-conditioning* is similar to architectures such as PerceiverIO [113]. With a single read-write, the performance saturates earlier as we increase model size. With more frequent read-writes, the model saturates later and with significantly better sample quality, validating the importance of iterative routing.

Tokenization. Recall that images are split into patches to form tokens on the interface. Fig. 2.8c shows that RINs can handle a wide range of patch sizes. For instance, it can scale to a large number of tokens (4096, for 1×1). While larger patch sizes force tokens to represent more information (i.e., with 8×8 patches), performance remains reasonable.

Effect of Noise Schedule. We find that the sigmoid schedule with an appropriate temperature is more stable training than the cosine schedule, particularly for larger images.

For sampling, the noise schedule has less impact and the default cosine schedule can suffice (see Figure 2.13).

2.3.5 Visualizing Adaptive Computation

To better understand the network's emergent adaptive computation, we analyze how information is routed by visualizing the attention distribution of read operations. For image generation, this reveals which parts of the image are most attended to for latent computation. Figure 2.6 shows the progression of two samples across the reverse process and the read attention (averaged over latents) through the blocks of the corresponding forward pass. As the generation progresses, the first read (guided by latent self-conditioning) is increasingly adapted to the sample. The read attention distribution becomes more sparse and favour regions of high information. Since the read attention loads information into the latents for high capacity computation, this suggests that the model learns to dynamically allocate computation on information as needed. Fig. 2.11 further shows similar phenomena in the video prediction setting, with the added effect of reading favouring information that cannot merely be copied from conditioning frames, such as object motion and panning.

2.4 Related Work

2.4.1 Neural architectures

Recurrent Interface Networks bear resemblance to architectures that leverage auxiliary memory to decouple computation from the input structure such as Memory Networks [251, 220], Neural Turing Machines [74], StackRNN [117], Set Transformer [137], Memory Transformers [23], Slot Attention [142], BigBird [260], and Workspace models [72]. While latents in our work are similar to auxiliary memory in prior work, we allocate the bulk of computation to latents and iteratively write back updates to the interface, rather than treating them simply as auxiliary memory. Recurrent Interface Networks are perhaps most similar to Set Transformers [136] and Perceivers [114, 113], which also leverage a set of latents for input-agnostic computation. Unlike these approaches, RINs alternate computation between the interface and latents, which is important for processing of information at both local and global levels without resorting to prohibitively many latents. Moreover, in contrast to existing architectures, latent self-conditioning allows RINs to leverage recurrence; this allows for propagation of routing context along very deep computation graphs to amortize the cost of iterative routing, which is crucial for achieving strong performance.

2.4.2 Adaptive computation

Other approaches for adaptive computation have mainly explored models with dynamic depth with recurrent networks [73, 57] or sparse computation [258], facing the challenges non-differentiability and dynamic or masked computation graphs. RINs are able to allocate

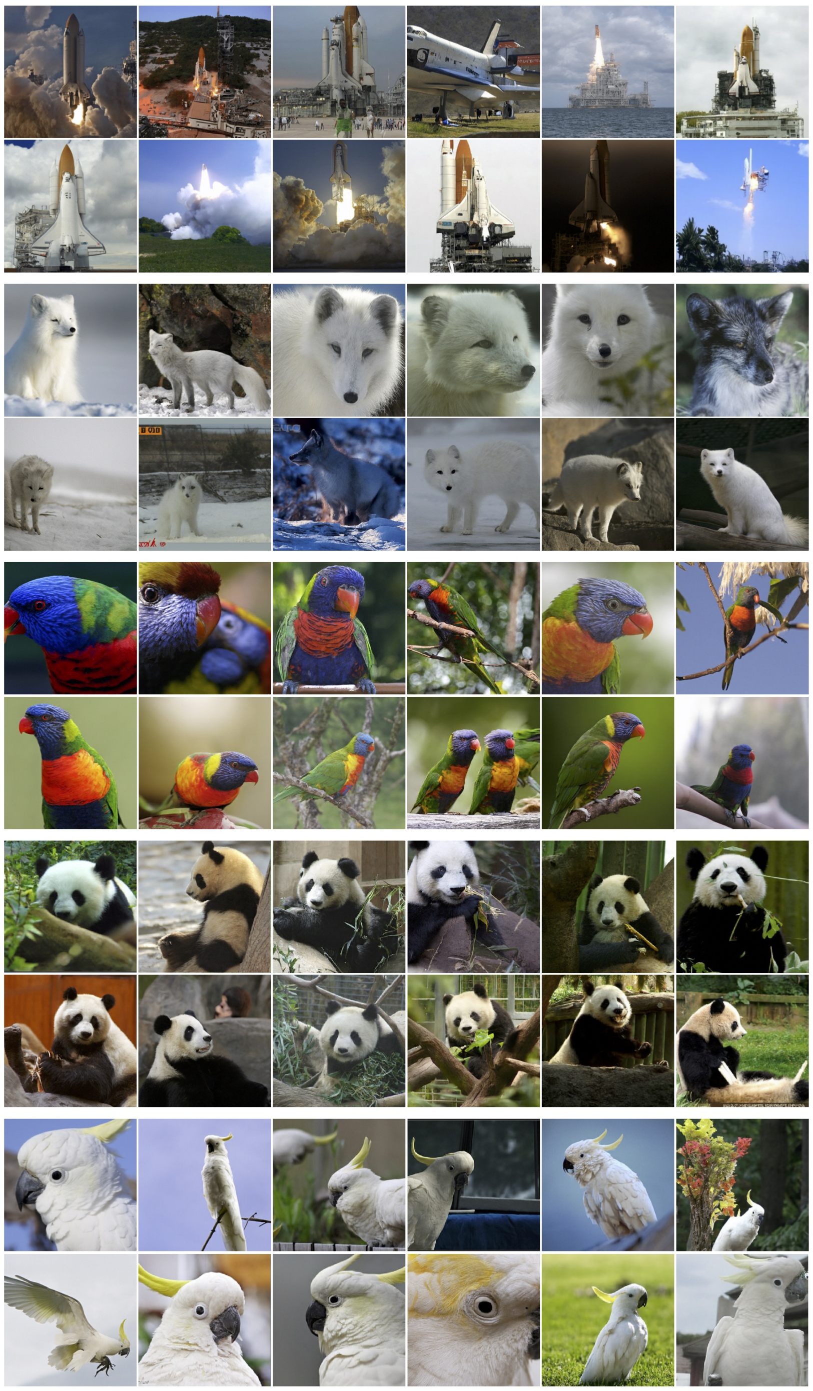


Figure 2.9: Class-conditional samples from a model trained on ImageNet 256×256. Classes from the top: space shuttle (812), arctic fox (279), lorikeet (90), giant panda (388), cockatoo (89).

compute non-uniformly despite having fixed computation graphs and being differentiable. RINs are closely related to recurrent models with input attention such as [80], but scale better by leveraging piecewise optimization enabled by diffusion models.

2.4.3 Diffusion models

Common diffusion models for images and videos can be roughly divided into pixel diffusion models [213, 101, 214, 44, 103, 120] and latent diffusion models [189]. In this work we focus on pixel diffusion models due to their relative simplicity. It is known to be challenging to train pixel diffusion models for high resolution images on ImageNet without guidance [44, 102] or cascades [103]. We show how improved architectures can allow for scaling pixel-level diffusion models to such large inputs without guidance and cascades, and we expect some insights to transfer to latent diffusion models [189].

The U-Net [190, 101] is the predominant architecture for image and video diffusion models [44, 103, 105]. While recent work [149] has explored pixel-level diffusion with Transformers, they have not been shown to attain strong performance or scale to large inputs. Concurrent work [176] has shown Transformers may be more tenable when combined with latent diffusion i.e. by downsampling inputs with large-scale pretrained VAEs, but reliance on uniform computation limits gracefully scaling to larger data. Our model suggests a path forward for simple performant and scalable iterative generation of images and video, comparing favourably to U-Nets in sample quality and efficiency, while based on domain-agnostic operations such as attention and fully-connected MLPs, and therefore more universal.

Self-conditioning for diffusion models was originally proposed in [35]. It bears similarity to step-unrolled autoencoders [198] and has been adopted in several existing work [219, 45, 36]. While these works condition on predictions of data, latent self-conditioning conditions a neural network on its own hidden activations, akin to recurrent neural network at inference while training without backpropagation through time.

2.5 Discussion

Recurrent Interface Networks are a family of neural networks that explicitly partition hidden units into the interface and latents. The interface links the input space to the core computation units operating on the latents, decoupling computation from data layout and allowing adaptive allocation of capacity to different parts of the input. We show the challenge of building latents can be amortized in recurrent computation settings – where the effective network is deep and persistent context can be leveraged – while still allowing for efficient training. While RINs are domain-agnostic, we found them to be performant and efficient for image and video generation tasks. As we look towards building more powerful generative models, we hope RINs can serve as a simple and unified architecture that scales to high-dimensional data across a range of modalities. To further improve RINs, we hope to better understand and enhance the effect of latent self-conditioning. Moreover, we hope to combine the advantages of RINs with orthogonal techniques, such as guidance and latent diffusion.

2.6 Appendix

2.6.1 Architecture Implementation Pseudo-code

2.6.2 More Details of Training / Sampling Algorithms, and Noise schedules

Algorithm 4 contains different choices of $\gamma(t)$, the continuous time noise schedule function.

Algorithm 4 Continuous time noise scheduling function.

```
def gamma_cosine_schedule(t, ns=0.0002, ds=0.00025):
  # A scheduling function based on cosine function.
  return numpy.cos(((t + ns) / (1 + ds)) * numpy.pi / 2)**2

def gamma_sigmoid_schedule(t, start=-3, end=3, tau=1.0, clip_min=1e-9):
  # A scheduling function based on sigmoid function.
  v_start = sigmoid(start / tau)
  v_end = sigmoid(end / tau)
  output = (-sigmoid((t * (end - start) + start) / tau) + v_end) / (v_end - v_start)
  return np.clip(output, clip_min, 1.)
```

Algorithm 5 contains DDIM [214] and DDPM [101] updating rules, as specified in [35].

Algorithm 5 x_t estimation with DDIM / DDPM updating rules.

```
def ddim_step(x_t, x_pred, t_now, t_next):
  # Estimate x at t_next with DDIM updating rule.
  $\gamma_{now}$ = gamma(t_now)
  $\gamma_{next}$ = gamma(t_next)
  x_pred = clip(x_pred, -scale, scale)
  eps = $\frac{1}{\sqrt{1-\gamma_{now}}}$ * (x_t - $\sqrt{\gamma_{now}}$ * x_pred)
  x_next = $\sqrt{\gamma_{next}}$ * x_pred + $\sqrt{1-\gamma_{next}}$ * eps
  return x_next

def ddpm_step(x_t, x_pred, t_now, t_next):
  # Estimate x at t_next with DDPM updating rule.
  $\gamma_{now}$ = gamma(t_now)
  $\alpha_{now}$ = gamma(t_now) / gamma(t_next)
  $\sigma_{now}$ = sqrt(1 - $\alpha_{now}$)
  z = normal(mean=0, std=1)
  x_pred = clip(x_pred, -scale, scale)
  eps = $\frac{1}{\sqrt{1-\gamma_{now}}}$ * (x_t - $\sqrt{\gamma_{now}}$ * x_pred)
  x_next = $\frac{1}{\sqrt{\alpha_{now}}}$ * (x_t - $\frac{1-\alpha_{now}}{\sqrt{1-\gamma_{now}}}$ * eps) + $\sigma_{now}$ * z
  return x_next
```

Sigmoid noise schedule. We use noise schedule function based the sigmoid function with different temperature (see Fig. 2.12a), which subtly shifts more weight to medium noise levels. We use a default temperature of 0.9, and its effect is ablated in our experiments.

2.6.3 Hyper-parameters and Other Training Details

We train most models on 32 TPUv3 chips with a batch size of 1024. Models for 512×512 and 1024×1024 are trained on 64 TPUv3 chips and 256 TPUv4 chips, respectively. All models are trained with the LAMB optimizer [259].

Table 2.4: Model Hyper-parameters.

Task	Input/Output	Blocks	$\|Z\|$	Z Dim	X Dim	Tokens $\|X\|$	Heads	Params	GFLOPs
IN	64×64	4×4	128	1024	256	256 (4×4)	16	280M	106
IN	128×128	6×4	128	1024	512	1024 (4×4)	16	410M	194
IN	256×256	6×4	256	1024	512	1024 (8×8)	16	410M	334
IN	512×512	6×6	256	768	512	4096 (8×8)	16	320M	415
IN	1024×1024	6×8	256	768	512	16384 (8×8)	16	415M	1120
Kin	16×64×64	6×4	256	1024	512	2048 (2×4×4)	16	411M	386

Table 2.5: Training Hyper-parameters.

Task	Input/Output	Updates	Batch Size	LR-decay	Optim β_2	WD	Self-cond.	EMA β
IN	64×64	300K	1024	cosine	0.999	0.01	0.9	0.9999
IN	128×128	600K	1024	cosine	0.999	0.001	0.9	0.9999
IN	256×256	600K	1024	cosine	0.999	0.001	0.9	0.9999
IN	512×512	1M	1024	cosine	0.999	0.01	0.9	0.9999
IN	1024×1024	1M	512	None	0.999	0.01	0.9	0.9999
Kin	16×64×64	500K	1024	cosine	0.999	0.001	0.85	0.99

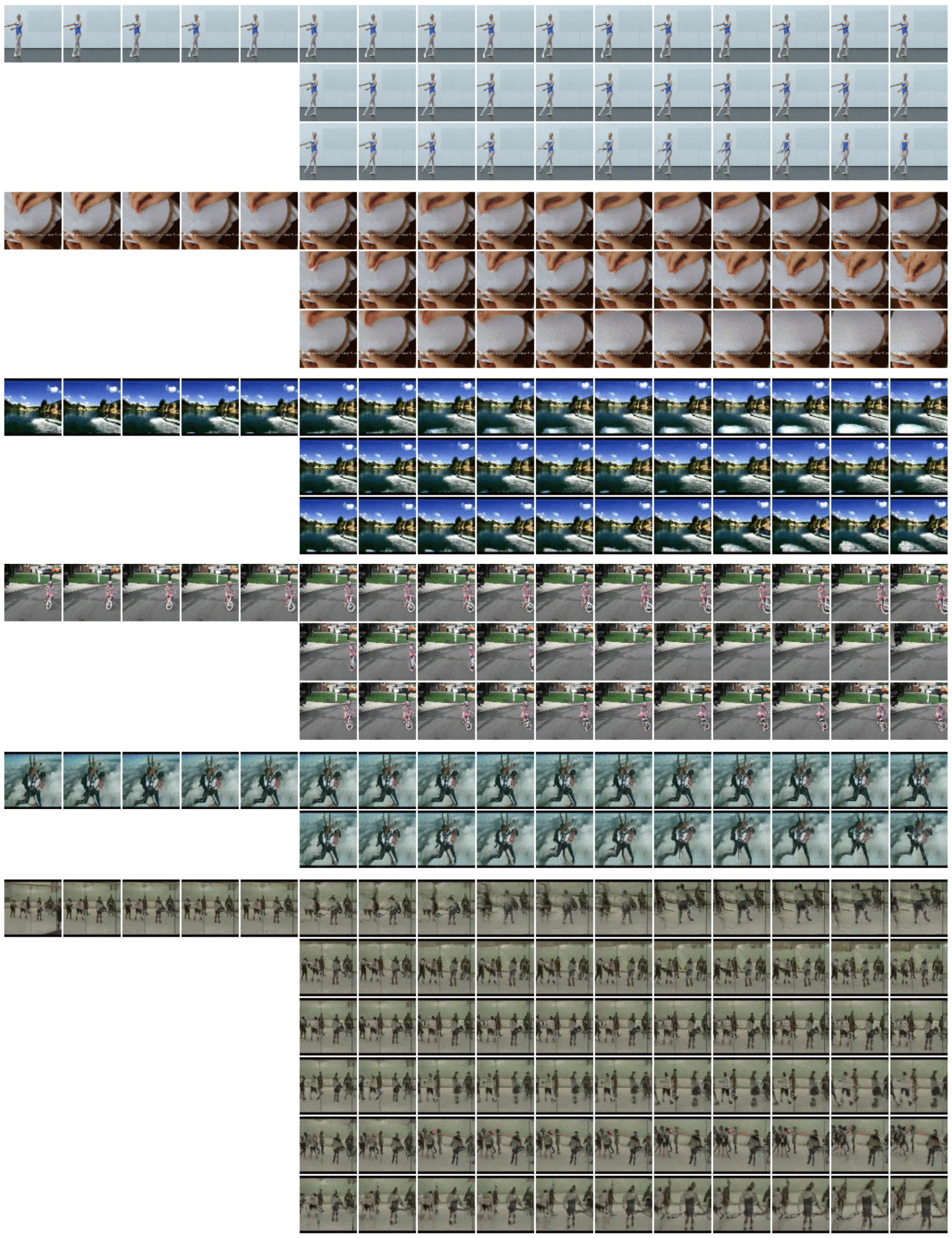


Figure 2.10: Selected samples of video prediction on Kinetics-600 at 16×64×64 showing examples of multi-modality across different future predictions, with conditioning frames from the test set. For example, the ballerina's arm and leg movements vary (first); the hand moves in different ways while sewing (second); the wakeboarder faces different waves (third); the bicyclist takes different turns; the sky-divers face different fates; the hockey scene (last) is zoomed and panned in different ways.

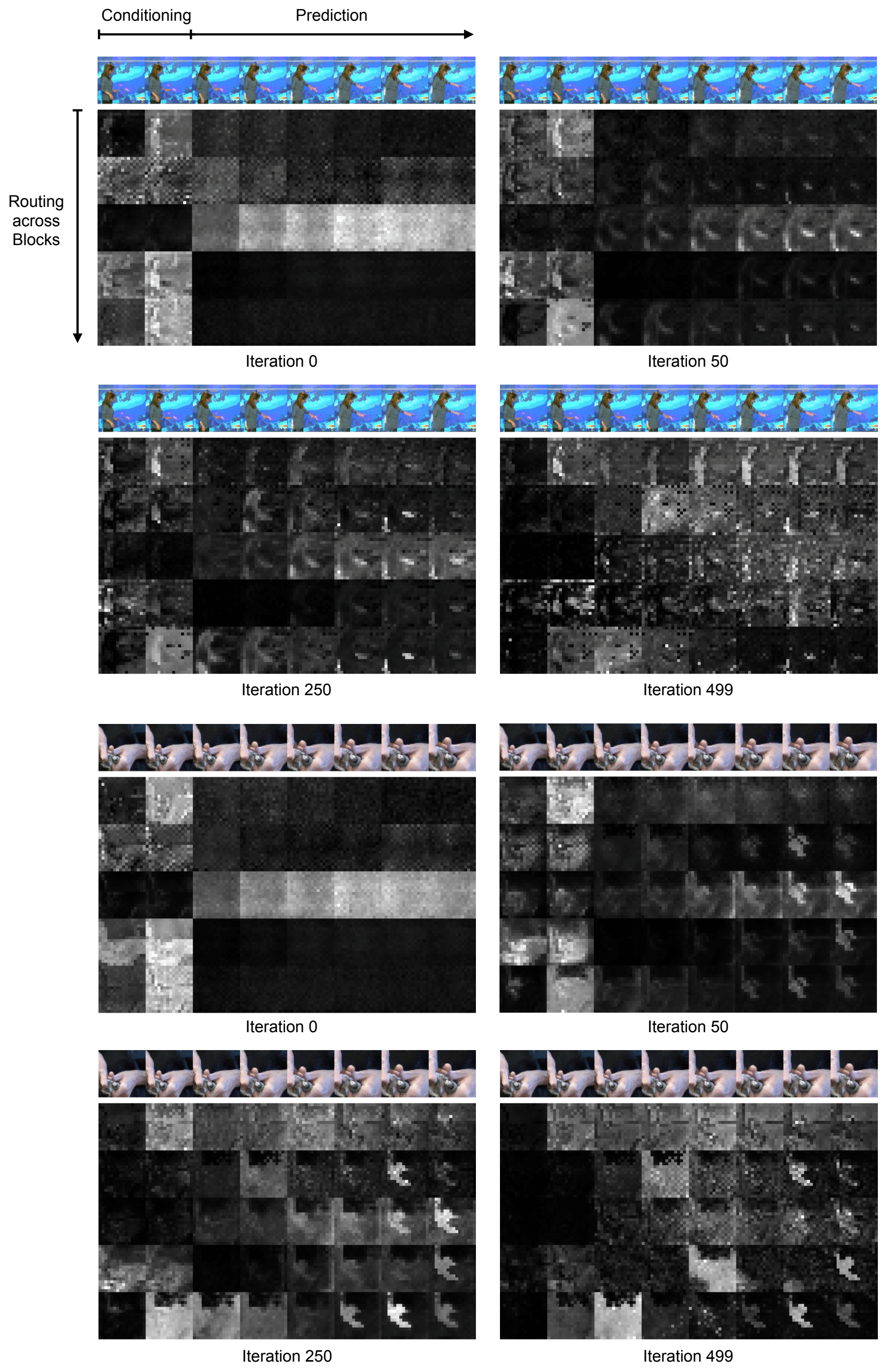
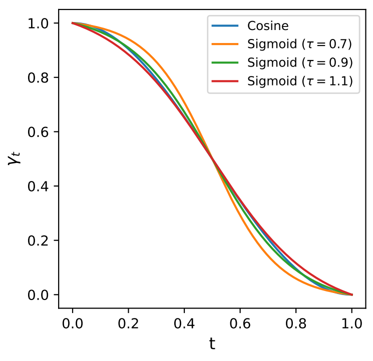


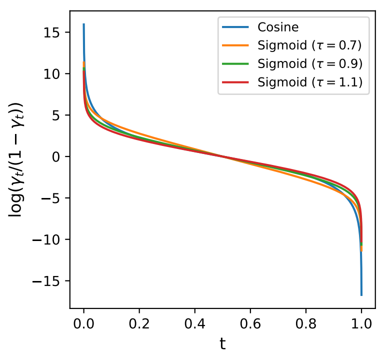
Figure 2.11: Visualization of emergent adaptive computation for video prediction on Kinetics-600. The samples are subsampled 2× in time to align with the attention visualization. In each column of the attention visualization, the first two columns are read attention on conditioning frames. We observe that read attention and hence computation is focused on regions of motion, that cannot be generated by simply copying from the conditioning frames.

Algorithm 3 RINs Implementation Pseudo-code.

```
def block(z, x, num_layers):
  """Core computation block."""
  z = z + multihead_attention(q=layer_norm(z), kv=x, n_heads=16)
  z = z + ffn(layer_norm(z), expansion=4)

  for _ in range(num_layers):
    zn = layer_norm(z)
    z = z + multihead_attention(q=zn, kv=zn, n_heads=16)
    z = z + ffn(layer_norm(z), expansion=4)

  x = x + multihead_attention(q=layer_norm(x), kv=z, n_heads=16)
  x = x + ffn(layer_norm(x), expansion=4)

  return z, x


def rin(x, patch_size, num_latents, latent_dim, interface_dim,
                        num_blocks, num_layers_per_block, prev_latents=None):
  """Forward pass of Network."""
  bsz, image_size, _, _ = x.shape
  size = image_size // patch_size

  # Initialize interface (with image tokenization as an example)
  x = conv(x, kernel_size=patch_size, stride=patch_size, padding='SAME')
  pos_emb = truncated_normal((1, size, size, dim), scale=0.02)
  x = layer_norm(x) + pos_emb


  # Initialize latents
  z = truncated_normal((num_latents, latent_dim), scale=0.02)

  # Latent self-conditioning
  if prev_latents is not None:
    prev_latents = prev_latents + ffn(stop_grad(prev_latents), expansion=4)
    z = z + layer_norm(prev_latents, init_scale=0, init_bias=0)

  # Compute
  for _ in range(num_blocks):
    z, x = block(z, x, num_layers_per_block)

  # Readout
  x = linear(layer_norm(x), dim=3*patch_size**2)
  x = depth_to_space(reshape(x, [bsz, size, size, -1]), patch_size)

  return z, x
```



(a) $\gamma(t)$



(b) $\log(\gamma(t)/(1-\gamma(t)))$

Figure 2.12: Compared to the cosine schedule, sigmoid (with appropriate τ) places less weight on extreme (high or low) noise levels.

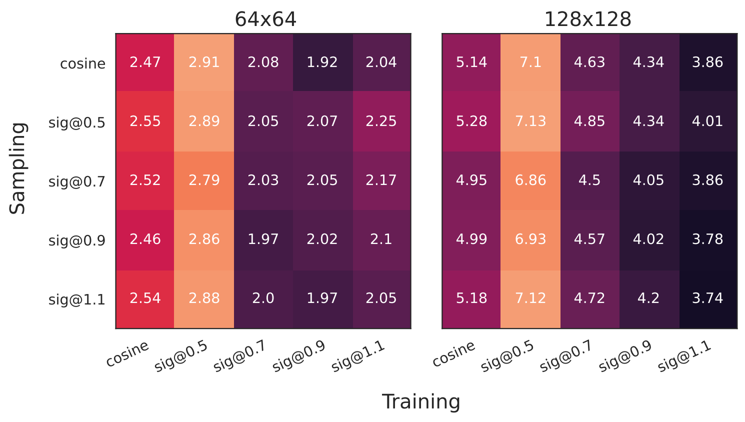


Figure 2.13: **Effect of noise schedule.** Comparing noise schedules for training and sampling, with corresponding FID score. The sigmoid schedule with an appropriate temperature is more stable during training than the widely used cosine schedule, particularly for larger images. For sampling, the noise schedule has less impact and the default cosine schedule can suffice.

Chapter 3

A Scalable Objective for Space-time Attention

In this chapter, we argue that temporal correspondence – the fundamental challenge in state representation of inferring *what goes where* – can be formulated as space-time attention, and propose a simple but general self-supervised objective for learning such an attention mechanism. We represent video as a graph, where nodes are e.g. image patches, and edges are attention weights between nodes of neighboring time-steps. Our aim is to learn a node representation such that temporal correspondence can be estimated by chaining attention in time to find paths between query and target nodes in the graph. This can be cast as a self-supervised contrastive learning problem by leveraging cycle-consistency, where the objective is to maximize the likelihood of returning to the initial node when walking along a graph constructed from a cyclic sequence (e.g. a palindrome) of frames. As a result, a single path-level constraint can implicitly supervise chains of intermediate comparisons.

The resulting similarity metric gives rise to an attention mechanism that can be used to propagate labels for object identity, part identity, and keypoints, which outperforms the self-supervised state-of-the-art on label propagation tasks involving objects, semantic parts, and pose. Moreover, we demonstrate that a technique we call edge dropout, as well as self-supervised adaptation at test-time, further improve transfer for object-centric correspondence.[1] Finally, we also show how the same objective can give rise to space-time attention for optical flow and object permanence.

3.1 Introduction

Video is often treated as a simple extension of an image into time, modeled as a spatio-temporal XYT volume [164, 262, 28]. Yet, treating time as yet another dimension is limiting [55]. One practical issue is the sampling rate mismatch between X and Y vs. T. But a more

[1]This work was published as *Space-time Correspondence as a Contrastive Random Walk* in NeurIPS 2020 [111].

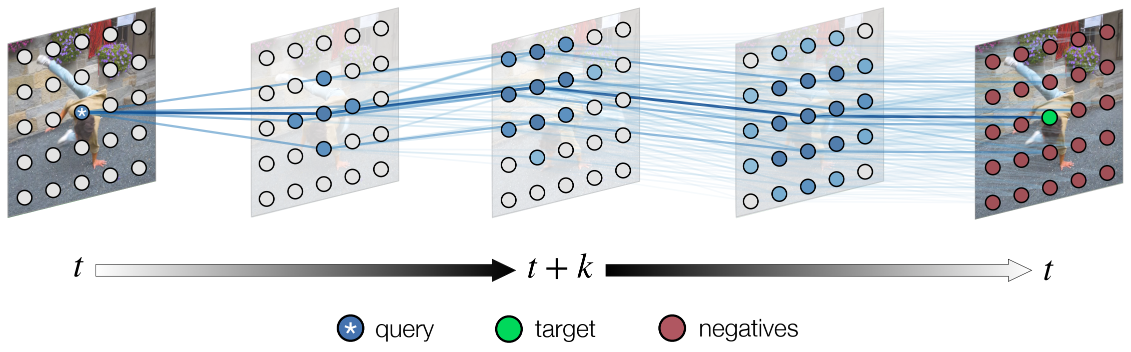


Figure 3.1: Video as a space-time graph. We represent video as a graph, where nodes are image patches, and edges are attention weights between nodes of neighboring frames. Our aim is to learn a node representation such that temporal correspondence is encoded by space-time attention, finding paths through the graph by chaining attention in time between query and target nodes. A contrastive loss encourages paths that reach the target, implicitly supervising latent correspondence along the path. Learning proceeds *without labels* by training on a *palindrome* sequence, walking from frame t to $t+k$, then back to t, using the initial node itself as the target

fundamental problem is that a physical point depicted at position (x, y) in frame t might not have any relation to what we find at that same (x, y) in frame $t+k$, as the object or the camera will have moved in arbitrary (albeit smooth) ways. This is why the notion of *temporal correspondence* — "what went where" [253] — is so fundamental for learning about objects in dynamic scenes, and how they inevitably change.

Recent approaches for self-supervised representation learning, such as those based on pairwise similarity learning [38, 48, 168, 205, 254, 100, 227, 94, 37], are highly effective when pairs of matching views are assumed to be known, e.g. constructed via data augmentation. Temporal correspondences, however, are *latent*, leading to a chicken-and-egg problem: we need correspondences to train our model, yet we rely on our model to find these correspondences. An emerging line of work aims to address this problem by bootstrapping an initially random representation to infer which correspondences should be learned in a self-supervised manner e.g. via cycle-consistency of time [245, 242, 139]. While this is a promising direction, current methods rely on complex and greedy tracking that may lead to local optima, especially when applied recurrently in time.

In this paper, we learn to associate features across space and time by formulating correspondence as pathfinding on a space-time graph. The graph is constructed from a video, where nodes are image patches and only nodes in neighboring frames share an edge. The strength of the edge is determined by similarity under a learned representation, whose aim is to place weight along paths linking visually corresponding patches (see Figure 3.1). Learning the representation amounts to fitting the transition probabilities of a walker stepping through

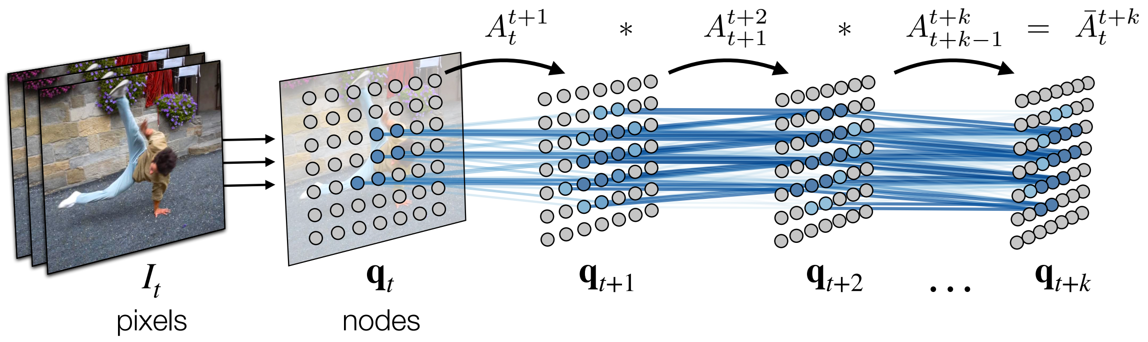


Figure 3.2: Correspondence as a Space-time Attention. We build a space-time graph by extracting nodes from each frame and allowing directed edges between nodes in neighboring frames. The transition probabilities of a random walk along this graph are determined by the attention matrix computed with pairwise similarity in the learned node representation.

time along the graph, reminiscent of the classic work of Meila and Shi [155] on learning graph affinities with a local random walk. This learning problem requires supervision — namely, the target that the walker should reach. In lieu of ground truth labels, we use the idea of cycle-consistency [266, 245, 242], by turning training videos into *palindromes*, e.g. sequences where the first half is repeated backwards. This provides every walker with a target — returning to its starting point. Under this formulation, we can view each step of the walk as a contrastive learning problem [38], where the walker's target provides supervision for entire chains of intermediate comparisons.

The central benefit of the proposed model is efficient consideration and supervision of many paths through the graph by computing the expected outcome of a random walk. This lets us obtain a learning signal from all views (patches) in the video simultaneously, and handling ambiguity in order to learn from harder examples encountered during training. Despite its simplicity, the method learns a representation that is effective for a variety of correspondence tasks. When used as a similarity metric without any adaptation, the representation outperforms state-of-the-art self-supervised methods on video object segmentation, pose keypoint propagation, and semantic part propagation. The model scales and improves in performance as the length of walks used for training increases. We also show several extensions of the model that further improve the quality of object segmentation, including an edge dropout [217] technique that encourages the model to group "common-fate" [250] nodes together, as well as test-time adaptation.

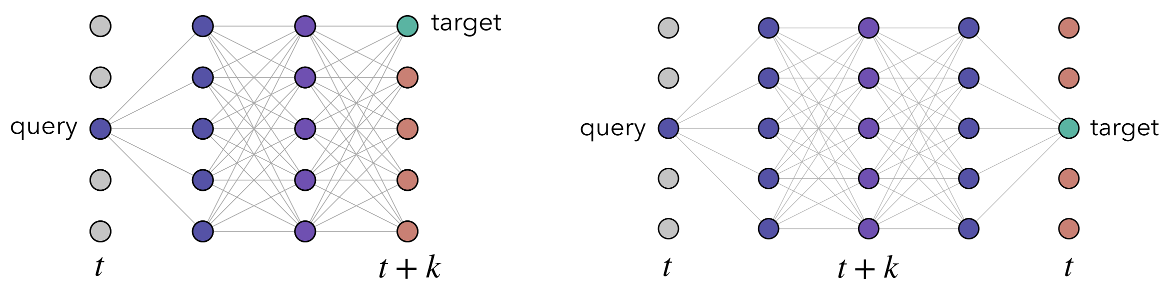


Figure 3.3: Learning Space-time Attention. (a) Specifying a target multiple steps in the future provides implicit supervision for *latent* correspondences along each path *(left)*. (b) We can construct targets for free by choosing palindromes as sequences for learning *(right)*.

3.2 Space-time Attention as a Contrastive Random Walk

We represent each video as a directed graph where nodes are patches, and weighted edges connect nodes in neighboring frames. Let $\mathbf{I}$ be a set of frames of a video and $\mathbf{q}_t$ be the set of N nodes extracted from frame $\mathbf{I}_t$, e.g. by sampling overlapping patches in a grid. An encoder ϕ maps nodes to l_2-normalized d-dimensional vectors, which we use to compute a pairwise similarity function $d_\phi(q_1, q_2) = \langle \phi(q_1), \phi(q_2) \rangle$ and an embedding matrix for $\mathbf{q}_t$ denoted $Q_t \in \mathbb{R}^{N \times d}$. We convert pairwise similarities into non-negative affinities by applying a softmax (with temperature τ) over edges departing from each node. For timesteps t and $t+1$, the stochastic matrix of affinities gives the attention matrix

$$A_t^{t+1}(i,j) = \texttt{softmax}(Q_t Q_{t+1}^\top)_{ij} = \frac{\exp(d_\phi(\mathbf{q}_t^i, \mathbf{q}_{t+1}^j)/\tau)}{\sum_{l=1}^N \exp(d_\phi(\mathbf{q}_t^i, \mathbf{q}_{t+1}^l)/\tau)}, \tag{3.1}$$

where the `softmax` is row-wise. Note that this describes only the *local* affinity between the patches of two video frames, $\mathbf{q}_t$ and $\mathbf{q}_{t+1}$. The affinity matrix for the entire graph, which relates all nodes in the video as a Markov chain, is block-sparse and composed of local affinity matrices.

Given the spatio-temporal connectivity of the graph, a step of a random walker on this graph can be viewed as performing tracking with attention, by *contrasting* similarity of neighboring nodes (using encoder ϕ). Let X_t be the state of the walker at time t, with transition probabilities $A_t^{t+1}(i,j) = P(X_{t+1} = j | X_t = i)$, where $P(X_t = i)$ is the probability of being at node i at time t. With this view, we can formulate long-range correspondence as walking multiple steps along the graph (Figure 3.2):

$$\bar{A}_t^{t+k} = \prod_{i=0}^{k-1} A_{t+i}^{t+i+1} = P(X_{t+k} | X_t). \tag{3.2}$$

Guiding the walk. Our aim is to train the embedding to encourage the random walker to follow paths of corresponding patches as it steps through time. While ultimately we will train without labels, for motivation suppose that we did have ground-truth correspondence between nodes in two frames of a video, t and $t+k$ (Figure **??**). We can use these labels to fit the embedding by maximizing the likelihood that a walker beginning at a *query* node at t ends at the *target* node at time $t+k$:

$$\mathcal{L}_{sup} = \mathcal{L}_{CE}(\bar{A}_t^{t+k}, Y_t^{t+k}) = -\sum_{i=1}^{N} \log P(X_{t+k} = Y_t^{t+k}(i) | X_t = i), \tag{3.3}$$

where $\mathcal{L}_{CE}$ is cross entropy loss and Y_t^{t+k} are correspondence labels for matching time t to $t+k$. Given the way transition probabilities are computed, the walk can be viewed as a chain of contrastive learning problems. Providing supervision at *every* step amounts to maximizing similarity between query and target nodes adjacent in time, while minimizing similarity to all other neighbors.

The more interesting case is supervision of longer-range correspondence, i.e. $k > 1$. In this case, the labels of t and $t+k$ provide *implicit* supervision for intermediate frames $t+1, ..., t+k-1$, assuming that latent correspondences exist to link t and $t+k$. Recall that in computing $P(X_{t+k}|X_t)$, we marginalize over all intermediate paths that link nodes in t and $t+k$. By minimizing $\mathcal{L}_{sup}$, we shift affinity to paths that link the query and target. In easier cases (e.g. smooth videos), the paths that the walker takes from each node will not overlap, and these paths will simply be reinforced. In more ambiguous cases – e.g. deformation, multi-modality, or one-to-many matches – transition probability may be split across latent correspondences, such that we consider distribution over paths with higher entropy. The embedding should capture similarity between nodes in a manner that hedges probability over paths to overcome ambiguity, while avoiding transitions to nodes that lead the walker astray.

3.2.1 Self-Supervision

How to obtain query-target pairs that are known to correspond, without human supervision? We can consider training on graphs in which correspondence between the first and last frames are known, by construction. One such class of sequences are *palindromes*, i.e. sequences that are identical when reversed, for which targets are known since the first and last frames are identical. Given a sequence of frames $(I_t, ..., I_{t+k})$, we form training examples by simply concatenating the sequence with a temporally reversed version of itself: $(I_t, ...I_{t+k}, ...I_t)$. Treating each query node's position as its own target (Figure **??**), we obtain the following cycle-consistency objective:

$$\mathcal{L}_{cyc}^{k} = \mathcal{L}_{CE}(\bar{A}_t^{t+k}\bar{A}_{t+k}^{t}, I) = -\sum_{i=1}^{N} \log P(X_{t+2k} = i | X_t = i) \tag{3.4}$$

By leveraging structure in the graph, we can generate supervision for chains of contrastive learning problems that can be made arbitrarily long. As the model computes a soft attention distribution at every time step, we can backpropagate error across – and thus learn from – the many alternate paths of similarity that link query and target nodes.

Contrastive learning with latent views. To better understand the model, we can interpret it as contrastive learning with latent views. The popular InfoNCE formulation [168] draws the representation of two views of the same example closer by minimizing the loss $\mathcal{L}_{CE}(U_1^2, I)$, where $U_1^2 \in \mathbb{R}^{n \times n}$ is the normalized affinity matrix between the vectors of the first and second views of n examples, as in Equation 3.1. Suppose, however, that we do not know *which* views should be matched with one another, merely that there should be a soft one-to-one alignment between them. We can formulate this as contrastive learning guided by a 'one-hop' cycle-consistency constraint, composing U_1^2 with the "transposed" stochastic similarity matrix U_2^1, to produce the loss $\mathcal{L}_{CE}(U_1^2 U_2^1, I)$, akin to Equation 3.4.

This task becomes more challenging with multiple hops, as avoiding spurious features that lead to undesirable diffusion of similarity across the graph becomes more important. While there are other ways of learning to align sets of features – e.g. by assuming soft bijection [40, 69, 256, 197] – it is unclear how they should extend to the multi-hop setting, where such heuristics may not always be desirable at each intermediate step. The proposed objective avoids the need to explicitly infer intermediate latent views, instead imposing a sequence-level constraint based on long-range correspondence known by construction.

3.2.2 Edge Dropout

One might further consider correspondence on the level of broader *segments*, where points within a segment have strong affinity to all other points in the segment. This inspires a trivial extension of the method – randomly dropping edges from the graph, thereby forcing the walker to consider alternative paths. We apply dropout [217] (with rate δ) to the transition matrix A to obtain $\tilde{A} = \texttt{dropout}(A, \delta)$, and then re-normalize. The resulting transition matrix B and noisy cycle loss are:

$$B_{ij} = \frac{\tilde{A}_{ij}}{\sum_l \tilde{A}_{il}} \qquad \mathcal{L}^k_{c\tilde{y}c} = \mathcal{L}_{CE}(B_t^{t+k} B_{t+k}^t, I).$$

Edge dropout affects the task by randomly obstructing paths, thus encouraging hedging of mass to paths correlated with the ideal path – i.e. paths of *common fate* [250] – similar to the effect in spectral-based segmentation [207, 155]. In practice, we apply edge dropout before normalizing affinities, by setting values to a negative constant. We will see in Section 3.3.2 that edge dropout improves object-centric correspondence.

3.2.3 Implementation

Algorithm 6 provides complete pseudocode for the method.

Algorithm 6 Pseudocode in PyTorch style.

```
for x in loader: # x: batch with B sequences
  # Split image into patches
  # B x C x T x H x W -> B x C x T x N x h x w
  x = unfold(x, (patch_size, patch_size))
  x = spatial_jitter(x)
  # Embed patches (B x C x T x N)
  v = l2_norm(resnet(x))

  # Transitions from t to t+1 (B x T-1 x N x N)
  A = einsum("bcti,bctj->btij",
             v[:,:,:-1], v[:,:,1:]) / temperature

  # Transition energies for palindrome graph
  AA = cat((A, A[:,::-1].transpose(-1,-2), 1)
  AA[rand(AA) < drop_rate] = -1e10 # Edge dropout
  At = eye(P)                      # Init. position

  # Compute walks
  for t in range(2*T-2):
     At = bmm(softmax(AA[:,t]), dim=-1), At)

  # Target is the original node
  loss = At[[range(P)]*B]].log()
```

`bmm`: batch matrix multiplication; `eye`: identity matrix; `cat`: concatenation.; `rand`: random tensor drawn from $(0, 1)$.

Pixels to Nodes. At training time, we follow [95], where patches of size 64×64 are sampled on a 7×7 grid from a 256×256 image (i.e. 49 nodes per frame). Patches are spatially jittered to prevent matching based on borders (see Appendix 3.7.3). At test time, we reuse the convolutional feature map between patches instead of processing the patches independently [143], making the features computable with only a single feed-forward pass of our network.

Encoder ϕ. We create an embedding for each image patch using a convolutional network, namely ResNet-18 [93] for fair comparison to baselines. We apply a linear projection and l_2 normalization after average pooling, obtaining a 128-dimensional vector. We reduce the stride of last two residual blocks (`res3` and `res4`) to be 1. Please see Appendix 3.7.8 for details.

Shorter paths. During training, we consider paths of multiple lengths. For a sequence of length T, we optimize all *sub*-cycles: $\mathcal{L}_{train} = \sum_{i=1}^{T} \mathcal{L}_{cyc}^{i}$. This loss encourages the sequence of nodes visited in the walk to be a palindrome, i.e. on a walk of length N, the node visited at step t should be the same node as $N - t$. It induces a curriculum, as short walks are easier to learn than long ones. This can be computed efficiently, since the losses share affinity matrices.

Training. We train ϕ using the (unlabeled) videos from Kinetics400 [28], with Algorithm 6. We used the Adam optimizer [124] for two million updates with a learning rate of 1×10^{-4}. We use a temperature of $\tau = 0.07$ in Equation 3.1, following [254] and resize frames to 256×256 (before extracting nodes, as above). Except when indicated otherwise, we report results with edge dropout rate 0.1 and a videos of length 10. Please find more details in Appendix 3.7.5.

3.3 Experiments

We evaluate the learned representation on video label propagation tasks involving objects, keypoints, and semantic parts, by using it as a similarity metric. We also study the effects of edge dropout, training sequence length, and self-supervised adaptation at test-time. In addition to comparison with the state-of-the-art, we consider a baseline of label propagation

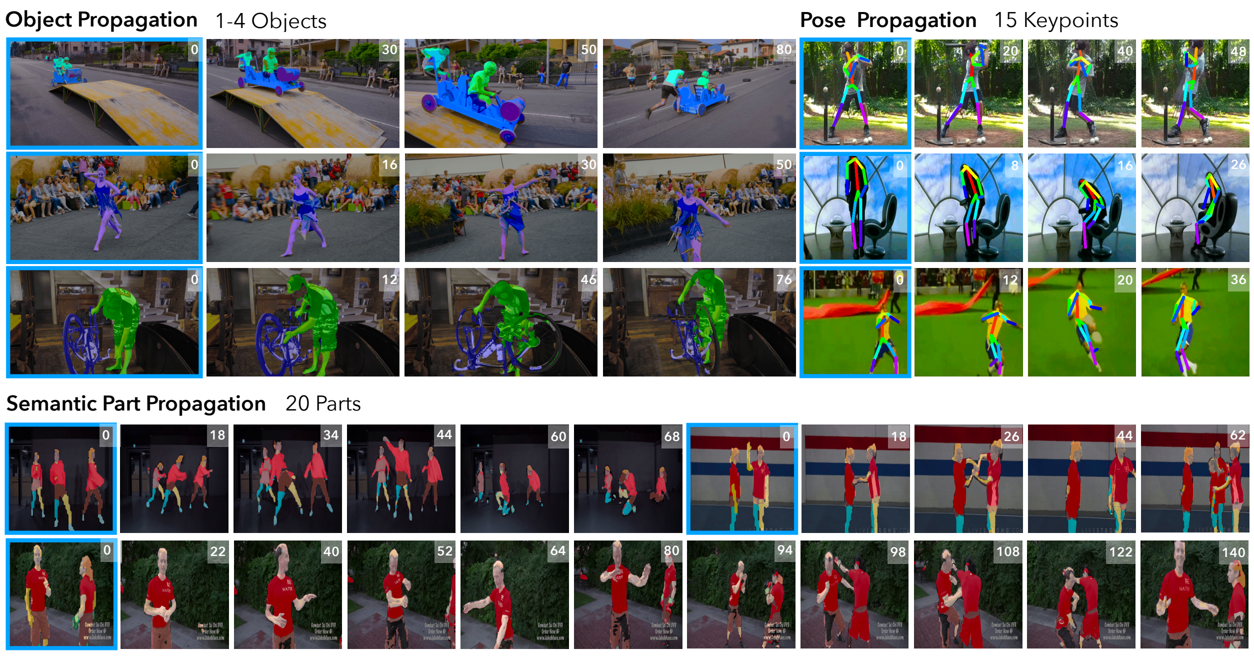

Object Propagation 1-4 Objects
Pose Propagation 15 Keypoints
Semantic Part Propagation 20 Parts
0
30
50
80
0
20
40
48
0
16
30
50
0
8
16
26
0
12
46
76
0
12
20
36
0
18
34
44
60
68
0
18
26
44
62
0
22
40
52
64
80
94
98
108
122
140

Figure 3.4: Qualitative results for label propagation under our model for object, pose, and semantic part propagation tasks. The first frame is indicate with a blue outline. Please see our webpage for video results, as well as a qualitative comparison with other methods.

with strong pre-trained features. Please find additional details, comparisons, ablations, and qualitative results in the Appendices.

3.3.1 Transferring the Learned Representation

We transfer the trained representation to label propagation tasks involving objects, semantic parts, and human pose. To isolate the effect of the representation, we use a simple inference algorithm based on k-nearest neighbors. Qualitative results are shown in Figure 3.4.

Label propagation. All evaluation tasks considered are cast as video label propagation, where the task is to predict labels for each pixel in *target* frames of a video given only ground-truth for the first frame (i.e. the *source*). We use the representation as a similarity function for prediction by k-nearest neighbors, which is natural under our model and follows prior work for fair comparison [245, 139].

Say we are given source nodes $\mathbf{q}_s$ with labels $L_s \in \mathbb{R}^{N \times C}$, and target nodes $\mathbf{q}_t$. Let K_t^s be the matrix of transitions between $\mathbf{q}_t$ and $\mathbf{q}_s$ (Equation 3.1), with the special property that only the top$-k$ transitions are considered per target node. Labels L_t are propagated as $L_t = K_t^s L_s$, where each row corresponds to the soft distribution over labels for a node, predicted by k-nearest neighbor in d_ϕ.

To provide temporal context, as done in prior work [245, 134, 139], we use a queue of the last m frames. We also restrict the set of source nodes considered to a spatial neighborhood of the query node for efficiency (i.e. *local* attention). The source set includes nodes of the first labeled frame, as well as the nodes in previous m frames, whose predicted labels are used for auto-regressive propagation. The softmax computed for K_t^s is applied over all source nodes. See Appendix 3.7.6 for further discussion and hyper-parameters.

Baselines. All baselines use ResNet-18 [93] as the backbone, modified to increase spatial resolution of the feature map by reducing the stride of the last two residual blocks to be 1. For consistency across methods, we use the output of the penultimate residual block as node embeddings at test-time.

Pre-trained visual features: We evaluate pretrained features from strong image- and video-based representation learning methods. For a strongly supervised approach, we consider a model trained for classification on **I**mageNet [42]. We also consider a strong self-supervised method, **MoCo** [94]. Finally, we compare with a video-based contrastive learning method, **VINCE** [70], which extends MoCo to videos (Kinetics) with views from data augmentation *and* neighbors in time.

Task-specific approaches: Wang et al. [245] uses cycle-consistency to train a spatial transformer network as a deterministic patch tracker. We also consider methods based on the Colorization approach of Vondrick et al. [239], including high-resolution methods: CorrFlow [134] and MAST [133]. CorrFlow combines cycle consistency with colorization. MAST uses a deterministic region localizer and memory bank for high-resolution colorization, and performs multi-stage training on [234]. Notably, both [134, 133] use feature maps that are significantly higher resolution than other approaches (2×) by removing max pooling from the network. Finally, UVC [139] jointly optimizes losses for colorization, grouping, pixel-wise cycle-consistency, and patch tracking with a deterministic patch localizer.

Video Object Segmentation

We evaluate our model on DAVIS 2017 [183], a popular benchmark for video object segmentation, for the task of semi-supervised multi-object (i.e. 2-4) segmentation. Following common practice, we evaluate on 480p resolution images. We apply our label propagation algorithm for all comparisons, except CorrFlow and MAST [134, 133], which require 4× more GPU memory. We report mean (m) and recall (r) of standard boundary alignment ($\mathcal{F}$) and region similarity ($\mathcal{J}$) metrics, detailed in [177].

As shown in Table 3.1, our approach outperforms other self-supervised methods, without relying on machinery such as localization modules or multi-stage training. We also outperform [133] despite being more simple at train and test time, and using a lower-resolution feature map. We found that when combined with a properly tuned label propagation algorithm, the more generic pretrained feature baselines fare better than more specialized temporal correspondence approaches. Our approach outperformed approaches such as MoCo [94] and VINCE [70], suggesting that it may not always be optimal to choose views for contrastive

Method	Resolution	Train Data	$\mathcal{J}\&\mathcal{F}_m$	$\mathcal{J}_m$	$\mathcal{J}_r$	$\mathcal{F}_m$	$\mathcal{F}_r$
VINCE [70]	1×	Kinetics	60.4	57.9	66.2	62.8	71.5
CorrFlow⋆ [134]	2×	OxUvA	50.3	48.4	53.2	52.2	56.0
MAST⋆ [133]	2×	OxUvA	63.7	61.2	73.2	66.3	78.3
MAST⋆ [133]	2×	YT-VOS	65.5	63.3	73.2	67.6	77.7
TimeCycle [245]	1×	VLOG	48.7	46.4	50.0	50.0	48.0
UVC+track⋆ [139]	1×	Kinetics	59.5	57.7	68.3	61.3	69.8
UVC [139]	1×	Kinetics	60.9	59.3	68.8	62.7	70.9
Ours w/ dropout	1×	Kinetics	**67.6**	**64.8**	**76.1**	**70.2**	**82.1**
w/ dropout & adapt.	1×	Kinetics	68.3	65.5	78.6	71.0	82.9

Table 3.1: Video object segmentation results on DAVIS 2017 val set Comparison of our method (2 variants), with previous self-supervised approaches and strong pretrained feature baselines. *Resolution* indicates if the approach uses a high-resolution (2x) feature map. *Train Data* indicates which dataset was used for pre-training. $\mathcal{F}$ is a boundary alignment metric, while $\mathcal{J}$ measures region similarity as IOU between masks. $\star$ indicates that our label propagation algorithm is not used.

learning by random crop data augmentation of frames. Finally, our model compares favorably to many supervised approaches with architectures designed for dense tracking [177, 24, 243] (see Appendix 3.7.2).

Pose Tracking

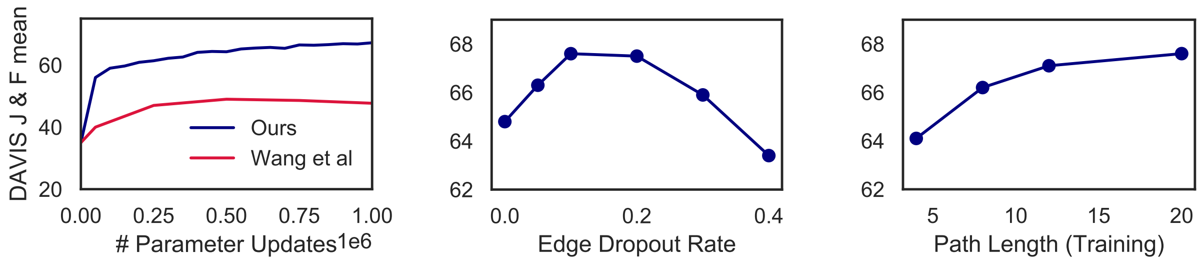
We consider pose tracking on the JHMDB benchmark, which involves tracking 15 keypoints. We follow the evaluation protocol of [139], using 320×320px images. As seen in Table 3.2, our model outperforms existing self-supervised approaches, including video colorization models that directly optimize for fine-grained matching with pixel-level objectives [139]. We attribute this success to the fact that our model sees sufficiently hard negative samples drawn from the same image at training time to learn features that discriminate beyond color. Note that our inference procedure is naive in that we propagate keypoints independently, without leveraging relational structure between them.

Video Part Segmentation

We consider the semantic part segmentation task of the Video Instance Parsing (VIP) benchmark [265], which involves propagating labels of 20 parts — such as arm, leg, hair, shirt, hand — requiring more precise correspondence than DAVIS. The sequences are longer and sampled at a lower frame rate. We follow the evaluation protocol of [139], using 560×560px images and $m = 1$. The model outperforms existing self-supervised methods, and when using more temporal context (i.e. $m = 4$), outperforms the baseline supervised approach of [265].

	Parts	Pose	
Method	mIoU	PCK@0.1	PCK@0.2
TimeCycle [245]	28.9	57.3	78.1
UVC [139]	34.1	58.6	79.6
Ours	36.0	59.0	83.2
Ours + context	**38.6**	**59.3**	**84.9**
ImageNet [93]	31.9	53.8	74.6
ATEN [265]	**37.9**	–	–
Yang et al. [257]	–	**68.7**	**92.1**

Table 3.2: Part and Pose Propagation tasks, with the VIP and JHMDB benchmarks, respectively. For comparison, we show supervised methods below.



(a) performance vs. training time (b) effect of edge dropout (c) effect of path length

Figure 3.5: Variations of the Model. (a) Downstream task performance as a function of training time. (b) Moderate edge dropout improves object-level correspondences. (c) Training on longer paths is beneficial. All evaluations are on the DAVIS segmentation task.

3.3.2 Variations of the Model

Edge dropout. We test the hypothesis (Figure 3.5b) that edge dropout should improve performance on the object segmentation task, by training our model with different edge dropout rates: {0, 0.05, 0.1, 0.2, 0.3, 0.4}. Moderate edge dropout yields a significant improvement on the DAVIS benchmark. Edge dropout simulates partial occlusion, forcing the network to consider reliable context.

Path length. We also asked how important it is for the model to see longer sequences during training, by using clips of length 2, 4, 6, or 10 (resulting in paths of length 4, 8, 12, or 20). Longer sequences yield harder tasks due to compounding error. We find that longer training sequences accelerated convergence as well as improved performance on the DAVIS task (Figure 3.5c). This is in contrast to prior work [245]; we attribute this success to considering multiple paths at training time via soft-attention, which allows for learning from longer sequences, despite ambiguity.

Improvement with training We found that the model's downstream performance on DAVIS improves as more data is seen during self-supervised training (Figure 3.5a). Compared to Wang et al [245], there is less indication of saturation of performance on the downstream task.

3.3.3 Self-supervised Adaptation at Test-time

A key benefit of not relying on labeled data is that training need not be limited to the training phase, but can continue during deployment [208, 160, 181, 222]. Our approach is especially suited for such adaptation, given the non-parametric inference procedure. We ask whether the model can be improved for object correspondence by fine-tuning the representation *at test time* on a novel video. Given an input video, we can perform a small number of iterations of gradient descent on the self-supervised loss (Algorithm 6) *prior* to label propagation. We argue it is most natural to consider an online setting, where the video is ingested as a stream and fine-tuning is performed continuously on the sliding window of k frames around the current frame. Note that only the raw, unlabeled video is used for this adaptation; we do not use the provided label mask. As seen in Table 3.1, test-time training improves object propagation. Interestingly, we see most improvement in the recall of the region similarity metric $\mathcal{J}_{recall}$ (which measures how often more than 50% of the object is segmented). More experiment details can be found in Appendix E.

3.4 Related Work

Temporal Correspondence. Many early methods represented video as a spatio-temporal XYT volume, where patterns, such as lines or statistics of spatio-temporal gradients, were computed for tasks like gait tracking [164] and action recognition [262]. Because the camera was usually static, this provided an implicit temporal correspondence via (x, y) coordinates. For more complex videos, optical flow [148] was used to obtain short-range explicit correspondences between patches of neighboring frames. However, optical flow proved too noisy to provide long-range composite correspondences across many frames. Object tracking was meant to offer robust long-range correspondences for a given tracked object. But after many years of effort (see [63] for overview), that goal was largely abandoned as too difficult, giving rise to "tracking as repeated detection" paradigm [187], where trained object detectors are applied to each frame independently. In the case of multiple objects, the process of "data association" resolves detections into coherent object tracks. Data association is often cast as an optimization problem for finding paths through video that fulfill certain constraints, e.g. appearance, position overlap, etc. Approaches include dynamic programming, particle filtering, various graph-based combinatorial optimization, and more recently, graph neural networks [263, 204, 15, 182, 33, 261, 118, 116, 129, 20, 26, 126]. Our work can be seen as contrastive data association via soft-attention, as a means for learning representations directly from pixels.

Graph Neural Networks and Attention. Representing inputs as graphs has led to unified deep learning architectures. Graph neural networks – versatile and effective across domains [235, 87, 128, 236, 246, 10, 26] – can be seen as learned message passing algorithms that iteratively update node representations, where propagation of information is dynamic, contingent on local and global relations, and often implemented as soft-attention. Iterative routing of information encodes structure of the graph for downstream tasks. Our work uses cross-attention between nodes of adjacent frames to learn to propagate node identity through a graph, where the task – in essence, instance discrimination across space and time – is designed to induce representation learning.

Graph Partitioning. Graphs have been widely used in image and video segmentation as a data structure. Given a video, a graph is formed by connecting pixels in spatio-temporal neighborhoods, followed by spectral clustering [206, 207, 64] or MRF/GraphCuts [19]. Most relevant is the work of Meila and Shi [155], which poses Normalized Cuts as a Markov random walk, describing an algorithm for learning an affinity function for segmentation by fitting the transition probabilities to be uniform within segments and zero otherwise. More recently, there has been renewed interest in the problem of unsupervised grouping [77, 127, 78, 53, 126]. Many of these approaches can be viewed as end-to-end neural architectures for graph partitioning, where entities are partitions of images or video inferred by learned clustering algorithms or latent variable models implemented with neural networks. While these approaches explicitly group without supervision, they have mainly considered simpler data. Our work similarly aims to model groups in dynamic scenes, but does so implicitly so as to scale to real, large-scale video data. Incorporating more explicit entity estimation is an exciting direction.

Graph Representation Learning. Graph representation learning approaches solve for distributed representations of nodes and vertices given connectivity in the graph [88]. Most relevant are similarity learning approaches, which define neighborhoods of positives with fixed (i.e. k-hop neighborhood) or stochastic (i.e. random walk) heuristics [178, 81, 225, 87], while sampling negatives at random. Many of these approaches can thus be viewed as fitting shallow graph neural networks with tasks reminiscent of Mikolov et al. [156]. Backstrom et al. [8] learns to predict links by supervising a random walk on social network data. While the above consider learning representations given a single graph, others have explored learning node embeddings given multiple graphs. A key challenge is inferring correspondence between graphs, which has been approached in prior work [256, 197] with efficient optimal transport algorithms [211, 40, 179]. We use graph matching as a means for representation learning, using cycle-consistency to supervise a chain of matches, without inferring correspondence between intermediate pairs of graphs. In a similar vein, cycle-consistency has also been shown to be a useful constraint for solving large-scale optimal transport problems [146].

Self-supervised Visual Representation Learning. Most work in self-supervised representation learning can be interpreted as data imputation: given an example, the task is to predict a part — or *view* — of its data given another view [11, 194, 38]. Earlier work leveraged unlabeled visual datasets by constructing *pretext* prediction tasks [46, 165, 264]. For video, temporal information makes for natural pretext tasks, including future prediction [71, 216, 153, 145, 151], arrow of time [158, 249], motion estimation [2, 115, 232, 140] or audio [172, 4, 171, 130]. The use of off-the-shelf tools to provide supervisory signal for learning visual similarity has also been explored [244, 67, 174]. Recent progress in self-supervised learning has focused on improving techniques for large-scale deep similarity learning, e.g. by combining the cross-entropy objective with negative sampling [84, 156]. Sets of corresponding views are constructed by composing combinations of augmentations of the same instance [48, 17, 254], with domain knowledge being crucial for picking the right data augmentations. Strong image-level visual representations can be learned by heuristically choosing views that are close in space [168, 100, 7, 94, 37], in time [205, 181, 89, 231, 70] or both [109, 227], even when relying on noisy negative samples. However, forcing random crops to be similar is not always desirable because they may not be in correspondence. In contrast, we implicitly determine which views to bring closer – a sort of automatic view selection.

Self-supervised Correspondence and Cycle-consistency. Our approach builds on recent work that uses cycle-consistency [266, 50] in time as supervisory signal for learning visual representations from video [245, 242]. The key idea in [245, 242] is to use self-supervised tracking as a pretext task: given a patch, first track forward in time, then backward, with the aim of ending up where it started, forming a cycle. These methods rely on trackers with hard attention, which limits them to sampling, and learning from, one path at a time. In contrast, our approach computes soft-attention at every time step, considering many paths to obtain a dense learning signal and overcome ambiguity. Li et al. [139] combines patch tracking with other losses including color label propagation [239], grouping, and cycle-consistency via an orthogonality constraint [68], considering pairs of frames at a time. Lai et al. [134, 133] refine architectural and training design decisions that yield impressive results on video object segmentation and tracking tasks. While colorization is a useful cue, the underlying assumption that corresponding pixels have the same color is often violated, e.g. due to lighting or deformation. In contrast, our loss is discriminative and permits association between regions that may have significant differences in their appearance.

3.5 Discussion

While data augmentation can be tuned to induce representation learning tasks involving invariance to color and local context, changes in other important factors of variation – such as physical transformations – are much harder to simulate. We presented a self-supervised approach for learning representations for space-time correspondence from unlabeled video data, based on learning to walk on a space-time graph. Under our formulation, a simple path-level constraint provides implicit supervision for a chain of contrastive learning problems. Our learning objective aims to leverage the natural data augmentation of dynamic scenes, *i.e.* how objects change and interact over time, and can be combined with other learning objectives. Moreover, it builds a connection between self-supervised representation learning and unsupervised grouping [155]. As such, we hope this work is a step toward learning to discover and describe the structure and dynamics of natural scenes from large-scale unlabeled video.

3.6 Extensions

As demonstrated, the CRW formulation can be used across a variety of space-time correspondence tasks. Here, we discuss two additional applications to the problems of optical flow and tracking with object permanence.

3.6.1 Optical Flow: Pixels as Nodes

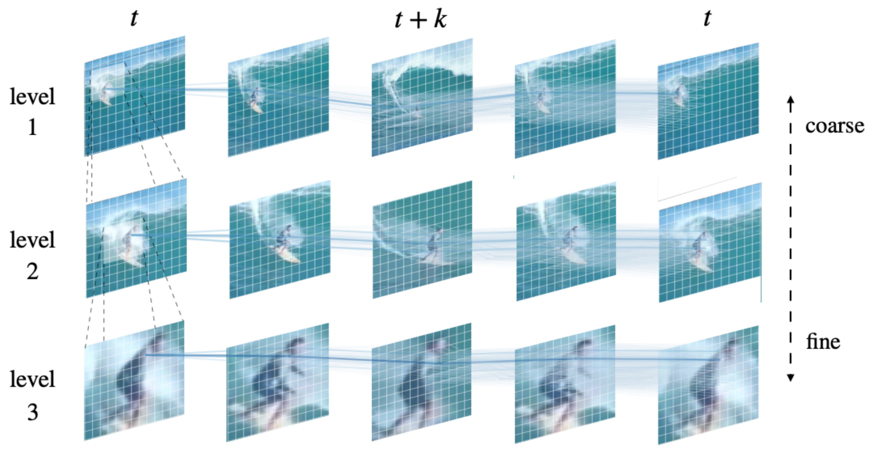


Figure 3.6: To scale to large space-time graphs (i.e. dense pixel graphs), we propose to use hierarchical attention, resulting in *multiscale* contrastive random walks.

The main challenge in applying CRW to the problem of optical flow is the quadratic cost of computing large attention matrices between pixels in high-resolution images, requiring much more dense pixel-level space-time graphs. To overcome this challenge, we can introduce hierarchy into the search problem by computing the attention matrix between two frames in a coarse-to-fine manner, forming a *multiscale* contrastive random walk when extended in time. This establishes a unified technique for self-supervised learning of optical flow, keypoint tracking, and video object segmentation.[2]

3.6.2 Object Permanence: Spatial Memories as Nodes

Object permanence is the idea that objects continue to exist even when they are not apparent. As presented, the CRW formulation makes the assumption that correspondence is Markovian.

[2]This work was published as *Learning Pixel Trajectories with Multiscale Contrastive Random Walks*, Bian et al, CVPR 2022 [16].

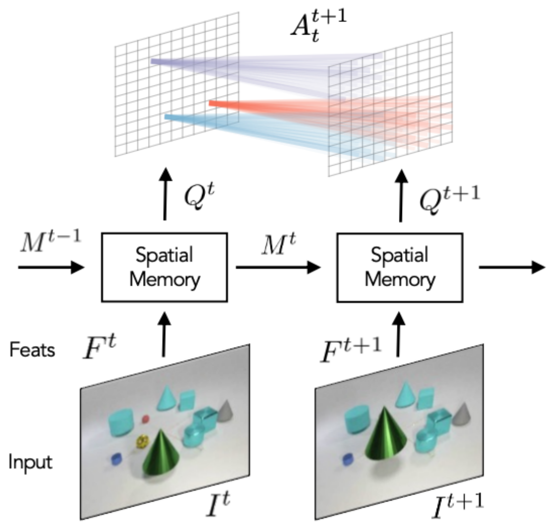


Figure 3.7: For object permanence, we can instead consider the spatio-temporal graph of an evolving spatial memory; here, we show one transition in time. To overcome partial observability, states Q_t are computed with a sequence encoder, allowing for transition probability A_t^{t+1} to model object permanence. Only a subset of the edges is shown for readability.

In reality, this assumption does not always hold due to partial observability, i.e. in cases of total occlusion. State-of-the-art approaches for object permanence, e.g. used for the task of video object detection or segmentation, typically rely on supervision during occlusion or simplifying assumptions such as constant velocity to overcome this challenge. Rather than directly supervising the locations of invisible objects, we can use the CRW objective to learn object permanence in a manner that does not requires neither human annotation nor assumptions about object dynamics. In particular, object permanence can emerge by optimizing for temporal coherence of memory: we fit a Markov walk along a space-time graph of spatial memories, where the states in each time step are non-Markovian features from a sequence encoder. This leads to a memory representation that stores occluded objects and predicts their motion, to better localize them. The resulting model outperforms existing approaches on several datasets of increasing complexity and realism, despite requiring minimal supervision, and hence being broadly applicable. [3]

[3]This work was published as *Object Permanence Emerges in a Random Walk along Memory*, Tokmakov et al, ICML 2022 [229].

3.7 Appendix

3.7.1 Derivation of Label Noise and Effect of Identical Patches

Here, we show that false negatives that are identical to the positive – for example, patches of the sky – do not change the sign of gradient associated with the positive. Let q be the query, u be the positive, V be the set of negatives. W.l.o.g, let the softmax temperature $\tau = 1$. The loss and corresponding gradient can be expressed as follows, where Z is the partition function:

$$L(q, u, V) = u^\top q - \log[\exp u^\top q + \sum_{v \in V} \exp v^\top q] = u^\top q - \log Z$$

$$\nabla_q L(q, u, V) = u - \frac{\exp u^\top q}{Z} u - \sum_{v \in V} \frac{\exp v^\top q}{Z} v = (1 - \frac{\exp u^\top q}{Z}) u - \sum_{v \in V} \frac{\exp v^\top q}{Z} v$$

Let V^- be the set of false negatives, such that $V^- \subseteq V$ and $V^+ = V \setminus V^-$. Consider the worst case, whereby $v_- = u, \forall v_- \in V^-$, so that false negatives are exactly identical to the positive:

$$\begin{aligned}
\nabla_q L(q, u, V) &= (1 - \frac{\exp u^\top q}{Z}) u - \sum_{v_- \in V^-} \frac{\exp v_-^\top q}{Z} v_- - \sum_{v_+ \in V^+} \frac{\exp v_+^\top q}{Z} v_+ \\
&= \underbrace{\left(1 - \frac{(1 + |V^-|) \exp u^\top q}{Z}\right)}_{\lambda_u} u - \sum_{v_+ \in V^+} \frac{\exp v_+^\top q}{Z} v_+
\end{aligned}$$

We see that the contribution of the negatives that are identical to the positive do not flip the sign of the positive gradient, i.e. $\lambda_u \geq 0$, so that in the worse case the gradient vanishes:

$$\begin{aligned}
\lambda_u &= 1 - \frac{(1 + |V^-|) \exp u^\top q}{Z} \\
&= 1 - \frac{(1 + |V^-|) \exp u^\top q}{(1 + |V^-|) \exp u^\top q + \sum_{v_+ \in V^+} \exp v_+^\top q} \\
&\geq 0
\end{aligned}$$

3.7.2 Comparison to Supervised Methods

The proposed method outperforms many supervised methods for video object segmentation, despite relying on a simple label propagation algorithm, not being trained for object segmentation, and not training on the DAVIS dataset. We also show comparisons to pretrained feature baselines with larger networks.

Method	Backbone	Train Data (#frames)	$\mathcal{J}\&\mathcal{F}_\mathrm{m}$	$\mathcal{J}_\mathrm{m}$	$\mathcal{J}_\mathrm{r}$	$\mathcal{F}_\mathrm{m}$	$\mathcal{F}_\mathrm{r}$
OSMN [257]	VGG-16	I/C/D (1.2M + 227k)	54.8	52.5	60.9	57.1	66.1
SiamMask [243]	ResNet-50	I/V/C/Y (1.2M + 2.7M)	56.4	54.3	62.8	58.5	67.5
OSVOS [24]	VGG-16	I/D (1.2M + 10k)	60.3	56.6	63.8	63.9	73.8
OnAVOS [237]	ResNet-38	I/C/P/D (1.2M + 517k)	65.4	61.6	67.4	69.1	75.4
OSVOS-S [152]	VGG-16	I/P/D (1.2M + 17k)	68.0	64.7	74.2	71.3	80.7
FEELVOS [238]	Xception-65	I/C/D/Y (1.2M + 663k)	71.5	69.1	79.1	74.0	83.8
PReMVOS [150]	ResNet-101	I/C/D/P/M (1.2M + 527k)	77.8	73.9	83.1	81.8	88.9
STM [166]	ResNet-50	I/D/Y (1.2M + 164k)	81.8	79.2	-	84.3	-
ImageNet [93]	ResNet-50	I (1.2M)	66.0	63.7	74.0	68.4	79.2
MoCo [94]	ResNet-50	I (1.2M)	65.4	63.2	73.0	67.6	78.7
Ours	ResNet-18	K (20M unlabeled)	67.6	64.8	76.1	70.2	82.1

Table 3.3: Video object segmentation results on DAVIS 2017 val set. We show results of state-of-the-art supervised approaches in comparison to our unsupervised one (see main paper for comparison with unsupervised methods). Key for *Train Data* column: I=ImageNet, K=Kinetics, V = ImageNet-VID, C=COCO, D=DAVIS, M=Mapillary, P=PASCAL-VOC Y=YouTube-VOS. $\mathcal{F}$ is a boundary alignment metric, while $\mathcal{J}$ measures region similarity as IOU between masks.

3.7.3 Using a Single Feature Map for Training

We follow the simplest approach for extracting nodes from an image without supervision, which is to simply sample patches in a convolutional manner. The most efficient way of doing this would be to only encode the image once, and pool the features to obtain region-level features [143].

We began with that idea and found that the network could cheat to solve this dense correspondence task even across long sequences, by learning a shortcut. It is well-known that convolutional networks can learn to rely on boundary artifacts [143] to encode position information, which is useful for the dense correspondence task. To control for this, we considered: 1) removing padding altogether; 2) reducing the receptive field of the network to the extent that entries in the center crop of the spatial feature map do not see the boundary; we then cropped the feature map to only see this region; 3) randomly blurring frames in each video to combat space-time compression artifacts; and 4) using *random* videos made of noise. Surprisingly, the network was able to learn a shortcut in each case. In the case of random videos, the shortcut solution was not nearly as successful, but we still found it surprising that the self-supervised loss could be optimized at all.

3.7.4 Frame-rate Ablation

Frame rate	$\mathcal{J}\&\mathcal{F}_\mathrm{m}$
2	65.9
4	67.5
8	67.6
30	62.3
∞	57.5

Effect of frame-rate at training time We ablate the effect of frame-rate (i.e. frames per second) used to generate sequences for training, on downstream object segmentation performance. The case of infinite frame-rate corresponds to the setting where the *same* image is used in each time step; this experiment is meant to disentangle the effect of data augmentation (spatial jittering of patches) from the natural "data augmentation" observed in video. We observe that spatio-temporal transformations is beneficial for learning of representations that transfer better for object segmentation.

3.7.5 Hyper-parameters

We list the key hyper-parameters and ranges considered at training time. Due to computational constraints, we did not tune the patch extraction strategy, nor several other hyper-parameters. The hyper-parameters varied, namely edge dropout and video length, were ablated in Section 3 (shown in bold). Note that the effective training path length is twice that of the video sequence length.

Train Hyper-parameters	Values
Learning rate	0.0001
Temperature τ	0.07
Dimensionality d of embedding	128
Frame size	256
Video length	**2, 4, 6, 10**
Edge dropout	**0, 0.05, 0.1, 0.2, 0.3**
Frame rate	**2, 4, 8, 30**
Patch Size	64
Patch Stride	32
Spatial Jittering (crop range)	(0.7, 0.9)

We tuned test hyper-parameters with the ImageNet baseline. In general, we found performance to increase given more context. Here, we show hyper-parameters used in reported experiments; we largely follow prior work, but for the case of DAVIS, we used 20 frames of context.

Test Hyper-parameters	Values
Temperature τ	0.07
Number of neighbors k	**10**, 20
Number of context frames m	Objects: 20
	Pose: 7
	Parts: 4
Spatial radius of source nodes	**12**, 20

3.7.6 Label Propagation

We found that the performance of baselines can be improved by carefully implementing label propagation by k-nearest neighbors. When compared to baseline results reported in [139] and [133], the differences are:

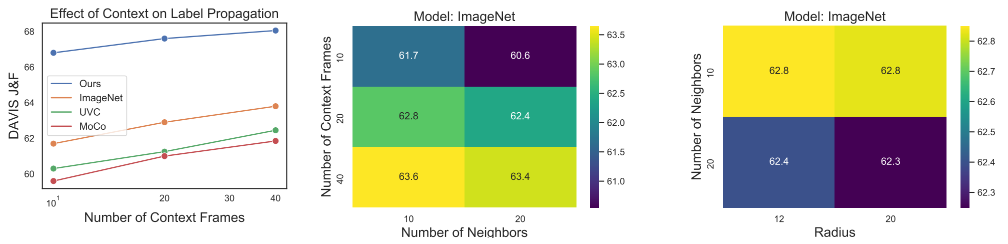
1. Restricting the set of source nodes (context) considered for each target node, on the basis of spatial locality, i.e. *local* attention. This leads to a gain of +4% J&F for the ImageNet baseline.

   Many of the task-specific approaches for temporal correspondence incorporate restricted attention, and we found this rudimentary form to be effective and reasonable.

2. Computing attention over all source nodes at once and selecting the top-k, instead of independently selecting the top-k from each frame. This leads to a gain of +3% J&F for the ImageNet baseline.

   This is more natural than computing nearest neighbors in each frame individually, and can be done efficiently if combined with local attention. Note that the softmax over context can be performed after nearest neighbors retrieval, for further efficiency.

3.7.7 Effect of Label Propagation Hyper-parameters



We study the effect of hyper-parameters of the label propagation algorithm, when applied with strong baselines and our method. The key hyper-parameters are the length of context m, the number of neighbors k, and the search radius r. In the figures above, we see the benefit of adding context (see left, with $k = 10, r = 12$), effect of considering more neighbors (middle, with $r = 12$), and effect of radius (right, with $m = 20$).

3.7.8 Encoder Architecture

We use the ResNet-18 network architecture, modified to increase the resolution of the output convolutional feature map. Specifically, we modify the stride of convolutions in the last two residual blocks from 2 to 1. This increases the resolution by a factor of four, so that the downsampling factor is 1/8. Please refer to Table 3.4 for a detailed description.

For evaluation, when applying our label propagation algorithm, we report results using the output of `res3` as node embeddings, for fair comparison to pretrained feature baselines ImageNet, MoCo, and VINCE, which were trained with stride 2 in `res3` and `res4`. We also found that `res3` features compared favorably to `res4` features.

Layer	Output	Details
`input`	$H \times W$	
`conv1`	$H/2 \times W/2$	7×7, 64, stride 2
`maxpool`	$H/4 \times W/4$	stride 2
`res1`	$H/4 \times W/4$	$\begin{bmatrix}3\times3,64\\3\times3,64\end{bmatrix}\times 2$, stride 1
`res2`	$H/8 \times W/8$	$\begin{bmatrix}3\times3,128\\3\times3,128\end{bmatrix}\times 2$, stride 2
`res3`	$H/8 \times W/8$	$\begin{bmatrix}3\times3,256\\3\times3,256\end{bmatrix}\times 2$, stride $\cancel{2}$ 1
`res4`	$H/8 \times W/8$	$\begin{bmatrix}3\times3,512\\3\times3,512\end{bmatrix}\times 2$, stride $\cancel{2}$ 1

Table 3.4: Modified ResNet-18 Architecture. Our modifications are shown in blue.

3.7.9 Test-time Training Details

We adopt the same hyper-parameters for optimization as in training: we use the Adam optimizer with learning rate 0.0001. Given an input video I, we fine-tune the model parameters by applying Algorithm 6 with input frames $\{I_{t-m}, ..., I_t, ..., I_{t+m}\}$, *prior* to propagating labels to I_t. For efficiency, we only finetune the model every 5 timesteps, applying Adam for 100 updates. In practice, we use $m = 10$, which we did not tune.

3.7.10 Utility Functions used in Algorithm 1

Algorithm 7 Utility functions.

```
// psize : size of patches to be extracted

import torch
import kornia.augmentation as K

# Turning images into list of patches
unfold = torch.nn.Unfold((psize, psize), stride=(psize//2, psize//2))

# l2 normalization
l2_norm = lambda x: torch.nn.functional.normalize(x, p=2, dim=1)

# Slightly cropping patches once extracted
spatial_jitter = K.RandomResizedCrop(size=(psize, psize), scale=(0.7, 0.9), ratio=(0.7, 1.3))
```

Chapter 4

Scalable Meta-RL with Self-supervised Rewards

In principle, meta-reinforcement learning algorithms leverage experience across many tasks to learn fast reinforcement learning (RL) strategies that transfer to similar tasks. However, current meta-RL approaches rely on manually-defined distributions of training tasks, and hand-crafting these task distributions can be challenging and time-consuming. Can "useful" pre-training tasks be discovered in an unsupervised manner? We develop an unsupervised algorithm for inducing an adaptive meta-training task distribution, i.e. an *automatic curriculum*, by modeling unsupervised interaction in a visual environment. The task distribution is scaffolded by a parametric density model of the meta-learner's trajectory distribution. We formulate unsupervised meta-RL as information maximization between a latent task variable and the meta-learner's data distribution, and describe a practical instantiation which alternates between integration of recent experience into the task distribution and meta-learning of the updated tasks. Repeating this procedure leads to iterative reorganization such that the curriculum adapts as the meta-learner's data distribution shifts. In particular, we show how discriminative clustering for visual representation can support trajectory-level task acquisition and exploration in domains with pixel observations, avoiding pitfalls of alternatives. In experiments on vision-based navigation and manipulation domains, we show that the algorithm allows for unsupervised meta-learning that transfers to downstream tasks specified by hand-crafted reward functions and serves as pre-training for more efficient supervised meta-learning of test task distributions.[*]

4.1 Introduction

The discrepancy between animals and learning machines in their capacity to gracefully adapt and generalize is a central issue in artificial intelligence research. The simple nematode *C.*

[*]This work was original published as *Unsupervised Curricula for Visual Meta-Reinforcement Learning*, Jabri et al, NeurIPS 2019[112].

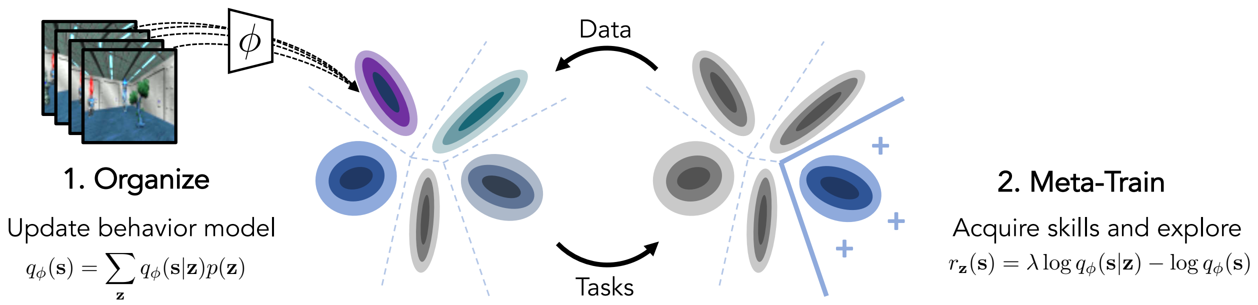

ϕ
Data
Tasks
1. Organize
Update behavior model
$q_\phi(\mathbf{s}) = \sum_{\mathbf{z}} q_\phi(\mathbf{s}|\mathbf{z})p(\mathbf{z})$
2. Meta-Train
Acquire skills and explore
$r_{\mathbf{z}}(\mathbf{s}) = \lambda \log q_\phi(\mathbf{s}|\mathbf{z}) - \log q_\phi(\mathbf{s})$

Figure 4.1: An illustration of CARML, our approach for unsupervised meta-RL. We choose the behavior model q_ϕ to be a Gaussian mixture model in a jointly, discriminatively learned embedding space. An automatic curriculum arises from periodically re-organizing past experience via fitting q_ϕ and meta-learning an RL algorithm for performance over tasks specified using reward functions from q_ϕ.

elegans is capable of adapting foraging strategies to varying scenarios [25], while many higher animals are driven to acquire reusable behaviors even without extrinsic task-specific rewards [252, 180]. It is unlikely that we can build machines as adaptive as even the simplest of animals by exhaustively specifying shaped rewards or demonstrations across all possible environments and tasks. This has inspired work in reward-free learning [91], intrinsic motivation [210], multi-task learning [30], meta-learning [201], and continual learning [226].

An important aspect of generalization is the ability to share and transfer ability between related tasks. In reinforcement learning (RL), a common strategy for multi-task learning is conditioning the policy on side-information related to the task. For instance, *contextual* policies [199] are conditioned on a task description (e.g. a *goal*) that is meant to modulate the strategy enacted by the policy. Meta-learning of reinforcement learning (meta-RL) is yet more general as it places the burden of inferring the task on the learner itself, such that task descriptions can take a wider range of forms, the most general being an MDP. In principle, meta-reinforcement learning (meta-RL) requires an agent to distill previous experience into fast and effective adaptation strategies for new, related tasks. However, the meta-RL framework by itself does not prescribe where this experience should come from; typically, meta-RL algorithms rely on being provided fixed, hand-specified task distributions, which can be tedious to specify for simple behaviors and intractable to design for complex ones [85]. These issues beg the question of whether "useful" task distributions for meta-RL can be generated automatically.

In this work, we seek a procedure through which an agent in an environment with visual observations can automatically acquire useful (i.e. utility maximizing) behaviors, as well as how and when to apply them – in effect allowing for *unsupervised* pre-training in visual environments. Two key aspects of this goal are: 1) learning to operationalize strategies so as to adapt to new tasks, i.e. meta-learning, and 2) unsupervised learning and exploration in the absence of explicitly specified tasks, i.e. skill acquisition *without* supervised reward functions. These aspects interact insofar as the former implicitly relies on a task curriculum, while the latter is most effective when compelled by what the learner can and cannot do. Prior work has offered a pipelined approach for unsupervised meta-RL consisting of unsupervised skill discovery followed by meta-learning of discovered skills, experimenting mainly in environments that expose low-dimensional ground truth state [83]. Yet, the aforementioned relation between skill acquisition and meta-learning suggests that they should not be treated separately.

Here, we argue for closing the loop between skill acquisition and meta-learning in order to induce an *adaptive* task distribution. Such co-adaptation introduces a number of challenges related to the stability of learning and exploration. Most recent unsupervised skill acquisition approaches optimize for the discriminability of induced modes of behavior (i.e. *skills*), typically expressing the discovery problem as a cooperative game between a policy and a learned reward function [79, 54, 1]. However, relying solely on discriminability becomes problematic in environments with high-dimensional (image-based) observation spaces as it results in an issue akin to mode-collapse in the task space. This problem is further complicated in the setting we propose to study, wherein the policy data distribution is that of a meta-learner rather than a contextual policy. We will see that this can be ameliorated by specifying a hybrid discriminative-generative model for parameterizing the task distribution.

The main contribution of this paper is an approach for inducing a task curriculum for unsupervised meta-RL in a manner that scales to domains with pixel observations. Through the lens of information maximization, we frame our unsupervised meta-RL approach as variational expectation-maximization (EM), in which the E-step corresponds to fitting a task distribution to a meta-learner's behavior and the M-step to meta-RL on the current task distribution with reinforcement for both skill acquisition and exploration. For the E-step, we show how deep discriminative clustering allows for trajectory-level representations suitable for learning diverse skills from pixel observations. Through experiments in vision-based navigation and robotic control domains, we demonstrate that the approach i) enables an unsupervised meta-learner to discover and meta-learn skills that transfer to downstream tasks specified by human-provided reward functions, and ii) can serve as pre-training for more efficient supervised meta-reinforcement learning of downstream task distributions.

4.2 Preliminaries: Meta-Reinforcement Learning

Supervised meta-RL optimizes an RL algorithm f_θ for performance on a hand-crafted distribution of tasks $p(\mathcal{T})$, where f_θ might take the form of an recurrent neural network (RNN) implementing a learning algorithm [49, 241], or a function implementing a gradient-based learning algorithm [58]. Tasks are Markov decision processes (MDPs) $\mathcal{T}_i = (\mathcal{S}, \mathcal{A}, r_i, P, \gamma, \rho, T)$ consisting of state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $r_i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, probabilistic transition dynamics $P(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$, discount factor γ, initial state distribution $\rho(\mathbf{s}_1)$, and finite horizon T. Often, and in our setting, tasks are assumed to share $\mathcal{S}, \mathcal{A}$. For a given $\mathcal{T} \sim p(\mathcal{T})$, f_θ learns a policy $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathcal{D}_\mathcal{T})$ conditioned on task-specific experience. Thus, a meta-RL

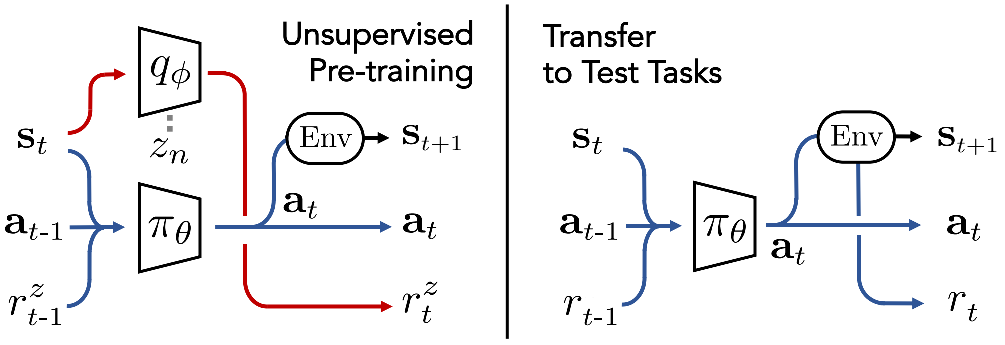


Figure 4.2: A step for the meta-learner. (Left) Unsupervised pre-training. The policy meta-learns self-generated tasks based on the behavior model q_ϕ. (Right) Transfer. Faced with new tasks, the policy transfers acquired meta-learning strategies to maximize unseen reward functions.

algorithm optimizes f_θ for expected performance of $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathcal{D}_\mathcal{T})$ over $p(\mathcal{T})$, such that it can generalize to unseen test tasks also sampled from $p(\mathcal{T})$.

For example, RL^2 [49, 241] chooses f_θ to be an sequence model with weights θ. For a given task $\mathcal{T}$, f_θ hones $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathcal{D}_\mathcal{T})$ as it recurrently ingests $\mathcal{D}_\mathcal{T} = (\mathbf{s}_1, \mathbf{a}_1, r(\mathbf{s}_1, \mathbf{a}_1), d_1, \dots)$, the sequence of states, actions, and rewards produced via interaction within the MDP. Crucially, the same task is seen several times, and the hidden state is not reset until the next task. The loss is the negative discounted return obtained by π_θ across episodes of the same task, and f_θ can be optimized via standard policy gradient methods for RL, backpropagating gradients through time and across episode boundaries.

Unsupervised meta-RL aims to break the reliance of the meta-learner on an explicit, upfront specification of $p(\mathcal{T})$. Following Gupta et al. [83], we consider a controlled Markov process (CMP) $\mathcal{C} = (\mathcal{S}, \mathcal{A}, P, \gamma, \rho, T)$, which is an MDP without a reward function. We are interested in the problem of learning an RL algorithm f_θ via unsupervised interaction within the CMP such that once a reward function r is specified at test-time, f_θ can be readily applied to the resulting MDP to efficiently maximize the expected discounted return.

Prior work [83] pipelines skill acquisition and meta-learning by pairing an unsupervised RL algorithm DIAYN [54] and a meta-learning algorithm MAML [58]: first, a contextual policy is used to discover skills in the CMP, yielding a finite set of learned reward functions distributed as $p(r)$; then, the CMP is combined with a frozen $p(r)$ to yield $p(\mathcal{T})$, which is fed to MAML to meta-learn f_θ. In the next section, we describe how we can generalize and improve upon this pipelined approach by jointly performing skill acquisition as the meta-learner learns and explores in the environment.

4.3 Curricula for Unsupervised Meta-Reinforcement Learning

Meta-learning is intended to prepare an agent to efficiently solve new tasks related to those seen previously. To this end, the meta-RL agent must balance 1) exploring the environment to infer which task it should solve, and 2) visiting states that maximize reward under the inferred task. The duty of unsupervised meta-RL is thus to present the meta-learner with tasks that allow it to practice task inference and execution, without the need for human-specified task distributions. Ideally, the task distribution should exhibit both structure and diversity. That is, the tasks should be distinguishable and not excessively challenging so that a developing meta-learner can infer and execute the right skill, but, for the sake of generalization, they should also encompass a diverse range of associated stimuli and rewards, including some beyond the current scope of the meta-learner. Our aim is to strike this balance by inducing an adaptive task distribution.

With this motivation, we develop an algorithm for unsupervised meta-reinforcement learning in visual environments that constructs a task distribution without supervision. The task distribution is derived from a latent-variable density model of the meta-learner's cumulative behavior, with exploration based on the density model driving the evolution of the task distribution. As depicted in Figure4.1, learning proceeds by alternating between two steps: **organizing experiential data** (i.e., trajectories generated by the meta-learner) by modeling it with a mixture of latent components forming the basis of "skills", and meta-reinforcement learning by **treating these skills as a training task distribution**.

Learning the task distribution in a data-driven manner ensures that tasks are feasible in the environment. While the induced task distribution is in no way guaranteed to align with test task distributions, it may yet require an implicit understanding of structure in the environment. This can indeed be seen from our visualizations in section 4.5, which demonstrate that acquired tasks show useful structure, though in some settings this structure is easier to meta-learn than others. In the following, we formalize our approach, CARML, through the lens of information maximization and describe a concrete instantiation that scales to the vision-based environments considered in section 4.5.

4.3.1 An Overview of CARML

We begin from the principle of information maximization (IM), which has been applied across unsupervised representation learning [12, 9, 168] and reinforcement learning [159, 79] for organization of data involving latent variables. In what follows, we organize data from our policy by maximizing the mutual information (MI) between state trajectories $\boldsymbol{\tau} := (\mathbf{s}_1, \ldots, \mathbf{s}_T)$ and a latent task variable $\mathbf{z}$. This objective provides a principled manner of trading-off structure and diversity: from $I(\boldsymbol{\tau}; \mathbf{z}) := H(\boldsymbol{\tau}) - H(\boldsymbol{\tau}|\mathbf{z})$, we see that $H(\boldsymbol{\tau})$ promotes coverage in policy data space (i.e. *diversity*) while $-H(\boldsymbol{\tau}|\mathbf{z})$ encourages a lack of diversity under each task (i.e. *structure* that eases task inference).

We approach maximizing $I(\boldsymbol{\tau};\mathbf{z})$ exhibited by the meta-learner f_θ via variational EM [9], introducing a variational distribution q_ϕ that can intuitively be viewed as a task scaffold for the meta-learner. In the E-step, we fit q_ϕ to a reservoir of trajectories produced by f_θ, re-organizing the cumulative experience. In turn, q_ϕ gives rise to a task distribution $p(\mathcal{T})$: each realization of the latent variable $\mathbf{z}$ induces a reward function $r_{\mathbf{z}}(\mathbf{s})$, which we combine with the CMP $\mathcal{C}_i$ to produce an MDP $\mathcal{T}_i$ (Line 8). In the M-step, f_θ meta-learns the task distribution $p(\mathcal{T})$. Repeating these steps forms a curriculum in which the task distribution and meta-learner co-adapt: each M-step adapts the meta-learner f_θ to the updated task distribution, while each E-step updates the task scaffold q_ϕ based on the data collected during meta-training. Pseudocode for our method is presented in Algorithm 8.

Algorithm 8 CARML – Curricula for Automatic Reinforcement of Meta-Learning

```
1: **Require:** $\mathcal{C}$, an MDP without a reward function
2: Initialize $f_\theta$, an RL algorithm parameterized by $\theta$.
3: Initialize $\mathcal{D}$, a reservoir of state trajectories, via a randomly initialized policy.
4: while not done do
5:     Fit a task-scaffold $q_\phi$ to $\mathcal{D}$, e.g. by using Algorithm 9.          E-step 4.3.2
6:     for a desired mixture model-fitting period do
7:         Sample a latent task variable $\mathbf{z} \sim q_\phi(\mathbf{z})$.
8:         Define the reward function $r_{\mathbf{z}}(\mathbf{s})$, e.g. by Eq. 4.8, and a task $\mathcal{T} = \mathcal{C} \cup r_{\mathbf{z}}(\mathbf{s})$.
9:         Apply $f_\theta$ on task $\mathcal{T}$ to obtain a policy $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathcal{D}_{\mathcal{T}})$ and trajectories $\{\boldsymbol{\tau}_i\}$.
10:        Update $f_\theta$ via a meta-RL algorithm, e.g. RL$^2$ [49].          M-step section 4.3.3
11:        Add the new trajectories to the reservoir: $\mathcal{D} \leftarrow \mathcal{D} \cup \{\boldsymbol{\tau}_i\}$.
12:    end for
13: end while
14: Return: a meta-learned RL algorithm $f_\theta$ tailored to $\mathcal{C}$
```

4.3.2 E-Step: Task Acquisition with Deep Clustering

The purpose of the E-step is to update the task distribution by integrating changes in the meta-learner's data distribution with previous experience, thereby allowing for re-organization of the task scaffold. This data is from the *post-update* policy, meaning that it comes from a policy $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathcal{D}_{\mathcal{T}})$ conditioned on data collected by the meta-learner for the respective task. In the following, we abuse notation by writing $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{z})$ – conditioning on the latent task variable $\mathbf{z}$ rather than the task experience $\mathcal{D}_{\mathcal{T}}$.

The general strategy followed by recent approaches for skill discovery based on IM is to lower bound the objective by introducing a variational posterior $q_\phi(\mathbf{z}|\mathbf{s})$ in the form of a classifier. In these approaches, the E-step amounts to updating the classifier to discriminate between data produced by different skills as much as possible. A potential failure mode of such an approach is an issue akin to mode-collapse in the task distribution, wherein the policy drops modes of behavior to favor easily discriminable trajectories, resulting in a lack of diversity in the task distribution and no incentive for exploration; this is especially problematic when considering high-dimensional observations. Instead, here we derive a generative variant, which allows us to account for explicitly capturing modes of behavior (by optimizing for likelihood), as well as a direct mechanism for exploration.

We introduce a variational distribution q_ϕ, which could be e.g. a (deep) mixture model with discrete $\mathbf{z}$ or a variational autoencoder (VAE) [125] with continuous $\mathbf{z}$, lower-bounding the objective:

$$I(\boldsymbol{\tau};\mathbf{z}) = -\sum_{\boldsymbol{\tau}} \pi_\theta(\boldsymbol{\tau}) \log \pi_\theta(\boldsymbol{\tau}) + \sum_{\boldsymbol{\tau},\mathbf{z}} \pi_\theta(\boldsymbol{\tau},\mathbf{z}) \log \pi_\theta(\boldsymbol{\tau}|\mathbf{z}) \tag{4.1}$$

$$\geq -\sum_{\boldsymbol{\tau}} \pi_\theta(\boldsymbol{\tau}) \log \pi_\theta(\boldsymbol{\tau}) + \sum_{\boldsymbol{\tau},\mathbf{z}} \pi_\theta(\boldsymbol{\tau}|\mathbf{z}) q_\phi(\mathbf{z}) \log q_\phi(\boldsymbol{\tau}|\mathbf{z}) \tag{4.2}$$

The E-step corresponds to optimizing Eq. 4.2 with respect to ϕ, and thus amounts to fitting q_ϕ to a reservoir of trajectories $\mathcal{D}$ produced by π_θ:

$$\max_\phi \; \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}),\boldsymbol{\tau}\sim\mathcal{D}}\big[\log q_\phi(\boldsymbol{\tau}|\mathbf{z})\big] \tag{4.3}$$

What remains is to determine the form of q_ϕ. We choose the variational distribution to be a state-level mixture density model $q_\phi(\mathbf{s},\mathbf{z}) = q_\phi(\mathbf{s}|\mathbf{z})q_\phi(\mathbf{z})$. Despite using a state-level generative model, we can treat $\mathbf{z}$ as a trajectory-level latent by computing the trajectory-level likelihood as the factorized product of state likelihoods (Algorithm 9, Line 4). This is useful for obtaining trajectory-level tasks; in the M-step (section 4.3.3), we map samples from $q_\phi(\mathbf{z})$ to reward functions to define tasks for meta-learning.

Algorithm 9 Task Acquisition via Discriminative Clustering

```
1: Require: a set of trajectories $\mathcal{D} = \{(\mathbf{s}_1,\dots,\mathbf{s}_T)\}_{i=1}^N$
2: Initialize $(\phi_w,\phi_m)$, encoder and mixture parameters.
3: while not converged do
4:     Compute $L(\phi_m;\boldsymbol{\tau},z) = \sum_{\mathbf{s}_t\in\boldsymbol{\tau}} \log q_{\phi_m}(g_{\phi_w}(\mathbf{s}_t)|z)$.
5:     $\phi_m \leftarrow \arg\max_{\phi'_m} \sum_{i=1}^N L(\phi'_m;\boldsymbol{\tau}_i,z)$ (via MLE)
6:     $\mathcal{D} := \{(\mathbf{s}, y := \arg\max_k q_{\phi_m}(z=k|g_{\phi_w}(\mathbf{s}))\}$.
7:     $\phi_w \leftarrow \arg\max_{\phi'_w} \sum_{(\mathbf{s},y)\in\mathcal{D}} \log q(y|g_{\phi'_w}(\mathbf{s}))$
8: end while
9: Return: a mixture model $q_\phi(\mathbf{s},z)$
```

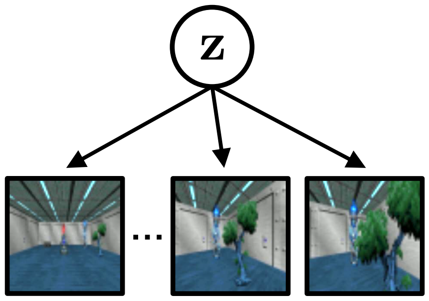


Figure 4.3: Conditional independence assumption for states along a trajectory.

Modeling Trajectories of Pixel Observations. While models like the variational autoencoder have been used in related settings [161], a basic issue is that optimizing for reconstruction treats all pixels equally. We, rather, will tolerate *lossy* representations as long as they capture *discriminative* features useful for stimulus-reward association. Drawing inspiration from recent work on unsupervised feature learning by clustering [17, 27], we propose to fit the trajectory-level mixture model via discriminative clustering, striking a balance between discriminative and generative approaches.

We adopt the optimization scheme of DeepCluster [27], which alternates between i) clustering representations to obtain pseudo-labels and ii) updating the representation by supervised learning of pseudo-labels. In particular, we derive a trajectory-level variant (Algorithm 9) by forcing the responsibilities of all observations in a trajectory to be the same (see Appendix 4.7.1 for a derivation), leading to state-level visual representations optimized with trajectory-level supervision.

The conditional independence assumption in Algorithm 9 is a simplification insofar as it discards the order of states in a trajectory. However, if the dynamics exhibit continuity and causality, the visual representation might yet capture temporal structure, since, for example, attaining certain observations might imply certain antecedent subtrajectories. We hypothesize that a state-level model can regulate issues of over-expressive sequence encoders, which have been found to lead to skills with undesirable attention to details in dynamics [1]. As we will see in section 4.5, learning representations under this assumption still allows for learning visual features that capture trajectory-level structure.

4.3.3 M-Step: Meta-Learning with RL^2

Using the task scaffold updated via the E-step, we meta-learn f_θ in the M-step so that π_θ can be quickly adapted to tasks drawn from the task scaffold. To define the task distribution, we must specify a form for the reward functions $r_{\mathbf{z}}(\mathbf{s})$. To allow for state-conditioned Markovian rewards rather than non-Markovian trajectory-level rewards, we lower-bound the trajectory-level MI objective:

$$I(\boldsymbol{\tau}; \mathbf{z}) = \frac{1}{T}\sum_{t=1}^{T} H(\mathbf{z}) - H(\mathbf{z}|\mathbf{s}_1, ..., \mathbf{s}_T) \geq \frac{1}{T}\sum_{t=1}^{T} H(\mathbf{z}) - H(\mathbf{z}|\mathbf{s}_t) \tag{4.4}$$

$$\geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}), \mathbf{s}\sim\pi_\theta(\mathbf{s}|\mathbf{z})}\big[\log q_\phi(\mathbf{s}|\mathbf{z}) - \log \pi_\theta(\mathbf{s})\big] \tag{4.5}$$

We would like to optimize the meta-learner under the variational objective in Eq. 4.5, but optimizing the second term, the policy's state entropy, is in general intractable. Thus, we make the simplifying assumption that the fitted variational marginal distribution matches that of the policy:

$$\max_\theta \; \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}), \mathbf{s}\sim\pi_\theta(\mathbf{s}|\mathbf{z})}\big[\log q_\phi(\mathbf{s}|\mathbf{z}) - \log q_\phi(\mathbf{s})\big] \tag{4.6}$$

$$= \max_\theta \; I(\pi_\theta(\mathbf{s}); q_\phi(\mathbf{z})) - D_{\mathrm{KL}}\pi_\theta(\mathbf{s}|\mathbf{z})q_\phi(\mathbf{s}|\mathbf{z}) + D_{\mathrm{KL}}\pi_\theta(\mathbf{s})q_\phi(\mathbf{s})) \tag{4.7}$$

Optimizing Eq. 4.6 amounts to maximizing the reward of $r_{\mathbf{z}}(\mathbf{s}) = \log q_\phi(\mathbf{s}|\mathbf{z}) - \log q_\phi(\mathbf{s})$. As shown in Eq. 4.7, this corresponds to information maximization between the policy's state marginal and the latent task variable, along with terms for matching the task-specific policy data distribution to the corresponding mixture mode and deviating from the mixture's marginal density. We can trade-off between component-matching and exploration by introducing a weighting term $\lambda \in [0, 1]$ into $r_{\mathbf{z}}(\mathbf{s})$:

$$\begin{aligned} r_{\mathbf{z}}(\mathbf{s}) &= \lambda \log q_\phi(\mathbf{s}|\mathbf{z}) - \log q_\phi(\mathbf{s}) && (4.8) \\ &= (\lambda - 1) \log q_\phi(\mathbf{s}|\mathbf{z}) + \log q_\phi(\mathbf{z}|\mathbf{s}) + C && (4.9) \end{aligned}$$

where C is a constant with respect to the optimization of θ. From Eq. 4.9, we can interpret λ as trading off between discriminability of skills and task-specific exploration. Figure 4.4 shows the effect of tuning λ on the structure-diversity trade-off alluded to at the beginning of section 4.3.

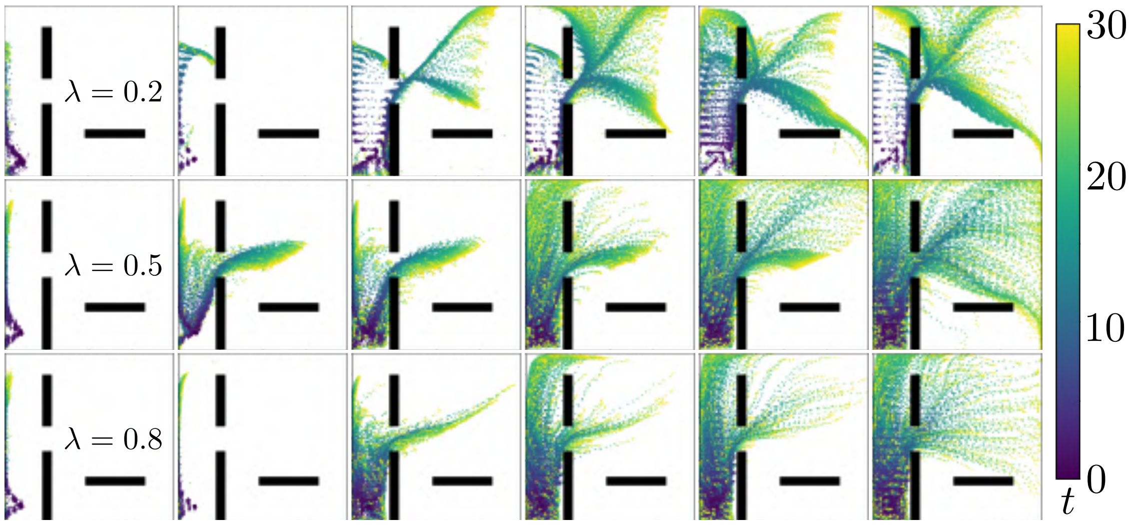


Figure 4.4: Balancing consistency and exploration with λ in a simple 2D maze environment. Each row shows a progression of tasks developed over the course of training. Each box presents the mean reconstructions under a VAE q_ϕ (Appendix 4.7.3) of 2048 trajectories. Varying λ of Eq. 4.8 across rows, we observe that a small λ (top) results in aggressive exploration; a large λ (bottom) yields relatively conservative behavior; and a moderate λ (middle) produces sufficient exploration and a smooth task distribution.

4.4 Related Work

Unsupervised Reinforcement Learning. Unsupervised learning in the context of RL is the problem of enabling an agent to learn about its environment and acquire useful behaviors without human-specified reward functions. A large body of prior work has studied exploration and intrinsic motivation objectives [200, 195, 173, 65, 22, 13, 138, 170]. These algorithms do not aim to acquire skills that can be operationalized to solve tasks, but rather try to achieve wide coverage of the state space; our objective (Eq. 4.8) reduces to pure density-based exploration with $\lambda = 0$. Hence, these algorithms still rely on slow RL [18] in order to adapt to new tasks posed at test-time. Some prior works consider unsupervised pre-training for efficient RL, but these works typically focus on settings in which exploration is not as much of a challenge [248, 59, 51], focus on goal-conditioned policies [175, 161], or have not been shown to scale to high-dimensional visual observation spaces [144, 209]. Perhaps most relevant to our work are unsupervised RL algorithms for learning reward functions via optimizing information-theoretic objectives involving latent skill variables [79, 1, 54, 247]. In particular, with a choice of $\lambda = 1$ in Eq. 4.9 we recover the information maximization objective used in prior work [1, 54], besides the fact that we simulatenously perform meta-learning. The setting of training a contextual policy with a classifier as q_ϕ in our proposed framework (see Appendix 4.7.1) provides an interpretation of DIAYN as implicitly doing trajectory-level clustering. Warde-Farley et al. [247] also considers accumulation of tasks, but with a focus on goal-reaching and by maintaining a goal reservoir via heuristics that promote diversity.

Meta-Learning. Our work is distinct from above works in that it formulates a meta-learning approach to explicitly train, without supervision, for the ability to adapt to new downstream RL tasks. Prior work [108, 122, 3] has investigated this unsupervised meta-learning setting for image classification; the setting considered herein is complicated by the added challenges of RL-based policy optimization and exploration. Gupta et al. [83] provides an initial exploration of the unsupervised meta-RL problem, proposing a straightforward combination of unsupervised skill acquisition (via DIAYN) followed by MAML [58] with experiments restricted to environments with fully observed, lower-dimensional state. Unlike these works and other meta-RL works [241, 49, 157, 186, 58, 107, 82, 192, 218, 223], we close the loop to jointly perform task acquisition and meta-learning so as to achieve an automatic curriculum to facilitate joint meta-learning and task-level exploration.

Automatic Curricula. The idea of automatic curricula has been widely explored both in supervised learning and RL. In supervised learning, interest in automatic curricula is based on the hypothesis that exposure to data in a specific order (i.e. a non-uniform curriculum) may allow for learning harder tasks more efficiently [52, 200, 75]. In RL, an additional challenge is exploration; hence, related work in RL considers the problem of *curriculum generation*, whereby the task distribution is designed to guide exploration towards solving complex tasks [61, 154, 60, 202] or unsupervised pre-training [221, 62]. Our work is driven by similar motivations, though we consider a curriculum in the setting of meta-RL and frame our approach as information maximization.

4.5 Experiments

We experiment in visual navigation and visuomotor control domains to study the following questions:

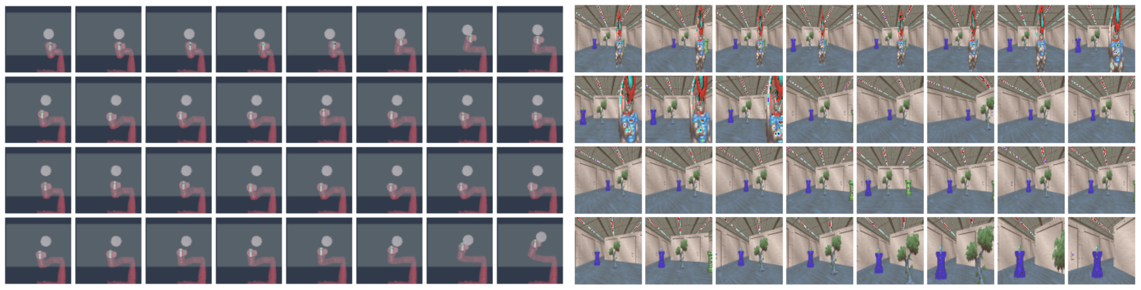


Figure 4.5: Example Observation Sequences from the Sawyer and Vizdoom environments.

- What kind of tasks are discovered through our task acquisition process (the E-step)?
- Do these tasks allow for meta-training of strategies that transfer to test tasks?
- Does closing the loop to jointly perform task acquisition and meta-learning bring benefits?
- Does pre-training with CARML accelerate meta-learning of test task distributions?

Videos are available at the project website `https://sites.google.com/view/carml`.

4.5.1 Experimental Setting

The following experimental details are common to the two vision-based environments we consider. Other experimental are explained in more detail in Appendix 4.7.2.

Meta-RL. CARML is agnostic to the meta-RL algorithm used in the M-step. We use the RL^2 algorithm [49], which has previously been evaluated on simpler visual meta-RL domains, with a PPO [203] optimizer. Unless otherwise stated, we use four episodes per trial (compared to the two episodes per trial used in [49]), since the settings we consider involve more challenging task inference.

Baselines. We compare against: 1) PPO from scratch on each evaluation task, 2) pre-training with random network distillation (RND) [22] for unsupervised exploration, followed by fine-tuning on evaluation tasks, and 3) supervised meta-learning on the test-time task distribution, as an oracle.

Variants. We consider variants of our method to ablate the role of design decisions related to task acquisition and joint training: 4) *pipelined* (most similar to [83]) – task acquisition with a contextual policy, followed by meta-RL with RL^2; 5) *online discriminator* – task acquisition with a purely discriminative q_ϕ (akin to online DIAYN); and 6) *online pretrained-discriminator* – task acquisition with a discriminative q_ϕ initialized with visual features trained via Algorithm 9.

4.5.2 Visual Navigation

The first domain we consider is first-person visual navigation in ViZDoom [121], involving a room filled with five different objects (drawn from a set of 50). We consider a setup akin to those featured in [32, 255] (see Figure 4.3). The true state consists of continuous 2D position and continuous orientation, while observations are egocentric images with limited field of view. Three discrete actions allow for turning right or left, and moving forward. We consider two ways of sampling the CMP $\mathcal{C}$. **Fixed**: fix a set of five objects and positions for both unsupervised meta-training and testing. **Random**: sample five objects and randomly place them (thereby randomizing the state space and dynamics).

Visualizing the task distribution. Modeling pixel observations reveals trajectory-level organization in the underlying true state space (Figure 4.6). Each map portrays trajectories of a mixture component, with position encoded in 2D space and orientation encoded in the jet color-space; an example of interpreting the maps is shown left of the legend. The components of the mixture model reveal structured groups of trajectories: some components correspond to exploration of the space (marked with green border), while others are more strongly directed towards specific areas (blue border). The skill maps of the fixed and random environments are qualitatively different: tasks in the fixed room tend towards interactions with objects or walls, while many of the tasks in the random setting sweep the space in a particular direction. We can also see the evolution of the task distribution at earlier and later stages of Algorithm 8. While initial tasks (produced by a randomly initialized policy) tend to be less structured, we later see refinement of certain tasks as well as the emergence

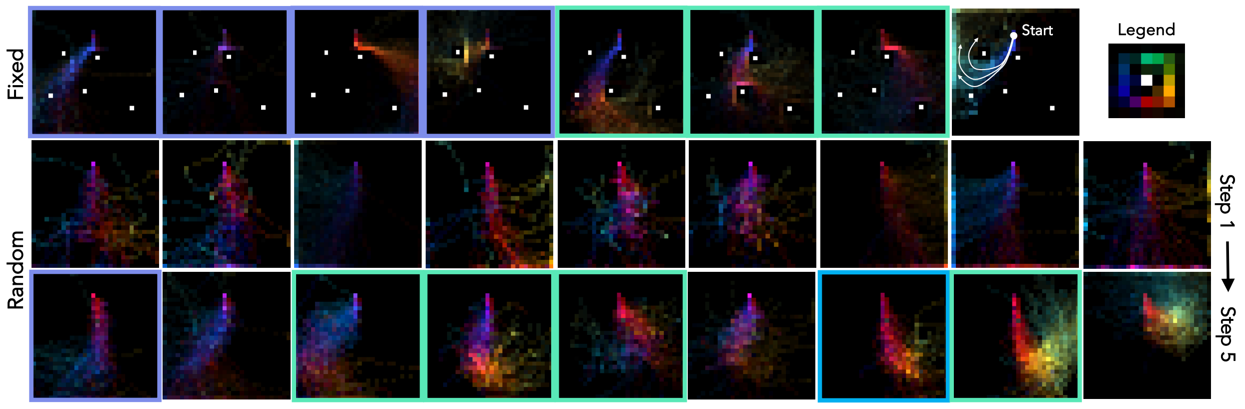
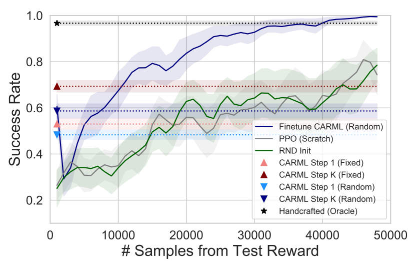
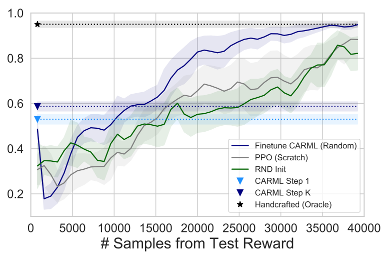


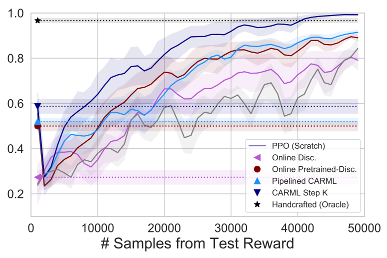
Figure 4.6: Skill maps for visual navigation. We visualize some of the discovered tasks by projecting trajectories of certain mixture components into the true state space. White dots correspond to fixed objects. The legend indicates orientation as color; on its left is an interpretation of the depicted component. Some tasks seem to correspond to exploration of the space (green border), while others are more directed towards specific areas (blue border). Comparing tasks earlier and later in the curriculum (step 1 to step 5), we find an increase in structure.



(a) ViZDoom



(b) Sawyer



(c) Variants (ViZDoom Random)

Figure 4.7: CARML enables unsupervised meta-reinforcemnt learning that transfer to downstream tasks. Direct transfer curves (marker and dotted line) represent a meta-learner deploying for just 200 time steps at test time. Compared to CARML, PPO and RND Init sample the test reward function orders of magnitude more times to perform similarly on a single task. Finetuning the CARML policy also allows for solving individual tasks with significantly fewer samples. The ablation experiments (c) assess both direct transfer and finetuning for each variant. Compared to variants, the CARML task acquisition procedure improves transfer by mitigating task mode-collapse and adapting the task distribution.

of others as the agent collects new data and acquires strategies for performing existing tasks.

Do acquired skills transfer to test tasks? We evaluate how well the CARML task distribution prepares the agent for unseen tasks. For both the fixed and randomized CMP experiments, each test task specifies a dense goal-distance reward for reaching a single object in the environment. In the randomized environment setting, the target objects at test-time are held out from meta-training. The PPO and RND-initialized baseline polices, and the finetuned CARML meta-policy, are trained for a single target (a specific object in a fixed environment), with 100 episodes per PPO policy update.

In Figure 4.7a, we compare the success rates on test tasks as a function of the number of samples with supervised rewards seen from the environment. Direct transfer performance of meta-learners is shown as points, since in this setting the RL^2 agent sees only *four episodes* (200 samples) at test-time, without any parameter updates. We see that direct transfer is significant, achieving up to 71% and 59% success rates on the fixed and randomized settings, respectively. The baselines require over two orders of magnitude more test-time samples to solve a single task at the same level.

While the CARML meta-policy does not consistently solve the test tasks, this is not surprising since no information is assumed about target reward functions during unsupervised meta-learning; inevitable discrepancies between the meta-train and test task distributions will mean that meta-learned strategies *will* be suboptimal for the test tasks. For instance, during testing, the agent sometimes 'stalls' before the target object (once inferred), in order to exploit the inverse distance reward. Nevertheless, we also see that finetuning the CARML meta-policy *trained on random* environments on individual tasks is more sample efficient

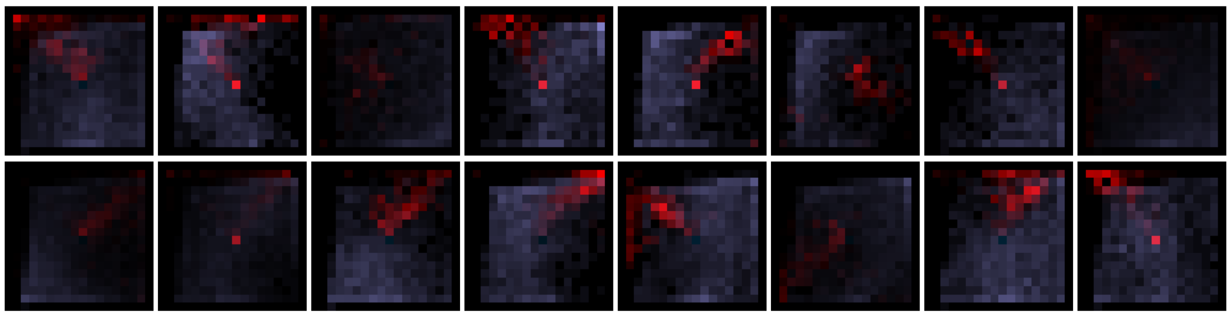


Figure 4.8: (Left) Skill maps for visuomotor control. Red encodes the true position of the object, and light blue that of the end-effector. Tasks correspond to moving the object to various regions (see Appendix 4.7.4 for more skills maps and analysis). (Right) Observation and third person view from the environment, respectively.

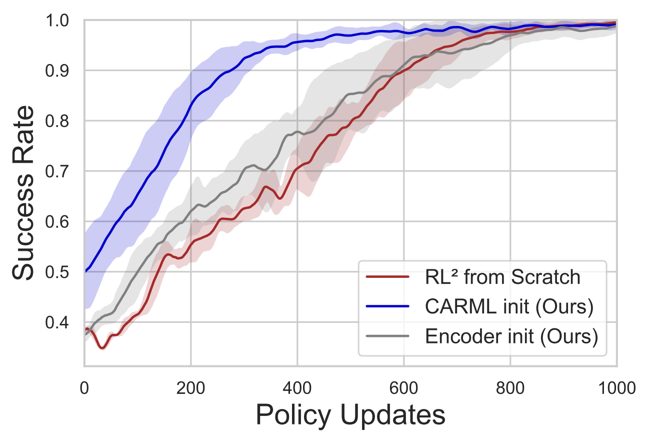
than learning from scratch. This suggests that deriving reward functions from our mixture model yields useful tasks insofar as they facilitate learning of strategies that transfer.

Benefit of reorganization. In Figure 4.7a, we also compare performance across early and late outer-loop iterations of Algorithm 8, to study the effect of adapting the task distribution (the CARML E-step) by reorganizing tasks and incorporating new data. In both cases, number of outer-loop iterations $K = 5$. Overall, the refinement of the task distribution, which we saw in Figure 4.6, leads improved to transfer performance. The effect of reorganization is further visualized in the Appendix 4.7.6.

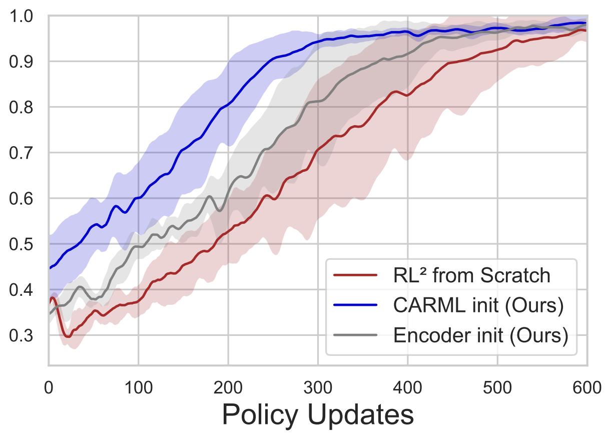
Variants. From Figure 4.7c, we see that the purely online discriminator variant suffers in direct transfer performance; this is due to the issue of mode-collapse in task distribution, wherein the task distribution lacks diversity. Pretraining the discriminator encoder with Algorithm 9 mitigates mode-collapse to an extent, improving task diversity as the features and task decision boundaries are first fit on a corpus of (randomly collected) trajectories. Finally, while the distribution of tasks eventually discovered by the pipelined variant may be diverse and structured, meta-learning the corresponding tasks from scratch is harder. More detailed analysis and visualization is given in Appendix 4.7.5.

4.5.3 Visual Robotic Manipulation

To experiment in a domain with different challenges, we consider a simulated Sawyer arm interacting with an object in MuJoCo [228], with end-effector continous control in the 2D plane. The observation is a bottom-up view of a surface supporting an object (Figure 4.8); the camera is stationary, but the view is no longer egocentric and part of the observation is proprioceptive. The test tasks involve pushing the object to a goal (drawn from the set of reachable states), where the reward function is the negative distance to the goal state. A subset of the skill maps is provided below.



(a) ViZDoom (random)



(b) Sawyer

Figure 4.9: Finetuning the CARML meta-policy allows for accelerated meta-reinforcement learning of the target task distribution. Curves reflect error bars across three random seeds.

Do acquired skills directly transfer to test tasks? In Figure 4.7b, we evaluate the meta-policy on the test task distribution, comparing against baselines as previously. Despite the increased difficulty of control, our approach allows for meta-learning skills that transfer to the goal distance reward task distribution. We find that transfer is weaker compared to the visual navigation (fixed version): one reason may be that the environment is not as visually rich, resulting in a significant gap between the CARML and the object-centric test task distributions.

4.5.4 CARML as Meta-Pretraining

Another compelling form of transfer is pretraining of an initialization for accelerated supervised meta-RL of target task distributions. In Figure 4.9, we see that the initialization learned by CARML enables effective supervised meta-RL with significantly fewer samples. To separate the effect of the learning the recurrent meta-policy and the visual representation, we also compare to only initializing the pre-trained encoder. Thus, while direct transfer of the meta-policy may not directly result in optimal behavior on test tasks, accelerated learning of the test task distribution suggests that the acquired meta-learning strategies may be useful for learning related task distributions, effectively acting as pre-training procedure for meta-RL.

4.6 Discussion

We proposed a framework for inducing unsupervised, adaptive task distributions for meta-RL that scales to environments with high-dimensional pixel observations. Through experiments in visual navigation and manipulation domains, we showed that this procedure enables unsupervised acquisition of meta-learning strategies that transfer to downstream test task distributions in terms of direct evaluation, more sample-efficient fine-tuning, and more sample-efficient supervised meta-learning. Nevertheless, the following key issues are important to explore in future work.

Task distribution mismatch. While our results show that useful structure can be meta-learned in an unsupervised manner, results like the stalling behavior in ViZDoom (see section 4.5.2) suggest that direct transfer of unsupervised meta-learning strategies suffers from a no-free-lunch issue: there will always be a gap between unsupervised and downstream task distributions, and more so with more complex environments. Moreover, the semantics of target tasks may not necessarily align with especially discriminative visual features. This is part of the reason why transfer in the Sawyer domain is less successful. Capturing other forms of structure useful for stimulus-reward association might involve incorporating domain-specific inductive biases into the task-scaffold model. Another way forward is the semi-supervised setting, whereby data-driven bias is incorporated at meta-training time.

Validation and early stopping: Since the objective optimized by the proposed method is non-stationary and in no way guaranteed to be correlated with objectives of test tasks, one must provide some mechanism for validation of iterates.

Form of skill-set. For the main experiments, we fixed a number of discrete tasks to be learned (without tuning this), but one should consider how the set of skills can be grown or parameterized to have higher capacity (e.g. a multi-label or continuous latent). Otherwise, the task distribution may become overloaded (complicating task inference) or limited in capacity (preventing coverage).

Accumulation of skill. We mitigate forgetting with the simple solution of reservoir sampling. Better solutions involve studying an intersection of continual learning and meta-learning.

4.7 Appendix

4.7.1 Derivations

Derivation for Trajectory-Level Responsibilities (Section 3.2.1)

Here we show that, assuming independence between states in a trajectory when conditioning on a latent variable, computing the trajectory likelihood as a factorized product of state likelihoods for the E-step in standard EM forces the component responsibilities for all states in the trajectory to be identical. Begin by lower-bounding the log-likelihood of the trajectory dataset with Jensen's inequality:

$$\begin{aligned}
\sum_i \log p(\boldsymbol{\tau}) &= \sum_i \log p(\mathbf{s}_1^i, \mathbf{s}_2^i, ..., \mathbf{s}_T^i) && (4.10)\\
&= \sum_i \log \sum_z p(\mathbf{s}_1^i, \mathbf{s}_2^i, ..., \mathbf{s}_T^i | z) p(z) && (4.11)\\
&\geq \sum_i \sum_z q_\phi(z|\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T) \log \frac{p(\mathbf{s}_1^i, \mathbf{s}_2^i, ..., \mathbf{s}_T^i | z) p(z)}{q_\phi(z|\mathbf{s}_1, \mathbf{s}_2 ... \mathbf{s}_T)} && (4.12)\\
&= \sum_i \mathbb{E}_{z \sim q_\phi(z|\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T)} \log \frac{p(\mathbf{s}_1^i, \mathbf{s}_2^i, ..., \mathbf{s}_T^i | z) p(z)}{q_\phi(z|\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T)}. && (4.13)
\end{aligned}$$

We have introduced the variational distribution $q_\phi(\boldsymbol{\tau}, z)$, where z is a categorical variable. Now, to maximize Eq. 4.13 with respect to $\phi := (\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \pi_1, ..., \boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N, \pi_N)$, we alternate between an E-step and an M-step, where the E-step is computing

$$\begin{aligned}
q_{ik} &= q_\phi(z = k|\mathbf{s}_1^i, \mathbf{s}_2^i, ..., \mathbf{s}_T^i) && (4.14)\\
&= \frac{q_\phi(\mathbf{s}_1^i, \mathbf{s}_2^i, ..., \mathbf{s}_T^i | z = k) q_\phi(z = k)}{\sum_j q_\phi(\mathbf{s}_1^i, \mathbf{s}_2^i, ..., \mathbf{s}_T^i | z = j) q_\phi(z = j)} && (4.15)\\
&= \frac{q_\phi(\mathbf{s}_1^i | z = k) q_\phi(\mathbf{s}_2^i | z = k) \cdots q_\phi(\mathbf{s}_T^i | z = k) q_\phi(z = k)}{\sum_j q_\phi(\mathbf{s}_1^i | z = j) q_\phi(\mathbf{s}_2^i | z = j) \cdots q_\phi(\mathbf{s}_T^i | z = j) q_\phi(z = j)}. && (4.16)
\end{aligned}$$

We assume that each $q_\phi(\mathbf{s}|z = k)$ is Gaussian; the M-step amounts to computing the maximum-likelihood estimate of ϕ, under the mixture responsibilities from the E-step:

$$\begin{aligned}
\boldsymbol{\mu}_k &= \frac{\sum_{i=1}^N \frac{q_{ik}}{T} \sum_{t=1}^T \mathbf{s}_t}{\sum_{i=1}^N q_{ik}} && (4.17)\\
\boldsymbol{\Sigma}_k &= \frac{\sum_{i=1}^N \frac{q_{ik}}{T} \sum_{t=1}^T (\mathbf{s}_t - \boldsymbol{\mu}_k)(\mathbf{s}_t - \boldsymbol{\mu}_k)^\top}{\sum_{i=1}^N q_{ik}} && (4.18)\\
\pi_k &= \frac{1}{N} \sum_{i=1}^N q_{ik}. && (4.19)
\end{aligned}$$

In particular, note that the expressions are independent of t. Thus, the posterior $q_\phi(z|\mathbf{s})$ will be, too.

CARML M-Step

The objective used to optimize the meta-RL algorithm in the CARML M-step can be interpreted as a sum of cross entropies, resulting in the mutual information plus two additional KL terms:

$$\begin{aligned}
&-\mathbb{E}_{\mathbf{s}\sim\pi_\theta(\mathbf{s}|\mathbf{z}),\mathbf{z}\sim q_\phi(\mathbf{z})}\left[\log q_\phi(\mathbf{s}) - \log q_\phi(\mathbf{s}|\mathbf{z})\right] && (4.20)\\
=&-\sum_{\mathbf{z}} q_\phi(\mathbf{z}) \sum_{\mathbf{s}} \pi_\theta(\mathbf{s}|\mathbf{z}) \left(\log q_\phi(\mathbf{s}) - \log q_\phi(\mathbf{s}|\mathbf{z})\right) && (4.21)\\
=&-\sum_{\mathbf{z}} q_\phi(\mathbf{z}) \sum_{\mathbf{s}} \pi(\mathbf{s}|\mathbf{z}) \left(\log \frac{q_\phi(\mathbf{s})}{\pi_\theta(\mathbf{s})} + \log \pi_\theta(\mathbf{s}) - \log \frac{q_\phi(\mathbf{s}|\mathbf{z})}{\pi_\theta(\mathbf{s}|\mathbf{z})} - \log \pi_\theta(\mathbf{s}|\mathbf{z})\right) && (4.22)\\
=&H(\pi_\theta(\mathbf{s})) + D_{\mathrm{KL}}\pi_\theta(\mathbf{s})q_\phi(\mathbf{s}) - H(\pi_\theta(\mathbf{s}|\mathbf{z})) - D_{\mathrm{KL}}\pi_\theta(\mathbf{s}|\mathbf{z})q_\phi(\mathbf{s}|\mathbf{z}) && (4.23)\\
=&I(\pi_\theta(\mathbf{s}); q_\phi(\mathbf{z})) + D_{\mathrm{KL}}\pi_\theta(\mathbf{s})q_\phi(\mathbf{s}) - D_{\mathrm{KL}}\pi_\theta(\mathbf{s}|\mathbf{z})q_\phi(\mathbf{s}|\mathbf{z}). && (4.24)
\end{aligned}$$

The first KL term can be interpreted as encouraging exploration with respect to the density of the mixture. The second KL term is the reverse KL term for matching the modes of the mixture.

Density-based exploration. In practice, we may want to trade off between exploration and matching the modes of the generative model:

$$\begin{aligned}
r_{\mathbf{z}}(\mathbf{s}) &= \lambda \log q_\phi(\mathbf{s}|\mathbf{z}) - \log q_\phi(\mathbf{s}) && (4.25)\\
&= (\lambda - 1)\log q_\phi(\mathbf{s}|\mathbf{z}) + \log q_\phi(\mathbf{z}|\mathbf{s}) - \log q_\phi(\mathbf{z}) && (4.26)\\
&= (\lambda - 1)\log q_\phi(\mathbf{s}|\mathbf{z}) + \log q_\phi(\mathbf{z}|\mathbf{s}) + C && (4.27)
\end{aligned}$$

where C is constant with respect to the optimization of θ. Hence, the objective amounts to maximizing discriminability of skills where $\lambda < 1$ yields a bonus for exploring away from the mode of the corresponding skill.

Discriminative CARML and DIAYN

Here, we derive a discriminative instantiation of CARML. We begin with the E-step. We leverage the same conditional independence assumption as before, and re-write the trajectory-level MI as the state level MI, assuming that trajectories are all of length T:

$$I(\boldsymbol{\tau}; \mathbf{z}) \geq \frac{1}{T}\sum_t I(\mathbf{s}_t; \mathbf{z}) = I(\mathbf{s}; \mathbf{z}) \qquad (4.28)$$

We then decompose MI as the difference between marginal and conditional entropy of the latent, and choose the variational distribution to be the product of a classifier $q_{\phi_c}(\mathbf{z}|\mathbf{s})$ and a density model $q_{\phi_d}(\mathbf{s})$:

$$\begin{aligned} I(\mathbf{s};\mathbf{z}) &= H(\mathbf{z}) - H(\mathbf{z}|\mathbf{s}) && (4.29)\\ &= -\sum_{\mathbf{z}} p(\mathbf{z}) \log p(\mathbf{z}) + \sum_{\mathbf{s},\mathbf{z}} \pi_\theta(\mathbf{s},\mathbf{z}) \log \pi_\theta(\mathbf{z}|\mathbf{s}) && (4.30)\\ &\geq -\sum_{\mathbf{z}} p(\mathbf{z}) \log p(\mathbf{z}) + \sum_{\mathbf{s},\mathbf{z}} \pi_\theta(\mathbf{s}|\mathbf{z}) p(\mathbf{z}) \log q_{\phi_c}(\mathbf{z}|\mathbf{s}) && (4.31) \end{aligned}$$

We fix $\mathbf{z}$ to be a uniformly-distributed categorical variable. The CARML E-step consists of two separate optimizations: supervised learning of $q_{\phi_c}(\mathbf{z}|\mathbf{s})$ with a cross-entropy loss and density estimation of $q_{\phi_d}(\mathbf{s})$:

$$\max_{\phi_c} \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}),\mathbf{s}\sim\pi_\theta(\mathbf{z})} \left[\log q_{\phi_c}(\mathbf{z}|\mathbf{s})\right] \qquad \max_{\phi_d} \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}),\mathbf{s}\sim\pi_\theta(\mathbf{z})} \left[\log q_{\phi_d}(\mathbf{s})\right] \tag{4.32}$$

For the CARML M-step, we start from the form of the reward in Eq. 4.26 and manipulate via Bayes':

$$\begin{aligned} r_{\mathbf{z}}(\mathbf{s}) &= \log q_{\phi_c}(\mathbf{z}|\mathbf{s}) + (\lambda-1)\log q_{\phi_c}(\mathbf{z}|\mathbf{s}) + (\lambda-1)\log q_{\phi_d}(\mathbf{s}) - (\lambda-1)\log p(\mathbf{z}) - \log p(\mathbf{z})\\ &= \lambda \log q_{\phi_c}(\mathbf{z}|\mathbf{s}) + (\lambda-1)\log q_{\phi_d}(\mathbf{s}) + C && (4.33) \end{aligned}$$

where C is constant with respect to the optimization of θ in the M-step

$$\max_{\theta} \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}),\mathbf{s}\sim\pi_\theta(\mathbf{z})} \left[\lambda \log q_{\phi_c}(\mathbf{z}|\mathbf{s}) + (\lambda-1)\log q_{\phi_d}(\mathbf{s})\right] \tag{4.34}$$

To enable a trajectory-level latent $\mathbf{z}$, we want every state in a trajectory to be classified to the same $\mathbf{z}$. This is achievable in a straightforward manner: when training the classifier $q_{\phi_c}(\mathbf{z}|\mathbf{s})$ via supervised learning, label each state in a trajectory with the realization of $\mathbf{z}$ that the policy $\pi_\theta(\mathbf{a}|\mathbf{s},\mathbf{z})$ was conditioned on when generating that trajectory.

Connection to DIAYN. Note that with $\lambda = 1$ in Eq. 4.34, we directly obtain the DIAYN [54] objective without standard policy entropy regularization, and we do away with needing to maintain a density model $\log q_{\phi_d}(\mathbf{s})$, leaving just the discriminator. If $\pi_\theta(\mathbf{a}|\mathbf{s},\mathbf{z})$ is truly a contextual policy (rather than the policy given by adapting a meta-learner), we have recovered the DIAYN algorithm. This allows us to interpret on DIAYN-style algorithms as implicitly doing trajectory-level clustering with a conditional independence assumption between states in a trajectory given the latent. This arises from the weak trajectory-level supervision specified when training the discriminator: all states in a trajectory are assumed to correspond to the same realization of the latent variable.

4.7.2 Additional Details for Main Experiments

CARML Hyperparameters

We train CARML for five iterations, with 500 PPO updates for meta-learning with RL^2 in the M-step (i.e. update the mixture model every 500 meta-policy updates). Thus, the CARML unsupervised learning process consumes on the order of 1,000,000 episodes (compared to the ~400,000 episodes needed to train a meta-policy with the true task distribution, as shown in our experiments). We did not heavily tune this number, though we noticed that using too few policy updates (e.g. ~100) before refitting q_ϕ resulted in instability insofar as the meta-learner does not adapt to learn the updated task distribution. Each PPO learning update involves sampling 100 tasks with 4 episodes each, for a total of 400 episodes per update. We use 10 PPO epochs per update with a batch size of 100 tasks.

During meta-training, tasks are drawn according to $z \sim q_\phi(z)$, the mixture's latent prior distribution. Unless otherwise stated, we use $\lambda = 0.99$ for all visual meta-RL experiments. For all experiments unless otherwise mentioned, we fix the number of components in our mixture to be $k = 16$. We use a reservoir of 1000 trajectories.

Temporally Smoothed Reward: At unsupervised meta-training time, we found it helpful to reward the meta-learner with the average over a small temporal window, i.e. $r_z^W(s_t) = \frac{1}{W}\sum_{i=t-W}^{t} r_z(s_i)$, choosing W to be $W = 10$. This has the effect of smoothing the reward function, thereby regularizing acquired task inference strategies.

Random Seeds: The results reported in Figure 6 are averaged across policies (for each treatment) trained with three different random seeds. The performance is averaged across 20 test tasks. The results reported in Figure 7 are based on finetuning CARML policies trained with three different random seeds. We did not observe significant effects of the random seed used in the finetuning procedure of experiments reported for Figure 7.

Model Selection: Models used for transfer experiments are selected by performance on a small held-out validation set (ten tasks) for each task, that does not intersect with the test task.

Meta-RL with RL^2

We adopt the recurrent architecture and hyperparameter settings as specified in the visual maze navigation tasks of Duan et al. [49], except we:

- Use PPO for policy optimization ($\text{clip} = 0.2, \text{value\_coef} = 0.1$)
- Set the entropy bonus coefficient α in an environment-specific manner. We use $\alpha = 0.001$ for MuJoCo Sawyer and $\alpha = 0.1$ for ViZDoom.
- Enlarge the input observation space to $84 \times 84 \times 3$, adapting the encoder by half the stride in the first convolutional layer.
- Increase the size of the recurrent model (hidden state size 512) and the capacity of the output layer of the RNN (MLP with one hidden layer of dimension 256).

- Allow for four episodes per task (instead of two), since the tasks we consider involve more challenging task inference.
- Use a multi-layer perceptron with one-hidden layer to readout the output for the actor and critic, given the recurrent hidden state.

Reward Normalization

A subtle challenge that arises in applying meta-RL across a range of tasks is difference in the statistics of the reward functions encountered, which may affect task inference. Without some form of normalization, the statistics of the rewards of unsupervised meta-training tasks versus those of the downstream tasks may be arbitrarily different, which may interfere with inferring the task. This is especially problematic for RL^2 (compared to e.g. MAML [58]), which relies on encoding the reward as a feature at each timestep. We address this issue by whitening the reward at each timestep with running mean and variance computed online, separately for each task from the unsupervised task distribution during meta-training. At test-time, we share these statistics across tasks from the same test task distribution.

Learning Visual Representations with Deep Clustering

To jointly learn visual representations with the mixture model, we adopt the optimization scheme of DeepCluster [27]. The DeepCluster model is parameterized by the weights of a convolutional neural network encoder as well as a k-means model in embedding space. It is trained in an EM-like fashion, where the M-step additionally involves training the encoder weights via supervised learning of the image-cluster mapping.

Our contribution is that we employ a modified E-step, as presented in the main text, such that the cluster responsibilities are ensured to be consensual across states in a trajectory in the training data. As shown in our experiments, this allows the model to learn trajectory-level visual representations. The full CARML E-step with DeepCluster is presented below.

Algorithm 10 CARML E-Step, a Modified EM Procedure, with DeepCluster

```
1: Require: a set of trajectories $\mathcal{D} = \{(\mathbf{s}_1, \dots, \mathbf{s}_T)\}_{i=1}^N$
2: Initialize $\phi := (\phi_w, \phi_m)$, the weights of encoder $g$ and embedding-space mixture model parameters.
3: while not converged do
4:     Compute $L(\phi_m; \boldsymbol{\tau}, z) = \sum_{\mathbf{s}_t \in \boldsymbol{\tau}} \log q_{\phi_m}(g_{\phi_w}(\mathbf{s}_t)|z)$.
5:     Update via MLE: $\phi_m \leftarrow \arg\max_{\phi'_m} \sum_{i=1}^N L(\phi'_m; \boldsymbol{\tau}_i, z)$.
6:     Obtain training data $\mathcal{D} := \{(\mathbf{s}, y := \arg\max_k q_{\phi_m}(z = k|g_{\phi_w}(\mathbf{s}))\}$.
7:     Update via supervised learning: $\phi_w \leftarrow \arg\max_{\phi'_w} \sum_{(\mathbf{s},y)\in\mathcal{D}} \log q(y|g_{\phi'_w}(\mathbf{s}))$.
8: end while
9: Return: a mixture model $q_\phi(\mathbf{s}, z)$
```

For updating the encoder weights, we use the default hyperparameter settings as described in [27], except 1) we modify the neural network architecture, using a smaller neural network, ResNet-10 [92] with a fixed number of filters (64) for every convolutional layer, and 2) we use number of components $K = 16$, which we did not tune. We tried using a more expressive Gaussian mixture model with full covariances instead of k-means (when training the visual representation), but found that this resulted in overfitting. Hence, we use k-means until the last iteration of EM, wherein a Gaussian mixture model is fitted under the resulting visual representation.

Environments

ViZDoom Environment The environment used for visual navigation is a 500x500 room built

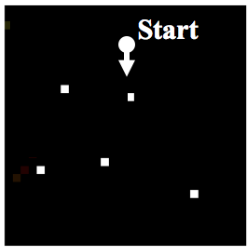


Figure 4.10: Top-down view of VizDoom environment, with initial agent position. White squares depict stationary objects (only relevant to fixed environment).

with ViZDoom [121]. We consider both fixed and random environments; for randomly placing objects, the only constraint enforced is that objects should not be within a minimal distance of one another. There are 50 train objects and 50 test objects. The agent's pose is always initialized to be at the top of the room facing forward. We restrict observations from the environment to be 84×84 RGB images. The maximum episode length is set to 50 timesteps. The hand-crafted reward function corresponds to the inverse l_2 distance from the specified target object.

The environment considered is relatively simple in layout, but compared to simple mazes, can provide a more complex observation space insofar as objects are constantly viewed from different poses and in various combinations, and are often occluded. The underlying ground-truth state space is the product of continuous 2D position and continuous pose spaces. There are three discrete actions that correspond to turning right, turning left, and moving forward, allowing translation and rotation in the pose space that can vary based on position; the result is that the effective visitable set of poses is not strictly limited to a subset of the pose space, despite discretized actions.

Sawyer Environment For visual manipulation, we use a MuJoCo [228] environment involving a simulated Sawyer 7-DOF robotic arm in front of a table, on top of which is an object. The Sawyer arm is controlled by 2D continuous control. It is almost identical to the

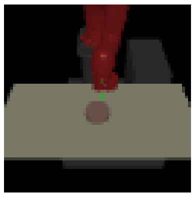


Figure 4.11: Third person view of the Sawyer environment

environment used by prior work such as [161], with the exception that our goal space is that of the object position. The robot pose and object are always initialized to the same position at the top of the room facing forward. We restrict observations from the environment to be 84×84 RGB images. The maximum episode length is set to 50 timesteps. The hand-crafted reward function corresponds to the negative l_2 distance from the specified target object.

4.7.3 Additional Details for Qualitative Study of λ

Instantiating q_ϕ as a VAE

Three factors motivate the use of a variational auto-encoder (VAE) as a generative model for the 2D toy environment. First, a key inductive bias of DeepCluster, namely that randomly initialized convolutional neural networks work surprisingly well, which Caron et al. [27] use to motivate its effectiveness in visual domains, does not apply for our 2D state space. Second, components of a standard Gaussian mixture model are inappropriate for modeling trajectories involving turns. Third, using a VAE allows sampling from a continuous latent, potentially affording an unbounded number of skills.

We construct the VAE model in a manner that enables expressive generative densities $p(\mathbf{s}|z)$ while allowing for computation of the policy reward quantities. We set the VAE latent to be $(\mathbf{z}, t)$, where $p(\mathbf{z}, t) = p(\mathbf{z})p(t) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})\frac{1}{T}$. The form of $p(t)$ follows from restricting the policy to sampling trajectories of length T. We factorize the posterior as $q_\phi(\mathbf{z}, t|\mathbf{s}_{t'}) = q(\mathbf{z}|\mathbf{s}_{t'})\delta(t - t')$. Keeping with the idea of having a Markovian reward, we construct the VAE's recognition network such that it takes as input individual states after training. To incorporate the constraint that all states in a trajectory are mapped to the same posterior, we adopt a particular training scheme: we pass in entire trajectories $\mathbf{s}_{1:T}$, and specify the posterior parameters as $\mu_z = \frac{1}{T}\sum_t g_\eta(\mathbf{s}_t)$ and $\sigma_z^2 = \frac{1}{T}\sum_t g_\eta(\mathbf{s}_t)$.

The ELBO for this model is

$$\mathbb{E}_{\mathbf{z},t\sim q_\phi(\mathbf{z},t|\mathbf{s}_{t'})}\big[\log q_\phi(\mathbf{s}_{t'}|\mathbf{z}, t)\big] - D_{\mathrm{KL}} q_\phi(\mathbf{z}, t|\mathbf{s}_{t'})p(\mathbf{z}, t) \tag{4.35}$$

$$=\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{s}_{t'})}\big[\log q_\phi(\mathbf{s}_{t'}|\mathbf{z}, t')\big] - D_{\mathrm{KL}} q_\phi(\mathbf{z}|\mathbf{s}_{t'})p(\mathbf{z}) - C \tag{4.36}$$

where C is constant with respect to the learnable parameters. The simplification directly follows from the form of the posterior; we have essentially passed t' through the network unchanged. Notice that the computation of the ELBO for a trajectory leverages the conditional independence in our graphical model.

CARML Details

Since we are not interested in meta-transfer for this experiment, we simplify the learning problem to training a contextual policy $\pi_\theta(\mathbf{a}|\mathbf{s}, z)$. To reward the policy using the VAE q_ϕ, we compute

$$r_z(\mathbf{s}) = \lambda \log q_\phi(\mathbf{s}|z) - \log q_\phi(\mathbf{s}) \tag{4.37}$$

where

$$\log q_\phi(\mathbf{s}|z) = \log \sum_t q_\phi(\mathbf{s}|z, t)p(t) = \log \frac{1}{T} \sum_t q_\phi(\mathbf{s}|z, t) \tag{4.38}$$

and we approximate $\log q_\phi(\mathbf{s})$ by its ELBO (Eq. 4.36), substituting the above expression for the reconstruction term.

4.7.4 Sawyer Task Distribution

Visualizing the components of the acquired task distribution for the Sawyer domain reveals structure and diversity related to the position of the object as well as the control path taken to effect movement. Red encodes the true position of the object, and light blue that of the end-effector. We find tasks corresponding to moving the object to various locations in the environment, as well as tasks that correspond to moving the arm in a certain way without object interaction. The tasks provide a scaffold for learning to move the object to various regions of the reachable state space.

Since the Sawyer domain is less visually rich than the VizDoom domain, there may be less visually discriminative states that align with semantics of test task distributions. Moreover, since a large part of the observation is proprioceptive, the discriminative clustering representation used for density modeling captures various proprioceptive features that may not involve object interaction. The consequences are two-fold: 1) the gap in the CARML and the object-centric test task distributions may be large, and 2) the CARML tasks may be too diverse in-so-far as tasks share less structure, and inferring each task involves a different control problem.

4.7.5 Mode Collapse in the Task Distribution

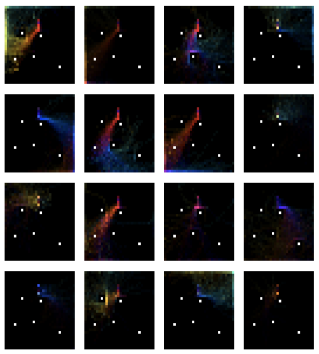
Here, we present visualizations of the task distributions induced by variants of the presented method, to illustrate the issue of using an entirely discrimination-based task acquisition approach. Using the fixed VizDoom setting, we compare:

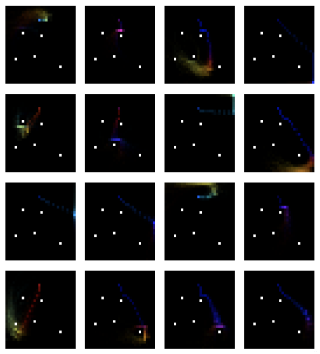
(i) CARML, the proposed method

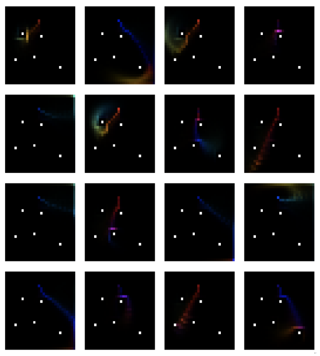
(ii) **online discriminator** – task acquisition with a purely discriminative q_ϕ (akin to an online, pixel-observation-based adaptation of [83]);

(iii) **online pretrained-discriminator** – task acquisition with a discriminative q_ϕ as in **(ii)**, initialized with pre-trained observation encoder.

For all discriminative variants, we found it crucial to use a temperature ≥ 3 to soften the classifier softmax to prevent immediate task mode-collapse.


(i) CARML (ours)


(ii) online discriminator [83]


(iii) pretrained online discriminator

We find the task acquisition of purely discriminative variants **(ii, iii)** to suffer from an effect akin to mode-collapse; the policy's data distribution collapses to a smaller subset of the trajectory space (one or two modes), and tasks correspond to minor variations of these modes. Skill acquisition methods such as DIAYN rely purely on discriminability of states/trajectories under skills, which can be more easily satisfied in high-dimensional observation spaces and can thus lead to such mode-collapse. Moreover, they do not a provide a direct mechanism for furthering exploration once skills are discriminable. On the other hand, the proposed task acquisition approach (Algorithm 9, section 4.3.2) fits a generative model over jointly learned discriminative features, and is thus not only less susceptible to mode-collapse (w.r.t the policy data distribution), but also allows for density-based exploration (section 4.3.3). Indeed, we find that **(iii)** seems to mitigate mode-collapse – benefiting from a pretrained encoder from **(i)** – but does not entirely prevent it. As shown in the main text (Figure 4.7c), in terms of meta-transfer to hand-crafted test tasks, the online discriminative variants **(ii, iii)** perform worse than CARML **(i)**, due to lesser diversity in the task distribution.

4.7.6 Evolution of Task Distribution

Here we consider the evolution of the task distribution in the Random VizDoom environment. The initial tasks (referred to as CARML It. 1) are produced by fitting our deep mixture model to data from a randomly-initialized meta-policy. CARML Its. 2 and 3 correspond to the task distribution after the first and second CARML E-steps, respectively.

We see that the initial tasks tend to be less structured, in so far as the components appear to be noisier and less distinct. With each E-step we see refinement of certain tasks as well as the emergence of others, as the agent's data distribution is shifted by 1) learning the learnable tasks in the current data-distribution, and 2) exploration. In particular, tasks that are "refined" tend to correspond to more simple, exploitative behaviors (i.e. directly heading to an object or a region in the environment, trajectories that are more straight), which may not require exploration to discover. On the other hand, the emergent tasks seem to reflect exploration strategies (i.e. sweeping the space in an efficient manner). We also see the benefit of reorganization that comes from refitting the mixture model, as tasks that were once separate can be combined.

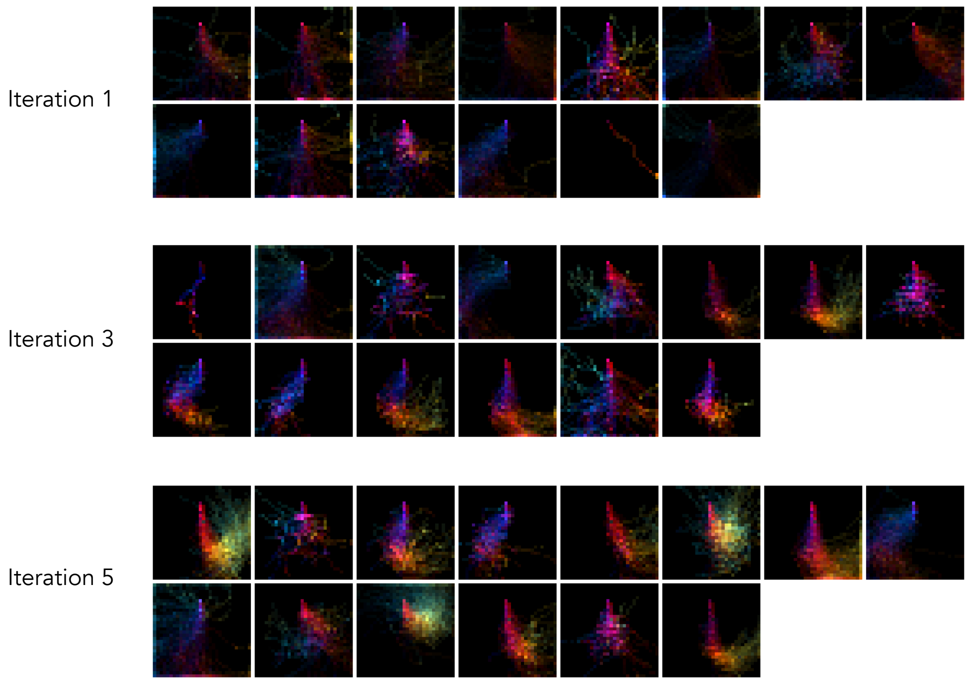


Figure 4.13: Evolution of the CARML task distribution over 3 iterations of fitting q_ϕ in the random ViZDoom visual navigation environment. We observe evidence of task refinement and incorporation of new tasks.

Chapter 5

Looking Forward

It has long been hypothesized that learning paradigms beyond supervised and reinforcement learning – i.e. relatively *unsupervised* learning – must play a key role in the development of artifical intelligence [99]. Recent trends in machine learning – driven by breakthroughs in deep learning, large-scale optimization, and internet-scale data curation – have enabled the rise of scalable pre-training techniques that are beginning to deliver on this promise [43, 21]. This thesis has explored inductive biases that enable scalable pre-training, from the design of architectures capable of adaptive computation for scalable generative modeling, to self-supervised objectives that prepare embodied agents with mechanisms for state representation and reward maximization.

While self-supervised learning and generative modeling both shift the burden of manual representation and annotation to that of computation and data curation, these two approaches span a trade-off of the role of domain knowledge in the design of pre-training schemes. Self-supervised learning requires more task-oriented design of annotation-free objectives, but this additional bias can in principle allow for more targeted pre-training. Generative models solve the more general and challenging task of maximizing the likelihood of data, but this learning objective is more universal and thus stands to scale more gracefully across domains. Taking the lessons of the past decade to heart, it is worth asking how our understanding of their relative advantages might evolve as underlying techniques for generative modeling improve.

5.1 Task-specific Binding as Conditional Inference

A key question is whether the task-oriented bias of self-supervised learning is necessary, or whether a similar effect can be achieved in a more scalable manner i.e. without the design of bespoke objectives. Though generative models optimize more universal objectives, they are not necessarily 'fully' unsupervised insofar as they rely on the curation of data with useful underlying conditional distributions. Indeed, many recent breakthroughs in generative modeling for image[188] and video[104] generation are instances of *conditional* generative models that rely on large datasets of curated data annotated with natural language. While this might suggest that data annotation remains a limiting factor, generative pre-training of large language models has demonstrated that much of the seemingly noisy ambient text data is rich with conditional distributions that capture useful task-specific structure. This is because language – even that of text spewed across the internet – is intrinsically an account of task-relevant latent variables and how they interact to produce goal-directed behaviour. As long as generative models can be formulated to model the joint distribution of a sufficiently universal data interface (e.g. in the case of NLP, variable length sequences with large vocabularies), learning new tasks amounts to specializing to specific conditional inference problems. The versatility of large language models – in terms of sample-efficient transfer to downstream tasks – has highlighted the potential of training on noisy datasets drawn from the joint distribution of tokens for efficiently learn more goal-directed conditional distributions, when paired with supervised finetuning and reinforcement learning from human feedback[169].

5.2 Universal Generative Models and Adaptive Computation

This points to the potential of yet untapped sources of useful conditional generation problems lying dormant in unlabeled data. A promising place to look is the increasing amounts of data across multiple modalities produced as a by-product of human activity, such as video and audio, in addition to text. A simple argument is that natural language is limited in bandwidth and finite in abundance. A more fundamental motivation is that language is just one account of the underlying state of the world and the coincidence of phenomena across modalities may play an important role in learning of more robust representations and algorithms [90, 212]. Large multi-modal data can thus provide additional sources of prior knowledge as well as useful constraints for how this knowledge should be structured. Looking forward, a key challenge will be scalable modeling of joint distributions of heterogeneous data sources that reflect phenenoma across different modalities.

Modeling continuous and discrete data. While we have seen breakthroughs in generative models for discrete and continuous data, these two domains are dominated by different approaches. Discrete data can be directly modeled with likelihood objectives such as auto-regressive models because normalizing probability distributions is tractable for categorical variables. Approaches such as diffusion models excel at modeling high-dimensional continuous because they avoid probability normalization. Recent work has shown how these objectives can be viewed from a unified perspective [106], suggesting the possibility of designing hybrid objectives that marry the best of both. In particular, generative models for text may benefit from advantages of diffusion models such as the ability of parallel refinement for more un-ordered generation (rather than a strict left-to-right bias), the ability to vary depth of the stochastic computation graph during generation, and the hierarchical coarse-to-fine nature of the increasing signal-to-noise ratio in the diffusion reverse process. Beyond the application to more universal generative models, this interplay between unordered and hierarchical generation may improve the coherence of generating long sequences that remain a challenge, such as theorem proving and document generation.

Adaptive computation. The coincidence of phenomena across modalities is useful for learning insofar as these different views of the same underlying latent structure share mutual information [109]. While this is a key motivation for modeling their joint distribution, it is also implies that multi-modal data is not only higher in dimension (e.g. video), but also inherently more redundant. The need for generative model architectures capable of adaptive computation presented in Chapter 2 thus becomes even more important. Understanding the scaling laws of generative models has driven our ability to improve existing systems in a more scientific manner [119, 97]. As we look towards building more universal generative models that span modalities, establishing their scaling laws will be instrumental in understanding the nature of transfer between modalities, and adaptive computation is sure to play a significant role.

Bibliography

[1] Joshua Achiam et al. "Variational option discovery algorithms". In: *arXiv preprint arXiv:1807.10299* (2018).

[2] Pulkit Agrawal, Joao Carreira, and Jitendra Malik. "Learning to See by Moving". In: *ICCV*. 2015.

[3] Antreas Antoniou and Amos Storkey. "Assume, augment and learn: unsupervised few-shot meta-learning via random labels and data augmentation". In: *arXiv preprint arXiv:1902.09884v3* (2019).

[4] Relja Arandjelovic and Andrew Zisserman. "Look, listen and learn". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 609–617.

[5] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. "Layer normalization". In: *arXiv preprint arXiv:1607.06450* (2016).

[6] Bernard J. Baars. "Global workspace theory of consciousness: toward a cognitive neuroscience of human experience". In: *The Boundaries of Consciousness: Neurobiology and Neuropathology*. Ed. by Steven Laureys. Vol. 150. Progress in Brain Research. Elsevier, 2005, pp. 45–53. DOI: `https://doi.org/10.1016/S0079-6123(05)50004-9`. URL: `https://www.sciencedirect.com/science/article/pii/S0079612305500049`.

[7] Philip Bachman, R Devon Hjelm, and William Buchwalter. "Learning representations by maximizing mutual information across views". In: *Advances in Neural Information Processing Systems*. 2019, pp. 15535–15545.

[8] Lars Backstrom and Jure Leskovec. "Supervised random walks: predicting and recommending links in social networks". In: *Proceedings of the fourth ACM international conference on Web search and data mining*. 2011, pp. 635–644.

[9] David Barber and Felix Agakov. "The IM algorithm: a variational approach to information maximization". In: *Neural Information Processing Systems (NeurIPS)*. 2004.

[10] Peter W Battaglia et al. "Relational inductive biases, deep learning, and graph networks". In: *arXiv preprint arXiv:1806.01261* (2018).

[11] Suzanna Becker and Geoffrey E Hinton. "Self-organizing neural network that discovers surfaces in random-dot stereograms". In: *Nature* 355.6356 (1992), pp. 161–163.

[12] Anthony J. Bell and Terrence J. Sejnowski. "An Information-maximization Approach to Blind Separation and Blind Deconvolution". In: *Neural Computation* 7.6 (1995).

[13] Marc Bellemare et al. "Unifying count-based exploration and intrinsic motivation". In: *Neural Information Processing Systems (NeurIPS)*. 2016.

[14] Yoshua Bengio, Aaron Courville, and Pascal Vincent. *Representation Learning: A Review and New Perspectives*. 2014. arXiv: `1206.5538 [cs.LG]`.

[15] Jerome Berclaz et al. "Multiple object tracking using k-shortest paths optimization". In: *IEEE transactions on pattern analysis and machine intelligence* 33.9 (2011), pp. 1806–1819.

[16] Zhangxing Bian et al. "Learning pixel trajectories with multiscale contrastive random walks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 6508–6519.

[17] Piotr Bojanowski and Armand Joulin. "Unsupervised learning by predicting noise". In: *International Conference on Machine Learning (ICML)*. 2017.

[18] Matthew Botvinick et al. "Reinforcement Learning, Fast and Slow". In: *Trends in Cognitive Science* 23.5 (2019).

[19] Yuri Boykov and Gareth Funka-Lea. "Graph cuts and efficient ND image segmentation". In: *International journal of computer vision* 70.2 (2006), pp. 109–131.

[20] Guillem Brasó and Laura Leal-Taixé. "Learning a neural solver for multiple object tracking". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 6247–6257.

[21] Tom Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[22] Yuri Burda et al. "Exploration by random network distillation". In: *International Conference on Learning Representations (ICLR)*. 2019.

[23] Mikhail S Burtsev et al. "Memory transformer". In: *arXiv preprint arXiv:2006.11527* (2020).

[24] Sergi Caelles et al. "One-shot video object segmentation". In: *CVPR*. 2017.

[25] Adam J. Calhoun, Sreekanth H. Chalasani, and Tatyana O. Sharpee. "Maximally informative foraging by *Caenorhabditis elegans*". In: *eLife* 3 (2014).

[26] Nicolas Carion et al. "End-to-End Object Detection with Transformers". In: *arXiv preprint arXiv:2005.12872* (2020).

[27] Mathilde Caron et al. "Deep clustering for unsupervised learning of visual features". In: *European Conference on Computer Vision (ECCV)*. 2018.

[28] Joao Carreira and Andrew Zisserman. "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset". In: *Computer Vision and Pattern Recognition (CVPR)*. 2017.

[29] Joao Carreira et al. "A short note about kinetics-600". In: *arXiv preprint arXiv:1808.01340* (2018).

[30] Rich Caruana. "Multitask Learning". In: *Machine Learning* 28.1 (1997).

[31] Stephanie Chan et al. "Data distributional properties drive emergent in-context learning in transformers". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 18878–18891.

[32] Devendra Singh Chaplot et al. "Gated-attention architectures for task-oriented language grounding". In: *AAAI Conference on Artificial Intelligence*. 2018.

[33] Albert YC Chen and Jason J Corso. "Temporally consistent multi-class video-object segmentation with the video graph-shifts algorithm". In: *2011 IEEE Workshop on Applications of Computer Vision (WACV)*. IEEE. 2011, pp. 614–621.

[34] Ting Chen. "On the importance of noise schedules for diffusion models". In: *arXiv preprint arXiv:2301.10972* (2023).

[35] Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. "Analog bits: Generating discrete data using diffusion models with self-conditioning". In: *arXiv preprint arXiv:2208.04202* (2022).

[36] Ting Chen et al. "A generalist framework for panoptic segmentation of images and videos". In: *arXiv preprint arXiv:2210.06366* (2022).

[37] Ting Chen et al. "A simple framework for contrastive learning of visual representations". In: *arXiv preprint arXiv:2002.05709* (2020).

[38] Sumit Chopra, Raia Hadsell, and Yann LeCun. "Learning a similarity metric discriminatively, with application to face verification". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. IEEE. 2005, pp. 539–546.

[39] Aidan Clark, Jeff Donahue, and Karen Simonyan. "Adversarial video generation on complex datasets". In: *arXiv preprint arXiv:1907.06571* (2019).

[40] Marco Cuturi. "Sinkhorn distances: Lightspeed computation of optimal transport". In: *Advances in neural information processing systems*. 2013, pp. 2292–2300.

[41] Zihang Dai et al. "Transformer-xl: Attentive language models beyond a fixed-length context". In: *arXiv preprint arXiv:1901.02860* (2019).

[42] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *Computer Vision and Pattern Recognition (CVPR)*. 2009.

[43] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[44] Prafulla Dhariwal and Alex Nichol. "Diffusion Models Beat GANs on Image Synthesis". In: *NeurIPS*. 2022.

[45] Sander Dieleman et al. “Continuous diffusion for categorical data”. In: *arXiv preprint arXiv:2211.15089* (2022).

[46] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. “Unsupervised Visual Representation Learning by Context Prediction”. In: *ICCV*. 2015.

[47] Alexey Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929* (2020).

[48] Alexey Dosovitskiy et al. “Discriminative unsupervised feature learning with exemplar convolutional neural networks”. In: *IEEE transactions on pattern analysis and machine intelligence* 38.9 (2015), pp. 1734–1747.

[49] Yan Duan et al. “RL^2: fast reinforcement learning via slow reinforcement learning”. In: *arXiv preprint arXiv:1611.02779* (2016).

[50] Debidatta Dwibedi et al. “Temporal cycle-consistency learning”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1801–1810.

[51] Frederik Ebert et al. “Self-Supervised Visual Planning with Temporal Skip Connections”. In: *Conference on Robotic Learning (CoRL)*. 2017.

[52] Jeffrey L Elman. “Learning and development in neural networks: the importance of starting small”. In: *Cognition* 48.1 (1993).

[53] Martin Engelcke et al. “Genesis: Generative scene inference and sampling with object-centric latent representations”. In: *arXiv preprint arXiv:1907.13052* (2019).

[54] Benjamin Eysenbach et al. “Diversity is all you need: learning skills without a reward function”. In: *International Conference on Learning Representations (ICLR)*. 2019.

[55] Christoph Feichtenhofer et al. “Slowfast networks for video recognition”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 6202–6211.

[56] Jerome Feldman. “The neural binding problem(s)”. In: *Cognitive Neurodynamics* 7 (2013), pp. 1–11.

[57] Michael Figurnov et al. “Spatially adaptive computation time for residual networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1039–1048.

[58] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks”. In: *International Conference on Machine Learning (ICML)*. Sydney, Australia, 2017.

[59] Chelsea Finn and Sergey Levine. “Deep Visual Foresight for Planning Robot Motion”. In: *International Conference on Robotics and Automation (ICRA)*. 2017.

[60] Carlos Florensa et al. “Automatic goal generation for reinforcement learning agents”. In: *International Conference on Machine Learning (ICML)*. 2017.

[61] Carlos Florensa et al. “Reverse curriculum generation for reinforcement learning”. In: *Conference on Robotic Learning (CoRL)*. 2017.

[62] Sébastien Forestier, Yoan Mollard, and Pierre-Yves Oudeyer. “Intrinsically motivated goal exploration processes with automatic curriculum learning”. In: *arXiv preprint arXiv:1708.02190* (2017).

[63] David A. Forsyth and Jean Ponce. *Computer Vision - A Modern Approach, Second Edition.* Pitman, 2012.

[64] Charless Fowlkes et al. “Spectral grouping using the Nystrom method”. In: *IEEE transactions on pattern analysis and machine intelligence* 26.2 (2004), pp. 214–225.

[65] Justin Fu, John Co-Reyes, and Sergey Levine. “EX^2: exploration with exemplar models for deep reinforcement learning”. In: *Neural Information Processing Systems (NeurIPS)*. 2017.

[66] Kunihiko Fukushima. “Neocognitron: A hierarchical neural network capable of visual pattern recognition”. In: *Neural networks* 1.2 (1988), pp. 119–130.

[67] Ruohan Gao, Dinesh Jayaraman, and Kristen Grauman. “Object-centric representation learning from unlabeled videos”. In: *Asian Conference on Computer Vision*. Springer. 2016, pp. 248–263.

[68] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. “A neural algorithm of artistic style”. In: *arXiv preprint arXiv:1508.06576* (2015).

[69] Aude Genevay, Gabriel Peyré, and Marco Cuturi. “Learning generative models with sinkhorn divergences”. In: *International Conference on Artificial Intelligence and Statistics*. 2018, pp. 1608–1617.

[70] Daniel Gordon et al. *Watching the World Go By: Representation Learning from Unlabeled Videos*. 2020.

[71] Ross Goroshin et al. “Unsupervised Learning of Spatiotemporally Coherent Metrics”. In: *ICCV* (2015).

[72] Anirudh Goyal et al. “Coordination among neural modules through a shared global workspace”. In: *arXiv preprint arXiv:2103.01197* (2021).

[73] Alex Graves. “Adaptive computation time for recurrent neural networks”. In: *arXiv preprint arXiv:1603.08983* (2016).

[74] Alex Graves, Greg Wayne, and Ivo Danihelka. “Neural turing machines”. In: *arXiv preprint arXiv:1410.5401* (2014).

[75] Alex Graves et al. “Automated curriculum learning for neural networks”. In: *International Conference on Machine Learning (ICML)*. 2017.

[76] Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. “On the Binding Problem in Artificial Neural Networks”. In: *ArXiv* abs/2012.05208 (2020).

[77] Klaus Greff, Sjoerd Van Steenkiste, and Jue rgen Schmidhuber. “Neural expectation maximization”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 6691–6701.

[78] Klaus Greff et al. “Multi-object representation learning with iterative variational inference”. In: *arXiv preprint arXiv:1903.00450* (2019).

[79] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. “Variational Intrinsic Control”. In: *arXiv preprint arXiv:1611.07507* (2016).

[80] Karol Gregor et al. “Draw: A recurrent neural network for image generation”. In: *International conference on machine learning*. PMLR. 2015, pp. 1462–1471.

[81] Aditya Grover and Jure Leskovec. “node2vec: Scalable feature learning for networks”. In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016, pp. 855–864.

[82] Abhishek Gupta et al. “Meta-Reinforcement Learning of Structured Exploration Strategies”. In: *Neural Information Processing Systems (NeurIPS)*. 2018.

[83] Abhishek Gupta et al. “Unsupervised Meta-Learning for Reinforcement Learning”. In: *arXiv preprint arXiv:1806.04640* (2018).

[84] Michael Gutmann and Aapo Hyvärinen. “Noise-contrastive estimation: A new estimation principle for unnormalized statistical models”. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 2010, pp. 297–304.

[85] Dylan Hadfield-Menell et al. “Inverse reward design”. In: *Neural Information Processing Systems (NeurIPS)*. 2017.

[86] Alon Halevy, Peter Norvig, and Fernando Pereira. “The unreasonable effectiveness of data”. In: *IEEE intelligent systems* 24.2 (2009), pp. 8–12.

[87] Will Hamilton, Zhitao Ying, and Jure Leskovec. “Inductive representation learning on large graphs”. In: *Advances in neural information processing systems*. 2017, pp. 1024–1034.

[88] William L Hamilton, Rex Ying, and Jure Leskovec. “Representation learning on graphs: Methods and applications”. In: *arXiv preprint arXiv:1709.05584* (2017).

[89] Tengda Han, Weidi Xie, and Andrew Zisserman. “Video representation learning by dense predictive coding”. In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2019, pp. 0–0.

[90] Stevan Harnad. “The symbol grounding problem”. In: *Physica D: Nonlinear Phenomena* 42.1-3 (June 1990), pp. 335–346. DOI: `10.1016/0167-2789(90)90087-6`. URL: `https://doi.org/10.1016%2F0167-2789%2890%2990087-6`.

[91] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. “Unsupervised learning”. In: *The Elements of Statistical Learning*. Springer, 2009.

[92] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *CoRR* abs/1512.03385 (2015). arXiv: `1512.03385`. URL: `http://arxiv.org/abs/1512.03385`.

[93] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *Computer Vision and Pattern Recognition (CVPR)*. 2016.

[94] Kaiming He et al. "Momentum contrast for unsupervised visual representation learning". In: *CVPR*. 2020.

[95] Olivier J Hénaff et al. "Data-efficient image recognition with contrastive predictive coding". In: *arXiv preprint arXiv:1905.09272* (2019).

[96] Dan Hendrycks and Kevin Gimpel. "Gaussian error linear units (gelus)". In: *arXiv preprint arXiv:1606.08415* (2016).

[97] Tom Henighan et al. *Scaling Laws for Autoregressive Generative Modeling*. 2020. arXiv: `2010.14701 [cs.LG]`.

[98] Martin Heusel et al. "Gans trained by a two time-scale update rule converge to a local nash equilibrium". In: *Advances in neural information processing systems* 30 (2017).

[99] Geoffrey Hinton and Terrence J Sejnowski. *Unsupervised learning: foundations of neural computation*. MIT press, 1999.

[100] R Devon Hjelm et al. "Learning deep representations by mutual information estimation and maximization". In: *International Conference on Learning Representations*. 2018.

[101] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising Diffusion Probabilistic Models". In: *NeurIPS* (2020).

[102] Jonathan Ho and Tim Salimans. "Classifier-Free Diffusion Guidance". In: *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*. 2021.

[103] Jonathan Ho et al. "Cascaded Diffusion Models for High Fidelity Image Generation". In: *JMLR* (2022).

[104] Jonathan Ho et al. "Imagen video: High definition video generation with diffusion models". In: *arXiv preprint arXiv:2210.02303* (2022).

[105] Jonathan Ho et al. "Video Diffusion Models". In: *NeurIPS*. 2022.

[106] Emiel Hoogeboom et al. *Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions*. 2021. arXiv: `2102.05379 [stat.ML]`.

[107] Rein Houthooft et al. "Evolved Policy Gradients". In: *Neural Information Processing Systems (NeurIPS)*. 2018.

[108] Kyle Hsu, Sergey Levine, and Chelsea Finn. "Unsupervised learning via meta-learning". In: *International Conference on Learning Representations (ICLR)*. 2019.

[109] Phillip Isola et al. "Learning visual groups from co-occurrences in space and time". In: *arXiv preprint arXiv:1511.06811* (2015).

[110] Allan Jabri, David Fleet, and Ting Chen. "Scalable Adaptive Computation for Iterative Generation". In: *International conference on machine learning* (2023).

[111] Allan Jabri, Andrew Owens, and Alexei Efros. “Space-time correspondence as a contrastive random walk”. In: *Advances in neural information processing systems* 33 (2020), pp. 19545–19560.

[112] Allan Jabri et al. “Unsupervised curricula for visual meta-reinforcement learning”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[113] Andrew Jaegle et al. “Perceiver io: A general architecture for structured inputs & outputs”. In: *arXiv preprint arXiv:2107.14795* (2021).

[114] Andrew Jaegle et al. “Perceiver: General perception with iterative attention”. In: *International conference on machine learning*. PMLR. 2021, pp. 4651–4664.

[115] Dinesh Jayaraman and Kristen Grauman. “Learning image representations tied to egomotion”. In: *ICCV*. 2015.

[116] Neel Joshi et al. “Real-time hyperlapse creation via optimal frame selection”. In: *ACM Transactions on Graphics (TOG)* 34.4 (2015), pp. 1–9.

[117] Armand Joulin and Tomas Mikolov. “Inferring algorithmic patterns with stack-augmented recurrent nets”. In: *Advances in neural information processing systems* 28 (2015).

[118] Armand Joulin, Kevin Tang, and Li Fei-Fei. “Efficient image and video co-localization with frank-wolfe algorithm”. In: *European Conference on Computer Vision*. Springer. 2014, pp. 253–268.

[119] Jared Kaplan et al. “Scaling laws for neural language models”. In: *arXiv preprint arXiv:2001.08361* (2020).

[120] Tero Karras et al. “Elucidating the Design Space of Diffusion-Based Generative Models”. In: *arXiv preprint arXiv:2206.00364* (2022).

[121] Michał Kempka et al. “ViZDoom: a Doom-based AI Research Platform for Visual Reinforcement Learning”. In: *Conference on Computational Intelligence and Games (CIG)*. 2016.

[122] Siavash Khodadadeh, Ladislau Bölöni, and Mubarak Shah. “Unsupervised Meta-Learning For Few-Shot Image and Video Classification”. In: *arXiv preprint arXiv:1811.11819v1* (2018).

[123] Diederik Kingma et al. “Variational diffusion models”. In: *Advances in neural information processing systems* 34 (2021), pp. 21696–21707.

[124] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv* (2014).

[125] Diederik P Kingma and Max Welling. “Auto-encoding variational Bayes”. In: *arXiv preprint arXiv:1312.6114* (2014).

[126] Thomas Kipf, Elise van der Pol, and Max Welling. “Contrastive Learning of Structured World Models”. In: *International Conference on Learning Representations*. 2020. URL: `https://openreview.net/forum?id=H1gax6VtDB`.

[127] Thomas Kipf et al. “Neural relational inference for interacting systems”. In: *arXiv preprint arXiv:1802.04687* (2018).

[128] Thomas N. Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: *International Conference on Learning Representations (ICLR)*. 2017.

[129] Shu Kong and Charless Fowlkes. “Multigrid Predictive Filter Flow for Unsupervised Learning on Videos”. In: *arXiv preprint arXiv:1904.01693* (2019).

[130] Bruno Korbar, Du Tran, and Lorenzo Torresani. “Cooperative learning of audio and video models from self-supervised synchronization”. In: *Advances in Neural Information Processing Systems*. 2018.

[131] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. “CIFAR-10 (Canadian Institute for Advanced Research)”. In: (). URL: `http://www.cs.toronto.edu/~kriz/cifar.html`.

[132] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012.

[133] Zihang Lai, Erika Lu, and Weidi Xie. “MAST: A Memory-Augmented Self-supervised Tracker”. In: *arXiv preprint arXiv:2002.07793* (2020).

[134] Zihang Lai and Weidi Xie. “Self-supervised learning for video correspondence flow”. In: *arXiv preprint arXiv:1905.00875* (2019).

[135] Yann LeCun et al. “Backpropagation applied to handwritten zip code recognition”. In: *Neural computation* 1.4 (1989), pp. 541–551.

[136] Juho Lee et al. “Set Transformer”. In: *CoRR* abs/1810.00825 (2018). arXiv: `1810.00825`. URL: `http://arxiv.org/abs/1810.00825`.

[137] Juho Lee et al. “Set transformer: A framework for attention-based permutation-invariant neural networks”. In: *International conference on machine learning*. PMLR. 2019, pp. 3744–3753.

[138] Joel Lehman and Kenneth O Stanley. “Abandoning objectives: evolution through the search for novelty alone”. In: *Evolutionary Computation* 19.2 (2011).

[139] Xueting Li et al. “Joint-task self-supervised learning for temporal correspondence”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 317–327.

[140] Yin Li et al. “Unsupervised Learning of Edges”. In: *CVPR*. 2016.

[141] Zeming Lin et al. “Evolutionary-scale prediction of atomic-level protein structure with a language model”. In: *Science* 379.6637 (2023), pp. 1123–1130.

[142] Francesco Locatello et al. “Object-centric learning with slot attention”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 11525–11538.

[143] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *CVPR*. 2015.

[144] Manuel Lopes et al. "Exploration in Model-based Reinforcement Learning by Empirically Estimating Learning Progress". In: *Neural Information Processing Systems (NeurIPS)*. 2012.

[145] William Lotter, Gabriel Kreiman, and David Cox. "Deep predictive coding networks for video prediction and unsupervised learning". In: *arXiv preprint arXiv:1605.08104* (2016).

[146] Guansong Lu et al. "Large-Scale Optimal Transport via Adversarial Training with Cycle-Consistency". In: *arXiv preprint arXiv:2003.06635* (2020).

[147] Pauline Luc et al. "Transformation-based adversarial video prediction on large-scale data". In: *arXiv preprint arXiv:2003.04035* (2020).

[148] Bruce D Lucas and Takeo Kanade. "An iterative image registration technique with an application to stereo vision". In: *IJCAI* (1981).

[149] Troy Luhman and Eric Luhman. "Improving diffusion model efficiency through patching". In: *arXiv preprint arXiv:2207.04316* (2022).

[150] Jonathon Luiten, Paul Voigtlaender, and Bastian Leibe. "PReMVOS: Proposal-generation, refinement and merging for video object segmentation". In: *Asian Conference on Computer Vision*. Springer. 2018, pp. 565–580.

[151] Zelun Luo et al. "Unsupervised learning of long-term motion dynamics for videos". In: (2017).

[152] K. K. Maninis et al. "Video Object Segmentation without Temporal Information". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.6 (2019), pp. 1515–1530.

[153] Michaël Mathieu, Camille Couprie, and Yann LeCun. "Deep multi-scale video prediction beyond mean square error". In: *arXiv* (2015).

[154] Tambet Matiisen et al. "Teacher-student curriculum learning". In: *Transactions on Neural Networks and Learning Systems* (2019).

[155] Meila, Marina and Shi, Jianbo. "Learning segmentation by random walks". In: *Advances in neural information processing systems*. 2001, pp. 873–879.

[156] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[157] Nikhil Mishra et al. "A simple neural attentive meta-learner". In: *International Conference on Learning Representations (ICLR)*. 2018.

[158] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. "Shuffle and Learn: Unsupervised Learning using Temporal Order Verification". In: *ECCV*. 2016.

[159] Shakir Mohamed and Danilo J. Rezende. “Variational Information Maximisation for Intrinsically Motivated Reinforcement Learning”. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. NIPS’15. Montreal, Canada: MIT Press, 2015, pp. 2125–2133. URL: `http://dl.acm.org/citation.cfm?id=2969442.2969477`.

[160] Ravi Teja Mullapudi et al. “Online model distillation for efficient video inference”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 3573–3582.

[161] Ashvin Nair et al. “Visual Reinforcement Learning with Imagined Goals”. In: *Neural Information Processing Systems (NeurIPS)*. 2018.

[162] Charlie Nash et al. “Transframer: Arbitrary Frame Prediction with Generative Models”. In: *arXiv preprint arXiv:2203.09494* (2022).

[163] Alex Nichol and Prafulla Dhariwal. “Improved denoising diffusion probabilistic models”. In: *arXiv preprint arXiv:2102.09672* (2021).

[164] Sourabh A Niyogi and Edward H Adelson. “Analyzing gait with spatiotemporal surfaces”. In: *Proceedings of 1994 IEEE Workshop on Motion of Non-rigid and Articulated Objects*. IEEE. 1994, pp. 64–69.

[165] Mehdi Noroozi and Paolo Favaro. “Unsupervised learning of visual representations by solving jigsaw puzzles”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 69–84.

[166] Seoung Wug Oh et al. “Video object segmentation using space-time memory networks”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 9226–9235.

[167] Catherine Olsson et al. “In-context learning and induction heads”. In: *arXiv preprint arXiv:2209.11895* (2022).

[168] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. “Representation learning with contrastive predictive coding”. In: *arXiv preprint arXiv:1807.03748* (2018).

[169] OpenAI. *GPT-4 Technical Report*. 2023. arXiv: `2303.08774 [cs.CL]`.

[170] Ian Osband, John Aslanides, and Albin Cassirer. “Randomized Prior Functions for Deep Reinforcement Learning”. In: *Neural Information Processing Systems (NeurIPS)*. 2018.

[171] Andrew Owens and Alexei A Efros. “Audio-visual scene analysis with self-supervised multisensory features”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 631–648.

[172] Andrew Owens et al. “Visually indicated sounds”. In: *Computer Vision and Pattern Recognition (CVPR)*. 2016.

[173] Deepak Pathak et al. “Curiosity-driven exploration by self-supervised prediction”. In: *International Conference on Machine Learning (ICML)*. 2017.

[174] Deepak Pathak et al. "Learning Features by Watching Objects Move". In: *CVPR*. 2017.

[175] Deepak Pathak et al. "Zero-Shot Visual Imitation". In: *International Conference on Learning Representations (ICLR)*. 2018.

[176] William Peebles and Saining Xie. "Scalable Diffusion Models with Transformers". In: *arXiv preprint arXiv:2212.09748* (2022).

[177] F. Perazzi et al. "A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation". In: *Computer Vision and Pattern Recognition*. 2016.

[178] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. "DeepWalk". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14* (2014). DOI: `10.1145/2623330.2623732`. URL: `http://dx.doi.org/10.1145/2623330.2623732`.

[179] Gabriel Peyré, Marco Cuturi, and Justin Solomon. "Gromov-wasserstein averaging of kernel and distance matrices". In: *International Conference on Machine Learning*. 2016, pp. 2664–2672.

[180] Jean Piaget. *The Construction of Reality in the Child*. Basic Books, 1954.

[181] Sören Pirk et al. "Online object representations with contrastive learning". In: *arXiv preprint arXiv:1906.04312* (2019).

[182] Hamed Pirsiavash, Deva Ramanan, and Charless C. Fowlkes. "Globally-optimal greedy algorithms for tracking a variable number of objects". In: *CVPR 2011*. 2011, pp. 1201–1208.

[183] Jordi Pont-Tuset et al. "The 2017 DAVIS Challenge on Video Object Segmentation". In: *arXiv:1704.00675* (2017).

[184] Alec Radford and Karthik Narasimhan. "Improving Language Understanding by Generative Pre-Training". In: 2018.

[185] Jack W Rae et al. "Compressive transformers for long-range sequence modelling". In: *arXiv preprint arXiv:1911.05507* (2019).

[186] Kate Rakelly et al. "Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables". In: *International Conference on Machine Learning*. 2019.

[187] Deva Ramanan, David A Forsyth, and Andrew Zisserman. "Strike a pose: Tracking people by finding stylized poses". In: *CVPR*. 2005.

[188] Aditya Ramesh et al. "Hierarchical text-conditional image generation with clip latents". In: *arXiv preprint arXiv:2204.06125* (2022).

[189] Robin Rombach et al. "High-resolution image synthesis with latent diffusion models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10684–10695.

[190] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[191] Frank Rosenblatt. "PRINCIPLES OF NEURODYNAMICS. PERCEPTRONS AND THE THEORY OF BRAIN MECHANISMS". In: *American Journal of Psychology* 76 (1963), p. 705.

[192] Jonas Rothfuss et al. "ProMP: proximal Meta-Policy Search". In: *International Conference on Learning Representations (ICLR)*. 2019.

[193] Olga Russakovsky et al. "Imagenet large scale visual recognition challenge". In: *International journal of computer vision* 115.3 (2015), pp. 211–252.

[194] Virginia R de Sa. "Learning classification with unlabeled data". In: *Advances in neural information processing systems*. 1994, pp. 112–119.

[195] Christoph Salge, Cornelius Glackin, and Daniel Polani. "Empowerment – an introduction". In: *Guided Self-Organization: Inception*. Springer, 2014.

[196] Tim Salimans et al. "Improved techniques for training gans". In: *Advances in neural information processing systems* 29 (2016).

[197] Paul-Edouard Sarlin et al. "Superglue: Learning feature matching with graph neural networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 4938–4947.

[198] Nikolay Savinov et al. "Step-unrolled denoising autoencoders for text generation". In: *arXiv preprint arXiv:2112.06749* (2021).

[199] Tom Schaul et al. "Universal value function approximators". In: *International Conference on Machine Learning*. 2015, pp. 1312–1320.

[200] Jürgen Schmidhuber. "Driven by Compression Progress: a Simple Principle Explains Essential Aspects of Subjective Beauty, Novelty, Surprise, Interestingness, Attention, Curiosity, Creativity, Art, Science, Music, Jokes". In: *Anticipatory Behavior in Adaptive Learning Systems*. Springer-Verlag, 2009.

[201] Jürgen Schmidhuber. "Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook". PhD thesis. Technische Universität München, 1987.

[202] Jürgen Schmidhuber. "POWERPLAY: Training an Increasingly General Problem Solver by Continually Searching for the Simplest Still Unsolvable Problem". In: *arXiv preprint arXiv:1112.5309* (2011).

[203] John Schulman et al. "Proximal Policy Optimization Algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[204] Steven M Seitz and Simon Baker. "Filter flow". In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE. 2009, pp. 143–150.

[205] Pierre Sermanet et al. "Time-contrastive networks: Self-supervised learning from video". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1134–1141.

[206] Jianbo Shi and Jitendra Malik. "Motion segmentation and tracking using normalized cuts". In: *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*. IEEE. 1998, pp. 1154–1160.

[207] Jianbo Shi and Jitendra Malik. "Normalized cuts and image segmentation". In: *IEEE Transactions on pattern analysis and machine intelligence* 22.8 (2000), pp. 888–905.

[208] Assaf Shocher, Nadav Cohen, and Michal Irani. ""Zero-Shot" Super-Resolution using Deep Internal Learning". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[209] Pranav Shyam, Wojciech Jaśkowski, and Faustino Gomez. "Model-Based Active Exploration". In: *International Conference on Machine Learning (ICML)*. 2019.

[210] Satinder Singh, Andrew G Barto, and Nuttapong Chentanez. "Intrinsically Motivated Reinforcement Learning". In: *Neural Information Processing Systems (NeurIPS)*. 2005.

[211] Richard Sinkhorn and Paul Knopp. "Concerning nonnegative matrices and doubly stochastic matrices". In: *Pacific Journal of Mathematics* 21.2 (1967), pp. 343–348.

[212] Linda B. Smith and Michael Gasser. "The Development of Embodied Cognition: Six Lessons from Babies". In: *Artificial Life* 11 (2005), pp. 13–29.

[213] Jascha Sohl-Dickstein et al. "Deep unsupervised learning using nonequilibrium thermodynamics". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 2256–2265.

[214] Jiaming Song, Chenlin Meng, and Stefano Ermon. "Denoising diffusion implicit models". In: *arXiv preprint arXiv:2010.02502* (2020).

[215] Yang Song et al. "Score-Based Generative Modeling through Stochastic Differential Equations". In: *International Conference on Learning Representations*. 2021.

[216] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. "Unsupervised Learning of Video Representations using LSTMs". In: *arXiv* (2015).

[217] Nitish Srivastava et al. "Dropout: A simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* (2014), pp. 1929–1958.

[218] Bradly C Stadie et al. "Some Considerations on Learning to Explore via Meta-Reinforcement Learning". In: *Neural Information Processing Systems (NeurIPS)*. 2018.

[219] Robin Strudel et al. "Self-conditioned Embedding Diffusion for Text Generation". In: *arXiv preprint arXiv:2211.04236* (2022).

[220] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. "End-to-end memory networks". In: *Advances in neural information processing systems* 28 (2015).

[221] Sainbayar Sukhbaatar et al. "Intrinsic motivation and automatic curricula via asymmetric self-play". In: *International Conference on Learning Representations (ICLR)*. 2018.

[222] Yu Sun et al. "Test-Time Training with Self-Supervision for Generalization under Distribution Shifts". In: *International Conference on Machine Learning (ICML)*. 2020.

[223] Flood Sung et al. "Learning to learn: meta-critic networks for sample efficient learning". In: *arXiv preprint arXiv:1706.09529* (2017).

[224] Richard Sutton. "The bitter lesson". In: *Incomplete Ideas (blog)* 13.1 (2019).

[225] Jian Tang et al. "Line: Large-scale information network embedding". In: *Proceedings of the 24th international conference on world wide web*. 2015, pp. 1067–1077.

[226] Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 1998.

[227] Yonglong Tian, Dilip Krishnan, and Phillip Isola. "Contrastive Multiview Coding". In: *CoRR* abs/1906.05849 (2019). arXiv: `1906.05849`. URL: `http://arxiv.org/abs/1906.05849`.

[228] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: a physics engine for model-based control". In: *International Conference on Intelligent Robots and Systems (IROS)*. 2012.

[229] Pavel Tokmakov et al. "Object Permanence Emerges in a Random Walk along Memory". In: *International conference on machine learning* (2022).

[230] Anne Treisman and Garry A. Gelade. "A feature-integration theory of attention". In: *Cognitive Psychology* 12 (1980), pp. 97–136.

[231] Michael Tschannen et al. "Self-Supervised Learning of Video-Induced Visual Invariances". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[232] Hsiao-Yu Tung et al. "Self-supervised learning of motion capture". In: *NIPS*. 2017.

[233] Thomas Unterthiner et al. "Towards accurate generative models of video: A new metric & challenges". In: *arXiv preprint arXiv:1812.01717* (2018).

[234] Jack Valmadre et al. "Long-term tracking in the wild: A benchmark". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 670–685.

[235] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[236] Petar Veličković et al. "Graph Attention Networks". In: *International Conference on Learning Representations*. 2018. URL: `https://openreview.net/forum?id=rJXMpikCZ`.

[237] Paul Voigtlaender and Bastian Leibe. "Online adaptation of convolutional neural networks for video object segmentation". In: *arXiv* (2017).

[238] Paul Voigtlaender et al. “MOTS: Multi-Object Tracking and Segmentation”. In: *CoRR* abs/1902.03604 (2019). arXiv: `1902.03604`. URL: `http://arxiv.org/abs/1902.03604`.

[239] Carl Vondrick et al. “Tracking emerges by colorizing videos”. In: *ECCV*. 2017.

[240] Jacob Walker, Ali Razavi, and Aäron van den Oord. “Predicting video with vqvae”. In: *arXiv preprint arXiv:2103.01950* (2021).

[241] Jane X Wang et al. “Learning to reinforcement learn”. In: *Annual Meeting of the Cognitive Science Society (CogSci)*. 2016.

[242] Ning Wang et al. “Unsupervised deep tracking”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1308–1317.

[243] Qiang Wang et al. “Fast online object tracking and segmentation: A unifying approach”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2019, pp. 1328–1338.

[244] Xiaolong Wang and Abhinav Gupta. “Unsupervised Learning of Visual Representations using Videos”. In: *ICCV*. 2015.

[245] Xiaolong Wang, Allan Jabri, and Alexei A Efros. “Learning correspondence from the cycle-consistency of time”. In: *CVPR*. 2019.

[246] Xiaolong Wang et al. “Non-local Neural Networks”. In: *CVPR*. 2018.

[247] David Warde-Farley et al. “Unsupervised Control Through Non-Parametric Discriminative Rewards”. In: *International Conference on Learning Representations (ICLR)*. 2019.

[248] Manuel Watter et al. “Embed to Control: a Locally Linear Latent Dynamics Model for Control from Raw Images”. In: *Neural Information Processing Systems (NeurIPS)*. 2015.

[249] Donglai Wei et al. “Learning and Using the Arrow of Time”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[250] Max Wertheimer. “Laws of Organization in Perceptual Forms”. In: *A source book of Gestalt psychology*. London: Routledge & Kegan Paul, 1938, pp. 71–88.

[251] Jason Weston, Sumit Chopra, and Antoine Bordes. “Memory networks”. In: *arXiv preprint arXiv:1410.3916* (2014).

[252] Robert W White. “Motivation reconsidered: the concept of competence”. In: *Psychological Review* 66.5 (1959).

[253] Josh Wills, Sameer Agarwal, and Serge Belongie. “What Went Where”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. Madison, WI, Jan. 1, 2003, pp. 37–44. URL: `/se3/wp-content/uploads/2014/09/01211335.pdf`.

[254] Zhirong Wu et al. "Unsupervised feature learning via non-parametric instance discrimination". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3733–3742.

[255] Annie Xie et al. "Few-shot goal inference for visuomotor learning and planning". In: *Conference on Robot Learning (CoRL)*. 2018.

[256] Hongteng Xu et al. "Gromov-wasserstein learning for graph matching and node embedding". In: *arXiv preprint arXiv:1901.06003* (2019).

[257] Linjie Yang et al. "Efficient video object segmentation via network modulation". In: (2018).

[258] Hongxu Yin et al. "AdaViT: Adaptive Tokens for Efficient Vision Transformer". In: *arXiv preprint arXiv:2112.07658* (2021).

[259] Yang You et al. "Large batch optimization for deep learning: Training bert in 76 minutes". In: *arXiv preprint arXiv:1904.00962* (2019).

[260] Manzil Zaheer et al. "Big bird: Transformers for longer sequences". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 17283–17297.

[261] Amir Roshan Zamir, Afshin Dehghan, and Mubarak Shah. "Gmcp-tracker: Global multi-object tracking using generalized minimum clique graphs". In: *European Conference on Computer Vision*. Springer. 2012, pp. 343–356.

[262] Lihi Zelnik-Manor and Michal Irani. "Event-based analysis of video". In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 2. IEEE. 2001, pp. II–II.

[263] Li Zhang, Yuan Li, and Ramakant Nevatia. "Global data association for multi-object tracking using network flows". In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2008, pp. 1–8.

[264] Richard Zhang, Phillip Isola, and Alexei A Efros. "Split-brain autoencoders: Unsupervised learning by cross-channel prediction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1058–1067.

[265] Qixian Zhou et al. "Adaptive Temporal Encoding Network for Video Instance-level Human Parsing". In: *ACM MM*. 2018.

[266] Tinghui Zhou et al. "Learning dense correspondence via 3d-guided cycle consistency". In: *CVPR*. 2016.