



AFRL-RI-RS-TR-2021-134

PROBABILISTIC LABEL-EFFICIENT DEEP GENERATIVE STRUCTURES (PLEDGES)

CHARLES RIVER ANALYTICS INC.

AUGUST 2021

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (<http://www.dtic.mil>).

AFRL-RI-RS-TR-2021-134 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /

PETER A JEDRYSIK
Work Unit Manager

/ S /

JULIE BRICHACEK
Chief, Information Systems Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.					
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) AUGUST 2021		2. REPORT TYPE FINAL TECHNICAL REPORT		3. DATES COVERED (From - To) AUG 2019 – APR 2021	
4. TITLE AND SUBTITLE PROBABILISTIC LABEL-EFFICIENT DEEP GENERATIVE STRUCTURES (PLEDGES)				5a. CONTRACT NUMBER FA8750-19-C-0515	
				5b. GRANT NUMBER N/A	
				5c. PROGRAM ELEMENT NUMBER 61101E	
6. AUTHOR(S) Avi Pfeffer Brad Rosenberg Ishaan Shah, Catherine Call Kirstin Bibbiani Deniz Erdogmus Frank Wood Leonid Sigal Sameer Singh Jan Willem van de Meent				5d. PROJECT NUMBER LWLL	
				5e. TASK NUMBER 00	
				5f. WORK UNIT NUMBER 02	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Charles River Analytics Inc. 625 Mount Auburn Street Cambridge MA 02138				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RISB DARPA/I2O 525 Brooks Road 675 N. Randolph St. Rome NY 13441-4505 Arlington, VA 22203-2114				10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/RI	
				11. SPONSOR/MONITOR'S REPORT NUMBER AFRL-RI-RS-TR-2021-134	
12. DISTRIBUTION AVAILABILITY STATEMENT Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT The enormous commercial success of machine learning has failed to translate into high performance military applications. Though deep learning is beginning to show impressive results in several specific military tasks, current capabilities fail to perform sufficiently due to their requirement for extremely large, labeled training sets. Under the Probabilistic Label-Efficient Deep Generative Structures (PLEDGES) project, part of DARPA's Learning with Less Labels (LwLL) program, we pursued pioneering research at the interface of probabilistic modeling and deep learning. We investigated three main directions. First, we developed Structured Deep Probabilistic Models (SDPMs), which define structured and disentangled joint probability distributions over unlabeled data observations. Second, we pursued efficient and accurate algorithms for high-capacity probabilistic models. Third, we developed probabilistic variants of deep learning models for semi-supervised and weakly supervised learning.					
15. SUBJECT TERMS Probabilistic models, Scruff, structured deep probabilistic models, semi-supervised learning, model transfer, active learning, variational methods, separable belief propagation					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 71	19a. NAME OF RESPONSIBLE PERSON PETER A. JEDRYSIK
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U			19b. TELEPHONE NUMBER (Include area code) N/A

TABLE OF CONTENTS

List of Figures	ii
List of Tables	iii
Acknowledgments.....	iv
1.0 Summary	1
2.0 Introduction.....	3
2.1 Problem Description	3
2.2 Technical Approach	4
2.3 Technical Objectives.....	5
2.4 Related Work	5
3.0 Methods, Assumptions, and Procedures	7
3.1 Probabilistic Model Components.....	7
3.1.1. Structured Deep Probabilistic Models	7
3.1.2. Tractable High-Capacity Models	14
3.2 Model Transfer.....	26
3.2.1. Overview	26
3.2.2. Simple CNAPS Algorithm for Few-shot Visual Classification.....	26
3.2.3. Extending Simple CNAPS to Few-Shot Object Detection	27
3.2.4. Few-Shot/Zero-Shot Object Detection	29
3.3 Semi-Supervised Learning.....	30
3.3.1. Overview	30
3.3.2. Baseline Methods.....	30
3.3.3. Semi-Supervised Extensions of Simple CNAPS	34
3.3.4. Other Semi-Supervised Approaches Explored	38
3.4 Active Learning	42
3.4.1. Overview	42
3.4.2. Active Learning Evaluation Preparation.....	44
3.4.3. Active Learning with Simple CNAPS Experimentation	45
3.4.4. Data Influence	46
4.0 Results and Discussion	49
4.1 Summary of Results	49
4.2 Internal Evaluation Results	49
4.3 Phase I Evaluation.....	50
5.0 Conclusions.....	51
6.0 Recommendations.....	52
7.0 References.....	53
Appendix A. Publications.....	56
Appendix B. Meetings, Presentations, and Personnel Changes	63
8.0 List of Symbols, Abbreviations, and Acronyms	64

LIST OF FIGURES

Figure 1: SDPM for Multiple MNIST Digits.	9
Figure 2: Results for conjugate energy-based models	11
Figure 3: Latent Space of a GMM-VAE.....	12
Figure 4: Constructing a Separable DBN	14
Figure 5: Error of Separable and Ordinary BP on Marginal Query.....	20
Figure 6: Error of Separable and Ordinary BP on Conditional Query	20
Figure 7: Inference Times for Various Algorithms (Log Scale).....	21
Figure 8: Training Time for Separable Models	22
Figure 9: Error of Separable and Ordinary BP for Models with Variable Range Size.....	22
Figure 10: Relative Error of Algorithms on Square Networks	24
Figure 11: Runtime of Algorithms on Square Models.....	24
Figure 12: Visualization of the Latent Space for a VAE Trained on MNIST. (<i>Left</i>) Standard VAE; (<i>right</i>) VAE + VIB Objective.....	33
Figure 13: Impact of β on the Quality of the Latent Representation	33
Figure 14: Impact of Dimensionality of $\mathbf{z}$ on the Quality of the Latent Representation	34
Figure 15: Supervised Meta-Learning Architecture	34
Figure 16: Meta-Learning a Semi-Supervised VAE.....	35
Figure 17: Classification Results: Transductive (top) and Inductive (bottom).....	39
Figure 18: Separable DBN with Thin Layer for Label	40
Figure 19: Limitations of Active Learning	46
Figure 20: Three-Class, Unbalanced Dataset.....	47
Figure 21: Results on Three-Class Dataset	47
Figure 22: Results on CIFAR-10	48

LIST OF TABLES

Table 1: Preliminary Results for Few-Shot Object Detection with Weak Supervision.....	30
Table 2: Performance of SSL Objectives on Various Datasets	32
Table 3: Transductive Learning Results	36
Table 4: Weak Supervision Results	37

ACKNOWLEDGMENTS

This work was performed under AFRL/DARPA-I2O contract number FA8750-19-C-0515. The authors thank Wade Shen and Bruce Draper for their support and direction on this project. We would also like to thank our extended project team, who were essential to making PLEDGES a successful effort, including Jarred Barber, Peyman Bateni, Andac Demir, Babak Esmaeili, Raghav Goyal, Siddhesh Khandelwal, Aziz Kocaogullari, Pouya Pezeshkpour, Niklas Smedemark-Marguiles, Byron Wallace, Hao Wu, and Heiko Zimmermann. We thank Kimberley Brady for assistance in putting together this report.

1.0 SUMMARY

The enormous commercial success of machine learning has failed to translate into high performance military applications. Though deep learning is beginning to show impressive results in several specific military tasks, current capabilities fail to perform sufficiently due to their requirement for extremely large, labeled training sets. The military needs a world-class team of artificial intelligence and machine learning experts to create the requisite innovations in label-efficient semi-supervised learning, model transfer, and active learning needed to solve real-world military learning problems. To address this need, Charles River Analytics assembled an outstanding team of probabilistic modeling and machine learning researchers and practitioners to provide Probabilistic Label-Efficient Deep Generative Structures (PLEDGES).

Under this effort, we pursued pioneering research at the interface of probabilistic modeling and deep learning. Probabilistic models are beneficial for LwLL because unlike neural networks, which require a label-based error function, they can use any kind of observation, which can include labels but is not dependent on them. Probabilistic models are therefore useful for semi-supervised learning because they can learn as much knowledge as possible from unlabeled data and augment it using a small amount of labeled data. They are also useful for model transfer because they are structured using semantically meaningful components, which can be combined and reapplied to transfer models efficiently to new domains. Probabilistic models also directly support active learning by providing well-calibrated uncertainty estimates, which are the basis for informed decisions about which instances to label.

To ensure high performance of probabilistic models, they must be integrated with deep learning representations. We investigated three main directions. First, we developed Structured Deep Probabilistic Models (SDPMs), which define structured and disentangled joint probability distributions over unlabeled data observations. SDPMs provide neural-like representations, enabling comparable performance to state-of-the-art neural methods while learning from few labels. Second, we pursued efficient and accurate algorithms for high-capacity probabilistic models that achieve orders of magnitude improvement over traditional probabilistic models, enabling the use of models that mirror the structure of state-of-the-art neural networks. Third, we developed probabilistic variants of deep learning models for semi-supervised and weakly supervised learning, including meta-learning methods for few-shot visual classification and object detection, as well as weakly supervised methods for zero-shot object detection.

Specifically, under the PLEDGES effort, we:

- Developed probabilistic models and model components to support learning with less labels. These models include variational SDPMs and tractable high-capacity models.
- Developed methods to learn a model from a large amount of unlabeled training data and a small amount of labeled training data.
- Developed methods to transfer a model learned from a general-purpose dataset to a specific application using very few labels for the new application.
- Developed methods to recommend instances for labeling by an expert.

We used these probabilistic models and methods to provide the key ingredients of LwLL: semi-supervised learning, model transfer, and active learning. We were able to make strong advances for both SDPMs and tractable high-capacity models and our work shows that both methods have significant promise, with considerable accuracy and scalability improvements over previous probabilistic models. We also showed that probabilistic formulations of neural models

can lead to significant advances. Our Simple CNAPS method achieved a dramatic improvement over state-of-the-art methods, including CNAPS, at few-shot visual classification. Simple CNAPS also formed the basis of a variety of extensions, including a transductive version that showed further improvements. For active learning, our foundational work led to a suite of new methods based on data importance, and we were able to show improvement over existing methods based on uncertainty sampling.

2.0 INTRODUCTION

2.1 Problem Description

The enormous commercial application success of machine learning has failed to translate into high performance military applications. Though deep learning is beginning to show impressive results in several specific military tasks, current capabilities fail to perform sufficiently due to their requirement for extremely large, labeled training sets. Unlike commercial applications, most important military applications have target objects, events, or untranslated words that are often novel and do not have large-scale ground truth labels. Also, labeling expertise is limited, which prohibits crowd-sourced creation of labeled data sets. Military learning-based applications require new approaches to unsupervised and semi-supervised learning that achieve the performance of deep learning systems but with as few as 10 to 100 labels. Additionally, learning algorithms must efficiently use limited access to military experts and training data, for example, by actively identifying the most representative and least confident learned instances. Finally, the learned models used to identify objects and activities of military interest must have a new level of component reuse to adapt through model recombination. Machine learning applications are notorious for their jungle of mechanisms that are complex to extend and fail to generalize across tasks and applications. Addressing these challenges requires fundamental new thinking to surpass current machine learning limitations, with requisite innovations in label-efficient semi-supervised learning, model transfer, and active learning needed to solve real-world military learning problems.

Today, approaches based on deep neural networks dominate research in machine learning. Despite numerous successes, neural networks have several critical limitations, including that they are extremely data hungry and require massive amounts of labeled data. This makes probabilistic methods, which can require significantly less labeled data, extremely attractive. Probabilistic models are beneficial for label-efficient learning because unlike neural networks, which require a label-based error function, they can use any kind of observation, which can include labels but is not dependent on them. Probabilistic models are therefore useful for semi-supervised learning because they can learn as much knowledge as possible from unlabeled data and augment it using a small amount of labeled data. They are also useful for model transfer because they are structured using semantically meaningful components, which can be combined and reapplied to transfer models efficiently to new domains. Probabilistic models also directly support active learning by providing well-calibrated uncertainty estimates, which are the basis for informed decisions about which instances to label.

In fact, probabilistic models have been one of the primary methods for unsupervised and semi-supervised learning, with methods like k-means clustering being extremely widely used. As this example shows, however, the probabilistic models used for unsupervised learning and semi-supervised learning have tended to be simple; k-means is essentially a mixture of Gaussians. In the past, the performance of probabilistic methods has lagged behind neural networks due primarily to the difficulty of scaling up inference and learning to rich models. Therefore, to achieve the required practical effectiveness for significant DoD applications, probabilistic models must be integrated with deep learning approaches.

We expect the results of our effort will create revolutionary advances in the ability to infer with and learn rich probabilistic models efficiently, enabling deep probabilistic approaches, at long last, to be viable alternatives to non-probabilistic deep learning for real-world problems.

These advances will lead to radically better methods for semi-supervised learning, model transfer, and active learning than current neural network-based approaches.

2.2 Technical Approach

We built on recent work by our team on integrating probabilistic models with deep representations. We pursued three parallel directions. First, we developed probabilistic variants of deep learning models for semi-supervised and weakly supervised learning. We call these models Structured Deep Probabilistic Models (SDPMs). This work built on recent innovations in languages, such as Edward, Pyro, and Probabilistic Torch, the latter of which is led by co-PI Jan-Willem van de Meent. These frameworks use variational methods to train probabilistic models in which neural networks define conditional distributions, combining the flexibility of deep learning with the data-efficiency of probabilistic models. Initial work with these methods proved remarkably effective. With only 10 labels per class, we achieved classification accuracies of over 90% on datasets, such as the Modified National Institute of Standards and Technology (MNIST) dataset, using Probabilistic Torch. During the effort, we extended these methods in a variety of ways to obtain significantly greater accuracy than other probabilistic variational methods.

For our second direction, we pursued efficient and accurate algorithms for high-capacity probabilistic models that mirror the structure of state-of-the-art neural networks while remaining fully probabilistic. This approach was based on recent work by PI Avi Pfeffer, which shows that any directed probabilistic model can be converted into a secondary model in which inference is linear-time and accurate, by encoding information about the joint distribution, using a method called Separable Belief Propagation (SBP). The success of deep learning, which relies on the fact that most observed data lives on a low-dimensional manifold that can be sparsely represented, provides us with strong reason to believe that the secondary model will be practically small for real-world military applications. As a result, high-capacity probabilistic models using SBP can achieve similar tractability to neural network approaches while achieving better performance on low-label real-world DoD problems.

For the third direction, we developed probabilistic reinterpretations and variations of existing non-probabilistic deep learning methods. In particular, we developed the Simple CNAPS algorithm, which is a meta-learning algorithm for few-shot visual classification with a neural feature extractor and probabilistic classifier that is learned from data with limited labels. We also developed a transductive version of Simple CNAPS, as well as a weakly supervised learning method for zero-shot/any-shot object detection.

We used the probabilistic models developed under these two threads to provide the key ingredients of LwLL: semi-supervised learning, model transfer, and active learning. Success at semi-supervised learning relies on learning most knowledge about the domain without labels, leaving only a small number of parameters to be learned from the labeled data. Probabilistic models naturally lend themselves to semi-supervised learning, because a rich, highly structured model can be learned from unlabeled data alone. This model can be a complex network with many parameters that can be fit accurately from a large amount of unlabeled data, and then augmented by a thin labeling layer with few parameters to learn from few instances. We augmented this insight with pairwise learning, in which we compare and contrast different data instances to learn about their comparative labels, even when they are not labeled themselves. Pairwise comparisons are less noisy and more informative than class labels, thereby making better use of sparse labels. In initial work, we established that, by incorporating comparisons, we

can train and optimize a neural network with 5,974,577 parameters on a dataset of only 80 samples.

For model transfer, probabilistic approaches let us compose semantically meaningful model components and reapply them to new situations. We built on this insight with several practical ideas. Non-parametric methods enable us to recognize new classes of object even with zero labeled instances of that class by recognizing clusters of similar instances that do not belong to any known class. Probabilistic zero-shot learning methods also enable us to use side information to identify instances of a class with no labeled instances. Reprogramming methods enable us to transfer the original model to a new application by mapping concepts between the applications, such as inputs, outputs, and internal representations of concepts.

For active learning, we used a decision-theoretic approach based on our probabilistic models. Most existing work on active learning has two goals: to pick least confident instances, and to pick most representative instances. Although combining these objectives has been achieved for simpler tasks where a similarity metric can be directly defined, measures of similarity are not readily available for most practical applications. We developed a new approach to active learning based on the concept of data importance and integrated these methods with SDPMs.

2.3 Technical Objectives

Charles River Analytics, along with our collaborators at Northeastern University, the University of British Columbia (UBC), and the University of California, Irvine (UCI), developed Probabilistic Label-Efficient Deep Generative Structures (PLEDGES) for Learning with Less Labels (LwLL). We pursued pioneering research at the interface of probabilistic modeling and deep learning. Unlike neural networks, which require a label-based error function, probabilistic models can use any kind of observation, which can include labels but is not dependent on them. We developed Structured Deep Probabilistic Models (SDPMs), which define structured and disentangled joint probability distributions over unlabeled data observations; tractable high-capacity probabilistic models, which enable fast and accurate inference over large networks; and probabilistic reinterpretations and variations of neural algorithms.

Our research goals for SDPMs followed three lines. First, for semi-supervised learning, SDPMs provide neural-like representations, enabling comparable performance to state-of-the-art neural methods while learning from few labels. SDPMs are therefore ideal for semi-supervised learning because they can learn as much knowledge as possible from unlabeled data and augment it using a small amount of labeled data. Second, to transfer learned knowledge from one model to a new model with few labels, SDPMs are also ideal for model transfer because they are structured using semantically meaningful components, which can be combined and reapplied to transfer models efficiently to new domains. Third, for active learning in which the learning system can ask questions about the data, probabilistic models also directly support active learning by providing well-calibrated uncertainty estimates, which are the basis for informed decisions about which instances to label.

2.4 Related Work

To develop practical probabilistic models for large scale applications, we built on recent developments in variational inference. In this effort, we developed new variants of *amortized* variational methods [2–4], which train a neural network to carry out inference using stochastic gradient descent. Amortized variational methods have been operationalized in languages, such as Edward [5], Pyro [6], and Probabilistic Torch [7]. Probabilistic Torch has been designed with

semi-supervised learning strategies in mind from the start and has already proven remarkably effective; with only 10 labels per class, we have achieved classification accuracies of over 90% on datasets, such as MNIST. Probabilistic Torch also provides an expressive library of combinators that enables assembling sophisticated models from components. This allows us to, for example, embed a model for object detection into a second model that identifies regions of interest in an image, resulting in a model for multiple object detection that can be trained in an end-to-end, semi-supervised manner. We built on our current work on Probabilistic Torch with a number of innovative advances, including amortized sequential inference for problems that are sequential in nature, like activity recognition and machine translation, and adapting techniques from probabilistic program inference to estimate global variables.

Our research into high-capacity, fully probabilistic models was inspired by an approach developed by Wood called inference compilation [8], which learns a neural network to answer a query defined by a probabilistic program. While effective, the learned network is not a declarative probabilistic model and is only able to answer a single kind of query, rather than providing a data structure for reasoning. Our key insight was that we could compile the original probabilistic model into an alternative probabilistic model that supports all the capabilities of the original model but in which inference is linear time. This enables all the advantages of probabilistic models to be brought to bear on models with millions of parameters, like state-of-the-art neural networks. There are other methods that compile probabilistic models into data structures that support linear-time inference, such as arithmetic circuits [9] and sum-product networks [10], but these circuits are exponential-sized in general. In contrast, our data structures capture potentially complex information in compact networks, in a similar way to compact neural networks that represent rich functions for real-world applications. Enabling probabilistic models to have the flexibility and performance of neural networks will have far-reaching effects, not only to learning with fewer labels but beyond. Probabilistic models address some of the essential limitations of neural networks, such as the difficulty of incorporating prior knowledge, the need for large amounts of data, and the difficulty of modeling causation.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

In the Learning with Less Labels (LwLL) program, we addressed three technical problems:

- Model transfer: Given a model that has been trained for a general application, transfer the model to a specific application using a small amount of additional labeled data (Section 3.2).
- Semi-supervised learning: Learn to map a data instance to a label using a combination of a small amount of labeled training data and a large amount of unlabeled data (Section 3.3).
- Active learning: Given a dataset of unlabeled data and access to an expert who can label a limited number of instances, decide which instances the expert should label. This task included two modes: Selecting the next individual instance to label, and selecting a set of instances to label as a batch (Section 3.4).

The basis for our approach to addressing these problems is a new kind of probabilistic model that is integrated with deep learning representations. Our investigation into new probabilistic model components is described in Section 3.1.

3.1 Probabilistic Model Components

Under the PLEDGES effort, we developed a wide range of probabilistic models and model components to support learning with less labels. These included two main threads: Structured deep probabilistic models based on variational methods, and tractable high-capacity models based on separable methods. We describe the progress we made on each of these threads, beginning with an overview and proceeding with the new work done on each thread.

3.1.1. Structured Deep Probabilistic Models

3.1.1.1. Overview

One of the challenges in developing robust label-efficient approaches for machine learning is that current-generation techniques rely on a hodge-podge of mechanisms that often do not readily generalize across tasks and applications. Our first thread built on the framework of deep probabilistic models as a means of unifying methods for learning with fewer labels in a coherent, principled, and generalizable manner. To do so, we developed new variants of Structured Deep Probabilistic Models (SDPMs), a class of models that inherits the flexibility of deep generative models, whilst achieving higher label-efficiencies through the use of structured priors that reflect statistical regularities in the problem domain. In addition to developing task-specific SDPMs for object classification, object recognition, activity recognition, and machine translation, we integrated a number of innovations into the training of these networks. To train more ambitious SDPM variants, we developed novel strategies for *amortized sequential inference*, which build on advances at the interface of deep learning and probabilistic programming, where team members van de Meent and Wood have contributed pioneering research.

Background

Autoencoding deep probabilistic models (also known as a variational autoencoders) provide a general framework for unsupervised and semi-supervised learning, which holds the promise of combining the power of neural networks with the data-efficiency of probabilistic models. These models consist of two components. The first component is a deep generative model $p_{\theta}(\mathbf{x}, \mathbf{z})$, which is a probabilistic model in which conditional distributions are parameterized by neural

networks. This model defines a prior $p_\theta(\mathbf{z})$ over latent variables of interest (e.g., the objects in a scene), and a likelihood $p_\theta(\mathbf{x} | \mathbf{z})$ of data given latent variables (e.g., the pixels in an image, given objects in a scene). The second component is an inference model $q_\phi(\mathbf{z} | \mathbf{x})$, which maps from data to the latent variables. This component sets autoencoding models apart from other deep generative models, such as generative adversarial networks [11], which learn a model that simulates data but do not reason about latent variables in existing data.

While deep generative models have garnered tremendous interest from researchers, their development is still in its early stages in important respects. To date, the majority of models assume a generic unstructured prior, such as a Gaussian with diagonal covariance, and rely on the generative network architecture to learn a realistic likelihood model. This results in a model in which latent variables take the form of a vector of features, which is often referred to as an embedding. This embedding is a compressed representation of the data, in which individual dimensions correspond to some unknown, *entangled* combination of characteristics.

From the perspective of label-efficient learning, an entangled representation is undesirable. Suppose one were to train a model to recognize K categories of shapes that can have L categories of colors. If a model can accurately represent the shape and color, then it should generalize to combinations of shapes and colors that were not present in the training data. This means that we can hope to train this model on $O(K + L)$ examples, rather than a dataset in which all combinations of shape and color are present, which can require $O(KL)$ examples. In other words, if we can design models that learn *disentangled* representations, which factorize a high-dimensional feature space into lower-dimensional feature spaces, then this provides us with a means for learning representations in a more data-efficient and label-efficient manner.

More generally, models with structured representations provide a path towards achieving generalization performance more similar to that possessed by humans. A four-year old child must see a juggler only once to understand that this activity involves a person throwing and catching multiple objects. A child can readily generalize this concept to different types of objects, such as balls and clubs. If a model can represent the concept of a person, a ball, and the actions of throwing and catching, then this model is more likely to recognize new activities given a small number of labeled examples. More generally, structured models also offer opportunities to design more efficient models by exploiting regularities in the data; in a scene containing a bike, there are generally wheels, whereas a scene with football players likely contains a ball.

Approach

In this program, we developed SDPMs for object classification, object detection, activity recognition, and machine translation. In these models, we replace the generic Gaussian prior $p_\theta(\mathbf{z})$ with a probabilistic model whose structure reflects causal or statistical relationships between latent variables. Figure 1 shows an example of a previously developed SDPM for multiple object detection, which was implemented using the Probabilistic Torch programming framework [7].

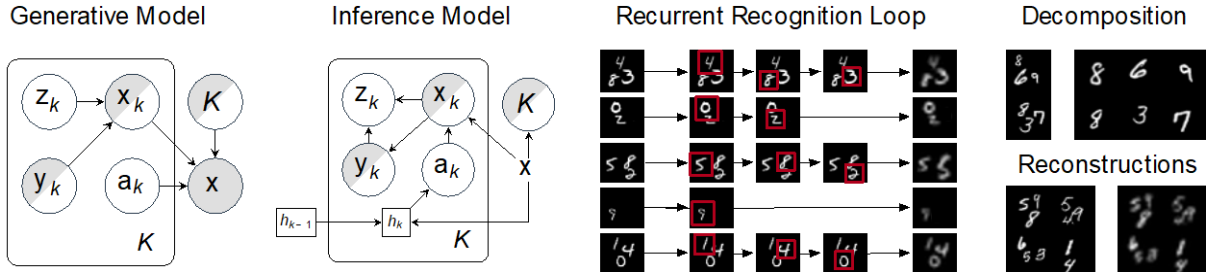


Figure 1: SDPM for Multiple MNIST Digits. This model performs object localization using a recurrent inference model to repeatedly identify regions of interest until a stochastic stopping condition is reached. It then calls a single-digit model as a subroutine to infer the digit label y and style variables z for each region.

In this model, we employ a form of curriculum learning. We first train a model for individual MNIST digits, which infers a discrete digit label and a set of continuous features that characterize the handwriting style. We then incorporate this individual digit model into a model for images that contain multiple digits. This second model repeatedly identifies regions of interest in the image, and then uses the first model to identify the digit in each region. The result is a highly structured model that can be trained in a semi-supervised manner by specifying values for some subset of variables in the model, like the number of digits in an image and the location of individual digits, and the digit label. This strategy is very label-efficient; the model for individual digits achieves a classification accuracy over 91% with only 10 labels per class, whereas the full model correctly identifies the digits in 85% of the images with partial supervision of variables, such as the number of digits and their locations.

We believe that SDPMs constitute a comparatively low-risk path towards developing a widely usable class of technologies that will enable practitioners to replicate and improve upon classification accuracies with far fewer labeled examples. It is important to note that SDPMs do not reinvent the wheel; the inference model in an SDPM can be parameterized using any deep neural network, including popular convolutional and recurrent architectures. What SDPMs provide in addition to these models is the ability to map the features that are returned by these models (which are generally unstructured and entangled) onto a new set of features (which are structured and disentangled), resulting in models which have better generalization properties.

Technical Innovations

Several new technologies are needed to realize orders of magnitude improvement in label efficiency across a broad range of tasks. We describe strategies for semi-supervised learning, model transfer, and active learning later in this report. In addition to these innovations, we developed new scalable methods for *amortized sequential inference*.

Many of the tasks that we considered in this program are sequential in nature; activity recognition involves modeling time series, object detection involves a spatial sequence of visual patterns, and machine translation involves sequential generation of language. In each of these tasks, we want to characterize some combination of global, time-invariant variables (e.g., the person in the video) and local variables (e.g., the activity that is carried out in each frame). Estimating the global variables is particularly challenging, since reasoning about these variables involves a judgement of all of the input data (e.g., all the frames of the video). To train these models in an efficient manner, we adapted innovations from probabilistic program inference to

SDPMs, and developed a number of novel strategies for sampling global variables that build on this work. The generative and inference models in an SDPM are trained using stochastic gradient descent. For the generative model, we maximize the marginal likelihood $p_{\theta}(\mathbf{x})$ with respect to the weights θ . For the inference model, we minimize the Kullback-Leibler (KL) divergence. Variational methods optimize the former objective, whereas wake-sleep variants (Bornschein & Bengio, 2014; Hinton et al., 1995; Le et al., 2018) optimize the latter. Computing the gradient of each of these objectives involves an intractable integral over the latent variables $\mathbf{z}$, which must be approximated using a Monte Carlo estimator that generates samples from $q_{\phi}(\mathbf{z} | \mathbf{x})$.

The core technical innovation in our methods for amortized sequential inference were to define gradient estimators that are sufficiently low variance when applied to models with highly structured priors. Recent work by Wood has shown that Sequential Monte Carlo (SMC) methods can be combined with optimization of either variational or a wake-sleep style objectives (Le et al., 2017; Le et al., 2018). Work by van de Meent will integrate SMC methods into an autoencoding Gibbs sampler, which iterates between updates to local and global variables to obtain a more accurate gradient estimate. Together, these innovations greatly increase the robustness of training for SDPMs, which enabled us to realize the model designs in this effort.

The following sections describe the research and development work we did to develop these models and capabilities.

3.1.1.2. Amortized Population Gibbs Samplers

We develop amortized population Gibbs (APG) samplers, a new class of autoencoding variational methods for deep probabilistic models. APG samplers construct high-dimensional proposals by iterating between updates to lower-dimensional blocks of variables. Each conditional update is a neural proposal, which we train by minimizing the inclusive Kullback-Leibler (KL) divergence relative to the conditional posterior. To appropriately account for the size of the input data, we develop a new parameterization in terms of neural sufficient statistics, resulting in quasi-conjugate variational approximations. Experiments demonstrate that learned proposals converge to the known analytical conditional posterior in conjugate models, and that APG samplers can learn inference networks for highly structured deep generative models when the conditional posteriors are intractable.

3.1.1.3. Conjugate Energy-Based Models

We improved the performance of generative probabilistic models at few-shot visual classification. While these generative models do not currently perform on the same level as discriminative models, we believe they have significant potential because they can be trained in absence of supervision by maximizing the probability of the training data. In our work on Conjugate Energy-Based Models (CEBMs), we define probabilistic models in terms of discriminative network architectures. The initial results from this method are encouraging—CEBMs produce an average improvement of about 6% on baseline generative models based on variational autoencoders on a range of datasets in the 10-label regime.

This work aims to close the gap between discriminative and generative models. A traditional advantage of (deep) generative models is that they can be trained in absence of supervision by maximizing the probability of the training data. When the underlying model incorporates the right prior, we can perform tasks such as tracking in a completely unsupervised manner (as we demonstrated in our work on amortized population Gibbs samplers). However, in practice, the vast majority of current models for label-efficient classification, including Simple CNAPS, are

discriminative; they employ a network that defines a mapping from data into a feature space. These models lack a notion of probability of data, which means they require labels (or pseudo-labels) for training.

This work bridges the gap between generative and discriminative models by using the framework of energy-based models (EBMs) to define probabilistic models in terms of discriminative network architectures. Concretely, we show how we can combine an energy-based likelihood with a generative prior in a manner that ensures that the posterior (i.e., the distribution over latent variables given data) is tractable. We believe that this work will allow us to reinterpret discriminative architectures like Simple CNAPS as probabilistic models, which will allow us to train them on a combination of labeled and unlabeled data.

This work is currently in its early stages, but initial results are encouraging; Conjugate Energy-based Models attain substantial improvements in label-efficiency relative to generative baselines based on Variational Autoencoders (VAEs) on datasets such as CIFAR-10 and SVHN house numbers.

Models	MNIST				Fashion-MNIST				CIFAR-10				SVHN			
	1	10	100	<i>full</i>	1	10	100	<i>full</i>	1	10	100	<i>full</i>	1	10	100	<i>full</i>
VAE	42	85	92	95	41	63	72	81	16	22	31	38	13	13	16	36
GMM-VAE	53	86	93	97	49	68	79	84	19	23	33	39	13	14	23	56
IGEBM	63	89	95	97	50	70	79	83	16	26	33	42	10	16	35	49
CEBM	67	89	95	97	52	70	77	83	19	30	42	52	12	25	48	70
CEBMM	67	91	97	98	52	71	80	85	16	28	42	51	10	17	39	60

Figure 2: Results for conjugate energy-based models

3.1.1.4. Evaluating Combinatorial Generalization in Variational Autoencoders

We evaluated the ability of variational autoencoders to generalize to unseen examples in domains with a large combinatorial space of feature values. Our experiments systematically evaluated the effect of network width, depth, regularization, and the typical distance between the training and test examples. We found that increasing network capacity benefits generalization in easy problems, where test-set examples are similar to training examples.

In more difficult problems, increasing capacity deteriorates generalization when optimizing the standard VAE objective, but once again improves generalization when we decrease the level of KL regularization. These qualitatively different forms of dependence on model capacity indicate that the properties of the training data and the level of KL regularization should be an integral part of any evaluation of generalization in this model class.

3.1.1.5. Gaussian Mixture Variational Autoencoders

We investigated methods to combine VAEs with Gaussian mixture models. Our experience from working with VAEs indicated that a unit Gaussian is not a suitable prior for latent space, especially if the goal is to have latent variables with high predictive powers. We believed that a possible key to drive VAEs to perform well on semi-supervised learning tasks was to employ

more complex priors that are designed to encourage separating the data based on the class in the latent space.

Some of the previous work on deep unsupervised clustering suggested that modeling the latent space as a mixture of Gaussians can result in a potentially useful latent structure (for prediction) [19–21]. To test whether these priors are helpful for classification, we used some of these models for the latent space and trained them with a semi-supervised objective. The first model we considered was simply a GMM-VAE [21] with 20 mixtures and a support-vector machine (SVM) classifier attached to the latent space z to predict the class from z . This model achieved 98% accuracy on MNIST with 10 label per class which is close to the state of the art. The latent space is shown in Figure 3. In this figure, the dimensionality of the GMM-VAE is 305. The model was trained on MNIST and projected down to two dimensions using t-distributed stochastic neighbor embedding (tSNE; left) and uniform manifold approximation and projection (UMAP; right).

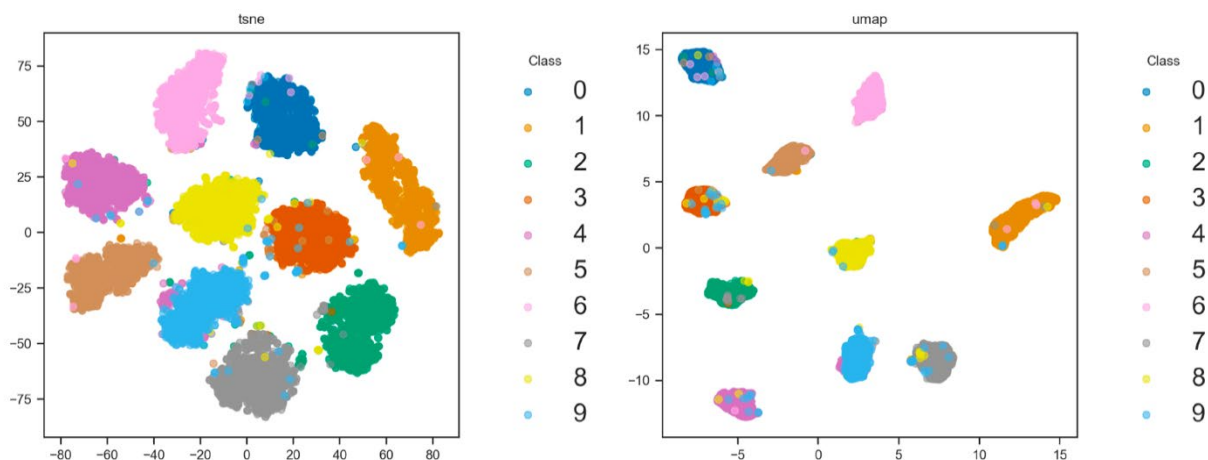


Figure 3: Latent Space of a GMM-VAE

3.1.1.6. Label Efficiency for Disentangled Representations

In recent years, there has been a concerted effort to develop models that learn “disentangled” representations, which explicitly separate factors of variation in a fully unsupervised manner. This work has, in certain cases, been motivated by the desire to learn models that will more readily generalize to unseen combinations of features. While there have been attempts to characterize the generalization performance of disentangled models on held-out combinations of features (Eastwood and Williams, 2018; Esmaeili et al., 2019; Locatello et al., 2019), precisely how disentangling affects generalization, and whether it improves generalization, is currently not well understood.

This project is explicitly testing to what extent disentangled representations improve label efficiency in downstream tasks. We did this in three stages:

- First, we considered standard approaches that learn “flat” representations that define a product space (e.g., shape, position, orientation, and color of objects).

- Next, we considered methods that induce hierarchical representations (e.g., multiple objects that each comprise several factors).
- Finally, we considered unsupervised methods for learning representations of video. In each case, we evaluated label efficiency for regression and classification tasks in terms of true labels.

3.1.1.7. *Bayesian Non-Parametric Models*

Learning classes of data where there are no labels is a clustering problem. When there are very few labels, we continue to use a clustering-like approach, while taking advantage of the few labels available to label classes. Bayesian non-parametric models represent an ideal framework for this because they can automatically adjust the number of classes to match the data. In PLEDGES LwLL effort, we adapted the SDPMs to include non-parametric labels.

From the original dataset, we learned a very good model of the intrinsic structure of the data. We identified the information that is essential to derive from the data to support labeling. This information can be identified with the state of hidden variables in the model. For example, if we used the thin layer models of Section 3.3.4.2, this information could be associated with the latent variables z , which are a distributed representation of the label.

Using a non-parametric approach, we created label variables, whose values include all the known labels for the new application along with an unspecified number of other values for the classes that have no labeled instances. For each class, we have a probabilistic model, with parameters to be learned, that generates the critical information about each instance. In training, we recognized when an instance did not fit any existing label and created a new class to correspond to the instance. Future instances can be assigned to this new class if they are similar to previous instances assigned to the class, which makes them fit the class well.

We can also integrate these probabilistic models with zero-shot learning methods that use side information. In these approaches, side information about a class can be used to classify instances of the class without seeing any training instances, by comparing the side information to that of other known classes. For example, animals might have verbal descriptions of their properties as side information (e.g., a zebra is black and white, has four legs, and lives on land). Even without having seen a zebra, it is possible to take an image that has characteristics of black and white objects, four-legged objects, and land-living objects, and classify it as a zebra.

In a probabilistic generative model, the class determines the properties of the object, which determines how the object is represented in data. Therefore, the side information, which represents the properties of the object, fulfills the same functional role in the network as the z variables in the thin layer. As we discussed, we can learn $P(h, k, x | z)$ from the original dataset. In Section 3.3.4.2, we showed how we can learn $P(z|y)$ from a small amount of labeled training data. In the presence of side information, we can do even better, because the side information provides explicit information about the relationship between y and z .

By combining the non-parametric and zero-shot learning methods, we can learn a better model of the class than we can from either source alone. To achieve this, we make the variables z include both semantically meaningful variables in the side information and other latent variables that have no semantic meaning. From the unlabeled data and semantically meaningful variables, we learn both the assignment of instances to classes and the deep structure of the new classes.

This combination of methods provides a flexible approach to applications with different amounts of side information. If no side information is available, we use the non-parametric model to learn the existence of new classes. With side information, we assign instances to named classes. If only some classes have side information, we use a combination of zero-shot learning for classes with side information and the non-parametric model for classes that do not.

3.1.2. Tractable High-Capacity Models

3.1.2.1. Overview

Our second thread also fits in the framework of variational methods and amortized inference. We begin with a probabilistic graphical model and construct an inference model. However, unlike the previous thread, the inference model is another declarative and generative probabilistic model, rather than a neural network, and it is constructed automatically. This has several advantages. First, we can construct the original model to mimic neural network architectures as closely as we like. Second, we do not have to worry about whether we can construct a good variational model for the architecture, as it will automatically be provided by the method. Third, and perhaps most importantly, the inference model is a generative probabilistic model in its own right, it has all the advantages of probabilistic models, including the ability to observe and query any variables and the ability to compose with other probabilistic models, using any of its variables as connection points. These properties make this method ideal for semi-supervised learning and model transfer.

Our approach is based on an inference algorithm called belief propagation (BP). BP is a variational message-passing method that can be used on any model but can be highly inaccurate, in some cases not even converging. Furthermore, the traditional BP algorithm is expensive in dense networks, because the size of messages can be exponential in the number of variables involved. In our approach, we compile our network into an alternative network for which BP is very fast and accurate by decomposing the messages to avoid the exponential blowup, and the decomposition makes the message passing more accurate. In fact, BP works by doing forward and backward passes on this alternative network, in a manner very similar to neural networks.

Figure 4 shows an example of this construction.

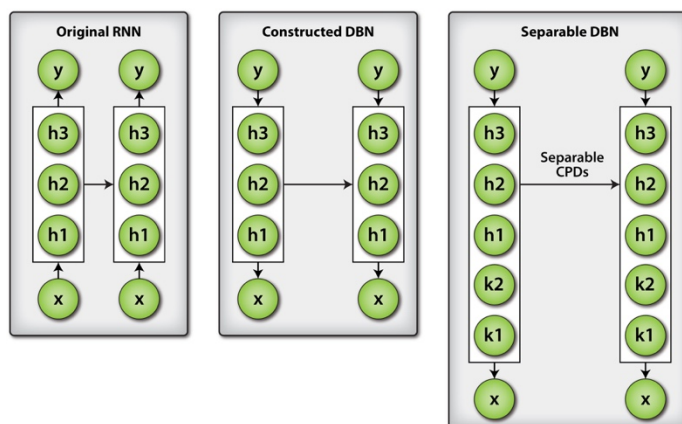


Figure 4: Constructing a Separable DBN

We begin with a neural network model, such as a recurrent neural network (RNN). The data is represented by x and the labels by y ; h represents the hidden nodes. We construct a probabilistic model with analogous structure to the RNN. In this case, the constructed model is a dynamic Bayesian network (DBN). The DBN is exactly like the RNN except that the arrows go from y through h to x rather than the other way around. This is the generative direction of the DBN; inference can still infer the label y from the data x . The next step is to convert the DBN into a separable DBN. This separable DBN consists of the nodes of the constructed DBN, augmented with auxiliary variables k ; in addition, we make each of the conditional probability distributions (CPDs) separable, as we now describe.

The origins of our approach are due to Pfeffer (2001), who introduced the concept of a separable CPD. A separable CPD is a representation of the dependence of a node in a probabilistic network on its parents. Separable CPDs can be used for nodes with any number of parents, but are easiest to explain for nodes with two parents.

Let a node X have parents Y and Z . In a network, the CPD of X is a representation of the conditional probability $P(X | Y, Z)$. In general, X depends on the joint state of Y and Z . However, if the CPD of X is separable, we can write

$$P(X | Y, Z) = \beta_1 P_1(X | Y) + \beta_2 P_2(X | Z)$$

Ordinarily, to compute $P(X)$ using belief propagation, you must know the joint probability $P(Y, Z)$, not just the separate marginal probabilities $P(Y)$ and $P(Z)$. This is why, in belief propagation, computing messages is ordinarily exponential in the number of parents of a node. However, Pfeffer showed that if the CPD of X is separable, we can compute $P(X)$ exactly knowing only $P(Y)$ and $P(Z)$. Furthermore, this only takes time linear in the number of parents. Pfeffer also extended this result to networks consisting of layers of variables in which every node has a separable CPD; we can propagate messages for the marginals over all the nodes exactly in linear time. In a subsequent paper (Pfeffer, 2006), Pfeffer showed that even in the presence of observations, the marginal probabilities are highly accurate for separable models.

Unfortunately, most real-world CPDs are not separable, so this result has had limited practical application. However, the result suggests that if we could create a model with separable CPDs, it can have fast and accurate inference. In fact, it is always possible to convert a non-separable CPD into a separable CPD by adding latent variables. Trivially, if we added a latent variable representing the joint distribution over the parents, the CPD of the child can be separable with this added parent. Our key insight was that we could use a similar construction to create a model that is approximately separable without representing the full joint distribution. In most real-world situations, knowing the exponentially many interactions between the parents to compute accurate marginals over the child is not necessary. Instead, we summarized what we must know about the joint distribution with a reasonably sized set of latent variables.

The critical question was how large this set of latent variables must be to ensure accurate inference. A priori, we did not have a definitive answer to this question, but we had three strong reasons to believe that the number of latent variables required would not be large.

The first comes from the overwhelming success of neural networks themselves, which is often attributed to the fact that they are universal function approximators. However, obtaining this result requires that we allow exponentially many nodes in the hidden layers. Practical neural

networks do not have exponentially many nodes, but they can still represent a wide variety of real-world functions. It has been shown (Brahma et al., 2016) that most observed data lives on a manifold that can be represented sparsely. We believe that the reason for this is that the real world has regularity and does not have the kinds of combinatorial interactions between parents that require exponentially many hidden nodes. The same regularity should enable us to represent the information required from the joint distribution over the parents compactly.

The second reason comes from Pfeffer's results. He defined the notion of approximate separability. A CPD is α -separable if it can be decomposed into a combination of a separable CPD with weight α and a non-separable CPD with weight $(1 - \alpha)$. The higher the value of α , the more separable the CPD, with $\alpha = 1$ corresponding to fully separable. Pfeffer (2006) showed empirically for dynamic reasoning problems that even for moderately sized values of α around 0.8, reasoning as if the CPD is separable produces highly accurate results. This shows that we only have to capture enough information in the hidden nodes to make the CPDs moderately separable.

The final reason arises from the way the separable model is trained. We generate a large number of simulated examples from the original network and then train the parameters of the separable model to explain these simulated examples using separable belief propagation. As a result, the separable model is trained to accurately predict the data generated by the original model using its own inference algorithm. If we view the separable model and separable belief propagation as a unit, then they are jointly trained to implement the model well. With a reasonable number of latent variables, they will have adequate capacity to implement the model.

In fact, we found that in some cases, the separable model provides more accurate inference than the original model even with no additional latent variables. For these models, it turned out that the benefit of more accurate inference outweighed the cost of using an approximate representation. However, we have only experimented with a small number of models, so the question of how many latent variables are needed requires further investigation.

Our correct reasoning that the number of latent variables needed is reasonably small for real-world problems led to large payoff, both for LwLL and for general applicability of probabilistic models. For LwLL, it led to exceptional performance on the challenge problems and on similar operational applications. Going beyond, probabilistic models have been touted as addressing some of the limitations of neural networks, such as the difficulty of incorporating prior knowledge, the inability to distinguish between causation from correlation, and the need for large amounts of data. To this point, although the situation is improving, the ability of probabilistic models to address these issues while being competitive with neural networks has been limited due to the difficulty of inference. The success of this research enables the ability to create and reason with probabilistic models without limitations, which will have profound impact on AI.

3.1.2.2. *Separable Model Learning Algorithms*

Separable models are high-capacity probabilistic models with fast and accurate inference. Separable Bayes nets are a unique form of Bayes nets that admit a fast, nearly exact inference algorithm based on belief propagation. This inference algorithm produces highly accurate marginal distributions over individual variables, though it does not explicitly represent the joint distribution of combinations of variables. Although this is a potential limitation in the inference, separable models with sufficient width can still describe arbitrary joint distributions over variables. We designed two learning algorithms for separable models, implemented them and

experimented with separable belief propagation, then evaluated their performance on alternative inference tasks. As our experiments below show, we were able to achieve good results on specific model classes.

We also implemented basic image classifiers. Our approach was to construct models where individual image pixels are each represented as separate leaf variables in a separable Bayes net. After fitting the parameters of the model to some data distribution, we constructed a fully Bayesian neural network equivalent with a “built-in” inversion procedure—a model trained to generate images from labels can be inverted to calculate the distribution over the labels (without an auxiliary “encoder” or “discriminator” model as employed in VAEs and generative adversarial networks (GANs)). However, we did not achieve good results with these models, most likely because the model architecture was too simple. We were hindered in our search for better architectures by the lack of GPU implementations for the separable algorithms. These algorithms are naturally tensorial and lend themselves to GPU implementation, but we were unable to explore this path under this effort.

3.1.2.3. *Expectation-Maximization Learning Algorithm*

We designed an Expectation-Maximization (EM) algorithm that computes the expected sufficient statistics of a variable with a separable model as a byproduct of belief propagation. We derived equations for statistics that enable them to be computed in time linear in the number of parents of a separable variable, as opposed to ordinary Bayesian network variables, which are exponential in the number of parents. This result made using this algorithm feasible on dense layered networks in which a node has all the nodes in the previous layer as parents, and therefore, it can provide the basis for tractable high-capacity models, one of our main goals.

We implemented this algorithm in Scruff. Scruff allows many kinds of models, each of which provides operations through a standard interface. Algorithms can then work on any model that provides the appropriate operations, in a manner agnostic to the specific kinds of models used. We developed a general-purpose EM algorithm in Scruff that can work on any model. This algorithm supports operations to initialize, calculate, and accumulate expected sufficient statistics, and to choose parameter values that maximize expected sufficient statistics. We have developed variants of these operations for ordinary conditional probability distributions and for separable models.

3.1.2.4. *Variational Method Learning Algorithm*

We also designed a variational method learning algorithm, which treats belief propagation (BP) inference as a variational approximation to the latent node posterior distribution and efficiently maximizes the evidence lower bound (ELBO) via Stochastic Gradient Descent (SGD). Essentially, we view separable BP as the inference model for a variational autoencoder.

The forward operation of a generative separable Bayes net is in many ways analogous to the “decoder” of a VAE, while the inference procedure producing latent labels $P(\text{lynx})$ is analogous to the “encoder” of a VAE. Through this lens, we constructed a variational bound on the probability of data and train the parameters of the model via SGD. A particularly interesting point is that we are propagating gradients through the operation of the belief propagation algorithm, enabled in large part by Julia, a new language with a number of functionalities aiming to enable automatic differentiation of arbitrary code. This algorithm has shown encouraging results demonstrating learning of a generative model on simple image tasks.

One issue we discovered with this approach is that the “decoding” operation of our model is only an approximation of $P(x|y)$. Because it uses the belief propagation inference procedure, we can only infer the marginal distributions over each pixel. This is a major limitation for more realistic tasks. We have a number of approaches in mind to alleviate this issue. Ideally, we calculate the gradient of the log parameter posterior, $\log p(\theta | \text{data})$. In general, derivatives with respect to the full vector argument for log conditional distributions are equivalent to derivatives taken entrywise across the vector conditioned on the other entry:

$$\frac{d \log p(\theta_0, \theta_1 | \text{data})}{d\theta_0} = \frac{d}{d\theta_0} [\log p(\theta_0 | \text{data}, \theta_1) + \log p(\theta_1 | \text{data}) - \log p(\theta_0 | \text{data})]$$

So, any algorithm that can produce accurate marginal distributions of single variables (as in the case of our inference algorithm based on belief propagation) can be automatically differentiated to produce a maximum a posteriori (MAP) learning algorithm as are typically employed in training deep neural networks.

Because we were able to derive efficient update methods for EM, we dropped work on this algorithm and pursued the EM approach because we expected it to require fewer iterations to converge.

3.1.2.5. *Implementation in Scruff*

We implemented learning for separable models in our Scruff AI framework, which is based on probabilistic programming, and conducted experiments. As part of this work, we made significant progress on the overall design of Scruff, and our separable model implementation was performed in conjunction with this design. We developed an application programming interface (API) for the Scruff runtime engine that manages inference across models and designed a message passing algorithm that uses this API to perform inference on separable models.

3.1.2.6. *Experimentation with Separable Belief Propagation*

We conducted extensive experiments with separable belief propagation on a predefined family of models. These models are a three-layer Bayesian network (BN) with a single node in the first layer, a variable number of nodes in the middle layer, and a single node in the third layer that depends jointly on all the nodes in the middle layer. The number of values in the range of each node is allowed to vary. The parameters of the models were set randomly.

The networks are designed to be a microcosm of the types of structures that make BN inference hard. Variable elimination (VE), while exact, is exponential in the number of nodes in the middle layer. BP is a popular approximate inference algorithm, but it is exponential in the number of parents of a node, so it is also exponential in the number of nodes in the middle layer.

One of the major advantages of separable BP is that inference is linear in the number of parents of a node. Therefore, we would expect separable inference to be quadratic for these models; one factor from the number of nodes and another factor from the number of parents of a node. Therefore, we expect to see exponential savings in inference time compared to VE and BP. On the other hand, separable models require a possibly expensive training process. This training can be amortized over many runs of inference. In particular, the model can be trained in the lab while inference is performed in real time in a fielded system.

The major unknown in the experiments was the degree of accuracy that the separable models would achieve. VE was used as an exact baseline to compare other algorithms to. We compared a range of variants of separable models to BP for accuracy. The separable models varied along two dimensions. The first is the number of values in the range of the nodes in the middle layer. We hypothesized that allowing more values for nodes in the middle layer could lead to more accurate results, the reason being that separable models have many fewer parameters than the original models and increasing the number of values in the middle layer could mitigate this. In particular, we hypothesized that having more values in the middle layer would enable those nodes to capture some of the coordinated influences that are ordinarily ignored by separable models.

The second dimension of variation of the separable models is the number of samples of the original model used to train the separable model. The separable model is trained by EM using data generated from the original model. Inaccuracy of this training is a major source of error in separable models, so we expect more samples to lead to more accurate results. We expect the training time per EM iteration to grow linearly with the number of samples, but the total number of iterations required might vary. We could also have experimented with the convergence criterion used by EM. However, initial investigations showed that a more stringent convergence criterion greatly increased the training time without yielding significantly more accurate results, so our extensive experiments were run with a looser convergence criterion in which all parameters must change by less than 0.01.

We considered two alternative inference tasks. The first, which we call the marginal query, is to predict the marginal probability of the leaf node with no evidence. The second, called the conditional query, is to infer the probability of the root node conditioned on the leaf node. We expected separable BP to be more accurate on the marginal query than the conditional query. The reason is that marginal inference using BP on separable models is an exact algorithm, while conditional inference is not, even though it has been shown empirically to be a good approximation.

Figure 5 shows the relative error of several variants of separable BP as well as ordinary BP on the marginal query, as a function of the number of nodes in the middle layer. All results are averaged over ten trials.

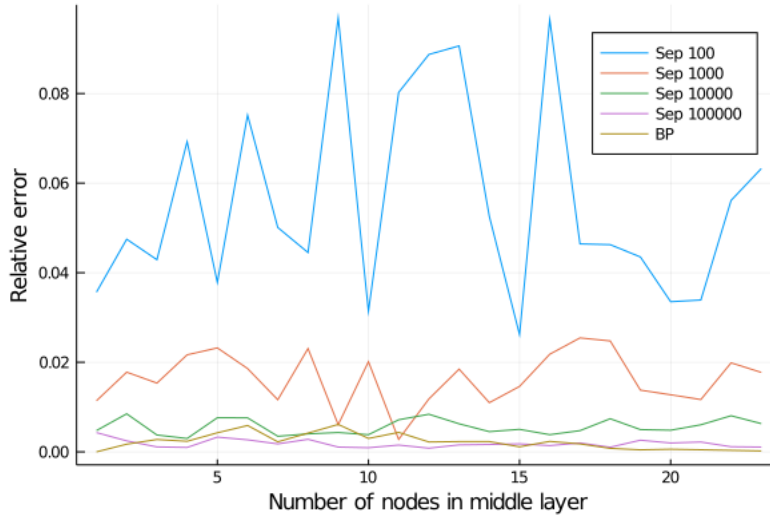


Figure 5: Error of Separable and Ordinary BP on Marginal Query

In these experiments, all the nodes had two values in their range. The separable algorithms also used just two values in the range of the nodes in the middle layer. The number in the legend is the number of samples used for training the separable model. We see that, as expected, the accuracy of separable BP increases significantly with the number of samples used for training. With 100,000 samples, accuracy is comparable with ordinary BP, and even with 10,000 samples the accuracy is not too much worse. These results indicate that as long as the separable model can be trained reasonably well, the ability to perform inference exactly makes up for the loss of information in the separable model compared to the original model.

Figure 6 shows a comparable plot for the conditional query.

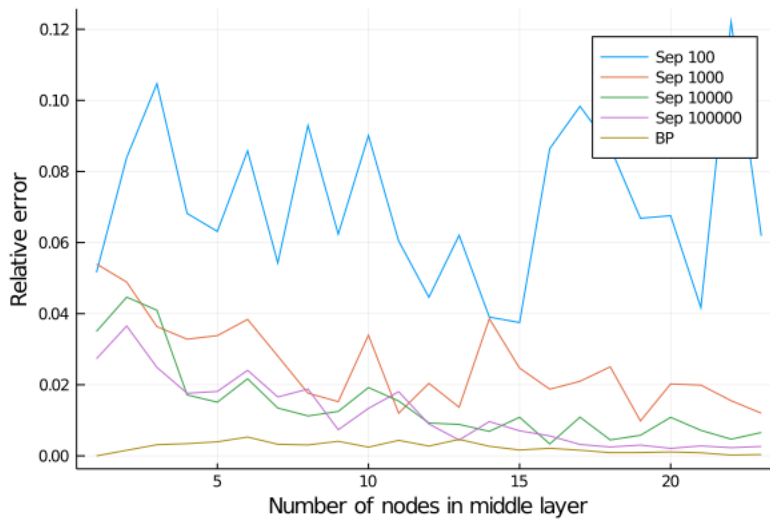


Figure 6: Error of Separable and Ordinary BP on Conditional Query

The results are similar, except that separable BP is less accurate for smaller networks than ordinary BP, even with 100,000 training samples. However, for larger networks, separable BP gets very close to the error of BP. Encouragingly, the error of separable BP decreases as the model grows more complex. Because separable BP is exponentially faster than ordinary BP (see below), its benefits will be most apparent on larger models, and it is encouraging that not only does its performance not degrade on larger models, but it also actually seems to improve, at least in this experiment.

Figure 7 shows the inference time for separable BP, ordinary BP, and VE on the same models. The log scale plot clearly shows the exponential growth of VE and BP, while the separable trend appears to be sub-exponential and is consistent with quadratic complexity.

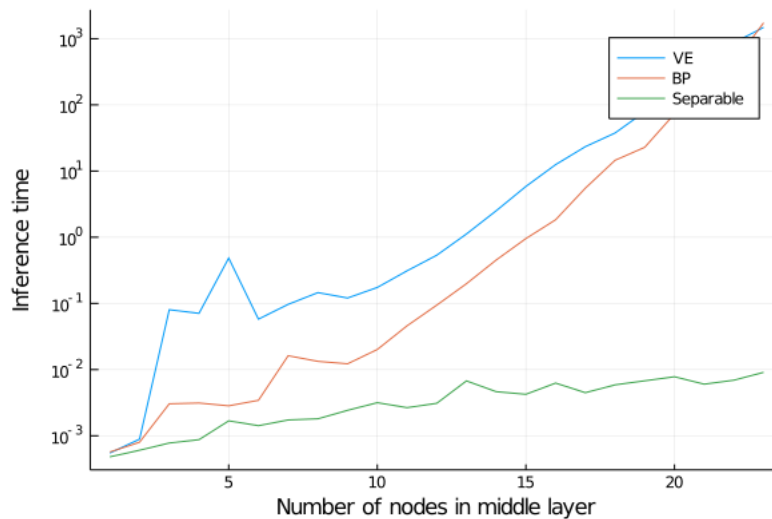


Figure 7: Inference Times for Various Algorithms (Log Scale)

Figure 8 shows the training time (not on a log scale) for separable models with different numbers of samples.

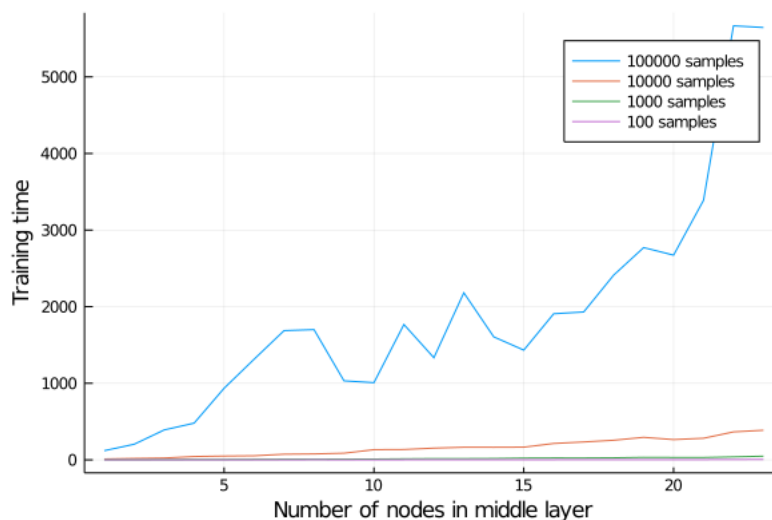


Figure 8: Training Time for Separable Models

The curves are consistent with a quadratic cost as a function of the size of the network. With 100,000 samples, there is a sharp spike at 21 middle nodes. This might be because of memory issues. These training times are amortized. However, with quadratic complexity for training, even without amortization, separable BP is exponentially faster than ordinary BP on these models.

We also experimented with varying the range size of nodes in the original network. Figure 9 shows the results for the conditional query and five nodes in the middle layer, where the range size varied from 3 to 10. All results are with 1,000 training samples.

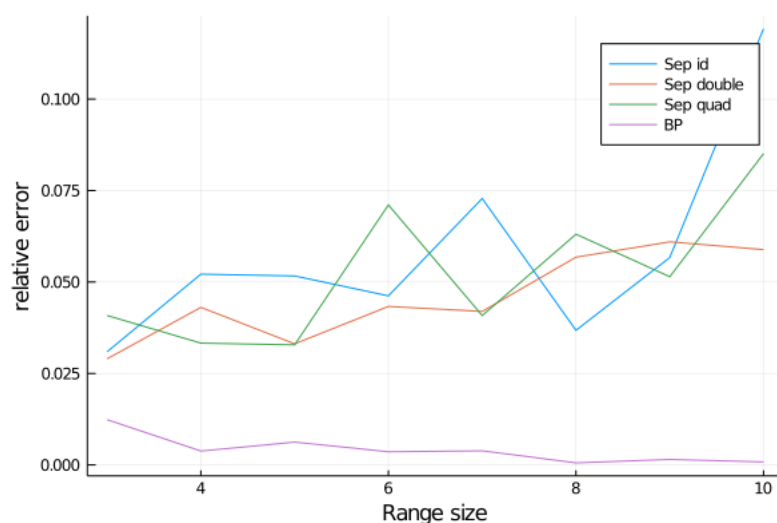


Figure 9: Error of Separable and Ordinary BP for Models with Variable Range Size

We have already seen that 1,000 training samples do not yield accurate results, so the fact that the separable models are less accurate than BP is not surprising. What is slightly concerning is that there is possibly a trend towards increased error (although this needs more investigation to confirm) as a function of the range size, whereas BP seems to improve as a function of range size. Our main interest in this experiment was to study the effect of the number of values in the range of middle node layers in the separable model. We experimented with three variants. The first, *id*, uses the same number of values as in the original model; *double* uses twice the number of values; and *quad* uses the square of the number of values. Surprisingly, we see no benefit to accuracy of allowing more values. However, more investigation is needed; it is possible that this is because 1,000 samples is not sufficient to accurately train the models with more values.

3.1.2.7. *Separable Model Scalability*

We improved the scalability of separable model learning and inference. As we were experimenting with separable models, we discovered that the original implementation was too slow for practical use. We developed some deconvolutional models to experiment with the MNIST dataset; unfortunately, even the simplest model took many hours to run, and the more useful models we were planning would have taken weeks. We therefore decided to improve the runtime of our algorithms.

We began by leveraging multithreading and parallelism. Our expectation-maximization (EM) algorithm can be naturally made parallel, where we compute statistics for each data instance separately in parallel. Locking is required to ensure correct aggregation of statistics atomically, but this does not significantly reduce the benefit of parallelism. Running on a 14-core machine, we obtained an order of magnitude reduction in runtime using parallelism.

A second order of magnitude reduction was obtained through code specialization. Operations used in our original belief propagation (BP) and EM algorithms were written in an abstract, general way that could apply to any number of parents, and to any target parent for the messages. However, when the number of parents and the index of the target parent are known, the operations can be streamlined significantly. Furthermore, the Julia compiler can specialize on the types of arguments, and types in Julia can include numeric values such as the number of possible values of a variable. The compiler can therefore use techniques, such as loop unrolling, to produce highly efficient straight-line code for specific models. Since the same model is used for many data instances and learning iterations, the benefits of this compilation are significant.

After experimenting on three-layer models with a large dense middle layer, we next ran experiments on larger, many-layer network models. We used square networks with n layers and n nodes in each layer. The layers are dense; each node depends on all the nodes in the previous layer. We ran separable models of size 50x50, with hundreds of thousands of parameters. However, variable elimination (VE) cannot handle more than 12x12, so our experiments show results for even values of n from 2 to 12.

Figure 10 shows accuracy results.

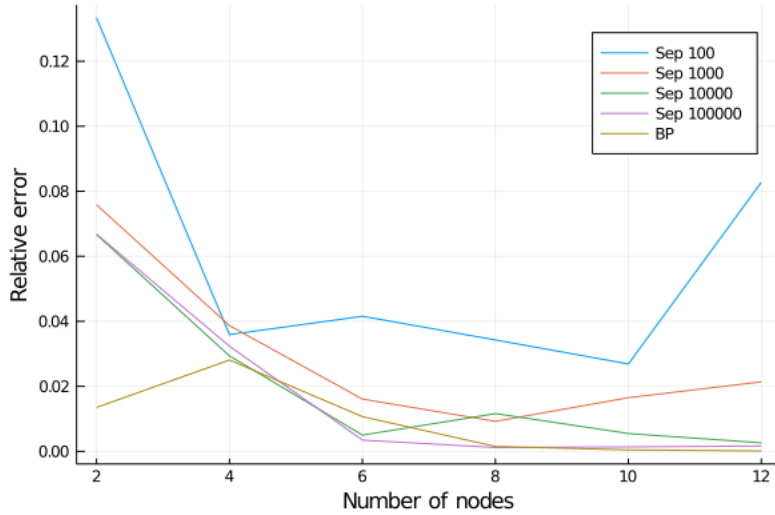


Figure 10: Relative Error of Algorithms on Square Networks

The plot shows the relative error of different algorithms as a function of the number of nodes in a layer (n), which is also the number of layers. Results for five algorithms are shown: four variants of the separable model with different numbers of training examples, and belief propagation (BP). Errors are shown compared to the exact solution of VE. As expected, the accuracy of the separable models tends to improve with the number of training examples. For larger values of n , the separable model with 100,000 training examples achieves comparable accuracy to BP. These results are similar to those we observed for the three-layer model.

Figure 11 shows computational performance results.

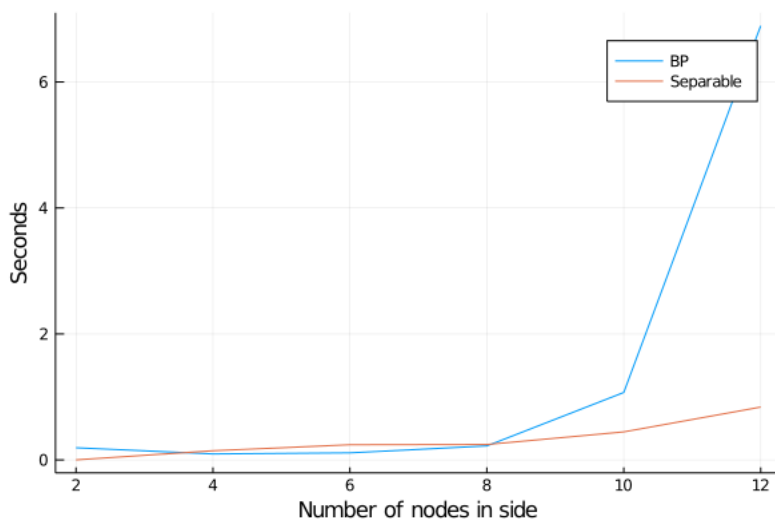


Figure 11: Runtime of Algorithms on Square Models

This plot shows the runtime of the separable model compared to BP. VE is not shown as it was far slower. The curve for BP shows the expected exponential growth in runtime. For separable, we would expect the runtime to be cubic in n , because there are n^2 nodes and each node has n parents. The curve for the separable model is consistent with subexponential growth; more data points would be needed to confirm cubic growth.

While our results show tremendous improvement in scalability of separable models compared to other probabilistic graphical models, they still do not quite reach the level of efficient neural network implementations, which typically operate on many millions of parameters. The next stage of our research will be GPU implementation of these algorithms; we have developed some initial designs in this direction.

3.1.2.8. *Neuro-Probabilistic Modeling Approach*

We developed a general formulation of our neuro-probabilistic modeling approach to facilitate integration of our separable models into our learning framework. This was not a completely new formulation, but a better understanding of the approach we are taking that facilitates integration of new probabilistic models in our framework.

Our approach is founded on the idea that probabilistic models are ideal for LwLL because they can learn from any combination of observations, including labeled, unlabeled, and with incomplete inputs. They can also encode prior knowledge and build composable, reusable models. However, experience has shown that neural models, which have highly tuned loss functions and efficient algorithms for large amounts of data, outperform probabilistic models at many learning tasks. Therefore, our approach has been to combine neural and probabilistic models.

Specifically, we formulated our approach to combine a neural feature extractor with a probabilistic decision maker. The neural feature extractor extracts features from the image. The probabilistic decision maker, represented by a directed probabilistic graphical model (PGM), infers the decisions from the features using Bayes rules.

To learn a neuro-probabilistic model, we must learn both the weights of the neural network and the parameters of the PGM. This is achieved by a combination of stochastic gradient descent (SGD) for the neural network and EM for the PGM. For SGD, we need a loss function on the output of the neural network; that is, the features. We use negative log likelihood of the predicted features under the true decision as the loss function. The procedure for updating the weights of the neural network on a training example is as follows:

Given a training example (which may have complete, partial, or no label):

1. Use the neural network to extract features from the image.
2. Observe whatever is available from the label (if anything) as evidence in the PGM.
3. Compute the gradient of the negative log probability of the features using PGM inference.
4. Apply this gradient to the output layer (extracted features) of the neural network and use backpropagation to update the weights. In practice, this step is performed on a whole mini-batch of examples.

The procedure for updating the parameters of the PGM using EM is as follows:

Given the extracted features from the neural network and the label (complete, partial, or missing)

1. Observe the features and label as evidence in the PGM.
2. Use the PGM inference to infer distributions over all the unobserved variables in the PGM.
3. Collect expected sufficient statistics from these distributions.
4. Use the statistics to re-estimate parameters of the PGM to maximize the likelihood of the expected sufficient statistics. In practice, this step is also performed on a whole mini-batch of examples.

The key to this combined learning working is that both the SGD on the neural network and the EM on the PGM are optimizing the same quantity. They are both trying to maximize the log likelihood of the features. SGD is known to lead to a local minimum of the objective function (negative log likelihood). EM is also known to lead to a local maximum of the log likelihood. Therefore, there is strong reason to believe that this combined neuro-probabilistic learning algorithm will also lead to a local maximum of the log likelihood. Although we were unable to test this algorithm under LwLL, we plan to continue to investigate this approach under future programs.

3.2 Model Transfer

Although model transfer is Task 3 under our work plan and semi-supervised learning is Task 2, our methods for model transfer became the basis for our methods for semi-supervised learning, so we present them first. Under the PLEDGES effort, we developed methods to transfer a model learned from a general-purpose dataset to a specific application using very few labels for the new application.

3.2.1. Overview

As labeled data for some military applications is extremely scarce, transferring knowledge learned from general applications to these highly specific applications is essential. This task requires learning the new application from very few labels; there may be fewer than one label per class to be learned. Some labels will have no examples at all, but our approach should use those labels that are available.

Our primary method for model transfer is Simple CNAPS, which uses meta-learning to learn how to classify images accurately in a new dataset from few labeled examples. Simple CNAPS is the basis for many of the methods we investigated during this effort and we describe it first. During the course of the effort, we also investigated several other approaches to model transfer, including approaches to object detection.

3.2.2. Simple CNAPS Algorithm for Few-shot Visual Classification

We designed and experimented with Simple CNAPS, a few-shot learning method that builds on and simplifies Conditional Neural Adaptive Processes (CNAPS) [22]. CNAPS is an algorithm for few-shot visual classification that uses a ResNet backbone, consisting of a feature extractor network and a classifier network, along with an adaptation network that adapts the weights of the backbone for a particular dataset. The adaptation network is learned using meta-learning, where it is trained with the objective of achieving good validation set performance with few labeled training examples.

Simple CNAPS takes a Bayesian view of the model, and replaces the classification network of CNAPS with a probabilistic classifier. Another difference is that the adaptation network updates the weights of only the feature extraction network and not the classification network (which has been disabled). Simple CNAPS uses meta-learning in a similar way to CNAPS. Our work on Simple CNAPS resulted in a paper in CVPR [23]. As reported in the paper, Simple CNAPS significantly outperformed previous methods, achieved a 6% average improvement at out-of-domain classification, on standard benchmarks used in the few-shot learning field. Out-of-domain accuracy represents performance on entirely new datasets not seen during meta-training.

Simple CNAPS's probabilistic classifier model is a mixture of Gaussians model that can quickly be trained using a small amount of labeled data. This model includes the means of all the classes, in feature space, together with a covariance matrix for each class. To classify an instance, it uses the Mahalanobis distance between the features and the class mean. This enables much better, domain-adapted classification. It turns out that just a few labeled examples in the new domain are enough to learn the covariance matrix and achieve good model transfer.

To evaluate Simple CNAPS, we experimented with META-DATASET, a standard benchmark for few-shot learning and model transfer. In our experiments, we trained our model on 8 object classification datasets and tested it on 5 new datasets, with few labeled examples for each of the test datasets. Overall, Simple CNAPS obtained 69.7% classification accuracy on the new datasets, which was 8.2% higher than the previous state-of-the-art (CNAPS).

We evaluated the performance of Simple CNAPS as a function of the number of labeled examples for a class and found that even with as few as 4 examples of a new class, we could classify that example with over 60% accuracy. We also found that Simple CNAPS improves on CNAPS consistently, regardless of how many classes there are in the task.

These results are reported in a paper that appeared at CVPR 20 [23].

3.2.3. Extending Simple CNAPS to Few-Shot Object Detection

To design our few-shot detection algorithm, we considered two alternative designs based on Simple CNAPS. In the first design, we place a region proposal network (RPN) in front of Simple CNAPS, which learns domain-specific classifications. The RPN in this design is domain-general and is not adapted specifically to the target domain. The idea behind this design is that a RPN that has been trained on a variety of domains to be robust to domain variations can propose good regions without fine-tuning it to the new domain.

In the second design, the RPN is adapted to the new domain. We place the RPN between the meta-learned feature representation of Simple CNAPS and the output. Ultimately, we felt that it would be difficult to accurately train this RPN on just a few examples, and that using a general RPN with a domain-adapted classifier would likely yield better results. In addition, the design approach is simpler from an engineering point of view, so we selected the first design to implement within our 12-month evaluation system.

3.2.3.1. Few-Shot Algorithm Implementation

To implement the few-shot object detection algorithm, our general approach was to use an RPN to discover the bounding boxes of objects, and attach the RPN to Simple CNAPS for classification. Simple CNAPS is meta-learned and adapted to the object detection dataset, allowing it to classify patches produced by the RPN of different sizes, treating the patches as the input images for classification. Because of previous experience, where we found that few-shot

adaptation of the RPN is not helpful, we meta-learn only the Simple CNAPS classifier and not the RPN. For the RPN, we use FasterRCNN. Specifically, we looked at Facebook’s Detectron implementation of Faster RCNN

In the 12-month evaluation, we pretrained the RPN on any object detection datasets in the whitelist. We implemented this method and tested it using COCO as the pretraining set and PASCAL-VOC as the target dataset. We compared performance to two recent papers, “Meta-RCNN: Meta Learning for Few-Shot Object Detection,” submitted by anonymous authors to ICLR 2020, and “Frustratingly Simple Few-Shot Object Detection,” submitted to arXiv in March 2020. These papers use the same datasets.

We focused on two aspects of model transfer: Transfer of knowledge from pretraining to operation, and transfer of knowledge between the base dataset and the adaptation dataset during the evaluation.

We first developed methods to transfer knowledge for classification from pretraining to operation using Simple CNAPS. For object detection, we also need to pretrain the RPN. This model is pretrained on the COCO dataset. We wanted to know how well this pretrained model performs on unseen classes on which it was not trained.

To investigate this question, we experimented with pretraining the RPN on COCO and testing on Pascal-VOC. We removed all the classes from the training set that are present in PASCAL-VOC. We compared two different versions of pretraining. One version used the classification and regression layers of Faster RCNN, using the class labels of COCO during training, to train the Faster RCNN model. The second used only the bounding boxes in the data to train the RPN alone, without the classification and regression layers. We wanted to see whether training on classes, even though they were different classes, would improve performance on new classes. Unfortunately, the results were inconclusive. Both methods performed poorly on new classes.

The other form of model transfer we investigated during the 12-month evaluation was from the base dataset to the adaptation dataset. It is our understanding that there is some relationship between the base dataset and adaptation dataset, but we do not know what that relationship is. However, we believe that finetuning the model on the base dataset will be beneficial for the adaptation dataset.

Since the relationship between the base dataset and adaptation dataset is unclear, we do not know how much finetuning to do. We do not want to forget everything that was previously learned during pretraining, but we want to obtain as much benefit from finetuning the model to the base dataset as possible. Our approach to this problem was to use the adaptation set itself as a validation set for finetuning on the base dataset. In particular, we believe that the second checkpoint, with ten images per class, provides enough data for a validation set. If the base dataset is highly similar to the adaptation dataset, our method should perform a lot of finetuning on the base dataset; while if they are dissimilar, there should be less finetuning.

3.2.3.2. Object Detection Algorithm Performance Improvements

After implementing and testing the basic few-shot algorithm, we discussed how to improve its performance. One way is to make it transductive, similar to our transductive classification algorithm. In this approach, we create an entire dataset of patches using our RPN (which is not adapted and can therefore be used for this purpose). We then use this dataset transductively to learn a model for the patches, in a similar way in which we use the target dataset transductively

when we do classification. Another way to improve performance is to use ensemble methods. For example, we can use different crops of the input image and take a vote across the ensemble. This method can also be used during training, where it is a form of data augmentation.

To improve performance, we implemented a new approach to few-shot object detection, which adapts the region proposal network (RPN) of Faster-RCNN using a Simple CNAPS adaptation network and Gaussian mixture model classifier.

We first reviewed how we were using Faster RCNN to generate region proposals and then classifying the patches using Simple CNAPS. Faster RCNN, in addition to the region proposal network (RPN), has a classification head and a regression head. The regression head is potentially useful to us as it refines the region proposals based on the specific objects. However, the regression and classification heads cannot be pretrained, because they are specific to a set of classes and we do not know the classes in the test datasets in advance. Therefore, our initial plan was just to use the RPN of Faster RCNN, pretrained on COCO, and not to use the regression head.

In our new approach, we train the regression head on the small number of labeled images in the test dataset. We do not update the RPN itself, since prior studies have shown that the RPN cannot be easily learned using small amounts of data. However, the regression head can be updated to improve the regions. In fact, the goal of the regression head is to tailor a pretrained RPN to a specific dataset.

We then adapt the RPN itself, as well as the classifier. The first step is to replace the RPN classification head with a Gaussian mixture model (GMM) classifier, as used in Simple CNAPS. In our experiments, the RPN with GMM head, with no adaptation, performs almost as well as the standard RPN. However, the advantage of using the GMM head is that it enables adaptation using a method similar to Simple CNAPS.

Finally, we meta-learn an adaptation network that sets Feature-wise Linear Modulation (FiLM) parameters of the RPN backbone. Given a new dataset, we first set the FiLM parameters using the adaptation network. We then learn the GMM classifier in the same way as Simple CNAPS for classification.

3.2.4. Few-Shot/Zero-Shot Object Detection

We investigated a new model transfer method for object detection. Our method uses zero-shot learning to detect objects in a new domain with no labeled examples, and we improve the zero-shot object detector with a few examples to create a few-shot or zero-shot object detector. To achieve this, the object detector uses two cues:

- The classification of the image, as produced by a classifier – For example, if the classifier predicts that there is a car and a pedestrian in the image, that gives the object detector an idea of what to look for. In the few-shot setting this classifier will be a few-shot classifier such as Simple CNAPS.
- The linguistic label of the image, as expressed in a word embedding such as word2vec – The idea is that even in a new domain, the name of the label might be a clue about what to look for when detecting the object.

Our method combines weak image-level supervision with few- or zero-shot object detection. It relies on the ability to learn classifiers, an easier task, from image data. For example, we can use Simple CNAPS to learn a classifier from few labels. The method also assumes abundant

detection data for base classes to learn their detectors, but few or no detection data for novel classes. Instead, detection for the novel classes is performed by adapting the most relevant base class detectors.

Specifically, we use similarities in the visual and lingual cues (described above) between base and novel classes to express the detectors of novel classes as linear combinations of base class detectors. This provides a zero-shot detection method. Then, given a small number of examples of detection with a novel class, we refine these novel class detectors for few-shot learning.

We implemented and tested a first version of our method. Table 1 shows preliminary results on the Pascal VOC dataset with 15 base classes and 5 novel classes. We compared to state-of-the-art few-shot object detection methods. The results are not directly comparable, since our method uses weak supervision whereas the competing methods do not. However, the results clearly show the potential gains from weak supervision.

Table 1: Preliminary Results for Few-Shot Object Detection with Weak Supervision

		Number of Shots (k)				
Method		0	1	2	3	5
Few-shot	YOLOv2 + FeatReweight	-	14.8	15.5	26.7	33.9
	Faster R-CNN + MetaDet	-	18.9	20.6	30.2	36.8
Weak	Fixed Classifier	21.7	51.4	60.4	63.3	66.9
+	Fixed Classifier + Top-5 deltas (LSDA)	26.2	-	-	-	-
Any-shot	Proposed approach	52.9	60.7	64.9	66.8	67.9

3.3 Semi-Supervised Learning

Under the PLEDGES effort, we developed methods to learn a model from a large amount of unlabeled training data and a small amount of labeled training data.

3.3.1. Overview

To learn models with fewer labels, we begin with semi-supervised learning—learning an accurate model from a large amount of unlabeled data and limited amount of labeled data. We investigated a variety of methods, including Simple CNAPS, which significantly improved on the state-of-the-art at few-shot learning.

3.3.2. Baseline Methods

We reviewed relevant literature, implemented baseline methods, and benchmarked these methods on a number of datasets.

3.3.2.1. Literature Review

To establish a baseline for future methods, we reviewed current state-of-the-art methods for semi-supervised learning in computer vision [24–31]. In general, semi-supervised methods optimize an objective of the form:

$$L_{total} = \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})} [L_{unsup}(\mathbf{x})] + \alpha \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim p_{data}(\mathbf{x}, \mathbf{y})} [L_{sup}(\mathbf{x}, \mathbf{y})]$$

Here L_{unsup} and L_{sup} correspond to the unsupervised and supervised losses respectively. Different approaches can essentially be distinguished by how they define these two terms. There are two families of models that are currently performing at the state-of-art: GAN-based objectives, discriminative objectives.

Standard GANs are normally trained with a discriminator that is in charge of classifying whether an input data is “real” (comes from the training set), or “fake” (comes from the generative model). In semi-supervised learning, GANs rely on the same trick. Consider a standard classifier for classifying a data point $\mathbf{x}$ into one of the K possible classes. GANs extends this set of labels $\mathbf{y}$ by adding an extra label $y = K + 1$ that corresponds to the “fake” label. The objective takes the following form:

$$L_{sup}(\mathbf{x}, \mathbf{y}) = \log p_{\theta}(\mathbf{y} | \mathbf{x}, y < K)$$

$$L_{unsup}(\mathbf{x}) = -\log[1 - p_{\theta}(y = K + 1 | \mathbf{x}) - \mathbb{E}_{\mathbf{x} \sim p_{\theta}(\mathbf{x})} [\log p_{\theta}(y = K + 1 | \mathbf{x})]]$$

Discriminative objectives have also been used for semi-supervised learning in a variety of forms [24, 30, 31]. One of the most common approaches that have shown significant improvements in classification accuracy is to encourage the predictions for the unlabeled data points to have a strong confidence, which can be done by penalizing the entropy. The objective can take the following form:

$$L_{sup}(\mathbf{x}, \mathbf{y}) = \log p_{\theta}(\mathbf{y} | \mathbf{x})$$

$$L_{unsup}(\mathbf{x}) = -H(p_{\theta}(\mathbf{y} | \mathbf{x}))$$

Where $H(\cdot)$ is the entropy. There have also been other approaches to improve accuracy in semi-supervised learning in both approaches such as consistency regularization [26] where there is a constraint on similar data to have similar output (labels), and traditional regularization [26] we believe some of these techniques can be useful in other deep generative models, such as variational autoencoders (VAEs).

In the last decade, three datasets have been commonly used as benchmarks for semi-supervised learning: MNIST, SVHN, and CIFAR10. MNIST is no longer used due to almost all state-of-art semi-supervised approaches achieving >99% accuracy (with only 10 labels per class). A list of state-of-the-art SSL objectives and their performance on various datasets is shown in Table 2.

Table 2: Performance of SSL Objectives on Various Datasets

Method	MNIST	CIFAR-10	SVHN	F-MNIST
Salimans et al. [11]	99.07 (100)	-	-	-
Dai et al. [5]	99.2 (100)	85.59 (4000)	95.75 (1000)	-
Balestrierio et al. [3]	99.14 (50)	77.75 (4000)	89.75 (500)	-
Xu et al. [14]	98.38 (100)	81.79 (4000)	-	86.74 (100)
Athiwaratkun et al. [2]	-	95.0 (4000)	-	-
Tarvainen and Valpola [12]	-	87.69 (4000)	96.05 (1000)	-
Berthelot et al. [4]	-	95.05 (4000)	96.22 (250)	-
Miyato et al. [7]	99.11 (100)	-	94.86 (1000)	-

3.3.2.2. VAE-Based Methods

The Northeastern team developed approaches to semi-supervised learning that incorporate generative models as inductive biases. We initially focused on VAE-based objectives. Within the VAE framework, we considered a variety of approaches, including:

- Employing a prior that is more suitable for classification, such as a Mixture of Gaussians [32] or a VampPrior [21]. A similar approach is to reduce the penalty on the KL regularizer, which implicitly enables to learn a less restricted latent representation. We believe one of the limitations of a VAE’s ability to achieve good classification accuracy is a bad choice of prior. VAEs try to organize their data based on standard Gaussian $N(0,1)$, which is not suitable for classification.
- Modifying the objective itself. These approaches include introducing a term for encouraging to learn a more disentangled representation to separate the class and the style [7, 33], or step towards a variation-informational-bottleneck (VIB) approach [34].

Concretely, let us assume we have three variables: the data $\mathbf{x}$, the target $\mathbf{y}$ and the latent variable $\mathbf{z}$. Let $p_\theta(\mathbf{x}, \mathbf{y}, \mathbf{z})$ and $q_\phi(\mathbf{x}, \mathbf{y}, \mathbf{z})$ be the generative and the inference model respectively. The VIB objective maximizes the mutual information $I_\phi[\mathbf{y}; \mathbf{z}]$ between the target variable and the latent variable, whilst minimizing the mutual information $I_\phi[\mathbf{x}; \mathbf{z}]$ between the data and the latent variable:

$$\max_\phi (I_\phi[\mathbf{y}; \mathbf{z}] - \beta I_\phi[\mathbf{x}; \mathbf{z}])$$

Both these mutual information terms are intractable, and are, in practice, approximated by a lower and upper bound respectively by introducing a likelihood $p_\theta(\mathbf{x}, \mathbf{y}|\mathbf{z})$ and a prior $p_\theta(\mathbf{z})$. This results in the variational objective:

$$L_{sup}(\mathbf{x}, \mathbf{y}) = \mathbb{E}_{q(\mathbf{x}, \mathbf{y}) \sim q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x}, \mathbf{y}|\mathbf{z})] - \beta \mathbb{E}_{q(\mathbf{x})} [\text{KL}(q_\phi(\mathbf{z}|\mathbf{x}) || p_\theta(\mathbf{z}))]$$

We evaluated two classes of generative models:

1. $p_\theta(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p(\mathbf{z})p_\theta(\mathbf{y}|\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z})$
2. $p_\theta(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p(\mathbf{z})p_\theta(\mathbf{y}|\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{y})$

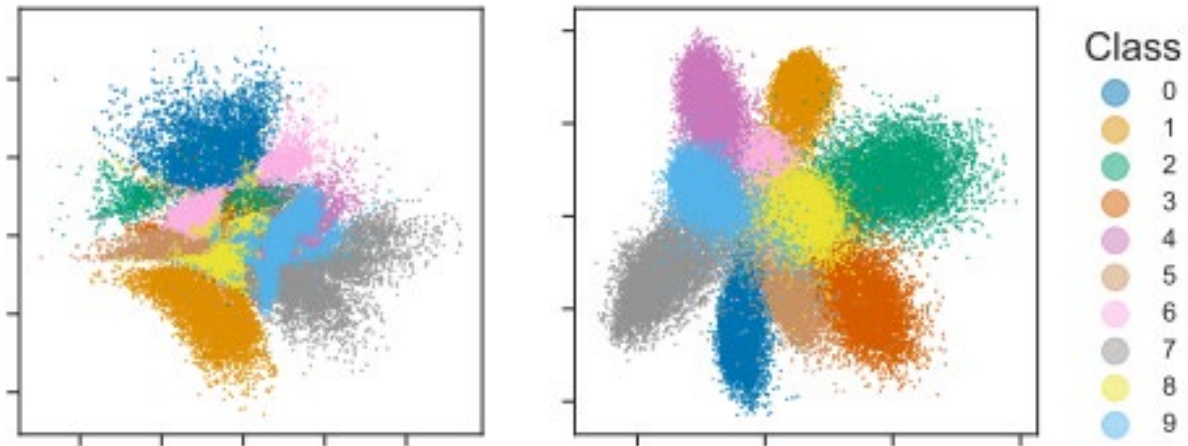


Figure 12: Visualization of the Latent Space for a VAE Trained on MNIST. (Left) Standard VAE; (right) VAE + VIB Objective.

We observed that Equation 1 showed promising results with certain adjustments. The adjustment we employed was to use only two dimensions in z to predict y and use all the dimensions to reconstruct the image. This approach not only allows us to visualize the latent space to see the clusters (Figure 12), but also yields around 91% using simple MLP architectures, and without any of the techniques currently being used in the literature, such as entropy minimization or consistency regularization. We evaluated the effect of regularization methods and objective terms that induce disentangled representations to encourage the separation of classes and the style within each class.

To gain a better understanding of the quality of latent representations (with respect to classification) in VAEs, we first trained several models in a completely unsupervised fashion, and then trained low-capacity classifiers (kNN, Naive-Bays, Logistic Regression, SVC) on the learned z . For testing purposes, we initially only considered MLPs on the MNIST dataset while studying several other influential factors such as β (the amount of KL regularization), dimensionality of z , and number of layers.

Figure 13 and Figure 14 shows the results of this analysis. We found that lowering β generally improves classification accuracy (which is consistent with existing results [34]). We also observed that even a 5-dimensional latent space can be sufficient to distinguish the classes.

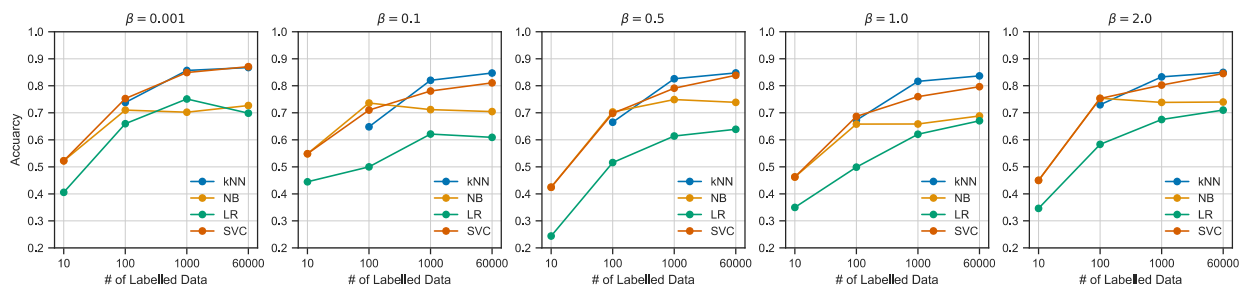


Figure 13: Impact of β on the Quality of the Latent Representation

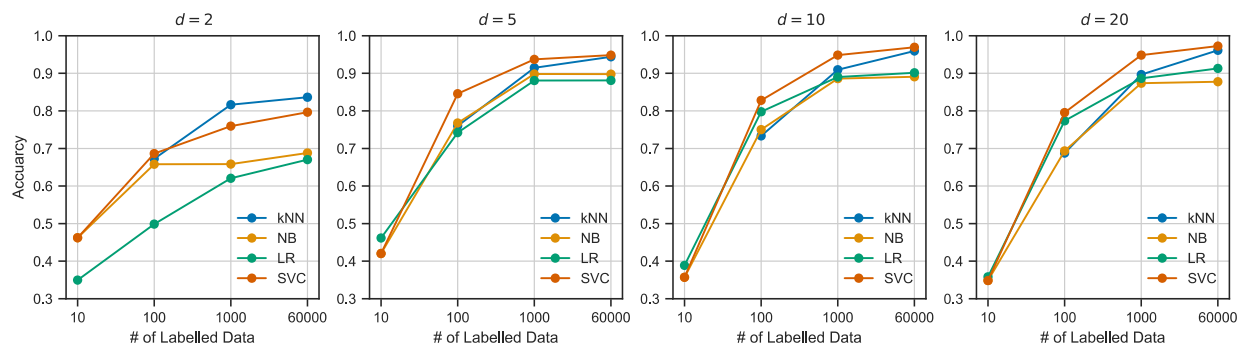


Figure 14: Impact of Dimensionality of z on the Quality of the Latent Representation

3.3.3. Semi-Supervised Extensions of Simple CNAPS

3.3.3.1. Initial Semi-Supervised Extensions

We developed a semi-supervised extension of our Simple CNAPS few-shot learning algorithm. We began by looking into methods for extending our meta-learning-based Simple CNAPS architecture from a low-shot supervised model to a low-shot *semi*-supervised model.

During our investigation into semi-supervised extensions of Simple CNAPS, we initially developed several potential concepts, including:

Semi-supervision of the underlying feature extractor by using a variational auto-encoder architecture. Figure 15 outlines the architecture of our meta-learned feature extractor. The ResNet and meta-learner are trained on only supervised data (such as ImageNet). However, if we have access to unlabeled data, we could extend this architecture to an encoder-decoder architecture, such as the one shown in Figure 16. This architecture can exploit unlabeled data by requiring that the learned features can properly reconstruct the unlabeled data.

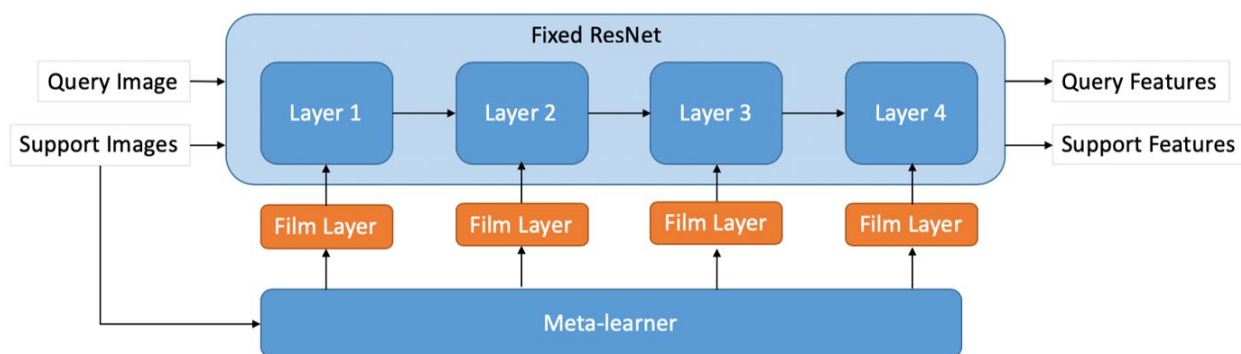


Figure 15: Supervised Meta-Learning Architecture

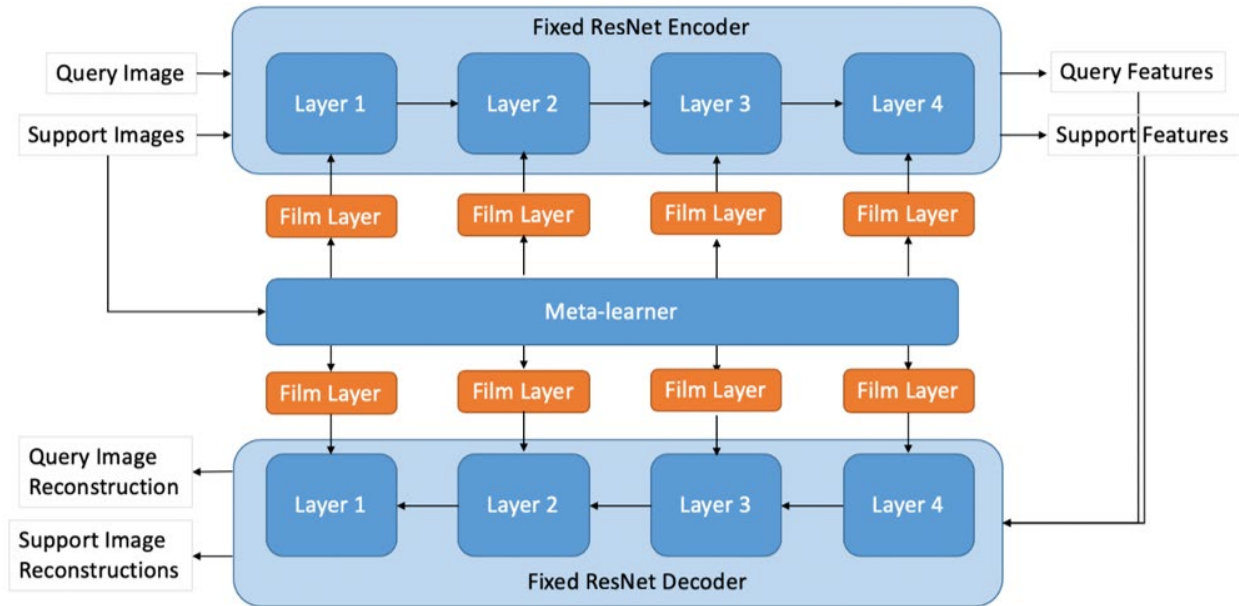


Figure 16: Meta-Learning a Semi-Supervised VAE

Exploit the unlabeled data for construction of the Simple CNAPS covariance matrix.

Simple CNAPS gets its name by replacing a classification network (built on top of the feature extractors in the preceding figures) with a simple distance-based classifier; the key insight is the choice of distance, which is a Mahalanobis distance based on per-class estimates of the feature covariance matrix. However, in the low-shot setting, the number of support images is much lower than the number of features. This causes the sample covariance matrix to be rank-deficient, and the distance is undefined (since it involves inverting the covariance matrix).

Simple CNAPS handles this by using a *shrinkage estimator*, which combines the sample covariance estimate with some “prior” full-rank covariance matrix (such as an identity matrix). This framework makes it easy to exploit unlabeled data by using a sample covariance matrix formed from all of the unlabeled training data as a shrinkage target; this has the benefit of both being (in scenarios with reasonable amounts of unlabeled data) full-rank *and* much closer to the “true” covariance. We expected this to improve performance both through just being a better estimator and by reducing the number of hand-tuned heuristics needed to properly condition the covariance matrix.

3.3.3.2. Transductive Simple CNAPS

After exploring these methods, we converged on a simpler idea, which was to use an EM algorithm to construct the class means and covariance matrices for Simple CNAPS. Given labeled training examples, the means and covariance matrices can be estimated directly. However, this formulation also suggests that the means and covariance matrices can be learned from unlabeled examples using the EM algorithm, in a way similar to the popular k-means clustering algorithm. We use EM to iteratively refine the class probabilities and the parameters

using both labeled and unlabeled examples. When the unlabeled data is the test set, this corresponds to transductive learning. When the unlabeled data is additional training data, this is semi-supervised learning. We experimented with the transductive setting; Table 3 shows the results.

Table 3: Transductive Learning Results

Dataset	Domain?	Simple CNAPS (Baseline)	Simple CNAPS (Transductive)	Delta
ImageNet	In	58.6	59.4	0.8
OmniGlott	In	91.7	95.0	3.3
Aircraft	In	82.4	83.4	1.0
Birds	In	74.9	76.5	1.6
Textures	In	67.8	68.3	0.5
Fungi	In	46.9	49.2	2.3
vgg_flower	In	90.7	92.6	1.9
Traffic Sign	Out	73.5	70.8	-2.7
MSCOCO	Out	46.2	53.3	7.1
MNIST	Out	93.9	94.4	0.5
CIFAR10	Out	74.3	74.5	0.2
CIFAR100	Out	60.5	62.3	1.8

The experiment was performed on the same 12 datasets used to evaluate Simple CNAPS, with 7 in-domain datasets used to learn the adaptation model and 5 out-of-domain datasets for transfer. The table compares the baseline Simple CNAPS algorithm that only uses labeled training data with the Transductive Simple CNAPS algorithm that also uses all the unlabeled data in the test set. We see that on 11 out of 12 datasets, the transductive version achieves better results, with a median 1.6% accuracy improvement. We continued to experiment with Transductive Simple CNAPS on a variety of datasets and obtained consistently good results, but without showing dramatic improvements over the fully unsupervised version. Our results are summarized in an arXiv paper [35].

3.3.3.3. *Alternative Transductive Method*

As an alternative to the above transductive algorithm, we also experimented with a different, very simple algorithm. This algorithm uses Simple CNAPS as is, fully unsupervised, and then labels all the unlabeled examples using the results of Simple CNAPS learning. These are then reinterpreted as labeled examples, and a standard ResNet model is relearned using supervised learning from the entire dataset. This can be viewed as Simple CNAPS providing a weak supervision signal for supervised learning.

We designed an experiment to test this approach. We worked with the Google Meta-Dataset used in the original Simple CNAPS paper. The Meta-Dataset produces meta-training and meta-testing tasks. Each task has a support set, used for training with labels, and a query set, used for prediction and evaluation. In the original Simple CNAPS paper, the query sets are small, but our approach sought to use large sets of unlabeled data, so we modified the Meta-Dataset to produce large query sets.

We ran three experiments:

1. We used the pretrained Simple CNAPS from the original paper, and evaluated it on meta-test instances with large query sets, comparing our new approach with the original Simple CNAPS.
2. We meta-trained Simple CNAPS with large query sets and performed the same evaluation.
3. We meta-trained Simple CNAPS using the full loop, where in each meta-training episode, we labeled the unlabeled data as learned by Simple CNAPS and performed supervised learning before evaluating. Our hypothesis was that this will produce Simple CNAPS models that are more tuned to the new use, but at the cost of increased training time.

We selected six datasets for experimentation: ImageNet, CIFAR100, MNIST, Aircraft, Flower, and Traffic Signs. For each dataset, we sampled between 5-10 labeled examples per class. The remaining dataset is split into training, validation, and test sets, where we treat training and validation set as unlabeled.

We first finetuned Simple CNAPS on the support set. We then classified the training and validation set using our finetuned Simple CNAPS model. We also measured the accuracy of Simple CNAPS on the test set using the ground truth labels.

Next, we trained a Resnet18 model on the training and validation set, using the classifications of the Simple CNAPS model as the labels. Resnet18 is the backbone used by Simple CNAPS, so provides an apples-to-apples comparison. We then measured the accuracy of the Resnet18 on the test set. Note that the Resnet18 model was pretrained on the training classes of ImageNet.

Table 4 shows the results.

Table 4: Weak Supervision Results

Dataset	Num Classes	Resnet18	Simple CNAPS
ImageNet	100	50.12	53.13
Aircraft	15	51.25	45.83
Cifar100	100	38.23	39.45
MNIST	10	91.2	91.6
Flower	15	88.843	83.72
Traffic Signs	43	83.78	82.91

Each entry in the table is averaged over three trials. Each entry represents the accuracy on the test set. We see that the weakly supervised Resnet18 does quite a bit worse (about 3%) than Simple CNAPS on ImageNet, but much better on Aircraft and Flower (over 5% on each). This may be because the Resnet18 is pretrained on ImageNet so there is little to be gained by further training, but may also be because Aircraft and Flower need specialized feature extraction (e.g., reading labels on aircraft for model) that require further training of the Resnet. The upshot of this experiment is that this approach has promise on new datasets, but the results are inconclusive compared to our original transductive method.

3.3.4. Other Semi-Supervised Approaches Explored

3.3.4.1. *Graph-Based Semi-Supervised Classification*

The idea of using graphs is that we can express similarity between data instances in a graph and use those similarities to help classify when only a few labels are available. Specifically, we looked at spectral clustering algorithms which finds eigenvectors of the adjacency matrix. The typical pipeline for graph-based semi-supervised learning is as follows:

1. Choose a kernel function to construct an adjacency matrix.
2. Transform the adjacency matrix to improve separation of clustered data points.
3. Apply standard clustering and classification methods or apply graph-based classification methods to infer missing labels.

We identified a number of opportunities for improvement in this basic pipeline:

- Use a fixed kernel function that is more sensitive to local structure, or learn a kernel function, so that the induced graph is well clustered according to class label. This kernel function can be the output of a neural network, trained to produce a similarity score that induces a well-clustered graph.
- Given a certain kernel function, find sparse or randomized approximations to the kernel matrix that is used to construct the graph.
- Instead of computing a kernel function on the raw data points, use a learned embedding function on the data points. For example, this function can be the inference model in a variational autoencoder, or the latent features from a standard convolutional network trained for classification. We can also use meta-learning techniques to help produce highly optimized features that induce a successful Label Propagation, and data augmentation to help learn a better embedding that is smooth and covers the entire space of our dataset.
- In the spirit of CNAPs and other meta-learning algorithms, we can learn an amortized kernel matrix, so that at test time we can map a new data point to a vector of similarities with each other point in the training dataset. (Note that this is not the same as a learned kernel function that accepts a pair of vectors and outputs a single similarity score.)
- Instead of working with an exact graph, use Spectral Sparsification to work with an approximation to our graph, for faster and lower-memory algorithms.
- Given a certain graph, we have a variety of methods to choose from for transductive inference:
 - Label propagation (iterative spreading using weighted adjacency matrix) [36]
 - Label spreading (iterative spreading using graph Laplacian or normalized Laplacian).
 - Using signal processing techniques like Fourier smoothing [37]
 - Graph convolutional networks [38] and variational graph auto-encoders [39]
- Construct a graph and perform label inference to evaluate the quality of an embedding (rather than for the purpose of using the inferred labels).
- Inductive or amortized classification methods.

To get a baseline for the methods we developed, we conducted two experiments. First, since Label Propagation (LP) is one of the simplest models to build from, and incorporates some of the

key components mentioned above, we ran this method on a variety of datasets commonly used for semi-supervised learning. We constructed a weighted adjacency matrix using the top-K entries of the RBF kernel. This matrix was the used to perform LP. The results are shown in Figure 17.

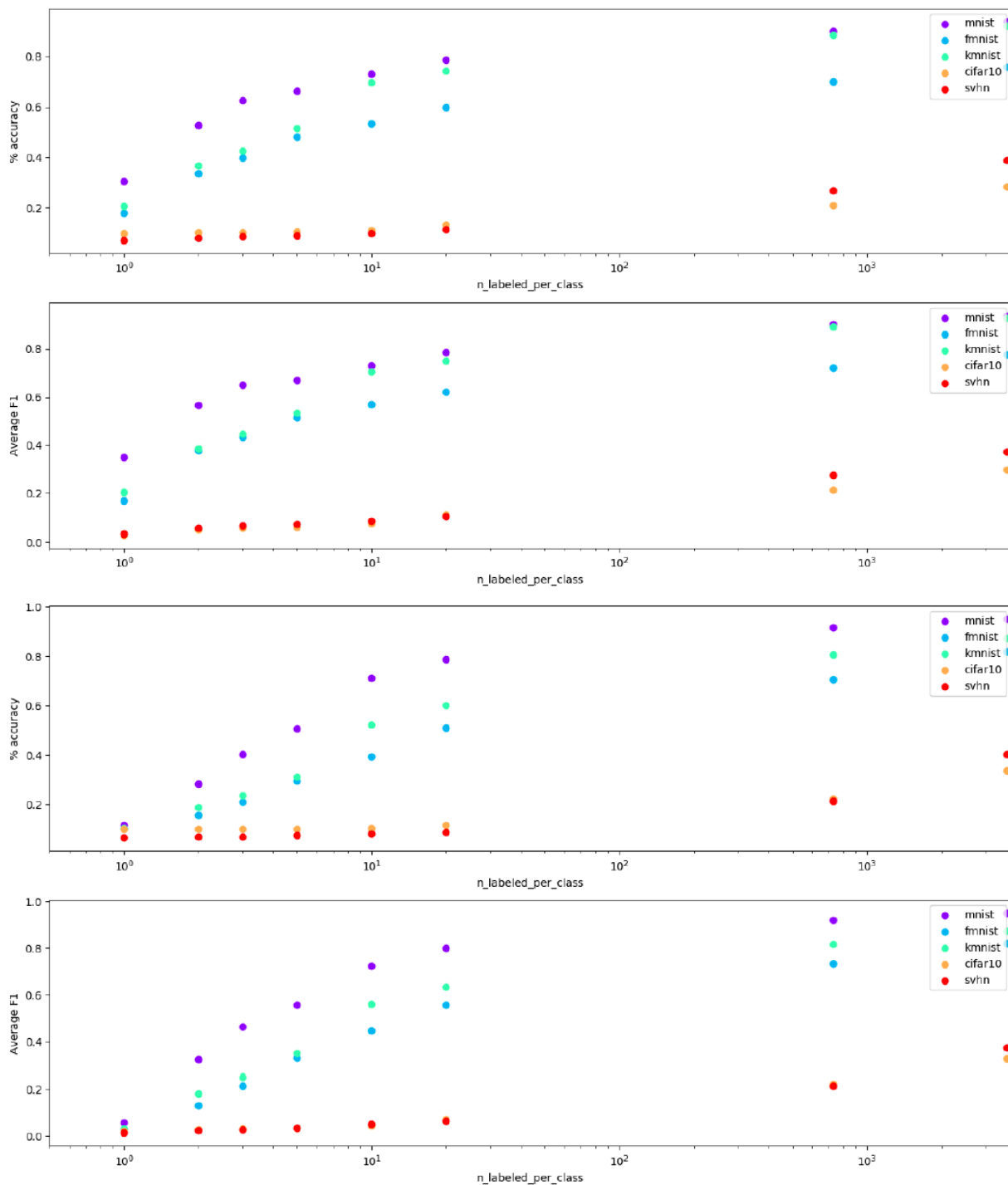


Figure 17: Classification Results: Transductive (top) and Inductive (bottom)

Second, as mentioned above, one possible modification to the graph-based classification pipeline is to learn a useful embedding in which graph-based methods like Label Propagation work well (i.e., where points from the same class map to nearby regions of the embedding space). To evaluate a method, we show that it takes advantage of both the structure in the dataset, as well as the provided labels. Therefore, we first evaluate the performance of embedding methods that are not aware of labels, and can only use structure in the input data.

3.3.4.2. *Structure Augmentation*

One of the advantages of a probabilistic model is that we can use any available data to learn it. In particular, we can gain a large amount of knowledge from unlabeled data alone using unsupervised learning. We can learn the deep structured model of the domain and its components that explains the data.

During the effort, we learned how the label relates to this structure, despite a small number of labeled instances. One of our key insights was that because we learned a structured representation without labels, we could augment it with a relatively simple model that could be learned efficiently from few labels. In other words, we learn to extract almost all the useful information from a training instance so that it becomes easy to predict the label. One way this works is to introduce a simple, thin layer to relate the labels to this structure.

Figure 18 shows how this might work for the example of Figure 4 (Section 3.1).

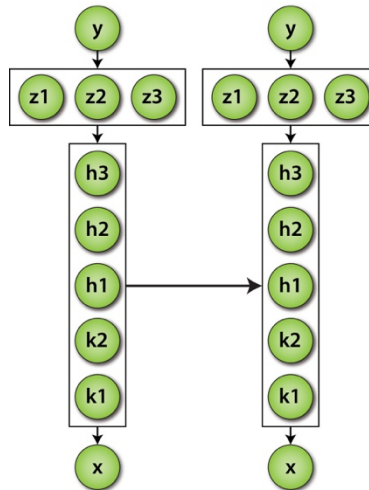


Figure 18: Separable DBN with Thin Layer for Label

If we only used the model on the right of Figure 4, we would have to learn a complete relationship between the label y and the hidden layer h , which might be hard to learn from few labels. Instead, we insert a thin layer of latent variables z between y and h . We can view z as a distributed representation of y and we can stipulate, for example, that the components of this distributed representation are conditionally independent given y . We also use this idea (latent

distributed representation) to summarize the data in our approaches to model transfer and active learning.

From unlabeled data, we can learn, using the methods of Section 3.1, a highly accurate conditional probability $P(h, k, x | z)$ that represents the entire deep structure of the data. From the small amount of data, we can learn the conditional probability $P(z | y)$. Because of our assumptions, this conditional probability distribution only has a small number of parameters. Although z was unobserved, we could infer the likelihood of the data x given z from our learned $P(h, k, x | z)$ and use that to learn $P(z | y)$.

Using the separable models, we described in Section 3.1.2, we were able to implement models of this sort. However, we have not yet been able to test them on datasets to evaluate the effectiveness of this approach.

3.3.4.3. *Pairwise Learning*

Another method we explored to improve semi-supervised learning was *pairwise learning*. Contrary to class labels, which are assigned to individual samples, pairwise learning receives multiple samples as inputs, and learns pairwise relationships pertinent to class membership. Examples of pairwise relationships include similarity (Bromley et al., 1994) and comparisons [41]. In neural network architectures, both approaches lead to training via Siamese networks [40]. Incorporating pairwise learning had tremendous potential to enhance weakly supervised methods for LwLL. Intuitively, this is because, in the presence of N labeled samples, the number of pairs is quadratic in N . Hence, pairwise learning naturally increases the number of samples available for training.

Similarity learning tries to learn lower dimensional representations that preserve the similarity structure, under an appropriate penalty loss, such as contrastive (Hadsell et al., 2006) or triplet loss (Norouzi et al., 2012). Similarity learning is an excellent means for training latent representations exploiting weak supervision. In particular, Siamese networks for similarity can naturally be used to embed unlabeled data in the same latent space as data for which labels exist. A classifier trained on the labeled data can then be directly applied to unlabeled data, operating on their images in the joint (and similarity-preserving) latent space. This embedding in latent space fits in very well with our probabilistic approach, where our SDPMs represent structured models with latent variables in which we can explicitly create components related to similarity. For example, we can create a probabilistic variant of the contrastive loss function by making the two class labels together generate the hidden units, rather than each single class individually. When the two classes are the same, this joint generation will tend to produce hidden unit values that are similar, while when the classes are different, they will cause the hidden units to diverge.

An alternative to similarity-based learning is *learning via comparisons* between instances, such as which of two instances is more typically a member of a class. Incorporating comparison labels to the training process has two significant advantages over class labels:

- Comparisons are often less noisy than class labels. Indeed, human labelers disagreeing when generating class judgments often exhibit reduced variability when asked to compare pairs of items instead.
- Even in the absence of noise, comparisons can be more informative compared to traditional class labels. This is because a labeler can produce both inter- *and* intra-class comparisons; though the former are implied by class labels, the latter are not revealed via class labels alone.

Both advantages directly motivate the study of comparison-based learning as a means of addressing LwLL. Our team members Ioannidis and Erdogmus recently proposed a Siamese network architecture that is trained on both class and comparison labels. Our research has shown that incorporating comparisons into the training pipeline can lead to a significant improvement: We established that, by incorporating comparisons, we could train and optimize a neural network with 5,974,577 parameters on a dataset of *only 80 samples*. In this setting, our combined model attains a classification performance of 0.914 AUC, 0.883 accuracy, and 0.64 F1 score, whereas training only on class labels can only achieve 0.835 AUC, 0.705 accuracy, and 0.517 F1 score.

Although our previous results with comparison-based learning were with neural networks, the idea of learning from comparisons is not neural-specific. We created probabilistic models with this capability to integrate the idea with our other advances, such as structure augmentation.

3.4 Active Learning

Under the PLEDGES effort, we developed methods to recommend instances for labeling by an expert.

3.4.1. Overview

3.4.1.1. *Decision-Theoretic Methods Based on SDPMs*

The goal of active learning is to select a set of instances from a large, unlabeled pool of instances that, when labeled, will improve a given trained model the most. However, estimating the effect of labeling any data point can be challenging, because we must estimate how similar the instance is to others already seen and how representative it is of future labels, in addition to considering the probability distribution over labels and the effect of each label on the model. In complex domains, such as images and natural language, notions of similarity and representativeness are hard to even define. As a result, much of active learning for these domains has been restricted to variants of “uncertainty sampling” (Holub et al., 2008; Joshi et al., 2009) that aims to pick least confident instances without regard to representativeness and diversity.

Our formulation of the active learning problem was to identify a set of K instances $\{x_1, \dots, x_K\}$ that have a high degree of uncertainty in the model posterior $p(y | x_k)$, are representative according to some measure $R(x_k)$, and are diverse according to some measure $D(\{x\})$. The Structured Deep Probabilistic Models (SDPMs) that we describe in Section 3.1 define tractable probability distributions that enable us to use effective strategies for this unified formulation. Instead of working with the original input space x , we used our encoder/decoder models to perform computations in the latent space z , in which characterizing the relative degree of similarity between examples, particularly for text, is much easier. This allowed us to effectively and accurately estimate representativeness and diversity, and address scalability issues in the optimization for active learning, as described below.

Representativeness. One of the key requirements for picking an instance to label is to be confident that the change in the model will actually result in a change in the test set performance. For example, we should not pick an unlabeled instance x that is an outlier in the data (i.e., $p(x)$ is small). Many of the existing techniques that rely on model uncertainty/confidence ignore this important aspect, in part because computing arbitrary $p(x)$ is impossible/intractable for most domains of interest. With our deep, structured representations, $p(z)$ is well defined and can be

computed if needed, and thus, by building on the intuition that $p(x)$ is well approximated by $p(z)$, we can estimate the representativeness of each instance.

Diversity. It is important for the instances that are being selected to: (1) be different from the training data to avoid redundant information, and (2) be different from each other, if multiple instances are selected to be labeled. Incorporating this notion into active learning, which is known as *density-based sampling*, has not received much attention recently due to the tractability considerations associated with estimating $p(x)$. An additional advantage of learning latent structured representations is that it can be used to define a metric (i.e., semantic distance between two instances x and x' can be well approximated with z and z'). We used this property to develop novel density-based active learning techniques for complex domains.

Scalability. Even after computing the uncertainty and representativeness of individual instances, and computing semantic distances between pairs of instances was efficient, a significant computing challenge remained: How do we search the combinatorial space over a large collection of unlabeled instances to select the *best* subset? Our deep representation provides further benefits for scalability by directly incorporating the representation in the optimization itself. Specifically, instead of performing the optimization over sets of x , we focused on finding the ideal set z efficiently by exploiting the structures and continuous spaces, followed by *decoding* them to find the best instances to label. As a concrete example, if the latent space z was continuous, we would optimize the search for a set of points as a continuous optimization problem solved using gradient descent, which is substantially more effective than a discrete search.

3.4.1.2. Active Learning Methods

The space of latent representations for SDPMs provided a suitable space in which query candidates for active learning are assessed. Computationally tractable methods for calculating the objectives and query set selections to achieve representative yet diverse sampling are essential. Team members Ioannidis and Erdogmus developed single and batch active learning methods based on several objectives, including label-parameter mutual information, parameter-entropy, Fisher information ratio, as well as D-Optimality and E-Optimality objectives (Sourati et al., 2016, 2017, 2018). We explored this spectrum of active learning objectives to determine the best tradeoff between expressiveness and computational complexity.

In Section 3.3.4.3, we discuss how we used pairwise learning to enhance the effectiveness of semi-supervised learning. Naturally, we sought to label the pairs of instances that lead to the best improvement in the model. To that end, we explored selection objectives, such as mutual information, entropy, Fisher Information, etc., in the context of pairwise learning.

These active learning objectives are monotone and submodular with respect to pairs of objects selected; therefore, selection is amenable to approximation via a greedy algorithm. However, applying the greedy algorithm to pairwise learning poses a significant challenge compared to class labels, because finding the pair that yields the largest marginal gain is quadratic, rather than linear, in the dataset size. In recent work, Ioannidis and Erdogmus investigated using geometric properties of sample pairs to accelerate the greedy algorithm. Our resulting accelerated implementation executes in less than one hour over a dataset with 108 comparisons; a naive algorithm on the same dataset can require more than 10 days to terminate.

For active learning, we considered the possibility of including pairwise comparison queries between examples, rather than simply querying the label of a single example. Pairwise

comparisons have the potential to improve active learning by providing more information with a lesser acquisition effort. For example, when labels are on a scale representing the degree to which an example has a certain property, asking which of two examples more has the property can be easier for a human annotator than coming up with a precise label. However, because the LwLL evaluation is fully automated, it was going to be hard to evaluate pairwise comparisons queries in general.

This led us to the question of whether pairwise learning queries would actually be beneficial in the LwLL setting. Pairwise comparisons are particularly beneficial where there are semantic relationships between classes, for example when classes lie on a scale. We anticipated that the types of classification problems arising in LwLL might have many different classes that do not necessarily have semantic relationships, such as lying on a scale.

There are other pairwise comparison queries we could ask, such as whether two examples have the same label, and this kind of query is possible with an automated evaluation. However, when there are no semantic relationships between variables, this query might be less useful than getting the labels of the two examples explicitly. Therefore, we believed that the lack of pairwise comparison queries would not severely impact active learning and we decided not to include pairwise comparisons in the evaluation, at least for our team.

Instead, we used uncertainty sampling for the evaluation, a simple algorithm that has proven hard to beat.

3.4.2. Active Learning Evaluation Preparation

We maintained and prepared active learning methods for the evaluation. To evaluate active learning, the process begins in the same way as for model transfer. We train a model on a number of diverse datasets. For evaluation, we are given a new dataset, different from the ones we trained on, with different class labels. We are given zero labeled examples from this dataset. We are given a budget for active learning. Using this budget, we choose which questions to ask. In response to these questions, we get answers, which could come, for example, from an expert or Mechanical Turk contributors, or be automatically generated.

To prepare for the evaluation, we updated our implementation to conform to the changing Jet Propulsion Laboratory (JPL) API and worked with JPL to fix a bug in their API.

We also refined our active learning in preparation for the program evaluation. Although we have developed a novel active learning algorithm based on data influence, and we showed that this method yields improvements compared to baseline active learning methods on standard models, we were not able to demonstrate benefit in our internal evaluations using this method with Simple CNAPS. Instead, we used margin sampling, in which we choose images to label that minimize the difference in probability between the most likely classification and the second most likely, as an indication of which image needs the most help to disambiguate the classification.

During active learning, we obtain images one by one, and update our estimate of the GMM weights each time. Due to computational limitations, we do not consider all images for labeling at each step; rather, we randomly subsample the images to compute the margin and select the minimum margin image in the subsample for labeling.

Active learning is potentially the most time-consuming step of our implementation. For each checkpoint in the evaluation, we want to ask for labels of many images. Ideally, we do this one image at a time, getting the result before deciding which image to query next. At each step,

deciding which label to query next requires examining each unlabeled image and computing its margin of uncertainty. On large datasets with millions of images, this can be expensive, and it is repeated many times for a checkpoint. However, subsampling the images to consider for querying makes the process much faster and can avoid diminishing the effectiveness of active learning too significantly if the subsamples are large enough. Therefore, we developed a subsampling budget that was designed to optimize our usage of computational resources during the test.

Next, we integrated active learning into our learning system run. Since we planned to use Simple CNAPS, the crucial question was how to integrate active learning into Simple CNAPS. We discovered that there is a natural way to do it. In Simple CNAPS, we learn representatives of each class and employ a distance metric known as the Mahalanobis distance to compute the distance of a point from each of the representatives. The chosen classification is the class with the nearest representative.

In this approach, the distance of a point from its nearest class representative (under the Mahalanobis distance) is a measure of the degree to which the point is typical of the class. In active learning, we generally want to obtain classifications about points that are not typical of any class, as we can learn the most from them. This suggests a simple strategy: choose the points to label that are furthest from their nearest class representatives. We began with this heuristic and then investigated more advanced methods.

Finally, we investigated integrating active learning into Simple CNAPS. Simple CNAPS is trained to extract features that can be classified using quadratic discriminants, consistent with Gaussian class-conditional probability density functions. This kind of model is amenable to semi-supervised learning of Gaussian mixture models (GMMs). In previous work, our teammate Deniz Erdogmus and his group developed methods to apply active learning to such models [46]. This work provides methods such as uncertainty sampling, density-weighted uncertainty sampling, and methods based on data influence and importance.

One challenge in these methods is the computational difficulty of computing the relevance of unlabeled examples to known examples. Sourati et al. developed submodular optimistic (upper bound) and pessimistic (lower bound) approximations of mutual information between known and unknown labels. These approximations can be the basis of a data influence active learning method for Simple CNAPS.

3.4.3. Active Learning with Simple CNAPS Experimentation

We ran experiments to determine the limits of active learning with Simple CNAPS. We found that active learning methods that work well with other learning approaches do not provide much benefit with Simple CNAPS. We wanted to investigate why that might be the case.

Active learning involves trying to select training examples that will best improve the accuracy of the model. This selection is performed without knowing the actual labels of those examples. We could not possibly do any better than “cheating” and looking at the label. This cheating performance provides an upper bound on the performance of active learning. We measured this cheating performance compared to active learning methods, as well as no active learning (random choosing of instances to label).

The results are shown in Figure 19. On CIFAR 10, there is only a small benefit to cheating compared to random sampling in the low-label case. Interestingly, this expands significantly with more labels. On SVHN, the results are similar, with slightly greater improvement.

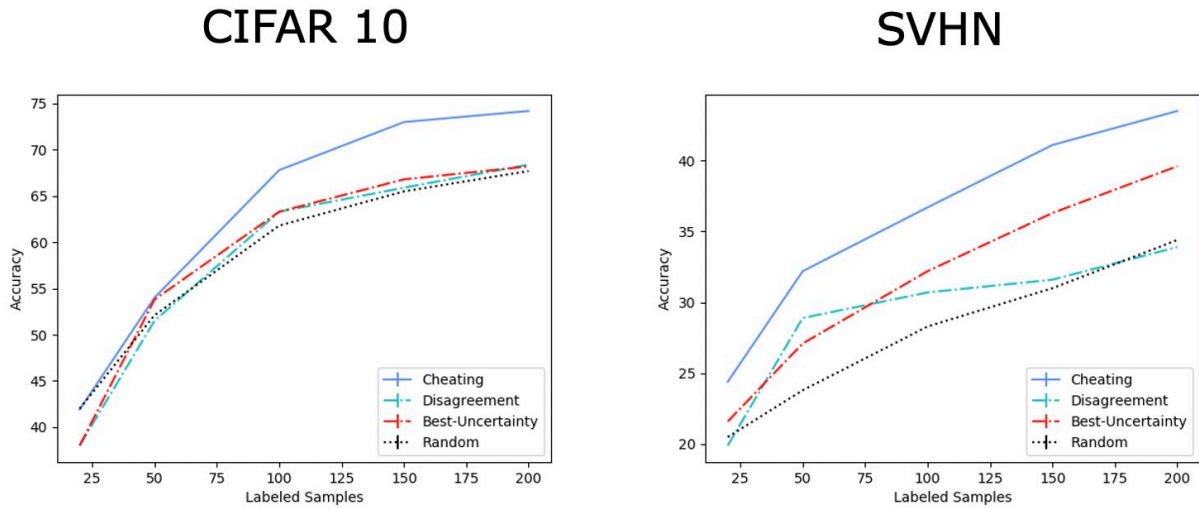


Figure 19: Limitations of Active Learning

3.4.4. Data Influence

For active learning, uncertainty sampling, in which the learner picks the instances to label based on what it is most uncertain about, is a commonly used method. It presents a strong baseline that is often hard to beat. As a result, we used it as a baseline in our system. However, there are some drawbacks to uncertainty sampling:

- Samples that are inherently difficult are uncertain, even if they do not add too much to the learned model.
- Outliers are uncertain, and we do not want to focus our learning on them.
- Learning algorithms, such as deep neural networks, are often miscalibrated and overconfident, estimating uncertainty incorrectly.
- Even if we correctly identify uncertain samples, it is hard to estimate their effect on predictions.

Because of these drawbacks, researchers search for alternative measures of desirability of a sample to be labeled. One approach is to use the concept of data influence, which asks which training sample was the most responsible for a given test set prediction. Intuitively, we should pick samples to label that have high predicted influence on future predictions. We can formally define the following questions, all of which measure data influence in some way:

- Which unlabeled instance is least affected by the existing training instances?
- Which unlabeled instance, if labeled, would be very different from training data?
- Which unlabeled instance, when added to training data, would be most important?

We experimented with methods that attempt to answer the third question. In our experiments, we call this Importance. We compare it to uncertainty sampling, which is labeled Entropy, and to the Random method that randomly selects samples to label.

We first experimented with a toy dataset designed to bring out some challenges in active inference. This dataset, shown in Figure 20, consists of three classes of very different sizes. There is a small amount of overlap between the classes. As seen in Figure 21, the Importance method is significantly more accurate than the other two methods, even with as few as ten samples.

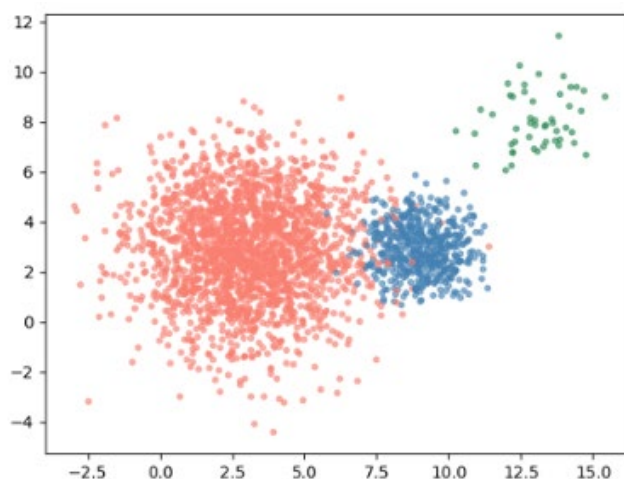


Figure 20: Three-Class, Unbalanced Dataset

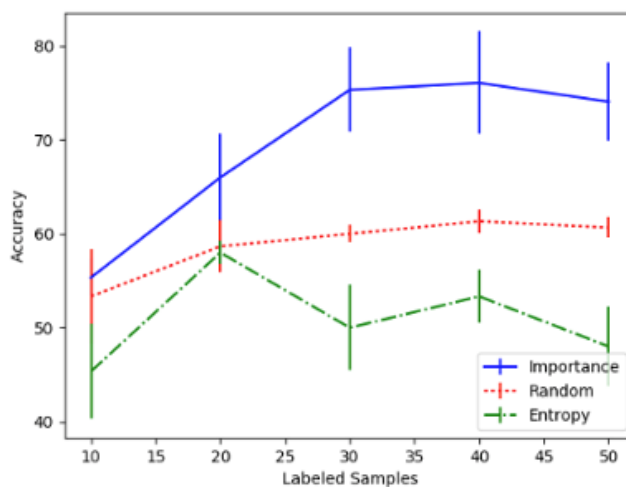


Figure 21: Results on Three-Class Dataset

We also experimented with a real dataset, CIFAR-10. Here, we found we had to add an estimate of the distance of a sample from the training set to the measure of importance; without this, Importance performed quite badly. However, we found that the combination of Importance + Distance outperformed all other methods, including Entropy + Distance combined.

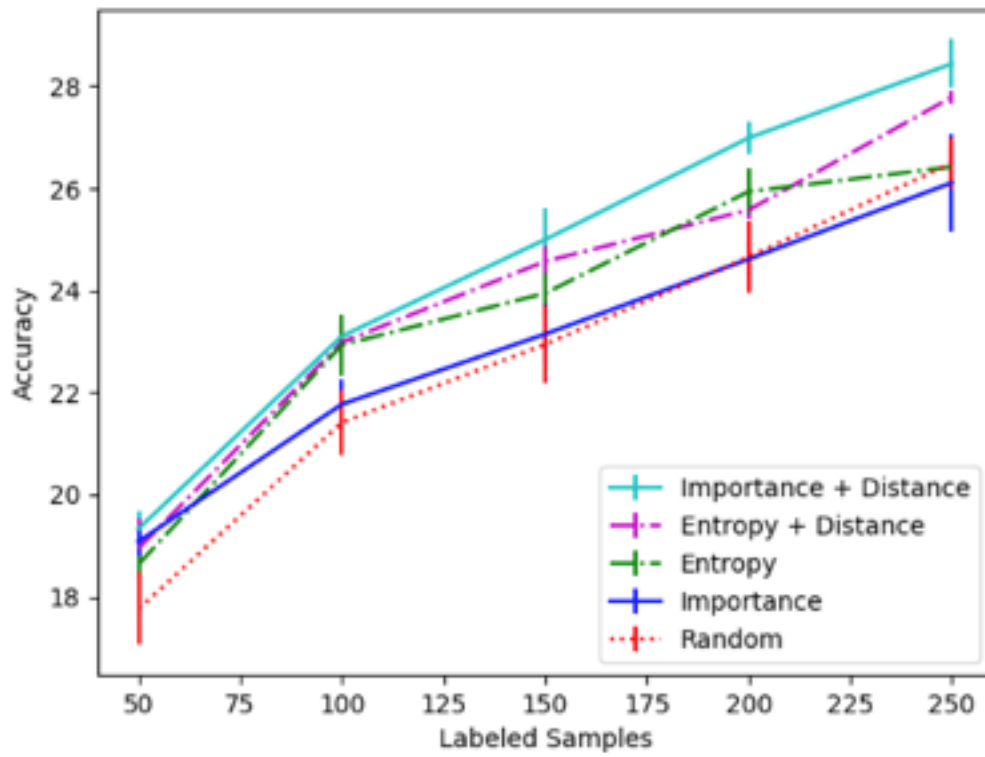


Figure 22: Results on CIFAR-10

4.0 RESULTS AND DISCUSSION

4.1 Summary of Results

Under *Task 1: Probabilistic Model Components*, we made significant progress on the theory and design of structured deep probabilistic models (SDPMs) and tractable high-capacity probabilistic models. We researched a number of advances in SDPMs, including amortized population Gibbs samplers, conjugate energy-based models, Gaussian mixture variational autoencoders, Bayesian nonparametric models, and label efficiency for disentangled representations. We also made significant progress on separable models, which are a form of tractable high-capacity probabilistic models. We implemented inference and learning algorithms for these models, and held demonstrations showing orders of magnitude improvements in scalability over traditional probabilistic graphical models.

Our methods for *Model Transfer*, under *Task 3*, were the basis for our work on semi-supervised learning. We designed, developed, and experimented with the Simple CNAPS meta-learning algorithm, showing significant improvements over the state of the art at few-shot visual classification. We also developed extensions of Simple CNAPS to few-shot object detection. In addition, we developed a zero-shot/few-shot algorithm for object detection based on weak supervision.

Under *Task 2: Semi-Supervised Learning*, we designed several extensions of Simple CNAPS to the semi-supervised setting, including transductive Simple CNAPS, which showed additional improvement at few-shot visual classification over Simple CNAPS. We integrated and tested our transductive Simple CNAPS algorithm with the JPL API for both visual classification and object detection. We also investigated a range of other approaches to semi-supervised learning including graph-based approaches, structure augmentation, and pairwise learning.

Under *Task 4: Active Learning*, we researched the foundations of active learning, which led to new methods based on data importance. We compared active learning methods based on data importance to baseline methods based on uncertainty sampling and showed improvement. We also conducted experiments that demonstrated that ultimate improvements in accuracy that could be expected from active learning were limited.

Our work on the project led to 18 publications in conferences and workshops (see Appendix A).

4.2 Internal Evaluation Results

Prior to the program evaluation, we ran an internal evaluation, testing transductive semi-supervised learning and active learning, on standard benchmarks. One of our goals was to determine to what degree our methods meet the Phase I goal of a 3 orders of magnitude reduction in the number of labels required to achieve a given level of accuracy. Our experiments were on CIFAR-10. We ran 2 versions of the experiments, one with 10 classes and one with 2 classes (ship versus truck). We compared a baseline of training Resnet 18 using fully supervised learning on the given number of training examples to Simple CNAPS with much fewer examples.

First, we report results for semi-supervised learning. With ten classes, we consistently see slightly better than two orders of magnitude improvement in the number of labels needed. For example, with 50 examples, Simple CNAPS with best uncertainty exhibits 53.8% accuracy,

while Resnet 18 has 51.8% accuracy with 5,000 examples. With 200 examples, Simple CNAPS with best uncertainty has 68.2% accuracy, while Resnet 18 gets 66.2% with 20,000 examples.

For 2 classes, we see a 250x improvement in the number of labels needed. With 40 examples, simple CNAPS with best uncertainty exhibits 94.2% accuracy, while Resnet 18 has 94.4% accuracy with 10,000 examples. Even with just 20 examples, Simple CNAPS with best uncertainty shows 91.9% accuracy, for a 500x improvement in the number of samples needed to obtain accuracy close to the baseline.

Second, for active learning, we compared three active learning methods: no active learning (random sampling); best uncertainty, which is the best of several algorithms related to uncertainty sampling; and our own active learning method based on data influence. However, our data influence method was not yet integrated with Simple CNAPS, so we compared active learning results on the baseline model. We found between 1-3% improvement of the data influence method over best uncertainty. This is consistent with other experience that the benefits of active learning may be limited, but nevertheless, data influence seems to be a superior active learning method.

4.3 Phase I Evaluation

Unfortunately, the official program evaluation yielded very poor results. While our methods were able to perform few-shot learning on occasion, they did not generally improve on the baseline method. We do not have a good explanation of this, considering that our methods had significantly improved on the state-of-the-art in published experiments on standard benchmarks, and our end-to-end self-test had been encouraging. It is possible that we were plagued by engineering issues and possible bugs that were unfortunately not addressed for the second evaluation.

5.0 CONCLUSIONS

The main premise of our approach was that probabilistic methods could provide a strong foundation for label-efficient learning when using advanced representations such as structured deep probabilistic models and separable models. We were able to make strong advances for both these kinds of models and our work shows that both methods have significant promise, with considerable accuracy and scalability improvements over previous probabilistic graphical models. However, neither approach was mature enough to match the performance of neural approaches on well-understood practical applications like visual classification and object detection, so we did not use them in the program evaluations.

We also showed that probabilistic formulations of neural models can lead to significant advances. Our Simple CNAPS method is a probabilistic reinterpretation of the CNAPS meta-learning algorithm. Simple CNAPS achieved a dramatic improvement over state-of-the-art methods, including CNAPS, at few-shot visual classification. Simple CNAPS also formed the basis of a variety of extensions, including a transductive version that showed further improvements.

For active learning, our foundational work led to a suite of new methods based on data importance, and we were able to show some improvement over existing methods based on uncertainty sampling. However, our experiments also showed that the potential improvements in classification accuracy from active learning may be limited.

6.0 RECOMMENDATIONS

Our research showed that structured deep probabilistic models and separable models have considerable promise but are currently lacking in maturity compared to well-established neural methods. Therefore, further research into these methods is recommended. This research should focus on three main goals: (1) scalable implementations with efficiency comparable to that of optimized neural implementations; (2) development of model designs that are able to express application-relevant concepts (such as convolutions for image recognition); (3) explorations of problem classes for which large-scale probabilistic methods may be well suited.

We also recommend further investigation of Simple CNAPS and its variants as a basis for practical few-shot learning. Our in-house experimentation showed the method to be extremely effective, but the program evaluations were less successful, for unknown reasons. A wider range of experiments on a large number of tasks and datasets will help evaluate the utility of the approach.

7.0 REFERENCES

1. Sculley, D., Phillips, T., Ebner, D., Chaudhary, V., & Young, M. (2014). Machine learning: The high-interest credit card of technical debt.
2. Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *ArXiv Preprint ArXiv:1312.6114*.
3. Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. *ArXiv Preprint ArXiv:1401.4082*.
4. Le, T. A., Baydin, A. G., & Wood, F. (2016). Inference compilation and universal probabilistic programming. *ArXiv Preprint ArXiv:1610.09900*.
5. Tran, D., Kucukelbir, A., Dieng, A. B., Rudolph, M., Liang, D., & Blei, D. M. (2016). Edward: A library for probabilistic modeling, inference, and criticism. *ArXiv Preprint ArXiv:1610.09787*.
6. Eli, B., Chen, J. P., Jankowiak, M., Karaletsos, T., Obermeyer, F., Pradhan, N., ... Goodman, N. (2017). Pyro: Deep probabilistic programming.
7. Narayanaswamy, S., Paige, T. B., Van de Meent, J.-W., Desmaison, A., Goodman, N., Kohli, P., ... Torr, P. (2017). Learning disentangled representations with semi-supervised deep generative models (pp. 5925–5935). Presented at the Advances in Neural Information Processing Systems.
8. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... Isard, M. (2016). TensorFlow: A System for Large-Scale Machine Learning. In *OSDI* (Vol. 16, pp. 265–283).
9. Huang, J., Chavira, M., & Darwiche, A. (2006). Solving MAP Exactly by Searching on Compiled Arithmetic Circuits. (Vol. 6, pp. 3–7). Presented at the AAAI.
10. Poon, H., & Domingos, P. (2011). Sum-product networks: A new deep architecture. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on* (pp. 689–690). IEEE.
11. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems* (pp. 2672–2680).
12. Bornschein, J., & Bengio, Y. (2014). Reweighted wake-sleep. *ArXiv Preprint ArXiv:1406.2751*.
13. Hinton, G. E., Dayan, P., Frey, B. J., & Neal, R. M. (1995). The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214), 1158–1161.
14. Le, T. A., Kosiorek, A. R., Siddharth, N., Teh, Y. W., & Wood, F. (2018). Revisiting Reweighted Wake-Sleep. *ArXiv Preprint ArXiv:1805.10469*.
15. Le, T. A., Igl, M., Rainforth, T., Jin, T., & Wood, F. (2017). Auto-Encoding Sequential Monte Carlo. *ArXiv Preprint ArXiv:1705.10306*.
16. Pfeffer, A. (2001). Sufficiency, separability and temporal probabilistic models (pp. 421–428). Presented at the Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc.
17. Pfeffer, A. (2006). Approximate separability for weak interaction in dynamic systems. In *Uncertainty in Artificial Intelligence*.

18. Brahma, P. P., Wu, D., & She, Y. (2016). Why Deep Learning Works: A Manifold Disentanglement Perspective. *IEEE Trans. Neural Netw. Learning Syst.*, 27(10), 1997–2008.
19. Jiang, Z., Zheng, Y., Tan, H., Tang, B., & Zhou, H. (2016). Variational deep embedding: An unsupervised and generative approach to clustering. *ArXiv Preprint ArXiv:1611.05148*.
20. Kingma, D. P., Mohamed, S., Rezende, D. J., & Welling, M. (2014). Semi-supervised learning with deep generative models (pp. 3581–3589). Presented at the Advances in neural information processing systems.
21. Tomczak, J. M., & Welling, M. (2017). VAE with a VampPrior. *ArXiv Preprint ArXiv:1705.07120*.
22. Requeima, J., Gordon, J., Bronskill, J., Nowozin, S., & Turner, R. E. (2019). Fast and flexible multi-task classification using conditional neural adaptive processes (pp. 7959–7970). Presented at the Advances in Neural Information Processing Systems.
23. Bateni, P., Goyal, R., Masrani, V., Wood, F., & Sigal, L. (2020). Improved few-shot visual classification (pp. 14493–14502). Presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.
24. Athiwaratkun, B., Finzi, M., Izmailov, P., & Wilson, A. G. (2018). There are many consistent explanations of unlabeled data: Why you should average. *ArXiv Preprint ArXiv:1806.05594*.
25. Balestrierio, R., Roger, V., Glotin, H. G., & Baraniuk, R. G. (2017). Semi-Supervised Learning via New Deep Network Inversion. *ArXiv Preprint ArXiv:1711.04313*.
26. Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., & Raffel, C. (2019). Mixmatch: A holistic approach to semi-supervised learning. *ArXiv Preprint ArXiv:1905.02249*.
27. Dai, Z., Yang, Z., Yang, F., Cohen, W. W., & Salakhutdinov, R. R. (2017). Good semi-supervised learning that requires a bad GAN (pp. 6510–6520). Presented at the Advances in neural information processing systems.
28. Oliver, A., Odena, A., Raffel, C. A., Cubuk, E. D., & Goodfellow, I. (2018). Realistic evaluation of deep semi-supervised learning algorithms (pp. 3235–3246). Presented at the Advances in Neural Information Processing Systems.
29. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved techniques for training GANs (pp. 2234–2242). Presented at the Advances in neural information processing systems.
30. Tarvainen, A., & Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results (pp. 1195–1204). Presented at the Advances in neural information processing systems.
31. Xu, J., Zhang, Z., Friedman, T., Liang, Y., & Broeck, G. V. den. (2017). A semantic loss function for deep learning with symbolic knowledge. *ArXiv Preprint ArXiv:1711.11157*.
32. Nalisnick, E., Hertel, L., & Smyth, P. (2016). Approximate inference for deep latent gaussian mixtures (Vol. 2). Presented at the NIPS Workshop on Bayesian Deep Learning.
33. Locatello, F., Tschannen, M., Bauer, S., Rätsch, G., Schölkopf, B., & Bachem, O. (2019). Disentangling factors of variation using few labels. *ArXiv Preprint ArXiv:1905.01258*.
34. Alemi, A. A., Fischer, I., Dillon, J. V., & Murphy, K. (2016). Deep variational information bottleneck. *ArXiv Preprint ArXiv:1612.00410*.

35. Bateni, P., Barber, J., van de Meent, J.-W., & Wood, F. (2020). Improving Few-Shot Visual Classification with Unlabeled Examples. *ArXiv Preprint ArXiv:2006.12245*.
36. Zhu, X., & Ghahramani, Z. (2002). Learning from labeled and unlabeled data with label propagation.
37. Ortega, A., Frossard, P., Kovačević, J., Moura, J. M., & Vandergheynst, P. (2018). Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5), 808–828.
38. Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *ArXiv Preprint ArXiv:1609.02907*.
39. Kipf, T. N., & Welling, M. (2016). Variational graph auto-encoders. *ArXiv Preprint ArXiv:1611.07308*.
40. Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., & Shah, R. (1994). Signature verification using a “Siamese” time delay neural network (pp. 737–744). Presented at the Advances in Neural Information Processing Systems.
41. Sculley, D. (2010). Combined regression and ranking (pp. 979–988). Presented at the Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM.
42. Hadsell, R., Chopra, S., & LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping (pp. 1735–1742). Presented at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
43. Norouzi, M., Fleet, D. J., & Salakhutdinov, R. R. (2012). Hamming distance metric learning (pp. 1061–1069). Presented at the Advances in neural information processing systems.
44. Holub, A., Perona, P., & Burl, M. C. (2008). Entropy-based active learning for object recognition (pp. 1–8). Presented at the IEEE Computer Vision and Pattern Recognition Workshops., IEEE.
45. Joshi, A. J., Porikli, F., & Papanikolopoulos, N. (2009). Multi-class active learning for image classification.
46. Sourati, J., Akcakaya, M., Dy, J. G., Leen, T. K., & Erdogmus, D. (2016). Classification active learning based on mutual information. *Entropy*, 18(2), 51.
47. Sourati, J., Akcakaya, M., Erdogmus, D., Leen, T. K., & Dy, J. G. (2018). A probabilistic active learning algorithm based on fisher information ratio. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(8), 2023–2029.
48. Sourati, J., Akcakaya, M., Leen, T. K., Erdogmus, D., & Dy, J. G. (2017). Asymptotic analysis of objectives based on fisher information in active learning. *Journal of Machine Learning Research*, 18(34), 1–41.

Appendix A. Publications

The following publications at conferences or workshops were supported by PLEDGES/LwLL.

Biza, O., van de Meent, J.-W., Wong, L., & Platt, R. *Action Priors for Large Action Spaces in Robotics*. International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2021.

In robotics, it is often not possible to learn useful policies using pure model-free reinforcement learning without significant reward shaping or curriculum learning. As a consequence, many researchers rely on expert demonstrations to guide learning. However, acquiring expert demonstrations can be expensive. This paper proposes an alternative approach where the solutions of previously solved tasks are used to produce an action prior that can facilitate exploration in future tasks. The action prior is a probability distribution over actions that summarizes the set of policies found solving previous tasks. Our results indicate that this approach can be used to solve robotic manipulation problems that would otherwise be infeasible without expert demonstrations. Source code is available at https://github.com/ondrejba/action_priors.

Wu, H., Zimmermann, H., Sennesh, E., Le, T. A., & van de Meent, J.-W. *Amortized Population Gibbs Samplers with Neural Sufficient Statistics*. In Proceedings of the International Conference on Machine Learning, 2020.

We develop amortized population Gibbs (APG) samplers, a class of scalable methods that frame structured variational inference as adaptive importance sampling. APG samplers construct high-dimensional proposals by iterating over updates to lower-dimensional blocks of variables. We train each conditional proposal by minimizing the inclusive KL divergence with respect to the conditional posterior. To appropriately account for the size of the input data, we develop a new parameterization in terms of neural sufficient statistics. Experiments show that APG samplers can be used to train highly structured deep generative models in an unsupervised manner, and achieve substantial improvements in inference accuracy relative to standard autoencoding variational methods.

Wu, H., Esmaili, B., Wick, M. L., Tristan, J.-B., & van de Meent, J.-W. *Conjugate Energy-Based Models*. Third Symposium on Advances in Approximate Bayesian Inference, 2021.

We propose conjugate energy-based models (EBMs), a class of deep latent-variable models with a tractable posterior. Conjugate EBMs have similar use cases as variational autoencoders, in the sense that they learn an unsupervised mapping between data and latent variables. However, these models omit a generator, which allows them to learn more flexible notions of similarity between data points. Our experiments demonstrate that conjugate EBMs achieve competitive results in terms of image modelling, predictive power of latent space, and out-of-distribution detection on a variety of datasets.

Smedemark-Margulies, N., Park, J. Y., Daniels, M., Yu, R., van de Meent, J.-W., & Hand, P. *Generator Surgery for Compressed Sensing*. NeurIPS Workshop on Deep Learning and Inverse Problems, 2020.

Image recovery from compressive measurements requires a signal prior for the images being reconstructed. Recent work has explored the use of deep generative models with low latent dimension as signal priors for such problems. However, their recovery performance is limited by high representation error. We introduce a method for achieving low representation error using generators as signal priors. Using a pre-trained generator, we remove one or more initial blocks at test time and optimize over the new, higher-dimensional latent space to recover a target image. Experiments demonstrate significantly improved reconstruction quality for a variety of network architectures. This approach also works well for out-of-training-distribution images and is competitive with other state-of-the-art methods. Our experiments show that test-time architectural modifications can greatly improve the recovery quality of generator signal priors for compressed sensing.

Biza, O., Platt, R., van de Meent, J.-W., & Wong, L. L. S. *Learning Discrete State Abstractions with Deep Variational Inference*. Third Symposium on Advances in Approximate Bayesian Inference, 2020.

Abstraction is crucial for effective sequential decision making in domains with large state spaces. In this work, we propose an information bottleneck method for learning approximate bisimulations, a type of state abstraction. We use a deep neural encoder to map states onto continuous embeddings. We map these embeddings onto a discrete representation using an action-conditioned hidden Markov model, which is trained end-to-end with the neural network. Our method is suited for environments with high-dimensional states and learns from a stream of experience collected by an agent acting in a Markov decision process. Through this learned discrete abstract model, we can efficiently plan for unseen goals in a multi-goal Reinforcement Learning setting. We test our method in simplified robotic manipulation domains with image states. We also compare it against previous model-based approaches to finding bisimulations in discrete grid-world-like environments.

Zimmermann, H., Wu, H., Esmaili, B., Stites, S., & van de Meent, J.-W. *Nested Variational Inference*. Third Symposium on Advances in Approximate Bayesian Inference, 2021.

We develop nested variational inference (NVI), a family of methods that learn proposals for nested importance samplers by minimizing an inclusive or exclusive KL divergence at each level of nesting. NVI is applicable to many commonly used importance sampling strategies and additionally provides a mechanism for learning intermediate densities, which can serve as heuristics to guide the sampler. Our experiments apply NVI to learn samplers targeting (a) an unnormalized density using annealing and (b) the posterior of a hidden Markov model. We observe improved sample quality in terms of log average weight and effective sample size.

McInerney, D. J., Dabiri, B., Touret, A.-S., Young, G., van de Meent, J.-W., & Wallace, B. C. Query-Focused EHR Summarization to Aid Imaging Diagnosis. Machine Learning for Healthcare, 2020.

Electronic Health Records (EHRs) provide vital contextual information to radiologists and other physicians when making a diagnosis. Unfortunately, because a given patient's record may contain hundreds of notes and reports, identifying relevant information within these in the short time typically allotted to a case is very difficult. We propose and evaluate models that extract

relevant text snippets from patient records to provide a rough case summary intended to aid physicians considering one or more diagnoses. This is hard because direct supervision (i.e., physician annotations of snippets relevant to specific diagnoses in medical records) is prohibitively expensive to collect at scale. We propose a distantly supervised strategy in which we use groups of International Classification of Diseases (ICD) codes observed in ‘future’ records as noisy proxies for ‘downstream’ diagnoses. Using this we train a transformer-based neural model to perform extractive summarization conditioned on potential diagnoses. This model defines an attention mechanism that is conditioned on potential diagnoses (queries) provided by the diagnosing physician. We train (via distant supervision) and evaluate variants of this model on EHR data from Brigham and Women’s Hospital in Boston and MIMIC-III (the latter to facilitate reproducibility). Evaluations performed by radiologists demonstrate that these distantly supervised models yield better extractive summaries than do unsupervised approaches. Such models may aid diagnosis by identifying sentences in past patient reports that are clinically relevant to a potential diagnosis. Code is available at <https://github.com/dmccinerney/ehr-extraction-models>.

Bozkurt, A., Esmaili, B., Brooks, D. H., Dy, J. G., & van de Meent, J.-W. *Rate-Regularization and Generalization in VAEs*. Artificial Intelligence and Statistics (AISTATS), 2020.

Variational autoencoders (VAEs) optimize an objective that comprises a reconstruction loss (the distortion) and a KL term (the rate). The rate is an upper bound on the mutual information, which is often interpreted as a regularizer that controls the degree of compression. We here examine whether inclusion of the rate term also improves generalization. We perform rate-distortion analyses in which we control the strength of the rate term, the network capacity, and the difficulty of the generalization problem. Lowering the strength of the rate term paradoxically improves generalization in most settings, and reducing the mutual information typically leads to underfitting. Moreover, we show that generalization performance continues to improve even after the mutual information saturates, indicating that the gap on the bound (i.e., the KL divergence relative to the inference marginal) affects generalization. This suggests that the standard spherical Gaussian prior is not an inductive bias that typically improves generalization, prompting further work to understand what choices of priors improve generalization in VAEs.

Pouya Pezeshkpour, Zhengli Zhao, Sameer Singh. *Data Importance-Based Active Learning for Limited Labels*. CVPR Workshop on Visual Learning with Limited Labels (VL3), 2020.

Active learning aims to reduce the required labeled data by selecting which unlabeled instances should be labeled next for a given model, instead of picking instances randomly. To select the instances to label, most existing approaches primarily rely on some form of uncertainty in the current predictions, as obtaining labels for these instances is expected to be the most useful for the current model. Despite the success of these approaches, uncertainty sampling does not directly evaluate what will change if an instance is labeled. In this work, we introduce a formulation of active learning that directly estimates this effect of adding an instance to the training data on the model, and uses this so-called data importance to select the next instance to label. In particular, we use a recently introduced representer point representation of the model to efficiently estimate the importance of samples on the model predictions, and use it for active

learning by selecting points that will have the largest impact. We evaluate our model on both synthetic and real-world datasets, demonstrating the utility of data importance in active learning for the limited label regime.

Pouya Pezeshkpour, Zhengli Zhao, Sameer Singh. *On the Utility of Active Instance Selection for Few-Shot Learning*. NeurIPS Workshop on Human and Model in the Loop Evaluation and Training Strategies (HAMLETS), 2020.

Few-shot learning aims to understand new concepts by transferring knowledge and utilizing very few randomly selected labeled samples. Instead of selecting these instances randomly, active learning provides a promising alternative. In this work, we investigate the effectiveness of actively identifying informative samples on the performance of few-shot learning models. We show that despite the fact that regular classification tasks with larger amounts of labeled data benefit from active learning approaches, these benefits do not reliably generalize to the few shot learning task. We characterize the best possible active few-shot learning performance, by introducing Single-Instance-Oracle and Batch-Oracle as active methods that assume access to labels of the unlabeled pool and the test set, and show via these “upper bounds” that we do not have a significant room for improving few-shot models through actively selecting instances.

Vaden Masrani. *All in the (Exponential) Family: Information Geometry and Thermodynamic Variational Inference*. International Conference on Machine Learning (ICML), 2020.

The recently proposed Thermodynamic Variational Objective (TVO) leverages thermodynamic integration to provide a family of variational inference objectives, which both tighten and generalize the ubiquitous Evidence Lower Bound (ELBO). However, the tightness of TVO bounds was not previously known, an expensive grid search was used to choose a ‘schedule’ of intermediate distributions, and model learning suffered with ostensibly tighter bounds. In this work, we propose an exponential family interpretation of the geometric mixture curve underlying the TVO and related path sampling methods, which allows us to characterize the gap in likelihood bounds as a sum of KL divergences. We propose a principled technique for choosing intermediate distributions using equal spacing in the moment parameters of our exponential family, and empirically demonstrate that this schedule can improve performance by adapting to the shape of the integrand across training. Finally, we derive a reparameterized gradient estimator which improves model learning and allows the TVO to benefit from more refined bounds. To further contextualize our contributions, we provide a unified framework for understanding thermodynamic integration and the TVO using Taylor series remainders.

William Harvey, Andreas Munk, Frank Wood. *Attention for Inference Compilation*. ProbProg, 2020.

We present a new approach to automatic amortized inference in universal probabilistic programs which improves performance compared to current methods. Our approach is a variation of inference compilation (IC) which leverages deep neural networks to approximate a posterior distribution over latent variables in a probabilistic program. A challenge with existing IC network architectures is that they can fail to model long-range dependencies between latent variables. To address this, we introduce an attention mechanism that attends to the most salient variables previously sampled in the execution of a probabilistic program. We demonstrate that

the addition of attention allows the proposal distributions to better match the true posterior, enhancing inference about latent variables in simulators.

Andrew Warrington, Saeid Naderiparizi, Frank Wood. *Coping with Simulators That Don't Always Return*. AISTATS, 2020.

Deterministic models are approximations of reality that are easy to interpret and often easier to build than stochastic alternatives. Unfortunately, as nature is capricious, observational data can never be fully explained by deterministic models in practice. Observation and process noise need to be added to adapt deterministic models to behave stochastically, such that they are capable of explaining and extrapolating from noisy data. We investigate and address computational inefficiencies that arise from adding process noise to deterministic simulators that fail to return for certain inputs; a property we describe as "brittle." We show how to train a conditional normalizing flow to propose perturbations such that the simulator succeeds with high probability, increasing computational efficiency.

Andreas Munk, Adam Ścibior, Frank Wood. *Deep Probabilistic Surrogate Networks for Universal Simulator Approximation*. ProbProg 2020.

We present a framework for automatically structuring and training fast, approximate, deep neural surrogates of existing stochastic simulators. Unlike traditional approaches to surrogate modeling, our surrogates retain the interpretable structure of the reference simulators. The particular way we achieve this allows us to replace the reference simulator with the surrogate when undertaking amortized inference in the probabilistic programming sense. The fidelity and speed of our surrogates allow for not only faster "forward" stochastic simulation but also for accurate and substantially faster inference. We support these claims via experiments that involve a commercial composite-materials curing simulator. Employing our surrogate modeling technique makes inference an order of magnitude faster, opening up the possibility of doing simulator-based, non-invasive, just-in-time parts quality testing; in this case inferring safety-critical latent internal temperature profiles of composite materials undergoing curing from surface temperature profile measurements.

Christian Weilbach, Boyan Beronov, William Harvey, Frank Wood. *Efficient Inference Amortization in Graphical Models using Structured Continuous Conditional Normalizing Flows*. AISTATS, 2020.

We introduce a more efficient neural architecture for amortized inference, which combines continuous and conditional normalizing flows using a principled choice of structure. Our gradient flow derives its sparsity pattern from the minimally faithful inverse of its underlying graphical model. We find that this factorization reduces the necessary numbers both of parameters in the neural network and of adaptive integration steps in the ODE solver. Consequently, the throughput at training time and inference time is increased, without decreasing performance in comparison to unconstrained flows. By expressing the structural inversion and the flow construction as compilation passes of a probabilistic programming language, we demonstrate their applicability to the stochastic inversion of realistic models such as convolutional neural networks (CNN).

Peyman Bateni, Raghav Goyal, Vaden Masrani, Frank Wood, Leonid Sigal. *Improved Few-Shot Visual Classification*. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020

Few-shot learning is a fundamental task in computer vision that carries the promise of alleviating the need for exhaustively labeled data. Most few-shot learning approaches to date have focused on progressively more complex neural feature extractors and classifier adaptation strategies, and the refinement of the task definition itself. In this paper, we explore the hypothesis that a simple class-covariance-based distance metric, namely the Mahalanobis distance, adopted into a state of the art few-shot learning approach (CNAPS) can, in and of itself, lead to a significant performance improvement. We also discover that it is possible to learn adaptive feature extractors that allow useful estimation of the high dimensional feature covariances required by this metric from surprisingly few samples. The result of our work is a new "Simple CNAPS" architecture which has up to 9.2% fewer trainable parameters than CNAPS and performs up to 6.1% better than state of the art on the standard few-shot image classification benchmark dataset.

Christian Weilbach, Boyan Beronov, William Harvey, Frank Wood. *Structured Conditional Continuous Normalizing Flows for Efficient Amortized Inference in Graphical Models*. AISTATS, 2020.

We exploit minimally faithful inversion of graphical model structures to specify sparse continuous normalizing flows (CNFs) for amortized inference. We find that the sparsity of this factorization can be exploited to reduce the numbers of parameters in the neural network, adaptive integration steps of the flow, and consequently FLOPs at both training and inference time without decreasing performance in comparison to unconstrained flows. By expressing the structure inversion as a compilation pass in a probabilistic programming language, we are able to apply it in a novel way to models as complex as convolutional neural networks. Furthermore, we extend the training objective for CNFs in the context of inference amortization to the symmetric Kullback-Leibler divergence, and demonstrate its theoretical and practical advantages.

S. Khandelwal, R. Goyal, L. Sigal. *UniT: Unified Knowledge Transfer for Any-shot Object Detection and Segmentation*. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021.

Methods for object detection and segmentation rely on large scale instance-level annotations for training, which are difficult and time-consuming to collect. Efforts to alleviate this look at varying degrees and quality of supervision. Weakly-supervised approaches draw on image-level labels to build detectors/segmentors, while zero/few-shot methods assume abundant instance-level data for a set of base classes, and none to a few examples for novel classes. This taxonomy has largely siloed algorithmic designs. In this work, we aim to bridge this divide by proposing an intuitive and unified semi-supervised model that is applicable to a range of supervision: from zero to a few instance-level samples per novel class. For base classes, our model learns a mapping from weakly supervised to fully supervised detectors/segmentors. By learning and leveraging visual and lingual similarities between the novel and base classes, we transfer those mappings to obtain detectors/segmentors for novel classes; refining them with a few novel class instance-level annotated samples, if available. The overall model is end-to-end trainable and

highly flexible. Through extensive experiments on MS-COCO and Pascal VOC benchmark datasets we show improved performance in a variety of settings.

Appendix B. Meetings, Presentations, and Personnel Changes

We participated in the following program-level meetings:

- 26-27 August 2019. Program Kickoff Meeting at DARPA. Attended by Jan-Willem van de Meent, Michael Harradon, Joe Gorman, Frank Wood, and Sameer Singh. We presented a kickoff briefing describing our effort.
- 3-5 February 2020. LwLL Hackathon, at JPL, Pasadena, CA. Attended by Jarred Barber and Michael Harradon.
- 24 March 2020. Virtual site visit with Dr. Draper. Presenters included Avi Pfeffer, Jan-Willem van de Meent, Frank Wood, Leon Sigal, Sameer Singh, Deniz Erdogmus, and Jarred Barber.
- 13-14 October 2020. LwLL PI meeting.

In addition, throughout the program, we engaged in regular virtual meetings with the whole PLEDGES team.

8. List of Symbols, Abbreviations, and Acronyms

Acronym	Definition
APG	amortized population Gibbs
API	application programming interface
BN	Bayesian network
BP	belief propagation
CPDs	conditional probability distributions
CNAPS	conditional neural adaptive processes
DBN	dynamic Bayesian network
ELBO	evidence lower bound
EM	expectation maximization
FILM	feature-wise linear modulation
GAN	generative adversarial network
GMM	Gaussian mixture model
I3D	inflated 3D
JPL	Jet Propulsion Laboratory
KL	Kullback-Leibler
LP	label propagation
MNIST	Modified National Institute of Standards and Technology
NLP	natural language processing
NMT	neural machine translation
PGM	probabilistic graphical model
PLEDGES	Probabilistic Label-Efficient Deep Generative Structures
RNN	recurrent neural network
RPN	region proposal network
SBP	separable belief propagation
SDPM	structured deep probabilistic models
SGD	stochastic gradient descent
SMC	sequential Monte Carlo
SVM	support-vector machine
TSM	Temporal Shift Modules
UBC	University of British Columbia
UCI	University of California, Irvine
VAEs	Variational autoencoders
VE	variable elimination
VIB	variation-informational-bottleneck
X3D	expand 3D