

https://dspace.org/

Incremental similarity learning for knowledge retention in deep neural networks

Huo, Jiahao

2022
https://hdl.handle.net/10539/33881

Downloaded from a DSpace repository. https://dspace.org/

RESEARCH THESIS

Incremental similarity learning for knowledge retention in deep neural networks

Author:
Jiahao HUO (1475365)

Supervisor:
Prof. Terence L VAN ZYL



UNIVERSITY OF THE
WITWATERSRAND,
JOHANNESBURG

submitted to
the Faculty of Science, in fulfilment of the requirements for the degree of
Master of Science in Computer Science

in the

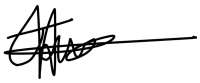
Wits School of Computer Science
School of Computer Science and Applied Mathematics

February 15, 2022

Declaration of Authorship

I, Jiahao HUO (1475365), declare that this research thesis titled, "Incremental similarity learning for knowledge retention in deep neural networks" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this research thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this research thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the research thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 

Date: 15 February 2022

UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

Abstract

Faculty of Science
School of Computer Science and Applied Mathematics

Master of Science in Computer Science

Incremental similarity learning for knowledge retention in deep neural networks

by Jiahao HUO (1475365)

The study investigate the open research problem of overcoming catastrophic forgetting during incremental similarity learning. The study examines the extent of forgetting during regular training on four well-known similarity learning loss functions: angular, contrastive, triplet, and center loss. We adapted three state-of-the-art existing learning techniques for incremental classification learning to incremental similarity learning. Furthermore, we present our novel incremental learning technique inspired by previous research. We compared our approach to the three existing techniques. Finally, we examine the forgetting problem on a variety of simple to complex datasets. We looked into different architecture setups to comprehensively examine how the different methods fare. The results confirm that catastrophic forgetting does occur during incremental similarity learning. Furthermore, we have shown the importance of good mining techniques for the similarity learning loss functions to reduce catastrophic forgetting. The results show that our approach outperformed the three existing techniques on average for incremental similarity learning. In addition, we note that our technique retained the highest ratio of base knowledge regardless of the training setup with the different datasets or network architectures. However, other methods learn new knowledge better. Nevertheless, our approach yields better average knowledge performance across all experiments. Furthermore, our approach can improve performance on unseen classes during incremental similarity learning. The results show that regularization techniques do not work as well as exemplar retention techniques in most of our experiments. In addition, we have shown that different combinations of training setups affect how each of the different techniques effectively reduces catastrophic forgetting. Further investigation into the relationship between transfer learning and similarity learning and the protection of intermediate layer embedding space for catastrophic forgetting is required.

Portions of this dissertation have appeared in:

- 2020 7th International Conference on Soft Computing, and Machine Intelligence (ISCMI) as Comparative Analysis of Catastrophic Forgetting in Metric Learning by Huo *et al.* [1].
- Incremental Class Learning using Variational Autoencoders with Similarity Learning by Huo *et al.* [2] currently under minor revisions at NCAA Springer Journal.

Acknowledgements

I would like to thank my family, my supervisor, Terence L van Zyl, and my good friends for the continuous support during the past years of study that lead me to what I am currently doing. The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

Contents

Declaration of Authorship i

Abstract ii

Acknowledgements iv

1 Introduction 1
- 1.1 Introduction . . . 1
- 1.2 Problem Statement . . . 2
- 1.3 Aim and Objectives . . . 3
- 1.4 Research Questions . . . 4
- 1.5 Contributions . . . 4
- 1.6 Limitations . . . 5
- 1.7 Structure . . . 5

2 Background and Related Work 7
- 2.1 Introduction . . . 7
- 2.2 Catastrophic Forgetting in Neural Networks . . . 8
- 2.3 Elastic Weight Consolidation . . . 9
- 2.4 Information-Back Discrete Representation Replay . . . 9
- 2.5 Encoder-Based Lifelong Learning . . . 10
- 2.6 Incremental Classifier and Representation Learning . . . 12
- 2.7 Ensemble-based Methods . . . 13
- 2.8 Dynamically Expanding Networks . . . 13
- 2.9 Similarity Learning . . . 14
  - 2.9.1 Triplet Loss . . . 14
  - 2.9.2 Contrastive Loss . . . 14
  - 2.9.3 Angular Loss . . . 15
  - 2.9.4 Center loss . . . 15
- 2.10 Conclusion . . . 16
- 2.11 Outline . . . 16

3 FullVAE and ConVAEr 17
- 3.1 Introduction . . . 17
- 3.2 Autoencoders . . . 17
  - 3.2.1 Vanilla Autoencoders . . . 17
  - 3.2.2 Variational Autoencoders . . . 17
- 3.3 Overview of Our Approach . . . 18
  - 3.3.1 FullVAE . . . 20
  - 3.3.2 ConVAEr . . . 20

3.4 Conclusion . . . 22

4 Research Methodology **23**
4.1 Introduction . . . 23
4.2 Research Design . . . 23
4.3 Method . . . 23
4.3.1 Research Instruments . . . 23
4.3.2 Modifications of the Existing Methods . . . 24
4.3.2.1 EWC . . . 24
4.3.2.2 EBLL . . . 25
4.3.2.3 iCaRL . . . 26
4.3.3 Datasets . . . 27
4.3.3.1 Motivating Datasets . . . 27
4.3.3.2 Primary Datasets . . . 27
4.3.4 Experimental Setup . . . 27
4.3.4.1 Preprocessing . . . 27
4.3.4.2 Training Sample Mining . . . 28
4.3.4.3 Hyper-parameters . . . 28
4.3.4.4 Exemplars for iCaRL . . . 28
4.3.5 Architecture of the CNNs and VAEs . . . 29
4.3.5.1 Architecture for Motivating Datasets . . . 29
4.3.5.2 Architecture for Primary Datasets . . . 30
4.3.6 Training and Testing . . . 30
4.3.6.1 Setup on Motivating Datasets . . . 30
4.3.6.2 Setup on Primary Datasets . . . 32
4.3.7 Analysis . . . 33
4.3.8 Hardware and software . . . 34
4.4 Conclusion . . . 34

5 Results and Discussions **35**
5.1 Evaluating Catastrophic Forgetting . . . 35
5.1.1 Pair and Triplet Mining . . . 35
5.1.2 Base and Overall Performance . . . 35
5.2 FullVAE . . . 36
5.2.1 Importance of Mining Techniques . . . 37
5.2.2 Performance on Base Test Set . . . 43
5.2.3 Impact of Loss . . . 44
5.2.4 Impact of Architecture . . . 45
5.3 ConVAEr . . . 46
5.3.1 Pair and Triplet Mining . . . 46
5.3.2 Knowledge Preservation . . . 47
5.3.3 Performance on Unseen Classes . . . 48

6 Conclusion **50**
6.1 Future work . . . 51

Bibliography **52**

Chapter 1

Introduction

1.1 Introduction

In machine learning, incremental learning refers to updating a model as new data becomes available or extended to support further tasks. An incrementally trained model should ideally retain previously attained knowledge while incorporating any new knowledge made available as it trains [3], [4]. Unfortunately, many machine learning algorithms cannot retain prior knowledge or do so in an unsatisfactory manner. Models that do not incrementally learn new tasks or classes whilst retaining prior knowledge suffer from catastrophic forgetting. Catastrophic forgetting in the machine learning context is when a machine learning model forgets how to perform well on previously learned tasks or classes during incremental learning. Catastrophic forgetting typically occurs during training on new data that contains no or highly imbalanced examples drawn from prior learned distributions [5], [6]. Catastrophic forgetting in deep neural networks and many of the tasks supported by them remains an open research problem [7]–[10]. Previous research has focused almost entirely on incremental classification learning in multi-layer perceptrons (MLP) neural networks and convolutional neural networks (CNNs). Typically encountered in computer vision tasks [11]–[13] and even Generative Adversarial Networks (GANs) [14]. Incremental classification learning refers to training a classification model to learn to perform classification on new tasks or classes incrementally.

Similarity learning is an area of supervised learning that looks into how similar objects are using a distance metric. In classification, the network outputs a probability for a given image for each class the network thinks it may belong to. However, a similarity network outputs a representation for the given image. We can use a distance metric to measure how related an image is to another image by their distance apart. So, for example, we can assign a label to an image given the image's closest neighbour. A similarity network can have any number of nodes for the output layer. Incremental similarity learning refers to the update of a similarity learning network with new tasks or classes.

There have been approaches to incremental classification learning by using regularization to stop drastic weight updates. This process can reduce catastrophic forgetting, such as Elastic Weight Consolidation (EWC). The EWC [13] approach calculates a Fisher Information Matrix learning each task and the Fisher Information Matrix regularizes updates of the parameters of the network so that it can find a

balance between the tasks. EWC has also shown success in reducing catastrophic forgetting when incrementally training GANs [15]. Another regularization method is Encoder-Based Lifelong Learning (EBLL) [12] whereby an autoencoder is trained before the network learns a new task. First, the autoencoder is trained to reconstruct outputs from the feature extractor section of the network. Then, all the autoencoders learned from the previous tasks are used for the next incremental step. The autoencoders regularize the update of feature extraction layers to perform reasonably well between all the tasks. The limitation of the method is it requires the autoencoders to be preserved for future incremental learning steps.

There have also been successful attempts to reduce catastrophic forgetting by giving reminders to the network in the incremental learning step about what it has learned before. One well-known example is Incremental Classifier and Representation Learning (iCaRL) [11], which stores a pool of exemplar images for each class seen in each incremental step. The exemplars combine with the new training data to update the network in each step. The exemplars can also perform classification on the test data. Recent research such as Information-Back Discrete Representation Replay [16] has also looked into encoding exemplars image into one dimension datapoints and stored be reconstructed for incremental learning in classification tasks. iCaRL puts a limit on the total number of exemplars that we can keep. The total number is divided between all the classes learned to get the number of exemplars per class. Thus, the number of exemplars per class is reduced as the network learns new classes. Unfortunately, the reduced exemplars per class may not be enough to remind the network to preserve knowledge as the network incrementally train.

A different take on incremental learning of tasks is the use of network architectures designed for incremental learning. One such example is dynamically expandable networks [17] that can expand their capacity when necessary to learn more tasks. The network tries to find unused neurons in the network not occupied by the previous tasks. Then, it selectively retrains the unused neurons. If it cannot find any, the network expands each layer where necessary. However, the main limitation of dynamically expandable networks is the growing number of parameters to compute as the network grows. In addition, the hardware may not handle a lot of network expansion due to memory limitations.

Our research investigates incremental similarity learning for learning new classes sequentially on the following datasets: MNIST [18], FMNIST [19], EMNIST [20], and CIFAR10 [21], which we denote as motivating datasets. However, experiments using only the motivating datasets only partially represent performance expected on real-world data. Therefore, we further experimented using CUB200 [22] and CAR196 [23] to better represent real-world performance which we denote as primary datasets.

1.2 Problem Statement

The common similarity between previous research is that they primarily focus on catastrophic forgetting during incremental classification learning. However, there persists a lack of evidence detailing the degree to which similarity learning tasks and

the underpinning pair mining and loss functions are affected by catastrophic forgetting during incremental similarity learning. Most techniques aimed at overcoming catastrophic forgetting in deep neural networks have been designed with incremental classification learning in mind [11]–[13], [16]. There is currently a lack of insights into how well existing incremental classification learning approaches would work for incremental similarity learning.

1.3 Aim and Objectives

The study aims to adapt and implement four different incremental-learning techniques, including our novel method, to overcome catastrophic forgetting during incremental similarity learning. Our study aims to fill the gap and investigate incremental similarity learning with deep convolution neural networks and deep neural networks (CNNs and DNNs). Incremental similarity learning will enable the networks to incrementally learn new knowledge without the need to be retrained on the entire dataset. The goal is for the similarity learning networks to continue to perform well in old classes and continuously improve while learning new knowledge. The objectives of our study are to:

- Implement similarity learning in siamese networks using the following loss functions: Triplet, Contrastive, Center loss, and Angular loss.
- Compare the differences in catastrophic forgetting on the four different loss functions without incremental learning techniques.
- Adapt parameter consolidation incremental learning technique, Elastic Weight Consolidation (EWC), to the siamese networks as a baseline.
- Adapt the Encoder-Based Lifelong Learning (EBLL) incremental learning technique to the siamese networks as a baseline.
- Adapt the rehearsal based incremental Classifier and Representation Learning (iCaRL) incremental learning technique to the siamese networks as a baseline.
- Implement our novel incremental learning approach FullVAE and ConVAEr on the siamese networks.
- Compare how the networks perform with and without any of the incremental learning techniques on motivating datasets.
- Compare how the techniques perform on incremental similarity learning, using our incremental similarity learning test metric. Our test metric is a minor variation of the test metric from Kemker *et al.* [24] for catastrophic forgetting assessment of incremental classification learning.
- Compare the best performing existing technique to our FullVAE and ConVAEr technique on primary datasets with a complex network during incremental similarity learning.

1.4 Research Questions

We present the following research questions:

1. How do siamese networks that apply incremental learning techniques compare to siamese networks without incremental-learning techniques?
2. Which of the loss functions is the most robust to catastrophic forgetting on motivating and primary datasets?
3. How do the three techniques from existing literature and our novel approach compare during incremental similarity learning on the motivating datasets using simple and complex CNN architectures?
4. How does the best performing existing technique compare to our FullVAE and ConVAEr technique on the primary datasets?

1.5 Contributions

The contributions of this study is:

- The results show that the four different loss functions were not capable of incremental similarity without the aid of any incremental learning technique (Section 5.1 and 5.2).
- We showed successful adaptation of the EWC, EBLL and iCaRL technique to incremental similarity learning and was able to reduce catastrophic forgetting (Section 5.2).
- We introduced our novel method (FullVAE and ConVAEr) using simple Variational Autoencoders (VAE) (Chapter 3).
- The results show that our FullVAE approach outperformed the other three existing solutions for incremental similarity learning (Section 5.2.3).
- The results show the importance of good mining techniques for the loss functions (Section 5.2.1).
- We reinforce that an increased number of initial classes comes with increased catastrophic forgetting on motivating datasets. We illustrate that the more complex the task, the greater the catastrophic forgetting (Both in Section 5.2.2).
- The results show that Elastic Weight Consolidation and Encoder Based Lifelong are less effective than other methods for incremental similarity learning (Section 5.2.2).
- The dissertation introduces the importance of keeping some form of previous knowledge during incremental similarity learning (Section 5.2.2 and 5.3).
- We highlight that with good mining techniques, Angular loss is least affected by catastrophic forgetting on the motivating datasets (Section 5.2.3).

- The results show that Contrastive loss with good mining technique with our approach is the most robust for simple and real-world datasets (Section 5.2.3 and 5.3).
- We demonstrate that generated representations from VAEs work just as well as images for exemplars on motivating and primary datasets (Section 5.2.3 and 5.3.2).
- The results show that ConVAEr yields better average knowledge retention across all experiments using a more complex network architecture on primary datasets (Section 5.3).
- The results show that intercepting exemplars from the convolutional layers (ConVAEr) retained the highest ratio of base knowledge compared to FullVAE and iCaRL (Section 5.3.2).
- We note that using embeddings from the linear layers (FullVAE) leads to better performance on new knowledge than convolutional embeddings (ConVAEr) (Section 5.3.2).
- We show potential improvements to unseen class performance on primary datasets when performing incremental similarity learning using our approach (Section 5.3.3).

1.6 Limitations

The limitations of our study are that we have not looked into the entirety of all the possible similarity loss functions available in the similarity learning space. In addition, we have not looked into how the methods would perform on all possible image datasets in the computer vision field because that is not feasible. Furthermore, we have not investigated how all available network architectures would be affected by catastrophic forgetting because it is not feasible. However, we give a general idea of how the techniques would perform using well-known datasets with well-known CNN architectures. Finally, we do not include some recent advances for overcoming catastrophic forgetting.

1.7 Structure

The structure of our study is as follows: In Chapter 2, we discuss the related research and concepts related to catastrophic forgetting and the different incremental learning technique that aims to overcome catastrophic forgetting. Furthermore, including details of various similarity loss functions, we implement for the research. In Chapter 3, we present the details of our novel approach and illustrate how it works. In Chapter 4, we discuss the aims and objectives of our research and the methodology detailing the training setups of our experiments and the various metrics we use to measure each of the models' performance. Chapter 5 presents many results from our

investigation, comparing the different approaches and presenting detailed key observations and discussions. Finally, we provide a summary of our research and possible future work in Chapter 6.

Chapter 2

Background and Related Work

2.1 Introduction

Incremental learning is the idea of a machine learning model that can continuously learn new information and adapt to real-world tasks. Incremental learning enables a model to progressively learn and develop its skills and knowledge based on the needs of the real world. In machine learning, incremental learning refers to the incremental update of a machine learning model with new data to perform new tasks. The model should still retain the model's previous knowledge to be usable on previously learned tasks or classes [11], [12], [16], [25].

Catastrophic forgetting in contrast to gradual or partial forgetting is when a machine learning model completely or abruptly forgets previously learned information. The forgetting occurs during the process in which it is learning new information on new data [24]. However, humans can retain the knowledge of previous tasks while learning a new task. We should aim for the same ability with neural networks in machine learning.

Previous research on incremental learning commonly also tries to overcome catastrophic forgetting because incremental learning and catastrophic forgetting are related to one another [8], [11], [13], [26], [27]. Previous research looked into the use of biologically inspired network architectures [28], network parameter consolidation by constraining newly learned parameters [13], and network architectures such as dynamically expanding networks [17]. The methods allow a machine learning model to learn more than one task. An example of incremental learning that used fake detection was a research by Marra *et al.* [15]. The authors adapted the Incremental Classifier and Representation Learning (iCaRL) technique for deep fake detection of GANs. iCaRL is technique recently proposed for incremental object classification [11]. The research proposed an incremental-learning strategy to address the problem of DNN's performing well on one type of GAN but not on another type of GANs. The reason being different GANs generate fake images differently. The incremental-learning strategy allows them to incrementally train a fake detection model that performs well against new unseen GANs and previously learned GANs.

This section covers the various challenges with incremental learning and presents previous related research on various machine learning algorithms that overcame catastrophic forgetting during incremental learning to a certain extent. Furthermore, in a

later section, we present our own novel incremental learning approach designed for similarity learning networks using inspiration from the previous incremental learning techniques.

2.2 Catastrophic Forgetting in Neural Networks

Goodfellow *et al.* [7] investigated catastrophic forgetting in gradient-based neural networks used for classification. The results showed that various combinations of activation function and learning were affected differently by catastrophic forgetting. Rannen *et al.* [12] demonstrated the problem of catastrophic forgetting in deep neural networks (DNN) using the AlexNet architecture. The research highlighted the classification performance drop in a previously learned task when a DNN is fined-tuned for newer classification tasks. The authors proposed using lightweight autoencoders to preserve the feature representations learned by the base network (AlexNet) that were optimal for the task before learning the next task. An autoencoder is trained after the network learns each new task, increasing storage requirements for storing each autoencoder. The methods proposed significantly reduced catastrophic forgetting that occurred during incremental classification learning.

Further research by Thanh-Tung *et al.* [14] showed an in-depth analysis of catastrophic forgetting in Generative Adversarial Networks (GAN). They focused on explaining the causes and effects of catastrophic forgetting and how it relates to mode collapse and non-convergence of the GANs. The research highlighted the problem of GANs suffering from forgetting even when trained on eight datasets generated from a single mixture of eight Gaussian models. Further analysis into incremental learning real-world image datasets such as CIFAR-10 and CelebA exhibited the same problem in GANs. The first main reason is that the new task does not use the knowledge learned from earlier tasks. Second, the current task is too different from previous tasks, and therefore knowledge is not reused and is instead overwritten. Existing research by Seff *et al.* [29] demonstrated the use of EWC [13] to try and overcome catastrophic forgetting in GANs during incremental training on the set of distributions. The results showed less forgetting on the GAN when trained on The Street View House Number (SVHN) digit classification dataset using EWC compared to standard training but it still suffered some forgetting. However, the experiments only performed one step of incremental learning whereby they initially trained the GAN with data from 0 through 4. In the incremental step, the GAN was trained with data on digits 5 through 9. A single step of incremental learning in the experiments does not show how well the EWC method would aid the GAN over long term incremental learning.

The research by Choi *et al.* [30] proposed the use of an autoencoder-based incremental classification learning technique without the use of a softmax classification layer like conventional classification models. The research is premised on the notion of a metric-based classification method, nearest-class-mean (NCM), an earlier research by Mensink *et al.* [31]. The idea is to make use of a pre-trained fixed network as a feature extractor for the autoencoder. The autoencoder is trained on the feature embeddings. The encoded representations from the encoder are used for cosine similarity-based classification. The problem of catastrophic forgetting occurs when the autoencoder is fine-tuned for feature embedding from new classes learned

incrementally. To overcome this difficulty, the authors use regularization techniques: Synaptic Intelligence (SI) [32], and Memory Aware Synapses (MAS) [33]. The techniques add a term to the existing loss function during the incremental class learning phase. The methods demonstrated good memory retention without the need to train on older data.

2.3 Elastic Weight Consolidation

Elastic Weight Consolidation (EWC) is a method proposed by Kirkpatrick *et al.* [13] aimed at overcoming the limitations of catastrophic forgetting in neural networks during incremental class learning. EWC selectively slows down the learning of weights that are of importance to the previously learned tasks. The constraint used to slow down weight updates is the Gaussian distribution modelled using the network weights as the mean and diagonal of the Fisher information matrix as the diagonal precision from previous tasks. The update constraint is shown as:

$$L(\theta) = L_t(\theta) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{t-1,i})^2 \,, \tag{2.1}$$

where $L(\theta)$ is the combined loss. θ are the network parameters. $L_t(\theta)$ is loss of the current training session at time t. λ is a hyper-parameter that indicates the importance of the old tasks compared to the new tasks. i represents each parameter of the network. F is the Fisher Information Matrix used to constrain the weights important for previously learned tasks to the original values [34]. θ^*_{t-1} are the trainable parameters from the previously trained model from the previous training session, $t-1$. Kirkpatrick *et al.* computed the Fisher Information Matrix using the gradient of the cross-entropy loss from the output of the network. The Fisher Information matrix is computed using the following:

$$F = \frac{1}{N} \sum_i^N \nabla \log p(x_i|\theta) \nabla \log p(x_i|\theta)^T \,, \tag{2.2}$$

where $\nabla \log p(x_i|\theta)$ is the gradient of the the log likelihood. N is the number of training samples, x_i is the i^{th} training sample, and θ is the parameters in the network.

However, EWC is an incremental learning technique aimed at classification, and the Fisher information matrix [34] used was designed for classification networks using cross-entropy. Therefore, to use EWC for similarity learning, we applied the gradient of our respective loss functions to compute the Fisher Information Matrix instead of cross-entropy likelihood, which we show in Section 4.3.2.

2.4 Information-Back Discrete Representation Replay

Incremental Learning with Information-Back Discrete Representation Replay (IB-DRR) is an approach proposed by Jiang *et al.* [16] for incremental classification learning. The approach formalized incremental learning as a latent representation compression problem whereby image exemplars are stored using as little memory

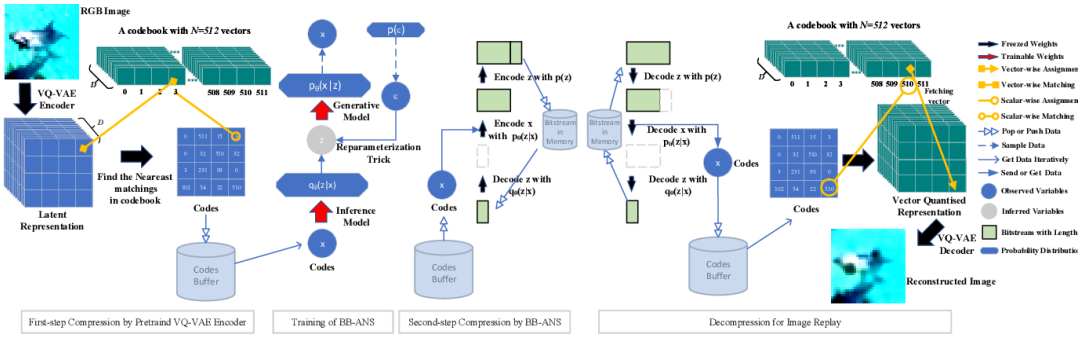


FIGURE 2.1: A high-level diagram depicting the IB-DRR algorithm [16].

as possible. The main idea is to compress and save image exemplars in representation for replay during incremental learning. The approach involves two steps of compression. Firstly, the authors used a pre-trained Vector-Quantized Variational Autoencoder (VQ-VAE) for compressing image exemplars into discrete latent representation to store into a buffer. Second, a Bits-Back Asymmetric Numeral Systems (ANS) was used to compress the latent representations into one dimension datapoints in the bit-streams in the memory. The discrete representations are replayed to the network during incremental learning by decompressing the bit-streams to latent representations and then back into the image domain. The decompressed images feed into the network alongside new images. The approach delivered good performance comparable and even slightly outperforms the state-of-the-art techniques on CIFAR100. We have included an illustration of how IB-DRR process works in Figure 2.1.

2.5 Encoder-Based Lifelong Learning

Encoder-Based Lifelong Learning (EBLL) was proposed by Rannen *et al.* [12] for incremental classification learning. EBLL modifies how the convolutional layers of a network are updated. After each incremental learning task, the authors train an autoencoder to reconstruct the "images" at the output of the convolutional layers. The reconstructed images are passed through the network's remaining fully connected layers to calculate their resulting classification loss. The reconstruction loss, together with the classification loss, is used to update the autoencoder's weights. For incremental learning, the previous tasks classification layer detaches for each new incremental learning task, and a new classification layer is attached. A frozen copy of the previous optimal network is preserved before training the next task. The new images pass through both the new and frozen networks during training for the new task. However, only the new network is updated. The image's convolution output of the new and frozen network are propagated through the autoencoder up to the bottleneck layer. The mean square error of the encoded outputs from the bottleneck layer of the autoencoder is calculated between the two networks and added to the final classification loss. Distillation loss is also calculated between all tasks learned previously and added to the final loss. The final classification loss propagated through the network's weights to update the network's weights. The process constrains the weight update of the convolutional network layers to compromise between new and old tasks. The

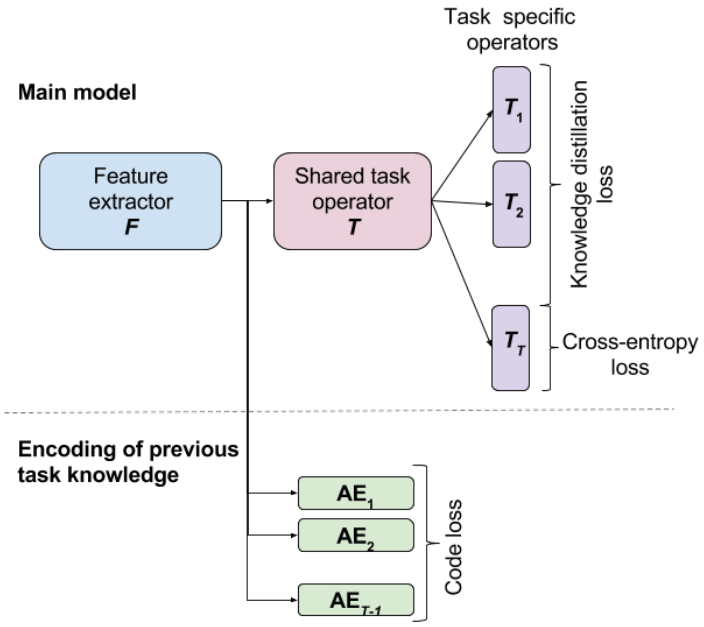


FIGURE 2.2: A high-level diagram depicting the EBLL algorithm [12].

loss to update the network during the incremental learning step is:

$$
\begin{aligned}
L = & \frac{1}{N} \sum_{i=1}^{N} \Bigg(\ell_{\textbf{cross entropy}} \left(T_T \circ T \circ F \left(X_i^{(T)} \right), Y_i^{(T)} \right) \\
& + \sum_{t=1}^{T-1} \ell_{dist} \left(T_t \circ T \circ F \left(X_i^{(T)} \right), T_t^* \circ T^* \circ F^* \left(X_i^{(T)} \right) \right) \\
& + \sum_{t=1}^{T-1} \frac{\alpha_t}{2} || \sigma \left(W_{enc,t} F \left(X_i^{(T)} \right) \right) - \sigma \left(W_{enc,t} F^* \left(X_i^{(T)} \right) \right) ||_2^2 \Bigg) ,
\end{aligned}
\tag{2.3}
$$

where N is the number of training samples in the current task T. The $T_t^* \circ T^* \circ F^*(X_i^{(T)})$ is the respective optimal feature extractor F, shared task operator T and task specific operators for task t.

$$
\ell_{\textbf{cross entropy}} \left(T_T \circ T \circ F \left(X_i^{(T)} \right), Y_i^{(T)} \right) ,
$$

is the cross entropy loss of the task the network is currently learning.

$$
\sum_{t=1}^{T-1} \ell_{dist} \left(T_t \circ T \circ F \left(X_i^{(T)} \right), T_t^* \circ T^* \circ F^* \left(X_i^{(T)} \right) \right) ,
$$

is the sum of distillation loss of all the previously learned tasks not including the current task. $F(X_i^{(T)})$ is the feature extractor outputs for the image $X_i^{(T)}$ from the in-training network for the current task. $F(X_i^{(T)})$ is the feature extractor outputs for the image $X_i^{(T)}$ from the frozen network copy made before training on the current

task.

$$\sum_{t=1}^{T-1} \frac{\alpha_t}{2} ||\sigma(W_{enc,t}F(X_i)) - \sigma(W_{enc,t}F^*(X_i))||_2^2) \ ,$$

is the sum of the mean square error of the encoded outputs between the teacher and student feature extractor given the image X_i^T. The distillation loss used is a modification of cross entropy from Hinton *et al.* [35]. The loss for training the autoencoder is:

$$\begin{aligned} L_{ae} =& \lambda \cdot \left|\left| r\left(F^*\left(X_i^{(T)}\right)\right) - F^*\left(X_i^{(T)}\right)\right|\right| \\ &+ \ell_{\textbf{cross entropy}}\left(T_T \circ T \circ F\left(X_i^{(T)}\right), Y_i^{(T)}\right) \ , \end{aligned} \tag{2.4}$$

where L_{ae} is the total loss of the autoencoder.

$$\left|\left| r\left(F^*\left(X_i^{(T)}\right)\right) - F^*\left(X_i^{(T)}\right)\right|\right| \ ,$$

is the reconstruction mean square error loss, $X_i^{(T)}$ is the input image, $Y_i^{(T)}$ is the label for the input image, $F^*(X)$ is the feature extractor output for image X , $r\left(F^*\left(X_i^{(T)}\right)\right)$ is the reconstructed feature extractor output, with

$$\ell_{\textbf{cross entropy}}\left(T_T \circ T \circ F\left(X_i^{(T)}\right), Y_i^{(T)}\right) \ ,$$

as the cross entropy loss using the optimal set of parameters learned. Let λ be the importance we place on the reconstruction loss per Rannen *et al.* [12]. F^* refers to the optimized convolutional layers of the network for the task. We have included an illustration of how EBLL process works in Figure 2.2.

2.6 Incremental Classifier and Representation Learning

Rehearsal methods try to decrease catastrophic forgetting by mixing some data from earlier learned classes to the present learning session. However, the method's cost requires storing some past data learned, which is not memory efficient. Existing pseudo-rehearsal attempt to counter the problem by making use of a network to generate pseudo-patterns that resembles previously learned data [10], [36]. The generated data is combined with the data in the present learning session. Pseudo-patterns allow the networks to see older data without the need for storing them.

One such example of rehearsal method is Incremental Classifier and Representation Learning (iCaRL) is a method proposed by Rebuffi *et al.* [11] for reducing catastrophic forgetting. iCaRL can learn a strong classifier and a data representation simultaneously. As a result, it is compatible with deep neural networks. iCaRL primarily relies on the storing of exemplars from previously seen classes. The authors construct each class's exemplar set by storing k images ranked by the closeness of their representation to the class's mean representation. This selection of the k closest images is known as the herding algorithm. The stored exemplars are used to supplement the incremental learning phase of new classes and use knowledge distillation.

Classification is performed using the stored exemplars following the nearest-mean-of-exemplars rule: A new image is classified to the class of the exemplar closest to it. iCaRL is reported to learn classes incrementally over a longer period where other methods failed more quickly. The distillation loss used was also a modification of cross-entropy from Hinton *et al.* [35] similar to EBLL [12] discussed above.

2.7 Ensemble-based Methods

Ensemble-based incremental-learning methods try to decrease catastrophic forgetting by explicitly or implicitly training multiple classifiers. The classifiers will each predict what new unseen data is. The predictions are combined to make a final prediction based on the highest votes. XGBoost [37] is an example of an explicit ensemble-based incremental-learning method. The problem with explicit methods is that the memory usage will scale according to the number of learning sessions, which is not the desired outcome. PathNet [8] is an example of an implicit ensemble-based method. PathNet uses a genetic algorithm to find an optimal path through a fixed size network for each training session. The weights of the previous paths are frozen during the subsequent training sessions of new tasks or classes. The network will retain the earlier knowledge learned because one task uses a unique path, and previously found paths are frozen for subsequent training sessions for new tasks. Therefore the limitation to PathNet is that it can only learn a limited amount of training sessions.

2.8 Dynamically Expanding Networks

Dynamically Expanding Networks (DEN) are network architectures that can actively decide their network capacity during training. During incremental learning, the network will actively expand or decrease its capacity depending on the sequence of trained tasks. Thus, the networks can learn a dense overlapping knowledge-sharing structure among the sequentially learned tasks or classes. DENs is a trained online approach by doing selective retraining on the previously trained old network. The old network's capacity will dynamically expand if necessary. It decides on the optimal capacity during the training phase on the new task or class. Previous research by Yoon *et al.* [17] looked into a network architecture that can learn overlapping knowledge between tasks and dynamically expand its network capacity when necessary given a set of rules. The dynamically expanding networks by Yoon *et al.* procedure are as follows:

- Selective retraining: retraining the whole network would be time inefficient. Therefore, selective retraining aims to find weights that are affected by the new tasks. The network is trained with $L1$-regularization to promote sparsity between the weights. During the incremental procedure for new tasks, a new output for the new task is attached to the model while the old output layer is stored aside. All the layers before the output layer of the network are frozen. The network is trained through the new data to find all weights affected by the new tasks using a breadth-first search starting from the last layer up top the input layer. The selected weights update during the partial training of the network.

- Dynamic network expansion: If the selective retraining procedure is not sufficient for the new task, then the network is allowed to expand by adding a certain amount of neurons to each layer decided by the limits set by the user. The network can drop some of the added neurons based on how much was necessary.
- Network split/duplication: Some of the network's neurons are essential for the new and old task and require a significant update to the neurons during incremental classification learning. The network will split/duplicate the neurons for the new and old tasks to update the neurons without affecting the old task.

2.9 Similarity Learning

Similarity learning is an area of supervised learning focused on learning how similar two or more objects are related to one another. Siamese networks are a class of neural network architecture that contains two or more identical subnetworks. The subnetworks have identical configurations with the same parameters and weights. Siamese networks are commonly used to find similarities between images [38], [39]. Since the research focuses on catastrophic forgetting in similarity learning methods, we consider four prominent loss functions:

2.9.1 Triplet Loss

Triplet loss by Wang *et al.* [38], [40] has been shown to learn good feature representations for determining image and video similarity [41]. The triplet comprises an anchor ground truth image, a positive image and a negative image. The positive image belongs to the same identity as ground truth, and a negative image is selected from an identity that differs from the anchor. The loss is given as:

$$\mathcal{L} = \max(d(a,p) - d(a,n) + \text{margin}, 0)\,, \tag{2.5}$$

where d is euclidean distance, a is the anchor ground truth image, p is the positive image, n is the negative image. The margin is a radius around the anchor and determines the degree to which the negative image is pushed away. The function optimizes the distance between the anchor-positive and anchor-negative simultaneously by bringing the positive pair closer and pushing the negative pairs apart.

2.9.2 Contrastive Loss

Contrastive loss finds optimal features by using pairs of positive and negative non-matching pairs of images. The function is given as:

$$\mathcal{L} = \frac{1}{2}(1-Y)(d(\tilde{y}_i,\tilde{y}_j))^2 + \frac{1}{2}(Y)\{\max(0, \text{margin} - d(\tilde{y}_i,\tilde{y}_j))\}^2\,, \tag{2.6}$$

where Y is the label 0 or 1 and is 0 if the input pair are from the same class and 1 otherwise. $d(\tilde{y}_i,\tilde{y}_j)$ is the Euclidean distance between the output feature representations of the network for the pair of images. The loss function differs from triplet loss

in that it tries to minimize the distance between positive pairs and maximize negative pairs in separate steps.

2.9.3 Angular Loss

Wang *et al.* [42] aimed to overcome some of the challenges with triplet loss. These problems include the infeasibility of exhaustive sampling and the use of a single global margin m. Angular loss tries to address these by constraining the angle of the triangle between the triplets. The resulting rotation and scale invariance make the loss function more robust to significant variations in features. The angular loss of a batch $\mathcal{B}$ of size N is defined as:

$$l_{ang}(\mathcal{B}) = \frac{1}{N} \sum_{x_a \in \mathcal{B}} \{\log[1 + \sum_{\substack{x_n \in \mathcal{B} \\ y_n \neq y_a, y_p}} \exp(f_{a,p,n})]\} , \tag{2.7}$$

where x_a is the anchor image. x_n is a negative image (different class from anchor). Function f is defined as

$$f_{a,p,n} = 4\tan^2\alpha(x_a + x_p)^T x_n - 2(1 + \tan^2\alpha)x_a^T x_p, \tag{2.8}$$

where x_p is a positive image (same class as anchor image). α is the degree of the angle. y_n is the class label of the negative image. y_a is the class label of the anchor image. y_p is the class label of the positive image.

2.9.4 Center loss

Wen *et al.* [43] try to enhance the discriminative power of learned features specifically for facial recognition. The loss function learns a center for features of unique classes. It simultaneously penalizes the distances between the features of the images and their corresponding class centers that maximize inter-class separation and intra-class compactness. Center loss cannot be used directly as a loss function and is therefore paired with softmax as defined by:

$$L_{s+c} = -\sum_{i=1}^{m} \log \frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T x_i + b_j}} + \frac{\lambda}{2} \sum_{i=1}^{m} \|x_i - c_{y_i}\|_2^2, \tag{2.9}$$

where L_{s+c} denotes the loss comprising of both softmax and center loss. The left term is the formula for softmax function. The right term is the formula for center loss. x_i denotes the features from the network. c_{y_i} denotes the center value for class y_i. λ is a scalar that is the importance of weighting between the two losses. In the paper, a scalar value, α, that controlled the learning rate of the centers.

2.10 Conclusion

In this section, we have outlined the problem of catastrophic forgetting during incremental learning in neural networks. We have also covered the various previous research that looked into overcoming catastrophic forgetting in classification networks. While these approaches have demonstrated good results in overcoming catastrophic forgetting in classification, no such research has been done to investigate the effect of catastrophic forgetting in similarity learning. Thus our research aims to fill the gap and investigate incremental similarity learning with deep convolution neural networks and deep neural networks (CNNs and DNNs). Incremental similarity learning will enable the networks to incrementally learn new knowledge without the need to be retrained on the entire dataset. The goal is for the similarity learning networks to continue to perform well in old classes and continuously improve while learning new knowledge.

2.11 Outline

This section has covered the related research to catastrophic forgetting and incremental learning for convolutional neural networks. Next, we cover how we take inspiration from the previous studies to construct our novel approach to incremental similarity learning and illustrate how the approach works.

Chapter 3

FullVAE and ConVAEr

3.1 Introduction

The previous section covered some research done on incremental learning and the methods to overcome catastrophic forgetting while performing incremental learning. The following section presents details of our novel approach to incremental similarity learning known as FullVAE and ConVAEr. We took inspiration from previous related research and constructed a framework for incremental similarity learning that can use any CNN architecture.

3.2 Autoencoders

3.2.1 Vanilla Autoencoders

Autoencoders (AEs) are a class of neural networks that aim to compress the input into a latent representation. AEs can also reconstruct the input from the latent representation first introduced by Hinton *et al.* [44]. Autoencoders consists primarily of two parts: the encoder and the decoder. The encoder is responsible for compressing the original input, such as an image, into a latent representation that is commonly a much lower dimension than the input. The decoder is responsible for taking in the latent representation and reconstructing the original input. The encoder and decoder are trained together.

3.2.2 Variational Autoencoders

Variational Autoencoders (VAEs) is an alternative improvement on autoencoders. Aside from the ability to reconstruct the original input from the latent representation. We can also generate new data that are related to the input data. The VAEs assume that the input data is from a parameterized distribution and find the parameters for the distribution. For example, in the case of the Gaussian distribution, the VAE tries to find the mean and variance parameters for the Gaussian distribution. We can generate outputs from VAE by sampling random latent representations from a standard normal distribution and inputting it into the decoder. A tutorial on VAEs can be found in the study by Carl [45] that gives additional detail on how VAEs work.

3.3 Overview of Our Approach

Thus, with the understanding of how VAEs work, we formulate our FullVAE and ConVAEr approaches. Rannen *et al.* constrain the weights of the feature extraction layers (convolutions) that were optimal for previous tasks with an autoencoder. The solution is effective when reusing the feature extraction layers (convolutions) on new tasks. Each task is tested independently from the others with its classification layer. The approach yields promising results by finding a middle-ground across tasks.

The iCaRL method by Rebuffi *et al.* largely depends on the storage and usage of exemplars. As reported, the performance of iCaRL decreases with time as the number of exemplars per class is reduced to accommodate new classes. Eventually, the stored exemplars may not be sufficient to represent all classes.

We propose a novel approach that brings together ideas from Rannen *et al.* [12] and Rebuffi *et al.* [11]. Before each incremental learning step, we train a new variational autoencoder (VAE) for the new class. The VAEs learn representations at the end of the convolutional layers. VAEs allow us to sample previously seen classes as an output from the convolutional or fully connected layers instead of reconstructing the images from each class. A complete CNN approach using images is more computationally expensive and requires more complex VAEs but is not infeasible.

However, the distillation loss presented and used by iCaRL and EBLL is a modification of cross entropy [35] that is not suitable for similarity learning. Instead, we used a different form of distillation loss more suitable to similarity learning. Furthermore, the research by Park *et al.* [46] introduces different types of distillation loss that we can use for similarity learning. Therefore, we use the angle-wise distillation loss to preserve the relationship between exemplars while incrementally learning. The angle-wise distillation loss is defined as:

$$L_A(X_s, X_t) = \sum_{\substack{s_i,s_j,s_k \in X_s \\ t_i,t_j,t_k \in X_t}} \ell_h(V_A(t_i,t_j,t_k), V_A(s_i,s_j,s_k)), \tag{3.1}$$

where X_s is network outputs for a given batch of images from the student network (model that is being updated). X_t is network outputs for a given batch of images from teacher network (model that is trained and frozen). The t_i, t_j, t_k is the triplets of image representations formed from X_t and s_i, s_j, s_k is the triplets of image representations formed from X_s. The loss penalizes the angular differences between the two networks. An angle is a higher-order property than a distance which may be able to transfer relational knowledge better.

With reference to the Figures 3.1 and 3.2 that shows the setup of FullVAE and ConVAEr methods respectively. Our method requires that the convolutional layers be frozen after initial training. Alternatively, pre-trained frozen convolutional layers from a base model can be used. The convolutional layers are frozen since the reconstructions from the VAEs will not match if the weights in the convolutional layers change. We first create a frozen copy of the complete network at the start of each incremental learning step denoted as the teacher network, to calculate the distillation loss to regularize the update. The trainable network at the current incremental learning step is denoted as the student network. Then, the VAEs generate samples

from previously seen classes combined with the new classes to perform incremental similarity learning on the student network. The autoencoder's reconstruction loss function varies depending on the network's last convolutional layer's activation function. For example, in our case, the last convolutional layers use sigmoid activation. Therefore we used the Binary Cross-Entropy objective function to calculate the reconstruction errors VAEs summed with the Kullback-Leibler divergence. The loss function to update the VAEs is as:

$$L_{VAE} = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(p(y_i)) + (1-y_i)\cdot\log(1-p(y_i)) \\ +\frac{1}{2}(\exp(\sigma^2)+\mu^2-1-\sigma^2)\,, \tag{3.2}$$

where σ^2 is the variance of the full dataset and μ is the mean. N is the total number of values in the output vector of the network. y_i is i^{th} the value of the output vector. $p(y_i)$ represents the probability of y_i. The first term is the Binary Cross-Entropy reconstruction loss and the second term is the Kullback–Leibler divergence. Binary Cross-Entropy reconstruction loss is used for the case whereby the input data and output of the network is in the range of 0 and 1. In the case whereby the inputs are not between 0 and 1, typically mean square error or cross-entropy is used.

We further make use of the angle-wise distillation loss in Equation 3.1 on the generated examples from the VAEs. We only use angle-wise distillation when updating the network during incremental learning similar to the approach by iCaRL [11] and EBLL [12].

Our incremental learning setup makes use of the following loss function during the incremental step defined as:

$$L_{\text{incremental}} = \frac{1}{N}\sum_{i=1}^{N} \ell_{\text{similarity learning}}(x_{s,i}^{all}, y_i^{all}) + \lambda_{\text{our}} \cdot L_A(x_{s,i}^{VAE}, x_{t,i}^{VAE})\,, \tag{3.3}$$

where $\ell_{\text{similarity learning}}$ are the contrastive, angular and, triplet loss functions, we state below. L_A is the angle-wise distillation for similarity learning in Equation 3.1. N refers to the total number of training batches. $x_{s,i}^{all}$ is the output of the student network for both the real images and the VAE samples and their respective labels y_i^{all} for i^{th} batch. $x_{s,i}^{VAE}$ represents the student network outputs for the VAE samples for i^{th} batch. $x_{t,i}^{VAE}$ represents the teacher network outputs for the VAE samples for i^{th} batch. $L_A(x_{s,i}^{VAE}, x_{t,i}^{VAE})$ represents the angle-wise distillation loss between the student and teacher network. λ_{our} is the importance placed on the angle-wise distillation loss shown in Equation 3.1. The student network refers to the network being trained during the current incremental step. The teacher network refers to the frozen copy of the optimal network at the start of each incremental learning step.

We made use of a different distillation loss for center loss that has been presented in research by Li *et al.* [47] and is defined as:

$$L_D(t_o, s_o) = -\sum_{i=1}^{l} t_o^{'(i)} log\left(s_o^{'(i)}\right)\,, \tag{3.4}$$

where l is the number of labels, $t_o^{'(i)}$ and $s_o^{'(i)}$ are the modified versions of the teacher model outputs and the current student model outputs $t_o^{(i)}$, $s_o^{(i)}$.

$$t_o^{'(i)} = \frac{\left(t_o^{(i)}\right)^{(1/T)}}{\sum_j \left(t_o^{(j)}\right)^{(1/T)}}, \quad s_o^{'(i)} = \frac{\left(s_o^{(i)}\right)^{(1/T)}}{\sum_j \left(s_o^{(j)}\right)^{(1/T)}}, \tag{3.5}$$

where T is the temperature and higher value of T increases the weight of smaller logit values and encourages the network to better encode similarities among classes as discussed by Li *et al.* [47]. We use T=2 for our experiments.

The modified loss function during the incremental step for center loss is defined as:

$$L_{\text{incremental}} = \frac{1}{N}\sum_{i=1}^{N} l_{\text{similarity learning}}\left(x_i^{all}, y_i^{all}\right) + \lambda_{\text{our}} \cdot L_D\left(x_{s,i}^{VAE}, x_{t,i}^{VAE}\right), \tag{3.6}$$

where L_D is defined above, $x_{s,i}^{VAE}$ and $x_{t,i}^{VAE}$ now represents output for the VAE samples from the classification layer of student network and teacher network respectively (We can see the classification layer for center loss in our architectures shown in Figure 4.1 and 4.2) because center loss uses softmax as shown in Equation 2.9, and the rest remains the same.

3.3.1 FullVAE

Figure 3.1 shows a high-level diagram depicting our FullVAE approach. FullVAE uses the VAEs that we discussed earlier to generate image representations just before the fully connected layers. The fully connected layer weights are trainable in the incremental step while the feature extractor (convolutional layers) are frozen. We pass the new images from new classes in the incremental learning step through the feature extractors to get image representations for the new images. Then we generate image representations from previously seen classes with the VAEs and concatenate them together to pass them through the fully connected layers. We regularized the network update using the angle-wise distillation loss as shown on Equation 3.3. After each incremental learning step, we train additional VAEs for the new classes using Equation 3.2. Note that the network only has one output layer for all the classes because it is similarity learning. The network outputs a representation for each input image to calculate angle-wise distillation loss between the images. The distillation loss is only used on images from old classes seen in the previous incremental learning step. In our approach, the image representations are generated with the VAEs to regularize the network update.

3.3.2 ConVAEr

Figure 3.2 shows a high-level diagram depicting our ConVAEr approach. ConVAEr is an extension to FullVAE. The difference is that we use the VAEs to generate representations from before an unlocked convolutional layer. We allow one convolutional

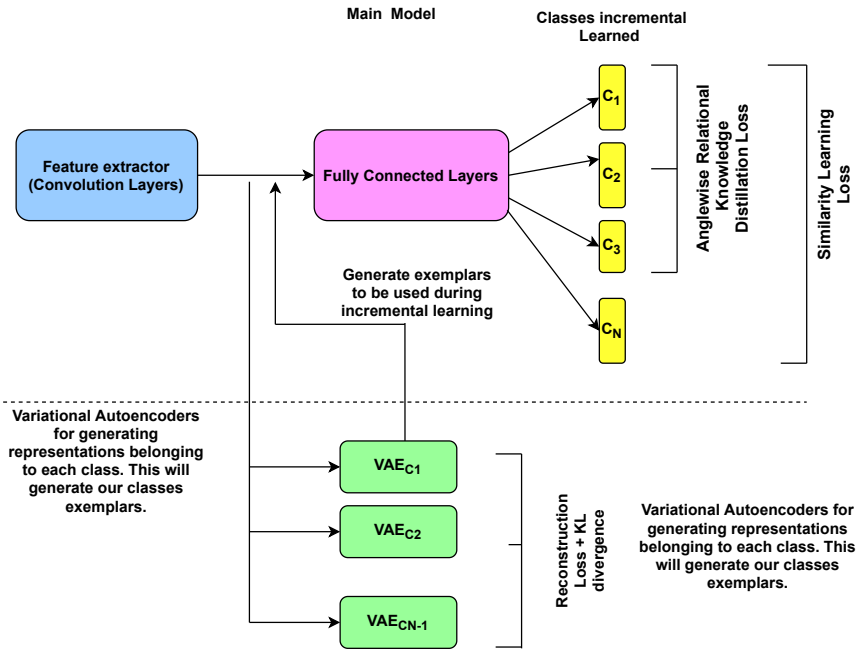


FIGURE 3.1: A diagram demonstrating how our FullVAE approach works. Feature extractor represents the convolutions layers of our network shown in Figure 4.1. After initially training on base-training set, the feature extractor is frozen so that the VAEs would generate useful representations on how the convolutional layers view each class after passed through [2].

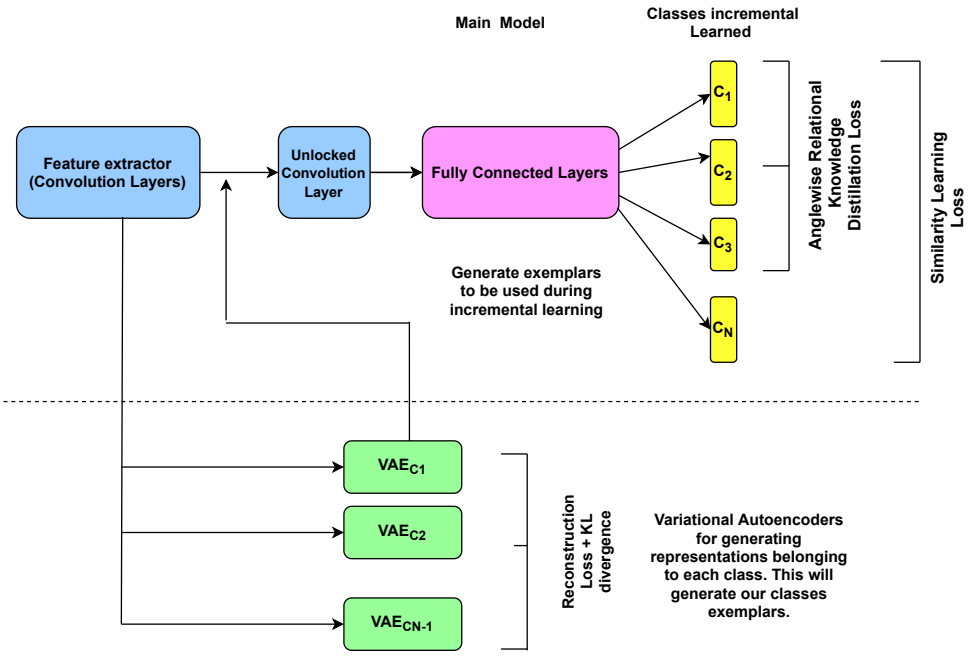


FIGURE 3.2: A diagram demonstrating how our ConvVAEr approach works. The feature extractor is the convolutions layers of our VGG11 network. After initial training on the base-training set the feature extractor is frozen except the last set of convolutional layers.

layer to be trainable. We want to see if we can learn new classes better with the convolutional layer over fully connected layers. Convolutional layers are generally better at capturing good features for images than fully connected layers. The unlocked convolutional and fully connected layers weights are trainable in the incremental step. The feature extractor (convolutional layers) are frozen. The unlocked convolutional layer can update to better capture features that distinguish new classes from the previously learned classes. We pass the new images from new classes in the incremental learning step through the feature extractors to get image representations for the new images. Then we generate image representations from previously seen classes with the VAEs and concatenate them together to pass them through the fully connected layers. We regularized the network update using the angle-wise distillation loss as shown on Equation 3.3. After each incremental learning step, we train additional VAEs for the new classes using Equation 3.2. Note that the network only has one output layer for all the classes because it is similarity learning. The network outputs a representation for each input image to calculate angle-wise distillation loss between the images. The distillation loss is only used on images from old classes seen in the previous incremental learning step. In our approach, the image representations are generated with the VAEs to regularize the network update.

3.4 Conclusion

This section provided details and high-level diagrams depicting how our FullVAE and ConVAEr approach works. The approach was designed for incremental similarity learning, taking inspiration from previous research on incremental learning. We cover the methodology of our experiments in the next section to provide details on the experiment setup and all the instruments we need.

Chapter 4

Research Methodology

4.1 Introduction

This section covers the research design and methodology for our experiments and testing to achieve our aims and objectives. The methodology covers the following contents: the research instruments, modifications needed for the existing methods to work for incremental similarity, datasets, experiment setups, and test metrics we use to measure performance.

4.2 Research Design

We are undertaking experimental research. Experimental research is a study that strictly follows along with scientific research design. It includes a hypothesis, variables that we can manipulate. The variables can be measured, calculated, and compared in a controlled environment. Our research determines if the application of incremental learning techniques to siamese networks help alleviate catastrophic forgetting during incremental learning. We implement and adapt siamese networks with the following incremental learning techniques to similarity learning instead of classification: EWC, rehearsal based technique iCaRL, Encode-based Lifelong learning, and finally, our unique approach that takes inspiration from knowledge from the previous literature. We use the techniques for incremental similarity training to learn new classes incrementally. We measure and compare the performances of the four methods against one another to see how well they reduce catastrophic forgetting and whether our novel approach outperform existing methods.

4.3 Method

4.3.1 Research Instruments

- CUB200 dataset [22] and CARS196 [23] dataset.
- MNIST [18], FashionMNIST [19], EMNIST [20], and CIFAR10 [21] datasets.
- Python 3.6 programming language to implement the coding for the study. Pytorch and Tensorflow library for Python is used for forwarding training and backpropagation of the siamese networks.
- Pytorch Metric Learning library [48].

- Scikit learn Python library [49].
- Steps to building the VGG11 and Resnet9 network architecture.
- Weight consolidation algorithm for controlled parameter updates such as the Elastic Weight Consolidation (EWC) algorithm [13]. Algorithm for rehearsal incremental-learning techniques iCaRL [11], and Encoder-based lifelong Learning [12] algorithm to which we can adapt to similarity learning.
- Angle-wise distillation loss and classification distillation loss algorithm for regularization.
- Similarity learning test metric 4.6 and incremental learning test metrics [24].

4.3.2 Modifications of the Existing Methods

The EWC, EBLL and iCaRL methods we discussed in Chapter 2 were initially designed for incremental classification learning. Therefore, we need to make modifications to make the methods suitable for incremental similarity learning. We cover the required changes for each of the approaches.

4.3.2.1 EWC

We computed the Fisher information matrix different for similarity learning because similarity does not make use of likelihood probabilities, like classification. Instead, we computed the Fisher Information Matrix using the gradient of the calculated loss of respective similarity loss functions as shown in Section 2.9. Therefore the Fisher Information Matrix is calculated as:

$$F = \frac{1}{N}\sum_{i}^{N} \nabla_\theta L(b_i|\theta) \nabla_\theta L(b_i|\theta)^T \,, \tag{4.1}$$

where N is the number of training batches, b_i represents the i^{th} training batch, and θ is the network parameters .$\nabla_\theta L(b_i|\theta)$ represents the gradient computed using Pytorch or Tensorflow by backpropagating the loss computed using the training batch b_i with respect to the network parameters, θ. After computing the Fisher Information Matrix, the loss to update the network is:

$$L = \ell_{\text{similarity learning}} + \sum_{i} \frac{\lambda_{\text{ewc}}}{2} F_i(\theta_{t,i} - \theta^*_{t-1,i})^2 \,, \tag{4.2}$$

where $\ell_{\text{similarity learning}}$ is the loss four loss functions defined in Section 2.9. F_i is the Fisher Information Matrix computed using Equation 4.1. λ_{ewc} is the importance of the new and old knowledge. θ_{t-1}, i is the network parameters from the previous incremental training step. $\theta_{t,i}$ is the trainable parameters from the current training step.

4.3.2.2 EBLL

EBLL approach was initially designed for incremental classification learning and therefore, we have to modify how the EBLL works for incremental similarity learning. Previously the authors used EBLL for incrementally learning new tasks [12] but in our case we are incrementally learning new classes. Therefore we need modifications for the EBLL to suit our purpose of incremental similarity learning. Firstly, the incremental learning update for the encoders has to be modified to suit our purpose. Therefore, we have formulated a modified loss function. The loss for optimizing the autoencoder for similarity learning is:

$$L_{ae} = \lambda_{ae} \cdot \left|\left| r\left(F^*\left(X_i^{(T)}\right)\right) - F^*\left(X_i^{(T)}\right)\right|\right|_2^2 + \ell_{\text{similarity learning}}\left(X_i^{(T)}, Y_i^{(T)}\right), \tag{4.3}$$

where L_{ae} is the total loss of the autoencoder. Let

$$\left|\left| r\left(F^*\left(X_i^{(T)}\right)\right) - F^*\left(X_i^{(T)}\right)\right|\right|_2^2,$$

be the reconstruction mean square error loss. With X_i^T representing the input image batch, $Y_i^{(T)}$ represents the labels for the image batch. $F^*\left(X_i^{(T)}\right)$ represents the optimal feature extractor output for image X , and $\ell_{\text{similarity learning}}$ is the respective similarity learning loss functions we would be using as shown in Section 2.9, and λ is the importance we put on the reconstruction loss as stated in the paper by Rannen *et al.* [12]. We use the optimized feature extractor layers before each incremental learning step to train the autoencoders using Equation 4.3.

Furthermore, similarity learning does not use a classification layer but rather a shared output layer. Therefore, we do not need to add additional output neurons to the output layer. Distillation loss is different for similarity because we are not performing classification; the distillation loss from Hinton *et al.* [35] does not suit our purpose. Instead, we used angle-wise distillation as shown in Equation 3.1. We do not have different task-specific operators for different tasks because we are not learning tasks incrementally but rather classes. We only have one shared output layer because it is similarity learning. Therefore, we do not use the sum of all distillation loss like previously shown in Equation 2.3. However, we are still using all autoencoders to perform the update. Hence the modified EBLL loss for similarity learning is:

$$\begin{aligned} L = & \frac{1}{N}\sum_{i=1}^{N}\Bigg(\ell_{\textbf{similarity learning}}\left(X_i^{(T)}, Y_i^{(T)}\right) \\ & + \ell_A\left(T\circ T\circ F\left(X_i^{(T)}\right), T^*\circ T^*\circ F^*\left(X_i^{(T)}\right)\right) \\ & + \sum_{t=1}^{T-1}\frac{\alpha_t}{2}\left|\left|\sigma\left(W_{enc,t}F\left(X_i^{(T)}\right)\right) - \sigma\left(W_{enc,t}F^*\left(X_i^{(T)}\right)\right)\right|\right|_2^2\Bigg), \end{aligned} \tag{4.4}$$

where N is the number of training batches in the each incremental step. We do not have separate task specific operators for similarity learning therefore $T \circ T \circ F$ represents the full network updated at each incremental step. $X_i^{(T)}$ represents the i^{th}

train batch in the T^{th} incremental step with the batch labels Y_i^T. The similarity loss:

$$\ell_{\textbf{similarity learning}}\left(X_i^{(T)}, Y_i^{(T)}\right) ,$$

for the training image batch X_i with batch labels Y_i using the different loss functions from Section 2.9. The angle-wise distillation loss:

$$\ell_A\left(T \circ T \circ F\left(X_i^{(T)}\right), T^* \circ T^* \circ F^*\left(X_i^{(T)}\right)\right) ,$$

between the student and teacher network uses Equation 3.1. In the case of center loss, the distillation loss used is shown by Equation 3.4. $T \circ T \circ F(X_i^{(T)})$ represents the output of the network currently being trained (student network) for image batch $X_i^{(T)}$. $T^* \circ T^* \circ F^*(X_i^{(T)})$ represents the output of the frozen copy of optimal network made before each incremental train step (teacher network) for image batch $X_i^{(T)}$. $W_{enc,t}$ represents the encoding portion of the autoencoders up to the bottleneck layer for t^{th} incremental step. $F\left(X_i^{(T)}\right)$ is the feature extractor outputs for the image $X_i^{(T)}$ from the student network. $F^*\left(X_i^{(T)}\right)$ is the feature extractor outputs for the image $X_i^{(T)}$ from the teacher network.

$$\sum_{t=1}^{T-1} \frac{\alpha_t}{2} \left|\left|\sigma\left(W_{enc,t}F\left(X_i^{(T)}\right)\right) - \sigma\left(W_{enc,t}F^*\left(X_i^{(T)}\right)\right)\right|\right|_2^2 ,$$

represents the sum of the mean square error of the encoded outputs between the teacher and student feature extractor given the image X_i^T. α_t is the importance we put on the encoded loss. In our experiments we set α_t to be always 2.

4.3.2.3 iCaRL

We implemented the exemplar herding algorithm as shown in the iCaRL paper to retain exemplars. The main point of iCaRL is keeping some exemplars of previously learned classes for each incremental step. iCaRL is designed to be used with a classification network that uses cross-entropy, so for similarity, we made use of the four similarity loss functions shown in Section 2.9 instead. The distillation loss presented and used by iCaRL is a modification of cross entropy [35] that is not suitable for similarity learning. We use a different form of distillation loss more suitable to similarity learning as shown in Equation 3.1. The distillation loss is only used on images from old classes seen in the previous incremental learning step. A frozen copy of the network is made before each incremental learning step is denoted as the teacher network. In contrast, the trainable network is the student network for the current incremental step. In this case of iCaRL, the exemplars passed through the network during the incremental learning step to constrain the update. The loss for iCaRL approach to update the trainable network during the incremental steps is:

$$L_{\text{iCaRL}} = \frac{1}{N}\sum_{i=1}^{N}\left(\ell_{\text{similarity learning}}\left(x_{s,i}^{all}, y_i^{all}\right) + \lambda_{\text{iCarL}} \cdot L_A\left(x_{s,i}^{Exemplars}, x_{t,i}^{Exemplars}\right)\right) , \tag{4.5}$$

where $\ell_{\text{similarity learning}}$ are the contrastive, angular, center and triplet loss functions. L_A is the angle-wise distillation for similarity learning in Equation 3.1. In the case of center loss, the distillation loss used is shown by Equation 3.4. N is the total number of training batches. $x_{s,i}^{all}$ represents the student network outputs for both the new images and the exemplar images in i^{th} training batch. λ_{iCarL} is the importance placed on the distillation loss. $x_{s,i}^{Exemplars}$ and $x_{t,i}^{Exemplars}$ are the student and teacher network outputs for only the exemplars that are in the i^{th} training batch so we can calculate distillation loss between the student and teacher outputs.

4.3.3 Datasets

4.3.3.1 Motivating Datasets

To analyze the impact of catastrophic forgetting in incremental similarity learning and to motivate the purpose of our study, all methods are subjected to incremental learning scenarios on well-known datasets. The datasets used are MNIST, EMNIST, FashionMNIST, and CIFAR10. MNIST is handwritten digits [18]. EMNIST is upper and lower case letters from the English alphabet [20]. We regard the upper/lower cases as coming from the same class to increase the difficulty. Fashion-MNIST obtained images of clothes and shoes from Zalando's articles [19]. CIFAR10 is a low-resolution objects [21]. The MNIST classes are unbalanced, but the sample sizes for each class are very similar. For all other datasets, the classes are balanced.

4.3.3.2 Primary Datasets

We subjected the two best methods to incremental similarity learning scenarios on well-known datasets to evaluate the real-world impact of catastrophic forgetting for similarity learning. The datasets utilised are CUB200 [22] and Cars196 [23] . The CUB200 and CARS196 are widely used for classification [50] and similarity learning [51] and are suitable for incremental similarity learning. Ordinarily, the first half of the classes are used to train the network, while the second half is utilized for testing.

4.3.4 Experimental Setup

4.3.4.1 Preprocessing

Preprocessing done on the MNIST and EMNIST dataset comprised normalizing the pixel values with the mean and standard deviation [0.1307, 0.3801]. For Fashion-MNIST, we normalize the pixel values with mean and standard deviation [0.2860, 0.3530]. On CIFAR10, the pixels were normalized with the mean and standard deviation [(0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)]. Additionally, padding of size 4 and data augmentation through horizontal flipping during training was used. The padding and augmentation are used to overcome the dataset's toughness and, in so doing, obtain sufficient performance to demonstrate the effects of catastrophic forgetting. Pre-processing done on the CUB200 and CARS196 datasets included normalizing the pixel values with mean and standard deviation [(0.485, 0.456, 0.406), (0.229, 0.224, 0.225)]. Additionally, padding of size 4 and data augmentation by

horizontal flipping when training was used. The additional padding and augmentation are used to overcome the dataset's complexity and, in so doing, obtain sufficient performance to demonstrate the negative effects of catastrophic forgetting.

4.3.4.2 Training Sample Mining

Training Pairs and Triplets were generated online. Triplets were mined employing semi-hard negative mining on positive and negative image pairs. To create pairs of images for contrastive loss, we performed pair margin mining. An angular miner was used to generate triplets for angular loss that are of an angle greater than α. All mining was performed using the Pytorch Metric Learning library [48] with the hyper-parameters specified in Section 4.3.4.3.

4.3.4.3 Hyper-parameters

The positive and negative margins for contrastive loss were $[0, 0.3841]$ for CUB200 and $[0.2652, 0.5409]$ for CARS196. These margins were obtained from Musgrave *et al.* [51]. For triplet loss, the margin for CUB200 and CARS196 were $[0.0961, 0.1190]$, respectively. The hyper-parameters [λ, α] for centre loss were [1.0, 0.5] respectively previously shown to have excellent outcomes for class separation [42].

The angle, α, for the angular loss in our implementation was 45° previously found as optimal for object detection [42] for MNIST and EMNIST datasets. An angle of $[35, 40]$ was used for CIFAR10 and FashionMNIST, obtained by grid-search on a range between 30 and 50 in steps of 5. The margin for contrastive and triplet loss functions were 1.0 and 1.25, respectively. The margin was determined through grid-search on the range 0.25 to 2.0 in steps of 0.25. All grid searches were done using the validation set from FashionMNIST, MNIST, and EMNIST. The margins for contrastive (0.3841) and triplet (0.0961) loss for CIFAR10 were adapted from Musgrave [51] who used these for CUB200. CUB200 is similar to the CIFAR10 dataset in terms of complexity. We used an importance value of 150 for λ_{ewc} across all the experiments for EWC. We weighted the similarity learning loss and distillation loss equally for iCaRL and our approach by setting λ_{iCaRL} and λ_{our} to 1.0 . We set λ_{ae} to 1.0 when training the autoencoders for EBLL.

4.3.4.4 Exemplars for iCaRL

Rebuffi *et al.* [11], used 2000 exemplars for CIFAR100, which results in an average ratio of 20 images per class. Therefore in our experiments, we limited the total number of exemplars for MNIST, FashionMNIST, and CIFAR10 to 200 exemplars. For EMNIST, we limited the number of exemplars to 520 for the 26 classes.

There is a limited number of images per class for CUB200 and CARS196 (60 or fewer images per class), which is substantially less than CIFAR100. Therefore, we allow iCaRL to retain 480 exemplars from both datasets' during training. The retained exemplars roughly account for 5% and 6% of the total training images from 160 classes for CARS196 and CUB200. This percentage of retained exemplars is initially equivalent to 9% of the images per class and slowly decreases as additional classes are incrementally learned until the 5/6 % is reached.

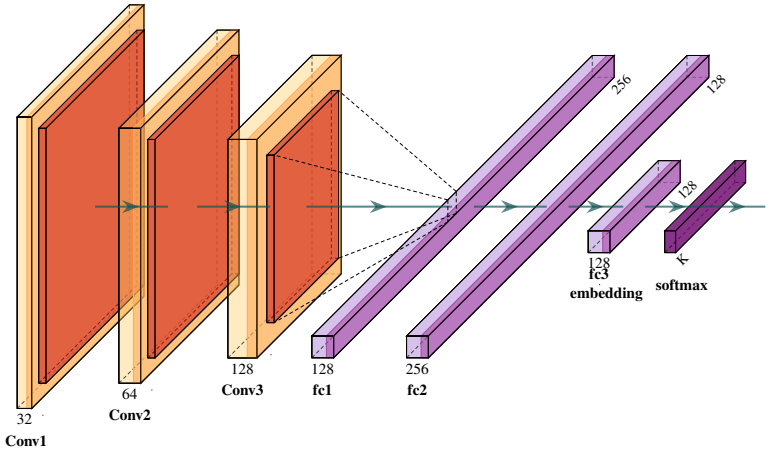


FIGURE 4.1: Architecture of our convolution neural network. The yellow layers represents the convolution, the orange represents the pooling layers, and the purple are fully connected layers. The K refers to the number of classes in the data. Note that the softmax fully connected layer is only used for Center Loss. For the rest of the loss functions, we only use the network up to fc3.

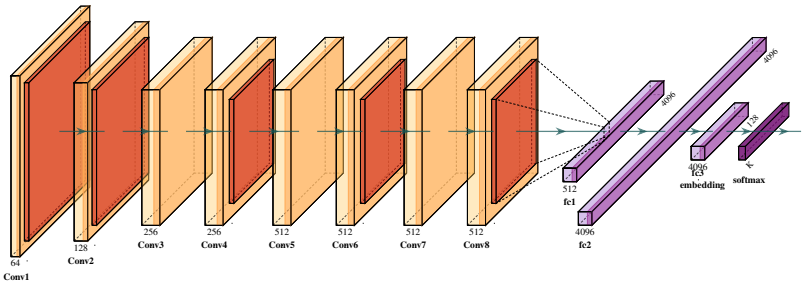


FIGURE 4.2: Convolution neural network Architecture. Yellow represents the convolution layers, orange the pooling layers, and purple the fully connected layers. K is the number of classes. The softmax fully connected layer is only used for Center Loss. For the other loss functions, we use the network up to fc3.

4.3.5 Architecture of the CNNs and VAEs

4.3.5.1 Architecture for Motivating Datasets

Our simple convolutional neural network, as shown in Figure 4.1, made use of three 2D convolutional layers with max-pooling layer between each of them. We add a flatten layer at the end of the last max-pooling layer. Finally, we added three linear layers to the network. Every layer is ReLU activated beside the output layer, which is linearly activated. The output layer is of size 128. We have ensured that the simple network performed reasonably well on MNIST, FMNIST, EMNIST, and CIFAR10 to observe the effect of catastrophic forgetting. For our FullVAE method, we used the sigmoid activation function for the last 2D convolutional layer instead of ReLU. The process allows binary cross-entropy as our reconstruction loss for our VAEs. Our

VAE architecture consisted of an input layer of size 512 (same size as out after the convolution layers), followed by 256, 128, and 128 for the bottleneck. Then these layers are reconstructed symmetrically in reverse. All kernels for the convolution layers are of size $(3, 3)$.

4.3.5.2 Architecture for Primary Datasets

For the experiments with the primary datasets, we utilized a VGG-11 backbone deep neural network architecture as show in Figure 4.2. After the convolution layers, the first fully connected layer were changed from the original 4096 to 512 size, the output layer was modified to 128 for similarity learning purposes. Thus, the network serves as an example of a commonly employed deep neural network architecture to assess the real-world performances of the respective methods more accurately. For our FullVAE approach, our VAE architectures consisted of an input layer of size 512 (same size as out after the convolution layers), followed by 256, 128, and 128 for the bottleneck. Then these layers are reconstructed symmetrically in reverse. For our ConVAEr approach, our VAE architecture consisted of a 2D Convolution layer with 512 input channels, 2D Convolution with 32 input channels, 2D Convolution with 16 input channels, and finally a bottleneck linear layer of input size 1600 and 256 output. These layers are reconstructed symmetrically in reverse so they can perform reconstructions with deConvolutional layers. All kernels for the deConvolution and convolution layers are $(3, 3)$ size.

4.3.6 Training and Testing

4.3.6.1 Setup on Motivating Datasets

The section covers the training setup we have used for MNIST, FMNIST, EMNIST, and CIFAR10 on the simple CNN shown in Figure 4.1 and Resnet9 architecture. We used a minor variation of the training procedure for incremental learning to that of Kemker *et al.* [24]. We start with half (5) of the classes from each dataset: MNIST, Fashion-MNIST, and CIFAR10. For EMNIST, half of the classes is 13. Subsequent training contains data from a single new unseen class. A high-level overview of the steps, followed, are:

1. The motivating datasets already provide testing data separate from the training data. We take all the classes in the training data and split them into two sets of classes. A baseline-set and an incremental-set of classes.

2. I split the samples in the baseline-set into base-training and base-validation data sets (80/20).

3. I split the samples in the incremental-set into inc-training and inc-validation data sets (80/20).

4. To get our base-test and inc-test data sets, we extract the images from the testing data belonging to two sets of classes (base-training set and inc-training sets).

5. I use the base-training and base-validation set to train our initial base models for incremental learning.

6. I take one unseen class from our incremental set of classes and the last seen class from the previous incremental learning step to supplement the unseen class.

7. I retrain our base model with the inc-training data set for the unseen class and the last seen class.

8. I use the base-test data set from our baseline set to record the mAP@R (mean Average Precision at R) after each step. We use the inc-test to record mAP@R for Ω_{new} after each step. I use base-test plus inc-test to record mAP@R for Ω_{all} after each step. Each of the equations is present in Equations 4.7.

9. We repeat from step 6 until all of the incremental sets' classes are exhausted.

Since similarity learning loss functions require at least two classes, we take the last seen class from the previous training step to pair with the new class. All the models were trained for a maximum of 50 epochs. The best models were selected using early stopping on the validation set. The Adam optimizer was used with a learning rate of 0.001, β_1 value of 0.9, and β_2 value of 0.999. For the FullVAE method, we trained one VAE for each class that the network has seen for each incremental train step. The same Adam optimizer setup was used for training the VAEs. The EBLL method consisted of training one autoencoder after each incremental class learning step using the same Adam optimizer setup specified above.

We randomly split the data into two sets of classes consisting of baseline set and incremental set. We can get an average result for different combinations of class splits of incremental learning by doing this. Then, we repeat the experiment ten times for each incremental learning method model on each dataset (total: 800) using randomly seeded baseline and incremental splits as previously specified to get average performance for each method on each dataset. We use the same training and validation splits to save the best models for each method to keep results consistent. In addition, the base-test and inc-test sets splits are the same for each incremental learning method to keep results consistent.

The models' output is a feature representation of size 128 per image evaluated using mean Average Precision at R (mAP@R). Average precision at R (AP@R) is calculated using a single query identity used to retrieve the top R related relevant images from the database. The AP at R is given by:

$$\text{AP@}R = \frac{1}{\text{R}} \sum_{k=1}^{R} \text{P@}k \times \text{Rel@}k, \tag{4.6}$$

where R is the total number of images in the database that belong to the same class as the query image. P@k refers to the precision at k and Rel@k is a relevance score which equals 1 if the document at rank k is relevant and equals 0 otherwise. mAP is the average of the AP over all possible image queries and k up to R.

4.3.6.2 Setup on Primary Datasets

The section covers the training setup that we use for incremental similarity learning on CUB200 and CARS196 on VGG11. We used a similar procedure for incremental learning to that of Kemker *et al.* [24]. We start with 120 of the classes from each of the datasets: CUB200 and CARS196. We decided to start with 120 of the classes during our base-training is to start the network with moderately good base knowledge before incremental learning. The process will allow us to observe the severity of catastrophic forgetting on the base knowledge during the incremental learning steps. Subsequent training contains data from a single new unseen class up to 160 classes. A high-level overview of the steps, followed, are:

1. The primary datasets do not provide testing data separate from the training data. We take all the classes and split them into two sets of classes. A baseline-set and the incremental-set of classes.

2. I split the baseline-set into base-training and base-test data sets (80/20).

3. I split the incremental-set into inc-training and inc-test data set (70/30).

4. I use the base-training and train our initial base models for incremental learning.

5. I take one unseen class from our incremental set of classes and the last seen class from the previous incremental learning step to supplement the unseen class.

6. I retrain our base model with the inc-training data set for the unseen and last seen class.

7. I use the base-test data set from our baseline set to record the mAP@R after each step. We use base-test plus inc-test to record mAP@R for Ω_{all} for all classes seen so far after each step. I use the inc-test to record mAP@R for Ω_{new} after each step. We use the unseen classes from 161 to 200 on CUB200 and 161 to 196 on CARS196 to record mAP@R for Ω_{unseen}. Each of the equations is presented in Equations 4.7 and 4.8.

8. I repeat from step 5 until all of the incremental sets' classes are exhausted.

As with the training setup for motivating datasets in Section 4.3.6.1, we take the last seen class from the previous training step to pair with an all-new class. The Adam optimizer was used with a learning rate of 0.001 was used for base set training and a learning rate of 0.0001 for incrementally learning new classes, β_1 value of 0.9, and β_2 value of 0.999. The Adam optimizer was used for iCaRL and FullVAE approaches. We use the SGD optimizer for ConvVAEr because it is better for the method with the same learning rate as the Adam optimizer. For FullVAE and ConVAEr, we trained one variational autoencoder for each class that the network has seen for each incremental train step. The Adam optimizer was used for training the VAEs for FullVAE and ConVAEr with a constant learning rate of 0.001. After training, we measure the mean average precision (mAP@R) on the new class to assess if the models are still learning. All the models were trained for 200 epochs for the base

set training and 100 epochs for inc-set training. Due to the limited training samples in each class, we could not do early stopping with validation. However, we first experimented with validation during "dummy" training to check for the best average epochs needed to reach decent performance before we started our experiments. We settled on 200 and 100 epochs for the base set and inc-set training, respectively.

We randomly split the data into two groups of classes consisting of a baseline set and an incremental set using three random seeds. We repeated and averaged the performance over the three random seeds. Thus, each run consisted of the same training and testing splits of the data to keep results consistent between all methods, and we can average the performance.

The models' output is a feature representation of size 128 per image evaluated using mean Average Precision at R (mAP@R). Average precision at R (AP@R) is calculated using a single query identity used to retrieve the top R related relevant images from the database. The mAP at R is defined in the Equation 4.6.

4.3.7 Analysis

The test metrics we used are a minor variation to the incremental class evaluation metrics of Kemker *et al.* [24] to support learning based on distance rather than classification. We measured performance (mAP@R) on the base-test set after learning classes incrementally. We use mAP@R since, in similarity learning, the loss functions we used do not make use of a classification layer, but instead, the networks are used to retrieve all related images of a query. mAP@R is considered a superior metric for similarity learning in this regard [51]. We tracked the model's performance on the new class to ensure that it was still learning. We measured how well a model retains prior knowledge and learns new knowledge by measuring the mean mAP@R performance on each class learned so far during each training session. We used mAP shown in Equation 4.6 to measure the performance of our model instead of the classification accuracy metric as our models learn based on distance.

The evaluation metrics used are modified to include mAP@R, versions of the original metrics by Kemker *et al.* defined as:

$$\begin{aligned} \Omega_{base} &= \frac{1}{T-1}\sum_{i=2}^{T}\frac{\alpha_{base,i}}{\alpha_{ideal}} \\ \Omega_{new} &= \frac{1}{T-1}\sum_{i=2}^{T}\alpha_{new,i} \\ \Omega_{all} &= \frac{1}{T-1}\sum_{i=2}^{T}\frac{\alpha_{all,i}}{\alpha_{ideal}} \end{aligned} \tag{4.7}$$

,where T is the total number of training sessions, $\alpha_{new,i}$ is the test mAP@R for session i immediately after it is learned. $\alpha_{base,i}$ is the test mAP@R on the first session (base-test set) after ith new sessions have been learned. $\alpha_{all,i}$ is the test mAP@R of all of the inc-test data and base-test set for the classes seen so far. α_{ideal} is the offline model mAP on the base-test set, which is the ideal performance. Ω_{base} measures a model's retention of the base knowledge after sequential training sessions. Ω_{new} measures the model's performance on new classes. Ω_{all} indicates how well a model

both retains prior knowledge and acquires new information (how well we retrieve new learnt class among previously seen classes). Ω_{base} and Ω_{all} are normalized with α_{ideal}. The evaluation metrics are between [0,1] unless the results exceed the offline model. The offline models are trained on all the data. mAP@R is evaluated by extracting a feature embedding of size 128 for every image. The images' embeddings are compared to all other embeddings and ranked in order of cosine similarity. Each test image was treated as a query once while comparing it's similarity to the remaining images.

The training setup for primary datasets in Section 4.3.6.2 covered incremental learning up to 160 classes. However, CUB200 and CARS196 have a total of 200 and 196 classes, respectively. We have left out 40 classes from CUB200 and 36 classes from CARS196. We use the remaining classes to observe if unseen classes performance improves from incremental learning. We introduce a new metric to measure improvement or decay on unseen classes performance and is defined as:

$$\Omega_{unseen} = \frac{1}{T-1}\sum_{i=2}^{T}\frac{\alpha_{unseen,i}}{\alpha_{\text{unseen_base}}} \tag{4.8}$$

where T is the total number of training sessions, $\alpha_{unseen,i}$ is the test mAP@R for the unseen classes of the CUB200 and CARS196 datasets immediately after session i has been learned. $\alpha_{\text{unseen_base}}$ is the test mAP@R for the unseen classes after training on the first session with the base-train set. Normalizing by $\alpha_{\text{unseen_base}}$ helps us to observe any performance gains or decay on the unseen classes due to incremental learning steps over compared to just training on the base-train set of data. In the case of this evaluation metric, if Ω_{unseen} is below 1 it means there is decay in unseen classes performance when we incremental train for the respective methods. If Ω_{unseen} is above 1, it means we are improving unseen classes performances during incremental learning.

4.3.8 Hardware and software

We used two machines. An Intel(R) Xeon(R) CPU E5-2683 processor and an Intel(R) Core(TM) i7-5820K, both have 32 GB of RAM, and a GTX 1080 TI 11GB GPU. Both machines used Linux, Python 3.6, Pytorch 1.7.1 [52], Scikit-learn 0.23.2, and Pytorch Metric Learning 0.9.97 [48].

4.4 Conclusion

In this section, we have covered the methodology and method we followed to conduct our study. First, it outlines the various research instruments we need to implement the incremental learning techniques to the siamese networks. Second, it consisted of the algorithms, software, and datasets we implemented to answer our research questions. Third, the methodology covered the structure of the output data that will produce by the experiments. Finally, it covers how we interpret the output results using existing information retrieval and catastrophic forgetting test metrics. The next section will cover the results and discussions of our research experiments.

Chapter 5

Results and Discussions

5.1 Evaluating Catastrophic Forgetting

The following section involves the investigation and discussion of how catastrophic forgetting affects similarity learning. We first look into the extent of catastrophic forgetting during incremental similarity learning without the aid of any learning techniques. We investigated four well-known similarity learning loss functions: angular, contrastive, triplet, and center-loss on the simple CNN shown in Figure 4.1. The experiments followed the training setup on motivating datasets shown in Section 4.3.6.1.

5.1.1 Pair and Triplet Mining

In the following experiments, only semi-hard triplet mining was used for mining triplets for training with triplet loss. Random positive and negative pairs were mined for contrastive and angular loss. Center loss does not need mining techniques. Later down in the results section, we show the importance of good pair and triplet mining for all the other loss functions to reduce catastrophic forgetting.

5.1.2 Base and Overall Performance

Figure 5.1 shows to what extent catastrophic forgetting affects initial base knowledge learned during incremental similarity learning. The offline ideal models are trained with all available classes and tested on the base-test set to get the ideal mAP. We get the random model performance by randomly initializing weights on the CNN and measuring the mAP retrieval performance on the base-test set. We can observe that most models drop below the random model's performance after learning two to three new classes due to forgetting. However, triplet loss is the exception, with mAP on base set dropping much slower, not falling below random performance.

Figure 5.2 shows how well each loss functions retain previously and newly learnt information by testing on the base-test set plus the inc-test sets from all the classes learned. Random performance was obtained by testing the random model on entire test set (base-test set and inc-test set combined). Ideal performance was obtained by testing the offline ideal model on the entire test set.

The results further support the findings in Figure 5.1 that each similarity loss function is affected differently by catastrophic forgetting. The angular loss was affected the worst, and triplet loss was the affected least. We notice a steeper drop in performance depending on the number of classes learned, as shown on the EMNIST

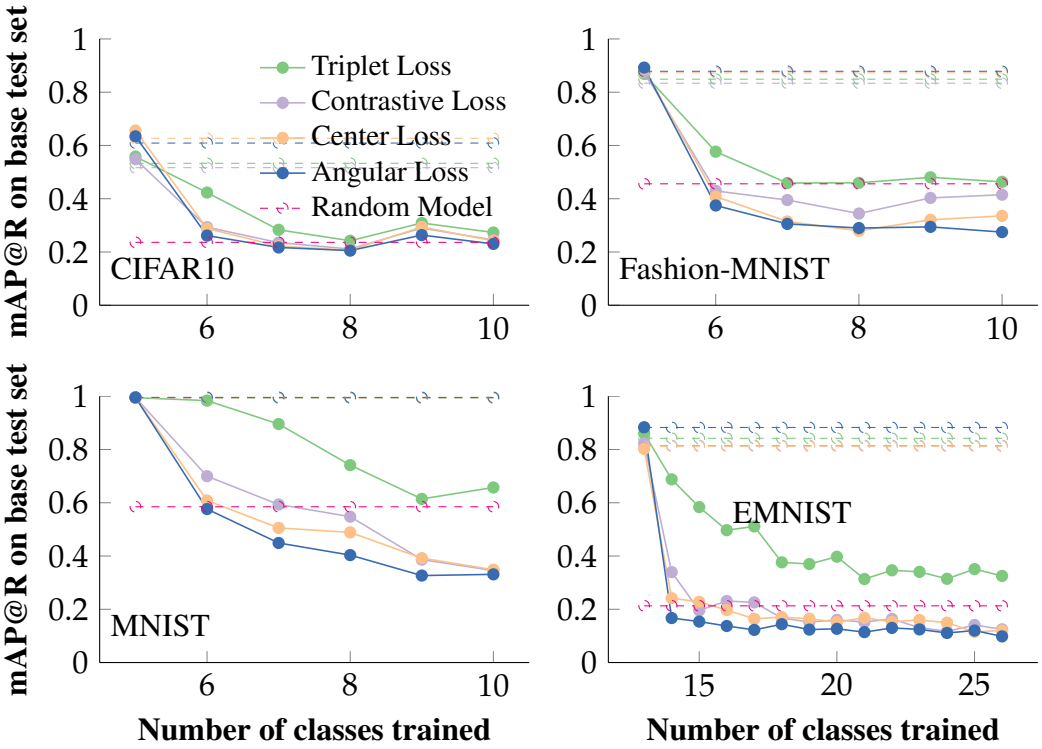


FIGURE 5.1: Left to right, top to bottom: CIFAR10, Fashion-MNIST, MNIST, EMNIST. Mean average precision on base-test set. The solid lines indicate the incremental learning models. The dotted lines of the same colour refer to the performance of the respective offline ideal models [1]

dataset. The complexity of the dataset equally affects the steepness of the drop, as seen in the CIFAR10 case.

Figure 5.1 and 5.2 illustrates the extent of catastrophic forgetting during incremental learning using the four different loss functions without the aid of any incremental learning techniques. All the loss functions drop to random performance during incremental learning except for triplet loss. We also observe that how well the network is able perform on the datasets affects how much prior knowledge is forgotten. The least being MNIST, followed by FashionMNIST, EMNIST and finally CIFAR10. The results in this experiment showed that triplet loss has the least amount of forgetting compared to the rest of the loss functions given the current training setups. We observe in later sections the importance of mining techniques and how it affects forgetting.

5.2 FullVAE

This section covers the experiment results on the importance of mining techniques for the different loss functions during incremental learning. We also cover the implementation of the four different incremental learning techniques consisting of EBLL, EWC, iCaRL, and our FullVAE approach. Finally, we compared how they perform

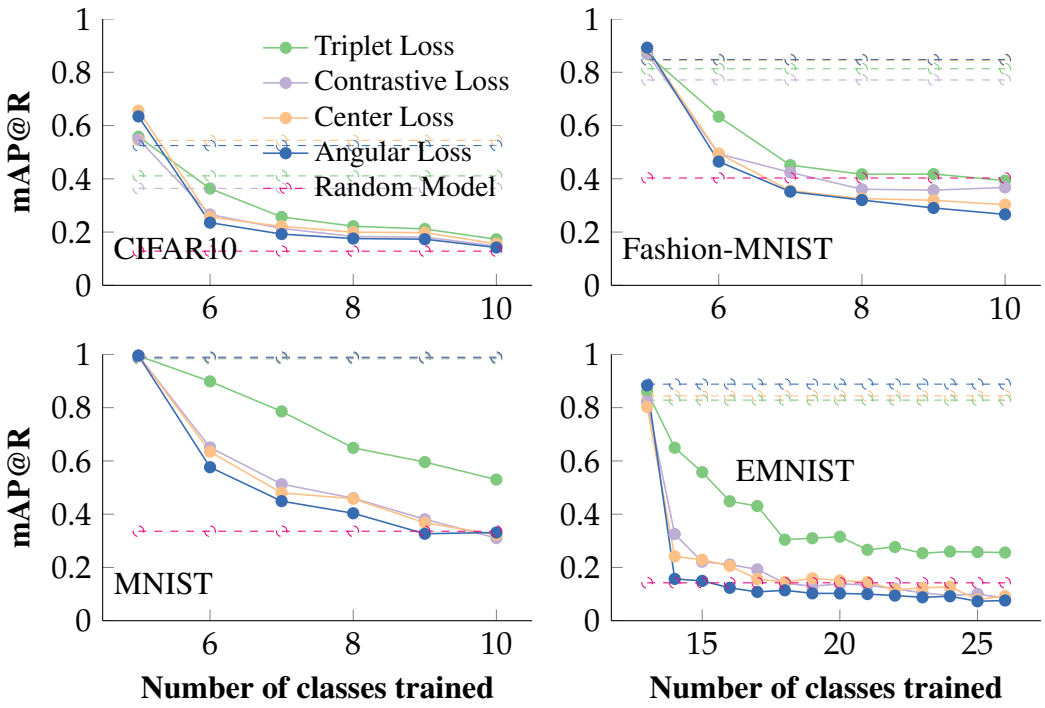


FIGURE 5.2: Left to right, top to bottom: CIFAR10, Fashion-MNIST, MNIST, EMNIST. Mean average precision on classes learned so far. The mAP retrieval test was performed on the base-test plus inc-test sets from all the classes learned. The solid lines indicate the incremental learning models while the dotted lines of the same colour refer to the respective offline ideal models [1].

against one another using the same training setup on motivating datasets in Section 4.3.6.1.

5.2.1 Importance of Mining Techniques

In the following experiments, we have implemented appropriate pair and triplet mining for all the loss functions besides center loss that does not use any mining techniques during training. We made use of pair margin mining for contrastive loss, semi-hard mining for triplet loss, and finally, angular mining for angular loss.

Figure 5.3 shows to what extent catastrophic forgetting affects initial base knowledge learned during incremental learning. The earlier results in Section 5.1 showed that triplet loss is less affected by catastrophic forgetting when compared to other similarity learning functions. However, there were no special mining techniques for contrastive and angular loss in the earlier experiments. We introduced a pairwise and angular miner for contrastive and angular loss in the current setup. With the correct setup of pairs and triplets mining for the similarity learning functions, the results show that the various methods suffer from catastrophic forgetting at approximately the same rate. Figure 5.3 shows that centre loss is still the most impacted by catastrophic forgetting than the other similarity learning loss functions. The EMNIST

results highlight the complete forgetting of base knowledge if there has been no effort to preserve knowledge over multiple steps of incremental class learning. The results reinforce this research's motivation. We note a steeper drop in performance dependent on the number of classes previously learned, as shown for the EMNIST dataset. The complexity of the dataset equally affects the drop's steepness, as seen in the CIFAR10 case. We see earlier and more severe catastrophic forgetting in more complex datasets.

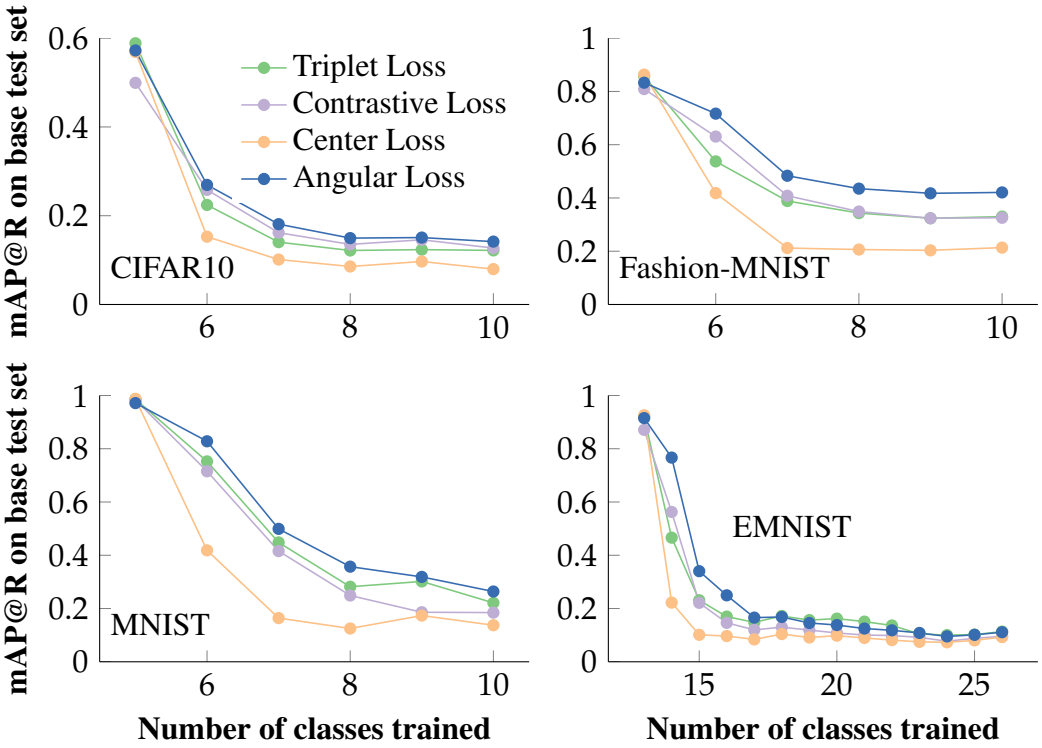


FIGURE 5.3: Mean average precision on base-test set without using any incremental learning techniques. Using appropriate mining techniques for each method [2].

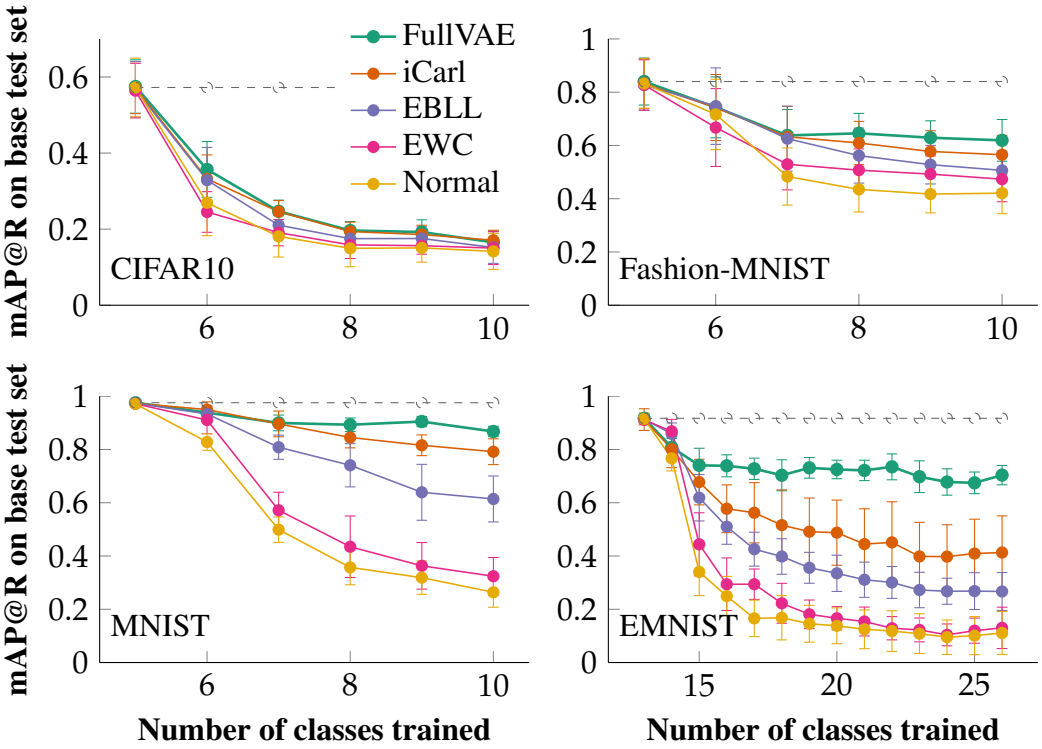


FIGURE 5.4: The figure compares the four different incremental learning approaches to the baseline normal train for **Angular loss**. Mean average precision (mAP@R) on base-test set. Solid lines indicate incremental learning models. Dotted line indicates offline ideal [2].

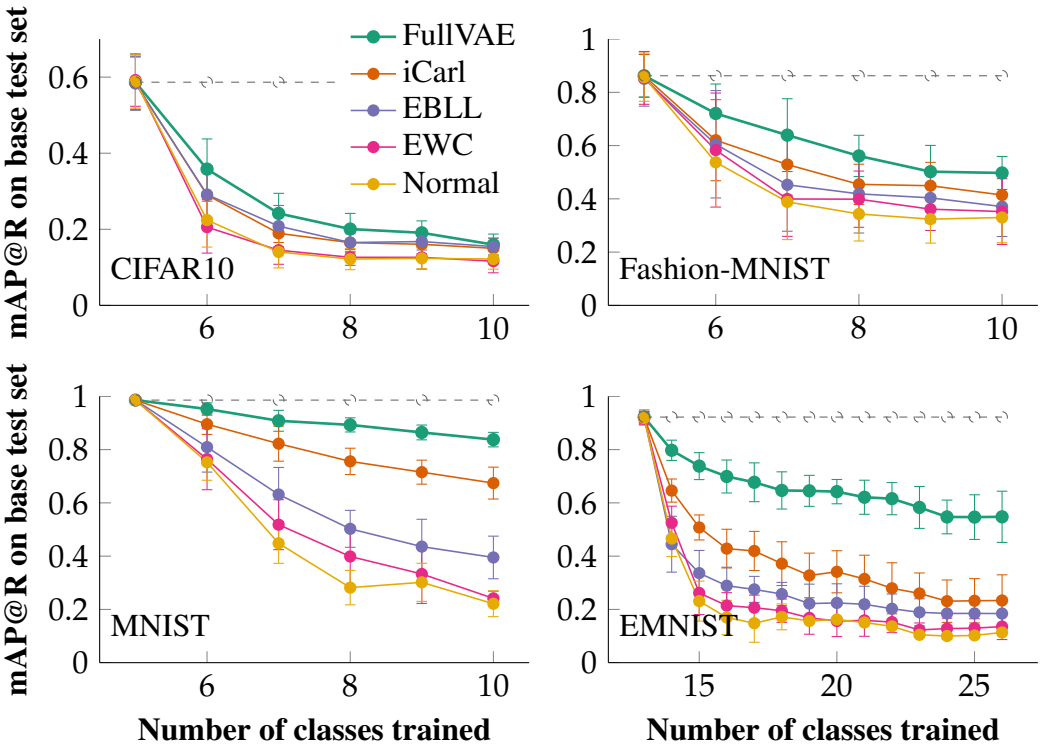


FIGURE 5.5: The figure compares the four different incremental learning approaches to the baseline normal train for **Triplet loss**. Mean average precision (mAP@R) on base-test set. Solid lines indicate incremental learning models. Dotted line indicates offline ideal [2].

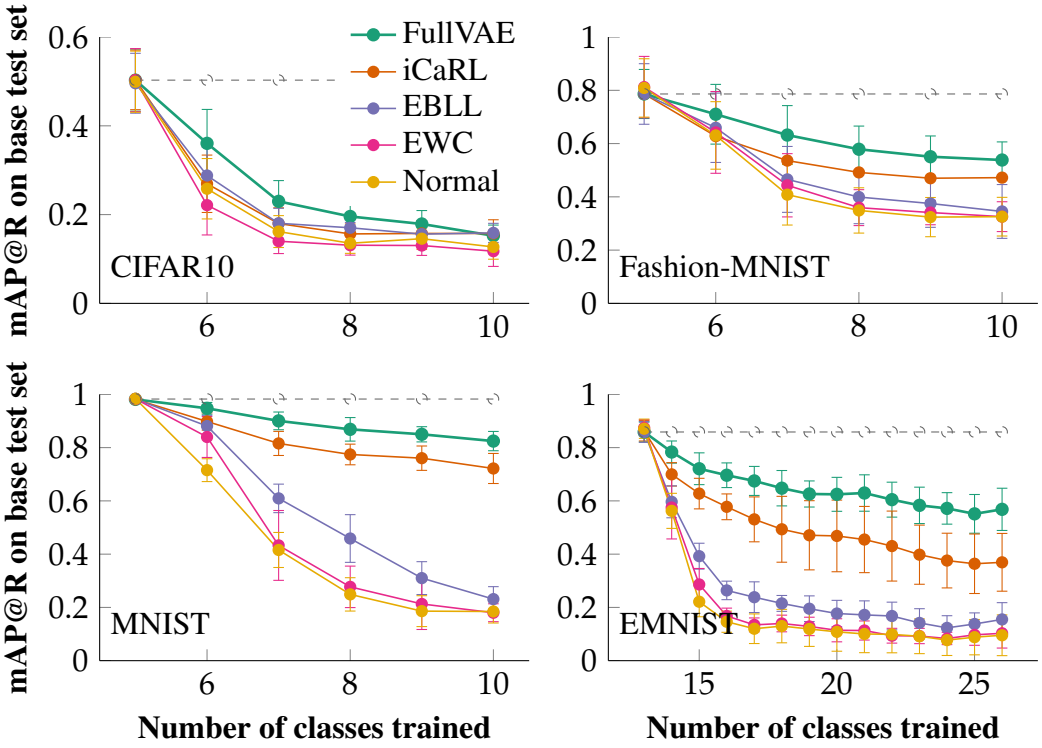


FIGURE 5.6: The figure compares the four different incremental learning approaches to the baseline normal train for **Contrastive loss**. Mean average precision (mAP@R) on base-test set. Solid lines indicate incremental learning models. Dotted line indicates offline ideal [2].

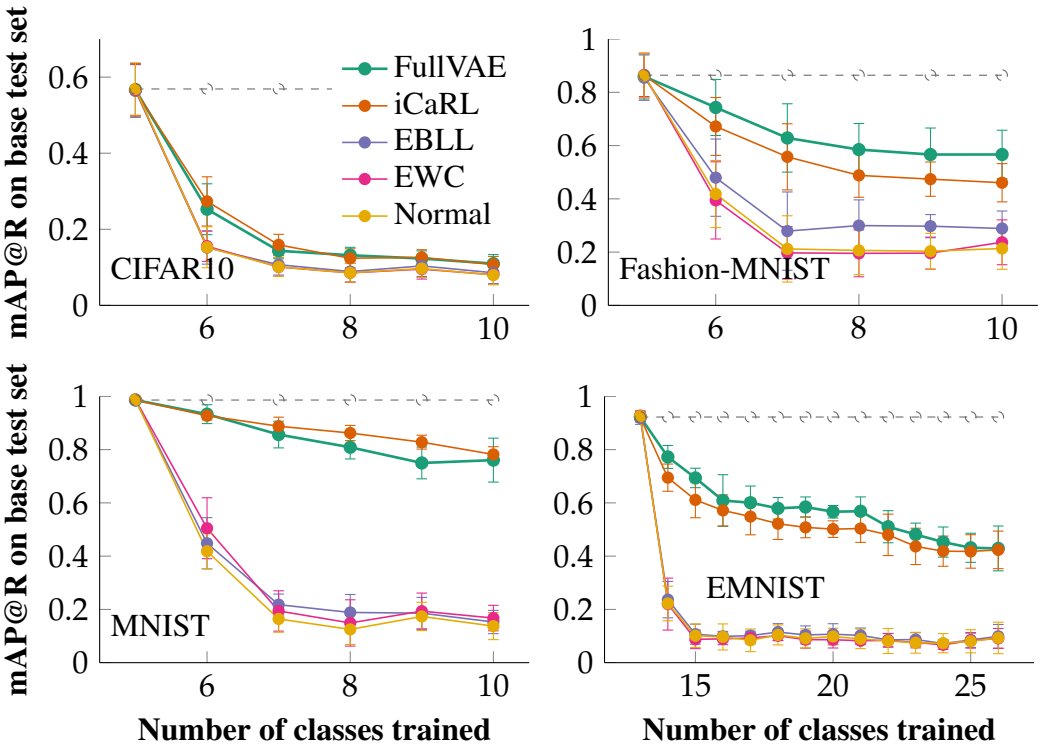


FIGURE 5.7: The figure compares the four different incremental learning approaches to the baseline normal train for **Center loss**. Mean average precision (mAP@R) on base-test set. Solid lines indicate incremental learning models. Dotted line indicates offline ideal [2].

5.2.2 Performance on Base Test Set

Figures 5.4, 5.5, 5.6, and 5.7 highlight how each of the methods implemented reduces catastrophic forgetting when learning classes incrementally by testing on a base-test set after each new class is introduced. Offline models were trained on all available classes and tested on the base-test set to get the ideal mAP@R shown by a dotted line. The normal models are trained without the aid of any incremental learning techniques. In Figure 5.4 to 5.7, we observe that EWC and EBLL do not provide many benefits for incremental class learning but still retain more initial knowledge than incremental training normally would. We see that EWC was the least effective in retaining base knowledge across all datasets and loss functions. We note that the EWC technique is still effective for a smaller number of incremental learning steps. Alternatively, EBLL provided decent base knowledge retention without exemplars and suffered less from catastrophic forgetting over a more significant number of incremental steps. The EMNIST results in Figure 5.4 also illustrate some of the challenges associated with the iCaRL method's retaining of exemplars. First, as the number of exemplars from each previously seen classes decreases, the network suffers from increased catastrophic forgetting. Secondly, variations of images are essential and keeping only exemplars closest to the class's mean does not represent a class well. Finally, the effectiveness of the exemplars seems to depend on how the mining and loss function work together. The figures show differences in rates of forgetting between the loss functions with the same number of exemplars for iCaRL.

However, the overall best performers for gaining new knowledge and retaining knowledge are iCaRL and FullVAE as supported by the plots and reinforced further by the discussion around Table 5.1.

TABLE 5.1: Incremental class test's mean average precision (mAP@R) starting from the memorised Ω_{base} with the new classes Ω_{new} added and the overall result Ω_{all}. Bold is best method for Ω_* per loss. Highlight is best method for Ω_* per dataset.[2]. † indicates that the best method is not significant at $p < 0.05$ using the student t-test when compared to the second best method.

		Normal			**EWC**			**EBLL**			**iCaRL**			**FullVAE**		
Loss	**Dataset**	Ω_{base}	Ω_{new}	Ω_{all}	Ω_{base}	Ω_{new}	Ω_{all}	Ω_{base}	Ω_{new}	Ω_{all}	Ω_{base}	Ω_{new}	Ω_{all}	Ω_{base}	Ω_{new}	Ω_{all}
Contrast'	CIFAR10	.33	.14	.20	.29	.16	.19	.38	.11	.21	.36	**.22†**	.24	**.44†**	.19	**.31**
	MNIST	.35	.58	.36	.39	.59	.37	.50	.47	.43	.80	**.90**	**.91†**	**.89**	.86	.85
	Fashion-M'	.51	.49	.36	.53	.45	.37	.57	.46	.37	.66	**.84†**	**.60**	**.76**	.73	**.61**
	EMNIST	.17	.23	.15	.19	.23	.16	.26	.27	.22	.56	**.55**	.51	**.74**	.54	**.69**
Angular	CIFAR10	.31	.16	.22	.31	.11	.19	.36	.13	.23	.39	**.20**	.27	**.40**	.18	**.39**
	MNIST	.46	.47	.45	.53	.51	.43	.76	.51	.60	.88	**.85**	**.87**	**.92†**	**.85**	.85
	Fashion-M'	.58	.52	.42	.63	.53	.40	.70	.49	.45	.74	**.74**	.55	**.77†**	.72	**.56**
	EMNIST	.22	.24	.30	.27	.23	.21	.43	.23	.30	.55	.57	.47	**.78**	**.63**	**.70†**
Triplet	CIFAR10	.24	.13	.17	.24	.12	.16	.33	.13	.21	.32	**.19**	**.23**	**.39**	.16	**.23**
	MNIST	.40	.52	.40	.45	.62	.50	.56	.41	.47	.78	**.90**	.77	**.90**	.84	**.84**
	Fashion-M'	.44	.41	.25	.48	.48	.27	.52	.42	.31	.57	**.74**	.49	**.67**	.73	**.52**
	EMNIST	.18	**.52**	.16	.21	.23	.16	.26	.16	.18	.38	.41	.36	**.69**	**.52**	**.60**
Center	CIFAR10	.18	.10	.12	.18	.10	.13	.18	.10	.14	**.27**	**.16**	.19	.26	.15	**.26**
	MNIST	.20	.31	.15	.24	.53	.27	.24	.63	.30	**.86**	**.91†**	**.85**	.83	.85	**.85**
	Fashion-M'	.29	.48	.20	.28	.43	.20	.38	.50	.25	.61	**.76**	.50	**.71**	.69	**.59**
	EMNIST	.10	.19	.09	.10	.20	.10	.11	.25	.10	.55	**.59**	.49	**.60**	.52	**.52**

Table 5.1 presents the evaluation metric results using Equation 4.7 for each of the models. The values: Ω_{base}, Ω_{new}, and Ω_{all} range between $[0, 1]$. 0 indicates the model retains no knowledge, and 1 indicates it retains all knowledge. The Ω_{new} results show the mAP@R performance on test data of the newly learned class. The Ω_{all} shows how well the models retain prior knowledge and acquire new knowledge. The Ω_{new} results show the normal models are learning new knowledge at a very low rate, and it would not be useful to be used as it is. In Table 5.1 we evaluated how methods retained previously and newly learnt knowledge by testing on base-test set (old learned classes) and inc-test set (newly learned classes). The results are normalized with the offline models' ideal performance using Equation 4.7. The offline ideal models' performances were obtained by measuring the mAP@R on the base-test set.

5.2.3 Impact of Loss

We observe from bold and highlighted results in Table 5.1 that the four loss functions with the FullVAE approach provided the best performance in terms of retaining base knowledge. The exception is Center loss, where iCaRL outperformed FullVAE in some cases as shown by Ω_{base} results. The Ω_{new} results showed that iCaRL mostly outperformed all other approaches when learning new classes. Overall, the bold and highlighted results of Ω_{all} shows that we mostly perform better overall across all loss functions. We do observe that different loss functions have different extents of forgetting during incremental learning. The best performer was angular, followed by contrastive, triplet, and center loss. Although the different incremental learning techniques reduced the amount of forgetting, we can still observe differences in the extent of forgetting of the four different loss functions. Angular loss combined with FullVAE is the best performer overall, followed by contrastive, triplet, and center loss. The results align with earlier observations. The different losses have a different impact on forgetting, even with good incremental learning techniques.

Table 5.1 show FullVAE as the most robust over multiple steps of incremental class learning, as highlighted by the EMNIST Angular results. Since we are not required to use actual images as exemplars, we can still represent a class well during incremental learning. VAEs are noisy, but they can still represent previously learned classes well. The ability to represent previous classes is vital due to the similarity learning loss functions prioritising separating classes from each other and forming regions [42], [53], [54] for the new class to occupy in the embedding space. We observe in Table 5.1 that methods that do not preserve some form of information about previously learnt knowledge are more adversely impacted by catastrophic forgetting.

We observe some interesting differences between iCaRL and our FullVAE approach. FullVAE outperforms iCaRL overall in terms of overall knowledge retention, but iCaRL is better in learning new classes. Overall, Ω_{all} shows our FullVAE approach is better on average mAP@R across all learnt classes. The results further supports that we are not required to have images as exemplars to represent previously learnt classes to preserve previous knowledge's embedding space. We can represent images in the form of a representation that can be passed through intermediate layers and get similar or better performance compared to iCaRL. However, we observe in

the CIFAR10 results that simple VAEs might not represent detailed images on more complicated datasets.

Finally, Table 5.1 showed that angular loss retained the most base knowledge followed by the contrastive, triplet, and center loss as shown by Ω_{base} value. Again we note that loss functions with correct mining perform similar to each other in contrast to previous results in Section 5.1. The results reinforces the importance of good miners for similarity learning loss functions.

5.2.4 Impact of Architecture

In this section, we explore if these outcomes might be attributable to the network architecture rather than the methods and losses.

TABLE 5.2: Incremental class test's mean average precision (mAP@R) starting from the memorised Ω_{base} with the new classes Ω_{new} added and the overall result Ω_{all}. Bold is best method for Ω_* per loss. Highlight is best method for Ω_*. [2]

Architecture	Loss	Normal			EWC			EBLL			iCaRL			FullVAE		
		Ω_{base}	Ω_{new}	Ω_{all}	Ω_{base}	Ω_{new}	Ω_{all}	Ω_{base}	Ω_{new}	Ω_{all}	Ω_{base}	Ω_{new}	Ω_{all}	Ω_{base}	Ω_{new}	Ω_{all}
ResNet9	Contrast'	.24	.12	.13	.25	.13	.15	.35	.12	.20	.30	**.14**	.19	**.68**	.05	**.36**
	Angular	.31	.12	.18	.41	.12	.24	.43	.12	.26	.34	**.13**	.21	**.74**	.07	**.36**
	Triplet	.25	.13	.16	.26	.12	.16	.34	.12	.20	.35	**.13**	.21	**.57**	.07	**.27**
	Center	.21	.10	.12	.23	**.11**	.14	.20	.11	.10	.22	**.11**	.14	**.47**	.05	**.24**

To assess if the network architecture might be a factor in the above results, we select a ResNet9 [55] as representative of an alternative deep neural network architecture. ResNet9 is a 9 layer deep residual network. A building block of a ResNet is called a residual block. A residual block is simply when the activation of a layer is fast-forwarded to a deeper layer in the neural network. We select the CIFAR10 dataset as we have previously observed this to be the worst-performing dataset for FullVAE. We repeat the above experimental setup once with the same metrics using CIFAR10 and a ResNet9. In Table 5.2 we note that the results are representative of those in Table 5.1, with similar techniques outperforming for each metric. FullVAE remains the best overall performing technique. It is worth noting that the results are normalized with respect to a baseline ideal and, as such, not directly comparable with Table 5.1 due to Resnet9 achieving a higher baseline idea compared to the simpler CNN.

We see that in Table 5.2 that the ordering of methods, other than FullVAE, has changed when compared with Table 5.1. Comparing the results in Table 5.1 and 5.2 shows an interesting perspective on how different architectures affects the effectiveness of each technique. We can observe that on the simple network, Figure 4.1, the order of best performing technique to worst was first FullVAE, iCaRL, EBLL, and finally EWC. For the Resnet9 architecture, we can observe that the best performance to worst were FullVAE, EBLL, iCaRL, and EWC.
There are significant differences between each class of CIFAR10, and base convolution knowledge learnt may not necessarily transfer well to new classes during incremental learning. We speculate that when it comes to learning new classes, the frozen convolution layers of our FullVAE approach limit its ability to learn new features.

The new features could be important for the network to differentiate between new and old classes. The results inspire our ConVAEr approach, covered in the following section.
Further, we observe that the ResNet9 architecture improves most methods' ability to retain base knowledge at the cost of new knowledge. These results are exciting as they point to another possible area of research for catastrophic forgetting in incremental similarity learning: network architecture.

5.3 ConVAEr

Results in the Table 5.1 in Section 5.2 showed the two best performing techniques as iCaRL and our FullVAE. Therefore to extend our investigation to represent real-world applications better. We expanded our FullVAE approach into ConVAEr as shown in Figure 3.2 to see if we get better performance on the primary datasets. The experiments are done on the more complex primary datasets using iCaRL, FullVAE, and ConVAEr approaches on VGG11 network architecture shown in Figure 4.2. The results better represent how the techniques would fare in real-world scenarios. We made use of the training setup in Section 4.3.6.2. We expand on the FullVAE approach by investigating using VAE to represent outputs from intermediate convolution layers. We have only investigated three loss functions: contrastive, triplet, and center loss, due to difficulty applying angular loss on CUB200 and CARS196. We experimented with numerous angles for angular loss but still could not train models successfully with the angular loss on CUB200 and CARS196.

5.3.1 Pair and Triplet Mining

In the following experiments, we have implemented appropriate pair and triplet mining for all the loss functions besides center loss that does not use any mining techniques during training. In addition, we have made use of pair margin mining for contrastive loss and semi-hard mining for triplet loss.

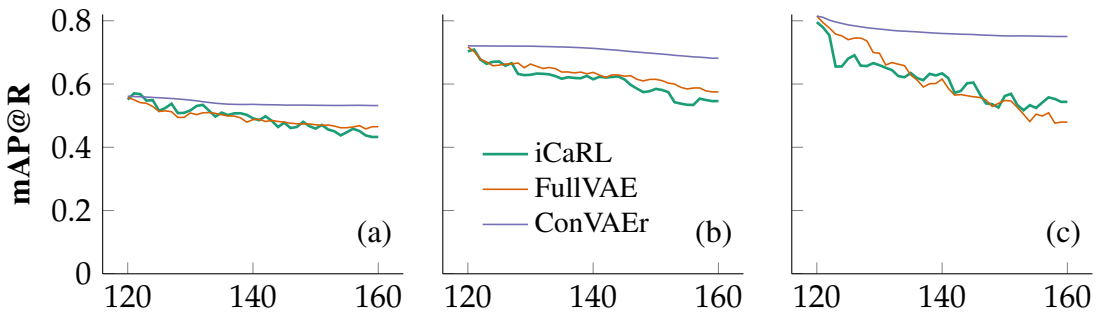


FIGURE 5.8: **CARS196**: mean average precision (mAP@R) on base-test for *(a) Triplet*, *(b) Contrastive*, and *(c) Center* loss combined for each incremental learning step.

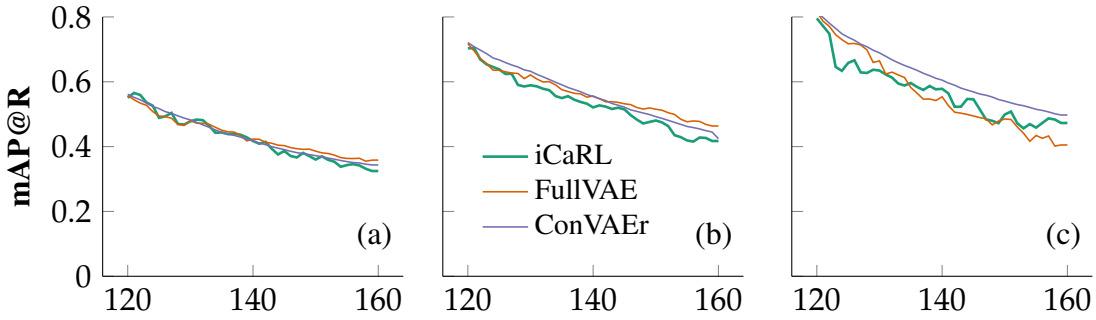


FIGURE 5.9: **CARS196**: mean average precision (mAP@R) on base-test plus inc-test for *(a) Triplet*, *(b) Contrastive*, and *(c) Center* loss combined for each incremental learning step.

5.3.2 Knowledge Preservation

Figures 5.8, and 5.9 highlight how each of the methods implemented reduces catastrophic forgetting when learning classes incrementally by testing on a base-test set after each new class is introduced. We observe that the FullVAE and ConVAEr retain better base knowledge compared to iCaRL during incremental learning. We further observe from Figures 5.8 that as progress with incremental steps, iCaRL performance drops quicker compared to the FullVAE and ConVAEr due to the number of samples for each class being reduced.

TABLE 5.3: Incremental class test's mean average precision (mAP@R) starting from the memorised Ω_{base} with the new classes Ω_{new} added and the overall result Ω_{all}.

		Ω_{base}			Ω_{new}			Ω_{all}		
Dataset	**Loss**	**iCARL**	**F'VAE**	**ConVAEr**	**iCARL**	**F'VAE**	**ConVAEr**	**iCARL**	**F'VAE**	**ConVAEr**
CUB-200	Contrast'	0.8461	0.9272	**0.9920**	0.2102	0.1299	0.1243	0.7380	0.7856	**0.7879**
	Triplet	0.8593	0.8651	0.9710	0.2282	0.2084	0.1300	0.7476	0.7531	0.7784
	Center	0.8008	0.7564	0.8380	0.3020	**0.3052**	0.0985	0.7235	0.6990	0.6761
CARS-196	Contrast'	0.8683	0.8800	**0.9814**	0.4148	0.3843	0.1911	0.7573	**0.7805**	0.7801
	Triplet	0.8952	0.8809	0.9623	0.3467	0.2961	0.1658	0.7683	0.7765	0.7615
	Center	0.7650	0.7499	0.9393	**0.5006**	0.4219	0.1620	0.7090	0.6879	0.6173

Table 5.3 shows the evaluation metric results utilising Equation 4.7 for all of the models. The values: Ω_{base}, Ω_{new}, and Ω_{all} span within $[0, 1]$. 0 suggests the model retains no knowledge, and 1 means it preserves all knowledge. The Ω_{new} results show the mAP@R performance on the test data of the newly learned class. The Ω_{all} presents how well the models recall prior knowledge and obtain new knowledge. The Ω_{new} results show the models are learning new knowledge very slowly and would not be useful as presented. The results are normalized with the offline models' ideal performance using Equations 4.7.

The results in Table 5.3 show our ConVAEr approach as the most robust over multiple steps of incremental class learning for retaining base knowledge. Even though ConVAEr is noisy and does not use actual images as exemplars, it can still preserve a class well during incremental learning. However, we observe that center loss does

not perform as well as contrastive and triplet loss with VAEs exemplars and has difficulty learning new classes. We observe that ConVAEr is better than iCaRL in terms of overall knowledge retention (Ω_{all}), but iCaRL is better at learning new classes. We observe that center loss has the highest drop in base knowledge on FullVAE and iCaRL. However, they gain a higher rate of new knowledge. The results support our argument that we can preserve previous knowledge's embedding space by representing images in a representation that can be passed through intermediate layers. We get similar or better performance than iCARL. Finally, in Table 5.3 we see that contrastive loss retains the most base knowledge followed by triplet and center loss as shown by Ω_{base} value.

The results in Table 5.3 show that iCaRL, FullVAE, and ConVAEr are on par with one another. It is a contrast to earlier results observed in Table 5.1 and 5.2 where we observed more significant differences between the techniques. The difference in performances between the different setups indicates that architecture plays a significant role in how each incremental technique performs to reduce catastrophic forgetting.

5.3.3 Performance on Unseen Classes

The following section presents the Ω_{unseen} results on the iCaRL, FullVAE, and ConVAEr on CUB200 and CARS196 dataset with the VGG11 architecture using Equation 4.8. As a reminder, Ω_{unseen} refers to the average mAP@R performance on the unseen classes on CUB200 and CARS196 after each incremental training step. The Ω_{unseen} results are normalized by the base trained models unseen classes performance. The normalization allows us to observe any decay or improvements from incrementally introducing the models with new classes. The results in Table 5.4. The Ω_{unseen} less than 1 indicates decay in unseen classes performance while more than 1 indicates improvement in unseen classes performance.

TABLE 5.4: Incremental class test's mean average precision (mAP@R) for Ω_{unseen} following Equation 4.8

		Ω_{unseen}		
Dataset	**Loss**	**iCARL**	**F'VAE**	**ConVAEr**
CUB-200	Contrast'	0.7355	0.9525	**1.004**
	Triplet	0.7822	0.8236	0.9853
	Center	0.5597	0.5528	0.4134
CARS-196	Contrast'	0.8247	0.8751	0.9799
	Triplet	0.9140	0.9689	**1.0116**
	Center	0.5661	0.4785	0.5056

The results in Table 5.4 shows that ConVAEr's performance decayed the least with unseen performance, followed by FullVAE and lastly, iCaRL. The exception being the center loss function. Furthermore, triplet loss on Cars196 and contrastive loss on CUB200 using the ConVAEr approach improved the unseen classes performances over the model that has only trained on the base classes. Taking into account the results from both Table 5.3 and 5.4, the decay in base performance, Ω_{base}, are

mostly correlated to decay in performances on unseen classes, Ω_{unseen}. The models utilizing the FullVAE and ConVAEr approach were the most robust to unseen class performance decay and even, in some cases, improved. Except for center loss having the much higher deterioration in unseen performance, Ω_{unseen}, compared to the base performance, Ω_{base}. The results further reinforce that center loss is not a suitable loss function for incremental similarity learning. Finally, we can conclude that even though the Ω_{base} and Ω_{new} performance drops as we incrementally train. It does not always result in the decay of unseen classes performance. With ConVAEr, we can observe some improvement over not incrementally training the model.

Chapter 6

Conclusion

In this research, we investigated to what extent similarity-based loss functions were affected by catastrophic forgetting during incremental similarity learning. We do this by implementing a minor variation of the existing catastrophic forgetting testing procedure and testing metric by Kemker *et al.* to accommodate testing similarity learning.

Firstly, we experimented on motivating datasets that we can easily achieve moderately high accuracy on. The results show that the extent of catastrophic forgetting was different on each of the respective loss functions. Angular loss was the least affected amongst the loss functions with good pair and triplet mining techniques for the loss functions. The centre loss was the most affected. We have found that the severity and rapidness of forgetting depend on the complexity of data (i.e. number of classes, the task's difficulty) in similarity-based loss functions. Therefore, we have shown that retrieval models with similarity-based loss functions are unsuitable for incremental learning without any techniques to aid the training procedure.

Secondly, we adapted iCaRL, EBLL and EWC and our FullVAE approach to address the issue. The results showed that all the methods helped reduce forgetting in the networks compared to regular training. FullVAE outperformed the other three methods when measured on base knowledge retention and overall knowledge retention on motivating datasets. FullVAE method differentiates from traditional exemplar methods that require us to keep images. Since using VAEs to generate actual images is complicated, we proposed using VAE to generate intermediate representations before the fully connected layers. The results show that we do not require actual images as exemplars during incremental learning. Instead, it is essential to remind the network of the previously seen knowledge. Thirdly, we investigated how the four methods would perform on Resnet9 architecture on CIFAR10 to verify if the behaviour is similar to the experiment on a simple CNN, which held.

Next, we extended the FullVAE approach to ConVAEr, taking the convolutional layer into account. We compared ConVAEr to iCaRL and FullVAE on primary datasets using the VGG11 network. We investigated the effectiveness of each method using contrastive, center, and triplet loss functions. The results showed that contrastive loss with ConVAEr was most robust to catastrophic forgetting on the primary datasets followed by triplet loss. However, iCaRL was more effective when used with center loss. iCaRL was also more effective at learning new classes. Finally, our ConVAEr and FullVAE approaches were better than iCaRL on unseen class performance on CUB200 and CARS196.

The results showed that catastrophic forgetting in incremental similarity learning remains a complex research area. However, our research highlighted different aspects that caused the forgetting.

6.1 Future work

We observe from Table 5.3 that iCaRL performed slightly better than FullVAE when learning new classes but vastly outperformed ConVAEr in some cases. ConVAEr should perform well for new classes with an unlocked convolutional layer inline with or better than FullVAE. The unlocked convolutional layer should learn better features for new images compared to just fully connected layers. We theorize that the VAE representation for simpler intermediate layers (fully connected layers) of the network was more accurate than the VAE representation from intermediate layers (CNN layers). Therefore, representation from the intermediate layers could be more difficult to replicate. The increased difficulty can lead to noisier VAE representations that make it difficult to learn new classes, which we can observe from Table 5.3. We speculate the need for a better VAE architecture or possibly the use of GANs for ConVAEr.
The datasets and network architecture combination could also affect the performance of the incremental learning techniques. The datasets and network architecture combination could also affect the performance of the incremental learning techniques. The CIFAR10 results in Tables 5.1 and 5.2 provided evidence to our speculation. We observed that the ordering of methods has changed. But that requires further investigation on this particular issue to be certain. In addition, we highlighted possible further areas for investigation concerning the choice of network architectures as one of the main targets to overcome catastrophic forgetting in incremental similarity learning. Finally, the loss function and mining function choice also matter. There is also future work possibility for designing loss functions suited for incremental learning. In general, different architectures will give different performances for different data sets. It seems that the network used was most appropriate for EMNIST data (given a loss only after 10 incremental classes). The NN architecture selected, merits further study

Bibliography

[1] J. Huo and T. L. Van Zyl, "Comparative analysis of catastrophic forgetting in metric learning," in *2020 7th International Conference on Soft Computing Machine Intelligence (ISCMI)*, 2020, pp. 68–72. DOI: 10.1109/ISCMI51676.2020.9311580.

[2] J. Huo and T. L. van Zyl, *Incremental class learning using variational autoencoders with similarity learning*, 2021. arXiv: 2110.01303 [cs.LG].

[3] N. A. Syed, S. Huan, L. Kah, and K. Sung, "Incremental learning with support vector machines," 1999.

[4] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*, vol. 31, no. 4, pp. 497–508, 2001.

[5] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*, vol. 24, Elsevier, 1989, pp. 109–165.

[6] R. Ratcliff, "Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions.," *Psychological review*, vol. 97, no. 2, p. 285, 1990.

[7] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," *arXiv preprint arXiv:1312.6211*, 2013.

[8] C. Fernando, D. Banarse, C. Blundell, *et al.*, "Pathnet: Evolution channels gradient descent in super neural networks," *arXiv preprint arXiv:1701.08734*, 2017.

[9] A. Robins, "Catastrophic forgetting, rehearsal and pseudorehearsal," *Connection Science*, vol. 7, no. 2, pp. 123–146, 1995.

[10] T. J. Draelos, N. E. Miner, C. C. Lamb, *et al.*, "Neurogenesis deep learning: Extending deep networks to accommodate new classes," in *2017 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2017, pp. 526–533.

[11] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "Icarl: Incremental classifier and representation learning," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2017, pp. 5533–5542.

[12] A. Rannen, R. Aljundi, M. B. Blaschko, and T. Tuytelaars, "Encoder based lifelong learning," in *2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE Computer Society, 2017, pp. 1329–1337.

[13] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[14] H. Thanh-Tung and T. Tran, "On catastrophic forgetting in generative adversarial networks," *arXiv preprint arXiv:1807.04015*, 2018.

[15] F. Marra, C. Saltori, G. Boato, and L. Verdoliva, "Incremental learning for the detection and classification of gan-generated images," in *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, IEEE, 2019, pp. 1–6.

[16] J. Jiang, E. Cetin, and O. Celiktutan, "Ib-drr-incremental learning with information-back discrete representation replay," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3533–3542.

[17] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," *arXiv preprint arXiv:1708.01547*, 2017.

[18] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

[19] H. Xiao, K. Rasul, and R. Vollgraf. "Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms." arXiv: cs.LG/1708.07747 [cs.LG]. (Aug. 28, 2017).

[20] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "Emnist: Extending mnist to handwritten letters," in *2017 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2017, pp. 2921–2926.

[21] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[22] P. Welinder, S. Branson, T. Mita, *et al.*, "Caltech-UCSD Birds 200," California Institute of Technology, Tech. Rep. CNS-TR-2010-001, 2010.

[23] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3d object representations for fine-grained categorization," in *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[24] R. Kemker, M. McClure, A. Abitino, T. L. Hayes, and C. Kanan, "Measuring catastrophic forgetting in neural networks," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[25] Z. Yang, Y. Wu, J. Cheng, S. Peng, L. Wang, and D. Tao, "Incremental xqda metric learning for person reidentification," in *2018 IEEE International Conference on Information and Automation (ICIA)*, IEEE, 2018, pp. 433–438.

[26] R. Coop, A. Mishtal, and I. Arel, "Ensemble learning in fixed expansion layer networks for mitigating catastrophic forgetting," *IEEE transactions on neural networks and learning systems*, vol. 24, no. 10, pp. 1623–1634, 2013.

[27] A. Douillard, M. Cord, C. Ollion, T. Robert, and E. Valle, "Podnet: Pooled outputs distillation for small-tasks incremental learning," in *European Conference on Computer Vision*, Springer, 2020, pp. 86–102.

[28] A. Gepperth and C. Karaoguz, "A bio-inspired incremental learning architecture for applied perceptual problems," *Cognitive Computation*, vol. 8, no. 5, pp. 924–934, 2016.

[29] A. Seff, A. Beatson, D. Suo, and H. Liu, "Continual learning in generative adversarial nets," *arXiv preprint arXiv:1705.08395*, 2017.

[30] E. Choi, K. Lee, and K. Choi, "Autoencoder-based incremental class learning without retraining on old data," *arXiv preprint arXiv:1907.07872*, 2019.

[31] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka, "Distance-based image classification: Generalizing to new classes at near-zero cost," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2624–2637, 2013.

[32] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," *Proceedings of machine learning research*, vol. 70, p. 3987, 2017.

[33] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 139–154.

[34] R. Pascanu and Y. Bengio, "Revisiting natural gradient for deep networks," *arXiv preprint arXiv:1301.3584*, 2013.

[35] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[36] M. Riemer, I. Cases, R. Ajemian, *et al.*, "Learning to learn without forgetting by maximizing transfer and minimizing interference," in *International Conference on Learning Representations*, 2018.

[37] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[38] J. Wang, Y. Song, T. Leung, *et al.*, "Learning fine-grained image similarity with deep ranking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1386–1393.

[39] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, IEEE, vol. 2, 2006, pp. 1735–1742.

[40] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[41] J. Huo and T. L. van Zyl, "Unique faces recognition in videos," in *2020 23rd International Conference on Information Fusion (FUSION 2020)*, 2020, ISBN: 978-0-964527-6-2.

[42] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin, "Deep metric learning with angular loss," in *2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE Computer Society, 2017, pp. 2612–2620.

[43] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *European conference on computer vision*, Springer, 2016, pp. 499–515.

[44] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[45] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[46] W. Park, D. Kim, Y. Lu, and M. Cho, "Relational knowledge distillation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3967–3976.

[47] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[48] K. Musgrave, S. Belongie, and S.-N. Lim, *Pytorch metric learning*, 2020. arXiv: 2008.09164 [cs.CV].

[49] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[50] J. He, J.-N. Chen, S. Liu, *et al.*, "Transfg: A transformer architecture for fine-grained recognition," *arXiv preprint arXiv:2103.07976*, 2021.

[51] K. Musgrave, S. Belongie, and S.-N. Lim, "A metric learning reality check," in *European Conference on Computer Vision*, Springer, 2020, pp. 681–699.

[52] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035.

[53] P. Ghosh and L. S. Davis, "Understanding center loss based network for image retrieval with few training data," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 0–0.

[54] B. Yu, T. Liu, M. Gong, C. Ding, and D. Tao, "Correcting the triplet selection bias for triplet loss," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 71–87.

[55] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.