

Incremental Learning of Object Detectors using Limited Training Data



By
Muhammad Abdullah Hafeez
00000119675

Supervisor
Dr. Faisal Shafait
Department of Computing

A thesis submitted in partial fulfillment of the requirements for the degree
of Master of Science in Computer Science (MS CS)

In
School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST),
Islamabad, Pakistan.

(August 2020)

Approval

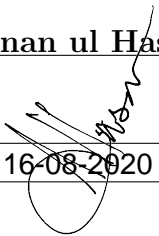
It is certified that the contents and form of the thesis entitled “**Incremental Learning of Object Detectors using Limited Training Data** ” submitted by **Muhammad Abdullah Hafeez** have been found satisfactory for the requirement of the degree.

Advisor: **Dr. Faisal Shafait**

Signature: 


Date: 14-08-2020

Committee Member 1: **Dr. Adnan ul Hasan**

Signature: 

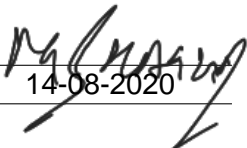
Date: 16-08-2020

Committee Member 2: **Dr. Muhammad Imran Malik**

Signature: 

Date: 17-08-2020

Committee Member 3: **Dr Muhammad Shahzad**

Signature: 

Date: 14-08-2020

Thesis Acceptance Certificate

Certified that final copy of MS/MPhil thesis entitled “**Incremental Learning of Object Detectors using Limited Training Data** ” written by **Muhammad Abdullah Hafeez**, (Registration No 00000119675), of SEECS has been vetted by the undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/MPhil degree. It is further certified that necessary amendments as pointed out by GEC members of scholar have also been incorporated in the said thesis.

Signature: 

Name of Advisor: Dr. Faisal Shafait

Date: 14-08-2020

Signature (HOD): _____

Date: _____

Signature (Dean/Principal): _____

Date: _____

Dedication

I dedicate this thesis to my Parents, my *Nano*, my brother and sisters, who supported me to their fullest, as it was their constant prayers, utmost affection, moral and logistic support due to which I have been able to accomplish this achievement.

Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: Muhammad Abdullah Hafeez



Signature: _____

Acknowledgment

First and foremost, I pay my gratitude to ALLAH ALMIGHTY for His countless blessings upon me and without His blessings, I could not accomplish this goal. I am thankful to my Advisor Dr. Faisal Shafait and committee members Dr. Adnan ul Hasan, Dr. M Imran Malik and Dr. Muhammad Shahzad for their guidance and support which helped me to conclude my thesis successfully.

I am also grateful to the National University of Sciences and Technology (NUST) for providing me a great learning atmosphere, to the faculty of the School of Electrical Engineering and Computer Science (SEECS), family of TUKL-NUST Lab, course-mates, my younger sister Hamna Hafeez and brother Hamza Hafeez who helped and encouraged me during the ups and downs of the research phase of my thesis.

Last but not least, I would like to thank my Parents, my *Nano*, my brother and sisters as this research thesis stands as a testament to their prayers, unconditional love, constant support and encouragement.

Muhammad Abdullah Hafeez

Table of Contents

1	Introduction	1
1.1	Incremental Learning Overview	2
1.2	Background	3
1.2.1	Dynamic nature of the world	4
1.2.2	Limitations of the DL models	4
1.3	Requirement of Incremental Learning	5
1.4	Challenges	6
1.4.1	Catastrophic Forgetting	6
1.4.2	Data Availability	6
1.4.3	Scalability	7
1.5	Applications	7
1.5.1	Autonomous systems	8
1.5.2	Computer Vision	8
1.5.3	Continuous stream of data	8
1.6	Motivation	9
1.7	Problem statement	9
1.8	Scope	10
1.9	Thesis Contribution	10
1.10	Thesis Structure	11
2	Literature review	12
2.1	Incremental Learning Overview	12
2.1.1	Working mechanism of Deep Learning Models	12
2.1.2	Catastrophic Forgetting	14
2.2	Typical model-design philosophies	15
2.2.1	Retraining Phases	15
2.2.2	Network Extension	16
2.2.3	Distillation Loss	16
2.2.4	Nearest Class Mean Classification	17
2.2.5	Network in Network	18
2.3	Summary	19

3	Proposed Model	20
3.1	Coming up with the architecture design	20
3.2	Overview of proposed model	21
3.3	Detailed architecture of proposed model	23
3.3.1	Training pipeline	27
3.3.2	Testing pipeline	27
3.4	Search for optimal classifier	27
3.4.1	Nearest task mean classifier	28
3.4.2	Nearest class mean classifier	28
3.4.3	Mean mode classifier	29
3.4.4	Confidence estimation based classifier	29
3.4.5	Maxout of stacked logits	30
3.5	Implementation Details	31
3.5.1	Hardware and Software	32
3.5.2	Data set and pre-processing	32
3.5.3	Training setup	32
3.6	Summary	33
4	Performance Evaluation	34
4.1	Accuracy Comparisons	34
4.1.1	05 increments	35
4.1.2	10 increments	35
4.1.3	20 increments	35
4.2	Confusion matrices	36
4.2.1	Observations	36
4.2.2	Analysis	37
4.3	Accuracy of separate tasks - with separate test data set	38
4.3.1	Observations	40
4.3.2	Analysis	40
4.4	Individual accuracies of all tasks - on combined test dataset	40
4.4.1	Observations	41
4.4.2	Analysis	41
4.5	Summary	42
5	Conclusions and Future Prospects	45
5.1	Future research prospects	45
5.1.1	Architectural Based Designs	46
5.1.2	Rehearsal Based Designs	47

List of Figures

1.1	Basic structure of Artificial Neural Networks. Input layer has 3 blue, Hidden layer has 4 orange and Output layer has 2 green neurons. Arrows show the learnable parameters called weights.	2
2.1	Basic working principle of single neuron in the Artificial Neural Network [43]	13
2.2	Continual learning based on Generative strategy [36]	16
2.3	Continual learning based on Lwf [22]	17
2.4	PackNet pruning and training strategy on 5x5 filter [24]	18
3.1	Residual Learning: a building block, Concept which revolutionized the deep learning models to go very deep without the vanishing/exploding gradient issue [12]	22
3.2	Architecture of Resnet model in different number of layers configurations [12]	23
3.3	Architecture of Proposed model: Shared and fixed layers of the model are taken from the pre-trained Resnet34. While Network Extension part of the model has one dedicated Convolutional and Fully connected layer, which are added for each increment, during training phase of incremental setting.	24
3.4	Training pipeline of model: Two increments are shown i.e. Base increment, first increment. Legend is in Fig 3.5	25
3.5	Testing pipeline of model: Combined testing for 'n' increments. Data passed is combined testing data set from all 'n' increments	26
3.6	Confidence density comparison: It is clearly shown in the graph, that Gaussian, and Uniform noise being Out-of-Distribution data set has high density in very low confidence range, while In-Distribution data set has very high density in high confidence range.	30

4.1	Incremental accuracy comparison over 05 increments. Brackets indicate the pre-trained model used in part of proposed model	35
4.2	Incremental accuracy comparison over 10 increments. Brackets indicate the pre-trained model used in part of proposed model	36
4.3	Incremental accuracy comparison over 20 increments. Brackets indicate the pre-trained model used in part of proposed model	37
4.4	Accuracy of each task separately, with only task's test data, at all succeeding increments. Starting point of the graph lines from the left shows the task number	39

List of Tables

4.1	Individual Accuracies of each task over 5 increments	42
4.2	Individual Accuracies of each task over 10 increments	42
4.3	(a) Individual Accuracies of each task over 20 increments	44
4.4	(b) Individual Accuracies of each task over 20 increments	44

Abstract

State of the art Deep learning models, despite being at par to human level in some of the challenging tasks, still suffer badly when they are put in the condition where they have to learn with the passage of time. This open challenge problem of making deep learning model learn with the passage of time is often called with synonymous names like Lifelong Learning, Incremental Learning or Continual Learning etc. In each increment new classes / tasks are introduced to the existing model and trained on them while maintaining the accuracy on the previously learnt classes / tasks. But accuracy of the deep learning model on the previously learnt classes / tasks decreases with each increment. Main reason behind this accuracy drop is catastrophic forgetting, an inherent flaw in the deep learning models, where weights learnt during the past increments, get disturbed while learning the new classes / tasks from new increment. Approaches have been proposed to mitigate or avoid this catastrophic forgetting, such as use of knowledge distillation, rehearsal over previous classes, or dedicated paths for different increments etc.

Here in my work I proposed a novel approach based on transfer learning methodology, which uses a combination of pre-trained shared and fixed network as backbone, along with a dedicated network extension in incremental setting for the learning of new tasks incrementally. Results have shown that proposed architecture successfully bypasses the catastrophic forgetting issue and completely eradicate the need of saved exemplars or retraining phases which are required by the current state of the art model to maintain performance, and still have performance comparable to the state of the art incremental learning model.

Chapter 1

Introduction

Machine learning (ML) is one of the most prominent and active research branch of Artificial intelligence (AI). A Machine learning algorithm, enables the computation system to be able to learn and improve from the training data (which is sometimes referred as *experience* E in the literature), without being specifically programmed. Field of ML is mainly focused on finding useful patterns automatically from the data given to its model, without any involvement or assistance from human, and then use those patterns for predictive purpose.

Machine learning approaches are often broadly categorized into three major sub-fields based on the nature of the “*signal*” or “*feedback*”, which is available to the learning system [4].

- Supervised learning: In this learning scenario, each input training example has its corresponding desired output, usually known as label. Therefore model will learn the generalized mapping rule from the input and output pairs.
- Unsupervised learning: In this setting no pairs of inputs and their corresponding outputs are given to the model. Instead, inputs are given to the model without labels and model has to extract the useful patterns and structures from the given inputs on its own.
- Reinforcement learning: Model being as an agent, interacts with the given dynamic environment, upon which model receives feedback (referred as reward) from its environment. Goal of the learning in this setting is to maximize its reward received from the environment.

There are various families of ML algorithms: like Naïve Bayes, Random forests, Support Vector Machines (SVMs) and Artificial Neural Networks

(ANNs) to name a few, which are commonly used in vast variety of scenarios. All ML models have their own merits and demerits.

Among these models, Artificial Neural Networks are loosely designed on the structure of neurons and synapses of the brain. As show in the Figure 1.1. Initially, design was intended to work out problems like a human brain, however, with the passage of time, ANNs was moulded to achieve specific task with distinctive performance. Since then ANNs have been successfully applied to vast variety of domains like vision, speech, translation and games. ANNs have outperformed other ML models multiple times in different scenarios. In 2012, success of AlexNet [21] in the ILSVRC2012 [34] competition gave breakthrough to the Deep Learning (DL) and as of 2020, Deep Learning is the most prominent approach to work on in the field of ML [2].

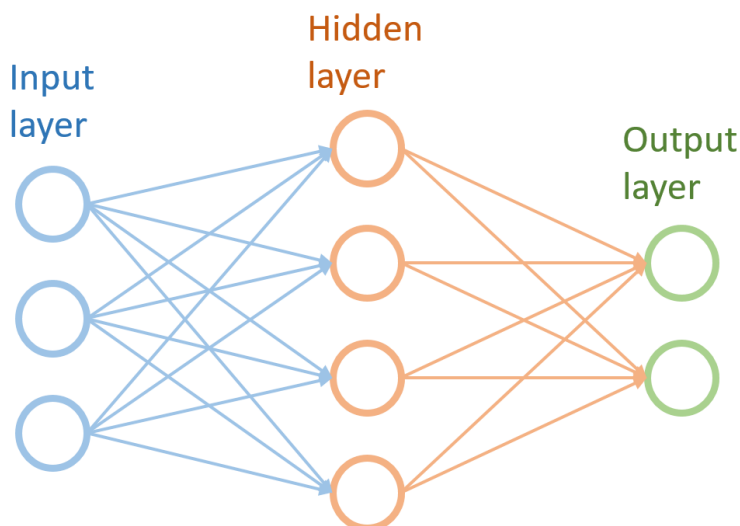


Figure 1.1: Basic structure of Artificial Neural Networks. Input layer has 3 blue, Hidden layer has 4 orange and Output layer has 2 green neurons. Arrows show the learnable parameters called weights.

As the research done under this thesis is mainly focused on the domain of Deep Learning (DL) so we will discuss the Neural Networks (NNs) and closely related models in incremental setting in detail.

1.1 Incremental Learning Overview

Humans and other higher mammals have distinctive ability to learn and acquire knowledge in incremental fashion, throughout their life span [6]. Their brain have a good balance of synaptic stability-plasticity [1]. Therefore, they

can incorporate new skills and information without disturbing the old one, which they have acquired in the past.

In Machine Learning, method which makes an ML model capable of extending its knowledge by learning new tasks with time, when and as new tasks become available, without degrading the performance of model on the tasks learned in past is called *Incremental Learning*. In the real world scenario, information is being generated continuously with the passage of time. Therefore, Incremental Learning capability for an ML model is an essential feature for a true artificially-intelligent computational system. Despite being a very crucial feature, design of a true incremental learning model is still a long standing challenge in the field of machine learning [11].

Although, it has been proved multiple times that, in the field of machine learning, Deep learning models are the best performing models in vast variety of domains – due to the reason, that Deep learning models have excellent ability to learn the specific task in a single go. Still DL model suffer badly, when model is put in a setting where it has to learn tasks incrementally with the passage of time. Main reason for DL model to suffer badly in incremental setting is because all the training data is not available at the same time so they suffer from phenomena called *catastrophic forgetting* or *catastrophic inference* [[25], [7]].

1.2 Background

Since the mid of 20th century, computers alone have revolutionize the world more than any other technology else. It has impacted almost all industries and all sectors of the world. It will be safe to say that computers have impacted every person's life directly or indirectly. Since 1970s, computation power of computers kept increasing, with doubling every two years as depicted by the Moore's Law in 1965. With the help of computers we were able to do enormous amount of computations which could never be thought before, would be possible. This trend is still followed today and will keep continuing, given the Moore's law is followed.

At very early stage, people in the computer industry realized that, computers despite being extremely powerful in computations, are simply dumb machines. Computers require to be programmed in very fine and clear details and that they cannot do a simplest task or take a simplest decision on their own. Therefore, in mid 1950s, people in the computing field started to work on the design of algorithms which could make computers behave intelligently and thus the field of Artificial Intelligence research was established.

Field of Artificial intelligence itself has a very large landscape. Which

involves, logic building, reasoning, planning and decision making, search and optimization and learning. Learning part of AI developed in the field prominently known as Machine Learning (ML). Machine learning plays a very important role in the making a better artificial intelligent (AI) system, because ML makes an agent well informed and educated about its environment, and better the agent is informed about its surroundings better will be the response it can generate.

Learning of an intelligent agent is ideally a continuous process. Whereas, ML state of the art models lack this ability in true sense. To further discuss it in detail we break the discussion in to two parts:

1.2.1 Dynamic nature of the world

World is fundamentally dynamic in nature. Change is simply happening all the time. Among those changes is the generation of data, and with data comes information. Data is being produced by different sources and in different forms. With the digitalization of technology, rate of data production has increased by many folds. Global level interconnectivity via World Wide Web, smart phone technology and Internet of Things (IoT) has increased this data generation rate exponentially. The rate itself is growing with time. Therefore enormous amount of new information is taking form every day. So there is always new stuff to learn.

New concepts are emerging and new products are being introduced. Even concept for same thing changes with time. For example with the perspective of classification: specific car name can refer to all of its generations yet each generation is visually quite different from each other. Moreover, concept changes can be gradual as well as abrupt.

Distribution of the data can change with time as well, called as *concept drift*. Concept drift can itself be of different types: if change is only to the distribution, it is called *virtual concept drift*. If change is in the functionality as well then it is called *real concept drift*. Then concept drift could be local if changes are in the specific range of data space [39].

Conclusively, world is dynamic in nature and Incremental learning capability is an essential feature for a true Artificial Intelligent (AI) system.

1.2.2 Limitations of the DL models

Currently deep learning models have outperformed other ML model families in various domains. Deep learning models have even surpassed humans in performance in specific settings. There are multiple reason behind their success. Some of them are as follows

- ANNs by design, have the capacity to learn very good approximation of even very complex functions.
- They can almost always make good use of data available. Which means that more data is always helpful in the performance improvement of ANNs. Now a days, data is already available in abundance. Thus ANNs can get better and better with the availability of data, where other models usually stop learning any further.
- Their design is dynamic and can be extended to virtually any number of layers, which usually helps in increasing performance. With the introduction of state of the art designs in the field of Deep learning, research have shown that networks can be extended extended from 100s of layers to even well over 1000 layers.
- Deep learning models can be computation hungry. But with the advancement and availability of powerful hardware, we can easily parallelize and deploy big architectures.

Despite being outperforming and powerful models their design has some limitations. Most prominent is that, data used for training a DL model must be available right at the time of training. But in real world scenario world being dynamic in nature as discussed in the section 1.2.1, this cannot always be possible to have all the training data available during training phase. This raises the requirement of DL models to must have feature of Incremental learning capability.

1.3 Requirement of Incremental Learning

Keeping in view of Dynamic nature of the real world scenarios. We now know that we cannot always have all the data available at the time of training phase or in some cases data distribution can change overtime or itself the concept for a single thing can change. These all mentioned scenarios lead us to fill the gap of learning within the dynamic environment of the world

Whereas DL models are not yet capable of learning with time, as and when data become available. Therefore, a strong need arise for the requirement of model which can learn within the dynamic environment of the world by incrementally learning about new tasks / classes when and as their training data become available, without forgetting about the tasks / classes learnt it past also and without degrading the performance.

1.4 Challenges

Challenges involved in designing DL models capable of incrementally learning with time, are being studied since early 90s. Although DL models are very flexible in design from one perspective but still there are some inherent issues with their design which makes them difficult to work around and make them fully incremental learning model. Some of main issues are below:

1.4.1 Catastrophic Forgetting

One of the main issue which is considered the core issue for making DL models to have true incremental learning capability, is the catastrophic forgetting. By design DL models have learn-able parameters called weights. Basically these weights are learned (adjusted) iteratively, during the training phase. These learned weights collectively depict the approximation of function which the model has learnt. Therefore in the incremental setting when the new data becomes available. Then during the training phase when model tries to learn on this newly available data, weights of the models adjust themselves according to this new data and hence the adjustment of weights for the previously learnt data gets disturbed. As the result model performance on the previously learnt tasks / classes starts to degrade while learning the new ones. This issue is referred as *catastrophic forgetting*. Whereas in DL models, this balance between plasticity (ability to learn new tasks / classes) and stability (ability to retain the performance on the old tasks /classes) is called *stability-plasticity dilemma*.

1.4.2 Data Availability

In the mitigation of catastrophic forgetting issue, some proposed incremental designs have used the technique of rehearsals or retraining of old tasks / classes, during the training phase of new tasks / classes. In this method some of the examples from old distributions are passed to the models along with the new training data. Therefore, where model learns about the new task / classes in the current increment, it also retains the accuracy to some extent on the tasks/ classes learnt in the past, by the rehearsal on their own data distribution examples.

- There are scenarios where data is coming in streams. Therefore data cannot always be available for the retraining phase. Especially in online learning strategies this situation arises. So retraining strategy could become ineffective.

- As in the retraining phase, old training data examples are required for the rehearsal purposes. Therefore old training data examples are saved on the disk for later used in the rehearsal purpose. But in some cases data is sensitive and could possibly arise the privacy issue, due to which data cannot be saved on the disk anymore. Therefore it will be not possible to rehearse the old training. Thus it will hinder in making the incremental training for the model effective. Another reason which could become issue in saving data will be the data rights. Which could arise the issue where we cannot save the data on the disk. Thus rehearsal will be not possible or may become ineffective resulting in degrading the true incremental ability of the DL model

1.4.3 Scalability

Scalability is another one of the most important and difficult challenge in design of incremental learning model. Where an incremental learning algorithm can keep learning without becoming an issue in the space and time complexity. Which should be linear or even less in its growth rate at and best if it is not increasing at all. Scalability is further discussed in the following points:

- As the true incremental ability require the model to learn for very long periods of time which could be analogous to lifelong learning of humans. But models having incremental ability based on rehearsal strategy requires data to be stored on the disk. But for the very long term incremental learning scenarios very large number of increments should incorporated. By design, for each increment some data examples should be stored on the disk. But for the very large number of increments scenario, this disk requirement will become humongous.
- By design, DL model have to incorporate some new connections, every time an increment is added. Hence model size increases by some factor on each increment. Therefore for the long term incremental scenario, size of the model kept increasing ultimately becoming model of enormous size.

1.5 Applications

A true incremental learning model will be beneficial in multiple fields. Artificial intelligence is already being applied in numerous sectors and fields of the world. Healthcare, automotive industry, Finance and economics, security

and cybersecurity, Gaming industry, government, military, media and art industry are one of the few fields benefiting from the capabilities of AI and ML. If we look closely all these fields are dynamic in nature. Therefore AI and ML models featuring incremental learning capability will be extremely beneficial.

1.5.1 Autonomous systems

Currently world is shifting from automatic to autonomous machines. Where less supervision will be necessary to be given to machines to complete the desired tasks. For the robots working in the industry will have more generalized knowledge base and their knowledge can be extended according to the specific needs of the domain they will be applied on, rather every time training them from scratch for any new task they are shifted on. Medical robots' training can be extended further with time without any overhead of complete retraining. Autonomous vehicles skill and knowledge base can be extended with the new data and environments easily on the go.

1.5.2 Computer Vision

In the field of image classification it can be beneficial in many ways, some of them are as follows: Image classifiers working for search engines can expand their knowledge base as and when new classes and examples become available. Auto tagging tools can learn new tags with time without requiring the overall training again and again. Face detection systems working can learn about new people from time to time, without any overhead of complete training data. Object detectors working in e-commerce domain can be trained on the new objects as and when required, without training again and again on the data.

1.5.3 Continuous stream of data

In the era of big data, data is being generated at a fast pace and pattern analysis of the data distributions where concept drift is already in place. Incremental learning models can work way better than simple static ML / DL models. In such online based scenarios data cannot even be saved, just in case we train the complete model again and again, where distribution or concepts can even change with time. Thus, incremental learning models can easily accommodate these changes in distributions and change in concepts along with the completely novel concepts coming at a fast pace in with the time.

1.6 Motivation

AI is getting better and better with each passing year. This improvement rate in the performance has even increased by many folds especially with the advancement in the field of Deep Learning and by the advancement and availability of the parallelizable hardware usually know as Graphic processing units (GPUs). Therefore with AI models' improvement, these models are now widely accepted in multiple fields all around the world and has been successfully applied in them. AI is now directing the whole industry of the world towards new direction. But still there are huge gaps in the filed yet which need to be addressed to make AI systems fully autonomous and dependable. In the current age, we have lots and lots of data available. Which can be used to improve the capability of AI models.

Keeping in the view of gaps available in the field and availability of data we can use make more robust and intelligent models. One feature among those fields is Incremental learning, which will have a very huge impact in the filed of ML in specific and on AI in general. Therefore, with growing and ever changing data, introduction of new knowledge by every passing day and need to have model for the sake of Artificial general intelligence (AGI), capable of learning with time just we humans do, it has become exigent necessity to have model with Incremental learning capability.

1.7 Problem statement

Although problem with the Deep learning models regarding incremental learning capability has been identified in early 90s as *catastrophic forgetting* but still this feature is an open research problem. Despite being numerous efforts by the research community to design the model with good balance between plasticity and stability, still deep learning models' performance degrades with every increment during the learning process. Current Incremental learning models are still far from being application worthy to real world problems. Most of those models even rely on some level of retraining and disk requirement for saving data for old classes' representation as well. Research efforts are being made by the research community to improve the Incremental learning performance, while keeping the memory and disk requirement as less as possible

1.8 Scope

From the perspective of architecture, research scope of the work done under this thesis is designing of Deep Learning (DL) architecture based model, capable of Incremental Learning. Proposed model is based on the Convolutional Neural Networks (CNNs). Therefore it is targeting the domain of Computer Vision and is trained and tested on the classification problem of image based datasets.

Regarding Incremental scenarios, we have worked on task based incremental learning setup. Hence, complete dataset is divided equally as per total number of classes. Each sub-dataset of classes represent a unique task. Therefore during each increment only one task is added to the model. During training phase, only training data from current task is passed to the model. But during testing, combined test data from all the learned tasks is passed to the model.

1.9 Thesis Contribution

In this thesis work is done on designing the Incremental learning model based on the architectural strategy rather rehearsal strategy in order to avoid catastrophic forgetting. Results have shown that proposed model have almost same performance results to that of state of the art incremental model, along with the following plus points:

- **By passing the effect of catastrophic forgetting:** In proposed model, I have designed the architecture mainly in two major parts - first, combination of fixed shared parameters and second, single-layer dedicated network extension. Thus the designing the model in the said way has eliminated the effects of catastrophic forgetting.
- **Eliminating the Retraining Requirement:** Model is designed in such a way that it eliminates the requirement of retraining over the past classes at any stage.
- **No disk space required:** As the design of the model does not require retraining. Therefore no data or distribution samples are needed to be saved on the disk. Thus, no disk space is required.
- **No data rights / privacy issue:** There could be the case when the training data is protected with rights or there could be privacy concerns. Proposed model being independent of rehearsal phase, does

not require to save any data for the rehearsal. Which automatically eliminates the issue of data privacy or data rights related issues.

- **Separate task performance preserved:** As model desing has two parts one as shared and fixed used by all increments. Second dedicated and learnable. Therefore, due to this scheme of layer design, model accuracy over the separate tasks is fully preserved.

1.10 Thesis Structure

This thesis has 5 chapters. Chapter-1 has through introduction of this thesis. Chapter-2 discusses related work done by the research community in the related field. In Chapter-3, proposed work is discussed in detail along with implementation details of the proposed model. While all the experiments and comparisons are in Chapter-4. Chapter-5 contains conclusions on the research done in this thesis and future prospects.

Chapter 2

Literature review

In this chapter, related work done in the field of incremental learning is reviewed on the landscape of deep learning. Different architectures towards modeling incremental learning algorithms are discussed. Merits and demerits of each algorithm design is discussed in detail.

2.1 Incremental Learning Overview

Based on Machine learning models belonging to similar family, Machine learning has multiple sub fields. Here we will discuss Incremental learning, a sub filed of Machine learning, in prospect of Deep learning.

2.1.1 Working mechanism of Deep Learning Models

Deep Learning models has advanced in the current decade than any other family of machine learning models after the AlexNet [21] success on dataset from ILSVRC-2010, and iLSVRC-2012 [34]. Since then, many state of the art deep learning architectures like Resnet [13], GoogleNet [37], Inception [18], and Densenet [17], have been proposed. These state of the art architectures has performed extremely well on vast variety of datasets in different domains like Image recognition, Object detection, Object recognition, Natural Language Processing etc. But still there is one big constrain on these deep learning models. That deep learning models cannot learn with time, just like we humans do. They require all the training data during the training phase in order to perform at their best.

This can be explained easily if we look closely on the working principals of the deep learning architecture. Complete structure of the model is based on Neurons and connections which connects the neurons of the model. These

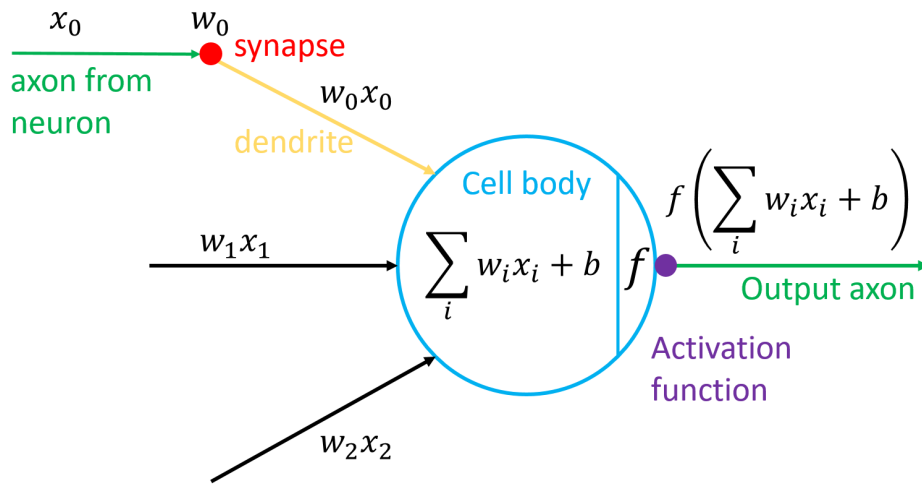


Figure 2.1: Basic working principle of single neuron in the Artificial Neural Network [43]

connections among the neurons are usually called weights, which are adjusted (referred as learning) during the training phase in order to approximate the function on which model is set to learn. The learning mechanism is based on the backpropagation algorithm [33], in which relative loss is back propagated. This loss is calculated based on the derivative of loss function with respect to weights.

1. **Feed forward:** During the feed forward pass. Input is passed to the model. Where each input is multiplied with the weights of the neurons and then bias for each neuron is added. This weighted input is then passed to the activation function to get the output from that specific neuron. This process is done for each neuron in the layer and multiple layers can be combined to make the network deeper. All combined activation from all neurons of the last layer is the final output. Simple feed forward of for single input vector is given in the equation below.
2. **Feed backward:** The output from the model is compared with the target (true labels given for training purpose). Thus the total loss is calculated and with the help of derivatives calculated-loss-contribution of each weight is then propagated back to all the weights of the network. Therefore, weights of each layer are then updated by the learning factor during the back-propagation process. This feed forward and backward process is usually iterated thousands of time until global maxima of the loss function is reached.

From this feed forward and backward mechanism we can get the insight, that in DL model weights are getting adjusted at each iteration therefore effect of this process on the incremental learning setting will be discussed in the next section.

$$z_n^l = \sum_{i=1}^m (w_{n,i}^l x_i^{l-1} + b_n^l) \quad (2.1)$$

$$a_n^l = \sigma(z_n^l) \quad (2.2)$$

where: l = number of layer in the network
 n = number of neuron in the specific layer
 m = number of the neurons (activations) in the previous layer
 w = weights of specific neuron in a layer
 b = bias of specific neuron in a layer
 z = weighted input of the specific neuron in a layer
 $\sigma()$ = activation function
 a = activation of specific neuron of a layer

2.1.2 Catastrophic Forgetting

From the working principle of the DL model it is clear that weights of the DL model gets adjusted in every iteration for the learning during the training process. That is why all the training data must be available to the DL model during the training phase, because if part of training data is not passed to the DL model during training phase, it will later disturb the weights learnt previously, when this data will be passed later to the model for learning. Therefore performance of the whole model drops on the data distribution learnt in the older training phase. This issue is referred as catastrophic forgetting / inference [25], [7]. This issue was identified by researchers in the late 80s and in 90s. Since then it is the major long standing issue in DL model architecture for making them a good incremental model.

DL models can easily adopt to the new data distribution and hence considered as very plastic nature. From one perspective it is very good attribute for a model to have incremental capability. But this attribute comes at the price of the stability of the model performance on the previous data distributions. It means that when the new data distribution is introduced to the

model, it learns the new distributions in a way that it starts to forget about the data distributions learnt in the past. So the stability of the model suffers at the cost of plasticity. This dilemma is called plasticity-stability dilemma. This dilemma is tightly related to the catastrophic forgetting.

2.2 Typical model-design philosophies

Catastrophic forgetting being the inherent issue in the deep learning models. To mitigate the issue different design philosophies has been proposed. Some designs have tried with the approach to mitigate the issue of catastrophic forgetting. While other have tried to avoid it at all in first place. Some of them are discussed below.

2.2.1 Retraining Phases

One design philosophy used in incremental learning models is “retraining phase”. Where, during the training phase of new task, some part of previous tasks’ training data is also used with the training data of new task. In this regard some mechanism should be devised to select some part of training data form previously trained classes.

In iCaRL [31], they used herding technique from [41], to get the best exemplars for the retraining phase. Herding was used in order to achieve two objectives: first, to make exemplars set to exhibit true class mean approximation at initial stage, second, herding algorithm could remove exemplars when necessary while keeping the approximation as close as possible, thus usage of disk space for saving the exemplars was kept constant.

Another approach for the retraining phase was the use of Generative adversarial networks [9], [30]. In this setting training data of old classes were generated with the use of generator of the generative adversarial network and thus augmented with the training data for the new increment. In [36] they used the combination of generator and solver, making complete model for current state as scholar as shown in Figure 2.2. On new increment, generator and solver of old scholar generated pseudo training samples for old data and their labels respectively. Which were then combined with the data and labels for new classes to train the new scholar. This helped in keeping the accuracies over previously trained classes as high as possible while learning the new classes with time.

Advantage of this generative adversarial network based retraining approach, is retaining the accuracy over the old classes along with learning the

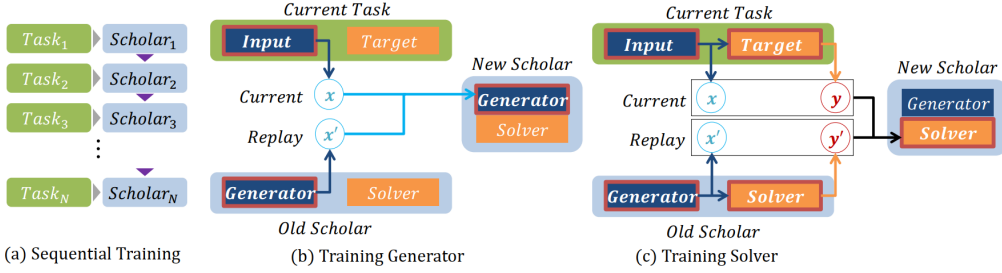


Figure 2.2: Continual learning based on Generative strategy [36]

new classes while eliminating the requirement of saving old classes' data on disk.

Limitation of this method is that generator has difficulty in generating visually complex dataset classes like CIFAR100 [20], CUBs200 [40], ImageNet [34] etc, as mentioned in [19]. Therefore performance drops when generator based architecture are put to test with such datasets. This results in limiting its application over the visually complex datasets.

2.2.2 Network Extension

During the each increment, model is further trained on some specific number of new classes. Which requires the addition of output node in the output layer of the network. When new nodes are added to the network, new set of weights making connection between output and last hidden layer are also added [31].

In Learning without forgetting [22], they used the task incremental strategy instead of class incremental strategy. Where their network was divided into two categories depending on the parameters' access to specific task. One is shared part of network having shared parameters θ_s across all tasks, other part is network extension with parameters θ_n which was added on introduction of new task. While at the same level of θ_n , parameters of network extensions added for old tasks were referred as θ_o , as elaborated in Figure 2.3 part (e).

2.2.3 Distillation Loss

During the training phase of new increment, retraining part is required. Where some mechanism of loss calculation over the previous training dataset becomes necessary, in order to maintain some balance between the training quality of new classes and retaining the accuracy of old classes. This is

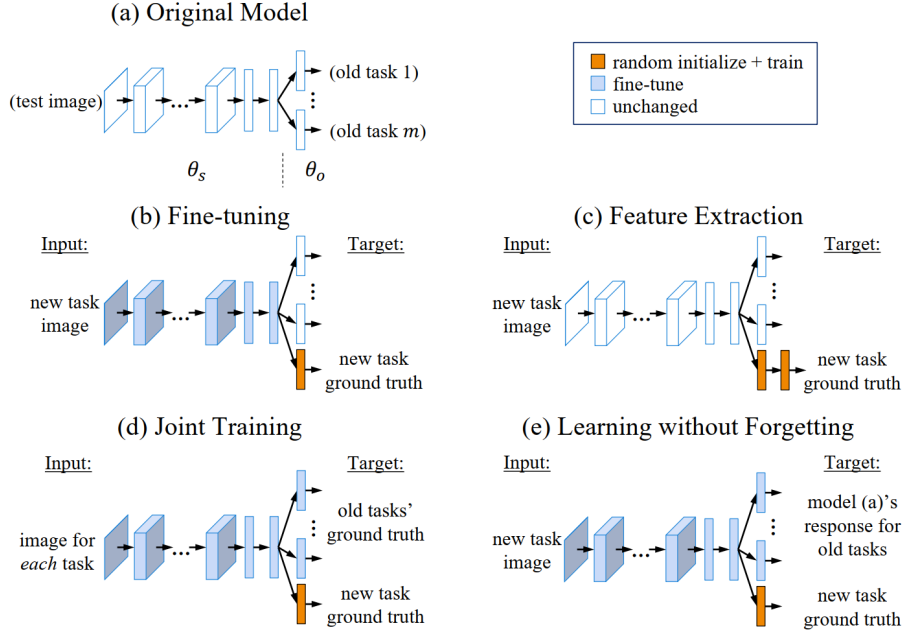


Figure 2.3: Continual learning based on Lwf [22]

usually done using the changes in the in loss calculation method in the loss function for the model. In [31] they used the loss function as the combination of two loss functions: classification loss and distillation loss. The later term was used to prevent the catastrophic forgetting by using the variation of loss function used in [16]. Equation (2.1) was used by [31] for the calculation of loss during training. Which was originally proposed for the information-transfer between neural networks. But in [31] it was used along different time stamps in same neural network.

The same technique of Distillation loss, applied in [31] was also proposed by [22] to keep the accuracy over the old task as high as possible after the addition of new task. But in [22] they used it for task incremental scenario rather class incremental scenario as used in [31].

2.2.4 Nearest Class Mean Classification

Although this thesis is focused on deep learning for incremental learning scenario. In [26] for the first time, it was exhibited that nearest-class-mean classifier can accommodate new classes in incremental manner, by keeping the mean of feature vectors of all examples form all classes. During testing, feature vector of example from test data was classified with the label of

mean-of-feature-vectors, which was most similar on metric to test feature vector. This proposed method exhibited to work well when applied in the incremental learning scenario [26], [27], [32].

In [31] they used different family of algorithm for the classification purpose only, along with the neural network as backbone of the whole algorithm. They used the nearest-mean-of-exemplars as their classification algorithm. In this algorithm they used to assign the label of class to the test image which had closest mean-of-exemplars to the feature vector of that test image.

Main idea of using this algorithm for classification was taken from nearest-class-mean classifier from [26]. The difference between both implementations [31], [26] is the use of only feature representation of exemplars in [31] instead of feature representation of whole data set examples. This is due to the reason that they cannot have all the training data stored for the above mentioned purpose in the incremental setting which will eat up the disk space with time.

Usage of this algorithm in [31] made the classifier robust to changes in feature representation, as class-prototypes could automatically adjust themselves to changes in feature representation.

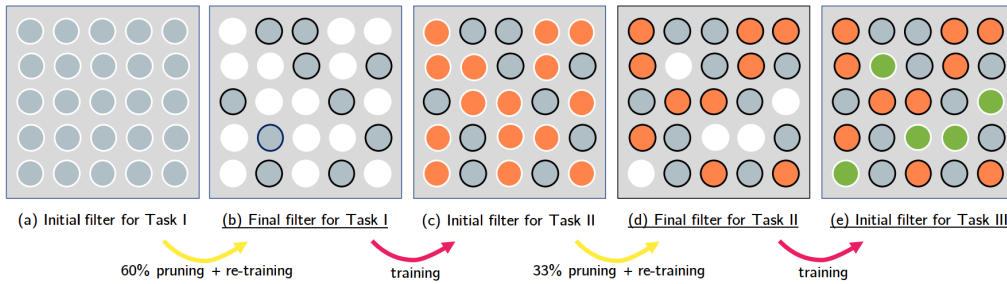


Figure 2.4: PackNet pruning and training strategy on 5x5 filter [24]

2.2.5 Network in Network

Inspired by pruning techniques in neural networks, and redundancies in large deep learning models, PackNet [24] proposed the method for learning of tasks with time in incremental manner, which is relatively different than all the other models discussed here yet. They used the redundancies of the deep learning model to their advantage and freed up the excess parameters for the learning of new tasks without the drop of performance in any task learnt. Using network retraining along with iterative pruning they were able to make the model learn multiple tasks with minimal drop in performance. Another major difference from other tasks was that they always used to optimize for the current (new) task, rather optimizing the balance between the new and

old tasks using some proxy loss functions. Their strategy is illustrated in Figure 2.4

Limitation of PackNet [24] is that it cannot keep on adding new tasks with time, a property, which a true incremental learning algorithm should have. This is due to the reason that after some increments there will be no more room left for pruning in the model thus model will reach its limit and will not be able to learn any further tasks afterwards.

2.3 Summary

Incremental learning, despite being the much needed capability of the machine learning model, is long standing open research problem. Which has not been addressed properly before the recent advancement in the field of deep learning. With the successful application of deep learning in this decade, in vast variety of fields has again given the spark to the need of Incremental learning. In this regard major work done in the domain of deep learning can be categorized as architectural based designs and rehearsal based designs. Both have improved the results slowly but yet the state of the art results are far from being application ready.

Chapter 3

Proposed Model

This chapter discusses the main work this thesis in detail. Basic design philosophies involved in designing the model. High level architecture of the model and training and testing pipeline are described in detail.

3.1 Coming up with the architecture design

Incremental learning, as discussed in Chapter 1, is much required capability of machine learning algorithm. In machine learning algorithms, family of Deep learning algorithms out perform all other families of machine learning algorithms by healthy margin in vast variety of scenarios. So, in this thesis, focus of the work is to design a deep learning based model capable of learning new classes incrementally with time.

Therefore, first it is to point out the issues involved in the design of the model so that it is clear what are the objectives and what are the main issues involved in achieving the required objectives.

Main issues faced by the researchers in designing deep learning models capable of true incremental learning ability are as follows:

- I Catastrophic forgetting: This is the most prominent issue of all in Deep learning models in the domain of incremental learning. Which even raises further issues as well. All of those major issues will be discussed here. Deep learning model must have design which eliminates this issue in first place, or it should at least minimize the effect of catastrophic forgetting, which can be done with rehearsal strategies, which leads to further issues.
- II Constant requirement of old training data: While model design can employ the retraining phases on each increment, but this leads to the issue

of constant requirement of training data of old classes. But training data of old classes might not be available in all cases.

- III Privacy of training data: Another reason is, because retraining phase requires old training data, which must be saved on the disk for rehearsal purpose so limitation could arise when the training data is proprietary or has privacy issues, and cannot be saved on the disk for later use in retraining phase [31].
- IV Increasing data: Requirement of old data should be saved on disk as discussed in point (iii). But in true incremental scenario for every new classes data will be saved on the disk and this saved data for old classes will keep on increasing with every increment.
- V Generative approach: All the three issues (II, III, IV) discussed above can be eliminated, by incorporating the generative approaches [9], [30], [36] for retraining phase, in the model for generating the samples for old training data rather having them saved on the disk. But this approach has its own limitation, because generative models' performance decreases significantly when samples of visually complex datasets are generated for rehearsal [19].
- VI Requirement of Proxy losses: Issues in points (II – IV) require some method, during the retraining phase, to maintain balance between keeping the accuracy over old classes as high as possible and training the new classes as good as possible. This was done with the help of making the loss function with the combination of two losses: Classification loss and Distillation loss [31], [22]. Although this works well but still there is much room for improvement and requires a lot of optimization still to be done.

Therefore, it is necessary to come up with the design which addresses the above discussed issues being faced in the design of Deep learning algorithms in Incremental setting.

3.2 Overview of proposed model

Basic design of the proposed model is based on the design philosophy of avoiding the catastrophic forgetting in first place. In this way, chain of issues involved with “minimizing the effect of catastrophic forgetting” approach can be avoided as well. For example in PackNet [24], they used the same architectural approach where catastrophic forgetting was not an issue. Therefore,

during the increment, they only focused on optimizing for the task in hand, rather to worry about retraining phases and all complexities involved along with that phase as discussed in points II – IV of section 3.1.

Basic architecture backbone, for our model is one of the state of the art deep learning model named ResNet34 [12] with some modifications. Before the introduction of ResNet in 2016, adding layers to deep learning network was not simply beneficial due to vanishing/exploding gradient problem [3], [8]. For the first time ResNet [12] architecture made it possible for deep learning model to have layers up to hundreds or even thousands of layers, while achieving compelling performance [14]. In ResNet [12] this was achieved by architecture design called identity shortcut connection, in which one or more layers were skipped, shown in Figure 3.1.

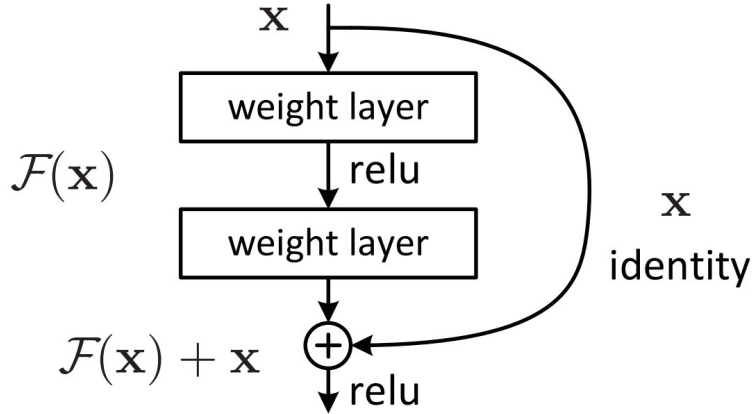


Figure 3.1: Residual Learning: a building block, Concept which revolutionized the deep learning models to go very deep without the vanishing/exploding gradient issue [12]

ResNet34 has, first layer as convolution layer followed by four intermediate layers, while fc layer is at the end of fourth intermediate layer. There are 6, 8, 12 and 6 bottleneck blocks in intermediate layers from. Each bottleneck block has two layers of convolution layer. The detail of layers are shown in Figure 3.2. ResNet34 has 3.6billion FLOPS.

In proposed model here, I have used the ResNet34 network per-trained on the ImageNet [34] 1000-class dataset. In presented model fully connected layer of the ResNet34 is removed and one dedicated convolution layer and fully connected layer is add for each increment.

During training phase the shared part of the model remains fixed as discussed above. However, for each increment right from zero increment

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10 ⁹	3.6×10 ⁹	3.8×10 ⁹	7.6×10 ⁹	11.3×10 ⁹

Figure 3.2: Architecture of Resnet model in different number of layers configurations [12]

(base increment), network extension will be added for the current increment in hand. While network extensions from other previous increments will not be accessible in the current increment. With this setting, the whole model will then be trained on the dataset of the current increment only and optimized for the current training dataset. No retraining or dataset from old increments will be required at all in any future increment. Parameters learnt for old increments will be saved on the disk during each new increment and remain inaccessible to any other training phase except their own training phase.

During testing phase shared part of the model along with network extensions from all the increments will be loaded to the model in *torch.eval()* mode. Combined test data from all increments will be passed at once to model in all the network extensions. Each extension will make prediction for both in-distribution and out-of-distribution examples. Prediction from all network extensions will then be combined in manner specified in the section 3.4 for final prediction.

3.3 Detailed architecture of proposed model

As already discussed in the section 3.2 the proposed model is based on the idea of avoiding the Catastrophic forgetting therefore complete model can be divided in to two main parts according to the sharing of parameters. One part of the model will have parameters shared along all the increments but this part will remain fixed through all increments as to abide by the chosen philosophy, while the other part will have the dedicated parameters.

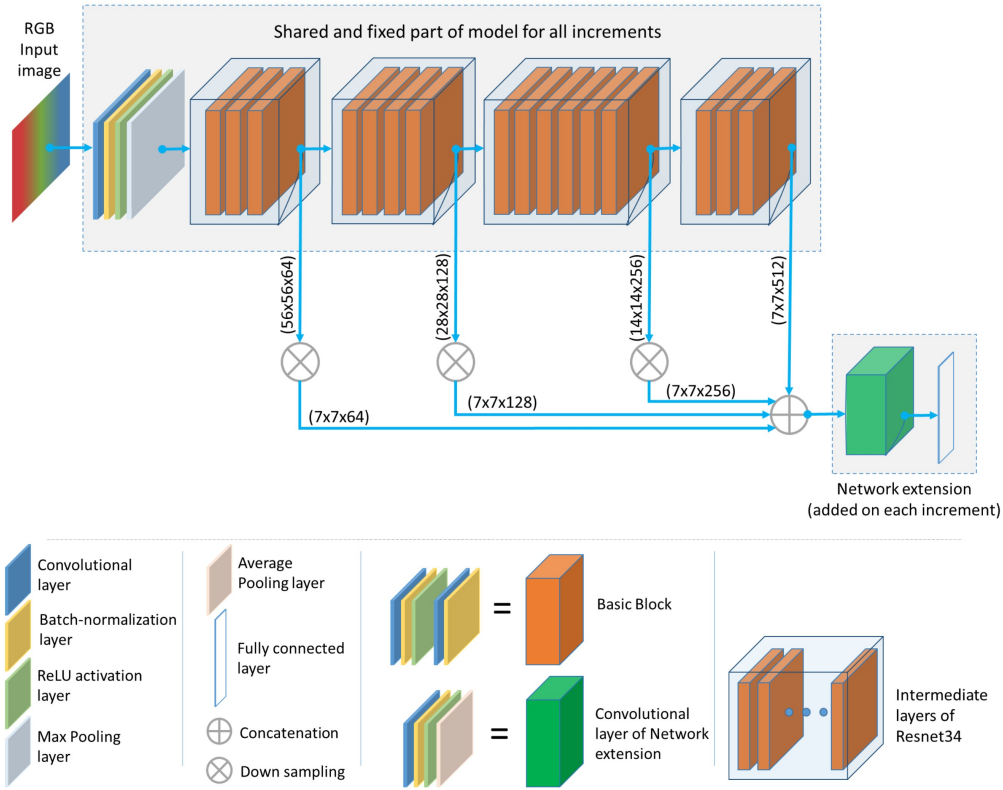


Figure 3.3: Architecture of Proposed model: Shared and fixed layers of the model are taken from the pre-trained Resnet34. While Network Extension part of the model has one dedicated Convolutional and Fully connected layer, which are added for each increment, during training phase of incremental setting.

The shared and fixed part of the model is taken from the ResNet34, pertained on the ImageNet [34] 1000-class dataset. Pre trained is chosen due to the reason that being this part of the model as shared and fixed, it should have, out of the box, rich capability of extracting the features from input.

Output features from all four intermediate layers of ResNet part are taken in order to get all level i.e. from high level to all the way to low level features are gathered. Outputs from first, second and third intermediate layers are of size (height x width) 56x56, 28x28 and 14x14 respectively. Which are then resized to 7x7 and then concatenated with the output of fourth intermediate layer which has already output of size 7x7.

In the ResNet34 model, last layer i.e. fully connected layer is removed and a small dedicated network extension is added for each increment. This network extension has one convolution layer and one fully connected layer.

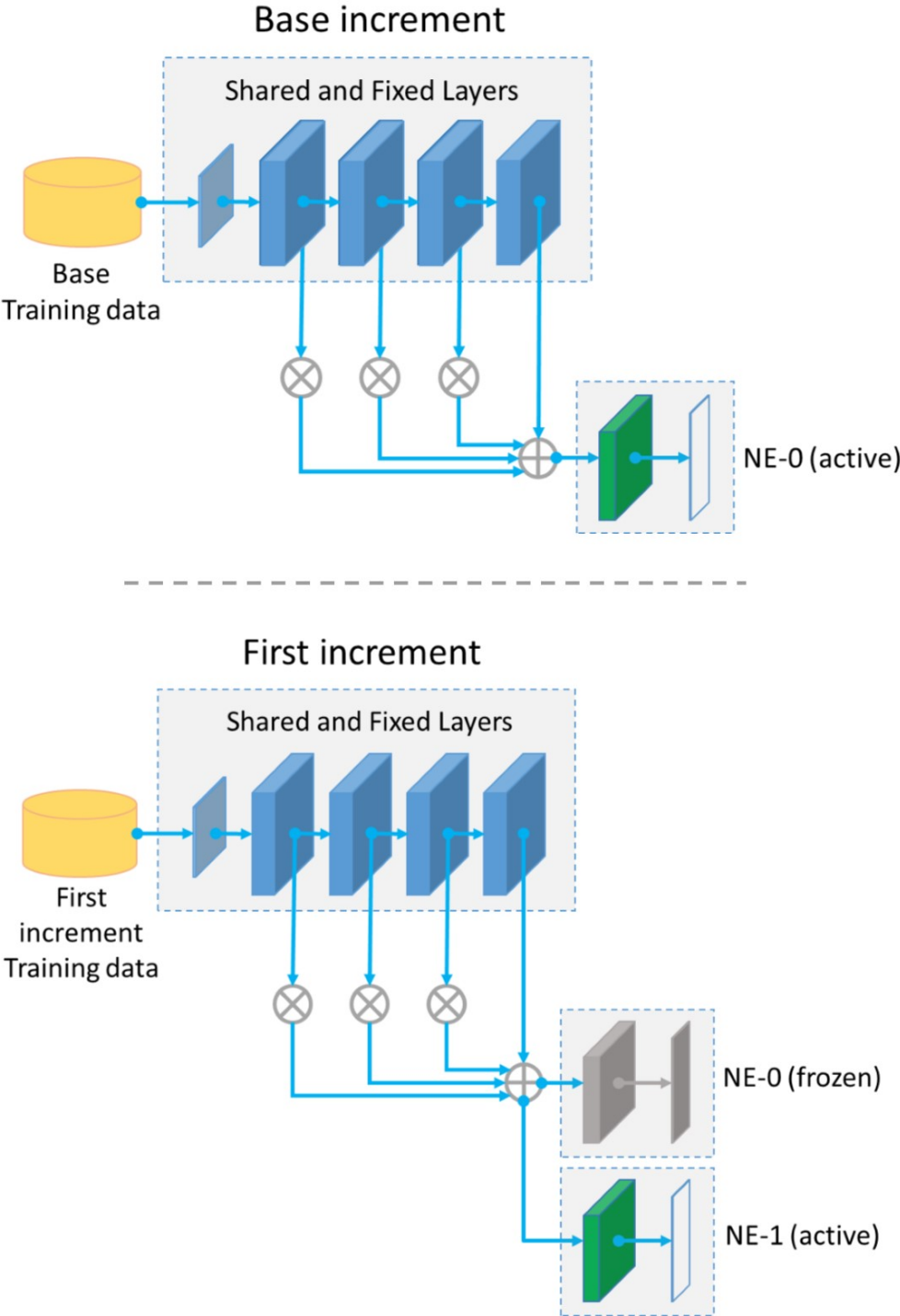


Figure 3.4: Training pipeline of model: Two increments are shown i.e. Base increment, first increment. Legend is in Fig 3.5

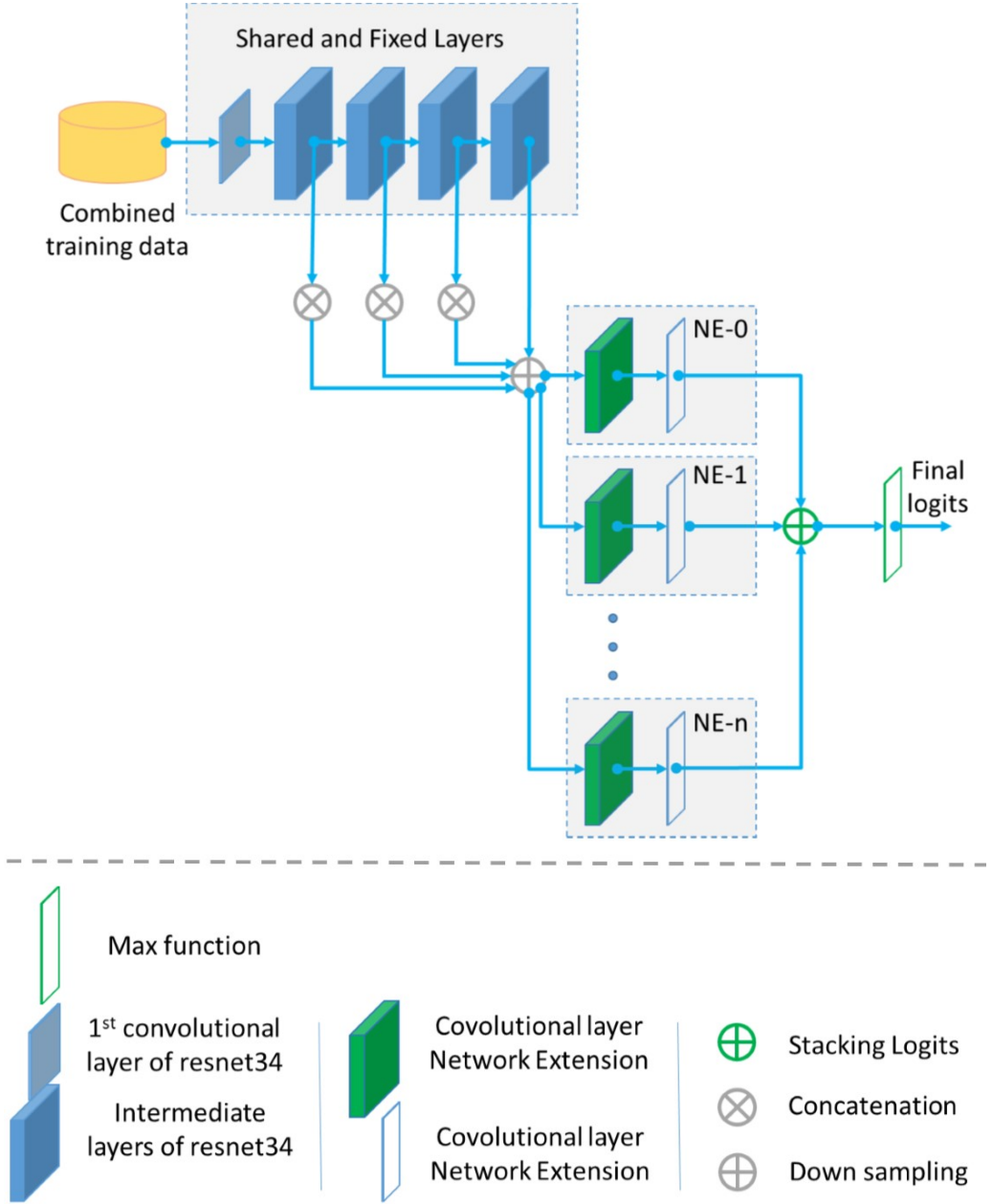


Figure 3.5: Testing pipeline of model: Combined testing for 'n' increments. Data passed is combined testing data set from all 'n' increments

The convolution layer of the network extension has 512 kernels of size 3x3, followed by batch normalization layer and ReLU activation function respectively. Output is of dimension 7x7x512 as height x width x depth (number of feature maps). This output is passed through maxpooling layer before passing to the fully connected layer of the network extension. This fully connected layer has 512 input neurons with dropout probability of 0.5 and output neuron are equal to the number of classes in that specific increment. *Softmax()* activation function is used on the final output. Complete architecture of the proposed model is shown in the Figure 3.3.

3.3.1 Training pipeline

During training phase the shared part of the model remains fixed as discussed above. However, for each increment right from zero increment (base increment), network extension will be added for the current increment in hand. While network extensions from other previous increments will not be accessible in the current increment. With this setting, the whole model will then be trained on the dataset of the current increment only and optimized for the current training dataset. No retraining or dataset from old increments will be required at all in any future increment. Parameters learnt for old increments will be saved on the disk during each new increment and remain inaccessible to any other training phase except their own training phase. Pipeline of training phase is shown in Figure 3.4.

3.3.2 Testing pipeline

During testing phase shared part of the model along with network extensions from all the increments will be loaded to the model in *torch.eval()* mode. Combined test data from all increments will be passed at once to model in all the network extensions. Each extension will make prediction for both in-distribution and out-of-distribution examples. Prediction from all network extensions will then be combined in manner specified in the section 3.4 for final prediction. Pipeline of the testing phase is shown in the Figure 3.5

3.4 Search for optimal classifier

As discussed in the section 3.3, there are different pipelines for testing and training phases. Moreover, model is independent of retraining phase for the rehearsal over old increments' training data. Therefore, distributions of each increment during training phase and during combined and separate testing

phases differ from each other. So model must have a classifier which could work well in all types of settings i.e. incremental training phase as well as in combined and separate testing phases. In this regard, extensive search for the optimal classifier was done. Details are as follows:

3.4.1 Nearest task mean classifier

Nearest mean classifier [26] can classify between classes based on the mean-feature-vector-of-class. . Implementation Idea was taken from iCaRL [31], but here in proposed model it is used for differentiating between the tasks rather than classes. In the proposed model, each increment is regarded as task. Where each task has equal number of classes. Task mean classifier (TCM) will have the updated mean feature vector for each task during the training phase.

During the test phase, the combined distribution of test data from all increments is passed to the model. Feature vectors of all the test data is passed to Task mean classifier before passing it to fully connected layer in each network extension. At this stage, task mean classifier will help in differentiating each passed sample between tasks. So that, samples belonging to the specific task can be passed to only fully connected layer of that specific network extension while samples from other tasks are suppressed.

In theory this design seems to work out well, but when implemented, task mean classifier was barely able to differentiate between the tasks. Reason identified behind this behavior was that, mean of feature vectors of each class in the specific task has its own uniqueness. Whereas, when these feature vectors from all the classes of one task are combined in one mean, the resultant mean loses the uniqueness with the means overlap it should mathematically poses.

3.4.2 Nearest class mean classifier

From section 3.4.1, Task mean classifier being not suitable for the classifier in the given setting. New strategy was devised to use the Class mean classifier. In which, mean of feature vectors of all classes from all tasks are stored with respect to the task it belongs.

Therefore, during the testing phase, instead of differentiating each test sample from other tasks based on the mean of task, each sample is compared with mean of feature vectors of all classes. Based on this comparison, each sample can be identified that in which class and task it belongs. Thus sample belonging to specific task will be passed to its respective Network extension's

fully connected layer, while other samples will be suppressed for that fully connected layer.

Implementation of the idea showed no adequate results. Mean of feature vectors from inter task-classes were not different at the scale where, based on its feature vector, it could be used to identify the task of test sample. Reason, behind this failure was, classes of one task was trained independent of other task classes. Therefore, model did not learned to make adequate differentiation (gap) among all the learned feature vectors of all the classes coming from different tasks. This caused the huge overlap in inter task-classes and resulted in bad classification accuracy.

3.4.3 Mean mode classifier

In proposed model, convolution layer of network extension gives the feature vector of 512 size. Techniques described in section 3.4.1 and 2 were unable to give desired results. Those techniques were using the complete feature vector as single identifier. Which was considered as waste of information. Therefore, new approach was devised. In this approach, idea was to use each feature of the feature vector independently for the comparison with the respective feature of mean of all other classes from all tasks. Then to each feature class label was assigned. In this way 512 labels were assigned to a single sample i.e. each label for single feature in the whole vector. After labeling each feature in the vector, then final label to the sample is given by taking the mode of all 512 labels given on each feature. This technique was named as mean mode classifier.

Results on the technique was not up to mark. Reason behind this was exactly as discussed in the section 3.4.2.

3.4.4 Confidence estimation based classifier

Deep learning models are very good at learning when the train and test data is from same distribution. But research have shown that, still deep learning models tend to give very high score or missclassify with high confidence to test images, which could be perturbed in distribution, out of distribution or not even recognizable images in first place [28], [10], [38].

This leads to many research frontiers in the domain of Deep learning, one of which is formulated as the problem of Out-of-distribution detection [15]. Model being able to differentiate between the in-distribution and out-distribution for a specific task will be a major advantage in classification during the combined test phase. Therefore, Out-of-distribution technique

from [5] was used to have layer for confidence measure along with the classification layer. In this way test examples from other increments could be identified as out-of-distribution dataset, and thus it was suppressed in their non-relevant fully connected layer.

Although, this technique was able to identify the out of distribution examples in the incremental setting but the confidence measure had a confidence overlap in out-of-distribution and in-distribution datasets, which was below the acceptable level. Overlap is shown in the Fig (3.3)

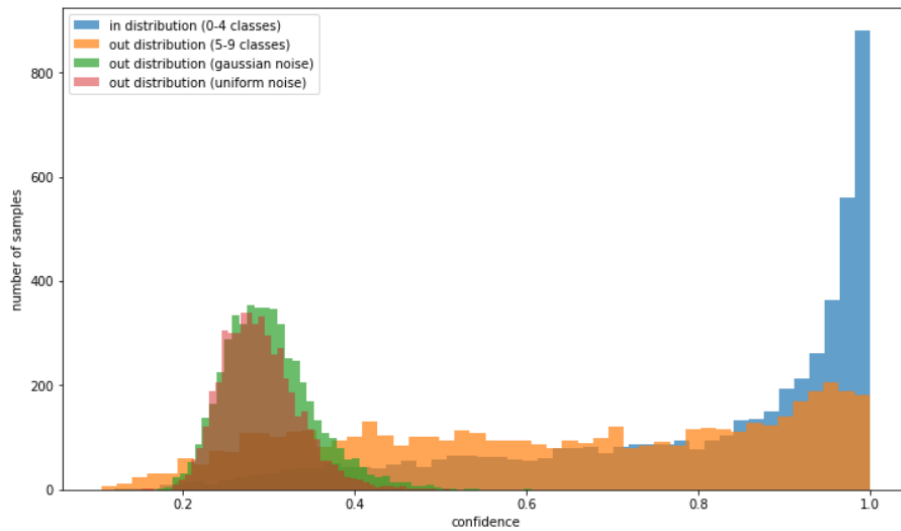


Figure 3.6: Confidence density comparison: It is clearly shown in the graph, that Gaussian, and Uniform noise being Out-of-Distribution data set has high density in very low confidence range, while In-Distribution data set has very high density in high confidence range.

3.4.5 Maxout of stacked logits

Fully connected layer in the each network extension, gives output for all passed samples i.e. both in-distribution and out-of-distribution samples. It was observed that logits outputted for most of out-of-distribution samples from network extension has values less than values outputted for in-distribution samples. Therefore, idea is to use these logits produced for all

samples from all the network extensions. Then stack them sample-wise and use the max function i.e. *torch.max()* along the stacked dimension. Final classification will be made on the output of the max function.

$$y'_f = \operatorname{argmax} \left(\operatorname{concat} \left[y'_i \right]_{i=0}^n \right) \quad (3.1)$$

where: y'_f = logits of i^{th} Network Extension (NE)
 n = Total number of increments at specific time stamp
 $\operatorname{concat}()$ = concatenation function: concatenates logits in specified dimension
 $\operatorname{argmax}()$ = function to get the index of max-value in specified dimension

Advantages of this relatively straight and simple technique, produced results equal to the results produced by the state-of-the-art incremental model, along with following pros.

- I No requirement of retraining over the training data of old increments is required at all.
- II No need to save the training data of old increments.
- III Possible privacy issue is eliminated.
- IV Accuracy for each increment is completely preserved in separate testing setting.

3.5 Implementation Details

In this section all the implementation details of the model are discussed, which are required to reproduce the results given in this thesis.

In proposed model, for the shared and fixed part, Resnet18, Resnet34 and Densenet201 are used. There are some implementation difference in the implementation model, based on these architectures.

3.5.1 Hardware and Software

- In the implementation of all versions of proposed model, for the GPU acceleration, hardware used are NVIDIA GeForce GTX TITAN X and GeForce GTX 1060 GPUs.
- On software part, libraries used for the model development are as follows. The whole model is used in Python language, version 3.7. Anaconda3 is used as default package manager for the development environment. While the deep learning frame work used is PyTorch [29] version 0.4.0 with CUDA version x.

3.5.2 Data set and pre-processing

- Dataset used is CIFAR100 [20]. There are 100 classes in total. The complete dataset has 60000 examples. Where training set has 50000 examples, with 500 examples for each class. While test set has 10000 examples, with 100 examples for each class.
- For the pre-processing part: Dataset has each example of size 32x32x3 which defines as height, width and RGB channels, respectively. Each example was up sampled to 224x224x3 and then normalized before passing to the model.
- During training phase two augmentation techniques are used: Random horizontal flip and Random cropping the image.

3.5.3 Training setup

- For incremental setting complete dataset is divided into equal number of classes per increment – i.e. for total of twenty increments, each increment has 5 classes. For 10 increments, each increment has 10 classes. For 5 increments, each increment has 20 classes.
- Each model version is trained for 150 epochs. Cross Entropy Loss is used for loss calculation. While standard Stochastic Gradient Descent (SGD) is used with learning rate of 0.01 along with 0.9 momentum. Batch size of 64 is used for 5 and 10 increment setting, while for 20 increments setting 32 batch size is used.

3.6 Summary

Proposed model fundamentally is based on the architectural design philosophy in contrast with the rehearsal based designs. Architectural strategy was chosen to avoid catastrophic forgetting in first place. Thus, all the issues related to catastrophic forgetting and rehearsal phase are avoided automatically. Model has two main parts: One is shared and fixed part. This part is actually the back bone of the model and taken from the ReNet34 (except FC layer). The ResNet part of the model is pre-trained on the ImageNet dataset. pretrained model was taken for the reason, that this part is shared and fixed part we must have this part already good enough feature extractor in first place for all the classes in the coming increments without training. Second is the dedicated part which are small CNN layers sub module and will be learned on ever task. Testing and training pipeline are different due to the dynamic architecture of the model. Final classification is done on the basis of max function on the stacked logits from all NEs.

Chapter 4

Performance Evaluation

In this chapter performance of the proposed model is evaluated. Different aspects of the proposed model are evaluated against multiple quantitative measures. Performance of the model is also directly evaluated with the performance of different architectures in different settings.

4.1 Accuracy Comparisons

In my proposed model I used pre-trained Resnet 18, Resnet 34 and Densenet 201 for the shared and fix part of the model. In my proposed model when I used pre-trained resnet18, resnet34, the output of intermediate layers except the last intermediate layer were down- sampled and then concatenated with the output of last intermediate layer. Then all the concatenated outputs are passed to the network extension of the current increment. While, in the implementation of pre-trained Densenet 201, no output of intermediate layers are concatenated. This is because, in Densenet architecture the output of previous layers all already concatenated.

Resnet architecture is one of the best model in deep learning family and used in wide variety of domains. In icarl [9], they used Resnet 34 as well. So in my model I used the resnet architecture for a fair comparison. While, advantage of Densenet architecture, over the resnet architecture is that it can give the comparable performance at less parameters and computation cost. All the results are compared in the coming sections.

Accuracy comparisons were done in two settings: First use of combined testing data from all increments i.e. all previous and current increment at the time of testing, second use of separate testing data from all increments is used - at each increment in testing phase. Therefore for the combined testing data we compared results with state of the art iCaRL [9], and baseline Lwf

[11]. While performance of the proposed model on separate testing data for each increment, results are compared with the results from PackNet[20].

4.1.1 05 increments

In the setting of total of five increments. During each increment, model is trained on 20 new classes. Comparison of our model in different configurations with other implementations are shown in Figure 4.1.

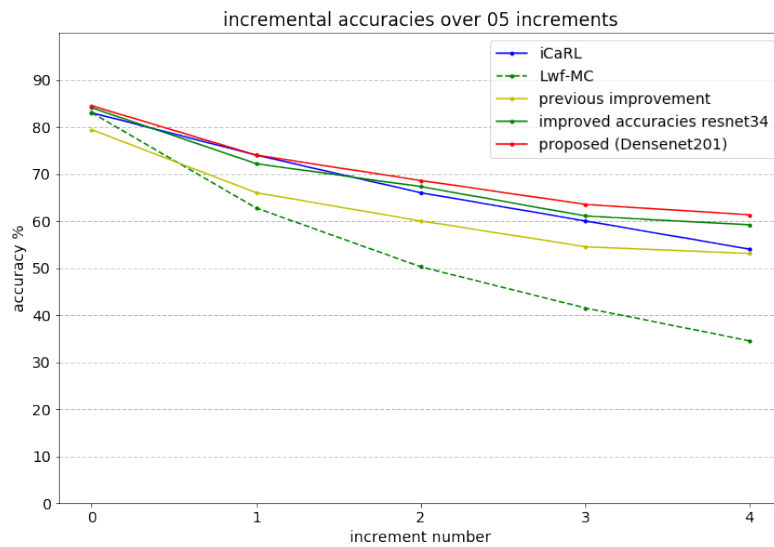


Figure 4.1: Incremental accuracy comparison over 05 increments. Brackets indicate the pre-trained model used in part of proposed model

4.1.2 10 increments

In the setting of total of ten increments. Model is trained on 10 new classes in each increments. Comparison of our model in different configurations with other implementations are shown in Figure 4.2.

4.1.3 20 increments

In the setting of total of twenty increments. Model is trained on 5 new classes in each increments. Comparison of our model in different configurations with other implementations are shown in Figure 4.3.

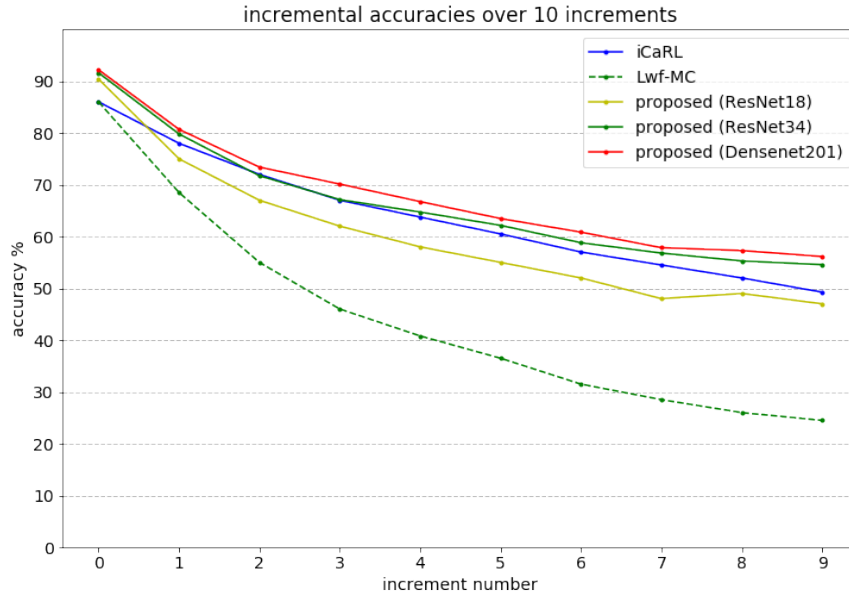


Figure 4.2: Incremental accuracy comparison over 10 increments. Brackets indicate the pre-trained model used in part of proposed model

4.2 Confusion matrices

Figure x, y and z shows the confusion matrices of the 20, 10 and 5 increments. Shown confusion matrices are of model with ResNet34 backbone. Each matrix represents learning over 100 classes in 20, 10 and 5 classes as discussed previously.

4.2.1 Observations

From all the 3 figures x, y and z we can clearly make following observations:

- I Generally, true positive rates for all classes are significantly higher than its respective false negative and false positives
- II Spread of predictions made by the model for each class are almost uniformly distributed.
- III True positive rates in the Matrix where the number of increments are less has generally higher true positive rates than matrix where the number of increments are higher. Same behavior can be shown and verified from accuracy graph figures and accuracy tables.

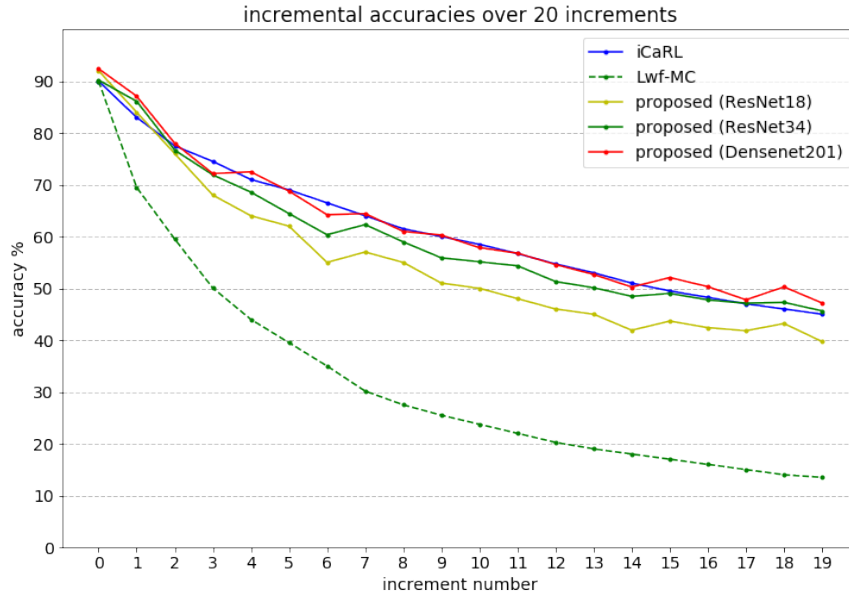
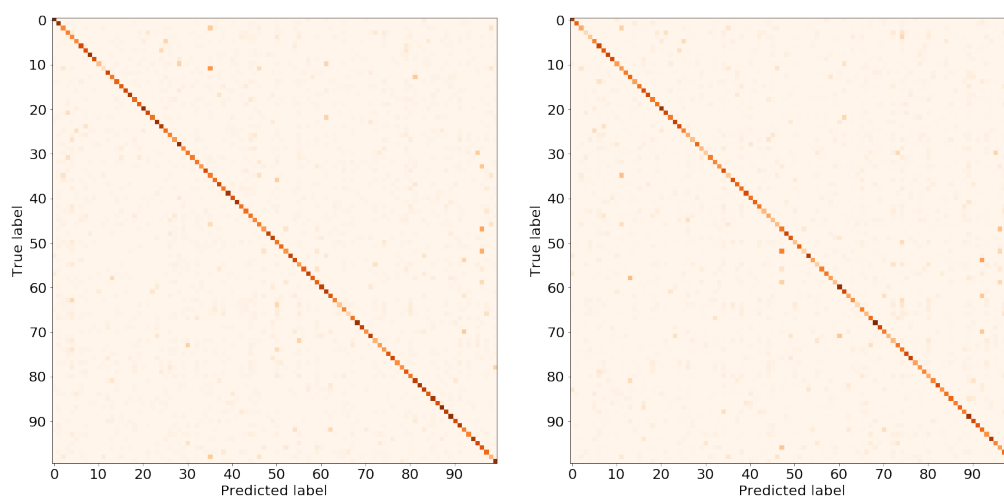


Figure 4.3: Incremental accuracy comparison over 20 increments. Brackets indicate the pre-trained model used in part of proposed model

4.2.2 Analysis

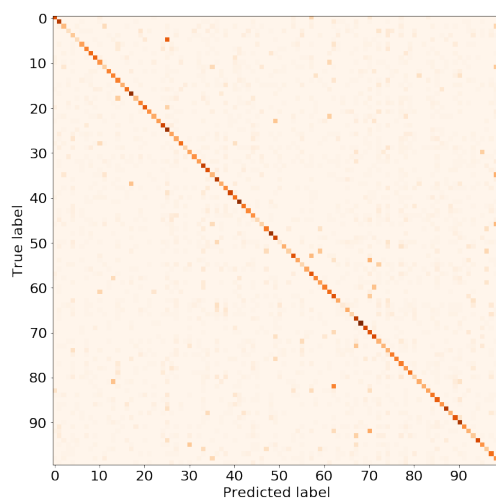
Analysis on the basis of observations made in section 4.2.1 on the confusion matrices of 20, 10, and 5 incremental setting.

- Point (I) shows that learning of the model for all classes from all increments is optimized
- point (I) and point (II) shows that model is capable of retaining the performance for all classes and increments at the balance rate.
- point (II) shows that Learning of the model is not biased towards some specific class(es) or increment(s)
- from point (III) we can conclude that less increments has better overall performance than more increments, although number of classes at respective task are same.



(a) Confusion matrix over 5 increments

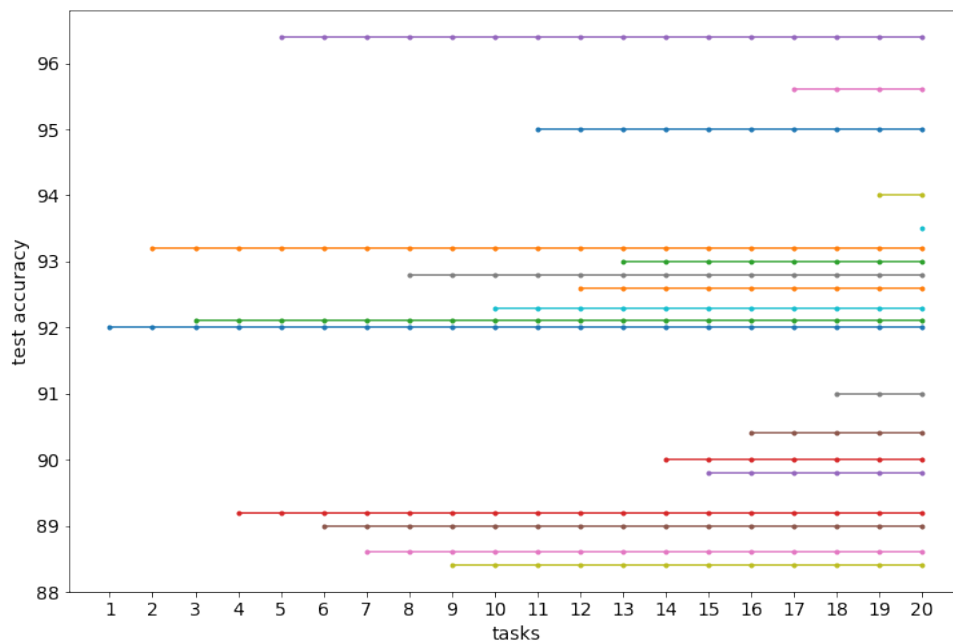
(b) Confusion matrix over 10 increments



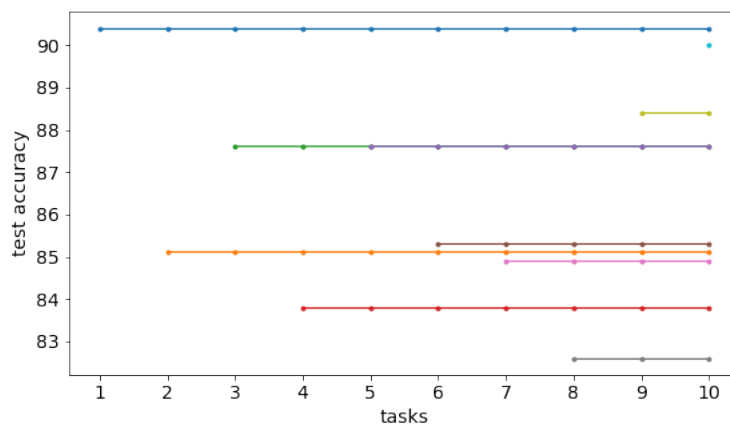
(a) Confusion matrix over 20 increments

4.3 Accuracy of separate tasks - with separate test data set

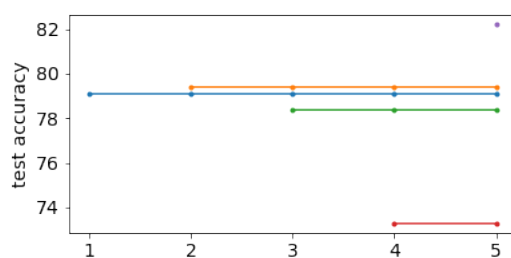
Discretely task specific test data is passed to the model to observe the model performance on separate tasks setting, discretely on tasks' own test data over all the increments. Following analysis is made on the basis of observations from Figure 4.4



(b) 05 increments



(c) 10 increments



(d) 20 increments

Figure 4.4: Accuracy of each task separately, with only task's test data, at all succeeding increments. Starting point of the graph lines from the left shows the task number

4.3.1 Observations

These are the following observations made on the accuracy of separate task tested separately.

- I It is clear from the graphs in figure 4.4, that performance of each task is sustained for all the succeeding increments, at the point where it was during the training time.
- II Overall accuracy for all the tasks differ for all the three increment-settings. For more increment-setting number of classes per task are less so the learning is better and model accuracy is higher and vice versa.

4.3.2 Analysis

From the point (I) and (II) in the section 4.3.1. we can conclude the reasons for the sustained accuracy on the separate task are as follows.

- Design of the model incorporates the dedicated network extension for each new increment. Only the parameters of the dedicated part will be learned for the task in hand.
- Training pipe line of the model train the only the dedicated part of the model specifically added for that specific task.
- Shared part of the model is fixed for all increments. So no shared parameters are changing. Therefore weights does not get disturbed during the training phase of new increment. So from the results it is clear that catastrophic forgetting has avoided in by the model, as discussed in section 3.3.2 in detail

4.4 Individual accuracies of all tasks - on combined test dataset

Table x, y and z shows the observation of the overall accuracy performance trend of each task individually over the span of all succeeding increments. Following observation can be made from the analysis of the table:

4.4.1 Observations

From the the data available in the Table x, y, and z we can make the following observations.

- I General trend for all classes is, the gradual drop in the accuracy with each increment, except a few points where accuracy increases for a single increment in the middle but then again follows the general trend.
- II Latest task added at each increment has generally more accuracy than all preceding tasks.
- III Performance for each task has same behavior, showing the un-biasness of the model. Contradicting observation of presence of slight biasness in the model, as discussed in the point above is resolved in succeeding increments.

4.4.2 Analysis

On the basis of observations made in the section 4.4.1, we can conclude the following points on the design and performance of the proposed model:

- Point (I) shows the sustained behaviour of the model for all the tasks and classes added during the course of 20, 10 and 5 increments.
- From point(II) we can conclude that although shared part of the model has fixed parameters. But by the design of the pipeline of the model, *running mean* and *running variance* for shared part are different for all the increments.
- Therefore, Each increment uses the shared part of the model which has fixed parameters but training pipeline of the model uses the shared part from the latest increment. Which has *running mean* and *running variance* from latest increment. Therefore, generally performance is slightly higher for the latest task.
- from point (III) we can conform that proposed model exhibits the un-biasness in learning and in incremental performance for all the classes and task learned.

Table 4.1: Individual Accuracies of each task over 5 increments

Inc → Task ↓	1	2	3	4	5
1	83.15	68.3	62.25	56.3	56.15
2	-	74.15	63.1	59.8	57.6
3	-	-	67.15	61.6	55.85
4	-	-	-	57.5	51.6
5	-	-	-	-	63.9
total Acc	83.15	71.25	64.16	58.80	57.02

Table 4.2: Individual Accuracies of each task over 10 increments

Inc → Task ↓	1	2	3	4	5	6	7	8	9	10
1	90.9	78.0	69.9	64.5	65.2	62.6	55.6	54.6	56.6	53.3
2	-	77.2	68.3	65.3	62.2	59.9	56.3	53.6	53.1	52.1
3	-	-	72.1	62.1	58.9	56.4	53.4	52.7	54.4	53.8
4	-	-	-	70.4	64.2	59.3	57.0	54.4	51.7	49.8
5	-	-	-	-	63.1	56.5	53.2	53.3	52.8	47.5
6	-	-	-	-	-	59.4	50.3	46.0	46.1	41.4
7	-	-	-	-	-	-	65.2	58.3	51.3	54.4
8	-	-	-	-	-	-	-	53.9	45.0	48.6
9	-	-	-	-	-	-	-	-	63.4	54.2
10	-	-	-	-	-	-	-	-	-	55.7
total Acc	90.9	77.6	70.1	65.6	62.7	59.0	55.8	53.4	52.7	51.1

4.5 Summary

In this chapter, we have evaluated the performance of proposed model in comparison with the other models and state of the art models in the incremental setting. We have evaluated the model with in different task sizes (number of classes in each tasks) and different number of increments.

Results on Aggregate Incremental accuracy have shown that our model has performed better or better or equal to that of the state of the art model in different incremental and task size settings, without any requirement of rehearsal phase, retraining data and disk requirement for the data saving, which was absolutely necessary for the state of the art model.

From the analysis of confusion matrices, it has been analyzed that our model learning is completely unbiased and has not shown any biasness towards any increment, task or classes. Furthermore, analysis on the individual

tasks' accuracy in combined test data setting, conforms the same unbiased learning behavior of our model. Which is the much required feature of any incremental learning model.

From the perspective of separate task performance, when tested separately on the test data of its own task only, it has been observed that our model has sustained the performance on each task without any level of degradation.

Table 4.3: (a) Individual Accuracies of each task over 20 increments

Inc → Task ↓	1	2	3	4	5	6	7	8	9	10
1	91.2	83.6	70.4	65.4	62.8	63.8	57.8	55.2	55.0	51.2
2	-	84.2	80.0	74.8	69.2	59.8	49.0	54.2	49.4	44.4
3	-	-	78.8	71.6	68.8	66.0	58.0	60.4	53.0	54.6
4	-	-	-	66.8	60.2	60.0	59.4	56.6	54.8	52.4
5	-	-	-	-	79.0	74.0	63.6	67.6	63.2	54.0
6	-	-	-	-	-	60.0	59.4	56.8	50.6	46.2
7	-	-	-	-	-	-	70.8	65.2	66.2	58.0
8	-	-	-	-	-	-	-	63.8	58.8	58.2
9	-	-	-	-	-	-	-	-	61.4	57.6
10	-	-	-	-	-	-	-	-	-	60.4
total Acc	91.2	83.9	76.4	69.6	68.0	63.9	59.7	59.9	56.9	53.7

Continued ...

Table 4.4: (b) Individual Accuracies of each task over 20 increments

in continuation with table 4.3										
Inc → Task ↓	11	12	13	14	15	16	17	18	19	20
1	53.0	45.0	50.0	39.2	41.6	44.0	38.8	36.4	40.6	39.2
2	47.6	48.8	43.4	41.4	41.4	40.6	46.8	42.0	43.6	40.6
3	51.2	52.4	46.6	46.8	41.4	39.8	43.0	44.2	43.2	39.4
4	53.6	52.4	48.2	48.2	45.8	46.2	43.0	42.0	44.0	46.6
5	49.4	56.0	51.2	54.8	45.0	41.0	41.6	46.0	44.6	46.6
6	45.2	48.8	44.2	46.0	43.4	43.4	42.8	46.4	41.2	42.8
7	58.2	53.6	58.8	55.6	53.6	55.8	44.0	47.2	49.6	48.2
8	49.6	54.2	47.8	53.6	47.2	47.6	43.2	51.0	46.6	47.6
9	54.6	55.8	57.4	52.6	54.8	60.0	44.2	43.4	47.6	52.6
10	56.8	54.8	55.4	50.8	51.2	48.8	47.0	46.4	46.0	45.0
11	61.2	44.0	46.8	35.4	37.2	40.6	43.4	28.6	43.4	31.0
12	-	50.2	47.8	44.8	44.2	38.4	43.0	41.4	41.2	38.4
13	-	-	49.4	51.6	43.2	44.6	35.4	37.0	35.2	40.0
14	-	-	-	58.4	54.4	55.0	47.8	50.8	51.6	53.4
15	-	-	-	-	48.0	43.8	33.0	32.8	38.0	42.4
16	-	-	-	-	-	48.4	42.2	38.6	43.0	45.4
17	-	-	-	-	-	-	54.6	41.2	46.2	33.2
18	-	-	-	-	-	-	-	56.4	50.4	51.2
19	-	-	-	-	-	-	-	-	48.2	46.0
20	-	-	-	-	-	-	-	-	-	43.4
total Acc	52.8	51.3	49.8	48.5	46.2	46.1	43.2	42.9	44.4	43.6

Chapter 5

Conclusions and Future Prospects

Incremental learning is an active research problem in the field of machine learning. It is much needed capability for a machine learning algorithm to learn with time just like as we humans do.

In the research work done under this thesis, I have proposed the deep learning based model for the incremental learning. Results have shown that proposed model has performed equal to the state of the art incremental algorithm, in the incremental setting of 20, 10 and 05 increments on the total of 100 different classes. Our model has achieved results without any need of retraining and requirement of saving training data of old classes on the disk, which was required by the state of the art algorithm. This is achieved by designing the model architecture on the design philosophy of avoiding the catastrophic forgetting.

Accuracy achieved in this research work and by iCaRL [31], both fall from somewhere 90% to almost 50% in just 20 increments. This performance is still far from being application ready. Therefore, there is a large performance gap, which yet to be covered by the incremental learning algorithms. In this regard, some research directions are discussed in the following section.

5.1 Future research prospects

Incremental learning is an active research topic and still has large gap to improve. Main gaps are the reduction of accuracy drop rate and keeping the rate of model size-increase as low as possible. Accuracy drops by 40% for 100 classes' dataset on the span of 20 increments. Which is still too high for the model to considered application ready. Similarly model size increases

with every increment, which must be controlled to the factor where it can be applied for long-term incremental setting.

There are generally two main directions, i.e. architecture based and rehearsal based designs, in which research community has worked to improve incremental learning capability of the model. Therefore, some possible future research in the aforementioned directions are discussed in the section below

5.1.1 Architectural Based Designs

- It is obvious from the design of all discussed incremental models, their model size increases every time new increment is added to the model. For a true incremental learning model, will have to learn for long span of time. But increase in size with every increment can become the bottle neck at some point so this issue must be addressed by effectively minimizing the rate at which model size increases on increment.
- Model capable of knowing what it does not know could have a great impact on the improvement. This due to the fact that in single increment model automatically learn to differentiate between intra-task classes. But how would it will able to learn inter-task classes. This can be achieved by making the model know what it does not know. Therefore, model will tend to avoid from giving classification probabilities to classes it has not trained on. This will significantly reduce the false positive rate and improve the overall accuracy of the model.
- In the light of above point, one approach used in the model could be – Out of Distribution detection capability [15], [23]. In this way model will be able to differentiate between the tasks learnt in different increments on different time stamps. Therefore, false positives rate of the model will be less and overall accuracy will improve.
- Another approach which could achieve the similar results discussed in the above point is, the use of Bayesian Neural Networks (BNNs). As BNNs can be employed for differentiating between the examples from in-distribution datasets and out-of-distribution datasets. Thus false positive rates for other tasks can be mitigated and overall accuracy can be improved.
- Deep neural network has inherent issue of catastrophic forgetting. Design changes like, structure semantics to the core of deep learning architecture can make them even more robust and semantically intelligent. This might help neural network know what it don't know and

could simply eliminate the issue of catastrophic forgetting also. A similar approach in CapsNet [35] proposed where neurons of previous layer transfer their knowledge only to the relevant neurons of preceding layer only.

5.1.2 Rehearsal Based Designs

- In the rehearsal based models, we have to use modified loss function which can keep the balance between the performance of the old tasks and learning of the new task. Like the loss function used in [31]. A better loss function can deliver even better results, in terms of retaining accuracies over previous increments as well as validation accuracy over the current task.
- Rehearsal based model clearly require time for the rehearsal part. Model can be designed in order to incorporate the changes required by the new task, while retaining the crucial parameters for the performance of old task in a way that each increment requires: 1) less time, 2) Less Examples from old tasks' training data and 3) fast convergence of loss to the minima.
- - Another technique used for the rehearsal based models is, Generative approach. As discussed in the section 3.1, this can have a great impact, if generative models are able to generate semantically correct images donating the same data distribution as in the real training data for the all the classes from the previous tasks. This will eliminate the saving requirement of training data on the disk. Privacy concern for the training data will also not be an issue any more if any.

Bibliography

- [1] W. Abraham and A. Robins. Memory retention—the synaptic stability versus plasticity dilemma. *Trends in neurosciences*, 28(2):73–78, 2005.
- [2] E. Alpaydin. *Introduction to machine learning*. MIT press, 2020.
- [3] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.
- [4] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.
- [5] T. DeVries and G. Taylor. Learning confidence for out-of-distribution detection in neural networks. *arXiv preprint arXiv:1802.04865*, 2018.
- [6] J. Fagot and R. Cook. Evidence for large long-term memory capacities in baboons and pigeons and its implications for learning and the evolution of cognition. *Proceedings of the National Academy of Sciences*, 103(46):17564–17567, 2006.
- [7] R. French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.
- [8] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.
- [9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.
- [10] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

- [11] D. Hassabis, D. Kumaran, C. Summerfield, and M. Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, 2017.
- [12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [13] 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*, pages 770–778, 2016.
- [14] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 630–645, Cham, 2016. Springer International Publishing.
- [15] D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.
- [16] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network, 2015.
- [17] G. Huang, Z. Liu, L. van der Maaten, and K. Weinberger. Densely connected convolutional networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [18] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- [19] J. Kim, J. Kim, and N. Kwak. Stacknet: Stacking feature maps for continual learning. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.
- [20] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- [21] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [22] Z. Li and D. Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

- [23] S. Liang, Y. Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.
- [24] A. Mallya and S. Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.
- [25] M. McCloskey and N. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.
- [26] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka. Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, editors, *Computer Vision – ECCV 2012*, pages 488–501, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
- [27] 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*, 35(11):2624–2637, 2013.
- [28] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.
- [29] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimeshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.
- [30] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.
- [31] SA. Rebuffi, A. Kolesnikov, G. Sperl, and C. Lampert. icarl: Incremental classifier and representation learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [32] M. Ristin, M. Guillaumin, J. Gall, and L. Van Gool. Incremental learning of ncm forests for large-scale image classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

- [33] D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.
- [34] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
- [35] S. Sabour, N. Frosst, and G. Hinton. Matrix capsules with em routing. In *6th international conference on learning representations, ICLR*, pages 1–15, 2018.
- [36] H. Shin, J. Lee, J. Kim, and J. Kim. Continual learning with deep generative replay. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2990–2999. Curran Associates, Inc., 2017.
- [37] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.
- [38] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.
- [39] A. Tsymbal. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 106(2):58, 2004.
- [40] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.
- [41] M. Welling. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1121–1128, 2009.