Summer 8-9-2022

Towards General AI using Continual, Active Learning in Large and Few Shot Domains

Jaya Krishna Mandivarapu

Towards General AI using Continual, Active Learning in Large and Few Shot Domains

by

Jaya Krishna Mandivarapu

Under the Direction of Rolando Estrada, Ph.D.

A Dissertation Submitted in Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2022

ABSTRACT

Lifelong learning a.k.a Continual Learning is an advanced machine learning paradigm in which a system learns continuously, assembling the knowledge of prior skills in the process. The system becomes more proficient at acquiring new skill using its accumulated knowledge. This type of learning is one of the hallmarks of human intelligence. However, in the prevailing machine learning paradigm, each task is learned in isolation: given a dataset for a task, the system tries to find a machine learning model which performs well on the given dataset. Isolated learning paradigm has led to deep neural networks achieving the state-of-the-art performance on a wide variety of individual tasks. Although isolated learning has achieved much success in a number of applications, it has wide range of struggles while learning multiple tasks in sequence. When trained on a new task using the isolated network performing well on prior task, standard neural network forget most of the information related to previous task by overwriting the old parameters for learning the new task at hand, a phenomenon often referred to as "catastrophic forgetting". In comparison, humans can learn effectively new task without forgetting the old task and we can learn the new task quickly because we have gained so much knowledge in the past, which allows us to learn the new task with little data and lesser effort. This enables us to learn more and more continually in a self-motivated manner. We can also adapt our previous knowledge to solve unfamiliar problems, an ability beyond current machine learning systems.

INDEX WORDS: Continual learning, Active learning, Deep learning, Meta-learning

Towards General AI using Continual, Active Learning in Large and Few Shot Domains

by

Jaya Krishna Mandivarapu

Committee Chair: Rolando Estrada

Committee: Rajshekhar Sunderraman

Xiaojun Cao

Yanqing Zhang

Ying Zhu, Member

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

August 2022

DEDICATION

I dedicate this work God, Mother, Father and Brother. I'm dedicating it to my mom as I want something that stays longer than me to show that there was once a women named Lakshmi in Bobbili, India, who loved her son very much, set me on right path even through her difficulties and taught me "No matter what happens in life be good to people. Being good to people is a wonderful legacy to leave behind but don't waste time to prove it to others".

I also dedicate this book to my Wife who somehow managed to be nothing but supportive. Who think the same thoughts without need of speech and chatter the same speech without need of the meaning. To my wife, for all you have done I know it wasn't always fun.

ACKNOWLEDGEMENTS

A significant credit for my Ph.D. goes to my advisor Dr. Rolando Estrada who was always there for when you need it as a mentor, as a good human being and made the whole Ph.D experience as gratifying journey. I cannot miss my Blake Camp who is my close friend, researcher partner who provided me with the excellent opportunity to work along with him and have intense intellectual discussions ranging boyish to latest AI related for the last 5 years. I also want to specially thank Dr. Raj Sunderraman who always helped me whenever needed and initially gave excellent opportunity to learn in their lab(CDC Research) as a research assistant during my master's which led to foundations of what I am today. It was truly an exciting experience where I learned a lot of valuable things. I would also like to acknowledge the department of computer science for providing me this opportunity to pursue PhD. I am also truly honoured to be working along side of my fantastic colleagues.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1| INTRODUCTION

Lifelong learning a.k.a Continual Learning is an advanced machine learning paradigm in which a system learns continuously, assembling the knowledge of prior skills in the process. The system becomes more proficient at acquiring new skill using its accumulated knowledge. This type of learning is one of the hallmarks of human intelligence. However, in the prevailing machine learning paradigm, each task is learned in isolation: given a dataset for a task, the system tries to find a machine learning model which performs well on the given dataset. Isolated learning paradigm has led to deep neural networks achieving the state-of-the-art performance on a wide variety of individual tasks. Although isolated learning has achieved much success in a number of applications, it has wide range of struggles while learning multiple tasks in sequence. When trained on a new task using the isolated network performing well on prior task, standard neural network forget most of the information related to previous task by overwriting the old parameters for learning the new task at hand, a phenomenon often referred to as "catastrophic forgetting". In comparison, humans can learn effectively new task without forgetting the old task and we can learn the new task quickly because we have gained so much knowledge in the past, which allows us to learn the new task with little data and lesser effort. This enables us to learn more and more continually in a self-motivated manner. We can also adapt our previous knowledge to solve unfamiliar problems, an ability

beyond current machine learning systems.

1.1 Motivation

Many modern machine learning algorithms face catastrophic forgetting because of the way in which they are trained. Current deep learning models are trained end to end in which all the network parameters are adjusted for decrease the loss and increasing the performance on the current task. All the loss is adjusted using some optimization algorithms such as standard SGD ..etc. While this kind of training is proven to be successful for gaining or performing well on individual task. But as the same network is trained on new task and trained in the same way in which all the network parameters are tuned. But this will lead to degrading the performance on the old task and this degradation is oftern termed as " catastrophic forgetting". For example , if a network with parameters as θ is trained on a Task A (classification or regression) after which the model parameters reach a new state θ_1, then the Task B is learning using this new parameters and the latter training on TASK B will modify the weights learned for TASK A as after training on TASK B will change the parameters to new state from θ_1 to θ_2 , thus likely reducing the network's performance on this task.

Although deep neural networks (DNNs) demonstrated state of the art (SOTA) accuracy on several supervised learning tasks such as as classification (He et al., 2016; Krizhevsky et al., 2012), object detection (Redmon et al., 2016; Ren et al., 2015), and semantic segmentation. But most of the deep neural networks (DNNs) require large set of labeled data to achieve this feet. The challenges of labeling huge datasets in real world setting are many: expensive, limited time available by domain business experts, long labeling time per for

large-scale sample such as videos and time-series data, financial constraints, or to minimize the model's carbon footprint. These all drawback does inherit the application of deep neural networks (DNNs) to more research areas and more organization.

In order to overcome the above drawbacks, Active Learning(AL) system try to select to most informative samples from the pool of unlabeled data points at each stage and send them for annotation to maximize the accuracy of the model. Active learning uses a fixed budget at each stage of learning to select and label a subset of a data points from the unlabeled pool where budget(b) refers to cost associated with annotation by oracle($\mathcal{O}$). The model will be trained on the current labeled pool along with the newly annotated data points. At the end of active learning process model's performance would be nearly the same accuracy as model by utilized fraction of data when compared to the model trained on all the data. Active Learning(AL) also highlights the fact that there exists a non-linear relationship between the model's performance and the amount of training data used. There exists most representative subset of the unlabeled data and selecting those data points to label will provide most of the information needed to learn to solve a task. In this case, we can achieve nearly the same performance by selecting that representative subset for annotation (and training on) only using data points from that representative subset samples, rather than the entire dataset.

1.2 Publications

1. Self-Net: Lifelong Learning via Continual Self-Modeling

2. Continual Learning using Deep Artificial Neurons

3. Deep Active learning using Open set recognition

4. Deep Active Learning using BarlowTwins

5. Efficient Document Image Classification Using Region-Based Graph Neural Network

6. Cross Domain Few-Shot Learning for Document Intelligence.

2| PROBLEM BACKGROUND

Lifelong learning or Continual Learning is an advanced machine learning paradigm in which a system learns continuously, assembling the knowledge of prior skills in the process. The system becomes more proficient at acquiring new skill using its accumulated knowledge. This type of learning is one of the hallmarks of human intelligence. However, in the prevailing machine learning paradigm, each task is learned in isolation: given a dataset, the system tries to find a machine learning model which performs well on the given dataset. Although isolated learning has achieved much success in a number of applications, it requires a large number of training samples to achieve good performance on a given dataset and is only suitable for well-defined tasks. In comparison, humans can learn effectively with a few examples because we have gained so much knowledge in the past, which allows us to learn with little data or effort. This enables us to learn more and more continually in a self-motivated manner. We can also adapt our previous knowledge to solve unfamiliar problems, an ability beyond current machine learning systems. While several approaches have recently emerged, which address this problem indifferent capacities, they are generally marred by either insufficient scalability or inflexibility. In this paper, we present a scalable approach to multi-context continual learning (MCCL) via lifelong skill encoding which greatly improves training efficiency, storage scalability, and which may offer new insights

into knowledge conceptualization. Motivated by the biological process of hippocampus responsible for consolidating knowledge and compressing experiences for long term storage, we show that it is highly beneficial to compress entire networks in sequential fashion. We equate trained networks to learned skills, and to the best of our knowledge, we are the first to demonstrate the efficacy of using Auto-encoders, and their variants, to encode learned skills in order to mimic some of the fundamental memory encoding functions of the hippo-campus and to facilitate efficient continual learning.

3| LITERATURE REVIEW

3.1 Continual Learning Scenarios :

Continual learning is a scenario in which sequence of tasks need to learned by neural network following any of training protocol. But Continual learning can be subdivided into three scenarios based on availability of task id during the phase of the evaluation or testing time. Consider a standard neural network provided with stream of sequential tasks T1 to tn and the tasks are assumed to be clearly separated. During the evaluation time the network is required to inference the probabilities for the task provided. This scenario makes either the evaluation at test time difficult or not for the model as if the task identity is also provided for the network along with the test data it makes easy for the model just to make the prediction, this case is called s task-incremental learning (Task-IL). The subcase of task identity not being provided during the evaluation can be subdivided into two cases 1. In which the model need to self identify the task which is provided and does to inference based on the respective task based parameters, this case is called class-incremental learning (Class-IL) 2. In which model doesn't need to identify the task but just produce the inference results for the task. This type of condition is called "domain-incremental learning (Domain-IL)" (Paper is Three scenarios for continual learning).

The recent works of continual learning can be subdivided into the following methods

1. Regularization based methods 2. Replay based methods 3. Growing network parameters along with the overall–architectural or parameter isolation methods

4| FIRST IDEA: SELF-NET LIFELONG LEARNING VIA CONTINUAL SELF-MODELING

4.1 Problem Formulation

Continual learning problem can be subdivided into different set of related problems such as fixed architecture, no access to prior training data, inference of task id etc. and each needs to be tackled in different way. As mentioned in the Section 3. One such scenario which is handled during our current idea is during training, only data from the current task is available and task id is provided during the inference time. Such that model doesn't need to infer the task during the testing time.

The overall continual learning setup for the current idea can be considered as below **(1)** System learns one new task at a time, **(2)** Each task can be solved independently of other tasks, **(3)** Each task is provided with task label (i.e., the system knows which task to solve at any point), and **(4)** the system has *no access* to old training data. In particular, our problem differs from settings in which a single task grows more difficult over time (e.g., class-incremental learning (CIL)).

One of the primary goals of the continual learning is to avoid catastrophic forgetting in which learning a larger of numbers of tasks $\mathcal{T}_0, \mathcal{T}_1, ..., \mathcal{T}_n$ in a sequential way but in such a way that after sequential learning average performance on all previous tasks learnt until

that time is high. More concretely, each task T_i is specified by a training set, $D_i = \{X_i, Y_i\}$, consisting of n_i different $\{x, y\}$ training pairs. However in our case of continual learning that the system has access only to the data D_i for the current task only. The system is sequentially trained on each D_i dataset, using either a supervised or reinforcement learning paradigm, as applicable with no access to previous task data. That is, the system is first exposed to D_1 (and thus must learn T_1), then D_2, D_3, up to D_k, where k is the total number of tasks encountered during its lifetime. Note that, in this paradigm, datasets are *not* required to be disjoint, i.e., any two datasets D_i and D_j many contain some common $\{x, y\}$ pairs.

Critically, the system is trained on each D_i only once during its lifetime. The system is not allowed to store any exemplars from previous tasks or revisit old data when training on new tasks. We do, however, allow multiple passes over the data when first learning the task, as is standard in machine learning. We also assume that task labels are known; inferring the desired task from the input data is important but is outside the scope of this paper.

As noted in Sec. 3, there are two common types of solutions for this CL problem. Regularization methods estimate a single set of parameters θ^* for all tasks, while growth-based approaches learn (and store) a new set of weights θ_i for each new task. The former uses constant storage (w.r.t to the number of tasks) but has bad performance, while the latter achieves good performance but is asymptotically equivalent to storing independent networks. Below, we detail our proposed approach, which has nearly the same performance as growth-based methods, but uses significantly less storage.

4.2 Methodology

Figure 9.1 provides a high-level overview of our proposed approach. The proposed system is named as **Self-Net** as the name suggests the proposed system self models the independent Task network (TN) for each task and single autoencoder (AE) for all tasks to solve the problem of catasrophic forgetting in continual learning. At any given time step after solving the first task the proposed system utilizes a m-dimensional Buffer for storing newly learned tasks, an $O(n)$ lifelong autoencoder (AE) for storing older tasks (at the end of the lifecycle of learning tasks only decoder would be sufficient) and single s-dimensional latent vector for each task. This s-dimensional latent vector for each task is saved, where $s << n$ which means that the size of the latent vector far smaller than the size of the task network which was used to solve the current task. Assuming that c and m are constants, our space complexity is $O(n + ks)$, where k is the number of learned tasks. In particular, the proposed approach achieves asymptotic space savings compared to storing kn independent networks if s is sub-linear w.r.t. n, (i.e., $s = \omega(n)$ in asymptotic notation).

One of the main advantage of the proposed approach is that each task network (TN) is just an independent and standard neural network, which can learn regression, classification, or reinforcement learning (RL) tasks (or some combination of the three as shown in the experimental section). For ease of discussion, we will focus on the case where there is a single TN and the Buffer can hold only one network; this can be easily extended to multiple networks. The AE is made up of an *encoder* that compresses an input vector into a lower-dimensional, latent vector e and a *decoder* that maps e back to the higher-dimensional space. Our system can produce high-fidelity recollections of the learned weights, despite this

intermediate compression. In our experiments, we used a contractive autoencoder (CAE) due to its ability to quickly incorporate new values into its latent space.

In CL, we must learn k different tasks sequentially. To learn these tasks independently, one would need to train and save k networks, with $O(n)$ parameters each, for a total of $O(kn)$ space. In contrast, we propose using our AE to encode each of these k networks as an s-dimensional latent vector, with $s << n$. Thus, our method uses only $O(n+ks)$ space, where the $O(n)$ term accounts for the TNs and the fixed-size Buffer. Despite this compression, our experiments show that we can obtain a high-quality approximation of previously learned weights, even when the number of tasks exceeds the number of parameters in the AE. Below, we first describe how to encode a single task-network before discussing how to encode multiple tasks in continual fashion.

4.2.1 *Single-network encoding*

For a simple explanation encoding of one skill is explained in this setion and can be easily extended to encoding multiple skills. Let t be a task (e.g., classifying digits) and let f(θ) be the network used to solve that task in which once flattened then θ is a $O(n)$-dimensional vector of parameters of a network trained to solve t. That is, using a task-network with parameters θ, we can achieve performance p on t (e.g., a classification accuracy of 95%). After flattening the Now, let $\hat{\theta}$ be the approximate reconstruction of θ by our autoencoder is training with an objective of reconstructing the $O(n)$-dimensional vector. Let $\hat{p}$ be the performance that we obtain by using these reconstructed weights for task t. Our goal is to minimize any performance loss w.r.t. the original weights. If the performance of the

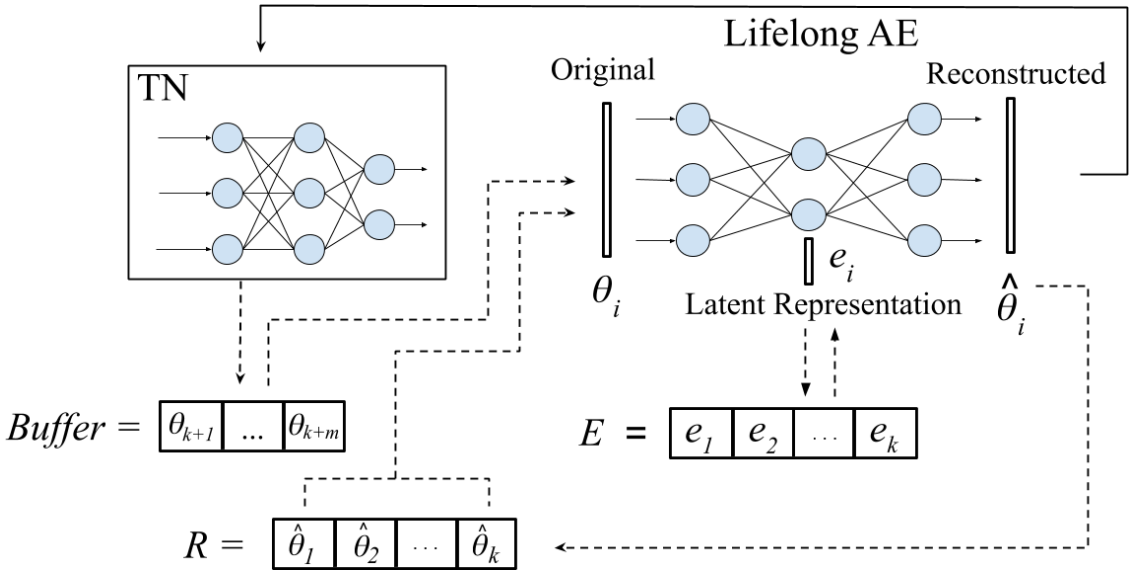


Figure 4.1: **Framework overview:** Our proposed system has a set of reusable *task-specific networks* (TN), a *Buffer* for storing the latest m tasks, and a lifelong, *auto-encoder* (AE) for long-term storage. Given new tasks $\{t_{k+1}, ..., t_{k+m}\}$, where k is the number of tasks previously encountered, we first train m task-networks independently to learn $\{\theta_{k+1}, ..., \theta_{k+m}\}$ optimal parameters for these tasks. These networks are temporarily stored in the Buffer. When the Buffer fills up, we incorporate the new networks into our long-term representation by retraining the AE on both its approximations of previously learned networks and the new batch of networks. When an old network is needed (e.g., when a task is revisited), we reconstruct its weights and load them onto the corresponding TN (solid arrow). Even when the latent representation e_i is asymptotically smaller than θ_i, the reconstructed network closely approximates the performance of the original.

reconstructed weights is acceptable, then we can simply store the $O(s)$ latent vector e, instead of the $O(n)$ original vector θ. If we had access to the test data for t, we could assess this difference in performance directly and train our AE until we achieved an acceptable margin ϵ:

$$p - \hat{p} \leq \epsilon. \tag{4.1}$$

For example, for a classification task we could stop training our AE if the drop in accuracy is less than 1%.

In a continual learning setting, though, the above scheme requires storing validation data for each old task. Instead, we measure a distance between the original and reconstructed weights and stop training when we achieve a suitably close approximation. Empirically, we determined that the cosine similarity,

$$\cos(\theta, \hat{\theta}) = \frac{\theta \cdot \hat{\theta}}{\|\theta\|\|\hat{\theta}\|} = \frac{\sum_{i=1}^{n} \theta_i \hat{\theta}_i}{\sqrt{\sum_{i=1}^{n} \theta_i^2} \sqrt{\sum_{i=1}^{n} \hat{\theta}_i^2}}, \tag{4.2}$$

is an excellent proxy for a network's performance. Unlike the mean-squared error, this distance metric is scale-invariant, so it is equally suitable for weights of different scales, which may be the case for separate networks trained on distinct tasks. As detailed in Section 7.4, cosine similarity close to 0.99 yielded excellent performance for a wide variety of tasks and architectures.

4.2.2 *Continual encoding*

In this section We will now detail now to use our propsed system Self-Net to encode a sequence of trained networks in a continual fashion by overcoming the catastrophic forgetting. Let m be the size of the Buffer, and let k be the number of tasks which have been previously encountered. As noted above, we train each of these m task-networks using conventional backpropagation, one per task. Now, assume that our AE has already learned to encode the first k task-networks. We will now show how to encode the most recent batch of m task-networks corresponding to tasks $\{t_{k+1}, ..., t_{k+m}\}$ into compressed representations $\{e_{k+1}, ..., e_{k+m}\}$ while still remembering all previously trained networks.

Let E be the set of latent vectors for the first k networks. In order to integrate m

new networks $\{\theta_{k+1}, ..., \theta_{k+m}\}$ into the latent space, we first recollect all previously trained networks by feeding each $e \in E$ as input to the decoder of the AE. We thus generate a set R of recollections, or approximations, of the original networks (see Fig. 9.1). We then append each θ_i in the Buffer to R and retrain the AE on all $k+m$ networks until it can reconstruct them, i.e., until the average of their respective cosine similarities is above the predefined threshold. Algorithm 3 summarizes our CL strategy.

As our experiments show, our compressed representations achieve excellent performance compared to the original parameters. Since each $\hat{\theta} \in R$ is simply a vector of network parameters, it can easily be loaded back onto a task-network with the correct architecture. We can thus discard the original networks and store k networks using only $O(n+ks)$ space. In addition, our framework can encode many different types and sizes of networks in a continual fashion. In particular, we can encode a network of arbitrary size q using a constant-size AE (that takes inputs of size n) by splitting the input network into r subvectors[1], such that $(n = q/r)$. As we verify in Section 7.4, we can effectively reconstruct a large network from its subvectors and still achieve a suitable performance threshold.

As Fig. 4.8 illustrates, we empirically found a strong correlation between a reconstructed network's performance and its cosine similarity w.r.t. to the original network. Intuitively, this implies that vectors of network parameters that have a cosine similarity approaching 1 will exhibit near-identical performance on the underlying task. Thus, the cosine similarity can be used as a terminating condition during retraining of the AE. In practice, we found a threshold of .997 to be sufficient for most experiments.

[1]We pad with zeros whenever q and n are not multiples of each other.

4.3 Task network fine-tuning

As an additional optimization, one can improve the speed with which the AE learns a new task by encouraging the parameters of new task-networks to be are as similar as possible to previously learned ones. This can be accomplished by fine-tuning all networks from a common source and penalizing large deviations from this initial configuration with a regularization term. Note that training new task networks in this manner differs from standard regularization methods (e.g., EWC) because the weights learned for older tasks are not modified (and hence their performance does not degrade).

Formally, let θ^* be the source parameters, ideally optimized for some highly-related task. Without loss of generality, we can define the loss function of task-network θ_i for task t_i as:

$$TaskNetLoss_i = TaskLoss + \lambda MSE(\theta^*, \theta_i) \tag{4.3}$$

where λ is a regularization coefficient that determines the importance of remaining close to the source parameters vs. optimizing for the current task. By encouraging the parameters for all task-networks to remain close to one another, we make it easier for the AE to learn a low-dimensional representation of the original space. We employ this scheme for the experiments section with $\lambda = 0.001$.

4.4 Results

We carried out a range of CL experiments on a variety of datasets, in both supervised and reinforcement-learning (RL) settings. First, we performed a robustness analysis in order to empirically establish how precise an approximation of a network must be in order to retain comparable performance on a task. Then, we analyzed our system's ability to encode a very large number of tasks, thus validating that the AE does simply memorize the TNs. We then evaluated the performance of our approach on the following CL datasets: Permuted MNIST (Kirkpatrick et al., 2017), Split MNIST (Nguyen et al., 2018), Split CIFAR-10 (Zenke et al., 2017), Split CIFAR-100 (Zenke et al., 2017), and successive Atari games (Mnih et al., 2013) (we describe each dataset below). Finally, we also analyzed our system's performance when using **(2)** different sizes of AEs, and **(3)** different TN architectures.

4.4.1 *Robustness analysis*

In our initial experiments, we added different levels of i.i.d, zero-mean Gaussian noise to the weights of a trained network. Our goal was twofold: **(1)** to verify that approximate weights can differ from their original values while still retaining good performance and **(2)** to establish a threshold at which to stop training our AE. Since we assume no access to data from previously learned tasks, we need a way to estimate the performance of a reconstructed network without testing on a validation set.

Figure 4.8 shows performance as a function of deviations from the original parameters as measured by cosine similarity, for three datasets (described below). Under this metric, there is a clear correlation between the amount of parameter dissimilarity and the probability of a

decrease in performance. The red line indicates a cosine similarity of 0.997. Weights above this value had nearly identical performance to the original values. Thus, unless otherwise noted, we used this threshold as a terminating condition in our subsequent experiments.

4.4.2 *Performance and storage scalability*

In the next set of experiments, we verified that our method retains excellent performance even when the number of TN parameters exceeds the number of parameters in the AE. In other words, here we confirmed that our AE is *compressing previously learned weights*, not simply memorizing them. More generally, there is a trade-off in CL between storage and performance. Using different networks for k tasks yields optimal performance but uses $O(kn)$ space, while regularized methods such as Online EWC (Huszár, 2018) only require $O(n)$ space but suffer a steep drop in performance as the number of tasks grows. For any method, we can quantify performance as a *compression factor*, i.e., the number of additional parameters it stores per task; in our case, our compression factor is k/s because we store an s-dimensional vector per task.

Here, our experimental paradigm was as described in Sec. 4.2.2: we first trained the TN on m tasks independently, storing each set of learned weights in the Buffer. Once the Buffer became full, we trained the AE to encode these weights into its latent space, only storing the latent vectors after training. We then continued to train the TN on new batches of m tasks (saving the new weights to the Buffer). Every time the Buffer became full, we trained the AE on *all tasks*, using the stored latent vectors and the new m weights. After the initial batch, we fine-tuned all networks from the mean of the initial set of m networks and penalized deviations from this source vector (using $\lambda = 0.001$), as described in Section 7.3.

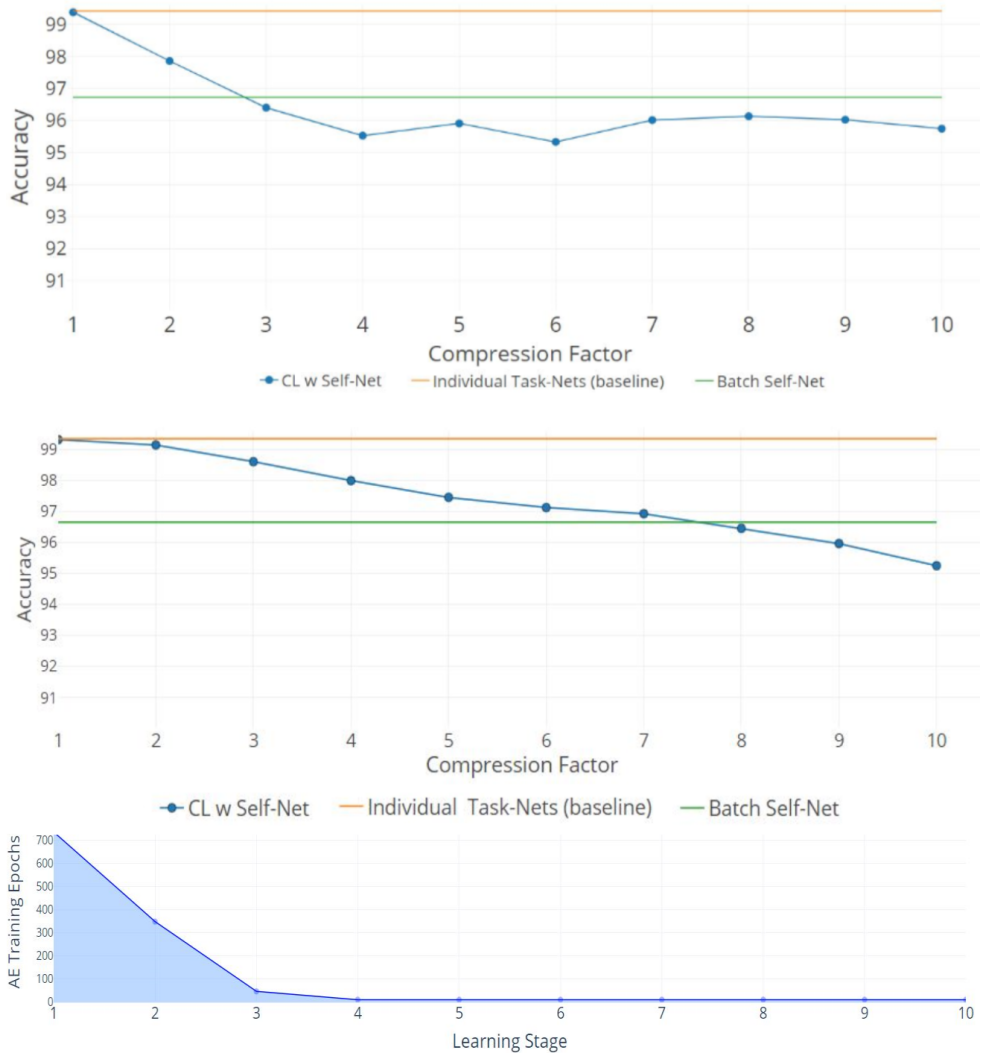


Figure 4.2: **10X Compression for Split-MNIST:** Orange lines denote the average accuracy achieved by individual networks, one per task. Green lines denote the average accuracy when training the AE to encode all networks as a single batch. Blue lines indicate the average accuracy obtained by Self-Net at each CL Stage. **Top:** 50 tasks with latent vectors of size 5 and a Buffer of size 5. **Middle:** 100 tasks with latent vectors of size 10 and Buffer of size 10. The x-axis (top and middle) denotes the compression factor achieved at each learning stage. **Bottom:** the training epochs required by the 5-dimensional AE to incorporate new networks decreases rapidly over time.

For these experiments, we used the Split MNIST dataset (Nguyen et al., 2018), which consists of different binary subsets of the MNIST dataset (LeCun et al., 1998), drawn randomly. In other words, tasks were defined by tuples comprised of the positive and negative digit class(es), e.g., ([pos={1}, neg={6,7,8,9}], [pos={6}, neg={1,2,3,4}], etc.). Here, the training and test sets consisted of approximately 40% positive examples and 60% negative examples. For this experiment, we trained a deep convolutional task network with 2 convolution layers (kernels of size 5x5 and stride 1x1), 1 hidden layer (320x50), and 1 output layer (50x10)—21,840 parameters in total. Our task network used ReLU activation units. Our AE, on the other hand, had one fully connected hidden layer with either 5 or 10 units. We used a Buffer of the same size as the latent vector, i.e., either 5 or 10. These values were chosen so that each new batch of networks yielded an integer compression factor, e.g., encoding 15 networks with a latent vector of size 5 gives 3X compression ($k/s = 3$). We used decreasing thresholds to stop training our AE: 0.9996 for the initial batch, 0.987 for the second batch, and 0.986 for subsequent batches.

The top two plots of Fig. 4.2 show the mean performance for up to 50 and 100 Split-MNIST tasks, given latent vectors of size 5 and 10, resp. All figures show the average accuracy across all tasks learned up to that point. For comparison, we also plotted the original networks' performance and the performance of the reconstructions when the AE learned all the tasks in a single batch (green and orange lines, resp.). The line with dots represents the CL system; each dot indicates the point where the AE had to encode a new set of m networks. For 10X compression, the Self-Net with a latent vector of size 5 retained ∼95.7% average performance across 50 Split-MNIST tasks, while the Self-Net with 10-dimensional latent vectors retained ∼95.2% across 100 tasks. This represents a relative

change of only ∼3.3% compared to the original performance of ∼99%. In other words, our approach is able to compress 21,840 parameters into 5 or 10 values with little performance loss, even when trained in a continual fashion. In contrast, existing methods dropped to ∼50% performance after learning only 10 tasks on this dataset (see Fig. 4.3 below). Finally, we note that by initializing each new network from the mean of the initial batch, our AE was able to incorporate subsequent networks with very little additional training (see stages 4-10 in bottom image of Fig. 4.2).

4.4.3 *Permuted MNIST*

In the next set of experiments, we compared our approach to state-of-the-art methods across multiple datasets. First, we trained convolutional feed-forward neural networks with 21,840 parameters on successive tasks, each defined by distinct permutations of the MNIST dataset (LeCun et al., 1998), for 10-digit classification. We used networks with 2 convolution layers (kernels of size 5x5, and stride 1x1), 1 hidden layer (320x50), and 1 output layer (50x10). Our AE had three, fully connected layers with 21,840, 2000, and 20 parameters, resp. Thus, our latent vectors were of size 20. For this experiment, we used a Buffer of size 1. Each task network was encoded by our AE in sequential fashion, and the accuracies of all reconstructed networks were examined at the end of each learning stage (i.e., after learning a new task). Figure 4.3 (top) shows the mean performance after each stage for all tasks learned up to that point. Our technique almost perfectly matched the performances achieved by independently trained networks, and it dramatically outperformed other state-of-the-art approaches including EWC (Kirkpatrick et al., 2017), Online EWC (the correction to EWC proposed in (Huszár, 2018)), and Progress & Compress (Schwarz et al., 2018). As a base-

line, we also show the results for SGD (no regularization), L2-based regularization in which we compare new weights to all the previous weights, and Online L2, which only measures deviations from the weights learned in the previous iteration. Our technique remember old tasks without inhibiting new learning.

4.4.4 *Split MNIST*

We the compared our method to the same set of prior approaches on the Split MNIST (described above). Our task-networks, CAE, and Buffer size were the same as for Permuted MNIST (except that the outputs of the task-networks were binary, instead of 10 classes). In this domain, too, our technique dramatically outperformed competing approaches, as seen in Figure 4.3 (middle).

4.4.5 *Split CIFAR-10*

We then verified that our proposed approach could reconstruct larger, more sophisticated networks. Similar to the Split MNIST experiments above, we divided the CIFAR-10 dataset (Krizhevsky, 2009a) into multiple training and test sets, yielding 10 binary classification tasks (one per class). We then trained a task-specific network on each class. Here, we used TNs having an architecture which consisted of 2 convolutional layers, followed by 3 fully connected hidden layers, and a final layer having 2 output units. In all, these task networks consisted of more than 60K parameters. Again, for this experiment we used a Buffer of size 1. Our AE had three, fully connected layers with 20442, 1000, and 50 parameters, resp. As described in Sec. 7.3, we split the 60K networks into three subvectors to encode them with our autoencoder; by splitting a larger input vector into smaller subvectors, we can encode

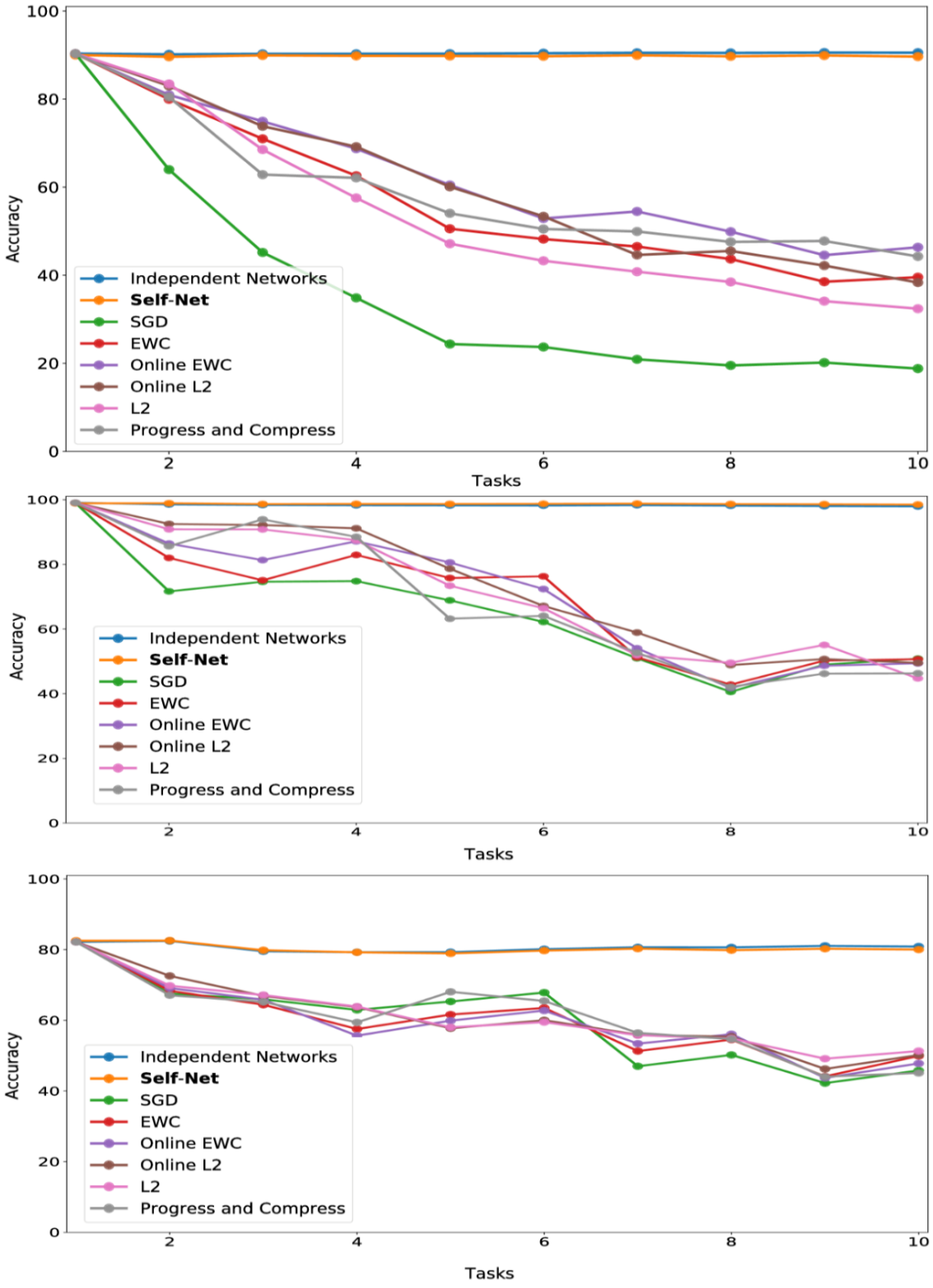


Figure 4.3: CL performance comparisons with average test set accuracy on all observed tasks at each stage for **(top)** Permuted MNIST, **(middle)** Split MNIST, and **(bottom)** Split CIFAR-10.

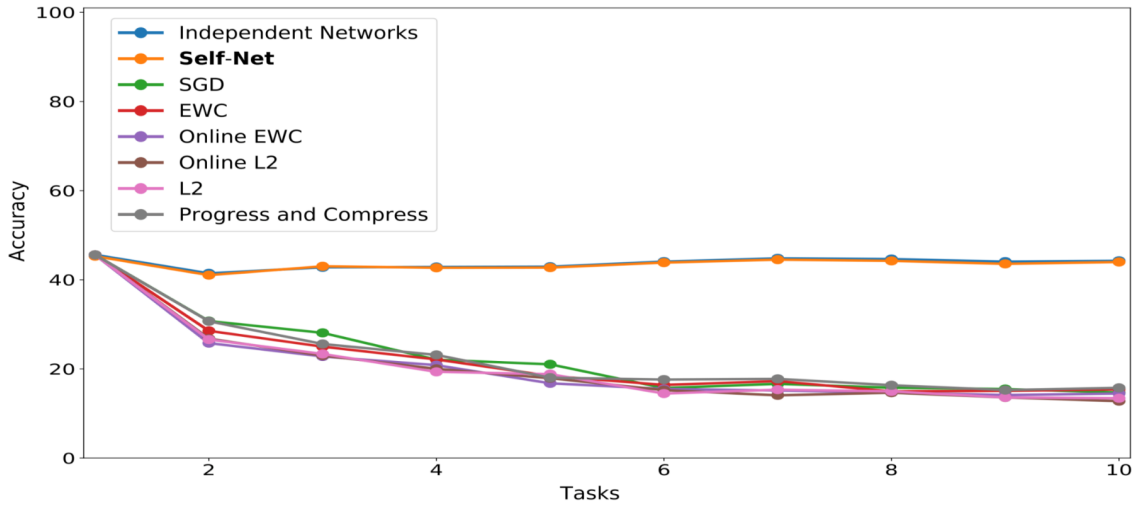


Figure 4.4: CL performance comparisons with average test set accuracy on all observed tasks at each stage for CIFAR-100.

networks of arbitrary sizes. The individual task-networks achieved accuracies ranging from 78% to 84%, and a mean accuracy of approximate 81%. Importantly, we encoded these larger networks using almost the same AE architecture as the one used in the MNIST experiments. As seen in Figure 4.3 (bottom), the accuracies of the reconstructed CIFAR networks also nearly matched the performances of their original counterparts, while also outperforming all other techniques.

4.4.6 *Split CIFAR-100*

We applied a similar approach for the CIFAR-100 dataset (Krizhevsky, 2009a). That is, we split the dataset into 10 distinct batches comprised of 10 classes of images each. We used the same task-network architecture and Buffer size as in our CIFAR-10 experiments, modified slightly to accommodate a 10-class classification objective. The trained networks achieved accuracies ranging from 46% to 49%. We then encoded these networks using the same AE architecture described in the previous experiments, again accounting for the input

size discrepancy by splitting the task-networks into smaller subvectors. As seen in Figure 4.4, our technique almost perfectly matched the performances achieved by independently trained networks.

4.4.7 *Incremental Atari*

To evaluate the CL performance of Self-Net in the challenging context of reinforcement learning, we used the code available at (Greydanus, 2017) to implement a modified Async Advantage Actor-Critic (A3C) framework; this architecture, originally introduced in (Mnih et al., 2016), can learn successive Atari games while retaining good performance across all games. The model we used had 4 convolutional layers (kernals of size 3x3, and strides of size 2x2), a GRU layer (800x256), and two ouput layers: an Actor (256xNum_Actions), and Critic (256x1), resulting in a complex model architecture and over 800K parameters. Critically, this entire model can be flattened and encoded by the single AE in our Self-Net framework having three, fully connected layers with 76863, 2000, and 200 parameters, resp. For these experiments we also used a Buffer of size 1.

Similar to previous experiments, we trained our system on successive tasks, specifically the following Atari games: Boxing, Star Gunner, Kangaroo, Pong, and Space Invaders. Figure 4.5 shows the near-perfect retention of performance on each of the 5 games over the lifetime of the system. This was accomplished by training on each game only once, never revisiting the game for training purposes. The dashed, vertical lines demarcate the different stages of continual learning. That is, each stage indicates that a new network was trained for a new game, over 40M frames. Afterwards, the mean (dashed, horizontal black lines) and standard-deviation (solid, horizontal black lines) of the network's performance were

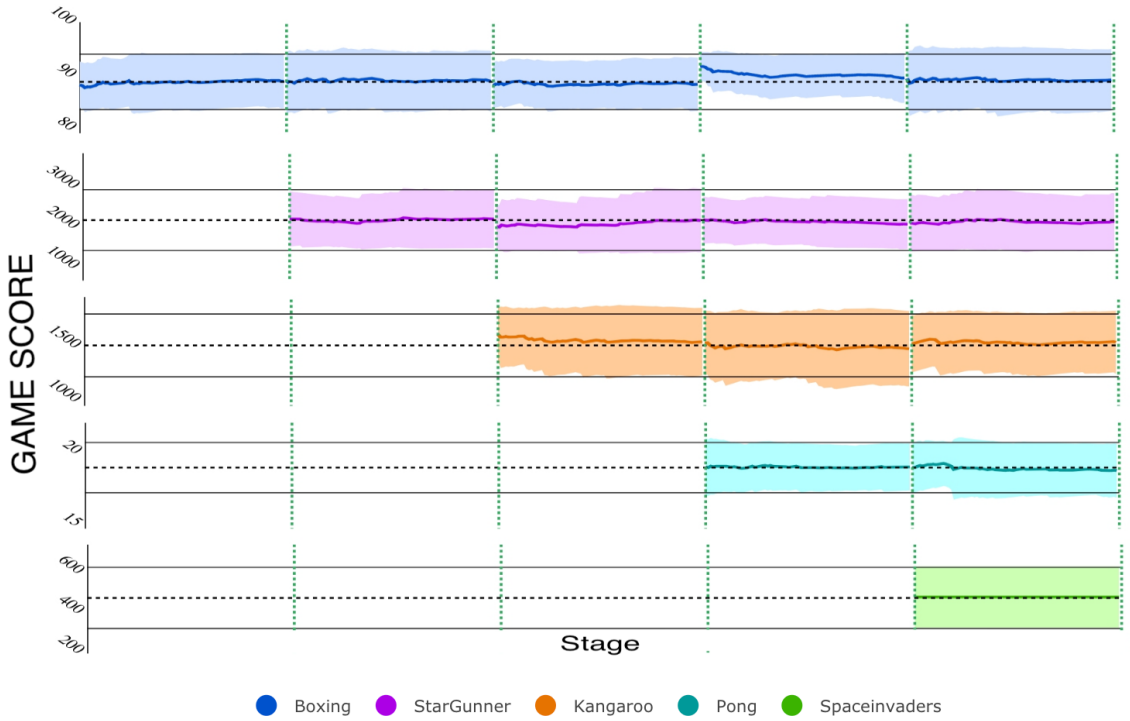


Figure 4.5: **CL on five Atari games with Self-Net:** To evaluate the reconstruction score at each stage, we ran the reconstructed networks for 80 full game episodes. The cumulative mean score is nearly identical to the original TN at each stage.

computed by allowing it to play the game, unrestricted, for 80 episodes. After each stage, the performances of all reconstructed networks were examined by re-playing each game with the appropriate reconstructed network. As Figure 4.5 shows, the cumulative means and SD's of the reconstructed networks closely mimic those achieved by their original counterparts.

4.4.8 *Split networks and multiple architectures*

Finally, we verified that **(1)** a smaller AE can encode multiple network splits in substantially less time than a larger one can learn the entire network and **(2)** that the same AE can be used to encode trained networks of different sizes and architectures. Figure 4.7 (left) shows the respective training rates of an AE with 20,000 input units (blue line)—trained to reconstruct 3 sub-vectors of length 20,000—compared to that of a larger one, with 61,000

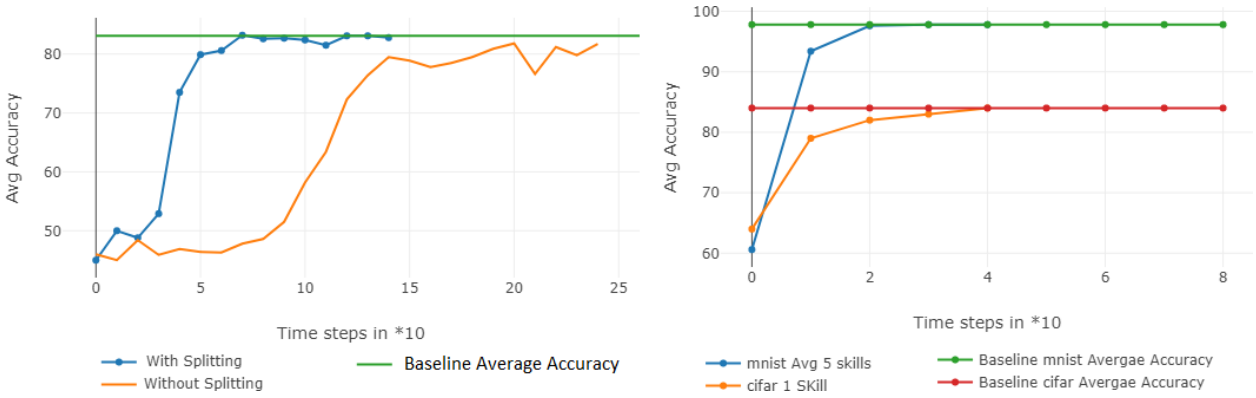


Figure 4.6: **Additional analyses: Left:** the AE training efficiency is improved when large networks are split into smaller subvectors. **Right:** a single AE can encode networks of different architectures and sizes.

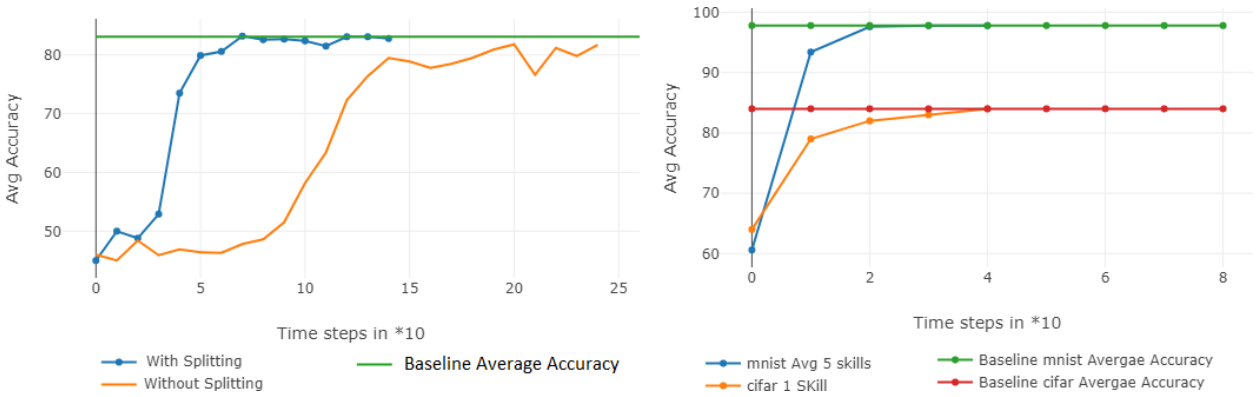


Figure 4.7: **Additional analyses: Left:** the AE training efficiency is improved when large networks are split into smaller subvectors. **Right:** a single AE can encode networks of different architectures and sizes.

input units (yellow line), trained on a single 60K CIFAR-10 network. Clearly, using more inputs for a smaller AE enables us to more quickly encode larger networks. Finally, Figure 4.7 (right) shows that the same AE can simultaneously reconstruct 5 MNIST networks and 1 CIFAR network so that all networks approach their original accuracies.

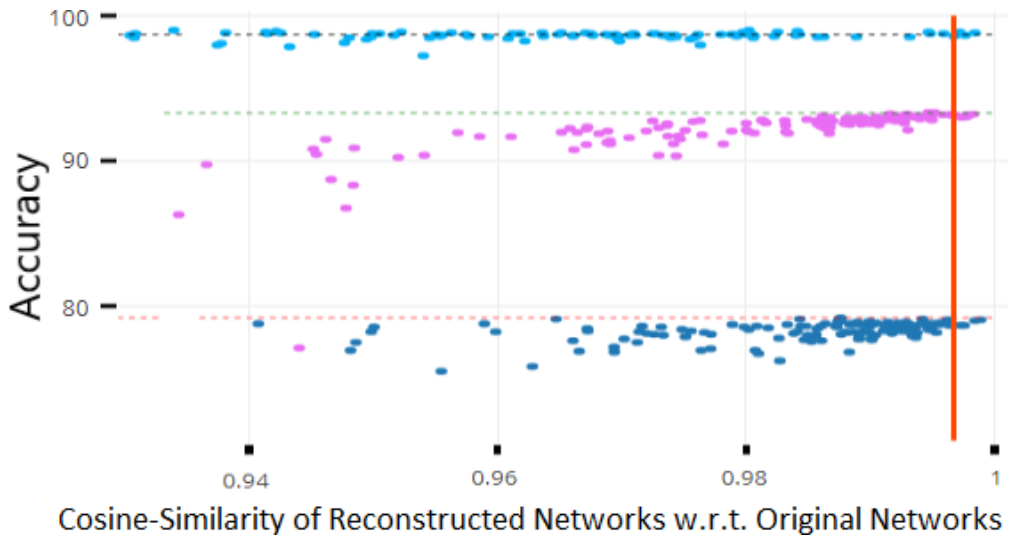


Figure 4.8: **Robustness analysis of network performance as a function of cosine similarity:** Each dot represents the accuracy of a reconstructed network and the dotted lines are the baseline performances of the original networks. The above values for three datasets (Permuted MNIST (in pink), MNIST (in cyan), and CIFAR-10 (in blue), show that cosine similarity values above 0.997 guarantee nearly optimal performance for these datasets.

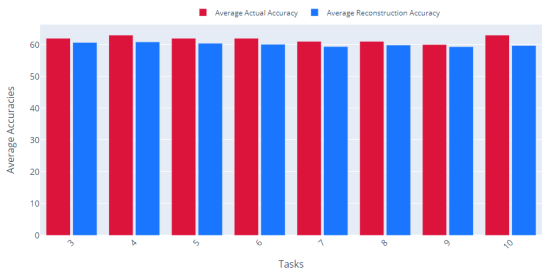


Figure 4.9: **Reconstructing Multiple Networks For The Same Cifar-10 Task**

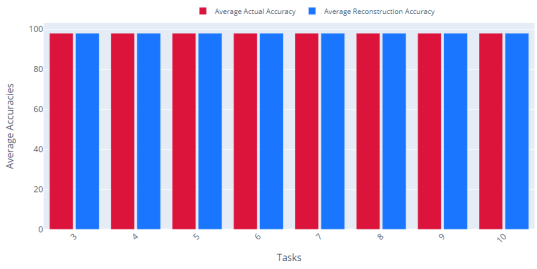


Figure 4.10: **Reconstructing Multiple Networks For The Same MNIST Task**

4.5 Conclusions and future work

In this paper, we introduced a scalable approach for multi-context continual learning that decouples how to learn a set of parameters from how to store them for future use. Our proposed framework uses state-of-the-art autoencoders to facilitate lifelong learning via continual self-modeling. Our empirical results confirm that our method can efficiently acquire and retain large numbers of tasks in continual fashion. In future work, we plan to further improve our autoencoder's capacity and explore how to use the latent space to extrapolate to new tasks using little or no training data. We also intend to compress the latent space even further (e.g., using only $\log(k)$ latent vectors for k tasks). Promising approaches include clustering the latent vectors into sets of related tasks or using sparse latent representations. Finally, we will also investigate how to infer the current task automatically.

5| SECOND IDEA: CONTINUAL LEARNING USING DEEP ARTIFICIAL NEURONS

5.1 Problem Formulation

In this work, we propose to meta-learn a single parameter vector φ, shared by all DANs in the model, which can mitigate catastrophic forgetting in a network that *learns during deployment* by updating its Synapses with standard backpropagation. We call this parameter vector a neuronal phenotype, since it defines the behavior of each DAN in the network, and it is kept fixed during intra-lifetime deployment. Specifically, we consider Continual Learning Trajectories (e.g. $[\mathcal{T}_0, \mathcal{T}_1, ..., \mathcal{T}_k]$) which are comprised of sequences of tasks $\mathcal{T}$. These task sequences are drawn uniformly from some underlying task distribution $p(\mathcal{T})$. We add a single-context stipulation stating that within a sequence of tasks, each data sample $x \in \mathcal{T}_i$ is mapped to one and only one target value y, and each sample x belongs to one and only one task $\mathcal{T}_i$. In our experiments, we assume that tasks are disjoint, and therefore our model must learn one task before moving on to the next, though we anticipate that we can relax this assumption to handle overlapping or evolving task distributions in future work. The model is therefore allowed to perform a fixed number of updates on data from each task, where tasks are encountered one after the other. We seek a model which retains good performance over the whole task trajectory, without being allowed to revisit data from previously encountered

tasks in the sequence.

We formulated an experiment similar to the one proposed in Flennerhag et al. (2020), and originally in Finn et al. (2017a). More specifically, we consider the problem of sequential non-linear regression, wherein a model must try to fit to a complete function, when exposed to data from only part of that function in distinct time-intervals. In other words, it must learn the complete function in a *piece-wise*, or incremental manner, since it cannot revisit data to which it was exposed during previous intervals. As in Flennerhag et al. (2020), we split the input domain $[-5, 5] \subset \mathbb{R}$ into 5 consecutive sub-intervals, which correspond to 5 distinct tasks. Task_1 therefore corresponds to the sub-function falling within $[-5, -3)$; Task_2 corresponds to the sub-function within $[-3, -1)$, and so on. The model is exposed to Tasks 1 through 5 in sequential manner. During each sub-task, the network is exposed to 100 data points, drawn uniformly from the current task window. That is, during Task_1 the model performs 100 updates on data sampled from $[-5, -3)$. In our experiments, we perform 1 update on every sample, equating to a batch size of 1. Sub-tasks are thus defined by their respective windows in the input domain.

We slightly modify the target functions used in Flennerhag et al. (2020). We define a task sequence by a target function that is a mixture of two sine functions with varying amplitudes, phases, and x-offsets. At the beginning of each meta-epoch, we randomly sample two amplitudes $\alpha_{(0,1)} \in (0,2)$, phases $\rho_{(0,1)} \in (0, \pi/3)$, and x-offsets $\phi_{(0,1)} \in [-5, 5]$. Summing two such sine functions yields a target function of the form:

$$y = \alpha_0 sin((\rho_0 x) + \phi_0) + \alpha_1 sin((\rho_1 x) + \phi_1) \tag{5.1}$$

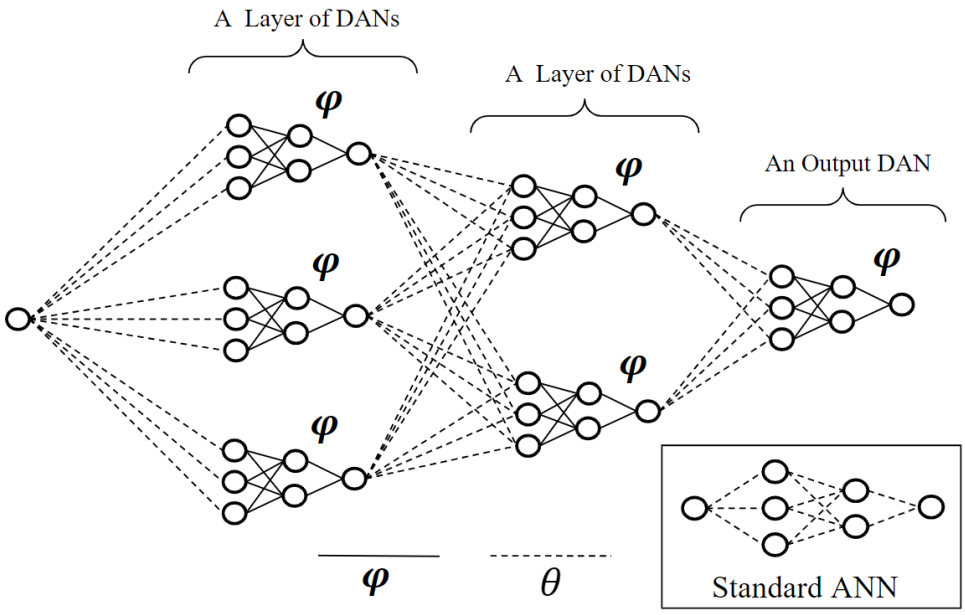


Figure 5.1: A Network of Deep Artificial Neurons (DANs). DANs are connected to one another by parameters θ, which can be regarded as vectorized synapses, or VECs. All DANs share parameters φ, which we dub a neuronal phenotype. The strength of the connection existing between any 2 connected neurons is therefore a function of the state of the n-dimensional synaptic vector.

Afterwards, we discard any target functions from the training distribution meeting the following criteria: $y_{max} > .8$; $y_{min} > -.8$; $y_{max} - y_{min} < .4$. This yields a final input domain of $[-5, 5]$ and output range of $[-.8, .8]$.

5.2 Model Architecture

Deep Artificial Neurons, or DANs, are themselves realized as multi-layer neural networks. For the purposes of demonstration, consider a DAN instantiated as a 2-layer neural network, with a single layer of hidden nodes, and a single output node (i.e. the output activation of the neuron). In practice, we apply a *tanh* non-linear activation to the output of the hidden layer, as well as to the output layer of each DAN. See the bottom of Fig.5.1 for an illustration. Let n_channels denote both (1) the size of the input vector to this network, and (2) as we will

see, the number of connections between pairs of DANs. Conceptually, we can distribute this single DAN amongst all nodes of a traditional neural network. Fig.5.1 offers an illustration of how to convert a standard ANN into a network of DANs with n_channels=3.

More generally, consider the topology of a standard, fully-connected, feed-forward neural network with l layers of nodes, and let n_l denote the number of nodes in layer l. Let l_0 be a special case, denoting the layer of input nodes, which are *not* DANs. We can convert this topology to a network of DANs in the following way. For each layer of nodes, up to but not including the layer of output nodes, we instantiate a layer of Synapses as a standard, fully-connected weight-matrix θ_l with dimensions $n_l \times (n_{l+1} \times n\_channels)$. Synapses connect layers of DANs to one another. Feed-forward propagation of a signal along these connections is therefore facilitated in the standard way, by computing the dot product of the activation vector σ_{l_out} from the previous layer and this layer of Synapses θ_l. This yields a large input vector $\sigma_{(l+1)\_in}$, to be processed by the DANs in the next layer:

$$\sigma_{(l+1)\_in} = \sigma_{l\_out} \cdot \theta_l = \sum_{i}^{n_{l+1}} \theta_i^j \sigma_{l\_out}^j + b_i, \tag{5.2}$$

where j denotes the index of nodes in layer l, and i is the index of nodes in layer $l+1$. Note that in practice we do *not* apply a non-linear activation function to this vector. Rather, n slices of the raw dot product $\sigma_{(l+1)\_in}$ are fed as input to the n DANs in the next layer. That is, since all DANs share parameters φ, the same DAN model processes each slice of $\sigma_{(l+1)\_in}$. Said another way, the input vector to the layer of DANs is sliced into n_{l+1} equally sized sub-vectors, where n is the number of DANs in layer $l+1$. The output vector of a layer of DANs is obtained by passing each of these separate slices through the DAN, and

concatenating the resulting activations. This can be done very efficiently by simply reshaping the inbound synaptic activations $\sigma_{(l+1)\_in}$ and using our single DAN phenotype to process all slices as a batch. This results results in $\sigma_{(l+1)\_out}$ which constitute the output activations of the DANs in layer $l+1$.

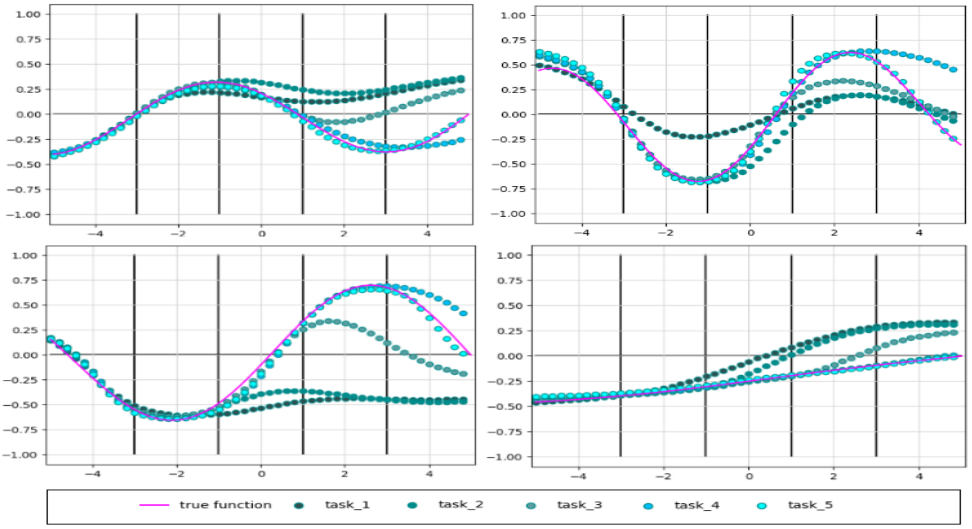


Figure 5.2: Continual Learning during Deployment on 4 non-linear functions, each divided into 5 sub-tasks, using a meta-learned neuronal phenotype which is held fixed. Synapses are updated with standard Backpropagation. In each of the 4 plots, each color in the plot depicts the predictions of the model over the whole function after performing 100 updates on data from the current sub-task only. In other words, the darkest plot represents the model's predictions over the whole function after training only on task_1 $[-5,-3)$. During the next stage of learning, the model performs 100 updates on data from task_2 only $[-3,-1)$. The lightest plot (cyan) depicts the model's predictions after the last round of learning: 100 updates on data from task_5 $[3,5]$. As shown, the model retains a good fit over the whole function even when it learns these sub-tasks in a sequential manner.

In practice, we also use skip connections, inspired by Deep Residual Networks He et al. (2015) and Flennerhag et al. (2020), in order to facilitate efficient learning. Skip connections are realized as additional layers of Synapses $\theta_{skip(j,k)}$, with dimensions $n_j \times (n_k \times n\_channels)$, which bypass layers of DANs by providing a direct pathway from nodes in layer j to layer k, where $k = j+2$. This allows some information to be sent directly downstream, without being subjected to processing by the intermediate layer of DANs. When using skip conenctions,

the input vector to a layer of DANs in layer k is obtained by summing the vector computed in Equation 5.2 with the vector signal traveling along $\theta_{skip(j,k)}$:

$$\sigma'_{k\_in} = \sigma_{k\_in} + (\sigma_{j\_out} \cdot \theta_{skip(j,k)}) \tag{5.3}$$

The result is a complete model, comprised of 2 distinct sets of parameters: Synapses, parameterized by θ, and DANs parameterized by φ. As we will show in coming sections, Synapses are intended to be fully plastic at all times. The parameters of our DANs, our so-called neuronal *phenotype*, are meta-learned and then held fixed during deployment.

As in Flennerhag et al. (2020), we leverage the benefits of a unique set of parameters φ which can be shown to warp the gradients applied to another set of parameters θ in order to prevent catastrophic forgetting. In contrast to WGD Flennerhag et al. (2020), however, we show that a single, small network, parameterized by φ, is sufficient to facilitate our meta-objective, rather than unique, separate layers of warp parameters. Additionally, we feel that our approach offers an additional layer of biological plausibility, and might help to explain some of the behavior and responsibilities of real neurons.

5.3 Methodology

Meta-Learning is generally concerned with optimizing some meta-objective over a distribution of tasks in order to attain some innate proficiency at comparable tasks likely to be encountered during a separate, deployment phase. To accomplish this, most meta-learning algorithms employ an inner-loop/out-loop framework, wherein optimization over several specific tasks occurs in the inner-loop, and proficiency at the meta-objective is evaluated and

optimized in the outer-loop. Inspired by Warped Gradient Descent (WGD) Flennerhag et al. (2020), we adopt such an approach, and wish to meta-learn parameters which facilitate continual learning during deployment.

During meta-training, we randomly sample target functions of the form defined in Equation 5.1. These target functions are split into 5 sub-tasks, as explained in Section 5.1. We deploy our model on each target function, and sub-tasks are encountered sequentially. Optimization over a single sub-task is done by performing backpropagation on both sets of parameters, θ and φ, using the loss over the current sub-task. This is known as an inner-loop epoch. At the end of each inner-loop epoch, we quantify the Meta-Loss over subtasks [0,...,*cur*], where *cur* is the current sub-task. Meta-Optimization is done by performing backpropagation on parameters φ ***only***, using the Meta-Loss. Repetition of this process over a sequence of 5 sub-tasks, given the current target function, is known as a meta, or outer-loop epoch. At the end of each outer-loop epoch, we sample a new target function, and repeat the process.

More formally, let the Historical Learning Trajectory $\mathcal{H}_t$ represent the dataset [x_0, x_1, ..., x_t] comprised of all data encountered by the system, prior to and including timestep t. Note that $\mathcal{H}_t$ therefore contains data from one or more sub-tasks $\mathcal{T}$. We have Synapses, parameterized by θ, and DANs, parameterized by φ, which together define the complete Model.

Note that since DANs are distributed throughout the network, the gradients for Synapses ∇_θ depend on parameters φ, and that the meta-gradient ∇_φ depends on parameters θ. This is true, since each set of parameters θ and φ are factors of both gradients.

We can define a Model State at timestep t as $\theta_t\varphi_t$. Given a new sample at timestep

$t+1$, this Model State will result in a measurable Task Loss $\mathcal{L}_{\mathcal{T}}$, for which we can compute a gradient $\nabla_{\theta_t \varphi_t} \mathcal{L}_{\mathcal{T}}^{t+1}$. Note that we can factor this gradient into its distinct components:

$$\nabla_{\theta_t \varphi_t} \mathcal{L}_{\mathcal{T}}^{t+1} = \nabla_{\theta_t} \nabla_{\varphi_t} \mathcal{L}_{\mathcal{T}}^{t+1} \tag{5.4}$$

This is desirable since we may want to assign separate learning rates to each set of parameters. For instance, let α denote the learning rate for parameters θ, and let γ denote the learning rate for parameters φ. Since we update both θ and φ in the inner loop, when learning individual sub-tasks, performing an inner-loop update on data from the current task at timestep $t+1$, like so:

$$\theta_{t+1} \varphi_{t+1} \leftarrow \theta_t \varphi_t - \alpha \nabla_{\theta_t} \gamma \nabla_{\varphi_t} \mathcal{L}_{\mathcal{T}}^{t+1} \tag{5.5}$$

...results in the new Model State $\theta_{t+1} \varphi_{t+1}$. Note that this update, $\theta_{t+1} \varphi_{t+1} \leftarrow \theta_t \varphi_t$, may have caused forgetting over $\mathcal{H}_{t+1}$, which now includes the latest data sample x_{t+1}.

We can quantify the Memory Loss over $\mathcal{H}_{t+1}$, defined as $\mathcal{L}_{\mathcal{M}}^{\mathcal{H}_{t+1}}$. This constitutes the meta-loss, which we wish to minimize in order to *facilitate* our meta-objective during inner-loop deployment.

Specifically, we seek an optimal neuronal phenotype, defined by a single parameter vector φ^*, shared by all DANs, which would have resulted in the least amount of forgetting over $\mathcal{H}_{t+1}$. Said another way, had the original state of the model been $\theta_t \varphi_t^*$, instead of $\theta_t \varphi_t$, then the inner loop update would have been:

$$\theta_{t+1}^* \varphi_{t+1}^* \leftarrow \theta_t \varphi_t^* - \alpha \nabla_{\theta_t} \gamma \nabla_{\varphi_t^*} \mathcal{L}_{\mathcal{T}}^{t+1} \tag{5.6}$$

This would have resulted in an alternative Model State $\theta^*_{t+1}\varphi^*_{t+1}$, ideally resulting in less forgetting than that originally induced by $\theta_t\varphi_t$.

Therefore, we calculate the Memory Loss across $\mathcal{H}_{t+1}$, using the current model state $\theta_{t+1}\varphi_{t+1}$, and compute the gradient w.r.t. this quantity. By taking a step towards φ^*_t, we update the phenotype φ, and in the process attempt to minimize the meta-loss. We can do this by factoring the gradient and isolating the update to φ only:

$$\varphi'_{t+1} = \varphi_{t+1} - \gamma \nabla_{\varphi_{t+1}} \mathcal{L}_{\mathcal{M}}^{\mathcal{H}_{t+1}} \quad \text{s.t.} \quad \varphi'_{t+1} \approx \varphi^*_t \tag{5.7}$$

Algorithm 1 Meta-Learning a Neuronal Phenotype for Continual Learning

Require: $p(\mathcal{T})$: distribution over target functions
Require: α, γ: learning rate hyperparameters
Require: inner_steps: number of inner loop steps
$\theta \leftarrow \theta_0, \varphi \leftarrow \varphi_0$: randomly initialize the model
while *not done* **do**
    Sample a Target Function $\mathcal{T} \sim p(\mathcal{T})$ **for** *sub-task st in* $\mathcal{T}$ **do**
    t in inner_steps
    Perform an update: $\theta_{t+1}\varphi_{t+1} \leftarrow \theta_t\varphi_t - \alpha \nabla_{\theta_t} \gamma \nabla_{\varphi_t} \mathcal{L}^t_{st}$ ▷ Equation 5.5

    $\mathcal{H}_{t+1} \leftarrow$ data from sub_tasks[0,...,st] $\subset \mathcal{T}$
    Compute Memory-Loss over $\mathcal{H}_{t+1}$
    Update Phenotype: $\varphi'_{t+1} = \varphi_{t+1} - \gamma \nabla_{\varphi_{t+1}} \mathcal{L}_{\mathcal{M}}^{\mathcal{H}_{t+1}}$ ▷ Equation 5.7

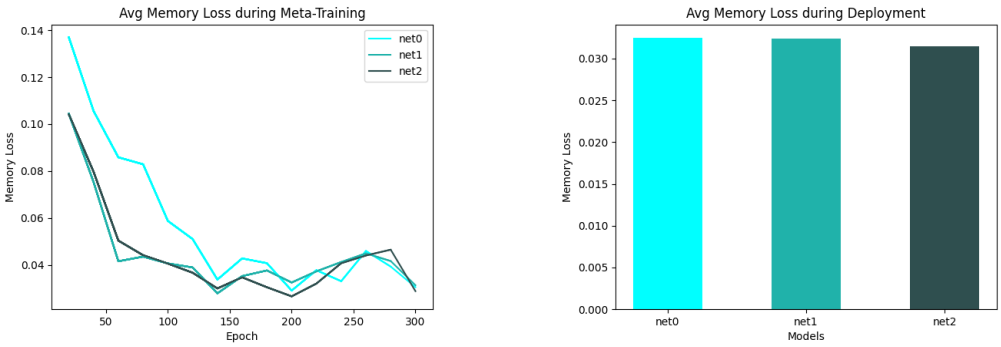
$\theta \leftarrow \theta_0$: reset VECs to initialization

The full meta-training procedure is outlined in Algorithm 1.

After meta-training is completed, the model is *deployed*. DAN parameters φ are held ***fixed***, and the model is obligated to learn continually, without forgetting, using standard backpropagation. That is, θ update normally, while DANs remain fixed. We offer empirical validation of our approach in the next section.

5.4 Experiments and Results

For all experiments, we used a network topology of 1 input node, 2 hidden layers of 40 nodes each, and a single output node. Recall that, apart from the single node in the input layer, each node represents a DAN, and the topology is therefore *converted* to a network of DANs. To this topology, we added 2 skip layers, as described in Section 5.2: from layer 0 to layer 2, and also from layer 1 to layer 3. The DAN itself is a 3 layer neural network with $n\_channels$ input nodes, followed by a hidden layer with 15 nodes, another hidden layer with 8 nodes, and a single output node, parameterized by φ. We applied *tanh* activation's to the hidden and output layers of the DAN. For all experiments except that depicted in Fig.5.1, we set $n\_channels = 40$. For Meta-Training, we set the learning rate for Synaptic parameters $\theta = .001$, and the learning rate for DAN parameters $\varphi = .0001$.



(a) (b)

Figure 5.3: Model definitions: **net0** uses a single, shared phenotype (cell-type); **net1** uses one phenotype for each layer; **net2** does not enforce parameter sharing among any DANs. (left) Minimization of the Memory-Loss during Meta-Training. (right) Avg Memory Loss during Deployment is nearly equal for all models, and nearly identical to the loss achieved near the end of meta-training, $\approx .03$ (mean squared error) across the full task-trajectory after learning 5 sub-tasks in sequence.

Fig.5.2 shows the ability of a meta-trained model to learn continually during ***deployment***; when it encounters tasks in a sequential manner, and is obligated to retain a good

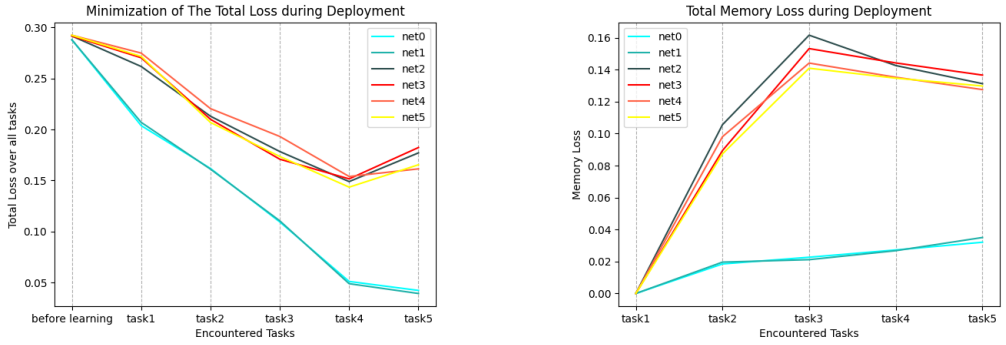
fit over previous sub-tasks, even though it is exposed to data from each task only once. During this experiment, DAN parameters φ were held fixed, and the network learns by using standard backpropagation to update Synapses θ.

Fig.5.4 depicts minimization of the Memory-Loss, our meta-objective, during meta-training. We found that the model converges relatively quickly, requiring only 200-300 meta-epochs to find a suitable phenotype, though this is likely due to task simplicity. Additionally, as the Figure shows, we sought to isolate the effect of using a single set of parameters for DANs in the whole network. To do this, we compared 3 models: one which used a single parameter vector for all DANs (net0: a single phenotype throughout the network), another which used a separate parameter vector for each *layer* of DANs (net1: phenotypes unique to each layer), and a third which did not enforce any parameter sharing amongst DANs (net2; 81 unique DANs in the network).

In above fig we compare the abilities of various models during deployment, when they are confronted with tasks in a sequential manner, and obligated to learn continually. Specifically, we sought to verify whether the meta-learning procedure was indeed endowing the DANs with an innate ability to assist in learning without forgetting. These plots confirm that hypothesis, showing that a meta-learned phenotype outperforms random parameter vectors, regardless of whether they are fully plastic during deployment, or fixed.

Finally, we investigated the effect of the size of *n_channels* on the ability of the model to minimize Memory-Loss during meta-training. Specifically, we asked, is there indeed a benefit to vectorizing the connections between pairs of DANs, and in the process increasing the size of the input to each DAN Above figure shows that the answer to that question was also yes. The plot shows that as the number of (1) connections between pairs of neurons

and (2) the size of the input to each DAN grows, the speed, or efficiency, with which the Memory-Loss is minimized is increased. In other words, vectorized connections accelerated optimization of our meta-objective.



(a) (b)

Figure 5.4: a) Model definitions: **net0** uses a single, meta-learned phenotype, shared by all DANs, fixed during deployment; **net1** uses the same meta-learned single phenotype as **net0**, but it is fully plastic during deployment (updates to the φ are allowed); **net2** uses a random, shared phenotype, fixed during deployment; **net3** uses a random, shared phenotype, fully plastic during deployment; **net4** uses random, but completely unique DANs (no parameter sharing), fixed during deployment; **net5** uses random, but unique DANs, fully plastic during deployment. Once before learning begins, and after training on each successive task, the Total-Loss over the complete function is calculated. Clearly, the meta-learned phenotype outperforms random DANs. b)Total amount of Memory-Loss experienced by different models during deployment. Clearly, the meta-learned phenotype outperforms random DANs

5.5 Conclusions and Future Work

In this work, we offered a framework for thinking about artificial neurons as much more powerful functions, realized as deep artificial networks, which can be embedded inside larger plastic networks. We showed that it is possible to meta-learn a single parameter vector for such a model that, when held fixed, can facilitate a meta-objective during deployment. In the process, we hope to inspire a deeper understanding about the responsibilities of neurons in both artificial neural networks, as well as real brains. In future work, we plan to investigate

the potential of DANs in real-world vision and reinforcement-learning settings, as well as the possibility of optimizing several meta-objectives at once.

6| THIRD IDEA: DEEP ACTIVE LEARNING VIA OPEN-SET RECOGNITION

6.1 Introduction

In many applications, data is easy to acquire but expensive and time-consuming to label prominent examples include medical imaging and NLP. This disparity has only grown in recent years as our ability to collect data improves. Under these constraints, it makes sense to select only the most informative instances from the unlabeled pool and request an oracle (e.g., a human expert) to provide labels for those samples. The goal of active learning is to infer the informativeness of unlabeled samples so as to minimize the number of requests to the oracle. Here, we formulate active learning as an open-set recognition problem. In this paradigm, only some of the inputs belong to known classes; the classifier must identify the rest as unknown. More specifically, we leverage variational neural networks (VNNs), which produce high-confidence (i.e., low-entropy) predictions only for inputs that closely resemble the training data. We use the inverse of this confidence measure to select the samples that the oracle should label. Intuitively, unlabeled samples that the VNN is uncertain about are more informative for future training. We carried out an extensive evaluation of our novel, probabilistic formulation of active learning, achieving state-of the-art results on MNIST, CIFAR-10, and CIFAR-100. Additionally, unlike current active learning

methods, our algorithm can learn tasks without the need for task labels. As our experiments show, when the unlabeled pool consists of a mixture of samples from multiple datasets, our approach can automatically distinguish between samples from seen vs. unseen tasks.

6.2 Methodology

As noted above, our active learning approach iteratively selects samples from an unlabeled pool based on the confidence level of its OSR classifier. Below, we first formalize the active learning paradigm we are tackling, then detail our proposed system. In particular, we provide an overview of VNNs and explain how we use their outputs to select new samples to label.

6.2.1 *Formal problem definition*

Formally, an active learning problem is denoted as $P = (C, D_{train}, D_{eval})$, where C indicates the number of classes, D_{train} is the training set, and D_{eval} is the evaluation set, s.t. $D_{train} \cap D_{eval} = \emptyset$.

Let $\mathcal{D}_{train} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ be a dataset consisting of N i.i.d. data points where only m of them are labeled ($m \ll N$). Each sample $\boldsymbol{x}_i \in \mathbb{R}^d$ is a d-dimensional feature vector, and $y_i \in \{1, 2, \ldots, C\}$ represents the target label. At the start, $\mathcal{D}_{train}$ is partitioned into two disjoint subsets: a labeled set $\mathcal{L}$ which consists of the m labeled data points, and an unlabeled set $\mathcal{U}$ which consists of the remaining $N - m$ data points with unknown target labels. We will update both $\mathcal{L}$ and $\mathcal{U}$ after each iteration of our algorithm. We denote the state of a subset at a given timestep as $\mathcal{L}^t$ and $\mathcal{U}^t$, respectively, for $t \in \{0, 1, \ldots\}$.

In active learning, we first train a classifier f, with parameters θ, on $\mathcal{L}^0$. Afterwards we select b data points from $\mathcal{U}^0$ using our OSR criterion (see Sec. 6.2.2). These b data points

are then sent to the oracle for annotation. The annotated samples are removed from the unlabeled pool and added to the labeled pool, along with their newly acquired target labels. The updated labeled and unlabeled data pools become $\mathcal{L}^1$, of size $m+b$, and $\mathcal{U}^1$, respectively. Thus, the labeled pool grows in size as training progresses. We continue this process until the size of the labeled pool reaches a predefined limit (40% of D_{train} in our experiments).

Importantly, unlike other formulations of AL, we allow for the unlabeled pool $\mathcal{U}$ to contain training data from *multiple datasets*. As we show in our experiments, our OSR-based AL method can automatically ignore samples that do not belong to the target classes.

Algorithm 2 Active Learning

Input: Unlabeled pool $\mathcal{U}^0$, labeled pool $\mathcal{L}^0$ for $t \in \{0, 1, \ldots\}$ where size of $\mathcal{L}^0 = m_0$.
Require: Active Learning Model, Optimizer, Sampling Strategy
Require: initialize b (budget), θ (Model parameters), Epochs
repeat
  Train Active Learning Model on Labeled Pool ($\mathcal{L}^t$) using selected optimizer.
  Give trained model f_θ on Labeled Pool ($\mathcal{L}^t$), Sampling Strategy (6.2.3 or 6.2.4) selects the uncertain data points according to budget size b.
  Send the selected data points to Oracle for annotation.
  Add the annotated data points to the Labeled Pool ($\mathcal{L}^t$)
until *stopping criterion (size of Labeled Pool* ($\mathcal{L}^t$) *equals 40% of* D_{train}*)*;

6.2.2 *Active learning system*

Algorithm 3 summarizes our AL approach, which has two main components: a variational neural network (VNN) Mundt et al. (2019b) that serves as our classifier and an OSR selection mechanism based on the loss function of the VNN. We discuss each component below.

Variational Neural Networks (VNNs)

Variational neural networks (VNNs) Mundt et al. (2019b) are a supervised variant of β-variational autoencoders (β-VAE) Higgins et al. (2017). The latter is itself a variant of

VAEs Doersch (2016) but with a regularized cost function. That is, the cost function for a β-VAE consists of two terms: the reconstruction error, as with a regular VAE, and an *entanglement* penalty on the latent vector. This penalty forces the dimensions of the latent space to be as uncorrelated as possible, making them easier to interpret.

A VNN combines the encoder-decoder architecture of a β-VAE with a probabilistic linear classifier (see Fig. 9.1 for a visual representation). As such, its loss function includes a classification error, i.e., a supervised signal, in addition to the reconstruction and entanglement terms:

$$\begin{aligned} L(\theta, \phi, \xi) = \mathbb{E}_{q_\theta(\boldsymbol{z}|\boldsymbol{x})} \left[\log p_\phi(\boldsymbol{x}|\boldsymbol{z}) + \log p_{\boldsymbol{\xi}}(\boldsymbol{y}|\boldsymbol{z})\right] \\ - \beta \, \mathrm{KL} \left(q_\theta(\boldsymbol{z}|\boldsymbol{x}) \| p(\boldsymbol{z})\right) \end{aligned} \tag{6.1}$$

As detailed in Mundt et al. (2019b), θ, ϕ, and ξ are the parameters of the encoder, decoder, and classifier, resp., while $p_\phi(\boldsymbol{x}|\boldsymbol{z})$ and $p_{\boldsymbol{\xi}}(\boldsymbol{y}|\boldsymbol{z})$ are the reconstruction and classification terms. The last term is the entanglement penalty, which is given by the Kullback-Leibler divergence between the latent vector distribution and an isotropic Gaussian distribution.

As in Mundt et al. (2019b), we evaluated both the full framework discussed above (dubbed M_2 in our experiments), which uses the loss function in Eq. 6.1, and a simplified version (M_1) without the reconstruction error:

$$L(\theta, \xi) = \mathbb{E}_{q_\theta(\boldsymbol{z}|\boldsymbol{x})} \left[\log p_{\boldsymbol{\xi}}(\boldsymbol{y}|\boldsymbol{z})\right] - \beta \, \mathrm{KL} \left(q_\theta(\boldsymbol{z}|\boldsymbol{x}) \| p(\boldsymbol{z})\right) \tag{6.2}$$

As our experiments show, both versions outperform the state of the art, but M_2 achieves better results overall.

Sample Selection

We wish to leverage the class disentanglement penalty defined in Eq. 6.1. Specifically, our aim is to select b data points from the unlabeled pool $\mathcal{U}$ that the VNN is highly uncertain about. Following Mundt et al. (2019a), in our experiments we investigated two sampling algorithms for OSR: *uncertainty sampling* and *Weibull distribution sampling*. The former is simpler, but the latter allows one to better reject outliers. We briefly describe each sampling strategy below.

6.2.3 *Uncertainty sampling*

Here, we select a data point $\boldsymbol{x}_i$ based directly on how uncertain the VNN is about it. Specifically, we rank all unlabeled samples by the value of the most likely class label and select the b samples with the lowest maximum values. Since the sum of class likelihoods is normalized, the value of the maximum class probability will approach one for highly certain samples and approach $\frac{1}{|C|}$, where $|C|$ is the number of classes, for highly uncertain samples. In other words, the class likelihoods of uncertain samples have higher entropy than those for which the VNN is certain about.

6.2.4 *Wiebull distribution sampling*

As our experiments show, uncertainty sampling is suitable for active learning problems in which all unlabeled samples belong to known classes. However, for the case where the unlabeled pool also contains samples from unknown classes, we need a more robust way to exclude outliers. For this latter case, we employed the sampling procedure defined in Mundt

et al. (2019a), which leverages a Wiebull distribution to estimate the model's uncertainty w.r.t a specific sample.

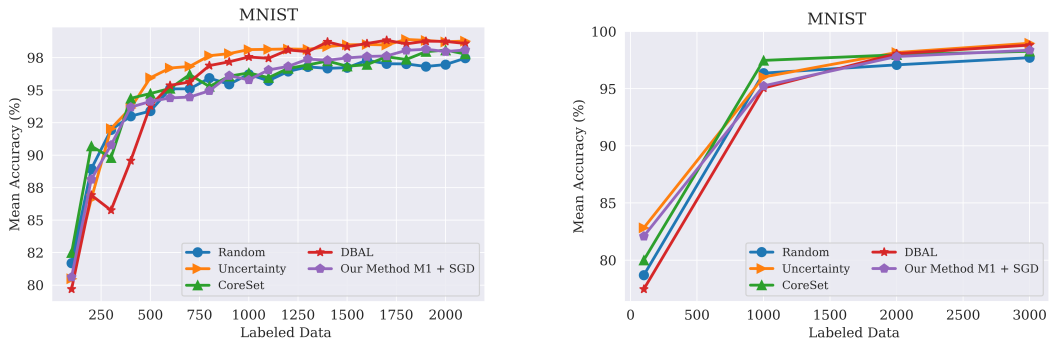
For completeness, here we will briefly outline the methodology proposed in Mundt et al. (2019a). Intuitively, it can be shown that it is useful to quantify the probability that a given data sample is an outlier, herein defined as a sample which is not sufficiently similar to those which have already been correctly classified. Mundt et al. (2019a) show that this can be accomplished as follows. First, for each class, we compute the mean of the latent vectors of all samples that have been correctly predicted by the model. Second, we compute the distances from each class mean for all latent vectors, which Mundt et al. (2019a) showed can be modeled with a Wiebull distribution. As such, a sample's likelihood under this distribution constitutes the minimum probability that the sample does ***not*** belong to any previously known class. In other words, the lower this value, the more likely that the sample is an outlier.

6.3 Experimental Results

We performed experiments on three image classification datasets—MNIST, CIFAR-10, and CIFAR-100—following the methodology defined in Section 7.3. Below, we first present our implementation details, then discuss our results.

6.3.1 *Implementation Details*

Budget: For CIFAR-10 and CIFAR-100, we used a max budget of 40%, and stage budgets b of 10%, 15%, 20%, 25%, 30%, 35%, and 40%. For MNIST, we used stage budgets of 100 and 1000 images.



(a) (b)

Figure 6.1: Performance on MNIST classification tasks using different query sizes for model M_1. (a) Query batch size of 100; (b) Query batch size of 1000 compared to Core-set Sener and Savarese (2017), DBAL Gal et al. (2017), Random Sampling and Uncertainty Sampling. M1 indicates our model with Encoder and Classifier. Best visible in color. Prior results adapted from Sinha et al. (2019).

Runs: For all three datasets, we measured performance by computing the average accuracy across 5 independent runs.

State of the art comparison: We compared our method against several recent AL approaches including Variational Adversarial Active Learning (VAAL) Sinha et al. (2019), Core-Set Sener and Savarese (2017), Monte-Carlo Dropout Gal and Ghahramani (2016), Ensembles using Variation Ratios (Ensembles w. VarR) Freeman (1965) Beluch et al. (2018), and Deep Bayesian AL (DBAL) Gal et al. (2017). As a baseline, we also included uniform random sampling (Random) since it remains a competitive strategy in the field of active learning.

Architectures: For experiments on CIFAR-10 and CIFAR-100 we used a VGG16 network Simonyan and Zisserman (2014a) as the encoder for both models, M_1 and M_2, and a decoder based on 14-layer residual networks Higgins et al. (2017); Zagoruyko and Komodakis (2016). We used latent vectors of size 60. As noted in Sec. 7.3, the classifier consists of a single linear layer. For MNIST, we used a LeNET network Lecun et al. (1998) as our encoder

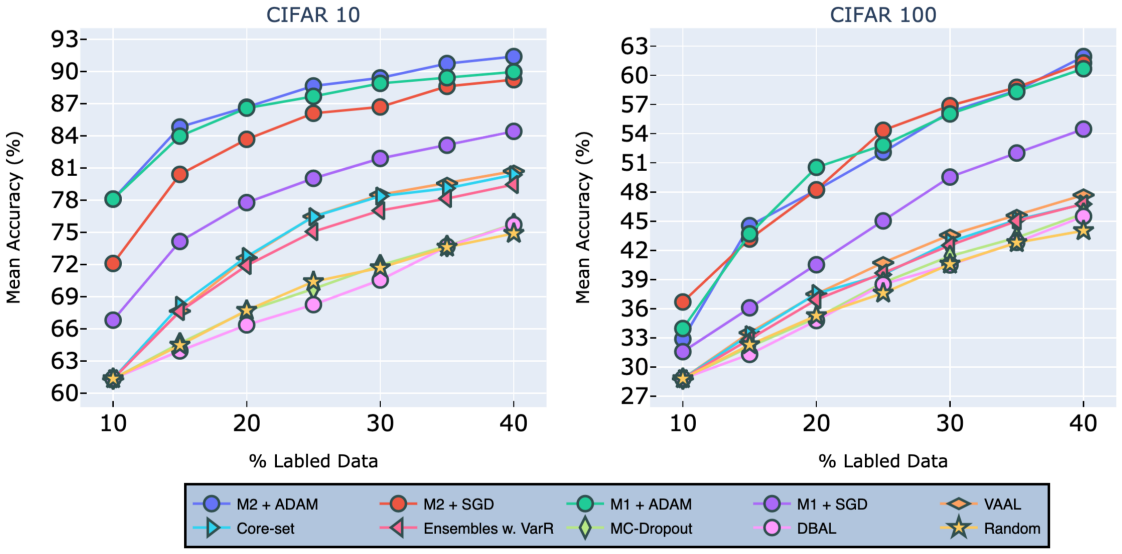
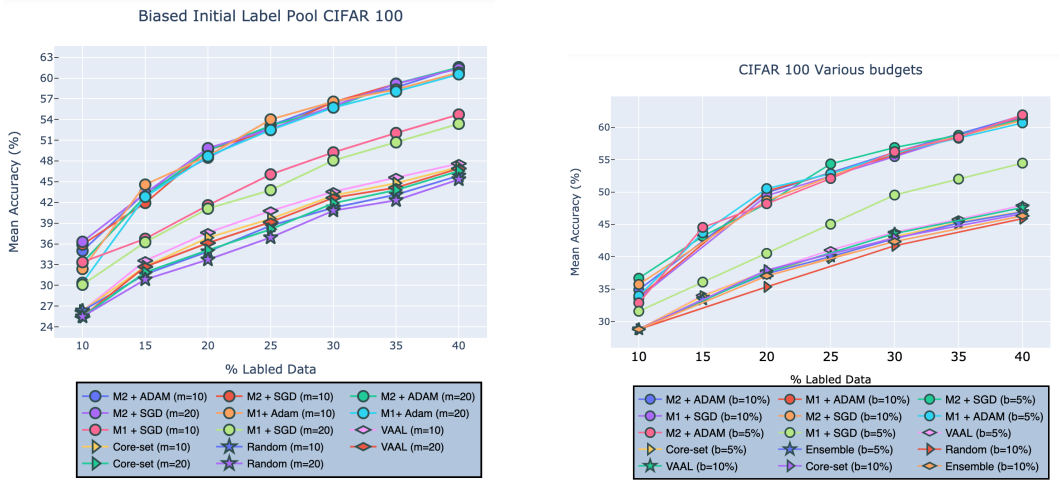


Figure 6.2: Performance on classification tasks for CIFAR-10 (left) and CIFAR-100 (right) compared to VAAL Sinha et al. (2019), Core-set Sener and Savarese (2017), Ensembles w. VarR Beluch et al. (2018), MC-Dropout Gal and Ghahramani (2016), DBAL Gal et al. (2017), and Random Sampling. M1 indicates our model (6.2) and M2 indicates our model (6.1). All the legend names are in descending order of final accuracies. Best visible in color. Prior results adapted from Sinha et al. (2019).



(a) (b)

Figure 6.3: Robustness of our approach on CIFAR-100 given (a) biased initial labeled pool or (b) different budget sizes compared to VAAL Sinha et al. (2019), Core-set Sener and Savarese (2017) , Ensembles w. VarR Beluch et al. (2018), MC-Dropout Gal and Ghahramani (2016), DBAL Gal et al. (2017), and Random Sampling. M1 indicates our model (6.2) and M2 indicates our model (6.1). Best visible in color. Prior results adapted from Sinha et al. (2019).

and a latent vector of size 60.

Optimization: We optimized all models using a mini-batch size of 128, a learning rate of 0.001, and a weight decay of 10^{-5}. We tested two different optimizer, SGD and ADAM Kingma and Ba (2014), for both M_1 and M_2, for a total of four combinations:

- M_1^{sgd} - Model M_1 as shown in Eq. 6.2 with SGD optimizer.
- M_1^{adam} - Model M_1 as shown in Eq. 6.2 with Adam optimizer.
- M_2^{sgd} - Model M_2 as shown in Eq.6.1, with SGD optimizer.
- M_2^{adam} - Model M_2 as shown in Eq.6.1 with Adam optimizer.

Oracle queries: We defined a learning stage (i.e., a period of training between queries to the oracle) as lasting 150 epochs on CIFAR-10 and CIFAR-100 and 10 epochs on MNIST. At the completion of a stage, we requested labels for b images from the unlabeled pool. These were added to the labeled pool and used in the subsequent learning stages.

6.3.2 *Image classification results*

MNIST: Our results were comparable with the state of the art on MNIST. However, as Figs. 6.1(a) and Fig. 6.1(b) show, random sampling is already a highly successful strategy on MNIST, leaving little room for improvement on this dataset. In particular, as illustrated in Fig. 6.1(b), all methods obtained statistically similar results as the batch size increased. However, as shown in Fig. 6.1(a) methods such as DBAL or Coreset have lower accuracies at the initial stages when using smaller batch sizes.

CIFAR-10 & CIFAR-100: As Fig. 6.2 clearly shows, we achieved state-of-the-art performance by a considerable margin on both CIFAR-10 (left) and CIFAR-100 (right).

On CIFAR-10, models $[M_1^{sgd}, M_1^{adam}, M_2^{sgd}, M_2^{adam}]$ achieved mean accuracies of [84.4%, 89.24%, 89.97%, 91.4%], respectively. To put this in perspective, the original accuracy for this VNN using the entire CIFAR-10 dataset was 92.63%. VAAL came in second, with an accuracy of only 80.71% , followed by Core-Set with an accuracy of 80.37%, and then Ensemble w VarR at 79.465%. Random sampling, DBAL and MC-Dropout all trailed significantly behind other methods. Finally, we found that our models trained with ADAM, on average, outperform those trained with SGD.

On CIFAR-100, models $[M_1^{sgd}, M_1^{adam}, M_2^{sgd}, M_2^{adam}]$ achieved mean accuracies of [54.47%, 60.68%, 61.25%, 61.93%], resp. The original accuracy with the entire CIFAR-100 dataset was 63.14%. VAAL once again came in second, with an accuracy of 54.47 %, followed by Core-Set, and Ensemble w VarR.

6.3.3 *Additional experiments*

In addition to our classification experiments, we replicated and extended the experiments of the same name put forth in Sinha et al. (2019) in order to investigate the robustness of our approach. Unless otherwise stated, we used CIFAR-100 for these experiments. Finally, we also tested our methods' ability to learn when the unlabeled pool contained out-of-distribution samples, a case which, to the best of our knowledge, cannot be handled by any existing methods.

Effect of Biased Initial Pool: We first investigated the effect of bias that may be present in the initial labeled pool, $\mathcal{L}_0$. As stated in Sinha et al. (2019), bias can negatively impact the training of an active learner because it means that the initial labeled pool may not be representative of the true underlying data distribution. Unless explicitly accounted

for, this will cause a system to learn an incomplete, or biased, model of the latent space. Following the protocol defined in Sinha et al. (2019), we removed all data points for c classes from $\mathcal{L}_0$, thereby unbalancing the dataset and thus introducing bias. As shown in Fig. 6.3(a), our method outperformed VAAL, Core-set, and random sampling w.r.t selecting useful data points from classes that were underrepresented in the initial labeled pool. Models $[M_1^{sgd}, M_1^{adam}, M_2^{sgd}, M_2^{adam}]$ achieved accuracies of [53.35%, 60.54%, 61.36%, 61.55%], respectively, when $c = 20$ and [54.72%, 60.79%, 61.53%, 61.57] when $c = 10$ (as noted above, c is the number of classes from which to exclude data). VAAL, by comparison, came in second, followed by Core-set, exhibiting accuracies [46.91%, 46.55%] for c=20 and [47.10%, 47.63%] for c=20, respectively. Random sampling achieved an accuracy of 45.33% for $c =$ 10 and 45.87% for $c = 20$.

Effect of Budget Size on Performance: In this section, we tested the effect of different budget sizes b on performance. Specifically, we investigated the effect of budgets of size b = 5% and b = 10%, referring to percentage of samples taken from $\mathcal{D}_{train}$ at each stage of learning. As shown in Fig. 6.3(b), our model outperformed VAAL, Core-Set, Ensemble, and random sampling over both the budget sizes. VAAL comes in second followed by Core-set and Ensemble. Models $[M_1^{sgd}, M_1^{adam}, M_2^{sgd}, M_2^{adam}]$ achieve accuracies of [61.52%, 61.57%, 61.07%, 61.82%] for $b = 10$ and [54.32%, 60.68%, 61.29%, 61.9%] for $b = 20$.

Noisy Oracle: Next, we investigated the performance of our approach in the presence of noisy data caused by an inaccurate, or noisy oracle. As in Sinha et al. (2019), we assumed that incorrect labels can be caused by the natural ambiguity which exists between examples drawn from 2 separate classes, rather than adversarial attacks. CIFAR-100 has both classes and super-classes, so, following Sinha et al. (2019), we randomly modified the labels of either

Figure 6.4: Robustness of our approach on CIFAR-100 given a noisy oracle. M_1 indicates our model (6.2) and M_2 indicates our model (6.1). All legend names are in descending order of final accuracies.

10%, 20% or 30% of the samples by replacing them with a label from another class within the same super-class. As shown in Fig. 6.4, our models consistently outperformed existing approaches *across all noise levels*. In other words, our M_1 model with 30% noise was *more accurate* than VAAL, etc. with 10% noise.

Sampling Time Analysis We also replicated the sampling time analysis put forth in Sinha et al. (2019). Table 6.1 shows that our method is competitive with other state-of-the-art techniques w.r.t. execution time, thereby offering strong empirical evidence that our method offers large performance advantages with minimal additional computation.

Out-of-distribution samples in unlabeled pool: Finally, we also tested an extreme case of active learning in which data samples from other datasets are mixed into the current unlabeled pool. We used CIFAR-10 for these experiments. Here, we intentionally added 20% data (10,000 images) from other datasets to the unlabeled pool; thus, the network must

Table 6.1: Sampling Time Analysis: Mean time to select a sample from the unlabeled pool of CIFAR-100.

Method	**Time (Seconds)**
VAAL	10.69
Uncertainty sampling	**10.89**
DBAL	11.05
Weibull sampling	**20.41**
Ensembles w. VarR	20.48
Core-set	75.33
MC-Dropout	83.65

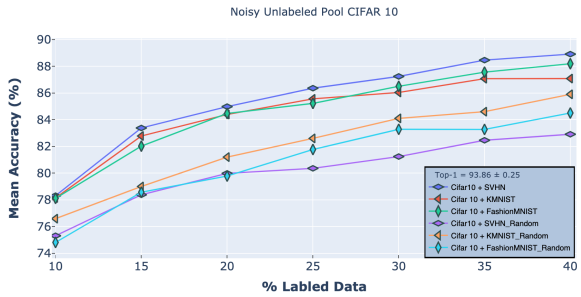


Figure 6.5: Robustness of our approach on CIFAR10 classification tasks when the unlabeled pool includes samples from either the SVHN, KMNIST, or FashionMNIST datasets. The first three curves used the M_2 classifier, while the ones with the 'Random' subscript used random sampling. Our results confirm that our approach significantly outperforms this baseline.

distinguish not only between informative and non-informative samples but also distinguish *in-distribution* data samples from *out-of-distribution* samples. Whenever our model selected an OOD sample, the oracle discarded the sample, thus reducing the overall budget size. The discarded samples were placed back in the unlabeled pool (so the total number of OOD samples remained at 10,000).

Figure 6.5 shows our M_2 method's performance on CIFAR-10 when the unlabeled pool contained images from either SVHN, KMNIST, or FashionMNIST. Here, we used Weibull sampling (Sec. 6.2.4) due to its better outlier rejection compared to uncertainty sampling.

For comparison, we also tested random sampling as a baseline. Impressively, despite the presence of 20% OOD samples, our method significantly outperformed existing state-of-the-art methods trained on the regular unlabeled pool (Fig. 6.2). And its performance, regardless of the second dataset, was only slightly below the standard M_2 method.

6.4 Conclusions and Future work

We have presented a novel approach for deep active learning using open-set recognition. To the best of our knowledge, we are the first to merge AL with OSR. Extensive experiments conducted over several image classification datasets have verified the effectiveness of our approach and established new state-of-the-art benchmarks. Specifically, we empirically demonstrated that the samples most worth labeling are those which are most different from the current labeled pool. Training on such samples allows the model to learn features underrepresented in the existing training data. We extensively tested the robustness of our approach using different budget sizes, a noisy oracle, and an unlabeled pool comprised of multiple datasets. In future work, we plan to test our approach on continual learning problems, in which the system must learn to solve different problems over time. We also plan to test our method on other problems, including image segmentation and document classification.

7| FOURTH IDEA: DEEP ACTIVE LEARNING USING BARLOW TWINS

The generalisation performance of a convolutional neural networks (CNN) is majorly predisposed by the quantity, quality, and diversity of the training images. All the training data needs to be annotated in-hand before, in many real-world applications data is easy to acquire but expensive and time-consuming to label. The goal of the Active learning for the task is to draw most *informative samples* from the unlabeled pool which can used for training after annotation. With total different objective, self-supervised learning which have been gaining meteoric popularity by closing the gap in performance with supervised methods on large computer vision benchmarks. self-supervised learning (SSL) these days have shown to produce low-level representations that are invariant to distortions of the input sample and can encode invariance to artificially created distortions, e.g. rotation, solarization,cropping etc. self-supervised learning (SSL) approaches rely on simpler and more scalable frameworks for learning. In this paper, we unify these two families of approaches from the angle of active learning using self-supervised learning mainfold and propose **D**eep **A**ctive **L**earning using **Barlow Twins** (DALBT), an active learning method for all the datasets using combination of classifier trained along with self-supervised loss framework of Barlow Twins to a setting where the model can encode the invariance of artificially created distortions, e.g. rotation, solarization,cropping etc.. We propose to use joint loss function which consist classifier

loss and self-supervised loss borrowed from Barlow twins to jointly learn an encoder that produces representations invariant across such pairs. DALBT is a method that is simple, easy to implement and train, and of broad applicability. We carried out an extensive evaluation of our novel proposed method of active learning, achieving state-of-the-art results on MNIST, Fashion-MNSIT, CIFAR-10. Additionally, to show the robustness of the proposed model we also showed the results on where the unlabeled pool consists of a mixture of samples from multiple datasets, proposed model can successfully distinguish between samples from seen vs. unseen datasets.

7.1 Introduction

Although deep neural networks (DNNs) demonstrated state of the art (SOTA) accuracy on several supervised learning tasks such as as classification (He et al., 2016; Krizhevsky et al., 2012), object detection (Redmon et al., 2016; Ren et al., 2015), and semantic segmentation. But most of the deep neural networks (DNNs) require large set of labeled data to achieve this feet. The challenges of labeling huge datasets in real world setting are many: expensive, limited time available by domain business experts, long labeling time per for large-scale sample such as videos and time-series data, financial constraints, or to minimize the model's carbon footprint. These all drawback does inherit the application of deep neural networks (DNNs) to more research areas and more organization.

In order to overcome the above drawbacks, Active Learning(AL) system try to select to most informative samples from the pool of unlabeled data points at each stage and send them for annotation to maximize the accuracy of the model. Active learning uses a fixed budget

at each stage of learning to select and label a subset of a data points from the unlabeled pool where budget(b) refers to cost associated with annotation by oracle($\mathcal{O}$). The model will be trained on the current labeled pool along with the newly annotated data points. At the end of active learning process model's performance would be nearly the same accuracy as model by utilized fraction of data when compared to the model trained on all the data. Active Learning(AL) also highlights the fact that there exists a non-linear relationship between the model's performance and the amount of training data used. There exists most representative subset of the unlabeled data and selecting those data points to label will provide most of the information needed to learn to solve a task. In this case, we can achieve nearly the same performance by selecting that representative subset for annotation (and training on) only using data points from that representative subset samples, rather than the entire dataset.

In contrast self-supervised learning which learns useful information from the dataset without relying on human annotations. Most of the work in the field of self-supervised learning work on goal of leaning good low level representations of input data without access to data labels. With the current advances in the field of Self-supervised learning (SSL)which is rapidly closing the dap with supervised learning methods on large datasets and computer vision taks. Most of the methods in SSL work with the goal of learning representations that are invariant under different distortions such as random cropping, resizing, horizontal flipping, color jittering, converting to grayscale, Gaussian blurring, and solarization.(also referred to as 'data augmentations').

From the high level view Active learning reduce the label-effort and Self-supervised learn-

ing aim to use the unlabeled data. With the goal of merging self-supervised learning along with supervised learning we propose **D**eep **A**ctive **L**earning using **Barlow Twins** (DALBT). The proposed method aims to combine self-supervised learning along with supervised learning in each step. Our proposed work is different from the previous work in the field which achieved this by doing pre-training on all the entire/partial unlabeled dataset using self-supervised learning then use the pre-trained model for active learning training step using supervised learning loss. But this approach increases the overall training time as in some cases the overall size of the unlabeled pool can be really big and this process is not feasible at times. In this paper we propose an Active learning system that utilizes both the supervise learning and unsupervised learning to achieve the goal of selecting the most informative samples from the unlabeled pool.

Our paper is organized as follows: In section 2 we describe the related work. Next, in section 3 we introduce the proposed framework. Section 4 present the experimental setup and the evaluations on the datasets we used. Finally, section 6 conclusion and discusses an interesting finding we observed in the proposed work.

7.2 Related Work

Here we detail previous work done in each of these directions.

7.2.1 *Active Learning*

Active learning methodologies were recently reviewed by Settles(Settles, 2010)), discussed more here (Dasgupta, 2011; Hanneke et al., 2014) and it has been shown that there exists

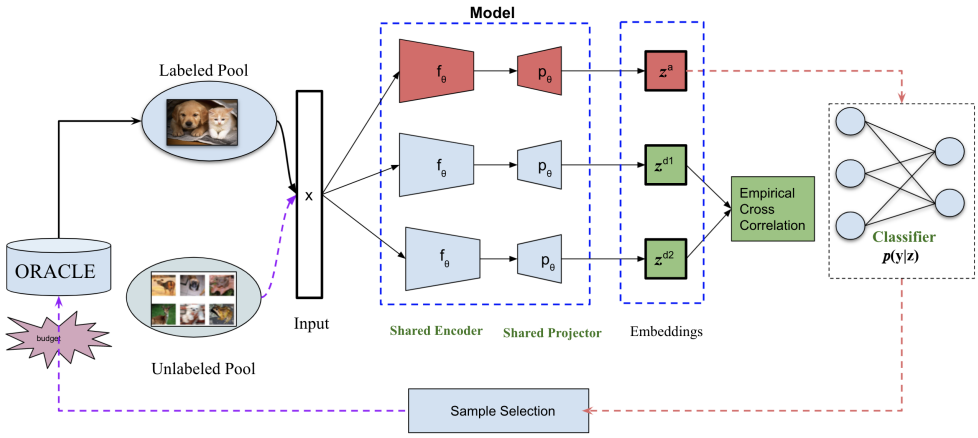


Figure 7.1: **Framework overview:** Our proposed active learning system based on BarlowTwins consists of encoeder(E),Projector(P),Classifer(C). Encoder takes two distorted versions of same input as its input. Output of encoder is fed into projector network which projects both the distorted versions into lower dimension. Classifier take latent vector(z) produced by passing the actual input image without distortion through the encoder(E). Overall system is trained using joint loss of cross-corelation loss and classification loss. Wiebull Sampling method to identify which samples from the unlabeled pool to label.

informative samples which contribute to performance than the other training samples. Thus, the overall goal of active learning is to learn or use an acquisition function along with model to chooses the best data points for which a label should be requested from a large unlabeled pool of data.

Existing Active learning approaches can be divide into Pool based methods or Query Synthesizing methods. Pool based methods tries to find the most informative samples from the unlabeled data using different sampling strategies which are more discussed in detailed in this section. Query Synthesizing methods (Mahapatra et al., 2018; McCallum and Nigam, 1998; Zhu and Bento, 2017) use generative models to genearate the informative samples.

Active learning sampling strategies can be sub-divided into following catgeories a) **Uncertainity Sampling** This is one of the popular sampling methodology in which where the

model queries data points about which it is most uncertain about. Recent research (Beluch et al., 2018; Gorriz et al., 2017; Lewis and Gale, 1994; Scheffer et al., 2001; Wang et al., 2017b) shows that uncertainity sampling approaches have proven effective in deep learning models such as CNNs. b) **Diversity sampling** This sampling method aims to choose samples which are more diversify from the existing labeled samples. c) **Representative Sampling** This sampling method aims to choose the samples from the unlabeled pool which are representative of the whole dataset. There exists subfield in the research which uses combination of features from these three disjoint groups mentioned above to increase the performance of activelearing system.

(Schohn and Cohn, 2000) uses active learning to enhace the performance on document classification tasks using support vector machines (SVM) by labeling examples that lie closest to the SVM's dividing hyperplane. The authors also proposed stopping heuristic for AL on when the model reach the peak generalization performance.

(Tong and Koller, 2001) applied active learning to Text Classification using SVM by implicitly projecting the training data into a different (often higher dimensional) feature space which is linearly separable. Then projects the query selection problem as version space optimization problem in which version space optimizates as quickly as possible still obeying the SVM constraints which is equivalent to finding informative samples quickly.

(Tur et al., 2005) combines active and semi-supervised learning methods int the domain of spoken language understanding.

(Wang et al., 2017a) combines uncertainty based active learning algorithm with diversity

constraint by sparse selection in which sample selection is represented as a sparse modeling problem.

(Sener and Savarese, 2017) projected the problem of active learning as core-set selection problem in set of points are choosen such that selected points should be dissimilar both to each other and the labelled set, representative of the unlabelled set and competitive for the remaining data points

(Zhu et al., 2009) combined uncertainty and density (SUD) and density-based re-ranking to overcome the problem outlier selection problem present in Uncertainty sampling. By combining uncertainty along with density-based re-ranking which selects the samples which are most informative example in terms of uncertainty criterion, but also the most representative example in terms of density criterion.

Similar to core-set approach of selecting batch of images in pool-based setting (Geifman and El-Yaniv, 2017) selects the points for each class by using farthest-first(FF) traversal principle or famously known as Gonzalez algorithm (Gonzalez, 1985). FF principle states that traversal for a set of points can be constructed by selecting the first point x randomly then next point is selected which is farthest from previously selected point x by greedily choosing the point farthest away from any of the points already chosen. Set of point obtained using the neural activation over a representation layer by forward passing all the unlabeled data. farthest-first(FF) traversal principle is similar to building long-tail wiebull distribution.

(Gissin and Shalev-Shwartz, 2019) motivated by selecting sample for which the probabil-

ity of distinguishing it unlabeled pool and labeled pool is the highest. Such that selecting such kind of the samples for labeling should be informative and helps in increasing the performace of the model.

(Beluch et al., 2018) showed ensembles perform better and lead to more calibrated predictive uncertainties which can be used for ActiveLearning Uncertanity strategy. Authors also showed that this method performs better than the Monte-Carlo Dropout and geometric approaches

7.2.2 *Self-Supervised Learning*

In recent years, self-supervised learning has achieved comparable performance w.r.t to suprvised learning (Caron et al., 2020; Chen et al., 2020; Grill et al., 2020). Most of the self-supervised learning methods work with a goal of achieving where representations are learned that are invariant to distortions present in the input data. Distorted inputs are created using different data augmentation applied to input randomly. Different research methods try to achieve this goal using different approaches such as in SIMCLR (Chen et al., 2020) achieved this by creating 'positive' and 'negative' sample pairs from the input data and treating each pair differently in the loss function, BarlowTwins (Zbontar et al., 2021) achieves this using variance and invariance terms in which two distorted versions of single sample should produce sample low level represntaion which is achieved using custom loss function consists of variance and redundancy reduction term.

(Ash et al., 2019) proposed Batch Active learning by Diverse Gradient Embeddings (BADGE) method in which d to incorporate both predictive uncertainty and sample diversity into every selected batch. Authors achieved this by calculating the gradient embedding for hypothetical label and used Kmean++ seeding algorithm to choose the batch to be labelled.

Previous works in the field of researchers trying to merge active learning and self-supervised learning are in Graphical domain (Zhu et al., 2020) applied self-supervised learning along with active learing to Graph Neural Networks by considering the information propa-gation scheme of GNN and selecting the central nodes from homophilous ego networks, (Bengar et al., 2021) utlized autoencoder architecturem SSL technique SIMCLR to form postive and negative pairs. In NLP for text classification task (Yuan et al., 2020) used self-supervised learning as a pre-training step for training the language model and the samples which the language model is uncertainare sent for labelling and for efficient fine-tuning. Similar approach of large-scale pseudo training data by randomly adding or deleting words from unlabeled data is followed by (Wang et al., 2021) for disfluency detection heavily rely on human-annotated data for solving sentence classificationt task. (Bengar et al., 2021) Model is trained on the entire dataset to get the frozen backbone. Now linear classifier or an SVM, decoder is fine-tuned on top of the features in supervised way, inference is run on the entire unlabeled data and top-k samples are collected via acquisition function. In medical domain (Mahapatra et al., 2021) collect the salency maps of medical images and project it as self-supervised learning problem where the autoencoder reconstructs the saliency maps of medical images followed by clustering the latent space to collect the top-k and Query labels of the most representative sample per cluster. Our work is particulary different from the other work in the field as all the previous work concentrates on high amount of pre-training

on all the data which is not feaisable as the unlabeled pool size is pretty high which add hughe overhead training time, creating "positive" and "negative" pairs for training is not feaisable when the overall dataset size is pretty large. (Mandivarapu et al., 2020b) merged the fields of active learning and open-set recognition in which model is trained using information bottleneck loss along with wiebull long tail distribution to find the outlier per class and achieved the state of the art in the field of active learning.

This proposed approach explores a active learning method including self-supervised learing which can be used for informative sample selection for labelling, in which the wiebull sampling was used as acquistion function. This work proposes to address all of above mentitone issues with a single approach, driven by a distinct business need.

7.3 Methodology

In this section, we briefly review the setup of the pool based active learning for computer vision classification tasks. We also discuss about used self supervised learning approach barlow twins and the intuitions behind using it. We then describe our proposed approach Deep Active Learning using Barlow Twins (DALBT). Throughout the sections, we refer to the model being trained as f and denote its corresponding weights/parameters θ. Given an unlabeled pool($\mathcal{U}$) of examples X without label, in each sampling iteration, our sampling method selects a diverse set of examples on which the model is least confident and useful for training at next iteration of active learning.

7.3.1 *Problem Definition*

Formally, pool based active learning problem is denoted as $P = (\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}})$; $\mathcal{D}_{\text{train}}$ is the training set from where initial pool of samples are taken. $\mathcal{D}_{\text{train}}$ can be sub divided into $(\mathcal{D}_{\text{L}}, \mathcal{D}_{\text{U}})$; $\mathcal{D}_{\text{L}}$ is the labeled pool where each sample consist of pair of input and label denoted by $((x_{\text{L}}, y_{\text{L}}))$. $\mathcal{D}_{\text{U}}$ denotes a much larger pool of samples $((x_{\text{L}})$ which are not yet labeled. The goal of the active learning model is to train on labeled pool$((x_{\text{L}})$ and used it along with sampling method to iteratively querying the most label-efficient samples present in the ublabeled pool $\mathcal{D}_{\text{U}}$ to be annotated by the oracle such that the expected loss is minimized by a fixed sampling budget(b). b is the total no of most informative samples that can be selected from the unlabeled pool at each stage of active learning setup. These selected b sampled will be sent to oracle for annotation. We denote the state of a subset at a given timestep as $\mathcal{L}^t$ and $\mathcal{U}^t$, respectively, for $t \in \{0, 1, \ldots\}$ where t indicates the current stage of active learning stages.

In standard pool based active learning setup, we train model our active learning model(f) with parameters θ on the initial labeled pool ($\mathcal{L}^0$) at stage t=0. After the initial stage t=0, b datapoints are sampled from the unlabeled pool using some predefined sampling method (eg: uncertainty measure, confidence estimate ..etc). These selected b data points will be removed from unlabeled pool ($\mathcal{U}^0$) and sent to oracle ($\mathcal{O}$) for annotation. These annotated datapoints are then added to labeled pool ($\mathcal{L}^0$) which now becomes labeled pool (c) and unlabeled pool($\mathcal{U}^0$) becomes ($\mathcal{U}^1$). Now the model again will be trained on new labeled pool ($\mathcal{L}^1$) at next stage t=1. In the current experimental setup we consider two scenarios where unlabeled pool($\mathcal{U}$) contains samples from the same datasets and mixture of multiple

datasets.

7.3.2 *Active Learning System*

With the goal of active learning using self-supervised learning. Our system consists of and encoder(E) followed by a projector (P), followed by a classifier(C) as shown in the Fig 7.1. Our goal is to learn an encoder f_θ can encode the invariance of artificially created distortions, e.g. rotation, solarization,cropping etc. The proposed model f_θ takes as input two distorted versions of the vector $x \in \mathbb{R}^D$ and outputs a corresponding reduced vector $z = f_\theta(x) \in \mathbb{R}^d$, with $d << D$. Without loss of generality, we define the encoder to be a neural network with learnable parameters θ. Let $\mathcal{L}^0$ be a initial labeled pool training set of datapoints in $\mathbb{R}^D$, the D-dimensional input space. Let $x \in \mathbb{R}^D$ be a vector from $\mathcal{X}$.

Barlow Twins

As mentioned in the Section Introduction we have used Barlowtwins (Zbontar et al., 2021) for finding the low-level representation of our inputs. In this we explain about barlow twins in more detailed fashion. Barlow twins networks consists of encoder(E) appended with projector network (p) as shown in Fig 7.1 excluding the classifier. For simplicity of explanation let's consider the case where the batchsize is 1. For each input image two distorted versions are produced using different types of random data augmentations applied during the training. Lets consider d^1 and d^2 as two distorted version of same input image x. These two distorted inputs are then fed into encoder(E) followed by a projector network (p) both with trainable parameters. The model then produces two output low level representation of the same input image but one each for each distorted version. Lets say z^{d1} and z^{d2} as two

low-level representations of d^1 and d^2.

Barlowtwins uses unique loss function as mentioned in the paper "" which is different from other SSL methods as shown below

$$\mathcal{L}_{\mathcal{BT}} \triangleq \underbrace{\sum_i (1 - \mathcal{C}_{ii})^2}_{\text{invariance term}} + \lambda \underbrace{\sum_i \sum_{j \neq i} \mathcal{C}_{ij}^2}_{\text{redundancy reduction term}} \tag{7.1}$$

where $\mathcal{C}$ indicates the cross-corelation matrix computed between the two identical networks which is given below and λ is hyper-parameter for defining the importance between the first and second terms of the loss.

$$\mathcal{C}_{ij} \triangleq \frac{\sum_b z_{b,i}^{d^1} z_{b,j}^{d^2}}{\sqrt{\sum_b \left(z_{b,i}^{d^1}\right)^2} \sqrt{\sum_b \left(z_{b,j}^{d^2}\right)^2}}$$

where b indexes batch samples and i, j index the vector dimension of the networks' outputs. C is a square matrix with size the dimensionality of the network's output with range of values from -1 to 1 where -1 indicated no-corelation between the $z_{b,i}^{d^1}$ and $z_{b,i}^{d^2}$, where 1 indicated perfect corelation between $z_{b,i}^{d^1}$ and $z_{b,i}^{d^2}$.

Active Learning using Barlow Twins

With the intention of merging both the self supervised learning method Barlow twins and active learning we proposed new changes to the existing barlow twins architecture as show

in Fig 7.1 and explained further. We appended additional classifier(C) to existing model of encoder(E),projector(P). Overall system is trained using the modified innovative loss as shown below

$$\mathcal{L}_{\mathcal{BT}} \triangleq \underbrace{\log p_{\boldsymbol{\xi}}(\boldsymbol{y}|\boldsymbol{z})}_{\text{Classifer term}} + \gamma * (\underbrace{\sum_{i} (1 - \mathcal{C}_{ii})^2}_{\text{invariance term}} + \underbrace{\lambda \sum_{i} \sum_{j \neq i} \mathcal{C}_{ij}^2}_{\text{redundancy reduction term}}) \tag{7.2}$$

where γ indicates the amount of importance given to the barlow twins loss and first term indicates the classifier loss. The overall system is optimized using the joint loss as shown in Eq 7.2. As you can see that the input to the classifier is the latent vector produced by actualling passing the input image without distortions through the model(f).

7.3.3 *Sampling technique*

With an aim to select b most informative data points from the unlabeled pool along with the trained model.

In depth the objective of the loss function in the Eq 7.2 is finidng the low-level representations that captures as much information as possible about the inputs while being least informative about the distortion applied to the input. We used wiebull sampling technique proposed by (Weibull, 1951) . Using the wiebull sampling method can be used to quantify weather a sample is an outlier or not. In our case if the latent representation is very different from the labeled pool latent represention it is considered as an outlier. Usage of long-tail distribution for finding the informative samples in the field of active learning is shown by (Gonzalez, 1985) and (Mandivarapu et al., 2020b). Firstly collect all the latent vectors of images which are classified correctly by the model at any stage of active learning. These

latent vectors are sub-divided into the respestive clusters depending on their class label. Now mean of each cluster is calculated and distance between mean of each class to rest of the points was calculated. Wiebull distribution is modeled using these distances for each class cluster. Finally any new images with out label will be pass through the wiebull model to check the percentage by which this image sample is considered as an outlier for all images and top such images are collected for labeling or for getting annontated by the oracle.

Algorithm 3 Active Learning

Require: Unlabeled pool $\mathcal{U}^0$, labeled pool $\mathcal{L}^0$,number of labeling iterations t, initialize b (budget)
Require: Active Learning Model(f_θ), Optimizer
for $k = 1$ **to** t **do**
  Train f_θ on Labeled Pool ($\mathcal{L}^k$)
  Z $\leftarrow$ Collect the latent vectors of all correctly classified samples in Labeled Pool ($\mathcal{L}^t$)
  $\mathcal{C}_i \leftarrow$ Mapping of Z_i onto separate cluster per class $\mathcal{C}_i$
  $\lceil_i \leftarrow$ Calculate the distance of each point to its cluster $\mathcal{C}_i$
  $\mathcal{W}_i \leftarrow$ Map the distances by fitting them to a wiebull model
  Z $\leftarrow$ Collect the latent vectors of all the samples Unlabeled Pool ($\mathcal{U}^k$)
  for $Z = 1$ **to** n **do**
    Collect the outlier probability score using previously fitted wiebull model.
  Request labels for top b samples
  $L^{k+1} \leftarrow L^k \cup b$ samples.

  Train f on L^{k+1}.

7.4 Experimental Results

We performed experiments on four image classification datasets: MNIST (LeCun et al., 2015), CIFAR-10 (Krizhevsky, 2009a), and FashionMNIST (Xiao et al., 2017)—following the methodology defined in Sec. 7.3. Below, we first present our implementation details, then discuss our results.

7.4.1 *Implementation Details*

Hardware: We carried out our experiments on a Dell Precision 7920R server with two Intel Xeon Silver 4110 CPUs, two GeForce GTX 1080 Ti graphics cards, and 128 GBs of RAM.

Dataset sizes and budgets: As previously explained in methodology section, *budget* refers to the number of samples labeled by the oracle in each round of active learning. Budget of the each experiments is shown in the legend of the each result. MNIST dataset consists of 50,000 images as part of the training set out of which is sub-divided into 100 images for the initial labeled pool, 5000 images as a validation set, and the remaining 44,900 images as part of the unlabeled pool. MNIST dataset also consists test set of size 10,000 images and we used it to check the performance of our model after each stage of our active learning setup. We used budgets of 100 and 1000 samples for experiments 7.2, resp. We used a similar setup for FashionMNIST. For CIFAR-10 which is similar to MNIST w.r.t total of no of images in train and test sets. CIFAR-10 training set out of which is sub-divided into 5000 images for the initial labeled pool, 5000 images as a validation set, and the remaining images as part of the unlabeled pool, we used a budget of 2500 images per round of active learning, up to 40% of the training data. CIFAR-10 test set consists of 10,000 images and we used it to check the performance of our model after each stage of our active learning setup

Runs: For all the experiments, we measured performance by computing the average accuracy across 5 independent runs.

State of the art comparison: We compared our method against several recent AL approaches including DAL-OSR(Mandivarapu et al., 2020b), Variational Adversarial Active

Learning (VAAL) (Sinha et al., 2019), Core-Set (Sener and Savarese, 2017), Monte-Carlo Dropout (Gal and Ghahramani, 2016), Ensembles using Variation Ratios (Ensembles w. VarR) (Freeman, 1965) (Beluch et al., 2018), Deep Bayesian AL (DBAL) (Gal et al., 2017), BatchBALD (Kirsch et al., 2019), and WAAL((Shui et al., 2020)). As a baseline, we also included uniform random sampling (Random) since it remains a competitive strategy in the field of active learning.

Architectures: For experiments on MNIST and Fashion-MNIST we used a LeNET network (Lecun et al., 1998) as the encoder, projector network, followed by a classifier. We used latent vectors of size 60. As noted in Sec. 7.3, the classifier consists of a single linear layer. For CIFAR-10, we used a VGG16 network (Simonyan and Zisserman, 2014a) as our encoder and a latent vector of size 512 followed by classifier with single layer.

Optimization: We optimized the overall system using a mini-batch size of 64, a learning rate of 0.001, barlow twins constant of 0.001 and a weight decay of 10^{-5}. We optimized the system for 150 epochs at each stage and 20 epochs on MNIST. At the completion of a stage, using wiebull sampling method we requested labels for b images from the unlabeled pool. Once the labels for the images are received from the oracle. These labeled images were added to the labeled pool and used in the subsequent learning stages.

Image Augmentations We use the augmentations similar to BYOL (Grill et al., 2020) which is used by major SSL approaches. As shown in the Fig 7.1 two distorted images are produced from given single input image by applying different kind of transformations. The image augmentation pipeline starts with random cropping, resizing which was applied to all images. Followed by Gaussian blurring, color jittering, converting to grayscale,horizontal flipping, and solarization which were e last five are applied randomly,

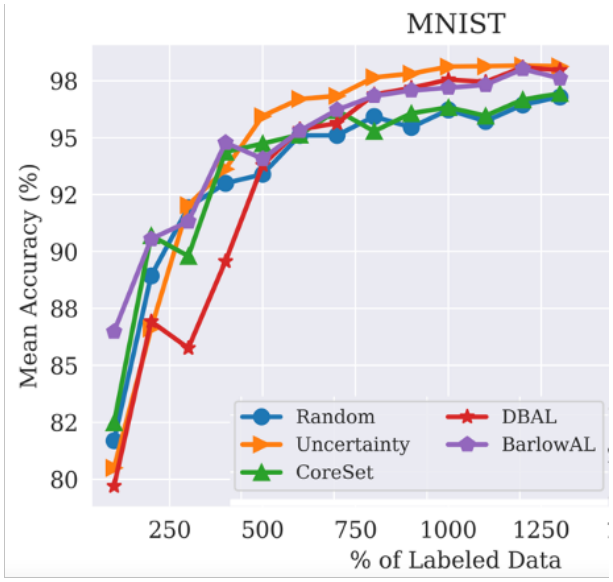


Figure 7.2: Robustness of our approach on MNIST classification . Our results confirm that our approach significantly outperforms this baseline.

Computer Vision Task results: To evaluate the effectiveness of our method we tested our method on MNIST,CIFAR-10, Fashion MNSIT and mixture of multiple datasets in the unlabeled pool.

MNIST: We conducted on MNIST dataset where size of initial labeled pool is 100 and using budget size of 100 at every stage of active learning. As it is shown in Fig 6.1 our method performed on-par with the rest of exisitng method. As this is a easier computer vision task all the methods performed within the range.

CIFAR-10: We conducted two separate experiments for CIFAR-10 with different budget sizes. We conducted the experiment where the initial labeled pool is of size 5000 and budget(b) is 2500 at each stage. As shown in Fig. 7.4 our proposed method performed on-par with the existing state of art method DAL-OSR and VAAL came in third, with an accuracy of only 80.71% , followed by Core-Set with an accuracy of 80.37%, and then Ensemble w

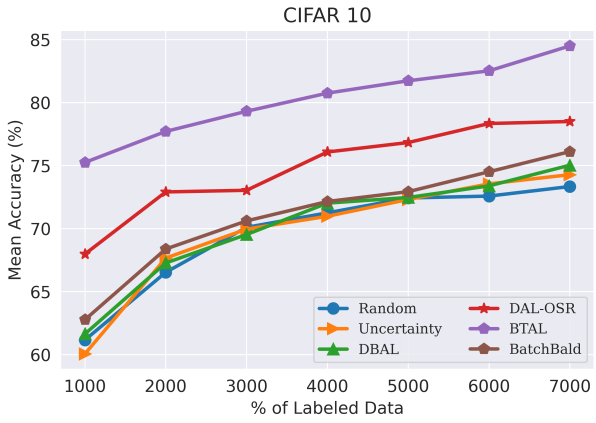


Figure 7.3: Robustness of our approach on CIFAR10 classification . Our results confirm that our approach significantly outperforms this baseline.

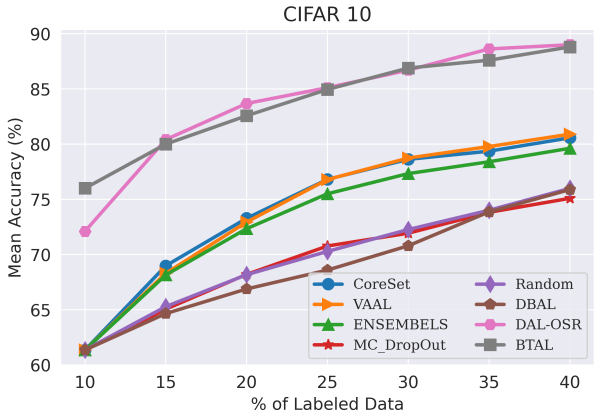


Figure 7.4: Robustness of our approach on CIFAR10 classification . Our results confirm that our approach significantly outperforms this baseline.

VarR at 79.465%. Random sampling, DBAL and MC-Dropout all trailed significantly behind other methods.

To evaluate the effectiveness of the proposed model when compared to other methods for small budgets we designed an experiments where the initial labeled pool is of size 5000 and budget(b) is 1000 at each stage of active learning. As shown in the Fig. 7.3 the proposed method overcomes all the existing methods by a huge margin and DAL-OSR method comes second followed by BAtchBALD and rest of the methods are in similar range. This proves

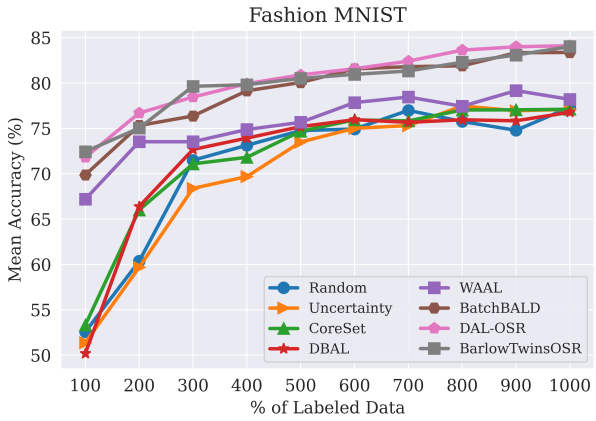


Figure 7.5: Robustness of our approach on CIFAR10 classification . Our results confirm that our approach significantly outperforms this baseline.

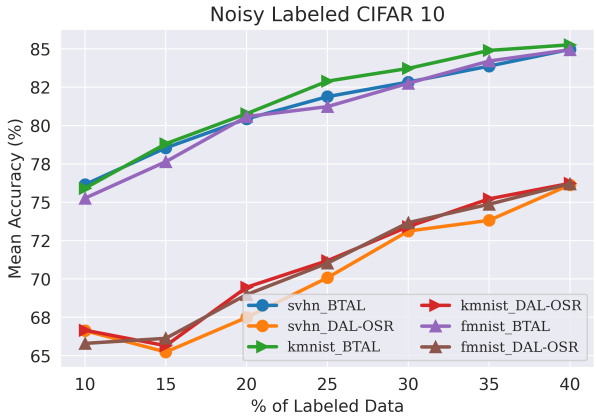


Figure 7.6: Robustness of our approach on CIFAR10 classification tasks when the unlabeled pool includes samples from either the SVHN, KMNIST, or FashionMNIST datasets. The first three curves used the M_2 classifier, while the ones with the 'Random' subscript used random sampling. Our results confirm that our approach significantly outperforms this baseline.

the proposed method is very effective when the budget size if pretty low. The original accuracy which can be achieved using the entire CIFAR-10 dataset was 92.63%.

FashionMNIST: To evaluate the robutness of our approach to different datasets we conducted experiemnt on another standard benchmark dataset FashionMNIST. Similar to previous experiments we compared our method with other exisiting state of the art methods like DAL-OSR, Core-Set (Sener and Savarese, 2017), Deep Bayesian AL (DBAL) (Gal et al.,

2017),WAAL, BatchBALD (Kirsch et al., 2019), and WAAL((Shui et al., 2020)). As shown in the Fig 7.5 our method outperforms all the methods and comes in-par performance with DAL-OSR.

Mixed UnLabeled Pool: Finally, we also tested the extreme case of active learning as proposed in DAL-OSR. We followed the similar setup follwed by DAL-OSR in which 10,000 images from other datasets like SVHN,KMNIST,KMNIST was mixed into source dataset of CIFAR-10. Thus the proposed method should distinguish not only between informative and non-informative samples but also distinguish in-distribution data samples(CIFAR-10) from out-of-distribution samples(SVHN,KMNIST,KMNIST. The better model untilizes the budget well and picks the informative in-dataset samples. Eg: In the case of where the budget is 1000, If the model picks 1000 samples out of which 400 belongs to samples from out-of-distribution dataset. Then only 600 samples are sent for annonatation and add to labeled pool dataset and rest of the 400 samples added back to unlabled pool which makes total of 10,000 out of distribution data samples in the unlabeled pool at every state of active learning. As the active learning increses in stages it makes more and more difficult for the model to pick the in-label samples as unlabeled pool contains less in-label and more out of distribution samples.

7.5 Conclusions and Future work

In this work, we proposed a novel method for Active learning under iid and non-iid shift based unlabeled pool for computer vision based tasks using self-supervised learning technique along with wiebull sampling method. We evaluated our work by extensive com-

parisons with existing methods on three open source datasets. We rigorously benchmarked our method against the state-of-the-art active learning models on computer vision tasks. We also presented different budget based and mixed unlabeled pool setup studies to show the effectiveness of the proposed method with respect to the other methods. The results showed our method consistently performed on par and better than existing baselines on computer vision tasks. For future work, we would like to further explore more effective self supervised methods for handling active learning at scale.

8| FIFTH IDEA: EFFICIENT DOCUMENT IMAGE CLASSIFICATION USING REGION-BASED GRAPH NEURAL NETWORK

Gartner has estimated 80% of enterprises data is unstructured (emails, PDF and other documents). These documents contain rich information and knowledge about internal and external business communication and transactions. And they have ubiquitous applications in numerous industrial sectors such as finance, health care, and law etc. Therefore, being able to automatically and efficiently sort, analyze, and extract structure and content from document images can improve efficiency and reduce cost for many business workflows. Document image classification is an import task in these automation solutions, and has been a popular research area for decades. Early works usually build classifiers that rely on Optical Character Recognition (OCR) to extract text information, and employ heuristics to model layout structural features. In light of the advancement of computer vision and deep learning, VGG-16 Simonyan and Zisserman (2014b) pre-trained on ImageNet Deng et al. (2009) reported good classification performance on data sets mixed of business letters, print advertisement, emails and magazine articles Kumar et al. (2014). Both Denk and Reisswig (2019) and Xu et al. (2019) created document representations by encoding layout coordinates into positional embeddings as inputs to pre-trained BERT Devlin et al. (2018) or transformer architectures. The latest PubLayNet Zhong et al. (2019) addresses the limited public available

document image data sets by training a Mask R-CNN He et al. (2017) model on 360k images of scientific articles, and enables transfer learning to other document domains. Motivated by the development of graph neural network algorithms Wu et al. (2019); Zhang et al. (2018), researchers Liu et al. (2019) attempted to use graph convolutions to model the interactions among structural components of a document and between the visual and textual features, as an alternative to pixel level or token level document modeling.

In contrast to fast moving research progress in document analysis and classification, few have systematically studied the time and hardware resources when using different methods and the financial implications of the model design. However, as document image classifications have been primarily motivated by its potential in commercialization, it is imperative to study its model performance with computing resources requirements and financial implications. In this paper we propose an efficient document image classification framework as shown in Fig. 9.1. Semantics regions of a document is extracted by pre-trained PubLayNet, textual features are extracted by text embedding models and the image features are extracted by a pre-trained VGG-16 model. Graphs formed for the document, with the document class labels are used to train a sort pooling graph convolution network Zhang et al. (2018) which normalizes and classify arbitrary graphs therefore documents. The major contributions of our papers are as follows:

- We propose a novel document image classification framework which applies a graph convolution neural network to a document image graph formed by semantic regions extracted from a pre-trained document segmentation model. Moreover both image and text features of the regions are extracted and assigned to the nodes so that information

from both modalities are captured and propagated in the graph convolutions. To our best knowledge, our framework is the the first in effectively and economically integrating image, text, and layout information for document image classification using a graph convolution neural network.

- We have rigorously bench marked our proposed method against state-of-the-art pre-trained vision models and transformer language models on document image data sets. These include an insurance related document image data set consisted of 11 classes and an open source data set of 10 classes. The results showed the classification results of our method are comparable to those of baseline models, if not better.

- We extensively bench marked the computing resources required by all methods. The results showed our framework needs substantially less computing resources and less time, further indicating the cost advantages of training, deployment and hosting at scale. Efficient model also helps accelerate model iterations and update.

We also discussed a few potential document image classification applications and the infrastructure to deploy our framework. The potentially large scale adoption of document image classification further reinforced the need for an efficient document image classification method.

8.1 Methodology

In this section, we briefly review the advantages and limitations when using either CNNs or large language models in document classification. We also discuss document segmentation and the intuitions of using region based representations. We then describe our proposed

efficient graph neural network, **Eff-GNN**

8.1.1 *Deep Convolution Neural Network Learning Approaches*

When using deep convolutional neural networks for document image classification, the document is treated as an image and is ingested as tensor representing the pixel values of the image. VGG-16 pre-trained on ImageNet can achieve good results on general business documents Das et al. (2018). Even for the insurance data set, VGG-16 pre-trained on ImageNet can be a powerful visual feature extractor.

8.1.2 *Language Model based Approaches*

In general, BERT-like pre-trained language models achieve superior performance on natural language processing tasks by adding self attention mechanism and positional information to the encoder-decoder architecture. When using BERT-like models to classify document images, we classify based on the contextualized embedding of text extracted from image. DocBert Adhikari et al. (2019) assumes syntax features matter less if only the categories of the document need to be decided. And DocBert successfully distills trained BERT into a much smaller LSTM model. This gives us some insight that token level modeling for document classification may not be necessary.

8.1.3 *Document Segmentation*

A business document contains visually salient structural components such as header, footer, paragraph, table etc. Intuitively one can classify a document image by its layout and structural components without accessing much of its content. The regions of structural

components are usually pixels or tokens with similar appearances or groupings. Hence regions are a higher level abstraction and representation which we can leverage to classify this document. Research in the computer vision community has provided plenty of tools of segmenting document images. The latest advancement is PubLayNet which trained a Mask R-CNN model for 360 thousand document images from scientific articles. The segmented region results from PubLayNet

8.1.4 *Efficient GNN for Document Image Classification*

Ideally an effective document classification method need to leverage both textual, image and layout information. However, training or fine tuning CNNs or large language models do not only run into resource constraints (e.g. GPUs, memory), but also prevent fast model iterations. We attempted to address this dilemma by graph representations using graph representations to represent document. We then assign image and text features to the nodes of the graph and apply a graph convolution neural network. Finally we classify the document as classifying a graph.

Details of training our proposed **Eff-GNN** can be found in Algorithm 1. For each image with class label, we extract its text using OCR PyTesseract and we extract its semantic regions using PubLayNet. Each image is converted into a graph where each node is a region. To generate the text feature of the nodes, we use Word2Vec to create the embeddings for words in the region; to generate image feature of the nodes, we extract visual features from that region using VGG-16 pre-trained on ImageNet. Text features and image features of the node can be concatenated and assigned to the nodes. A graph convolution neural network classifier with a SortPooling Zhang et al. (2018) layer is then trained on this data. We

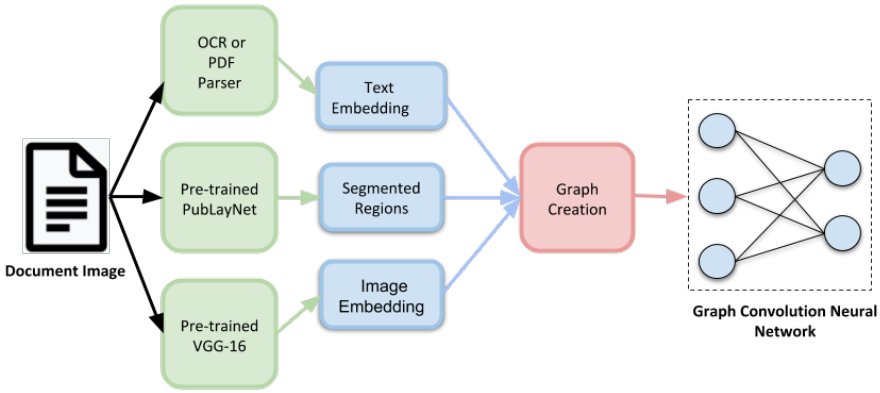


Figure 8.1: **Eff-GNN Framework overview:** textual embedding, segmented regions and image embeddings of an image are integrated when the graph of document is formed. Created graph is fed into the Graph Convolution Neural Network for graph classification as document image classification.

class label	0	1	2	3	4	5	6	7	8	9	10
Insurance	13	5	6	18	5	12	3	11	8	13	11
Tobacco	5	9	3	11	10	5	2	9	10	2	-

Table 8.1: **Median number of nodes per class for the Insurance and Tobacco data sets.**

adapted to this specific graph convolution neural network because it preserve features of individual nodes and also enforces learning from graph global topology.

Till this end we have integrated textual, image and layout information into a document classification task using a graph convolution neural network.

8.2 Experimental Setup

8.2.1 *Datasets*

For the Insurance dataset, we use the splits of 4544 images for train set and 1280 images for test set; for Tobacco-3482 dataset, we use the standard splits of 2482 images for training,

800 images for testing, and rest 200 images for validation set.

8.2.2 *Document Pre-processing*

To construct a graph for each document image we need to utilize the different layout regions' information such as paragraphs, title, list, table present in each document. We process the scanned document images using pre-trained PubLayNet to obtain the necessary bounding boxes for each region. To extract the textual information present in the bounding box of each region in the document images we apply PyTesseract, an open-source python OCR package. All the results of the PubLayNet and PyTesseract were then serialized and stored in tensor format using PyTorch. Paszke et al. (2019).

8.2.3 *Hyper parameters and infrastructure*

We use Hedwig[1], an open-source deep learning toolkit with a number of implemented of document classification models. We use a Tesla K80 GPU for all models requiring GPU for train, and use amazon EC2-t2.micro and EC2-c type machine when only CPU is needed . We use PyTorch 1.5 as the backend framework, and gensim Řehůřek and Sojka (2010) package for computing the node feature vectors using word2vec.

[1]https://github.com/castorini/hedwig

8.3 Results and Discussions

8.3.1 *Comparing classification accuracy*

We compare our proposed approach with the state-of-the-art deep learning models, VGG-16 pre-trained on ImageNet and BERT$_{base}$ pre-trained on Wikipedia. When using BERT$_{base}$ for classification, we extracted tokens from document images as input and fine tune BERT$_{base}$ with class labels. When using pre-trained VGG-16, we used it directly to extract document image features for classification. No further fine-tuning is used. We also compared our model with DocBERT which is specially designed for document level classification by simplifying BERT models with knowledge distillation.

Table 8.2 shows the model comparison results of classification AUC on the two data sets. In the insurance data set, our proposed model achieves 90.7% to 91.0 %, very competitive as compared to models in BERT families (91.95 %) and VGG-16 (90.6 %). In the Tobacco-3482 data set, our models achieved 73.5 % to 77.5 % , comparable to models in BERT families (82.3 %) and VGG-16 (81.5 %). The fact our proposed model shows more advantages when classifying the insurance data set could be due to the high intra-class variance and low inter-class variance in the Tobacco-3482 data set Kölsch et al. (2017).

We also experimented with combining text and image embedding features as node features in the graph neural network. The combination does provides ample AUC improvements in our proposed method on Tobacco-3482 dataset i.e. 73.5 % for Eff-GNN + Word2Vec and for 77.5 % for Eff-GNN + Word2Vec + Image Embedding. In the insurance data set, Eff-GNN + Word2Vec + Image Embedding shows little improvement over using Word2Vec text features alone. That could be due to the fact that both the textual and image content in

the insurance data set provides enough information for classification. This assumption can be further justified by the similar classification results achieved by models in BERT family and VGG-16.

In addition to classification performance, we compare the number of trainable parameters of each model. In both the insurance data set and Tobacco-3482, our model size is drastically smaller than models in BERT families and VGG-16. We calculate the parameters of our model as the sum of parameters in the graph neural network and the parameters in trained word2vec. The small sizes of graph neural network model (Table 8.1) results in only 160,000 parameters. The Word2Vec model is also relatively light weight because each of the data set contains a very limited vocabulary. Note we did not include the 44.2 million parameters of PubLayNet, because in our framework we do not train or fine-tune any parameters of the PubLayNet.

8.3.2 *Comparing computing resources*

We also bench marked the time and memory required for training our proposed Eff-GNN against other models. Table 8.3 reports the statistics on the insurance data set, 4544 images for training and 1280 images for inference. Eff-GNN models take less than 5 minutes to train 50 epochs whereas VGG-16 or models in BERT Family take hours to train less number of epochs (15 epochs and 28 epochs respectively). In particular Eff-GNN can run on CPU alone and its model training time is comparable to its GPU counterpart. This is consistent with the small size of our model (See Table 8.2, "Parameters" column). Eff-GNN can achieve these advantages because it models the documents using a graph formed by regions extracted by PubLayNet. The time of using PubLayNet to extract regions for training

images are negligible. The size of the resulting graph leads to a small model compared to deep models trained on pixel level information or transformers trained on token level information. Therefore Eff-GNN only uses 470MB in GPU memory with additional 3.5 GB for using PubLayNet. Consequently Eff-GNN requires drastically less time for the inference of 1280 images (0.79 seconds) as compared VGG-16 (103 seconds) and BERT (40 seconds). Note the time of document pre-porcessing steps such as OCR and training Word2Vec model are not included in the table. Although these two steps are extra for our proposed framework, we contend that their addition does not nullify the efficiencies gained through graph neural nets. Even BERT based models require OCR extraction pre-processing step. Just one Word2Vec needs to be trained for the entire data sets and OCR can be optimized by e.g. parallel processing.

Compared with the SOTA pre-trained large models, our proposed Eff-GNN framework achieved competitive classification results on our insurance document image data sets, and achieved comparable results on the the open source Tobacco-3482 data set. We also showed that combining text and image information as the node features in our graph neural network can be advantageous when OCR fails to extract text information or when the two modalities are complimentary. Our proposed method models document representations using extracted semantic regions, instead of using token level or pixel level information. Therefore our model size is dramatically less than other methods, and can be run on CPU machines.

8.4 Applications and Deployment

Document image classification can find many enterprise level applications in insurance companies to reduce manual review and stream line claims processes. For example, casualty

Data Set	Model	AUC	# Parameters
Insurance	DocBert Adhikari et al. (2019)	91.95 %	110M
	BERT Devlin et al. (2018)	91.95 %	110M
	VGG-16 Simonyan and Zisserman (2014b)	90.6 %	130M
	Eff-GNN + Word2Vec Mikolov et al. (2013)	91.0 %	124k + 610k
	Eff-GNN + Word2Vec Mikolov et al. (2013) + Image Embedding	91.0 %	126k + 610k
Tobacco-3482	VGG-16 Simonyan and Zisserman (2014b)	81.5%	130M
	DocBERT Adhikari et al. (2019)	82.3 %	110M
	BERT Devlin et al. (2018)	79.0 %	110M
	Eff-GNN+ Word2Vec Mikolov et al. (2013)	73.5 %	124k + 610k
	Eff-GNN + Word2Vec Mikolov et al. (2013) + Image Embedding	77.5 %	126k + 610k

Table 8.2: **Classification accuracy on the Insurance, Tobacco-3482 dataset.**

Insurance	Batch Size	Epochs	Training Time	GPU Memory	Inference Time
VGG	32	28	2.30 hours	7.08GB	103 seconds
Eff-GNN (GPU)	32	50	3.5 mins.	470MB + 3.5 GB	0.79 seconds
Eff-GNN (CPU)	32	50	4.1 mins.	NA	0.79 seconds
BERT	16	15	6.2 hours	10.5GB	40 seconds
DocBERT	16	15	6.3 hours	8.1GB	"40 times faster than BERT" Adhikari et al. (2019)

Table 8.3: **GPU Memory(training), hardware and time required by different models on the Insurance dataset. The numbers are reported for training 4544 images and inference for 1280 Images.**

injuries claims usually have multiple correspondences with hospitals and clinics, each involving documents to be processed by the insurance company. Personal automobile claims processing can use automatic document classification of letters of guarantee, purchase receipts, and others in order to auto-approve a certain percentage of auto claim reimbursements. Each year, millions of claim related document images are sent to insurance companies belonging to hundreds of categorizes related to financial institutions, medical providers, or legal organizations. A scalable document classifier can assign the correct categories as meta data to a document, which can be used to route to appropriate downstream tasks. Given the scale document image classification can be deployed in an enterprise, an efficient framework for model training and iterations, together with an economical model hosting solutions has clear cost advantages.

To support all those applications and achieve scalability, we have deployed this trained model using AWS Serverless Application Model framework as an API via AWS Lambda. We have packaged the model artifacts and inference code in a Docker image, and used the container image support functionality to deploy this to as a Lambda function, surfacing this all through AWS API Gateway. This will host the model behind an Amazon API Gateway, which serves as the front end and handles authentication when user submit the request. All the deployments, security permissions, and advanced configuration capabilities are doing using YAML.

8.5 Conclusion:

Millions of business document images, such as medical bills, attorney letters, contracts, bank statements and personal checks are processed in insurance companies to support a wide range of business workflows and applications. A scalable and efficient automation that is more intelligent than the brittle OCR-template based method is desired.

In this paper we proposed a novel document image classification that uses graph convolution neural network to integrate text, image, and layout information of a document. We rigorously bench marked our method against the SOTA computer vision and language models on both the insurance dataset and Tobacco dataset. We also compared computing time and hardware resources required for training those models. The results showed our method is not only competitive on classification performance but also is much smaller in size therefore requires much less time and resource. This could translate to big cost advantages of hosting and deployment in real world applications. We are also working on enabling general document classification that can handle hundreds of document classes. A few options include

training larger models for domain specific transfer learning, enabling few shot learning and continual learning when dynamically adding new document classes. In addition, we would like to further explore more effective document representations including more sophisticated graph representations or jointly trained layout Xu et al. (2019).

9| SIXTH IDEA: DOMAIN AGNOSTIC FEW-SHOT LEARNING FOR DOCUMENT INTELLIGENCE

The generalisation performance of a convolutional neural networks (CNN) is majorly predisposed by the quantity, quality, and diversity of the training images. All the training data needs to be annotated in-hand before, in many real-world applications data is easy to acquire but expensive and time-consuming to label. The goal of the Active learning for the task is to draw most *informative samples* from the unlabeled pool which can used for training after annotation. With total different objective, self-supervised learning which have been gaining meteoric popularity by closing the gap in performance with supervised methods on large computer vision benchmarks. self-supervised learning (SSL) these days have shown to produce low-level representations that are invariant to distortions of the input sample and can encode invariance to artificially created distortions, e.g. rotation, solarization,cropping etc. self-supervised learning (SSL) approaches rely on simpler and more scalable frameworks for learning. In this paper, we unify these two families of approaches from the angle of active learning using self-supervised learning mainfold and propose **D**eep **A**ctive **L**earning using **Barlow Twins** (DALBT), an active learning method for all the datasets using combination of classifier trained along with self-supervised loss framework of Barlow Twins to a setting where the model can encode the invariance of artificially created distortions, e.g. rotation,

solarization,cropping etc.. We propose to use joint loss function which consist classifier loss and self-supervised loss borrowed from Barlow twins to jointly learn an encoder that produces representations invariant across such pairs. DALBT is a method that is simple, easy to implement and train, and of broad applicability. We carried out an extensive evaluation of our novel proposed method of active learning, achieving state-of-the-art results on MNIST, Fashion-MNSIT, CIFAR-10. Additionally, to show the robustness of the proposed model we also showed the results on where the unlabeled pool consists of a mixture of samples from multiple datasets, proposed model can successfully distinguish between samples from seen vs. unseen datasets.

Few-shot learning aims to generalize to novel classes with only a few samples with class labels. Research in few-shot learning has borrowed techniques from transfer learning, metric learning, meta-learning, and Bayesian methods. These methods also aim to train models from limited training samples, and while encouraging performance has been achieved, they often fail to generalize to novel domains. Many of the existing meta-learning methods rely on training data for which the base classes are sampled from the same domain as the novel classes used for meta-testing. However, in many applications in the industry, such as document classification, collecting large samples of data for meta-learning is infeasible or impossible. While research in the field of the cross-domain few-shot learning exists, it is mostly limited to computer vision. To our knowledge, no work yet exists that examines the use of few-shot learning for classification of semi-structured documents (scans of paper documents) generated as part of a business workflow (forms, letters, bills, etc.). Here the domain shift is significant, going from natural images to the semi-structured documents of interest. In this work, we address the problem of few-shot document image classification under domain

shift. We evaluate our work by extensive comparisons with existing methods. Experimental results demonstrate that the proposed method shows consistent improvements on the few-shot classification performance under domain shift.

9.1 Related Work

This work explores a method which can be used for few-shot learning on multi-channel document data, in which the meta-training is done on a distinct domain of open source documents. Work has been done in this area addressing the separate problems of:

1. Meta-training a few-shot model on document data,
2. Combining visual and textual feature channels via canonical correlation, and
3. Domain adaptation of models trained on image data.

This work proposes to address all of these issues with a single approach, driven by a distinct business need. Here we detail previous work done in each of these directions.

9.1.1 *Meta-learning*

Meta-learning has been a powerful tool to answer the challenge of the large data requirements that many deep learning models seem to face. So far, many of the applications of deep meta-learning have been in few-shot image classification Finn et al. (2019); Ravi and Larochelle (2016); Snell et al. (2017). Meta-learning for few shot image classification has often been evaluated on data sets such as ImageNet Russakovsky et al. (2015), CIFAR-10 and CIFAR-100 Krizhevsky (2009b), and Omniglot Lake et al. (2011). However, there has been little done to apply the methods of meta-learning to industry level document images.

9.1.2 *Canonical Correlation*

One aspect of enterprise level document images is that they typically have two feature channels; a visual channel, and a text channel. Each has useful information that can be leveraged for document classification. However, a challenge to overcome with this is that typically pre-trained models are used as feature extractors, which are further fine tuned during meta-training. The vectors extracted by these pre-trained models (one for each channel mentioned above) are typically not the same length, and encode the information of the channel in semantically different ways. One method of overcoming this challenge is the use of Canonical Correlation Akaho (2006); Andrew et al. (2013); Bach and Jordan (2002); Hotelling (1992); Melzer et al. (2001), which in a sense aligns two vectors via projections in such a way that the projections are maximally correlated. We use a later iteration of this method called Deep Canonical Correlation Andrew et al. (2013). This allows us to efficiently combine the two modalities of visual and text features occurring in enterprise document images for the purpose of few-shot meta-learning.

9.1.3 *Domain adaptation*

In the traditional machine learning setting, the data samples used for training and testing an algorithm are assumed to come from the same distribution. In practical applications however, this is not always a valid assumption; the data available for training may fall into a different distribution than the data the model is expected to perform on in a live system. A typical example of this is a model which is trained on an open source data set is then desired to be used for inference in a smaller, proprietary data set, perhaps for a

slightly different task. Domain adaptation is a subfield of machine learning that attempts to overcome this challenge. Typical approaches include transfer learning Pan and Yang (2010), semi-supervised learning Pise and Kulkarni (2008); Zhu (2008), multi-task learning Caruana (2004), and meta-learning Huisman et al. (2021). Recent methods include (Liu et al., 2020) which uses feature transformation ensemble model with batch spectral regularization, (Cai and Shen, 2020) aims train specific layers of network using MAML, (Jiang et al., 2020) uses a new prediction head and global classification network based on semantic information for addressing the cross-domain adaptation.

9.2 Methodology

In this section, we briefly formally define the few-short learning and cross-domain few shot learning problems. We then describe our proposed method, Cross Domain Few-Shot Learning using **D**eep **C**anonical **C**orrelation for **D**ocument **I**ntelligence dubbed as **DCCDI** in later parts of the paper.

9.2.1 *Formal problem definition*

Formally, a few-shot learning problem is denoted as $P = (\mathcal{D}_{\text{source}}, \mathcal{D}_{\text{target}})$; $\mathcal{D}_{\text{source}}$ is the meta-train set from where base classes as sampled for episodic training. Novel classes during meta-test are sampled from the target domain $\mathcal{D}_{\text{target}}$, such that $D_{\text{source}} \cap D_{\text{target}} = \emptyset$. For brevity, we will define the ***domain*** of the source dataset $\mathcal{D}_{source}$ to be

$$d_{source} = \{\mathcal{X}, \mathcal{Y}, P_{source}\} \tag{9.1}$$

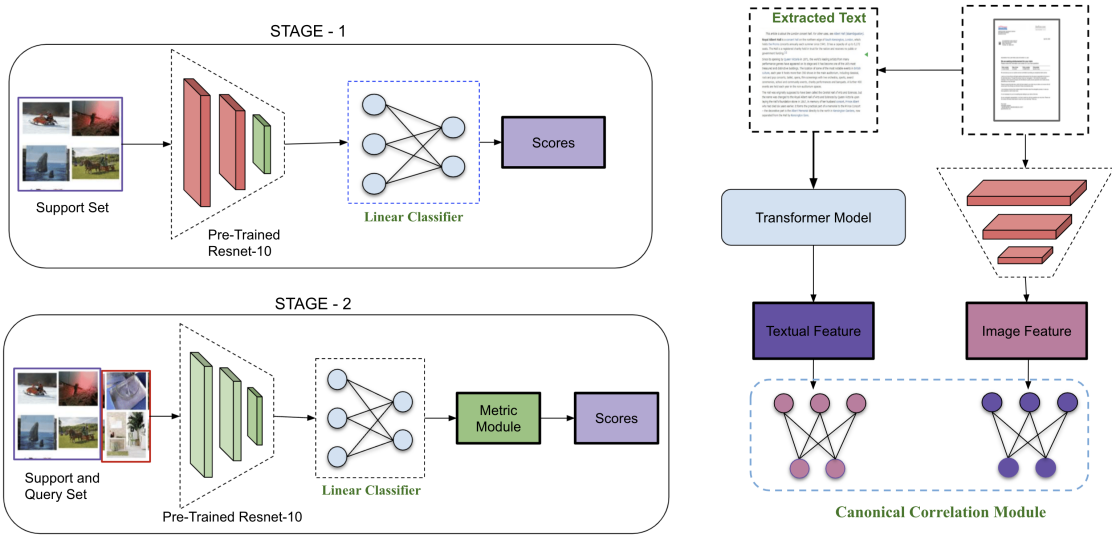


Figure 9.1: The overall architecture of our approach. LEFT(Stage-1): Using episodic training paradigm train the last k layers of ResNet-10 (shown in green color) using Cross-entropy loss by support set . Left(Stage-2): Using episodic training paradigm train all layers of ResNet-10 using Cross-entropy loss by support, Query set and by including the Metric-Learning Module. During Meta-testing all the Resnet-10 layers will be fixed. RIGHT: Canonical Correlation Block: During Meta-testing all the text extracted from document image and both image, textual features was trained using canonical correlation loss

where $\mathcal{X}$ is the feature space of all the inputs in d-dimensional space; $\mathcal{X} \subset \mathbb{R}^d$ and $\mathcal{Y}$ is the label space of all the labels; $\mathcal{Y} \subset \{1, \ldots C\}$ where is the number of classes, and P_{source} is the joint probability distribution over the feature,label pairs of $\{\mathcal{X}, \mathcal{Y}\}$ denoted by $p(x, y)$. A similar definition can be made for the target dataset.

We focus on few-shot settings for document classification using a model f with parameters θ, wich we will denote f_θ, via meta learning tasks using episodic training from the meta-train set $\mathcal{D}_{source}$ and aim to demonstrate generalization to novel classes present in the meta-test set $\mathcal{D}_{target}$.

During meta training the model f_θ is provided with a wide range of classification tasks $\mathcal{T}_i$

drawn from the dataset $\mathcal{D}_{\text{source}} = \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$ where each episodic task is $\mathcal{T}_i = \{(x_1^i, y_1^i), \ldots, (x_k^i, y_k^i)\}$, and where x_i represents image i and y_i its corresponding label.

Each task $\mathcal{T}_i$ is further partitioned into a *support set* $\mathcal{S}_i$ used for training, and a *query set* $\mathcal{Q}_i$ used for testing. That is, $\mathcal{T}_i$ can be written as the disjoint union $\mathcal{T}_i = \mathcal{S}_i \dot{\cup} \mathcal{Q}_i$. Overall, $\mathcal{D}_{source}$ can be written as $\mathcal{D}_{\text{source}} = \{(\mathcal{S}_1, \mathcal{Q}_1), \ldots, (\mathcal{S}_n, \mathcal{Q}_n)\}$.

We follow the conventional way of preparing the support and query sets for each task ($\mathcal{T}_i$), which is a C-way, N-shot classification problem in which C classes are randomly drawn from the entire set of classes from $\mathcal{D}$. Furthermore, for each of the sampled classes, N and M examples are sampled for the support and query set respectively such that each task $\mathcal{T}_i$ consists of $(\mathcal{S}_i, \mathcal{Q}_i)$ where $\mathcal{S}_i = \{(\mathbf{x}_i, y_i)\}_{i=1}^{C \times N}$ is a support set consisting of N labeled images for each of the C classes and the query set $\mathcal{Q}_i = \{\tilde{\mathbf{x}}_i, \tilde{y}_i\}_{i=1}^{C \times M}$ with M samples per class and $y, \tilde{y} \in \{1, \ldots, C\}$ are the corresponding class labels.

The cross-domain few-shot learning scheme matches closely with our real-world industry setting where the source domain $\mathcal{D}_{source}$ and the target domain $\mathcal{D}_{target}$ belong to different distributions. As in Eq. 9.1, the joint distribution of the source dataset is indicated as P_{source} and the target domain distribution can be denoted as P_{target}. Furthermore, as is the case in a cross-domain few-shot learning setting, $P_{source} \neq P_{target}$ and $\mathcal{Y}_s$ is disjoint from $\mathcal{Y}_t$. Also, similarly to a few-shot learning paradigm, during the episodic meta-training phase, the model is trained on a large number of tasks $\mathcal{T}_i$ sampled from the source domain $\mathcal{D}_{source}$.

During the meta-testing phase, the model is presented with a support set $S = \{x_i, y_i\}_{i=1}^{K \times N}$ consisting of N examples from K novel classes and $\mathcal{Q} = \{x_i, y_i\}_{i=1}^{K \times M}$ consisting of M examples which are very different from the meta-training classes.

After the meta-trained model $\hat{f}_\theta$ is adapted to the support set, a query set from novel

classes is used to evaluate the model performance.

9.2.2 *Canonical Correlation*

Ideally, an effective document classification method needs to leverage both textual and image (including layout) information. When using deep convolutional neural networks for document image classification, the document is treated as an image and is ingested as tensor representing the pixel values of the image to get the visual feature vector of the document images.On the other hand, all the text from the document image is extracted, converted into tokens and passed through a BERT-like pre-trained transformer-based language model to obtain textual features. Some of the ways of utilizing both the textual and visual features during the classification are to concatenate or average them before passing them through the final classification layer. Some of the disadvantages of doing this are a) More computational resources are required for model training for large dimensional features b) Difficult to maintain the synchronization between both the visual and textual modalities, which might impact model performance.

We address this dilemma by introducing the Deep Canonical Correlation for Document Intelligence Module (DCCDI) during meta-test phase to represent a document utilizing both the textual and visual features. By using the proposed DCCDI module, we produce highly correlated transformations of multiple modalities of data such as textual and visual using complex non-linear transformations. Canonical correlation was proposed by Hotelling (Hotelling, 1992). It is a widely used technique in the statistics community to measure the linear relationship between two multidimensional variables, used in finding linear projections of two multidimensional vectors that are maximally correlated. Later on it was applied to ma-

chine learning by different researchers (Akaho, 2006; Andrew et al., 2013; Bach and Jordan, 2002; Melzer et al., 2001). We use deep canonical corelation method propsed by (Andrew et al., 2013) in our DCCM module with the goal of achieving fine-grained cross-modality alignment between the visual and textual modalities.

As shown in RIGHT Figure 9.1; $V \in \mathbb{R}^{N \times d_1}$ is the multidimensional vector for the visual modality where d_1 is total number of dimensions and $T \in \mathbb{R}^{N \times d_2}$ is the multidimensional vector for the textual modality where d_2 is total number of dimensions. N is the total number of inputs available in each modality. The input multidimensional vectors in different modalities are transformed using two neural networks g with parameters ϕ_1, h with parameters ϕ_2

$$\mathcal{Z}_1 = g_{\phi_1}(V), \;\; \mathcal{Z}_2 = h_{\phi_2}(T) \tag{9.2}$$

$\mathcal{Z}_1 \in \mathbb{R}^{N \times d}$ and $\mathcal{Z}_2 \in \mathbb{R}^{N \times d}$ are the d dimensional outputs of the neural networks. Both the neural networks g, h are optimized jointly with a goal of making the correlation between $\mathcal{Z}_1$ and $\mathcal{Z}_2$ as high as possible:

$$(\phi_1^*, \phi_2^*) = \arg\max_{\phi_1, \phi_2} \text{corr}(g_{\phi_1}(V), h_{\phi_2}(T))$$

The above equation can be solved in multiple ways. For this work we chose an approach suggested by (Martin and Maes, 1979) and that utilizes Singular Value Decomposition. Define the centered output matrices by $\bar{\mathcal{Z}}_i = \mathcal{Z}_i' - \frac{1}{N}\mathcal{Z}_i'\mathbf{1}$. Then define

$$\hat{\Sigma}_{11} = \frac{1}{d-1}\bar{\mathcal{Z}}_1\bar{\mathcal{Z}}_1' + r_1 \tag{9.3}$$
$$\hat{\Sigma}_{22} = \frac{1}{d-1}\bar{\mathcal{Z}}_2\bar{\mathcal{Z}}_2' + r_1$$
$$\hat{\Sigma}_{12} = \frac{1}{d-1}\bar{\mathcal{Z}}_1\bar{\mathcal{Z}}_2'$$

where $r_1 > 0$ is a regularization constant. As discussed in Andrew et al. (2013), the correlation of the top k components of $\mathcal{Z}_1$ and $\mathcal{Z}_2$ is the sum of the top k singular values of the matrix $T = \hat{\Sigma}_{11}^{-1/2}\hat{\Sigma}_{12}\hat{\Sigma}_{22}^{-1/2}$. If we take $k = d$, then this is exactly the matrix trace norm of T; $\text{corr}(\mathcal{Z}_1, \mathcal{Z}_2) = \|T\|_{\text{tr}} = \text{tr}(T'T)^{1/2}$.

Both the networks parameters are updated by computing the gradient of $\text{corr}(\mathcal{Z}_1, \mathcal{Z}_2)$ and update the parameters using backpropagation. If the singular value decomposition of T is $T = UDV'$, then

$$\frac{\partial \,\text{corr}(\mathcal{Z}_1, \mathcal{Z}_2)}{\partial \mathcal{Z}_1} = \frac{1}{d-1}(2\nabla_{11}\bar{\mathcal{Z}}_1 + \nabla_{12}\bar{\mathcal{Z}}_2) \tag{9.4}$$

where

$$\nabla_{12} = \hat{\Sigma}_{11}^{-1/2}UV'\hat{\Sigma}_{22}^{-1/2}$$

and

$$\nabla_{11} = -\frac{1}{2}\hat{\Sigma}_{11}^{-1/2}UDU'\hat{\Sigma}_{11}^{-1/2}$$

9.2.3 *DCCDI Model*

The meta-training approach of our proposed **DCCDI** method can be found in Algorithm 1. Our primary focus in this work is to improve the generalization ability of few-shot classification models to unseen domains by learning a prior on the model weights which is suitable for Meta-Fine-tuning during the meta-testing phase on document datasets. We have also proposed a canonical-correlation-based layer in the model to integrate effectively both the textual and visual modalities of the document images which can be seen as a fine-grained cross-modality alignment task.

Domain Agnostic Meta-Learning for Document Intelligence

Our focus in this work is to improve the generalization ability of our meta-trained model to arbitrary unseen document intelligence domains. The Meta-training that incorporates **CCDI** its described in detail in Algorithm 1.

We aim to train a model that can adapt swiftly to novel unseen classes. This problem setting is often formalized as cross domain few-shot learning. In this proposed approach, the model is meta-trained on a set of tasks generated using $\mathcal{D}_{source}$, such that the meta-trained model can quickly adapt to new unseen novel tasks using only a small number of examples or trials generated using $\mathcal{D}_{target}$. In this section, we formally state the problem and present the general form of our algorithm. Similar to Meta-Learning algorithms the proposed algorithm can be subdivided into following phases.

Meta-Training Phase

We used ResNet-10 as our visual feature extractor or encoder. It have been shown recently that this pre-training process significantly improves the generalization (Gidaris and Komodakis, 2018; Lifchitz et al., 2019; Rusu et al., 2018). We pre-train the visual feature encoder on a source dataset (miniImageNet or tiredImageNet) by incorporating a final linear layer.

After the pre-training stage, we start our meta-training process of few-shot classification training stage. First, we train and update the last k layers of the visual feature encoder E followed by a linear classifier layer. We minimize the standard cross-entropy loss on the meta-training dataset by using only the support set images as shown in the Stage-1 of Figure 9.1. After this Stage-1 training process, all the layers of the visual feature encoder block of the model f_θ will be unfrozen.

In the Stage-2 phase, we train the proposed model using the traditional episodic meta-learning paradigm. For each episode a new task $\mathcal{T}_i$ is sampled from $\mathcal{D}_{source}$, the model f_θ is trained with N samples present in the support set. The model is then tested on query samples from the same task. The prior parameters of the model f are then updated by considering the test error on the query set. Actually, the test error on sampled tasks $\mathcal{T}_i$ serves as the training error of the meta-learning process. All the parameters in the network are updated using the MAML (Finn et al., 2017b) first order approach. For this stage-2, we proceed similarly to (Cai and Shen, 2020; Chen et al., 2021; Guo et al., 2020) which successfully use a metric mapping module to project the final linear classifier scores onto a metric space which can be used to compare support and query samples, hence increasing the

overall accuracy. A graph neural network is used for the Metric-Learning module which is similar to architecture used in few-shot graph neural networks (Garcia and Bruna, 2018).

Meta-Testing or Meta-Deployment Phase

At the start of the meta-test phase the first l layers of the visual feature extractor was frozen and the last k layers are left unfrozen. With the main goal of adapting the meta-trained model for the business document domain, we introduce the CCDI module during our deployment phase. During Meta-Testing, a new task $\mathcal{T}_i$ is drawn from the $\mathcal{D}_{target}$. The input document images are resized to 224 $\times$ 224 then fed into the visual feature blocks. Visual features are extracted for each of the document image present in the support set using the visual feature encoder block from the meta-trained model $\hat{f}_\theta$. Similarly for each of the document images, text is extracted using Pytesseract and then fed into pre-trained longformer model (Beltagy et al., 2020) to extract textual embedding features. Both the textual and visual modalities are passed through its corresponding deep Canonical Correlation bock and jointly optimized. Training the canonical co-relation block results in representations that aligns both the modalities (Image and text). The resulting meta-trained model along with the metric module, which consists of an ensemble of a graph convolution neural network classifier and a linear classifier layer is then trained on this data. Finally both the scores are combined and treated as the final classification scores.

9.3 Experiments and Results

In this section, we first introduce the different document image datasets used in our training and evaluation process, then show quantitative comparisons against other baseline

Algorithm 4 DCCDI Meta-Test Protocol

Require: $p(\mathcal{T}) \leftarrow \mathcal{T}_{1..n}$; n meta-testing tasks generated from $\mathcal{D}_{target}$ in the form of K shot-N-way classification

Require: $\hat{f}_\theta \leftarrow$ meta-trained model

Require: $\hat{f}_{bert} \leftarrow$ pre-trained longformer model

Require: $g_{\phi_1}, g_{\phi_2} \leftarrow$ Canonical Corelation Module

Meta-Test

for *i=1,...,n* **do**

Sample Task $\mathcal{T}_i$ from $p(\mathcal{T})$

Sample N samples from support set $\mathcal{S}$ and M samples from query set $\mathcal{Q}$.

Obtain visual, texual modality $\mathcal{Z}_1 = g_{\phi_1}(\hat{f}_\theta(\mathcal{S}))$ and $\mathcal{Z}_2 = g_{\phi_2}(\hat{f}_\theta(\mathcal{S}))$ Train the canonical co relation module for 20 steps using canonical loss as shown in Eq 9.4 $\nabla_{\phi_1,\phi_2}\mathcal{L}_{\mathcal{T}_i}(\mathcal{Z}_1, \mathcal{Z}_2)$ by following the

Concatenate the visual features and low dimensional textual features found using the trained canonical co-relation module $\hat{g}_{\phi_1}, \hat{g}_{\phi_2}$

Feed the concatenated feature vectors through feature extractor and then through the metric learning module

Calculate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(\hat{f}_\theta(\mathcal{Q}))$ using the query set samples

Compute adapted parameters with gradient descent and update the model parameters.

methods in the following sections. The code will be attached as supplementary material and will be released publicly after the conference.

9.3.1 *Datasets*

Robustness of the proposed approach has been tested using standardized few-shot classification datasets **miniImageNet** (Ravi and Larochelle, 2016), **tieredImageNet** as the single source domain, and evaluate the trained model on two document domain datasets as target domain. The two training datasets are MiniImageNet dataset which is a subset of bigger ImageNet derived from ILSVRC- 2012 (Russakovsky et al., 2015) which is used in our meta-training process consists of 60,000 images from 100 classes, and each class has 600 images in which 64 classes for training, 16 classes for validation, 20 classes for the test.

		INSR Dataset (5-way)			RVL Dataset (5-way)		
Baselines	**Embedding Net**	**5-shot**	**10-shot**	**20-shot**	**5-shot**	**10-shot**	**20-shot**
ProtoTypical Networks	Conv-4	57.97 %	62.21 %	65.4%	44.64%	48.85 %	53.42 %
ProtoTypical Networks	Resnet-10	50.04 %	53.33%	52.78%	49.50 %	52.58 %	53.25 %
Relational Networks	Conv-4	55.47 %	56.58 %	57.27%	41.28 %	46.93 %	49.10 %
Relational Networks	ResNet-10	32.42 %	43.08%	46.53%	33.08 %	34.68 %	37.57%
Matching Networks	ResNet-10	48.53 %	50.21%	58.92%	40.57 %	44.76 %	52.21%
DCCDI without CCA	ResNet-10	**65.21** %	**70.93**%	**77.2**%	**60.85** %	**66.45** %	**70.32** %
DCCDI	ResNet-10	**67.82** %	**72.79** %	**79.78**%	**61.76** %	**67.40** %	**72.93** %

Table 9.1: **Few Classification accuracy on the INSR, miniRVL dataset when source domain is miniImagenet.**

		INSR Dataset (5-way)			RVL Dataset (5-way)		
Baselines	**Embedding Net**	**5-shot**	**10-shot**	**20-shot**	**5-shot**	**10-shot**	**20-shot**
ProtoTypical Networks	Conv-4	61.09 %	61.41 %	65.18%	44.42%	46.94 %	54.50 %
ProtoTypical Networks	Resnet-10	52.21 %	52.52%	53.76%	46.86 %	46.18 %	52.30 %
Relational Networks	Conv-4	50.28 %	56.82 %	60.29%	41.30 %	48.61 %	47.61 %
Relational Networks	ResNet-10	29.36%	39.36 %	54.82%	26.58 %	29.81 %	44.29 %
Matching Networks	ResNet-10	40.09 %	45.88%	48.10 %	33.68%	42.37 %	43.92%
DCCDI without CCA	ResNet-10	**65.73** %	**71.75**%	**77.11**%	**61.32** %	**66.92** %	**71.04** %
DCCDI	ResNet-10	**66.68** %	**74.01** %	**78.85** %	**62.05** %	**67.55** %	**72.61** %

Table 9.2: **Few Classification accuracy on the INSR, miniRVL dataset when source domain is tieredImageNet.**

We have used 64 classes using our meta-training phase for episodic training of the proposed model. The TieredImageNet dataset is another dataset derived from ImageNet, it consists 608 classes in total derived from 34 high-level categories. These categories are split into 20 meta-training, 6 meta-validation and 8 meta-test , which corresponds to 351 train classes, 97 validation classes and 160 test classes respectively. We have used 351 train classes for the training the model using episodic training.

We use the following two datasets to evaluate our proposed model. The Insurance company dataset, dubbed as **INSR** for anonymity contains 5772 document images which spans across 11 categories. Some Categories from the INSR dataset include Medical Bills, Medical Authorizations, Medical Records etc.

The second dataset is a few-shot learning dataset dubbed as The **miniRVL** dataset

Harley et al. (2015) that has been generated from a larger RVL dataset which consists of 400,000 images which spans across 16 categories. The data relates to document classification and include Advertisements, Emails among other document types. The **miniRVL** dataset consists of16 classes with 1000 images per class and it is designed to keep the inter-class similarity sufficiently high to purposely pose a few-shot learning document classification challenge.

9.3.2 *Document Pre-processing*

To construct the textual features for each document image, we use PyTesseract to extract all the text present in the document. All the extracted text is then passed through longformer model pre-trained on *longformer-base-4096* (Beltagy et al., 2020) and the textual feature vector is then collected for each image. We have used longformer models due to it's increased capacity to handle documents of thousands of tokens or longer as we want to utilize all the textual information present in the document images.

For all the experiments, we measured performance by computing the average accuracy across 3 independent runs.

9.3.3 *Implementation Details:*

We compared our method to three metric-based learning methods: Matching Networks (Vinyals et al., 2016), Relation Networks (Sung et al., 2018), and Prototypical Networks (Snell et al., 2017). We also compared the proposed method using visual and textual features to the proposed method using only visual features. Prior to our meta training phase we also pre-trained our image feature extractor by minimizing the standard supervised cross-entropy

loss on the source domain dataset such as miniImageNet or TieredImageNet. This is similar to several recent works (Gidaris and Komodakis, 2018; Lifchitz et al., 2019; Rusu et al., 2018) that have shown significant improvement in classification accuracy via this method. In all our experiments, we used ResNet-10 model as the backbone of the visual encoder. We also included ablation studies to see the effect of our proposed method for various hyper parameters.

9.3.4 *Hyper parameters and infrastructure*

We use easyfsl[1], an open-source deep learning toolkit with a number of implementations of metric-based learning methods. We use a Tesla V100 GPU, consisting of 16GB of memory for all models requiring GPU for train, and use amazon EC2-t2.micro. We use PyTorch 1.9 as the backend framework, and the Pytesseract package for extracting the text from the document images. We used the public architecture implementation from official matching networks (Chen et al., 2019) and relational networks, and the graph neural network is trained using the official implementation for few-shot graph convolutional network[2]. We used the canonical correlation block containing two neural networks, their architectures as shown in Table 9.3. Both the network parameters were optimized together using canonical loss using the RMSprop optimizer with a learning rate of 0.001

[1]https://github.com/sicara/easy-few-shot-learning
[2]https://github.com/vgsatorras/few-shot-gnn

Layer	Visual Feature	Text-Feature
Input	512	786
1st layer	1024, tanh	1024, tanh
2nd layer	1024, tanh	1024, tanh
3rd layer (output)	20, linear	20, linear

Table 9.3: Architecture of CCDI Block

9.3.5 *Comparing classification accuracy*

MiniImageNet

Results comparing the baselines to our model on meta-trained on miniImagenet and deployed on document image datasets are shown in Table 9.1. For 5-shot 5-way, 10-shot 5-way, and 20-shot 5-way, our proposed model outperforms all the existing baselines. As shown in the Table 9.1 all the baseline models, Prototypical Networks, Matching Networks and Relational Networks work well when the embedded model is Conv-4 and the performance degrades rapidly when a Resnet-10 block was used as an embedding model for each metric-based baseline method. As shown, the baseline method which closest performance to our proposed approach Prototypical network which achieves 61.09% accuracy for (5-shot, 5-way). However, performance doesn't improve much for both the (10-shot, 5-way) and the (20-shot, 5-way) classification. The proposed CCDI Model without the canonical co-relation block achieved an accuracy of 65.73% and the one with the canonical co-relation block has achieved an state-of-the-art accuracy of 66.68% which shows the significance of proposed method during the meta-testing phase.

Our experiments followed the same setup described above for meta-testing on the open source RVL dataset. Results are shown in Table 9.2. However the miniRVL dataset classi-

fication task is more challenging as it contains many documents that doesn't follow specific layout structures within classes. As shown Table 9.2 (b), the proposed method also outperforms the rest of the baseline methods and achieves state-of-the-art performance on this dataset by a large margin of up to 12% when compared to its closer competitor: Relational Networks.

TieredImageNet

To test the effectiveness of the proposed approach when meta-trained on a bigger domain, we repeated the experiments of the previous section. Results are shown in Table 9.2. On the INSR dataset, the models PrototTypical Networks(Conv-4), PrototTypical Networks(ResNet10), Relational Networks(Conv-4),Relational Networks(ResNet10)and Matching Networks(ResNet10) achieved mean accuracies of [61.09%, 52.21%, 50.28 %, 39.36%, 48.10%], respectively for (5-way, 5-shot) classification. In comparison, our proposed method achieved an accuracy of 66.68%. PrototTypical Networks (Con-4) came in second, with an accuracy of only 61.09% , followed by Relational Networks Networks(Con-4) with an accuracy of 50.28%. Finally, we found that our proposed model outperform those existing baselines in all the scenarios of (5-way, 5-shot), (10-way, 5-shot) , (20-way, 5-shot) and results can be found in the Table 9.2.

Vector Size	**Accuracy**
15	64.75%
20	**65.92%**
25	64.88%

Table 9.4: Effect on Output Vector dimension on Accuracy

No of Aug Images	Accuracy
0- Augmented Images	**66.68%**
5- Augmented Images	65.44%
10- Augmented Images	65.39%
15- Augmented Images	64.59%

Table 9.5: Evaluation Augmentation Results on INSR Dataset when meta-trained on TieredImagenet

9.3.6 *Ablation Studies*

Several ablation studies has been done to understand the significance and impact of the different hyperparameters on the proposed model. To evaluate the effectiveness of our proposed canonical correlation module for combining the textual and image features for document image classification, we first investigate the performance of the canonical correlation module for different output dimensions for $\mathcal{Z}$ based on the textual content and visual features. We did grid search the output dimensions from the set of [10,15,20,25] and results are shown in Table 9.4. The vector of size 20 gave the best performance results for different experiments.

We also evaluate the effectiveness our proposed approach for aligning the textual and image features instead of directly concatenating the textual and image features for document classification. As it is shown in Table 9.6, the DCCDI method achieves higher accuracies than the widely used concatenation of the image and textual features. We also performed an ablation study to see the effect of data augmentation on the generalization of performance on unseen data during the meta-test phase. As it is shown in various studies, data augmentation helps in the process of generalization for visual tasks which provide view-point invariance for each visual image. For this, we sampled additional images from the support set, and

perform jitter, random crops, and horizontal flips on a randomized basis. As we can see in Table 9.5 applying data augmentation does not have a positive effect on the solution. One reason for this might be that document images contain specific structure in them.

Method	5-Shot	20-Shot
Text-Concat	66.64%	78.51%
DCCDI	**67.8%**	**79.78%**

Table 9.6: Effect on CCDI block vs text-concatenation

9.4 Conclusion and Future Work

In many companies, millions of unlabeled documents containing information relevant to many business-related workflows have to be processed to be classified or / and to extract key information. Unfortunately, a large percentage of these documents consists of unstructured formats in the form of images and PDF documents. Examples of these types of documents include: medical bills, attorney letters, contracts, bank statements and personal checks. This process automation it usually refer as "Document intelligence" and relies on the use of models that combine both the image and text information to classify, categorize, and extract the relevant information. However, the labeled data needed by traditional learning approaches maybe too expensive and taxing on business experts and hence not practical in real-world industry settings. Hence, a few-shot learning pipeline is highly desired.

In this work, we proposed a novel method for few-shot document image classification under domain shift for semi-structured business documents, using the canonical correlation block to align extracted text and image feature vectors. We evaluate our work by extensive comparisons with existing methods on two datasets. We rigorously benchmarked our method

against the state-of-the-art few-shot computer vision models on both an insurance process derived dataset and the miniRVL dataset. We also presented the different ablation studies to show the effectiveness of the proposed method. The results showed our method consistently performed better than existing baselines on few-shot classification tasks. For future work, we would like to further explore more effective document representations including more sophisticated graph representations , or jointly trained layouts Mandivarapu et al. (2021a) and future directions of implementing the continual learning in document classification Camp et al. (2020); Mandivarapu et al. (2020a,2,2).

10| DISCUSSION AND CONCLUSION

10.1 Conclusion

In Conclusion, With the goal of achieving General AI which would be possible only with system of capabilities such as continual learning, active learning, efficent learning both in the fields of fewshot and large data pool and made significant contributions towards achieving general AI. Published state of the art work in the field of continual, active learning with showing range of usecases on the computervision tasks.

10.2 Summary of Contributions

This section will describe the contributions of the proposal. These can be summarized as follows:

1. **First Idea:** Made contributions towards experience replay based method in the field of continual learning using "Self-Net: Lifelong Learning via Continual Self-Modeling."(Accepted at Frontier's in AI) and supported others works

2. **Second Idea:** Made contributions towards Continual learning by using biologically motivated neuron structures and presented the work "Continual Learning using Deep Artificial Neurons." (Accepted at ICML-W)

3. **Third Idea:** Made contributions towards Active learning by first to propose solving the Active learning problem using Open set recognition approaches. (Accepted at Frontier's in AI)

4. **Fourth Idea:** Made contributions towards active learning by using the self-supervised methods and proposing "Deep Active Learning using Barlow Twins".

5. **Fifth Idea:** Made contributions towards efficient based learning by using Graph Neural networks (Efficient Document Image Classification Using Region-Based Graph Neural Network at Accepted at KDD-W)

6. **Sixth Idea:** Made contributions towards cross-domain learning by proposing "Cross Domain Few-Shot Learning for Document Intelligence". (In-Review at AAAI Conference)

10.3 Future Work

Some potential areas for future efforts could include the following:

1. Planning on extending the work of Active learning approach to graph neural networks and combing the active learning along with continual learning.

References

Adhikari, A., Ram, A., Tang, R., and Lin, J. (2019). Docbert: BERT for document classification. *CoRR*, abs/1904.08398.

Akaho, S. (2006). A kernel method for canonical correlation analysis. *arXiv preprint cs/0609071*.

Andrew, G., Arora, R., Bilmes, J., and Livescu, K. (2013). Deep canonical correlation analysis. In *International conference on machine learning*, pages 1247–1255. PMLR.

Ash, J. T., Zhang, C., Krishnamurthy, A., Langford, J., and Agarwal, A. (2019). Deep batch active learning by diverse, uncertain gradient lower bounds. *arXiv preprint arXiv:1906.03671*.

Bach, F. R. and Jordan, M. I. (2002). Kernel independent component analysis. *Journal of machine learning research*, 3(Jul):1–48.

Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer. *arXiv:2004.05150*.

Beluch, W. H., Genewein, T., Nurnberger, A., and Kohler, J. M. (2018). The power of ensembles for active learning in image classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9368–9377.

Beluch, W. H., Genewein, T., Nürnberger, A., and Köhler, J. M. (2018). The power of ensembles for active learning in image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9368–9377.

Bengar, J. Z., van de Weijer, J., Twardowski, B., and Raducanu, B. (2021). Reducing label effort: Self-supervised meets active learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 1631–1639.

Cai, J. and Shen, S. M. (2020). Cross-domain few-shot learning with meta fine-tuning. *arXiv preprint arXiv:2005.10544*.

Camp, B., Mandivarapu, J. K., and Estrada, R. (2020). Continual learning with deep artificial neurons.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. (2020). Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924.

Caruana, R. (2004). Multitask learning. *Machine Learning*, 28:41–75.

Chen, D., Chen, Y., Li, Y., Mao, F., He, Y., and Xue, H. (2021). Self-supervised learning for few-shot image classification. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1745–1749. IEEE.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C. F., and Huang, J.-B. (2019). A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*.

Das, A., Roy, S., and Bhattacharya, U. (2018). Document image classification with intra-domain transfer learning and stacked generalization of deep convolutional neural networks. *CoRR*, abs/1801.09321.

Dasgupta, S. (2011). Two faces of active learning. *Theoretical computer science*, 412(19):1767–1781.

Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Li, F. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255. IEEE Computer Society.

Denk, T. I. and Reisswig, C. (2019). Bertgrid: Contextualized embedding for 2d document representation and understanding. *CoRR*, abs/1909.04948.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Doersch, C. (2016). Tutorial on variational autoencoders. *arXiv e-prints*, page arXiv:1606.05908.

Finn, C., Abbeel, P., and Levine, S. (2017a). Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400.

Finn, C., Abbeel, P., and Levine, S. (2017b). Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR.

Finn, C., Rajeswaran, A., Kakade, S., and Levine, S. (2019). Online meta-learning.

Flennerhag, S., Rusu, A. A., Pascanu, R., Yin, H., and Hadsell, R. (2020). Meta-learning with warped gradient descent. *ArXiv*, abs/1909.00025.

Freeman, L. C. (1965). *Elementary applied statistics: for students in behavioral science*. John Wiley & Sons.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059.

Gal, Y., Islam, R., and Ghahramani, Z. (2017). Deep bayesian active learning with image

data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1183–1192. JMLR. org.

Garcia, V. and Bruna, J. (2018). Few-shot learning with graph neural networks.

Geifman, Y. and El-Yaniv, R. (2017). Deep active learning over the long tail. *arXiv preprint arXiv:1711.00941*.

Gidaris, S. and Komodakis, N. (2018). Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4367–4375.

Gissin, D. and Shalev-Shwartz, S. (2019). Discriminative active learning. *arXiv preprint arXiv:1907.06347*.

Gonzalez, T. F. (1985). Clustering to minimize the maximum intercluster distance. *Theoretical computer science*, 38:293–306.

Gorriz, M., Carlier, A., Faure, E., and Giró-i-Nieto, X. (2017). Cost-effective active learning for melanoma segmentation. *CoRR*, abs/1711.09168.

Greydanus, S. (2017). baby-a3c. `https://github.com/greydanus/baby-a3c`.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. (2020). Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33:21271–21284.

Guo, Y., Codella, N. C., Karlinsky, L., Codella, J. V., Smith, J. R., Saenko, K., Rosing, T., and Feris, R. (2020). A broader study of cross-domain few-shot learning. In *European Conference on Computer Vision*, pages 124–141. Springer.

Hanneke, S. et al. (2014). Theory of disagreement-based active learning. *Foundations and Trends® in Machine Learning*, 7(2-3):131–309.

Harley, A. W., Ufkes, A., and Derpanis, K. G. (2015). Evaluation of deep convolutional nets for document image classification and retrieval. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 991–995. IEEE.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. B. (2017). Mask R-CNN. *CoRR*, abs/1703.06870.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

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

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017). beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2(5):6.

Hotelling, H. (1992). Relations between two sets of variates. In *Breakthroughs in statistics*, pages 162–190. Springer.

Huisman, M., van Rijn, J. N., and Plaat, A. (2021). A survey of deep meta-learning. *Artificial Intelligence Review*, 54(6):4483–4541.

Huszár, F. (2018). Note on the quadratic penalties in elastic weight consolidation. *Proceedings of the National Academy of Sciences*, 115(11):E2496–E2497.

Jiang, J., Li, Z., Guo, Y., and Ye, J. (2020). A transductive multi-head model for cross-domain few-shot learning.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Kirsch, A., Van Amersfoort, J., and Gal, Y. (2019). Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *Advances in neural information processing systems*, 32.

Kirsch, A., van Amersfoort, J., and Gal, Y. (2019). BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning. *arXiv e-prints*, page arXiv:1906.08158.

Kölsch, A., Afzal, M. Z., Ebbecke, M., and Liwicki, M. (2017). Real-time document image classification using deep CNN and extreme learning machines. In *14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, November 9-15, 2017*, pages 1318–1323. IEEE.

Krizhevsky, A. (2009a). Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Krizhevsky, A. (2009b). Learning multiple layers of features from tiny images.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Kumar, J., Ye, P., and Doermann, D. (2014). Structural similarity for document image classification and retrieval. *Pattern Recognition Letters*, 43:119–126.

Lake, B., Salakhutdinov, R., Gross, J., and Tenenbaum, J. (2011). One shot learning of simple visual concepts. *Cognitive Science*, 33.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

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

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lewis, D. D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer.

Lifchitz, Y., Avrithis, Y., Picard, S., and Bursuc, A. (2019). Dense classification and implanting for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9258–9267.

Liu, B., Zhao, Z., Li, Z., Jiang, J., Guo, Y., and Ye, J. (2020). Feature transformation ensemble model with batch spectral regularization for cross-domain few-shot classification.

Liu, X., Gao, F., Zhang, Q., and Zhao, H. (2019). Graph convolution for multimodal information extraction from visually rich documents. In Loukina, A., Morales, M., and Kumar, R., editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019*, pages 32–39. Association for Computational Linguistics.

Mahapatra, D., Bozorgtabar, B., Thiran, J., and Reyes, M. (2018). Efficient active learning for image classification and segmentation using a sample selection and conditional generative adversarial network. *CoRR*, abs/1806.05473.

Mahapatra, D., Poellinger, A., Shao, L., and Reyes, M. (2021). Interpretability-driven sample selection using self supervised learning for disease classification and segmentation. *IEEE Transactions on Medical Imaging*, 40(10):2548–2562.

Mandivarapu, J., Camp, B., and Estrada, R. (2020a). Deep active learning via open set recognition. *CoRR*, abs/2007.02196.

Mandivarapu, J. K., Bunch, E., You, Q., and Fung, G. (2021a). Efficient document image classification using region-based graph neural network.

Mandivarapu, J. K., Camp, B., and Estrada, R. (2020b). Deep active learning via open set recognition. *arXiv preprint arXiv:2007.02196*.

Mandivarapu, J. K., Camp, B., and Estrada, R. (2020c). Self-net: Lifelong learning via continual self-modeling. *Frontiers in Artificial Intelligence*, 3:19.

Mandivarapu, J. K., Camp, B., and Estrada, R. (2021b). Deep active learning via open set recognition.

Martin, N. and Maes, H. (1979). *Multivariate analysis*. Academic press London.

McCallum, A. and Nigam, K. (1998). Employing em and pool-based active learning for text classification. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, page 350–358, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Melzer, T., Reiter, M., and Bischof, H. (2001). Nonlinear feature extraction using generalized canonical correlation analysis. In *International Conference on Artificial Neural*

Networks, pages 353–360. Springer.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. *ArXiv e-prints*.

Mundt, M., Majumder, S., Pliushch, I., and Ramesh, V. (2019a). Unified probabilistic deep continual learning through generative replay and open set recognition. *CoRR*, abs/1905.12019.

Mundt, M., Pliushch, I., Majumder, S., and Ramesh, V. (2019b). Open set recognition through deep neural network uncertainty: Does out-of-distribution detection require generative classifiers? *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 753–757.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. (2018). Variational continual learning. In *International Conference on Learning Representations*.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22:1345–1359.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., and Lin (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Pise, N. N. and Kulkarni, P. (2008). A survey of semi-supervised learning methods. In *2008 International Conference on Computational Intelligence and Security*, volume 2, pages 30–34.

Ravi, S. and Larochelle, H. (2016). Optimization as a model for few-shot learning.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing*

systems, 28.

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

Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., and Hadsell, R. (2018). Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*.

Scheffer, T., Decomain, C., and Wrobel, S. (2001). Active hidden markov models for information extraction. In *International Symposium on Intelligent Data Analysis*, pages 309–318. Springer.

Schohn, G. and Cohn, D. (2000). Less is more: Active learning with support vector machines. In *ICML*, volume 2, page 6. Citeseer.

Schwarz, J., Luketina, J., Czarnecki, W. M., Grabska-Barwinska, A., Whye Teh, Y., Pascanu, R., and Hadsell, R. (2018). Progress & Compress: A scalable framework for continual learning. *ArXiv e-prints*.

Sener, O. and Savarese, S. (2017). Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*.

Settles, B. (2010). Active learning literature survey. *University of Wisconsin, Madison*, 52.

Shui, C., Zhou, F., Gagné, C., and Wang, B. (2020). Deep active learning: Unified and principled method for query and training. In *International Conference on Artificial Intelligence and Statistics*, pages 1308–1318. PMLR.

Simonyan, K. and Zisserman, A. (2014a). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Simonyan, K. and Zisserman, A. (2014b). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sinha, S., Ebrahimi, S., and Darrell, T. (2019). Variational adversarial active learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5972–5981.

Snell, J., Swersky, K., and Zemel, R. S. (2017). Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*.

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H., and Hospedales, T. M. (2018). Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208.

Tong, S. and Koller, D. (2001). Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66.

Tur, G., Hakkani-Tür, D., and Schapire, R. E. (2005). Combining active and semi-supervised learning for spoken language understanding. *Speech Communication*, 45(2):171–186.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. (2016). Matching networks for one shot learning. *Advances in neural information processing systems*, 29:3630–3638.

Wang, G., Hwang, J.-N., Rose, C., and Wallace, F. (2017a). Uncertainty sampling based active learning with diversity constraint by sparse selection. In *2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE.

Wang, K., Zhang, D., Li, Y., Zhang, R., and Lin, L. (2017b). Cost-effective active learning for deep image classification. *CoRR*, abs/1701.03551.

Wang, S., Wang, Z., Che, W., Zhao, S., and Liu, T. (2021). Combining self-supervised learning and active learning for disfluency detection. *Transactions on Asian and Low-Resource Language Information Processing*, 21(3):1–25.

Weibull, W. (1951). A statistical distribution function of wide applicability. *Journal of applied mechanics*.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. (2019). A comprehensive survey on graph neural networks. *CoRR*, abs/1901.00596.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., and Zhou, M. (2019). Layoutlm: Pre-training of text and layout for document image understanding. *CoRR*, abs/1912.13318.

Yuan, M., Lin, H.-T., and Boyd-Graber, J. (2020). Cold-start active learning through self-supervised language modeling.

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146*.

Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. (2021). Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pages 12310–12320. PMLR.

Zenke, F., Poole, B., and Ganguli, S. (2017). Improved multitask learning through synaptic intelligence. *CoRR*, abs/1703.04200.

Zhang, M., Cui, Z., Neumann, M., and Chen, Y. (2018). An end-to-end deep learning architecture for graph classification. In McIlraith, S. A. and Weinberger, K. Q., editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 4438–4445. AAAI Press.

Zhong, X., Tang, J., and Jimeno-Yepes, A. (2019). Publaynet: Largest dataset ever for document layout analysis. In *2019 International Conference on Document Analysis and Recognition, ICDAR 2019, Sydney, Australia, September 20-25, 2019*, pages 1015–1022. IEEE.

Zhu, J. and Bento, J. (2017). Generative adversarial active learning. *CoRR*, abs/1702.07956.

Zhu, J., Wang, H., Tsou, B. K., and Ma, M. (2009). Active learning with sampling by

uncertainty and density for data annotations. *IEEE Transactions on audio, speech, and language processing*, 18(6):1323–1331.

Zhu, X. (2008). Semi-supervised learning literature survey. *Comput Sci, University of Wisconsin-Madison*, 2.

Zhu, Y., Xu, W., Liu, Q., and Wu, S. (2020). When contrastive learning meets active learning: A novel graph active learning paradigm with self-supervision.