Crafting Deep Learning Models for Reinforcement Learning and Computer Vision Applications: Novel Representation Learning Frameworks for Strong Inductive Biases

Junxia Deng
University of Southern California
California, USA

Ceil Hong. Zhang
Massachusetts Institute of Technology
Massachusetts, USA
zhanghongceil@pku.org.cn

SUMMARY

This thesis *Crafting Deep Learning Models for Reinforcement Learning and Computer Vision Applications* focuses on designing novel and effective representation learning frameworks. There are two major aspects in our proposed approaches: neural network model architecture design and objective engineering. To demonstrate how each aspect can be maneuvered, we delve into representative applications from two important areas of studies in artificial intelligence, namely reinforcement and computer vision. In both areas, we emphasize how to manipulate abstract representations to build in strong inductive biases from the target tasks and the type of available data. We hope our examples may shed light on future endeavors in tackling problems from related areas and beyond.

The first part of the thesis looks into representative tasks in reinforcement learning. Our contributions are listed as follows:

- As a starting point, we aim at improving general and exploratory behavior and reflect environment uncertainty for a popular class of model-free, on-policy RL algorithm, actor critic methods. To this end, we propose ***stochastic actor-critic methods*** (Shang et al., 2019b; Chapter 2). It incorporates an effective and flexible way to inject randomness into actor-critic models. The randomness is injected to the high-level abstract representations. We test several actor-critic models enhanced with stochastic activations and demonstrate their effectiveness in a wide range of Atari 2600 games, a continuous control problem and a car racing task.

- Next, we turn our attention to how to allow structured exploration in a more specific albeit common RL problem setting: a persistent environment, or world, that hosts a diverse suite of tasks. To this end, we propose ***world graph decompositions over the environments to accelerate reinforcement learning*** (Shang et al., 2019a; Chapter 3). The nodes of

a world graph are important *waypoint* states and edges represent feasible traversals between them. After identifying the world graph, our framework then applies it to a hierarchical RL algorithm to bias exploration towards task-relevant waypoints and regions. We thoroughly evaluate our approach on a suite of challenging maze tasks and show that using the world graph abstraction of the environment significantly accelerates RL, achieving higher reward and faster learning.

- Lastly, we consider the scenario where multiple agents must cooperate to achieve a common goal, a subset of multi-agent RL. Specifically, we propose ***to incorporate agent-centric representations for multi-agent reinforcement learning*** (Shang et al., 2020a; Chapter 4) in two ways. First, we introduce an agent-centric attention module with explicit connections across the agents. The attention module is built over the abstract representations of the agents. Second, we leverage an agent-centric unsupervised predictive objective, to be used as an auxiliary loss or as the basis of a pre-training step. We evaluate these approaches on the Google Research Football environment as well as DeepMind Lab 2D and show they lead to the emergence of more complex cooperation strategies between agents as well as enhanced sample efficiency and generalization.

The second part of the thesis shifts its focus to unsupervised learning for various computer vision tasks and domains. Our contributions are listed as follows:

- To better leverage unlabeled data and enhance unsupervised image modeling, we propose ***channel-recurrent variational autoencoders*** (crVAE) (Shang et al., 2018; Chapter 5). It integrates recurrent connections across channels of the abstract convolutional features to both inference and generation steps, allowing the resulting high-level features to be captured in global-to-local, coarse-to-fine manners. Combined with adversarial loss, the resulting channel-recurrent VAE-GAN (crVAE-GAN) outperforms the baseline VAE-GAN in generating a diverse spectrum of high resolution images while maintaining the same level of computational efficacy.

- As an immediate next step, we further extend the channel-recurrent framework and propose ***attentive conditional channel-recurrent autoencoding*** (acVAE) (Shang and Sohn, 2019; Chapter 6) for attribute-conditioned face

synthesis. Evaluations are performed through both qualitative visual examination and quantitative metrics, namely inception scores, human preferences, and attribute classification accuracy.

- Finally, we consider unsupervised learning with unlabeled video sequences and propose ***to learn video-level statics and dynamics representations*** (Shang et al., 2020b; Chapter 7) by decomposing videos from temporal coherence and dynamics. We demonstrate the significance of the learned representations over several applications, including a novel dynamics retrieval task, on a face, a human activity, and a robotics grasping datasets.

LIST OF PUBLICATIONS AND WORKING PAPERS

This thesis is based on the following publications:

- **Wenling Shang**, Douwe van der Wal, Herke van Hoof and Max Welling (2019). "Stochastic Activation Actor-Critic Methods." In: *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*.

- **Wenling Shang**, Stephan Zheng*, Alex Trott*, Caiming Xiong and Richard Socher (2019). "Learning World Graphs for Accelerated Hierarchical Reinforcement Learning" In: *ICML Real-world Sequential Decision Making Workshop*.

- **Wenling Shang**, Lasse Espeholt, Anton Raichuk, and Tim Salimans (2020). "Agent-centric Representations for Multi-agent Reinforcement Learning."

- **Wenling Shang**, Kihyuk Sohn and Yuandong Tian (2019). "Channel-Recurrent Autoencoding for Image Modeling." In: *Winter Conference on Applications of Computer Vision (WACV)*.

- **Wenling Shang** and Kihyuk Sohn (2020). "Attentive Conditional Channel-Recurrent Autoencoding for Attribute Conditioned Face Synthesis." In: *Winter Conference on Applications of Computer Vision (WACV)*.

- **Wenling Shang**, Sifei Liu, Arash Vahdat, Shalini De Mello and Jan Kautz (2020). "Unsupervised Learning from Temporal Coherence for Dynamics-based Retrieval."

Main ideas, experiments and paper writings are mainly contributions from the first author. The co-authors have played essential roles in advising and editing.

CONTENTS

1 INTRODUCTION

The introduction chapter provides a general overview and background. It starts with an introduction to the topic of artificial intelligence. It then further introduces a primary subject within AI that has predominantly propelled its advancements in recent years, deep representation learning, along with two domains of applications, reinforcement learning, and computer vision. This thesis centers around these two domains of applications. Finally, we discuss two crucial ingredients in crafting deep learning models, neural network architectures, and objective function engineering.

The remainder of the thesis studies and demonstrates in detail how to maneuver these critical ingredients for several applications. Particularly, we manipulate these ingredients to better shape the abstract representations and incorporate the essential inductive biases for a target task and available data. Starting from applications in reinforcement learning, Part I of the thesis first introduces a layer with injected noise to generally improve exploration for many RL problems in Chapter 2. Then Part I looks into a more specific setup where we can represent the environment in a graph format to enable structured exploration in Chapter 3. Finally, in Chapter 4 of Part I, we tackle another important class of problem, multi-agent RL, by incorporating agent-centric representation learning. Part II of the thesis moves on to computer vision. We begin with a novel technique to modulate image latent space in Chapter 5 and extend this technique to a conditional version in Chapter 6. Lastly, in Chapter 7, we propose a novel framework to learn video-level features representing the statics and dynamics aspects of given video sequences.

1.1 ARTIFICIAL INTELLIGENCE AND DEEP LEARNING

Charles Darwin once wrote: “There is no fundamental difference between man and the higher mammals in their mental faculties" and believed that the differences between human and animal intelligence are “of degree, not of kind". Indeed, human intelligence, including our mental feats to learn, to adapt, and to reason, plays an instrumental role in the thriving of humanity but, at the same time, holds potential limitations. Many scientific disciplines are motivated by our desire to understand better and exploit human intelligence, among which is the study of the intelligence manifested by machines, that is, *artificial intelligence* (AI). By comprehending and borrowing insight from human intelligence, the ultimate practical goal of AI is to develop tools and algorithms that help further advance human society.

A rapidly growing branch of AI is *machine learning*, which builds and improves mathematical models to solve problems through past experiences instead of explicit rule-based programming. The experiences used to construct the mathematical models are termed training data. Most commonly, we extract *features* from the raw data and then operate these models on top of the extracted features instead of the raw data. This process of extracting features is *representation learning*. The advancement in representation learning significantly contributes to essential progress in machine learning.

In the recent decade, an immensely successful approach in machine learning is *deep learning*. The “deep” part refers to using multiple layers of non-linear transformation to extract features from the raw data. The resulting features are referred to as *deep representations* and the class of mathematical models as *deep neural networks* (DNNs). Although earlier attempts, such as deep Boltzmann machines (Salakhutdinov and Hinton, 2009), have performed layer-by-layer pre-training, most recent breakthroughs (Krizhevsky et al., 2012) directly train DNNs to optimize a desirable objective via gradient descents in an end-to-end manner.

Deep learning has achieved success in solving many tasks in various domains, to name a few, image classification in computer vision (Krizhevsky et al., 2012), playing GO in reinforcement learning (Silver et al., 2016), and machine translation in natural language processing (Vaswani et al., 2017). A core ingredient partially contributing to each success is to leverage the *inductive bias* (Mitchell, 1980) well inherent to the task by, for example, constructing suitable model architectures and optimization objectives. However, each task can call for deep representations that incorporate its inductive bias in its particular way. Many challenges in deep learning research and applications lie in designing the optimal model to produce such representations.

In this thesis, we delve into how to effectively structure deep representation learning frameworks for a diversity of tasks in the domains of reinforcement learning and computer vision. These tasks are highly relevant to many real-life scenarios but, at the same time, challenging and less explored in their specific setups. The remainder of this section introduces further background information on the application of deep learning in reinforcement learning and computer vision.

1.1.1 Deep Reinforcement Learning

Reinforcement learning is an important field in AI. It aims at enabling autonomous agents to optimally interact with their environments, i.e. maximizing an accumulative reward. Examples range from game playing (Mnih et al., 2013) to robotics manipulation (Gu et al., 2017). This process is formally described as a Markov Decision Process (MDP) (Sutton and Barto, 1998a), consisting of

- states S, which can be finite, infinite or even continuous,
- actions A, which can be discrete or continuous,
- a transition probability $P(s_{t+1}|a_t, s_t)$ describing the dynamics of the interaction,
- and a reward function $r(s_t, a_t, s_{t+1}) = \mathbb{E}[R_{t+1}|s_t, a_t, s_{t+1}]$.

RL algorithms can generally be categorized into model-based or model-free (Sutton and Barto, 1998a). Model-based algorithms utilize the underlying environment dynamics for planning. Model-free algorithms directly use a learned model to estimate the value of each action directly given current state or observation. In this thesis, we focus on model-free RL. RL algorithms are trained either on-policy or off-policy (Sutton and Barto, 1998a). On-policy is trained on top of trajectories generated by the target policy, whereas off-policy can be done on other trajectories.

The objectives used in RL can generally be categorized into value function approximation, policy function approximation, and a combination of both. Value function approximation targets at estimating either the state values, V (or V_t at time t) or the state-action values, Q (or Q_t at time t), mathematically

$$Q_t{=}Q(s_t, a_t) = \mathbb{E}_{s_{t+1:\infty}, a_{t+1:\infty}}\left[\Sigma_{i\geq 0}\gamma^i r_{t+i}\right], V_t = V(s_t) = \mathbb{E}_{a_t}\left[Q(s_t, a_t)|s_t\right], \tag{1.1}$$

where γ ($0 \leq \gamma \leq 1$) is the *discount factor*. Then based on the estimated values, one can plan to act optimally. Policy gradient methods (Sutton et al., 2000) are based on the ground of the policy gradient theorem. A commonly used algorithm is REINFORCE (Williams, 1992), a gradient-based policy learning approach. Intuitively REINFORCE encourages policy parameters to produce actions resulting in positive rewards. But the naive application of policy gradients faces a major issue of high variance. One variance reduction technique is to approximate Q instead of using rewards. In addition, an approximated V is often adopted as a baseline and subtracted from Q when forming the policy gradient to reduce variance further, referred to as the *advantage*. In this thesis, the foundation RL algorithm we consider is based on on-policy actor-critic methods, and more mathematical details are in Section 2.

Although RL has been a long-standing area of research, it was limited to low-dimensional simple problems due to the complexity from the state and action spaces, as well as the lack of high-level abstract features (Bellemare et al., 2012). The recent rise of deep learning has brought breakthroughs in RL across many tasks, from mastering the game of GO (Silver et al., 2016) to robot manipulation (Gu et al., 2017). DNNs can not only perform as powerful value and policy function approximators but also eases the training of RL objectives by extracting features more effectively. The first major success of deep reinforcement learning (DRL) is the deep Q-networks (DQN). DQN processes the observations using convolutional neural networks (Mnih et al., 2013). It then feeds the subsequent features into a value function approximator, also parametrized by a DNN. DQN outperforms humans on many Atari 2600 games. In a similar spirit, DRL-based advantage actor-critic methods use DNNs to represent both values and policies (Mnih et al., 2016a), which, as aforementioned, serves as our algorithmic backbone in this thesis.

1.1.2 Deep Learning in Computer Vision

Computer vision (CV) is a branch of AI that allows computers to understand the visual world. Although “seeing” and “understanding” images and videos are effortless to the human vision system, it poses tremendous challenges to build a similar computational system both due to a lack of thorough understanding of the biological perception system and the inherent complexity of the visual domain. Traditional CV frameworks usually combine hand-crafted features, such as SIFT (Lowe, 1999), and traditional ML techniques, such as support vector machines, to solve CV tasks. The hand-crafted feature engineering step requires many trials and errors as well as domain expertise. However, for more complicated tasks, such as large-scale image classification, even carefully tuned features by hand are not sufficiently effective.

DL frameworks have revolutionized CV, starting with the seminal AlexNet (Krizhevsky et al., 2012). Alex Net introduced the essential concept of end-to-end training, where, instead of manual engineering, DNNs are stacked on top of raw input images and explicitly optimized to output the highest scores to the correct class via a softmax function. Another crucial component in AlexNet is the utilization of convolutional layers. Convolutional layers exploit an important inductive bias of images, translation invariance, which we will revisit in Section 1.2. Combined with other essential architectural components (e.g., data augmentation, ReLU activation, max pooling, drop out, etc.), mini-batch stochastic gradient descent, and GPU acceleration, AlexNet achieved state-of-the-art results in ImageNet classification and outperforming the best traditional CV competitors at the time by a large margin. More crucially, it sets the course for a wave of follow-up works in applying DL to computer vision tasks, from object detection (Girshick et al., 2014), activity recognition (Simonyan and Zisserman, 2014), to image synthesis (Brock et al., 2018).

Problems in computer vision can be classified into supervised, semi-supervised, and unsupervised learning based on the amount of labeled data provided during training. Supervised learning trains on fully labeled datasets. That is, for each input, there is an associated answer for the models to evaluate its performance. On the other hand, unsupervised learning is not provided with any labels but relies on the intrinsic patterns of the training datasets. Lastly, semi-supervised learning is equipped with partially labeled data. In this thesis, we mostly focus on investigating unsupervised learning for CV applications on images and videos.

1.2 DEEP NEURAL NETWORK ARCHITECTURES

A central component of DL is DNN architecture. This thesis especially explores how to improve and maneuver architectural building blocks to better incorporate the inductive biases of the tasks in hand (Chapter 2, 4, 5, and 6). In this section, we introduce the most relevant DNN building blocks.

1.2.1 Fully Connected, Convolutional and Recurrent Neural Networks

The most generic DNNs—also considered in our thesis—is a composition of non-linear transformation functions, or *layers*. Each layer typically comes with learnable parameters θ. Together, a DNN is a complicated and powerful function approximator:

$$\mathrm{DNN}_{\boldsymbol{\theta}} = f^L \circ f^{L-1} \circ \cdots \circ f^2 \circ f^1. \tag{1.2}$$

A fully connected (FC) layer is mathematically defined as:

$$f(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}), \tag{1.3}$$

where $\mathbf{W}$ is the *weight matrix* and $\mathbf{b}$ the *bias vector*, both learnable. The non-linearity of the FC layer comes from the activation function σ, e.g. a tanh function. But most widely used is the rectified linear units (ReLU) defined as $\mathrm{ReLU}(x) = \max(0, x)$.

The seminal AlexNet belongs to the class of convolutional neural networks (CNNs) (LeCun et al., 1998), whose core building block is the *convolutional (conv) layers*. A conv layer uses the same operation as an FC layer on local patches, termed as a kernel, and then spatially convolves the kernel across the input. As a result, a conv layer is not only efficient in the number of parameters and fast to be computed on GPUs but also produces translation-invariant features.

Recurrent neural networks (RNNs) are another important class of neural networks that takes a sequence of inputs $\{\mathbf{x}_i\}$, where the input at the i-th step takes in both $\mathbf{x}_i$ and a hidden state $\mathbf{h}_{i-1}$ and outputs a new hidden state $\mathbf{h}_i$.

In other words, the hidden states carry over the information from previous outputs. There are a few variations of RNN layers. In our thesis, we mostly use the long-short-term-memory recurrent layer (Hochreiter and Schmidhuber, 1997).

We refer the readers to the *Deep Learning* book by Goodfellow et al. (2016) for full details on CNNs and RNNs,

1.2.2 Attention in Deep Neural Networks

The *attention* mechanism (Kahneman and Treisman, 1984) incorporates another important inductive bias of how human intelligence functions to DNNs, that is, how we pay attention to extract the most relevant information.

In computer vision, spatial attention is well exploited. One ubiquitous tool to express spatial attention is through spatial transformation networks (STN). It applies an affine transformation to the feature of an input image to deform it into another image spatially transformed from the input image (Jaderberg et al., 2015). It is composed of a localization network to estimate the affine transformation parameters, a grid generator, and a sampler to apply this transformation to the input features. We refer the readers to the STN paper by Jaderberg et al. (2015) for full details.

Recently, a more general and powerful self-attention module has enabled new state-of-the-art performances in many ML tasks, ranging from machine translation (Vaswani et al., 2017) to image generation (Brock et al., 2018). In a self-attention module, each input component calculates a query, key, and value vector. Then, based on the key and query, each component calculates a score indicating how much attention to pay across different components. Then the final feature for each component is a weighted sum of the values based on the scores. We refer the readers to the paper by Vaswani et al. (2017) for full details.

1.3 REPRESENTATION LEARNING OBJECTIVES

The training of DNNs is usually done through backpropagating error derivatives from one or multiple objective functions (Werbos, 1982) using stochastic gradient descent algorithms (Bottou, 1991) such as Adam (Kingma and Ba, 2014). The choice of objective functions is another central component of DL. This thesis also explores how to construct the most effective objective functions for the tasks in hand (Chapter 3, 5, 6 and 7).

An objective function can be part of supervised learning, unsupervised learning, or reinforcement learning. In supervised learning, the target labels are given, and the objective function aims at mapping the inputs to the labeled outputs, such as classification. In reinforcement learning, there are no explicit labels, but rather the rewards inspire the objectives. In unsupervised learning, we utilize properties that do not require explicit labeling as objectives to shape the representations better. The works in this thesis focus on improving unsupervised representation learning objectives. The rest of this section introduces the main unsupervised approaches used in this thesis to form objectives.

1.3.1 Maximum Likelihood Learning

One long-standing approach to unsupervised learning is the probability density estimation of the variables of interest. A popular solution to density estimation is *maximum likelihood estimation* (MLE). Given we observe $\mathbf{x} \sim p(\mathbf{x})$, MLE directly optimizes a set of parameters θ to a probability distribution that best approximates the true distribution

$$p_\theta(\mathbf{x}) \approx p(\mathbf{x}). \tag{1.4}$$

The parameterizations that we focus in this thesis are DNNs. Often, we want to leverage additional a priori knowledge of the data space structure and learn a conditional model

$$p_\theta(\mathbf{x}|\mathbf{y}) \approx p(\mathbf{x}|\mathbf{y}), \tag{1.5}$$

rather than the unconditional $p_\theta(\mathbf{x})$. For instance, given some sequential data, we can form an unsupervised objective to predict the future from the present, such as in Chapter 1.2, and learn $p_\theta(\mathbf{x}_{t+1}|\mathbf{x}_t)$.

1.3.2 Latent Variable Models and Variational Inference

A popular approach for maximum likelihood learning is to explicitly learn a latent representation by assuming the observed variables $\mathbf{x}$ are generated conditioning on some unobserved *latent variables* $\mathbf{z}$. The marginal distribution of $\mathbf{x}$ can be written as:

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z})\ d\mathbf{z} = \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})\, d\mathbf{z}\,, \tag{1.6}$$

where $p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})$ is termed *the generative model* and $p(\mathbf{z})$ the prior. However, the integral in equation 1.6 usually does not have a closed analytic form or an efficient way to perform estimation. Therefore, we borrow techniques from variational inference and maximize a variational lower bound (VLB) of $\log p_\theta(\mathbf{x})$ by introducing an approximated posterior $q_\phi(\mathbf{z}|\mathbf{x})$:

$$\begin{aligned} \log p_\theta(\mathbf{x}) &= \log \int \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})\, d\mathbf{z} \\ &\geq \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\, d\mathbf{z} \\ &= \mathbb{E}_{q_\phi}(\mathbf{z}|\mathbf{x})[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{\mathrm{KL}}[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})] \end{aligned} \tag{1.7}$$

The DL framework, where both q_ϕ and p_θ are parameterized by DNNs, to optimize the VLB is the variational autoencoder (VAE). The first term of the VLB, $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$, can be viewed as a reconstruction term. To make the training of VAEs end-to-end differentiable, we often use the *reparameterization trick* (Kingma and Welling, 2013) to perform a Monte Carlo approximation of the reconstruction term. The prior $p(z)$ can either be parameterized or predefined to a fixed distribution, such as the standard multivariant Gaussian. We will revisit VAE and its variants in Chapter 3, 5 and 6 and also refer the readers to the VAE paper by Kingma and Welling (2013) for more details.

1.3.3 Contrastive Learning

Another approach to learn from the structure of the data space in an unsupervised fashion is through *contrastive learning*. Given an input, it essentially contrasts data points related to the input against those unrelated. Contrastive loss dated back more than a decade, including noise contrastive estimation (NCE) (Gutmann and Hyvärinen, 2010; Mnih and Teh, 2012) and similarity metric learning (Chopra et al., 2005; Hadsell et al., 2006). A series of promising adaptations of the unsupervised contrastive loss to deep representation learning have recently been proposed, such as contrastive predictive coding (CPC) (Oord et al., 2018) and momentum contrastive coding (MoCo) (He et al., 2020; Chen et al., 2020b).

In Chapter 7, we utilize CPC objective in unsupervised learning over unlabeled video sequences to extract dynamics information. Mathematically, CPC learns an encoder $\mathbf{z}_i = f_\theta(\mathbf{x}_i)$ to encode observations parametrized by a DNN and another DNN $\hat{\mathbf{z}}_{i+t} = g_\theta(\mathbf{x}_i)$ to perform predictive coding of $\mathbf{z}_{i+t}$. Then we minimize the following CPC objective inspired by NCE, *infoNCE*:

$$\begin{aligned}\mathcal{L} &= -\sum_{i,t} \log p(f_\theta(\mathbf{x}_{i+k} | g_\theta(\hat{\mathbf{x}}_i), \{f_\theta(\hat{\mathbf{x}}_l\}) \\ &= -\sum_{i,t} \log \frac{\exp(\hat{\mathbf{z}}_{i+k}^T \mathbf{z}_{i+k})}{\exp(\hat{\mathbf{z}}_{i+k}^T \mathbf{z}_{i+k}) + \sum_l \exp(\hat{\mathbf{z}}_{i+k}^T \mathbf{z}_l)},\end{aligned} \tag{1.8}$$

where x_l are the collection of negative examples that are unrelated to x_i.

For more details on CPC, we refer the readers to the CPC paper by Oord et al. (2018).

1.4 SCOPE AND RESEARCH QUESTIONS

The thesis is formulated into two parts, based on the domains of applications. Part 1 discusses how to craft deep learning models for RL applications and Part 2 computer vision. The research questions and contributions of this thesis are summarized as follows:

Research Question 1: *How can we effectively inject noise in actor-critic methods in RL to improve exploration and uncertainty understanding?*

In recent works (Plappert et al., 2017; Fortunato et al., 2017), parametric noise has been applied to value iteration and policy methods to improve exploration behaviors. However, this approach has proven less effective for actor-critic methods. In Chapter 2 and Shang et al. (2019b), we propose an alternative way to effectively inject noise into actor-critic methods. Inspired by the local reparameterization trick, randomness is added at the level of activations that feed into both policy and value functions. By entertaining a combination of deterministic and stochastic units, the model can learn the desired noise level for downstream computations. In experiments, we show that this method is highly effective in modeling both irreducible uncertainty due to stochasticity of the environment, as well as reducible uncertainty due to insufficient experiences. We test a variety of models enhanced with stochastic activations and demonstrate their effectiveness to a range of Atari 2600 games and a continuous control problem.

Research Question 2: *Given a persistent environment and a suite of diverse tasks, how can we effectively learn a representation of the environment to enable structured exploration in downstream RL?*

Efficiently learning to solve a suite of diverse tasks in a complex, albeit persistent environment is a crucial challenge for reinforcement learning (RL) agents. To this end, in Chapter 3 and Shang et al. (2019a), we propose to decompose a complex environment using a task-agnostic *world graph*, an abstraction that accelerates learning by enabling agents to focus exploration on a subspace of the environment. The nodes of a world graph are important *waypoint* states, and edges represent feasible traversals between them. Our framework has two learning phases: 1) identifying world graph nodes and edges by training a binary recurrent VAE on trajectory data and 2) a hierarchical RL framework that leverages structural and connectivity knowledge from the learned world graph to bias exploration towards task-relevant waypoints and regions. We thoroughly evaluate our approach on a suite of challenging maze tasks and show that using world graphs significantly accelerates RL.

Research Question 3: *How can we incorporate general agent-centric representation learning to improve multi-agent RL?*

Object-centric representations have recently enabled significant progress in tackling relational reasoning tasks (Battaglia et al., 2018; Santoro et al., 2017). By building a strong object-centric inductive bias into neural architectures, recent efforts have improved generalization and data efficiency of machine learning algorithms for these problems. One problem class involving relational reasoning that remains under-explored is multi-agent reinforcement learning (MARL). Here, in Chapter 4 and Shang et al. (2020a) we investigate whether object-centric representations are also beneficial in the fully cooperative MARL setting. Specifically, we study two ways of incorporating an agent-centric inductive bias into our RL algorithm: 1, introducing an agent-centric attention module with explicit connections across agents 2. Adding an agent-centric unsupervised predictive objective (i.e., not using action labels), to be used as an auxiliary loss for MARL, or as the basis of a pre-training step. We evaluate these approaches on the Google Research Football environment as well as DeepMind Lab 2D. Empirically, agent-centric representation learning leads to the emergence of more complex cooperation strategies between agents as well as enhanced sample efficiency and generalization.

Research Question 4: *How can we improve the architectural design of VAEs for more complex image modeling?*

Despite recent successes in synthesizing faces and bedrooms, existing generative models struggle to capture more complex image types (unless using large-scale modeling and training (Brock et al., 2018), potentially due to the oversimplification of their latent space constructions. In Chapter 5 and Shang et al. (2018), to tackle this issue, building on VAEs, we integrate recurrent connections across *channels* to both inference and generation steps, allowing the high-level features to be captured in global-to-local, coarse-to-fine manners. Combined with adversarial loss, our channel-recurrent VAE-GAN (crVAE-GAN) outperforms VAE-GAN (Larsen et al., 2016) in generating a diverse spectrum of high-resolution images while maintaining the same level of computational efficacy. Our model produces interpretable and expressive latent representations to benefit downstream tasks such as image completion. Moreover, we propose two novel regularizations, namely the KL objective weighting scheme over time steps and mutual information maximization between transformed latent variables and the outputs, to enhance the training.

Research Question 5: *How can we effectively integrate attention mechanisms to achieve improved and more interpretable conditional image modeling?*

As an extension of Research Question 4, we further probe whether we can apply the structured latent space to attribute-conditioned image modeling, such that different attributes are embedded in different components of the latent space. In Chapter 6 and Shang and Sohn (2019), we address this question by studying an important real-world problem: attribute-conditioned face synthesis. Building on top of a conditional version of VAE-GAN, we augment the pathways connecting the latent space with channel-recurrent architecture, to provide not only improved generation qualities but also interpretable high-level features. In particular, to better achieve the latter, we further propose an attention mechanism over each attribute to indicate the specific latent subset responsible for its modulation. Thanks to the latent semantics formed via the channel-recurreny, we envision a tool that takes the desired attributes as inputs and then performs a 2-stage general-to-specific generation of diverse and realistic faces. Lastly, we incorporate the progressive-growth training scheme to the inference, generation, and discriminator networks of our models to facilitate higher resolution outputs. Evaluations are performed through both qualitative visual examination and quantitative metrics, namely inception scores, human preferences, and attribute classification accuracy.

Research Question 6: *How can we learn sequence-level representations for the dynamics and statics aspects of videos in an unsupervised fashion?*

Video frames present both strong temporal coherence and visual dynamics. In Chapter 7 and Shang et al. (2020b), we leverage the interplay between these two fundamental properties as an important inductive bias and propose a fully unsupervised framework. Specifically, we learn decomposed video representations using two central objectives: i) a spatial transformation objective that summarizes temporally coherent content by transforming it into individual video frames; ii) a latent prediction objective that extracts video-level dynamics into a representation capable of reasoning frame-level dynamics autoregressively. In experiments, our proposed framework demonstrates competitive performance over a diversity of applications, including i) video retrieval based on either temporally coherent or dynamical content, ii) pretraining for downstream tasks such as facial expression and human action recognition, and iii) interpretable activity classification under the linear protocol.

Part I

Crafting Deep Learning Models for Reinforcement Learning Applications

MOTIVATION AND SUMMARY

The ultimate goal of reinforcement learning is how to optimize sequential decision makings, which is at the core to many real-world problems: from investments in the stock market, at-home robot assistance, to autonomous drone exploration on in the inhabitable area. However, RL is known to be tricky to train due to its noisey learning signals and the exploration-exploitation tradeoff. Therefore, crafting DL models to either directly alleviate these challenges is essential to the success of RL.

In Part I of the thesis, we examine several RL setups and investigate means to construct DRL frameworks to effectively and efficiently conduct learning to solve a diversity of tasks.

In Chapter 2, we explore a general, effective and flexible technique to improve exploratory behavior and to better reflect uncertainties in the context of actor-critic methods, termed ***stochastic actor-critic methods*** (Shang et al., 2019b).

In Chapter 3, we look at an RL setup that represents many real-world scenarios, such as an at-home robot doing various chores under the same household: given a persistent albeit complex environment, how we can efficiently learn to solve a diversity of tasks. We propose ***world graph decompositions over the environments to accelerate reinforcement learning*** (Shang et al., 2019a). The nodes of a world graph are important *waypoint* states, and edges represent feasible traversals between them. After identifying the world graph, our framework then applies it to a hierarchical RL algorithm to bias exploration towards task-relevant waypoints and regions.

In Chapter 4, we turn to another highly relevant RL problem class: fully cooperative multi-agent RL. We propose ***to incorporate agent-centric representations for multi-agent reinforcement learning*** in two ways. First, we introduce an agent-centric attention module with explicit connections across agents. Second, we leverage an agent-centric unsupervised predictive objective, to be used as an auxiliary loss or as the basis of a pre-training step.

2

STOCHASTIC ACTIVATION ACTOR CRITIC METHODS

2.1 INTRODUCTION

Deep reinforcement learning (DRL)—that is, using deep neural networks (DNNs) in reinforcement learning—has allowed tremendous progress in areas from game playing (Mnih et al., 2015) to continuous control (Lillicrap et al., 2016). These DNNs generally serve to approximate value functions (Sutton and Barto, 1998b), such as in deep Q-network (DQN) and its variants (Mnih et al., 2015), or to represent policies (Sutton and Barto, 1998b) such as in policy-gradient methods (Schulman et al., 2015). Another family of Deep RL (DRL) methods is the hybrid actor-critic approach, which employs DNNs to represent value functions as well as policies (Mnih et al., 2016a; Wang et al., 2016b) and has achieved state-of-the-art performances on highly complex RL problems.

Uncertainties play a crucial role in RL, including probabilistic state transitions, noisy reward functions, non-deterministic action outcomes (Gal and Ghahramani, 2016), and exploration of infrequently tested actions. Earlier DRL works addressing uncertainty have proposed the use of stochastic neural networks (SNNs). SNNs such as Bayesian Neural Networks (BNNs) and NoisyNets (Blundell et al., 2015; Fortunato et al., 2017; Plappert et al., 2017) improve exploration through injecting parametric noise. Nevertheless, parametric noise has not been equally successful in actor-critic methods ((Fortunato et al., 2017), (Plappert et al., 2017)), which are of particular interest because they have performed at a state-of-the-art level in many environments, including Atari games (Bellemare et al., 2013) and continuous robotics control (Wang et al., 2016b). Similar to other model-free approaches, DRL-based actor-critic methods are also highly sensitive concerning model architecture and other hyperparameter selections. It is therefore essential yet non-trivial to discover means to strengthen actor-critic methods with stochastic modeling components. We propose to directly sample intermediate latent representations shared by both the policy and value network to propagate more complex, structured perturbations, contrasting parametric noise where the weights for the two networks are jittered independently[1]. Particularly, we contribute to the development of a family of stochastic activation A3C models that effectively incorporate stochastic activations on top of the LSTM-Asynchronous Advantage Actor-Critic (Mnih et al., 2016a, A3C-LSTM), a framework representing the current state-of-the-art in many RL tasks.

An important subsequent contribution of this work is a thorough investigation of the empirical performance of stochastic activation A3C on 7 Atari 2600 benchmark games with stochastic skip frames, where our models generally outperform both the SOTA baseline A3C-LSTM and its NoisyNet variant with stochastic weights. Further examination over these experiments demonstrates the decrease of variance over approximated values from multiple samples of stochastic activations during training, indicating a reduction of model uncertainty. An empirical analysis of the converged value and policy networks also shows our proposed models reflecting the environment's intrinsic stochasticity. We then provide a mathematical link between stochastic activations and a special case of stochastic weights yet highlight their essential practical discrepancies. As an additional contribution, we advance beyond the on-policy A3C-LSTM and incorporate stochastic activations to methods with experience replay and continuous action spaces, namely deep deterministic policy gradients (DDPG) (Lillicrap et al., 2016) and actor-critic with experience replay (ACER) (Wang et al., 2016b). Pseudocode and full experimental details are in the Appendix; code and video demos are in the Supplementary Materials.

The rest of this paper is organized as follows: first, we discuss related works and preliminaries. Then, we motivate and introduce stochastic activation A3C and our primary model, Stochastic A3C (SA3C), along with two important variants to underline the flexibility of this technique. Next, we present our experimental setup and results, evaluating stochastic units' overall performance against baselines and stochastic weights. Finally, we analyze and interpret

1 This chapter is based on our ECML 2019 paper (Shang et al., 2019b).

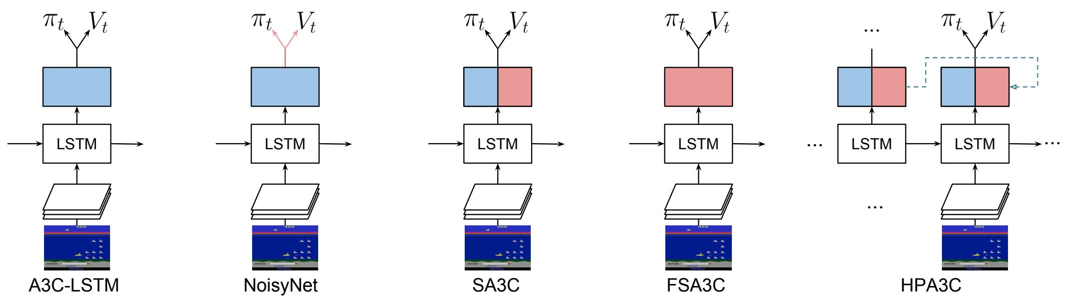


Figure 2.1: The baseline **A3C-LSTM** has deterministic latent units and weights only (blue). NoisyNet has stochastic weights (red) over policy and value networks independently. Our proposed methods have stochastic units shared by the two pathways with different configurations. **SA3C** has half deterministic and half stochastic units in the intermediate layer; **FSA3C** has only stochastic units. **HPA3C** is the same as SA3C but regularized with hierarchical prior from preceding time step during training.

these results and provide practical advice such as model and hyperparameter selections along with algorithm limitations.

2.2 RELATED WORK

The treatment of uncertainty has been a long-standing challenge in RL, and several lines of research have studied how to address this challenge. Our work is most connected to two general directions, incorporating stochastic components during (1) exploration and (2) inference process.

Epistemic uncertainty, i.e., model uncertainty, reduces as the agents gather more information via exploration. Many exploration mechanisms employ randomized actions instead of always using the best current model (exploitation) to gather more information. These mechanisms include Bayesian methods such as Thompson sampling (Ghavamzadeh et al., 2015), action-dithering schemes such as ϵ-greedy (Sutton and Barto, 1998b), value randomization such as randomized least-squares value iteration (RLSVI) or with Gaussian Processes (Osband et al., 2016; Kuss and Rasmussen, 2004), et cetera. Many of these mechanisms have also been adapted to the context of DRL, such as Thompson sampling via BNNs (Blundell et al., 2015; Wang and Zhou, 2019) and deep value randomization (Osband et al., 2017; Touati et al., 2020).

One approach developed in the related field of stochastic optimal control (SOC) uses inference techniques for finding the best actions under uncertain dynamics. The return (or related RL objectives) is defined as a factor in a graphical model, and probabilistic inference is applied to determine a sequence of actions optimizing this objective (Todorov, 2008). These probabilistic frameworks have inspired DRL algorithms, such as distributive DQN (Bellemare et al., 2017) and deep probabilistic inference for learning control (PILCO) (Gal et al., 2016). Recently, DRL models built upon the maximum entropy framework (Ziebart et al., 2008) by augmenting the standard RL objective with an entropy maximization term to achieve probabilistic inference have gained much attention, thanks to the potential of improving exploration and generalization in the face of uncertainty (Haarnoja et al., 2018b). These works also shed light on our proposed framework in retaining a distributive perspective over values and allowing stochastic policies.

The partially-observable setting explicitly addresses uncertainty about the state that the agent is in. A common strategy compresses the unbounded history of observations into belief states and then applies RL to the belief states (Murphy, 2000). Analytical belief state updates require knowledge of the observation model and transition model – even then is exponential in the number of state variables (Murphy, 2000). DRL-based algorithms that incorporate recurrent modules (Mnih et al., 2016a) implicitly maintain analogous internal states. However, these internal states are usually deterministic; in contrast, our model samples its internal states from Gaussian distributions, more similar to belief state approximations in continuous state systems (Prentice and Roy, 2007).

Our proposed technique fills an important gap in DRL-based actor-critic methods such as A3C-LSTM, where there has been lacking a general yet effective way to include stochastic elements. We apply high-level insights from Bayesian deep learning (Kingma et al., 2015), in particular the use of SNNs, to RL. Applications of SNNs in ML have a long history, (Neal, 1990) (Tang and Salakhutdinov, 2013) to list a few. In the regime of RL, SNNs have also shown promising results. For instance, recently, (Fortunato et al., 2017) and (Plappert et al., 2017) concurrently proposed to add independent parametric noises to the FC layers for better exploration, resembling BNNs but without the convergence to a posterior. In contrast, our model perturbs (part of) the intermediate activations, which are eventually shared by the actor and critic, allowing structured exploration via better-correlated randomness on both paths. Similar SNNs have been employed in several hierarchical RL systems to embed complex skills in an abstract form for higher-level tasks (Florensa et al., 2017). (Pritzel et al., 2017) leverages a special case, the variational autoencoder Kingma and Welling, 2013, to extract latent representations from raw observations for measuring similarities between states. In these works, the SNNs are separately trained. We directly alter a part of the deterministic units within the baseline models to become stochastic and train the model end-to-end. Finally, recent works propose to measure model uncertainty using DNNs with a special type of stochastic unit, dropout unit, in the context of, e.g. better safety (Gal and Ghahramani, 2016; Kahn et al., 2016).

2.3 PRELIMINARIES

We consider the standard discrete time step, discounted RL setup. An agent at time t observes o_t, which is a function of its state s_t, and chooses an action a_t guided by a policy π_t. Its ultimate objective is to maximize the accumulative expected return over time $R = \mathbb{E}_{(s_t, o_t, a_t) \sim \pi_t}[r_t]$, where r_t is the reward at time t. This section focuses on introducing the primary baseline algorithm used in our work, batch A3C-LSTM. To demonstrate the generalizability of our proposed method, we perform additional experiments using actor-critic methods with off-policy replays, and the descriptions of these models are introduced later in Section 2.4.2.

Asynchronous advantage actor-critic (A3C) (Mnih et al., 2016a) is a model-free, on-policy RL algorithm. Multiple agents are spawned to concurrently interact with the environments with different random seeds and optimize a shared model that approximates both the policy and value functions through asynchronous gradient descent to trade-off bias and variance. A3C models can either be composed of only Convolutional Neural Networks (CNNs) or with an additional recurrent module, usually an LSTM cell. We choose the latter, for it can learn more complex state representations to tackle, e.g. partially observable environments with longer time dependencies. Recently, batch A3C-CNN was developed for faster training and efficient utilization of GPUs (Adamski et al., 2016). We also take advantage of mini-batch training on A3C-LSTM for better stability and apply synchronous descents (Adamski et al., 2016), where backpropagation waits for all agents to finish their actions to avoid stale gradients (Chen et al., 2016a). Some also refer to similar algorithms as A2C (Wang et al., 2016a).

A3C-LSTM (Figure 2.1) consists of a CNN to extract features from raw observations, an LSTM cell to compress history, and a value and policy networks. We denote the features extracted by the LSTM as $h_t = f_{\text{LSTM}}(\text{CNN}(o_t), h_{t-1})$. In order to be consistent with models introduced later on, we further add two sets of Fully-Connected (FC) layers on top of h_t, obtaining their concatenation $[f_{\text{FC1}}(h_t), f_{\text{FC2}}(h_t)]$ as inputs to the value and policy networks. This structure allows us to later make either or both of these pathways stochastic. The objective for the value network is to estimate the state value V_t by regressing the estimated $t_{\max}$-step discounted returns with discount rate $\gamma \in (0, 1)$ (Equation 2.1); the policy network proposes a policy π_t and is guided by advantage-based policy gradients using the generalized advantage estimation $\hat{A}$ (details see (Schulman et al., 2016)), regularized by an entropy term to encourage exploration (Equation 2.2).

$$\text{Value estimation objectvie: } \mathcal{L}_{V_t} = \mathbb{E}_{s_t, o_t, a_t}(\Sigma_{t'=t}^{t_{\max}} \gamma^{t'-t} r_{t'} - V_t)^2, \qquad (2.1)$$

$$\text{Policy gradient with entropy regularization: } \nabla_\theta \log \pi_t(\hat{A}_t) + \beta \nabla H(\pi_t). \qquad (2.2)$$

Finally, we also compare our proposed method with a sthochastic weight variant of A3C-LSTM, NoisyNet A3C (NN-A3C, Figure 2.1). The construction mostly follows (Fortunato et al., 2017) and more details are illustrated in the Appendix.

Our architecture and training protocol produce a state-of-the-art level A3C-LSTM, which is an essential component in our work since we aim at developing a technique that is highly competitive, even surpassing the performance of a very powerful baseline. We compare the baseline A3C implementation replicated by us with another mainstream version as well as human players in Table 2.1.

2.4 ACTOR CRITIC METHODS WITH STOCHASTIC ACTIVATION

This section first illustrates how to integrate stochastic activations into A3C-LSTM, arriving at a family of models named stochastic activation A3C. We

Algorithm 1 SA3C

```
Initialize network parameters $\theta$
Fix variance $\sigma^2$
for $k = 0, 1, 2, \cdots$ do
    Clear gradients $d\theta \leftarrow 0$
    Simulate under current policy $\pi_{t-1}$ until $t_{\max}$ steps are obtained, where, $h_t = f_{\text{LSTM}}(\text{CNN}(o_t), h_{t-1})$, $\mu_t = f_{\text{mean}}(h_t), k_t = f_{\text{d}}(h_t)$, $z_t \sim \mathcal{N}(\mu_t, \sigma_t^2 = \sigma^2)$, $V_t = f_v(z_t, k_t)$, $\pi_t = f_p(z_t, k_t)$, $t = 1, \cdots t_{\max}$
    $R = \begin{cases} 0, \text{ if terminal} \\ V_{t_{\max}+1}, \text{otherwise} \end{cases}$
    for $t = t_{\max}, \cdots 1$ do
        $R \leftarrow r_t + \gamma R$
        $A_t \leftarrow R - V_t$
        Accumulate gradients from value loss: $d\theta \leftarrow d\theta + \lambda \frac{\partial A_t^2}{\partial \theta}$
        $\delta_t \leftarrow r_t + \gamma V_{t+1} - V_t$
        $\hat{A}_t \leftarrow \gamma \tau \hat{A}_{t-1} + \delta_i$
        Accumulate policy gradients with entropy regularization: $d\theta \leftarrow d\theta + \nabla \log \pi_t(a_t) \hat{A}_t + \beta \nabla H(\pi_t)$
    end
end
```

then describe in details how to construct the primary stochastic activation A3C used in our work, along with two additional variants. Finally, we extend the technique of stochastic units to actor-critic methods with off-policy training, namely DDPG and ACER.

2.4.1 Stochastic Activation A3C

Inspired by the SNN design from (Tang and Salakhutdinov, 2013) whose intermediate units are half deterministic and half stochastic in order to encode information of different uncertainty levels, we craft an initial version of stochastic activation A3C in a similar manner, termed stochastic A3C (SA3C). Following the output of the LSTM hidden state h_t, the next layer is split into a deterministic channel and a stochastic channel. The deterministic channel $k_t{=}f_{\text{det}}(h_t)$ is parameterized by a FC layer. The stochastic units follow factored Gaussian distributions. The variance is, for now, set to a fixed value and treated as a hyperparameter, but note that subsequent layers can learn to rely on the deterministic or the stochastic units in any proportion to manage the amount of noise in the value and policy functions. The mean $\mu_t{=}f_{\text{mean}}(h_t)$ is also parameterized by a FC layer. The pseudocode for SA3C is in Algorithm 1.
Fully-Stochastic A3C (FSA3C) is an interesting control setup that replaces the deterministic channel with a stochastic one and attains a fully-stochastic intermediate representation. Hierarchical Prior stochastic activation A3C (HPA3C)

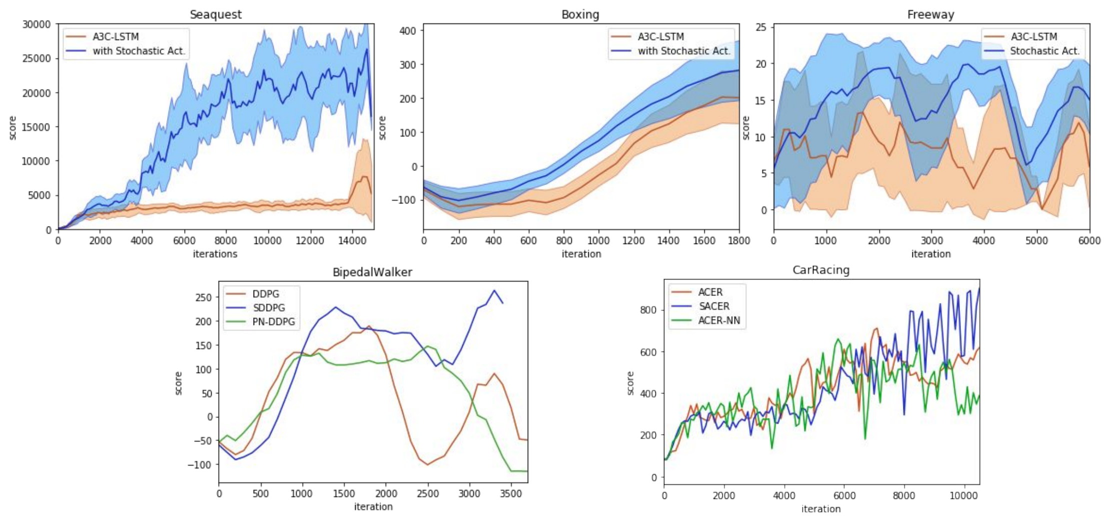


Figure 2.2: Training curves over 3 runs (median); vertical: rewards; horizontal: iterations. For Atari games, we plot the curves for the baseline and the best stochastic activation model along with the interquartile range. For the rest, we compare among the baseline, SA3C and then stochastic weights.

is inspired by BNNs that craft their priors to the model parameters in order to achieve certain effects, such as inducing sparsity (Louizos et al., 2017a). Analogously, HPA3C adds a KL-divergence between the stochastic activation distribution and a prior to the objective function. Specifically, the prior for the variance is fixed to a value σ^2, treated as a hyperparameter, and the prior for μ_t is derived from the previous step stochastic latent sample.[2] Our design is also similar in spirit to latent forward modeling (Tian and Gong, 2017) where the history predicts and guides the future, but in a more implicit form of prior regularization:

$$\text{Derivation of } \mu_t^p : z_{t-1} \sim \mathcal{N}(\mu_{t-1}, \sigma^2), \mu_t^p = f^p(z_{t-1}),$$

$$\text{Prior regularization: KL}\left[\mathcal{N}(\mu_t, \sigma_t^2) \middle\| \mathcal{N}(\mu_t^p, \sigma^2\right] = \log \frac{\sigma}{\sigma_t} + \frac{\sigma_t^2 + (\mu_t - \mu_t^p)^2}{2(\sigma)^2} - \frac{1}{2}.$$

We found that a proper prior choice is critical — omitting either the prior on the mean or the variance significantly deteriorates the model performance.The pseudocode for HPA3C is in the Appendix.

Forward propagation through stochastic activation A3C is identical to A3C-LSTM, except that the stochastic activations z_t are sampled from $\mathcal{N}(\mu_t, \sigma_t)$ and then concatenated with the deterministic counterpart k_t as the inputs for the

2 All operations are element-wise because of the factored Gaussian assumption.

Game	Human*	A3C†	NN-A3C†
Seaquest	28010	1744±0	943±41
BeamRider	5775	9214±608	11237±1582
MsPacman	15693	2436±249	3401±761
Boxing	4.3	91±1	100±0
Breakout	31	496±56	347±27
Qbert	13455	18586±574	17896±1522
Freeway	29.6	0±0	18±13

Game	A3C	NN-A3C	Stochastic Act.
Seaquest	13922±4920	894±313	29656±5317
BeamRider	9214±608	6117±1808	13779±3605
MsPacman	4670±1864	4096±1351	5590±1521
Boxing	99.5±1.0	94±4.4	100±0.0
Breakout	588±180	570±252	621±194
Qbert	15333±2462	14352±1335	16045±556
Freeway	23.3±1.2	22.4±0.8	23.9±1.3

Table 2.1: Results with * and † are cited from (Mnih et al., 2015) and (Fortunato et al., 2017). Due to stochastic frame skipping in our setups which generally yield more difficult environments, our results (last 3 columns) are not precisely comparable to †. Nonetheless we can still clearly conclude the competitiveness of our baseline implementation. The last column presents the results from the optimal stochastic activation models.

policy and value networks. Backpropagation via the stochastic units is done by the reparametrization trick (Kingma and Welling, 2013).

Lastly, it is worth noting that while our models employ Gaussian units thanks to their flexibility and ease to train, the proposed framework can adopt other stochastic units as well. We conduct preliminary experiments with dropout stochastic units in the Appendix and leave further investigation along this direction to future works.

2.4.2 DDPG and ACER

Deep Deterministic Policy Gradients (DDPG) (Lillicrap et al., 2016) is an off-policy actor-critic method. It explores via injecting action space noise, commonly from the Ornstein-Uhlenbeck process. We equip DDPG with parametric noise (Plappert et al., 2017) (PG-DDPG) or stochastic activation (SDDPG). We do not incorporate an LSTM module to DDPG and its variants. Thus, the baseline algorithm follows exactly as in (Lillicrap et al., 2016) and its parametric noise version, PN-DDPG, exactly as in (Plappert et al., 2017) but without randomizing the convolutional layers. Unlike A3C-LSTM, DDPG keeps separate encoders for the actor and critic. We only use stochastic activations to the behavior actor network and not to off-policy training.

Actor Critic with Experience Replay (ACER) (Wang et al., 2016b) is a sample-efficient actor-critic algorithm with a hybrid of on/off-policy gradients. We compare amongst ACER and its variants with stochastic units or noisy layer. Augmenting ACER with stochastic activation (SACER) follows the same protocol as augmenting A3C-LSTM with stochastic activation and we also use stochastic activations for off-policy training. As an additional comparison, we construct a NoisyNet version of ACER, NN-ACER by similarly randomizing the value and policy networks as in NN-A3C. The pseudocode of our ACER and its stochastic variants are in the Appendix.

2.5 EXPERIMENTAL SETUP AND RESULTS

This section first introduces the environments used in our experiments. Extensive ablation studies are done on the Atari games. We then discuss the empirical advantages of stochastic activation A3C over its deterministic baseline and how its design flexibility can adapt well to a variety of environments and tasks. Finally, we present additional results generalizing SA3C to off-policy methods, namely DDPG and ACER, on BipedalWalker2D and CarRacing.

2.5.1 Environments

Our experiments are primarily done in an on-policy manner on seven selected classic Atari 2600 games contained in the Arcade Learning Environment (Bellemare et al., 2013) and interfaced via OpenAI Gym (Brockman et al., 2016) to cover a diverse range of tasks and exploration types (Bellemare et al., 2016). Full descriptions of these games are in the Appendix. To avoid memorization and impose more randomness, we use the stochastic frame-skipping: each action is repeated for a number, uniformly sampled between 2 and 4, of consecutive frames. Exploration type is categorized by the taxonomy from (Bellemare et al., 2016). The stochasticity of Atari games originates from multiple sources, including frame-skipping, partial observation of some environments,

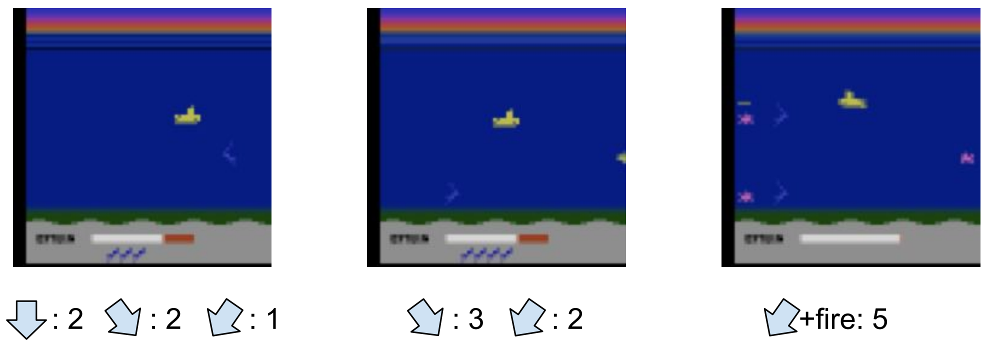


Figure 2.3: For SA3C, stochastic activations can result in stochastic policies. The less ambiguous the environment is, the more certain the policies become (left to right). Arrows indicate the direction of movement, followed by the number of times this action being selected (out of 5 samples).

the non-stationary policy during training, approximation errors, et cetera. For preprocessing, we crop Atari games to display only the game playing region, subtract estimated mean and divide standard deviation, and rescale to 80×80.

DDPG models are tested on a continuous task, BipedalWalker2D, where a robot needs to reach the end of a path within a time limit and positive reward is given for moving forward, totaling ≥ 300 for reaching the end, while a negative reward of -100 is given for falling. No preprocessing is done for this environment. ACER models are tested with CarRacing, a simple driving simulator whose observations consist of an RGB top-view of a race car and a black bar containing other driving information. We only receive the pixel-valued observations and also discretize its action space. More details on the preprocessing of CarRacing is provided in the Appendix.

2.5.2 Stochastic Activation A3C Results

HYPERPARAMETERS AND MODEL ARCHITECTURE For Atari, hyperparameters are tuned on Seaquest A3C-LSTM and then transferred to other games. We inherit all common hyperparameters from A3C-LSTM to stochastic activation A3Cs and only tune the additional ones, namely σ^2 for SA3C and FSA3C, HPA3C and the KL term weight for HPA3C. In particular, we would like to emphasize the coefficient for entropy regularization is tuned to perform optimally on the baseline–a higher value in fact deteriorates its performance; in other words, any performance gain via stochastic activations cannot be replaced by increasing the entropy term. For other environments, hyperparameters are tuned on the baseline and then transferred to stochastic weight/activation models. Most games share a common model architecture, but we use a slightly slimmer CNN for Boxing, Breakout and Freeway. Since HPA3C needs to learn σ_t, more variance is introduced to the gradients and the resulting stochastic activations require further normalization. After trial-and-error with several techniques such as Batch (Ioffe and Szegedy, 2015) and Layer (Ba et al., 2016) Normalization, we pick the most effective option–concatenated ReLU (Shang et al., 2016). The full details are in the Appendix.

EVALUATION PROTOCOL We report Atari results on A3C-LSTM, NN-A3C, SA3C and the best performing stochastic activation A3C variant in Table 2.2 following the protocol:

1. Train 3 independent runs—a standard DRL practice (Ostrovski et al., 2017; Wu et al., 2017).

2. For each run, validate the current model on a non-training random seed, select the best (validated) one after training.

3. Test the selected model for each run on 10 other random seeds not included in training or validation, obtaining $\mu_1 \pm \sigma_1$, $\mu_2 \pm \sigma_2$, $\mu_3 \pm \sigma_3$.

4. Report the best $\mu_i \pm \sigma_i$ under the column "Best".

5. Average across the 3 models, i.e. $\mu \pm \sigma$ over μ_1, μ_2, μ_3, and report under "Avg.".

The proceeding protocol not only showcases how good a policy the algorithm can attain if optimized well but also indicates variances in performance due to policy gradient training. We also plot the training curves composed of average validation scores with the standard deviation bars for Seaquest, Boxing and Freeway in Figure 2.2, other games in the Appendix.

INFERENCE AND STOCHASTIC POLICIES Based on the protocol from (Fortunato et al., 2017), NoisyNet is tested by setting the stochastic weights equal to the learned mean. For stochastic activation A3Cs, there are multiple possibilities during evaluation time. One is to only use the mean from the stochastic units, referred to as Maximum A Posteriori (MAP)–borrowing the Bayesian terminology. Alternatively, we sample the stochastic activations and vote the majority decision, leading to stochastic policies. Figure 2.3 shows the decisions out of 5 sampled policies for selected states: when there is no clear immediate goal, e.g. no enemy around, decisions tend to diverge, but otherwise they agree. Videos of Seaquest with deterministic polices from A3C-LSTM versus stochastic policies from SA3C are included in the Supplementary Materials. Table 2.3 compares different evaluation schemes from Seaquest — for stochastic policies we attempt 1, 5, and 50 samples — and 5 samples give the optimal results for most models. If not mentioned otherwise, all stochastic activation A3C results are obtained by voting among policies from 5 sampled activations.[3]

2.5.3 Actor-Critic Models with Experience Replay

We further integrate this technique to actor-critic methods with experience replay, namely ACER and DDPG. Complete hyperparameter details are in the Appendix. For DDPG, we plot the median training curves out of 3 independent runs in Figure 2.2. We found that DDPG is much less stable in training comparing with A3C-LSTM. Adding stochasticity to DDPG does not improve its training stability, which remains an open question. Nonetheless, SDDPG tends to converge significantly faster than DDPG and DDPG-PN, at iteration 4000, 5900, and 6300, respectively (Table 2.4). For ACER, we plot the median training curves out of 3 independent runs in 10K iterations in Figure 2.2. However, note that we stop the training once the environment is solved, i.e., average validation score over 10 random runs ≥ 900 . Out of the 3 runs (with maximum 10K iterations), only SACER manages to solve the environment. The median best scores attained by ACER is 891 and NN-ACER 859 (Table 2.5).

Stochastic activations boost the performances of the baselines and outperform parametric space noise. These results confirm the effectiveness of our proposed method when coupled with experience replay.

2.6 DISCUSSION

To further investigate the roles of stochastic activations, we first inspect the significance of a shared intermediate stochastic layer and then study the value

3 We use 1 sample for Freeway. As it only has 2 actions, voting would strongly diminish policy stochasticity.

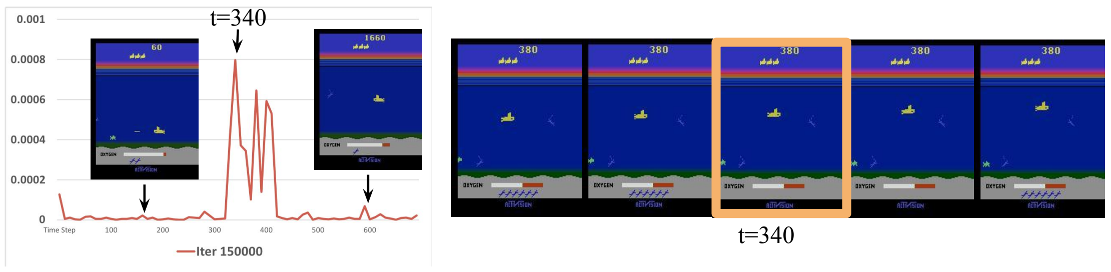


Figure 2.4: A zoom-in of the variance plot at convergence (iter 150K). The variance is generally low except for periods with high unpredictability. Around step 340, the submarine has reached divers' max capacity while the enemy is still emerging; the submarine chooses to surface over shooting the enemy to gain more points. More details see Section 2.6.2.

and policy learning process of SA3C. Our analyses support our conjecture that the stochastic units implicitly reflect the unpredictability of the environment. Lastly, we compare stochastic activations with stochastic weights, both theoretically and empirically. All of the experiments in this section are done on Seaquest.

2.6.1 Shared Intermediate Stochastic Layer

One feature of our proposed SA3C is to incorporate the stochasticity in the intermediate hidden layer shared by the value and policy networks[4]. As a result, the two paths adapt to a systematically propagated randomness instead of their own independent jittering as in (Fortunato et al., 2017; Plappert et al., 2017). Experimentally, we compare passing the perturbed mean (i.e., stochastic activations) to the value network only, to the policy network only, and to both as proposed. Injecting randomness to just the value network hamper the training (2438 ± 372), indicating the difficulty of capturing the value uncertainty without acting accordingly. The policy network only version leads to a comparable performance as the baseline (13410 ± 6281) but much worse than SA3C, meaning that the stochastic signals cannot be fully leveraged if not well coordinated with the critic.

4 For DDPG, since the encoding network is not shared, the stochastic activations are only for the policy.

2.6.2 Value Learning

Motivated by recent works using dropout units to estimate uncertainty (Gal and Ghahramani, 2016; Kahn et al., 2016), we obtain and analyze uncertainty estimations by calculating the sample variance of multiple approximated values over sampled stochastic activations. Concretely, for SA3C models at different training stages, we sample stochastic activations 5 times for each time step, calculate the variances of those resulting values and plot these in Figure 2.5 for the first 700 time steps, that is

$$\hat{\sigma_v}^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(f_V(k_t, z_t)_i - \bar{f_V}\right)^2, \bar{f_V} = \frac{1}{N}\sum_{i=1}^{N} f_V(k_t, z_t)_i, N=5, t=1\cdots700.$$

The variances tend to go down over training (Figure 2.5). This behavior is reminiscent of (Bayesian) models employing distributions over the weights, where these distributions reflect parametric uncertainty. Indeed, there is a connection between stochastic weights and activations that we will discuss more later. Despite the fixed variance of the noise in the stochastic units, the value network is clearly capable of gradually adapting the variance through learning. This can be achieved by shifting focus from stochastic units to deterministic units in downstream computations.

At convergence time, the value network usually approximates with little variance, reflecting low uncertainty, except for a surge period around step 340 (Figure 2.4). This corresponds to a special event where different actions can lead to varying amounts of rewards: the submarine has reached the maximum capacity of rescued divers, and it can either shoot the upcoming enemy to gain some points or surface to collect a large number of rewards by releasing the divers—eventually the submarine chooses the latter. The ambiguous situation can increase the variance compared to those with a clear target, compared to 340, 600, and 160. Therefore, we argue that SA3C's

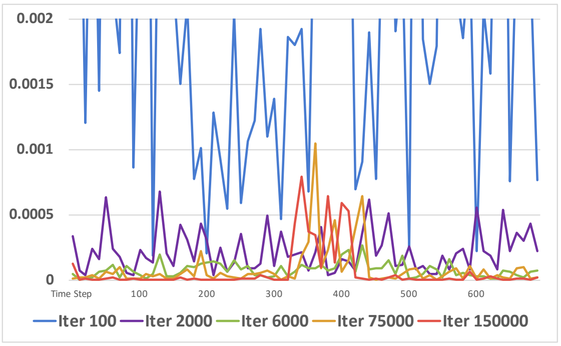


Figure 2.5: At different training iterations for SA3C, we sample 5 V_ts via stochastic activations for $t = 1..700$, plot their variances and observe the general trend of variances descends over training.

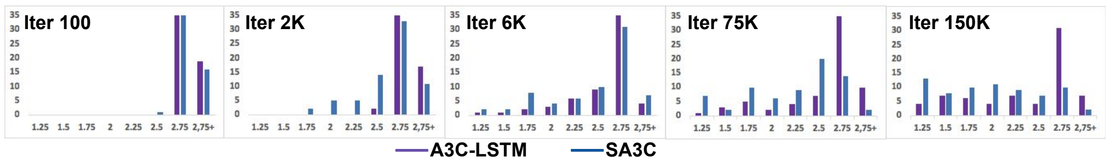


Figure 2.6: The entropy distributions of actions sampled via Boltzmann exploration for A3C-LSTM and via Boltzmann exploration plus stochastic activations for SA3C. The entropy of SA3C spreads evenly over time, allowing structured exploration, while A3C-LSTM stays around 2.75.

adaption to stochastic activations is not merely reducing the influence of the stochastic units, but rather carefully balancing between the deterministic and stochastic signals to implicitly establish a distributional perspective over returns and values, a beneficial trait for RL (Barth-Maron et al., 2018) and a proper reflection of the environment unpredictability.

2.6.3 Policy Learning

Next, we study the distributions of actions sampled during training. The default exploration for A3C-LSTM samples the next action from the distribution output by the policy network. In the case of stochastic activation A3Cs, on top of the default exploration, there is another sampling step over latent representations, i.e., to obtain stochastic activations. We are interested in how this additional modulation can make a difference. At different training iterations, we sample 50 actions from A3C-LSTM and SA3C for each time step following the above protocols, calculate the empirical entropy based on the sampled action spaces and plot the distributions in Figure 2.6 (over time step 1-700).

Initially (iteration 100), the sampled actions from both models are of high entropy. While training progresses, the entropy of SA3C starts to spread over a relatively even spectrum of values, whereas that of A3C-LSTM still concentrates around 2.75. The entropy regularization can explain the persistent high entropy for A3C-LSTM. Nonetheless, SA3C is given the same objective. Thus we reckon SA3C overcomes the entropy objective once the model has reached some level of parametric certainty. Then, based on how ambiguous the current environment is, SA3C modulates the amount of uncertainty communicated through the policy network distributions and leads to more structured action sampling for exploration compared to the closer-to-uniform action sampling in A3C-LSTM.

2.6.4 Comparison to Parametric Noise

Mathematically, stochastic activation SA3Cs can be linked to parametric noise with the aid of the local reparametrization trick (Kingma et al., 2015), which theoretically corroborates the reduction of model uncertainty. As an example, we derive the correspondence between SA3C and a special case of parametric noise where the hyperparameter-variance of the weights are fixed:

Proposition 1. *For an MLP layer, denote its input $X \in \mathbb{R}^n$ and output $Y \in \mathbb{R}^m$, if the posterior approximation of the weights to the MLP follows the distribution of a factored Gaussian $W \sim \mathcal{N}(U, V)$*[5] *with $U, V \in \mathbb{R}^{n \times m}$ and $v_{i,j} = \sigma^2$ to be a fixed value, then sampling the weights is equivalent to sampling its activation $Y \sim \mathcal{N}(U^T X, \sigma^2 \|X\|^2)$.*

Proof. With the chosen posterior, the activation of the FC layer with stochastic weights follows the distribution $y_i \sim \mathcal{N}(\sum_j^n u_{ij} x_j, \sum_j^n v_{ij} x_j^2)$, where y_i is the ith entry in Y. Since $v_{ij} = \sigma^2$ for any i, j, we conclude $Y \sim \mathcal{N}(U^T X, \|X\|_2^2 V)$. □

In practice, Atari experiments skip the scaling of V in proportion to $\|X\|_2^2$, as the pre-activation h_t's are generally of similar magnitude thanks to the LSTM module.

Despite the connection, in practice, SA3C differs from parametric noise from (Plappert et al., 2017) as it does not need adaptive variance adjustment to perform competitively; it also differs from NoisyNet, as it does not learn the variance—in fact, we empirically observe naive learning of the variance can hurt the training. But similarly to NoisyNet and in contrast to BNNs, approximated posteriors to the weights are absent to avoid complication over optimization. This practical trick has appeared in other algorithms derived from variational Bayes (Sohn et al., 2015). Meanwhile, the general framework of stochastic activation A3C still allows variance learning and the presence of a prior, as in the case of HPA3C. The setup of HPA3C, shown to be highly effective for several Atari games, is arduous to formulate in the context of stochastic weights, highlighting another important advantage of our proposed method—design flexibility.

There are other differences between stochastic weights and activations, which we will explain using NoisyNet. First, generally, NoisyNet does not surpass the

5 We abuse the notation here to represent factored Gaussian: U, V are both matrices of dim $n \times m$ and each entry of V represents the variance of the corresponding entry in U.

performance of deterministic A3C-LSTMs; we conjecture that although parametric noise properly facilitates optimization in value methods, it might not be suitable when the actor and critic are intertwined. In contrast, stochastic activations add randomness on top of representations simultaneously impacting both the actor and critic, henceforth yielding more structured perturbations than independently jittering the two networks' weights. Also, in (Fortunato et al., 2017), during training, stochastic weights for each agent are sampled per iteration (update). In contrast, we sample stochastic activation per time-step (game frame) and potentially induces a more complex change over policy and value learning. Although in practice, stochastic weights can also be sampled either way, empirical comparison (Table 2.5, both evaluated with MAP) confirms that NoisyNet is more compatible with per-iteration sampling and SA3C with per-time step. Consequently, the per-time step sampling can give rise to policy stochasticity based on the states' ambiguity level, as discussed in Section 2.5.2.

2.7 PRACTICAL ADVICE AND ALGORITHM LIMITATIONS

Stochastic activation is a general approach to improve A3C but not the panacea to every environment and task. Fully observable environments, such as Breakout, benefit less from stochastic activations. Environments with sparse rewards like Freeway also receive a more limited performance boost. RL problems with sparse rewards and/or more complex logic will require more specialized systems combined with stochastic units, such as curiosity-driven exploration.

The flexibility of stochastic activation A3C allows an effective application to a diversity of tasks, but the model selection can appear labor-demanding at first glance. From our experiences, SA3C is the go-to model as an initial attempt; if more aggressive exploration seems appropriate, FSA3C is a good candidate; if forecasting the upcoming states is essential in solving the task or rewards are sparse, HPA3C will likely perform better and more stable. One can thus always easily customize the stochastic activation A3C to meet the need of the task.

2.8 CONCLUSION

We proposed a flexible and straightforward technique to improve DRL-based actor-critic methods by adding stochastic activations. The proposed method outperforms existing state-of-the-art baselines on a variety of benchmarks. In future work, we hope to integrate the proposed technique with curiosity-driven exploration to address problems with sparse rewards and experiment with other types of stochastic units such as binary units for feature level count-based exploration (Ostrovski et al., 2017) or other intrinsic motivations (Aubret et al., 2019).

Game	A3C-LSTM		SA3C		NoisyNet	
	Best	Avg.	Best	Avg.	Best	Avg.
Seaquest	13922 ±4920	6785 ±5050	**28876** ±4270	23411 ±4783	849 ±313	1332 ±367
BeamRider	9214 ±608	8723 ±627	**9994** ±3717	8966 ±1013	6117 ±1808	5838 ±287
MsPacman	4670 ±1864	3973 ±543	**4960** ±1639	4743 ±220	4096 ±1351	3705 ±297
Boxing	**100** ±0.0	99.7 ±0.2	**100** ±0.0	99.9 ±0.0	94 ±4.4	11.6 ±59.6
Breakout	588 ±180	560 ±22	**621** ±194	556 ±45	570 ±252	551 ±25
Qbert	15333 ±2462	14732 ±482	**15560** ±184	15365 ±150	14352 ±1335	11231 ±3348
Freeway	**23.3** ±1.2	22.8 ±0.7	22.4 ±1.1	21.6 ±0.5	22.4 ±0.8	21.5 ±0.6

Game	Optimal Model		
		Best	Avg.
Seaquest	HPA3C	**29656**±5317	24992±3356
BeamRider	FSA3C	**13779** ±3605	10551 ±2341
MsPacman	FSA3C	**5590** ±1521	5382 ±268
Boxing	HPA3C	**100.0**±0.0	99.6±0.23
Breakout	HPA3C	596±197	569±22
Qbert	HPA3C	**16045** ±556	15365 ±150
Freeway	HPA3C	**23.9** ±1.3	23.2±0.5

Table 2.2: We report Atari results following the evaluation protocol in Sec 2.5. SA3C outperforms the baselines most of the time. The last column displays the results from the optimal stochastic activation variants for each game which can further boost the testing scores.

Seaquest	SA3C	FSA3C	HPA3C
MAP	27695±9096	4387±171	25474±8067
1	27081±6817	**5090**±1099	24475±4765
5	**28876**±4270	4453±592	**29656**±5317
50	28341±9839	4794±523	28341±9839

Table 2.3: Compare evaluating using μ only (MAP), using a single sampled stochastic activation and averging over 5 or 50 stochastic activation outcomes. Sampling and averaging 5 activations tend to be optimal.

	DDPG	SDDPG	PN-DDPG
BipedalWalker	5900	**3500**	6300
	ACER	SACER	NN-ACER
CarRacing	891	**900**+	859

Table 2.4: Compare various actor-critic baselines with their stochastic-activation/weight variants, tested on BipedalWalker and CarRacing. We report the median iteration of solving the environmen for BipedalWalker and the median best score in 10K iteration for CarRacing, where 300+ and 900+ are considered solved for each task respectively.

Seaquest	Per Iteration	Per Time Step
NoisyNet	894±313	536±359
SA3C	3796±26	27695±9096

Table 2.5: Comparing per iteration and time-step noise injection to NoisyNet and SA3C.

CHAPTER APPENDIX

2.A MORE ATARI RESULTS

We provide more experimental results for Atari games. Table 2.G.1 compares different evaluation schemes for stochastic activation A3Cs. Figure 2.G.2 compares the training curves among the stochastic activation A3C variants from the main text with the baseline A3C-LSTM.

2.B PSEUDOCODE

Pesudocode for baseline A3C-LSTM is Algorithm 2, for HPA3C is Algorithm 3, ACER is Algorithm 4, and SACER is Algorithm 5.

2.C ATARI 2600 ENVIRONMENT

See Table 2.C.1.

2.D CARRACING ENVIRONMENT

CarRacing is a simple driving environment from the Box2D module of OpenAI Gym. The observations consist of 96×96 RGB top-view of the race car and a black bar containing information regarding speed, ABS sensor outputs per each wheel, steering wheel position, and gyroscope. During training, we only receive pixel-valued observations. The observations are normalized by subtracting mean and dividing by standard deviation and then resized to 80×80. The beginning of the game shows a top view of the entire track and slowly

	Exploration Type	Remarks
Seaquest-v4	easy, score exploit	Players drive around a submarine to eliminate enemies and save divers.
Boxing-v4	easy, human optimal	Players hit the opponent in a boxing game. max score: 100
BeamRider-v4	easy, score exploit	Players pilot a fixed ship to clear enemy sectors in the outer space.
Breakout-v4	easy, human optimal	Players bounce a traveling ball to hit and eliminate bricks. max score: 864
MsPacman-v4	hard, dense reward	Players navigate Pacman through a 2D maze to eat pellets and avoid ghosts.
Qbert-v4	hard, dense reward	Players need to alter the cube color in a pyramid by making Qbert hop around while avoiding obstacles and enemies.
Freeway-v4	hard, sparse reward	Players control chickens to run across highway filled with traffic to reach the other side.

Table 2.C.1: Selected Atari Game Descriptions.

zooms in. As this period is irrelevant to our task, we set the zoom to a static value. The original action space consists of 3 continuous values, namely breaking (0 to 1), steering wheel left to right (-1 to 1), and acceleration (0 to 1). We discretize the continuous space to 4 general categories: braking, acceleration, turning right, and turning left. We further provide 3 levels of intensity in applying the action for each category, namely soft, medium, and full-throttle, in total yielding 12 discrete action options. The maximum total score possible, if without any negative rewards, is 1000. Every tile the car clears is given $1000/N$ points where N is the total number of tiles on the track, and every second used for the game is given -5 points. We also reduce the FPS from 50 to 12.5 and repeat the action 4 times, to simulate the effect of (deterministic) frame-skipping, while making the game computationally more efficient. The environment is considered to be solved when the agent consistently achieves more than 900 points, and the OpenAI oracle agent scores 837, averaging over 100 games. We test on 10 randomly generated games with different random seeds outside of the training seeds at each iteration.

2.E MORE ON ARCHITECTURE AND HYPERPARAMETERS

We document the complete details on model architecture and hyperparameters in this section.

MODEL ARCHITECTURE All models start with a CNN encoder. Most environments also use an LSTM module to incorporate longer time dependencies, except for DDPG where we find that the training is much more stable with a single CNN and stacked frames (4 frames). DDPG also uses separate encoders on the actor and critic's observations, whereas the other models share the same CNN encoder. The CNN or CNN-LSTM part of the model architecture is documented in Table 2.G.2.

The CNN encoders are a generic composition of convolutional layers and LeakyReLU nonlinearities. Two types of CNN encoders are used for Atari games: one with fewer parameters for simpler games (Breakout, Boxing, and Freeway) and the other with more parameters for the rest of the games. The LSTM module has 512 hidden units. DDPG uses the same CNN encoder as in (**a3c2018**) but without the LSTM module. ACER shares the same architecture as A3C-LSTM for Seaquest, except that the value network now outputs 12 (number of actions for CarRacing) state-action value estimations instead of a single state value estimation.

	Seaquest	BeamRider	MsPacman	Boxing	Breakout	Qbert	Freeway
MAP	24601±4614	9886±1936	3665±1942	99.7±0.9	427±19	14772±2257	21.7±2.3
1	18749±6386	7331±1135	3368±1540	99.5±1.2	410±30	13535±2886	5.1±1.1
5	26550±8240	7826±1519	3344±1788	100.0±0.0	495±152	14772±1033	10.2±1.0
50	24552±6911	8580±1824	4267±1962	99.9±0.3	448±159	15227±290	19.8±0.9

Table 2.E.1: Dropout stochastic units tested on 10 random seeded games with MAP, and voting from 1, 5, and 50 stochastic policies.

NORMALIZATION OF HPA3C. Since HPA3C needs to learn σ_t, more variance is introduced to the gradients and the resulting stochastic activations require further normalization. Because LeakyReLU allows the activations to concentrate below 0 in avoidance of the noisy gradients from variance learning, stochastic units' effects are diminished. After trial-and-error with techniques such as Batch (Ioffe and Szegedy, 2015) and Layer (Ba et al., 2016) Normalization, we pick the most effective option—concatenated ReLU (Shang et al., 2016). In theory, normalization techniques such as BatchNorm and LayerNorm can also adjust the output distribution, but they are not as effective in practice. We note that with the right normalization, HPA3C exhibits more stable and consistent training than A3C-LSTM and the other stochastic activation A3Cs.

NORMALIZATION OF DDPG In (Plappert et al., 2017), the authors achieve meaningful perturbation from a uniform spherical Gaussian noise distribution by normalizing the noisy layer's activations. We also discover that without proper normalization, the additional randomness from either weights or activations can catastrophically hamper optimization. We conjecture that the lack of tanh non-linear activation from LSTM can contribute to this issue. Therefore we apply layer normalization after the linear layers with stochastic weights or activations in DDPG.

HYPERPARAMETERS Our models are optimized with Adam (Kingma and Ba, 2014) through gradient descent using mini-batches of size 64, thus spawning 64 asynchronous agents—except for CarRacing, where we use 8 agents. For Atari, hyperparameters are tuned on Seaquest A3C-LSTM baseline and then transferred to other games, with only small adjustments if necessary. We inherit all common hyperparameters from A3C-LSTM to stochastic activation A3Cs and only tune the additional ones, namely the variance for SA3C and FSA3C, the variance prior and the KL term weight for HPA3C. For other environments, hyperparameters are tuned on the baseline algorithm then transferred to other related models, such as with parametric noise and stochastic activations.

For Adam optimizer, initial learning rate for all Atari games is 0.0005, $\epsilon = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$; gradients over 40 are clipped to 40. Discount rate γ for return is 0.95. The objective for value estimation is weighted with $\lambda = 5$ except for Freeway[6] where $\lambda = 0.5$. The entropy term is weighted with $\beta = 0.01$. The rewards are clipped between -1 and 1 except for Freeway, where the reward is not clipped. The maximum rollout for LSTM is 20 time steps except for Freeway's 30 steps. The trace decay parameter for the generalized advantage estimator is $\tau = 1.0$ except for Freeway $\tau = 0.92$. For the variance, we set $\log(\sigma^2) = -6$ for Seaquest, MsPacman and Breakout and $\log(\sigma^2) = -4$ for the rest. The variance prior of HPA3C is set in the same way, and the KL term is weighted with $\psi = 0.0001$. The maximum training iteration is $150K$,

6 Freeway is a sparse reward game hence its hyperparameters are additionally tuned.

but training is stopped early if model performance reaches a plateau or the score is maximized.

In our ACER experiments, we follow most of the hyperparameters used for Seaquest and train 10K iterations. Additional ACER hyperparameters include the replay ratio to be 1, replay buffer size 15000, and replay start when the buffer reaches 5000, trust decay for TRPO 0.99, trust threshold 1, model averaging ratio 0.99, and importance weight truncation 10.

In our DDPG experiments, we use batch size 128, initial learning rate 0.0005. For parametric noise, the initial noise standard deviation 0.05, the distance threshold is 0.3, and the adapt coefficient is 1.05. For stochastic activation, we set $\log(\sigma^2) = -5$.

2.F DROPOUT STOCHASTIC UNITS

Existing works (Gal and Ghahramani, 2016) have interpreted hidden units equipped with stochastic dropout, traditionally a means to prevent overfitting, from a Bayesian point of view. We test this alternative by replacing the Gaussian units in SA3C with dropout units of an optimal dropout rate 0.25. We test with no dropout (MAP) during the evaluation, a single forward pass with dropout, and averaging over 5 or 50 forward passes with dropout. Dropout can improve upon baseline A3C-LSTM in some of the games but does not outperform the best Gaussian stochastic activation model (Table 2.G.1). However, it is a worthwhile future direction to more closely investigate the link with, and potential benefits of, a formal Bayesian treatment of stochastic activations with the aid of dropout stochastic units.

2.G MORE ON NOISYNET A3C-LSTM

As there is no existing published implementation of NoisyNet to A3C-LSTM frameworks, we experimented with different configurations as shown in Figure 2.G.1. NoisyNet-v1 only randomize the value and policy networks (in red). As we have additional FC layers after LSTM before the policy and value networks, NoisyNet-v2 attempts to randomize those connecting FC layers only. Finally, NoisyNet-v3 randomize both connecting FC layers and the policy and

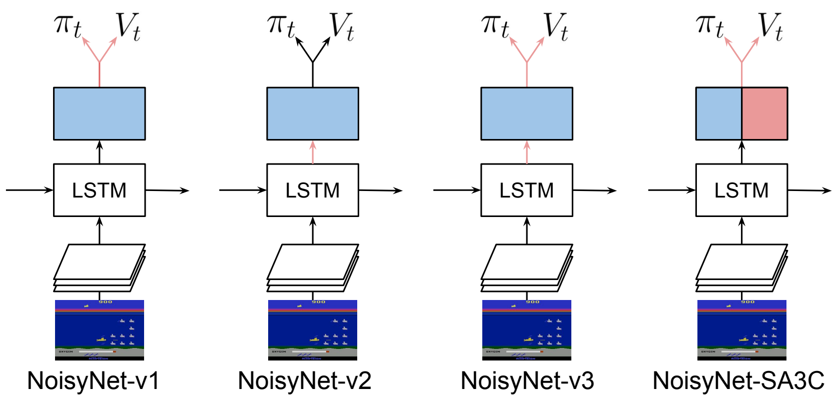


Figure 2.G.1: Different variations of NoisyNet that we test for A3C-LSTM as well as an integration with SA3C. Stochastic connections and activations are in red.

value networks, following the protocol of the A3C-NoisyNet in Fortunato et al., 2017.

For fairness, all models are evaluated using the MAP protocol. We test the 3 NoisyNet configurations on Seaquest. NoisyNet-v2 and v3 results in performance at the level of random moves, indicating that no meaningful training happened, and randomizing the connecting FC layer is detrimental to optimization. NoisyNet-v1 indeed learns up to some degree, but the performance (894 ± 313) is significantly worsened from the baseline A3C-LSTM model (13922 ± 4920); however, since it is the most promising configurations out of all possibilities, throughout the main text, we apply this architecture for any NoisyNet related experiment.

Lastly, we investigate how weight randomization affects stochastic activations and integrate NoisyNet to SA3C, forming NoisyNet-SA3C. NoisyNet-SA3C achieves 16037 ± 7438, better than NoisyNet-A3C-LSTM or A3C-LSTM, but substantially worse than SA3C (27695 ± 9096). Indeed, stochastic activations can help alleviate the difficulty of taming stochastic weights to some degree. Still, the two are not necessarily complementary, and stochastic weights can lower stochastic activations' performance.

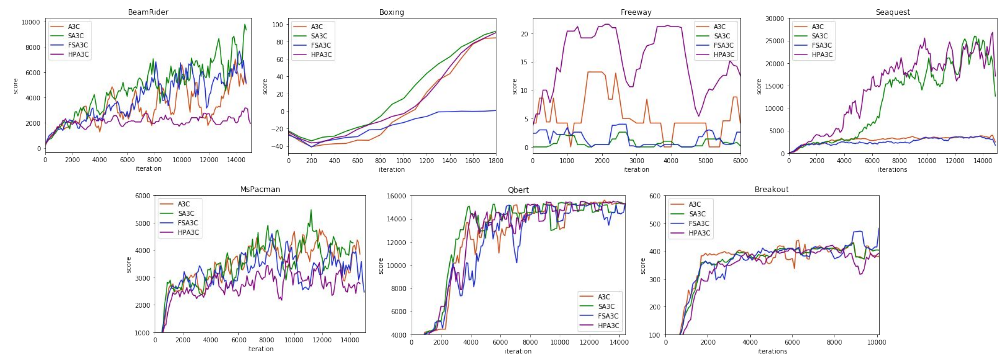


Figure 2.G.2: Compare different stochastic activation A3C with baseline. Training curves over 3 runs (median).

	Seaquest	BeamRider	MsPacman	Boxing	Breakout	Qbert	Freeway
SA3C							
MAP	27695±9096	9970±3428	4816±1569	99.4±1.2	497±139	15530±480	22.4±1.1
1	27081±6817	9648±2623	4819±1569	99.5±0.8	524±180	15330±146	22.4±1.1
5	28876±4270	9319±2125	4830±1595	99.2±1.6	493±135	15570±462	21.3±0.6
50	28341±9839	10540±3176	5096±1283	99.9±0.3	552±162	15530±480	21.2±0.6
FSA3C							
MAP	4287±171	11105±3402	5590±1521	99.4±1.0	409±14	14095±1762	21.2±2.3
1	5090±1099	10108±2858	4464±1638	98.7±2.0	422±27	14592±1943	24.2±0.6
5	4453±592	13779±3650	5304±1920	99.6±0.8	412±18	14017±1935	21.2±0.6
50	4794±523	9401±2207	3988±1697	99.4±1.8	423±89	14412±1652	21.3±0.6
HPA3C							
MAP	25474±8067	3349±2171	4515±2116	98.9±1.4	437±197	15302±2535	21.2±0.6
1	24475±4765	4338±3017	4287±1765	99.6±1.2	564±167	13160±147	23.9±1.3
5	29656±5317	4951±2171	4131±1345	100±0.0	596±197	12035±4306	21.3±0.6
50	28341±9839	5278±2234	3988±1697	99.2±1.6	378±192	12030±4923	21.2±0.6

Table 2.G.1: We compare testing on 10 random seeded games with MAP, and voting from 1, 5, and 50 stochastic policies.

Algorithm 2 A3C-LSTM

```
Initialize network parameters $\theta$ for $k = 0, 1, 2, \cdots$ do
    Clear gradients $d\theta \leftarrow 0$ Simulate under current policy $\pi_{t-1}$ until $t_{\max}$ steps are obtained, where, $h_t = f_{\text{LSTM}}(\text{CNN}(o_t), h_{t-1})$, $w_t = f_{\text{FC1}}(h_t), k_t = f_{\text{FC2}}(h_t)$, $V_t = f_v(w_t, k_t)$, $\pi_t = f_p(w_t, k_t)$, $t = 1, \cdots t_{\max}$ $R = \begin{cases} 0, \text{ if terminal} \\ V_{t_{\max}+1}, \text{otherwise} \end{cases}$ for $t = t_{\max}, \cdots 1$ do
        $R \leftarrow r_t + \gamma R$ $A_t \leftarrow R - V_t$ Accumulate gradients from value loss: $d\theta \leftarrow d\theta + \lambda \frac{\partial A_t^2}{\partial \theta}$ $\delta_t \leftarrow r_t + \gamma V_{t+1} - V_t$ $\hat{A}_t \leftarrow \gamma\tau\hat{A}_{t-1} + \delta_i$ Accumulate policy gradients with entropy regularization: $d\theta \leftarrow d\theta + \nabla \log \pi_t(a_t)\hat{A}_t + \beta\nabla H(\pi_t)$
    end
end
```

Algorithm 3 HPA3C

```
Initialize network parameters $\theta$
Fix variance $\sigma^2$
for $k = 0, 1, 2, \cdots$ do
    Clear gradients $d\theta \leftarrow 0$
    Simulate under current policy $\pi_{t-1}$ until $t_{\max}$ steps are obtained, where, $h_t = f_{\text{LSTM}}(\text{CNN}(o_t), h_{t-1})$, $\mu_t^p = f_{\text{mean}}^p(z_{t-1})$, $\mu_t = f_{\text{mean}}(h_t)$, $\sigma_t^2 = f_{\text{var}}(h_t)$, $k_t = f_{\text{d}}(h_t)$, $z_t \sim \mathcal{N}(\mu_t, \sigma_t^2)$, $V_t = f_v(z_t, k_t)$, $\pi_t = f_p(z_t, k_t)$, $t = 1, \cdots t_{\max}$
    $R = \begin{cases} 0, \text{ if terminal} \\ V_{t_{\max}+1}, \text{otherwise} \end{cases}$
    for $t = t_{\max}, \cdots 1$ do
        $R \leftarrow r_t + \gamma R$
        $A_t \leftarrow R - V_t$
        Accumulate gradients from value loss: $d\theta \leftarrow d\theta + \lambda \frac{\partial A_t^2}{\partial \theta}$
        $\delta_t \leftarrow r_t + \gamma V_{t+1} - V_t$
        $\hat{A}_t \leftarrow \gamma\tau \hat{A}_{t-1} + \delta_i$
        Accumulate policy gradients with entropy regularization: $d\theta \leftarrow d\theta + \nabla \log \pi_t(a_t)\hat{A}_t + \beta \nabla H(\pi_t) + \phi D_{KL}(\mathcal{N}(\mu_t, \sigma_t^2) || \mathcal{N}(\mu_t^p, (\sigma^p)^2))$
    end
end
```

Algorithm 4 ACER

```
Initialize network parameters $\theta$
Initialize average network parameters $\theta_a$
for $k = 0, 1, 2, \cdots$ do
    Clear gradients $d\theta \leftarrow 0$
    if on policy then
        Simulate under current policy $\pi_{t-1}$ until $t_{\max}$ steps are obtained, where,
        $h_t = f_{\text{LSTM}}(\text{CNN}(o_t), h_{t-1})$, $w_t = f_{\text{FC1}}(h_t), k_t = f_{\text{FC2}}(h_t)$, $Q_t = f_v(w_t, k_t)$,
        $\pi_t = f_p(w_t, k_t)$, $V_t = Q_t \cdot \pi_t$, $t = 1, \cdots t_{\max}$, set $\bar{\rho}_t = 1$
    else
        Retrieve experience $(o_{1 \cdots t_{\max}}, r_{1 \cdots t_{\max}}, a_{1 \cdots t_{\max}}, \mu_{1 \cdots t_{\max}})$ from the memory
        buffer, compute $h_i = f_{\text{LSTM}}(\text{CNN}(o_t), h_{t-1})$, $w_t = f_{\text{FC1}}(h_t), k_t = f_{\text{FC2}}(h_t)$,
        $Q_t = f_v(w_t, k_t)$, $\pi_t = f_p(w_t, k_t)$, $V_t = Q_t \cdot \pi_t$, $\bar{\rho}_t = \pi_t / \mu_t$,$t = 1, \cdots t_{\max}$.
    end
    $R = \begin{cases} 0, \text{ if terminal} \\ V_{t_{\max}+1}, \text{otherwise} \end{cases}$
    for $t = t_{\max}, \cdots 1$ do
        $R \leftarrow r_t + \gamma R$
        $A_t \leftarrow R - V_t$
        $\mathcal{L}_V \leftarrow \frac{1}{2}(R - Q_t(a_t))^2$
        Calculate advantage policy gradients: $g \leftarrow \nabla \log \pi(a_t) \hat{A}_t$
        Calculate KL gradients: $k \leftarrow \nabla D_{KL}(\pi_t^a || \pi_t)$
        Accumulate trust region policy gradients with entropy regularization: $d\theta \leftarrow d\theta + \nabla\theta(g - \max(0, \frac{k^T g - \delta}{||k||_2^2})k) + \beta \nabla H(\pi_t)$
        Accumulate gradients from value loss: $d\theta \leftarrow d\theta + \lambda \frac{\partial \mathcal{L}_V}{\partial \theta}$
        Update Retrace target: $R \leftarrow \rho_t(R - Q_t(a_t)) + V_t$
    end
    Update average model parameter: $\theta_a \leftarrow 0.99 * \theta_a + 0.01 * \theta$
end
```

Algorithm 5 SACER

```
Initialize network parameters $\theta$  Fix variance $\sigma^2$  Initialize average network parameters $\theta_a$  for $k = 0, 1, 2, \cdots$ do
    Clear gradients $d\theta \leftarrow 0$  if on policy then
        Simulate under current policy $\pi_{t-1}$ until $t_{\max}$ steps are obtained, where,
        $h_t = f_{\text{LSTM}}(\text{CNN}(o_t), h_{t-1})$, $\mu_t = f_{\text{mean}}(h_t), k_t = f_{\text{d}}(h_t)$, $z_t \sim \mathcal{N}(\mu_t, \sigma^2)$,
        $Q_t = f_v(z_t, k_t)$, $\pi_t = f_p(z_t, k_t)$, $V_t = Q_t \cdot \pi_t$, $t = 1, \cdots t_{\max}$
    else
        Retrieve experience $(o_{1\cdots t_{\max}}, r_{1\cdots t_{\max}}, a_{1\cdots t_{\max}}, \mu_{1\cdots t_{\max}})$ from the memory
        buffer, compute $h_t = f_{\text{LSTM}}(\text{CNN}(o_t), h_{t-1})$, $\mu_t = f_{\text{mean}}(h_t), k_t = f_{\text{d}}(h_t)$,
        $z_t \sim \mathcal{N}(\mu_t, \sigma^2)$, $Q_t = f_v(z_t, k_t)$, $\pi_t = f_p(z_t, k_t)$, $V_t = Q_t \cdot \pi_t$, $\bar{\rho}_t = \pi_t / \mu_t$, $t = 1, \cdots t_{\max}$.
    end
    $R = \begin{cases} 0, \text{ if terminal} \\ V_{t_{\max}+1}, \text{otherwise} \end{cases}$  for $t = t_{\max}, \cdots 1$ do
        $R \leftarrow r_t + \gamma R$  $A_t \leftarrow R - V_t$  $\mathcal{L}_V \leftarrow \frac{1}{2}(R - Q_t(a_t))^2$  Calculate advantage policy gradients: $g \leftarrow \nabla \log \pi(a_t) \hat{A}_t$  Calculate KL gradients:
        $k \leftarrow \nabla D_{KL}(\pi_t^a || \pi_t)$  Accumulate trust region policy gradients with
        entropy regularization: $d\theta \leftarrow d\theta + \nabla\theta(g - \max(0, \frac{k^T g - \delta}{||k||_2^2})k) + \beta \nabla H(\pi_t)$
        Accumulate gradients from value loss: $d\theta \leftarrow d\theta + \lambda \frac{\partial \mathcal{L}_V}{\partial \theta}$  Update Retrace
        target: $R \leftarrow \rho_t(R - Q_t(a_t)) + V_t$
    end
    Update average model parameter: $\theta_a \leftarrow 0.99 * \theta_a + 0.01 * \theta$
end
```

layer	input	output size	parameters
Default			
conv1	observation	32×80×80	32, 5, 1, 2
conv2	conv1	32×40×40	32, 3, 2, 1
conv3	conv2	32×40×40	32, 5, 1, 2
conv4	conv3	32×20×20	32, 3, 2, 1
conv5	conv4	64×20×20	64, 3, 1, 1
conv6	conv5	64×10×10	64, 3, 2, 1
conv7	conv6	64×10×10	64, 3, 1, 0
conv8	conv7	64×5×5	64, 3, 2, 1
lstm	conv8	512	1024
[FC1, FC2]	lstm	1024	
Slimmer			
conv1	observation	32×80×80	16, 5, 1, 2
conv2	conv1	32×40×40	32, 3, 2, 1
conv3	conv2	16×40×40	32, 5, 1, 2
conv4	conv3	32×20×20	32, 3, 2, 1
conv5	conv4	64×20×20	32, 3, 1, 1
conv6	conv5	64×10×10	64, 3, 2, 1
conv7	conv6	32×10×10	64, 3, 1, 0
conv8	conv7	64×5×5	64, 3, 2, 1
lstm	conv8	512	1024
[FC1, FC2]	lstm	1024	
1D			
conv1	observation	32×24	32, 3, 1, 1
conv2	conv1	32×24	32, 3, 1, 1
conv3	conv2	64×25	64, 2, 1, 1
conv4	conv3	64×25	64, 1, 1, 0

Table 2.G.2: The encoder and recurrent modules for A3C-LSTM and ACER. The parameter tuple for convolutional layers corresponds to number of filters, kernel size, stride size and padding size and for LSTM corresponds to number of hidden units. After the conv layers are followed by LeakyReLU activation function, except some of the slimmer CNN's conv layers with the number of channels doubled at output stage are followed by CReLU. The 1D CNN encoder is for DDPG on BipedalWalker2D.

3 WORLD GRAPH DECOMPOSITION TO ACCELERATE REINFORCEMENT LEARNING

3.1 INTRODUCTION

Having investigated a general method of stochastic activation to improve exploration in actor-critic methods for general RL problems, we turn our attention to improve exploration and sample efficiency for a more specific scenario. Many real-world applications, e.g., self-driving cars and in-home robotics, require an autonomous agent to execute different tasks within a single environment that features, e.g., high-dimensional state space, complex world dynamics, or structured layouts. In these settings, model-free reinforcement learning (RL) agents often struggle to learn efficiently, requiring a large number of experience collections to converge to optimal behaviors. Intuitively, an agent could learn more efficiently by focusing its exploration in *task-relevant regions*, if it knows the high-level structure of the environment.

We propose a method[1] to 1) learn and 2) use an environment decomposition in the form of a *world graph*, a *task-agnostic* abstraction. World graph nodes are *waypoint* states, a set of salient states that can summarize agent trajectories and provide meaningful starting points for efficient exploration (Chatzigiorgaki and Skodras, 2009; Jayaraman et al., 2018; Ghosh et al., 2018). The directed and weighted world graph edges characterize feasible traversals among the waypoints. To leverage the world graph, we model hierarchical RL (HRL) agents where a high-level policy chooses a waypoint state as a goal to guide ex-

1 This chapter is based on our ICML 2019 paper (Shang et al., 2019a).

ploration towards task-relevant regions, and a low-level policy strives to reach the chosen goals.

Our framework consists of two phases. In the task-agnostic phase, we obtain world graphs by training a recurrent variational auto-encoder (VAE) (Chung et al., 2015; Gregor et al., 2015; Kingma and Welling, 2013) with binary latent variables (Nalisnick and Smyth, 2016) over trajectories collected using a random walk policy (Ha and Schmidhuber, 2018) and a curiosity-driven goal-conditioned policy (Ghosh et al., 2018; Nair et al., 2018). World graph nodes are states that are most frequently selected by the binary latent variables, while edges are inferred from empirical transition statistics between neighboring waypoints. In the task-specific phase, taking advantage of the learned world graph for structured exploration, we efficiently train an HRL model (Taylor and Stone, 2009).

In summary, our main contributions are:

- A task-agnostic unsupervised approach to learn world graphs, using a recurrent VAE with binary latent variables and a curiosity-driven goal-conditioned policy.
- An HRL scheme for the task-specific phase features multi-goal selection (Wide-then-Narrow) and navigation via world graph traversal.
- Empirical evaluations on multiple tasks in complex 2D grid worlds to validate that our framework produces descriptive world graphs and significantly improves both sample efficiency and final performance on these tasks over baselines, especially thanks to transferring learning from the unsupervised phase and world graph traversal.

3.2 RELATED WORK

An understanding of the environment and its dynamics is essential for effective planning and control in model-based RL. For example, a robotics agent often locates or navigates by interpreting a map (Lowry et al., 2015; Thrun, 1998; Angeli et al., 2008). Our exploration strategy draws inspiration from active localization, where robots are actively guided to investigate unfamiliar regions (Fox et al., 1998; Li et al., 2016). Besides mapping, recent works (Azar

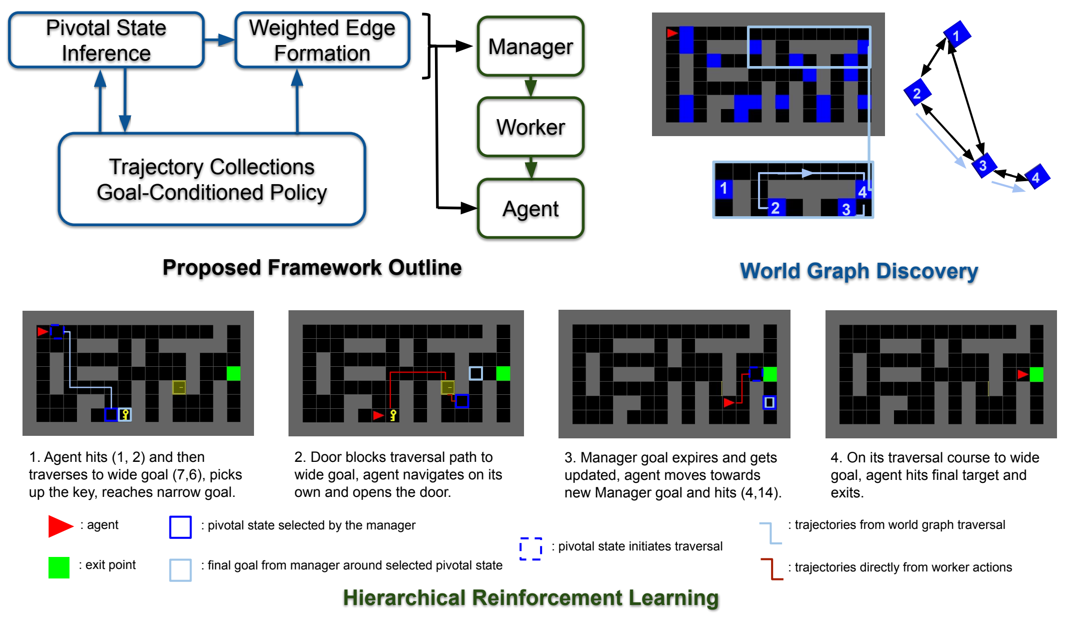

Pivotal State Inference
Weighted Edge Formation
Trajectory Collections Goal-Conditioned Policy
Manager
Worker
Agent
Proposed Framework Outline
World Graph Discovery
1. Agent hits (1, 2) and then traverses to wide goal (7,6), picks up the key, reaches narrow goal.
2. Door blocks traversal path to wide goal, agent navigates on its own and opens the door.
3. Manager goal expires and gets updated, agent moves towards new Manager goal and hits (4,14).
4. On its traversal course to wide goal, agent hits final target and exits.
: agent
: exit point
: pivotal state selected by the manager
: final goal from manager around selected pivotal state
: pivotal state initiates traversal
: trajectories from world graph traversal
: trajectories directly from worker actions
Hierarchical Reinforcement Learning

Figure 3.1: Top Left: overall pipeline of our 2-phase framework. Top Right (*world graph discovery*): a subgraph exemplifies traversal between waypoint states (in blue), see Section 3.3 for more details. Bottom (*Hierarhical RL*): an example rollout from our proposed HRL policy with Wide-then-Narrow Manager instructions and world graph traversals, solving a challenging *Door-Key* task, see Section 3.4 for more details.

et al., 2019; Ha and Schmidhuber, 2018; Guo et al., 2018) learn to represent the world with generative latent states (Tian and Gong, 2017; Haarnoja et al., 2018a; Racanière et al., 2017). If the latent dynamics are also extrapolated, the latent states can assist planning (Mnih et al., 2016b; Hafner et al., 2018) or model-based RL (Gregor and Besse, 2018; Kaiser et al., 2019).

While also aiming to model the world, we approach this as abstracting both the structure and dynamics of the environment in a graph representation. The nodes are states from the environment and edges encode actionable, efficient transitions between nodes. Existing works (Metzen, 2013; Mannor et al., 2004; Eysenbach et al., 2019; Entezari et al., 2010) have shown benefits of such graph abstractions but typically select nodes only subject to a good coverage the observed state space. Instead, we identify a parsimonious subset of states that can summarize trajectories and provide more useful intermediate landmarks, i.e., waypoints, for navigating complex environments.

Our method for estimating waypoint states can be viewed as performing automatic (sub)goal discovery. Subgoal and subpolicy learning are two major approaches to identify a set of temporally-extended actions, "skills", that allow agents to learn to solve complex tasks efficiently. Subpolicy learning identifies

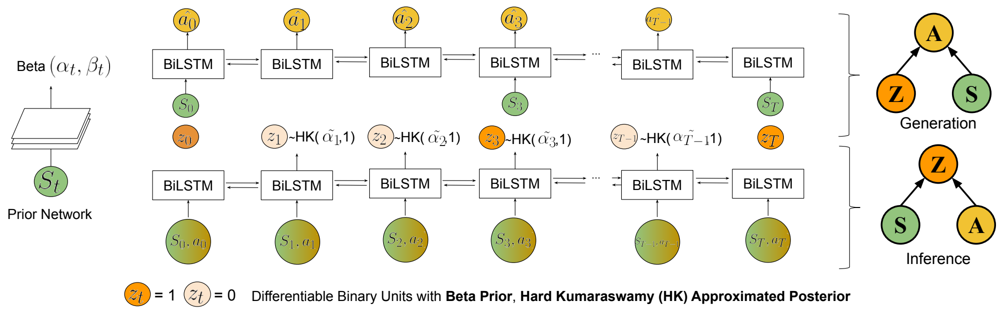

Beta (α_t, β_t)
Prior Network
BiLSTM
z_1~HK($\tilde{\alpha}_1$,1)
z_2~HK($\tilde{\alpha}_2$,1)
z_3~HK($\tilde{\alpha}_3$,1)
z_{T-1}~HK(α_{T-1}1)
Generation
Inference
z_t = 1 z_t = 0 Differentiable Binary Units with Beta Prior, Hard Kumaraswamy (HK) Approximated Posterior

Figure 3.2: Our recurrent latent model with differentiable binary latent units to identify waypoint states. A prior network (left) learns the state-conditioned prior in Beta distribution, $p_\psi(z_t|s_t)$=Beta(α_t, β_t). An inference encoder learns an approximate posterior in HardKuma distribution inferred from the state-action sequence input, $q_\phi(z_t|\boldsymbol{a}, \boldsymbol{z})$=HardKuma$(\tilde{\alpha}_t, 1)$. A generation network p_θ reconstructs $\boldsymbol{a}$ from $\{s_t|z_t{=}1\}$.

policies useful to solve RL tasks, such as option-based methods (Daniel et al., 2016; Bacon et al., 2017) and subtask segmentations (Pertsch et al., 2019; Kipf et al., 2018). Subgoal learning, on the other hand, identifies "important states" to reach (Şimşek et al., 2005).

Previous works consider various definitions of "important" states: frequently visited states during successful task completions (Digney, 1998; McGovern and Barto, 2001), states introducing the most novel information (Goyal et al., 2019), bottleneck states connecting densely-populated regions (Chen et al., 2007; Şimşek et al., 2005), or environment-specific heuristics (Ecoffet et al., 2019). Our work draws intuition from unsupervised temporal segmentation (Chatzigiorgaki and Skodras, 2009; Jayaraman et al., 2018) and imitation learning (Abbeel and Ng, 2004; Hussein et al., 2017). We define "important" states (waypoints) as the most critical states in recovering action sequences generated by some agents, which indicates that these states contain the richest information about the executed policy (Azar et al., 2019).

3.3 LEARNING WORLD GRAPHS

We propose a method for learning a *world graph* $\mathcal{G}_w$, a task-agnostic abstraction of an environment that captures its high-level structure and dynamics. In this work, the primary use of world graphs is to accelerate reinforcement learning of downstream tasks. The nodes of $\mathcal{G}_w$, denoted by a set of *waypoints* states

$s_p \in \mathcal{V}_p$, are generically "important" for accomplishing tasks within the environment, and therefore useful as starting points for exploration. Our method identifies such waypoint states from interactions with the environment. In addition, we embed feasible transitions between nearby waypoint states as the edges of $\mathcal{G}_w$.

In this work, we define important states in the context of *learning* $\mathcal{G}_w$ (see Section 3.2 for alternative definitions). That is, we wish to discover a small set of states that, when used as world graph nodes, concisely summarize the structure and dynamics of the environment. Below, we describe 1) how to collect state-action trajectories and an unsupervised learning objective to identify world graph nodes, and 2) how the graph's edges (i.e., how to transition between nodes) are formed from trajectories.

3.3.1 Waypoint State Identification

The structure and dynamics of an environment are implicit in the state-action trajectories observed during exploration. To identify world graph nodes from such data, we train a recurrent variational autoencoder (VAE) that, given a sequence of state-action pairs, identifies a subset of the states in the sequence from which the full action sequence can be reconstructed (Figure 3.2). In particular, the VAE infers *binary latent variables* that controls whether each state in the sequence is used by the generative decoder, i.e., whether a state is "important" or not. **Binary Latent VAE** The VAE consists of an inference, a generative and a prior network. These are structured as follows: the input to the inference network q_ϕ is a trajectory of state-action pairs observed from the environment $\boldsymbol{\tau}=\{(s_t, a_t)\}_{t=0}^{T}$, with $\boldsymbol{s}=\{s_t\}_{t=0}^{T}$ and $\boldsymbol{a}=\{a_t\}_{t=0}^{T}$ denoting the state and action sequences respectively. The output of the inference network is the approximated posterior over a sequence $\boldsymbol{z}=\{z_t\}_{t=0}^{T}$ of binary latent variables, denoted as $q_\phi(\boldsymbol{z}|\boldsymbol{a}, \boldsymbol{s})$. The generative network p_θ computes a distribution over the full action sequence $\boldsymbol{a}$ using the *masked* state sequence, where s_t is masked if $z_t=0$ (we fix $z_0=z_T=1$ during training), denoted as $p_\theta(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z})$.

Finally, a *state-conditioned* $p_\psi(z_t|s_t)$ given by the prior network p_ψ for each s_t encodes the empirical average probability that state s_t is activated for reconstruction. This choice encourages inference to select within a consistent subset of states for use in action reconstruction. In particular, *the waypoint states $\mathcal{V}_p$ are* *chosen as the states with the largest prior means* and during training, once every few iterations, $\mathcal{V}_p$ is updated based on the current prior network.

Objective Formally, we optimize the VAE using the following evidence lower bound (ELBO):

$$\text{ELBO} = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{a},\boldsymbol{s})}\left[\log p_\theta(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})\right] - D_{\text{KL}}\left(q_\phi(\boldsymbol{z}|\boldsymbol{a},\boldsymbol{s})|p_\psi(\boldsymbol{z}|\boldsymbol{s})\right). \tag{3.1}$$

To ensure differentiability, we apply a continuous relaxation over the discrete z_t. We use the Beta distribution $p_\psi(z_t) = \text{Beta}(\alpha_t, \beta_t)$ for the prior and the Hard Kumaraswamy distribution $q_\psi(z_t|\boldsymbol{a},\boldsymbol{z}) = \text{HardKuma}(\tilde{\alpha}_t, \tilde{\beta}_t)$ for the approximate posterior, which resembles the Beta distribution but is outside the exponential family (Bastings et al., 2019). This choice allows us to sample 0s and 1s without sacrificing differentiability, accomplished via the stretch-and-rectify procedure (Bastings et al., 2019; Louizos et al., 2017b) and the reparametrization trick (Kingma and Welling, 2013). Lastly, to prevent the trivial solution of using all states for reconstruction, we use a secondary objective $\mathcal{L}_0$ to regularize the L_0 norm of $\boldsymbol{z}$ at a targeted value μ_0 (Louizos et al., 2017b; Bastings et al., 2019), the desired number of selected states out of T steps, e.g., for when $T = 25$, we set $\mu_0 = 5$, meaning ideally 5 out of 25 states are activated for action reconstruction. Another term $\mathcal{L}_T$ to encourage temporal separation between selected states by targeting the number of $0/1$ switches among $\boldsymbol{z}$ at $2\mu_0$:

$$\mathcal{L}_0 = \left\|\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})}[\|\boldsymbol{z}\|_0] - \mu_0\right\|^2, \quad \mathcal{L}_T = \left\|\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})}\left[\sum_{t=0}^{T}\mathbb{1}[z_t \neq z_{t+1}]\right] - 2\mu_0\right\|^2. \tag{3.2}$$

See Appendix 3.A for details on training the VAE with binary z_t, including integration of the Hard Kumaraswamy distribution and how to regularize the statistics of $\boldsymbol{z}$.

3.3.2 Exploration for World Graph Discovery

Naturally, the latent structure learned by the VAE depends on the trajectories used to train it. Hence, collecting a rich set of trajectories is crucial. Here, we propose a strategy to bootstrap a useful set of trajectories by alternately exploring the environment based on the current iteration's $\mathcal{V}_p$ and updating the VAE and $\mathcal{V}_p$, repeating this cycle until the action reconstruction accuracy plateaus (Algorithm 6).

Algorithm 6 Identifying waypoint states $\mathcal{V}_p$ and learning a goal-conditioned policy π_g

```
Result: Waypoint states $\mathcal{V}_p$ and a goal-conditioned policy $\pi_g$
Initialize network parameters for the recurrent variational inference model $V$
Initialize network parameters for the goal-conditioned policy $\pi_g$
Initialize $\mathcal{V}_p$ with the initial position of the agent, i.e. $\mathcal{V}_p = \{s_0 = (1, 1)\}$
while VAE reconstruction error has not converged do
    for $n \leftarrow 1$ to $N$ do
        Sample random waypoint $s_p \in \mathcal{V}_p$
        Navigate agent to $s_p$ and perform $T$-step rollout using a randow walk policy: $\tau^r_n \leftarrow \{(s_0 = s_p, a_0), ..., (s_T, a_T)\}$
        $g_n \leftarrow s_T$
        Navigate agent to $s_p$ and perform $T$-step rollout using $\pi_g$ with goal $g_n$: $\tau^\pi_n \leftarrow \{(s_0 = s_p, a_0), ..., (s_T, a_T)\}_{a_t \sim \pi_g(\cdot|s_t, g_n)}$
        Re-label $\pi_g$ rewards with action reconstruction error as curiosity bonus: $r^\pi_n \leftarrow \{\mathbb{1}_{s_{t+1}=g_n} - \lambda \cdot p_\theta(a_t|\boldsymbol{s}, \boldsymbol{z})\}^T_{t=0}$
    end
    Perform policy gradient update of $\pi_g$ using $\tau^\pi$ and $r^\pi$
    Update $V$ using $\tau^r$ and $\tau^\pi$
    Update $\mathcal{V}_p$ as set of states with largest prior mean $\frac{\alpha_s}{\alpha_s+\beta_s}$.
end
```

During exploration, we use action replay to navigate the agent to a state drawn from the current iteration's $\mathcal{V}_p$. Although resetting via action replay assumes our underneath environment to be deterministic, in cases where this resetting strategy is infeasible, it may be modified so long as to allow the exploration starting points to expand as the agent discovers more of its environment. For each such starting point, we collect two rollouts. In the first rollout, we perform a random walk to explore the nearby region. In the second rollout, we perform actions using a goal-conditioned policy π_g (GCP), setting the final state reached by the random walk as the goal. Both rollouts are used for training the VAE, and the latter is also used for training π_g. GCP provides a venue to integrate intrinsic motivation, such as curiosity (Burda et al., 2018; Achiam and Sastry, 2017; Pathak et al., 2017; Azar et al., 2019) to generate more diverse rollouts. Specifically, we use the action reconstruction error of the VAE as an intrinsic reward signal when training π_g. This choice of curiosity also prevents the VAE from collapsing to the simple behaviors of a vanilla π_g.

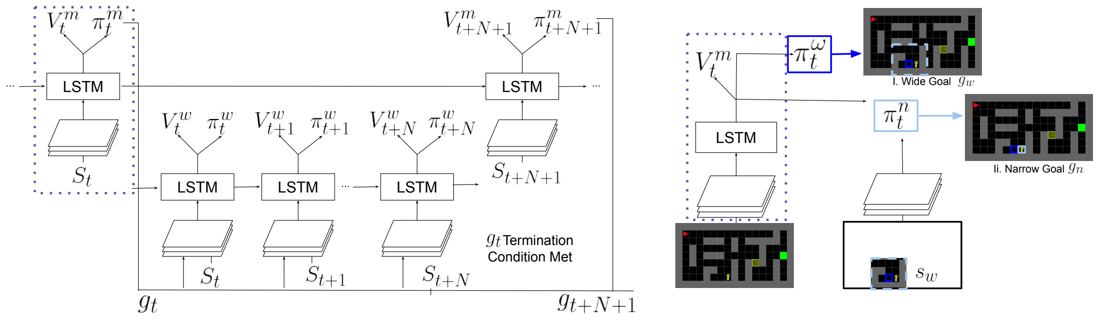


Figure 3.3: Left: a standard Feudal Network. Right: using Wide-then-Narrow goals. The Manager first outputs a waypoint state as the wide goal g^w, then attends to a closer-up area around g^w to narrow down the final goal g^n.

3.3.3 Edge Formation

The final stage is to construct the edges of $\mathcal{G}_w$, which should ideally capture the environment dynamics, i.e., how to transition between waypoint states. Once VAE training is complete and $\mathcal{V}_p$ is fixed, we collect random walk rollouts from each of the waypoints $s_p \in \mathcal{V}_p$ to estimate the underlying adjacency matrix (Biggs, 1993). More precisely, we claim a directed edge $s_p \rightarrow s_q$ if there exists a random walk trajectory from s_p to s_q that does not intersect a third waypoint. We also consider paths taken by π_g (starting at s_p and setting s_q as the goal) and keep the shortest observed path from s_p to s_q as a world graph edge transition. We use the action sequence length of the edge transition between adjacent waypoints as the *weight* of the edge. As shown experimentally, a key benefit of our approach is the ability to plan over $\mathcal{G}_w$. To navigate from one waypoint to another, we can use dynamic programming (Sutton and Barto, 1998b; Feng et al., 2004) to output the optimal traversal of the graph.

3.4 ACCELERATING REINFORCEMENT LEARNING WITH WORLD GRAPHS

World graphs present a high-level, task-agnostic abstraction of the environment through waypoints and feasible transition routes between them. A key example of world graph applications for task-specific RL is *structured exploration*: instead of exploring the entire environment, RL agents can use world graphs to quickly identify task-relevant regions and bias low-level exploration to these regions. Our framework to leverage world graphs for structured exploration consists of two parts:

1. Hierarchical RL wherein the high-level policy selects subgoals from $\mathcal{V}_p$.
2. Traversals using world graph edges.

3.4.1 Hierarchical RL over World Graphs

Formally, an RL agent learning to solve a task is formulated as a Markov Decision Process: at time t, the agent is in a state s_t, executes an action a_t via a policy $\pi(a_t|s_t)$ and receives a rewards r_t. The agent's goal is to maximize its cumulative expected return $R = \mathbb{E}_{(s_t,a_t)\sim\pi,p,p_0}\left[\sum_{t\geq 0}\gamma^t r_t\right]$, where $p(s_{t+1}|s_t, a_t), p_0(s_0)$ are the transition and initial state distributions.

To incorporate world graphs with RL, we use a hierarchical approach based on the Feudal Network (FN) (Dayan and Hinton, 1993; Vezhnevets et al., 2017), depicted in Figure 3.3. A standard FN decomposes the policy of the agent into two separate policies that receive distinct streams of reward: a high-level policy ("Manager") learns to propose subgoals; a low-level policy ("Worker") receives subgoals from the Manager as inputs and is rewarded for taking actions in the environment that reach the subgoals. The Manager receives the environment reward defined by the task and therefore must learn to emit subgoals that lead to task completion. The Manager and Worker do not share weights and operate at different temporal resolutions: the Manager only outputs a new subgoal if either the Worker reaches the chosen one or a subgoal horizon c is exceeded.

For all our experiments, policies are trained using advantage actor-critic (A2C), an on-policy RL algorithm (Wu and Tian, 2016; Pane et al., 2016; Mnih et al., 2016a). To ease optimization, the feature extraction layers of the Manager and Worker that encode s_t are initialized with the corresponding layers from π_g, the GCP learned during the world graph discovery phase. More details are in Appendix 3.B.

3.4.2 Wide-then-Narrow Goals and World Graphs

To incorporate the world graph, we introduce a Manager policy that factorizes subgoal selection as follows: a *wide* policy $\pi^w(g_t^w|s_t)$ selects a waypoint state as the *wide* goal $g^w \in \mathcal{V}_p$, and a *narrow* policy $\pi^n(g_t^n|s_t, g_t^w)$ selects a state within a local neighborhood of g_t^w, i.e. its ϵ-net (Mahadevan and Maggioni, 2007), as the *narrow* goal $g^n \in \{s : \mathcal{D}(s, g_t^w) \leq \epsilon\}$. The Worker policy $\pi^{\text{worker}}(a_t | s_t, g_t^n, g_t^w)$ chooses the action taken by the agent given the current state and the wide and narrow goals from the Manager. A visual illustration is in Figure 3.3 and training details in Appendix 3.C.2.

3.4.3 World Graph Traversal

The wide-then-narrow subgoal format simplifies the search space for the Manager policy. Using waypoints as wide goals also makes it possible to leverage the world graph's edges for planning and executing the planned traversals. This process breaks down as follows:

1. **When to Traverse:** When the agent encounters a waypoint state $s_t \in \mathcal{V}_p$, a "traversal" is initiated if s_t has a feasible connection in $\mathcal{G}_w$ to the active wide goal g_t^w.

2. **Planning:** Upon triggering a traversal, the optimal traversal route from the initiating state to g_t^w is estimated from the $\mathcal{G}_w$ edge weights using classic dynamic programming planning (Sutton and Barto, 1998b; Feng et al., 2004). This yields a sequence of intermediate waypoint states.

3. **Execution:** The execution of graph traversals depends on the nature of the environment. If deterministic, the agent simply follows the action sequences given by the edges of the traversal. Otherwise, the agent uses the pretrained GCP π_g to sequentially reach each of the intermediate waypoint states along the traversal (we fine-tune π_g in parallel where applicable). If the agent fails to reach the next waypoint state within a certain time limit, it stops its current pursuit, and a new (g^w, g^n) pair is received from the Manager.

World graph traversal allows the Manager to assign task-relevant wide goals g^w that can be far away from the agent yet still reachable, which consequentially accelerates learning by focusing exploration around the task-relevant region near g^w.

Task	Task Description	Environment Characteristics
MultiGoal	Collect randomly spawned balls. Each ball gives +1 reward. To end an episode, the agent has to exit at a designated point.	Balls are located randomly, dense reward.
MultiGoal-Sparse	Agents receive a single reward $r \leq 1$ proportional to the number of balls collected upon exiting.	Balls are located randomly, sparse reward.
MultiGoal-Stochastic	Spawn lava blocks at random locations each time step that immediately terminates the episode if stepped on.	*Stochastic* environment. Multiple objects: lava and balls are randomly located, dense reward.
Door-Key	Agent has to pick up a key to open a door (reward +1) and reach the exit point on the other side (reward +1).	Walls, door, and key are located randomly. Agents have additional actions: pick and toggle.

Table 3.1: An overview of tasks used to evaluate the benefit of using world graphs. Visualizations can be found in Appendix 3.D.

3.5 EXPERIMENTAL VALIDATION

We now assess each component of our framework on a set of challenging 2D grid worlds. Our ablation studies demonstrate the following benefits of our framework:

- It improves sample efficiency and performance over the baseline HRL model.
- It benefits tasks varying in environment scale, task type, reward structure, and stochasticity.
- The identified waypoints provide superior world representations for solving downstream tasks compared to graphs using randomly selected states as nodes.

Implementation details, snippets of the tasks and mazes are in Appendix 3.C-3.D.

3.5.1 Ablation Studies on 2D Grid Worlds

For our ablation studies, we construct 2D grid worlds of increasing sizes (small, medium, and large) along with challenging tasks with different reward structures, levels of stochasticity, and logic (summarized in Table 3.1). In all tasks, every action taken by the agent receives a negative reward penalty. We follow a rigorous evaluation protocol (Wu et al., 2017; Ostrovski et al., 2017; Henderson et al., 2018): each experiment is repeated with 3 training seeds. 10 additional

Task	**Size**	A2C	FN $+\pi_g$ **init**	**Ours**		
				$+\pi_g$**-init**	$+\ \mathcal{G}_w$**-traversal**	$+\ \pi_g$**-init** $+\ \mathcal{G}_w$**-traversal**
MultiGoal	Small	2.04±0.05	2.93±0.74	**5.25±0.13**	3.92±0.22	5.05±0.03
	Medium	-	-	**5.15±0.11**	2.56±0.09	3.00±0.90
	Larger	-	-	-	2.18±0.12	**2.72±0.59**
MultiGoal-Sparse	Small	-	-	0.39±0.09	0.24±0.04	**0.42±0.07**
	Medium	-	-	-	0.20±0.04	**0.25±0.03**
	Larger	-	-	-	0.16±0.22	**0.26±0.11**
MultiGoal-Stochastic	Small	1.38±1.20	1.93±0.16	**3.06±0.31**	-	2.92±0.45
	Medium	-	-	2.99±0.12	2.42±0.24	**2.64±0.14**
	Larger	-	-	-	-	**0.60±0.12**
Door-Key	Small	-	-	**0.99±0.00**	0.37±0.15	0.92±0.02
	Medium	-	-	0.56±0.02	-	**0.76±0.06**
	Larger	-	-	-	-	**0.26±0.19**

Table 3.2: On a variety of tasks and environment setups, we evaluate RL models trained with GCP π_g initialization, with $\mathcal{G}_w$ world graph traversal, and with both. All models on the right are equipped with WN. Left are baselines for additional comparison. We report final rewards for *MultiGoal* tasks, and success rates for *Door-Key* are reported. If no result was reported, the agent failed to solve the task.

Waypoint type	*MultiGoal*	*MultiGoal-Sparse*	*MultiGoal-Stochastic*	*Door-Key*
Learned	**2.72**±0.59	**0.26**±0.11	**0.60**±0.12	**0.26**±0.19
Random	2.30±0.49	0.19±0.11	0.41±0.25	**0.27**±0.40

Table 3.3: Comparing learned $\mathcal{V}_p$ versus random $\mathcal{V}_{\text{rand}}$ as wide subgoals on large mazes, all trained with π_g initialization and graph traversal. $\mathcal{V}_p$ generally is superior in terms of performance and consistency. We report final rewards for *MultiGoal* tasks and success rates for *Door-Key* are reported.

validation seeds are used to pick the model with the best reward performance. This model is then tested on 100 testing seeds. We report the mean reward and standard deviation.

We ablate each of the following components in our framework and compare against non-hierarchical (A2C) and hierarchical baselines (FN):

1. initializing the feature extraction layers of the Manager and Worker from π_g,
2. applying Wide-then-Narrow Manager (WN) goal instruction, and
3. allowing the Worker to traverse along $\mathcal{G}_w$.

Results are shown in Table 3.2. In sum, each component improves performance over the baselines.

WIDE AND NARROW GOALS Using two-goal types is a highly effective way to structure the Manager instructions and enables the Worker to differentiate the transition and local task-solving phases. We note that for small *MultiGoal*, agents do not benefit much from $\mathcal{G}_w$ traversal: it can rely solely on the guidance from WN goals to master both phases. However, with increasing maze size, the Worker struggles to master traversals on its own and thus fails to solve the tasks.

WORLD GRAPH TRAVERSAL As conjectured in Section 3.4.3, the performance gain of our framework can be explained by the broader range and more targeted exploration strategy. In addition, the Worker does not have to learn long-distance transitions with the aid of $\mathcal{G}_w$ traversals. Figure 3.4 confirms that $\mathcal{G}_w$ traversal speeds up convergence and its effect becomes more evident with larger mazes. Note that the graph learning stage only needs 2.4K iterations to converge. Even when taking these additional environment interactions into account, $\mathcal{G}_w$ traversal still exhibits superior sample efficiency, not to mention that the graph is shared among all tasks. Moreover, solving *Door-Key* involves a complex combination of sub-tasks: find and pick up the key, reach and open the door and finally exit. With limited reward feedback, this is particularly difficult to learn. The ability to traverse along $\mathcal{G}_w$ enables longer-horizon planning on top of the waypoints, thanks to which the agents boost the success rate on medium *Door-Key* from 0.56 ± 0.02 to 0.75 ± 0.06.

BENEFITS OF LEARNED WAYPOINTS To highlight the benefit of establishing the waypoints learned by the VAE as nodes for $\mathcal{G}_w$, we compare against results using a $\mathcal{G}_w$ constructed around randomly selected states ($\mathcal{V}_{\text{rand}}$). The random-node graph edges are formed in the same way as described in Section 3.3.3, and its feature extractor is also initialized from π_g. Although granting knowledge acquired during the unsupervised phase to $\mathcal{V}_{\text{rand}}$ is unfair to $\mathcal{V}_p$, deploying both initialization and traversal while only varying $\mathcal{V}_{\text{rand}}$ and $\mathcal{V}_p$ isolates the effect from the nodes to the best extent. The comparative results (in Table 3.3, learning curves for *MultiGoal* in Figure 3.4) suggest $\mathcal{V}_p$ generally outperforms $\mathcal{V}_{\text{rand}}$. *Door-Key* is the only task in which the two matches. However, $\mathcal{V}_{\text{rand}}$ exhibits a large variance, implying that certain sets of random states can be suitable for this task, but using learned waypoints gives the strong performance more consistently.

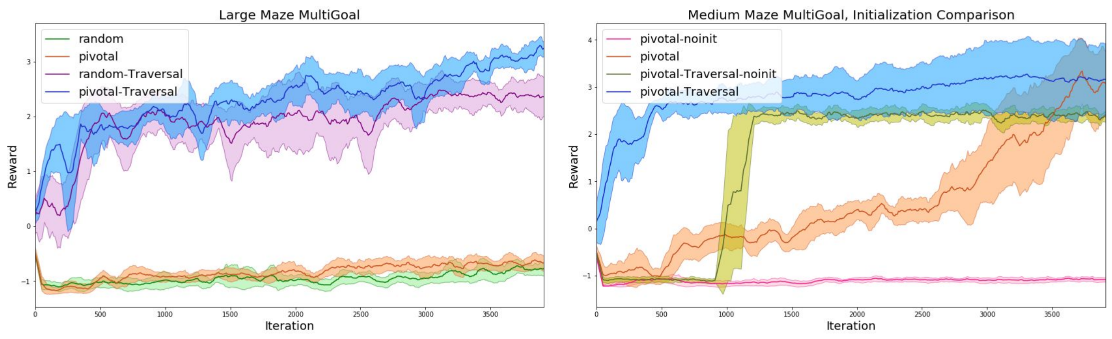


Figure 3.4: Validation performance during training (mean and standard-deviation of reward, 3 seeds) for *MultiGoal*. Left: Comparing $\mathcal{V}_p$ and $\mathcal{V}_{\text{rand}}$, with or without traversal, all models use WN and π_g initialization. We see that 1) traversal speeds up convergence, 2) $\mathcal{V}_{\text{rand}}$ gives higher variance, and slightly worse performance than $\mathcal{V}_p$. Right: comparing with or without π_g initialization on $\mathcal{V}_p$, all models use WN. We see that initializing the task-specific phase with the task-agnostic goal-conditioned policy boosts learning.

INITIALIZATION WITH GCP Initializing the weights of the Worker and Manager feature extractors from π_g (learned during the task-agnostic phase) consistently benefits learning.We observe that models starting from scratch fail on almost all tasks within the maximal number of training iterations unless coupled with $\mathcal{G}_w$ traversal, which is still inferior to using π_g-initialization. Notably, for the small *MultiGoal-Stochastic* environment, there is a high chance that a lava square blocks traversal; therefore, without the environment knowledge from π_g transferred by weight initialization, the interference created by the episode-terminating lava prevents the agent from learning the task.

3.6 CONCLUSION

We have shown that world graphs are powerful environment abstractions, which, in particular, are capable of accelerating reinforcement learning. Future works may extend their applications to more challenging RL setups, such as real-world multi-task learning and navigation. It is also interesting to generalize the proposed framework to learn dynamic world graphs for evolving environments and apply world graphs to multi-agent problems, where agents become part of other agents' world graphs.

CHAPTER APPENDIX

3.A RECURRENT VAE WITH DIFFERENTIABLE BINARY LATENT VARIABLES

As illustrated in the main text, the main objective for the recurrent VAE is the following evidence lower bound with derivation:

$$\begin{aligned}
\log p(a|s) &= \log \int p(a|s,z)dz \\
&= \log \int p(a|s,z)p(z|s)\frac{q(z|a,s)}{q(z|a,s)}dz \\
&= \log \int p(a|s,z)\frac{p(z|s)}{q(z|a,s)}q(z|a,s)dz \\
&\geq \mathbb{E}_{q(z|a,s)}[\log p(a|s,z) - \log \frac{q(z|a,s)}{p(z|s)}] \\
&= \mathbb{E}_{q(z|a,s)}[\log p(a|s,z)] - D_{\mathrm{KL}}(q(z|a,s)||p(z|s))
\end{aligned}$$

The inference network q_ψ takes in the trajectories of state-action pairs τ and at each time step approximates the posterior of the corresponding latent variable z_t. The prior network p_ψ takes the state s_t at each time step and outputs the state-conditioned prior $p_\psi(s_t)$. We choose Beta as the prior distribution and the Hard Kuma as the approximated posterior to relax the discrete latent variables to continuous surrogates.

The Kuma distribution Kuma(α, β) highly resembles the Beta Distribution in shape but does not come from the exponential family. Similar to Beta, the Kuma distribution also ranges from bimodal (when $\alpha \approx \beta$) to unimodal ($\alpha/\beta \to 0$ or $\alpha/\beta \to \infty$). Also, when $\alpha = 1$ or $\beta = 1$, Kuma$(\alpha, \beta) =$ Beta(α, β). We observe empirically better performance when we fix $\beta = 1$ for the Kuma approximated posterior. One major advantage of the Kuma distribution is its simple Cumulative Distribution Function (CDF):

$$F_{\mathrm{Kuma}}(x, \alpha, \beta) = (1 - (1 - x^\alpha))^\beta. \tag{3.3}$$

It is therefore amendable to the reparametrization trick (Kingma and Welling, 2013; Rezende et al., 2014; Maddison et al., 2017) by sampling from uniform distribution $u \sim \mathcal{U}(0,1)$:

$$z = F^{-1}_{\text{Kuma}}(u; \alpha, \beta) \sim \text{Kuma}(\alpha, \beta). \tag{3.4}$$

Lastly, the KL-divergence between the Kuma and Beta distributions can be approximated in closed form (Nalisnick and Smyth, 2016):

$$\begin{aligned} D_{\text{KL}}(\text{Kuma}(a,b)|\text{Beta}(\alpha,\beta)) &= \frac{a-\alpha}{a}\left(-\gamma - \Psi(b) - \frac{1}{b}\right) \\ &+ \log(ab) + \log \text{Beta}(\alpha,\beta) - \frac{b-1}{b} + (\beta-1)b\sum_{m=1}^{\infty}\frac{1}{m+ab}\text{Beta}\left(\frac{m}{a}, b\right), \end{aligned} \tag{3.5}$$

where Ψ is the Digamma function, γ the Euler constant, and the approximation uses the first few terms of the Taylor series expansion. We take the first 5 terms here.

Next, we make the Kuma distribution "hard" by following the steps in Bastings et al., 2019. First stretch the support to $(r = 0 - \epsilon_1, l = 1 + \epsilon_2)$, $\epsilon_1, \epsilon_2 > 0$, and the resulting CDF distribution takes the form:

$$F_S(z) = F_{\text{Kuma}}\left(\frac{z-l}{r-l}; \alpha, \beta\right). \tag{3.6}$$

Then, the non-eligible probabilities for 0's and 1's are attained by rectifying all samples below 0 to 0 and above 1 to 1, and other value as it is, that is

$$P(z=0) = F_{\text{Kuma}}\left(\frac{-l}{r-l}; \alpha, \beta\right), \quad P(z=1) = 1 - F_{\text{Kuma}}\left(\frac{1-l}{r-l}; \alpha, \beta\right). \tag{3.7}$$

Lastly, we impose two additional regularization terms $\mathcal{L}_0$ and $\mathcal{L}_\mathcal{T}$ on the approximated posteriors. As described in the main text, $\mathcal{L}_0$ prevents the model from selecting all states to reconstruct $\{a_t\}_0^{T-1}$ by restraining the expected L_0 norm of $\boldsymbol{z} = (z_1 \cdots z_{T-1})$ to approximately be at a targeted value μ_0 (Louizos et al., 2017b; Bastings et al., 2019). In other words, this objective adds the constraint that there should be μ_0 of activated $z_t = 1$ given a sequence of length T. The other term $\mathcal{L}_\mathcal{T}$ encourages temporally isolated activation of z_t, meaning the number of transitions between 0 and 1 among z_t's should roughly be $2\mu_0$. Note that both expectations in Equation 3.2 have closed forms for HardKuma.

$$\mathcal{L}_0 = \left\| \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})} \left[\|\boldsymbol{z}\|_0 \right] - \mu_0 \right\|^2, \text{ where} \tag{3.8}$$

$$\begin{aligned}\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})} \left[\|\boldsymbol{z}\|_0 \right] &= \sum_{t=1}^{T} \mathbb{E}_{q(z_t|\boldsymbol{s},\boldsymbol{a})} \left[\mathbb{1}_{z_t \neq 0} \right] \\ &= \sum_{t=1}^{T} 1 - p\left(z_t = 0\right) = \sum_{t=1}^{T} 1 - F_{\text{Kuma}} \left(\frac{-l}{r-l}; \alpha_t, \beta_t \right),\end{aligned} \tag{3.9}$$

$$\mathcal{L}_T = \|\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})} \sum_{t=1}^{T-1} \mathbb{1}_{z_t \neq z_{t+1}} - 2\mu_0\|^2, \text{ where} \tag{3.10}$$

$$\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})}[\sum_{t=1}^{T-1} \mathbb{1}_{z_t \neq z_{t+1}}] = \sum_{t=1}^{T-1} p\left(z_t = 0\right) \left(1 - p\left(z_{t+1} = 0\right)\right) + \left(1 - p\left(z_t = 0\right)\right) p\left(z_{t+1} = 0\right). \tag{3.11}$$

LAGRANGIAN RELAXATION. The overall optimization objective consists of action sequence reconstruction, KL-divergence between the posterior and prior, $\mathcal{L}_0$ and $\mathcal{L}_T$ (Equation 3.12). We tune the objective weights λ_i using Lagrangian relaxation (Higgins et al., 2017a; Bastings et al., 2019; Bertsekas, 1999), treating λ_i's as learnable parameters and performing alternative optimization between λ_i's and the model parameters. We observe that as long as their initialization is within a reasonable range, λ_i's converge to a local optimum:

$$\max_{\{\lambda_{1,2,3}\}} \min_{\{\theta,\phi,\psi\}} -\mathbb{E}_{q_\psi(\boldsymbol{z}|\boldsymbol{a},\boldsymbol{s})} \left[\log p_\theta(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z}) \right] + \lambda_1 D_{\text{KL}} \left(q_\phi(\boldsymbol{z}|\boldsymbol{a},\boldsymbol{s}) | p_\psi(\boldsymbol{z}|\boldsymbol{s}) \right) + \lambda_2 \mathcal{L}_0 + \lambda_3 \mathcal{L}_T. \tag{3.12}$$

We observe this approach to produce efficient and stable mini-batch training.

3.B GOAL-CONDITIONED POLICY INITIALIZATION FOR HRL

Optimizing composite neural networks like HRL (Co-Reyes et al., 2018) is sensitive to weight initialization (Mishkin and Matas, 2015; Le et al., 2015), due to its complexity and lack of clear supervision at various levels. Therefore, taking inspiration from prevailing pre-training procedures in computer vision (Russakovsky et al., 2015; Donahue et al., 2014) and NLP (Devlin et al., 2018;

Radford et al., 2019), we take advantage of the weights learned by π_g during world graph discovery when initializing the Worker and Manager policies for downstream HRL, as π_g has already implicitly embodied much environment dynamics information.

More specifically, we extract the weights of the feature extractor, i.e., the state encoder, and use them as the initial weights for the state encoders of the HRL policies. Our empirical results demonstrate that such weight initialization consistently improves performance and validates the value of skill/knowledge transfer from GCP (Taylor and Stone, 2009; Barreto et al., 2017).

3.C ADDITIONAL IMPLEMENTATION DETAILS

Model code folder including all architecture details is shared in comment.

3.C.1 Hyperparameters for VAE training

Our models are optimized with Adam (Kingma and Ba, 2014) using mini-batches of size 128, thus spawning 128 asynchronous agents to explore. We use an initial learning rate of 0.0001, with $\epsilon = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$; gradients are clipped to 40 for inference and generation nets. For HardKuma, we set $l = -0.1$ and $r = 1.1$. The maximum sequence length for BiLSTM is 25. The total number of training iterations is 3600 and model usually converges around 2400 iterations. We train the prior, inference, and generation networks end-to-end.

We initialize λ_i's (see **Lagrangian Relaxation**) to be $\lambda_1 = 0.01$ (KL-divergence),$\lambda_2 = 0.06$ ($\mathcal{L}_0$), $\lambda_3 = 0.02$ ($\mathcal{L}_T$). After each update of the latent model, we update λ_i's, whose initial learning rate is 0.0005, by maximizing the original objective in a similar way as using Lagrangian Multiplier. At the end of optimization, λ_i's converge to locally optimal values. For example, with the medium maze, $\lambda_1 = 0.067$ for the KL-term, $\lambda_2 = 0.070$ for the $\mathcal{L}_0$ and $\lambda_3 = 0.051$ for the $\mathcal{L}_T$ term. The total number of waypoints $|\mathcal{V}_p|$ is set to be 20% of the full state space.

3.C.2 Training HRL models

The procedure of the Manager and the Worker in sending/receiving orders using either traversal paths among $\mathcal{V}_p$ from replay buffer for deterministic environments or with π_g for stochastic ones follows:

1. The Manager gives a wide-narrow subgoal pair (g_w, g_n).

2. The agent takes action based on the Worker policy π^ω conditioned on (g_w, g_n) and reaches a new state s'. If $s' \in \mathcal{V}_p$, g_w has not yet met, and there exists a valid path basing on the edge paths from the world graph $s' \rightarrow g_w$, agent then either follows replay actions or π_g to reach g_w. If π_g still does not reach the desired destination in certain steps, then stop the agent wherever it stands; also, π_g can be finetuned here.

3. The Worker receives a positive reward for reaching g_w for the first time.

4. If the agent reaches g_n, the Worker also receives positive rewards and terminates this horizon.

5. The Worker receives negative for every action taken except for during traversal; the Manager receives a negative reward for every action taken, including traversal.

6. When either g_n is reached or the maximum time step for this horizon is met, the Manager renews its subgoal pair.

The training of the Worker policy π^ω follows the same A2C algorithm as π_g. The training of the Manager policy π^m also follows a similar procedure, but as it operates at a lower temporal resolution, its value function regresses against the t_m-step discounted reward where t_m covers all actions and rewards generated from the Worker.

When using the Wide-then-Narrow instruction, the policy gradient for the Manager policy π_m becomes:

$$\mathbb{E}_{(s_t,a_t)\sim\pi,p,p_0}\left[A_{m,t}\nabla \log\left(\pi^\omega\left(g_{w,t}|s_t\right)\pi^n\left(g_{n,t}|s_t, g_{w,t}, s_{w,t}\right)\right)\right] + \nabla\left[\mathcal{H}\left(\pi^\omega\right) + \mathcal{H}\left(\pi^n(\cdot|g_{w,t})\right)\right],$$

where $A_{m,t}$ is the Manager's advantage at time t. Also, for Manager, as the size of the action space scales linearly with $|\mathcal{S}|$, the exact entropy for the π^m can quickly become intractable. Essentially there are $O\left(|\mathcal{V}| \times (N^2)\right)$ possible actions. To calculate the entropy exactly, all of them has to be summed, making it easily computationally intractable:

$$\mathcal{H} = \sum_{w \in \mathcal{V}} \sum_{w_n \in s_w} \pi^n(w_n|s_w, s_t)\pi^\omega(w|s_t) \log \nabla \pi^n(w_n|s_w, s_t)\pi^\omega(w|s_t).$$

Thus in practice we resort to an effective alternative $\mathcal{H}(\pi^\omega) + \mathcal{H}(\pi^n(\cdot|g_{w,t}))$.

Psuedo-code for Manager training is in Algorithm 7.

Algorithm 7 Training of π^m for HRL models

```
Initialize network parameters $\theta$ for $\pi^m$, here $\pi^{m,t}$ refers to the policy at time rollout time step $t$  Given a map of $\mathcal{V}$, $s_\mathcal{V}$  for iter $= 0, 1, 2, \cdots$ do
    Clear gradients $d\theta \leftarrow 0$  Reset the set of time steps where $\pi^{m,t}$ omits a new subgoal $S_m = \{\}$ and $t_m = 0$.  while $t <= t_{\max}$ or episode not terminated do
        Simulate under current policy $\pi^{m,t-1}, \pi^{w,t-1}$  if the Worker has met the previous subgoal or exceeded the horizon $c$ then
            Sample a new subgoal $g_{m,t}$ from $\pi^{m,t}$ $z_{m,t} = f_{\text{LSTM}}(\text{CNN}(s_{m,t}, s_\mathcal{V}), h_{m,t_m})$, $V_{m,t} = f_v(z_{m,t})$, $\pi_t = f_p(z_{m,t})$
        end
        $S_m = S_m \cup \{t_m\}$ and $t_m = t$
    end
    $R = \begin{cases} 0, \text{ if terminal} \\ V_{t_{\max}+1}, \text{otherwise} \end{cases}$  for $t = t_{\max}, \cdots 1$ do
        $R \leftarrow r_t + \gamma R$  if $t \in S_m$ then
            $A_{m,t} \leftarrow R - V_{m,t}$  Accumulate gradients from value loss: $d\theta \leftarrow d\theta + \lambda \frac{\partial A^2_{m,t}}{\partial \theta}$  Accumulate policy gradients with entropy regularization: $d\theta \leftarrow d\theta + \nabla \log \pi_{m,t}(g_{m,t}) A_{m,t} + \beta \nabla H(\pi_{m,t})$
        end
    end
end
```

3.c.3 Hyperparameters for HRL

For training the HRL policies, we inherit most hyperparameters from those used when training π_g, as the Manager and the Worker both share similar architectures with π_g. The hyperparameters used when training π_g follow those from Shang et al., 2019b. Because the tasks used in HRL experiments are more difficult than the generic goal-reaching task, we set the maximal number of training iterations to 100K, and training is stopped early if model performance reaches a plateau. The rollout steps for each iteration is 60. Hyperparameters specific to HRL is the horizon $c = 20$ and the size of the Manager's local attention range (that is, the neighborhood around g^w within which g^n is selected), which are $N = 5$ for small and medium mazes, and $N = 7$ for the large maze.

3.D 2D GRID WORLD VISUALIZATIONS

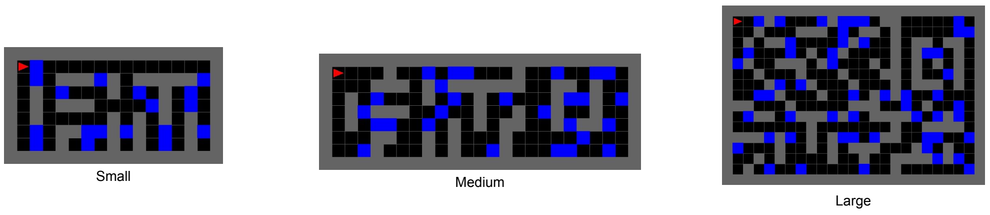


Figure 3.D.1: Visualization of the 2D grid environments in our experiments, along with the learned waypoints in blue.

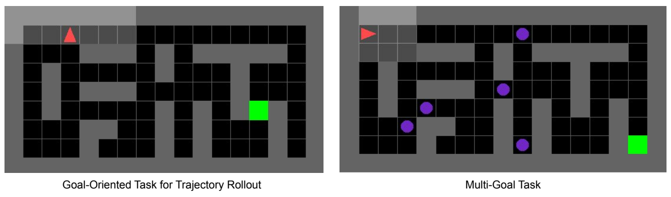


Figure 3.D.2: Visualization of tasks in our experiments.

4 AGENT-CENTRIC REPRESENTATIONS FOR MULTI-AGENT REINFORCEMENT LEARNING

4.1 INTRODUCTION

So far, we have tackled single-agent RL problems. Meanwhile, many real-world settings involve multiple agents. In this chapter, we look at the important problem class, multi-agent reinforcement learning (MARL). A distinct feature of MARL is the intrinsic relational reasoning among agents for e.g. cooperation. Of course, relational reasoning is fundamental to human intelligence. Human perception and understanding of the world are structured around objects. Inspired by this cognitive foundation, many recent efforts have successfully built strong object-centric inductive biases into neural architectures and algorithms to tackle relational reasoning tasks, from robotics manipulation (Devin et al., 2018) to visual question answering (Shi et al., 2019). Nonetheless, in MARL, such relational reasoning is still under-explored.

This work studies how agent-centric representations can benefit model-free MARL where each agent generates its policy independently. We consider a fully cooperative scenario, which can be modeled as a Multi-Agent Markov Decision Process (MAMDP), an extension of the single-agent MDP (Boutilier, 1996). In light of recent advances in model-free RL and neural relational reasoning (Jaderberg et al., 2016; Zambaldi et al., 2018), we study two ways of incorporating agent-centric inductive biases into our algorithm[1].

1 This chapter is based on our paper (Shang et al., 2020a) on arxiv.

First, we introduce an attention module (Vaswani et al., 2017) with explicit connections across the decentralized agents. Existing RL works (Zambaldi et al., 2018; Mott et al., 2019; Liu et al., 2019) have adapted similar self-attention modules on top of a single network. In our setup, the agents share a model to generate their actions individually to ensure scalability when the number of agents increases. The proposed attention module is then implemented *across* intermediate features from forward passes of the agents, explicitly connecting them. As we will show in experiments, this leads to the emergence of more complex cooperation strategies as well as better generalization. The closest approach to ours is Multi-Attention-Actor-Critic (MAAC) (Iqbal and Sha, 2018), which also applies a shared encoder across agents and aggregate features for an attention module. However, each agent has its unique critic that takes actions and observations of all agents through the attention features.

Secondly, we develop an unsupervised trajectory predictive task—i.e., without using action labels—for pre-training and/or as an auxiliary task in RL. Observations, without action labels, are often readily available, which is a desirable property for pre-training. Unlike prior works in multi-agent trajectory forecasting (Yeh et al., 2019; Sun et al., 2019), we consider an *agent-centric* version where the location of each agent position is predicted separately. This task encourages the model to reason over an agent's internal states, such as its velocity, intent, etc. We explore two ways to leverage the pre-trained models in RL: (1) as weight initialization and (2) as a frozen column inspired by Progressive Neural Networks (Rusu et al., 2016) (Figure 4.1c). Furthermore, we investigate whether the agent-centric predictive objective serves as a suitable auxiliary loss for MARL. Auxiliary tasks have been used to facilitate RL representation learning in terms of stability and efficiency (Oord et al., 2018; Jaderberg et al., 2016).

Our key contributions are as follows: (1) we introduce an agent-centric attention module for MARL to encourage complex cooperation strategies and generalization (2) we employ an agent-centric predictive task as an auxiliary loss for MARL and/or for pre-training to improve sample efficiency and (3) we assess incorporating agent-centric inductive biases on MARL using the proposed approaches on tasks from Google Research Football and DeepMind Lab 2D.

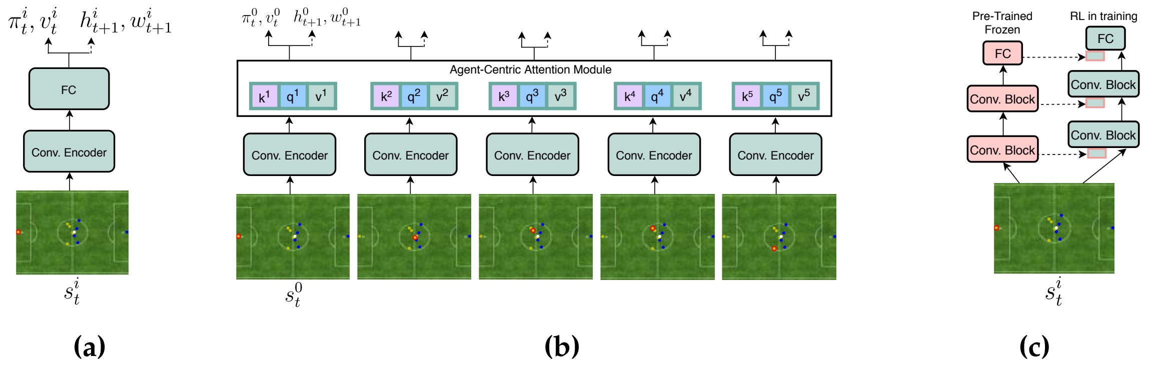


Figure 4.1: (a) MARL CNN baseline, agents share the model but with separate forward passes (b) ACNN with the agent-centric attention module (c) Progressive CNN (PrCNN) with pre-trained model as a frozen column.

4.2 METHODS

Section 4.2.1 describes our problem setup, fully cooperative multi-agent reinforcement learning (MARL), in a policy gradient setting. Section 4.2.2 introduces the agent-centric attention module. Section 4.2.3 motivates the agent-centric prediction objective to learn from observations as an unsupervised pre-training step and/or as an auxiliary loss for MARL.

4.2.1 Multiagent Reinforcement Learning

We consider a group of N agents, denoted by $\mathcal{N}$, operating cooperatively in a shared environment towards a common goal. It can be formalized as a multi-agent Markov decision process (MAMDP) (Boutilier, 1996). A MAMDP is a tuple $(S, \{A^i\}_{i\in\mathcal{N}}, P, \{R^i\}_{i\in\mathcal{N}})$ where S is the shared state space, A^i the set of actions for the i-th agent, $\mathbf{A} = A^1 \times \cdots A^N$, $P : S \times \mathbf{A} \times S \to [0, 1]$ the transition function, and $R : S \times \mathbf{A} \to \mathbb{R}$ the reward function. In our experiments, S are 2D maps of the environments.

We adapt an actor-critic method, V-trace (Espeholt et al., 2018) implemented with SEED RL (Espeholt et al., 2019). The actor and critic are parameterized by deep neural networks. Each agent receives a state input s^i, covering the global $s \in S$ and a specification of the agent's location, and generates its policy $\pi^i = \pi(s^i)$ and state value $V^i = V(s^i)$. The model is shared between agents, adding scalability when the number of agents grows larger and the environment more complex (Iqbal and Sha, 2018; Jiang and Lu, 2018; Jeon et al., 2020). It also

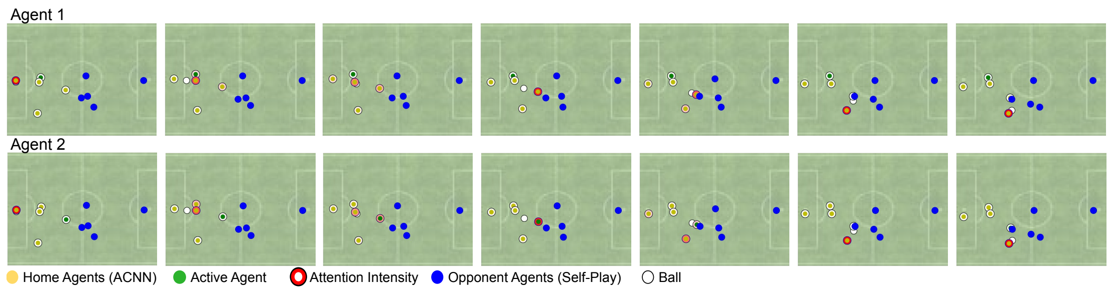


Figure 4.2: Visualization of attention for two different home player agents. Green is the active agent being controlled. Yellow are the other home agents. The red borders signify attention intensity, blue the opponents, and white the ball. Attention is mostly on the agents around the ball, but the two agents' attention weights differ. E.g., in the 5th frame, one looks at the agent possessing the ball, the other at the agent to whom it is passing the ball. Complete video replays of all players are in the Supplementary Materials.

potentially alleviates instability due to the non-stationary nature of multi-agent environments by sharing the same embedding space (Lowe et al., 2017).

The goal for all agents is to maximize the expected long term discounted global return

$$J(\theta) = \mathbb{E}_{s\sim d,\mathbf{a}\sim\pi}\left[\Sigma_{t\geq 0}\gamma^t R(s_t,\mathbf{a})\right] = \mathbb{E}_{s\sim d}\left[V(s)\right], \tag{4.1}$$

where $0 \leq \gamma \leq 1$ is the discount factor, $V(s) = \mathbb{E}_\pi\left[\Sigma_{t\geq t_0}\gamma^t R(s_t,\mathbf{a}_t)|s_0 = s\right]$ the state value, and θ the parameterization for policy and value functions. In a decentralized cooperative setting, (Zhang et al., 2018) proves the policy gradient theorem

$$\nabla_\theta J(\theta) = \mathbb{E}_{d,\pi}\left[\nabla \log \pi^i(s,a^i)Q(s,\mathbf{a})\right] = \mathbb{E}_{d,\pi}\left[\nabla \log \pi(s,\mathbf{a})A(s,\mathbf{a})\right], \tag{4.2}$$

where $\mathbf{a} = (a_1, \cdots a_N)$, $Q(s,\mathbf{a}) = \mathbb{E}_\pi\left[R(s_t,\mathbf{a}_t) + \gamma V(s_{t+1})\right]$ is the state-action value and $A(s,\mathbf{a}) = Q(s,\mathbf{a}) - V(s)$ the advantage for variance reduction. In theory, $V^i \approx V$, and we can directly apply V-trace for each agent with the policy gradient above. In practice, we slightly shape each agent's reward (see Section 4.3), but the policy gradient direction is approximately retained (proof in Appendix 1).

4.2.2 Agent-Centric Attention Module

In the MARL baseline CNN (Figure 4.1a), an agent makes independent decisions without considering the high-level representations from other agents.

This can hinder the formation of more complex cooperation strategies. To address the issue, we propose a novel attention module built upon the multi-head self-attention mechanism (Vaswani et al., 2017) to enable relational reasoning across agents explicitly. As shown in Figure 4.1b and Equation 4.3, the forward pass of each agent produces the key, query, and value independently, which are congregated to output the final features for each agent. These features are then sent to the RL value and policy heads. The attention module allows easy model interpretation (Mott et al., 2019), see Figure 4.2 with a detailed explanation in Section 4.4.3. We term the model ACNN.

Many RL related works for both multi-agent and single-agent setups (Zambaldi et al., 2018; Mott et al., 2019; Liu et al., 2019; Sun et al., 2019; Yeh et al., 2019; Parisotto et al., 2019) that use self-attention usually implement this module on top of a single network. To scale better with multiple agents, many attempts to apply attention across features from independent agents with model sharing, similarly to this work. A multi-agent extension of DDPG (Jiang and Lu, 2018) shares an actor and critic but its attention module is a binary classifier designed to narrow down a subset of agents for communication. The aforementioned MAAC (Iqbal and Sha, 2018), based on soft actor-critic, only shares an encoder across agents to extract features for the attention module. But each agent is equipped with its own critic that takes in all of the outputs from the attention module, essentially covering observations and actions from all agents.

The following summarizes the operations of our attention module:

$$\begin{gathered}
z^i = f_{\text{fc}}(f_{\text{cnn}}(s^i)), q^i = q\left(\text{LN}(z^i)\right), k^i = k\left(\text{LN}(z^i)\right), v^i = v\left(\text{LN}(z^i)\right); \\
K = (k^1 \cdots k^N), Q = (q^1 \cdots q^N), V = (v^1 \cdots v^N), \widetilde{V} = \text{Attn}(Q, K, V) = \sigma(\frac{QK^T}{\sqrt{d_k}})V; \\
\tilde{z}^i = \text{LN}(z^i + \tilde{v}^i) \longrightarrow \pi^i = f_\pi(\tilde{z}^i), v^i = f_v(\tilde{z}^i),
\end{gathered} \tag{4.3}$$

where K, Q, V are the key, query and value as used in the transformer paper (Vaswani et al., 2017). We find a proper placement of Layer Normalization (Ba et al., 2016) within the attention module is crucial.

4.2.3 Multi-agent Observation Prediction

When developing a new skill, humans often extract knowledge by simply observing how others approach the problem. We do not necessarily pair the observations with well-defined granular actions but grasp the general patterns and logic. Observations without action labels are often readily available, such as recordings of historical football matches and team-based FPS gameplay videos. Therefore, in this work, we explore how to transfer knowledge from existing observations to downstream MARL without action labels.

A useful supervision signal from observation is the agent's location. Even when not directly accessible, existing techniques (He et al., 2017; Ren et al., 2015) can extract this information in many cases. We thus adopt an agent's future location as a prediction target. It is worth noting that the same action can lead to different outcomes for different agents depending on their internal states, such as velocity. Therefore, we expect the location predictive objective to provide cues, independent of actions taken, for the model to comprehend an agent's intent and internal states.

Recent works (Yeh et al., 2019; Sun et al., 2019; Zhan et al., 2018) develop models that predict trajectories over all agents at once. We, on the other hand, task each agent to predict its future location, arriving at an agent-centric predictive objective, as illustrated in Figure 4.1a. The motivation for the agent-centric objective is two-fold. For one, as discussed later in this section, the agent-centric loss can be integrated as an auxiliary loss to MARL in a straightforward manner.

Concretely, after collecting observation rollouts, we minimize a prediction loss over the observations instead of maximizing return as in RL training (for details see Section 4.4.2). Our observation is in a 2D map format, and the prediction task for each agent consists of predicting location heatmaps along height σ_h^i and width σ_w^i as softmax vectors. We train these predictions by minimizing the negative log likelihood via cross-entropy loss,

$$\arg\min_{\theta} \mathbb{E}_{\sigma_h^i}\left[-\log \sigma_h^i[h_{t+1}^i]\right], \arg\min_{\theta} \mathbb{E}_{\sigma_w^i}\left[-\log \sigma_w^i[w_{t+1}^i]\right], \qquad (4.4)$$

where h_{t+1}^i and w_{t+1}^i are the ground truth next step locations of the ith agent. The pre-training is applied to both the CNN and ACNN architectures. Next, we investigate two ways to transfer knowledge from the pre-trained models to MARL.

Weight Initialization Transfer via weight initialization is a long-standing method for transfer learning in many domains (Donahue et al., 2013; Devlin et al., 2018). Often, the initialization model is trained on larger related supervised task (Donahue et al., 2013; Carreira and Zisserman, 2017), but unsupervised pre-training (He et al., 2019; Devlin et al., 2018) has also made much progress. In RL, some prior works initialize models with weights trained for another relevant RL task, such as in Chapter 3 but general pre-training through a non-RL objective has been explored less.

Progressive Neural Networks Progressive Neural Networks (Rusu et al., 2016) are originally designed to prevent catastrophic forgetting and enable transfer between RL tasks. They build lateral connections between features from an existing RL model—on a relevant task—to the RL model in training. The setup is also a promising candidate for transferring knowledge from our pre-trained models to MARL. Specifically, a pre-trained model becomes a *frozen column* from which the intermediate features, after a non-linear function, are concatenated to the corresponding activation from the RL model, as shown in Figure 4.1c. We experiment with Progressive Networks combined with CNN and ACNN, called PrCNN and PrACNN.

Multi-agent Observation Prediction as Auxiliary Task for MARL

Auxiliary objectives for single-agent RL in terms of stability, efficiency, and generalization have been widely studied (Oord et al., 2018; Jaderberg et al., 2016). In light of prior works, we assess using the agent-centric prediction objective as an auxiliary task for MARL. Thanks to the convenient formulation of the agent-centric objective, we can simply add prediction heads in juxtaposition with the policy and value heads, as in Figure 4.1a.

4.3 ENVIRONMENTS AND SETUPS

We consider two multi-agent domains, Google Research Football (Kurach et al., 2019) (GRF) and DeepMind Lab 2D (DeepMind, 2020) (DMLab2D), with a focus on the more dynamically complex GRF.

4.3.1 Google Research Football

Google Research Football (Kurach et al., 2019) (GRF) is a FIFA-like environment. Home player agents match against the opponent agents in the game of 5-vs-5 football. All agents are individually controlled except for the goalie, which is operated under the built-in rules by default. Each agent picks 1 out of 19 actions to execute at each step, with 3000 frames per game. We compete against two types of opponent policies: the built-in AI and the self-play AI. The built-in AI is a rule-based policy adapted from (Schuiling, 2017). Its behavioral patterns reflect simple and logical football strategies and can be exploited easily[2] . To add more challenge to training against built-in AI, we also take control of the goalie. The other more robust and generalized opponent policy is trained via self-play(for more details in Table 4.1 and Appendix 3). It requires more advanced cooperative strategies to win against the self-play AI. Our experiments also include ablation studies on a single-agent setup, 11-vs-11 "Hard Stochastic", from (Kurach et al., 2019). In this case, one active player is being controlled at one time.

The home agents are all rewarded $+1$ after scoring and -1 if a goal is conceded. To speed up training, we reward the agent in possession of the ball an additional $+0.1$ if it advances towards the opponent's goal (see full details in (Kurach et al., 2019)).

Observations are in the Super Mini Map (SMM) (Kurach et al., 2019) format. An SMM is a 2D bit map of spatial dimension 72×96, covering the entire football field. It comprises four channels: positions of the home agents, the opponents, the ball, and the active agent. SMMs across four time steps are stacked to convey information such as velocity. We predict an agent's height and width locations on the SMM by outputting 72- and 96-dim heatmap vectors separately when learning from observations.

4.3.2 DeepMind Lab 2D

DeepMind Lab 2D (DMLab2D) (DeepMind, 2020) is a framework supporting multi-agent 2D puzzles. We experiment on the task "Cleanup" (Figure 4.3). At the top of the screen, mud is randomly spawned in the water, and at the bottom, apples spawned at a rate inversely proportional to the amount of mud.

2 video demo: https://www.dropbox.com/s/zofgl382xv9hnff/neurips2020.mp4?dl=0

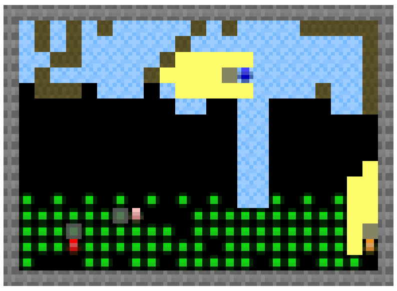


Figure 4.3: DMLab2D "Cleanup"

The agents are rewarded $+1$ for each apple eaten. Four agents must cooperate between cleaning up and eating apples. The list of actions is in Appendix 2. There are 1000 RGB frames per episode with a spatial dimension 72×96. The state input for each agent colors itself blue and the others red. Again, frames across four time steps are stacked for temporal information.

4.4 EXPERIMENTS AND DISCUSSIONS

This section describes our model architectures, implementation details for pre-training from observations, and then for MARL. Next, we dive into analyzing the efficacy of agent-centric representations for MARL. Finally, we conduct two additional ablation studies: comparing agent-centric representation learning against an agent-agnostic alternative on MARL and evaluating agent-centric representation learning on single-agent RL.

4.4.1 Model Architecture

The baseline in our experiments is a Convolutional Neural Network (CNN) followed by a Fully Connected (FC) layer (Figure 4.1a). The policy, value, and height/width predictive heads are immediately after the FC layer. In experiments with the attention module, the FC layer is replaced by a 2-head attention module as illustrated in Figure 4.1 and Equation 4.3. The detailed architectures are adapted from (Espeholt et al., 2019) and attached in the Supplementary Materials.

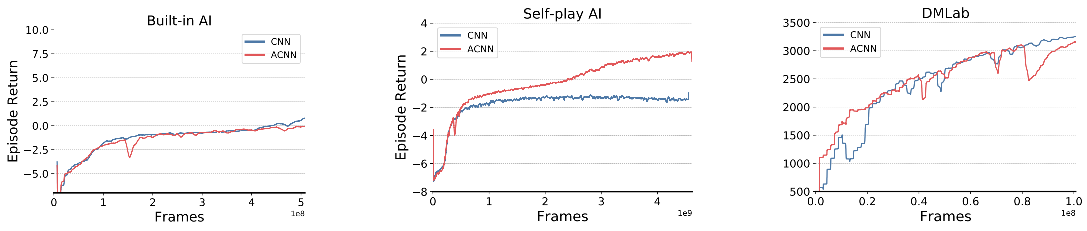


Figure 4.4: Comparison of baseline CNN and ACNN with agent-centric attention. On the simpler DMLab2D and GRF build-in AI, ACNN does not improve over CNN. But on the harder self-play GRF which requires more advanced relational reasoning, attention becomes crucial—without it, the average return does not pass the zero mark. Plots are averaged over 3 random seeds.

4.4.2 Representation Learning from Observations

We collect replays without action labels for the *unsupervised* representation learning from observations. For GRF built-in AI and DMLab2D "Cleanup", we record replays evenly throughout RL training, from the early stage of training to convergence. For GRF self-play, we record checkpoints evenly throughout self-play training (details in Appendix 3) and sample checkpoints to play against each other. We collect approximately 300K frames from each task. In principle, one can utilize *any* reasonable replays. Unlike learning from demonstration, the unsupervised approaches we study do not require action annotations, enabling potential usage of observations for which it would be infeasible to label actions (Schmeckpeper et al., 2019).

We learn two predictive models from observations as described in Section 4.2.3, based on CNN and ACNN. The predictive objectives are optimized via negative log-likelihood (NLL) minimization using Adam (Kingma and Ba, 2014) with default parameters in TensorFlow (Abadi et al., 2016) and a sweep over learning rate.

The batch size is 32, and we train till convergence on the validation set. More training details and NLL results are summarized in Appendix 4. The attention module does not offer a significant advantage in terms of NLL. We conjecture that because each agent's location is predicted independently, the CNN architecture has sufficient model complexity to handle the tasks used in our experiments.

4.4.3 MARL and Main Results

The RL training procedures closely follow SEED RL (Espeholt et al., 2019). When adding the auxiliary loss, we sweep its weighting coefficient over a range of values. See the full set of hyperparameters in Appendix 5. Figures 4.4, 4.5 and 4.6 show plots of our core results, where the horizontal axis is the number of frames and the vertical axis the episode return. We train on 500M frames for GRF built-in AI, 4.5G for self-play AI, and 100M for DMLab2D. All plots are averaged over 3 random seeds of the best performing set of hyperparameters. The rest of this section examines and discusses the experiments.

Attention is essential to form complex strategies among agents. Figure 4.4 compares the baseline CNN with the attention-based ACNN, all trained from random initialization. Although performing comparably on the simpler tasks DMLab2D "Cleanup" and GRF built-in AI, ACNN clearly outperforms CNN when tackling the much more challenging self-play AI in GRF. By examining the replays against the built-in AI and self-play AI (attached in the Supplementary Materials), we find it is possible to exploit the built-in AI with elementary tactics. In other words, **the most critical phase in GRF** is when the agent transitions from passively defending to actively scoring, i.e. when the scores cross the 0 mark. This transition signifies the agent's understanding of the opponent's offensive as well as defensive policy. After the transition, the agent may excessively exploit the opponent's weaknesses. For example, a flaw of built-in AI is that it can easily be tricked into an offside position (see the demo video) and thus CNN with auxiliary loss scoring more than ACNN with auxiliary loss in absolute term merely means the former is good at exploiting this weakness. On the other hand, when against self-play AI, ACNN can pass the 0 mark to start winning, while none of the CNN models are able to achieve so. This strongly indicates that the attention module plays an essential role in providing the reasoning capacity to form complex cooperative strategies. That is, it demands more sophisticated play to perform well against the more general and robust self-play policies. It indicates the attention module plays an essential role in forming cooperation strategies complex enough to allow winning against the self-play AI.

Figure 4.2 takes two players from ACNN trained against the self-play AI and visualize their attention patterns. Green dots are the active player agents, and yellow ones are the other agents on the home team. We visualize the attention

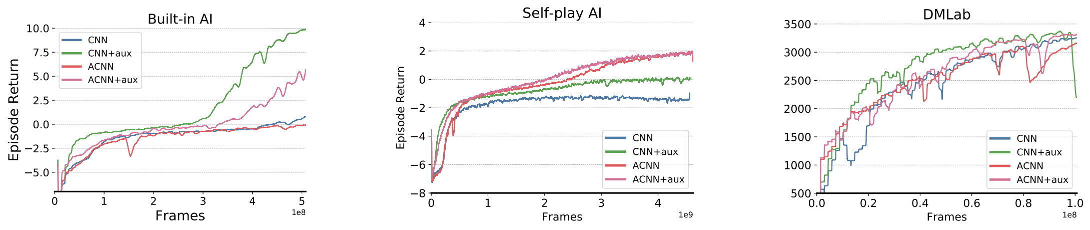


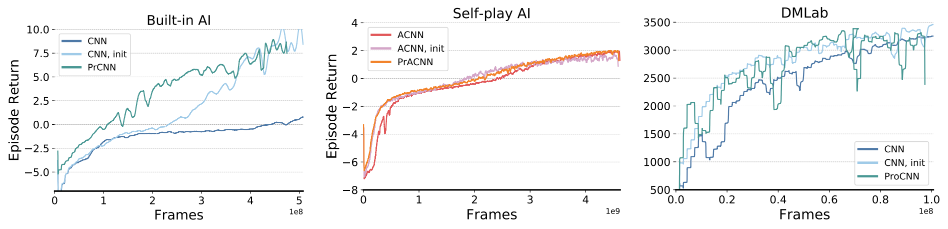
Figure 4.5: Comparison of models trained without and with the auxiliary loss. On both GRF tasks, the auxiliary loss improves performance and sample efficiency. Plots are averaged over 3 random seeds.

weights from one of the two heads. The intensity of the red circles surrounding the home agents reflects the weights. The most-watched area is in the vicinity of the ball. E.g., in the 5th frame, one agent (top row) focuses on the player in possession of the ball, whereas the other agent (bottom row) is looking at the player to whom it is passing the ball.

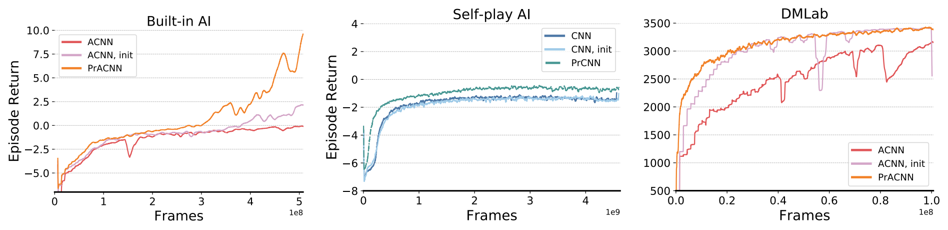
Finally, note that for two out of three tasks in our experiments, namely self-play GRF and DMLab2D, PrACNN are the best performing models (Figure). Particularly for DMLab2D, PrACNNconverges significantly faster and more stably than its CNN counterpart without the attention module.

The agent-centric auxiliary loss complements MARL. Figure 4.5 compares the effects of adding the agent-centric auxiliary loss as described in Section 4.2.3 to the CNN and ACNN model, both trained from random initialization. As RL is sensitive to tuning, an incompatible auxiliary loss can hinder its training. In our experiments, however, the auxiliary loss for the most part improves the models' performance and efficiency, particularly for the CNN models. This suggests the agent-centric loss is supportive of the reward structure in the game.

Unsupervised pre-training improves sample-efficiency. We compare training the CNN (Figure 4.6a) and ACNN (Figure 4.6b) from scratch with the two ways of integrating unsupervised pre-training to MARL, namely weight initialization and progressive neural networks. For a fair comparison, we use the same hyperparameter tuning protocol from baseline training to tune models involving pre-training. Both integration methods provide significant improvements to sample-efficiency, especially on the simpler DMLab2D and GRF with the built-in AI. In some cases, progressive models can achieve better performance and efficiency than those with weight initialization. Even when the impact of pre-training is limited, it does not hurt the performance of MARL. Hence in practice, it can be beneficial to perform a simple pre-training step



(a)



(b)

Figure 4.6: Comparison of training from scratch, using pre-trained models as initialization and as a frozen column in progressive networks, on top of (a) CNN and (b) ACNN models. Initialization and progressive networks show better or comparable performance as well as sample efficiency, with more evident effects on the simpler tasks, built-in AI and DMLab2D. Plots are averaged over 3 random seeds.

with existing observation recordings to speed up downstream MARL. We repeat the same control experiments on RL models trained with the auxiliary loss in Appendix 5. The same trend is observed, albeit to a lesser degree, as expected because the auxiliary objectives and pre-training carry overlapping information.

4.4.4 Agent-Centric Representation Learning for Single-Agent RL

We have demonstrated the efficacy of agent-centric representation learning for multi-agent RL. To evaluate whether similar conclusions hold for single-agent RL, we use the 11-vs-11 "Hard Stochastic" task from (Kurach et al., 2019), where only one player is controlled a time (similarly to FIFA).

Although the game logic has much in common between single and multiple players, the experimental outcome is different, as in Figure 4.7. The attention module brings down baseline performance, likely because cooperation matters less here—the rest of the home players are controlled by the built-in AI—, and the attention module is harder to optimize. The agent-centric prediction task still helps as auxiliary loss or for pre-training, but to a limited extent.

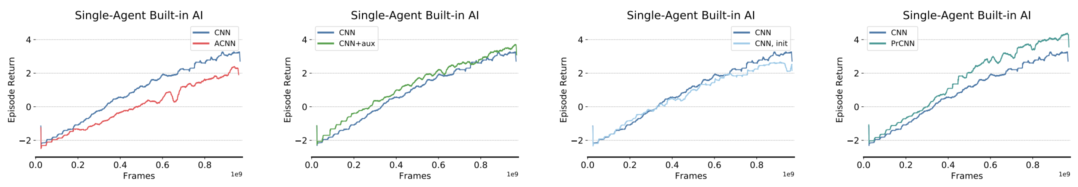


Figure 4.7: Results of incorporating agent-centric inductive biases to *single-agent RL* (GRF 11-vs-11 Hard Stochastic). The attention module is no longer optimal as co-operation matters less for single-agent RL. The auxiliary loss and pre-training from observations still help albeit to a lesser degree.

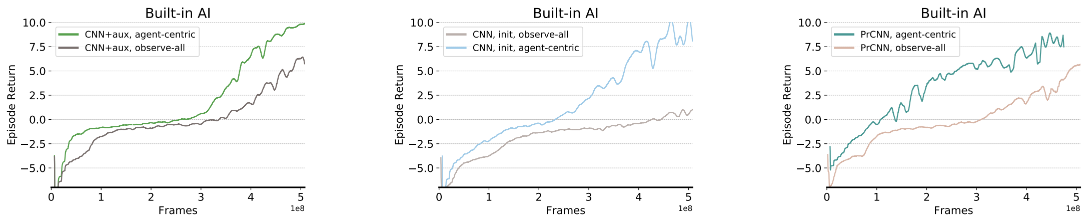


Figure 4.8: Comparison between the agent-centric objective and the observe-all objective that predicts for all agents' location at once for MARL on GRF Built-in AI. The former exhibits clear advantage.

4.4.5 Agent-agnostic Observe-All Representation Learning for MARL

Finally, to verify the necessity for MARL representations to be *agent-centric*, we implement an alternative agent-agnostic observation prediction objective, referred to as *observe-all*, and test with playing against the GRF built-in AI. Concretely, the height and width predictive heads for observe-all output 72− and 96− dimension binary masks respectively, where 1 indicates player occupation and 0 vacancy, taking over *all agents*. In this way, the prediction is agnostic of agent identity and can predict for all players at once. First, we use the observe-all objective as an auxiliary loss to RL and train from scratch. Next, we apply the observe-all objective to pre-training from observations, which is then used for MARL, either via weight initialization or as a progressive frozen column. Figure 4.8 clearly shows the agent-centric auxiliary loss is more competitive than the observe-all one in all setups. It confirms that the agent-centric nature of the prediction task is indeed important for MARL.

	CNN	+aux.	+init.	+prgs.	ACNN	+aux.	+init.	+prgs.	rating
CNN	╲	5/8/7	9/1/10	7/5/8	4/4/12	5/2/13	2/7/11	3/5/12	−323
+aux.	7/8/5	╲	6/4/10	6/1/13	3/4/13	2/4/14	2/8/10	2/5/13	−420
+init.	10/1/9	10/4/6	╲	6/5/9	5/5/10	2/5/13	3/6/11	4/6/10	−283
+prgs.	8/5/7	13/1/6	9/5/6	╲	2/8/10	2/4/14	5/1/14	2/5/13	−322
ACNN	12/4/4	13/4/3	10/5/5	10/8/2	╲	8/4/8	7/7/6	10/6/4	**147**
+aux.	13/2/5	14/4/2	13/5/2	14/4/2	8/4/8	╲	11/3/6	7/5/8	**182**
+init	11/7/2	10/8/2	11/6/3	14/1/5	6/7/7	6/3/11	╲	8/4/8	24
+prgs.	12/5/3	13/2/5	10/6/4	13/5/2	4/6/10	8/5/7	8/4/8	╲	35

GRF Leaderboard Results

Agent	Opponent	Rating
ACNN+aux	CNN-v1/v2	**1992**
ACNN	CNN-v1/v2	1841
CNN-v1	itself	1659
CNN-v2	itself	1630
CNN+aux	Built-in	1048
Built-in	NA	1000

Table 4.1: Left: selected agents play 20 matches among each other, all trained against the self-play AI. Each entry records win/tie/loss between row agent and column agent. Ratings are estimated ELO scores. ACNNs shows clear superiority over CNNs. ACNN trained from scratch generalizes better. **Right:** GRF Multi-Agent public leaderboard results by the time of paper submission. Opponent refers to the policy the agent trained against. Each submitted agent plays 300 games. ACNNs trained against self-play AI perform the best. The CNN trained against built-in AI performs poorly and does not generalize. The self-play AI used in our experiments CNN-v1/v2 are clearly superior to the built-in AI.

4.5 GRF AGENT TOURNAMENT AND PUBLIC LEADERBOARD

Table 4.1 (Left) investigates how various agent-centric representation learning components generalize by hosting a tournament among selected agents. We include agents trained from scratch, from scratch plus auxiliary loss, from initialization plus auxiliary loss, and with progressive column plus auxiliary loss. All are trained against self-play AI for a total of 4.5 billion frames, and each play 20 matches against another. Each entry records win/tie/loss between row agent (home) and column agent (opponent). We also estimate their ELO ratings (for details, see Appendix 6). Clearly, ACNN based models outperform CNN models, corroborating the claim that the agent-centric attention module enhances generalization. Meanwhile, the ACNN models using pre-training are inferior to the ones trained from scratch. It suggests that although pre-training speeds up convergence, it can also limit the model's ability to generalize.

Finally, we upload our best performing agent trained against the built-in AI, i.e., CNN with auxiliary loss, and best agents against the self-play AI, i.e., ACNN and ACNN with auxiliary loss, to the public GRF Multi-agent League (Zurich, 2020). For comparison, we also upload the two self-play AI models used in our experiments, i.e., CNN-v1 and CNN-v2. Each of the submitted agents plays 300 games against agents submitted by other participants, and their ELO ratings are listed in Table 4.1 (Right). The self-play AI agents deliver a decent performance, showing clear advantages over the built-in AI. The agents trained against the self-play AI overall perform the best. In contrast, the agent trained against the built-in AI, although dominating the built-in AI, is hugely fragile against other agents. This supports our observation that the built-in AI can be exploited with little cooperation. It is also worth mentioning that, at the time of submission, our ACNN agent with auxiliary loss ranks top 1 on the leaderboard.

4.6 CONCLUSIONS

We propose integrating novel agent-centric representation learning components, namely the agent-centric attention module and the agent-centric predictive objective, to multi-agent RL. In experiments, we show that the attention module leads to complex cooperative strategies and better generalization. In addition, leveraging the agent-centric predictive objectives as an auxiliary loss and/or for unsupervised pre-training from observations improves sample efficiency.

CHAPTER APPENDIX

4.A POLICY GRADIENT FOR MAMDP

Recall that we consider a fully cooperative multi-agent MDP with N agents and finite horizon with a maximum of $t_{\max}$ steps. A MAMDP is a tuple $(S, \{A^i\}_{i \in \mathcal{N}}, P, \{R^i\}_{i \in \mathcal{N}})$ where S is the shared state space, A^i the set of actions for the i-th agent and $\mathbf{A} = A^1 \times \cdots A^N$, $P : S \times \mathbf{A} \times S \to [0, 1]$ the transition function, and $R : S \times \mathbf{A} \to \mathbb{R}$ the reward function. For a policy $\pi : S \times \mathbf{A}$, the respective state value function is defined as

$$V(s) = \mathbb{E}_\pi \left[\Sigma_{t \geq t_0} \gamma^t R(s_t, \mathbf{a}_t) | s_0 = s \right], \tag{4.5}$$

or in the Bellman form

$$V(s) = \mathbb{E}_\pi \left[R(s_0, \mathbf{a}_0) + \gamma V(s_1) | s_0 = s \right],$$

where $0 \leq \gamma < 1$ is the discount factor over time. The respective state-action value function can also be defined in Monte Carlo, Bellman forms as well as in terms of V

$$Q(s, \mathbf{a}) = \mathbb{E}_\pi \left[\sum_{t_0}^{t_{\text{end}}} \gamma^t R(s_t, \mathbf{a}_t) | s_0 = s, a_0 = \mathbf{a} \right], \tag{4.6}$$

$$Q(s, \mathbf{a}) = \mathbb{E}_\pi \left[R(s_t, \mathbf{a}_t) + \gamma \mathbb{E}_\pi \left[Q(s_{t+1}, \mathbf{a}_{t+1}) \right] \right] = \mathbb{E}_\pi \left[R(s_t, \mathbf{a}_t) + \gamma V(s_{t+1}) \right].$$

The common goal for all agents is to identify policy π that maximizes the expected long term discounted global return

$$J(\theta) = \mathbb{E}_{s \sim d} \left[V(s) \right] = \mathbb{E}_{s \sim d, \mathbf{a} \sim \pi} \left[\Sigma_{t \geq 0} \gamma^t R(s_t, \mathbf{a}) \right], \tag{4.7}$$

where d is the stationary distribution of states.

A popular method in tackling this task is via policy gradient (Williams, 1992) by directly ascending gradients of π along the direction $\nabla_\theta J$. The single-agent policy gradient theorem rewrites $\nabla_\theta J$ in the following form

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim d, a \sim \pi} \left[\nabla \log \pi(s, a) Q(s, a) \right].$$

Often Q is also approximated by a parameterized function, leading to actor-critic algorithms (Konda and Tsitsiklis, 2000). To reduce variance, a baseline is usually subtracted from Q. One candidate for baseline, used in V-trace (Espeholt et al., 2018), is the state-value V, resulting in the advantage actor-critic algorithm (Mnih et al., 2016a), that is

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim d, a\sim\pi}\left[\nabla \log \pi(s,a) A(s,a)\right], \text{ where } A(s,a) = Q(s,a) - V(s).$$

In the case of fully-cooperative MARL, Zhang et al. (2018) proves the corresponding policy gradient theorem, following the original proof in Williams (1992)

Theorem 1. (Policy Gradient Theorem for MARL.) *For any $\theta \in \Theta$ be parametrizations for actor and critic. Let $\pi^i : S \times A^i$ be a policy for i-th agent, let $J(\theta)$ be the global discounted long term return defined in 4.7, Q and V defined in 4.6 and 4.5. The gradient of $J(\theta)$ with respect to θ^i, the parametrization for i-th agent, is given by*

$$\begin{aligned} \nabla_\theta J(\theta) &= \mathbb{E}_{s\sim d, \mathbf{a}\sim\pi}\left[\nabla \log \pi^i(a^i|s) Q(s,\mathbf{a})\right] && (4.8)\\ &= \mathbb{E}_{s\sim d, \mathbf{a}\sim\pi}\left[\nabla \log \pi^i(a^i|s)\left(Q(s,\mathbf{a}) - V(s)\right)\right]. && (4.9)\end{aligned}$$

Our setup closely follows that in Theorem 1. We parameterize π and V using deep neural networks, denoted by θ. The state-action Q in practice is thus derived through $Q(s,\mathbf{a}) = R(s_t,\mathbf{a}_t) + \gamma V(s_{t+1})$. All agents share the same π and v. In our case, the policy for each agent is differentiated by adding additional agent specific information to the input state s^i. As we update the gradients for all agents altogether, it does not affect the policy gradient formulation in practice.

Since all agents share a common global reward, ideally $V(s_i) = V(s_j) = V(s)$, and denote $V^i(s) = V(s_i)$. We add a small amount of additional reward for faster convergence and ease of training if an agent is explicitly making positive contributions. For example, an active player, i.e., the player in control of the ball, receives 0.1 if advancing to the opponent's half of the field. However, this slight reshaping of reward does not deviate the empirical $\nabla\hat{J}$ too far away from the true ∇J.

Proposition 1. *Consider for each agent, when taking action, it receives a reward*

$$R^i(s_t,\mathbf{a}_t) = R(s_t,\mathbf{a}_t) + \beta_t\epsilon,$$

where $\beta_t \sim \text{Bernoulli}(p)$ *is i.i.d and* $\epsilon > 0$. *Assume* V^i *converges, that is* $V^i(s_t) = \mathbb{E}_\pi\left[\Sigma_{t \geq t_0} \gamma^t R^i(s_t, \mathbf{a}_t)\right]$, *then we have*

$$|Q(s_t, \mathbf{a}_t) - Q^i(s_t, \mathbf{a}_t)| = |V(s_t) - V^i(s_t)| \leq p\epsilon \frac{1 - \gamma^{t_{\max}}}{1 - \gamma},$$

and

$$\|\nabla_\theta J(\theta) - \nabla_\theta \hat{J}(\theta)\| \leq p\epsilon \frac{1 - \gamma^{t_{\max}}}{1 - \gamma} \|\mathbb{E}_{s \sim d, a \sim \pi}\left[\nabla \log \pi^i(a|s)\right]\|$$

Proof. We start with the value functions

$$\begin{aligned} V^i(s_t) - V(s_t) &= \mathbb{E}_\pi\left[\Sigma_{t \geq t_0} \gamma^t R^i(s_t, \mathbf{a}_t)\right] - \mathbb{E}_\pi\left[\Sigma_{t \geq t_0} \gamma^t R(s_t, \mathbf{a}_t)\right] \\ &= \mathbb{E}_\pi\left[\Sigma_{t \geq t_0} \gamma^t \left(R(s_t, \mathbf{a}_t) + \beta_t \epsilon\right) - \Sigma_{t \geq t_0} \gamma^t R(s_t, \mathbf{a}_t)\right] \\ &= \mathbb{E}_\pi\left[\Sigma_{t \geq t_0} \gamma^t \beta_t \epsilon\right] \\ &= \Sigma_{t \geq t_0} \gamma^t p\epsilon. \end{aligned}$$

Therefore

$$|V^i(s_t) - V(s_t)| \leq \Sigma_{t \geq t_0} \gamma^t p\epsilon = \frac{p\epsilon(1 - \gamma^{t_{\text{end}} - t_0})}{1 - \gamma} \leq p\epsilon \frac{1 - \gamma^{t_{\max}}}{1 - \gamma}.$$

Since we have

$$\begin{aligned} Q^i(s_t, \mathbf{a}_t) - Q(s_t, \mathbf{a}_t) &= \mathbb{E}_\pi\left[R^i(s_t, \mathbf{a}_t) + \gamma V^i(s_{t+1}) - R(s_t, \mathbf{a}_t) - \gamma V(s_{t+1})\right] \\ &= p\epsilon + \gamma(V^i(s_{t+1}) - V(s_{t+1})) = V^i(s_t) - V(s_t), \end{aligned}$$

from there

$$|Q^i(s_t, \mathbf{a}_t) - Q(s_t, \mathbf{a}_t)| \leq p\epsilon \frac{1 - \gamma^{t_{\max}}}{1 - \gamma}.$$

Now turn to the policy gradients

$$\begin{aligned} \nabla_\theta J(\theta) - \nabla_\theta \hat{J}(\theta) &= \mathbb{E}_{s \sim d, a \sim \pi}\left[\nabla \log \pi^i(a|s) Q(s, \mathbf{a})\right] - \mathbb{E}_{s \sim d, a \sim \pi}\left[\nabla \log \pi^i(a|s) Q^i(s, \mathbf{a})\right] \\ &= \mathbb{E}_{s \sim d, a \sim \pi}\left[\nabla \log \pi^i(a|s) \left(Q(s, \mathbf{a}) - Q^i(s, \mathbf{a})\right)\right] \end{aligned}$$

therefore

$$\begin{aligned} \|\nabla_\theta J(\theta) - \nabla_\theta \hat{J}(\theta)\| &= \|\mathbb{E}_{s \sim d, a \sim \pi}\left[\nabla \log \pi^i(a|s) \left(Q(s, \mathbf{a}) - Q^i(s, \mathbf{a})\right)\right]\| \\ &= \|\mathbb{E}_{s \sim d, a \sim \pi}\left[\nabla \log \pi^i(a|s) \Sigma_{t \geq t_0} \gamma^t \beta_t \epsilon\right]\| \\ &= \|\mathbb{E}_{s \sim d, a \sim \pi}\left[\nabla \log \pi^i(a|s) p\epsilon \frac{1 - \gamma^{t_{\text{end}}}}{1 - \gamma}\right]\| \\ &= p\epsilon \frac{1 - \gamma^{t_{\text{end}}}}{1 - \gamma} \|\mathbb{E}_{s \sim d, a \sim \pi}\left[\nabla \log \pi^i(a|s)\right]\| \\ &\leq p\epsilon \frac{1 - \gamma^{t_{\max}}}{1 - \gamma} \|\mathbb{E}_{s \sim d, a \sim \pi}\left[\nabla \log \pi^i(a|s)\right]\|. \end{aligned}$$

□

4.B ACTION LISTS

GOOGLE RESEARCH FOOTBALL There are in total 19 actions, which are: idle, left, top left, top, top right, right, bottom right, bottom, bottom left, long pass, high pass, short pass, shot, keeper rush, sliding, pressure, team pressure, switch, sprint, dribble, release direction, release long pass, release high pass, short pass, release shot, release keeper rush, release sliding, release pressure, release team pressure, release switch, release spring, release dribble.

Top	Bottom	Left	Right
Top-Left	Top-Right	Bottom-Left	Bottom-Right
Short Pass	High Pass	Long Pass	Shot
Do-Nothing	Sliding	Dribble	Stop-Dribble
Sprint	Stop-Moving	Stop-Sprint	—

DEEPMIND LAB 2D "CLEANUP" In Cleanup, there are 4 categories of action. At each step, an agent chooses one act from each. The categories are move (up, down, right, left, idle), turning (left, right, idle), firing cleaning laser (fire, no fire), firing fine laser (fire, no fire). Thus in total, there are 60 combinations. The leaning laser is for cleaning up the mud. The fine laser is purposed to freeze other agents when they become too greedy and keep consuming apples without cleaning.

4.C GRF SELF-PLAY TRAINING

Self-play follows the same general training protocol as the non-self-play MARL. The biggest differences are (1) the opponent policy and (2) policy gradient updates on the opponents. For (1), the opponent policies are a mix of the current model and the built-in AI. Concretely, we control a subset of opponent player agents using the training policy and the other player agents using built-in AI. The two self-play policies on the GRF Leaderboard (Zurich, 2020), *smarl:b3* (CNN-v1 from Section 5) and *smarl:b2* (CNN-v2 from Section 5) are trained with controlling 2 and 1 agents with the current training policy respectively. For (2), we calculate the policy gradients for the opponent players controlled by the model in training in the same way as home players and backpropagate the gradients as well.

4.D MORE DETAILS AND RESULTS ON MULTI-AGENT OBSERVATION PREDICTION

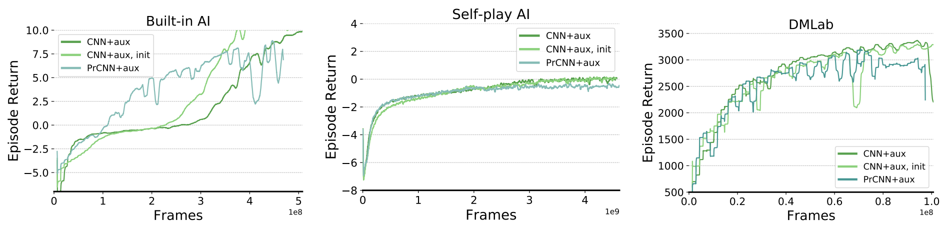
The input format for multi-agent observation prediction is exactly the same as used for RL. This consistency allows straightforward transfer of the pre-trained model to MARL. The Adam optimizer has parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-07$ and initial learning rate 0.002. For validation, we set aside 1/15 of the data and train till NLL plateau on the validation data. The results are summarized in Table 4.D.1. The attention module does not offer significant advantage in terms of NLL. We conjecture that because each agent's location is predicted independently, the CNN architecture has sufficient model complexity to handle the tasks used in our experiments.

	DMLab		built-in		self-play	
CNN	4.81	5.18	8.11	8.06	7.70	7.98
ACNN	4.56	4.80	8.22	7.91	7.74	7.92

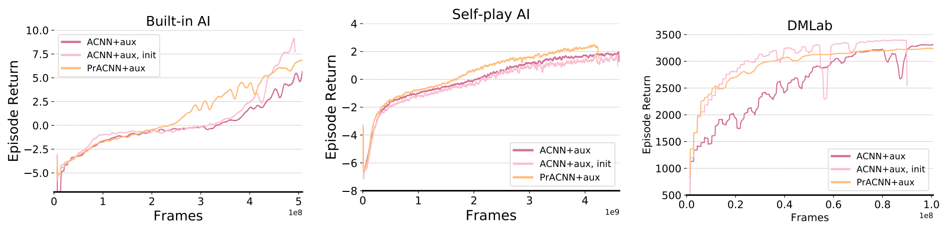
Table 4.D.1: Negative log likelihood results for CNN, ACNN on observations from the 3 tasks.

4.E MORE DETAILS AND RESULTS ON MULTI-AGENT RL

For all MARL experiments, we sweep the initial learning rate over the set of values $(0.00007, 0.00014, 0.00028)$, for the auxiliary loss coefficient $(0.0001, 0.0005)$. The loss coefficient for value approximation is 0.5. We use TensorFlow Abadi et al., 2016 default values for other parameters in ADAM. The unroll length is 32, discounting factor 0.99, entropy coefficient in V-trace 0.0005. The multi-agent experiments for GRF are run on 16 TPU cores, 10 of which are for training and 6 for inference, and 2400 CPU cores for actors with 18 actors per core. The batch size is 120. We train 500M frames when against the built-in AI and 4.5G frames when against the built-in AI. The single-agent experiments for GRF are run on 32 TPU cores, 20 of which for training and the rest for inference, and



(a) Comparison of CNN, CNN using initialization, and PrCNN when trained with auxiliary loss



(b) Comparison of ACNN, ACNN using initialization, and PrACNN when trained with auxiliary loss

Figure 4.E.1: Plots are averaged over 3 random seeds.

480 CPU cores for actors with 18 actors per core. The batch size is 160. We train 900M frames for the single-agent RL tasks. The DMLab2D experiments are run on 8 TPU cores, 5 of which for training the other for inference, and 1472 CPU cores with 18 actors per core.

When analyzing the effects of incorporating pre-trained models to MARL, besides the results in Section 4 of the main text, we also conduct experiments by adding initialization or progressive frozen column to CNN and ACNN models with auxiliary loss in Figure 4.E.1. The same trend is observed, albeit to a lesser degree, as expected because the auxiliary objectives and pre-training carry overlapping information.

4.F ELO

The ELO rating system is designed to compare a group of players in zero-sum games. In our tournament (Table 1 Left), we compute the ELO scores among the selected agents in the following way.

1. Initialize all agents to have an ELO score 1000.
2. Shuffle the order of the matches taken place.

3. Go through the matches and update the ELO score for the two players A and B in two steps. First, calculate the expected scores based on their current ELO ratings R_A and R_b:

$$E_A = \frac{1}{1 + 10^{(R_B - R_A)/400}}, \quad E_B = \frac{1}{1 + 10^{(R_A - R_B)/400}}.$$

Then update the ELO ratings based on the expected scores and the actual score, where winning is 1, losing -1, and tie 0:

$$R_A = R_A + k(S_A - E_A), \quad R_B = R_B + k(S_B - E_B),$$

where k is the k-factor for ELO, which we set to be 16.

Part II

Crafting Deep Learning Models for Computer Vision Applications

MOTIVATION AND SUMMARY

Computer vision is fundamental to AI, as it is one of the dominating venues for the digital world to interact with the physical world. Computer vision tools can not only be directly useful in applications such as image recognition, activity recognition, face verification, etc., but also serve as a crucial basis for much larger scale problems such as autonomous driving, robotic arm manipulation, virtual reality applications, etc. Recently, there has been tremendous progress in computer vision research, thanks to the rise of deep learning. It has achieved human-level performance in many supervised learning tasks, such as image recognition. Meanwhile, the physical world provides us with a wealth of unlabeled data. Therefore, it has been a central research challenge to better craft deep computer vision representations from data that is either unlabeled or with limited labels.

Part II investigates how to enhance unsupervised image modeling, conditional image modeling, and unsupervised video representation learning.

In Chapter 5, we propose ***channel-recurrent variational autoencoders*** (crVAE (Shang et al., 2018) for image modeling. It integrates recurrent connections across channels to both inference and generation steps, allowing the high-level features to capture global-to-local, coarse-to-fine content. Combined with adversarial loss, the resulting channel-recurrent VAE-GAN (crVAE-GAN) outperforms the baseline VAE-GAN in generating a diverse spectrum of high-resolution images while maintaining the same level of computational efficacy.

In Chapter 6, we further extend the channel-recurrent framework and propose ***attentive conditional channel-recurrent autoencoding*** (acVAE) (Shang and Sohn, 2019) for attribute-conditioned face synthesis. Evaluations are performed through both qualitative visual examination and quantitative metrics, namely inception scores, human preferences, and attribute classification accuracy.

Finally, in Chapter 7, we propose ***to learn video level statics and dynamics representations*** (Shang et al., 2020b) by decomposing videos from temporal coherence and dynamics. We demonstrate the significance of the learned representations over several applications, including a novel dynamics retrieval task, on a face, and a human activity datasets.

5

CHANNEL-RECURRENT AUTOENCODING FOR IMAGE MODELING

5.1 INTRODUCTION

Tremendous progress has been made in generative image modeling in recent years. Autoregressive models, such as PixelRNN (Oord et al., 2016a), describe image densities autoregressively in the pixel level, leading to sharp generations, albeit at a high computational cost and without providing latent representations. Generative adversarial networks (GANs) (Goodfellow et al., 2014) have shown promise, but are limited to modeling high-density regions of data distributions (Arora et al., 2017; Wu et al., 2016; Theis et al., 2016) and difficult to train (Radford et al., 2016; Arjovsky et al., 2017).

Variational Autoencoder (VAE) (Kingma and Welling, 2013) is a directed graphical model that approximates a data distribution through a variational lower bound (VLB) of its log-likelihood. VAE introduces an approximate posterior parameterized by a deep neural network (DNN) that probabilistically

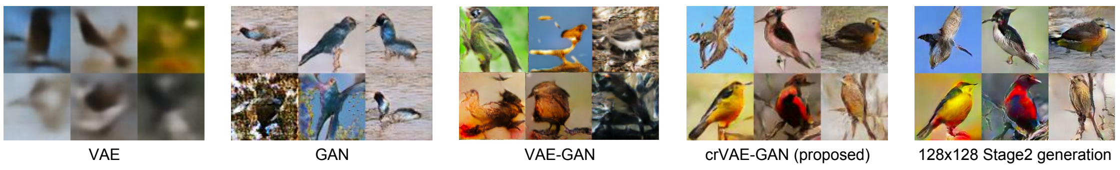


Figure 5.1: Comparison demonstrating our channel-recurrent VAE-GAN's superior ability to model complex bird images. Based on the high-quality generation of Stage1 64×64 images, higher-resolution Stage2 images can be further synthesized unsupervisedly.

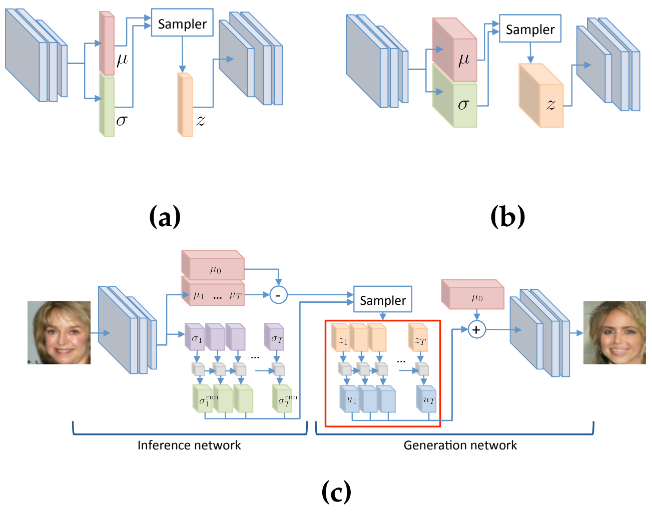


Figure 5.2: Illustrations of (a) standard VAE, (b) its convolutional variant cVAE and (c) the proposed crVAE.

encodes the input to a latent representation. It is directly applicable for a wide range of downstream tasks from photo-editing (Guo et al., 2017) to policy learning (Higgins et al., 2017b), which neither GANs nor autoregressive models are equipped with. Inference in VAE can be made efficiently by a forward pass of the DNN, making it promising for real-time applications. As opposed to GANs, the reconstruction objective of VAE assures a comprehensive input space mode coverage. However, such comprehensive mode coverage, when combined with the KL regularization to the approximated posterior, becomes a downside for modeling complex and high-dimensional images, resulting in blurry image generation (Bousquet et al., 2017). To resolve blurriness while preserving a meaningful latent space, (Larsen et al., 2016) augments a VAE with an auxiliary adversarial loss, obtaining VAE-GAN.

However, recent works for high resolution (64×64 and above) unsupervised image modeling are restricted to images such as faces and bedrooms, whose intrinsic degrees of freedom are low (Larsen et al., 2016; Cha, 2017; Mescheder et al., 2017). Once images become spatially and contextually complex, e.g. birds photographed in their natural habitats, aforementioned models struggle to produce sensible outputs (Figure 5.1), likely due to their latent space constructions lacking the capacity to represent complex input distributions.

In this work, we aim at resolving the limitations of VAE, such as blurry image generation and lack of expressiveness to model complex input spaces, in an unsupervised way[1]. While keeping graphical model unchanged to retain its original efficacy such as efficient probabilistic inference and generation, we propose to augment the architecture of inference and generation networks via recurrent connections across channels of convolutional features, leading to the *channel-recurrent VAE* (crVAE). Our approach is motivated by observing a common drawback to VAE and VAE-GAN: the fully-connected (FC) layers between the latent space and convolutional encoder/decoder. Although FC layers can extract abstract information for high-level tasks such as recognition (Krizhevsky et al., 2012), it omits much local descriptions that are essential for detailed image modeling as in our case. Instead, we build latent features on convolutional activation without FC layers. The proposed architecture sequentially feed groups of convolutional features sliced across channels into an LSTM (Hochreiter and Schmidhuber, 1997), so that for each time step, the associated latent channels are processed based upon accumulated information from previous time steps to ensure temporal coherence while reducing the redundancy and rigidity from FC layers by representing distinguishing information at different time steps. As a result, our model disentangles factors of variation by assigning general outlining to early time steps and refinements to later time steps. Analogously to VAE-GAN, We derive crVAE-GAN by adding an additional adversarial loss, along with two novel regularization methods to assist training further.

We evaluate the performance of our crVAE-GAN in generative image modeling of a variety of objects and scenes, namely birds (Van Horn et al., 2015; Berg et al., 2014; Wah et al., 2011), faces (Liu et al., 2015), and bedrooms (Yu et al., 2015). We demonstrate the superiority of our crVAE-GAN qualitatively via 64×64 image generation, completion, and an analysis of semantic contents for blocks of latent channels, as well as quantitatively via inception scores (Salimans et al., 2016) and human evaluation. Specifically, significant visual enhancement is observed on the more spatially and contextually complex birds dataset. We provide further empirical evidence through higher-resolution (128×128 or 224×224) image synthesis by stacking an extra generation network on top of 64×64 generations from crVAE-GAN and VAE-GAN, similarly to (Zhang et al., 2016a). Unlike (Zhang et al., 2016a), the success of generating higher-resolution 2nd stage images without condition variables is heavily dependent on the quality of the 1st stage generations. Our results verify the importance

1 This chapter is based on our WACV 2018 paper (Shang et al., 2018).

of channel-recurrent architecture in providing a solid 1st stage foundation to achieve high-quality 2nd stage generation. Lastly, we remark on the computational virtues of crVAE-GAN.

The merits of our method are summarized as follows:

- We integrate temporal structure to the latent space via LSTMs to replace the rigid FC layers to recurrently process latent channels, attaining a global-to-local, coarse-to-fine generation.
- Our framework not only preserves the beneficial probabilistic latent space from VAE, allowing wide mode coverage, efficient posterior inference, and training but improves its expressiveness and interpretability.
- Our crVAE-GAN, combined with two novel regularization methods, can model complex input spaces when existing models fail. We visually and quantitatively demonstrate significant improvement in high-resolution image generation and related tasks over VAE-GAN.
- Our model, while producing state-of-the-art level image generations, maintains the computational efficacy from VAE.

Code and pretrained models are published.

5.2 RELATED WORKS

Recent advances in deep generative modeling predominantly come from autoregressive models, Generative Adversarial Networks (GANs), and Variational Autoencoders (VAEs). Autoregressive models such as PixelRNN and PixelCNN (Oord et al., 2016b; Oord et al., 2016a; Salimans et al., 2017) directly characterize the probability density function over the pixel space. Although these models produce sharp images, they have slow inference, demand heavy GPU parallelization for training, and do not explicitly learn a latent space. GAN (Goodfellow et al., 2014; Zhang et al., 2019; Brock et al., 2018) is another popular method in which a generator competes against a discriminator, producing outputs that imitate the inputs. GANs suffer from several notable issues: limited distribution coverage (Arora et al., 2017; Wu et al., 2016; Theis et al., 2016), training instability (Radford et al., 2016; Arjovsky et al., 2017; Mao et al., 2016). It also lacks probabilistic latent spaces to encode a given input, although there have been attempts to build latent representations on top of GANs (Donahue and Simonyan, 2019). VAEs (Kingma and Welling, 2013) consist of a bottom-up inference network and a top-down generation network parameterized by DNNs that are jointly trained to maximize the VLB of the data log-likelihood. Although VAEs are mathematically elegant, easy to train, fast in inference, and less GPU demanding than autoregressive models, its KL divergence penalty paired with reconstruction objective hampers realistic image generation since it overly stretches the latent space over the entire training set (Theis et al., 2016; Bousquet et al., 2017).

Attempts are made to combine the methods as mentioned above. PixelVAE (Gulrajani et al., 2016) integrates PixelCNN into VAE decoder but is still computationally heavy. RealNVP and other flow-based methods (Dinh et al., 2017; Kingma and Dhariwal, 2018) employ an invertible transformation between latent space and pixel space that enables exact log-likelihood computation and inference, but the model is restricted by the invertibility requirement. Adversarial Variational Bayes (AVB) (Mescheder et al., 2017) theoretically builds more flexible approximated posterior via adversarial learning but empirically still outputs blurry generations. VAE-GAN (Larsen et al., 2016) stitches VAE with GAN to enhance the generation quality while preserving an expressive latent space without introducing excessive computational overhead. However, VAE-GANs are still not competitive in complex image classes, as we note from Figure 5.1. To tame complex input spaces, recent works (Reed et al., 2016; Yan et al., 2016; Odena et al., 2017) leverage side information such as text description, foreground mask, and class labels as conditional variables hoping that the consequential conditional distributions are less tangled. But learning a conditional distribution requires additional labeling efforts both at training and downstream applications. Our approach handles complex input spaces well without the aid of conditional information. It follows the pipeline of VAE-GAN but employs a core channel recurrency to transform convolutional features into and out of the latent space. Such changes are similar in spirit as recent works on improving approximate posterior and prior for VAEs (Kingma et al., 2016; Chen et al., 2016c; Rezende and Mohamed, 2015), however, we do not change the prior or the posterior to maintain algorithmic simplicity and computational efficiency. Our recurrent module builds lateral connections between latent channels following a similar philosophy as in the deep autoregressive networks (DARN) (Gregor et al., 2014). But latent variables in DARN are sequentially drawn conditioned on the samples from the previous time steps and

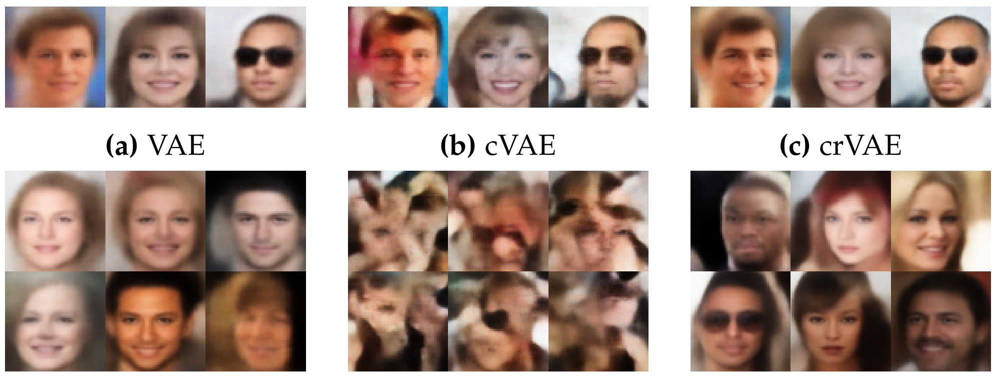


Figure 5.3: (top) reconstructions with latent variables drawn from the approximated posterior and (bottom) generations from the prior for VAE, convolutional VAE (cVAE), and the proposed crVAE.

can be slow in inference. DRAW networks (Gregor et al., 2015; Gregor et al., 2016) are related to ours as they also recurrently iterate over latent variables for generation. DRAW iterates over the entire latent variables multiple times and incrementally reconstructs pixel-level input at each iteration. In contrast, we only iterate between blocks of latent channels and reconstruct once. Thus it is computationally more efficient and learns interpretable latent subspaces.

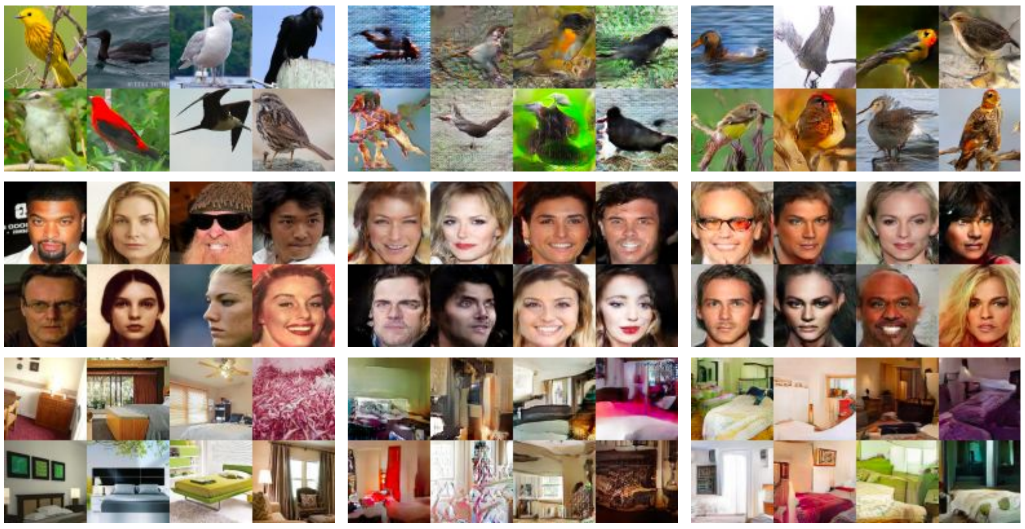
5.3 CHANNEL-RECURRENT AUTOENCODING

This section introduces the proposed channel-recurrent architecture, motivated by observing the limitations of the standard VAE and convolutional VAE (cVAE). Then, we extend crVAE-GAN with an adversarial loss to render realistic images. Furthermore, we introduce two latent space regularization techniques specific to our proposed channel-recurrent architecture, namely, the KL objective weighting and the mutual information maximization.

5.3.1 Latent Space Analysis of VAEs

VAE approximates the intractable posterior of a directed graphical model with DNNs (Figure 5.2a), maximizing a VLB of the data log-likelihood:

$$\mathcal{L}_{\text{VAE}} = -\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] + D_{\text{KL}}(q_\phi(z|x)\|p(z))]$$



(a) ground truth **(b)** VAE-GAN **(c)** crVAE-GAN

Figure 5.4: 64×64 resolution image generation of (top) Birds, (middle) CelebA and (bottom) LSUN using (b) baseline VAE-GAN and (c) our proposed crVAE-GAN along with (a) real examples.

where the approximate posterior $q_\phi(z|x)$ is modeled as a diagonal Gaussian and the prior $p(z)$ as a standard Gaussian. We refer to $q_\phi(z|x)$ as an inference network and $p_\theta(x|z)$ a generation network. The latent space of standard VAE is modeled as 1-dim vector $z \in \mathbb{R}^c$ (Kingma and Welling, 2013), whereas for cVAE the latent space is modeled as a 3-dim tensor $z \in \mathbb{R}^{w \times h \times c}$ (Sohn et al., 2015).

Overly smoothed reconstructions (Figure 5.3a) and generations (Figure 5.3d) as well as a lack of sample diversity are major downsides of VAEs. A potential cause is that the naive parameterization of the latent space, its associated prior and approximated posterior, may not be able to reflect a complex data distribution (Theis et al., 2016; Kingma et al., 2016; Chen et al., 2016c). One would be tempted to provide a fix by adding an image-specific prior to the latent space, such as spatial structure, leading to cVAE (Figure 5.2b), whose inference and generation networks, different from standard VAE, are fully convolutional. By making the approximated posterior spatially correlated, cVAE can learn with more local details during inference, reflected by higher quality reconstructions (Figure 5.3b). However, the latent variables sampled from spatially independent prior of cVAE ignores the global structure of face shapes and produce chaotic samples (Figure 5.3e).

5.3.2 Channel-Recurrent Variational Autoencoder

Employing a prior with spatially dependent latent variables, such as a full-covariance Gaussian prior, is one remedy to the problem of chaotic image generation in cVAEs. However, such prior complicates the optimization due to significantly increased number of parameters (e.g., full covariance matrix $\sim O((w \times h \times c)^2)$), especially when the latent space is large (Duchi, 2007; Gregor et al., 2014). Alternatively, we can structure the covariance to have dense dependencies across the spatial dimension and conditional dependencies along channels, hoping that such a setup can guide each channel to model different aspects of an image.

One possible way to apply the desired structure is to introduce hierarchy to the latent variables, which requires sequential sampling and complicates the training. Thus, tackling from a different direction, we opt to adapt to the network architecture. We propose to factorize the convolutional latent space into blocks across channels, flatten the activation to model spatial dependency, and connect the blocks via an LSTM (Hochreiter and Schmidhuber, 1997) to allow communication and coordination among them for coherency across channels. That is, the transformation of the current block of latent channels always takes account of the accumulative information from the preceding time steps, serving as guidance. Concretely, during generation, $z=[z_1, \cdot\cdot, z_T]$ with $z_i \in \mathbb{R}^{w \times h \times \frac{c}{T}}$ sampled from standard Gaussian prior is passed through an LSTM to obtain a transformed representation $u = [u_1, \cdot\cdot, u_T]=\text{LSTM}(z)$, which is then projected back to the pixel space. Similarly, during inference, the mean path shares the same architecture as in cVAE; the variance path slices latent variables into T blocks of size $w \times h \times \frac{c}{T}$, each referred to as σ_i and feeds σ_i's into another LSTM to output σ_i^{rnn} as the final variances for the approximate posterior. Our proposed model, referred to as the channel-recurrent VAE (crVAE), can both reconstruct and generate with higher visual quality than VAE and cVAE, as shown in Figure 5.3c and 5.3f. More mathematical intuition and details for our design are in the Supplementary Materials.

5.3.3 Additional Regularization

Inspired by (Larsen et al., 2016), we adopt an adversarial loss to generate realistic images, leading to crVAE-GAN. Additionally, we propose two novel regu-

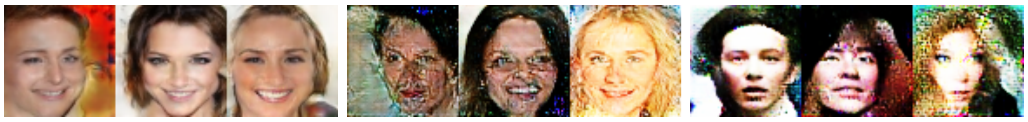


Figure 5.5: (Left) generations from the baseline VAE-GAN without MI regularization has much less artifacts than those from models trained with MI regularization on z (middle) or $\mathrm{FC}_{\text{gen}}(z)$ (right), implying such regularization is not compatible with VAE-GAN.

larizers to enhance the latent space quality of crVAE-GAN for better semantic disentanglement and more stable optimization.

Generating Realistic Images with crVAE-GAN

We extend crVAE to crVAE-GAN with an auxiliary adversarial loss on top of the generation network outputs for realistic image synthesis. The discriminator D maps an image sampled from either the posterior or prior into a binary value:

$$\max_{\phi,\theta} \mathcal{L}_{\text{VAE}} + \beta \mathbb{E}_{z\sim\{q_\phi(z|x),p(z)\}} \left[\log D(p_\theta(x|z))\right]$$

$$\max_{D} \mathbb{E}_{x\sim X} \left[\log D(x)\right] + \mathbb{E}_{z\sim\{q_\phi(z|x),p(z)\}} \left[\log(1-D(p_\theta(x|z))\right].$$

Training can be done by min-max optimization as in (Larsen et al., 2016).

Weighting the KL Objective

The KL objective of crVAE can be written as follows:

$$\sum_{t=1}^{T} (1-\alpha_t) D_{\text{KL}}(q_\phi(z_t|x) \| p(z_t)), \tag{5.1}$$

where $\alpha_t{=}0,\ \forall t{\in}\{1,\cdots,T\}$ yields the equivalent expression to standard VAE objective. The channel recurrent architecture additionally enables different weights to regularize the KL objective at each time step. Specifically, noting that the earlier ones can heavily influence the later time steps due to the recurrent connection, we gradually reduce the regularization coefficients of the KL divergence to balance. From an information-theoretic perspective (Alemi et al., 2016; Tishby et al., 2000), the earlier time steps with larger coefficients hold a tighter information bottleneck, meaning that these latent channels shall convey general outlines, whereas the later time steps with smaller coefficients, i.e., a more flexible information flow, constitute diverse details conditioned on the sketched outlines. Figure 5.8 demonstrates the resulting effects where earlier time steps output rough profiles and later ones craft the details.

Mutual Information Regularization

To increase training stability, we borrow the idea of mutual information (MI) maximization from (Chen et al., 2016b) as an additional regularization. By recovering the latent variables from the generated image, the generation network encourages the administration of latent information to the output space. The regularization objective is written as:

$$\mathbb{E}_{z\sim\{q_\phi(z|x),p(z)\},\tilde{x}\sim p_\theta(x|z)}\left[q_\psi(z|\tilde{x})\right] \tag{5.2}$$

where z can be sampled either from approximate posterior $q_\phi(z|x)$ or prior $p(z)$. The CNN encoding path is shared between q_ψ and D as in (Chen et al., 2016b), mapping generated image to reconstruct z and to a binary value, respectively.

Note that the formulation in Equation (5.2) is not restricted to z, and we empirically found that relating the transformed representation $u{=}\text{LSTM}_{\text{gen}}(z)$ with the output of q_ψ under crVAE-GAN framework is much more effective. However, similar regularization is found detrimental for VAE-GAN, both with z and the transformed representation $\text{FC}_{\text{gen}}(z)$, as shown in Figure 5.5. We compare samples generated from the baseline VAE-GAN and those trained with additional regularization to relate z or $\text{FC}_{\text{gen}}(z)$ with the outputs, but the outcomes of the latter are visibly worse. We contemplate that VAE-GAN lacks an equivalent transformation step as in crVAE-GAN via channel-recurrent architecture, which particularly functions to enhance the latent representations. More empirical demonstrations are in Section 5.4.2 and implementation details in the Supplementary Materials.

5.4 EXPERIMENTS

For evaluation, we first synthesize 64×64 natural images, followed by a 2nd stage generation to 128×128 or 224×224 on top of the 1st stage generation. To demonstrate the latent space capacity, image completion tasks are performed via optimization based on latent representations. Finally, we explore the se-

Model (64×64)	Birds	CelebA	LSUN
VAE-GAN	5.81±0.09	21.70±0.15	16.6%
crVAE-GAN	10.62±0.12	24.16±0.33	29.9%
crVAE-GAN + MI	**11.07**±0.12	**26.20**±0.19	**53.5**%
Model (128 or 224)	**Birds**	**CelebA**	**LSUN**
VAE-GAN	14.97±0.11	19.09±0.19	20.5%
VAE-GAN + Perc.	14.61±0.24	27.09±0.26	10.5%
crVAE-GAN	29.14±0.45	**42.66**±0.45	**34.8**%
crVAE-GAN + Perc.	**32.13**±0.37	35.03±0.39	**34.2**%

Table 5.1: Quantitative evaluation on generating 64×64 and higher-resolution images. For Birds and CelebA, inception scores are reported. For LSUN, the frequency of selection by mechanical turk workers as the most realistc generation among all models is reported.

	Occlusion	VAE-GAN	crVAE-GAN
Birds	lower	28.9%	**71.1%**
	upper	35.2%	**64.8%**
CelebA	eye	18.0%	**82.0%**
	mouth	22.7%	**77.3%**
	half	34.4%	**65.6%**
LSUN	center	23.4%	**76.6%**

Table 5.2: The frequency of selection by mechanical turk workers as the more realistic completion using VAE-GAN and crVAE-GAN.

mantics of the learned latent channels, particularly with respect to different time steps.

Three datasets, covering a diverse spectrum of contents, are used for evaluation. Birds dataset is composed of three datasets, namely Birdsnap (Berg et al., 2014), NABirds (Van Horn et al., 2015) and Caltech-UCSD Birds-200-2011 (Wah et al., 2011), containing 106,474 training and 5974 validation images. CelebA (Liu et al., 2015) contains 163,770 training and 19,867 validation images of face. LSUN bedroom (LSUN) (Yu et al., 2015) contains 3,033,042 training and 300 validation images. The ROIs are cropped and scaled to 64×64 and 128×128 for Birds and CelebA; the images are scaled and cropped to 64×64 and 224×224 for LSUN. Complete Implementation details are in the Supplementary Materials.

5.4.1 Stage 1: 64×64 Image Generation

We compare our crVAE-GAN to the baseline VAE-GAN for 64×64 image generation (Figure 5.4). This also serves as the first step towards generating higher-resolution images later on. For Birds, VAE-GAN generates colorful images (Figure 5.4b), but details are highly obscured, indicating the latent space conveys mostly low-level information such as color and edges, but not much high-level semantic concepts. By contrast, crVAE-GAN generates significantly more realistic birds with decent diversity in color, background, and poses (Figure 5.4c). Generating aligned faces and structured bedrooms are less difficult than birds, but crVAE-GAN still exhibits clear superiority over VAE-GAN.

For quantitative evaluation, we measure inception scores on Birds and CelebA using ImageNet pretrained VGG11 models finetuned on bird and face recognition task (Yi et al., 2014), respectively. As no bedroom classification dataset is available, we instead conduct a user study for LSUN. We ask a mechanical turk worker to select the most realistic out of 3 generated images (corresponding to VAE-GAN and crVAE-GAN, with and without MI regularization), repeating for 2000 times. We observe improved inception scores in Table 5.1, which agrees with the visual observation. The frequency of selection of each model by mechanical turk workers on generated LSUN images in Table 5.1 also verifies that our proposed model outperforms the baseline with a significant margin.

5.4.2 Effect of Mutual Information Regularization.

We examine the effects of Mutual Information (MI) regularization on training crVAE-GAN by projecting the same z to the output pixel space via generation networks through the last 10 epochs of training. Figure 5.6a and 5.6b show the results without and with MI regularization, respectively, where the top 4 rows are successful samples, 5th and 6th rows lower quality samples, and the bottom 2 rows failure cases. First, note that both models supply crisp, coherent, and realistic samples when successful, and their inception scores are also close (Table 5.1). Nonetheless, without MI regularization, generated samples between consecutive epochs oscillate, and failure cases tend to collapse to the same mode despite originating from different z's. By contrast, with the regularization, the convergence becomes stable, and the mode collapsing phenomenon no longer exists even for failed generations. High variance and mode collapsing are two well-known issues of adversarial training (Radford et al., 2016). MI maximization overcomes these issues by (1) enforcing the latent messages to be passed to the outputs and (2) regulating the adversarial gradients–recall that MI and adversarial objectives share the same encoding path. Unless specified otherwise, the crVAE-GAN results reported are trained with MI regularization.

5.4.3 Stage2: Higher Resolution Image Generation

To further assess the quality of Stage1 64×64 generations, we raise the generation resolution to 128×128 for Birds and CelebA,[2] and to 224×224 for LSUN. Our Stage2 generation network is designed similarly to that of StackGAN (Zhang et al., 2016a), but the generation is done in an unsupervised way without any condition variables. Thanks to the nature of our framework, the Stage1 outputs are composed of both generated and reconstructed samples. We can utilize not only both sources of "fake" images in training but also an additional perceptual loss (Johnson et al., 2016) of the reconstructed images to regularize the Stage2 network. Figure 5.9 presents generated samples from Stage2 networks with and without perceptual loss while taking generation outputs of VAE-GAN and crVAE-GAN Stage1 models as input. Table 5.1 provides quantitative evaluations following the protocol of Section 5.4.1. The qualitative and quantitative results imply that a high-quality Stage1 generation is essential for Stage2 success, despite that the Stage2 network can correct some of the Stage1 mistakes. Since crVAE-GAN supplies much higher quality Stage1 generations than VAE-GAN, it also produces more visually pleasing Stage2 generations. We also observe that the inception scores, in this case, can diverge from visual fidelity, e.g., the Stage2 CelebA results with perceptual loss exhibit higher visual quality than without but lower inception scores. Lastly, other mechanisms besides stacking generation networks can be applied to further raise image resolution, such as variations of the recently proposed progressive GAN (Karras et al., 2017), a worthy future direction but out of the scope of this paper.

2 We decide to generate higher-resolution images of 128×128 for Birds and CelebA since the ROIs of are approximately of this resolution.

5.4.4 Image Completion

To verify how faithfully the latent manifold from crVAE-GAN reflects the semantic meaning of the input space, we conduct an image completion task using Stage1 models. We occlude parts of validation images, namely right-half, eye and mouth regions for CelebA, upper and lower part in Birds and blocks for LSUN, and then optimize their latent representations z's to fill in the missing parts (Yan et al., 2016):

$$\min_{z} \left[\|\hat{x} \odot m - x \odot m\|_2^2 + \gamma \log \mathcal{N}(z; 0, \mathbf{I}) + \tau \log(1 - D(\hat{x})) \right],$$

where $\hat{x} = \text{gen}(z)$ refers to the output of the generation network, $m \in \{0,1\}^{3\times 64\times 64}$ is a mask whose entries are 0 if corresponding pixel locations are occluded and 1 otherwise, and $\odot$ represents element-wise multiplication. Qualitative examples are in Figure 5.7. VAE-GAN struggles to complete some missing regions, e.g., right half of the faces and sunglasses in Figure 5.7a, or generates excessive noise, e.g., the sky in Figure 5.7b. By contrast, crVAE-GAN is more competent in retracting off-orbit latent points back to the actual manifold by better embedding the high-level semantic information. Since there may exist multiple solutions in completing an image besides the ground truth, reconstruction error is not an ideal metric for quantitative measurement. Instead, we conduct human evaluation by presenting completed results from VAE-GAN and crVAE-GAN to mechanical turk workers and asking them to select the more realistic one. The selection frequency of each model out of 128 randomly selected pairs is reported in Table 5.2. Overall our crVAE-GAN again outperforms VAE-GAN with a substantial margin.

5.4.5 Latent Channel Semantics

Our crVAE-GAN processes the latent variables sequentially, allowing a global-to-local and coarse-to-fine progression. Figure 5.8 highlights this progression by first initializing all latent variables to be zero, and then gradually sampling blocks of latent variables from the standard Gaussian prior. Due to the weighted KL penalty, the first four time steps tend to operate on an image's overall tone: defining the background color theme, outlining the general shapes, etc. The second half attends the details: the texture of feathers for Birds, facial expressions for CelebA, lighting for LSUN, etc. For a concrete example, the first two rows from CelebA demonstration both start with an outline of men with glasses, then gradually diverge to different hairstyles, opposite poses, one taking off while the other solidifying the presence of glasses, etc. Such a progressing phenomenon also suggests that latent channels at different time steps carry their interpretable semantics.

To investigate this hypothesis, we first draw random samples from the prior across 8 time steps, denoted by $z = [z_1, \cdot\cdot, z_t, \cdot\cdot, z_8]$; we then draw a new sample $\tilde{z} = [z_1, \cdot\cdot, \tilde{z}_t, \cdot\cdot, z_8]$ by only changing a representation at time t that shows semantically meaningful changes from z; finally we generate images by interpolating between z_t and $\tilde{z}_t$ while fixing other z_i's. We note an interesting tendency where images sharing certain characteristics can be manipulated in the same way by interpolating towards the same $\tilde{z}_t$. For example, the CelebA samples in Figure 5.8 suggest that t=5 decides to pose variation; indeed, if two faces both look into the right (Figure 5.10b), traversing z_5's of these faces to the same $\tilde{z}_5$ (selected by visual inspection) will smoothly frontalize their poses while retaining other factors. Similar observations can also be made with Birds and LSUN.

Additionally, we conjecture that our model grants more freedom for semantic variations by associating an explainable factor to a latent subspace rather than a single latent unit (Higgins et al., 2017a; Chen et al., 2016b). To investigate this hypothesis, we generate several different *styles* of the same factor by sampling different z_t's while fixing the other z_i's. For instance, Figure **??** not only takes the glasses on/off but also switches from eyeglasses to sunglasses, from thin frame to thick frame. In comparison, existing works on controlling factors of variation through latent unit manipulation, such as infoGAN (Chen et al., 2016b) and β-VAE (Higgins et al., 2017a), can only shift the controlled factor along a single direction, e.g., a latent unit that controls the existence of glasses does not allow different glasses styles.

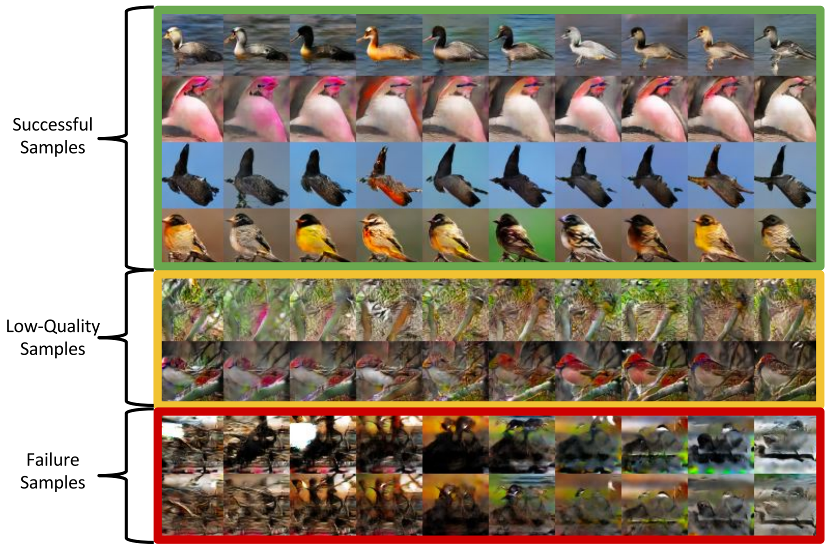
Channel recurrency is not yet perfect in explaining latent semantics. In particular, determining the direction representing a certain factor of variation still requires visual inspection. Nevertheless, the preliminary demonstrations of this intriguing property shed light on future research in learning more semantically meaningful latent spaces.

5.5 COMPUTATIONS

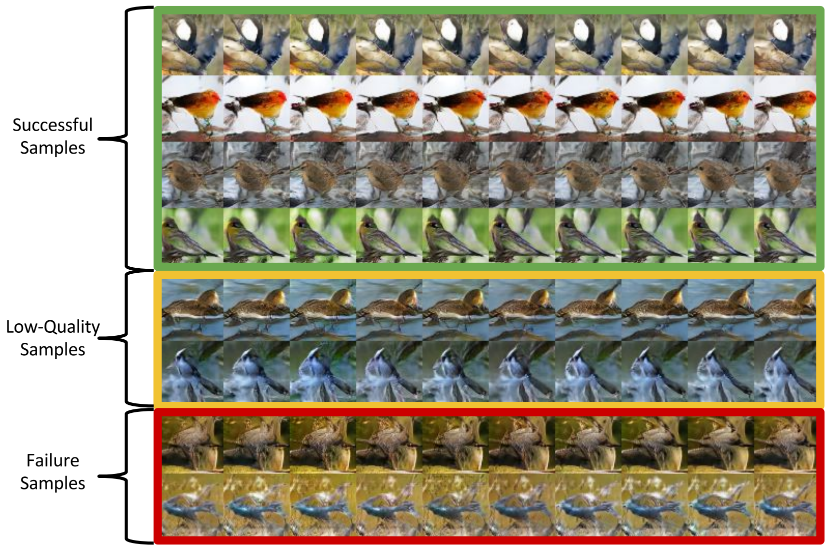
Another prominent advantage of crVAE-GAN to the baseline as well as other state-of-the-art autoregressive models (Oord et al., 2016a; Kingma et al., 2016) is found from computational aspects. First, as LSTMs share weights over time, for Stage1 generations, our proposed model has 130M parameters, when the baseline VAE-GAN with the same number of latent variables and the same encoder/decoder architecture has 164M. For the same reason, training our model consumes much less GPU memory. For example, in our implementation, during Stage1 optimization, crVAE-GAN requires around 4.5GB memory with a batch size of 128 while VAE-GAN requires 6.2GB. Finally, the inference and generation complexities for crVAE-GAN is on the same order as those for VAE-GAN. In wall clock time, for a mini-batch of 128 images using a Titan X, crVAE-GAN on average takes 5.8 ms for inference and 4.0 ms for generation; VAE-GAN takes 2.6 ms for inference and 2.2 ms for generation. Meanwhile, autoregressive models are significantly slower in evaluation: even under careful parallelization, it is reported that PixelCNN (Oord et al., 2016a) takes 52K ms and inverse autoregressive flow (Kingma et al., 2016) 50 ms to generate a single 32×32 image on a Titan X.

5.6 CONCLUSION

We propose the channel-recurrent autoencoding framework to improve the latent space constructions for image modeling upon the baseline VAE models. We evaluate the performance of our proposed framework via generative image modeling, such as image generation, completion, and latent space manipulation. Future research includes building more interpretable features via channel recurrency and extrapolating our framework to other tasks.

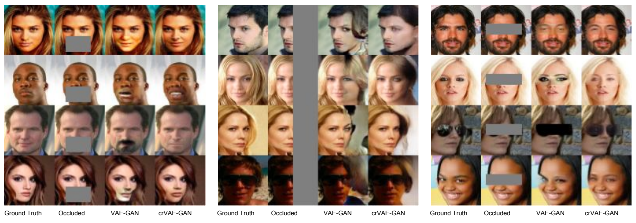


(a) crVAE-GAN without MI regularization

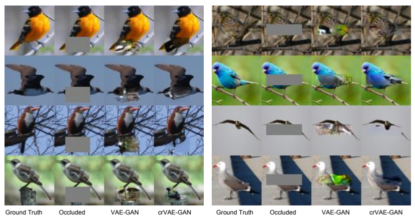


(b) crVAE-GAN with MI regularization

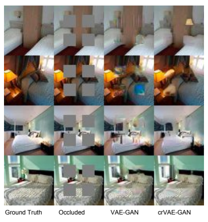
Figure 5.6: The same z's are sampled for each row from a standard Gaussian prior and projected back to the pixel space using the snapshot of decoders at last 10 epochs of training. (a) and (b) correspond to crVAE-GAN trained without and with the MI regularization. Top 4 rows are successful samples, 5th and 6th are low-quality ones, and bottom 2 are failure cases. Clear improvement in stability is observed for (b) in terms of color oscillations between consecutive epochs and failure case mode collapsing.



(a) CelebA: mouth, right-half, eyes



(b) Birds: lower, upper



(c) LSUN: blocks

Figure 5.7: Image completion with VAE-GAN and crVAE-GAN.

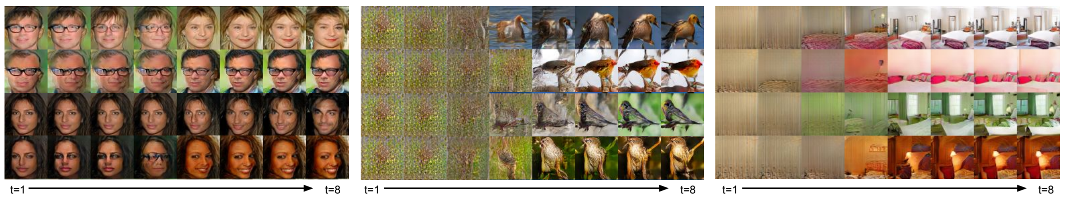


Figure 5.8: Progressively drawing samples from a standard Gaussian prior over time steps. We observe how image generation evolves by determining global structure at earlier time steps and gradually adding more details later on. Recall the first 3 time steps carry more KL-weight than the rest hence a visual leap occurs around $t = 3$ or 4.

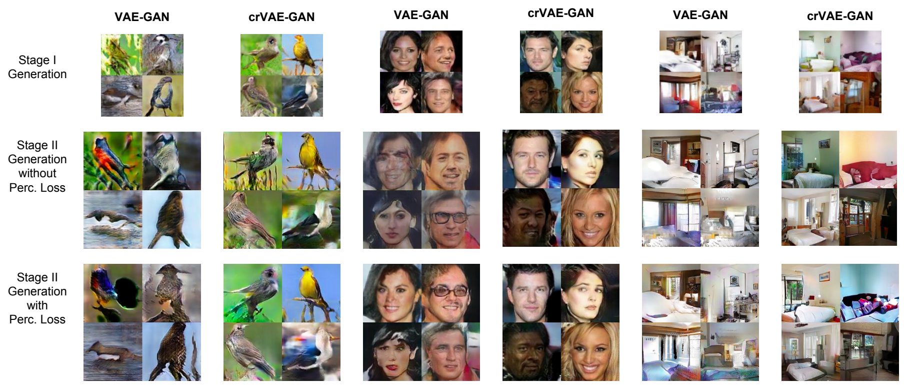
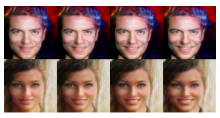
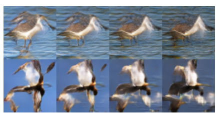
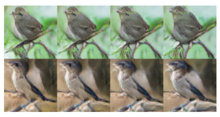

VAE-GAN
crVAE-GAN
VAE-GAN
crVAE-GAN
VAE-GAN
crVAE-GAN
Stage I Generation
Stage II Generation without Perc. Loss
Stage II Generation with Perc. Loss

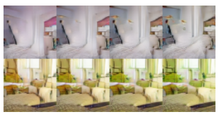
Figure 5.9: Stage2 generation from 64×64 to 128×128 (Birds and CelebA) and 224×224 (LSUN). Proposed crVAE-GAN provides a solid foundation to assure Stage2 success, clearly contrasting the baseline VAE-GAN.


(a) expression ($t = 8$)

(b) azimuth ($t = 5$)


(c) background ($t = 5$)


(d) background ($t = 5$)


(e) color ($t = 3$)

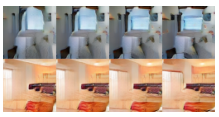

(f) window ($t = 7$)

Figure 5.10: Interpolating between z_t and $\tilde{z}_t$, for a selected t, while fixing other z_i's. We observe a gradual shift of an attribute towards the semantic direction encoded by $\tilde{z}_t$ while preserving most of the other factors.

CHAPTER APPENDIX

5.A MATHEMATICAL INTUITION ON MODEL DESIGN

We provide more details on the mathematics that motivates our proposed channel-recurrent architecture. Particularly, our architecture attempts to imitate the effects of having a non-diagonal multivariate Gaussian approximated posterior and prior, which is too computationally expensive to achieve exactly in practice when the latent dimension is big (Duchi, 2007). In practice, to draw a sample from a non-diagonal multivariate Gaussian, one first draw from the standard multivariate Gaussian and operate the following procedures are performed:

$$z = A\epsilon + \mu,\ AA^\top = \Sigma, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

Therefore, to imitate the variance transformation via multiplication of A, we apply an LSTM layer on the variance inference path to allow all latent variables to communicate with one another and similarly, to imitate the inverse of that, we apply another LSTM layer on the generation path. To imitate the mean addition, on the mean inference path, we integrate an element-wise addition layer and on the generation path, we add another element-wise addition layer after the LSTM layer.

5.B ABLATION STUDIES ON MNIST

We ablate our model architecture on MNIST, by comparing models including standard VAE, cVAE, crVAEs with channel recurrency on the inference path only, the generation path only and on both paths. We also conduct control experiments to search the optimal number of recurrent channels on crVAE.

The negative log-likelihood (NLL) of MNIST has been used as a benchmark to evaluate probabilistic generative models. NLL is approximated via impor-

Model	specifications	NLL	Model	specifications	NLL
VAE	–	82.04	crVAE	$T = 4$	81.09
cVAE	–	83.12	crVAE	$T = 4$	80.84
crVAE	@inference	81.86	crVAE	$T = 16$	81.35
crVAE	@generation	81.71	crIWAE	–	80.02
crVAE	both	80.84	crVAE	2 stochastic layers	79.65

Table 5.B.1: Ablation studies on MNIST comparing VAE, cVAE, and crVAE for different configurations.

tance sampling (Rezende et al., 2014), i.e., for each image from the statistically binarized MNIST test set (Larochelle, 2015), we sample from its approximated posterior 10K times. We keep the same latent dimension for all the models. The results obtained with our and a number of previously proposed generative models are summarized in Table 5.B.1. The baseline, standard VAE (Kingma and Welling, 2013), obtains 82.04 NLL. As expected, cVAE shows worse performance (83.12), because it can not sufficiently capture globally coherent patterns. Among crVAE models, positioning the recurrency on either the inference path or the generation path marginally improves upon the standard VAE and cVAE. The most substantial improvement happens when the recurrency is used both for inference and generation paths, achieving 80.84 NLL.

To assess the impact of the number of recurrent time steps, we train crVAEs with $T = 4, 8, 16$ while maintaining other hyperparameters. Table 5.B.1 shows that $T = 8$ achieves the best NLL. We conjecture that including too many latent channels at one step ($T = 4$) burdens the recurrent layers to establish meaningful intra-block connections, whereas too few latent channels ($T = 16$) limits the expressive power of the LSTM.

Our crVAE is generalizable and compatible with many other advanced training methods of directed graphical models. For example, by integrating k-sample importance weighted estimation (Burda et al., 2015), we can improve the NLL to 80.02; a hierarchical crVAE with 2 stochastic layers reduces NLL to 79.65

5.C DETAILS OF NETWORK ARCHITECTURE

5.C.1 Stage1 Generation

Please refer to the paper repository (`https://github.com/WendyShang/crVAE`) for Stage1 model architecture details.

5.C.2 Stage2 for Birds

The Stage2 generation network for Birds follows similar architecture as in (Zhang et al., 2016a), but without the text encoding parts. We have uploaded the pre-trained Stage2 Birds models to the repository.

5.C.3 Stage2 for CelebA

For Stage2 generation of crVAE-GAN on CelebA dataset, we found the original stackGAN (Zhang et al., 2016a) architecture does not perform well. Instead, we replace the nearest neighbor upsampling followed by 3×3 convolutions with a deconvolution layer, i.e., a full convolution of kernel 4×4 and stride 2, padding 1. In the case of Stage2 generation of VAE-GAN, we found that the generation performance is sensitive to the generator architecture. We only reduce the spatial dimension once from 64×64 to 32×32 instead of downsampling to spatial dimension 16×16.

We also have uploaded the pretrained Stage2 CelebA models to the repository.

5.C.4 Stage2 for LSUN

We integrated more modifications over the model from (Zhang et al., 2016a) to generate 224×224 LSUN bedroom images and uploaded the pretrained LSUN model to the repository.

5.D IMPLEMENTATION DETAILS

MODIFICATION OF VAE-GAN OBJECTIVE. We empirically found that reconstructing the last convolutional layer of the discriminator to maximize $\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right]$ as proposed in (Larsen et al., 2016) leads to more instability in training comparing to reconstructing at the pixel-level. Hence, throughout our work, we reconstruct at the pixel level for both VAE-GAN and crVAE-GAN.

DATA AUGMENTATION. For Birds and CelebA, random horizontal flipping is used during training as data augmentation. For LSUN, random cropping and horizontal flipping are used.

OPTIMIZATION. Our models are optimized with ADAM (Kingma and Ba, 2014), where we set $\epsilon = 1 \times 10^{-8}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$. For the discriminator, we use a variation of ADAM with additional thresholding (Larsen and Sønderby, 2016): if the classification accuracy of the discriminator for a batch consisting of half real and half fake images is over 90%, we do not update the model parameters of the discriminator.

INITIAL LEARNING RATE. For Stage1 models, Birds has initial learning rate 0.0003, CelebA 0.003 for VAE-GAN and 0.001 for crVAE-GAN, LSUN 0.0003 for VAE-GAN and 0.0001 for crVAE-GAN. For all Stage2 networks, we use 0.0002 as initial learning rate.

OBJECTIVES. Recall the VAE objective:

$$\mathcal{L}_{\text{VAE}} = \underbrace{-\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right]}_{\mathcal{L}_{\text{recon}}} + \underbrace{D_{\text{KL}}(q_\phi(z|x)\|p(z))}_{\mathcal{L}_{\text{prior}}} \geq -\log(p(x)). \tag{5.3}$$

Conventionally, for RGB images, $p_\theta(x|z)$, a diagonal multivariate Gaussian, is assumed to have fixed variance σ and only the mean values are estimated which correspond to pixel values. We follow this convention and apply MSE loss at the pixel level, which we find effective even without feature-level reconstruction, and defer σ to be adjusted through weighting the $\mathcal{L}_{\text{prior}}$, i.e., the KL-divergence term, by introducing an additional hyperparameter α. Recall that we weigh the KL-divergence term differently for different time steps when training crVAE, i.e., for the first 3 time steps with α_1 and the rest α_2. Moreover, the adversarial loss is weighted by hyperparameter β. Together, we have the following training objective for RGB images:

$$\mathcal{L}_{\text{VAE-GAN}} = \mathcal{L}_{\text{recon}} + \alpha\mathcal{L}_{\text{prior}} + \beta\mathcal{L}_{\text{D}}, \tag{5.4}$$

where

$$\mathcal{L}_{\text{D}} = -\log(D(x)) - \log(1 - D(\text{gen}(z))), z \sim p(z), \text{ or } z \sim q_{\phi}(z|x).$$

It is worth mentioning that the generated samples, which are decoded from z sampled from the prior $p(z)$, and the reconstructed samples, which are decoded from z sampled from the approximated posterior $q_{\phi}(z|x)$, are both used for training with adversarial loss. Specifically, a mini-batch for discriminator update is assembled with 50% real examples (half), 25% generated samples (a quarter), and 25% reconstructed samples. Also, note that $\mathcal{L}_{\text{recon}}$ is divided not only by the batch size but also by the channel, width, and height of the images for implementation convenience, while $\mathcal{L}_{\text{prior}}$ or $\mathcal{L}_{\text{D}}$ are divided by the batch size only. This explains why the optimal α or β values are considerably smaller than one.

α AND β For Birds, VAE-GAN is trained with $\alpha = 0.0002$ and $\beta = 0.0125$ and crVAE-GAN $\alpha_1 = 0.0003$, $\alpha_2 = 0.0002$ and $\beta = 0.0125$; for CelebA, VAE-GAN is trained with $\alpha = 0.0003$ and $\beta = 0.01$ and crVAE-GAN $\alpha_1 = 0.0003$, $\alpha_2 = 0.0002$ and $\beta = 0.01$; for LSUN, VAE-GAN is trained with $\alpha = 0.0002$, $\beta = 0.025$ and crVAE-GAN $\alpha_1 = 0.0003$, $\alpha_2 = 0.0002$, $\beta = 0.0125$.

MI OBJECTIVES. In the main text, for crVAE-GAN, we introduce a regularization term based on the maximization of mutual information between the generation $\tilde{x}$ and its sampled latent variable z. We now mathematically justify and illustrate the formulation of the auxiliary objective:

$$\begin{aligned}
I(z, \tilde{x}) &= H(z) - H(z|\tilde{x}) \\
&= \mathbb{E}_{\tilde{x}\sim p_\theta(x|z)}\left[\mathbb{E}_{\tilde{z}\sim p(z|\tilde{x})}\left[\log p(\tilde{z}|\tilde{x})\right]\right] + H(z) \\
&\geq \mathbb{E}_{\tilde{x}\sim p_\theta(x|z)}\left[\mathbb{E}_{\tilde{z}\sim p(z|\tilde{x})}\left[\log q(\tilde{z}|\tilde{x})\right]\right] + H(z) \\
&= \mathbb{E}_{z\sim p(z),\tilde{x}\sim p_\theta(x|z)}\left[\mathbb{E}_{\tilde{z}\sim p(z|\tilde{x})}\left[\log q_\psi(\tilde{z}|\tilde{x})\right]\right] + H(z) \\
&= \mathbb{E}_{z\sim p(z),\tilde{x}\sim p_\theta(x|z)}\left[\log q_\psi(z|\tilde{x})\right] + H(z),
\end{aligned}$$

where $q_\psi(\tilde{z}|\tilde{x})$ is a variational approximation to $p(\tilde{z}|\tilde{x})$, parametrized by a neural network which shares the same encoding path with the discriminator. The last equality transformation comes from Lemma 5.1 in (Chen et al., 2016b). Similarly, as done in (Chen et al., 2016b), we bypass the optimization of $H(z)$ by treating it as a constant and leverage the reparametrization trick to sample from z. As in the case for the adversarial loss, when given the ground truth x, we sample z from its approximated posterior $q_\phi(z|x)$, otherwise sample from prior $p(z)$. Therefore, the overall objective for the auxiliary task becomes:

$$\mathcal{L}_{\text{MI}} = \mathbb{E}_{z\sim p(z),\tilde{x}\sim p_\theta(x|z)}\left[\log q_\psi(z|\tilde{x})\right] + \mathbb{E}_{x\sim X, z\sim q_\phi(z|x),\tilde{x}\sim p_\theta(x|z)}\left[\log q_\psi(z|\tilde{x})\right]$$

Recall that in RGB images, the formulation $p_\theta(x|z)$ estimates the mean of the Gaussian distribution while fixing variance. Therefore, in practice, we do not perform an additional sample from $p_\theta(x|z)$ but directly take the mean. Also, we assume $q_\psi(z|\tilde{x})$ to be a diagonal multivariate Gaussian with fixed σ. In this way, the auxiliary objective boils down to L^2 reconstruction of z from $\tilde{x}$. Finally, as explained in the main text, we found empirically reconstructing z does not work as well as reconstructing the transformed latent variable $u = \text{LSTM}(z)$. Thus we set the objective to be the latter. Now the overall objective for crVAE-GAN becomes

$$\mathcal{L}_{\text{VAE-GAN}} = \mathcal{L}_{\text{recon}} + \alpha_1\mathcal{L}^1_{\text{prior}} + \alpha_2\mathcal{L}^2_{\text{prior}} + \beta\mathcal{L}_{\text{D}} + \kappa\mathcal{L}_{\text{MI}},$$

where for Birds, $\kappa = 0.02$, for CelebA, $\kappa = 0.01$ and for LSUN, $\kappa = 0.01$.

STAGE2 OBJECTIVE. The objective for the discriminator during Stage2 training is to maximize:

$$\mathcal{L}_D = \mathbb{E}_{s\sim S}\left[\log(D(s))\right] + \mathbb{E}_{z\sim p(z),\tilde{x}\sim p_\theta(x|z)}\left[\log(1 - D(G(\tilde{x}))\right] + \mathbb{E}_{z\sim q_\phi(z|x),\tilde{x}\sim p_\theta(x|z)}\left[\log(1 - D(G(\tilde{x}))\right]$$

The loss for the generator is to minimize

$$\mathcal{L}_G = \mathbb{E}_{z\sim p(z),\tilde{x}\sim p_\theta(x|z)}\left[\log(1 - D(G(\tilde{x}))\right] + \mathbb{E}_{z\sim q_\phi(z|x),\tilde{x}\sim p_\theta(x|z)}\left[\log(1 - D(G(\tilde{x}))\right] + \gamma\mathbb{E}_{z\sim q_\phi(z|x),\tilde{x}\sim p_\theta(x|z)}\left[\|f(S(x)) - f(G(\tilde{x}))\|_2^2\right],$$

where $S(x)$ is the ground truth upsampled x and f represent feature extraction via some pre-trained deep CNNs, namely VGG11 for Birds and CelebA and ResNet50 for LSUN. For the experiments without perceptual loss, $\gamma = 0$, otherwise $\gamma = 1$ for Birds and CelebA, and $\gamma = 10$ for bedroom.

IMAGE COMPLETION HYPERPARAMETERS. Recall the image completion objective:

$$\min_{z} \left[\|\hat{x} \odot m - x \odot m\|_2^2 + \gamma \log \mathcal{N}(z; 0, \mathbf{I}) + \tau \log(1 - D(\hat{x}) \right].$$

We use $\gamma = 0.00001$ and $\tau = 0.003$.

6

ATTENTIVE CONDITIONAL CHANNEL-RECURRENT AUTOENCODING

6.1 INTRODUCTION

Having introduced the unsupervised unconditional channel-recurrent autoencoding, we proceed to extend the channel-recurrency to a conditional version. Conditional image generation can be of practical significance. Let's take a look at a motivational example. Assaults, burglaries, and thefts are among the top common crimes taking place in the United States (FBI, 2017). The witnesses' post-trauma memories become susceptible to even a short period of delay due to factors such as stress and potential contamination from other events. It is shown that the recall of a suspect's face from a witness is the most accurate within 3-4 hours after the incidence and drastically declines after 2 days (Frowd et al., 2005). However, traditional sketching methods such as Forensic artists or composition softwares (*FACES 4.0* 2016) both require a certain level of human expertise, where the former takes a long time to complete, and the latter can be less descriptive (Klum et al., 2013). To this end, we strive for a data-driven

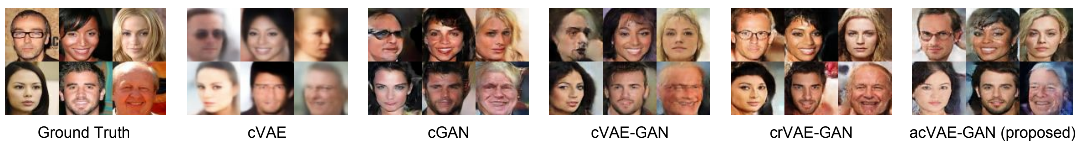
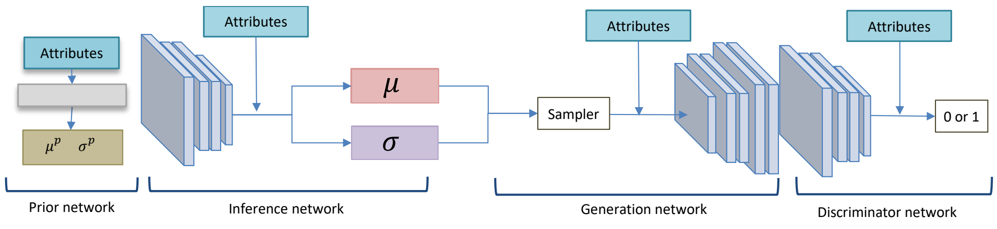
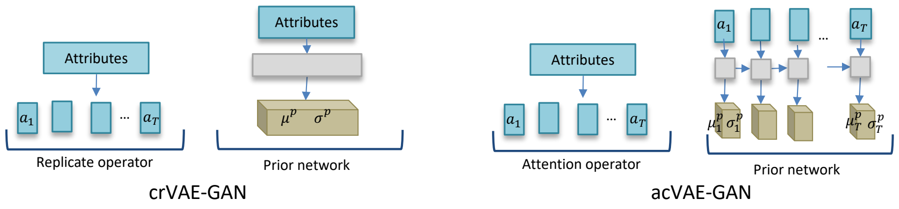


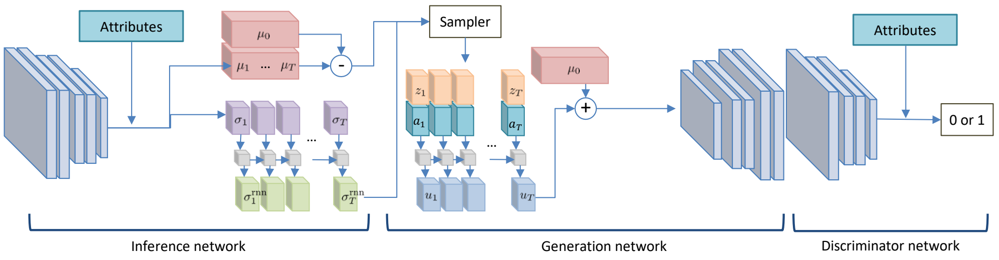
Figure 6.1: Generations using attributes from ground truth images. Comparison shows our attentive conditional channel-recurrent VAE-GAN's superiority in attribute-conditioned face synthesis.



(a) Prior, inference, generation and discrminator networks for cVAE-GAN



(b) Manipulation of attributes and prior for crVAE-GAN and acVAE-GAN



(c) Inference, generation and disriminator networks for cVAE-GAN and acVAE-GAN

Figure 6.2: Illustration of models used in our work.

tool to synthesize realistic and diverse face candidates basing on attribute information that returns instantaneous feedback to aid the identification of, e.g., a crime suspect[1].

Deep conditional generative modeling is a natural approach to such an attribute-conditioned face synthesis application. There has been tremendous recent progress in constructing such models for a variety of tasks. Applications related to generating face images have been especially well-studied, with techniques mainly derived from the 3 most dominating generative model frameworks, namely autoregressive models (Oord et al., 2016a), generative adversarial networks (GANs) (Goodfellow et al., 2014), and variational autoencoders (VAEs) (Kingma and Welling, 2013).

Conditional autoregressive models have been proposed to synthesize images from conditional variables such as text, key points, and the initial frame of a video (Reed et al., 2017). However, few works have applied this method on

1 This chapter is based on our WACV 2019 paper (Shang and Sohn, 2019).

faces, likely because the pixel level autoregression is computationally costly and slow in inference time without complex parallelization. Conditional GANs, on the other hand, have been more widely utilized on faces. In particular, (Gauthier, 2014) generates attribute-conditioned faces from scratch but the outputs (Figure 6.1) can be flawed with artifacts. Conditional VAE (Kingma et al., 2014) (cVAE)—an extension of VAE (Kingma and Welling, 2013)—is a conditional directed graphical model to approximate conditional data distribution through maximizing its variational lower bound. In (Yan et al., 2016), attributes and background masks are used as conditional variables in their cVAE formulation to generate faces. Although the reconstruction component in the variational lower bound gives the advantage of a more comprehensive input space coverage opposed to cGAN, cVAE suffers from a blurry generation when optimized in combination with a highly restricted KL regularization on the approximated posterior (Figure 6.1).

To alleviate the blurriness in generation while maintaining the probabilistic latent space, (Larsen et al., 2016) augments a VAE with an adversarial loss, arriving at VAE-GAN. As a follow-up, (Bao et al., 2017) formulates a conditional version of VAE-GAN, cVAE-GAN for face synthesis, where the conditional variables are identity labels, i.e., one-hot categorical vectors. However, this model is not directly applicable for our application as it would require 2^N-dimensional conditional variable to represent all possible attribute combinations, where N is the number of attributes. Therefore, in this work, we establish a cVAE-GAN baseline specifically targeting at generations out of high-dimensional attribute information (Figure 6.1), where the attribute condition is embedded in a N-dim binary vectors instead one-hot vectors.

On top of our baseline, we integrate the channel-recurrent architecture (Shang et al., 2018), where the latent space is divided into consecutive and non-overlapping blocks that are connected via a recurrent module, leading to conditional channel-recurrent VAE-GAN (crVAE-GAN). The benefit of doing so is two-fold. Firstly, by introducing a more complex architecture to maneuver the pathways into and out of the latent space characterized by simple diagonal Gaussians, the image generation quality is substantially enhanced (Figure 6.1). Secondly, the channel-recurrent architecture captures the high-level information in a course-to-fine manner, allowing more explainable latent features, which inspires us to propose our final model, the attentive conditional channel-recurrent VAE-GAN (acVAE-GAN). Our acVAE-GAN learns attention over the attribute vectors so that each attribute attends a specific latent block. In addition to being responsible for different attributes, the channel-recurrent layout also assigns different content for each latent block to modulate: some block predominately controls the global content, whereas others focus more on finetuning locally. This unique property of channel-recurrency enables us to envision an application tool to performs a 2-stage general-to-specific conditional face generation.

Lastly, we incorporate progressive-growth training (Karras et al., 2017) to the cVAE-GAN framework to facilitate higher resolution outputs.

The merits of our work are summarized as follows:

- Construct the cVAE-GAN baseline for attribute-conditioned face synthesis.
- Improve the generation quality of cVAE-GAN by integrating the channel-recurrent architecture, arriving at crVAE-GAN.
- Towards more interpretable models, learn an attention mask over attribute vectors so that each attribute "chooses" to be modulated by a specific latent block.
- Implement progressive-growth to our models to increase generation resolution.
- Envision a tool that performs 2-stage general-to-specific attribute-conditioned face generations.

6.2 RELATED WORKS

Traditional methods in controllable face synthesis (Yang et al., 2011; Laput et al., 2013; Kemelmacher-Shlizerman et al., 2014) typically require domain-specific knowledge (e.g., 3D face information) to control a limited set of attributes (e.g., illumination, expression, or age) by editing a reference image. Recently, deep generative models have been successful in learning visual object representations in a data-driven way while controlling many visual attributes for face editing and synthesis (Yan et al., 2016). In the regime of deep generative modeling, the most significant recent progress comes from autoregressive models, Generative Adversarial Networks (GANs), and Variational Autoencoders (VAEs). Framed initially as unsupervised learning, all three themes have also evolved to model conditional distributions with side information.

Pixel RNN and Pixel CNN (Oord et al., 2016b; Oord et al., 2016a; Salimans et al., 2017) explicitly model the pixel space distribution in an autogressive manner. Conditional versions (Reed et al., 2017; Gao et al., 2017) have been developed to generate a scene from text, a video from an initial frame, a segmentation mask from an image, etc. Although granting high-quality outputs and an exact log-likelihood, these methods do not learn a latent representation and demand heavy GPU parallelization for training.

GAN (Goodfellow et al., 2014) is another mainstream method to render sharp images in which a generator attempts to fool a discriminator with generations resembling examples from the input space. Conditional GANs (cGANs) (Mirza and Osindero, 2014; Kaneko et al., 2017; Choi et al., 2018) introduce conditional information to the generator, and the outputs are expected to not only imitate the inputs but also obey the conditions. Consequently, the discriminator must learn to distinguish correctly-matched real-condition pairs from both fake-condition pairs and wrongly-matched pairs (Reed et al., 2016). Adversarial training suffers from several notable drawbacks, such as limited input space coverage (Arora et al., 2017; Wu et al., 2016; Theis et al., 2016), generation artifacts, and lack of probabilistic latent representations. The training of GANs is also known to be unstable, albeit there have been many works attempting to address this, for both unsupervised and conditional cases (Arjovsky et al., 2017; Mao et al., 2016; Miyato and Koyama, 2018; Miyato et al., 2018). Figure 6.1 shows results from cGAN as done in (Gauthier, 2014), where we observe some of the aforementioned issues. Cycle-GAN (Zhu et al., 2017; Lu et al., 2017) is another conditional adversarial model basing on the concept of cycle-consistency. Since the publication of this work, there have also been efforts to build conditional cycle-GAN (Chen et al., 2020a). However, such models usually require an encoding image as the condition variable, hence it is not applicable in our case.

VAEs (Kingma and Welling, 2013) parameterize the inference and generation paths of a directed graphical model with DNNs and are trained to maximize the corresponding variational lower bounds of the data log-likelihood via stochastic backpropagation. A conditional VAE (Kingma et al., 2014) introduces a new node to the graphical model as conditions to assist and constraint the generation process. Despite the many merits of this method, e.g. training stability, fast inference, GPU efficiency, etc, the combination of KL regularization and reconstruction objectives in the VLB (Theis et al., 2016; Bousquet et

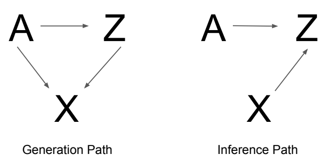


Figure 6.3: The generation and inference paths for the graphical model used in our work.

al., 2017) often result in blurry generations—Figure 6.1 displays an example of attribute-conditioned face generation (Yan et al., 2016).

Many research efforts have been made to alleviate VAEs' generation blurriness. One approach is to enrich the posterior/prior distributions or the model architecture itself (Kingma et al., 2016; Chen et al., 2016c; Rezende and Mohamed, 2015; Gregor et al., 2015; Gregor et al., 2016; Shang et al., 2018). In our work, we opt to enhance with the channel-recurrent autoencoding framework (Shang et al., 2018), motivated by its resulting latent semantics. The other approach combines VAE with autoregressive modules (Gulrajani et al., 2016) or adversarial training (Larsen et al., 2016), in the hope of taking the best from both worlds. Here we also employ an adversarial objective under a conditional setup, similar to (Bao et al., 2017). But (Bao et al., 2017) generates based on one-hot identity label, whereas we tackle the high-dimensional attribute information.

6.3 PRELIMINARIES

In this section, we first elaborate our cVAE formulation along with its graphical model, followed by an introduction of cGAN as well as their combination cVAE-GAN.

6.3.1 conditional VAE and Graphical Model

An important building block in our work is the conditional variational autoencoder (cVAE) (Kingma et al., 2014; Yan et al., 2016). In our work, we choose to follow the graphical model described in Figure 6.3, where the attributes A are always given in our case, Z are the latent variables and X images. The intuition behind our graphical model is to construct meaningful submanifolds within the latent space, where each submanifold associates with a designated combination of attributes. In other words, our graphical model choice assumes that $p(z|a)$ is more feasible than $p(z)$ as latent distributions, where a is the attribute condition. The variational lower bound can be derived as

$$\begin{aligned}
&\log(p(x|a)) = \log \int p(x,z|a)dz \\
&= \log \int p(x|a,z)p(z|a)\frac{q(z|x,a)}{q(z|x,a)}dz \\
&= \log \int p(x|a,z)\frac{p(z|a)}{q(z|x,a)}q(z|x,a)dz \\
&\geq \mathbb{E}_{q(z|x,a)}(\log p(x|a,z) - \log \frac{q(z|x,a)}{p(z|a)}) \\
&= -KL(q(z|x)||p(z|a)) + \mathbb{E}_{q(z|x,a)}[\log p(x|a,z)] = \mathcal{L}_{\text{cVAE}},
\end{aligned}$$

where the approximate posterior $q(z|x)$ and the prior $p(z|a)$ are modeled as diagonal Gaussians. As done in (Yan et al., 2016), we model the latent space as 1-dim vector space $z \in \mathbb{R}^c$, which is connected via fully-connected (FC) layers. The prior network also connects attribute vectors—an N-dim vector composed of ± 1 entries where N is the number of attributes, —to the distribution of $p(z|a)$ via an FC layer. The generations of cVAEs, as shown in Figure 6.1 and (Yan et al., 2016), are very blurry.

6.3.2 conditional GAN

In (Gauthier, 2014), the author uses attribute-conditioned cGAN to generate faces: on top of GAN, cGAN feeds attribute information $a(x)$ to the generator G and discriminator D. Also, the discriminator not only needs to detect generated images but also mismatched image-attribute pairs, which yields the following over-all game played by D and G:

$$\begin{aligned}
\min_G \max_D V(D,G) =& \mathbb{E}_{x\sim X}[\log D(x,a(x))] \\
&+\mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z,a),a))] \\
&+\mathbb{E}_{x,x'\sim X, x\neq x'}[\log(1 - D(x,a(x')))],
\end{aligned}$$

where z is sampled from a standard Gaussian distribution, a is the attribute condition and $a(x)$ is the attribute for x. The output images indeed reflect the

Step	Attributes
1	Heavy Makeup, Pale Skin, Rosy Cheeks
2	Brown Hair, Pointy Nose, Straight Hair
3	Arched Eyebrows, Attractive, Blond Hair
4	Blurry, Double Chin, High Cheekbones, Mouth Slightly Open, No Beard, Bags under Eyes
5	5'o clock shadow, Big Lips, Bushy Eyebrows, Chubby, Goatee, Gray Hair, Oval Face, Necktie
6	Bangs, Black Hair, Mustache, Receding Hairline, Smiling, Wavy Hair, Earrings, Hat
7	Bald, Eyeglasses, Male, Narrow Eyes
8	Big Nose, Lipstick

Table 6.1: In acVAE-GAN, each attribute attends a specific block.

specified attributes, but they contain many artifacts (Figure 6.1). Also, cGAN can suffer from low probability region coverage.

6.3.3 conditional VAE-GAN

The combination of VAE and GAN, VAE-GAN (Larsen et al., 2016) is proposed to take the best from both worlds: high input space coverage from VAE and sharp generations from GAN. In (Bao et al., 2017), a conditional version of VAE-GAN is crafted but limited to a single-class condition, i.e. identity label. But we have multiple attribute classes associated with a single image. Therefore, we propose our version of conditional VAE-GAN (cVAE-GAN), by directly combining cVAE from Section 6.3.1 and cGAN from Section 6.3.2 to obtain the following objective:

$$\begin{aligned} &\max \mathcal{L}_{\text{cVAE}} + \beta \mathbb{E}_{z\sim\{q(z|x,a(x)),p(z|a)\}} [\log D(z,a)] \\ &\max_D \mathbb{E}_{x\sim X} [\log D(x,a(x))] \\ &+\mathbb{E}_{z\sim\{q(z|x,a(x)),p(z|a)\}} [\log(1-D(z,a)] \\ &+\mathbb{E}_{x,x'\sim X,x\neq x'} [\log(1-D(G(x,a(x'))))], \end{aligned} \tag{6.1}$$

where β is the hyperparameter to weight the gradients from the adversarial loss. The model layout is shown in Figure 6.2a and generation examples in Figure 6.1, where the image quality is improved yet still distant from being realistic. As cVAE and cGAN are clearly inferior to cVAE-GAN in terms of visual quality, we regard cVAE-GAN as our main baseline to compare with for the rest of the paper.

6.4 METHOD

This section introduces the conditional channel-recurrent VAE-GAN (crVAE-GAN). It proposes an attention module such that each latent block focuses on a subset of attributes, arriving at the attentive conditional crVAE-GAN (acVAE-GAN), followed by a description of how to increase generation resolution via progressive-growth training.

6.4.1 Channel-Recurrency

To enhance the latent space construction, which in turn improves generation quality, (Shang et al., 2018) proposes the channel-recurrent architecture with which the inference and generation paths connect to the latent space. The distributions for the approximate posteriors and priors are still diagonal Gaussians, but since channel-recurrency imposes more complex manipulation to the latent space, the features, captured in a global-to-local, coarse-to-fine manner, become more abstract with more interpretable semantics.

Here, we integrate this technique to cVAE-GAN to similarly improve the conditional synthesis. The latent space is now a 3-dim space $z \in \mathbb{R}^{c\times w\times h}$. The prior network still connects the attribute vectors to the prior latent space via FC layers and reshape the resulting distributions to $\mu^p, \sigma^p|a \in \mathbb{R}^{c\times w\times h}$ (Figure 6.2b). Before being sent to the LSTM module, an attribute vector is simply repeated T times, i.e., $[a_1, a_2, \cdots a_T], a{=}a_i$, where T is the number of time steps. During generation, $z{=}[z_1, \cdot\cdot, z_T]$, with $z_i{\in}\mathbb{R}^{w\times h\times \frac{c}{T}}$, are sampled from $\mathcal{N}(\mu^p, \sigma^p)$ and concatenated with a_i at each time step i before passing through an LSTM to obtain a transformed representation $u = [u_1, \cdot\cdot, u_T]{=}\text{LSTM}(z, a)$, which is then projected back to the pixel space by a decoder. Similarly, during inference, we first transform the attribute vector with an FC layer $a'{=}\text{FC(a)} \in \mathbb{R}^{w}\times h\times \frac{c}{T}$, repeat the outputs T times, $[a'_1, a'_2, \cdots a'_T]$, and concatenate each to the T corresponding blocks within the convolutional features from the encoder. Then, the mean path performs convolution on the concatenated features; the variance path slices features back into T blocks, each referred to as σ_i, and feeds σ_i's into another LSTM to output σ_i^{rnn}.

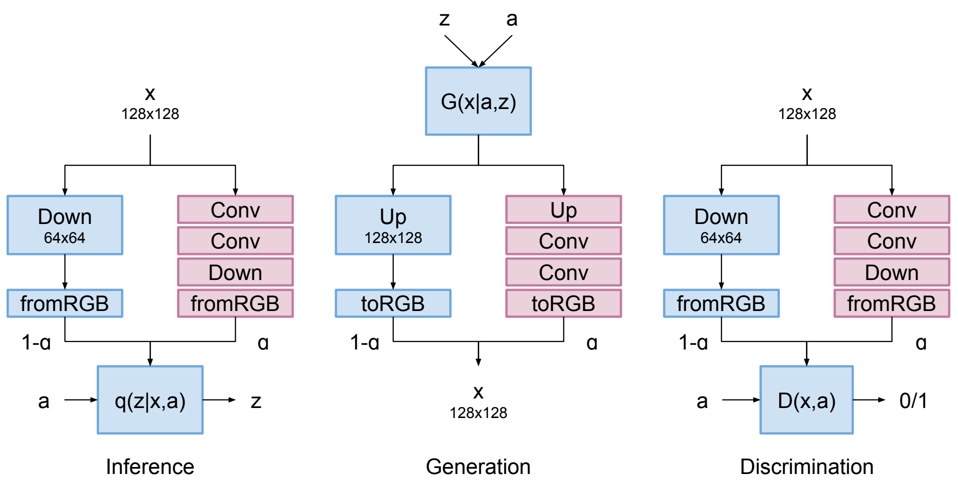


Figure 6.4: Illustration of progressive growing of acVAE-GAN. 64×64 acVAE-GAN is first trained without up/down sampling layers and used to initialize blue boxes of PG acVAE-GAN. The model is trained to generate 128×128 images with α being linearly interpolated from 0 to 1 over the first half of the training and remains 1 for the rest.

6.4.2 Attending Attributes

Repeating the same attribute vector T times for all of the LSTM time steps can appear redundant. To obtain better interpretable features, it is desirable to understand that for each block, which attributes are being modulated, motivating us to propose an *Attention Operator* over the attribute vectors. In essence, we learn a $T \times N$ mask matrix $\mathbf{M}$. Each row vector is constrained to be an T-dim probability vector, which is additionally regularized by minimizing its entropy $\mathcal{H}(\mathbf{M}_n)$ so that ideally, only one out of the T slots has a high probability. In other words, each attribute is learned to primarily focuses on one of the blocks.

Concretely, the Attention Operator first repeats the attribute vector T times to form a $T \times N$ attribute matrix $\mathbf{A}$, performs element-wise multiplication $\mathbf{A} \odot \mathbf{M}$, and separates the row vectors into $[a_1, a_2, \cdots, a_T]$. Thus instead of using the same attribute vector to all LSTM steps, we input a_t to the t-th step, reflecting the attributes attending this particular time step. Ideally, the attention from the attributes is divided evenly to all time steps: it is desirable for each time step to take in approximately N/T attributes rather than having a single time step getting most of the attribute information. Luckily, we discover no additional regularization is required to achieve such effects. For example, we summarize the attribute attention from our 8-time step acVAE-GAN in Table 6.1. The prior network (Figure 6.2b) now can also be meaningfully modeled with an

LSTM where the tth time step takes a_t and outputs μ_i^p, σ_i^p. The remaining of acVAE-GAN are the same as crVAE-GAN.

6.4.3 Progressive Growing Conditional Synthesis

While our model can generate 64×64 face images from attributes, generating higher-resolution remains a challenge. Following the idea of stackGAN (Zhang et al., 2016a), (Shang et al., 2018) achieves higher-resolution outputs by employing an upsampling network that takes generated low-resolution images as input. However, one drawback of such an approach is that the performance of the upsampling network is significantly limited by the initial image generation module as it is trained greedily without finetuning the entire system. Furthermore, the upsampling network is usually composed of multiple convolution and deconvolution layers, making the whole pipeline computationally inefficient.

Instead, we borrow the idea of progressive growing GAN (PGGAN) (Karras et al., 2017) for attribute-conditioned high-resolution image generation. Specifically, we first train a model to generate 64×64 images. Then, for the generation network, we append a nearest-neighbor upsampling layer and two 3×3 convolution followed by one image decoding layer (1×1 convolution with 3 output channels) to the last feature response map to generate 128×128 images. The original and generated images of size 128×128 are then fed to the encoder and discriminator heads, respectively, which are composed of 1×1 convolution followed by two 3×3 convolutions and average pooling layer with a pool size of 2 to generate 64×64 feature response maps. We illustrate the construction of progressive growing acVAE-GAN in Figure 6.4.

Similarly to (Karras et al., 2017), we train our progressive growing model by linearly interpolating the generated images of naive upsampling and the upsampling network. Note that our approach does not require to start from extremely low-resolution images (e.g., 4×4) as 64×64 generations can be directly achieved with decent quality. The overall training objective still follows that in Equation (6.1).

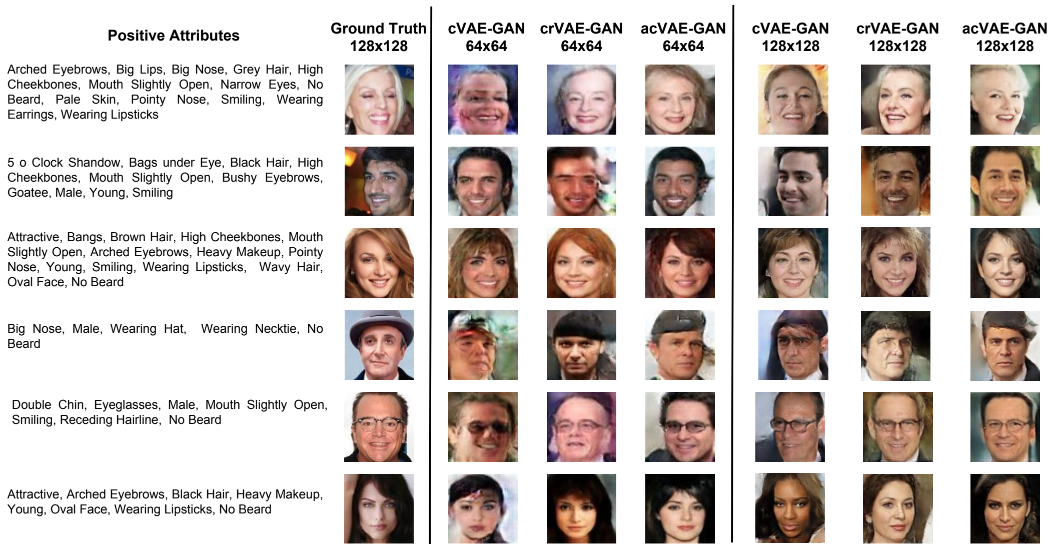

Positive Attributes
Ground Truth 128x128
cVAE-GAN 64x64
crVAE-GAN 64x64
acVAE-GAN 64x64
cVAE-GAN 128x128
crVAE-GAN 128x128
acVAE-GAN 128x128
Arched Eyebrows, Big Lips, Big Nose, Grey Hair, High Cheekbones, Mouth Slightly Open, Narrow Eyes, No Beard, Pale Skin, Pointy Nose, Smiling, Wearing Earrings, Wearing Lipsticks
5 o Clock Shandow, Bags under Eye, Black Hair, High Cheekbones, Mouth Slightly Open, Bushy Eyebrows, Goatee, Male, Young, Smiling
Attractive, Bangs, Brown Hair, High Cheekbones, Mouth Slightly Open, Arched Eyebrows, Heavy Makeup, Pointy Nose, Young, Smiling, Wearing Lipsticks, Wavy Hair, Oval Face, No Beard
Big Nose, Male, Wearing Hat, Wearing Necktie, No Beard
Double Chin, Eyeglasses, Male, Mouth Slightly Open, Smiling, Receding Hairline, No Beard
Attractive, Arched Eyebrows, Black Hair, Heavy Makeup, Young, Oval Face, Wearing Lipsticks, No Beard

Figure 6.5: We select attribute combinations from the testing set and compare 64×64 and 128×128 resolution image generations of cVAE-GAN (baseline), crVAE-GAN and acVAE-GAN, along with the ground truth images.

6.5 EXPERIMENTS

We introduce the dataset used in our experiments and describe important implementation details. For evaluation, we generate 64×64 attribute-conditioned face images, then progressively grow the resolution to 128×128. We perform a qualitative visual examination and quantitative assessments, namely the inception scores, human preferences, and attribute classification accuracy. Finally, we conduct more qualitative analysis, such as conditional latent space interpolation and progressively adding attributes to an image.

6.5.1 CelebA Dataset and Implementation Details

Our experiments are on the CelebA (Liu et al., 2015) dataset, containing 163,770 training, 19,867 validation, and 19,962 testing images of face. The face ROIs are cropped and scaled to 64×64/128×128. Random horizontal flipping is used during training as data augmentation. There are 40 binary attributes annotated in CelebA (see Table 6.1), making it a perfect venue for our experiments.

There have been many works to improve the stability of adversarial training, many of which we have experimented with, such as projective discriminator (Miyato and Koyama, 2018), self-attention (**zhang2018self**), hinge ad-

Models	64×64	128×128	128×128
cVAE-GAN	37.48±0.96	93.74±2.25	23%
crVAE-GAN	54.66±0.67	87.51±1.46	28%
acVAE-GAN	**55.12**±0.64	**102.23**±1.81	**47%**

Table 6.2: Inception Scores on 64×64/128×128 generated images from testing set attributes. We perform 10-fold calculation and report the mean±std. We also report the percentage humans prefer a model in a 3-way classification setup.

versarial loss (**lim2017geometric**), etc. In the end, our model adapts three main methods to stabilize the optimization: batch discriminator (Salimans et al., 2016), controlled discriminator update (Larsen and Sønderby, 2016), and mutual information regularization (Shang et al., 2018).

The inference, generation, and discriminator networks of our models share a common convolutional encoder/decoder architecture, composed of convolutional layers, batch normalization layers (Ioffe and Szegedy, 2015), and activation layers (**maas2013rectifier**; Shang et al., 2016). See the code in the Supplemental Materials for details.

Stocahstic gradinet descent is done using ADAM (Kingma and Ba, 2014) with $\epsilon = 1 \times 10^{-8}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ for 150 epochs. The initial learning rate is 0.0003 for (cVAE-GAN) and 0.001 for the rest. We follow the control protocol in (Larsen and Sønderby, 2016) to update the discriminator with a variation of ADAM that imposes a threshold for updating: if the classification accuracy for a batch consisting of a third of real pairs, a third fake of pairs, and a third of mis-matched pairs is over 90%, we skip the update. During progressive-growth training, the resolution transition is linearly done in 75 epochs. Training code is in the Supplemental Materials.

6.5.2 Attribute-Conditioned Face Synthesis

Evaluation of the conditional generative models in our work primarily focuses on the visual fidelity and the faithfulness to the assigned attributes. To this end, we first visually compare the baseline cVAE-GAN with crVAE-GAN and acVAE-GAN in Figure 6.5, where our proposed model produces more photo-realistic generations that satisfy the attribute conditions, even for some of the more challenging ones such as eyeglasses and hats.

		5 Shadow	Arch. Eyebrows	Attractive	Bags Un. Eyes	Bald	Bangs	Big Lips	Big Nose	Black Hair	Blond Hair	Blurry	Brown Hair	Bushy Eyebrows	Chubby	Double Chin	Eyeglasses	Goatee	Gray Hair	Heavy Makeup	H. Cheekbones	Male
64×64	Real Images	93	81	81	84	98	95	71	82	88	95	95	88	92	95	96	99	96	98	90	87	97
	cVAE-GAN	92	78	80	82	98	96	69	79	94	95	94	82	91	94	95	98	96	97	91	86	98
	crVAE-GAN	93	79	81	83	98	97	70	83	91	96	94	85	93	95	95	99	96	98	91	88	98
	acVAE-GAN	92	81	81	84	98	97	71	83	87	96	94	85	93	95	95	99	96	98	90	88	99
128×128	Real Images	94	83	82	84	98	96	70	84	89	95	95	88	92	95	96	99	97	98	91	87	97
	cVAE-GAN	92	83	84	83	98	98	70	83	87	97	95	87	93	95	95	99	96	98	92	88	99
	crVAE-GAN	91	81	81	83	98	98	70	83	89	97	94	85	93	95	95	99	96	98	91	88	98
	acVAE-GAN	92	83	84	83	98	98	70	83	87	97	95	87	93	95	95	99	96	98	92	88	99

		Mouth S. O.	Mustache	Narrow Eyes	No Beard	Oval Face	Pale Skin	Pointy Nose	Reced. Hairline	Rosy Cheeks	Sideburns	Smiling	Straight Hair	Wavy Hair	Wear. Earrings	Wear. Hat	Wear. Lipstick	Wear. Necklace	Wear. Necktie	Young	Average
64×64	Real Images	92	96	86	94	74	96	76	93	95	97	92	81	80	87	98	93	86	94	87	90.4
	cVAE-GAN	94	95	86	95	70	95	72	93	90	96	94	77	69	81	98	94	86	94	84	89.1
	crVAE-GAN	96	96	87	95	73	96	73	93	94	96	95	78	78	84	98	94	86	94	88	90.3
	acVAE-GAN	97	96	89	95	72	96	74	93	94	96	96	79	76	84	98	94	86	94	86	**90.4**
128×128	Real Images	93	96	87	95	75	97	77	93	94	97	92	81	82	89	99	94	86	95	88	90.9
	cVAE-GAN	98	96	89	96	75	96	76	93	93	96	97	80	79	88	99	94	85	95	87	90.7
	crVAE-GAN	97	96	88	94	73	96	74	93	93	96	96	79	77	87	98	94	84	94	87	90.6
	acVAE-GAN	98	96	89	96	75	96	76	93	93	96	97	80	79	88	99	94	85	95	87	**91.0**

Table 6.3: We train an independently trained classifier, we test attribute classification results on real images as benchmark and generated ones from the baseline cVAE-GAN, crVAE-GAN and acVAE-GAN. All models perform competatively, likely due to the adversarial loss and the proposed graphical model. All numbers are in %.

We also measure the inception scores (Salimans et al., 2016) for 64×64 and 128×128 images. The classification model used here is a VGG11 trained on CASIA (Yi et al., 2014) for face recognition with input resolution 128×128. During testing, we generate images conditioning on all attribute combinations from the testing set, randomly divide them to 10 subsets, obtain inception scores for these subsets and report their mean±standard deviation in Table 6.2. However, since inception score is not an absolutely truthful metric on visual quality (**barratt2018note**), we present 200 128×128 generated tuples from the 3 models with the same set of attributes to mechanical turks, out of which they select the one with the highest visual quality. The results are also summarized in Table 6.2. Our proposed model substantially outperforms the baseline in both metrics.

To quantitatively judge the attribute qualities, we train an attribute classifier on 64×64/128×128 CelebA images. Our classifier is composed of a standard convolutional encoder and a fully connected layer ending with 40 dimension binary predictions for the 40 attributes. We follow the standard training-validation procedures and report the testing results in Table 6.3. Note that our classifier achieves competitive accuracy comparing (Liu et al., 2015). Next, we

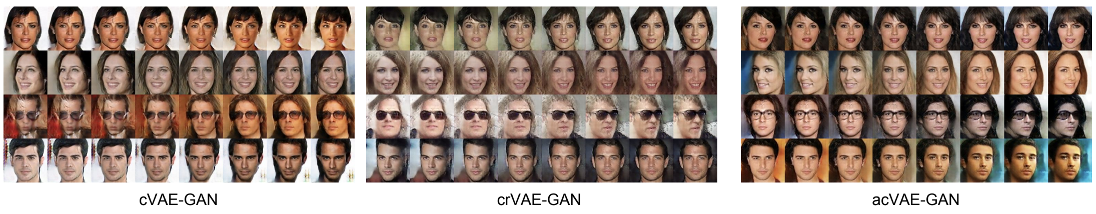


Figure 6.6: We generate faces by linearly interpolating latent variable between 2 samples drawn from the conditional priors conditioned with the same attributes. The conditional prior latent space in all models appears to deliver smooth transitioning while maintaining the attributes; acVAE-GAN especially excels at traversing across diverse samples and at the same time, maintaining image qualities.

run our attribute classifiers on the generated images, with accuracy results in Table 6.3. All of the models reach promising accuracy, meaning that our graphical model design is very effective in delivering the attributes, among which our proposed model has the highest score.

6.5.3 Content Interpolation

For our goal of generating diverse and realistic images given a set of attributes, it is important to learn a prior latent space that densely covers valid images in the pixel space conditioned on the attributes. This section investigates this quality of the latent spaces learned by our models by checking how well two distinct modes on the latent submanifold conditioned on the same attributes can mix, also an indicator of the representation abstractness (**bengio2013better**). Concretely, we select four sets of attribute combinations from the testing set, obtain the conditional prior space for these combinations, and sample 2 examples each to represent 2 modes Z_1 and Z_2 on the latent submanifold. Then we traverse from Z_1 to Z_2 using the interpolation formula

$$Z_i = \cos^2(\psi)Z_1 + (1 - \cos^2(\psi))Z_2, \psi \in [0, \pi/2].$$

The generation network projects the interpolated latent codes to the pixel space. Figure 6.6 shows, likely thanks to our proposed graphical model learning conditional submanifold distributions, the resulting latent representations from all models can be consistently mixed in a semantically meaningful manner. Our proposed acVAE-GAN especially excels at transiting through very distinct looking modes, such as the pose variations in the 2nd row, the glasses changing from clear to dark in the 3rd row, and the background color shifting

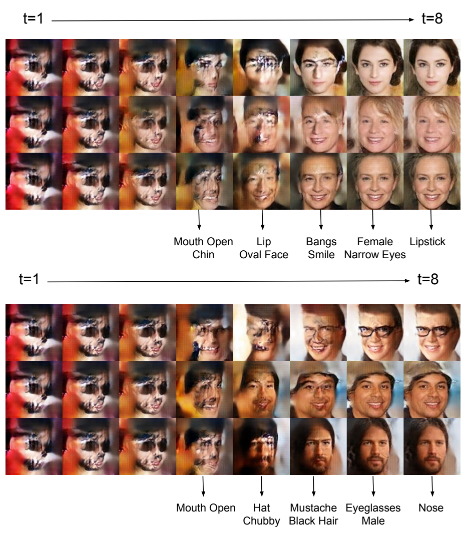


Figure 6.7: We progressively construct the conditional prior latent space by sending in the attended attributes for each time step, and then fill in the latent representations sequentially. The first few time steps do not carry enough content information to distinctively show the progression; for the latter ones, after sampling each latent block, its associated attributes indeed appear (correspondence see Table 6.1). We list out a few examples such as gender, hair color, etc.

from the last row. Meanwhile, some of the generations from the other models associated with certain attributes appear to be repetitive, such as the 2nd row of cVAE-GAN and crVAE-GAN; some other generations from cVAE-GAN are prone to artifacts such as the 3rd and 4th row.

6.5.4 Generation Progression

Our proposed acVAE-GAN has each attribute to focus its attention to a specific time-step latent block to achieve more interpretable high-level features, in the sense that we in theory can pinpoint the exact block modulating a given attribute. To verify this assertion, we attempt to visualize the emergence of attributes by progressively sampling each latent blocks. Recall that the conditional prior space is generated via sending the attended attribute a_t through an LSTM module at tth time-step, which then outputs the distribution $\mathcal{N}(\mu_t^p, \sigma_t^p)$ of the corresponding prior latent block. From $\mathcal{N}(\mu_t^p, \sigma_t^p)$, we sample z_t and, in combination of the previously sampled z's, decode $[(z_1, a_1), \cdots (z_i, a_i), (0,0), \cdots, (0,0)]$ via the generation network back to the pixel space. We plot the $t = 1 \cdots 8$ generations in Figure 6.7. The first 3 time steps do not carry enough content

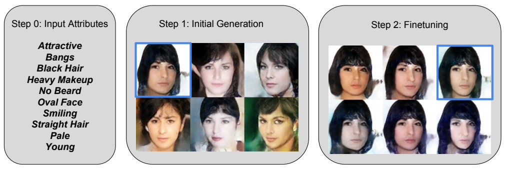


Figure 6.8: A 2-stage generation tool where the model first outputs a diversity of faces following the given attributes and then, after selecting an initial output, resamples one of the blocks in charge of minor modifications to give more finetuned options.

information to distinctively show the progression. However, for the latter ones, after sampling each z_t, its associated attributes a_t do emerge (attribute correspondence see Table 6.1). For example, we can already see the mouth open in step 3; hat forms in step 4 and glasses step 7; the gender is finalized at step 7; mustache and bangs become clear at step 6; step 8 brings subtle changes over the lip color and nose shape. The visualization demonstrates that up to a certain degree we can confirm a latent block is indeed modulating its attending attributes.

6.6 APPLICATION

The recurrent learning of latent blocks enables each of them to be in charge of a different aspect of the input content (Shang et al., 2018). In particular, some latent blocks can impose a more drastic global impact and some others in a more subtle way. We leverage this unique feature from channel-recurrency and propose the following 2-stage generation pipeline. As a preparation step, through trial-and-error, we locate the latent block that performs a minor adjustment on the content of the generation; in the case of the model used in Figure 6.7, a progressive-growing acVAE-GAN, is z_5. The first step generates a diversity of samples from the corresponding prior latent space based on given attributes, and the user selects the most desirable ones. For finetuning, the 2nd step resamples multiple z_5 of the selected example while fixing the rest of z_t's, and the user finalizes on the most closely-matching image. An example is demonstrated in Figure 6.8, where the first step outputs distinct faces obeying the same set of attributes and the second step finetunes face shape, skin tone, hair details, etc. The user can further use existing neural editing tools (**brock2016neural**) to refine the image, but it is out of scope of this work.

6.7 CONCLUSION

We propose an attentive conditional channel-recurrent VAE-GAN for high-quality attribute-to-face synthesis while learning better interpretable high-level features. We also incorporate progressive-growth training to generate higher-resolution images. We demonstrate the superiority of our models, both in quantitative and qualitative evaluations. In application, we envision a tool for a general-to-specific 2-stage attribute-conditioned face synthesis. Future research includes extending our framework in a semi-supervised manner and extrapolating our models to other tasks.

7

DECOMPOSE VIDEO REPRESENTATIONS FROM TEMPORAL COHERENCE AND DYNAMICS

7.1 INTRODUCTION

We have explored both unconditional and conditional modeling in image domain. Now, we switch our focus to another dominating data domain of the digital world, videos. Videos constitute a large portion of Internet traffic, posing great challenges and opportunities for video understanding. Yet, representation learning for videos is not as mature as for images. Most video applications parse videos frame-by-frame via an image-based model (Perazzi et al., 2016; Thies et al., 2016), ignoring their rich temporal structure. Two main factors have hindered progress in video understanding: videos contain orders of magnitude more raw information than images and annotation of videos, especially in a per-frame manner, requires excessive labor. On the bright side, video data possesses appealing intrinsic properties that can be leveraged as important inductive biases (Locatello et al., 2018). For one, since most video footage evolves continuously, raw video contains repeated objects and scenes that are highly correlated and redundant in appearance. For another, videos present strong semantic coherence in terms of object motion along the temporal direction, which can be modeled as rearrangement of objects via inter-frame transformation.

Many computer vision applications make effective use of the video appearance redundancy and temporal semantics coherence. *E.g.*, most video compression algorithms (Le Gall, 1991; Chen et al., 2001) only sparsely store full reference frames along with the motion differences for the in-between frames;

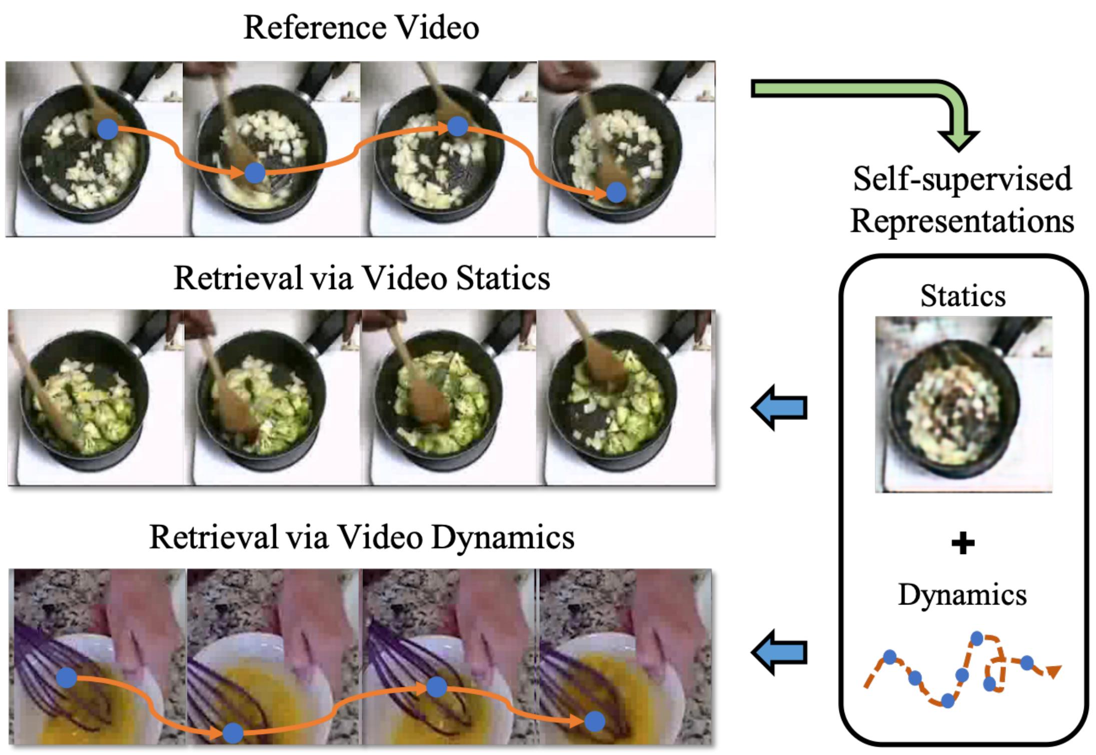

Reference Video
Self-supervised Representations
Retrieval via Video Statics
Statics
+
Dynamics
Retrieval via Video Dynamics

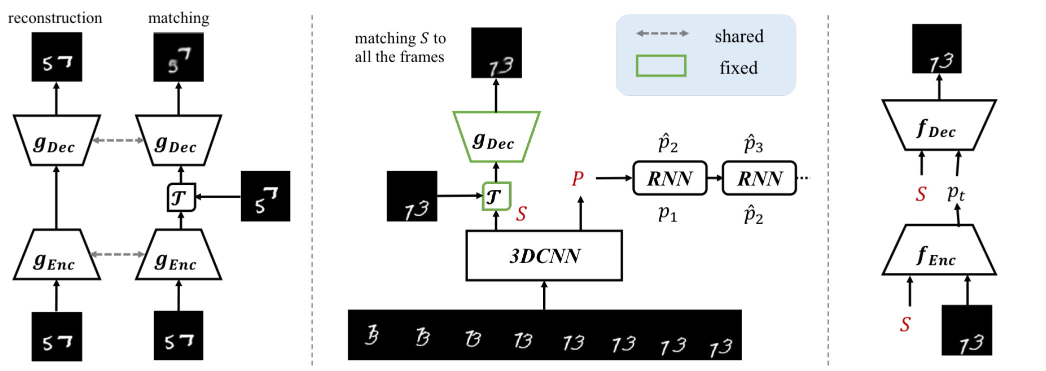
Figure 7.1: An overview of the proposed representation learning of video statics and dynamics. The videos retrieved based on the video statics (the 2nd row) and dynamics (the 3rd row) resemble the input reference video (the 1st row) in appearance and motion, respectively.

state-of-the-art video predictions (Denton, 2017; Li and Mandt, 2018; Hsieh et al., 2018) are accomplished by disentangling content and motion. However, these algorithms are application-specific and do not output broadly applicable representations.

In terms of representation learning, current efforts primarily rely on temporal coherence, specifically inter-frame correspondences, as self-supervision signals (Li et al., 2019; Wang et al., 2019a). However, their end products are still image features benefiting image-level downstream tasks. Meanwhile, less research has been carried out to model the temporal coherence as video-level representations.

This chapter fills these gaps[1]. Assuming a video clip is single-shot by one camera, we consider the interplay of video statics (the temporally coherent part) and dynamics (the temporally evolving part) as an important inductive bias and explicitly extract decomposed video-level representations of each part in an unsupervised fashion, as illustrated by Figure 7.1. Here, the video stat-

1 This chapter is based on our paper (Shang et al., 2020b) on arxiv.



(a) Transformer (b) Video Encoder (c) Frame Autoencoder

Figure 7.2: Our framework learns video statics and dynamics through (a) a spatial transformer $\mathcal{T}$ trained to warp between frames, (b)-left a video encoder extracting video statics S to represent the temporal coherent content that reproduces all frames to the best extend via $\mathcal{T}$, (c) a conditional (condition on S) autoencoder whose latent space corresponds to the frame-level dynamics p_t, and (b)-right the video encoder also generates video dynamics P which is encouraged to *reason* about temporal dynamics by predicting frame-level dynamics p_t.

ics is a compact representation holding visual resemblance to all frames when decoded. The video dynamics is another single representation encoding the abstract motion trajectories. Each can retrieve videos with similar appearance or dynamics. To extract video statics, we build on a key intuition that video statics should describe the common appearance to the best extent. To this end, we first train a spatial transformer to match frames to each other. The transformer then assists in learning a single common video-level representation, which can be deformed back to each frame. Video dynamics are hierarchically distilled from the frame-level up to the video-level. A low-dimensional frame-level dynamics representation is learned such that when being conditioned on video statics, it can reconstruct the corresponding frame. We then abstract the entire motion trajectory into a compact video-level dynamics representation via a frame-level dynamics prediction task.

To the best of our knowledge, we are the first to propose unsupervised learning of decomposed video representations by transforming video-level representations to frame-level. Our main contributions are:

- We leverage video temporal coherence to learn a video statics representation. In this process, we propose a novel transformation framework going from video-level to each individual frame-level representation.

- Conditioning on video statics, we learn an abstract video-level dynamics representation that describes motion trends independent of spatial locations.
- We demonstrate a competitive performance of our framework when applying the learned representations to a variety of applications.

7.2 RELATED WORK

Unsupervised video representation learning can be roughly categorized into two groups based on the signals used to learn from: temporal coherence or dynamics. There are also efforts to take advantage of both.

Works leveraging video statics include enforcing temporal feature steadiness (Jayaraman and Grauman, 2016; Mobahi et al., 2009), learning from egomotion (Agrawal et al., 2015; Jayaraman and Grauman, 2015) or pose transformation (Purushwalkam and Gupta, 2016), and matching correspondences (Schmidt et al., 2016; Wang et al., 2019a; Li et al., 2019; Wang and Gupta, 2015). However, statics cues alone generally provide frame-level feedback leading to image features and tend to overlook inter-frame connections. Our framework, although also leverages inter-frame spatial transformations, upgrades the output to an explicit representation for the entire video.

The temporal dimension of videos contains rich structural information that provides signals for many unsupervised frameworks. Frame interpolation not only builds context-aware features but is also practically useful, e.g., to increase frame rate (Long et al., 2016; Niklaus and Liu, 2018; Janai et al., 2018). Sequence ordering is another venue to impose temporal understanding (Misra et al., 2016; Lee et al., 2017; Fernando et al., 2017; Kim et al., 2019). Moreover, video prediction is often used to extract abstract features for high-level tasks (Srivastava et al., 2015; Han et al., 2019; Diba et al., 2019). Our framework also involves understanding video dynamics, using a training objective that shares similarities with the contrastive predictive loss in (Han et al., 2019). However, unlike (Han et al., 2019), our focus is not the prediction of the future but rather a compact summary of the current dynamics.

Several approaches combine ideas from both the temporal coherence and dynamics paradigms. Identification of time-invariant versus time-varying components has been at the core of many applications for video generation (Vondrick et al., 2016), video prediction (Denton, 2017; Hsieh et al., 2018; Li and Mandt, 2018; Villegas et al., 2017) and video style-transfer (Tulyakov et al., 2018; Siarohin et al., 2019). In terms of representation learning, (Wang et al., 2019b) regresses both motion and appearance statistics but requires RGB as well as optical flow inputs. In contrast, our framework considers RGB frames only. More importantly, we produce explicit representations reflecting video statics and dynamics, which can be directly used in a broad range of downstream tasks.

7.3 METHODOLOGY

We propose a unified unsupervised framework for video-level representation learning from both video statics and dynamics. Essentially, we decompose a video sequence $\mathbf{V} = \{x_t\}$ for $t \in \{1 \cdots T\}$, with x_t denoting the frame at time t, into two components: a video statics S, representing temporal coherent content and a video dynamics P, representing dynamics content.

We leverage inter-frame spatial transformers to distill video statics, detailed in Section 7.3.1. Ideally, the remaining information is thus related to video dynamics. To learn a video-level dynamics representation, inspired by the information bottleneck principle (Alemi et al., 2016), we formulate a hierarchical procedure that: (a) learns an individual frame-level representation p_t, which is the complement of the extracted video statics from each frame, and (b) learns a video-level representation P that summarizes all the frame-level dynamics p_t. These steps are detailed in Section 7.3.2. We use the Moving MNIST synthetic dataset (Srivastava et al., 2015) (MMNIST) as running examples to better describe our framework, since its statics and dynamics, *i.e.*, the digits and their trajectories, are clearly defined.

7.3.1 Learning Video Statics

A compact statics representation is encoded from the whole video, which can be spatially transformed into each frame. To learn such a representation, we propose two building blocks as shown in Figure 7.2(a) and (b): a *transformer* module $\mathcal{T}$ that performs inter-frame spatial transformation, and a 3D convolutional neural network (3DCNN) *video encoder* Enc_v that outputs the video-level statics S.

Inter-Frame Matching

A spatial transformer $\mathcal{T}$ is an operator that spatially rearranges pixels or features of one image to match another if the two share common content. The $\mathcal{T}$ itself can be any module that outputs either the parameters of a geometric transformation, such as an affine transformation, or a general pixel-to-pixel mapping, such as optical flow.

In the upcoming step, we rely on a transformer $\mathcal{T}$ that outputs a mapping $\mathcal{G}$ that matches video statics S to frames upon receiving S and frame features. Therefore, we first learn $\mathcal{T}$ to perform inter-frame transformation by minimizing the matching error of transforming one frame to another.

Concretely, we apply an autoencoder and conduct the transformation on its bottleneck latent space instead of the raw image space (Figure 7.2(a)-left). The encoder g_{enc} embeds an input frame x_i and a target frame x_j to

$$s_i{=}g_{\text{enc}}(x_i), s_j{=}g_{\text{enc}}(x_j), s_i.s_j \in \mathbb{R}^{c_{\mathcal{T}} \times h_{\mathcal{T}} \times w_{\mathcal{T}}}. \tag{7.1}$$

Then $\mathcal{T}$ produces the transformation mapping $\mathcal{G}_{ij}$ that acts on s_i to obtain $\hat{s}_j$. The decoder g_{dec} projects $\hat{s}_j$ to $\hat{x}_j$, the final transformed x_i to match the target frame x_j. Overall, we train the reconstruction and the transformer $\mathcal{T}$ (via spatial transformation) objectives:

$$\mathcal{L}_{\mathcal{T}}^{i,j}{=}\lambda_r \underbrace{|x_i{-}g_{\text{dec}}((s_i))\|^2}_{\text{reconstruction}} {+}\lambda_t \underbrace{\|x_j{-}g_{\text{dec}}(\mathcal{G}_{ij}(s_i))\|^2}_{\text{spatial transformation}}, \tag{7.2}$$

where λ_r and λ_t are coefficients to balance the losses. More implementation details on the transformers are in Section 7.4.2.

Video-Frame Matching

The $\mathcal{T}$ provides us with a latent space in which video frames can be reconstructed and spatially modified to generate other frames. To learn S, we seek a single video representation in the same latent space that can summarize temporal coherence across all frames to the best extent. To this end, we construct a 3DCNN video encoder Enc_v (see Figure 7.2(b)) that takes in raw video sequences and outputs $S \in \mathbb{R}^{c_{\mathcal{T}} \times h_{\mathcal{T}} \times w_{\mathcal{T}}}$. To ensure the re-projection of S through

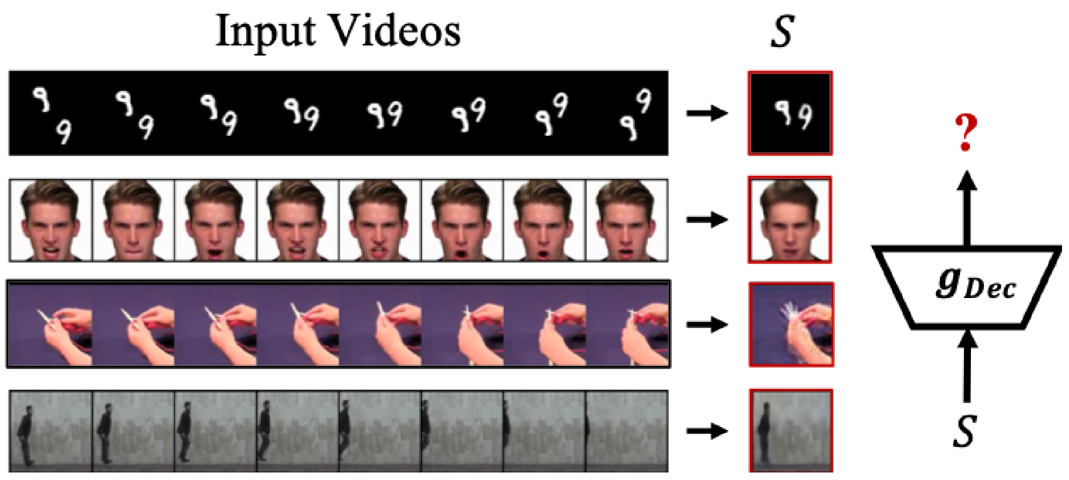


Figure 7.3: Visualization of video statics from different datasets by passing S through g_{dec}. S extracts time-invariant appearance.

the decoder g_{dec} back to the frames with minimal reconstruction error, we minimize:

$$\mathcal{L}_{\text{statics}} = \frac{1}{T}\sum_{t=1}^{T} \|x_t - g_{\text{dec}}(\mathcal{G}_{vt} \circ S)\|^2, \tag{7.3}$$

where we use $\mathcal{G}_{vt} = \mathcal{T}(S, s_t)$ to denote the transformation parameters between S and each frame feature s_t, and $\mathcal{G}_{vt} \circ S$ represents to warp S using $\mathcal{G}_{vt}$. Formulations of $\mathcal{G}_{vt}$ are specified in Section 7.4. Figure 7.3 shows that on different datasets our video statics representations express the common appearance of the sequences, e.g., lighting, objects and scene background.

7.3.2 Learning Video Dynamics

We regard video dynamics as a video-level representation that encodes temporal evolution. There is no explicit way to annotate such information, making unsupervised learning an attractive approach.

Assuming that the video statics part S extracts the content shared by all frames, we can learn a frame-level dynamic representation p_t describing the transition from S to each x_t, such that together with S, it can reconstruct x_t. Consequentially, p_t encodes the relative spatial locations of content in x_t w.r.t. content in S. We, on the other hand, desire an abstraction of motion trajectories that is *independent of locality*. Therefore, we further construct a video-level dynamics representation P using an objective function that predicts p_t. Taking MMNIST as an example, p_t describes the spatial transition of the digits and P the trajectory of how the digits move in a video. The rest of this section presents our unsupervised approach to learning the frame-level and then video-level dynamics representations.

Frame-wise dynamics representation

The information bottleneck principle (Saxe et al., 2018; Alemi et al., 2016) speculates that an autoencoder with a low-dimensional latent space condenses task-relevant information and discards the irrelevant part. Inspired by this principle, we construct a frame-level conditional autoencoder (Kingma et al., 2014) (see Figure 7.2(c)), where S is provided to both the encoder and decoder and the latent space is *much smaller* than the input space. Thus, the reconstruction objective encourages the network to take full advantage of S and squeeze information not conveyed through S into p_t. Therefore, p_t is forced to encode mostly dynamics content. The objective is formulated as follows:

$$\mathcal{L}^t_{frame} = \|x_t - f_{\text{Dec}}(p_t, S)\|^2, \text{ where } p_t = f_{\text{Enc}}(x_t, S). \tag{7.4}$$

Video-Level dynamics representation

Furthermore, we aim to attain a video-level dynamics representation P that summarizes motion trajectory independent of location. We achieve this using a latent prediction task. Concretely, the video encoder Enc_v outputs P in conjunction with S. We then condition on p_1, providing the initial location information, and P predicts future frame dynamics $p_{t>1}$ autoregressively. We apply a contrastive loss (Gutmann and Hyvärinen, 2010) for the latent prediction task, based on contrastive predictive coding (Oord et al., 2018; Sohn, 2016). In theory, the contrastive loss maximizes a lower bound on the mutual information between frame-level and video-level dynamics (Hjelm et al., 2018). Therefore, for a video v at time step t, the loss encourages the predicted $\hat{p}_{v,t}$ to be only close to the true frame-level representation for v at t, $p_{v,t}$, but far away from the rest. That is, distant from frame representations in either a different video ($p_{n,i}$ for $n \neq v$), or at a different time step ($p_{v,i}$ with $i \neq t$). In practice, the contrastive loss takes a batch of size $N \times (T-1)$ from N videos, consisting of $\{p^t_v\}_{v=1:N,t=2:T}$:

$$\mathcal{L}_{pred} = -\sum_{v,t} \left[\log \frac{\exp(\hat{p}^{\text{tr}}_{v,t} p_{v,t})}{\sum_{n,i} \exp(\hat{p}^{\text{tr}}_{v,t} p_{n,i})} \right], \tag{7.5}$$

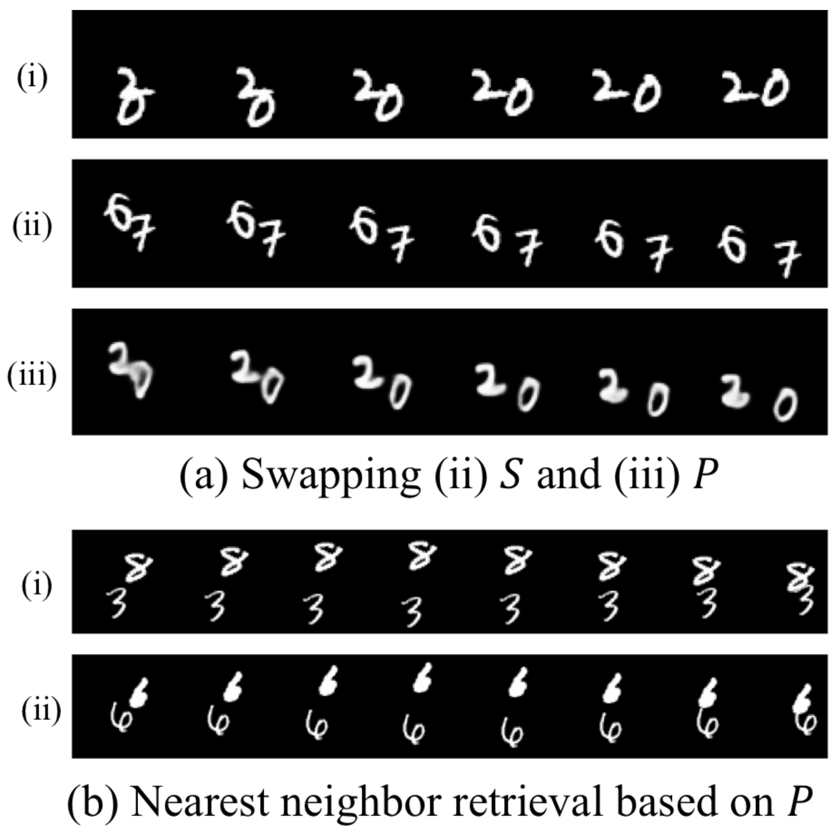

(i)
(ii)
(iii)
(a) Swapping (ii) S and (iii) P
(i)
(ii)
(b) Nearest neighbor retrieval based on P

Figure 7.4: (a): combine video statics from (i) and dynamics from (ii) to compose (iii). (b): visualization of retrieved MMNIST sequences based on video dynamics, (i) reference (ii) retrieved.

where $\hat{p}_t = f_{auto.}(P, p_{<t})$ is generated via an autoregressive model $f_{auto.}$ for $t = 2, \ldots, T$. Intuitively, this objective encourages the model to *reason* about the dynamics progression over frames using video dynamics.

7.4 IMPORTANT IMPLEMENTATION DETAILS

In this section, we discuss our training pipeline, transformers, and model architecture. Then we explore the synthetic MMNIST dataset to provide insights into our proposed framework. Full implementation details are in the Appendix.

7.4.1 Learning Pipeline

The full training pipeline of the proposed framework starts with the training of the transformer with the objective of matching one frame from a video sequence to (not necessarily consecutive) another in equation 7.2. Next, we freeze the transformer and train the rest of the model (see Figure 7.2(b)). Precisely, we first backpropagate the error derivatives for S in equation 7.3, then detach S from the computational graph while backpropagating to the frame-level models with the objective in equation 7.4. Lastly, we detach the obtained p from the computational graph to update Enc_v with the contrastive objective in equation 7.5

7.4.2 Self-supervised Transformer $\mathcal{T}$: BlockSTN

In our experiments, we adopt the block spatial transformation networks (blockSTN) as our transformer $\mathcal{T}$, an extension of the spatial transformation networks (STN) (Jaderberg et al., 2015). An STN applies an affine transformation to the feature of an input frame in order to deform it into another frame. It is composed of a localization network to estimate the affine transformation parameters, a grid generator, and a sampler to apply this transformation to the input features. But a single affine transformation can be too rigid. For instance, when digits in MMNIST are transitioning along two independent trajectories, a single affine transformation is insufficient. In this case, we turn to blockSTN, where the feature maps are divided into blocks, and each block applies its affine transformation. BlockSTN equivalently performs local rigid transformations, striking a balance between rigidity and flexibility. The transformation module takes the latent features of two frames and learns a stack of block-wise transformation parameters, which optimally warp one to the other, i.e., $\mathcal{G}_{ij} = \mathcal{T}(s_i, s_j)$.

7.4.3 Network Architecture

We describe our network architecture in the following.

The transformation module consists of an image encoder, decoder, and transformer $\mathcal{T}$. The encoder, in this case, uses a stack of 4 basic residual blocks (He et al., 2016). We split the end feature channels into blocks multiple blocks

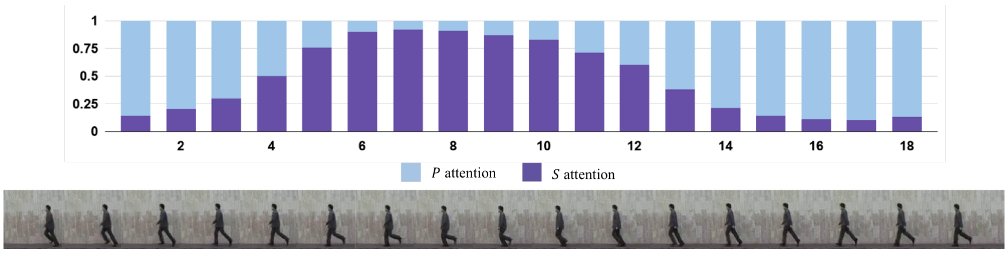


Figure 7.5: Visualization of how the final linear classifier attends to features from S and P for the Weizmann human action sequences.

according to the dataset complexity. Each block is processed following the original STN. The decoder is symmetric to the encoder, with the additional transposed convolution layers for upsampling.

The video encoder Enc_v follows a Res3D-18 backbone (Hara et al., 2018). After the first 3 residual blocks, Enc_v branches out to a convolution layer that outputs S and a fourth residual block followed by fully-connected (FC) layers to output P.

The frame-level encoder shares the same architecture as the transformation module's encoder for the initial convolution layers. Then, where the transformer originally outputs s, we condition S via concatenation, followed by more convolution and finally fully connected layers to output p. The frame-level decoder reverses the encoder operations.

The autoregressive model, where P conditioned on p_1, predicts the rest of the $p_{t>1}$, is a one-layer LSTM (Hochreiter and Schmidhuber, 1997) as in Figure 7.2(b).

7.4.4 A Toy Example: the MMNIST dataset

As a synthetic dataset, the video statics and dynamics of MMNIST are well defined and thus provide an intuitive understanding of the working of the proposed framework. For MMNIST, we use 2 blocks for BlockSTN.

Visualization of S. In Figure 7.3, we project S back to the pixel level through g_{dec}. Not only does S capture the temporal coherence, namely the two digits, but it also positions them in a way to be easily warped to each frame.

Unsupervised 2-digit classification. We examine the quality of the representation S via unsupervised 2-digit classification (55 classes) by adding a simple linear classifier directly on S. It achieves a test accuracy of 69.1%. As a baseline comparison, we perform the same experiment with s_t's (see equation 7.2) and

Dataset	Classification	Recon.	Ours
MMNIST	61	**24**	**24**
Weizmann	1.87	0.96	**0.92**
RADVESS	3.03	3.07	**2.75**

Table 7.1: Average keypoint displacement distance between reference video frames and retrieved video frames. Compare retrieving based on video features trained from supervised classification, unsupervised reconstruction and our proposed unsupervised framework. Retrieved videos from our proposed video dynamics representations more faithfully resemble the reference video dynamics and thus achieves the smallest keypoint displacement distances across all datasets.

aggregate votes from each frame to reach a final classification decision. The baseline network gives a test accuracy of 63.0%, which is significantly lower than using the over-all video statics S.

Dynamics-relevant retrieval. We conduct a retrieval task using P, by taking a reference sequence from the test set and searching for the nearest neighbors in the feature space of P. The results are shown in Figure 7.4(b) exhibiting motion trajectories highly resembling their corresponding references, whereas the digit contents can significantly differ. More detailed results on the dynamics-relevant retrieval are in Section 7.5.2.

Swapping video statics and dynamics. We also notice a curious tendency that even though our method is not explicitly optimized with the content-motion disentanglement objective (Li and Mandt, 2018; Hsieh et al., 2018; Denton, 2017), it still manages to distinguish them to a certain degree. Notably, in Figure 7.4(a), we pair S from the sequence in row (i) with p's from the sequence in row (ii) to compose a new sequence with digit appearances from the former and trajectory from the latter.

7.5 EXPERIMENTAL RESULTS

This section introduces the datasets in our experiments and then evaluates the proposed framework through several empirical tasks. Our primary task is to retrieve videos using the learned dynamics representations. The retrieved videos bear assembling dynamical content to the reference videos—a property that is infeasible to manually label but practically very useful—, albeit independent appearances. In this application, we follow the "frozen net" convention (Kolesnikov et al., 2019) and directly apply representations extracted using our unsupervised model. Additionally, we further analyze the statics and dynamics representations through classification tasks. First, we classify actions in videos under the linear protocol while focusing on how video statics and dynamics contribute to the classification decisions. Next, we employ our unsupervised phase as a pretraining step for expression recognition.

7.5.1 Datasets

To ensure generalization of the proposed framework, we evaluate on a diversity of video domains: the Weizmann human action dataset ("Actions as space-time shapes"), and RAVDESS (video only) facial expression dataset (Livingstone and Russo, 2018). Video length is kept at 8 frames for all datasets.

The Weizmann human action dataset, a widely-used benchmark, contains 93 sequences of 9 actors performing 10 actions. We crop and normalize each frame from resolution 180×144 to 96×96 as done in (Tulyakov et al., 2018). Data augmentation includes random horizontal flipping. We use 4 blocks for BlockSTN on Weizmann.

The RAVDESS dataset contains 2452 audio-visual sequences of 24 actors singing and speaking with 8 facial expressions. Our experiments use the video channel only. We preprocess faces using the normalization procedure from (Karras et al., 2019) after detecting facial landmarks (Zhang et al., 2016b), where we apply a similarity transform to remove the effect of in-plane rotation of faces and scale them to 128×128. Data augmentation includes random horizontal flipping. Uniform temporal downsampling by a factor of 3 is performed, i.e., taking every 3rd frame to form sequences. We use 8 blocks for BlockSTN on RAVDESS.

7.5.2 Video Retrieval based on Dynamics

A unique direct application of our learned representations is to retrieve relevant videos based on the video dynamics representations P, since manual annotation of video dynamics is infeasible. For example, given a facial expression sequence, if one desires to discover other sequences sharing the same facial movements while ignoring the appearances, we can extract P from the reference sequence and search for its nearest neighbors in the representation space. Another important downstream application is learning through demonstration without action labels: if we can recover the actions from a demonstration video by identifying a video sequence with similar motion trajectories whose actions are known, the rich amount of unlabeled demonstrations can be easily leveraged.

In this section, we experiment on 3 datasets, MMNIST, Weizmann, and RADVESS. For MMNIST, we hold out the validation set from training; for Weizmann and RADVESS, we hold out one identity from training. Then we take sequences from the holdout set as reference (top of each row) and retrieve videos from the rest of the database based on the high-level features. We compare and extract features from 3 types of models, namely, a supervised model trained on classification tasks, an unsupervised model trained on reconstruction loss—i.e., a standard autoencoder, and our proposed model, which is also unsupervised. The classification task for MMNIST is digit classification, Weizmann human activity recognition, and RADVESS facial expression recognition. To extract features after training, concretely, we take the last FC layer outputs before softmax from the classification models, the middle bottleneck layer outputs of unsupervised reconstructive autoencoders, and the video dynamics representations P of our proposed models. The model architecture is consistent across all models, and the high-level features are all of the same dimensions across models, which are 32 for MMNIST and 512 for the other datasets.

Quantitative evaluation of dynamics based retrieval is challenging since it is infeasible to explicitly "label" the dynamics of a video clip. Therefore, we approximate the retrieval results through *average keypoint displacement distance*. For each dataset, we define detectable key points that are most relevant to the motion and dynamics of the video. For MMNIST, we directly use the center coordinate of each digit from the simulator; for Weizmann, we apply a pre-trained human-pose estimator, OpenPose (Cao et al., 2019), to detect keypoints; for RADVESS, we use the same set of facial landmarks (Zhang et al., 2016b) as used in Section 7.5.1. Then we calculate the *displacements* of keypoints between frames instead of directly using their absolute locations for both reference and retrieved video sequences. Finally, Table 7.1 reports the L^1 norm of the key-point displacement distance between reference and retrieved videos averaged overall key points. The video dynamics representations from our proposed model achieve the smallest differences. Although the representations from the reconstruction models appear to perform well for MNIST and Weizmann, we qualitatively show that the retrieved videos bear a lot more static visual resemblance to the reference videos, i.e., the retrieval is based on both statics and dynamics instead of dynamics alone in the next paragraphs.

Figure 7.6 shows examples of reference (top of each pair) vs. retrieved (bottom) pairs. The videos retrieved with classification features in Figure 7.6a share high-level content as the reference videos, such as the digit classes, the human activities, and the facial expressions but differ significantly in detailed motion trajectories. On the one hand, when using reconstructive features, the MMNIST and Weizmann retrievals do cover similar motion content to the reference as these trajectories significantly contribute to reconstruction. This also explains the competitive keypoint displacement distance from reconstruction models in Table 7.1. On the other hand, for RADVESS, the retrieved sequences share much fewer motion dynamics with the reference. They mostly come from the same identity, which is highly resembling the held-out identity in appearance. Even in the case of MMNIST and Weizmann, the retrieved videos also share significant appearance cues with the referenced sequences. In contrast to both the classification model and the reconstruction model, retrieved sequences based on our proposed P share highly similar motion trajectories but distinct appearances. Take the last row of Weizmann from Figure 7.6c as an example: not only does the positioning of each actor agree between reference and retrieved, but also more subtle body movements such as the rhythm of the stride.

Therefore, using our representation learning framework, we can envision a set of practical applications that automatically identify sequences from a database carrying analogous video dynamics independent of statics appearance.

7.5.3 Additional Empirical Analysis

As a bonus, we further analyze the statics and dynamics representations through classification tasks. In Section 7.5.3, under the linear protocol on top of the learned video statics and dynamics representations, we perform human activity classification on Weizmann while focusing on how each component contributes to the classification decisions. Next, in Section 7.5.3, we employ our

Weizmann	Avg. accuracy
(**bregonzio2012fusing**)	96%
(**sharif2017framework**)	95%
supervised 3DCNN	75%
ours, $S+P$	**96**%

Table 7.2: Compare average action recognition accuracy using leave-one-actor-out cross validation on the Weizmann dataset.

unsupervised phase as a pretraining step for expression recognition to highlight the proposed framework arrives at a suite of meaningful weights for potentially higher level application.

Interplay between Statics and Dynamics

With the Weizmann dataset, we are interested in investigating how our video statics and dynamics contribute to the final action recognition task. Such analysis not only helps us provide insight about the extracted representations, but also sheds light towards the potential direction of more interpretable data-driven decision making.

Concretely, we train a linear classifier directly on top of S and P. Instead of naive concatenation and weighing the two equally, we learn an attention scalar $0 \leq \mathrm{a} \leq 1$ to balance their contributions. In other words, we apply linear transformations on both S and P to obtain feature vectors $g_S, g_P \in \mathbb{R}^{c_g}$ and then use another linear layer f to derive the attention scalar, $\mathrm{a}=\sigma(\mathrm{f}(g_S, g_P))$. The attention scalar re-distributes the two feature vectors to the concatenation $(\mathrm{a} \cdot g_S, (1-\mathrm{a}) \cdot g_D)$ as input for a final FC layer to make a classification decision. We average the logits over all sequences within a video during training and pool a single score to backpropagate through. We adopt leave-one-actor-out cross-validation (Kong and Fu, 2018).

Table 7.2 reports our result in comparison to a fully supervised 3DCNN baseline that has the same architecture as Enc_v. We also list other previously proposed systems that follow the same evaluation protocol. Since the Weizmann dataset has been thoroughly studied, although the frozen features with an attentive linear classifier achieve good results, our focus is the interplay between S and P in the decision making process. Figure 7.7 shows how the classifier attention distributes between S and P. Roughly half of the time, the classifier attends to S and the other half to P. However, there is a significant

RADVESS	Avg. accuracy
supervised 3DCNN	67%
pretrain with S only	68%
ours, $S+P$	**72%**

Table 7.3: We compare recognition accuracy on RADVESS.

shift from dynamics to statics then back to dynamics. Indeed, when the actor moves from the edge of the frame, the classifier attends more to P. When the actor is around the center of the frame, it turns to S.

We also attempted to assign all attention to S or P. But both fail, since averaging logits cause the false leads from either S or P to easily overshadow the right signals. This observation confirms that at different stages of the video, S and P contribute differently towards action recognition.

Pretraining for Expression Recognition

Another popular evaluation of unsupervised representation learning is to use the pretrained weights from the unsupervised phase as initialization for a downstream supervised task (Zhai and al., 2019). We follow the same procedure to assess our framework on RADVESS for expression recognition. We use 5-splits cross-validation for RADVESS We keep a vanilla evaluation protocol: all sequences within a video vote together to reach a final verdict. We strive to keep consistency across different methods for any comparative experiments, namely model architecture, preprocessing, data augmentation, and evaluation protocol.

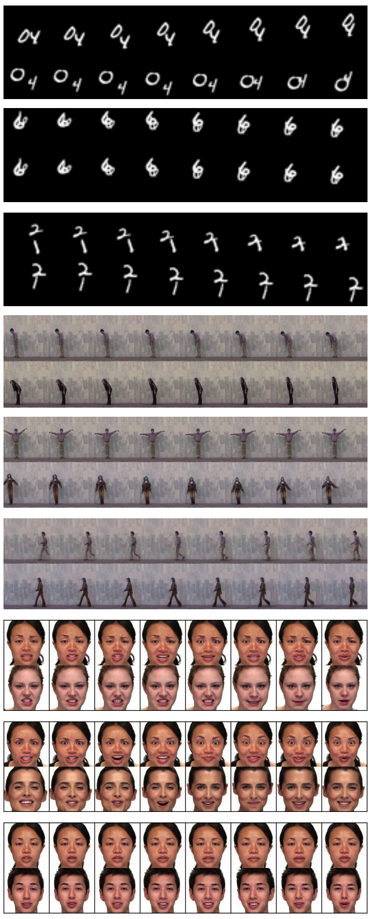
Specifically, we first conduct unsupervised pretraining as described in Section 7.4. Taking Enc_v, we replace the FC layer that originally outputs P with an average pooling layer followed by another linear classifier.

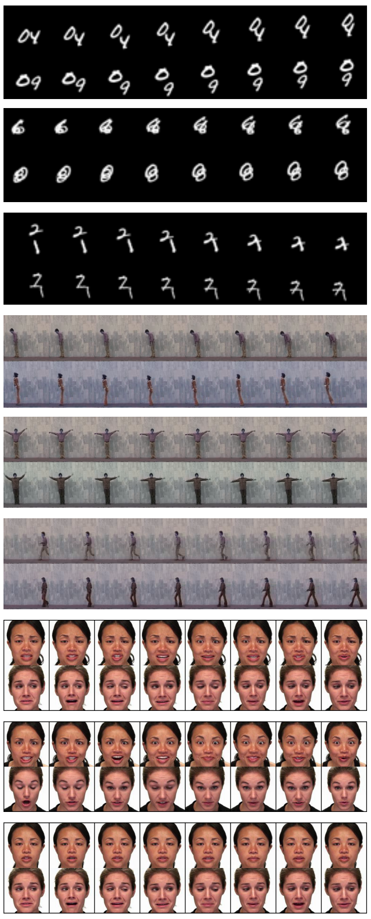
As a baseline, we train the same 3DCNN from scratch with random initialization. For both RADVESS, our unsupervised pretraining improves upon the random initialization, from 67% to 72% for RADVESS.

As another ablation, we perform unsupervised pretraining using the S objective in equation 7.3 only. It yields results comparable to the random baseline, 68% vs. 67% for RADVESS, demonstrating that joint learning from video statics and dynamics leads to better representations.

7.6 CONCLUSION

We introduce a novel unsupervised framework to explicitly learn decomposed video representations from video temporal coherence and dynamics. The learned representations are directly useful for interesting applications, particularly dynamics relevant retrieval. Our framework can also help other downstream tasks video-level classification tasks such as action and facial expression recognition. An immediate add-on in future works is spatial and/or temporal attention. For example, use representations from video statics to identify "moving" regions to attend to when distilling frame-wise dynamics. It is also worth integrating other existing unsupervised techniques in our framework. Lastly, by building an additional hierarchy on top of video-level representations, we can perform video understanding over much longer sequences.


(a) classification, supervised


(b) reconstruction, unsupervised

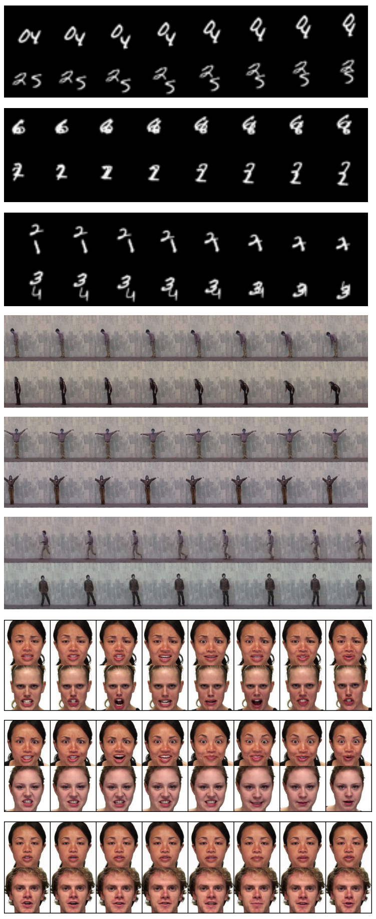

(c) proposed, unsupervised

Figure 7.6: Visualization of retrievals based on (a) the last FC layer before softmax of supervised classification models (b) the middle bottleneck layer of unsupervised reconstructive autoencoders and (c) the proposed video dynamics representations. In each pair, the reference is at the top and retrieved the bottom. Our video dynamics encode motion very well on all datasets independent of static appearance. Meanwhile, the classification features tend to ignore motion details; the reconstruction features tend to select retrievals assembling in appearance besides motion.

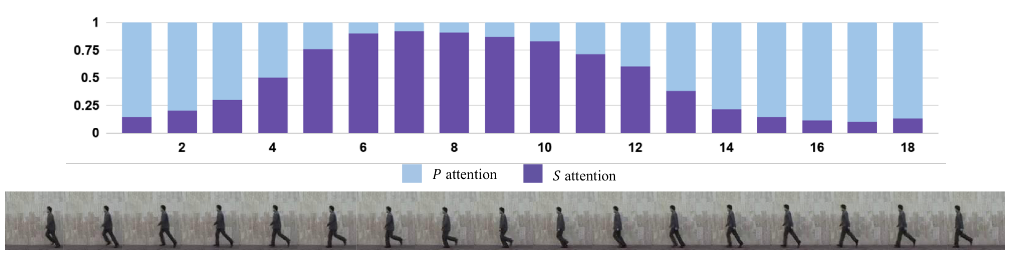


Figure 7.7: Visualization of how the final linear classifier attends to features from S and P for the Weizmann human action sequences.

CHAPTER APPENDIX

In the supplementary materials, we provide full implementation details, including model architecture and training details.

7.A MODEL ARCHITECTURE

All the experiments w.r.t different datasets share similar model layouts. More details are discussed in the following.

7.A.1 Transformer

The transformer module consists of three components: encoding, decoding, and transformation $\mathcal{T}$, BlockSTN. The encoders here use a stack of four basic residual blocks. The last residual output features of $64 \times 8 \times 8$ for MMNIST, $64 \times 12 \times 12$ for Weizmann and $128 \times 16 \times 16$ for RADVESS. Then we split the channels into blocks, MMNIST 2 blocks, Weizmann 4 blocks, and RADVESS 8 blocks. Each block is processed following the original STN (Jaderberg et al., 2015). Specifically, each block is connected to a localization network to output the 6 affine-transformation parameters, a gird generator to generate transformation grids from the transformation parameters, and a final step that transfers each block using the transformation grids. The transformed features then go through the decoder, which is symmetric to the encoder, with the additional transposed convolution layers for upsampling.

7.A.2 Video Encoder

The video encoder follows a 3D-Res18 backbone (Hara et al., 2018). After the first three 3D residual layers, the network branches out to output S and P. The former is done via merging features along time dimension then applying 2D convolutional layers. The latter first goes through more 3D layers, followed by fully connected layers. The final dimensionality of P is 512 for all datasets except MNIST's 32.

7.A.3 Frame Encoder and Decoder

The frame encoder and decoder copy the same architecture as the encoder and decoder from the transformation module up to the transformation step. Then instead of performing the transformation, we concatenate the feature with S, which is followed by additional convolutional layers and, finally, an FC layer outputting p. The dimensionality of p is 128 for all datasets except MNIST's 16.

7.A.4 Autoregressive Model

To predict p_t's, for $t = 2, \cdots T$, we use a one-layer LSTM with 1024 hidden units. At the first step, the input is a combination of p_1 and P. For the remaining time steps, the input is $\hat{p}_{t-1}$ and P, where $\hat{p}_{t-1}$ is the predicted previous time step.

7.B IMPLEMENTATION DETAILS

We set our sequence length to be 8 and batch size 64 for all datasets. We also use Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1e{-}08$ for all experiments.

7.B.1 MMNIST

Moving MNIST (Srivastava et al., 2015) (MMNIST), used by many previous works in video representation learning (Hsieh et al., 2018; Srivastava et al., 2015; Denton, 2017), is consist of two randomly sampled digits following independently generated random trajectories in 64×64-sized frames. We train with digits sampled from the MNIST training set, validate with digits from the validation set and fix a set of $10,000$ sequences of 8 frames with digits from the test set for testing.

To train the BlockSTN transformer, we warp between different random frames within 25 frames of each other. We set the initial learning rate 0.005, decayed by half every 50 epochs, and run at most 250 epochs. The reconstruction objective coefficient is 0.1, and the spatial transformation coefficient is 1.0. To train the rest of the proposed model, we set the initial learning rate at 0.001, decayed by half every 50 epochs, and run at most 150 epochs. The coefficient for the contrastive objective is 0.25, and the frame-level reconstruction is 1.0.

7.B.2 Weizmann

For preprocessing, we center and resize each frame to 96×96 as done in (Tulyakov et al., 2018), then map the pixel values to between -1 and 1. For data augmentation, we perform random horizontal flip over sequences.

To train the BlockSTN transformation module, we warp between different random frames within 25 frames of each other. We set the initial learning rate at 0.0001, decayed by half every 50 epochs, and run at most 250 epochs. To train the rest of the proposed model, we set the initial learning rate at 0.0005, decayed by half every 50 epochs, and run at most 150 epochs. The coefficient for the contrastive objective is 0.01. For this dataset, there are frames where humans do not appear or fully appear in the sequence, causing the learning of S difficult. For this reason, we add an additional regularization term: first use transformation error to identify which frame can be best warped to the other frames and then use a perceptual loss to encourage S close to that frame's s. This term has a coefficient 0.5.

The same "no-human" issue also affects action recognition. In this case, we use simple median-based background subtraction to automatically cut frames at the beginning or the end if there are no humans in them.

7.B.3 RADVESS

For preprocessing, we crop based on facial landmarks and resize each frame to 96×96 as done in (Karras et al., 2019), then map the pixel values to between -1 and 1. For data augmentation, we perform random horizontal flip over sequences.

To train the BlockSTN transformation module, we warp between different random frames within 25 frames of each other but at least 5 frames apart. We set the initial learning rate at 0.001, decayed by half every 50 epochs, and run at most 500 epochs. To train the rest of the proposed model, we set the initial learning rate at 0.0005, decayed by half every 50 epochs, and run at most 500 epochs. The coefficient for the contrastive objective is 0.1. To train the classifier, for both random initialization and initialization using the proposed model, we use an initial learning rate of 0.001, decayed by half every 50 epochs, and run at most 200 epochs.

8 CONCLUSION

Deep learning has become one of the most prominent ways to tackle problems in modern artificial intelligence. The tools in deep learning are expressive, abundant, and flexible. At the same time, given a specific application, it is becoming increasingly non-trivial to pinpoint the optimal framework in terms of e.g. model design and training. This thesis showcases how to effectively craft deep learning frameworks. In particular, we incorporate the essential inductive biases from the target tasks and the type of available data to high-level abstract representations. We demonstrate over a wide range of important example problems from reinforcement learning and computer vision applications.

For reinforcement learning, exploration and uncertainty are among its most unique challenges. First, we demonstrate a generic technique, stochastic activation, to improve exploration and reflect uncertainty on top of actor-critic methods. Then, we delve further into a special case of RL setup—a persistent environment coupling with a diverse suite of potential tasks —and provide a solution of decomposing the environment into a graph representation as a building block for downstream hierarchical RL. Lastly, we look at another important class of problem, multi-agent RL, and devise an attentive framework to coordinate exploration and execution among multiple agents.

For computer vision, it has been a long-standing goal to utilize the wealth of unlabeled digital data better. We are thus inspired to study unsupervised representation learning of images and videos. For images, we construct the latent space from a coarse to fine progression, from which we can generate realistic-looking, complex-structured images. A conditional version that further pays attention to different attributes during different stages of the progression is also deployed to a face synthesis application. For videos, we take advantage of the interplay of temporal coherence and dynamics to extract video-level statics and dynamics representations.

The field of deep learning and AI is ever-evolving. Therefore the possibilities in forming a deep learning framework seem to grow more overwhelming over time. Especially given limited resources, it has become more valuable to be able to streamline the design of deep learning frameworks. In this thesis, we show that, when tackling an ML problem, it is important to understand the nature of the problem and assemble the most suitable tools wisely together. We hope our works inspire future models and training designs for future research and applications.

BIBLIOGRAPHY

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. (2016). "Tensorflow: A system for large-scale machine learning." In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283.

Pieter Abbeel and Andrew Y Ng (2004). "Apprenticeship learning via inverse reinforcement learning." In: *Proceedings of the twenty-first international conference on Machine learning*. ACM, p. 1.

Joshua Achiam and Shankar Sastry (2017). "Surprise-based intrinsic motivation for deep reinforcement learning." In: *arXiv preprint arXiv:1703.01732*.

Igor Adamski, Robert Adamski, Tomasz Grel, Adam Jedrych, Kamil Kaczmarek, and Henryk Michalewski (2016). "Distributed Deep Reinforcement Learning: learn how to play Atari games in 21 minutes." In: *arXiv*.

Pulkit Agrawal, Joao Carreira, and Jitendra Malik (2015). "Learning to see by moving." In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 37–45.

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy (2016). "Deep variational information bottleneck." In: *arXiv preprint arXiv:1612.00410*.

Adrien Angeli, David Filliat, Stéphane Doncieux, and Jean-Arcady Meyer (2008). "A fast and incremental method for loop-closure detection using bags of visual words." In: *IEEE Transactions on Robotics*, pp. 1027–1037.

Martin Arjovsky, Soumith Chintala, and Léon Bottou (2017). "Wasserstein gan." In: *arXiv*.

Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang (2017). "Generalization and Equilibrium in Generative Adversarial Nets (GANs)." In: *ICML*.

Arthur Aubret, Laetitia Matignon, and Salima Hassas (2019). "A survey on intrinsic motivation in reinforcement learning." In: *arXiv preprint arXiv:1908.06976*.

Mohammad Gheshlaghi Azar, Biala Piot, Bernardo Avila Pires, Jean-Bastien Gril, Florent Altche, and Remi Munos (2019). "World discovery model." In: *arXiv*.

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

Pierre-Luc Bacon, Jean Harb, and Doina Precup (2017). "The option-critic architecture." In: *AAAI Conference on Artificial Intelligence*.

Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua (2017). "CVAE-GAN: fine-grained image generation through asymmetric training." In:

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver (2017). "Successor features for transfer in reinforcement learning." In: *Advances in neural information processing systems*, pp. 4055–4065.

Gabriel Barth-Maron, Matthew Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva TB, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap (2018). "Distributional Policy Gradients." In: *ICLR*.

Joost Bastings, Wilker Aziz, and Ivan Titov (2019). "Interpretable Neural Predictions with Differentiable Binary Variables." In: *Proceedings of the 2019 Conference of the Association for Computational Linguistics, Volume 1 (Long Papers)*. Florence, Italy: Association for Computational Linguistics.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. (2018). "Relational inductive biases, deep learning, and graph networks." In: *arXiv preprint arXiv:1806.01261*.

Marc Bellemare, Joel Veness, and Michael Bowling (2012). "Sketch-based linear value function approximation." In: *Advances in Neural Information Processing Systems*, pp. 2213–2221.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos (2016). "Unifying count-based exploration and intrinsic motivation." In: *NIPS*.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling (2013). "The arcade learning environment: An evaluation platform for general agents." In: *Journal of Artificial Intelligence Research*.

Marc G Bellemare, Will Dabney, and Rémi Munos (2017). "A distributional perspective on reinforcement learning." In: *ICML*.

Thomas Berg, Jiongxin Liu, Seung Woo Lee, Michelle L Alexander, David W Jacobs, and Peter N Belhumeur (2014). "Birdsnap: Large-scale fine-grained visual categorization of birds." In: *CVPR*.

Dimitri P Bertsekas (1999). *Nonlinear Programming*.

Norman Biggs (1993). *Algebraic Graph Theory*.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra (2015). "Weight uncertainty in neural networks." In: *ICML*.

Léon Bottou (1991). "Stochastic gradient learning in neural networks." In:

Olivier Bousquet, Sylvain Gelly, Ilya Tolstikhin, Carl-Johann Simon-Gabriel, and Bernhard Schoelkopf (2017). "From optimal transport to generative modeling: the VEGAN cookbook." In: *arXiv*.

Craig Boutilier (1996). "Planning, learning and coordination in multiagent decision processes." In: *Proceedings of the 6th conference on Theoretical aspects of rationality and knowledge*. Morgan Kaufmann Publishers Inc., pp. 195–210.

Andrew Brock, Jeff Donahue, and Karen Simonyan (2018). "Large scale gan training for high fidelity natural image synthesis." In: *arXiv preprint arXiv:1809.11096*.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba (2016). *OpenAI Gym*. eprint: arXiv.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov (2015). "Importance weighted autoencoders." In: *ICLR*.

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros (2018). "Large-scale study of curiosity-driven learning." In: *arXiv preprint arXiv:1808.04355*.

Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh (2019). "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Joao Carreira and Andrew Zisserman (2017). “Quo vadis, action recognition? a new model and the kinetics dataset.” In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6299–6308.

Junbum Cha (2017). *Implementations of (theoretical) generative adversarial networks and comparison without cherry-picking*. `https://github.com/khanrc/tf.gans-comparison`.

Maria Chatzigiorgaki and Athanassios N Skodras (2009). “Real-time keyframe extraction towards video content identification.” In: *2009 16th International conference on digital signal processing*. IEEE, pp. 1–6.

Fei Chen, Yang Gao, Shifu Chen, and Zhenduo Ma (2007). “Connect-based subgoal discovery for options in hierarchical reinforcement learning.” In: *Third International Conference on Natural Computation (ICNC 2007)*. Vol. 4. IEEE, pp. 698–702.

Jianmin Chen, Rajat Monga, Samy Bengio, and Rafal Jozefowicz (2016a). “Revisiting distributed synchronous SGD.” In: *arXiv*.

Jiawei Chen, Janusz Konrad, and Prakash Ishwar (2020a). “A cyclically-trained adversarial network for invariant representation learning.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 782–783.

Jie Chen, Ut-Va Koc, and KJ Ray Liu (2001). *Design of digital video coding systems: a complete compressed domain approach*. CRC Press.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton (2020b). “A simple framework for contrastive learning of visual representations.” In: *arXiv preprint arXiv:2002.05709*.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel (2016b). “Infogan: Interpretable representation learning by information maximizing generative adversarial nets.” In: *NIPS*.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel (2016c). “Variational Lossy Autoencoder.” In: *arXiv*.

Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo (2018). “StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation.” In: *CVPR*.

Sumit Chopra, Raia Hadsell, and Yann LeCun (2005). "Learning a similarity metric discriminatively, with application to face verification." In: *Conference on Computer Vision and Pattern Recognition (CVPR).*

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio (2015). "A recurrent latent variable model for sequential data." In: *Advances in neural information processing systems*, pp. 2980–2988.

John D Co-Reyes, YuXuan Liu, Abhishek Gupta, Benjamin Eysenbach, Pieter Abbeel, and Sergey Levine (2018). "Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings." In: *arXiv preprint arXiv:1806.02813*.

Christian Daniel, Herke Van Hoof, Jan Peters, and Gerhard Neumann (2016). "Probabilistic inference for determining options in reinforcement learning." In: *Machine Learning* 104.2-3, pp. 337–357.

Peter Dayan and Geoffrey E Hinton (1993). "Feudal reinforcement learning." In: *Advances in neural information processing systems*, pp. 271–278.

DeepMind (2020). *DeepMind Lab 2D*.

Emily L Denton et al. (2017). "Unsupervised learning of disentangled representations from video." In: *Advances in neural information processing systems*, pp. 4414–4423.

Coline Devin, Pieter Abbeel, Trevor Darrell, and Sergey Levine (2018). "Deep object-centric representations for generalizable robot learning." In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 7111–7118.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding." In: *arXiv*.

Ali Diba, Vivek Sharma, Luc Van Gool, and Rainer Stiefelhagen (2019). "Generating videos with scene dynamics." In: *ICCV*.

Bruce L Digney (1998). "Learning hierarchical control structures for multiple tasks and changing environments." In:

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio (2017). "Density estimation using Real NVP." In: *ICLR*.

Jeff Donahue and Karen Simonyan (2019). "Large scale adversarial representation learning." In: *Advances in Neural Information Processing Systems*, pp. 10542–10552.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell (2013). "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition." In: *arXiv preprint arXiv:1310.1531*.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell (2014). "Decaf: A deep convolutional activation feature for generic visual recognition." In: *International conference on machine learning*, pp. 647–655.

John Duchi (2007). "Derivations for linear algebra and optimization." In: *Berkeley, California*.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune (2019). "Go-Explore: a New Approach for Hard-Exploration Problems." In: *arXiv preprint arXiv:1901.10995*.

Negin Entezari, Mohammad Ebrahim Shiri, and Parham Moradi (2010). "A local graph clustering algorithm for discovering subgoals in reinforcement learning." In: *International Conference on Future Generation Communication and Networking*. Springer, pp. 41–50.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. (2018). "IMPALA: Scalable distributed Deep-RL with importance weighted actor-learner architectures." In: *International Conference on Machine Learning (ICML)*.

Lasse Espeholt, Raphaël Marinier, Piotr Stanczyk, Ke Wang, and Marcin Michalski (2019). "SEED RL: Scalable and Efficient Deep-RL with Accelerated Central Inference." In: *arXiv preprint arXiv:1910.06591*.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine (2019). "Search on the Replay Buffer: Bridging Planning and Reinforcement Learning." In: *arXiv preprint arXiv:1903.00606*.

FACES 4.0 (2016). `http://www.iqbiometrix.com`.

FBI (2017). *Preliminary Semiannual Uniform Crime Report, January–June, 2017*.

Zhengzhu Feng, Richard Dearden, Nicolas Meuleau, and Richard Washington (2004). "Dynamic programming for structured continuous Markov decision problems." In: *Proceedings of the 20th conference on Uncertainty in artificial intelligence*. AUAI Press, pp. 154–161.

Basura Fernando, Hakan Bilen, Efstratios Gavves, and Stephen Gould (2017). "Self-supervised video representation learning with odd-one-out networks." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3636–3645.

Carlos Florensa, Yan Duan, and Pieter Abbeel (2017). "Stochastic neural networks for hierarchical reinforcement learning." In: *International Conference on Learning Representations (ICLR)*.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. (2017). "Noisy networks for exploration." In: *arXiv*.

Dieter Fox, Wolfram Burgard, and Sebastian Thrun (1998). "Active markov localization for mobile robots." In: *Robotics and Autonomous Systems* 25.3-4, pp. 195–207.

Charlie D Frowd, Derek Carson, Hayley Ness, Dawn McQuiston-Surrett, Jan Richardson, Hayden Baldwin, and Peter Hancock (2005). "Contemporary composite techniques: The impact of a forensically-relevant target delay." In: *Legal and Criminological Psychology* 10.1, pp. 63–81.

Yarin Gal and Zoubin Ghahramani (2016). "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning." In: *ICML*.

Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen (2016). "Improving PILCO with Bayesian neural network dynamics models." In: *ICML Workshop*.

Hongyang Gao, Hao Yuan, Zhengyang Wang, and Shuiwang Ji (2017). "Pixel deconvolutional networks." In: *arXiv preprint arXiv:1705.06820*.

Jon Gauthier (2014). *Conditional generative adversarial nets for convolutional face generation*.

Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, Aviv Tamar, et al. (2015). "Bayesian reinforcement learning: A survey." In: *Foundations and Trends® in Machine Learning*.

Dibya Ghosh, Abhishek Gupta, and Sergey Levine (2018). "Learning Actionable Representations with Goal-Conditioned Policies." In: *arXiv preprint arXiv:1811.07819*.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik (2014). "Rich feature hierarchies for accurate object detection and semantic segmentation." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative Adversarial Nets." In: *Advances in Neural Information Processing Systems*.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville (2016). *Deep learning*. MIT press.

Anirudh Goyal, Riashat Islam, Daniel Strouse, Zafarali Ahmed, Matthew Botvinick, Hugo Larochelle, Sergey Levine, and Yoshua Bengio (2019). "Infobot: Transfer and exploration via the information bottleneck." In: *arXiv preprint arXiv:1901.10902*.

Karol Gregor and Frederic Besse (2018). "Temporal difference variational auto-encoder." In: *arXiv preprint arXiv:1806.03107*.

Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra (2014). "Deep AutoRegressive Networks." In: *ICML*.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra (2015). "DRAW: A recurrent neural network for image generation." In: *ICML*.

Karol Gregor, Frederic Besse, Danilo Jimenez Rezende, Ivo Danihelka, and Daan Wierstra (2016). "Towards conceptual compression." In: *NIPS*.

Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine (2017). "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates." In: *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, pp. 3389–3396.

Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vazquez, and Aaron Courville (2016). "PixelVAE: A Latent Variable Model for Natural Images." In: *arXiv*.

Qi Guo, Ce Zhu, Zhiqiang Xia, Zhengtao Wang, and Yipeng Liu (2017). "Attribute-controlled face photo synthesis from simple line drawing." In: *arXiv*.

Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Bernardo A Pires, Toby Pohlen, and Rémi Munos (2018). "Neural Predictive Belief Representations." In: *arXiv preprint arXiv:1811.06407*.

Michael Gutmann and Aapo Hyvärinen (2010). "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models." In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

David Ha and Jürgen Schmidhuber (2018). "World models." In: *arXiv preprint arXiv:1803.10122*.

Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine (2018a). "Latent space policies for hierarchical reinforcement learning." In: *arXiv preprint arXiv:1804.02808*.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine (2018b). "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor." In: *International Conference on Machine Learning (ICML)*.

Raia Hadsell, Sumit Chopra, and Yann LeCun (2006). "Dimensionality reduction by learning an invariant mapping." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson (2018). "Learning latent dynamics for planning from pixels." In: *arXiv preprint arXiv:1811.04551*.

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

Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh (2018). "Can Spatiotemporal 3D CNNs Retrace the History of 2D CNNs and ImageNet?" In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6546–6555.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep residual learning for image recognition." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick (2017). "Mask r-cnn." In: *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick (2019). "Momentum contrast for unsupervised visual representation learning." In: *arXiv preprint arXiv:1911.05722*.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick (2020). "Momentum contrast for unsupervised visual representation learning." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger (2018). "Deep reinforcement learning that matters." In: *Thirty-Second AAAI Conference on Artificial Intelligence*.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner (2017a). "beta-VAE: Learning basic visual concepts with a constrained variational framework." In: *ICLR)*.

Irina Higgins, Arka Pal, Andrei A Rusu, Loic Matthey, Christopher P Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner (2017b). "DARLA: Improving zero-shot transfer in reinforcement learning." In: *ICML*.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Adam Trischler, and Yoshua Bengio (2018). "Learning deep representations by mutual information estimation and maximization." In: *arXiv preprint arXiv:1808.06670*.

Sepp Hochreiter and Jürgen Schmidhuber (1997). "Long short-term memory." In: *Neural Computation* 9.8.

Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li F Fei-Fei, and Juan Carlos Niebles (2018). "Learning to decompose and disentangle representations for video prediction." In: *Advances in Neural Information Processing Systems*, pp. 517–526.

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne (2017). "Imitation learning: A survey of learning methods." In: *ACM Computing Surveys (CSUR)* 50.2, p. 21.

Sergey Ioffe and Christian Szegedy (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In: *arXiv preprint arXiv:1502.03167*.

Shariq Iqbal and Fei Sha (2018). "Actor-attention-critic for multi-agent reinforcement learning." In: *arXiv preprint arXiv:1810.02912*.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. (2015). "Spatial transformer networks." In: *Advances in neural information processing systems*, pp. 2017–2025.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu (2016). "Reinforcement learning with unsupervised auxiliary tasks." In: *arXiv preprint arXiv:1611.05397*.

Joel Janai, Fatma Guney, Anurag Ranjan, Michael Black, and Andreas Geiger (2018). "Unsupervised learning of multi-frame optical flow with occlusions." In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 690–706.

Dinesh Jayaraman and Kristen Grauman (2015). "Learning image representations tied to ego-motion." In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1413–1421.

Dinesh Jayaraman and Kristen Grauman (2016). "Slow and steady feature analysis: higher order temporal coherence in video." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3852–3861.

Dinesh Jayaraman, Frederik Ebert, Alexei A Efros, and Sergey Levine (2018). "Time-agnostic prediction: Predicting predictable video frames." In: *International Conference on LearningRepresentations*.

Wonseok Jeon, Paul Barde, Derek Nowrouzezahrai, and Joelle Pineau (2020). "Scalable Multi-Agent Inverse Reinforcement Learning via Actor-Attention-Critic." In: *arXiv preprint arXiv:2002.10525*.

Jiechuan Jiang and Zongqing Lu (2018). "Learning attentional communication for multi-agent cooperation." In: *Advances in neural information processing systems*, pp. 7254–7264.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei (2016). "Perceptual losses for real-time style transfer and super-resolution." In: *ECCV*.

Gregory Kahn, Adam Villaflor, Vitchry Pang, Pieter Abbeel, and Sergey Levine (2016). "Uncertainty-Aware Reinforcement Learning for Collision Avoidance." In: *arxiv*.

Daniel Kahneman and Anne Treisman (1984). *Changing views of Attention and Automaticity*. San Diego, CA: Academic Press, Inc.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. (2019). "Model-Based Reinforcement Learning for Atari." In: *arXiv preprint arXiv:1903.00374*.

Takuhiro Kaneko, Kaoru Hiramatsu, and Kunio Kashino (2017). "Generative Attribute Controller With Conditional Filtered Generative Adversarial Networks." In: *CVPR*.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen (2017). "Progressive Growing of GANs for Improved Quality, Stability and Variation." In: *arXiv*.

Tero Karras, Samuli Laine, and Timo Aila (2019). "A style-based generator architecture for generative adversarial networks." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410.

Ira Kemelmacher-Shlizerman, Supasorn Suwajanakorn, and Steven M Seitz (2014). "Illumination-aware age progression." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3334–3341.

Dahun Kim, Donghyeon Cho, and In So Kweon (2019). "Self-supervised video representation learning with space-time cubic puzzles." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33, pp. 8545–8552.

Diederik Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling (2016). "Improving Variational Inference with Inverse Autoregressive Flow." In: *NIPS*.

Diederik P Kingma and Jimmy Lei Ba (2014). "Adam: A method for stochastic optimization." In: *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma and Max Welling (2013). "Auto-encoding variational bayes." In: *arXiv preprint arXiv:1312.6114*.

Diederik P Kingma, Tim Salimans, and Max Welling (2015). "Variational dropout and the local reparameterization trick." In: *NIPS*.

Durk P Kingma and Prafulla Dhariwal (2018). “Glow: Generative flow with invertible 1x1 convolutions.” In: *Advances in neural information processing systems*, pp. 10215–10224.

Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling (2014). “Semi-supervised learning with deep generative models.” In: *Advances in neural information processing systems*, pp. 3581–3589.

Thomas Kipf, Yujia Li, Hanjun Dai, Vinicius Zambaldi, Edward Grefenstette, Pushmeet Kohli, and Peter Battaglia (2018). “Compositional Imitation Learning: Explaining and executing one task at a time.” In: *arXiv preprint arXiv:1812.01483*.

Scott Klum, Hu Han, Anil K Jain, and Brendan Klare (2013). “Sketch based face recognition: Forensic vs. composite sketches.” In: *International Conference on Biometrics (ICB)*.

Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer (2019). “Revisiting self-supervised visual representation learning.” In: *arXiv preprint arXiv:1901.09005*.

Vijay R Konda and John N Tsitsiklis (2000). “Actor-critic algorithms.” In: *Advances in neural information processing systems*, pp. 1008–1014.

Yu Kong and Yun Fu (2018). “Human action recognition and prediction: A survey.” In: *arXiv preprint arXiv:1806.11230*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton (2012). “Imagenet classification with deep convolutional neural networks.” In: *Advances in Neural Information Processing Systems*.

Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zając, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. (2019). “Google research football: A novel reinforcement learning environment.” In: *arXiv preprint arXiv:1907.11180*.

Malte Kuss and Carl E Rasmussen (2004). “Gaussian processes in reinforcement learning.” In: *NIPS*.

Gierad P Laput, Mira Dontcheva, Gregg Wilensky, Walter Chang, Aseem Agarwala, Jason Linder, and Eytan Adar (2013). “Pixeltone: A multimodal interface for image editing.” In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, pp. 2185–2194.

Hugo Larochelle (2015). *Binarized MNIST Dataset*. `http://www.cs.toronto.edu/~larocheh/public/datasets/binarized_mnist`.

Anders Boesen Lindbo Larsen and Søren Kaae Sønderby (2016). *Generating Faces with Torch.* http://torch.ch/blog/2015/11/13/gan.html.

Anders Boesen Lindbo Larsen, Soren Kaae Sonderby, Hugo Larochelle, and Ole Winther (2016). "Autoencoding beyond pixels using a learned similarity metric." In: *ICML*.

Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton (2015). "A simple way to initialize recurrent networks of rectified linear units." In: *arXiv preprint arXiv:1504.00941*.

Didier Le Gall (1991). "MPEG: A video compression standard for multimedia applications." In: *Communications of the ACM* 34.4, pp. 46–59.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner (1998). "Gradient-based learning applied to document recognition." In: *Proceedings of the IEEE* 86.11.

Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang (2017). "Unsupervised representation learning by sorting sequences." In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 667–676.

Alberto Quattrini Li, Marios Xanthidis, Jason M O'Kane, and Ioannis Rekleitis (2016). "Active localization with dynamic obstacles." In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 1902–1909.

Xueting Li, Sifei Liu, Shalini De Mello, Xiaolong Wang, Jan Kautz, and Ming-Hsuan Yang (2019). "Joint-task Self-supervised Learning for Temporal Correspondence." In: *Neruips*.

Yingzhen Li and Stephan Mandt (2018). "Disentangled sequential autoencoder." In: *arXiv preprint arXiv:1803.02991*.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra (2016). "Continuous control with deep reinforcement learning." In: *ICLR*.

Iou-Jen Liu, Raymond A Yeh, and Alexander G Schwing (2019). "PIC: Permutation Invariant Critic for Multi-Agent Deep Reinforcement Learning." In: *arXiv preprint arXiv:1911.00025*.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang (2015). "Deep learning face attributes in the wild." In: *ICCV*.

Steven R Livingstone and Frank A Russo (2018). "The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English." In: *PloS one* 13.5, e0196391.

Francesco Locatello, Stefan Bauer, Mario Lucic, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem (2018). "Challenging common assumptions in the unsupervised learning of disentangled representations." In: *arXiv preprint arXiv:1811.12359*.

Gucan Long, Laurent Kneip, Jose M Alvarez, Hongdong Li, Xiaohu Zhang, and Qifeng Yu (2016). "Learning image matching by simply watching video." In: *European Conference on Computer Vision*. Springer, pp. 434–450.

Christos Louizos, Karen Ullrich, and Max Welling (2017a). "Bayesian compression for deep learning." In: *NIPS*.

Christos Louizos, Max Welling, and Diederik P Kingma (2017b). "Learning Sparse Neural Networks through L_0 Regularization." In: *arXiv preprint arXiv:1712.01312*.

David G Lowe (1999). "Object Recognition from Local Scale-Invariant Features." In: *International Conference on Computer Vision (ICCV)*.

Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch (2017). "Multi-agent actor-critic for mixed cooperative-competitive environments." In: *Advances in neural information processing systems*, pp. 6379–6390.

Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J Leonard, David Cox, Peter Corke, and Michael J Milford (2015). "Visual place recognition: A survey." In: *IEEE Transactions on Robotics* 32.1, pp. 1–19.

Yongyi Lu, Yu-Wing Tai, and Chi-Keung Tang (2017). "Conditional cyclegan for attribute guided face image generation." In: *arXiv preprint arXiv:1705.09966*.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh (2017). "The concrete distribution: A continuous relaxation of discrete random variables." In: *International Conference on Learning Representations (ICLR)*.

Sridhar Mahadevan and Mauro Maggioni (2007). "Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes." In: *Journal of Machine Learning Research* 8.Oct, pp. 2169–2231.

Shie Mannor, Ishai Menache, Amit Hoze, and Uri Klein (2004). “Dynamic abstraction in reinforcement learning via clustering.” In: *Proceedings of the twenty-first international conference on Machine learning*. ACM, p. 71.

Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, and Zhen Wang (2016). “Multi-class Generative Adversarial Networks with the L2 Loss Function.” In: *arXiv*.

Amy McGovern and Andrew G Barto (2001). “Automatic discovery of subgoals in reinforcement learning using diverse density.” In:

Lars Mescheder, Sebastian Nowozin, and Andreas Geiger (2017). “Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks.” In: *ICML*.

Jan Hendrik Metzen (2013). “Learning graph-based representations for continuous reinforcement learning domains.” In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pp. 81–96.

Mehdi Mirza and Simon Osindero (2014). “Conditional generative adversarial nets.” In: *arXiv preprint arXiv:1411.1784*.

Dmytro Mishkin and Jiri Matas (2015). “All you need is a good init.” In: *arXiv preprint arXiv:1511.06422*.

Ishan Misra, C Lawrence Zitnick, and Martial Hebert (2016). “Shuffle and learn: unsupervised learning using temporal order verification.” In: *European Conference on Computer Vision*. Springer, pp. 527–544.

Tom M Mitchell (1980). *The need for biases in learning generalizations*. Department of Computer Science, Laboratory for Computer Science Research, Rutgers Univ. New Jersey.

Takeru Miyato and Masanori Koyama (2018). “cGANs with projection discriminator.” In: *arXiv preprint arXiv:1802.05637*.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida (2018). “Spectral normalization for generative adversarial networks.” In: *arXiv preprint arXiv:1802.05957*.

Andriy Mnih and Yee Whye Teh (2012). “A fast and simple algorithm for training neural probabilistic language models.” In: *arXiv preprint arXiv:1206.6426*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller (2013). "Playing atari with deep reinforcement learning." In: *arXiv preprint arXiv:1312.5602*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. (2015). "Human-level control through deep reinforcement learning." In: *Nature*.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu (2016a). "Asynchronous methods for deep reinforcement learning." In: *International Conference on Machine Learning*, pp. 1928–1937.

Volodymyr Mnih, John Agapiou, Simon Osindero, Alex Graves, Oriol Vinyals, Koray Kavukcuoglu, et al. (2016b). "Strategic attentive writer for learning macro-actions." In: *arXiv preprint arXiv:1606.04695*.

Hossein Mobahi, Ronan Collobert, and Jason Weston (2009). "Deep learning from temporal coherence in video." In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, pp. 737–744.

Alexander Mott, Daniel Zoran, Mike Chrzanowski, Daan Wierstra, and Danilo Jimenez Rezende (2019). "Towards interpretable reinforcement learning using attention augmented agents." In: *Advances in Neural Information Processing Systems*, pp. 12329–12338.

Kevin P Murphy (2000). *A survey of POMDP solution techniques*. Tech. rep.

Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine (2018). "Visual reinforcement learning with imagined goals." In: *Advances in Neural Information Processing Systems*, pp. 9191–9200.

Eric Nalisnick and Padhraic Smyth (2016). "Stick-breaking variational autoencoders." In: *arXiv preprint arXiv:1605.06197*.

Radford M Neal (1990). "Learning stochastic feedforward networks." In: *Technical Report*.

Simon Niklaus and Feng Liu (2018). "Context-aware synthesis for video frame interpolation." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1701–1710.

Augustus Odena, Christopher Olah, and Jonathon Shlens (2017). “Conditional image synthesis with auxiliary classifier gans.” In: *ICML*.

Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu (2016a). “Conditional Image Generation with PixelCNN Decoders.” In: *NIPS*.

Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu (2016b). “Pixel recurrent neural networks.” In: *ICML*.

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

Ian Osband, Benjamin Van Roy, and Zheng Wen (2016). “Generalization and exploration via randomized value functions.” In: *ICML*.

Ian Osband, Daniel Russo, Zheng Wen, and Benjamin Van Roy (2017). “Deep exploration via randomized value functions.” In: *arXiv*.

Georg Ostrovski, Marc G Bellemare, Aaron van den Oord, and Rémi Munos (2017). “Count-based exploration with neural density models.” In: *ICML*.

Yudha P Pane, Subramanya P Nageshrao, and Robert Babuška (2016). “Actor-critic reinforcement learning for tracking control in robotics.” In: *Decision and Control (CDC), 2016 IEEE 55th Conference on*. IEEE, pp. 5819–5826.

Emilio Parisotto, H Francis Song, Jack W Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant M Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. (2019). “Stabilizing transformers for reinforcement learning.” In: *arXiv preprint arXiv:1910.06764*.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell (2017). “Curiosity-driven exploration by self-supervised prediction.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 16–17.

Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung (2016). “A benchmark dataset and evaluation methodology for video object segmentation.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 724–732.

Karl Pertsch, Oleh Rybkin, Jingyun Yang, Kosta Derpanis, Joseph Lim, Kostas Daniilidis, and Andrew Jaegle (2019). “KeyIn: Discovering Subgoal Structure with Keyframe-based Video Prediction.” In: *arXiv preprint arXiv:1904.05869*.

Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz (2017). "Parameter space noise for exploration." In: *arXiv*.

Sam Prentice and Nicholas Roy (2007). "The Belief Roadmap: Efficient Planning in Linear POMDPs by Factoring the Covariance." In: *Makoto Kaneko and Yoshihiko Nakamura (Eds.)*

Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adria Puigdomenech, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell (2017). "Neural episodic control." In: *ICML*.

Senthil Purushwalkam and Abhinav Gupta (2016). "Pose from action: Unsupervised learning of pose features based on motion." In: *arXiv preprint arXiv:1609.05420*.

Sébastien Racanière, Théophane Weber, David Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adria Puigdomenech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. (2017). "Imagination-augmented agents for deep reinforcement learning." In: *Advances in neural information processing systems*, pp. 5690–5701.

Alec Radford, Luke Metz, and Soumith Chintala (2016). "Unsupervised representation learning with deep convolutional generative adversarial networks." In: *ICLR*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever (2019). "Language Models are Unsupervised Multitask Learners." In:

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee (2016). "Generative adversarial text to image synthesis." In: *ICML*.

Scott Reed, Aäron van den Oord, Nal Kalchbrenner, Sergio Gómez Colmenarejo, Ziyu Wang, Dan Belov, and Nando de Freitas (2017). "Parallel multiscale autoregressive density estimation." In: *arXiv preprint arXiv:1703.03664*.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun (2015). "Faster r-cnn: Towards real-time object detection with region proposal networks." In: *Advances in neural information processing systems*, pp. 91–99.

Danilo Rezende and Shakir Mohamed (2015). "Variational inference with normalizing flows." In: *ICML*.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra (2014). "Stochastic backpropagation and approximate inference in deep generative models." In: *International Conference on Machine Learning (ICML)*.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. (2015). "Imagenet large scale visual recognition challenge." In: *International journal of computer vision* 115.3, pp. 211–252.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell (2016). "Progressive neural networks." In: *arXiv preprint arXiv:1606.04671*.

Ruslan Salakhutdinov and Geoffrey Hinton (2009). "Deep boltzmann machines." In: *Artificial intelligence and statistics*, pp. 448–455.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen (2016). "Improved techniques for training gans." In: *NIPS*.

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma (2017). "PixelCNN++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications." In: *arXiv*.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap (2017). "A simple neural network module for relational reasoning." In: *Advances in Neural Information Processing Systems*.

Andrew Michael Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan Daniel Tracey, and David Daniel Cox (2018). "On the information bottleneck theory of deep learning." In:

Karl Schmeckpeper, Annie Xie, Oleh Rybkin, Stephen Tian, Kostas Daniilidis, Sergey Levine, and Chelsea Finn (2019). "Learning Predictive Models From Observation and Interaction." In: *arXiv preprint arXiv:1912.12773*.

Tanner Schmidt, Richard Newcombe, and Dieter Fox (2016). "Self-supervised visual descriptor learning for dense correspondence." In: *IEEE Robotics and Automation Letters* 2.2, pp. 420–427.

B.K. Schuiling (2017). *GamePlay Football*. `github.com/BazkieBumpercar/GameplayFootball`.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz (2015). "Trust region policy optimization." In: *ICML*.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel (2016). "High-dimensional continuous control using generalized advantage estimation." In: *ICLR*.

Wenling Shang and Kihyuk Sohn (2019). "Attentive conditional channel-recurrent autoencoding for attribute-conditioned face synthesis." In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, pp. 1533–1542.

Wenling Shang, Kihyuk Sohn, Diogo Almeida, and Honglak Lee (2016). "Understanding and improving convolutional neural networks via concatenated rectified linear units." In: *ICML*.

Wenling Shang, Kihyuk Sohn, and Yuandong Tian (2018). "Channel-recurrent autoencoding for image modeling." In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, pp. 1195–1204.

Wenling Shang, Alexander Trott, Stephan Zheng, Caiming Xiong, and Richard Socher (2019a). "Learning World Graphs to Accelerate Hierarchical Reinforcement Learning." In:

Wenling Shang, Douwe van der Wal, Herk van Hoof, and Max Welling (2019b). "Stochastic Activation Actor-Critic Methods." In: *ECML-PKDD*.

Wenling Shang, Lasse Espeholt, Anton Raichuk, and Tim Salimans (2020a). "Agent-centric Representations for Multi-agent Reinforcement Learning." In:

Wenling Shang, Sifei Liu, Arash Vahdat, Shalini De Mello, and Jan Kautz (2020b). "Decompose Video Representations from Temporal Coherence and Dynamics." In:

E Shechtman. "Actions as space-time shapes." In:

Jiaxin Shi, Hanwang Zhang, and Juanzi Li (2019). "Explainable and explicit visual reasoning over scene graphs." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8376–8384.

Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe (2019). "Animating arbitrary objects via deep motion transfer." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2377–2386.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. (2016). "Mastering the game of Go with deep neural networks and tree search." In: *Nature* 529.7587.

Karen Simonyan and Andrew Zisserman (2014). "Two-stream convolutional networks for action recognition in videos." In: *Advances in neural information processing systems*, pp. 568–576.

Özgür Şimşek, Alicia P Wolfe, and Andrew G Barto (2005). "Identifying useful subgoals in reinforcement learning by local graph partitioning." In: *Proceedings of the 22nd international conference on Machine learning*. ACM, pp. 816–823.

Kihyuk Sohn (2016). "Improved deep metric learning with multi-class n-pair loss objective." In: *Advances in Neural Information Processing Systems*, pp. 1857–1865.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan (2015). "Learning structured output representation using deep conditional generative models." In: *Advances in Neural Information Processing Systems*.

Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov (2015). "Unsupervised learning of video representations using lstms." In: *International conference on machine learning*, pp. 843–852.

Chen Sun, Per Karlsson, Jiajun Wu, Joshua B Tenenbaum, and Kevin Murphy (2019). "Stochastic Prediction of Multi-Agent Interactions from Partial Observations." In: *arXiv preprint arXiv:1902.09641*.

Richard S. Sutton and Andrew G. Barto (1998a). *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press. ISBN: 0-262-19398-1. URL: `http://www.cs.ualberta.ca/\%7Esutton/book/ebook/the-book.html`.

Richard S Sutton and Andrew G Barto (1998b). "Reinforcement Learning: An Introduction." In: *artificial intelligence*.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour (2000). "Policy gradient methods for reinforcement learning with function approximation." In: *Advances in neural information processing systems*, pp. 1057–1063.

Yichuan Tang and Ruslan R Salakhutdinov (2013). "Learning stochastic feed-forward neural networks." In: *NIPS*.

Matthew E Taylor and Peter Stone (2009). "Transfer learning for reinforcement learning domains: A survey." In: *Journal of Machine Learning Research* 10.Jul, pp. 1633–1685.

Lucas Theis, Aaron van den Oord, and Matthias Bethge (2016). "A note on the evaluation of generative models." In: *ICLR*.

Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner (2016). "Face2face: Real-time face capture and reenactment of rgb videos." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2387–2395.

Sebastian Thrun (1998). "Learning metric-topological maps for indoor mobile robot navigation." In: *Artificial Intelligence* 99.1, pp. 21–71.

Yuandong Tian and Qucheng Gong (2017). "Latent forward model for Real-time Strategy game planning with incomplete information." In: *NIPS Symposium*.

Naftali Tishby, Fernando C Pereira, and William Bialek (2000). "The information bottleneck method." In: *arXiv*.

Emanuel Todorov (2008). "General duality between optimal control and estimation." In: *CDC*.

Ahmed Touati, Harsh Satija, Joshua Romoff, Joelle Pineau, and Pascal Vincent (2020). "Randomized value functions via multiplicative normalizing flows." In: *Uncertainty in Artificial Intelligence*. PMLR, pp. 422–432.

Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz (2018). "Mocogan: Decomposing motion and content for video generation." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1526–1535.

Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie (2015). "Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection." In: *CVPR*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention is all you need." In: *Advances in Neural Information Processing Systems*.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu (2017). "FeUdal networks for hierarchical reinforcement learning." In: *International Conference on Machine Learning (ICML)*.

Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee (2017). "Decomposing motion and content for natural video sequence prediction." In: *arXiv preprint arXiv:1706.08033*.

Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba (2016). "Generating videos with scene dynamics." In: *Advances In Neural Information Processing Systems*, pp. 613–621.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie (2011). "The caltech-ucsd birds-200-2011 dataset." In:

Angelina Wang, Thanard Kurutach, Kara Liu, Pieter Abbeel, and Aviv Tamar (2019a). "Learning Robotic Manipulation through Visual Planning and Acting." In: *arXiv preprint arXiv:1905.04411*.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick (2016a). "Learning to reinforcement learn." In: *arXiv*.

Jiangliu Wang, Jianbo Jiao, Linchao Bao, Shengfeng He, Yunhui Liu, and Wei Liu (2019b). "Self-Supervised Spatio-Temporal Representation Learning for Videos by Predicting Motion and Appearance Statistics." In: *CVPR*, pp. 4006–4015.

Xiaolong Wang and Abhinav Gupta (2015). "Unsupervised learning of visual representations using videos." In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2794–2802.

Zhendong Wang and Mingyuan Zhou (2019). "Thompson Sampling via Local Uncertainty." In: *arXiv preprint arXiv:1910.13673*.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas (2016b). "Sample efficient actor-critic with experience replay." In: *ICLR*.

Paul J Werbos (1982). "Applications of advances in nonlinear sensitivity analysis." In: *System modeling and optimization*. Springer.

Ronald J Williams (1992). "Simple statistical gradient-following algorithms for connectionist reinforcement learning." In: *Machine learning* 8.3-4, pp. 229–256.

Yuhuai Wu, Yuri Burda, Ruslan Salakhutdinov, and Roger Grosse (2016). "On the quantitative analysis of decoder-based generative models." In: *arXiv*.

Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba (2017). "Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation." In: *NIPS*.

Yuxin Wu and Yuandong Tian (2016). "Training agent for first-person shooter game with actor-critic curriculum learning." In:

Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee (2016). "Attribute2image: Conditional image generation from visual attributes." In: *ECCV*.

Fei Yang, Jue Wang, Eli Shechtman, Lubomir Bourdev, and Dimitri Metaxas (2011). "Expression flow for 3D-aware face component transfer." In: *ACM Transactions on Graphics (TOG)* 30.4, p. 60.

Raymond A Yeh, Alexander G Schwing, Jonathan Huang, and Kevin Murphy (2019). "Diverse generation for multi-agent sports games." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4610–4619.

Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li (2014). "Learning face representation from scratch." In: *arXiv*.

Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao (2015). "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop." In: *arXiv*.

Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, et al. (2018). "Relational deep reinforcement learning." In: *arXiv preprint arXiv:1806.01830*.

Xiaohua Zhai and et al. (2019). "The Visual Task Adaptation Bencmark." In: *arXiv preprint*.

Eric Zhan, Stephan Zheng, Yisong Yue, Long Sha, and Patrick Lucey (2018). "Generative multi-agent behavioral cloning." In: *arXiv*.

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris Metaxas (2016a). "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks." In: *arXiv*.

Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena (2019). "Self-attention generative adversarial networks." In: *International Conference on Machine Learning*. PMLR, pp. 7354–7363.

Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao (2016b). "Joint face detection and alignment using multitask cascaded convolutional networks." In: *IEEE Signal Processing Letters* 23.10, pp. 1499–1503.

Kaiqing Zhang, Zhuoran Yang, Han Liu, Tong Zhang, and Tamer Başar (2018). "Fully decentralized multi-agent reinforcement learning with networked agents." In: *arXiv preprint arXiv:1802.08757*.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros (2017). "Unpaired image-to-image translation using cycle-consistent adversarial networks." In: *ICCV*.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey (2008). "Maximum entropy inverse reinforcement learning." In:

Google Brain Zurich (2020). *Google Research Football League*. `research-football.dev`.

SAMENVATTING - SUMMARY IN DUTCH

Dit proefschrift *Crafting Deep Learning Models for Reinforcement Learning and Computer Vision Applications* richt zich op het ontwerpen van nieuwe en effectieve kaders voor het leren van representaties. Er zijn twee belangrijke aspecten in onze voorgestelde benaderingen: architectuurontwerp van neurale netwerkmodellen en het opstellen van doelfuncties. Om aan te tonen hoe elk aspect kan worden ontworpen, verdiepen we ons in representatieve toepassingen uit twee belangrijke studiegebieden in de kunstmatige intelligentie, namelijk reinforcement learning en computervisie. In beide gebieden benadrukken we hoe abstracte representaties kunnen worden gemanipuleerd om sterke inductieve aannames in te bouwen over de doeltaken en het type beschikbare data. We hopen dat onze voorbeelden licht kunnen werpen op toekomstige inspanningen om problemen op aanverwante gebieden en daarbuiten aan te pakken. .