Université de Montréal

Emergence of Language-like Latents in Deep Neural Networks

par

Yuchen Lu

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée en vue de l'obtention du grade de
Philosophiæ Doctor (Ph.D.)
en Informatique

May 30, 2023

Université de Montréal

Faculté des arts et des sciences

Cette thèse intitulée

Emergence of Language-like Latents in Deep Neural Networks

présentée par

Yuchen Lu

a été évaluée par un jury composé des personnes suivantes :

Aishwarya Agrawal
(président-rapporteur)

Aaron Courville
(directeur de recherche)

Alessandro Sordoni
(membre du jury)

Stefan Lee
(examinateur externe)

Dhruv Batra
(remplaçante)

(représentant du doyen de la FESP)

Résumé

L'émergence du langage est considérée comme l'une des marques distinctives de l'intelligence humaine. Par conséquent, nous émettons l'hypothèse que l'émergence de latents ou de représentations similaires au langage dans un système d'apprentissage profond pourrait aider les modèles à atteindre une meilleure généralisation compositionnelle et hors distribution. Dans cette thèse, nous présentons une série de travaux qui explorent cette hypothèse dans différents domaines, notamment l'apprentissage interactif du langage, l'apprentissage par imitation et la vision par ordinateur.

Nous commençons par nous concentrer sur le maintien de la langue naturelle en tant que latents dans un système de réseau neuronal profond. Pendant l'apprentissage interactif du langage, les agents du réseau neuronal profond utilisent la communication pour collaborer, et les communications linguistiques entre eux peuvent être considérées comme des représentations latentes dans le système. Inspirés par la recherche en communication émergente, nous proposons l'Apprentissage Itératif Semé (AIS) pour maintenir la structure linguistique de cette communication lors de l'entraînement interactif. Les agents résultants obtiennent un meilleur score de généralisation lorsque les "latents linguistiques" qui sont communiqués ne dérivent pas.

Nous enquêtons ensuite sur l'ajout d'un biais inductif pour encourager l'émergence de structures similaires à la langue dans l'apprentissage par imitation. Nous nous concentrons sur la découverte de structures compositionnelles à partir de démonstrations non structurées lors de l'apprentissage par imitation. Nous proposons le Réseau de Politique de Mémoire Ordre (OMPN), qui favorise l'émergence d'une hiérarchie de sous-tâches dans ses mises en mémoire. Nous constatons que notre modèle peut segmenter les trajectoires en sous-tâches significatives et surpasser les références en termes d'achèvement des tâches.

Nous analysons enfin les propriétés ressemblant à un langage émergentes dans les méthodes de pointe existantes. Nous nous concentrons sur l'analyse des représentations dans l'apprentissage auto-supervisé. En établissant le lien entre l'apprentissage de représentations non supervisé et l'émergence du langage, nous identifions l'expressivité et l'apprentissage comme deux propriétés clés. Nous proposons un cadre d'évaluation pour les méthodes d'apprentissage auto-supervisé sans accès aux étiquettes en aval. Notre étude empirique à grande échelle

montre que les méthodes ayant une meilleure généralisation hors distribution produisent également des représentations à la fois apprenables et expressives.

Mots-clés: Apprentissage Profond, Émergence du Langage, Compositionnalité, Apprentissage par Imitation, Apprentissage Auto-supervisé

Abstract

The emergence of language is regarded as one of the hallmarks of human intelligence. Therefore, we hypothesize that the emergence of language-like latents or representations in a deep learning system could help models achieve better compositional and out-of-distribution generalization. In this thesis, we present a series of papers that explores this hypothesis in different fields including interactive language learning, imitation learning and computer vision.

We first focus on maintaining natural language as latents in a deep neural network system. During interactive language learning, deep neural network agents use communication to collaborate, and the language communications among them can be thought of as latent representations in the system. Inspired by research in emerging communication, we propose Seeded Iterated Learning (SIL) to maintain the linguistic structure of this communication during interactive training. The resulting agents achieve a better generalization score when the "language latents" being communicated do not drift.

We then investigate adding inductive bias to encourage the emergence of language-like structures in imitation learning. We focus on discovering compositional structures from unstructured demonstrations during imitation learning. We propose the Ordered Memory Policy Network (OMPN), which encourages the emergence of a subtask hierarchy in its memory layouts. We find that our model can segment the trajectories into meaningful subtasks and outperform baselines in terms of task completion

We finally analyze the emerging language-like properties in existing state-of-the-art methods. We focus on analyzing representations in self-supervised learning. By drawing the connection between unsupervised representation learning and language emergence, we identify expressiveness and learnability as two key properties. We propose an evaluation framework for self-supervised learning methods without access to downstream labels. Our large-scale empirical study shows that methods with better out-of-distribution generalization also produce representations that both are learnable and expressive.

Keywords: Deep Learning, Language Emergence, Compositionality, Imitation Learning, Self-Supervised Learning

Contents

List of tables

List of figures

Acknowledgements

As I reflect on my journey with Mila at the University of Montreal, I am filled with immense gratitude for the experience. When I first joined, I was a young undergraduate with a vague understanding of deep learning research, but a strong passion for understanding how human intelligence works. Witnessing the transformative power of deep learning in applications like smart assistants and self-driving cars, and being a part of these changes alongside my fellow lab mates, has been an unforgettable honor. It is with mixed emotions that I bid farewell to this chapter of my life, the people, and the lifestyle that have become such an integral part of me. I am grateful for the support and guidance of so many individuals who have helped me along the way, without whom this journey would not have been possible. These past six years have been a truly important and rewarding period in my life, and I will carry the lessons learned and the memories made with me always.

I would like to express my deepest gratitude to my advisor, Aaron Courville. Working with one of the most renowned researchers in the field of deep learning has been a true honor for me. Throughout my research, Aaron's guidance and input have been invaluable, and I am forever thankful for his mentorship. Aaron's influence on my research taste cannot be overstated, as he has taught me to see the big picture of the research landscape and to tackle ambitious fundamental problems instead of simply following trends. I vividly remember our first meeting, where I proposed to work on generative models like Variational Autoencoders, which were quite popular at the time. However, Aaron suggested that I should work on language learning and its intersection with reinforcement learning. This turned out to be an excellent decision, and I am grateful to have worked on some of the most interesting and unique problems in deep learning. Throughout this journey, Aaron's supportive and encouraging attitude has empowered me to try new approaches, even unconventional ones. He has also been extremely hands-on in helping me refine every aspect of my papers, particularly during those late nights before deadlines. I cannot thank him enough for the extra effort he has put into making my work the best it can be. On a personal note, Aaron is also a wonderful friend. His infectious laughter and positive outlook have always lifted my spirits, even during early morning meetings. Without Aaron as my advisor, my journey would have

been far less enjoyable, and I am proud to have been his student. Once again, thank you, Aaron, for your unwavering support, guidance, and friendship.

I would like to express my sincere gratitude to all my collaborators, whose contributions have been instrumental in shaping me into the researcher I am today. In particular, I would like to thank Alessandro Sordoni, Yikang Shen, and Florian Strub for their invaluable support and guidance.

Alessandro is an exceptionally imaginative and innovative researcher, whose critical eye has helped me examine and refine my ideas. Our slack conversations have been particularly enlightening, as Alessandro's fresh perspectives have shaped my research ideas and identified concrete next steps and experiments to test my hypotheses. I am deeply grateful for his mentorship and his contributions to my growth as a researcher. Yikang's expertise in natural language processing has been a tremendous asset to our collaboration on unsupervised task decomposition. I have learned a great deal from his experience in designing network architecture inductive bias, and our time spent hanging out has also been a source of joy. I am grateful for his friendship and his contributions to my intellectual growth. Finally, I would like to acknowledge Florian's exceptional knowledge and expertise in reinforcement learning and emergent communication. His guidance and support have been invaluable, and I have learned a great deal from him about the importance of good software engineering practices in research.

I want to also express my gratitude to Tong Che, George Tucker, Surya Bhupatiraju, Shane Gu, Sergey Levine, Yoshua Bengio, Philip Paquette, Seton Steven Bocco, Max Smith, Ortiz-Gagné Satya, Jonathan Kummerfeld, Joelle Pineau, Satinder Singh, Soumye Singhal, Olivier Pietquin, Ankit Vani, Max Schwarzer, Eeshan Dhekane, Zhen Liu, Aristide Baratin, Romain Laroche, Mengdi Xu, Shun Zhang, Ding Zhao, Joshua Tenenbaum, Chuang Gan, Siddharth Verma, Rui Hou, Hanchao Yu, Nicolas Ballas, Madian Khabsa, Amjad Almahairi. It's my honour to work closely with these people, and I thank them from the bottom of my heart for their mentorship, guidance, and friendship.

I want to thank the Mila community. It gathers a lot of brilliant researchers and created a unique atmosphere of collaboration. I met many interesting people in Mila. Among them, I especially want to thank Jie Fu, Min Lin, Xingdi Yuan, Shawn Tan, Jae Hyun Lim, Zhen Liu, Ying Zhang, Saizheng Zhang, Chen Xing, Olga Luo, Zhihao Luo, Yusong Wu, Erqun Dong, Jianan Zhao. I feel fortunate to form friendship to these people, and I cherish the time we spent together.

Last but not least, I also want to thank my parents for their unwavering support and love in this journey.

Chapter 1

Introduction

1.1. A Brief History of Deep Learning

Deep learning is a subset of machine learning that has emerged as a popular approach to artificial intelligence in recent years. It involves training artificial neural networks, which are designed to simulate the behavior of the human brain, to recognize patterns and make predictions. The origins of neural networks can be traced back to the 1940s, when Warren McCulloch and Walter Pitts created the perceptron, a mathematical model of a single neuron (Rosenblatt, 1958). However, it was criticized that such an approach to neural networks could not be well translated into multiple layers (Minsky and Papert, 2017). Nevertheless, it doesn't stop deep learning pioneers from the 1980s and 1990s from implementing these models on various applications. The seminal work of LeCun et al. (1998) proposes to use back-propagation to train a hand written digit classifier with multiple layers of neural network.

Over the years, deep learning has undergone a remarkable transformation due to the increased dataset sizes and advancements in computational power. As a result, it has revolutionized the field of artificial intelligence, offering state-of-the-art performance in various tasks such as image recognition, natural language processing, and speech recognition. Deep learning models can extract abstract information from massive datasets and store them into layers of representations, as demonstrated by (LeCun et al., 2015). The success of deep learning and neural networks is evident from their remarkable performance on various benchmark datasets. For instance, the ImageNet challenge, which is a popular competition for image recognition, was won by a deep neural network called AlexNet (Krizhevsky et al., 2012), with a significant margin over the previous state-of-the-art methods. Similarly, the language model GPT-3 (Brown et al., 2020) has shown impressive performance on various natural language processing tasks. It seems that stacking more layers along with proper training techniques and smart engineering tricks can lead to tremendous empirical success.

In the modern context, researchers are further pushing this trend and we are now squarely in the era of Large Language Models (Devlin et al., 2018; Raffel et al., 2020). These models,

which are trained with massive amounts of data, are demonstrating emergent abilities (Wei et al., 2022), such as few-shot in-context learning (Brown et al., 2020). The discovery of these abilities within deep neural networks has opened up exciting possibilities for industrial applications.

1.2. Motivation: The Lack of Systematicity

While deep neural networks have achieved astonishing results, there are still some problems that prevent these models from being mpre widely deployed in practical applications.

For example, in computer vision, a tiny change to a fraction of the image pixels (Goodfellow et al., 2014) can lead to a total failure in super-human image classification neural networks. In the Question Answering domain, Jia and Liang (2017) discovered that state-of-the-art models can be fooled by adding an additional distracting sentence to the end of the context paragraph. Even in those seemingly omniscient large language models, recent work shows that these models can surprisingly perform well with irrelevant prompts (Webson and Pavlick, 2022) or demonstrations with random labels (Min et al., 2022), which begs the question whether these models truly understand the text. Additionally, Razeghi et al. (2022) observe that large language models generalize poorly when it comes to rare numbers in the training data, raising the question of the extent of their generalization beyond the pretraining dataset. Although these failures of deep neural network models are sometimes referred to differently under different contexts (e.g., adversarial attack, spurious features, dataset bias, etc.), Geirhos et al. (2020) provide a good summary of these failures called "shortcut learning". By following an unintended "shortcut" solution, the model can behave well on standard benchmarks but fail to generalize systematically to real-world scenarios. In this thesis, we refer these symptoms of deep neural networks as the lack of systematicity (Fodor and Pylyshyn, 1988).

While this concept of lack of systematicity is commonly used to describe the generalization failure of seq2seq models (Lake and Baroni, 2018; Bastings et al., 2018), we argue that this notion can be extended to the problems discussed in this work. Specifically, even when our current models have learned solutions to particular tasks, they may fail to generalize to semantically related scenarios. Systematicity is a fundamental property of human cognition and natural language, and addressing this issue could potentially bring our models one step closer to achieving human-level intelligence.

1.3. Previous Work on Achieving Systematicity

The issue of achieving systematicity within neural networks has been previously addressed in the literature.

Tai et al. (2015) proposed integrating tree structures into a seq2seq model and demonstrated improvements in semantic related tasks. Andreas and colleagues Andreas et al.

(2016) introduced Neural Module Networks (NMN) to solve the problem of Visual Question Answering, where the computational flow or layout of the neural network can reflect the underlying linguistic structure of the problem description. Subsequent work explored similar ideas in reinforcement learning (Andreas et al., 2017), text reasoning (Gupta et al., 2020), few-shot classification (Andreas et al., 2018), among others. It has also been demonstrated that systematicity can potentially be achieved when ground-truth layouts (or programs) are given (Bahdanau et al., 2018), although annotating such layouts from text descriptions is a tedious task. With the emergence of large pretrained language models (Brown et al., 2020; Raffel et al., 2020), recent works suggest that simply scaling the model can help with achieving systematicity. Warstadt and colleagues Warstadt et al. (2020) argue that larger models would prefer linguistic features over surface features. However, some work also points out that scaling alone might have limited benefits in reducing the compositional generalization gap (Qiu et al., 2022), unless accompanied by clever prompting (Press et al., 2022). Therefore, the lack of systematicity remains a fundamental problem for deep learning, even in the context of large language models.

1.4. Emerging Language-like Latents in Neural Networks

We propose the hypothesis that *the emergence of language-like latents can facilitate systematic generalization in deep neural networks.*

Firstly, we define latents as anything situated between the inputs and outputs of a deep learning system. This can refer to a single neuron, the representation from the last convolutional layers, or even an intermediate natural language sequence, depending on the context. Secondly, "language-like" is an umbrella term for any linguistic properties that these latents may exhibit. Depending on the type of latents being studied, we may use existing metrics such as BLEU scores for natural language sentences, or develop our own metric based on abstract concepts such as learnability and expressiveness derived from the emergence of language (Kirby, 2002). Lastly, the systematic generalization in which we are interested, pertains to compositional or out-of-distributional generalization, which can be identified in various benchmark tasks.

While it is interesting to examine the emergence of language-like latents in the existing approaches, we are also interested in identifying the inductive bias that can enforce them. Inductive bias refers to the set of assumptions that a learner employs when presented with novel inputs (Mitchell, 1980). All machine learning methods require some degree of inductive bias or constraints to be effective. However, we believe that overly relying on hard-coded constraints may hinder the scalability of neural networks, which is one of their major strengths. This distinguishes our work from previous approaches such as neural symbolic methods and trivial neural module networks (Yi et al., 2018; Amizadeh et al., 2020), where a pre-defined set

of modules or vocabulary is necessary, thereby limiting their scalability. Instead, we construct our inductive bias by taking inspiration from emerging communication methods (Lazaridou et al., 2018; Kharitonov et al., 2020; Harding Graesser et al., 2019), which are less rigid. However, since these studies typically focus on small-scale toy experiments, it remains a challenge to apply their insights to larger models and datasets.

1.5. Relation to the Language of Thought Hypothesis

Our hypothesis is centered around the vital role that language plays in human cognitive development. Initially, it may seem closely linked to the Language of Thought Hypothesis (LOTH), which proposes that cognitive functions, such as productivity and systematicity, necessitate the existence of a linguistically-structured mental representation (Fodor, 1975). LOTH is commonly summarized as the idea that "thoughts are sentences in the head." However, classical LOTH considers mental representations as symbolic systems with combinatorial syntax and compositional semantics (Fodor and Pylyshyn, 1988), and is thus frequently used to challenge connectionist models like neural networks. In this thesis, our goal is to demonstrate the potential for neural networks to create latent representations with certain linguistic traits, such as compositionality. While these latent representations may lack the strictest forms of constituency or symbolic structure found in human language, their emergence could still enhance models in terms of generalization and systematicity.

1.6. Thesis Development

In the following thesis we are going to first introduce some preliminary concepts (Chapter 2) to help readers understand our papers. Now we summary the content of the papers and discuss how they are related to our central hypothesis.

Firstly, we show that preserving language-like latent representations can result in superior generalization. We focus on the problem of Language Drift (Chapter 3 and Chapter 4), which is the phenomenon where a goal-oriented dialogue agent (e.g., for ticket booking) that has been fine-tuned using self-play produces language that deviates from the pretraining corpus. We propose to study this problem in a translation game setting, where the agents collaboratively translate French to German, with English as the intermediate language. we propose an iterated learning algorithm inspired by emerging communication to counter language drift. Our approach not only addresses language drift but also improves generalization to novel inputs. In these works, the goal-oriented dialogue agents is a holistic system, and the natural language sentences exchanged among neural networks can be viewed as the system's "latent representations." Therefore, these papers demonstrate that maintaining language-like latents could result in improved generalization.

Secondly, we demonstrate that incorporating suitable neural architectural inductive bias can facilitate the emergence of language-like latents, resulting in improved compositional generalization. Our investigation focuses on the problem of unsupervised task decomposition through imitation learning, as discussed in Chapter 5. Specifically, given a lengthy sequence of demonstrations, we seek to develop a neural network agent that can effectively comprehend and parse the input into meaningful subtasks that can be applied in a range of contexts. The capacity to accomplish this objective is connected to the notion of compositionality. We propose leveraging an architectural inductive bias referred to as Ordered Memory to achieve the task. We demonstrate that by employing standard backpropagation in behavior cloning, the subtask structures emerge in the model's memory. While the degree to which these subtask structures resemble natural language is an area requiring further inquiry, our findings highlight the potential for designing neural architectural inductive bias to enable the emergence of language-like latents.

Thirdly, we assert that language-like representations may arise in existing scalable deep learning methods without any alteration of algorithms or architecture. In Chapter 6, we examine the relationship between the performance of state-of-the-art self-supervised methods and their representation properties through the lens of language emergence literature. We identify two crucial properties, namely expressiveness and learnability, and measure them respectively by Cluster Learnability (CL) and Intrinsic Dimension (ID). Our extensive empirical investigation of state-of-the-art models demonstrates that if the model is proficient at generalization, the resulting representations exhibit language-like characteristics in that they are both expressive and learnable. Through the integration of CL and ID into a single predictor, we empirically demonstrate that it can forecast the out-of-domain generalization of self-supervised models on various visual classification tasks, yielding improvements with respect to the competing baselines.

Finally, we conclude our thesis and propose potential future research directions on finding language-like latents in the current context of large-scale pretrained models.

1.7. Contributions

The contributions of this thesis are summarized as follows:

- I argue that the *emergence of language-like latents in neural networks can enable better systematic generalization*. I study the hypothesis across a wide spectrum of deep learning applications, including language learning, imitation learning and self-supervised learning.
- I propose to use algorithmic inductive bias to counter the problem of language drift. I was among the first to adapt techniques from emergent communications into large scale machine learning applications beyond toy datasets.

- I design network architectural inductive bias so that the compositional subtask structure emerges, enabling the model to perform unsupervised task decomposition.
- I empirically investigate the existence of language-like representations in the state-of-art self-supervised methods.

Chapter 2

Preliminaries

2.1. Reinforcement Learning

2.1.1. Markov Decision Process

Reinforcement learning focuses on how an agent can learn to make decisions through trial-and-error interactions with an environment. At the heart of reinforcement learning lies the Markov Decision Process (MDP), Markov Decision Processes (MDPs) are a popular framework for modeling decision-making problems in stochastic environments. An MDP is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the state transition probability function, $\mathcal{R}$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor. At each time step t, the agent observes the current state $s_t \in \mathcal{S}$, selects an action $a_t \in \mathcal{A}$, receives a reward $r_t = \mathcal{R}(s_t, a_t)$, and transitions to the next state s_{t+1} according to the probability distribution $\mathcal{P}(s_{t+1} \mid s_t, a_t)$. The goal of an agent in an MDP is to find a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expected cumulative discounted reward:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] \tag{2.1.1}$$

where $\mathbb{E}_\pi$ denotes the expectation with respect to the state-action distribution induced by the policy π.

2.1.2. Policy Gradient Method

One popular method for learning a policy in the MDP is the policy gradient method. It involves computing the gradient of the expected cumulative reward with respect to the parameters of the policy, and using this gradient to update the policy parameters in the direction of increasing reward. The policy gradient theorem is derived from the likelihood ratio method, and it provides a principled way to optimize policies in both discrete and continuous action spaces. Let θ be the parameter vector of the policy $\pi_\theta(a \mid s)$, which is a

probability distribution over actions given the current state. By taking the gradient of the objective function $J(\pi_\theta)$, we have

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) Q^{\pi_\theta}(s_t, a_t) \right] \tag{2.1.2}$$

where $Q^{\pi_\theta}(s_t, a_t)$ is the state-action value function under policy π_θ. Different realizations of the state-action value could lead to different variants of the algorithms. A common approach is to use the Bellman equation (Sutton and Barto, 2018) to re-write the state-action value as:

$$Q^{\pi_\theta}(s_t, a_t) = \mathbb{E}_{\pi_\theta} \left[\sum_{k=t}^{\infty} \gamma^{k-t} r_k \mid s_t, a_t \right] \tag{2.1.3}$$

which is the expected cumulative discounted reward starting from state s_t and taking action a_t under policy π_θ. Using this realization, we can now derive the REINFORCE algorithm (Williams, 1992) as follows:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \sum_{k=t}^{\infty} \gamma^{k-t} r_k \right] \tag{2.1.4}$$

Nevertheless, vanilla REINFORCE can have high variance, which can lead to slow convergence and unstable training. This high variance arises because the gradient is estimated from samples that can be noisy or biased, especially when dealing with high-dimensional and continuous action spaces. These limitations have led to the development of advanced techniques such as Advantage Actor Critic (A2C) (Mnih et al., 2016), Trust Region Policy Optimization (TRPO) (Schulman et al., 2015b), Proximal Policy Optimization (PPO) (Schulman et al., 2017) etc.

To encourage exploration, entropy regularization can be added to the objective function to encourage agents to take a diverse set of actions. The resulting gradient becomes:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \sum_{k=t}^{\infty} \gamma^{k-t} r_k + \beta \nabla_\theta H(\pi_\theta(\cdot|s_t)) \right] \tag{2.1.5}$$

where $H(\pi_\theta(\cdot|s_t))$ is the entropy of the policy distribution, and β is a hyper-parameter that controls the strength of the entropy regularization term.

2.1.3. Behavior Cloning

Behavior cloning is another popular approach for learning policies in reinforcement learning, which involves training a policy to mimic an expert policy using a supervised learning method. Specifically, behavior cloning involves collecting a dataset of expert trajectories $\mathcal{D} = \{ \tau_1, \tau_2, ..., \tau_N \}$ where N is the total number of trajectories. Each trajectory is comprised of a sequence of state-action pairs $\tau_i = (s_{i,1}, a_{i,1}, ..., s_{i,T_i}, a_{i,T_i})$ where T_i is the length of

trajectory. We then train a neural network to predict the expert action given the current state. The objective function of behavior cloning is the cross-entropy loss:

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T_i}\log \pi_\theta(a_{i,t} \mid s_{i,t}) \tag{2.1.6}$$

where N is the number of state-action pairs in the dataset, and $\pi_\theta(a \mid s)$ is the policy parameterized by the neural network with weights θ. The neural network is trained using stochastic gradient descent to minimize the cross-entropy loss.

Behavior cloning has the advantage of being fast and easy to implement, since it only requires a dataset of state-action pairs and a neural network. However, it is prone to the problem of distributional shift, where the training data may not cover the full range of states and actions encountered during execution of the policy, leading to poor generalization performance. Nevertheless, behavior cloning is a good choice when demonstrations are ready and can be used as good initialization for the policy network.

2.1.4. Partially Observable MDP and Recurrent Neural Networks

Partially Observable Markov Decision Processes (POMDPs) have become a more prevalent framework for modeling decision-making problems in uncertain environments. These processes are an extension of Markov Decision Processes (MDPs) that allow for incomplete observations of the state. Within a POMDP, the agent receives an observation that is dependent on the underlying system state, which is not directly observable. The agent must utilize this observation to make a decision that optimizes the expected cumulative reward over time. At each time step, denoted by t, the agent only has access to an observation, o_t, that is a partial representation of the true environment state, s_t. Subsequently, the agent selects an action, a_t, based on its current belief state, h_{t-1}, receives a reward, r_t, and transitions to a new belief state, h_t, based on the belief update function.

Recurrent neural networks (RNNs) have been shown to be effective as policy networks in POMDPs due to their ability to capture temporal dependencies in the observations. To be specific, an RNN is used to model $\pi_\theta(a_t|o_t, h_{t-1})$, and the underlying recurrent states update is $h_t = f_\theta(x_t; h_{t-1})$, where $x_t = (o_t, a_t)$. For example, the vanilla RNN update is

$$\mathbf{h}_t = \tanh(\mathbf{W}^{hh}h_{t-1} + \mathbf{W}^{hx}x_t + \mathbf{b}), \tag{2.1.7}$$

where $\mathbf{W}^{hh}, \mathbf{W}^{hx}$, $\mathbf{b}$ are the parameters to be learned.

In the literature, there are other commonly used variants of RNNs, including long short-term memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997) and gated recurrent units (GRUs) (Cho et al., 2014). With the use of recurrent neural networks, policy gradient methods and behavior cloning techniques can be directly applied. Recent studies have also

demonstrated the effectiveness of more advanced sequence modelling network architectures, such as memory networks (Chapter 5) and transformers (Chen et al., 2021a).

2.2. Emergent Communication

2.2.1. Referential Game

The referential game (Lewis, 1969) is a typical setting among emergent communication research in deep learning (Chaabouni et al., 2019, 2020; Kottur et al., 2017). It is an object selection game between two agents, a sender and a receiver, or sometimes referred to as Alice and Bob. During each round, the sender sees a object $\boldsymbol{x}$ from the object space $\mathcal{X}$, and send a discrete message of length l, $\boldsymbol{m} = (m_1, m_2, ..., m_l)$ to the receiver, where each word comes from a fixed vocabulary $\mathcal{V}$. We then show the receiver a set of $\boldsymbol{k}$ objects (or candidates) $\mathcal{C} = \{\boldsymbol{c}_1, \boldsymbol{c}_2, ..., \boldsymbol{c}_k\}$ where $\boldsymbol{x} \in \mathcal{C}$. Based on the message $\boldsymbol{m}$ and the candidates, the receiver makes a selection $\hat{\boldsymbol{c}}$. After the selection is made, a reward r is generated, which could be as simple as $r(\hat{\boldsymbol{c}}, \boldsymbol{x}) = [\hat{\boldsymbol{c}} == \boldsymbol{x}]$, where $[\cdot]$ is the Iverson bracket. The objects are shuffled and candidates are randomly selected in each round.

To be specific, the policies of the sender and the receiver are π_S and π_R, which are usually implemented with some off-the-shelf sequence modelling neural networks, e.g., a single-layer LSTM (Hochreiter and Schmidhuber, 1997). The sender models $\boldsymbol{p}_{\pi_S}(m_i|\boldsymbol{x}, m_{<i})$, which is the conditional distribution of generating next message token m_i, based on the history $m_{<i}$ and the object $\boldsymbol{x}$. We also denote the message probability as $\boldsymbol{p}_{\pi_S}(\boldsymbol{m}|\boldsymbol{x}) = \Pi_{i=1}^{l} \boldsymbol{p}_{\pi_S}(m_i|\boldsymbol{x}, m_{<i})$, sampling the message autoregressively. Meanwhile, the receiver models $\boldsymbol{p}_{\pi_R}(\hat{\boldsymbol{c}}|\boldsymbol{m}, \mathcal{C})$, the conditional distribution of the selection, given the message and the candidates. Then the objective to be maximized is the expected reward under the policies:

$$J(\pi_S, \pi_R) = \mathbb{E}_{\pi_S, \pi_R}[r] \tag{2.2.1}$$

To optimize this objective function, one applies REINFORCE algorithm (Williams, 1992). That is

$$\nabla J_{\pi_R} = \mathbb{E}_{\pi_S, \pi_R}[r \nabla \log \boldsymbol{p}_{\pi_R}(\hat{\boldsymbol{c}}|\boldsymbol{m}, \mathcal{C}) + \lambda \nabla H(\boldsymbol{p}_{\pi_R})] \tag{2.2.2}$$

$$\nabla J_{\pi_S} = \mathbb{E}_{\pi_S, \pi_R}[r \nabla \log \boldsymbol{p}_{\pi_S}(\boldsymbol{m}|\boldsymbol{x}) + \lambda \nabla H(\boldsymbol{p}_{\pi_S})] \tag{2.2.3}$$

where $H(\cdot)$ is the entropy function to encourage the policy exploration, and λ is a hyper-parameter.

While the above case describe the policy gradient algorithm, in some cases the reward function $r(\hat{\boldsymbol{c}}, \boldsymbol{x})$ is differentiable. Then a hybrid method (Schulman et al., 2015a) can be used

by directly applying backpropagation w.r.t. the receiver.

$$\nabla J_{\pi_R} = \mathbb{E}_{\pi_S,\pi_R}\left[\nabla r + \lambda \nabla H(\boldsymbol{p}_{\pi_R})\right] \tag{2.2.4}$$

For the sender, one can still use policy gradient above, or back-propogation with Gumbel Softmax (Jang et al., 2017a) as is shown in Chapter 3. Through the optimization, a specific communication protocol or language is emerged that can help agents successfully complete the task.

2.2.2. Compositionality in the Emergent Language

To investigate the presence of compositionality in an emergent protocol, a straightforward approach is to evaluate whether agents can utilize it to express novel composite meanings. For instance, can they refer to a blue square when encountering it for the first time, given that they have been exposed to other blue and square objects during training? To accomplish this, a compositional split of objects is typically generated for testing purposes.

However, in addition to task performance, it is also important to evaluate the underlying compositionality of the emergent language. Several commonly employed metrics for this purpose are detailed below:

- **Topological Similarity** Lazaridou et al. (2018) is firsly proposed by Brighton and Kirby (2006) in the context of language emergence. The fundamental concept is that objects that are semantically similar should possess comparable messages. To calculate this measure, two sets of numbers are computed: (i) the Levenshtein distances (minimum edit distance) between every pair of messages for the objects, and (ii) the cosine similarity between every pair of representation vectors for the objects. Then the topographic similarity can be defined as the negative Spearman ρ correlation, since we are correlating distances with similarities. As a result, if similar objects have a substantial amount of shared message structure, such as common prefixes or suffixes, and dissimilar objects have little common message structure, the topographic similarity should be high.
- **Positional Disentanglement (*posdis*)** is an entropy-based measure introduced by Chaabouni et al. (2020). The idea is that for a language to be compositional given our inputs, each position of the message should only be informative about a single attribute of the object. Formally, if we assume that an object can be described by m attributes, and each of which can take on k values, then we can represent the object as a vector $\boldsymbol{x} = (a_1,a_2,...,a_m)$, where $a_i \in 1,2,...,k$. The mutual information between attribute i and word j can then be defined as: $\mathcal{I}(a_i,m_j) = \sum_{a_i}\sum_{m_j} \boldsymbol{P}(a_i,m_j)\log\left(\frac{\boldsymbol{P}(a_i,m_j)}{\boldsymbol{P}(a_i)\boldsymbol{P}(m_j)}\right)$, where $\boldsymbol{P}(a_i,m_j)$, $\boldsymbol{P}(a_i)$, and $\boldsymbol{P}(m_j)$ are the joint and marginal distributions estimated by counting in the toy referential games. Denote $b_j^1 = \arg\max \mathcal{I}(\cdot,m_j)$ as the attribute variable with the largest mutual information

with respect to m_j, and b_j^2 as the one with the second largest mutual information. The positional disentanglement score is then calculated as:

$$posdis = \frac{1}{l}\sum_{i=1}^{l}\frac{\mathcal{I}(b_i^1, m_i) - \mathcal{I}(b_i^2, m_i)}{\mathcal{H}(m_i)} \tag{2.2.5}$$

where $\mathcal{H}(m_i)$ is the entropy of word i in the message and l is the length of the message.

- **Tree Reconstruction Error (TRE)** was introduced by Andreas (2019a) as a means of evaluating the degree of compositionality in the representation space. Notably, this measure is not limited to evaluating the compositionality of the representation space alone; it can also be applied to the emergent language space. Unlike previous measures, TRE prespecifies the underlying compositional structure and build a TRE approximator function that has a computation graph reflecting the underlying structure. The approximation error obtained from this process can be used to test whether the language conforms to a pre-specified compositional grammar. Nevertheless, the prerequisite of knowing the grammar beforehand limits its practical usage.

Despite their widespread use, current metrics for evaluating the compositionality of emergent languages are still significantly flawed. Chaabouni et al. (2020) have demonstrated that while these models can develop a language that generalizes successfully to a held-out test set, they score low on the existing compositionality measures presented above, thereby leading some to posit that compositionality is not necessary for robust generalization (Andreas, 2019b; Kharitonov and Baroni, 2020).

However, this interpretation presents a significant puzzle: if the languages emerging from these models lack compositionality, how do they still exhibit the generalization capabilities that enable successful communication about unseen and novel examples? One potential explanation, as argued by Conklin and Smith (2023), is that the languages that emerge between networks are indeed compositional, but with a degree of variation that can obscure their compositionality from existing measures, which only assess surface-level regularity.

Therefore, more investigation are still needed to establish a proper measure for the compositionality for emergent languages.

2.2.3. Neural Iterated Learning (NIL)

In addition to devising metrics for measuring compositionality, researchers have also focused on developing priors that facilitate the emergence of compositional languages, such as input representations (Lazaridou et al., 2018) and channel capacity (Kottur et al., 2017). Among them, a strong result is discovered under the framework of *neural iterated learning (NIL)* (Ren et al., 2020), claiming that the ease of generational transmission encourages compositionality..

NIL is a deep learning extension to the Iterated Learning Model (ILM), which is a cultural evolutionary account of the origins of compositionality (Kirby, 2001, 2002). Figure 1 provides an overview of the model. The ILM assumes the existence of two forms of language: *I-language* and *E-language*. The former refers to the internal language knowledge, such as grammar, while the latter represents the external language, such as utterances in daily usage. In this process, parents generate the E-language from their I-language, which in turn, infants acquire as their own I-language via learning. This process of learning and transmission continues across generations, resulting in the evolution of language through observations of the utterances of other agents who learn the language in a similar way (Kirby et al., 2014).

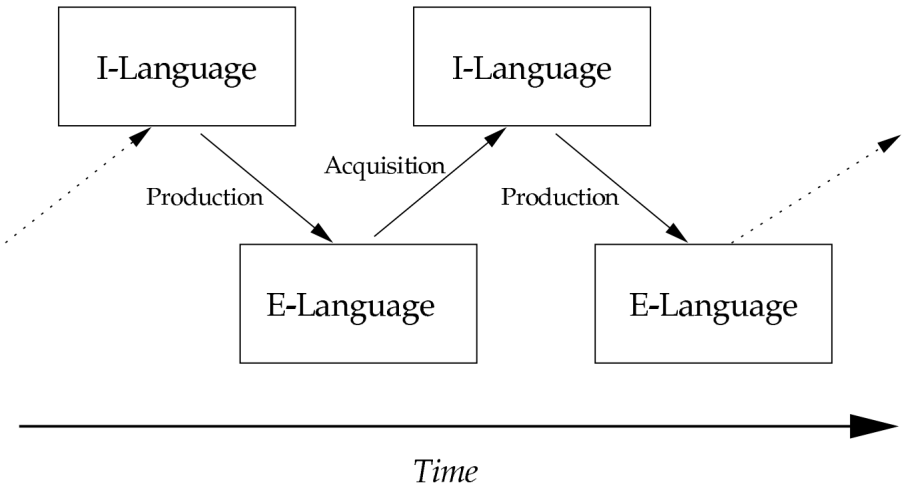


Fig. 1. Language transmission over time Kirby (2002). I-language is the internal language knowledge, while E-language is the external language like utterances.

One of the distinguishing characteristics of ILM is the presence of the "Learning Bottleneck". This phenomenon describes the challenge faced by language learners who must acquire a language system with infinite expressiveness, despite having access to only a limited amount of linguistic data. For instance, the vast number of possible sentences in a language cannot be explicitly enumerated by a parent or caregiver. As a result, linguistic structures, such as compositionality, is an adaptation to this selective pressure imposed by the Learning Bottleneck. Consequently, the languages that emerge from this process has great generalization capacity. Previous research has established the effectiveness of the Iterated Learning Model (ILM) as demonstrated by Kirby et al. (2014). However, these studies were primarily conducted with human participants, who tend to favor compositional languages due to the preference of the human brain for structured languages. As a result, it is unclear how the ILM can be applied to deep neural networks.

Recently, Ren et al. (2020) demonstrate that, like humans, neural networks also exhibit a preference for compositional languages. The authors found that neural networks can

acquire emergent languages with higher topological similarity scores faster, which they refer to as the “learning speed advantage". These observations were also made in other studies. For instance, Li and Bowling (2019) find that more structured emergent languages can be taught to the next generation more quickly. Similarly, Chaabouni et al. (2020) observe a strong positive correlation between the learning speed of new receivers and the degree of compositionality as well as generalization accuracy. These findings form the basis for the development of NIL. Therefore, to further leverage this learning speed advantage, Ren et al. (2020) propose a three-phase algorithm to simulate the ILM. The algorithm consists of the following phases:

- The first phase, referred to as the **Interaction Phase**, involves the standard training of both the sender and receiver to maximize the reward.
- The second phase, known as the **Transmitting Phase**, is characterized by generating a dataset $\mathcal{D}$ of object-message pairs through feeding all objects to the sender.
- In the third phase, the **Learning Phase**, both agents are reinitialized. A new sender is trained using supervised learning on the dataset $\mathcal{D}$, while the new receiver is trained with policy gradient, with the new sender fixed.
- Repeat the above phases.

The experiments results show that although the utilization of NIL exhibits negligible influence on game performance, it exhibit a distinct impact on the topological similarity score within the emergent language, provided with an adequate number of generations.

2.3. Self-Supervised Learning (SSL)

As our thesis revolves around learning language-like latent representations, we have explored our hypothesis within the realm of self-supervised learning (Chapter 6), which also involves unsupervised learning of latent representations. In the following sections, we present a literature review for the field.

2.3.1. The Dream of Unsupervised Representation Learning

Representation learning is about learning representations of the data that make it easier to extract useful information when building classifiers or other predictors (Bengio et al., 2013). While it is shown that with proper network architecture (e.g., convolutional neural networks) and large labeled dataset, we can achieve useful representations that can transfer well in the downstream tasks. It remains unclear how to obtain this useful representation without any labels. As a result, unsupervised representation learning is still a holy grail of deep learning.

Previous work on unsupervised representation learning mainly focus on generative models like Variational Autoencoders (VAE) (Kingma and Welling, 2013) and Generative Adversarial Networks (GAN) (Goodfellow et al., 2020). Generative models are a type of algorithm that

aims to model the data distribution $\mathcal{P}(X)$, which is derived from the traditional unsupervised learning perspective. However, despite improved image generation quality, the representations generated by these models are still less effective than their supervised counterparts.

To understand this limitation, it is crucial to recognize that obtaining representations that can be useful for downstream tasks like classification requires high-level semantic concepts that exhibit proper invariance and equivariance. Unfortunately, the representations obtained from generative models tend to focus on low-level details such as textures. Although these features may be desirable for generating accurate image pixels, they are not sufficiently high-level to support downstream tasks.

In contrast to generative models, self-supervised learning shifts its focus on auxiliary tasks that demand a deeper understanding of the data. By doing so, it moves beyond pixel space. Moreover, these techniques harness the vast amount of unlabeled data to construct annotations for their auxiliary tasks, rendering them the cutting-edge approach in unsupervised representation learning.

2.3.2. Pretext Tasks

In the early days, due to the lack of a principled method, researchers design ad-hoc pretext tasks for self-supervised learning. Solving these pretext tasks would require models to understand the underlying semantic concept within the data.

One such pretext task proposed by Gidaris et al. (2018) involves predicting the rotation angle of an image. The model is trained to classify images that have been rotated by 0, 90, 180, or 270 degrees, providing a 4-way classification task (Figure 2). By learning to recognize

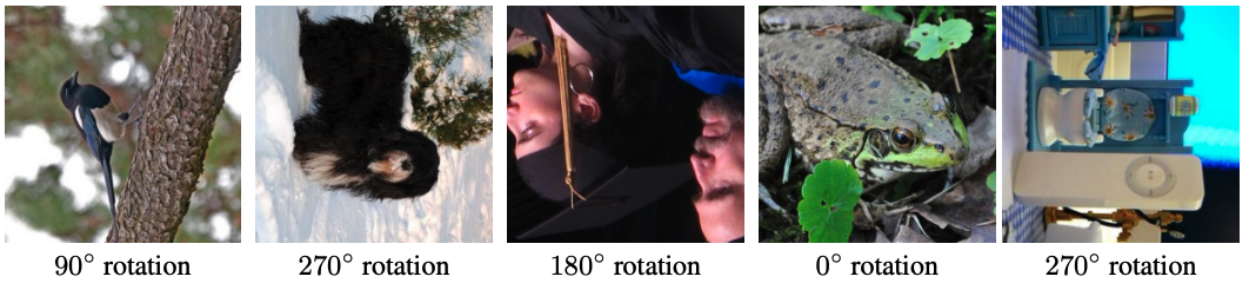


Fig. 2. Sample images rotated by random multiples of 90 degrees taken from Gidaris et al. (2018). The pretext task is to predict the orientation of the image.

the orientation of an image, the model is encouraged to learn features that are invariant to rotation, which can be useful for a wide range of downstream tasks. Noroozi and Favaro (2016) proposed a different pretext task, where the model is required to solve a jigsaw puzzle. The goal of this task is to encourage the model to learn and recognize the spatial relationships between image patches. By learning to reconstruct an image from shuffled patches, the model is forced to learn features that capture spatial information. Similarly, Doersch et al.

(2015) proposed a pretext task where the model is trained to predict the spatial configuration between two patches of an image. By learning to predict the relative position of image patches, the model is encouraged to learn features that capture spatial relationships between objects in an image. Misra and Maaten (2020) argued that semantic representations should be invariant to various pretext transformations and developed a technique called pretext-invariant representation learning (PIRL). Colorization (Larsson et al., 2017, 2016) is another powerful pretext task that has been proposed. Since a large number of colored images are already available, the model can be trained to predict the colors from grayscale images. Noroozi et al. (2017) introduced counting as a pretext task, which enforces constraints that the network should be able to recognize the number of visual primitives in an image. Other pretext tasks that have been investigated include in-painting (Pathak et al., 2016), video prediction (Wang and Gupta, 2015), and predicting the residual part of the image (Zhang et al., 2017). All of these pretext tasks provide different types of objectives that encourage the model to learn useful features that can be transferred to downstream tasks.

2.3.3. Contrastive Learning

Contrastive Learning is a family of SSL algorithms based on the principle of instance discrimination: Representations of different views of the same image should be closer than those of different images. To achieve this, contrastive approaches are constructed based on the idea of negative examples.

- **MoCo** (He et al., 2020b) is proposed to train a visual representation encoder. The core idea is to match an encoded query q with a set of encoded samples $\mathcal{K}$. It is assumed that there exists a single key $k_+ \in \mathcal{K}$ that matches with q. The contrastive loss function used in MoCo is based on the InfoNCE approach (Oord et al., 2018), and is given by:

$$\mathcal{L} = -\log \frac{\exp(q \cdot k_+/\tau)}{\sum_{k \in \mathcal{K}}^{K} \exp(q \cdot k/\tau)} \tag{2.3.1}$$

  Where τ is a temperature hyper-parameter that scales the logits to control the softmax sharpness. This loss function aims to maximize the similarity between q and k_+, while minimizing the similarity between q and the other samples in $\mathcal{K}$. Recent work also builds upon MoCo by using more data augmentation (Chen et al., 2020c) and using vision transformers architecture (Chen et al., 2021b).
- **SimCLR**, introduced by Chen et al. (2020a), presents a straightforward framework for contrastive learning that does not require special architecture or a memory bank. The approach involves randomly sampling N images and their corresponding augmented instances, resulting in a set of $2N$ images. Let $sim(\boldsymbol{u},\boldsymbol{v}) = \boldsymbol{u}^T\boldsymbol{v}/||\boldsymbol{u}||||\boldsymbol{v}||$ be the

cosine similarity. For each positive pair i,j, the loss is defined as follows:

$$l_{i,j} = -\log \frac{\exp(sim(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} 1_{k \neq i} \exp(sim(z_i, z_k)/\tau)} \tag{2.3.2}$$

where $1_{k \neq i} \in \{0, 1\}$ is an indicator function equals to 1 if and only if $k \neq i$. The final loss is computed over all positive pairs. Recent work also builds upon it by exploring deeper ResNet architectures and increasing the capacity of the projection head (Chen et al., 2020b).

Both the MoCo and SimCLR methods rely heavily on advanced data augmentation techniques, including cropping, resizing, and color distortion, to enhance the quality of their learned representations. This reliance on sophisticated data augmentation techniques becomes the main theme of self-supervised learning.

2.3.4. Beyond Contrastive Learning

Recent work has also shown that eliminating negative samples in contrastive learning is achievable. These frameworks place greater emphasis on the consistency between positive examples.

- **BYOL** (Grill et al., 2020) maintains an online network denoted as f_θ and a target network denoted as f_ϕ. In this framework, when an image is sampled, two views denoted as x and x' are obtained from the data augmentations. The representations of these views are obtained using the online network and target network, respectively $z = f_\theta(x)$ and $z' = f_\phi(x')$. In addition, the online network includes a predictor network q, which renders the architecture asymmetric between the online and target pipelines. This asymmetry has been argued to be a crucial factor contributing to the success of BYOL (Tian et al., 2021) Finally the loss is defined as mean squared error as follows:

  $$\mathcal{L} = 2 - 2 \cdot \frac{< q(z), z' >}{||q(z)||||z'||} \tag{2.3.3}$$

  During the optimization process, the stop-gradient operator is applied on the target network, and only the online network f_θ is updated with back-propagation. As with the MoCo framework, BYOL updates the target network f_ϕ with a slow-moving average of the online networks. Future work, e.g., DINO (Caron et al., 2021), further builds on this idea of self-distillation with a momentum target network by incorporating vision transformers.
- **SimSiam** further simplifying the above procedure by removing the momentum target network. Let the two views of the same image denoted as x_1 and x_2. An encoder f and a projection head h are used to process the two views. Let $p_1 = h(f(x_1))$ and $z_2 = f(x_2)$. The negative cosine similarity is then computed as $D(p_1, z_2) = -\frac{p_1}{||p_1||} \frac{z_2}{||z_2||}$.

Finally, a symmetric loss is defined as

$$L = \frac{1}{2}(D(p_1, sg(z_2)) + D(p_2, sg(z_1))) \tag{2.3.4}$$

where sg is a stop-gradient operator, and it is shown to prevent the representation from collapsing.

- **SwAV** (Caron et al., 2020) is among the series work on clustering-based SSL methods (Caron et al., 2018; Asano et al., 2019) that leverage clustering to facilitate representation learning. Unlike traditional SSL methods that compare image features, SwAV compares the cluster assignments or codes obtained from image views. Given two image features z_t and z_s from two different augmentations of the same image, the codes q_t and q_s are computed by matching these features to a set of K prototypes $\mathcal{C} = \{c_1,...,c_K\}$ with Sinkhorn-Knopp algorithm (Cuturi, 2013). Then the loss is computed as

$$L(z_t,z_s) = l(z_t, q_s) + l(z_s, q_t) \tag{2.3.5}$$

where $l(z_t, q_s) = -\sum_k q_s^k \log p_t^k$ is the cross-entropy loss between the code q_s and the probability p_t, which is computed by taking a softmax on the dot-product between z_t and all the prototypes in c. In the implementation, a multi-crop data augmentation strategy is leveraged to further increase the performance.

- **Barlow twins** (Zbontar et al., 2021) introduces a new loss function that aims to improve the similarity between the embeddings of positive examples, while reducing redundancy between the components of these vectors. Denote Z^A and Z^B as the batches of embeddings from two different views. A cross correlation matrix is computed along the batch dimension:

$$\mathcal{C}_{ij} = \frac{\sum_b Z^A_{b,i} Z^B_{b,j}}{\sqrt{\sum_b (Z^A_{b,i})^2}\sqrt{\sum_b (Z^B_{b,j})^2}} \tag{2.3.6}$$

The network is trained by making $\mathcal{C}$ closer to identity matrix. As a result, the loss function is

$$\mathcal{L} = \sum_i (1 - \mathcal{C}_{ii})^2 + \lambda \sum_i \sum_{j \neq i} C_{ij}^2 \tag{2.3.7}$$

where λ is a hyper-parameter trading off the importance between maintaining invariance (first term) and removing redundancy (second terms). Future work, e.g., VICReg (Bardes et al., 2021), builds upon Barlow twins by incorporating a weighted sum of the variance, invariance, and covariance.

2.3.5. Mask and Predict

Following the success of BERT (Devlin et al., 2018) and masked language modelling pretraining, recent work also try to adapt the idea of masked auto-encoders into computer vision along with the introduction of vision transformers (Dosovitskiy et al., 2020).

- **BEIT** (Bao et al., 2022) presents a self-supervised learning approach using a masked image modeling task. The authors tokenize the initial image by utilizing a pretrained dVAE(Ramesh et al., 2021) to produce visual tokens. Following this, they randomly mask certain patches of the image and input them into the backbone Transformer. The goal of pretraining is to recover the original visual tokens using the corrupted image patches. The authors demonstrate that the BEIT pretraining method can be interpreted as maximizing the evidence lower bound. They also show that the self-attention mechanism of self-supervised BEIT learns to distinguish semantic regions and object boundaries.
- **MAE** is a recent approach proposed by He et al. (2022) that simplifies the masked image modelling procedure by eliminating the need for a pretrained visual tokenizer. This is achieved by employing an asymmetric encoder-decoder architecture. Specifically, the encoder only processes the unmasked subset of patches and generates latent representations for each patch. In contrast, the decoder is a lightweight network that reconstructs the image patch from its corresponding latent representation and the masked tokens. The reconstruction is accomplished by predicting the mean pixel values of the patch. The authors also demonstrate that a higher masking rate, e.g., 80%, is beneficial which also makes the training significantly faster. The transfer performance in downstream also shows promising scaling behavior to larger transformer architectures.

Chapter 3

First Article: Countering Language Drift with Seeded Iterated Learning

Prologue

Article Details. **Countering Language Drift with Seeded Iterated Learning.** Yuchen Lu, Soumye Singhal, Florian Strub, Olivier Pietquin, Aaron Courville. This paper was published at ICML 2020.

Personal contributions. Aaron Courville proposed the problem of language drift, while I conceived the idea of applying iterated learning and conducted most of the experiments. Souyme Singhal helped run the experiment of the small scale toy dataset. I also drafted the most parts of the paper, with the assistance from all the other co-authors.

Discussion and Recent Developments. This project is inspired by the successful application of iterated learning for deep emergent communication (Ren et al., 2020). This can be viewed as one of the first attempt of bringing the techniques from emergent communications field to a more large-scale problem, beyond the toy signalling game.

Since its publication, several works have extended the proposed technique. Notably, Vani et al. (2021) propose to apply iterated learning to Visual Question Answering, and find that the iterated learning method can recover the ground truth layout in neural module networks. Rajeswar et al. (2022) propose to use iterated learning to resolve label ambiguity and to improve multi-label classification. Our successful application also inspired theoretical work on analyzing iterated learning (Jarvis et al., 2022).

Abstract

Pretraining on human corpus and then finetuning in a simulator has become a standard pipeline for training a goal-oriented dialogue agent. Nevertheless, as soon as the agents are finetuned to maximize task completion, they suffer from the so-called language drift phenomenon: they slowly lose syntactic and semantic properties of language as they only focus on solving the task. In this paper, we propose a generic approach to counter language drift called *Seeded iterated learning* (SIL). We periodically refine a pretrained student agent by imitating data sampled from a newly generated teacher agent. At each time step, the teacher is created by copying the student agent, before being finetuned to maximize task completion. SIL does not require external syntactic constraint nor semantic knowledge, making it a valuable task-agnostic finetuning protocol. We evaluate SIL in a toy-setting Lewis Game, and then scale it up to the translation game with natural language. In both settings, SIL helps counter language drift as well as it improves the task completion compared to baselines.

3.1. Introduction

Recently, neural language modeling methods have achieved a high level of performance on standard natural language processing tasks (Radford et al., 2019). Those agents are trained to capture the statistical properties of language by applying supervised learning techniques over large datasets (Bengio et al., 2003; Collobert et al., 2011). While such approaches correctly capture the syntax and semantic components of language, they give rise to inconsistent behaviors in goal-oriented language settings, such as question answering and other dialogue-based tasks (Gao et al., 2019). Conversational agents trained via traditional supervised methods tend to output uninformative utterances such as, for example, recommend generic locations while booking for a restaurant (Bordes et al., 2017). As models are optimized towards generating grammatically-valid sentences, they fail to correctly ground utterances to task goals (Strub et al., 2017; Lewis et al., 2017).

A natural follow-up consists in rewarding the agent to solve the actual language task, rather than solely training it to generate grammatically valid sentences. Ideally, such training would incorporate human interaction (Skantze and Hjalmarsson, 2010; Li et al., 2016a), but doing so quickly faces sample-complexity and reproducibility issues. As a consequence, agents are often trained by interacting with a second model to simulate the goal-oriented scenarios (Levin et al., 2000; Schatzmann et al., 2006; Lemon and Pietquin, 2012). In the recent literature, a common setting is to pretrain two neural models with supervised learning to acquire the language structure; then, at least one of the agents is finetuned to maximize

task-completion with either reinforcement learning, e.g., policy gradient (Williams, 1992), or Gumbel softmax straight-through estimator (Jang et al., 2017a; Maddison et al., 2017). This finetuning step has shown consistent improvement in dialogue games (Li et al., 2016b; Strub et al., 2017; Das et al., 2017), referential games (Havrylov and Titov, 2017; Yu et al., 2017) or instruction following (Fried et al., 2018).

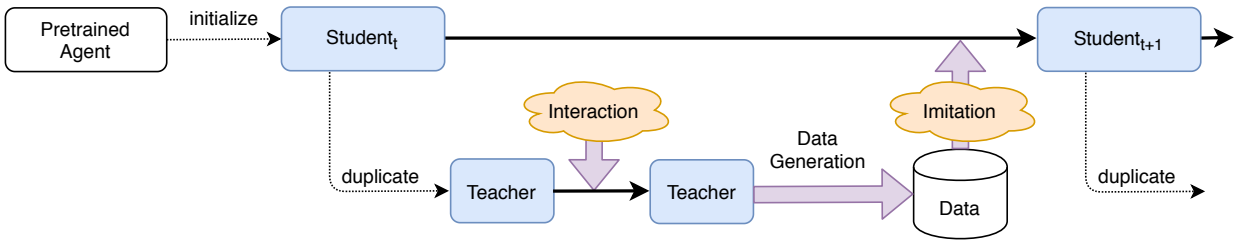


Fig. 3.3. Sketch of Seeded Iterated Learning. A **student** agent is iteratively refined using newly generated data from a **teacher** agent. At each iteration, a teacher agent is created on top of the student before being finetuned by interaction, e.g. maximizing a task completion-score. The teacher then generates a dataset with greedy sampling, which is then used to refine the student through supervised learning. Note that the interaction step involves interaction with another language agent.

Unfortunately, interactive learning gives rise to the *language drift* phenomenon. As the agents are solely optimizing for task completion, they have no incentive to preserve the initial language structure. They start drifting away from the pretrained language output by shaping a task-specific communication protocol. We thus observe a co-adaptation and overspecialization of the agent toward the task, resulting in significant changes to the agent's language distribution.

In practice, there are different forms of language drift (Lazaridou et al., 2020) including (i) structural drift: removing grammar redundancy (e.g. "is it a cat?" becomes "is cat?" (Strub et al., 2017)), (ii) semantic drift: altering word meaning (e.g. "an old teaching" means "an old man" (Lee et al., 2019)) or (iii) functional drift: the language results in unexpected actions (e.g. after agreeing on a deal, the agent performs another trade (Li et al., 2016b)). Thus, these agents perform poorly when paired with humans (Chattopadhyay et al., 2017; Zhu et al., 2017; Lazaridou et al., 2020).

In this paper, we introduce the *Seeded Iterated Learning* (SIL) protocol to counter language drift. This process is directly inspired by the iterated learning procedure to model the emergence and evolution of language structure (Kirby, 2001; Kirby et al., 2014). SIL does not require human knowledge intervention, it is task-agnostic, and it preserves natural language properties while improving task objectives.

As illustrated in Figure 3.3, SIL starts from a pretrained agent that instantiates a first generation of *student* agent. The teacher agent starts as a duplicate of the student agent and then goes through a short period of interactive training. Then the teacher generates a training dataset by performing the task over multiple scenarios. Finally, the student is finetuned – via supervised learning – to imitate the teacher data, producing the student for next generation, and this process repeats. As further detailed in Section 3.3, the imitation learning step induces a bias toward preserving the well-structured language, while discarding the emergence of specialized and inconsistent language structure (Kirby, 2001). Finally, SIL successfully interleaves interactive and supervised learning agents to improves task completions while preserving language properties.

Our contribution In this work, we propose Seeded Iterated Learning and empirically demonstrate its effectiveness in countering language drift. More precisely,

(1) We study core Seeded Iterated Learning properties on the one-turn Sender-Receiver version of the Lewis Game.
(2) We demonstrate the practical viability of Seeded Iterated Learning on the French-German translation game that was specifically designed to assess natural language drift (Lee et al., 2019). We observe that our method preserves both the semantic and syntactic structure of language, successfully countering language drift while outperforming strong baseline methods.
(3) We provide empirical evidence towards understanding the algorithm mechanisms[1].

3.2. Related Works

3.2.1. Countering Langauge Drift

The recent literature on countering language drift includes a few distinct groups of methods. The first group requires an external labeled dataset, that can be used for visual grounding (i.e. aligning language with visual cues (Lee et al., 2019)), reward shaping (i.e. incorporating a language metric in the task success score (Li et al., 2016b)) or KL minimization (Havrylov and Titov, 2017). Yet, these methods depends on the existence of an extra supervision signal and ad-hoc reward engineering, making them less suitable for general tasks. The second group are the population-based methods, which enforces social grounding through a population of agents, preventing them to stray away from the common language. The third group of methods involve an alternation between an interactive training phase and a supervised training phase on a pretraining dataset (Wei et al., 2018; Lazaridou et al., 2016). This approach has been formalized in Gupta et al. (2019a) as *Supervised-2-selfPlay* (S2P).

[1]Code for Lewis game and translation game

Empirically, the S2P approach has shown impressive resistance to language drift and, being relatively task-agnostic, it can be considered a strong baseline for SIL. However, the success of S2P is highly dependent on the quality of the fixed training dataset, which in practice may be noisy, small, and only tangentially related to the task. In comparison, SIL is less dependent on an initial training dataset since we keep generating new training samples from the teacher throughout training.

3.2.2. Iterated Learning in Emergent Communication

Iterated learning was initially proposed in the field of cognitive science to explore the fundamental mechanisms of language evolution and the persistence of language structure across human generations (Kirby, 2001, 2002). In particular, Kirby et al. (2014) showed that iterated learning consistently turns unstructured proto-language into stable compositional communication protocols in both mathematical modelling and human experiments. Recent works (Guo et al., 2019; Li and Bowling, 2019; Ren et al., 2020; Cogswell et al., 2019; Dagan et al., 2020) have extended iterated learning into deep neural networks. They show that the inductive learning bottleneck during the imitation learning phase encourages compositionality in the emerged language. Our contribution differs from previous work in this area as we seek to *preserve* the structure of an existing language rather than *emerge* a new structured language.

3.2.3. Lifelong Learning

One of the key problem for neural networks is the problem of catastrophic forgetting (McCloskey and Cohen, 1989). We argue that the problem of language drift can also be viewed as a problem of lifelong learning, since the agent needs to keep the knowledge about language while acquiring new knowledge on using language to solve the task. From this perspective, S2P can be viewed as a method of task rehearsal strategy (Silver and Mercer, 2002) for lifelong learning. The success of iterated learning for language drift could motivate the development of similar methods in countering catastrophic forgetting.

3.2.4. Self-training

Self-training augments the original labeled dataset with unlabeled data paired with the model's *own* prediction (He et al., 2020a). After noisy self-training, the student may out-perform the teacher in fields like conditional text generation (He et al., 2020a), image classification (Xie et al., 2019) and unsupervised machine translation (Lample et al., 2018).

This process is similar to the imitation learning phase of SIL except that we only use the self labeled data.

3.3. Method

3.3.1. Learning Bottleneck in Iterated Learning

The core component of iterated learning is the existence of the *learning bottleneck* (Kirby, 2001): a newly initialized student only acquires the language from a *limited number of examples* generated by the teacher. This bottleneck implicitly favors any structural property of the language that can be exploited by the learner to generalize, such as compositionality.

Yet, Kirby (2001) assumes that the student to be a perfect inductive learner that can achieve systematic generalization (Bahdanau et al., 2018). Neural networks are still far from achieving such goal. Instead of using a limited amount of data as suggested, we propose to use a regularization technique, like *limiting the number of imitation steps*, to reduce the ability of the student network to memorize the teacher's data, effectively simulating the learning bottleneck.

3.3.2. Seeded Iterated Learning

As previously mentioned, Seeded Iterated Learning (SIL) is an extension of Iterated Learning that aims at preserving an initial language distribution while finetuning the agent to maximize task-score. SIL iteratively refines a pretrained agent, namely the *student*. The *teacher* agent is initially a duplicate of the student agent, and it undergoes an interactive training phase to maximize task score. Then the teacher generates a new training dataset by providing pseudo-labels, and the student performs imitation learning via supervised learning on this synthetic dataset. The final result of the imitation learning will be next student. We repeat the process until the task score converges. The full pipeline is illustrated in Figure 3.3. Methodologically, the key modification of SIL from the original iterated learning framework is the use of the student agent to seed the imitation learning rather than using a randomly initialized model or a pretrained model. Our motivation is to ensure a smooth transition during the imitation learning and to retain the task progress.

Although this paper focuses on countering language drift, we emphasize that SIL is task-agnostic and can be extended to other machine learning settings.

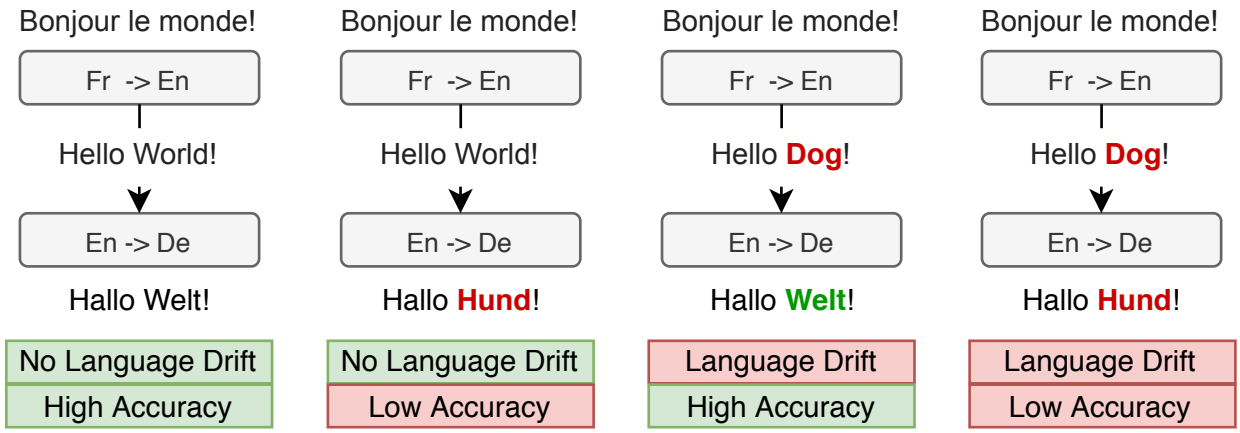


Fig. 3.4. In the translation game, the sentence is translated into English then into German. The second and fourth cases are regular failures, while the third case reveals a form of agent co-adaptation.

3.4. The Sender-Receiver Framework

We here introduce the experimental framework we use to study the impact of SIL on language drift. We first introduce the Sender-Receiver (S/R) Game to assess language learning and then detail the instantiation of SIL for this setting.

3.4.1. Sender-Receiver Games

S/R Games are cooperative two-player language games in which the first player, the *sender*, must communicate its knowledge to the second player, the *receiver*, to solve an arbitrary given task. The game can be multi-turn with feedback messages, or single-turn where the sender outputs a single utterance. In this paper, we focus on the single-turn scenario as it eases the language analysis. Yet, our approach may be generalized to multi-turn scenarios. Figures 3.4 and 3.5 show two instances of the S/R games studied here: the Translation game (Lee et al., 2019) and the Lewis game (Kottur et al., 2017).

Formally, a single-turn S/R game is defined as a 4-tuple $\mathcal{G} = (\mathcal{O}, \mathcal{M}, \mathcal{A}, R)$. At the beginning of each episode, an observation (or scenario) $\boldsymbol{o} \in \mathcal{O}$ is sampled. Then, the sender s emits a message $\boldsymbol{m} = s(\boldsymbol{o}) \in \mathcal{M}$, where the message can be a sequence of words $m = [w]_{t=1}^{T}$ from a vocabulary $\mathcal{V}$. The receiver r gets the message and performs an action $\boldsymbol{a} = r(\boldsymbol{m}) \in \mathcal{A}$. Finally, both agents receive the same reward $R(\boldsymbol{o}, \boldsymbol{a})$ which they aim to maximize.

Algorithm 1 Seeded Iterate Learning for S/R Games

Require: Pretrained parameters of sender $\boldsymbol{\theta}$ and receiver $\boldsymbol{\phi}$.
Require: Training scenarios $\mathcal{O}_{train}$ {or scenario generator}
Copy $\boldsymbol{\theta}$,$\boldsymbol{\phi}$ to $\boldsymbol{\theta}^S, \boldsymbol{\phi}^S$ {*Prepare Iterated Learning*}
repeat
  Copy $\boldsymbol{\theta}^S$,$\boldsymbol{\phi}^S$ to $\boldsymbol{\theta}^T, \boldsymbol{\phi}^T$ {*Initialize Teacher*}
  for $i = 1$ **to** k_1 **do**
    Sample a batch $\boldsymbol{o} \in \mathcal{O}_{train}$
    Get $\boldsymbol{m} = s(\boldsymbol{o}; \boldsymbol{\theta}^T)$ and $\boldsymbol{a} = r(\boldsymbol{m}; \boldsymbol{\phi}^T)$ to have $R(\boldsymbol{o},\boldsymbol{a})$
    Update $\boldsymbol{\theta}^T$ and $\boldsymbol{\phi}^T$ to maximize R
  end for {*Finish Interactive Learning*}
  for $i = 1$ **to** k_2 **do**
    Sample a batch of $\boldsymbol{o} \in \mathcal{O}_{train}$
    Sample $\boldsymbol{m} = s(\boldsymbol{o}; \boldsymbol{\theta}^T)$
    Update $\boldsymbol{\theta}^S$ with supervised learning on $(\boldsymbol{o}, \boldsymbol{m})$
  end for {*Finish Sender Imitation*}
  for $i = 1$ **to** k_2' **do**
    Sample a batch of $\boldsymbol{o} \in \mathcal{O}_{train}$
    Get $\boldsymbol{m} = s(\boldsymbol{o}; \boldsymbol{\theta}^S)$ and $\boldsymbol{a} = r(\boldsymbol{m}; \boldsymbol{\phi}^S)$ to have $R(\boldsymbol{o},\boldsymbol{a})$
    Update $\boldsymbol{\phi}^S$ to maximize R
  end for {*Finish Receiver Finetuning*}
until Convergence or maximum steps reached

3.4.2. SIL For S/R Game

We consider two parametric models, the sender $s(.; \boldsymbol{\theta})$ and the receiver $r(.; \boldsymbol{\phi})$. Following the SIL pipeline, we use the uppercase script S and T to respectively denote the parameters of the student and teacher. For instance, $r(.; \boldsymbol{\phi}^T)$ refers to the teacher receiver. We also assume that we have a set of scenarios $\mathcal{O}_{train}$ that are fixed or generated on the fly. We detail the SIL protocol for single-turn S/R games in Algorithm 1.

In one-turn S/R games, the language is only emitted by the sender while the receiver's role is to interpret the sender's message and use it to perform the remaining task. With this in mind, we train the sender through the SIL pipeline as defined in Section 3.3.2 (i.e., interaction, generation, imitation), while we train the receiver to quickly adapt to the new sender's language distribution with a goal of stabilizing training (Ren et al., 2020). First, we jointly train $s(.; \boldsymbol{\phi}^T)$ and $r(.; \boldsymbol{\phi}^T)$ during the SIL interactive learning phase. Second, the sender student imitates the labels generated by $s(.; \boldsymbol{\phi}^T)$ through greedy sampling. Third, the receiver student is trained by maximizing the task score $R(r(\boldsymbol{m}; \boldsymbol{\phi}^S), \boldsymbol{o})$ where $\boldsymbol{m} = s(\boldsymbol{o}; \boldsymbol{\theta}^S)$ and $\boldsymbol{o} \in \mathcal{O}_{train}$. In other words, we finetune the receiver with interactive learning while freezing the new sender parameters. SIL has three training hyperparameters: (i) k_1, the number of interactive learning steps that are performed to obtain the teacher agents, (ii) k_2, the number of sender imitation steps, (iii) k_2', the number of interactive steps that are

performed to finetune the receiver with the new sender. Unless stated otherwise, we define $k_2 = k_2'$.

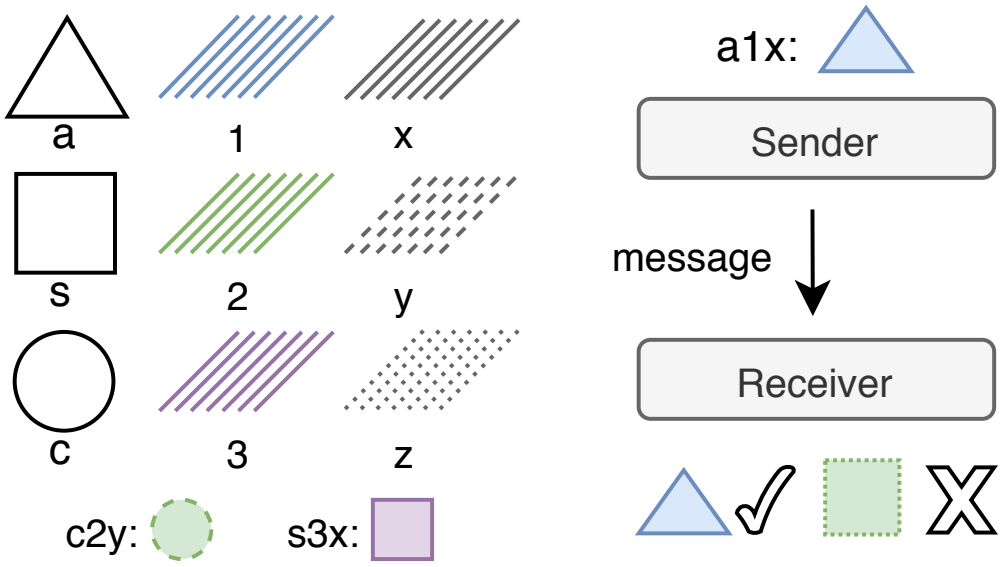


Fig. 3.5. Lewis game. Given the input object, the sender emits a compositional message that is parsed by the receiver to retrieve object properties. In the language drift setting, both models are trained toward identity map while solving the reconstruction task.

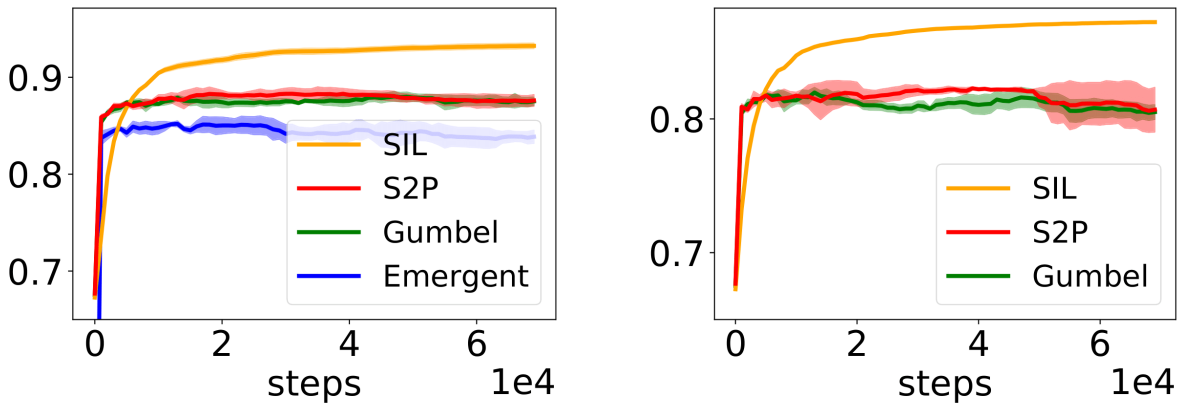
3.4.3. Gumbel Straight-Through Estimator

In the one-turn S/R game, the task success can generally be described as a differentiable loss such as cross-entropy to update the receiver parameters. Therefore, we here assume that the receiver r can maximize task-completion by minimizing classification or regression errors. To estimate the task loss gradient with respect to the sender s parameters, the receiver gradient can be further backpropagated using the Gumbel softmax straight-through estimator (GSTE) (Jang et al., 2017a; Maddison et al., 2017). Hence, the sender parameters are directly optimized toward task loss. Given a sequential message $\boldsymbol{m} = [w]_{t=1}^{T}$, we define $\boldsymbol{y}_t$ as follows:

$$\boldsymbol{y}_t = \text{softmax}\big((\log s(w|\boldsymbol{o}, w_{t-1}, \cdots, w_0; \theta) + g_t)/\tau\big) \tag{3.4.1}$$

where $s(w|\boldsymbol{o}, w_{t-1}, \cdots, w_0)$ is the categorical probability of next word given the sender observation o and previously generated tokens, $g_t \sim \text{Gumbel}(0,1)$ and τ is the Gumbel temperature that levels exploration. When not stated otherwise, we set $\tau = 1$. Finally, we sample the next word by taking $w_t = \arg\max \boldsymbol{y}_t$ before using the straight-through gradient estimator to approximate the sender gradient:

$$\frac{\partial R}{\partial \theta} = \frac{\partial R}{\partial w_t}\frac{\partial w_t}{\partial y_t}\frac{\partial y_t}{\partial \theta} \approx \frac{\partial R}{\partial w_t}\frac{\partial y_t}{\partial \theta}. \tag{3.4.2}$$



(a) Task Score **(b)** Sender Language Score

Fig. 3.6. Task Score and Language Score for SIL($\tau = 10$) vs baselines ($\tau = 1$). SIL clearly outperforms the baselines. For SIL: $k_1 = 1000, k_2 = k_2' = 400$. The emergent language score is close to zero. All results are averaged over four seeds.

SIL can be applied with RL methods when dealing with non-differential reward metrics (Lee et al., 2019), however RL has high gradient variance and we want to GSTE as a start. Since GSTE only optimizes for task completion, language drift will also appear.

3.5. Building Intuition: The Lewis Game

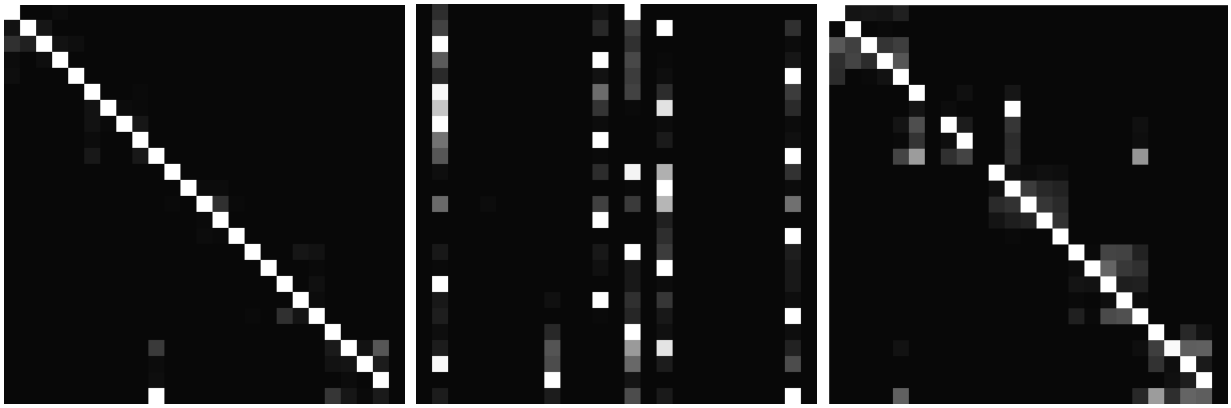
In this section, we explore a toy-referential game based on the Lewis Game (Lewis, 1969) to have a fine-grained analysis of language drift while exploring the impact of SIL.

3.5.1. Experimental Setting

We summarize the Lewis game instantiation described in Gupta et al. (2019a) to study language drift, and we illustrate it in Figure 3.5. First, the sender observes an object o with p properties and each property has t possible values: $o[i] \in [1 \dots t]$ for $i \in [1 \dots p]$. The sender then sends a message m of length p from the vocabulary of size $p \times t$, equal to the number of property values. Our predefined language $\mathcal{L}$ uniquely map each property value to each word, and the message is defined as $\mathcal{L}(o) = [o_1, t + o_2, ..., (p-1)t + o_p]$.

We study whether this language mapping is preserved during S/R training.

The sender and receiver are modeled by two-layer feed-forward networks. In our task, we use $p = t = 5$ with a total of 3125 unique objects. We split this set of objects into three parts: the first split(pre-train) is labeled with correct messages to pre-train the initial agents.



(a) SIL **(b)** Emergent **(c)** Gumbel

Fig. 3.7. Comparison of sender's map, where the columns are words and rows are property values. Emergent communication uses the same word to refer to multiple property values. A perfect mapped language would be the identity matrix.

The second split is used for the training scenarios. The third split is held out (HO) for final evaluation. The dataset split and hyper-parameters can be found in the Appendix A.2.1.

We use two main metrics to monitor our training: *Sender Language Score* (LS) and *Task Score* (TS). For the sender language score, we enumerate the held-out objects and compare the generated messages with the ground-truth language on a per token basis. For task accuracy, we compare the reconstructed object vs. the ground-truth object for each property. Formally, we have:

$$LS = \frac{1}{|\mathcal{O}_{HO}|p} \sum_{o \in \mathcal{O}_{HO}} \sum_{l=1}^{p} [\mathcal{L}(o)[l] == s(o)[l]], \tag{3.5.1}$$

$$TS = \frac{1}{|\mathcal{O}_{HO}|p} \sum_{o \in \mathcal{O}_{HO}} \sum_{l=1}^{p} [o[l] == r(s(o))[l]]. \tag{3.5.2}$$

where $[\cdot]$ is the Iverson bracket.

In our experiments, we compare SIL with different baselines. All methods are initialized with the same pretrained model unless stated otherwise. The *Gumbel* baselines are finetuned with GSTE during interaction. These correspond to naive application of interactive training and are expected to exhibit language drift.

Emergent is a random initializion trained with GSTE. *S2P* indicates that the agents are trained with Supervised-2-selfPlay. Our *S2P* is realized by using a weighted sum of the losses at each step: $L_{S2P} = L_{Gumbel} + \alpha L_{supervised}$ where $L_{supervised}$ is the loss on the pre-train dataset and α is a hyperparameter with a default value of 1 as detailed in (Lazaridou et al., 2016, 2020).

We present the main results for the Lewis game in Figure 3.6. For each method we used optimal hyperparameters namely $\tau = 10$ for SIL and $\tau = 1$ for rest. We also observed that

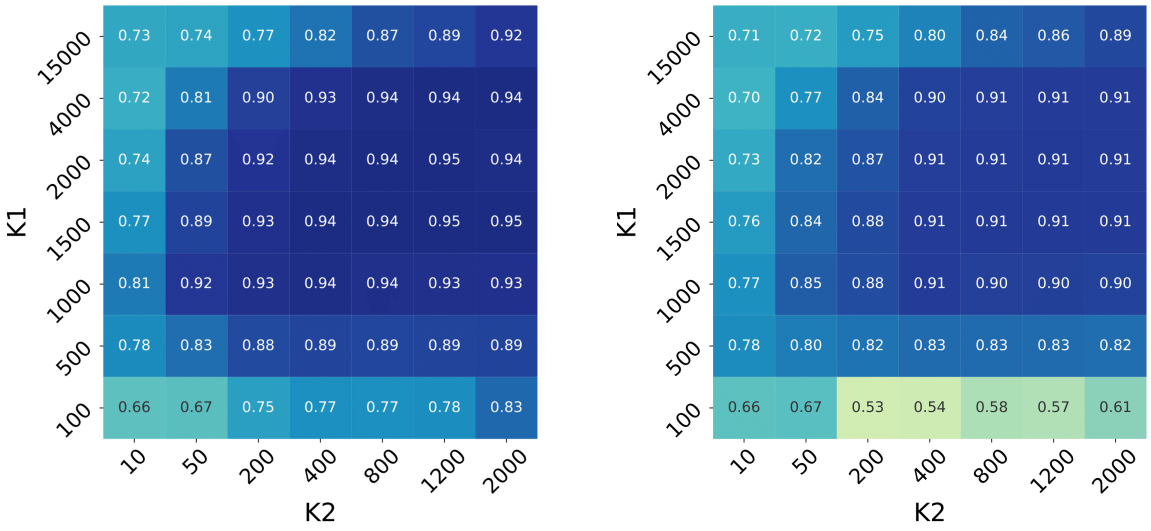
SIL outperforms the baselines for any τ. Additional results in Appendix A.2 (Figures A.26 & A.27).

The pretrained agent has an initial task score and language score of around 65%, showing an imperfect language mapping while allowing room for task improvement. Both Gumbel and S2P are able to increase the task and language score on the held-out dataset. For both baselines, the final task score is higher than the language score. This means that some objects are reconstructed successfully with incorrect messages, suggesting language drift has occurred.

Note that, for S2P, there is some instability of the language score at the end of training. We hypothesize that it could be because our pretrained dataset in this toy setting is too small, and as a result, S2P overfits that small dataset. Emergent communication has a sender language score close to zero, which is expected. However, it is interesting to find that emergent communication has slightly lower held-out task score than Gumbel, suggesting that starting from pretrained model provides some prior for the model to generalize better. Finally, we observe that SIL achieves a significantly higher task score and sender language score, outperforming the other baselines. A high language score also shows that the sender leverages the initial language structure rather than merely re-inventing a new language, countering language drift in this synthetic experiment.

To better visualize the underlying language drift in this settings, we display the sender's map from property values to words in Figure 3.7. We observe that the freely emerged language results in re-using the same words for different property values. If the method has a higher language score, the resulting map is closer to the identity matrix.

We perform a hyper-parameter sweep for the Lewis Game in Figure 3.8 over the core SIL parameters, k_1 and k_2, which are, respectively, the length of interactive and imitation training phase. We simply set $k_2' = k_2$ since in a toy setting the receiver can always adjust to the sender quickly. We find that for each k_2, the best k_1 is in the middle. This is expected since a small k_1 would let the imitation phase constantly disrupt the normal interactive learning, while a large k_1 would entail an already drifted teacher. We see that k_2 must be high enough to successfully transfer teacher distributions to the student. However, when a extremely large k_2 is set, we do not observe the expected performance drop predicted by the learning bottleneck: The overfitting of the student to the teacher should reduce SIL's resistance to language drift. To resolve this dilemma, we slightly modify our imitation learning process. Instead of doing supervised learning on the samples from teachers, we explicitly let student imitate the complete teacher distribution by minimizing $KL(s(;\theta^T)||s(;\theta^S))$. The result is in Figure 3.9, and we can see that increasing k_2 now leads to a loss of performance, which confirms our hypotheses. In conclusion, SIL has good performance in a (large) valley



(a) Task Score **(b)** Language Score

Fig. 3.8. Sweep over length of interactive learning phase k_1 and length of imitation phase k_2 on the Lewis game (darker is higher). Low or high k_1 result in poor task and language score. Similarly, low k_2 induces poor results while high k_2 do not reduce performance as one would expect.

of parameters, and a proper imitation learning process is also crucial for constructing the learning bottleneck.

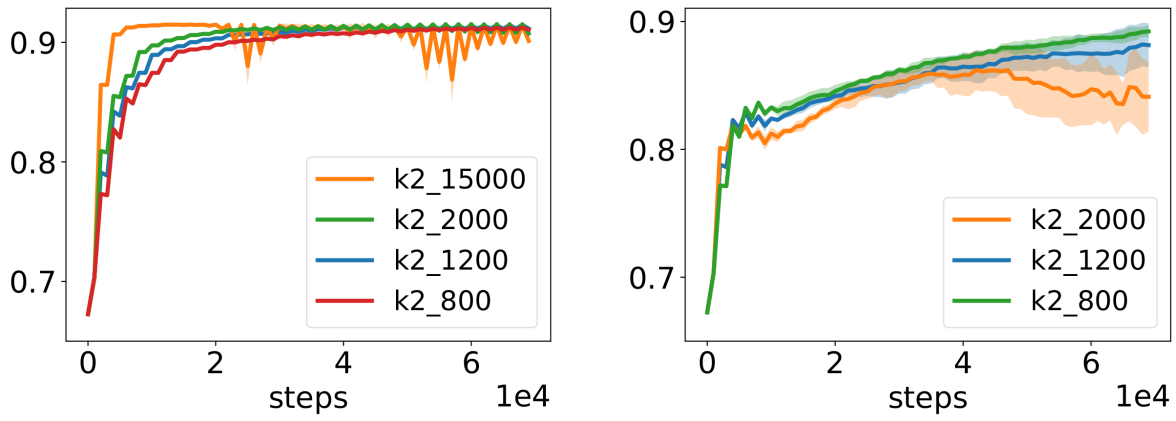
3.6. Experiments: The Translation Game

Although being insightful, the Lewis game is missing some core language properties, e.g., word ambiguity or unrealistic word distribution etc. As it relies on a basic finite language, it would be premature to draw too many conclusions from this simple setting (Hayes, 1988). In this section, we present a larger scale application of SIL in a natural language setting by exploring the translation game (Lee et al., 2019).

3.6.1. Experimental Setting

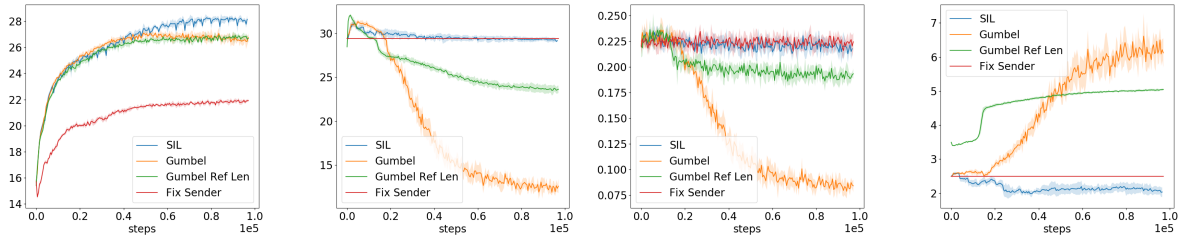
The translation game is a S/R game where two agents translate a text from a source language, French (FR), to a target language, German (De), through a pivot language, English (En). This framework allows the evaluation of the English language evolution through translation metrics while optimizing for the Fr→De translation task, making it a perfect fit for our language drift study.

The translation agents are sequence-to-sequence models with gated recurrent units (Cho et al., 2014) and attention (Bahdanau et al., 2015). First, they are independently pretrained



(a) argmax **(b)** KL Minimization

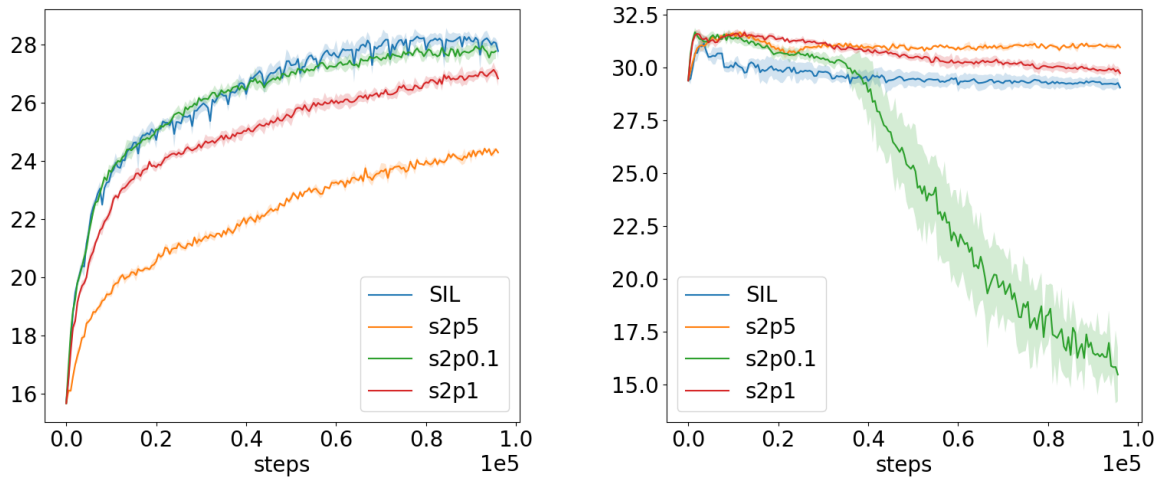
Fig. 3.9. Language score for different k_2 by imitating greedy sampling with cross-entropy (Left) vs distilling the teacher distribution with KL minimization (Right). As distillation relaxes the learning bottleneck, we observe a drop in language score with overfitting when the student imitation learning length increases.



(a) BLEU De **(b)** BLEU En **(c)** R1 **(d)** NLL

Fig. 3.10. The task score and the language score of NIL, S2P, and Gumbel baselines. Fix Sender indicates the maximum performance the sender may achieve without agent co-adaptation. We observe that Gumbel language start drifting when the task score increase. Gumbel Ref Len artificially limits the English message length, which caps the drift. Finally, SIL manages to both increase language and task score

on the IWSLT dataset (Cettolo et al., 2012) to learn the initial language distribution. The agents are then finetuned with interactive learning by sampling new translation scenarios from the Multi30k dataset (Elliott et al., 2016), which contains 30k images with the same caption translated in French, English, and German. Generally, we follow the experimental setting of Lee et al. (2019) for model architecture, dataset, and pre-processing, which we describe in Appendix A.3.2 for completeness. However, in our experiment, we use GSTE to optimize the sender, whereas Lee et al. (2019) rely on policy gradient methods to directly maximize the task score.



(a) BLEU De (Task Score)

(b) BLEU En

Fig. 3.11. S2P sweep over imitation loss weight vs. interactive loss. S2P displays a trade-off between a high task score, which requires a low imitation weight, and high language score, which requires high imitation weight. SIL appears less susceptible to a tradeoff between these metrics

3.6.2. Evaluation metrics

We monitor our task score with *BLEU(De)* (Papineni et al., 2002), it estimates the quality of the Fr→De translation by comparing the translated German sentences to the ground truth German. We then measure the sender language score with three metrics. First, we evaluate the overall language drift with the *BLEU(En)* score from the ground truth English captions. As the BLEU score controls the alignment between intermediate English messages and the French input texts, it captures basic syntactic and semantic language variations. Second, we evaluate the structural drift with the negative log-likelihood (*NLL*) of the generated English under a pretrained language model. Third, we evaluate the semantic drift by computing the image retrieval accuracy ($R1$) with a pretrained image ranker; the model fetches the ground truth image given 19 distractors and generated English. The language and image ranker models are further detailed in Appendix A.3.3.

3.6.3. Results

We show our main results in Figure 3.10, and a full summary in Table A.7 in Appendix A.3. Runs are averaged over five seeds and shaded areas are one standard deviation. The x-axis shows the number of interactive learning steps.

After pretraining our language agents on the IWSLT corpus, we obtain the single-agent BLEU score of 29.39 for Fr→En and 20.12 for En→De on the Multi30k captions. When combining the two agents, the Fr→De task score drops to 15.7, showing a compounding error in the translation pipeline. We thus aim to overcome this misalignment between translation agents through interactive learning while preserving an intermediate fluent English language.

As a first step, we freeze the sender to evaluate the maximum task score without agent co-adaptation. The Fix Sender then improves the task score by 5.3 BLEU(De) while artificially maintaining the language score constant. As we latter achieve a higher task score with Gumbel, it shows that merely fixing the sender would greatly hurt the overall task performance.

We observe that the Gumbel agent improves the task score by 11.32 BLEU(De) points but the language score collapse by 10.2 BLEU(En) points, clearly showing language drift while the two agents co-adapt to solve the translation game. Lee et al. (2019) also constrain the English message length to not exceed the French input caption length, as they observe that language drift often entails long messages. Yet, this strong inductive bias only slows down language drift, and the language score still falls by 6.0 BLEU(En) points. Finally, SIL improves the task score by 12.6 BLEU(De) while preserving the language score of the pretrained model. Thus, SIL successfully counters language drift in the translation game while optimizing for task-completion.

3.6.4. S2P vs SIL

We compare the S2P and SIL learning dynamics in Figure 3.11 and Figure A.29 in Appendix A.3. S2P balances the supervised and interactive losses by setting a weight α for the imitation loss (Lazaridou et al., 2016). First, we observe that a low α value, i.e, 0.1, improves the task score by 11.8 BLEU(De), matching SIL performances, but the language score diverges. We thus respectively increase α to 1, and 5, which stops the language drift, and even outperforms SIL language score by 1.2 BLEU(En) points. However, this language stabilization also respectively lowers the task score by 0.9 BLEU(De) and 3.6 BLEU(De) compared to SIL. In other words, S2P has an inherent trade-off between task score (with low α), and language score (with high α), whereas SIL consistently excels on both task and language scores. We assume that S2P is inherently constrained by the initial training dataset.

3.6.5. Syntactic and Semantic Drifts

As described in Section 3.6.1, we attempt to decompose the Language Drift into syntactic drifts, by computing language likelihood (NLL), and semantic drifts, by aligning images

	SIL successfully prevent language drift	*SIL can remain close to the valid pretrained models*
Human	**two men, one in blue and one in red, compete in a boxing match.**	**there are construction workers working hard on a project**
Pretrain	two men, one in blue and the other in red, fight in a headaching game	there are workers working hard work on a project.
Gumbel	two men one of one in blue and the other in red cfighting in a acacgame.....	there are construction working hard on a project .....
S2P	two men, one in blue and the other in red, fighting in a kind of a kind.	there are workers working hard working on a project ..
SIL	two men, one in blue and the other in red, fighting in a game.	there are workers working hard on a project .
	SIL partially recovers the sentence without drifting	*SIL/S2P still drift when facing rare word occurrences (shaped lollipop)*
Human	**a group of friends lay sprawled out on the floor enjoying their time together.**	**a closeup of a child's face eating a blue , heart shaped lollipop.**
Pretrain	a group of friends on the floor of fun together.	a big one 's face plan a blue box.
Gumbel	a group of defriends comadeof on the floor together of of of of of together.....	a big face of a child eating a blue th-acof of of of chearts.....
S2P	a group of friends of their commodities on the floor of fun together.	a big face plan of eating a blue of the kind of hearts.
SIL	a group of friends that are going on the floor together.	a big plan of a child eating a blue datadof the datadof the datadof the data@@

Table 3.1. Selected generated English captions. Vanilla Gumbel drifts by losing grammatical structure, repeating patches of words, and inject noisy words. Both S2P and SIL counter language drift by generating approximately correct and understandable sentences. However, they become unstable when dealing with rare word occurrences.

and generated captions ($R1$). In Figure 3.10, we observe a clear correlation between those two metrics and a drop in the language BLEU(En) score. For instance, Vanilla-Gumbel simultaneously diverges on these three scores, while the sequence length constraint caps the drifts. We observe that SIL does not improve language semantics, i.e., $R1$ remains constant during training, whereas it produces more likely sentences as the NLL is improved by 11%. Yet, S2P preserves slightly better semantic drift, but its language likelihood does not improve as the agent stays close to the initial distribution.

3.6.6. SIL Mechanisms

We here verify the initial motivations behind SIL by examining the impact of the learning bottleneck in Figure 3.12 and the structure-preserving abilities of SIL in Figure 3.13.

As motivated in Section 3.3.2, each imitation phase in the SIL aims to filtering-out emergent unstructured language by generating an intermediate dataset to train the student. To verify this hypothesis, we examine the change of negative language likelihood (NLL) from the teacher to the student after imitation. We observe that after imitation, the student consistently improves the language likelihood of its teacher, indicating a more regular language production induced by the imitation step. In another experiment, we stop the iterated learning loop after 20k, 40k and 60k steps and continue with standard interactive training. We observe that the agent's language score starts dropping dramatically as soon as we stop SIL while

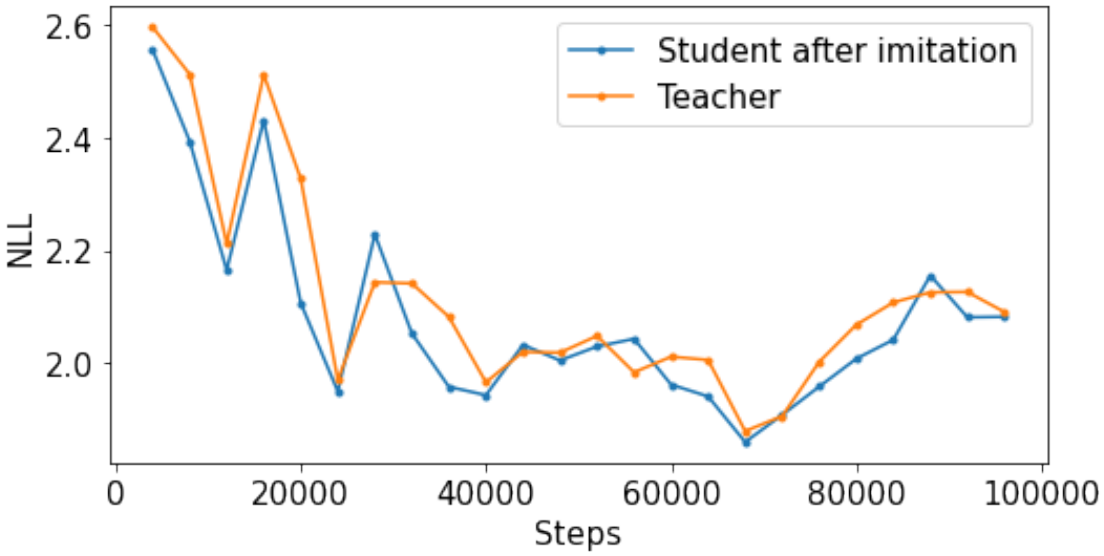


Fig. 3.12. *NLL* of the teacher and the student after imitation learning phase. In the majority of iterations, the student after imitation obtains a lower NLL than the teacher, after supervised training on the teacher's generated data.

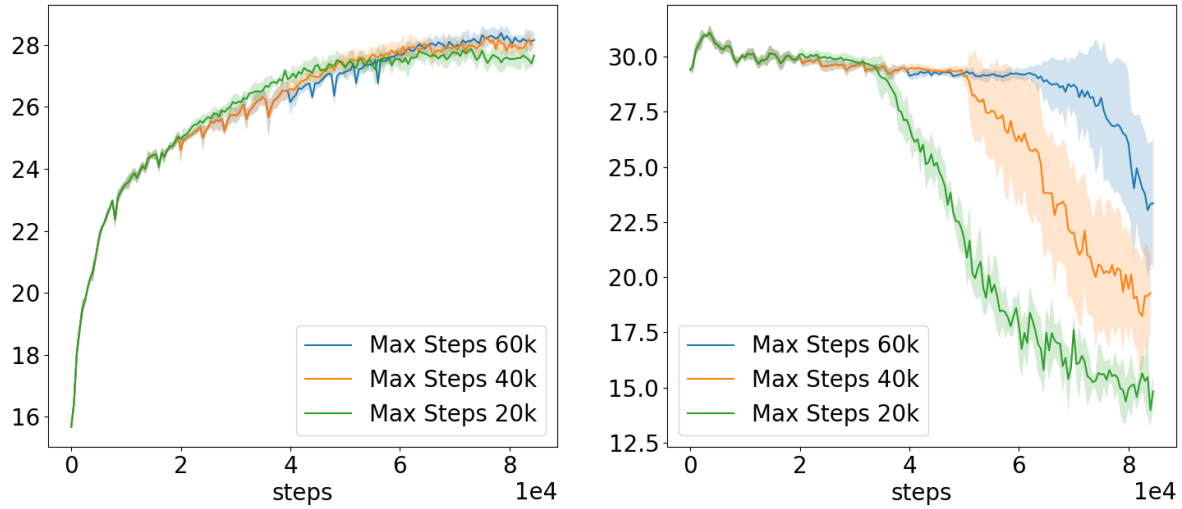
the task score keep improving. This finding supports the view that SIL persists in preventing language drift throughout training, and that the language drift phenomenon itself appear to be robust and not a result of some unstable initialization point.

3.6.7. Qualitative Analysis

In Table 3.1, we show some hand-selected examples of English messages from the translation game. As expected, we observe that the vanilla Gumbel agent diverges from the pretrained language models into unstructured sentences, repeating final dots or words. It also introduce unrecognizable words such as "cfighting" or "acacgame" by randomly pairing up sub-words whenever it faces rare word tokens. S2P and SIL successfully counter the language drift, producing syntactically valid language. However, they can still produce semantically inconsistent captions, which may be due to the poor pretrained model, and the lack of grounding (Lee et al., 2019). Finally, we still observe language drift when dealing with rare word occurrences. Additional global language statistics can be found in Appendix that supports that SIL preserves language statistical properties.

3.7. Conclusion

In this paper we proposed a method to counter language drift in task-oriented language settings. The method, named Seeded Iterated Learning is based on the broader principle of



(a) BLEU De **(b)** BLEU En

Fig. 3.13. Effect of stopping SIL earlier in the training process. SIL maximum steps set at 20k, 40k and 60k. SIL appears to be important in preventing language drift through-out training.

iterated learning. It alternates imitation learning and task optimisation steps. We modified the iterated learning principle so that it starts from a seed model trained on actual human data, and preserve the language properties during training. Our extensive experimental study revealed that this method outperforms standard baselines both in terms of keeping a syntactic language structure and of solving the task. As future work, we plan to test this method on complex dialog tasks involving stronger cooperation between agents.

Chapter 4

Second Article: Supervised Seeded Iterated Learning for Interactive Language Learning

Prologue

Article Details. **Supervised Seeded Iterated Learning for Interactive Langauge Learning.** Yuchen Lu, Soumye Singhal, Florian Strub, Olivier Pietquin, Aaron Courville. This paper was published at EMNLP 2020 as a short paper.

Personal contributions. I conducted all the experiments and wrote the first draft of the paper. All the co-authors helped provide feedback during the weekly meetings and polished the paper.

Discussion and Recent Developments. This is a follow up work to the first article, and it provides a more thorough analysis between the proposed Seeded Iterated Learning and the Supervised Selfplay. The paper presents an interesting failure case of Supervised Selfplay called late-stage collapse, and how gradient conflicting could be a potential explanation.

Abstract

Language drift has been one of the major obstacles to train language models through interaction. When word-based conversational agents are trained towards completing a task, they tend to invent their language rather than leveraging natural language. In recent literature, two general methods partially counter this phenomenon: Supervised Selfplay (S2P) and Seeded Iterated Learning (SIL). While S2P jointly trains interactive and supervised losses to counter the drift, SIL changes the training dynamics to prevent language drift from occurring. In this paper, we first highlight their respective weaknesses, i.e., late-stage training collapses and higher negative likelihood when evaluated on human corpus. Given these observations, we

introduce Supervised Seeded Iterated Learning (SSIL) to combine both methods to minimize their respective weaknesses. We then show the effectiveness of SSIL in the language-drift translation game.

4.1. Introduction

Since the early days of NLP (Winograd, 1971), conversational agents have been designed to interact with humans through language to solve diverse tasks, e.g., remote instructions (Thomason et al., 2015) or booking assistants (Bordes et al., 2017; El Asri et al., 2017). In this goal-oriented dialogue setting, the conversational agents are often designed to compose with predefined language utterances (Lemon and Pietquin, 2007; Williams et al., 2014). Even if such approaches are efficient, they also tend to narrow down the agent's language diversity.

To remove this restriction, recent work has been exploring interactive word-based training. In this setting, the agents are generally trained through a two-stage process (Wei et al., 2018; De Vries et al., 2017; Shah et al., 2018; Li et al., 2016a; Das et al., 2017): Firstly, the agent is pretrained on a human-labeled corpus through supervised learning to generate grammatically reasonable sentences. Secondly, the agent is finetuned to maximize the task-completion score by interacting with a user. Due to sample-complexity and reproducibility issues, the user is generally replaced by a game simulator that may evolve with the conversational agent. Unfortunately, this pairing may lead to the *language drift* phenomenon, where the conversational agents gradually co-adapt, and drift away from the pretrained natural language. The model thus becomes unfit to interact with humans (Chattopadhyay et al., 2017; Zhu et al., 2017; Lazaridou et al., 2020).

While domain-specific methods exist to counter language drift (Lee et al., 2019; Li et al., 2016b), a simple task-agnostic method consists of combining interactive and supervised training losses on a pretraining corpus (Wei et al., 2018; Lazaridou et al., 2016), which was later formalized as Supervised SelfPlay (S2P) (Lowe et al., 2020).

Inspired by language evolution and cultural transmission (Kirby, 2001; Kirby et al., 2014), recent work proposes Seeded Iterated Learning (SIL) (Lu et al., 2020) as another task-agnostic method to counter language drift. SIL modifies the training dynamics by iteratively refining a pretrained student agent by imitating interactive agents, as illustrated in Figure 4.14. At each iteration, a teacher agent is created by duplicating the student agent, which is then finetuned towards task completion. A new dataset is then generated by greedily sampling the teacher, and those samples are used to refine the student through supervised learning. The authors empirically show that this iterated learning procedure induces an inductive learning bias that successfully maintains the language grounding while improving task-completion.

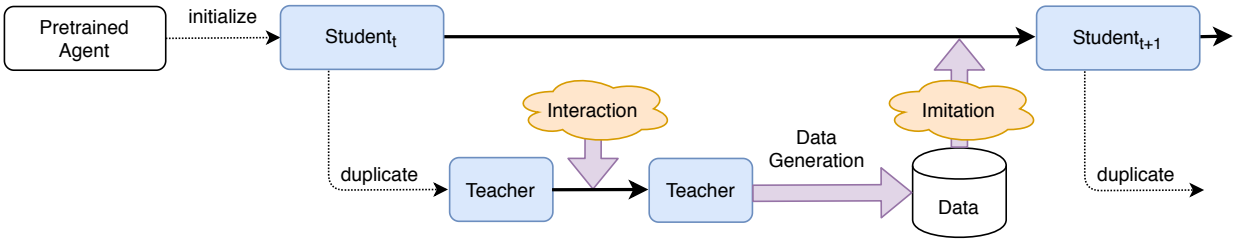


Fig. 4.14. SIL (Lu et al., 2020). A student agent is iteratively refined using newly generated data from a teacher agent. At each iteration, a teacher agent is created on top of the student before being finetuned by interaction, e.g. maximizing a task completion-score. Teacher generates a dataset with greedy sampling and students imitate those samples. The interaction step involves interaction with another language agent.

As a first contribution, we further examine the performance of these two methods in the setting of a translation game (Lee et al., 2019). We show that S2P is unable to maintain a high grounding score and experiences a late-stage collapse, while SIL has a higher negative likelihood when evaluated on human corpus.

We propose to combine SIL with S2P by applying an S2P loss in the interactive stage of SIL. We show that the resulting *Supervised Seeded Iterated Learning* (SSIL) algorithm manages to get the best of both algorithms in the translation game. Finally, we observe that the late-stage collapse of S2P is correlated with conflicting gradients before showing that SSIL empirically reduces this gradient discrepancy.

4.2. Preventing Language Drift

We describe here our interactive training setup before introducing different approaches to prevent language drift. In this setting, we have a set of collaborative agents that interact through language to solve a task. To begin, we train the agents to generate natural language in a word-by-word fashion. Then we finetune the agents to optimize a task completion score through interaction, i.e., learning to perform the task better. Our goal is to prevent the language drift in this second stage.

4.2.1. Initializing the Conversational Agents

For a language agent f parameterized by $\boldsymbol{\theta}$, and a sequence of generated words $\boldsymbol{w}_{1:i} = [w_j]_{j=1}^{i}$ and an arbitrary context $\boldsymbol{c}$, the probability of the next word w_i is $p(w_{i+1}|\boldsymbol{w}_{1:i}, \boldsymbol{c}) = f_{\boldsymbol{\theta}}(\boldsymbol{w}_{1:i}, \boldsymbol{c})$ We pretrain the language model to generate meaningful sentences by minimizing the cross-entropy loss $\mathcal{L}^{\text{CE}}_{pretrain}$ where the word sequences are sampled from a language corpus $D_{pretrain}$. Note that this language corpus may either be task-related or generic. Its role is to get our conversational agents a reasonable initialization.

4.2.2. Supervised Selfplay (S2P)

A common way to finetune the language agents while preventing language drift is to replay the pretraining data during the interaction stage. In S2P the training loss encourages both maximizing task-completion while remaining close to the initial language distribution. Formally,

$$\mathcal{L}^{\text{S2P}} = \mathcal{L}^{\text{INT}} + \alpha \mathcal{L}^{\text{CE}}_{pretrain} \tag{4.2.1}$$

where $\mathcal{L}^{\text{INT}}$ is a differentiable interactive loss maximizing task completion, e.g. reinforcement learning with policy gradients (Sutton et al., 2000), Gumbel Straight-through Estimator (STE) (Jang et al., 2017b) etc., $\mathcal{L}^{\text{CE}}_{pretrain}$ is a cross-entropy loss over the pretraining samples. α is a positive scalar which balances the two losses.

4.2.3. Seeded Iterated Learning (SIL)

Seeded Iterated Learning (SIL) iteratively refines a pretrained *student* model by using data generated from newly trained *teacher* agents (Lu et al., 2020). As illustrated in Figure 4.14, the student agent is initialized with the pretrained model. At each iteration, a new teacher agent is generated by duplicating the student parameters. It is tuned to maximize the task-completion score by optimizing the interactive loss $\mathcal{L}^{\text{TEACHER}} = \mathcal{L}^{\text{INT}}$ In a second step, we sample from the teacher to generate new training data $D_{teacher}$, and we refine the student by minimizing the cross-entropy loss $\mathcal{L}^{\text{STUDENT}} = \mathcal{L}^{\text{CE}}_{teacher}$ where sequence of words are sampled from $D_{teacher}$. This imitation learning stage can induce an information bottleneck, encouraging the student to learn a well-formatted language rather than drifted components.

4.2.4. SSIL: Combining SIL and S2P

S2P and SIL have two core differences: first, SIL never re-uses human pretraining data. As observed in Section 4.4.1, this design choice reduces the language modeling ability of SIL-trained agents, with a higher negative likelihood when evaluated on human corpus. Second, S2P agents merge interactive and supervised losses, whereas SIL's student never experiences an interactive loss. As analyzed in Section 4.4.3, the S2P multi-task loss induces conflicting gradients, which may trigger language drift. In this paper, we propose to combine these two approaches and demonstrate that the combination effectively minimizes their respective weaknesses. To be specific, we apply the S2P loss over the SIL teacher agent, which entails $\mathcal{L}^{\text{TEACHER}} = \mathcal{L}^{\text{INT}} + \alpha \mathcal{L}^{\text{CE}}_{pretrain}$. We call the resulting algorithm, Supervised Seeded Iterated Learning (SSIL). In SSIL, teachers can generate data that is close to the human distribution due to the S2P loss, while students are updated with a consistent supervised loss to avoid the potential weakness of multi-task optimization. In addition, SSIL still maintains the inductive

Finetuning Methods	*Training Losses*
Gumbel	$\mathcal{L}^{\text{INT}}$
S2P	$\mathcal{L}^{\text{INT}} + \alpha\mathcal{L}^{\text{CE}}_{pretrain}$
SIL (teacher)	$\mathcal{L}^{\text{INT}}$
SIL (student)	$\mathcal{L}^{\text{CE}}_{teacher}$
SSIL (teacher)	$\mathcal{L}^{\text{INT}} + \alpha\mathcal{L}^{\text{CE}}_{pretrain}$
SSIL (student)	$\mathcal{L}^{\text{CE}}_{teacher}$

Table 4.2. Finetuning with respective training objective.

learning bias of SIL. We list all these methods in Table 4.2 for easy comparison. We also experiment with other ways of combining SIL and S2P by mixing the pretraining data with teacher data during the imitation learning stage. We call this method *MixData*. We show the results of this approach in Appendix 4.4.2. We find that this approach is very sensitive to the mixing ratio of these two kinds of data, and the best configuration is still not as good as SSIL.

4.3. Experimental Setting

4.3.1. Translation Game

We replicate the translation game setting from (Lee et al., 2019) as it was designed to study language drift. First, a *sender* agent translates French to English (Fr-En), while a *receiver* agent translates English to German (En-De). The sender and receiver are then trained together to translate French to German with English as a pivot language. For each French sentence, we sample English from the sender, send it to the receiver, and sample German from the receiver.

The task score is defined as the BLEU score between generated German translation and the ground truth (*BLEU De*) (Papineni et al., 2002). The goal is to improve the task score without losing the language structure of the intermediate English language.

4.3.2. Training Details

The sender and the receiver are pretrained on the IWSLT dataset (Cettolo et al., 2012) which contains (Fr, En) and (En, De) translation pairs. We then use the Multi30k dataset (Elliott et al., 2016) to build the finetuning dataset with (Fr, De) pairs. As IWSLT is a generic translation dataset and Multi30k only contains visually grounded translated captions, we

also call IWSLT task-agnostic while Multi30K task-related. We use the cross-entropy loss of German as the interactive training objective, which is differentiable w.r.t. the receiver. For the sender, we use Gumbel Softmax straight-through estimator to make the training objective also differentiable w.r.t. the sender, as in Lu et al. (2020).

Implementation details are in Appendix B.2

4.3.3. Metrics for Grounding Scores

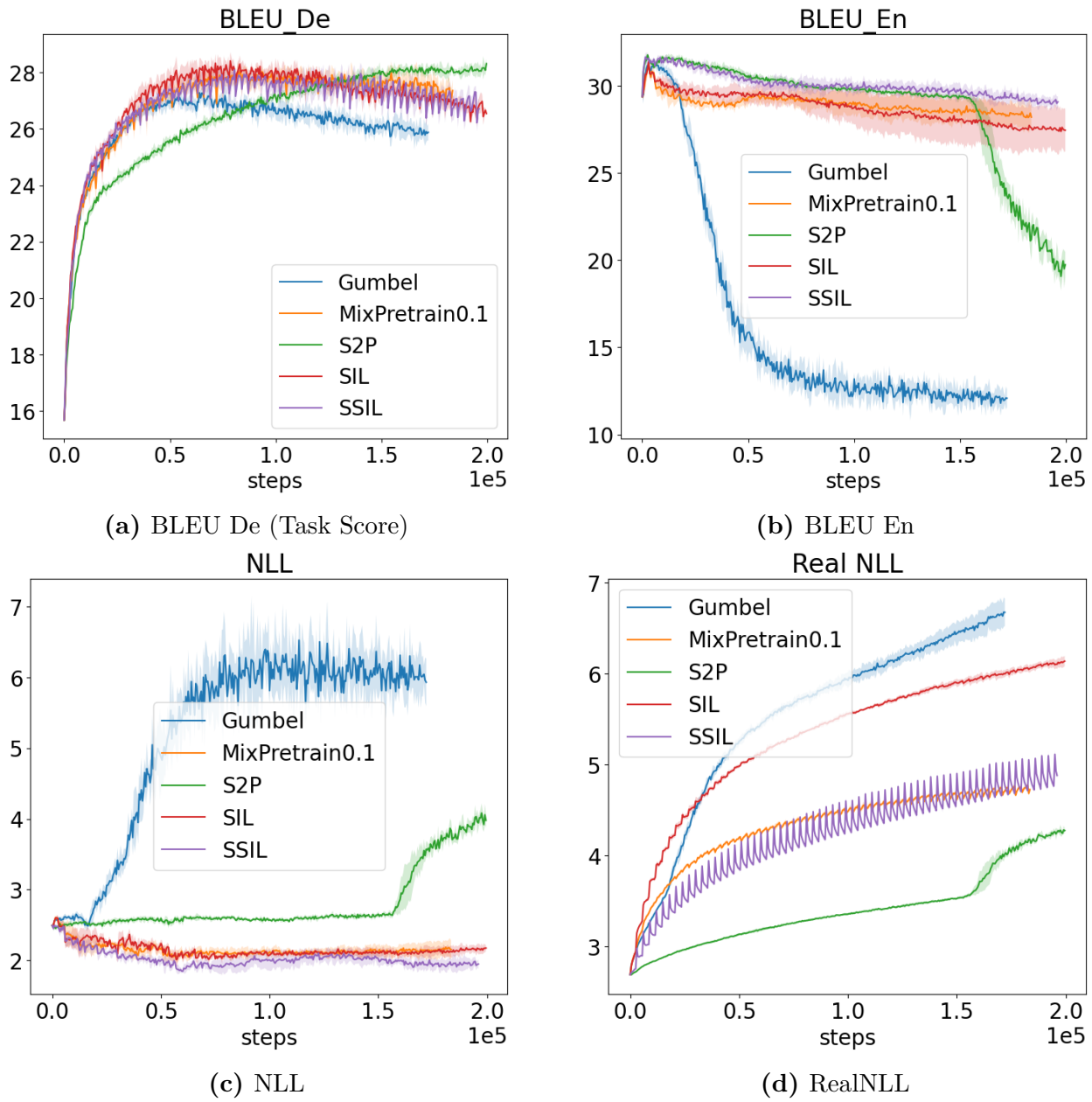
In practice, there are different kinds of language drift (Lazaridou et al., 2020) (e.g. syntactic drift and semantic drift). We thus have multiple metrics to consider when evaluating language drift. We first compute English BLEU score (*BLEU En*) comparing the generated English translation with the ground truth human translation. We include the negative log-likelihood (*NLL*) of the generated En translation under a pretrained language model as a measure of syntactic correctness. In line with (Lu et al., 2020) , we also report results using another language metric: the negative log-likelihood of human translations (*RealNLL*) given a finetuned Fr-En model. We feed the finetuned sender with human task-data to estimate the model's log likelihood. The lower is this score, the more likely the model would generate such human-like language.

4.4. Experiments

4.4.1. S2P and SIL Weaknesses

We report the task and grounding scores of vanilla Gumbel, S2P, SIL, and SSIL in Figure 4.15. The respective best hyper-parameters can be found in the appendix. As reported by Lu et al. (2020), vanilla Gumbel successfully improves the task score *BLEU De*, but the *BLEU En* score as well as other grounding metric collapses, indicating a language drift during the training. Both S2P and SIL manage to increase *BLEU De* while maintaining a higher *BLEU En* score, countering language drift. However, S2P has a sudden (and reproducible) late-stage collapse, unable to maintain the grounding score beyond 150k steps. On the other hand, SIL has a much higher RealNLL than S2P, suggesting that SIL has a worse ability to model human data.

SSIL seems to get the best of the two worlds. It has a similar task score *BLEU De* as S2P and SIL, while it avoids the late-stage collapse. It ends up with the highest *BLEU En*, and it improves the RealNLL over SIL, though still not as good as S2P. Also, it achieves even better NLL, suggesting that its outputs are favoured by the pretrained language model.



(a) BLEU De (Task Score)

(b) BLEU En

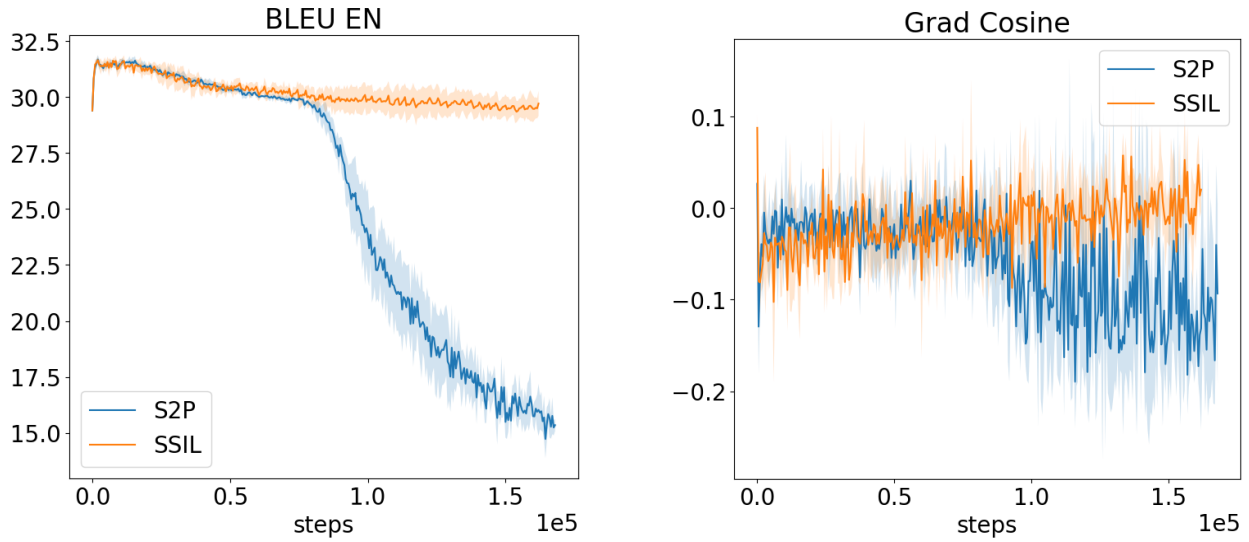
(c) NLL

(d) RealNLL

Fig. 4.15. Task and language metrics for Vanilla Gumbel, SIL, S2P, and SSIL in the translation game average over 5 seeds. We also show the results of mixing pretraining data in the teacher dataset (Section 4.4.2). The plots are averaged over 5 seeds with shaded area as standard deviation. Although SIL and S2P both counter language drift, S2P suffers from late collapse, and SIL has a high *RealNLL*, suggesting that its output may not correlate well with human sentences.

4.4.2. Mixing Teacher and Human Data

We also explore whether injecting pretraining data into the teacher dataset may be a valid substitute for the S2P loss. We add a subset of the pretraining data in the teacher dataset



(a) Bleu En **(b)** Cosine Similarity

Fig. 4.16. Cosine similarity between the gradients issued from $\mathcal{L}^{\text{INT}}$ and $\mathcal{L}^{\text{CE}}_{pretrain}$. The collapse of the BLEU En matches the negative cosine similarity.We here set $\alpha = 0.5$ but similar values yield identical behavior as shown in Figure B.31 in Appendix.

before refining the student, and we report the results in Figure 4.15 and B.33. Unfortunately, such an approach was quite unstable, and it requires heavy hyper-parameters tuning to match SSIL scores. As explained in (Kirby, 2001), iterated learning rely on inductive learning to remove language irregularities during the imitation step. Thus, mixing two language distributions may disrupt this imitation stage.

4.4.3. Why S2P collapses?

We investigate the potential cause of S2P late-stage collapse and how SSIL may resolve it. We firstly hope to solve this by increasing the supervised loss weight α. However, we find that a larger α only delays the eventual collapse as well as decreases the task score, as shown in Figure B.32 in Appendix B.4.

We further hypothesize that this late-stage collapse can be caused by the distribution mismatch between the pretraining data (IWSLT) and the task-related data (Multi30K), exemplified by their word frequencies difference. A mismatch between the two losses could lead to conflicting gradients, which could, in turn, make training unstable. In Figure 4.16, we display the cosine similarity of the sender gradients issued by the interactive and supervised losses $cos(\nabla_{sender}\mathcal{L}^{\text{INT}}, \nabla_{sender}\mathcal{L}^{\text{CE}}_{pretrain})$ for both S2P and SSIL for $\alpha = 0.5$ during training. Early in S2P training, we observe that the two gradients remain orthogonal on average, with the cosine oscillating around zero. Then, at the same point where the S2P *Bleu En* collapses, the cosine of the gradients starts trending negative, indicating that the gradients are pointing

in opposite directions. However, SSIL does not have this trend, and the *BLEU En* does not collapse. Although the exact mechanism of how conflicting gradients trigger the language drift is unclear, current results favor our hypothesis and suggest that language drift could result from standard multi-task optimization issues (Yu et al., 2020a; Parisotto et al., 2016; Sener and Koltun, 2018) for S2P-like methods.

Conclusion. We investigate two general methods to counter language drift: S2P and SIL. S2P experiences a late-stage collapse on the grounding score, whereas SIL has a higher negative likelihood on human corpus. We introduce SSIL to combine these two methods effectively. We further show the correlation between S2P late-stage collapse and conflicting gradients.

Acknowledgement. We thank Compute Canada (www.computecanada.ca) for providing the computational resources. We thank Miruna Pislar and Angeliki Lazaridou for their helpful discussions.

Chapter 5

Third Article: Learning Task Decomposition with Ordered Memory Policy Network

Prologue

Article Details. **Learning Task Decomposition with Ordered Memory Policy Network.** Yuchen Lu, Yikang Shen, Siyuan Zhou, Aaron Courville, Joshua B. Tenenbaum, Chuang Gan. This paper was published at ICLR 2021.

Personal contributions. I conceived the idea of adopting the inductive bias of ordered memory into the task of unsupervised decomposition. Yikang Shen and I jointly design the model architecture. Siyuan Zhou helped run the experiments for the robotics experiments. Chuang Gan, Aaron Courville, and Josh Tenenbaum helped polished the paper, which I drafted the most of.

Discussion and Recent Developments. While unsupervised task decomposition is widely studied under probabilistic graphic models with the seminal work of CompILE (Kipf et al., 2019), our work present the first solution to this problem that relies purely on network inductive bias. Compared with previous work, our method has less ad-hoc constraints.

Since publication, we adapt the proposed network architecture so that the model can leverage the learned subtask in the downstream hierarchical reinforcement learning (Zhou et al., 2022). Inspired by our approach, Inspired by our work, Gopalakrishnan et al. (2021) introduces slot-based transformers as another architectural inductive bias, which are originally used for unsupervised object discovery.

Abstract

Many complex real-world tasks are composed of several levels of sub-tasks. Humans leverage these hierarchical structures to accelerate the learning process and achieve better generalization. In this work, we study the inductive bias and propose Ordered Memory Policy Network (OMPN) to discover subtask hierarchy by learning from demonstration. The discovered subtask hierarchy could be used to perform task decomposition, recovering the subtask boundaries in an unstructured demonstration. Experiments on Craft and Dial demonstrate that our model can achieve higher task decomposition performance under both unsupervised and weakly supervised settings, comparing with strong baselines. OMPN can also be directly applied to partially observable environments and still achieve higher task decomposition performance. Our visualization further confirms that the subtask hierarchy can emerge in our model [1].

5.1. Introduction

Learning from Demonstration (LfD) is a popular paradigm for policy learning and has served as a warm-up stage in many successful reinforcement learning applications (Vinyals et al., 2019; Silver et al., 2016). However, beyond simply imitating the experts' behaviors, an intelligent agent's crucial capability is to decompose an expert's behavior into a set of useful skills and discover sub-tasks. The discovered structure from expert demonstrations could be leveraged to re-use previously learned skills in the face of new environments (Sutton et al., 1999; Gupta et al., 2019b; Andreas et al., 2017). Since manually labeling sub-task boundaries for each demonstration video is extremely expensive and difficult to scale up, it is essential to learn task decomposition *unsupervisedly*, where the only supervision signal comes from the demonstration itself.

This question of discovering a meaningful segmentation of the demonstration trajectory is the key focus of Hierarchical Imitation Learning (Kipf et al., 2019; Shiarlis et al., 2018; Fox et al., 2017; Achiam et al., 2018) These works can be summarized as finding the optimal behavior hierarchy so that the behavior can be better predicted (Solway et al., 2014). They usually model the sub-task structure as latent variables, and the subtask identifications are extracted from a learnt posterior. In this paper, we propose a novel perspective to solve this challenge: could we design a *smarter* neural network architecture, so that the sub-task structure can emerge during imitation learning? To be specific, we want to design a recurrent policy network such that examining the memory trace at each time step could reveal the underlying subtask structure.

[1] Project page: `https://ordered-memory-rl.github.io/`

Drawing inspiration from the Hierarchical Abstract Machine (Parr and Russell, 1998), we propose that each subtask can be considered as a finite state machine. A hierarchy of sub-tasks can be represented as different slots inside the memory bank. At each time step, a subtask can be internally updated with the new information, call the next-level subtask, or return the control to the previous level subtask. If our designed architecture maintains a hierarchy of sub-tasks operating in the described manner, then subtask identification can be as easy as monitoring when the low-level subtask returns control to the higher-level subtask, or when the high-level subtask expands to the new lower-level subtask.

We give an illustrative grid-world example in Figure 5.17. In this example, there are different ingredients like grass for the agent to pickup. There is also a factory where the agent can use the ingredients. Suppose the agent wants to complete the task of building a bridge. This task can be decomposed into a tree-like, multi-level structure, where the root task is divided into $GetMaterial$ and $BuildBridge$. $GetMaterial$ can be further divided into $GetGrass$ and $GetWood$. We provide a sketch on how this subtask structure should be represented inside the agent's memory during each time step. The memory would be divided into different levels, corresponding to the subtask structure. When $t = 1$, the model just starts with the root task, $MakeBridge$, and vertically expands into $GetMaterial$, which further vertically expands into $GetWood$. The *vertical expansion* corresponds to planning or calling the next level subtasks. The action is produced from the lowest-level memory. The intermediate $GetMaterial$ is copied for $t < 3$, but horizontally updated at $t = 3$, when $GetWood$ is finished. The *horizontal update* can be thought of as an internal update for each subtask, and the updated $GetMaterial$ vertically expands into a different child $GetGrass$. The root task is always copied until $GetMaterial$ is finished at $t = 4$. As a result, $MakeBridge$ goes through one horizontal update at $t = 5$ and then expands into $BuildBridge$ and $GoFactory$. We can identify the subtask boundaries from this representation by looking at the change of *expansion position*, which is defined to be the memory slot where vertical expansion happens. E.g., from $t = 2$ to $t = 3$, the expansion position goes from the lowest level to the middle level, suggesting the completion of the low-level subtask. From $t = 4$ to $t = 5$, the expansion position goes from the lowest level to the highest level, suggesting the completion of both low-level and mid-level subtasks.

Driven by this intuition, we propose the *Ordered Memory Policy Network* (OMPN) to support the subtask hierarchy described in Figure 5.17. We propose to use a bottom-up recurrence and a top-down recurrence to implement *horizontal update* and *vertical expansion* respectively. Our proposed memory-update rule further maintains a hierarchy among memories such that the higher-level memory can store longer-term information. At each time step, the model would softly decide the expansion position from which to perform vertical expansion based on a differentiable stick-breaking process, so that our model can be trained end-to-end.

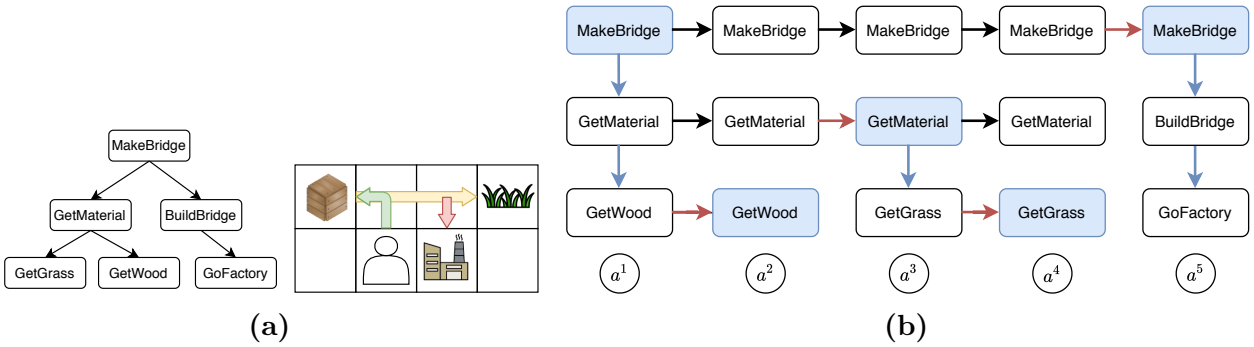


Fig. 5.17. (a) A simple grid world with the task "make bridge", which can be decomposed into multi-level subtask structure. (b) The representation of subtask structure within the agent memory with *horizontal update* and *vertical expansion* at each time step. The black arrow indicates a copy operation. The *expansion position* is the memory slot where the vertical expansion starts and is marked blue.

We demonstrate the effectiveness of our approach with multi-task behavior cloning. We perform experiments on both grid-world as well as more challenging robotic tasks. We show that OMPN is able to perform task decomposition in both an unsupervised and weakly supervised manner, comparing favorably with strong baselines. Meanwhile, OMPN still maintains the similar, if not better, performance on behavior cloning in terms of sample complexity and returns. Our ablation study shows the contribution of each component in our architecture. Our visualization further confirms that the subtask hierarchy emerges in our model's expanding positions.

5.2. Ordered Memory Policy Network

We describe our policy architecture given the intuition described above. Our model is a recurrent policy network $p(a^t|s^t, M^t)$ where $M \in \mathcal{R}^{n \times m}$ is a block of n memory while each memory has dimension m. We use M_i to refer to the ith slot of the memory, so $M = [M_1, M_2, ..., M_n]$. The highest-level memory is M_n while the lowest-level memory is M_1. Each memory can be thought of as the representation of a subtask. We use the superscript to denote the time step t.

At each time step, our model will first transform the observation $s^t \in \mathcal{S}$ to $x^t \in \mathcal{R}^m$. This can be achieved by a domain-specific observation encoder. Then we have an ordered-memory module $M^t, O^t = OM(x^t, M^{t-1})$ to generate the next memory and the output. The output O^t is sent into a feed-forward neural net to generate the action distribution.

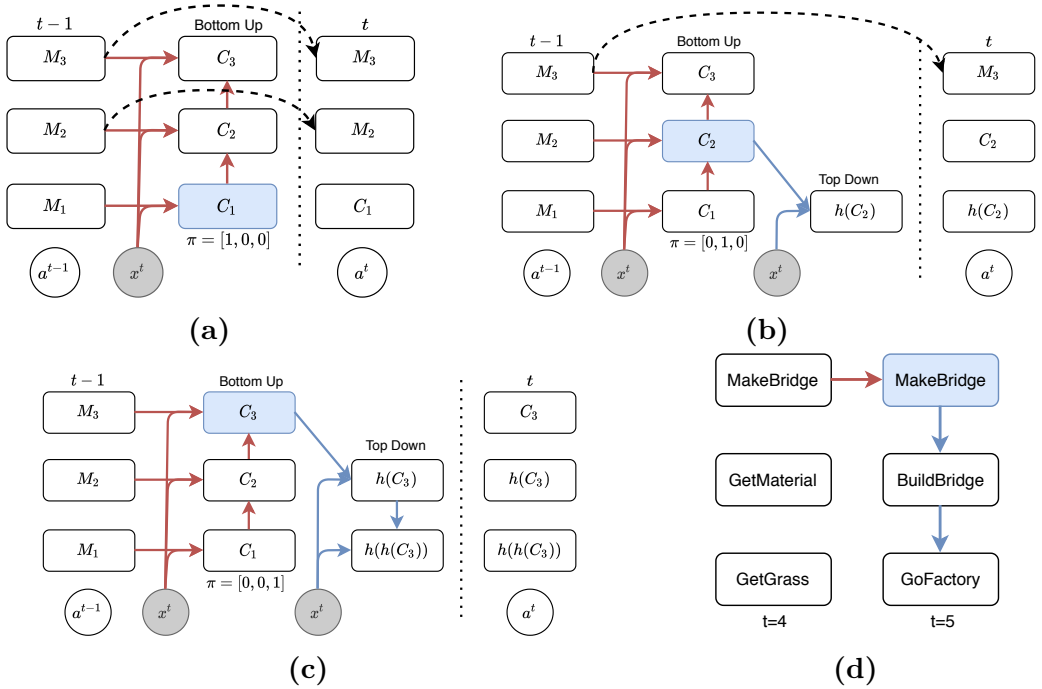


Fig. 5.18. Dataflow of how M^{t-1} will be updated in M^t for three memory slots when the expansion position is at a (a) low, (b) middle, or (c) high position. Blue arrows and red arrows corresponding to the vertical expansions and horizontal updates. (d) is a snapshot of $t = 5$ from the grid-world example Figure 5.17b. The subtask-update behavior corresponds to the memory-update when the expansion position is at the high position.

5.2.1. Ordered Memory Module

The ordered memory module first goes through a bottom-up recurrence. This operation implements the *horizontal update* and updates each memory with the new observation. We define C^t to be the updated memory:

$$C_i^t = \mathcal{F}(C_{i-1}^t, x^t, M_i^{t-1})$$

for $i = 1,...,n$ where $C_0^t = x^t$ and $\mathcal{F}$ is a cell function. Different from our mental diagram, we make it an recurrent process since the high-level memory might be able to get information from the updated lower-level memory in addition to the observation. In our experiment we find that such recurrence will help model perform better than in task decomposition. For each memory, we also generate a score f_i^t from 0 to 1 with $f_i^t = \mathcal{G}(x^t, C_i^t, M_i^t)$ for $i = 1,...,n$. The score f_i^t can be interpreted as the probability that subtask i is completed at time t.

In order to properly generate the final *expansion position*, we would like to insert the inductive bias that the higher-level subtask is expanded only if the higher-level subtask is not completed while all the lower-level subtasks are completed, as is shown in Figure 5.17b. As a

result we use a stick-breaking process as follows:

$$\hat{\pi}_i^t = \begin{cases} (1 - f_i^t) \prod_{j=1}^{i-1} f_j^t & 1 < i \leq n \\ 1 - f_1^t & i = 1 \end{cases}$$

Finally we have the expansion position $\pi_i^t = \hat{\pi}_i^t / \sum \hat{\pi}^t$ as a properly normalized distribution over n memories. We can also define the ending probability as the probability that every subtask is finished.

$$\pi_{end}^t = \prod_{i=1}^{n} f_i^t \tag{5.2.1}$$

Then we use a top-down recurrence on the memory to implement the vertical expansion. Starting from $\hat{M}_n^t = 0$, we have

$$\hat{M}_i^t = h(\overleftarrow{\pi}_{i+1}^t C_{i+1}^t + (1 - \overleftarrow{\pi}_{i+1}^t)\hat{M}_{i+1}^t, x^t),$$

where $\overrightarrow{\pi}_i^t = \sum_{j \geq i} \pi_j^t$, $\overleftarrow{\pi}_i^t = \sum_{j \leq i} \pi_j^t$, and h can be any cell function. Then we update the memory in the following way:

$$M^t = M^{t-1}(1 - \overrightarrow{\pi}^t) + C^t \pi^t + \hat{M}^t(1 - \overleftarrow{\pi}^t) \tag{5.2.2}$$

where the output is read from the lowest-level memory $O^t = M_1^t$. For better understanding purpose, we show in Figure 5.18 how M^{t-1} will be updated into M^t with $n = 3$, when the expansion position is at a high, middle and low position respectively. The memory higher than the expansion position will be preserved, while the memory at and lower than the expansion position will be over-written. We also take the snapshot of $t = 5$ from our the early example in Figure 5.17b and show that the subtask-update behavior corresponds to our memory-update when the expansion position is at the high position.

Although we show only the discrete case for illustration, the vector π^t is actually continuous. As a result, the whole process is fully differentiable and can be trained end-to-end. More details can be found in the appendix C.1.

The memory depths n is a hyper-parameter here. If n is too small, we might not have enough capacity to cover the underlying structure in the data, If n is too large, we might impose some extra optimization difficulty. In our experiments we investigate the effect of using different n.

5.2.2. Unsupervised Task Decomposition with Behavior Cloning

We assume the following setting. We firstly have a *training phase* to perform behavior cloning on an unstructured demonstration dataset with state-action pairs. Then during the

detection phase, the user would specify the number of subtasks K for the model to produce the task boundaries.

We firstly describe our *training phase*. We develop a unique regularization technique which can help our model learning the underlying hierarchy structure. Suppose we have an action space $\mathcal{A}$. We first augment this action space with $\mathcal{A}' = \mathcal{A} \cup \{done\}$, where *done* is a special action. Then we can modify the action distribution accordingly:

$$p'(a^t|s^t) = \begin{cases} p(a^t|s^t)(1 - \pi^t_{end}) & a^t \in \mathcal{A} \\ \pi^t_{end} & a^t = done \end{cases}$$

Then for each demonstration trajectory $\tau = \{s^t, a^t\}_{t=1}^T$, we transformed it into $\tau' = \tau \cup \{s^{T+1}, a^{T+1} = done\}$, which is essentially telling the model to output *done* only after the end of the trajectory. This process can be achieved on both discrete and continuous action space without heavy human involvement described in Appendix C.1. Then we will maximize $\sum_{t=1}^{T+1} \log p'(a^t|s^t)$ on τ'. We find that including π^t_{end} into the loss is crucial to prevent our model degenerating into only using the lowest-level memory, since it provides the signal to raise the expansion position at the end of the trajectory, benefiting the task decomposition performance. We also justify this in our ablation study.

Since the expansion position should be high if the low-level subtasks are completed, we can achieve unsupervised task decomposition by monitoring the behavior of π^t. To be specific, we define $\pi^t_{avg} = \sum_{i=1}^n i\pi^t_i$ as the expected expansion position. Given π_{avg}, we consider the following methods to recover the subtask boundaries.

Top-K. In this method we choose the time steps of K largest π_{avg} to detect the boundary, where K is the desired number of sub-tasks given by the user during the detection phase. We find that this method is suitable for the discrete action space, where there is a very clear boundary between subtasks.

Thresholding. In this method we standardize the π_{avg} into $\hat{\pi}_{avg}$ from 0 to 1, and then we compute a Boolean array $\mathbf{1}(\pi_{avg} > thres)$, where $thres$ is from 0 to 1. We retrieve the subtask boundaries from the ending time step of each $True$ segments. We find this method is suitable for continuous control settings, where the subtask boundaries are more ambiguous and smoothed out across time steps.

5.3. Related Work

Our work is related to option discovery and hierarchical imitation learning. The existing option discovery works have focused on building a probabilistic graphical model on the

trajectory, with options as latent variables. DDO (Fox et al., 2017) proposes an iterative EM-like algorithm to discover multiple level of options from the demonstration. DDO was later applied in the continuous action space (Krishnan et al., 2017) and program modelling (Fox et al., 2018). Recent works like compILE (Kipf et al., 2019) and VALOR (Achiam et al., 2018) also extend this idea by incorporating more powerful inference methods like VAE (Kingma and Welling, 2013). Lee (2020) also explore unsupervise task decompostion via imitation, but their method is not fully end-to-end, requires an auxiliary self-supervision loss, and does not support multi-level structure. Our work focuses on the role of neural network inductive bias in discovering re-usable options or subtasks from demonstration. We do not have an explicit "inference" stage in our training algorithm to infer the option/task ID from the observations. Instead, this inference "stage" is implicitly designed into our model architecture via the stick-breaking process and expansion position. Based on these considerations, we choose compILE as the representative baseline for this field of work.

Our work is also related to Hierarchical RL (Vezhnevets et al., 2017; Nachum et al., 2018; Bacon et al., 2017). These works usually propose an architecture that has a high-level controller to output a goal, while the low-level architecture takes the goal and outputs the primitive actions. However, these works mainly deal with the control problem, and do not focus on learning task decomposition from the demonstration. Recent works (Gupta et al., 2019b; Lynch et al., 2020) also include hierarchical imitation learning stage as a way to pretraining low-level policies before apply Hierarchical RL for finetuning, however they do not produce the task boundaries and therefore are not comparable to our works. Moreover, their hierarchical IL algorithm exploits the fact that the goal can be described as a point in the state space, so that they are able to re-label the ending state of an unstructured demonstrations as a fake goal to train the goal-conditioned policy. Meanwhile our approach is designed to be general. In addition to the option framework, our work is closely related to Hierarchical Abstract Machine (HAM) (El Hihi and Bengio, 1996). Our concept of subtask is similar to the finite state machine (FSM). The horizontal update corresponds to the internal update of the FSM, while the vertical expansion corresponds to calling the next level of the FSM. Our stick-breaking process is also a continuous realization of the idea that low-level FSM transfers control back to high-level FSM at completion.

Recent work (Andreas et al., 2017) introduces the modular policy networks for reinforcement learning so that it can be used to decompose a complex task into several simple subtasks. In this setting, the agent is provided a sequence of subtasks, called *sketch*, at the beginning. Shiarlis et al. (2018) propose TACO to jointly learn sketch alignment with action sequence, as well as imitating the trajectory. This work can only be applied in the "weakly supervised" setting, where they have some information like the sub-task sequence. Nevertheless, we also choose TACO (Shiarlis et al., 2018) as one of our baselines.

Incorporating varying time-scale for each neuron to capture hierarchy is not a new idea (Chung et al., 2016; El Hihi and Bengio, 1996; Koutnik et al., 2014). However, these works do not focus on recovering the structure after training, which makes these methods less interpretable. Shen et al. (2018) introduce Ordered Neurons and show that they can induce syntactic structure by examining the hidden states after language modelling. However ONLSTM does not provide mechanism to achieve the top-down and bottom-up recurrence. Our model is mainly inspired by the Ordered Memory (Shen et al., 2019). However, unlike previous work our model is a decoder expanding from root task to subtasks, while the Ordered Memory is an encoder composing constituents into sentences. Recently Mittal et al. (2020) propose to combine top-down and bottom-up process. However their main motivation is to handle uncertainty in the sequential prediction and they do not maintain a hierarchy of memories with different update frequencies.

5.4. Experiment

We would like to evaluate whether OMPN is able to jointly learning task decomposition during behavior cloning. We would like to answer the following questions in our experiments.

Q1: Can OMPN be applied in both continuous and discrete action space?

Q2: Can OMPN be applied in both unsupervised, as well as, weakly supervised setting for task decomposition?

Q3: How much does each component helps the task decomposition?

Q4: How does the task decomposition performance change with different hyper-parameters, e.g., memory dimension m and memory depths n?

5.4.1. Setup and Metrics

For the discrete action space, we use a grid world environment called Craft adapted from Andreas et al. (2017)[2]. At the beginning of each episode, an agent is equipped with a task along with the sketch, e.g. $makecloth = (getgrass, gofactory)$. The original environment is fully observable. To further test our model, we make it also support partial observation by providing a self-centric window. For the continuous action space, we have a robotic setting called Dial (Shiarlis et al., 2018) where a JACO 6DoF manipulator interact with a large number pad[3]. For each episode, the sketch is a sequence of numbers to be pressed. More details on the demonstration can be found in Appendix C.2.

[2] https://github.com/jacobandreas/psketch

[3] https://github.com/KyriacosShiarli/taco/tree/master/taco/jac

We experiment in both unsupervised and weakly supervised settings. For the unsupervised setting, we did not provide any task information. For the weakly supervised setting, we provide the subtask sequence, or sketch, in addition to the observation. We use *nosketch* and *sketch* to denote these two settings respectively. We choose compILE to be our unsupervised baseline while TACO to be our weakly supervised baseline.

We use the ground-truth K to get the best performance of both our models and the baselines. This setting is consistent with the previous literature (Kipf et al., 2019). The details about task decomposition metric can be found in the appendix C.3.

5.4.2. Task Decomposition Results

		Craft				Dial
		Full		Partial		
		Align Acc	F1(tol=1)	Align Acc	F1(tol=1)	Align Acc
NoSketch	OMPN	**93(1.7)**	95(1.2)	**84(6)**	**89(4.6)**	**87(4.0)**
	compILE	86(1.4)	**97(0.8)**	54(1.4)	57(4.8)	45(5.2)
Sketch	OMPN	**97(1.2)**	98(0.9)	**78(10.7)**	83(8.1)	84(5.7)
	TACO	90(3.6)	-	66(2.2)	-	**98(0.1)**

Table 5.3. Alignment accuracy and F1 scores with tolerance 1 on Craft and Dial. The results are averaged over five runs, and the number in the parenthesis is the standard deviation.

In this section, we mainly address the question **Q1** and **Q2**. Our main results for task decomposition in Craft and Dial are in Table 5.3. In Craft, we use the $TopK$ detection algorithm. Our results show that OMPN is able to outperform baselines in both unsupervised and weakly-supervised settings with a higher F1 scores and alignment accuracy. In general, there is a decrease of the performance for all models when moving from full observations to partial observations. Nevertheless, compared with the baselines, we find that OMPN suffers less performance drop than the baselines. The F1 score results with different K other than the ground truth is in Table C.14 and Table C.15.

In Dial, our results is able to outperform compILE for the unsupervised setting, but is not better than TACO when the sketch information is given. In general, our current model does not benefit from the additional sketch information, and we hypothesize that it is because we use a very trivial way of incorporating sketch information by simple concatenating it with the observation. A more effective way would be using attention over the sketch sequence to properly feed the related task information into the model. The F1 score results with different K other than the ground truth is in Table C.17.

5.4.3. Qualitative Analysis

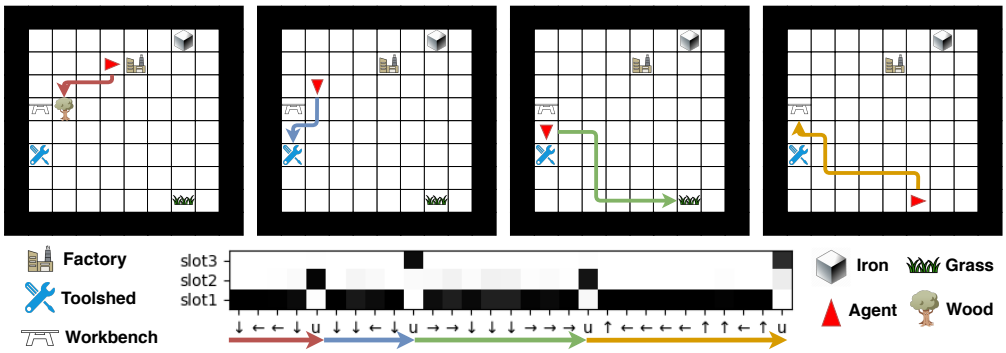


Fig. 5.19. Visualization of the learnt expanding positions on Craft. We present π and the action sequence. The four subtasks are *GetWood*, *GoToolshded*, *GetGrass* and *GoWorkbench*. In the action sequence, "u" is either picking up/using the object. Each ground truth subtask is highlighted with an arrow of different colors.

In this section, we provide some visualization of the learnt expanding positions to qualitatively evaluate whether OMPN can operate with a subtask hierarchy.

We firstly visualize the behavior of expanding positions for Craft in Figure 5.19. We find that at the end of each subtask, the model learns to switch to a higher expansion position when facing the target object/place, while within each subtask, the model learns to maintain the lower expansion position. This is our desired behavior described in Figure 5.17. What is interesting is that although the ground truth hierarchy structure might be only two-levels, our model is able to learn multiple level hierarchy if given some redundant depths. In this example, the model combines the first two subtasks into one group and the last two subtasks into another group. More results can be found in Appendix C.5.

In Figure 5.20, we show the qualitative result in Dial. We find that, instead of having a sharp subtask boundary, the boundary between subtasks is more ambiguous with high expanding positions across multiple time steps, and this motivates our design of the threshold algorithm. This happens because we use the observation to determine the expanding position. In Dial, the state difference are less significant due to a small time skip and continuous space, so the expanding position near the boundaries might be high for multiple time steps. Also just like in Craft, the number of subtasks naturally emerge from the peak patterns. We also provide addition qualitative result on Kitchen released by Gupta et al. (2019b) in Figure 5.21. More results on Kitchen can be found in Appendix C.8.

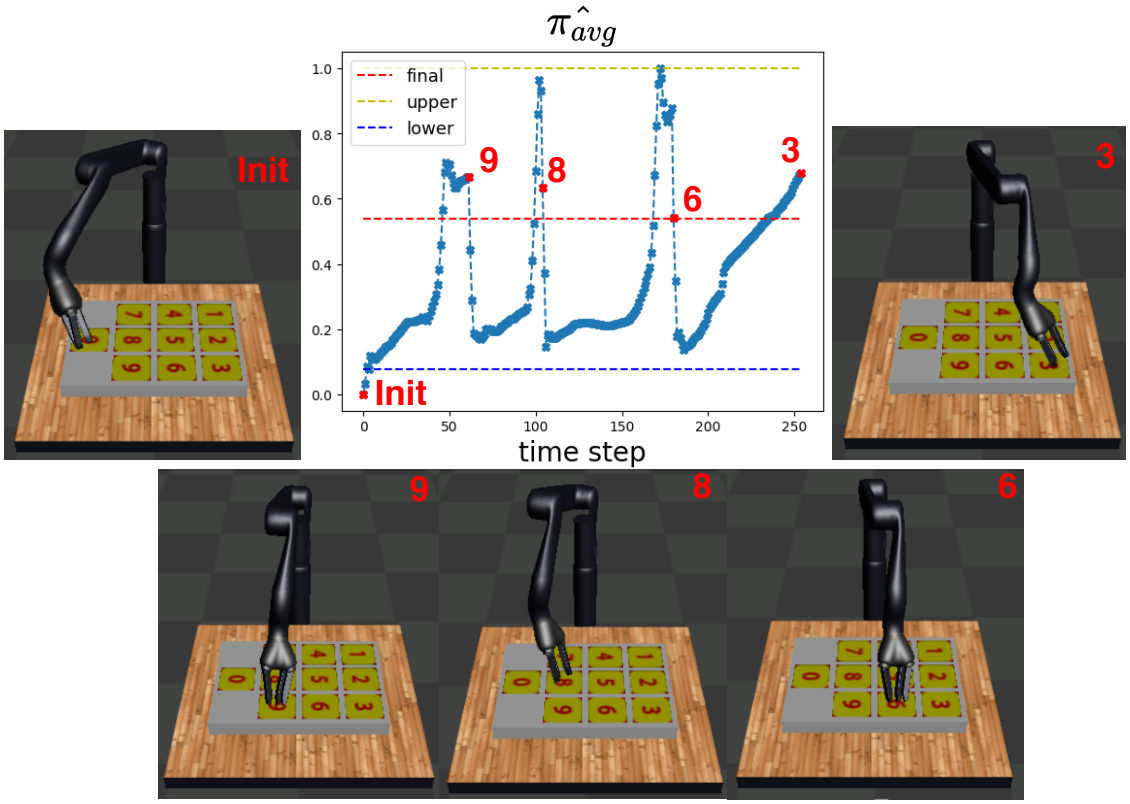


Fig. 5.20. Visualization of the learnt expanding positions of Dial domain. The task is to press the number sequence $[9, 8, 6, 3]$. We plot the $\hat{\pi}_{avg}$. Our threshold selection algorithm produce a upper bound and and lower bound, and the final threshold is computed as the average. The task boundary is detected as the last time step of a segment above the final threshold. The frames at the detected task boundary show that the robot just finishes each subtask.

5.4.4. Ablation Study

In this section, we aim to answer **Q3** and **Q4**. We perform the ablation study as well as some hyper-parameter analysis. The results are in Figure 5.22. We summarize our findings as below:

For *No Bottomup* and *No Topdown*, we remove completely either the bottom-up or the top-bottom process from the model. We find that both hurt the alignment accuracy and removing bottom-up process hurts more. This is expected since bottom-up recurrence updates the subtasks with the latest observations before predicting the termination scores, while the top-down process is more related to outputting actions.

For *No Bottomup Recurr*, we remove the recurrence in the bottom-up process by making $C_i^t = \mathcal{F}(\mathbf{0}, x^t, M_i^{t-1})$ so as to preserve the same number of parameters as OMPN. Although this hurts the alignment accuracy least, the existence of the performance drop confirms

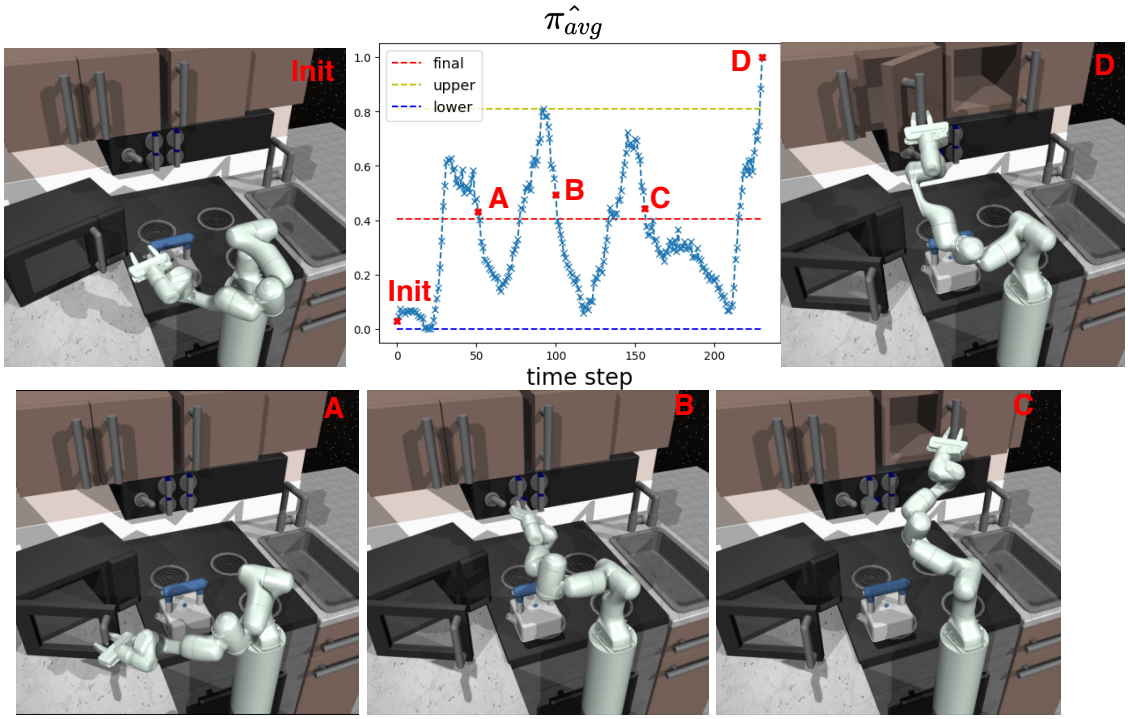
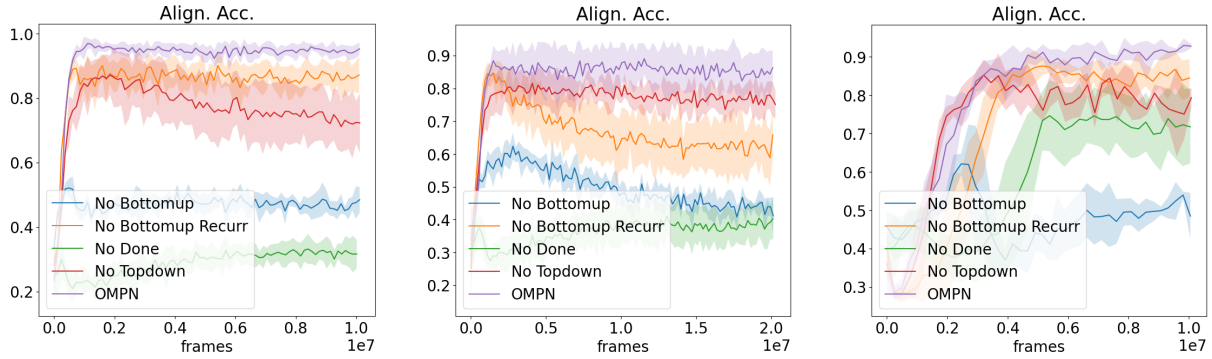


Fig. 5.21. Visualization of the learnt expanding positions of Kitchen. The subtasks are Microwave, Bottom Knob, Hinge Cabinet and Slider. We show the upper bound, lower bound and final threshold produced by our detection algorithm.



(a) NoSketch + Craft(Full) **(b)** NoSketch + Craft(Partial) **(c)** NoSketch + Dial

Fig. 5.22. Ablation study for task alignment accuracy for NoSketch settings

our intuition that the outputs of the lower-level subtasks are beneficial for the higher-level subtasks to predict termination scores, resulting in better task decomposition.

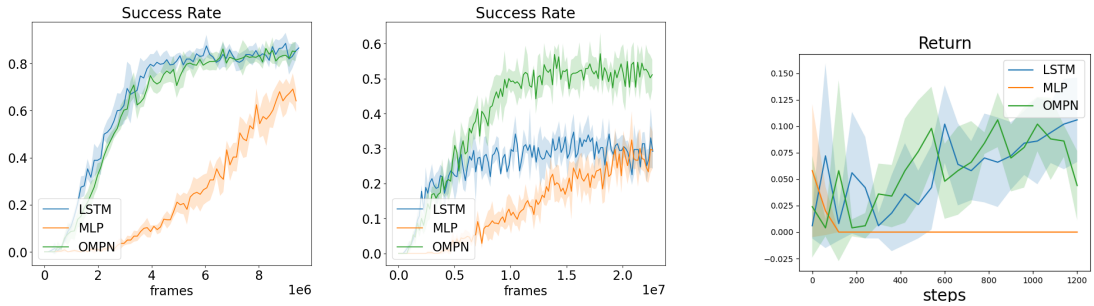
For *No Done*, we use the OMPN architecture but remove the π_{end} from the loss. We find that the model is still able to learn the structure based on the inductive bias to some degree, but the alignment accuracy is much worse.

We also perform hyper-parameter analysis and see its effect on the task decomposition results in Appendix C.7. We find that our task decomposition results is robust to the memory depths n and memory size m in most cases.

5.4.5. Behavior Cloning

We show the behavior cloning results in Figure 5.23 and the full results are in Figure C.39. For Craft, the sketch information is necessary to make the task solvable, so we don't see much difference when there is no sketch information. On the contrary, with full observation and sketch information (Figure 5.23a), this setting might be too easy to show the difference since a memory-less MLP can also achieve almost perfect success rate. As a result, only when the environment moves to a more challenging but still solvable setting with partial observation (Figure 5.23b), OMPN outperform LSTM on the success rate.

For Dial, we find that behaviour cloning alone is not able to solve the task and all of our models never generate the maximum return due to the exposure bias. This is consistent with the previous literature on applying behavior cloning to robotics tasks with continuous states (Laskey et al., 2017). More details can be found in Figure C.42.



(a) Craft Full **(b)** Craft Partial **(c)** Dial

Fig. 5.23. The behavior cloning results when sketch information is provided. For Craft, we define success as the completion of four subtasks. For Dial, the total maximum return is 4.

5.5. Conclusion

In this work, we investigate the problem of learning the subtask hierarchy from the demonstration trajectory. We propose a novel Ordered Memory Policy Network (OMPN) that can represent the subtask structure and leverage it to perform unsupervised task decomposition. Our experiments show that OMPN learns to recover the ground truth subtask boundary in both unsupervised and weakly supervised settings with behavior cloning. In the future, we plan to develop a novel control algorithm based on the inductive bias for faster adaptation to compositional combinations of the subtasks.

Chapter 6

Fourth Article: Using Representation Expressiveness and Learnability to Evaluate Self-Supervised Learning Methods

Prologue

Article Details. **Using Representation Expressiveness and Learnability to Evaluate Self-Supervised Learning Methods.** Yuchen Lu, Zhen Liu, Aristide Baratin, Romain Laroche, Aaron Courville, Alessandro Sordoni. This paper is ready to submit.

Personal contributions. I concieved the main idea, implemented the code, conducted the experiments and wrote the first draft of the paper. Zhen Liu helped with the experiments for some self-supervised learning baseline. Alessandro Sordoni closely follow the weekly experiment progress, provided concrete feedback on codebase, and helped running experiments. The rest of the co-authors participated in the weekly meetings and helped polishing the paper.

Discussion and Recent Developments. The main motivation of this work is mainly motivated under the idea that unsupervised representation learning works like a process of language evolution (Kirby, 2001), where there is a trade-off between expressiveness and learnability. The idea of Cluster Learnability is also inspired by the human study done by Laina et al. (2020). The connection between intrinsic dimension and expressiveness is also inspired by the connection between intrinsic dimension and the entropy as is shown in Levina and Bickel (2004).

Abstract

We address the problem of evaluating the quality of self-supervised learning (SSL) models without access to supervised labels, while being agnostic to the architecture, learning algorithm or data manipulation used during training. We argue that representations can be evaluated through the lens of *expressiveness* and *learnability*. We propose to use the Intrinsic Dimension (ID) to assess expressiveness and introduce Cluster Learnability (CL) to assess learnability. CL is measured in terms of the performance of a KNN classifier trained to predict labels obtained by clustering the representations with K-means. We thus combine CL and ID into a single predictor – CLID. Through a large-scale empirical study with a diverse family of SSL algorithms, we find that CLID better correlates with in-distribution model performance than other competing recent evaluation schemes. We also benchmark CLID on out-of-domain generalization, where CLID serves as a predictor of the transfer performance of SSL models on several visual classification tasks, yielding improvements with respect to the competing baselines.

6.1. Introduction

Despite impressive recent progress in self-supervised learning (SSL) (Chen et al., 2020a; Caron et al., 2021; Grill et al., 2020; Caron et al., 2020, 2018; He et al., 2020b; Chen and He, 2021), the problem of properly evaluating the quality of the learned representations *without using labelled data* has not been fully explored. This is an important problem: solving it could give us a practical tool to choose which SSL model to use when downstream task labels are not accessible, or when the costs of fine-tuning every model to choose the best one are prohibitive. It can also shed light on how these methods work.

In this paper, we approach it by drawing an analogy between unsupervised learning and the evolution of human language. It has been suggested in the language evolution literature (see e.g., Smith et al., 2013) that linguistic structure results from competing pressures for expressiveness (to discriminate objects and concepts of the world) and learnability (to be transmitted across generations). Here, we take the view that a similar guiding principle may be applied to representations of natural images. Just as a language associates objects and scenes with utterances, SSL algorithms do it with continuous vectors to images. We hypothesize that *a good representation should not only cover the diverse visual concepts in the datasets, but also compress them in a manner that could be efficiently learned*. Our goal is to test this hypothesis by (*i*) proposing tractable metrics to assess *expressiveness* and *learnability*, (*ii*) designing performance predictors based on these, and (*iii*) testing these predictors through a large-scale empirical study of a diverse class of SSL methods.

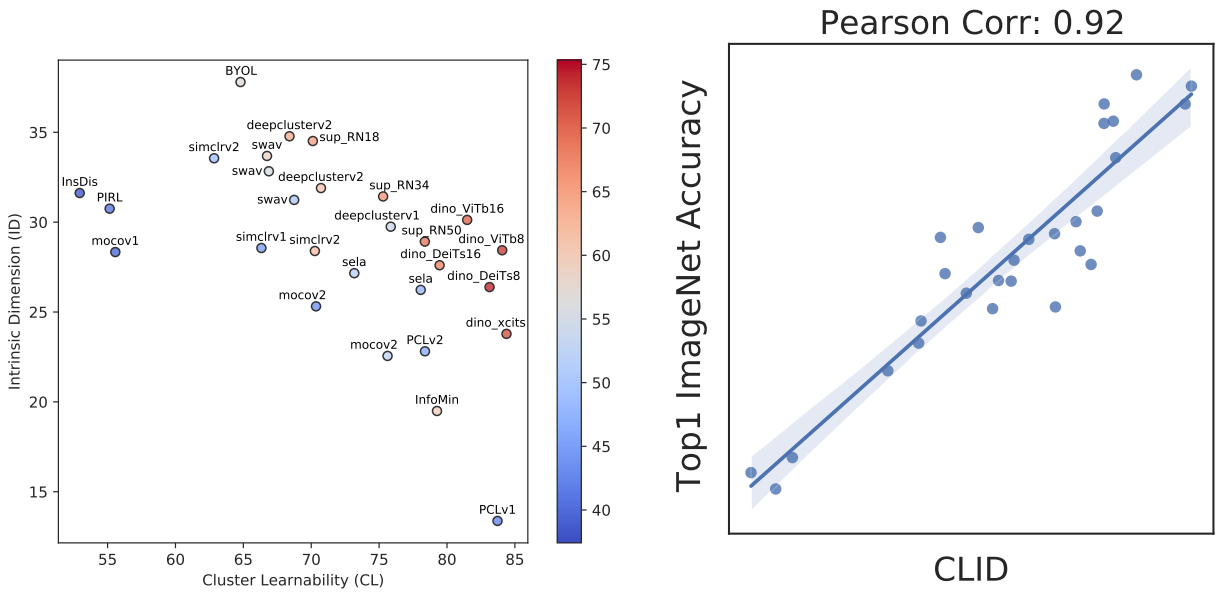


Fig. 6.24. We propose to use Cluster Learnability (CL) to measure learnability and Intrinsic Dimension (ID) for expressiveness. **Left**: Each circle is a SSL pre-trained checkpoint, and the color is used to show its KNN Top1 ImageNet accuracy. Red indicates high accuracy, while blue indicates low accuracy. Good self-supervised learning representations are learnable and expressive, distributed over the upper-right portion of the graph. **Right**: Our predictor is highly correlated with ImageNet performance, even without access to gold labels and by staying agnostic to the model's architecture or training algorithm.

Concretely, we propose to quantify expressiveness via an estimator of the intrinsic dimension (ID) of the data representations (Facco et al., 2017; Ansuini et al., 2019). To assess learnability, we take inspiration from Laina et al. (2020) and propose a novel method called Cluster Learnability (CL), based on the performance of a KNN learner trained to predict labels induced by clustering held-out representations with K-Means. We show that a combination of CL and ID, dubbed CLID, correlates with the model performance across different architectures (see Figure 6.24), better than existing evaluation scheme baselines (Wang and Isola, 2020; Reed et al., 2021; Yu et al., 2020b). To further demonstrate the usefulness of our framework, we conduct out-of-domain generalization prediction experiments on seven downstream transfer tasks. We find that the proposed CLID predictor outperforms baseline concurrent methods at predicting transfer performance.

Our contributions are summarized below:

- We propose an evaluation framework for self-supervised learning relying on expressiveness and learnability, yielding new insights into existing techniques in the literature.
- We propose tractable and generic metrics to quantify *(i)* expressiveness via the Intrinsic Dimension (ID) and *(ii)* learnability via our novel Cluster Learnability (CL) metric.

- We show that CLID predictors are well correlated with Top-1 KNN classification accuracies on ImageNet, and are robust with respect to the choice of hyperparameters.
- We show that CLID predictors predict the transfer performance of pretrained SSL models on several visual classification task, especially when used jointly with in-domain accuracies.

6.2. Learnability and Expressiveness

In this section, we briefly discuss existing evaluation schemes of self-supervised learning methods, which will be used as baselines in our experiments. We then describe our own evaluation scheme, based on specific metrics to quantify learnability and expressiveness.

6.2.1. Notation

We consider the following setting: we assume we have an unlabelled dataset $\{x_i\}_{i=1}^N$ represented in $\mathcal{X} \subset \mathbb{R}^d$, where $x_i \sim \mathcal{P}$ are sampled i.i.d. from some distribution $\mathcal{P}$ over $\mathcal{X}$. We also consider *representation maps* $\mathcal{F} : \mathcal{X} \to \mathbb{R}^m$, typically pretrained neural networks, which represent any input as an m-dimensional vector. Any such map forms a representation dataset $Z = \{z_i\}_{i=1}^N$ where $z_i = \mathcal{F}(x_i)$.

6.2.2. Existing SSL Evaluation

Alignment and Uniformity (Wang and Isola, 2020) decomposes contrastive learning loss and proposes to understand SSL with alignment and uniformity. Formally, they introduce two metrics,

$$\begin{aligned} \mathcal{L}_{align} &:= \mathbb{E}_{x\sim\mathcal{P},x'\sim\mathcal{P}_{\text{aug}}(\cdot|x)} \|\mathcal{F}(x) - \mathcal{F}(x')\|_2^\alpha, \quad \alpha > 0 \\ \mathcal{L}_{unif} &:= \log \mathbb{E}_{x_1,x_2\sim\mathcal{P}} \left[e^{-t\|\hat{\mathcal{F}}(x_1)-\hat{\mathcal{F}}(x_2)\|_2^2)} \right], \quad t > 0 \end{aligned} \tag{6.2.1}$$

where $\mathcal{P}_{\text{aug}}$ is the conditional distribution defined by the data augmentation procedure, $\hat{\mathcal{F}}(x) := \mathcal{F}(x)/\|\mathcal{F}(x)\|$ are the normalized features and t, α are tunable parameters. These two metrics respectively correspond to the intuition of learnability and expressiveness. By making the feature of positive pairs to be similar, the minimization of $\mathcal{L}_{align}$ induces learnable representations that can generalize from one positive example to another. Minimizing the other component $\mathcal{L}_{unif}$ ensures that the (normalized) features are distributed over a hyper-sphere, so that the representations can cover more concepts and achieve high expressiveness.

Maximal Coding Rate Reduction (MCR2) (Yu et al., 2020b) measures the reduction of the average coding length per sample, a.k.a. the coding rate, of a representation,

induced by the knowledge of some category structure (e.g., the membership of the samples in different classes). The goal of this approach is to learn representations that discriminates between classes while being maximally diverse MCR2 corresponds to the intuition of learning a diverse and yet compressible (w.r.t. the class partition) representations.

Maximizing mutual information between inputs and representations (Oord et al., 2018; Bachman et al., 2019a) can be viewed as increasing the expressiveness of the representations. However, as pointed out by Tschannen et al. (2020), MI is not a good predictor of the model performance. Our view is that this is because MI alone does not cover take into account the learnability of the representations. Similar arguments can be applied to pretext tasks evaluation scheme like rotation prediction and solving jigsaw puzzles (Reed et al., 2021).

6.2.3. Our Proposal

Here we describe our evaluation scheme under the view of learnability and expressiveness. Our proposed metrics can be efficiently evaluated on the validation set of the dataset[1].

6.2.3.1. Intrinsic Dimension (ID). We propose to use the notion of intrinsic dimensionality (ID) of the data in the representation space (Pettis et al., 1979a) to quantify expressiveness. Our intuition is that, as more and more fine-grained categories emerge in the representation space, we expect the manifold complexity to increase. Intrinsic dimension is a way to quantify this complexity, characterized as the number of parameters needed to describe the representation manifold without loss of information.

Inferring the intrinsic dimension of a highly nonlinear manifold is a challenging problem (e.g., Levina and Bickel, 2005). In this work, we leverage the nearest neighbor-based method of Facco et al. (2017) to estimate ID[2]. This estimator (TwoNN) is shown to be reliable with respect to representation dimensions and scalable to real-world datasets with deep neural networks (Ansuini et al., 2019). The concept of intrinsic dimension is also connected to entropy (we discuss this point in detail in Appendix D.2).

Formally, given $z_i \in Z$ and an integer $k \geq 1$, we denote by $r_{ik} = D(z_i, NN(z_i, k))$ the distance[3] of z_i to its k-th nearest neighbor $NN(z_i, k)$. Assuming that the points are sampled on a manifold with intrinsic dimension d, it can be shown that, under the assumption of

[1]We have also experimented with metrics computed on the train set but we observed no significant difference in the results.
[2]We experiment with the ID estimator based on the maximum likelihood estimation (Levina and Bickel, 2004; Ma et al., 2018), without noticing a difference in performance.
[3]While we generally consider nearest neighboors w.r.t Euclidean distance, in practice we will also use the cosine distance function, $D(z_1,z_2) = 2 - 2\cos(z_1,z_2)$. Both produce similar results in our experiments.

local uniformity[4], the ratio of distances $\mu_i := r_{i2}/r_{i1}, 1 \leq i \leq N$, follow a Pareto distribution with parameter $d+1$ on $[1, \infty)$, i.e., $\mu_i \sim P(\mu|d) := d\mu^{-(d+1)}$. While d can be computed by maximizing the likelihood, we follow a much simpler method proposed by Facco et al. (2017) based on the cumulative distribution $F(\mu) = 1 - \mu^{-d}$ associated with $P(\mu|d)$. The idea is to estimate d with a linear regression on the empirical cumulate of the distribution. Specifically, assuming we sort μ_i ascendingly, that is $\mu_1 \leq \mu_2 \leq ... \leq \mu_N$, we estimate the cumulative distribution as $F_i^{emp} = i/N$ and fit a straight line on the datasets $\{(\log \mu_i, -\log(1-F_i^{emp}))\}_{i=1}^N$ in the two dimensional plane. The slope is the estimated ID.

6.2.3.2. Cluster Learnability (CL). Let $\{z_i, \tilde{y}_i\}_{i=1}^N$ be the labelled dataset obtained by clustering the representation (e.g., with K-means). We define the *learnability* of the representation as the performance of a classifier on this labelled dataset. For practical purpose, we choose KNN classifier due to its efficiency. We re-use the dataset split to assess the performance of a KNN classifier on this labelled dataset. Let $\hat{y} = KNN(z; \{z^{train}, \tilde{y}^{train}\})$ be the prediction of z after seeing the training data. Learnability is defined as:

$$CL = \frac{1}{N} \sum_{i=1}^{N} [\hat{y}_i^{val} == \tilde{y}_i^{val}] \tag{6.2.2}$$

6.2.3.3. CLID Predictor. We apply our CL and ID in the context of predicting models performance ranking. As a result, for each checkpoint model, we combine the computed CL and ID values into a single numeric predictor, which can be used to rank the models. In the case of without any image labels, we simply adding up these values after standardization

$$\text{CLID}: \quad CL + ID$$

Furthermore, if we have access to the in-domain performance of the models, we can learn a weighted sum

$$\text{W-CLID}: \quad \mathbf{w}^\top [CL, ID]$$

where $\mathbf{w} \in \mathbb{R}^2$ is a weight that can be learned by linear regression on the in-domain performances. This predictor is mainly used when predicting out-of-domain performances.

[4]While the original derivation (Facco et al., 2017) assumes global uniformity, a post-hoc analysis shows that it only needs local uniformity up to the second neighbor.

6.3. Empirical Study

6.3.1. Setup

We select in total 28 self-supervised learning checkpoints trained on ImageNet over different algorithms, architecture, and training epochs. A complete list can be found in Table D.19 in the appendix. We use the KNN evaluation on the validation data using the ground-truth labels to measure the performance of the model, which has been shown to be well correlated with the linear evaluation but computationally less expensive (Caron et al., 2021).

For the computation of cluster learnability, we choose the square root of the dataset size as the number of clusters in Kmeans. We report results with 1 neighbor for our KNN learner. We also show that our results are robust to other choices of hyper-parameters. We normalize the features and use cosine distance for the K-means clustering and KNN learner[5]. Other configurations of cluster numbers and neighbor numbers are also explored in section 6.3.4. All our experiments are computed on a single V100 GPU.

6.3.2. Baselines

Wang and Isola (2020) also proposes to predict the ImageNet performance of pre-trained SSL checkpoints as an evaluation scheme. Therefore, in our experiments, we follow the official implementation[6] with $\alpha = 2$ and $t = 2$ as default values for the tunable parameters in Eqn 6.2.1. We additionally define $-\mathcal{L}_{contrast} = -\mathcal{L}_{align} - \mathcal{L}_{unif}$ to compute the predictor for this method, which reduces to the negative contrastive loss. We add a negative sign, since we require a predictor to be in proportional to the model performance.

The original MCR2 (Yu et al., 2020b) requires a class partition to be pre-specified. In order to adapt it into an unsupervised evaluation scheme, we use a K-means clustering as the dataset partition. We follow the default settings[7] and we normalize the features. We also experiment with using the ground-truth label as the dataset partition, but we found no improvements. We use the coding rate reduction ΔR (see Yu et al., 2020b, Equation 6) to be maximized as a predictor.

For the Mutual Information baseline, we use MINE (Belghazi et al., 2018) with a fixed student ResNet18 network (He et al., 2016) to estimate the mutual information between inputs and representation. We use a batch size 128, learning rate 0.0005 and weight decay 0.001. The network is trained for 50000 steps on the training images, and we report MINE

[5]We find similar results when using L2 distance.
[6]See https://github.com/SsnL/moco_align_uniform
[7]See https://github.com/ryanchankh/mcr2

Table 6.4. Correlation results between ImageNet performances and different predictors. We compute both Pearson ρ and Kendall τ. Our CLID predictors achieve the highest correlation.

Predictors	Pearson ρ	Kendall τ
$-\mathcal{L}_{align}$	0.42	0.26
$-\mathcal{L}_{unif}$	-0.05	0.03
$\mathcal{L}_{contrast}$	0.37	0.24
ΔR	-0.62	-0.33
MI	0.13	0.08
Pretext	0.61	0.27
Ours		
CL	0.74	0.44
ID	0.12	0.09
CLID	**0.92**	**0.75**

on the validation data. We find that training longer is computational intensive, while the results are similar.

For the pretext task, we experiment with rotation prediction by constructing a 4-way classification. We randomly rotate the training images by $0, 90, 180, 270$ degrees, train a KNN classifier on the training images to predict a 4-way classification, and then report the rotation prediction accuracy on the validation images. This is shown to be an effective evaluation in both self-supervised learning (Reed et al., 2021) and architecture search (Liu et al., 2020).

6.3.3. Is CLID scheme correlated with ImageNet performance?

We first investigate whether the proposed evaluation scheme is useful for in-domain (ImageNet) generalization. We perform both qualitative and a quantitative examination and show the results in Figure 6.24. On the left, we find that the self-supervised learning checkpoints with higher ImageNet accuracies tend to be both more learnable and more expressive (e.g., in the upper-right corner of the graph). In the meantime, we find that methods favouring only of these qualities at the expanse of the other, like PCLv1 and PIRL, also have poor ImageNet accuracies. In Figure 6.24 middle and right, we compute the correlation of our CLID predictors with respect to the ImageNet accuracy of the model considered and show that we achieve a Pearson ρ of 0.92 for CLID.

In Table 6.4, we compare our results with the baselines. The regression plots for the baselines can be found in Figure D.49. Our proposed predictors achieve the highest correlation both in terms of Pearson ρ and Kendall τ coefficient. We find that $\mathcal{L}_{contrast}$ achieves a relatively low correlation $\rho = 0.37$, which apparently contradicts the observation made in Wang and Isola (2020). We hypothesize that this is due to the fact that their analysis is restricted to SSL

checkpoints trained with contrastive learning, and therefore $\mathcal{L}_{contrast}$ might not be general enough to characterize representations obtained by alternative approaches. Additionally, the results in Wang and Isola (2020) are computed on the representations used to compute the noise-contrastive objective, which are usually transformed by a final projection layer that is not always present in other SSL methods. This limits its generality. Interestingly, we find that MCR2 (ΔR) gives negative correlation which warrants further investigation. While the MI baseline only achieves $\rho = 0.13$, the pretext task baseline is still surprisingly good with a $\rho = 0.61$, suggesting that the rotation prediction task remains a simple but effective baseline.

6.3.4. Is CLID sensitive to its hyper-parameters?

In this section, we check the sensitivity of CLID to its hyperparameters. Since ID does not have any hyper-parameters, we mainly examine the following hyper-parameters for CL: the number of clusters and the number of neighbors used in the KNN Learner.

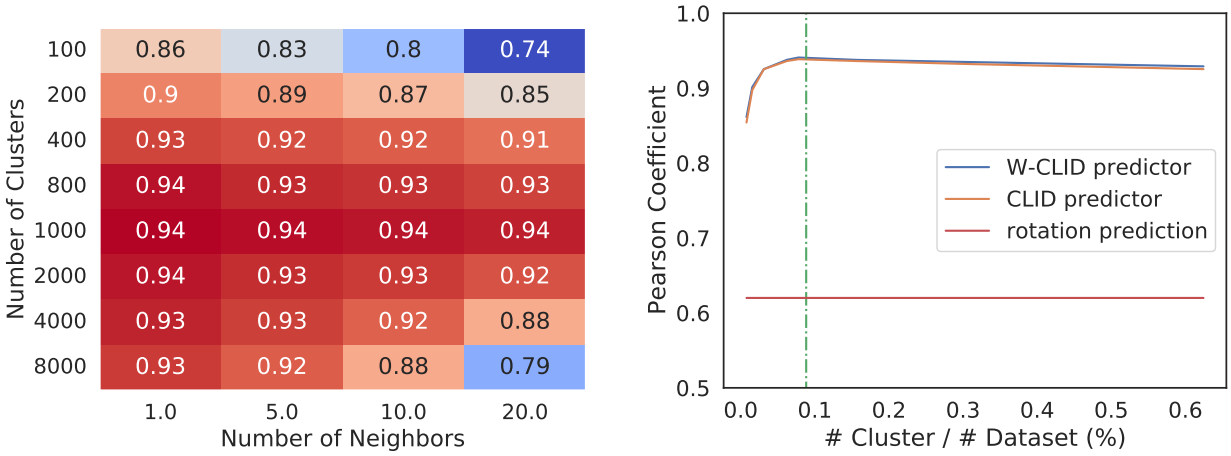


Fig. 6.25. Robustness Analysis for ImageNet. **Left**: The heat map of Pearson Coefficient between the Top-1 accuracy and CLID predictor. **Right**: Pearson coefficient vs. the ratio between the number of clusters and the dataset size when using one neighbor. The result is stable with a reasonably large number of clusters, e.g., the square-root of the dataset size **(Green Dashed Line)**.

The results can be found in Figure 6.25. The resulting predictors can still produce a higher correlation than the rotation prediction for a wide range of the configurations we tried. We observe that decreasing the number of neighbors leads to a better predictor. Given our result, we recommend setting this number to be 1.

We observe that there is an optimal cluster number so that the choice of neighbor numbers can be more flexible. If the cluster number is too low, the results become worse. Recall that for different checkpoints, the clusters are produced with the same K-means algorithm, so the separability among clusters might be roughly controlled. As a result, we hypothesize that the

Table 6.5. Kendall Ranking Coefficient results on Out-of-Domain Tasks when ImageNet labels are not available. The CLID predictor has the best predictive performance for transfer tasks. Full results in Table D.20, including results for the few-shot experiments.

	CLID	$-\mathcal{L}_{contrast}$	ΔR	MI	Pretext
foods	**0.75**	0.17	-0.35	0.01	0.51
flowers	**0.81**	0.25	-0.35	0.14	0.46
pets	**0.71**	0.23	-0.25	-0.08	0.54
caltech101	0.56	0.11	-0.18	-0.05	**0.67**
stl10	**0.56**	0.02	-0.34	-0.21	0.41
aircraft	**0.67**	0.21	-0.27	0.16	0.57
cars	**0.81**	0.32	-0.28	0.16	0.48
Avg	**0.70**	0.19	-0.29	0.02	0.52

Table 6.6. Kendall Ranking Coefficient results on Out-of-Domain Tasks when ImageNet labels are obtained. The results are computed with the *joint rank product* between the predictors and the ImageNet accuracy. While the ImageNet accuracy can predict the transfer accuracy reasonably, the proposed CLID predictor further enhances the predictive performance. Full results in Table D.21, which include results with few-shot experiments.

	Imagenet	CLID	W-CLID	Source Ent	Target Ent	$L_{contrast}$	Pretext
foods	0.86	0.84	0.81	**0.88**	0.84	0.56	0.75
flowers	0.70	**0.79**	0.74	0.74	0.71	0.58	0.64
pets	0.75	0.77	**0.78**	0.66	0.72	0.55	0.76
caltech101	0.62	0.59	0.58	0.60	0.66	0.41	**0.77**
stl10	**0.76**	0.71	0.71	0.61	0.71	0.36	0.64
aircraft	0.60	0.66	0.61	**0.68**	0.64	0.50	0.62
cars	0.68	**0.79**	0.74	0.72	0.74	0.64	0.66
Avg	0.71	**0.74**	0.71	0.70	0.72	0.52	0.69

number of clusters might control the underlying difficulty of the classification problem faced by the KNN learner. In the extreme case, we either have 1 cluster or as many clusters as data points. In the former, KNN accuracy would always be 100% and in the latter always be close to 0%[8]. As a result, neither of them would be suitable enough to distinguish the collected checkpoints. As a rule of thumb, we recommend using a reasonably large number of clusters, e.g., the square root of the dataset size.

6.3.5. Can CLID be used to predict transfer performance?

Here we investigate whether CLID can be a good predictor of the performance on downstream tasks. We collect 7 out-of-domain downstream visual classification tasks. For

[8]To be specific it's $1/N$, where N is the dataset size. It's close to 0% when N is large.

each domain, we compare the ranking of our SSL checkpoints induced by CLID, and the ranking induced by the actual test performance on that domain. We report the Kendall τ correlation score, which is consistent with the existing benchmarks on out-of-domain generalization (Vedantam et al., 2021).

We first investigate the scenario where the ImageNet labels are not accessible (Table 6.5). This is an extension of our previous scenario, and it is also a realistic assumption especially when the models are pretrained on web-scale data without clear annotations. We find that our CLID predictor is the best, even beating the W-CLID predictor. One reason is because the linear coefficients $\mathbf{w}$ are obtained from regressing in the ImageNet domain, which might increase overfitting and hurt transfer. We also find that $-\mathcal{L}_{contrast}$ is not a good predictor, and a simple pretext task like rotation prediction already has reasonable performance even in transfer settings.

In the second setting, we assume to have access to ImageNet labels (Table 6.6) and explore whether transfer performance prediction can be improved using this additional information. We find that the ImageNet accuracies have pretty high correlation with the downstream tasks, with an average Kendall coefficients of 0.71. To integrate ImageNet performance information to each our competing predictors, we proceed as follows. Let r^i_{pred} be the ranking of i-th checkpoint from the predictor, and r^i_{img} be that from the ImageNet accuracy. We compute a joint ranking by computing the rank product, which is the geometric mean of these two ranks:

$$r^i_{joint} = \sqrt{r^i_{pred} r^i_{img}}$$

In addition to comparing with previously presented baselines, we also add the strongest baselines from (Vedantam et al., 2021): we train a linear classifier and measure the negative label entropy $-H(Y|X)$. Since the entropy can be measured on either source domain or target domain, we denote each variant as "Source Ent" and "Target Ent". The underlying intuition is that, if the model has more confidence in the prediction (lower entropy), it should generalize better.

We find that the proposed CLID predictor can further enhance the ImageNet accuracies predictions, with an average Kendall coefficients of 0.74 for the joint ranking. We find that "Target Ent." and "Source Ent." both have a reasonable performance, which is consistent with observation in Vedantam et al. (2021), but they underperform our predictor. Note that computing "Target Ent." and "Source Ent." requires training an extra linear layer, which increases their computational requirements.

6.4. Related Work

6.4.1. Representation Evaluation

Several recent works address the question of representation evaluation in self-supervised learning. Whitney et al. (2021) propose to use the learning dynamics of the downstream classifiers to measure the representation complexity. However their method still depends on extra human labels. Pretext tasks like jigsaw or rotation prediction are shown to be well correlated with the self-supervised evaluation (Reed et al., 2021; Deng and Zheng, 2021) and architecture search (Liu et al., 2020). However, they also rely on crafting ad-hoc data augmentations. Ericsson et al. (2021) argues that it is important to look at the transfer performance of the SSL models in order to judge its quality. We agree with this statement and therefore also test our method for out-of-domain generalization settings (section 6.3.5). Closely related to our work, a very recent paper Garrido et al. (2022) proposes to evaluate joint-embedding self-supervised methods without access to supervised labels, by means of the effective rank of the learned embedding matrix.

The concepts of learnability and expressiveness, while not being explicitly mentioned, can be found in existing literature. For example, many SSL strategies emphasize maximizing the mutual information between inputs and representations (Vincent et al., 2008; Higgins et al., 2017; Oord et al., 2018; Bachman et al., 2019b). These can be viewed as attempts to increase expressiveness, since high mutual information leads to correspondence between inputs and representations, and thus the visual concepts among the inputs are mapped to the representation space. Recent works also pay attention to the emerging properties of learnability in the cluster structure from SSL models through human studies (Laina et al., 2020). The emphasis of learnability can be also found in the recent attempt to design a compression regularizor called Conditional Entropy Bottleneck (Lee et al., 2021). Finally, the intuition of a trade-off between learnability and expressiveness underpins other representation analysis frameworks, such as alignment and uniformity (Wang and Isola, 2020) or Maximal Coding Rate Reduction (Yu et al., 2020b), which are further explained in Section 5.2. In this paper, we aim at turning this intuition into a practical self-supervised evaluation tool, especially on predicting performances in out-of-distribution transfer tasks.

6.4.2. Learnability, Ease-of-Transmission and Compression

Learnability has been argued to be a hallmark of the human language in order to be effortlessly transmitted through generations (Kirby et al., 2014; Rafferty et al., 2011; Beckner et al., 2017; Zhou and Yurovsky, 2021; Kampen, 2004), and it is also true for visual concepts

like color (Xu et al., 2010), categories (Griffiths et al., 2006), shapes (Portelance et al., 2021) etc. In deep learning, it has been explored in the context of emergent communication (Ren et al., 2020; Guo et al., 2019; Li and Bowling, 2019), language drift (Lu et al., 2020), and neural module networks (Vani et al., 2021), but it is less explored for vision representation learning, except for a human study on just two SSL methods (Laina et al., 2020). While Lee et al. (2021) also investigates a regularizor for compressive self-supervised learning, it is unclear whether such notion is useful as a representation evaluation framework. Learnability has a tight connection to compression (Chaitin, 2007) and prequential codelength (Dawid, 1984), which quantifies the compression levels with the online learning error. Existing work (Blier and Ollivier, 2018) has uses it to support the generalization ability of the learner (e.g., deep neural nets) on the dataset (e.g., labeled images). Here, we use this concept to quantify the learnability of the representation, in the sense that if the emerged Kmeans clustering is more learnable, then the same KNN learner could achieve a lower compression bound via prequential coding.

6.4.3. Manifold Intrinsic Dimension

Intrinsic dimension can be thought of as the smallest number of variable needed to approximate the representation manifold. Applying local neighborhood information to estimate the intrinsic dimension is not a new idea, and it was shown to be more efficient than the global eigenvalue approach (Pettis et al., 1979b). Ansuini et al. (2019) apply the TwoNN estimator (Facco et al., 2017) to the non-linear representation manifold of modern deep neural nets. They find that the intrinsic dimension is inversely correlated to the classification accuracy Their work is further extended to confirm that natural images lies in a low-dimension manifold (Pope et al., 2021), and that lower ID datasets leads to better generalization. Recanatesi et al. (2019) further highlights the connections between intrinsic dimension and the generalization properties, by comparing the models before and after supervised training. Intrinsic dimension can also be estimated locally with a Maximum Likelihood Estimator (Levina and Bickel, 2004), which Ma et al. (2018) propose to use as a regularizor against overfitting noisy labels. While the above work mainly focus on ID with supervised learning models, our work on SSL models presents a more nuanced view about ID. It is indeed the case that lower intrinsic dimension can lead to better accuracy, especially among supervised checkpoints (see "sup_RN18", "sup_RN34", "sup_RN50"in Figure 6.24). However, we hypothesize that when learning representation from scratch without labels, e.g., self-supervised learning, the representation manifold still need a certain amount of complexity in order to include enough information from the dataset.

6.5. Limitation and Potential Negative Societal Impacts

6.5.1. Limitations

While our proposed evaluation scheme is designed to be general, a potential limitation is that it might be only suitable to the classification tasks, since we put emphasis on the emergence of learnable cluster structure. As a result, the proposed predictor might not be as useful for other downstream tasks like object detection or segmentation. While our results is true for a population of models, there are also some outliers. E.g., in Figure 6.24 "deepclusterv1" seems to have higher CL and ID than "simclrv2", but its accuracy is lower.

6.5.2. Societal impact

This paper follows a line of work aiming at a better understanding of deep learning algorithms. Even though it does not directly contribute to any specific application, it promotes the development and dissemination of the deep learning technology, which, as any technology, can be used for harmful purposes. Moreover, we acknowledge that deep learning has been proved in the past to potentially reduce or amplify bias.

6.6. Conclusion & Discussion

We propose a unifying view to evaluate self-supervised representation learning through expressiveness and learnability. We propose to estimate expressiveness with intrinsic dimension (ID), and learnability with the acquisition speed of a KNN learner on the K-means clustering of the representation. We show that the proposed CLID evaluation scheme better predicts the ImageNet accuracy than other evaluation schemes. We further demonstrate that our CLID can also predict the transfer task performance.

Our work is a solid step towards understanding the current SSL algorithms and opens up interesting research directions. Future works could further explore the expressiveness-learnability in a more theoretical context, extend this framework to other field like pretrained language models, as well as devise new SSL algorithms that directly maximize intrinsic dimension or cluster learnability.

Chapter 7

Conclusion

In this thesis, we described the results of some preliminary investigations towards about emerging language-like latents in deep neural networks. This perspective could also help us leveraging the insights from the emergent communication works, which have been primarily focused on small-scale signalling game. The papers we demonstrate here are by no means conclusive, but they help illustrate several interesting points around the language-like latents in deep learning in diverse machine learning contexts. To summarize,

- Emerging language-like latents could potentially help us develop models that can achieve systematic generalization, which has been one of the key problems in deep learning research (Chapter 3 and 4).
- It is possible to induce language-like latents with scalable model inductive bias or constraints, such as neural architectural design (Chapter 5).
- It is possible that there are already language-like latents emerging in the current state-of-art methods (Chapter 6).

Our work opens several interesting questions for further investigation. To start with, can we further develop new algorithm or model inductive bias that can help emerge language-like latents? Emerging language-like latents could further provide guidance on neural architectural design. The ability to emerge symbol-like objects is one of the hardest problem for connectionist models (Smolensky, 1991). Our work suggest that developing architectural inductive bias or learning procedure that can scale is a potential solution. This can be viewed as the continuation of the line of research that incorporating linguistic structure into neural networks such as seq2seq models (Shen et al., 2018) and transformers (Shen et al., 2021).

Conversely, one might still ask whether language-like latents can be found more easily as we scale up? Recently, self-supervised learning and large-scale pretrained models (Brown et al., 2020) have shown impressive results. As a result, it is interesting to see whether their intermediate representations would have more and more linguistic properties. This

direction can be purely analytical, and there is preliminary work showing that there are emerging syntactic structure in the patterns of self-attention layers (Manning et al., 2020), as well as emerging semantic segmentation from self-supervised vision transformers (Caron et al., 2021). In the future, it would be interesting to see more works like those exploring whether language-like latents, maybe even some re-usable modules, could appear even in a large-pretrained model, even when they are not expected. The success or failure in finding these language-like latents in big models might also help provide insights for the currently heated debate on whether scaling could lead to truly general intelligence like human beings.

References

Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

Saeed Amizadeh, Hamid Palangi, Alex Polozov, Yichen Huang, and Kazuhito Koishida. Neuro-symbolic visual reasoning: Disentangling. In *International Conference on Machine Learning*, pages 279–290. PMLR, 2020.

Jacob Andreas. Measuring compositionality in representation learning. In *International Conference on Learning Representations*, 2019a. URL https://openreview.net/forum?id=HJz05o0qK7.

Jacob Andreas. Measuring compositionality in representation learning. *arXiv preprint arXiv:1902.07181*, 2019b.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 39–48, 2016.

Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning*, pages 166–175. PMLR, 2017.

Jacob Andreas, Dan Klein, and Sergey Levine. Learning with latent language. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2166–2179, 2018.

Alessio Ansuini, Alessandro Laio, Jakob H Macke, and Davide Zoccolan. Intrinsic dimension of data representations in deep neural networks. *Advances in Neural Information Processing Systems*, 32:6111–6122, 2019.

Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. *arXiv preprint arXiv:1911.05371*, 2019.

Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *Advances in Neural Information Processing Systems*, 32:15535–15545, 2019a.

Philip Bachman, R. Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *NeurIPS*, pages 15509–15519, 2019b.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proc. of of International Conference on Learning Representations*, 2015.

Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. Systematic generalization: what is required and can it be learned? *arXiv preprint arXiv:1811.12889*, 2018.

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEit: BERT pre-training of image transformers. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=p-BhZSz59o4.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

Jasmijn Bastings, Marco Baroni, Jason Weston, Kyunghyun Cho, and Douwe Kiela. Jump to better conclusions: Scan both left and right. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 47–55, 2018.

Clay Beckner, Janet B Pierrehumbert, and Jennifer Hay. The emergence of linguistic structure in an online iterated learning task. *Journal of Language Evolution*, 2(2):160–176, 2017.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *International Conference on Machine Learning*, pages 531–540. PMLR, 2018.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(35): 1798–1828, 2013.

Léonard Blier and Yann Ollivier. The description length of deep learning models. *arXiv preprint arXiv:1802.07044*, 2018.

Antoine Bordes, Y-Lan Boureau, and Jason Weston. Learning end-to-end goal-oriented dialog. In *Proc. of International Conference on Learning Representations*, 2017.

Henry Brighton and Simon Kirby. Understanding linguistic evolution by visualizing the emergence of topographic mappings. *Artificial life*, 12(2):229–242, 2006.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 132–149, 2018.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *arXiv preprint arXiv:2104.14294*, 2021.

Mauro Cettolo, Christian Girardi, and Marcello Federico. Wit3: Web inventory of transcribed and translated talks. In *Proc. of Conference of european association for machine translation*, 2012.

Rahma Chaabouni, Eugene Kharitonov, Emmanuel Dupoux, and Marco Baroni. Anti-efficient encoding in emergent communication. In *Advances in Neural Information Processing Systems*, pages 6290–6300, 2019.

Rahma Chaabouni, Eugene Kharitonov, Diane Bouchacourt, Emmanuel Dupoux, and Marco Baroni. Compositionality and generalization in emergent languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4427–4442, 2020.

Gregory J Chaitin. *Thinking About Godel And Turing: Essays On Complexity, 1970–2007*. World scientific, 2007.

Prithvijit Chattopadhyay, Deshraj Yadav, Viraj Prabhu, Arjun Chandrasekaran, Abhishek Das, Stefan Lee, Dhruv Batra, and Devi Parikh. Evaluating visual conversational agents via cooperative human-ai games. In *Proc. of AAAI Conference on Human Computation and Crowdsourcing*, 2017.

Bernard Chazelle and Chu Wang. Self-sustaining iterated learning. In *Proc. of the Innovations in Theoretical Computer Science Conference*, 2017.

Bernard Chazelle and Chu Wang. Iterated learning in dynamic social networks. *The Journal of Machine Learning Research*, 20(1):979–1006, 2019.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021a.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020a.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in Neural Information Processing Systems*, 33:22243–22255, 2020b.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020c.

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9640–9649, 2021b.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proc. of Empirical Methods in Natural Language Processing*, 2014.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*, 2016.

Michael Cogswell, Jiasen Lu, Stefan Lee, Devi Parikh, and Dhruv Batra. Emergence of compositional language with deep generational transmission. *arXiv preprint arXiv:1904.09067*, 2019.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537, 2011.

Henry Conklin and Kenny Smith. Compositionality with variation reliably emerges in neural networks. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=-Yzz6vlX7V-.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

Gautier Dagan, Dieuwke Hupkes, and Elia Bruni. Co-evolution of language and agents in referential games. *arXiv preprint arXiv:2001.03361*, 2020.

Abhishek Das, Satwik Kottur, José MF Moura, Stefan Lee, and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning. In *Proc. of International Conference on Computer Vision*, 2017.

A Philip Dawid. Present position and potential developments: Some personal views statistical theory the prequential approach. *Journal of the Royal Statistical Society: Series A (General)*, 147(2):278–290, 1984.

Harm De Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron Courville. Guesswhat?! visual object discovery through multi-modal dialogue. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Weijian Deng and Liang Zheng. Are labels always necessary for classifier accuracy evaluation? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15069–15078, 2021.

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

Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. Frames: a corpus for adding memory to goal-oriented dialogue systems. In *Proceedings of the SIGdial Meeting on Discourse and Dialogue*, 2017.

Salah El Hihi and Yoshua Bengio. Hierarchical recurrent neural networks for long-term dependencies. In *Advances in neural information processing systems*, pages 493–499, 1996.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. Multi30k: Multilingual english-german image descriptions. In *Proc. of Workshop on Vision and Language*, 2016.

Linus Ericsson, Henry Gouk, and Timothy M Hospedales. How well do self-supervised models transfer? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5414–5423, 2021.

Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific reports*, 7(1):1–8, 2017.

Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. Vse++: Improving visual-semantic embeddings with hard negatives. *arXiv preprint arXiv:1707.05612*, 2017.

Jerry A Fodor. *The language of thought*, volume 5. Harvard university press, 1975.

Jerry A Fodor and Zenon W Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71, 1988.

Roy Fox, Sanjay Krishnan, Ion Stoica, and Ken Goldberg. Multi-level discovery of deep options. *arXiv preprint arXiv:1703.08294*, 2017.

Roy Fox, Richard Shin, Sanjay Krishnan, Ken Goldberg, Dawn Song, and Ion Stoica. Parametrized hierarchical procedures for neural programming. *ICLR 2018*, 2018.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. Speaker-follower models for vision-and-language navigation. In *Proc. of Neural Information Processing Systems*, 2018.

Jianfeng Gao, Michel Galley, Lihong Li, et al. Neural approaches to conversational ai. *Foundations and Trends in Information Retrieval*, 13(2-3):127–298, 2019.

Quentin Garrido, Randall Balestriero, Laurent Najman, and Yann Lecun. Rankme: Assessing the downstream performance of pretrained self-supervised representations by their rank. *arXiv:2210.02885 [cs.LG]*, 2022.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Anand Gopalakrishnan, Kazuki Irie, Jürgen Schmidhuber, and Sjoerd van Steenkiste. Unsupervised learning of temporal abstractions using slot-based transformers. In *Deep RL Workshop NeurIPS 2021*, 2021.

Thomas L Griffiths and Michael L Kalish. A bayesian view of language evolution by iterated learning. In *Proc. of the Annual Meeting of the Cognitive Science Society*, 2005.

Thomas L Griffiths, Brian R Christian, and Michael L Kalish. Revealing priors on category structures through iterated learning. In *Proceedings of the 28th annual conference of the Cognitive Science Society*, volume 199, 2006.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33, 2020.

Shangmin Guo, Yi Ren, Serhii Havrylov, Stella Frank, Ivan Titov, and Kenny Smith. The emergence of compositional languages for numeric concepts through iterated learning in neural agents. *arXiv preprint arXiv:1910.05291*, 2019.

Abhinav Gupta, Ryan Lowe, Jakob Foerster, Douwe Kiela, and Joelle Pineau. Seeded self-play for language learning. In *Proc. of Beyond Vision and LANguage: inTEgrating Real-world kNowledge (LANTERN)*, 2019a.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019b.

Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. Neural module networks for reasoning over text. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SygWvAVFPr.

Laura Harding Graesser, Kyunghyun Cho, and Douwe Kiela. Emergent linguistic phenomena in multi-agent communication games. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3700–3710, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1384. URL https://aclanthology.org/D19-1384.

Serhii Havrylov and Ivan Titov. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. In *Proc. of Neural Information Processing Systems*, 2017.

Patrick J Hayes. The second naive physics manifesto. *Formal theories of the common sense world*, 1988.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. Revisiting self-training for neural sequence generation. In *Proc. of International Conference on Learning Representations*, 2020a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020b.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

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

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *Proc. of International Conference on Learning Representations*, 2017a.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparametrization with gumble-softmax. In *International Conference on Learning Representations (ICLR)*, 2017b.

Devon Jarvis, Richard Klein, Benjamin Rosman, and Andrew M Saxe. The role of learning regime, architecture and dataset structure on systematic generalization in simple neural networks, 2022. URL https://openreview.net/forum?id=3r034NfDKnL.

Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *EMNLP*, 2017.

Michael L Kalish, Thomas L Griffiths, and Stephan Lewandowsky. Iterated learning: Intergenerational knowledge transmission reveals inductive biases. *Psychonomic Bulletin & Review*, 14(2):288–294, 2007.

Jacqueline van Kampen. The learnability of syntactic categories. *LOT Occasional Series*, 3: 245–256, 2004.

Eugene Kharitonov and Marco Baroni. Emergent language generalization and acquisition speed are not tied to compositionality. *arXiv preprint arXiv:2004.03420*, 2020.

Eugene Kharitonov, Rahma Chaabouni, Diane Bouchacourt, and Marco Baroni. Entropy minimization in emergent languages. In *International Conference on Machine Learning*, pages 5220–5230. PMLR, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Thomas Kipf, Yujia Li, Hanjun Dai, Vinicius Zambaldi, Alvaro Sanchez-Gonzalez, Edward Grefenstette, Pushmeet Kohli, and Peter Battaglia. Compile: Compositional imitation learning and execution. In *International Conference on Machine Learning*, pages 3418–3428. PMLR, 2019.

Simon Kirby. Spontaneous evolution of linguistic structure-an iterated learning model of the emergence of regularity and irregularity. *IEEE Transactions on Evolutionary Computation*, 5(2):102–110, 2001.

Simon Kirby. Natural language from artificial life. *Artificial life*, 8(2):185–215, 2002.

Simon Kirby, Tom Griffiths, and Kenny Smith. Iterated learning and the evolution of language. *Current opinion in neurobiology*, 28:108–114, 2014.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, pages 177–180, 2007.

Satwik Kottur, José MF Moura, Stefan Lee, and Dhruv Batra. Natural language does not emerge 'naturally' in multi-agent dialog. In *Proc. of Empirical Methods in Natural Language Processing*, 2017.

Jan Koutnik, Klaus Greff, Faustino Gomez, and Juergen Schmidhuber. A clockwork rnn. In *International Conference on Machine Learning*, pages 1863–1871, 2014.

Sanjay Krishnan, Roy Fox, Ion Stoica, and Ken Goldberg. Ddco: Discovery of deep continuous options for robot learning from demonstrations. *arXiv preprint arXiv:1710.05421*, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Iro Laina, Ruth Fong, and Andrea Vedaldi. Quantifying learnability and describability of visual concepts emerging in representation learning. *Advances in Neural Information*

Processing Systems, 33, 2020.

Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pages 2873–2882. PMLR, 2018.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Unsupervised machine translation using monolingual corpora only. *Proc. of Internation Conference on Learning Representations*, 2018.

Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 577–593. Springer, 2016.

Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Colorization as a proxy task for visual understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6874–6883, 2017.

Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan, and Ken Goldberg. Dart: Noise injection for robust imitation learning. *arXiv preprint arXiv:1703.09327*, 2017.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. In *Proc. of Internation Conference on Learning Representations*, 2016.

Angeliki Lazaridou, Karl Moritz Hermann, Karl Tuyls, and Stephen Clark. Emergence of linguistic communication from referential games with symbolic and pixel input. *arXiv preprint arXiv:1804.03984*, 2018.

Angeliki Lazaridou, Anna Potapenko, and Olivier Tieleman. Multi-agent communication meets natural language: Synergies between functional and structural language learning. In *Proc. of the Association for Computational Linguistics*, 2020.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Jason Lee, Kyunghyun Cho, and Douwe Kiela. Countering language drift via visual grounding. In *Proc. of Empirical Methods in Natural Language Processing*, 2019.

Kuang-Huei Lee, Anurag Arnab, Sergio Guadarrama, John Canny, and Ian Fischer. Compressive visual representations. *Advances in Neural Information Processing Systems*, 34, 2021.

Sang-Hyun Lee. Learning compound tasks without task-specific knowledge via imitation and self-supervised learning. In *International Conference on Machine Learning*, 2020.

Oliver Lemon and Olivier Pietquin. *Data-driven methods for adaptive spoken dialogue systems: Computational learning for conversational interfaces*. Springer Science & Business Media, 2012.

Olivier Lemon and Olivier Pietquin. Machine learning for spoken dialogue systems. In *Proceedings of European Conference on Speech Communication and Technologies (Interspeech)*, 2007.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on speech and audio processing*, 8(1):11–23, 2000.

Elizaveta Levina and Peter Bickel. Maximum likelihood estimation of intrinsic dimension. *Advances in neural information processing systems*, 17, 2004.

Elizaveta Levina and Peter Bickel. Maximum likelihood estimation of intrinsic dimension. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2005. URL https://proceedings.neurips.cc/paper/2004/file/74934548253bcab8490ebd74afed7031-Paper.pdf.

David K. Lewis. *Convention: A Philosophical Study*. Wiley-Blackwell, 1969.

Mike Lewis, Denis Yarats, Yann Dauphin, Devi Parikh, and Dhruv Batra. Deal or no deal? end-to-end learning of negotiation dialogues. In *Proc. of Empirical Methods in Natural Language Processing*, pages 2443–2453, 2017.

Fushan Li and Michael Bowling. Ease-of-teaching and language structure from emergent communication. *Advances in neural information processing systems*, 32, 2019.

Jiwei Li, Alexander H Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. Dialogue learning with human-in-the-loop. In *Proc. of International Conference on Learning Representations*, 2016a.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. Deep reinforcement learning for dialogue generation. In *Proc. of Empirical Methods in Natural Language Processing*, 2016b.

Junnan Li, Pan Zhou, Caiming Xiong, and Steven Hoi. Prototypical contrastive learning of unsupervised representations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=KmykpuSrjcq.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In

Proc. of European Conference on Computer Vision, 2014.

Chenxi Liu, Piotr Dollár, Kaiming He, Ross Girshick, Alan Yuille, and Saining Xie. Are labels necessary for neural architecture search? In *European Conference on Computer Vision*, pages 798–813. Springer, 2020.

Ryan Lowe, Abhinav Gupta, Jakob Foerster, Douwe Kiela, and Joelle Pineau. On the interaction between supervision and self-play in emergent communication. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rJxGLlBtwH.

Yuchen Lu, Soumye Singhal, Florian Strub, Olivier Pietquin, and Aaron Courville. Countering language drift with seeded iterated learning. In *Proceedings of International Conference of Machine Learning (ICML)*, 2020.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on Robot Learning*, pages 1113–1132. PMLR, 2020.

Xingjun Ma, Yisen Wang, Michael E Houle, Shuo Zhou, Sarah Erfani, Shutao Xia, Sudanthi Wijewickrema, and James Bailey. Dimensionality-driven learning with noisy labels. In *International Conference on Machine Learning*, pages 3355–3364. PMLR, 2018.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *Proc. of International Conference on Learning Representations*, 2017.

Christopher D Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences*, 117(48):30046–30054, 2020.

Mitch Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.emnlp-main.759.

Marvin Minsky and Seymour A Papert. *Perceptrons, Reissue of the 1988 Expanded Edition with a new foreword by Léon Bottou: An Introduction to Computational Geometry*. MIT press, 2017.

Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6707–6717, 2020.

Tom M Mitchell. *The need for biases in learning generalizations*. Citeseer, 1980.

Sarthak Mittal, Alex Lamb, Anirudh Goyal, Vikram Voleti, Murray Shanahan, Guillaume Lajoie, Michael Mozer, and Yoshua Bengio. Learning to combine top-down and bottom-up signals in recurrent neural networks with attention over modules. *arXiv preprint arXiv:2006.16981*, 2020.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3303–3313, 2018.

Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI*, pages 69–84. Springer, 2016.

Mehdi Noroozi, Hamed Pirsiavash, and Paolo Favaro. Representation learning by learning to count. In *Proceedings of the IEEE international conference on computer vision*, pages 5898–5906, 2017.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2018. URL http://arxiv.org/abs/1807.03748. cite arxiv:1807.03748.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2016.

Ronald Parr and Stuart J Russell. Reinforcement learning with hierarchies of machines. In *Advances in neural information processing systems*, pages 1043–1049, 1998.

Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

Karl W. Pettis, Thomas A. Bailey, Anil K. Jain, and Richard C. Dubes. An intrinsic dimensionality estimator from near-neighbor information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(1):25–37, 1979a. doi: 10.1109/TPAMI.1979.4766873.

Karl W. Pettis, Thomas A. Bailey, Anil K. Jain, and Richard C. Dubes. An intrinsic dimensionality estimator from near-neighbor information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(1):25–37, 1979b. doi: 10.1109/TPAMI.1979.4766873.

Phil Pope, Chen Zhu, Ahmed Abdelkader, Micah Goldblum, and Tom Goldstein. The intrinsic dimension of images and its impact on learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=XJk19XzGq2J.

Eva Portelance, Michael C Frank, Dan Jurafsky, Alessandro Sordoni, and Romain Laroche. The emergence of the shape bias results from communicative efficiency. *arXiv e-prints*, pages arXiv–2109, 2021.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*, 2022.

Linlu Qiu, Peter Shaw, Panupong Pasupat, Tianze Shi, Jonathan Herzig, Emily Pitler, Fei Sha, and Kristina Toutanova. Evaluating the impact of model scale for compositional generalization in semantic parsing. *arXiv preprint arXiv:2205.12253*, 2022.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog 1.8*, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

Anna N Rafferty, Thomas L Griffiths, and Marc Ettlinger. Exploring the relationship between learnability and linguistic universals. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*, pages 49–57, 2011.

Sai Rajeswar, Pau Rodriguez, Soumye Singhal, David Vazquez, and Aaron Courville. Multi-label iterated learning for image classification with label ambiguity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4783–4793, 2022.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/ramesh21a.html.

Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. Impact of pretraining term frequencies on few-shot reasoning. *arXiv preprint arXiv:2202.07206*, 2022.

Stefano Recanatesi, Matthew Farrell, Madhu Advani, Timothy Moore, Guillaume Lajoie, and Eric Shea-Brown. Dimensionality compression and expansion in deep neural networks. *arXiv preprint arXiv:1906.00443*, 2019.

Colorado J Reed, Sean Metzger, Aravind Srinivas, Trevor Darrell, and Kurt Keutzer. Selfaugment: Automatic augmentation policies for self-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2674–2683, 2021.

Yi Ren, Shangmin Guo, Matthieu Labeau, Shay B. Cohen, and Simon Kirby. Compositional languages emerge in a neural iterated learning model. In *Proc. of International Conference on Learning Representations*, 2020.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The knowledge engineering review*, 21(2):97–126, 2006.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. *Advances in neural information processing systems*, 28, 2015a.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015b.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 527–538, 2018.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association*

for Computational Linguistics (Volume 1: Long Papers), pages 1715–1725, 2016.

Pararth Shah, Dilek Hakkani-Tur, Bing Liu, and Gokhan Tur. Bootstrapping a neural conversational agent with dialogue self-play, crowdsourcing and on-line reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 41–51, 2018.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. Ordered neurons: Integrating tree structures into recurrent neural networks. In *International Conference on Learning Representations*, 2018.

Yikang Shen, Shawn Tan, Arian Hosseini, Zhouhan Lin, Alessandro Sordoni, and Aaron C Courville. Ordered memory. In *Advances in Neural Information Processing Systems*, pages 5037–5048, 2019.

Yikang Shen, Yi Tay, Che Zheng, Dara Bahri, Donald Metzler, and Aaron Courville. Structformer: Joint unsupervised induction of dependency and constituency structure from masked language modeling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7196–7209, 2021.

Kyriacos Shiarlis, Markus Wulfmeier, Sasha Salter, Shimon Whiteson, and Ingmar Posner. Taco: Learning task decomposition via temporal alignment for control. In *International Conference on Machine Learning*, pages 4654–4663, 2018.

Daniel L Silver and Robert E Mercer. The task rehearsal method of life-long learning: Overcoming impoverished data. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 90–101. Springer, 2002.

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. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Gabriel Skantze and Anna Hjalmarsson. Towards incremental speech generation in dialogue systems. In *Proc.of the Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, 2010.

Kenny Smith, Monica Tamariz, and Simon Kirby. Linguistic structure is an evolutionary trade-off between simplicity and expressivity. In *Proceedings of the annual meeting of the cognitive science society*, volume 35, 2013.

Paul Smolensky. The constituent structure of connectionist mental states: A reply to fodor and pylyshyn. In *Connectionism and the philosophy of mind*, pages 281–308. Springer,

1991.

Alec Solway, Carlos Diuk, Natalia Córdova, Debbie Yee, Andrew G Barto, Yael Niv, and Matthew M Botvinick. Optimal behavioral hierarchy. *PLOS Comput Biol*, 10(8):e1003779, 2014.

Florian Strub, Harm De Vries, Jeremie Mary, Bilal Piot, Aaron Courville, and Olivier Pietquin. End-to-end optimization of goal-driven and visually grounded dialogue systems. In *Proc. of International Joint Conferences on Artificial Intelligence*, 2017.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112 (1-2):181–211, 1999.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 2000.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, 2015.

Jesse Thomason, Shiqi Zhang, Raymond J Mooney, and Peter Stone. Learning to interpret natural language commands through human-robot dialog. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? *arXiv preprint arXiv:2005.10243*, 2020.

Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning dynamics without contrastive pairs. In *International Conference on Machine Learning*, pages 10268–10278. PMLR, 2021.

Michael Tschannen, Josip Djolonga, Paul K. Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkxoh24FPH.

Ankit Vani, Max Schwarzer, Yuchen Lu, Eeshan Dhekane, and Aaron Courville. Iterated learning for emergent systematicity in {vqa}. In *International Conference on Learning*

Representations, 2021. URL https://openreview.net/forum?id=Pd_oMxH8IlF.

Ramakrishna Vedantam, David Lopez-Paz, and David J Schwab. An empirical investigation of domain generalization with empirical risk minimizers. *Advances in Neural Information Processing Systems*, 34, 2021.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1703.01161*, 2017.

P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning proceedings*. 2008.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pages 9929–9939. PMLR, 2020.

Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2015.

Alex Warstadt, Yian Zhang, Xiaocheng Li, Haokun Liu, and Samuel Bowman. Learning which features matter: Roberta acquires a preference for linguistic generalizations (eventually). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 217–235, 2020.

Albert Webson and Ellie Pavlick. Do prompt-based models really understand the meaning of their prompts? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2300–2344, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.167. URL https://aclanthology.org/2022.naacl-main.167.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

Wei Wei, Quoc Le, Andrew Dai, and Jia Li. Airdialogue: An environment for goal-oriented dialogue research. In *Proc. of Empirical Methods in Natural Language Processing*, 2018.

William F Whitney, Min Jae Song, David Brandfonbrener, Jaan Altosaar, and Kyunghyun Cho. Evaluating representations by the complexity of learning low-loss predictors. In *Neural Compression: From Information Theory to Applications–Workshop@ ICLR 2021*, 2021.

Jason D Williams, Matthew Henderson, Antoine Raux, Blaise Thomson, Alan Black, and Deepak Ramachandran. The dialog state tracking challenge series. *AI Magazine*, 35(4): 121–124, 2014.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Terry Winograd. Procedures as a representation of data in a computer program for understanding natural language. Technical report, PhD thesis, Massachusets Institute of Technology, January 1971.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742, 2018.

Qizhe Xie, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Self-training with noisy student improves imagenet classification. *arXiv preprint arXiv:1911.04252*, 2019.

Jing Xu, Thomas Griffiths, and Mike Dowman. Replicating color term universals through human iterated learning. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 32, 2010.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. *Advances in neural information processing systems*, 31, 2018.

Licheng Yu, Hao Tan, Mohit Bansal, and Tamara L Berg. A joint speaker-listener-reinforcer model for referring expressions. In *Proc. of Computer Vision and Pattern Recognition*, 2017.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *arXiv preprint arXiv:2001.06782*, 2020a.

Yaodong Yu, Kwan Ho Ryan Chan, Chong You, Chaobing Song, and Yi Ma. Learning diverse and discriminative representations via the principle of maximal coding rate reduction. *Advances in Neural Information Processing Systems*, 33:9422–9434, 2020b.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pages 12310–12320. PMLR, 2021.

Richard Zhang, Phillip Isola, and Alexei A Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1058–1067, 2017.

Siyuan Zhou, Yikang Shen, Yuchen Lu, Aaron Courville, Joshua B. Tenenbaum, and Chuang Gan. Inducing reusable skills from demonstrations with option-controller network, 2022. URL https://openreview.net/forum?id=62r41yOG5m.

Yuchen Zhou and Dan Yurovsky. A common framework for quantifying the learnability of nouns and verbs. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 43, 2021.

Yan Zhu, Shaoting Zhang, and Dimitris Metaxas. Interactive reinforcement learning for object grounding via self-talking. *Visually Grounded Interaction and Language Workshop*, 2017.

Appendix A

Supplementary Material for Article One

A.1. Complementary Theoretical Intuition for SIL and Its Limitation

We here provide a complementary intuition of Seeded Iterated Learning by referring to some mathematical tools that were used to study Iterated Learning dynamics in the general case. These are not the rigorous proof but guide the design of SIL.

One concern is that, since natural language is not fully compositional, whether iterated learning may favor the emergence of a new compositional language on top of the initial one. In this spirit, Griffiths and Kalish (2005); Kalish et al. (2007) modeled iterated learning as a Markov Process, and showed that vanilla iterated learning indeed converges to a language distribution that (i) is independent of the initial language distribution, (ii) depends on the student language before the inductive learning step.

Fortunately, Chazelle and Wang (2017) show iterated learning can converge towards a distribution close to the initial one with high probability if the intermediate student distributions remain close enough of their teacher distributions and if the number of training observations increases logarithmically with the number of iterations.

This theoretical result motivates one difference between our framework and classical iterated learning: as we want to preserve the pretrained language distribution, we do not initialize the new students from scratch as in (Li and Bowling, 2019; Guo et al., 2019; Ren et al., 2020) because the latter approach exert a uniform prior on the space of language, while we would like to add a prior that favors natural language (e.g. favoring language whose token frequency satisfies Zipf's Law).

A straightforward instantiation of the above theoretic results is to initialize new students as the pretrained model. However we empirically observe that, periodically resetting the

model to initial pretrained model would quickly saturate the task score. As a result, we just keep using the students from the last imitation learning for the beginning of new generation, as well as retain the natural language properties from pretraining checkpoint.

However, we would also point out the limitation of existing theoretical results in the context of deep learning: The theoretical iterated learning results assume the agent to be perfect Bayesian learner (e.g. Learning is infering the posterior distribution of hypothesis given data). However, we only apply standard deep learning training procedure in our setup, which might not have this property. Because of the assumption of perfect Bayesian learner, (Chazelle and Wang, 2019) suggests to use training sessions with increasing length. However in practice, increasing k_2 may be counter-productive because of overfitting issues (especially when we have limited number of training scenarios).

A.2. Lewis Game

A.2.1. Experiment Details

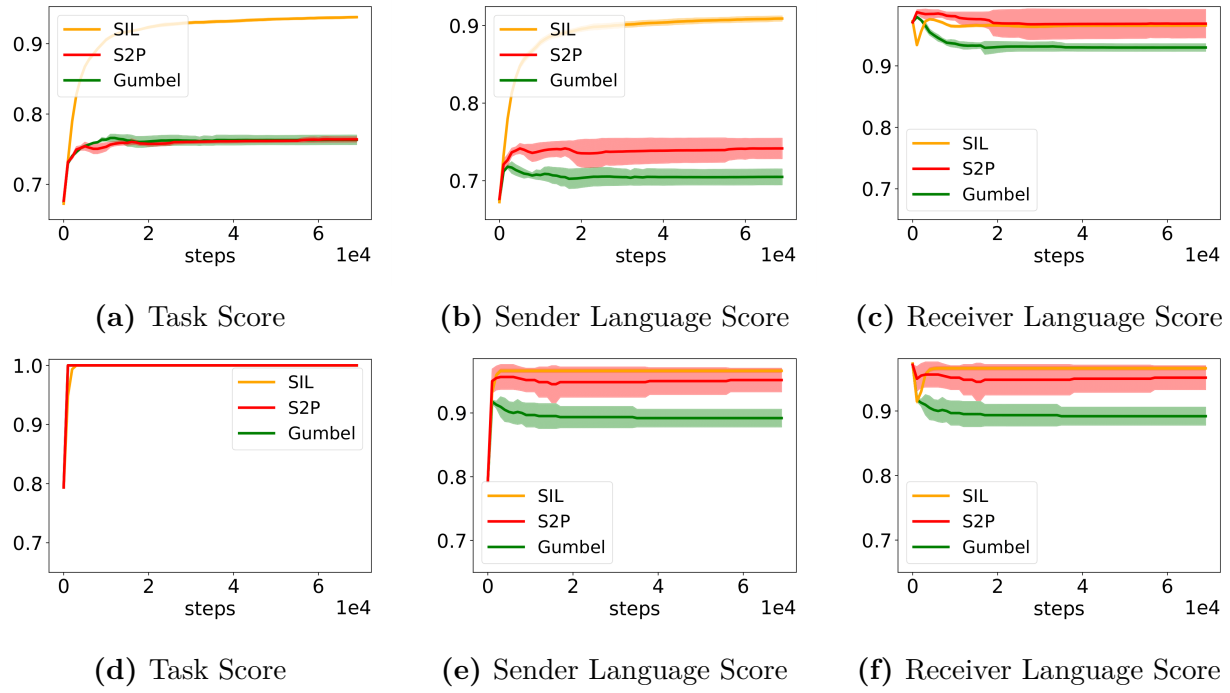
In the Lewis game, the sender and the receiver architecture are 2-layer MLP with a hidden size of 200 and no-activation ($ReLU$ activations lead to similar scores). During interaction learning, we use a learning rate of 1e-4 for SIL. We use a learning rate of 1e-3 for the baselines as it provides better performance on the language and score tasks. In both cases, we use a training batch size of 100. For the teacher imitation phase, the student uses a learning rate of 1e-4.

In the Lewis game setting, we generate objects with $p = 5$ properties, where each property may take $t = 5$ values. Thus, it exists 3125 objects, which we split into 3 datasets: the pretraining, the interactive, and testing datasets. The pretraining split only contains 10 combination of objects. As soon as we provide additional objects, the sender and receiver fully solve the game by using the target language, which is not suitable to study the language drift phenomenon. The interactive split contains 30 objects. This choice is arbitrary, and choosing a additional objects gives similar results. Finally, the 3.1k remaining objects are held-out for evaluation.

A.2.2. Additional Plots

We sweep over different Gumbel temperatures to assess the impact of exploration on language drift. We show the results with Gumbel temperature $\tau = 1, 10$ in Fig A.27 and Fig A.26. We observe that the baselines are very sensitive to Gumbel temperature: high temperature both decreases the language and tasks score. On the other side, Seeded Iterated

Learning perform equally well on both temperatures and manage to maintain both task and language accuracies even with high temperature.



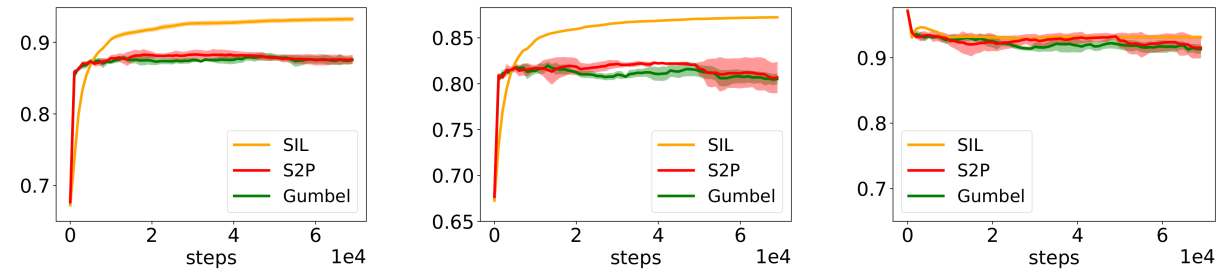
(a) Task Score **(b)** Sender Language Score **(c)** Receiver Language Score

(d) Task Score **(e)** Sender Language Score **(f)** Receiver Language Score

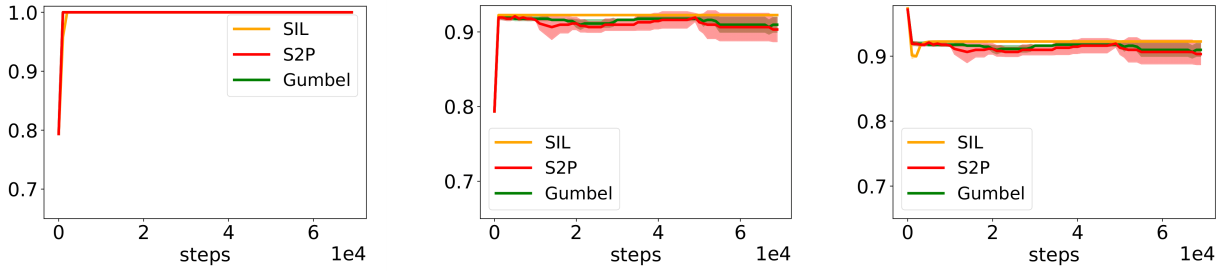
Fig. A.26. Complete training curves for Task score and sender grounding in Lewis Game comparing SIL vs baselines for $\tau = 10$ on the held-out dataset (bottom), and the interactive training split (bottom). The first row is on held-out set while the second row is on train set. We observe that the three methods reach 100% accuracy on the training task score, but their score differs on the held-out split. For SIL we use $k_1 = 1000, k_2 = k_2' = 400$.

A.2.3. Tracking Language Drift with Token Accuracy

To further visualize the language drift in Lewis game, we focus on the evolution of on the probability of speaking different word when facing the same concept. Formally, we track the change of conditional probability $s(w|c)$. The result is in Figure A.28.



(a) Task Score **(b)** Sender Language Score **(c)** Receiver Language Score



(d) Task Score **(e)** Sender Language Score **(f)** Receiver Language Score

Fig. A.27. Complete training curves for Task score and sender grounding in Lewis Game comparing SIL vs baselines for $\tau = 1$ on the held-out dataset (bottom), and the interactive training split (bottom). The first row is on held-out set while the second row is on train set. For SIL we use $k_1 = 1000, k_2 = k_2' = 400$.

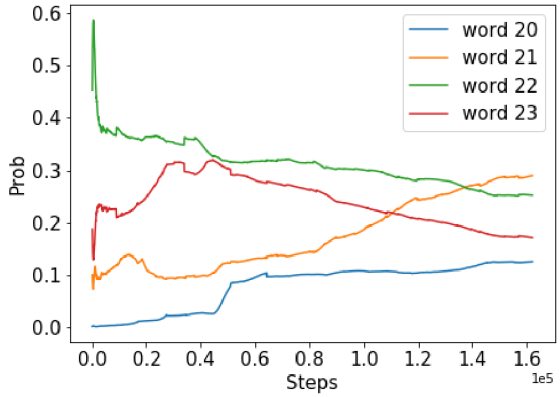
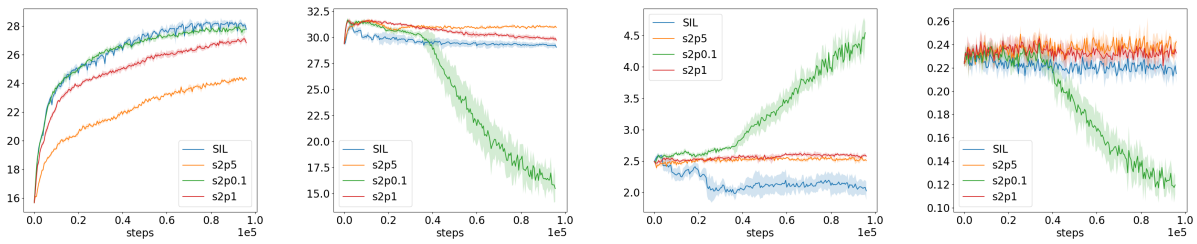


Fig. A.28. Change of conditional probability $s(w|c)$ where $c = 22$ and $w = 20, 21, 22, 23$. Following pretraining, $s(22|22)$ start with the highest probability. However, language drift gradually happens and eventually word 21 replaces the correct word 22.

Table A.7. Translation Game Results. The checkmark in "ref len" means the method use reference length to constrain the output during training/testing. ↑ means higher the better and vice versa. Our results are averaged over 5 seeds, and reported values are extracted for the best BLEU(De) score during training. We here use a Gumbel temperature of 0.5.

Method		ref len	BLEU↑		NLL↓	R1%↑
			De	En		
Lee et al. (2019)	Pretrained	N/A	16.3	27.18	N/A	N/A
	PG	✓	24.51	12.38	N/A	N/A
	PG+LM+G	✓	28.08	24.75	N/A	N/A
Ours	Pretrained	N/A	15.68	29.39	2.49	21.9
	Fix Sender	N/A	22.02 ± 0.18	29.39	2.49	21.9
	Gumbel		27.11 ± 0.14	14.5 ± 0.83	5.33 ± 0.39	9.7 ± 1.2
	Gumbel	✓	26.94 ± 0.20	23.41± 0.50	5.04 ± 0.01	18.9 ± 0.8
	S2P($\alpha = 0.1$)		27.43± 0.36	19.16 ± 0.63	4.05 ± 0.16	13.6 ± 0.7
	S2P($\alpha = 1$)		27.35± 0.19	29.73 ± 0.15	2.59 ± 0.02	**23.7 ± 0.7**
	S2P($\alpha = 5$)		24.64± 0.16	**30.84 ± 0.07**	2.51 ± 0.02	23.5 ± 0.5
	NIL		**28.29± 0.16**	29.4 ± 0.25	**2.15 ± 0.12**	21.7 ± 0.2



(a) BLEU De **(b)** BLEU En **(c)** NLL **(d)** R1

Fig. A.29. S2P has a trade-off between the task score and the language score while SIL is consistently high with both metrics.

A.3. Translation Game

A.3.1. Data Preprocessing

We use Moses to tokenize the text (Koehn et al., 2007) and we learn byte-pair-encoding (Sennrich et al., 2016) from Multi30K (Elliott et al., 2016) with all language. Then we apply the same BPE to different dataset. Our vocab size for En, Fr, De is 11552, 13331, and 12124.

A.3.2. Model Details and Hyperparameters

The model is a standard seq2seq translation model with attention (Bahdanau et al., 2015). Both encoder and decoder have a single-layer GRU (Cho et al., 2014) with hidden size 256. The embedding size is 256. There is a dropout after embedding layers for both encoder and decoder For decoder at each step, we concatenate the input and the attention context from last step.

Pretraining. For Fr-En agent, we use dropout ratio 0.2, batch size 2000 and learning rate 3e-4. We employ a linear learning rate schedule with the anneal steps of 500k. The minimum learning rate is 1e-5. We use Adam optimizer (Kingma and Ba, 2014) with $\beta = (0.9, 0.98)$. We employ a gradient clipping of 0.1. For En-De, the dropout ratio is 0.3. We obtain a BLEU score of 32.17 for Fr-En, and 20.2 for En-De on the IWSLT test dataset (Cettolo et al., 2012).

Finetuning. During finetuning, we use batch size 1024 and learning rate 1e-5 with no schedule. The maximum decoding length is 40 and minimum decoding length is 3. For iterated learning, we use $k_1 = 4000$, $k_2 = 200$ and $k_2' = 300$. We set Gumbel temperature to be 5. We use greedy sample from teacher speaker for imitation.

A.3.3. Language Model and Image Ranker Details

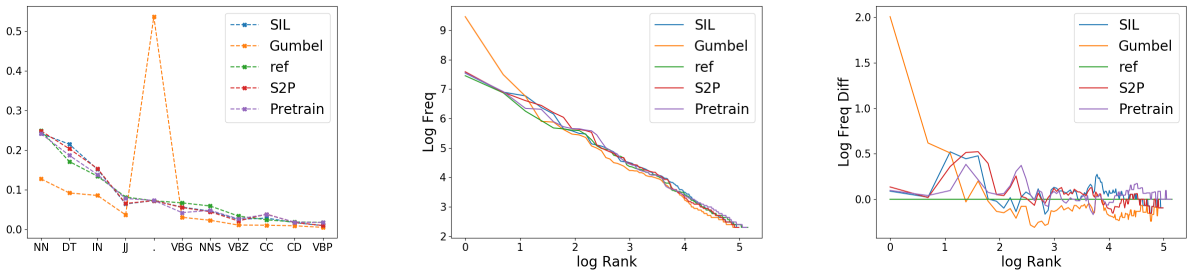
Our language model is a single-layer LSTM (Hochreiter and Schmidhuber, 1997) with hidden size 512 and embedding size 512. We use Adam and learning rate of 3e-4. We use a batch size of 256 and a linear schedule with 30k anneal steps. The language model is trained with captions from MSCOCO (Lin et al., 2014). For the image ranker, we use the pretrained ResNet-152 (He et al., 2016) to extract the image features. We use a GRU (Cho et al., 2014) with hidden size 1024 and embedding size 300. We use a batch size of 256 and use VSE loss (Faghri et al., 2017). We use Adam with learning rate of 3e-4 and a schedule with 3000 anneal steps (Kingma and Ba, 2014).

A.3.4. Language Statistics

We here compute several linguistic statistics on the generated samples to assess language quality.

POS Tag Distribution. We compute the Part-of-Speech Tag (POS Tag (Marcus et al., 1993)) distribution by counting the frequency of POS tags and normalize it. The POS tag are sorted according to their frequencies in the reference, and we pick the 11 most common POS tag for visualization, which are:

- NN Noun, singular or mass



(a) POS tag distribution. **(b)** Word Frequency Analysis **(c)** Difference of Log Frequency

Fig. A.30. Language statistics on samples from different method.

- DT Determiner
- IN Preposition or subordinating conjunction
- JJ Adjective
- VBG Verb, gerund or present participle
- NNS Noun, plural
- VBZ Verb, 3rd person singular present
- CC Coordinating conjunction
- CD Cardinal number

The results are shown in Figure A.30a. The peak on “period" show that Gumbel has tendency of repeating periods at the end of sentences. However, we observe that both S2P and

Word Frequency. For each generated text, we sort the frequency of the words and plot the log of frequency vs. log of rank. We set a minimum frequency of 50 to exclude long tail results. The result is in Figure A.30b.

Word Frequency Difference. To further visualize the difference between generated samples and reference, we plot the difference between their log of word frequencies in Figure A.30c.

A.4. Human Evaluation

We here assess whether our language drift evaluation correlates with human judgement. To do so, we performed a human evaluation with two pairwise comparison tasks.

- In Task1, the participant picks the best English semantic translation while observing the French sentence.
- In Task2, the participant picks the best English translation from two candidates.

Thus, the participants are likely to rank captions mainly by their syntax/grammar quality in Task2, whereas they would also consider semantics in Task1, allowing us to partially disentangle structural and semantic drift.

For each task, we use the validation data from Multi30K (1013 French captions) and generate 4 English sentences for each French caption from the Pretrain, Gumbel, S2P, and SIL. We also retrieved the ground-truth human English caption. We then build the test by randomly sampling two out of five English captions. We gathered 22 people, and we collect about 638 pairwise comparisons for Task2 and 315 pairwise comparisons for Task1. We present the result in Table A.9 and Table A.10. I also include the binomial statistical test result where the null hypothesis is *methods are the same*, and the alternative hypothesis is *one method is better than the other one*.

Unsurprisingly, we observe that the Human samples are always preferred over generated sentences. Similarly, Gumbel is substantially less preferred than other models in both settings.

In Task 1(French provided), human users always preferred S2P and SIL over pretrained models with a higher win ratio. Oh the other hand when French is not provided, the human users prefer the pretrain models over S2P and SIL. We argue that while the pretrained model keeps generating gramartically correct sentences, its translation effectiveness is worse than both S2P and SIL since these two models go through the interactive learning to adapt to new domain.

Finally, SIL seems to be preferred over S2P by a small margin in both tasks. However, our current ranking is not conclusive, since we can see the significance level of comparisons among Pretrain, S2P, and SIL is not smaller enough to reject null hypothesis, especially in task 1 where we have less data points. In the future we plan to have a larger scale human evaluation to further differentiate these methods.

Table A.8. The Win-Ratio Results. The number in row X and column Y is the empiric ratio that method X beats method Y according collected human pairwise preferences. We perform a naive ranking by the row-sum of win-ratios of each method. We also provide the corresponding P-values under each table. The null hypothesis is *two methods are the same*, while the alternative hypothesis is *two methods are different.*

Table A.9. With French Sentences

	Gumbel	Pretrain	S2P	SIL	Human
Gumbel	0	0.25	0.15	0.12	0
Pretrain	0.75	0	0.4	0.4	0.13
S2P	0.84	0.6	0	0.38	0.21
SIL	0.88	0.6	0.63	0	0.22
Human	1	0.87	0.79	0.77	0
Ranking	Human(3.4), SIL(2.3), S2P(2.0), Pretrain(1.7), Gumbel(0.5)				
	P-values				
	Gumbel	Pretrain	S2P	SIL	Human
Gumbel	-	$< 10^{-2}$	$< 10^{-2}$	$< 10^{-2}$	$< 10^{-2}$
Pretrain	$< 10^{-2}$	-	0.18	0.21	$< 10^{-2}$
S2P	$< 10^{-2}$	0.18	-	0.15	$< 10^{-2}$
SIL	$< 10^{-2}$	0.21	0.15	-	$< 10^{-2}$
Human	$< 10^{-2}$	$< 10^{-2}$	$< 10^{-2}$	$< 10^{-2}$	-

Table A.10. Without French Sentences

	Gumbel	Pretrain	S2P	SIL	Human
Gumbel	0	0.16	0.12	0.13	0.02
Pretrain	0.84	0	0.69	0.59	0.15
S2P	0.88	0.31	0	0.38	0.05
SIL	0.86	0.41	0.62	0	0.01
Human	0.98	0.85	0.95	0.98	0
Ranking	Human(3.8), Pretrain(2.3), SIL(1.9), S2P(1.6), Gumbel(0.4)				
	P-values				
	Gumbel	Pretrain	S2P	SIL	Human
Gumbel	-	$< 10^{-2}$	$< 10^{-2}$	$< 10^{-2}$	$< 10^{-2}$
Pretrain	$< 10^{-2}$	-	$< 10^{-2}$	0.08	$< 10^{-2}$
S2P	$< 10^{-2}$	$< 10^{-2}$	-	0.06	$< 10^{-2}$
SIL	$< 10^{-2}$	0.08	0.06	-	$< 10^{-2}$
Human	$< 10^{-2}$	$< 10^{-2}$	$< 10^{-2}$	$< 10^{-2}$	-

Appendix B

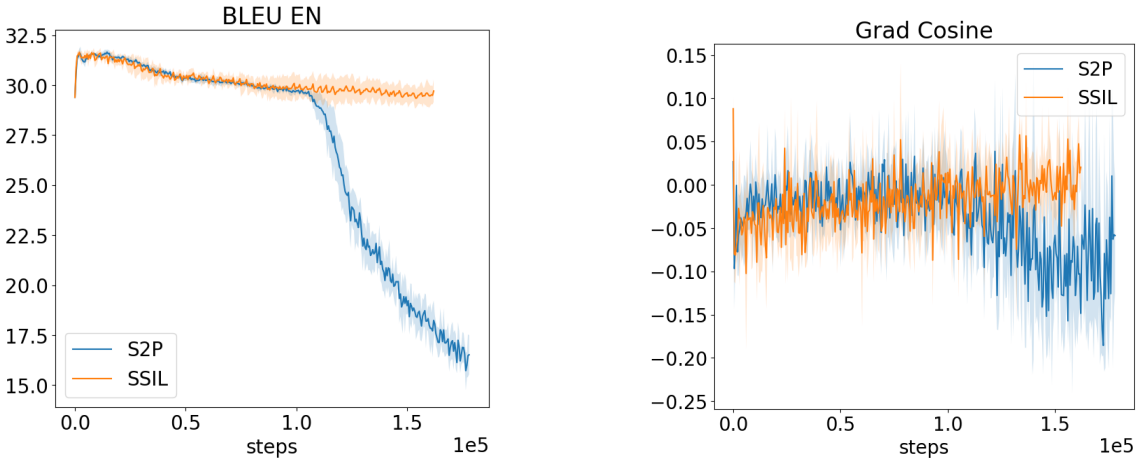
Supplementary Material for Article Two

B.1. Explicit losses in the Translation Game

Let $\mathcal{L}^{\text{GSTE}}(Fr, De)$ be the loss of Gumbel STE, when two agents is fed with Fr and the ground truth German translation De. Let $\mathcal{L}^{\text{CE}}(X, Y)$ to be the supervised training loss with source X and target Y. Then for each interactive training step, we have for both agents

$$\mathcal{L}^{\text{S2P}}_{sender} = \mathcal{L}^{\text{GSTE}}(\text{Fr}^{ft}, \text{De}^{ft}) + \alpha\mathcal{L}^{\text{CE}}(\text{Fr}^{pre}, \text{En}^{pre}) \tag{B.1.1}$$

$$\mathcal{L}^{\text{S2P}}_{receiver} = \mathcal{L}^{\text{GSTE}}(\text{Fr}^{ft}, \text{De}^{ft}) + \alpha\mathcal{L}^{\text{CE}}(\text{En}^{pre}, \text{De}^{pre}) \tag{B.1.2}$$



(a) Bleu En **(b)** Cosine Similarity

Fig. B.31. Cosine similarity bewteen $\mathcal{L}^{\text{CE}}_{pretrain}$ and $\mathcal{L}^{\text{INT}}$ when $\alpha = 0.7$

B.2. Translation Game Implementation Details

We use the Moses tokenizer Koehn et al. (2007) and we learn a byte-pair-encoding from Multi30K with all language. Then the same BPE is applied to different dataset. Our vocab size for En, Fr, De is 11552, 13331, and 12124. Our pretraining datasets are IWSLT while the finetuning datasets are Multi30K. Our language model is trained with captions data from MSCOCO Lin et al. (2014). For image ranker, we use the captions in Multi30K as well as the original Flickr30K images. We use a ResNet152 with pretrained ImageNet weights to extract the image features. We also normalize the image features. We follow the pretraining and model architecture from work Lu et al. (2020).

B.3. Hyper-parameters

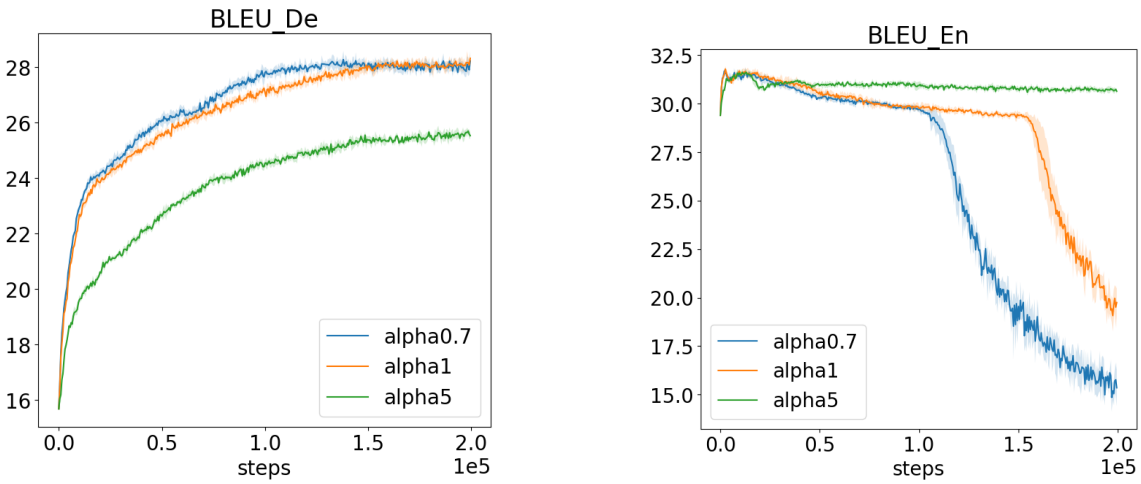
During finetuning, we set Gumbel temperature to be 0.5 and follow the previous work Lu et al. (2020) for other hyperparameters, e.g. learning rate, batch size, etc. We list our hyper-parameters and our sweep: We mainly use P100 GPU for our experiments. For training

Name	Sweep
k_1	3000, 4000
k_2	200, 300, 400
k_2'	200, 300, 400
α	0, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7

200k steps, Gumbel takes 17 hours, S2P takes 24 hours, SIL takes 18 hours and SSIL takes 24 hours. The best hyperparameters for SIL are $k_1 = 3000, k_2 = 200, k_2' = 300$. The best *alpha* for $S2P$ is 1, while for $SSIL$ we choose $\alpha = 0.5$.

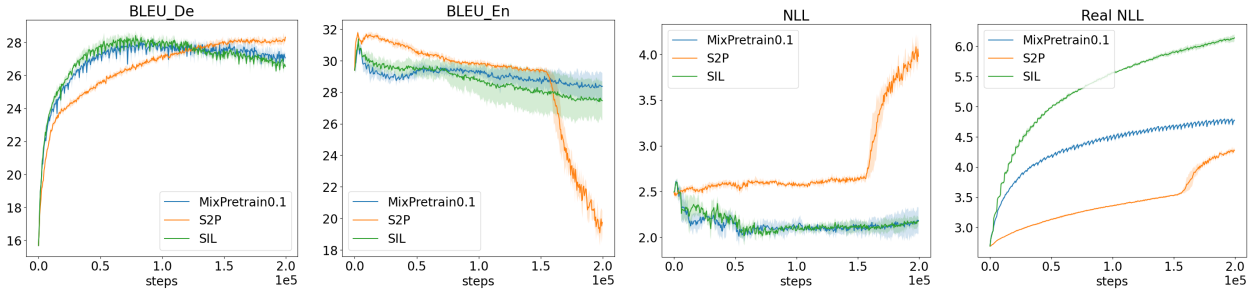
B.4. S2P Details

We show the results of S2P with varying α in Figure B.32. In general, one can find that for S2P there is a trade-off between grounding score and task score controlled by α. A larger α might delay the eventual collapse. However, if the α is too large, the task score will decrease significantly. As a result, even though increasing α seems to fit the intuition, it cannot fix the problem.



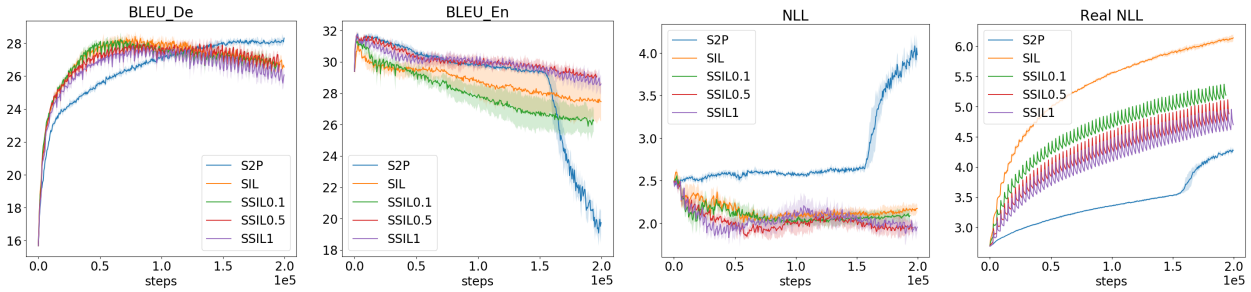
(a) BLEU De (Task Score) **(b)** BLEU En

Fig. B.32. S2P with different α. Increased α might delay or remove the late-stage collapse, but it might be at the cost of task score.



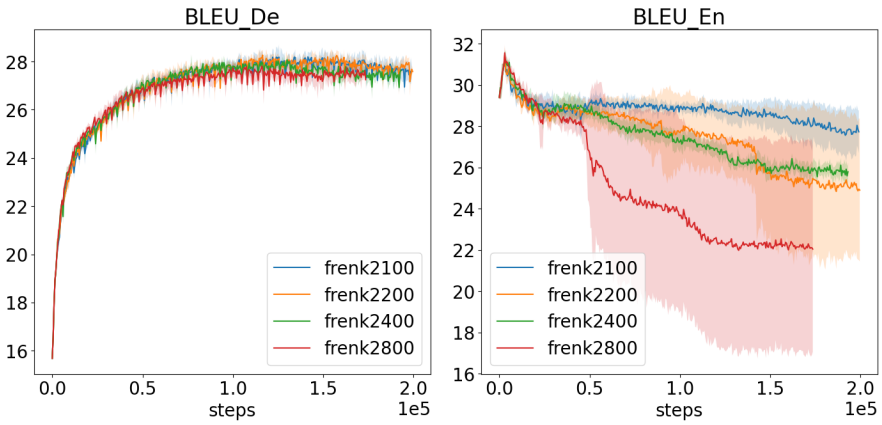
(a) BLEU De **(b)** BLEU En **(c)** NLL **(d)** RealNLL

Fig. B.33. Mix with Pretraining data in SIL.



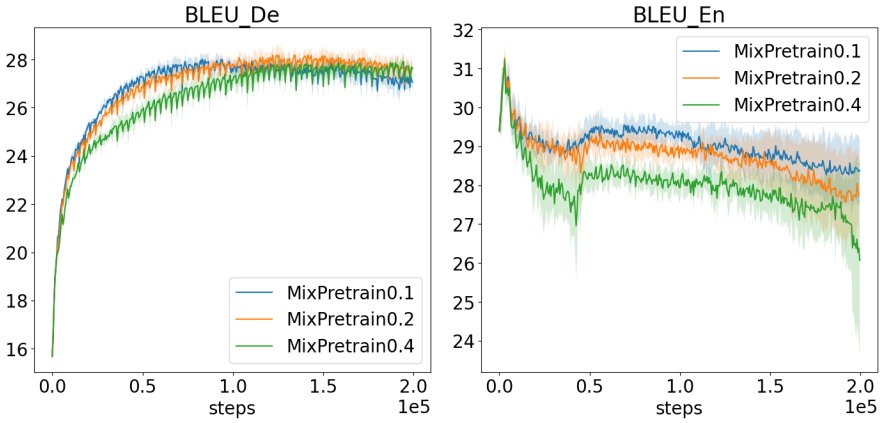
(a) BLEU De **(b)** BLEU En **(c)** NLL **(d)** RealNLL

Fig. B.34. SSIL with different α



(a) BLEU De (Task Score) **(b)** BLEU En

Fig. B.35. Effect of k_2 for MixData.$\alpha = 0.2$



(a) BLEU De (Task Score) **(b)** BLEU En

Fig. B.36. Effect of α for MixData. $k_2 = 100$

Appendix C

Supplementary Material for Article Three

C.1. OMPN Architecture Details

We use the gated recursive cell function from Shen et al. (2019) in the top-down and bottom up recurrence. We use a two-layer MLP to compute the score f_i for the stick-breaking process. For the initial memory M^0, we send the environment information into the highest slot while keep the rest of the slots to be zeros. If unsupervised setting, then the every slot is initialized as zero. At the first time step, we also skip the bottom-up process and hard code π^1 such that the memory expands from the highest level. This is used to make sure that at first time step, we could propagate our memory with the expanded subtasks from root task. In our experiment, our cell functions does not share the parameters. We find that to be better than shared-parameter.

We set the number of slots to be 3 in both Craft and Dial, and each memory has dimension 128. We use Adam optimizer to train our model with $\beta_1 = 0.9, \beta_2 = 0.999$. The learning rate is 0.001 in Craft and 0.0005 in Dial. We set the length of BPTT to be 64 in both experiments. We clip the gradients with L2 norm 0.2. The observation has dimension 1076 in Craft and 39 in Dial. We use a linear layer to encode the observation. After reading the output O^t, we concatenate it with the observation x^t and send them to a linear layer to produce the action.

In section 5.2, we describe that we augment the action space into $\mathcal{A} \cup \{done\}$ and we append the trajectory $\tau = \{s_t, a_t\}_{t=1}^T$ with one last step, which is $\tau \cup \{s_{t+1}, done\}$. This can be easily done if the data is generated by letting an expert agent interact with the environment. If you do not have the luxury of environment interaction, then you can simply let $s_{T+1} = s_T, a_{T+1} = done$. We find that augmenting the trajectory in this way does not change the performance in our Dial experiment, since the task boundary is smoothed out across time steps for continuous action space, but it hurts the performance for Craft, since the final action of craft is usually USE, which can change the state a lot.

C.2. Demonstration Generation

We use a rule-based agent to generate the demonstration for both Craft and Dial. For Craft, we train on 500 episodes each on $MakeAxe$, $MakeShears$ and $MakeBed$. Each of these task is further composed of four subtasks. For Dial, we generate 1400 trajectories for imitation learning with each sketch being 4 digits. For Craft with full observations, we design a shortest path solver to go to the target location of each subtask. For Craft with partial observation, we maintain an internal memory about the currently seen map. If the target object for the current subtask is not seen on the internal memory, we perform a left-to-right, down-to-top exploration until the target object appears inside the memory. Once the target object is seen, it defaults to the behavior in full observations. For Dial, we use the hand-designed controller in Shiarlis et al. (2018) to generate the demonstration.

C.3. Task Decomposition Metric

C.3.1. F1 Scores with Tolerance

For each trajectory, we are given a set of ground truth task boundary gt of length L which is the number of subtasks. The algorithm also produce L task boundary predictions. This can be done in OMPN by setting the correct K in $topK$ boundary detection. For compILE, we set the number of segments to be equal to N. Nevertheless, our definition of F1 can be extended to arbitaray number of predictions.

$$precision = \frac{\sum_{i,j} match(preds_i, gt_j, tol)}{\#predictions)}$$

$$precision = \frac{\sum_{i,j} match(gt_i, preds_j, tol)}{\#ground\ truth}$$

where the $match$ is defined as

$$match(x, y, tol) = [y - tol \leq x \leq y + tol]$$

where [] is the Iverson bracket. The tolerance

C.3.2. Task Alignment Accuracy

This metric is taken from Shiarlis et al. (2018). Suppose we have a sketch of 4 subtasks $b = [b1, b2, b3, b4]$ and we have the ground truth assignment $\xi_{true} = \{\xi^t_{true}\}^T_{t=1}$. Similar we

have the predicted alignment ξ_{pred}. The alignment accuracy is simply

$$\sum_t [\xi^t_{pred} == \xi^t_{true}]$$

For OMPN and compILE, we obtain the task boundary first and construct the alignment as a result. For TACO, we follow the original paper to obtain the alignment.

C.4. baseline

C.4.1. compILE Details

latent	[concrete, gaussian]
prior	[0.3, 0.5,0.7]
kl_b	[0.05, 0.1, 0.2]
kl_z	[0.05, 0.1, 0.2]

Table C.11. compILE hyperparameter search.

Our implementation of compILE is taken from the author github[1]. However, their released code only work for a toy digit sequence example. As a result we modify the encoder and decoder respectively for our environments. During our experiment, we perform the following hyper-parameter sweep on the baseline in Table C.11. Although the authors use latent to be concrete during their paper, we find that gaussian perform better in our case. We find that Gaussian with $prior = 0.5$ performs the best in Craft. For Dial, these configurations perform equally bad.

We show the task alignments of compILE for Craft in Figure C.37. It seems that compILE learn the task boundary one-off. However, since the subtask ground truth can be ad hoc, this brings the question how should we decide whether our model is learning structure that makes sense or not? Further investigation in building a better benchmark/metric is required.

C.4.2. TACO Details

dropout	[0.2, 0.4, 0.6, 0.8]
decay	[0.2, 0.4, 0.6, 0.8]

Table C.12. TACO hyperparameter search.

We use the implementation from author github[2] and modifiy it into pytorch. Although the author also conduct experiment on Craft and Dial, they did not release the demonstration

[1] https://github.com/tkipf/compile
[2] https://github.com/KyriacosShiarli/taco

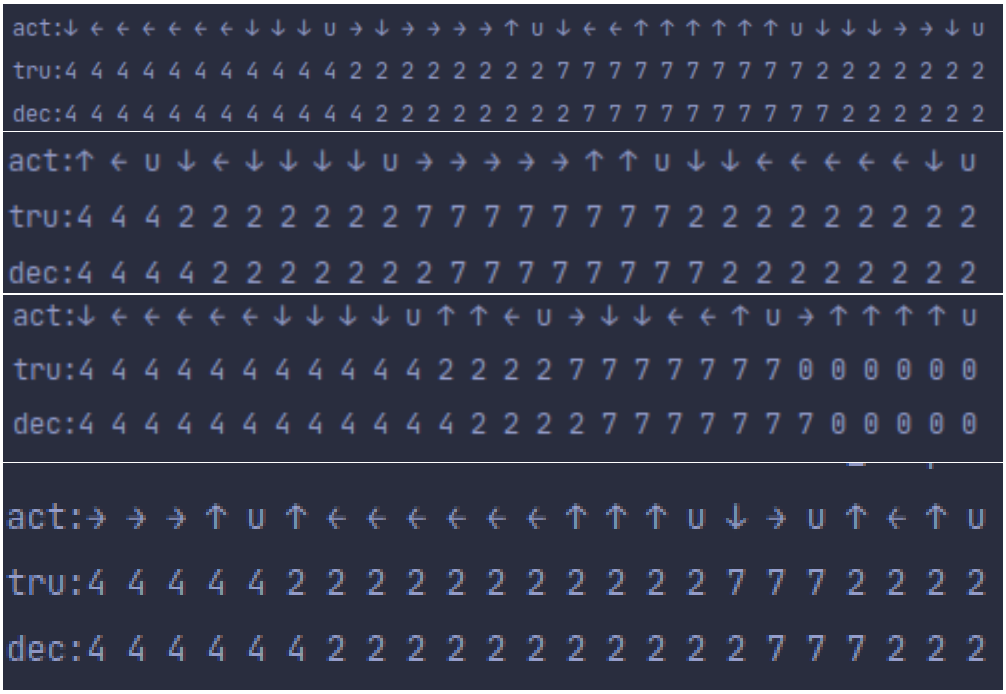


Fig. C.37. Task alignment results of compILE on Craft.

dataset they use. As a result, we cannot directly use their numbers from the paper. We also apply dropout on the prediction of $STOP$ and apply a linear decaying schedule during training. The hyperparameter search is in table C.12. We find the best hyperparameter to be $0.4, 0.4$ for Craft. For Dial, the result is not sensitive to the hyperparameters.

C.5. Craft

We train our models on *MakeBed*, *MakeAxe*, and *MakeShears*. The detail of their task decomposition is in Table C.13. We show the behavior cloning results for all settings in Figure C.39. We display more visuzliation of task decomposition results from OMPN in Figure C.38.

a	makebed	get wood, make at toolshed, get grass, make at workbench
	makeaxe	get wood, make at workbench, get iron, make at toolshed
	makeshears	get wood, make at workbench, get iron, make at workbench

Table C.13. Details of training tasks decomposition.

We show the results of task decomposition when the given K is different in table C.14 and table C.15. We find that when you increase the K, the recall increases while the precision decreases.

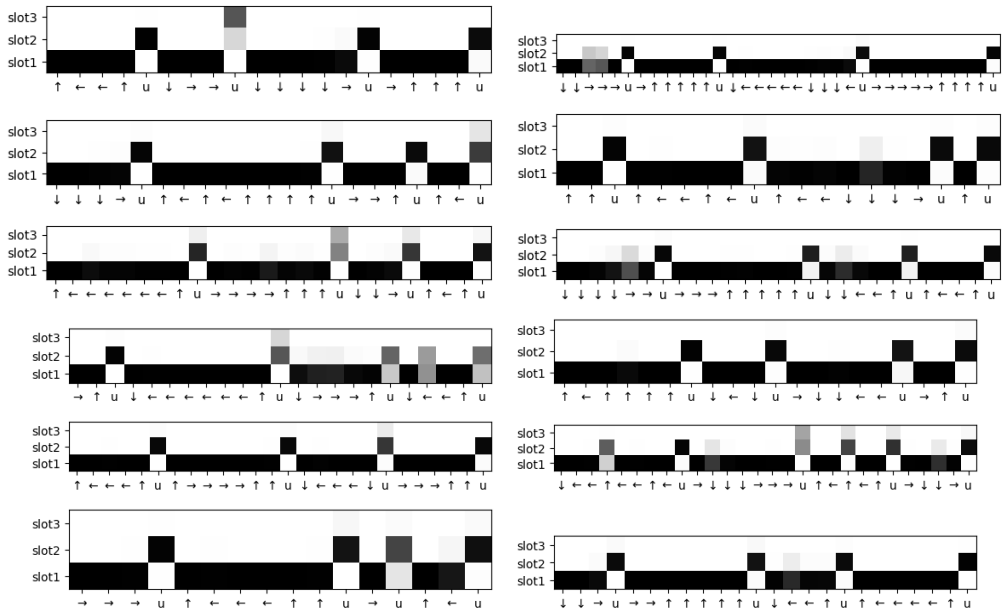
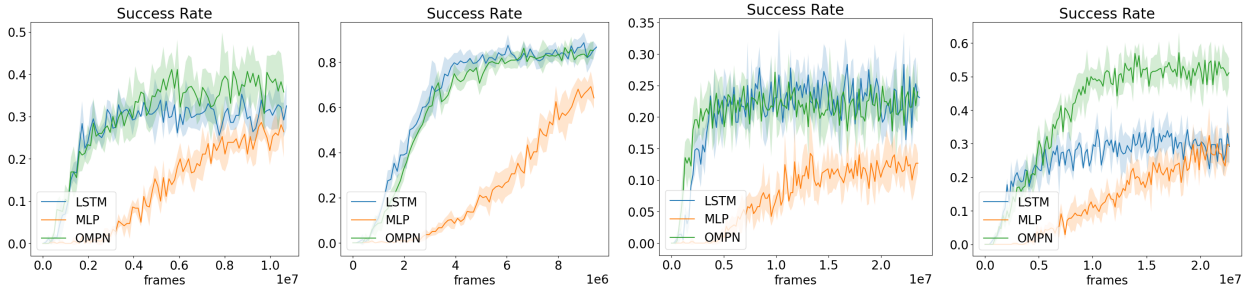


Fig. C.38. More results on π in Craft. The model is able to robustly switch to a higher expanding position at the end of subtasks. The model will also sometimes discover multi-level hierarchy.



(a) Full + NoSketch **(b)** Full + Sketch **(c)** Partial + NoSketch **(d)** Partial + Sketch

Fig. C.39. Behavior Cloning for Craft.

	Full NoSketch			Full Sketch		
	F1(tol=1)	Pre(tol=1)	Rec(tol=1)	F1(tol=1)	Pre(tol=1)	Rec(tol=1)
K=2	62(2.4)	93(3.7)	47(1.8)	62(2.4)	93(3.4)	46(1.8)
K=3	81(2)	95(2.4)	71(1.8)	81(2.2)	95(2.3)	71(2.0)
K=4	95(1.2)	95(1.8)	95(1.8)	98(0.9)	98(1.6)	97(2.1)
K=5	92(1.6)	87(2.8)	99(0.5)	93(6.4)	87(1.1)	99(0.5)
K=6	88(2.9)	78(3.9)	100(0)	88(0.6)	79(0.9)	99(0.2)

Table C.14. Parsing results for full observations with different K

	Parital NoSketch			Partial Sketch		
	F1(tol=1)	Pre(tol=1)	Rec(tol=1)	F1(tol=1)	Pre(tol=1)	Rec(tol=1)
K=2	64(1.8)	96(2.7)	48(1.4)	57(3.7)	88(5.6)	43(2.8)
K=3	83(1.9)	96(1.9)	72(1.9)	74(3.9)	88(4.6)	64(3.5)
K=4	89(4.6)	97(1.6)	82(2.7)	83(8.1)	88(4.4)	80(4.6)
K=5	93(0.8)	90(1.9)	96(2.7)	89(3.2)	85(3.7)	94(3.8)
K=6	90(1.9)	85(1.7)	98(2.3)	87(2.1)	813.4)	97(2.3)

Table C.15. Parsing results for partial observations with different K

C.6. Dial

We show in Table C.16 the task alignment result for different thresholds. We can see that optimal fixed threshold is around 0.4 or 0.5 for Dial, and our threshold selection algorithm could produce competitive results. We demonstrate more expanding positions in Figure C.40. We also show the failure cases, where our selected threshold fails to recover the skill boundary. Nevertheless, we can see that the peak pattern exists. We show the task alignment curves for different thresholds as well as the auto-selected threshold in Figure C.41.

	Align. Acc. at different threshold						
	0.2	0.3	0.4	0.5	0.6	0.7	Auto
OMPN + noenv	57(12.8)	76(9.5)	87(5.7)	89(1.4)	81(5.7)	70(9.7)	87(4)
OMPN + sketch	71(11.6)	81(10.6)	85(7.4)	84(6.6)	76(7.7)	60(6.4)	84(5.7)

Table C.16. Task alignment accuracy for different threshold in Dial. The result for automatic threshold selection is in the last column.

The behavior cloning results are in Figure C.42.

We show the task decomposition results when the K is mis-specified in Table C.17.

	NoSketch			Sketch		
	f1(tol=1)	rec(tol=1)	pre(tol=1)	f1(tol=1)	rec(tol=1)	pre(tol=1)
$K = 2$	59(2.9)	45(2.1)	90(4.4)	58(1.4)	44(1.1)	88(2.1)
$K = 3$	73(5.5))	63(5.1)	85(6.2)	72(3.8)	63(3.4)	84(4.5)
$K = 4$	84(7.1)	81(7.1)	84(6.9)	81(4.8)	81(4.6)	82(5.1)
$K = 5$	86(5.1)	91(3.6)	83(6.3)	83(5.2)	86(5.4)	82(5.3)
$K = 6$	87(4.2)	93(1.5)	82(6.2)	84(5.2)	88(5.5)	81(5.2)
$K = 7$	87(4.1)	93(1.4)	82(6.4)	84(5.2)	89(6)	81(5.2)
$K = 8$	86(4.1)	93(1.6)	82(6.6)	84(5.2)	90(6..1)	81(5.1)

Table C.17. F1 score, recall and precision with tolerance 1 computed at different k.

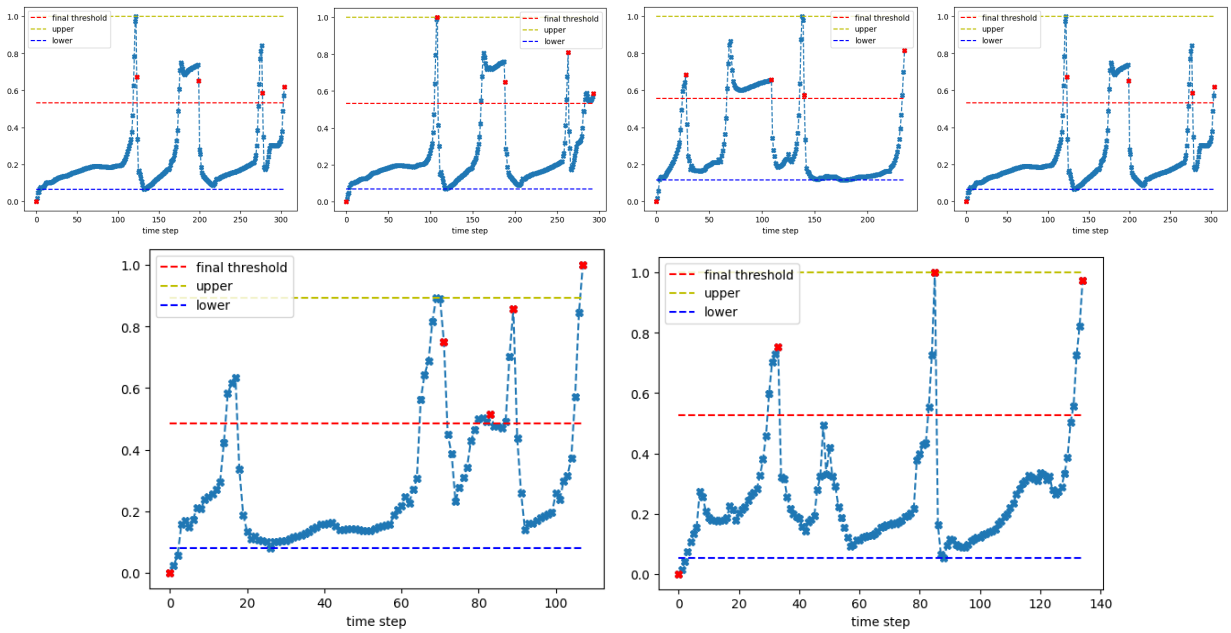
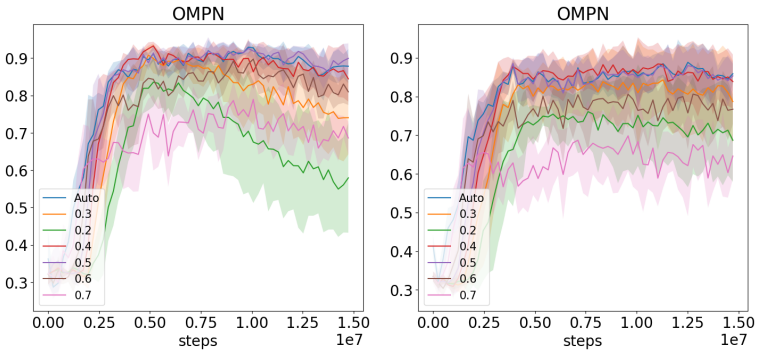


Fig. C.40. More task decomposition visualization in Dial. Our algorithm recovers the skill boundary for the first four trajectories but fails in the last two. Nevertheless, one can see that our model still display the peak patterns for each subtask, and a more advanced thresholding method could be designed to recover the skill boundaries.

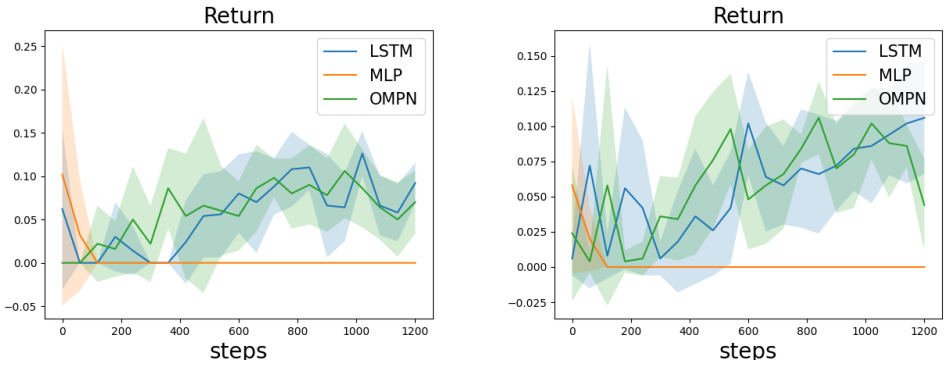


(a) Dial + NoSketch **(b)** Dial + Sketch

Fig. C.41. The learning curves of task alignment accuracy for different thresholds as well as the automatically selected one.

C.7. Hyperparameter Analysis

We discuss the effect of hyper-parameters on the task decomposition results. We summarize the results in Table C.18. We show the detailed alignment accuracy curves for Craft in Figure C.43 and for Dial in Figure C.44.



(a) Dial + NoSketch **(b)** Dial + Sketch

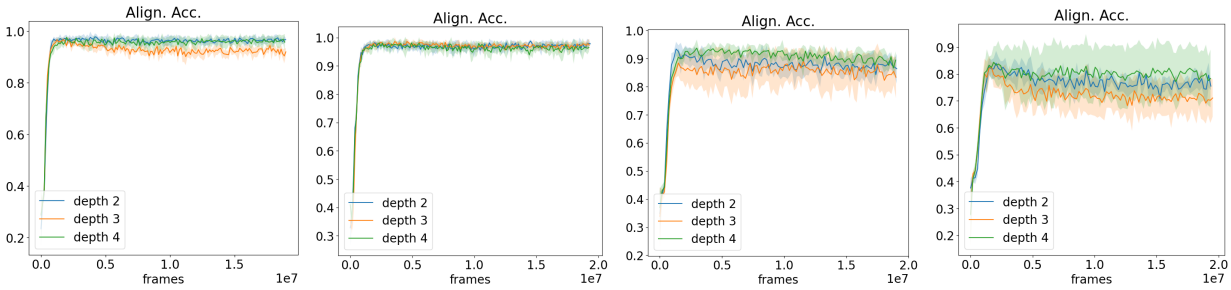
Fig. C.42. The behavior cloning results in Dial domain.

	Craft Full		Craft Partial		Dial	
	NoSketch	Sketch	NoSketch	Sketch	NoSketch	Sketch
$n = 3, m = 64$	93(1.7)	97(1.2)	84(6)	78(10.7)	87(4)	84(5.7)
$n = 2, m = 64$	96(1.4)	96(1.4)	87(3.3)	77(1.2)	88(3.2)	82(5.7)
$n = 4, m = 64$	96(0.6)	96(2.5))	88(3.1)	73(6.2)	86(9.8)	82(10)
$n = 3, m = 32$	92(4.2)	91(10.3)	88(6)	74(5.5)	88(4.9)	83(3.7)
$n = 3, m = 128$	96(1.5)	97(1.2)	86(3)	75(5.7)	87(2.4)	83(6.2)

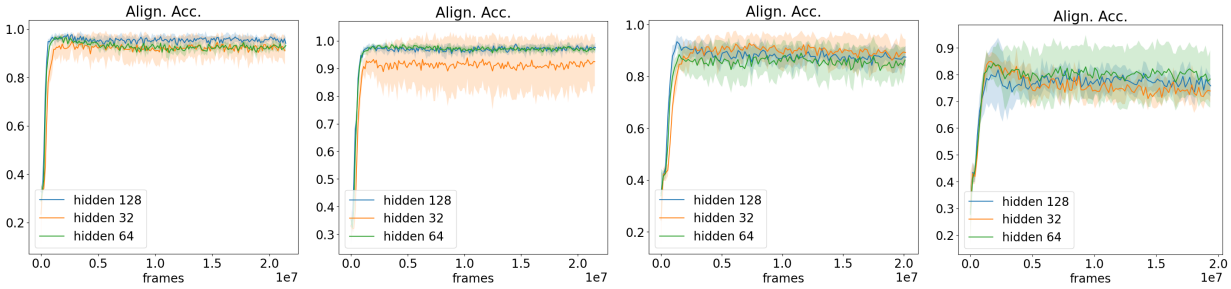
Table C.18. Task alignments accuracy for different memory dimension m and depths n. The default setting is on the first row. The next two rows change the depths, while the last two rows change the memory dimension. The number in the parenthesis is the standard deviation.

C.8. Qualitative Results on Kitchen

We display the qualitative results for Kitchen. We use the demonstration released by Gupta et al. (2019b). Since they do not provide ground truth task boundaries, we provide the qualitative results. We use a hidden size of 64 and the memory depths of 2. We set the learning rate to be 0.0001 with Adam and the BPTT length to be 150. The input dimension is 60 and action dim is 9. We train our model in the unsupervised (NoSketch) setting. We show the visualization of our expanding position, in a similar fashion of the Dial domain, in Figure C.46 and Figure C.47. We show the visualizations for other trajectories in Figure C.48

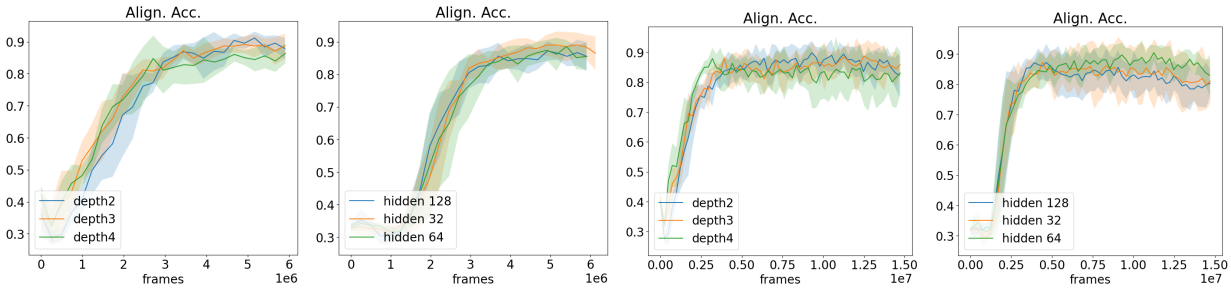


(a) Full + NoSketch **(b)** Full + Sketch **(c)** Partial + NoSketch **(d)** Partial + Sketch



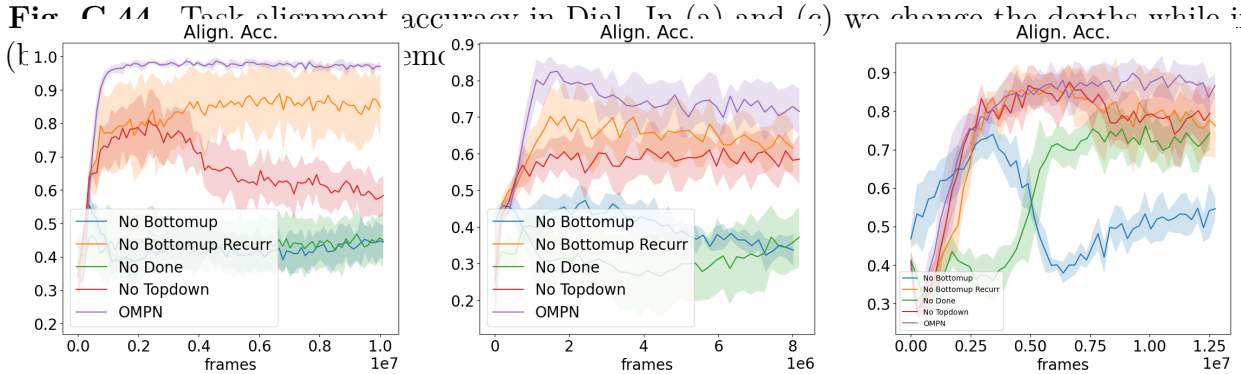
(e) Full + NoSketch **(f)** Full + Sketch **(g)** Partial + NoSketch **(h)** Partial + Sketch

Fig. C.43. Task alignment accuracy with varying hyperparameters in Craft. We change the depths in the first row and memory size in the second row.



(a) Dial + NoSketch **(b)** Dial + NoSketch **(c)** Dial + Sketch **(d)** Dial + Sketch

Fig. C.44. Task alignment accuracy in Dial. In (a) and (c) we change the depths while in (b) we change the memory size.



(a) Sketch + Craft(Full) **(b)** Sketch + Craft(Partial) **(c)** Sketch + Dial

Fig. C.45. Ablation study for task alignment accuracy for Sketch settings

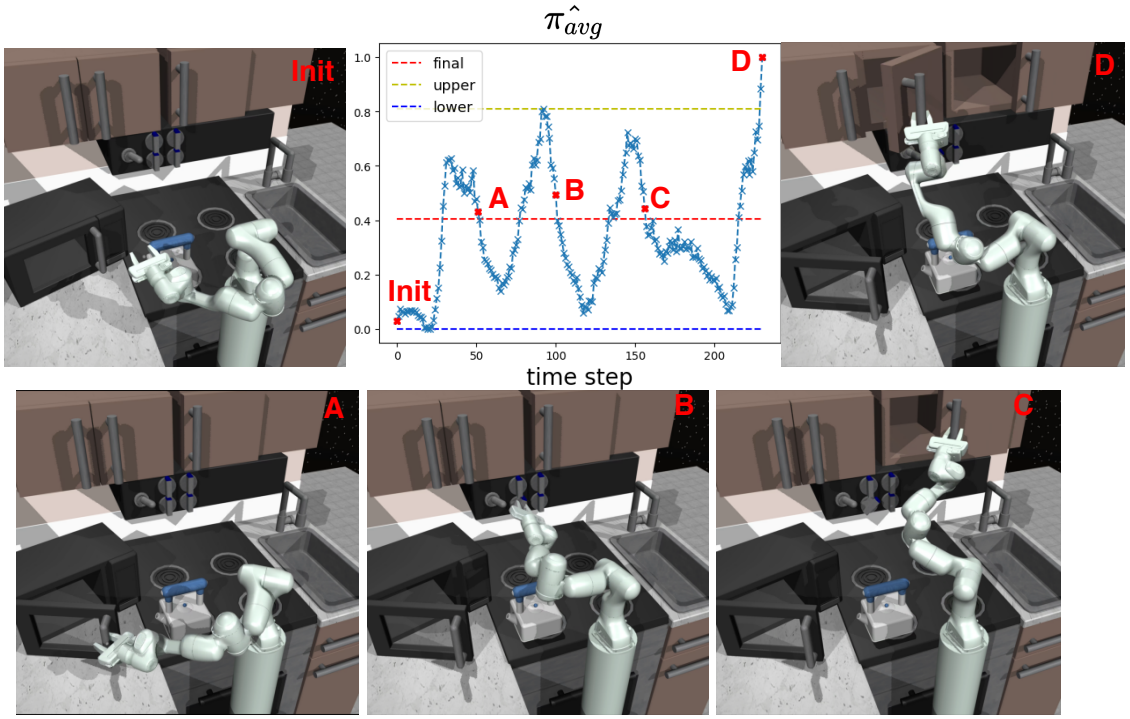


Fig. C.46. Visualization of the learnt expanding positions of Kitchen. The subtasks are Microwave, Bottom Knob, Hinge Cabinet and Slider.

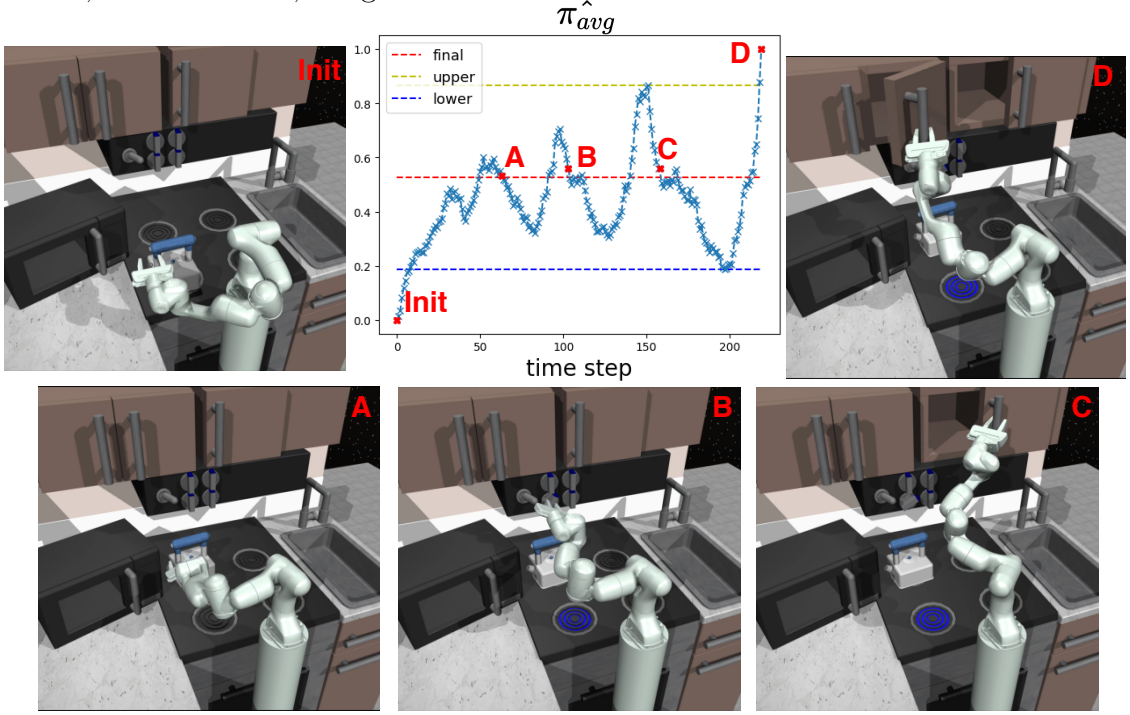


Fig. C.47. Visualization of the learnt expanding positions of Kitchen. The subtasks are Kettle, Bottom Knob, Hinge Cabinet and Slider.

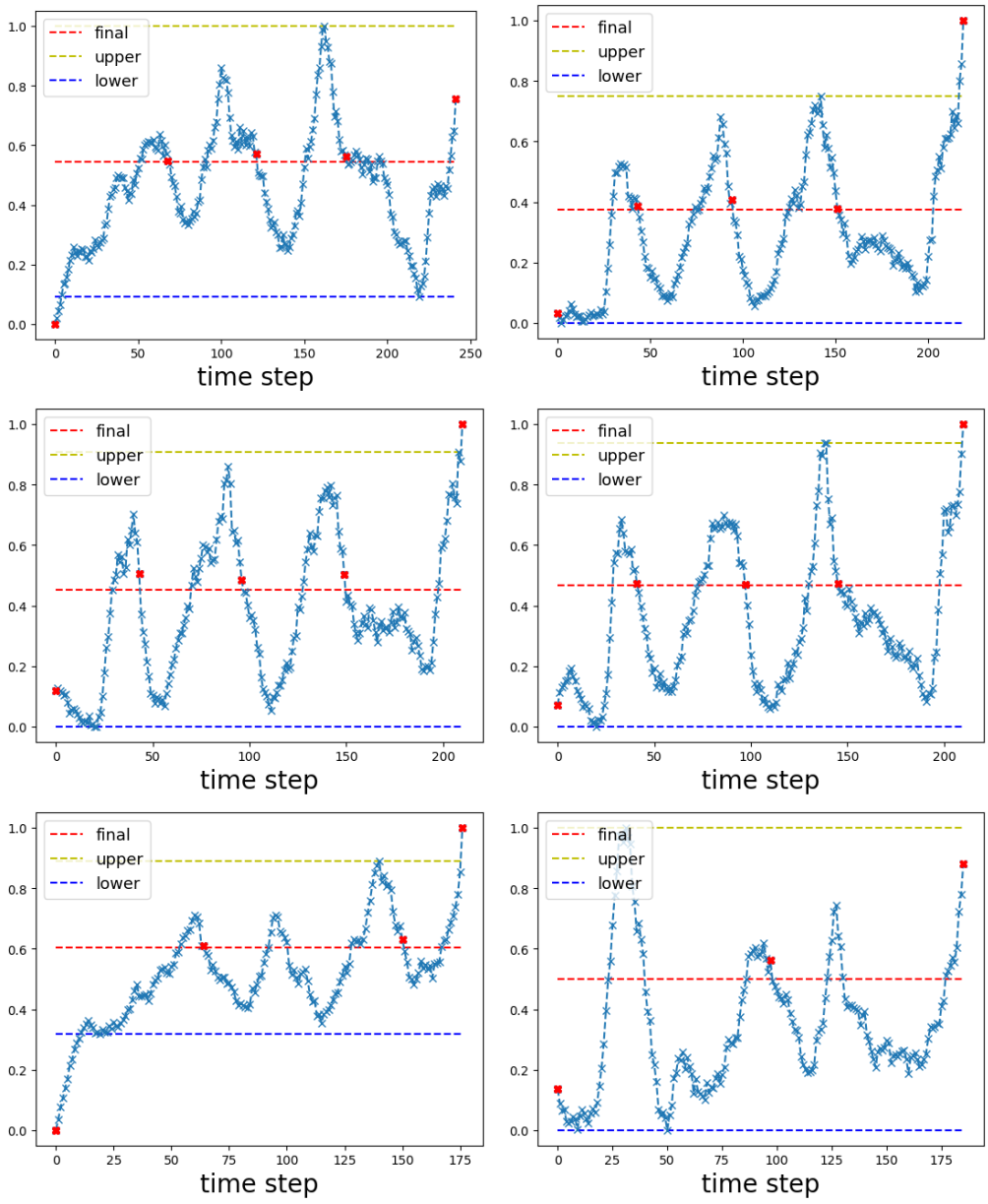


Fig. C.48. More task decomposition visualization in Kitchen. Our algorithm seems to recover the skill boundary for the first four trajectories but fails in the last two. Nevertheless, one can see that our model still display the peak patterns for each subtask, and a more advanced post-processing thresholding method might be able to recover the task boundary.

Appendix D

Supplementary Material for Article Four

D.1. Futher Discussion on Current SSL Practice

We further discuss the motivation behind our framework, and how can approach some of the SSL practice throught the lense of expressiveness and learnability.

Contrastive Learning. With our new view on representation quality by expressiveness-learnability, we can try to analyze some of the practice in the contrastive learning and see why it works and fails. Contrastive learning is a typical example of such a trade-off, and it has the following objective:

$$\mathbb{E}_{x,x^+,x^-} \log \left(\frac{e^{\mathcal{F}(x)^T \mathcal{F}(x^+)}}{e^{\mathcal{F}(x)^T \mathcal{F}(x^+)} + e^{\mathcal{F}(x)^T \mathcal{F}(x^-)}} \right)$$

It aims to output distinguishable feature vectors between positive and negative examples which turns out to increase the expressiveness. Meanwhile, it aims to make the output vectors among positive examples to be similar by which a higher value of learnability is ensured, since ideally the another learner only need to see one example to generalize to other positive examples. As a result, the degree of learnability here can be controlled by the definition of positive examples. A more diverse selection of the positive examples should make the $\mathcal{F}$ leaning toward the learnability side, since each example could represent a larger fraction of the inputs. It is well acknowledged in the self-supervised community that a diverse data augmentation is an essential part of the contrastive learning, and under this framework, it is a source of the learnability pressure.

Clustering-based Learning. Perhaps our expressiveness-learnability framework is more obvious in the clustering-based algorithm like DeepCluster (Caron et al., 2018) and SwAV (Caron et al., 2020). In these algorithms, you learn not only $\mathcal{F}$ but also a bank of cluster centroids $\mathcal{C}$ as well as assignments $\tilde{y} = \mathcal{C}^T \mathcal{F}(x)$. As a result, we can also view these algorithms as finding $\tilde{y} = \mathcal{G}(x)$, where $\mathcal{G}$ is composed of $\mathcal{F}$ and $\mathcal{C}$.

In these methods, the expressiveness is achieved by using a large number of clusters as well as some uniformity constraint, and this can ensure each data has a distinctive cluster assignment. SwAV achieves it with the SinkHorn iterations, while the DeepCluster, at least in the early version, would balance the sampling ratio of images to make the cluster uniform as well as resolving the empty cluster issues.

The learnability requirements are implicitly enforced as well. Since the space of $\tilde{y}$ is usually less than x, there are usually multiple data being assigned to the same cluster. This already ensure that a new learner should able to see a few examples to generalize within the cluster. In addition, these algorithms also leverage the data augmentation, so that $\mathcal{G}(x)$ and $\mathcal{G}(x^+)$ should have the same $\tilde{y}$, which further improves the learnability.

D.2. Intrinsic Dimension and Entropy Estimation

In this section, we briefly discuss how Intrinsic Dimension can be connected to Entropy Estimation for highly non-linear manifolds under the MLE principle.

Suppose we have i.i.d. dataset $X = \{X_1, X_2, ..., X_n\}$. Let $S(x,R)$ be a sphere around x and radius R. We assume when R is small, the local density is constant (local uniformity). That is we assume there is a constant $f(x)$ inside $S(x,R)$. Then we denote $N(r,x)$ as the number of data that appears inside the sphere $S(x,r)$, out of n data. When r goes from 0 to R, we assume the stochastic process $\{N(r,x)\}$ can be described as a homogeneous Poisson Process inside $S(x,R)$. Then the rate of such process w.r.t. r is

$$\lambda(r) = f(x)V(D)Dt^{D-1}$$

where D is the intrinsic dimension, and $V(D) = \pi^{D/2}[\Gamma(D/2+1)]^{-1}$ is the volume of unit sphere in R^D.

Let's say D and $\theta = \log f(x)$ is our parameters for this model. Then the likelihood is the conditional probability of the observation $\{N(r,x)\}$. That is

$$L(D,\theta) = \int_0^R \log \lambda(r) dN(r,x) - \int_0^R \lambda(r) dr$$

This is derived in the other paper that I do not fully understand now. But let's say this is true, we can estimate both parameter with MLE

$$\frac{\partial L}{\partial D} = 0, \frac{\partial L}{\partial \theta} = 0$$

This gives us the estimation of D w.r.t. the choice of sphere $S(x,R)$ as

$$\hat{D}(x,R) = \left[\frac{1}{N(R,x)} \sum_{j=1}^{N(R,x)} \log \frac{R}{T_j(x)} \right]^{-1}$$

where $T_j(x)$ is the distance of j-th neighbor to x. If we control R by the predefined the number of neighbor k, then

$$\hat{D}(x,k) = \left[\frac{1}{k-1} \sum_{j=1}^{k-1} \log \frac{T_k(x)}{T_j(x)} \right]^{-1}$$

which is the expected log distance ratio between k-th neighbor and other neighbors.

Also from above, we have

$$\frac{N(R,x)}{R^D} = e^{\theta} V(D)$$

to be true for all x. Therefore

$$\log f(x) = \theta = \log \frac{N(R)}{R^D V(D)} = \log N(R) - D \log R - \log V(D)$$

which suggest that given the local intrinsic dimension, we have an estimation of the local density $f(x)$. If we also use number of neighbors k to represent R, and use our local intrinsic dimension estimate $\hat{D}(x,k)$, we have estimated local density

$$\log \hat{f}(x) = \log(k-1) - \hat{D}(x,k) \log T_k(x) - \log V(\hat{D}(x,k))$$

Then our estimate of entropy based on dataset can be

$$\hat{H}(X,k) = -\frac{1}{n} \sum_{i}^{n} \log \hat{f}(X_i)$$

$$= \frac{1}{n} \sum_{i}^{n} \Big(\hat{D}(X_i,k) \log T_k(X_i) + \log V(\hat{D}(X_i,k)) \Big) - \log(k-1)$$

D.3. Cluster Learnability and Prequential Codelength

Note that Epn. 6.2.2 can be connected to the compression lower bound of the emerged cluster assignments datasets $\{z_i, \tilde{y}_i\}_{i=1}^{N}$ via prequential coding (Dawid, 1984). The prequential codelength is defined as

$$L_{preq} = -\sum_{i} \log p(\tilde{y}_i | (z_{k \le i}, \tilde{y}_{k<i})) \tag{D.3.1}$$

where $p(\tilde{y}_i | (z_{k \le i}, \tilde{y}_{k<i}))$ is a conditional model, which can be computed from our KNN classifier probabilities. If Eqn. 6.2.2 is the online learning accuracy, then Eqn. D.3.1 is online learning

cross-entropy loss. Higher CL lead to lower prequential compression bounds on the emerged clusters.

Name	Method	Architecture
infomin	Unsupervised	ResNet
insdis	Unsupervised	ResNet
pcl_v1	Unsupervised	ResNet
pcl_v2	Unsupervised	ResNet
pirl	Unsupervised	ResNet
byol	Unsupervised	ResNet
deepclusterv2_400ep_2x224	Unsupervised	ResNet
deepclusterv2_400ep	Unsupervised	ResNet
deepclusterv2_800ep	Unsupervised	ResNet
dino_deitsmall8	Unsupervised	ViT
dino_deitsmall16	Unsupervised	ViT
dino_vitbase8	Unsupervised	ViT
dino_vitbase16	Unsupervised	ViT
dino_xcit_small_12_p16	Unsupervised	XCiT
moco_v1_200ep	Unsupervised	ResNet
noco_v2_200ep	Unsupervised	ResNet
moco_v2_800ep	Unsupervised	ResNet
resnet18	Supervised	ResNet
resnet34	Supervised	ResNet
resnet50	Supervised	ResNet
selav2_400ep_2x224	Unsupervised	ResNet
selav2_400ep	Unsupervised	ResNet
simclrv2_r501xsk0	Unsupervised	ResNet
simclrv2_r501xsk1	Unsupervised	ResNet
simclrv1_resnet50_1x	Unsupervised	ResNet
swav_100ep	Unsupervised	ResNet
swav_200ep	Unsupervised	ResNet
swav_800ep	Unsupervised	ResNet

Table D.19. Full list of the pretrained checkpoints collected. The list of SSL methods are: MoCo-v1 (He et al., 2020b), MoCo-v2 (Chen et al., 2020c), SeLA-v2 (Caron et al., 2020), DeepCluster-v2 (Caron et al., 2020), SwAV (Caron et al., 2020), DINO (Caron et al., 2021), SimCLR v2 (Chen et al., 2020b), SimCLR-v1 (Chen et al., 2020a), PCL-v1 (Li et al., 2021), PCL-v2 (Li et al., 2021), PIRL (Misra and Maaten, 2020), BYOL (Grill et al., 2020), InfoMin (Tian et al., 2020), InsDis (Wu et al., 2018).

foods	0.37	0.12	0.75	0.72	0.27	-0.04	0.17	-0.36	-0.22	-0.35	0.01	0.51
flowers	0.22	0.30	0.81	0.73	0.21	0.07	0.25	-0.36	-0.14	-0.35	0.14	0.46
pets	0.40	0.07	0.71	0.70	0.33	-0.09	0.23	-0.25	-0.31	-0.25	-0.08	0.54
caltech101	0.36	-0.01	0.56	0.52	0.32	-0.17	0.11	-0.19	-0.26	-0.18	-0.05	0.67
stl10	0.68	-0.20	0.56	0.64	0.30	-0.22	0.02	-0.35	-0.46	-0.34	-0.21	0.41
aircraft	0.14	0.30	0.67	0.57	0.17	0.02	0.21	-0.28	-0.14	-0.27	0.16	0.57
cars	0.20	0.31	0.81	0.72	0.18	0.12	0.32	-0.28	-0.11	-0.28	0.16	0.48
Avg	0.34	0.13	0.70	0.66	0.25	-0.04	0.19	-0.29	-0.24	-0.29	0.02	0.52

Table D.20. Full Kendall Ranking Coefficient results on Out-of-Domain Tasks when ImageNet labels are not acquired. The CLID predictor has the best predictive performance for transfer tasks.

foods	0.86	0.65	0.59	0.84	0.81	**0.88**	0.84	0.57	0.32	0.56	0.19	0.32	0.18	0.55
flowers	0.70	0.50	0.68	**0.79**	0.74	0.74	0.71	0.46	0.30	0.58	0.14	0.27	0.14	0.60
pets	0.75	0.62	0.51	0.77	**0.78**	0.66	0.72	0.63	0.27	0.55	0.24	0.24	0.25	0.41
caltech101	0.62	0.53	0.40	0.59	0.58	0.60	0.66	0.60	0.12	0.41	0.23	0.15	0.25	0.37
stl10	0.76	**0.84**	0.28	0.71	0.71	0.61	0.71	0.61	0.13	0.36	0.11	0.11	0.12	0.29
aircraft	0.60	0.40	0.63	0.66	0.61	**0.68**	0.64	0.41	0.21	0.50	0.18	0.25	0.17	0.56
cars	0.68	0.47	0.72	**0.79**	0.74	0.72	0.74	0.46	0.33	0.64	0.21	0.33	0.21	0.63
Avg	0.71	0.57	0.55	**0.74**	0.71	0.70	0.72	0.53	0.24	0.52	0.19	0.24	0.19	0.49

Table D.21. Full Kendall Ranking Coefficient results on Out-of-Domain Tasks when ImageNet labels are obtained. The results are computed with the *joint rank product* between the predictors and the ImageNet accuracy. While the ImageNet accuracy can predict the transfer accuracy reasonabley, the proposed CLID predictor could further enhance the result.

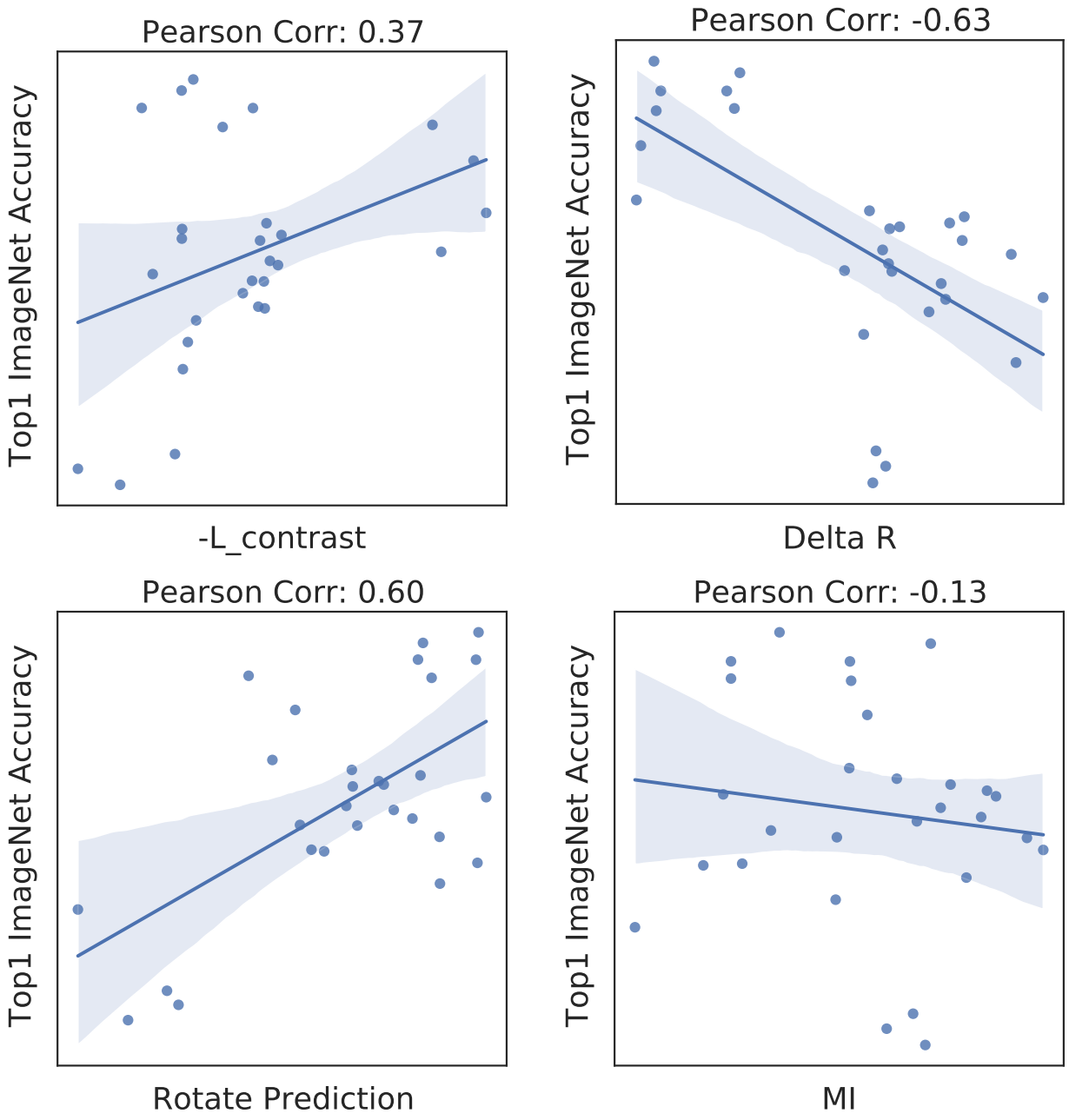


Fig. D.49. Regression results for baselines w.r.t ImageNet accuracies.