An Implementation of Patching Algorithm using DNN as feature extractor on image datasets

Master-Thesis von Shuai Jiang aus Darmstadt

Tag der Einreichung:

1. Gutachten: Prof. Dr. Johannes Fürnkranz
2. Gutachten: Sebastian Kauschke

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Computer Science Department
Knowledge Engineering Group

An Implementation of Patching Algorithm using DNN as feature extractor on image datasets
Master-Thesis von Shuai Jiang aus Darmstadt

Vorgelegte Master-Thesis von Shuai Jiang aus Darmstadt

1. Gutachten: Prof. Dr. Johannes Fürnkranz
2. Gutachten: Sebastian Kauschke

Tag der Einreichung:

Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den November 27, 2018

(Shuai Jiang)

Abbreviations

DNN Deep neural network

MLP Multi Layer Perceptron

MOA Massive On-line Analysis

ConvNets Convolution Neural Networks

AI Artificial Intelligence

E experience

T tasks

P performance measure

MNIST Mixed National Institute of Standards and Technology

Conv2D 2D convolution

FC Fully Connected Layer

MS Model selector

CBR Case-Based Reasoning

Nomenclature

List of Figures

List of Tables

Abstract

Deep neural networks are become popular in machine learning nowadays. The layered architecture makes it easy for continuous learning compared to traditional machine learning methods like decision tree, random forests, etc.. The hidden layers of the network represent features in different stage of abstraction. These layers can be attached and reused for modified or even new problems, namely for concept drift and transfer learning.

In the real world, the data are not always available immediately. Sometimes, data come in pieces and in a relative long time. The classifiers must therefore be trained continuously with the data currently available. The neural networks are suitable for such situations.

In this thesis the convolution neural networks (CNN) are used to implement the patching algorithm, which is an approach to deal with changed image inputs that represents concept drift and transfer learning. An original network is assumed to classify unchanged data very well. When changed data come, a special classifier is trained to predicate if the original network will misclassify. Data are then diverted to the patching network or original network accordingly. This way, the usually expensive original network can be reused and the whole network must not be trained from scratch.

Experiments on engagement in different layers of neural networks are also performed. Several datasets are applied for these experiments. We engage into each network layer in small dataset and engage into each block of network for relative big networks to find the best layer for engagement.

We find that the patching algorithm used in this thesis can adapt changed data well and quickly. For convolution neural network, which is applied in this thesis, the engagement on the last layers can achieve better performance, since special features of changed data are necessary and import for the classification.

Contents

1 Introduction

Neural network is an important technology in machine learning. In 1943, neurophysiologist Warren McCulloch and mathematician Walter Pitts published an article on how neurons might work, which first introduced the concept of neurons into the field of computation, and proposed the first artificial neuron model, which opened the door to neural network. But because of the limitation of the calculation method and capacity at that time, the research of neural network felled into trough. This changed since 2005, when new methods emerged and GPUs start to be used that allowed to expand the depth of the neural network and allowed the network to be trained faster. In the past ten years, with the developing and more detailed of the research, great progress has been made in neural network. Today, deep neural networks exhibit good intelligence in the fields of image classification, pattern recognition intelligent robotics, automatic control, predictive estimation, medicine, economy and many more, which has successfully solved many practical problems that are difficult to be solve in modern computers. Computers are now better than humans at recognising and sorting images. For example, Baidu's Minwa supercomputer can sort a million images into a thousand predefined categories with an error rate less than a typical person.

Deep neural network (DNN) is an artificial neural network with multiple layers between input and output layers. Because of the layered architecture, it can be adapted to new and complex problems relatively easier compared to other networks. The deep network architecture with an input layer, hidden layers and the output layer, is also called Multi Layer Perceptron(MLP). The hidden layers can be seen as a "distillation layer" that distill some of the important patterns from the inputs and pass it onto the next layers. It makes the network faster and efficient by identifying only the important information from the inputs leaving out the redundant information.

Deep learning algorithms are trained to learn progressively using data. Nowadays, huge amounts of data are available. Thus building a high-depth neural network for specialized tasks becomes possible. But large data sets are required to make sure that the machine delivers desired results. As human brain needs a lot of experiences to learn and deduce information, the analogous artificial neural network requires copious amount of data. The more powerful abstraction you want, the more parameters need to be tuned and more parameters require more data.

Convolution Neural Networks (ConvNets) is a specific type of artificial feed-forward neural network that has been successfully applied to image recognition. ConvNets is an efficient identification method that has been developed in recent years and has attracted widespread attention. In the 1960s, when Hubel and Wiesel studied the local sensitive and directional selection of neurons in the cat's cerebral cortex, they found that their unique network structure can effectively reduce the complexity of the feedback neural network, and then ConvNets was first proposed. Today, ConvNets have become one of the research hotspots in many scientific fields,especially in the field of pattern classification.

Like other kinds of artificial neural networks, a ConvNets has an input layer, an output layer and various hidden layers. Some of these layers are convolutional, using a mathematical model to pass on results to successive layers. This simulates some of the actions in the human visual cortex.

1.1 Motivation

In the practice, sometimes we will face scenarios where needed parameters and data can be changed over time. For example, a changing environment, a concept drift or transfer learning may happen. It is very difficult or impossible to adapt the legacy or expert models, because the required expertise for reprogramming them is no longer available. The result is, that originally learned model is underperforming or even unusable.

Nowadays, data was born at a high speed from a wide variety of data sources. Actually, all data can be seen as data stream. Due to the data stream arrives in high speed, the related algorithm for it has limitations in time and space. In order to break through this restriction, the algorithm must meet some requirements. ~~For example, despite there is only one instance can be observed at a time in a limited time, it should always ready to predict.~~

The most stream learning algorithms only do experiments on small datasets, which does not correspond with the real world and can not convince people. For this purpose, the Massive Online Analysis (MOA) framework[1] is introduced. MOA was built based on WEKA[2]. MOA can be used to evaluate data stream classification and clustering, it can be used for big data stream mining as well.

The goal of patching is to re-use old models, which were trained with a big amount of data and with high costs, for new problems, in order to reduce the costs. While this works for traditional machine learning techniques, the neural networks are more performant and more suitable for continuous training with data streaming. Thus, it is a promising way to use the patching technology on neural networks. The combination of advantages of both neuronal networks and patching can be very useful on application fields with concept drift or transfer learning.

1.2 Goal of this Thesis

In this Thesis, I try to find a way to reuse the existing models and to solve new problems by patching. Patching is usually applied to traditional Machine Learning (ML) technologies such as decision trees and rules. Here, it is tried extend a Patching-Method to a neural network (NN). ~~Theoritically, the extend can, but doesn't have to be a neuronal network. But we use convolution neural network (CNN) to implement the patching technology.~~

~~Meanwhile, different datasets are used to examine the performance of patching networks. Some metrics are defined to compare the classifiers with different classification algorithms. Engagement of patching network is also examined for a best engaging layer. Some data change variants that simulate concept drift and transfer learning are defined for experiments. The optimization of the patching network by adjusting parameters and add some special layers is performed.~~

~~Furthermore future work will be discussed xxxx.~~

1.3 Thesis Structure

I divide this thesis into six chapters,namely (1) Introduction, (2) Theoretical Background, (3) Related Work, (4) Datasets, (5) Experiments and their Evaluation, and (6) Conclusion and Future work.

In the Theoretical Background chapter, the fundamental concept of Machine Learning, Convolutional Neural Network and Patching is introduced. It helps to understand the principle of research object and clarify the method of evaluation. In chapter 3, existing approaches for patching and related themes are presented. The datasets chapter describes how the dataset and labels are prepared for this thesis. In the Experiments and Evaluation chapter, data change variants and detailed experiment setups are presented. Evaluation and analysis of the experiment results are also presented and discussed. In the last chapter, the results of the evaluation are reviewed and summarized and future work is proposed.

2 Theoretical Bachground

Convolutional neural networks and the patching approach are the main topic of this thesis. Like Wermter, Stefan et al.[3] mentioned, Deep Learning is a sub class of Machine Learning. Therefore, a general introduction of machine learning is introduced.

In this chapter, the basics of machine learning and Convolutional Neural Network are introduced, followed by the explanation of the Patching concept.

2.1 Machine Learning

Machine Learning Problem defined by Mitchell 1997 [4] :

"A computer program is said to **learn** from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

The task T defines the tasks that should be achieved by computer program. This can be Classification, Density estimation, Testing and matching and more others. For example, in recognizing Spam-Mail, the task T is to sort E-mails into categories. The performance measure P is a measurement of the quality of the learned task. For recognizing Spam-Mail, the Performance Measure P could be the weighted Sum of Mistakes. The experience E describes source of the data which is used to learn. In recognizing Spam-Mail the experience E could be handsorted a set of E-mail messages.

As shown in Figure 1, the basic workflow of machine learning has two phases: a training phase and an operational phase. In the first phase, a trainer or scientist can update the training dataset at any time to automatically train a new persisted model. In the second phase, any new coming input data will be processed with the updated model. Previous predictions can be re-computed with the updated model, new models also could be backtested on previously input and versioned data. This is certainly a very rough categorization. It can be categorized in finer phases like use-case conception, feasibility study, model design and deployment, and model maintenance etc.

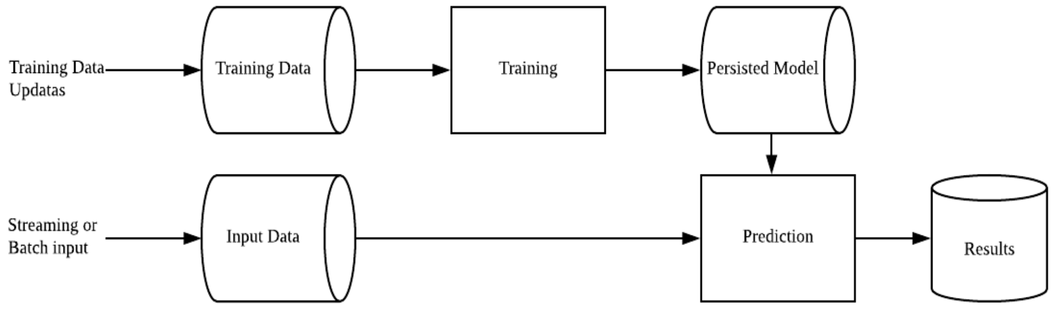


Figure 1: Basci Machine Learning 'workflow'

2.2 Neural Network

2.2.1 Artificial Neural Network

Artificial Neural Networks (ANN) [5] also called Neural Network (NN) have been used since 1940s. It is inspired by the learning system of the human brain.Since the brain is an extremely complex network with many neurons connected with each other,the neural network also has a series of units that are densely connected together. But this artificial neural network is much more simpler than the human brain. In the 1960s, Rosenblatt [6] proposed the perceptron model, which laid the foundation for today'neural network. A typical NN consist of an input layer, an output layer, and hidden layers are used for processing the inputs received from the input layers. NN is an excellent tool for finding patterns, which are far too complex or numerous for a human programmer to extract and teach the machine to recognize.

It is only in the last several decades, when NN has become a major part of artificial intelligence(AI). This is due to the arrival of a technique called "backpropagation", which allows networks to adjust their hidden layers of neurons in situations where the outcome doesn't match what the creator is hoping for. For example, a network is designed to recognize dogs, but it is often misidentifies as a cat.

Another big step forward is the arrival of deep learning neural networks, in which different layers of a multilayer network extract different features until it can recognize what it is looking for.

Nowadays, NN is under the name of deep learning and is widely used. There are two main reasons are mentioned by Goodfellow et al. [7]to explain why it has only recently become useful:

1. **computing power**

2. **large amounts of labeled data**

Deep learning requires large amounts of labeled data. For example, driverless car development requires millions of images and thousands of hours of video. Deep learning requires substantial computing power. High-performance GPUs have a parallel architecture that is efficient for deep learning. When combined with clusters or cloud computing, this enables development teams to reduce training time for a deep learning network from weeks to hours or less.

Now I will briefly introduce the basis of NN. In Figure 2, a perceptron is shown. Perceptron is a single layer neural network and a multi-layer perceptron is called Neural Networks. The perceptron consists of 4 parts: Input values or One input layer, Weights and Bias, Net sum, and Activation Functions. The perceptron has multiple inputs X_i with their weights W_i, for each input value is combined with the weights : $X_i \cdot W_i$.

The weights will be continusely learned during the training and at the end all the multiplied values are added that is called Weighted Sum. After the Weighted Sum is calculated,this Weighted Sum will be applied to the correct Activation Function. Activation Function is also known as Transfer Function, which is used to determin the output of neural network like yes or no. It maps the resulting values in between 0 to 1 or -1 to 1. (depending upon the function). This Activation Function activates the output, if the threshold is fulfilled.

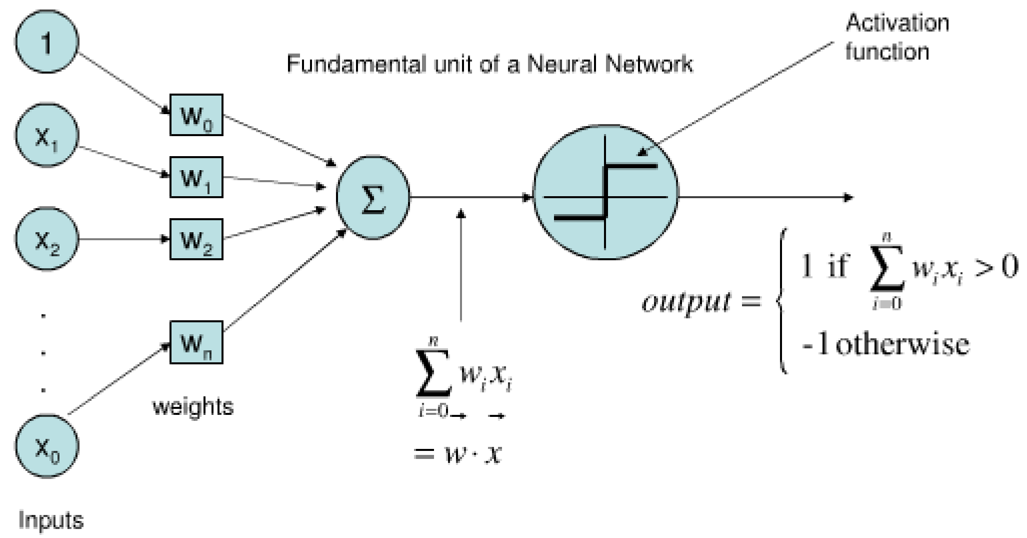

Fundamental unit of a Neural Network
Activation function
1
X_1
X_2
X_0
W_0
W_1
W_2
W_n
weights
Inputs
Σ
$\sum_{i=0}^{n} w_i x_i = \vec{w} \cdot \vec{x}$
$output = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 \text{ otherwise} \end{cases}$

Figure 2: Schematic of Rosenblatt's perceptron

This Section is only a short introduction to neural networks, and the information are taken from the MW Gardner et al.[8].

2.2.2 Convolutional Neural Network

Convolutional Neural Networks (ConvNets) is a kind of Artificial Neural Network. ConvNets have proven to be very effective in fields such as speech analysis and image recognition. It consists of an input layer, an output layer, multiple hidden layers, and usually, a fully connected layer is attached in the end. The hidden layers consist typically of convolutional layers, pooling layers. In this section, the ConvNets operation for the processing of images and individual layers and the details of their hyperparameters and their connectivities are briefly described.

Convolutionly Layer

The convolutional layer is very suitable for image recognition. Due to the large input size associated with images, a big number of parameters (weights) are necessary for each neuron in a fully connected layer. For example, a image with 200x200 pixels needs 40000 weights. A convolutional layer, on the contrary, with e.g. 5x5 kernel size needs only 25 learnable parameters, regardless of the image size. The goal of convolutional calculation is to extract features from images automatically. The general consensus is that the first layers extract information of basic edges and patterns, while the deeper layers become sensitized to specific shapes and patterns.

Convolution operation

If an image (black-white) has a height H and a width W, x is then the tensor x R(H×W). The kernel (filter) is also a tensor x R(H'xW'), where H' < H and W' < W. A convolution operation is to place the filter tensor on the upper left corner of the input image. Then multiplication is done with values of the input matrix and the corresponding values in the convolution filter. The values are then added together to a result scalar, which is placed in the first position of the result tensor. The filter is then moved some amount of steps to the right. The amount of steps is called stride. This is repeated for the entire row. Then the filter shifts down with the same stride until the whole image is covered. This process is illustrated in the following Figure3 :

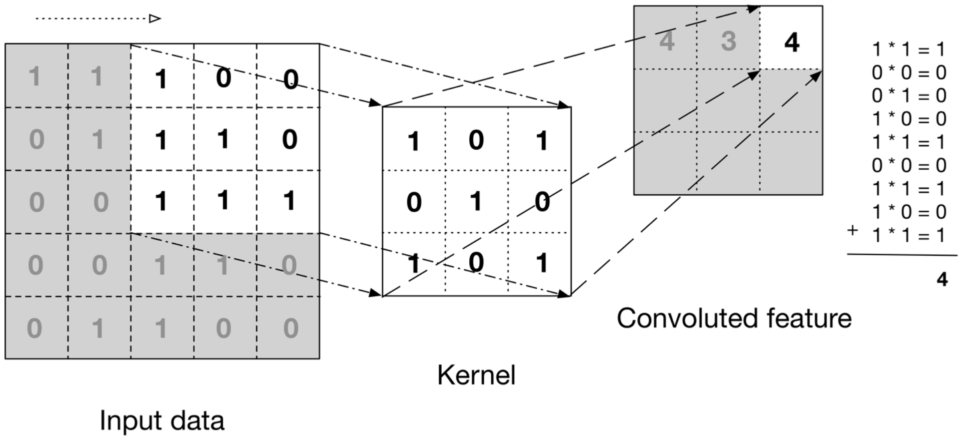


Figure 3: Basci Machine Learning 'workflow'

Pooling Layer

Pooling is another important concept in CNN. It is an art of downsampling layer. There are many kind of non-linear pooling functions. The most usual one is max-pooling. It divides the image into several subregions. The max values of subregions are used. The effectiveness of pooling can be explained that the position of a feature is less important than the relative position to other features found. Pooling can reduce the size of parameters and thus, reduce the calculation costs and overfitting.

Output layer

After the convolution and pooling layers, the features found are collected and flattened. Full connected layers are applied for classification. A typical output layer is softmax, which classifies the image into classes.

2.3 Neural Network Patching

2.3.1 The concept of patching

In this section the general idea of the patching algorithm as described in Kauschke and Fürnkranz(2018) [9] is introduced.

The concept of the patch principle is as follows. We assume a general instance space D of instances x with labels l(x), and a black-box classifier C_0. C_0 is immutable and can classify the examples of D well. We now receive new batches of examples D_i. Instead of addressing this problem directly by training a classifier C_i from instances of D_i, a Patching approach to create two classifiers (E_i and C_{ij}) is used. E_i is a binary classifier that can identify error regions, in which C_0 misclassifies data of D_i. C_{ij}) is a patch classifier to learn the error regions.

The idea of patching is related to several well-known concepts in machine learning, in particular the work on transfer learning and concept drift is relevant. ~~Some ensemble methods assign weights locally, but such weights are determined based on training data only.~~ There has not been much work on ensemble methods to address the transfer learning problem.

In Gao et al.2008 [10], they propose a locally weighted ensemble framework to transfer the combined knowledge to a new domain that is different from all the training domains. They analyze the optimality on expected error reduction by utilizing the locally weighted ensemble framework as compared to both single models and globally weighted ensembles.

By patching, old models trained with big amounts of data with a high cost can be furthermore applied for new problems. This way, new problems can be solved with relatively lower costs. This works well for traditional Machine learning (ML) techniques, but there are applications, in which Neural network(NN) are more performant. Thus, extending NN by a general patching technology is a promising way to benefit from the patching idea by keeping the advantages of the NN for the particular application field.

The concept of patiching is a bit similar to the one of Case-Based Reasoning (CBR). CBR [xy] is used since it is easier to adapt a former solution of a similar problem than to construct a new solution from scratch. The Case Base (CB) holds a set of pairs [problem, solution] of former problems and performs four steps, namely (1) case retrieval, (2) case reuse, (3) case revision, and (4) case retaining. More detail of CBR is described by E. Hullermeier [11].

2.3.2 Patching of Neural Networks

~~In this thesis I will use patching of CNN. Since neural networks are normally using Backpropagation method for training. Usually the newest coming data are adapted into neural network regarding a changed environment.~~ However, this can be costly and this may lead to forgetting (French 1999) [12]. In order to avoid this problem, a special method is learned, which trains only a part of the network (Yosinski et al.2014)[13]. This can handles data quickly, with limited resources and should have better accuracy and faster recovery speed than other methods.

The idea of patching is depicted in Figure 4 from Kauschke and Fürnkranz(2018) [9] followed in several steps, organge is the original network and green is the patch:

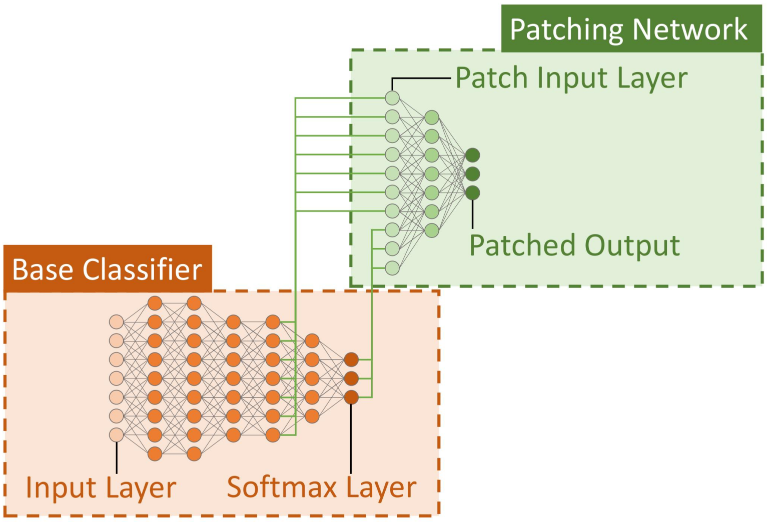


Figure 4: Patching of Feed-Forward Networks

1.Create and train a basic classifier C_0 using original data.

2.Create a binary classifier E_i.In this step we receive the new labeled data stream,we use these new data to build a classifer E_i to determine the error regions that C_0 will misclassify.

3.Create and train a patching classifier C_i using original data. The patching classifier C_i is used to learn the error regions.

4.Using classification C_i ,If $E_i$is true. In this step, E_i is executed and gives a positive result,then the classification will be divert from $C_0$to C_i.

Engagement is a technique to avoid "forgetting" and to improve performance of networks. As shown in Figure 4, some layers have to be selected and be engaged with a patching network. That means, the output of the engagement layer is used as input of the patching network.

In our experiment, different layers will be selected and be engaged under variety environments to determine the best engaging layer.

3 Related Work

Today,a lot of research has been performed due to concept drift [14] or other environmental changes,e.g. transfer learning [15].In this thesis ,our proposed solution (patching)to deal with the data changing via an adaptation mechanism is discussed. In this section,an overview of related work in this field is given.

3.1 Adaptive Learning with Neural Networks

In the real world, learning algorithms have to address several issues such as great amount of data, data which are underlying continuous change and generation over time. That means, the information and data are not available from beginning. They are received continuously and thus must be handled sequentially in real time. Albert Bifet et al.[1] handles data streaming, real time analytics with open source framework MOA (massive online analysis).

3.2 Concept Drift

Concept drift is a phenomenon in Machine learning .Concept drift has a great impact on accuracy and reliability of many applications in the real world. Indrė Žliobaitė [16]gives an overview of the concept drifing problem Because data is changing over time. The relationship between input and output is underlying problem. Geoffrey I. Webb et al. [14] addresses this problem and presents quantitative drift analysis techniques. A method for detection of changes in the probability distribution of examples is provided by J Gama et al.[17].

3.3 Transfer Learning

According to Lisa Torrey et al.[15]:

> **Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned.**

In the supervised machine learning, additional information of new task is combined with existing knowledges, so that transfer learning can be realized (Hoffman et al. 2014)[18]. In the paper of"How Transferable are Features in Deep Neural Networks"(J Yosinski et al. 2014)[13],the generality and specificity of neurons in each layer of a deep convolutional neural network are analyzed.

This thesis is related to both concept drift and transfer learning in that we want to enable an existing neural network classifier to adapt for concept drift on a stream of data, as well as to a completely new scenario as in transfer learning, ideally with as few labeled examples as possible. In the following sections we will describe the core idea of our method and how it can be used to achieve said goals.

4 The Datasets

This chapter gives an overview of the datasets we used. Making a right choice of datasets is very important for this thesis. In classification tasks, the datasets should have extensive images with labels, since it need enough training and testing examples for the neural network. In this thesis Mnist, Nist and a Dog-Monkey datasets,will be used. In the follow section, these datasets will be introduced along with methods on how to prepare these datasets for the intended usage.

4.1 Mnist

The MNIST (Mixed National Institute of Standards and Technology) [19] database is a large dataset for handwritten digits. The training set is consist of numbers that were written by 250 peoples. 50% of them are high school students and 50% are from Census Bureau staff. The test set has the same proportion of handwritten digital data. The MNIST database contains 60,000 training images and 10,000 testing images.

The dataset consists of pairs, “handwritten digit image” and “label”. Digit ranges from 0 to 9, meaning 10 patterns in total. Figure 5 show an example of “handwritten digit image” and its “label” from MNIST dataset.

- handwritten digit image: This is gray scale image with size 28 x 28 pixel.
- label : This is actual digit number this handwritten digit image represents. It is either 0 to 9

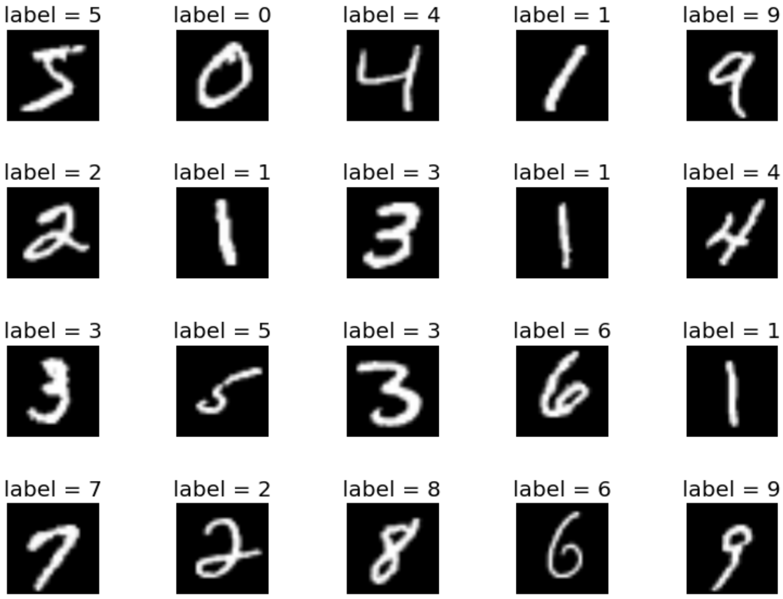


Figure 5: Several samples of “handwritten digit image” and its “label” from MNIST dataset.

The MNIST dataset is widely used for “classification”, “image recognition” task. This is considered as relatively simple task. It is also often used to compare algorithm performances in research.

4.2 NIST

Nist is the dataset of Handprinted Sample Forms published by 3600 writers.It contains 810,000 character images isolated from their forms. In this thesis, 36 classes including all digits 0-9 and uppercase characters A-Z are used. Each class has 2000 images.

4.3 Dog&Monkey Dataset

The third dataset is the so called Dog&Monkey Dataset. The dog dataset part are choosen from the stanford dogs dataset, which contains images of 120 breeds of dogs [20]. Each class has about 150 images. Another part dataset part is the 10-monkey-species [1] from kaggle. There are totally over 1000 monkey images.

In this thesis, 10 breeds of dogs and 10 breeds of monkeys are used for experiments. So there are totally 20 classes, the first 10 are dogs and the second 10 are monkeys.

[1] https://www.kaggle.com/slothkong/10-monkey-species

5 Experiments and Evaluation

In this chapter, the different transfer learning model architectures are introduced along with approaches, how patching works in practice. Furthermore, the experiments setup and the results of experiments are also presented and evaluated.

5.1 Preparation of the experiments

First, In this section five variants of data changes representing different types of concept drifting up to transfer of knowledge to unknown problem are introduced. Inputs are treated as stream data and occur batchwise. After the change point is reached, the five variants are used for data changing. All the datasets are experimented under the existing open source Codebase-Keras in the python environment.

Flip: Randomly flip the image in the horizontal or the vertical direction.In MNIST and NIST, the second half of the dataset, which consists of vertically and horizontally flipped digits are used.

Rotation: Randomly rotate the a certain angle of the image; Change the orientation of the image content. In our experiment, the second half of images are rotated with increasing degrees up to 180.

Appear: New data appear among old ones and thus, new concepts occur. In MNIST, 5-9 does not exist at beginning and only start to appear after change point (mixed with 0-4). In NIST, 0-9 does not exist at the beginning and only start to appear after change point (mixed with A-Z). In Dog-Monkey dataset,the monkey appear only after change point (mixed with dogs).

Transfer: One task is repurposed on a second task. In MNIST, 0-4 exist only in first half, while 5-9 only in second half. In NIST, 0-9 exist only in first half, while A-Z only in second half.In Dog-Monkey dataset,dogs exist only in the first half of images and monkeys exist only in the second half of images.

Remap: Transfers knowledge between domains with different feature spaces. In MNIST, only 0-4 exist in the first half. The inputs are replaced with 5-9 in the second half but the labels remain 0-4. In NIST, 0-9 exist only in the first half. The inputs are replaced with A-J in the second half but the labels remain 0-9.In Dog-Monkey dataset,dogs exist only in the first half of images. Monkies replace dogs in the second half but the labels remain 0-9.

5.2 Transfer learning Models

First of all, we need to introduce what transfer learning is, before we explain the transfer learning models are going to be used.Transfer learning make use of the knowledge gained while solving one problem and applying it to a different but related problem. The development of algorithms that facilitate transfer learning processes has become a goal of machine learning technicians as they strive to make machine learning as human-like as possible. As shown in Figure 6, transfer learning enables us to use pre-trained models from other people by making small changes. A pre-trained model is a model created by some one else to solve a similar problem. Instead of building a model from scratch to solve a similar problem, you use the model trained on another problem as a starting point. It means in transfer learning not only the complete model is transferred but also maybe only several first layers to be transfered.

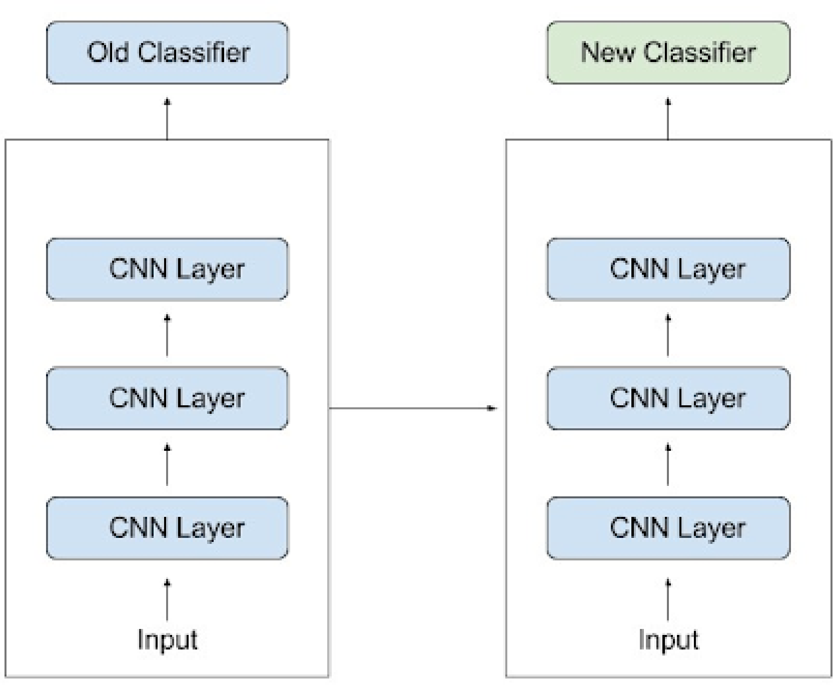


Figure 6: Basic transfer learning model

In transfer learning, we try to transfer as much knowledge as possible from the previous task, the model was trained on, to the new task at hand. This knowledge can be in various forms depending on the problem and the data.

There are two common approaches to do with the transfer leaning:

1.The pre-trained model is used as feature extractor by using early layers, which include generic informations. These extractive features can then be used as inputs for another learning model.

2.Fine-tuning pre-trained model. The last layer (softmax) of the pre-trained model is replaced with our own softmax layer that is relevant to the new problem. Some lower layers are frozen. Dataset-specific features are then be learned in the subsequent layers.

The second approach is mostly used in Computer Vision because the pre-trained model is usually trained on large dataset and have learned features relevant to our own classification problem.

5.2.1 simple CNN

The simple convolutional neural networks is straightforward and small. But it can still provide interesting and useful results. In our experiment a modification of the LeNet (LeCun Yann et al.1998) [21] architecture is used as simple CNN.

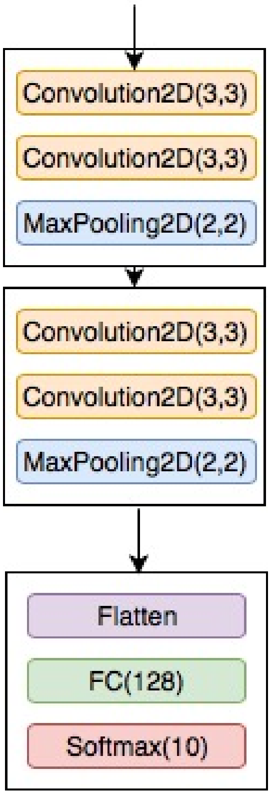


Figure 7: Simple modified CNN architecture

As shown in Figure 7, in our experiments, the following structure is being used: Conv2D(3,3) -> Conv2D(3,3) -> MaxPooling(2,2) -> Conv2D(3,3) -> Conv2D(3,3) -> MaxPooling(2,2) -> Dropout(0.25) -> Flatten -> FC(128) -> Dropout(0.5) -> Softmax(10)

where Conv2D(k): 2D convolution with k = kernel size; MaxPooling(k): Max pooling, k = size of the max pooling windows; Dropout(k): Dropout, k = dropout rate; FC(k): Fully Connected Layer, k = number of units; Softmax(k): Fully connected layer with softmax activation (k units).

The second two convolution layers have twice as many filters as the first 2 convolution layers. Since the MaxPooling layer reduces the image size after the first 2 convolution layers, the filters in the next Conv2D layers can be increased without increasing too much running time. This architecture of CNN is used for dataset MNIST.

5.2.2 Deeper CNN

For the dataset NIST, a deeper CNN model is used. This model is built in block-wise. Each block has the following structure: Conv2D(3,3) Conv2D(3,3) MaxPooling(2,2)

Totally 4 blocks are used. The whole structure of this CNN model is then: Block1 -> Block2 -> Block3 -> Block4 -> Dropout(0.25) -> Flatten -> FC (512) -> Dropout(0.5) -> Softmax(36)

For all Convolution layers and fully connected layers, the activation function "relu" is used.

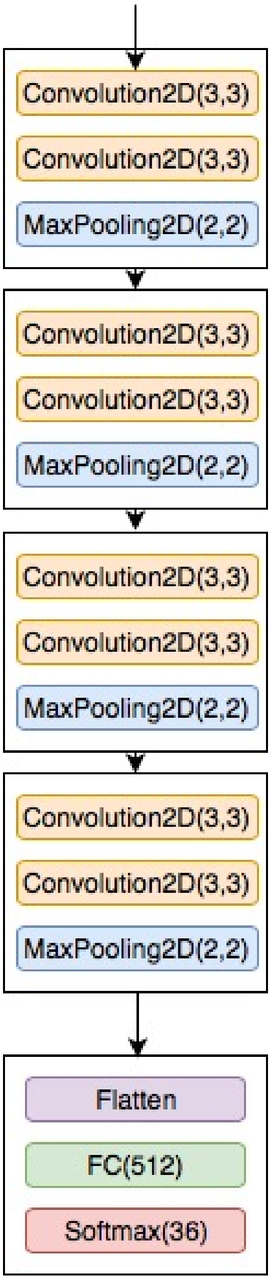


Figure 8: deeper CNN architecture with 4 blocks

In this thesis, a fully connected layer with 512 neurons are used in the patching-network. It can be described as following: FC(512)-Activation(relu)-softmax/sigmoid.

5.2.3 VGG16

The third transfer learning model used in our experiment is VGG16. VGG16 is a convolutional neural network architecture named after the Visual Geometry Group from Oxford[1].

In VGG16, the network uses only 3x3 convolutional layers stacked on top of each other in increasing depth. Max pooling is used to reduce image size. 16 stands for number of weight layers in the network[2].

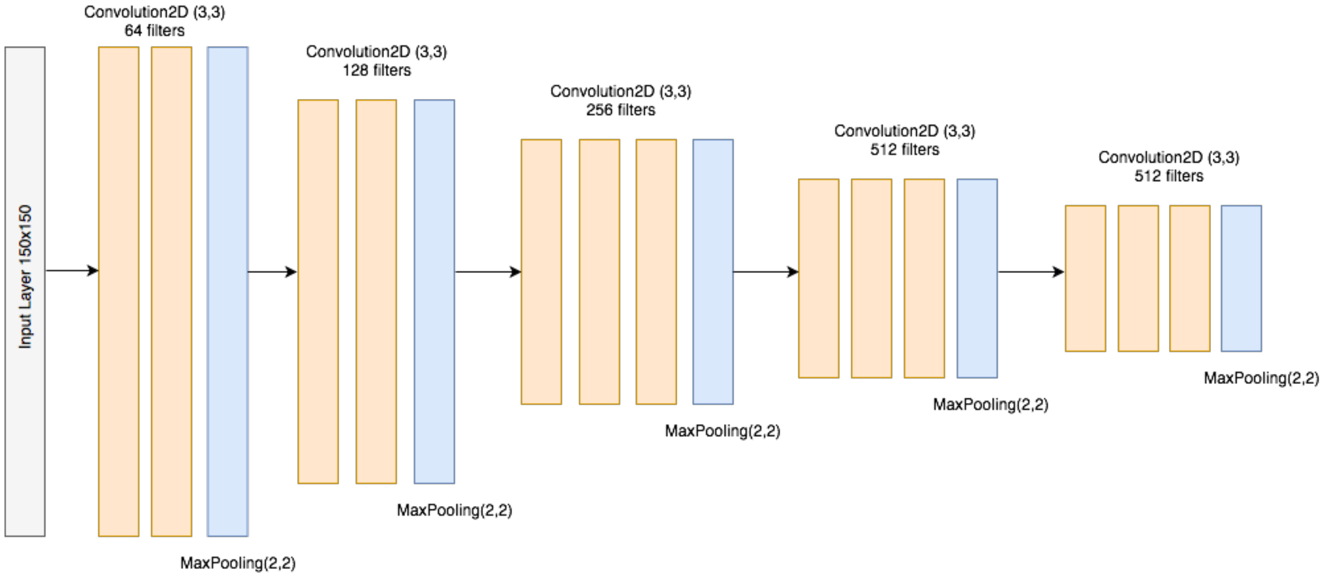


Figure 9: A visualization of the VGG architecture

In my experiment as shown in Figure 9, the structure of VGG16 without top layers can be described as following: Conv64 -> Conv64 -> MaxPooling -> Conv128 -> Conv128 -> MaxPooling -> Conv256 -> Conv256 -> Conv256 -> MaxPooling -> Conv512 -> Conv512 -> Conv512 -> MaxPooling -> Conv512 -> Conv512 -> Conv512 -> MaxPooling

The VGG16 network is used for Monkey-Dog dataset with 3 channels in our experiments.

5.3 Experiemnt setup

MNIST and NIST

In this thesis, datasets MNIST and NIST are used as inputs for the patching framework. Images of datasets are coming in batches. The first 20 batches are applied to train the base classifier C_0. After that, each batch of data is used for evaluation and training for the following classifiers:

E: determines if the base network C_0 misclassifies

P: patching network, which engages into an inner layer of the C_0 classifier. It will be applied if C_0 errs.

[1] http://www.robots.ox.ac.uk/ṽgg/research/very_deep/

[2] https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/

Base-Update: Classifier using base network, which is updated with all batches of data

Freezing: The layers of this classifier up to and including the engaged layer is frozen. The remaining layers are updated via backpropagation. It is initialized with weights of C_0 .

MS: Model selector which predicts that C_0 will misclassify and P will classify correctly. That means, the probability on correctness of C_0 and P is used to determine which classifier shall be use: if Pr(P) > Pr(C0), P shall be used.

All these 5 classifiers (E;P;Base-Update;Freezing;MS) are evaluated and trained after 20 batches. The classifiers "Base-Update" and "Freezing" are used as comparison to patching. The accuracy of each classifier is used for performance comparison.

To find the best inner layer of the base network that is engaged by patching network is one of the import targets of our experiments. So the performance of the patching at different engagement layers is recorded and compared.

The patching framework as described above is applied on the five data change variant: flip, rotate, appear, remap and transfer. More detail is described in chapter 2 and chapter 4.

Dataset	Init	CPs	Total	Chunks
Mnist				
flip	12K	#30K	60K	100
rotate	12K	#30K	60K	100
appear	12K	#12K	60K	100
remap	12K	#30K	60K	100
transfer	12K	#30K	60K	100

Table 1: Summary of the MNIST dataset used in the experiments

Dataset	Init	CPs	Total	Chunks
Nist				
flip	14.4K	#36K	72K	100
rotate	14.4K	#36K	72K	100
appear	21.6K	#21.6K	72K	100
remap	14.4K	#36K	72K	100
transfer	14.4K	#36K	72K	100

Table 2: Summary of the NIST dataset used in the experiments

The summary of the MNIST and NIST dataset for this thesis are shown in Table 1 and Table 2 For Dog&Monkey dataset, we must use data augmentation for better results, since we have to use data augmentation,the data change variant "flip" and "rotate"are not used. Thus, there are only three data change variants (appear, remap and transfer). In the Dog&Monkey dataset,three classifiers are trained: base C_0 , error detector E and patching network P. E and P are trained after 20 batches of data. More detail are described in chapter 2.

5.4 Evaluation

In this Section, I first define the performance metrics. Then, I propose the designs and results of various groups of experiments.

5.4.1 Evaluation Metrics

The general idea of the patching algorithm as described in Kauschke and Fürnkranz(2018) [9] is introduced. A typical run of patching consists of three phases:

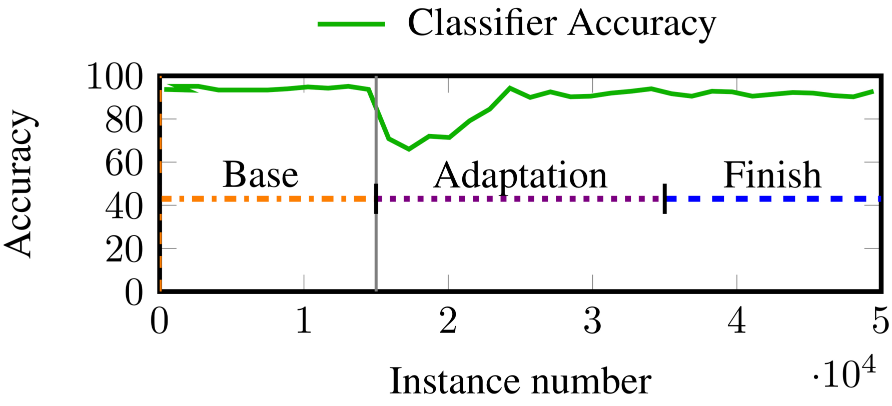


Figure 10: Example of a typical run of patching with data stream in batches

A typical run of patching with data stream in batches are shown in Figure 10, The vertical line marks the change point (drift of data).

In this thesis,the following metrics are being used:

1.Final accuracy: Accuracy measured in the final phase, which consists of the last 5 batches.

2.Average accuracy: Average accuracy after change point (adaption and finish phase).

3.Recovery Speed: Number of data instances that is required for a classifier to achieve 90% of the final accuracy.

4.Adaption Rank: Average rank of a classifier in the adaption phase.

5.Final Rank: Average rank of a classifier in the final phase.

6.Average Oscillations: Average accuracy fluctuation of the last 30 batches.

5.4.2 Evaluation Results

The results of 5 data change variants on dataset MNIST and NIST are evaluated in this section.For each of the 5variants(flip,rotate,appear,remap and transfer),the process of the training and evaluation of networks are shown.Accuracy changes against data stream in batches and performance fall and recovery in change points are presented in these figures.Engagement results are also shown in figures.For dataset MNIST we engage into each convolutional and max pooling layer,for dataset

NIST we engage into each Conv-block.These graphs show how different engagement layer affact the accuracy of networks.

MNIST - Flip:

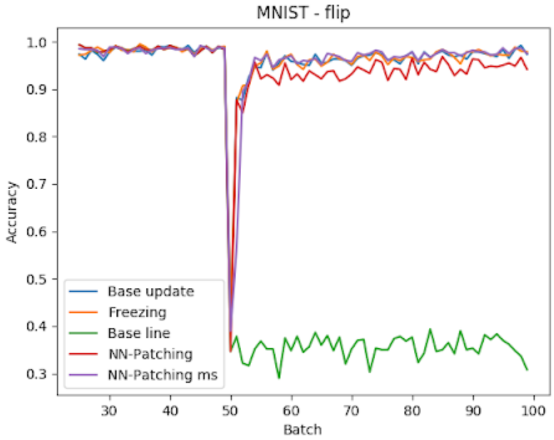


Figure 11: Result of data change variant "flip" on dataset MNIST

In this experiment,the input images are coming in batches.The base classifer is trained using the first 20 batches of data.The other 5 classifers start to be trained at batch number 20.At change point(batch number 50),the images are randomly flipped horizontally or vertically.The evaluated accuracy for 5 classifers are recorded and shown in Figure11.They are then trained continuously with totally 100 batches of a batch size 600.

All 6 Convolution and Maxpooling layers and the subsequent Dropout layer are engaged by patching network.The final accuracy of patching network is affected by selecting different engagement layer and are shown in Figure12 and Figure13.

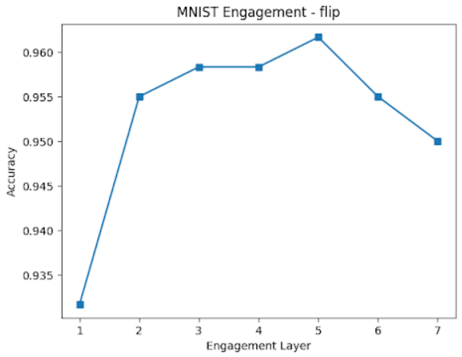


Figure 12: MNIST - Flip:Final accuracy with varying engagement layers in NN-Patching network

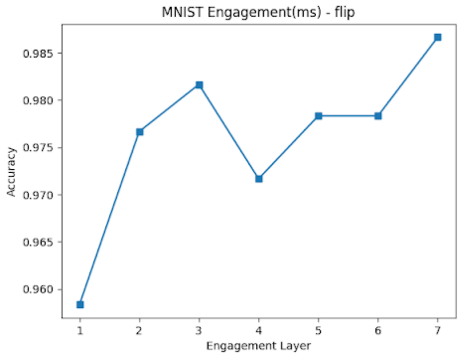


Figure 13: MNIST - Flip:Final accuracy with varying engagement layers in NN-Patching-MS network.

The left graph shows the result of the NN-Patching network, while the right one shows the result of NN-Patching-MS network.

MNIST - Rotate:

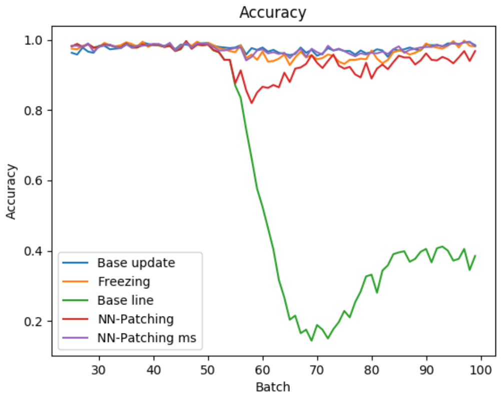


Figure 14: Result of data change variant “rotate” on dataset MNIST.

The MNIST-Rotate evaluated accuracy for 5 classifers are recorded and shown in Figure14

The final accuracy of patching network of MNIST-Rotate e is affected by selecting different engagementlayer and are shown in Figure15and Figure16

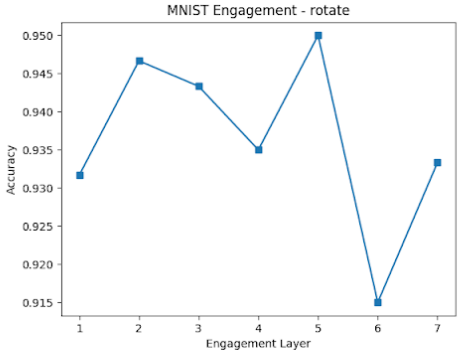


Figure 15: MNIST - Rotate:Final accuracy with varying engagement layers in NN-Patching network

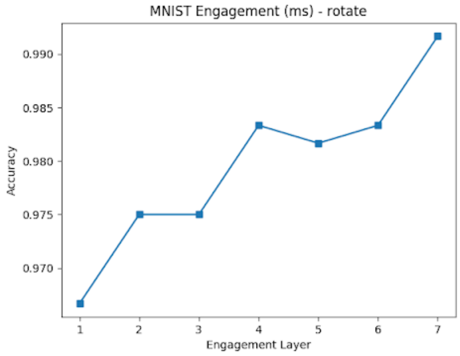


Figure 16: MNIST - Rotate:Final accuracy with varying engagement layers in NN-Patching-MS network.

The left graph shows the result of the NN-Patching network, while the right one shows the resultof NN-Patching-MS network.

MNIST - Appear:

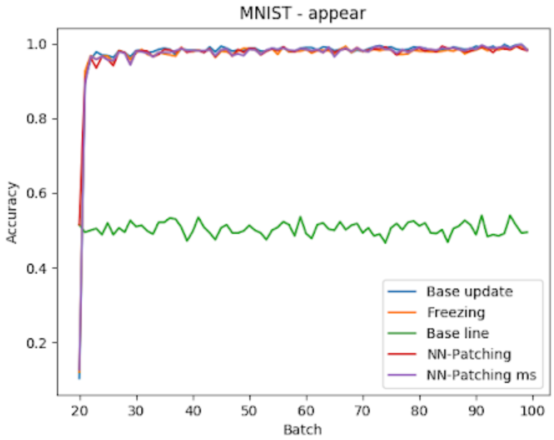


Figure 17: Result of data change variant “Appear”on dataset MNIST.

The MNIST-Appear evaluated accuracy for 5 classifers are recorded and shown in Figure17 The final accuracy of patching network of MNIST-Appear is affected by selecting different engagementlayer and are shown in Figure18and Figure19

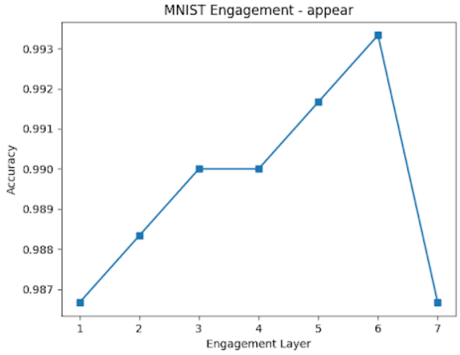


Figure 18: MNIST-Appear:Final accuracy with varying engagement layers in NN-Patching network

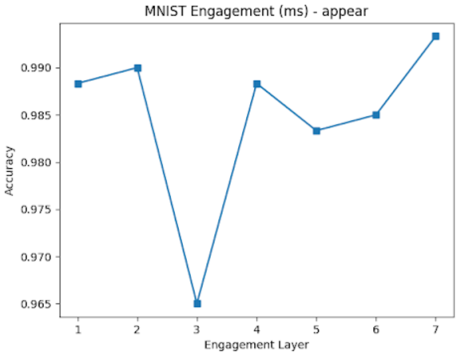


Figure 19: MNIST-Appear:Final accuracy with varying engagement layers in NN-Patching-MS network.

The left graph shows the result of the NN-Patching network, while the right one shows the resultof NN-Patching-MS network.

MNIST - Remap:

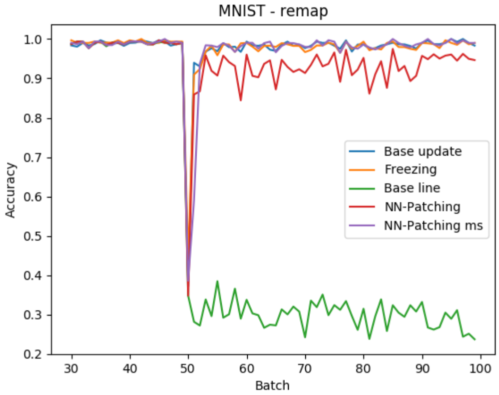


Figure 20: Result of data change variant “Remap”on dataset MNIST.

The MNIST-Appear evaluated accuracy for 5 classifers are recorded and shown in Figure20

The final accuracy of patching network of MNIST-Appear is affected by selecting different engagementlayer and are shown in Figure21and Figure22

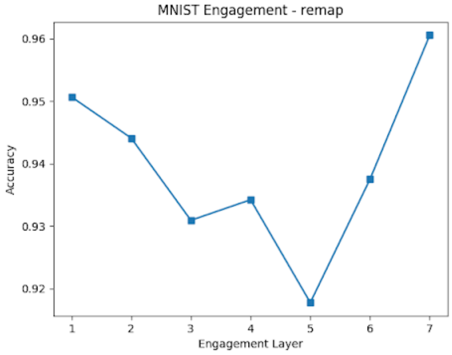


Figure 21: MNIST-Remap:Final accuracy with varying engagement layers in NN-Patching network

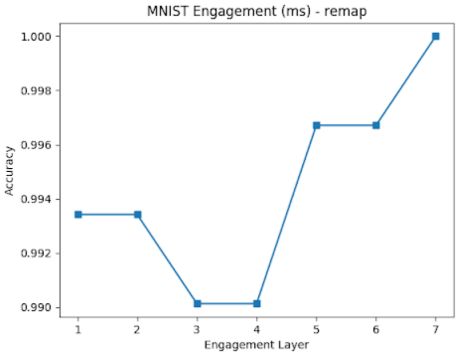


Figure 22: MNIST-Remap:Final accuracy with varying engagement layers in NN-Patching-MS network.

The left graph shows the result of the NN-Patching network, while the right one shows the resultof NN-Patching-MS network.

MNIST - Transfer:

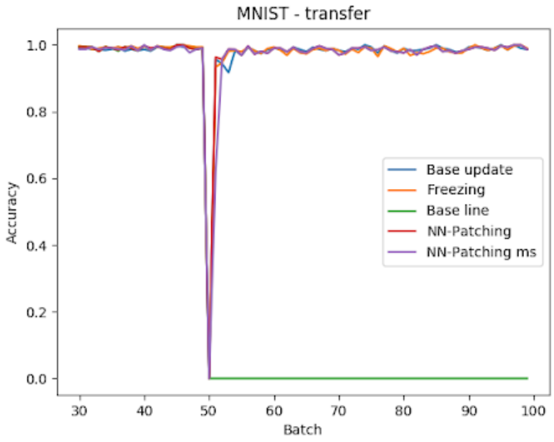


Figure 23: Result of data change variant “Transfer”on dataset MNIST.

The MNIST-Transfer evaluated accuracy for 5 classifers are recorded and shown in Figure23

The final accuracy of patching network of MNIST-Appear is affected by selecting different engagementlayer and are shown in Figure24and Figure25

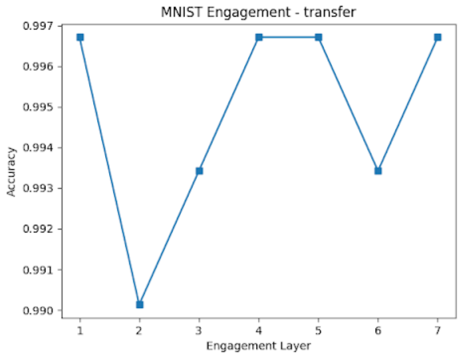


Figure 24: MNIST-Transfer:Final accuracy with varying engagement layers in NN-Patching network

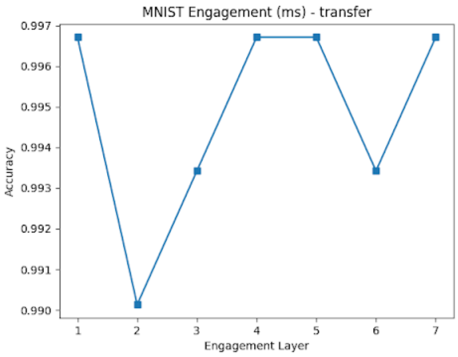


Figure 25: MNIST-Transfer:Final accuracy with varying engagement layers in NN-Patching-MS network.

The left graph shows the result of the NN-Patching network, while the right one shows the resultof NN-Patching-MS network.

NIST - Flip:

In this experiment,the input images are coming in batches.The base classifer is trained using the first 20 batches of data.The other 5 classifers start to be trained at batch number 20.At change point(batch number 50),the images are randomly flipped horizontally or vertically.The evaluated accuracy for 5 classifers are recorded and shown in Figure26.They are then trained continuously with totally 100 batches of a batch size 720.

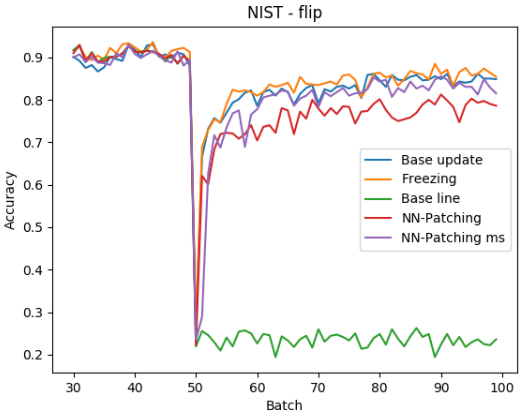


Figure 26: Result of data change variant “Flip”on dataset NIST.

All 6 Convolution and Maxpooling layers and the subsequent Dropout layer are engaged by patching network.The final accuracy of patching network is affected by selecting different engagement layer and are shown in Figure27 and Figure28.

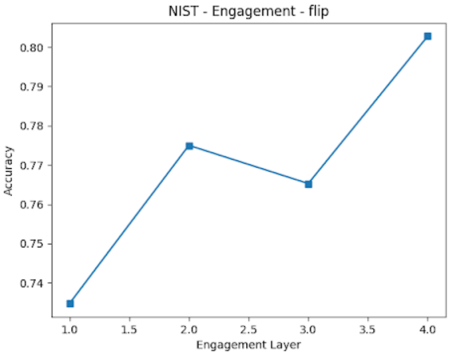


Figure 27: NIST-Flip:Final accuracy with varying engagement layers in NN-Patching network

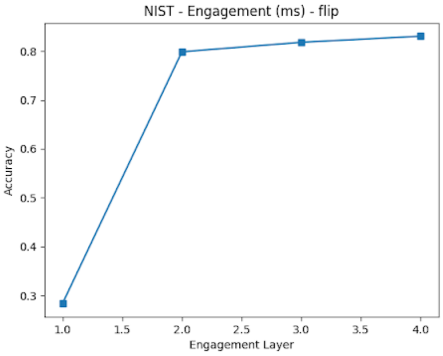


Figure 28: NIST-Flip:Final accuracy with varying engagement layers in NN-Patching-MS network.

The left graph shows the result of the NN-Patching network, while the right one shows the result of NN-Patching-MS network.

NIST - Rotate:

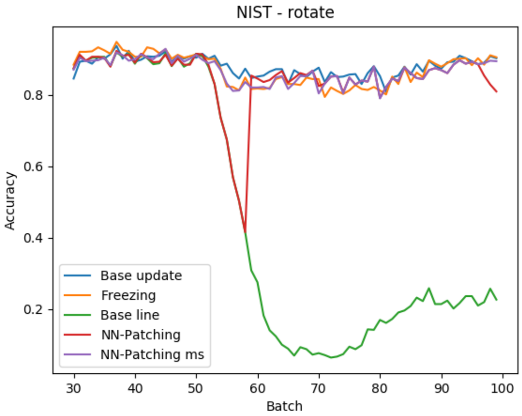


Figure 29: Result of data change variant “Rotate”on dataset NIST.

The NIST-Rotate evaluated accuracy for 5 classifers are recorded and shown in Figure29

The final accuracy of patching network of NIST-Rotate is affected by selecting different engagementlayer and are shown in Figure30and Figure31

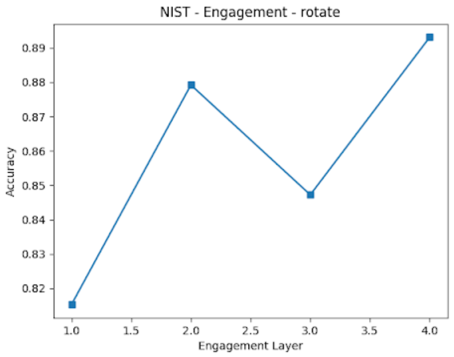


Figure 30: NIST-Rotate:Final accuracy with varying engagement layers in NN-Patching network

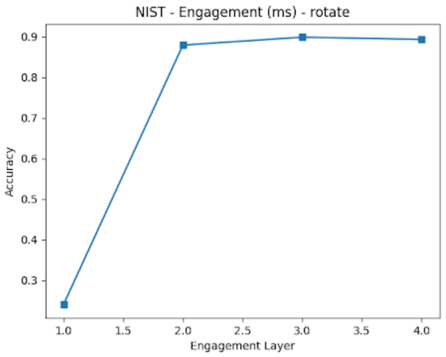


Figure 31: NIST-Rotate:Final accuracy with varying engagement layers in NN-Patching-MS network.

The left graph shows the result of the NN-Patching network, while the right one shows the resultof NN-Patching-MS network.

NIST - Appear:

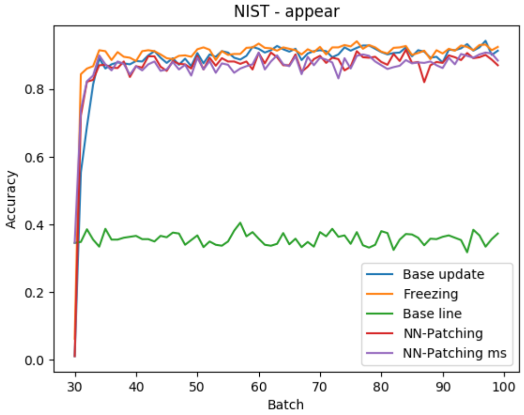
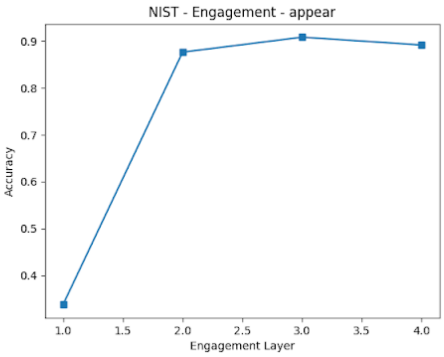


Figure 32: Result of data change variant “Appear”on dataset NIST.

The NIST-Appear evaluated accuracy for 5 classifers are recorded and shown in Figure32

The final accuracy of patching network of NIST-Appear is affected by selecting different engagementlayer and are shown in Figure33and Figure34



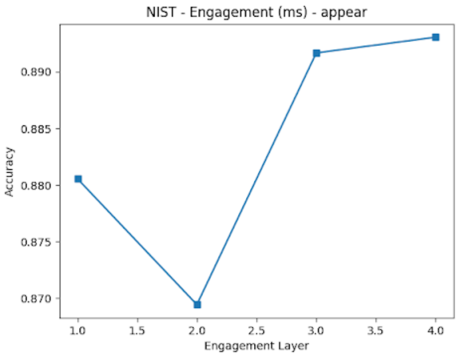


Figure 33: NIST-Appear:Final accuracy with varying engagement layers in NN-Patching network

Figure 34: NIST-Appear:Final accuracy with varying engagement layers in NN-Patching-MS network.

The left graph shows the result of the NN-Patching network, while the right one shows the resultof NN-Patching-MS network.

NIST - Remap:

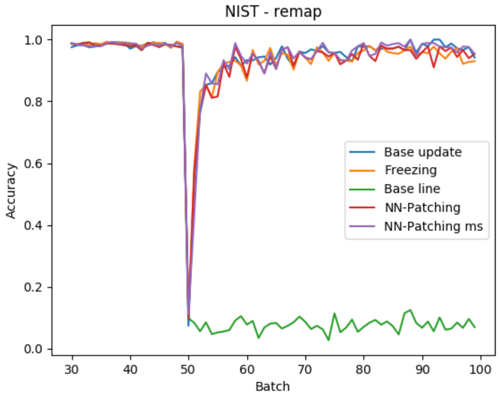


Figure 35: Result of data change variant "Remap"on dataset NIST.

The NIST-Remap evaluated accuracy for 5 classifers are recorded and shown in Figure35

The final accuracy of patching network of NIST-Remap is affected by selecting different engagementlayer and are shown in Figure36and Figure37

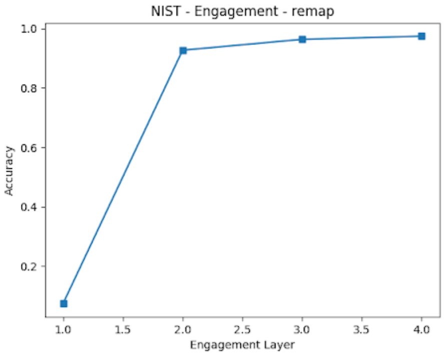


Figure 36: NIST-Remap:Final accuracy with varying engagement layers in NN-Patching network

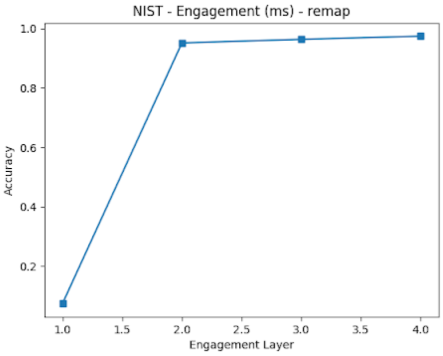


Figure 37: NIST-Remap:Final accuracy with varying engagement layers in NN-Patching-MS network.

The left graph shows the result of the NN-Patching network, while the right one shows the resultof NN-Patching-MS network.

NIST - Transfer:

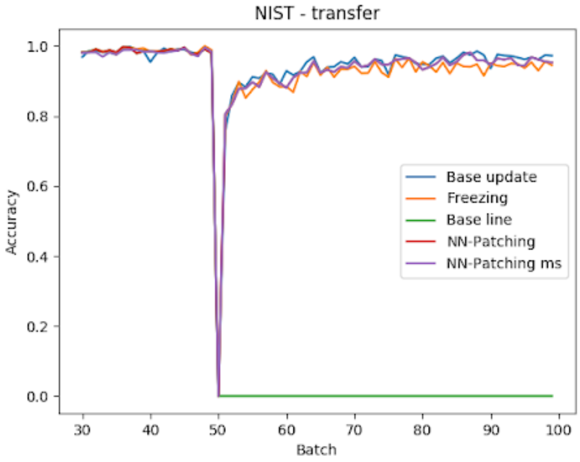


Figure 38: Result of data change variant “Transfer”on dataset NIST.

The NIST-Transfer evaluated accuracy for 5 classifers are recorded and shown in Figure38

The final accuracy of patching network of NIST-Transfer is affected by selecting different engagementlayer and are shown in Figure39and Figure40

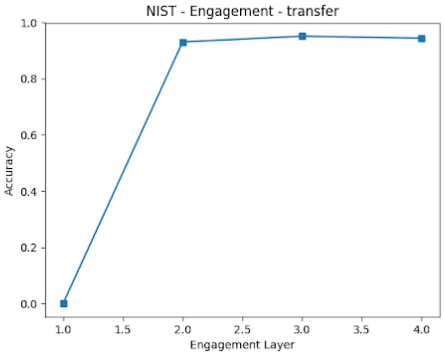


Figure 39: NIST-Transfer:Final accuracy with varying engagement layers in NN-Patching network

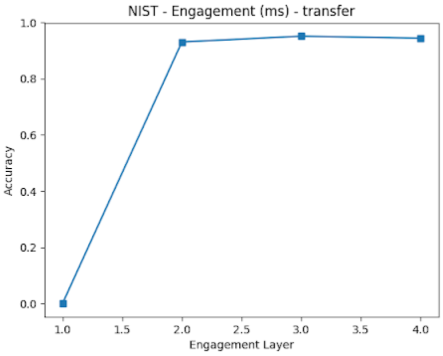


Figure 40: NIST-Transfer:Final accuracy with varying engagement layers in NN-Patching-MS network.

The left graph shows the result of the NN-Patching network, while the right one shows the resultof NN-Patching-MS network.

5.4.3 Evaluation analysis

In the Fitures above, we can see an adaptation phase, in which changed data are coming and the classifiers need some batches for recovery. The variant “rotation” is an exception. It shows no apparent adaptation phase.The reason is that the input images are rotated gradually and continuously. So the classifiers have to do the adaptation for all the coming batches of data. Since the change is not abrupt, there is also no abrupt drop on accuracy.

Performance of classifiers The metrics measured in experiments can be summarized in the following tables:

Classifier	Final Acc.	Avg. Acc.	Recovery Speed	Adap. Rank	Final Rank	Avg. Oscillation
MNIST - flip						
BaseLine	34.5	35.7	-	4.8	5.0	1.723
NN-Patching	34.5	35.7	-	4.8	5.0	1.723
NN-Patching ms	34.5	35.7	-	4.8	5.0	1.723
Freezing	34.5	35.7	-	4.8	5.0	1.723
Base-Update	34.5	35.7	-	4.8	5.0	1.723

Table 3: Measured performance of classifiers for MNIST - flip

Classifier	Final Acc.	Avg. Acc.	Recovery Speed	Adap. Rank	Final Rank	Avg. Oscillation
MNIST - rotate						
BaseLine	34.5	35.7	-	4.8	5.0	1.723
NN-Patching	34.5	35.7	-	4.8	5.0	1.723
NN-Patching ms	34.5	35.7	-	4.8	5.0	1.723
Freezing	34.5	35.7	-	4.8	5.0	1.723
Base-Update	34.5	35.7	-	4.8	5.0	1.723

Table 4: Measured performance of classifiers for MNIST - rotate

Classifier	Final Acc.	Avg. Acc.	Recovery Speed	Adap. Rank	Final Rank	Avg. Oscillation
MNIST - appear						
BaseLine	34.5	35.7	-	4.8	5.0	1.723
NN-Patching	34.5	35.7	-	4.8	5.0	1.723
NN-Patching ms	34.5	35.7	-	4.8	5.0	1.723
Freezing	34.5	35.7	-	4.8	5.0	1.723
Base-Update	34.5	35.7	-	4.8	5.0	1.723

Table 5: Measured performance of classifiers for MNIST - appear

Classifier	Final Acc.	Avg. Acc.	Recovery Speed	Adap. Rank	Final Rank	Avg. Oscillation
MNIST - remap						
BaseLine	34.5	35.7	-	4.8	5.0	1.723
NN-Patching	34.5	35.7	-	4.8	5.0	1.723
NN-Patching ms	34.5	35.7	-	4.8	5.0	1.723
Freezing	34.5	35.7	-	4.8	5.0	1.723
Base-Update	34.5	35.7	-	4.8	5.0	1.723

Table 6: Measured performance of classifiers for MNIST - remap

Classifier	Final Acc.	Avg. Acc.	Recovery Speed	Adap. Rank	Final Rank	Avg. Oscillation
MNIST - transfer						
BaseLine	34.5	35.7	-	4.8	5.0	1.723
NN-Patching	34.5	35.7	-	4.8	5.0	1.723
NN-Patching ms	34.5	35.7	-	4.8	5.0	1.723
Freezing	34.5	35.7	-	4.8	5.0	1.723
Base-Update	34.5	35.7	-	4.8	5.0	1.723

Table 7: Measured performance of classifiers for MNIST - transfer

Table # Result for dataset NIST

Classifier	Final Acc.	Avg. Acc.	Recovery Speed	Adap. Rank	Final Rank	Avg. Oscillation
Nist-flip						
BaseLine	34.5	35.7	-	4.8	5.0	1.723
NN-Patching	34.5	35.7	-	4.8	5.0	1.723
NN-Patching ms	34.5	35.7	-	4.8	5.0	1.723
Freezing	34.5	35.7	-	4.8	5.0	1.723
Base-Update	34.5	35.7	-	4.8	5.0	1.723

Table 8: Measured performance of classifiers for NIST - flip

Classifier	Final Acc.	Avg. Acc.	Recovery Speed	Adap. Rank	Final Rank	Avg. Oscillation
Nist-rotate						
BaseLine	34.5	35.7	-	4.8	5.0	1.723
NN-Patching	34.5	35.7	-	4.8	5.0	1.723
NN-Patching ms	34.5	35.7	-	4.8	5.0	1.723
Freezing	34.5	35.7	-	4.8	5.0	1.723
Base-Update	34.5	35.7	-	4.8	5.0	1.723

Table 9: Measured performance of classifiers for NIST - rotate

Classifier	Final Acc.	Avg. Acc.	Recovery Speed	Adap. Rank	Final Rank	Avg. Oscillation
Nist-appear						
BaseLine	34.5	35.7	-	4.8	5.0	1.723
NN-Patching	34.5	35.7	-	4.8	5.0	1.723
NN-Patching ms	34.5	35.7	-	4.8	5.0	1.723
Freezing	34.5	35.7	-	4.8	5.0	1.723
Base-Update	34.5	35.7	-	4.8	5.0	1.723

Table 10: Measured performance of classifiers for NIST - appear

Classifier	Final Acc.	Avg. Acc.	Recovery Speed	Adap. Rank	Final Rank	Avg. Oscillation
Nist-remap						
BaseLine	34.5	35.7	-	4.8	5.0	1.723
NN-Patching	34.5	35.7	-	4.8	5.0	1.723
NN-Patching ms	34.5	35.7	-	4.8	5.0	1.723
Freezing	34.5	35.7	-	4.8	5.0	1.723
Base-Update	34.5	35.7	-	4.8	5.0	1.723

Table 11: Measured performance of classifiers for NIST - remap

Classifier	Final Acc.	Avg. Acc.	Recovery Speed	Adap. Rank	Final Rank	Avg. Oscillation
Nist-transfer						
BaseLine	34.5	35.7	-	4.8	5.0	1.723
NN-Patching	34.5	35.7	-	4.8	5.0	1.723
NN-Patching ms	34.5	35.7	-	4.8	5.0	1.723
Freezing	34.5	35.7	-	4.8	5.0	1.723
Base-Update	34.5	35.7	-	4.8	5.0	1.723

Table 12: Measured performance of classifiers for NIST - transfer

In the experiments, we use the same model architecture for base network (C_0) and for Base-Update. The patching networks also use the C_0 to do the engagement.

As shown in the tables above, the average accuracy and final accuracy for all classifiers are almost the same. There is also no great difference on recovery speed. The Base-Update is trained on all its weights, which is a great benefit, especially when the input images are not very complex and there are enough data to train. For the Freezing-model, it is initialized with the weights of C_0. This can significantly improve the performance of the classifier.

The two NN-Patching networks are trained with random initialization. But they can also achieve good accuracy, in some cases actually the best. The advantage of patching is the ability to use both C_0 and the patch P depending on error decider or model selector.

The NN-Patching-ms with model selector achieves better accuracy in almost all experiments over NN-Patching. It has also smaller avg.oscillations for dataset MNIST as shown in table 7. The avg.oscillations are affected normally by batch size and learning rate. But since this two factors are same in our experiments, the difference of fluctuation here shows that the model selector diverts more data to the patch classifier P, which is advantageous for changed data.Later in this section,the optimization of patching network for NIST will be explaind.

Engagement

In a convolution neural network, the first layers represent general features like edges and shapes. Because these features are general, they can be transferred to other tasks. So it is benefit for concept drifting and transfer learning, that the first layer remains. In our experiments, the engagement begins after the first convolution layer for MNIST and after the first Conv2D-Block in NIST.

In general, the accuracy of classifiers increases when we engage into layers towards end of networks. These layers represent specific informations, which are useful for classifiers to learn new concepts.

For MNIST, it is interesting to see that the NN-patching network with model selector shows more regular behavior, i.e. engagement towards the end of network has better accuracy and engagement at the beginning of the network has poor performance. While networks using Error Detector shows more irregular behavior. This is because the model selector diverts more data to patching network, while the error detector fluctuate between C_0 and patching. The graph of the drift variant “transfer” for MNIST shows a strange form. But the absolute difference of accuracy is only 0.007. We can consider that the engagement in different layers has the same effect. A patching classifier with fewer layer can already do good job for this relative simple dataset.

Another phenomenon that can be observed for NIST is, the engagement after the second block makes a jump in accuracy (with the exception “Appear”). This indicates that the first 2 blocks are powerful enough for the classifiers to learn new things. This certainly depends on datasets and architecture of networks.

According to Yosinski et al. (2014)[13], fragile co-adaptation can occur when splitting networks. That means, neurons on neighboring layers co-adapt during training. If we engage between them, the neurons cannot be rediscovered. We can see this phenomenon in e.g. MNIST-Rotate, MNIST-Remap, MNIST-Appear (NN-Patching-ms), etc. In dataset NIST, we engage layers block-wise. It reduces the risk of co-adaptation, because the probability of co-adaptation between two blocks is small. But one exception can be observed in “appear” in network with model selector. Maybe there is co-adaptation between neighbor layers at boundary in this special case.

Conclusion

For datasets with simple images like MNIST and not very deep network, one can engage into each layer and measure the performance of the network to find an optimal engagement. For middle datasets like NIST and maybe also for big datasets ,one can build the model in blocks. Each block is then engaged and the performance is measured. This also reduces the risk of co-adaptation which causes the performance drop. In general, engaging into the last Conv2D/Maxpooling layer gives the best performance. But for dataset NIST, engaging into the second block gives already very good results.

5.4.4 Optimization of Patching Network (NIST)

Some experiments are made for the dataset NIST for optimization of the Patching network. I find out the following 3 ways to increase the performance of the patching network:

i) add an additional dropout layer

Dropout is a regurization technique to reduce overfitting and to avoid co-adaptation on training data. It ignores certain set of units chosen at random. The neurons develop co-dependency among each other during training, which can lead to overfitting of training data.

ii) add a normalization layer

As the network learns, the parameters (weights) are updated. Thus, the distribution of outputs of a specific layer changes (internal covariate shift). This forces higher layers to adapt to that drift and leads to slow down leaning. The layer is normalised in such a way, that it has a mean output activation of zero and a standard deviation of one. The normalization layer stabilizes the network and makes the data comparable across features.

iii) use the optimizer SGD with a learning rate of 0.001 for the classifier error detector and the classifier model selector.

Learning rate is a hyper-parameter which controls how much the weights of network are adjusted with respect to loss gradient. The lower the learning rate, the slower the network is learning. But big value can cause missing local minima. The experiments show that the value of 0.001 can lead to good results.

The structure of the patching network is then: Dropout(0.5) - Flatten - FC(512) - BatchNormalization - Activation(relu) - Dropout(0.25) - Softmax/Sigmoid

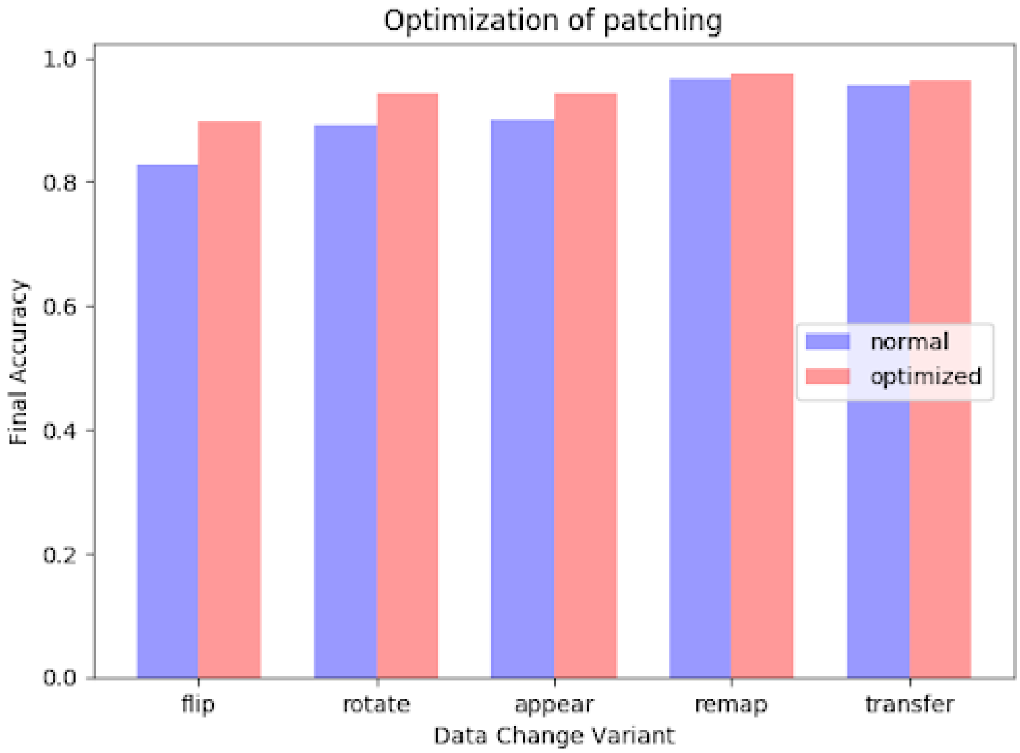


Figure 41: Increasement of final accuray by optimization in all 5 drift variants

The variants with lower accuracy benefit more from the optimization.

5.4.5 DataSet Dog-Monkey with VGG16

In this thesis the patch algorithm is also applies on the dataset Dog-Monkey, whose images have more pixels (150x150) and 3 channels. There are 10 categories of dogs 10 categories of monkeys, total 20 classes.

Because the dataset is relative complex, we use the pre-trained VGG16 network. All classifer used here are not trained from scratch.

Since we have relative small amount of images (with only about 1000 monkey images), the image augmentation is essential for reasonable results. Thus the data change variants “flip” and “rotate” have to be dropped out.

Base-Update as comparing algorithm cannot be applied due to very long running time, if all parameters are trained.

Patching using fine-tuning technology

The VGG16 network consists of 5 Conv-blocks. We first use the fine-tuning technique for the patching classifiers. That means, the first 4 Conv-blocks are frozen, only the last block is free (initialized with pre-trained weights). We use the keras VGG16 network without top and add our own fully connected layers and output layer.

Figure42,Figure43and Figure44 show the final accuracy of the patching network as the 100batches of data are coming with 3types of data change:appear,remap and transfer.The base classifer is trained using data of the first 20 batches,while the patching networks are trained after batch #20.

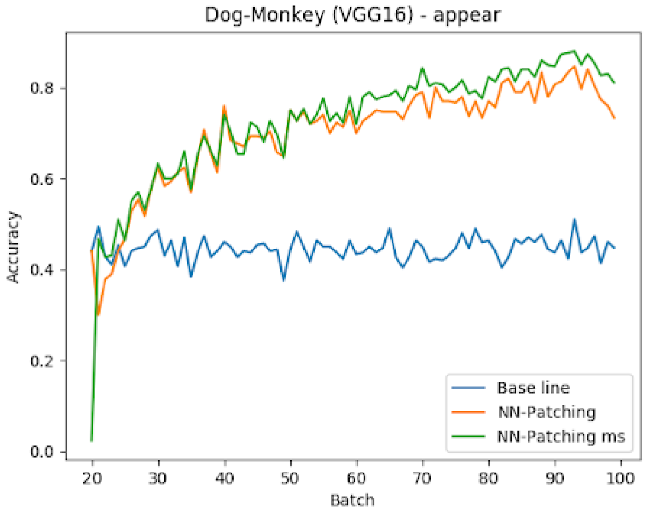


Figure 42: Result of data change variant “appear” on dataset Dog-Monkey

For ”appear”,the change point begins at batch#20,i.e.after batch#20,monkeys start to appear,mixed with dogs.

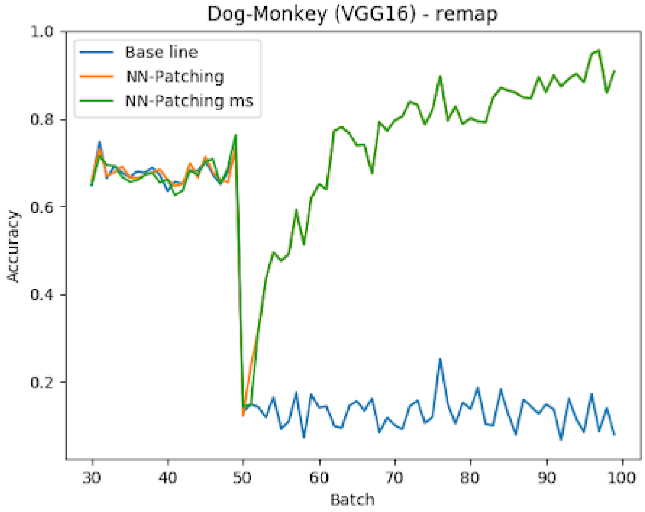


Figure 43: result of data change variant “Remap”on dataset Dog-Monkey

For ”remap”,only dogs exist for the first 50 batches.After 50 batches only monkeys exist,but the labels remain unchanged.

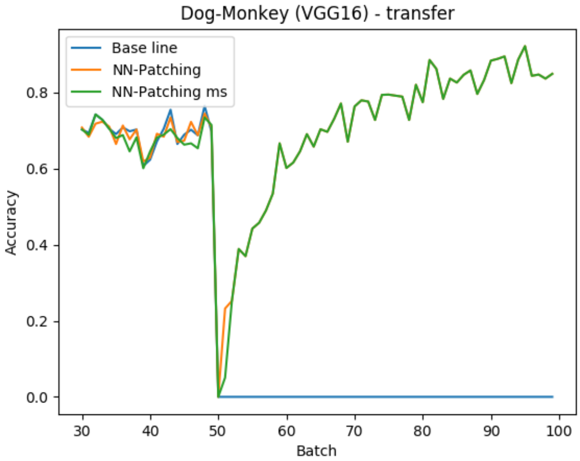


Figure 44: result of data change variant “transfer”on dataset Dog-Monkey

For”transfer”,only dogs exist in the first 50 batches and only monkeys exist in the second 50 batches.The task of classifiers are transferred from recognising dogs to recognising monkeys.

The accuracy of over 80% shows a good performance of the patching algorithm. With the error detector and model selector, we can achieve also relative good results with fewer data.

Engagement

Some experiments of engagement are done for the dataset Dog-Monkey.

Using only fully connected layers as patching network does not give good results. We added a convolution layer and a max-pooling layer in the patching network. The result is also not good. We then tried the following structure as our patching network:

Conv2D(3,3) - Conv2D(3, 3) - Conv2D(3, 3) - Maxpooling(2, 2) - FC(256) - Dropout(0.5) - Softmax

There is a limitation to use this structure. The last block of VGG16 cannot be engaged, because the number of kernel size (4) is too small for further convolution layers.So we engaged into block 1 to 4 of VGG16.

In Figure45,Figure46 and Figure47 show us the results of the engagement for ”apepar”,”remap” and ”transfer”.The final accuracy of patching network with model selector against engaging layer is presented.The last data point in the figures is the final accuracy using fine-tuning algorithm,it is also used as reference for the purpose of comparison.

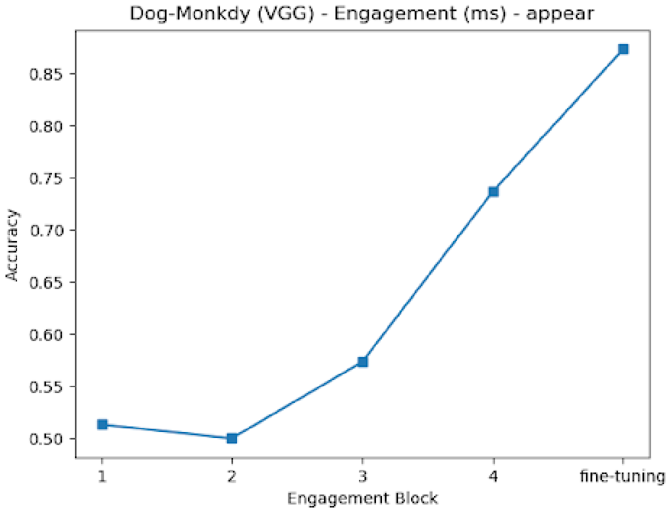


Figure 45: Final accuracy with different engagement blocks in CNN for “appear”

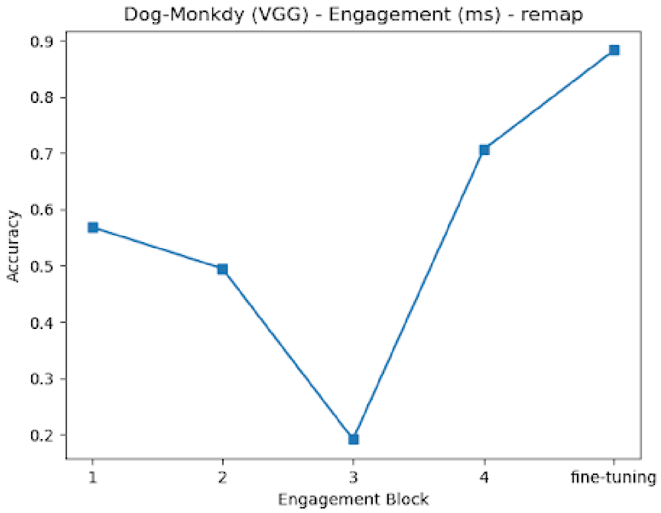


Figure 46: Final accuracy with different engagement blocks in CNN for“Remap”

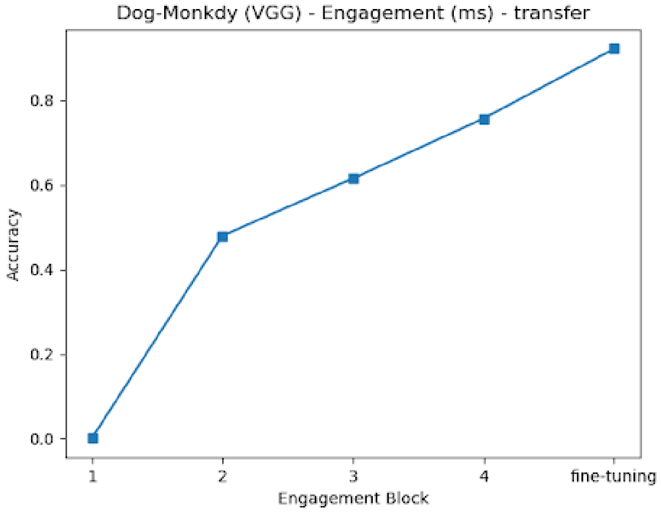


Figure 47: Final accuracy with different engagement blocks in CNN for"transfer"

As we can see in the graphs, the engagements in the layers towards end of the network has better performance, since the special features are important to distinguish monkeys from dogs. The fine-tuning method can achieve best results. This can be explained with the fact, that fine-tuning is using pre-trained weights, while the patching network is trained from scratch.

It is also interesting to see that the result of "appear" has a performance drop,when block 3 is engaged.These performance drops can be observed in all thress datasets we are using.It can be considered as "fragile co-adaptation" mentioned by Yosinski et al.(2014)[13]. So it is important to identify and avoid these "bad" layers,when engagement in inner layers.

We can conclude that for CNN, engagement at the end of network can lead to better results. For concept drift or transfer learning, the special features, which are extracted at the last layers, are essential for handling changed data and thus for the performance of networks.

5.4.6 CNN filter number

Experiments are made in MNIST-dataset to find the relationship of patching performance and the number of CNN filters used.

We use 16, 32, 64 and 128 filters for the patching network and the accuracy thereof is examined. The results show that more filters does not always mean more benefit for the performance or the benefit is too small. For example for the variant “rotate” the relationship of network accuracy and the number of filters is shown in the following graph:

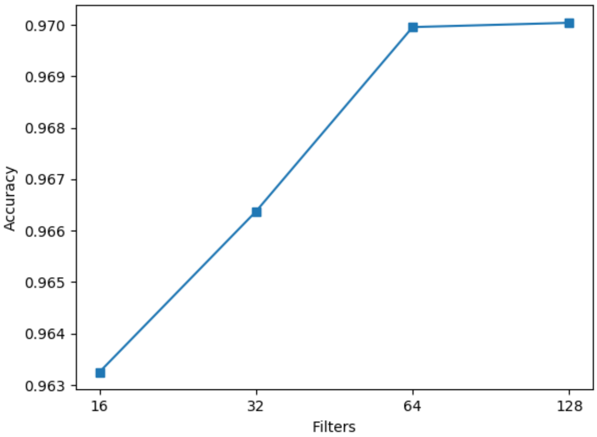


Figure 48: Relationship of patching network accuracy and number of filters used

Using 128 filters improves hardly the accuracy. In some cases, the accuracy is even lower with 128 filters.

If we look at the running time for different number of filters applied in CNN network:

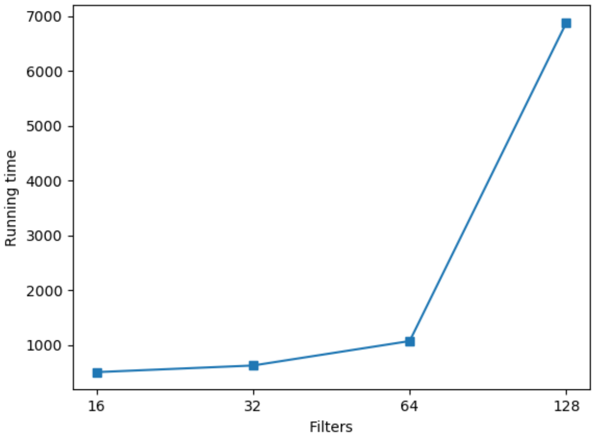


Figure 49: Relationship of running time and the number of filters

128 Filters consume 7-fold time as 64 filters. It is obviously not cost-efficient to use 128 filters. 64 filters for the patching network is good enough with rational resource consumption.

6 Conclusion and Future work

6.1 Conclusion

The number of hidden units depends from the kind of neural network used, the complexity of the learning task, and up to a certain extent the number of training examples.

The CNN is very suitable for continuous training on data stream, which are coming batchwise. The layered architecture of CNN makes it possible for engagement and reuse of pretrained models. Our experiments show that number of filters of CNN can affect the performance of the patching network. But the relation is not linear. For simple data, too many filters do not leads to better performance. The more filters, the more resources are requested. So one must find the balance between performance and costs, especially for complex datasets.

Five data change types (flip, rotate, appear, remap and transfer) for datasets MNIST and NIST and three data change types (appear remap and transfer) for dataset Dog-Monkey are applied on the patching network. The experiments show that the patching algorithm can adapt the data change quickly and well, using metrics we defined (final accuracy, recovery speed, etc.). Besides the error detector, a model selector is also used to determine if the original network misclassifies. This model selector is not just trained with changed data. It compares the probability on correct prediction of base network and patching network. The model selector shows a better performance over the error detector in most cases.

Engagement is also done on datasets. For MNIST every convolution/maxpooling layer is engaged. For NIST every Conv-block is engaged. The patching network has a hidden layer: fully connected layer with 512 units. It shows that generally engagement on layers towards end of network can achieve better performance. Since the inner layers of CNN represents the general information and the last layers the special features, the last layers are necessary for extraction of new features of changed data.

Optimizations are conducted for classifiers on dataset NIST. It shows that dropout, normalization and the selection of optimizer and its learning rate can affect the performance of classifiers. The experiments show a relative greater improvement on flip, rotate and appear, whose accuracy is lower before optimization. There are certainly many other possibilities to optimize the patching network, which can be done in the future work. An interesting phenomenon observed in our experiments is the so called ”fragile co-adaptation”. Two layers are correlated. If an engagement is done between them, the correlation is destroyed. This leads to performance drop. An engagement at these positions shall be avoided.

For dataset Dog-Monkey, VGG16 and the technique ”fine-tuning” is applied in the patching network. All Conv-blocks till the last one of VGG16 are frozen. Only the last block is trained. The accuracy of over 80% is quite good with fewer data (e.g. only about 1000 monkeys). This shows that the patching algorithm works also for complex datasets.

Experiments on engagement is also done for this dataset. Instead of just use fully connected layer, a Conv-block is also applied in the patch network. We can see the improvement of accuracy if the engagement moves towards the end of the network. This shows the same result as for other datasets. The experiments show that the fine-tuning method achieves better accuracy than engagement. The fine-tuning method has the advantage that it uses the pre-trained weights of VGG16, while the patch network is trained from scratch.

6.2 Future Work

This thesis mainly focused on network patching algorithm using streaming data. Experiments on engagements are also conducted. Several datasets are applied for our experiments. For the further exploring of patching and engagement we can 1) use more datasets. Besides MNIST, NIST and Dog-Monkey datasets, other datasets can be applied to research more aspects of patching performance and data properties (black-white vs. color, simple and complex images, ...)

2) use more CNN types. Besides VGG16, Resnet, Xception, Inception, MobileNet, etc. are also interesting to be applied to observe probably different behaviour of the patching network

3) find some methods to avoid ”fragile co-adaptation”. Research on relationship of these co-adaptation and data changed type (flip, appear, ...) is also interesting to explore

4) find more way to optimize the patching networks. Besides adding some dropout and normalization layers and modifying optimizer and learning rate we did, other hyperparameters and methods can be explored

5) build patch network differently. Other than one fully connected hidden layer or one Conv-block plus fully connected layer we used for Dog-Monkey dataset, other structures can be used and experimented on different datasets

6) build a website to receive data for evaluation and training. In this thesis the data stream in batches is a simulation of reality. A website that receives data all the time and evaluate and train the data ”live” is more realistic.

If the size of the bird’s mouth are different from various pictures,then the different size of filters should be used or pictures should be normailzed.But we can not solve this problem in CNN,this is a also a question that needs to be continued research.For example,in mentioned,we can add a new NN in front of the CNN,then the images can be processed in rotate and so on,finally we can get better performance.

References

[1] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa: Massive online analysis. *Journal of Machine Learning Research*, 11(May):1601–1604, 2010.

[2] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[3] Stefan Wermter, Cornelius Weber, Wlodzislaw Duch, Timo Honkela, Petia Koprinkova-Hristova, Sven Magg, Günther Palm, and Allessandro EP Villa. *Artificial Neural Networks and Machine Learning–ICANN 2014: 24th International Conference on Artificial Neural Networks, Hamburg, Germany, September 15-19, 2014, Proceedings*, volume 8681. Springer, 2014.

[4] Tom M Mitchell et al. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45(37):870–877, 1997.

[5] B Yegnanarayana. *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

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

[7] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[8] Matt W Gardner and SR Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636, 1998.

[9] Sebastian Kauschke and Johannes Fürnkranz. Batchwise patching of classifiers. In *AAAI*, 2018.

[10] Jing Gao, Wei Fan, Jing Jiang, and Jiawei Han. Knowledge transfer via multiple model local structure mapping. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 283–291. ACM, 2008.

[11] E Hüllermeier. Case-based approximate reasoning, volume 44 of theory and decision library, series b: Mathematical and statistical methods, 2007.

[12] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[13] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[14] Geoffrey I Webb, Loong Kuan Lee, François Petitjean, and Bart Goethals. Understanding concept drift. *arXiv preprint arXiv:1704.00362*, 2017.

[15] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, pages 242–264. IGI Global, 2010.

[16] Indrė Žliobaitė. Learning under concept drift: an overview. *arXiv preprint arXiv:1010.4784*, 2010.

[17] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Brazilian symposium on artificial intelligence*, pages 286–295. Springer, 2004.

[18] Judy Hoffman, Sergio Guadarrama, Eric S Tzeng, Ronghang Hu, Jeff Donahue, Ross Girshick, Trevor Darrell, and Kate Saenko. Lsda: Large scale detection through adaptation. In *Advances in Neural Information Processing Systems*, pages 3536–3544, 2014.

[19] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[20] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *Proc. CVPR Workshop on Fine-Grained Visual Categorization (FGVC)*, volume 2, page 1, 2011.

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