Summer 2019

Evaluation of CNN Models with Fashion MNIST Data

Yue Zhang
yuez@iastate.edu

Evaluation of CNN Models with Fashion MNIST Data

by

Yue Zhang

A report submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Electrical Engineering

Minor: Statistics

Program of Study Committee:
Joseph Zambreno, Major Professor
Alicia L Carriquiry

Iowa State University

Ames, Iowa

2019

TABLE OF CONTENTS

Page

LIST OF FIGURES iii

ABSTRACT iv

CHAPTER 1. INTRODUCTION 1

CHAPTER 2. BACKGROUND 2

CHAPTER 3. MODEL EXPLORATION 6
3.1 Kaggle Data 6
3.2 Data Preparation 7
3.3 Training Data 8
3.4 Four Models for Training and Testing (Results and Analysis) 9
3.4.1 LeNet-5 9
3.4.2 AlexNet-11 9
3.4.3 VGG-16 10
3.4.4 ResNet-20 11
3.5 Comparing Results of Four Models 11
3.6 Insights From Hierarchies 18

CHAPTER 4. MODEL APPLICATION 26
4.1 Data Preparation 26
4.2 Training Data 27
4.3 Best Model for Training and Testing (Results and Analysis) 27
4.4 Modified Model for Each Stage 28
4.4.1 Changing Pooling Layer For Each Stage 30

CHAPTER 5. CONCLUSIONS 33
BIBLIOGRAPHY 34

LIST OF FIGURES

		Page
Figure 3.1	LeNet [1]	9
Figure 3.2	AlexNet [2]	10
Figure 3.3	VGG [3]	11
Figure 3.4	AlexNet testing results.	12
Figure 3.5	LeNet testing results.	12
Figure 3.6	VGG testing results.	13
Figure 3.7	ResNet testing results.	13
Figure 3.8	LeNet Validation plot	14
Figure 3.9	LeNet Confusion Table	15
Figure 3.10	AlexNet first stage validation plot and confusion table.	18
Figure 3.11	AlexNet-Shoes Second Stage Result	19
Figure 3.12	AlexNet Tops Second Stage Result	19
Figure 3.13	LeNet-First Stage Result	20
Figure 3.14	LeNet-Shoes Second Stage Result	20
Figure 3.15	LeNet-Tops Second Stage Result	21
Figure 3.16	VGG-First Stage Result	21
Figure 3.17	VGG-Shoes Second Stage Result	22
Figure 3.18	VGG-Tops Second Stage Result	22
Figure 3.19	ResNet-First Stage Result	23
Figure 3.20	ResNet-Shoes Second Stage Result	23
Figure 3.21	ResNet-Tops Second Stage Result	24
Figure 3.22	Final Two Stage Model Confusion Table	25
Figure 4.1	Changing pooling layer result from one to four layers.	31
Figure 4.2	Two pooling layer of two-stage confusion table	32

ABSTRACT

The work seeks to evaluate the performance of four CNNs with respect to Fashion MNIST data set. Fashion MNIST is a dataset of images consisting of 70000 28×28 grayscale images, associated with a label from 10 classes. In this report, the accuracy of four popular CNN models that are LeNet-5, AlexNet, VGG-16 and ResNet for classifying MNIST-fashion data revealed that ResNet was the best suited for the selected dataset. The training process has been coded with Tensorflow. After the result accuracy improving, we could use the new model to the fashion company that can help the fashion company more accurately classify clothing. Moreover you could build your own closet online for your fashion.

CHAPTER 1. INTRODUCTION

Convolutional neural networks (CNN) are the most current progress in the image retrieval and classification domain. Research data that is entrenched and processed in industrial and commercial applications can be brought into productive use with CNN which is a form of a Deep Neural Network (DNN) [4]. It is applied to image processing tasks, human action recognition and fashion image classification [5] [6].

The aim of this work is to analyze a few different models of CNN namely, the LeNet-5, AlexNet-11, VGG-16 and ResNet-20. Analysis is carried out with a fashion dataset from the Kaggle website. A training process for the distance function is coded with TensorFlow. The results of each of the four CNN models are then analyzed and evaluated to identify the best out of the four models. The model that performs well is recommended to the fashion industry for their use in classification, and also for those third-party vendors and consumers who are interested in creating their own fashion closets with online data [7].

CHAPTER 2. BACKGROUND

CNN are specialized neural networks responsible for the processing of data with an input shape resembling a 2D matrix like image. They are made use of in image detection or pattern detection and image classification. Images or pictures are basically pixels in a matrix shape, and the trained distance function of the CNN is useful for identifying the image, and/or for classifying the image as per pre-annotated categories [8]. The CNN is a convoluted neural network, where convolution is basically a mathematic operation with input I and argument, and a kernel K producing an output useful for understanding how shapes are modified. "A convolution is a weighted sum of the pixel values of the image, as the window slides across the whole image. Turns out, this convolution process throughout an image with a weight matrix produces another image (of the same size, depending on the convention). Convolving is the process of applying a convolution" [9]. A feature map is drawn with the convolution function. If there is an image x in a given image with pixels in the 2D format, then the array is set out in different color channels of the RGB. A feature detector (also called the kernel) as represented by 'w' will output the feature map. From the function below, s[t] means feature map, x means input and w means kernel.

$$s[t] = (x \times w)[t] = \sum_{\alpha=-\infty}^{\alpha=\infty} x[\alpha]w[\alpha + t] \tag{2.1}$$

Similarity of signals is computed as part of convolution and this is how image identification and categorization are done. In any CNN, there are usually multiple convolution layers, meaning many alternate convolutions are generated. A weight matrix used for the calculation will therefore be a tensor in the form of $5 \times 5 \times n$, here 'n' refers to the number of convolutions in the CNN.

Schindler [10] present how CNN is used for fashion image classification. Researchers work with the photographs of cloths, and their aim is to categorize the photographs into pre-annotated cate-

gories, such as categories for skirts, jeans, sports shoes, etc. CNN is made use of for the classification. Some existing online e-commerce companies already make use of such classification. Companies like Asos-EU 1, Farfetch 2 or Zalando 3 present consumers access to the data of their products, including the metadata and the image data. Data provided however varies in terms of the quality of the data, the granularity which could become a challenge in the classification, and also the taxonomy.

Some of the fashion image domain specific challenges that have been observed in research works on mage processing with DNN are that of the varying styles, the textures, the shapes and colors [9]. Usually, an image quality is probably the biggest challenges. When the image quality is high and is professional produced, and then the classification is aided. However, not all images in the fashion domain databases are high resolution images. Usually, when analyzing for these different photographs and assignment of categories, there are two general ways that quality is understood. One way is the arrangement of products before a white background. This displays the product clearly, and only the product. However, as understood in the fashion domain, there is another form of presentation as well, where a person wears the product. The second form of image is that of a person or a part of the person (such as the leg of the person with a pant suit, or the wrist of a person with a watch, etc.). Semantic noise is introduced in the second category of pictures. The first category has reduced semantic noise. The second category suffers here because the person is wearing multiple items and hence categorization to a label becomes a challenge. Many of the existing studies report that there are concerns with how they define and extract content from one that produces a high semantic noise [11] [12]. Clothing items presented in images introduce high variability to the data set compared to general other data sets. There is also a high amount of deform-ability attached to this data set. Both these aspects introduce challenges for extraction and identification. CNN is a pre-trained neural network, and hence the distance function has to be well trained in order to assess similarities between the fashion images. The issues of semantic noise are thus handled much better when using CNN. In applied fashion research, researchers have worked on evaluating CNN for their classification accuracies for differentiating not just products

based on categories, but also aspects like 1) differentiating persons from products, 2) gender. For instance, for classifying persons from product images, a custom VGG CNN form has been used with stacked convolution layers and predictions with 91.07 percent of accuracy were gathered. In the case of category classification, options for researchers are to either train from scratch or to fine tune. In the case of gender prediction with fashion image data, product classification once again was dependent on pre-training, but it was possible to fine tune and obtain a performing value of around 88 percent. Most of the work with CNN architectures does support the notion that even if there are large amounts of high-quality fashion images, the amount of pretraining and fine tuning that was done to prepare the distance function was what helped in performance. The quality of the image itself was not just a singular deciding factor for performance [13] [14].

The form of data augmentation used will also affect the categorization and identification aspects. "Data augmentation is often used in order to improve generalization properties. Typically, random cropping of rescaled images together with random horizontal flipping and random RGB color and brightness shifts are used. Different schemes exist for rescaling and cropping the images (i.e. single scale vs. multi scale training). Multi-crop evaluation during test time is also often used, although computationally more expensive and with limited performance improvement" [?]. The aim of random rescaling or cropping is necessary to understand the features of the objects at different scales. However, when using Keras, it is noted that these aspects of data augmentation is not available as such, meaning the user has to develop the right data augmentation technique and implement it when training the data. Preprocessing functions like that of 'Image Data Generator' function is helpful for conducting data augmentation. In most applications of the CNN model on the Fashion MNIST data set, it is observed that the users divide the data set. The users divide the data set into training image data set and validation image data set. If the total image data set is 60000 images, then for training images around 48000 images are allocated. This is about 80 percent of the total images. The rest of the 20 percent which is 12000 images are allocated for validation. This form of division reportedly helps to understand the accuracy of the model, and whether data

is being over-fit. Lowering of the learning rate, and training for more epochs could be necessary if the validation accuracy is low [15].

CHAPTER 3. MODEL EXPLORATION

Data for the analysis of the CNN's was downloaded from the Kaggle website. The Kaggle is an open source online community that was developed and owned by Google. Users can download datasets for work with CNN, and users can also publish their own datasets as well. The Fashion MNIST dataset is more challenging than the existing MNIST datasets on the site, and the dataset basically includes as many as 60,000 training examples, 10,000 testing examples, and 10 classes. There is 28×28 gray scale, single channel images. It is a data set from Zalando originally and each of the gray scale images has the 10 classes.

3.1 Kaggle Data

All images of the Fashion MNIST basically are of 28×28 matrix, with 28 pixels in their height and 28 pixels in their width. The total is 784 pixels in the matrix. Each of the pixel in the matrix has an assigned value and this value is dependent on the lightness or the darkness of the pixel. The value is drawn from an integer data set with limits 0-255. A low value within this integer assignment for instance will be lighter, compared to a higher value from the set. The training data set, and the testing data set will have around 785 columns. First column in both will have class labels, as presented below (such as a sneaker, a dress or a coat etc.). Basically, the first column just represents the clothing category. The rest of the columns will have the pixel values for the image based on the lightness or darkness properties set with the 0-255 integer value. Pixels within the training and testing data sets are classified under these values. When a pixel has to be identified on the image, the pixel x is identified as i * 28 + j. Here i and j are integers anywhere between the numbers of 0 ad 27. Pixel is located on the row I, and the column j of the standard 28×28 matrix. Every row is a separate image, every column is a class label, and every value of the pixel is indicative of the darkness level of the pixel. The fashion class labels or categories that are identified for the data set

are in the following. These are the only images used, and therefore the CNNs tested and trained with these data sets won't be directly transferable to real time fashion data sets out in the fashion market.

Label	Classification
0	T-shirt
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle Boots

3.2 Data Preparation

The below are the steps for data preparation that was used in this work:

1. First the necessary libraries are imported such as tensorflow, keros, numpy, matplotlib.pylot.

2. The second step is to load the fashion MNIST dataset. The fashion MNIST dataset is downloaded from Kaggle. The dataset has the fashion images of clothing and other items.

3. From this dataset, training and evaluation datasets are then created. The dataset is downloaded from the keras dataset. Dataset contains fashion images of clothing items and accessories. From the dataset, we create training and evaluation dataset to the each data and label.

4. A target dictation connecting fashion images or the inputs with the target variables or the 10 classes is now coded. This corresponds to the existing data definition, and a typical assignment

would be as follows. target dict = 0: 'T-shirt/top', 1: 'Trouser', 2: 'Pullover', 3: 'Dress', 4: 'Coat', 5: 'Sandal', 6: 'Shirt', 7: 'Sneaker', 8: 'Bag', 9: 'Ankle boot'

5. The data set is then analyzed to check the training and the evaluation images. This is done using the print command for training and evaluation. The answer for the two print commands were (6000, 28, 28) and (10000, 28, 28).

6. This corresponds to the 60,000 images in the downloaded data set and the evaluation data set with the 10,000 images.

7. The data set was finally normalized. Normalization is necessary to ensure that all the data are on the same scale and this usually improves the performance. For the Fashion MNIST data set, the normalization for training dataset and the evaluation data set would be handled as mapping [0,255] to the [0,1] that will increase the training speed.

3.3 Training Data

Training with Keras is carried out by first initializing the environment and setting up the needed imports. The number of epochs to train for, and the base learning rate and base size are initialized as part of data preparation. In training, the hyperparameters are set in the end lines. The hyper parameter values are the learning rate, the batch size and the epochs of training [16].

For instance, in the VGG 16, after the Fashion MNIST data set is loaded, the training will carry out the generation of a history plot and montage visualization for the user to view the results of training. The resulting model can be evaluated with the data being output and classification reports can also be drawn at this point [17].

3.4 Four Models for Training and Testing (Results and Analysis)

3.4.1 LeNet-5

The LeNet-5 originally was used for the classification of digits and for identifying hand written numbers in checks that were digitized in a 32*32-pixel input image. More convolutional layers contributed to the use of the LeNet-5 for high resolution images. The architecture of the LeNet-5 convolutional neural network is shown below [1]. The LeNet-5 is one with five convolutional layers with different planes. The planes are basically feature maps and show the set of units with weights that are identified as being similar.

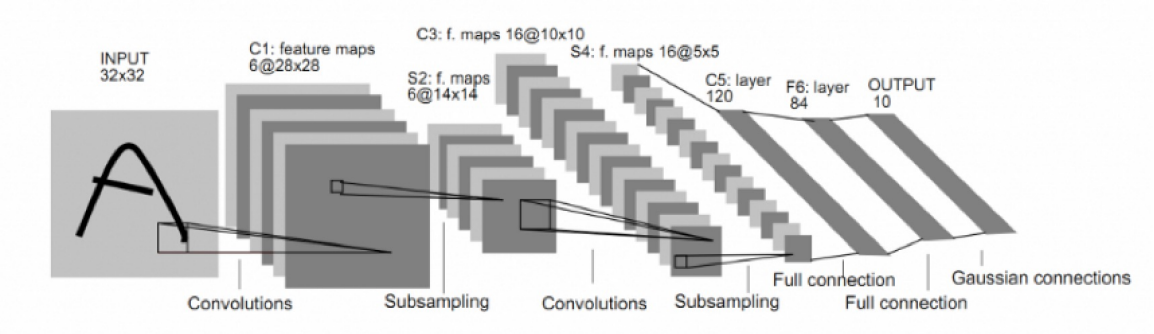


Figure 3.1: LeNet [1]

With five convolutional layers, the first layer has 6 filters for learning. Filters have to be stepped across in order to update values. Kernel values are calculated and are then fed forward for calculating the loss and then the update is once again executed by means of backpropagation. Different convolution filter can be used for the layers and hence different sizes are possible.

3.4.2 AlexNet-11

AlexNet was a much larger CNN compared to others like VGG in its time [18] and was used initially in computer vision tasks. There were 60 million parameters in AlexNet. It has 650,000 neurons and yet in current times, is supported by much faster GPUs compared to the GTX 580 3GB GPUs in the past.

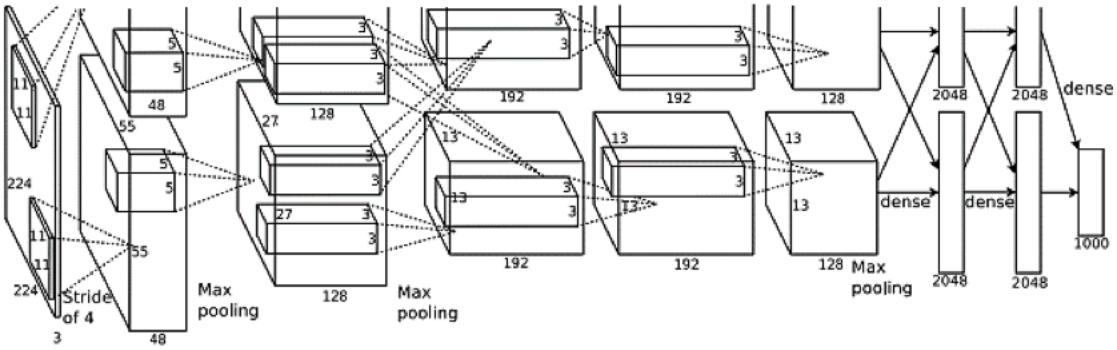


Figure 3.2: AlexNet [2]

It has 5 convolutional layers and has 3 fully connected layers. Kernels are multiple convoluted and have the capability to extract features from single images [19] [20]. As many as 96 kernels are present in the first convoluted layer and the kernels are of size 11*11*3. The width and the height of the kernels match and are set at size 11. The depth will equal the number of channels used [21]. There are max pooling layers in AlexNet, with the fifth convolutional layer followed by the max-pooling layer whose output is presented in the connected series of layers. A soft-max classifier is used here.

3.4.3 VGG-16

VGG Net was developed by Simonyan and Zisserman. The VGG Net or VGG-16 as it is called has around 16 convolutional layers. It has a very uniform architecture, and this is one of its advantageous aspects. The VGG Net shares similarities with AlexNet. Similar to AlexNet, the VGG also has only 3x3 convolutions. However, it is different from the AlexNet because of the numbers of filters used. Compared to the AlexnNet, VGG makes use of lots of filters. The training for this CNN happens for a time period of 2-3 weeks. Four GPUS are used for the training. In the contemporary situations of performance and CNNs, this appears to be a preferred choice because of its uniform architecture, and the filters it uses that makes it ideal for complex mage extraction or classification. In fact, the weight configuration of the VGG-16 is available publicly. Many applications make use of the weights as a baseline for developing their feature extraction. While these are some of the

positive aspects of the VGG Net that makes it simpler and advantageous for use, some aspects like its parameters introduces a complexity. VGG Net as around 138 million parameters, and this could become complex and challenging for some users.

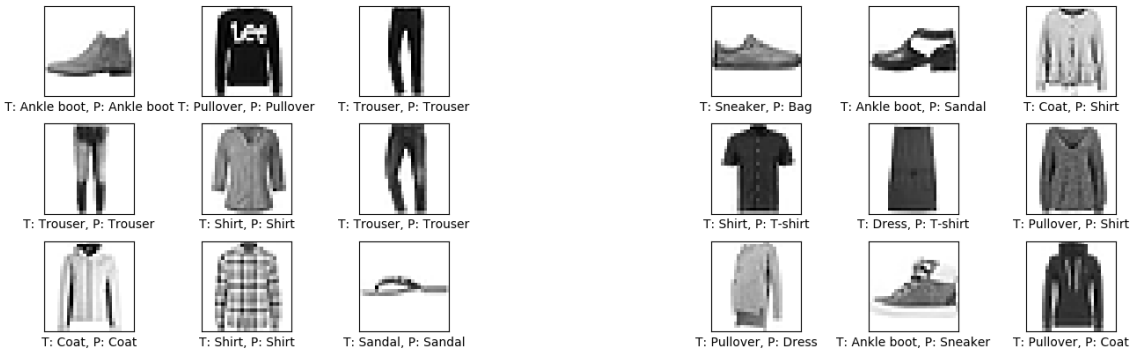
Figure 3.3: VGG [3]

3.4.4 ResNet-20

The residual neutral network was introduced with an architecture that was novel compared to the other existing CNN architectures. It has what is called the skip connections or gated units (or gated recurrent units). One of its main features is its batch normalization. Because of this feature, the ResNet-20 is able to train neural networks with as many as 152 layers [22] [23]. These many layers could introduce a very high level of complexity in other CNNs, but this is not the case with ResNet. With very low complexity and with an error rate at 3.57 percent on datasets, this CNN offers top performance.

3.5 Comparing Results of Four Models

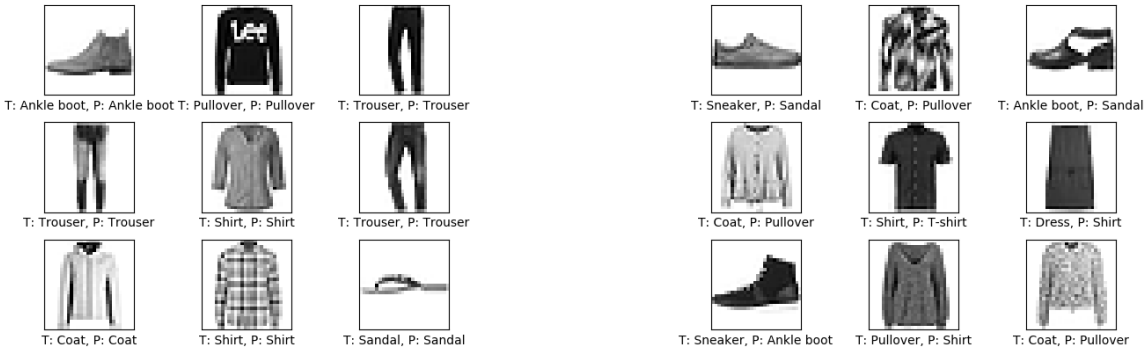
The results for each of the CNN are compared and analyzed in this section.For Alex Net, the input and the training and prediction data set are as follows. It can be observed that there are issues in prediction for sneakers, for ankle boots and for pullover etc. Some of the images for Shirt are categorized correctly, but many other images were wrongly identified and categorized as shirts in AlexNet. The code for AlexNet includes the definition for depth, the numbers of classes, width and height. The model is first initialized and the convolutional layers, and max pooling layers size and strides are defined by using the kernel initializer function and the model adds function.



(a) (b)

Figure 3.4: AlexNet testing results.

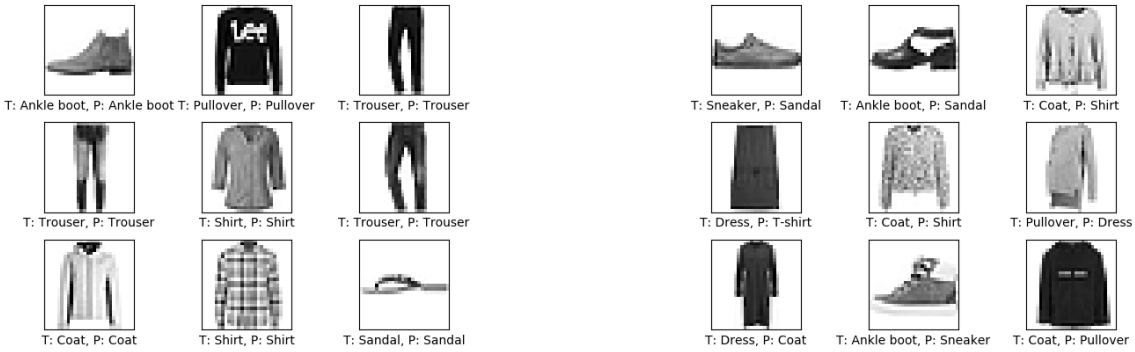
For LeNet the image set used as input for training and prediction is the same as for AlexNet, and the results are as follows. LeNet also suffers same performance concerns like AlexNet and appears to be more accurate only with the Shirt image classification, compared to others. Similar to AlexNet, for the LeNet as well, the initialization, convolutional layer activation and the maxpooling are specified. Batch normalization functions are run, too. The same is repeated for VGG as well, except here the number of convolutional layers is higher.



(a) (b)

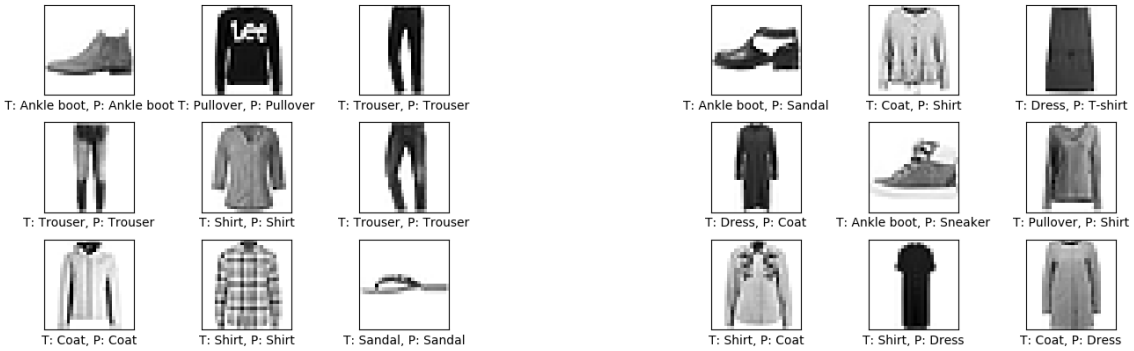
Figure 3.5: LeNet testing results.

As the AlexNet and LeNet, the image categorization output for VGG and ResNet are respectively presented below.



(a) (b)

Figure 3.6: VGG testing results.



(a) (b)

Figure 3.7: ResNet testing results.

Now although in image categorization it does appear like AlexNet and LeNet have done better, the entire performance as understood from the accuracy curves and confusion matrix present ResNet as showing better performance. For each of the CNNs, the Keras and deep learning fashion MNIST

training plot helps reveal the accuracy/loss curves. The below are the accuracy loss curves generated for the CNNs.

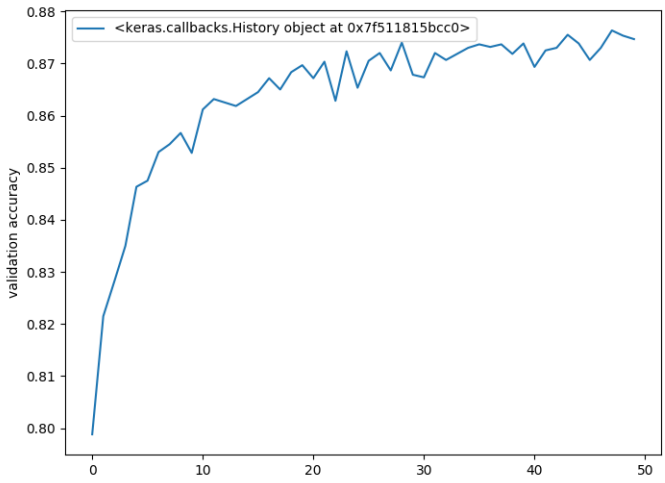


Figure 3.8: LeNet Validation plot

The below are the confusion matrix for the four CNNs. The plot confusion matrix is basically the technique for evaluating the performance of the classification algorithm. Using classification accuracy as identified alone can be misleading, especially in those instances where there is an unequal number of observations in each class. The confusion matrix is better in this case. Additionally, when more than two classes are used in the data set, such as in this examination, it is better to make use of the confusion matrix. Classification accuracy has also been presented as hiding those details that are needed for better improving the performance of the classification model. For instance, in data with more than 2 classes, even when one gets a classification accuracy of 80 percent, then it would still be difficult to determine if the classes are predicted well or if one or two classes are neglected by the model/algorithm. Similarly, in situations when the data being used has classes in an odd number, then even if an accuracy of 90 percent is achieved then this might not be a good score because the 90 records for every 100 could actually belong to one class, meaning the score pertains to predictions from the most common class. Therefore, the calculation of the confusion matrix will

show how the classification model works and what forms of errors are generated. It shows how the classification model is confused in its predictions. The numbers of correct and incorrect predictions are usually summarized with count values and are broken down by class.

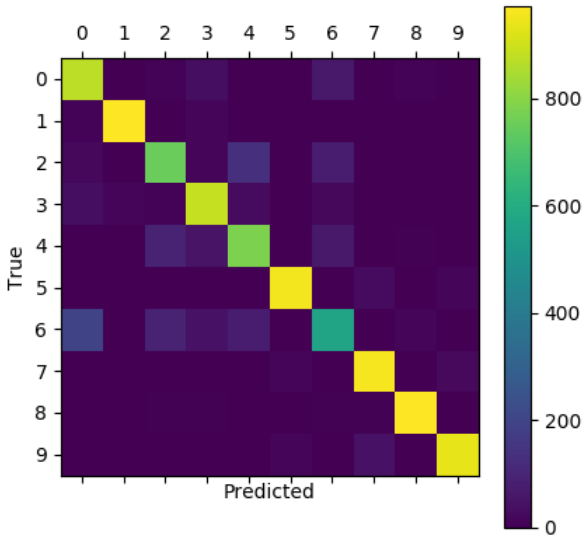


Figure 3.9: LeNet Confusion Table

In the above matrices for AlexNet, LeNet, VGG and ResNet, the rows of the matrix correspond to the predicted class, and the column in the matrix corresponds to an actual class. ResNet is identified as giving the best performance compared to the other models. Confusion matrix indicates a true prediction above 800 for most of its predictions. ResNet code for the work includes the definition for convolutional layer initializations, setting of kernel size, strides and the running of the batch normalization function. The accuracy plotting also shows that ResNet CNN is accurate in the range of 92% to 93% better than the other models. VGG is at the rate of 91% and 92%. AlexNet is 89% to 91% and LeNet is 85% to 88%. As the epoch value is increased in the training for AlexNet, it is observed that the validation accuracy also improves. There are some dips in accuracy but there is also a constantly steady increase. Similarly, in the case of LeNet, as epoch is increased, the validation accuracy increases. The same trend is noticed in the case of VGG and ResNet, but

in ResNet the validation accuracy rates are high. In the case of ResNet, at epoch value reaching 30, there appears to be a dip in validation accuracy rates; however, the relative performance is still good compared to the rest.

In ResNet, usually, it is assumed that adding more layers would result in complex function and this in fact could cause a failure in performance because overfitting issue could arise [22] [23]. However, in actual conditions, the error was observed because of other issues. More training errors could result in a 56 layer or more convoluted layer network. ResNet works with as many as 152 layers, and therefore it was natural to have a concern on whether the error of over-fitting influences performance. However, the plots showed no issues in performance, and the ResNet with the high number of layers did exhibit better performance. This can be explained in the words of one of the users of ResNet, "the failure of the 56-layer CNN could be blamed on the optimization function, initialization of the network, or the famous vanishing/exploding gradient problem. Vanishing gradients are especially easy to blame for this; however, it is argued that the use of Batch Normalization ensures that the gradients have healthy norms. Amongst the many theories explaining why Deeper Networks fail to perform better than their Shallow counterparts, it is sometimes better to look for empirical results for explanation and work backwards from there" [24].

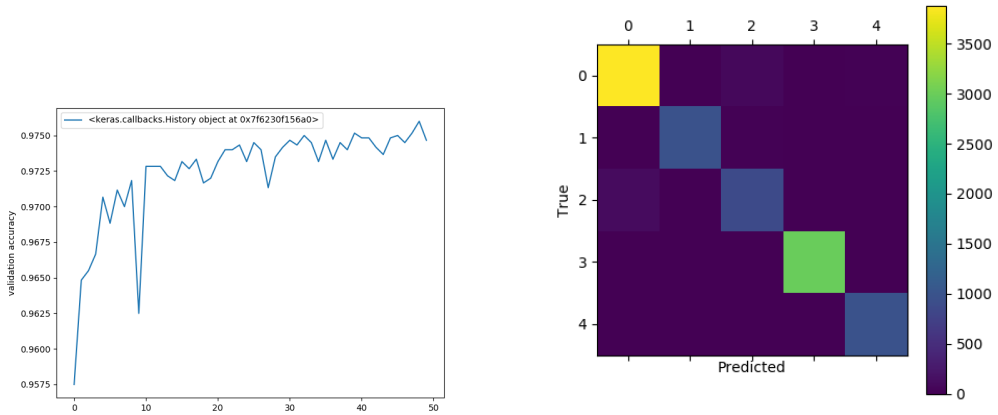
For almost all the models, a better accuracy is obtained in the testing set if the training epoch is set higher. A more than 90 percent accuracy can be observed for at least three models of the four, but there are some struggles seen in classification in some of the models. The plots for the models show that there is little over-fitting, and data augmentation could lead to more accuracy as well. It was presented in the research background section that despite the CNN architecture that was used, and the amount and quality of the provided fashion images, it is the pre-training and the fine tuning of the models that lead to better performance [25]. Analyzing models on a scale of two million or more images, and with more deep training could reveal much better insights on the CNN architecture that will be better suited for image identification and categorization. Some CNN architectures

are more constrained than others. For instance, the LeNet-5 architecture has the ability to process high resolution images, and is better only with higher resolution images, and this is a limitation in itself. The availability of the computing resources will challenge the use of LeNet-5. Not all fashion industry users or customers who want to build their own fashion set might be able to afford the computing resources especially if their data set is going to be bigger. In such cases, either of the other models would also offer better use. The ResNet however is the one with more promising accuracy with respect to the Fashion dataset and hence is recommended for use in the fashion domain.

There are however some limitations and considerations that has to be understood by a user when they use the models and MNIST data. For instance, the model and the data set that has been trained with for this work will not be applicable to other images that fall outside the Fashion MNIST data set. It is necessary to consider training based on the data set. The images that are used for this current evaluation of CNN models furthermore relies on certain aspects of the images, such as the color of the background and the foreground. The pixel color and number assignment, the resizing of matrix to 28×28 pixels for instance, are not how images exist in the real world. In the context of real-world fashion and for clothing images, the data set itself must be preprocessed and prepared in order to use them similar to the Fashion MNIST dataset. Even if the data set is processed to meet the requirements like the current set, the model still cannot be transferable to the real-world image without the needed training. As observed from the evaluation, the ResNet offers better accuracy, so the ResNet can be chosen and then it can be trained with real world images. Different classifications and multi output networks may need to be used based on the real-world situation. Only when such extra work is done, can the results of the work be replicated on newer data sets. Even then, there might not be guaranteed that the same level of validation accuracy would be achieved [25]. Epoch levels adjustments might be needed, and further observation and tuning could help develop a CNN that presents better validation accuracy for the new data sets. These however are not within the scope of this work, and the work has merely attempted to evaluate how four different CNNs perform in image classification with the Fashion MNIST.

3.6 Insights From Hierarchies

Convolutional layer size can be optimized. Kernel size can have a significant effect on the computational efficiency and can even affect over-fitting. This has been proved in research works too and was identified as a significant factor affecting computation in this work. For instance, in 2012 when AlexNet was introduced, it had a convolution size of 11*11, later GoogleNet had a convolutional layer of around 7*7 with a kernel at 5*5. It was identified that the kernel size affected the number of parameters. In fact, the parameters exhibit a quadratic growth with respect to kernel size. As the kernel size grows and the parameters or factors grow, then the convolutional kernels are no longer cost efficient. A number of channels are needed at this point. The number of channels is defined by $n = k^2 \times c - in \times c - out$. It is recommended that the kernel size be maintained as 3*3 or 5*5. The first convolutional layer size is usually kept larger than the other and convolutions are often stacked vertically as a best practice. The following shows how the CNNs fare across stages. AlexNet first stage validation accuracy is presented below.

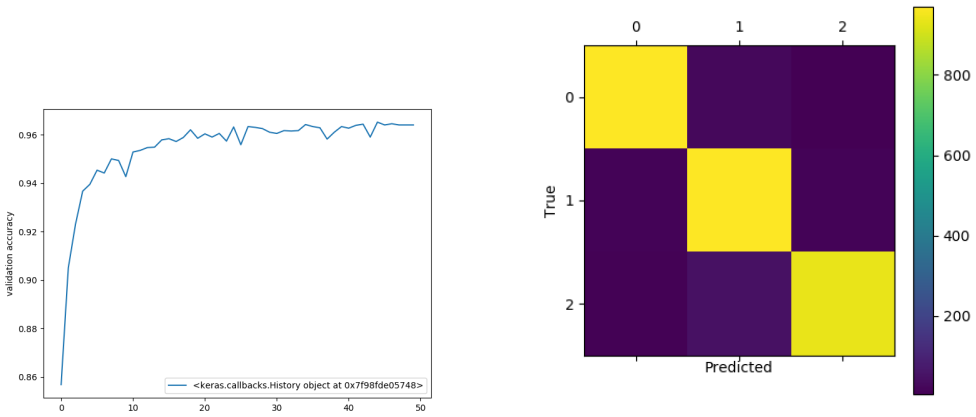


(a) (b)

Figure 3.10: AlexNet first stage validation plot and confusion table.

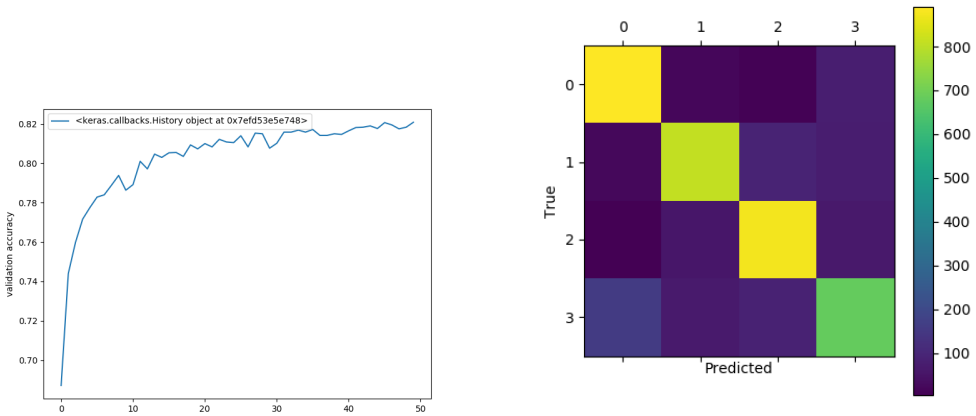
AlexNet second stage shoe validation accuracy is presented below along with the corresponding confusion matrix with predicted and true states. It can be seen that the validation accuracy dips

slightly below 97% to 96% for shoe analysis. For top analysis, the validation accuracy reaches a higher value, compared to even the first stage at 81.45%.



(a) (b)

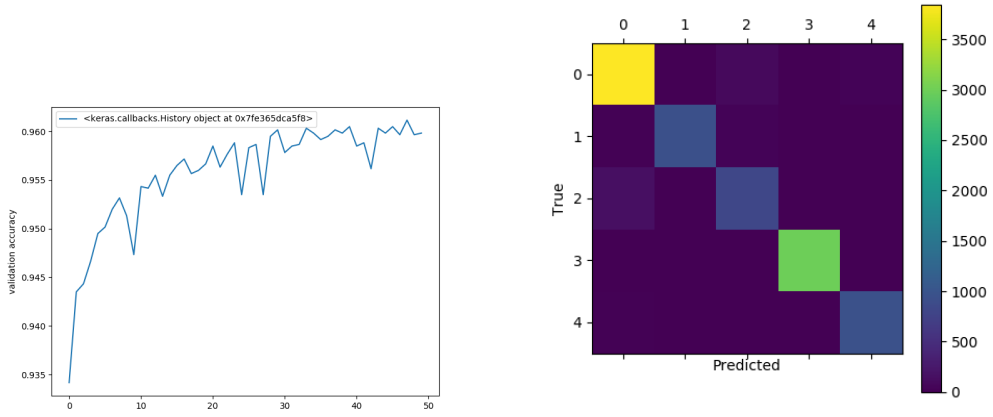
Figure 3.11: AlexNet-Shoes Second Stage Result



(a) (b)

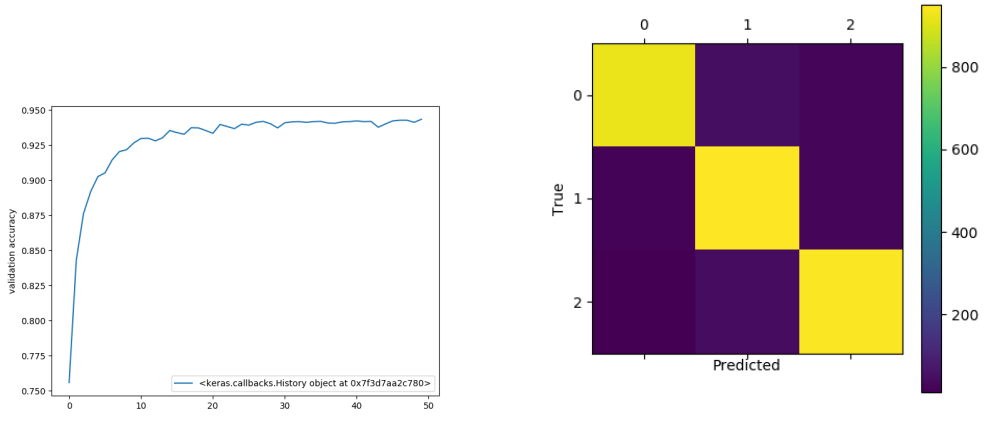
Figure 3.12: AlexNet Tops Second Stage Result

Data for LeNet first stage, shoe and top analysis is respectively presented below. The accuracy of first stage is 95.53%, the accuracy of shoes is 93.93%, and the accuracy of Tops is 74.75%. From the result, the accuracy of tops is too low because the purpose of accuracy is 95.00%.



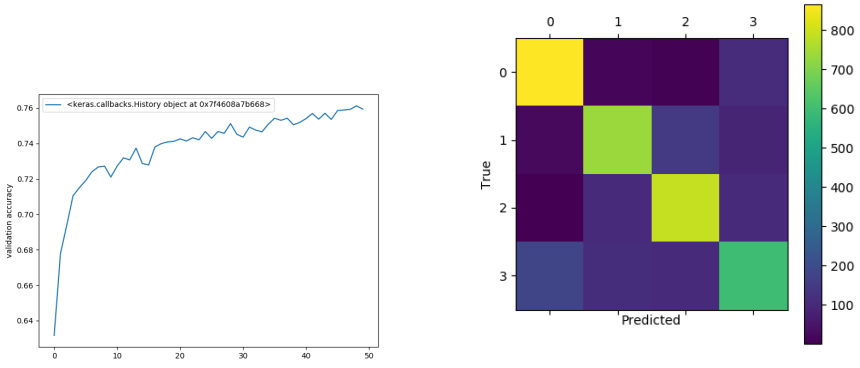
(a) (b)

Figure 3.13: LeNet-First Stage Result



(a) (b)

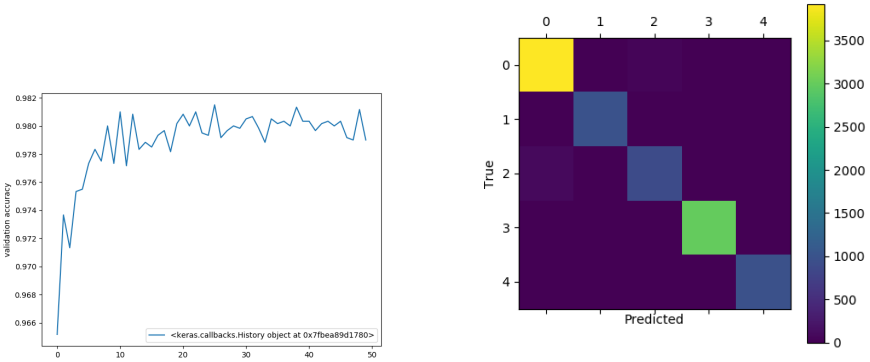
Figure 3.14: LeNet-Shoes Second Stage Result



(a) (b)

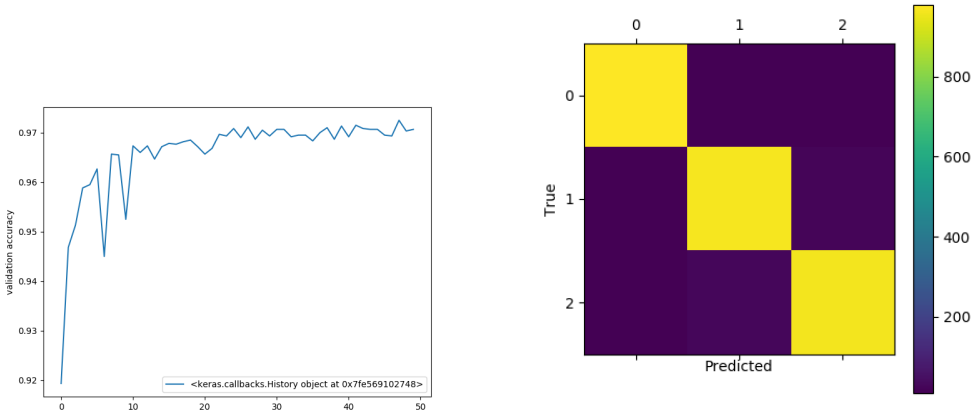
Figure 3.15: LeNet-Tops Second Stage Result

Data for VGG first stage, shoe and top analysis is respectively presented below. The accuracy of first stage is 97.79%, the accuracy of shoes is 97.00%, and the accuracy of Tops is 85.42%. The accuracy of first stage and shoes stage are almost higher than 95.00%. The problem of accuracy is happened in the Tops stage.



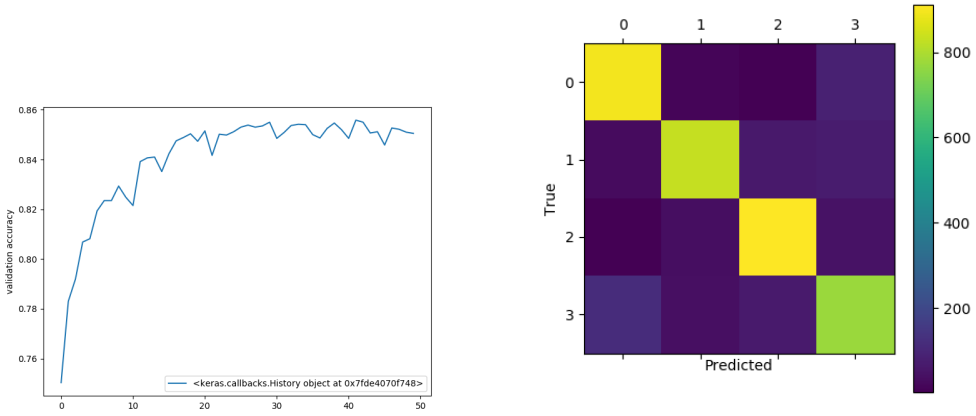
(a) (b)

Figure 3.16: VGG-First Stage Result



(a) (b)

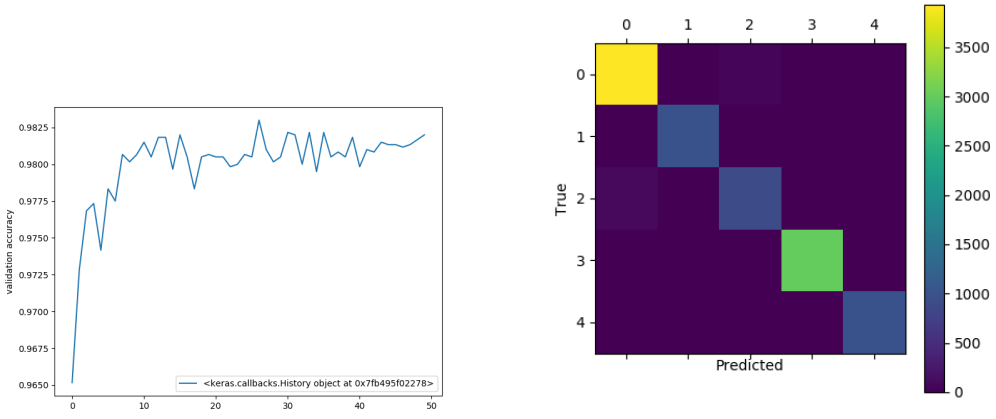
Figure 3.17: VGG-Shoes Second Stage Result



(a) (b)

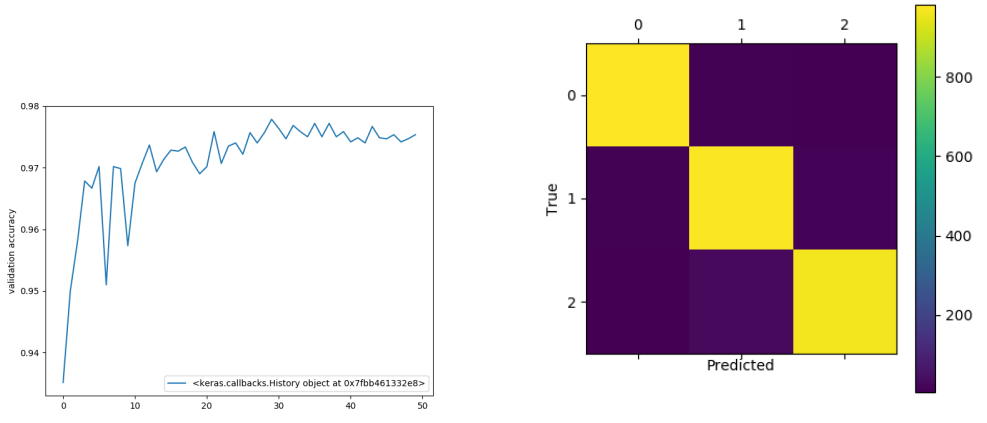
Figure 3.18: VGG-Tops Second Stage Result

Data for ResNet first stage, shoe and top analysis is respectively presented below. The accuracy of first stage is 98.05%, the accuracy of shoes is 97.47%, and the accuracy of Tops is 85.50%. The accuracy of tops stage is not arrive the 95.00% but it is highest than others.



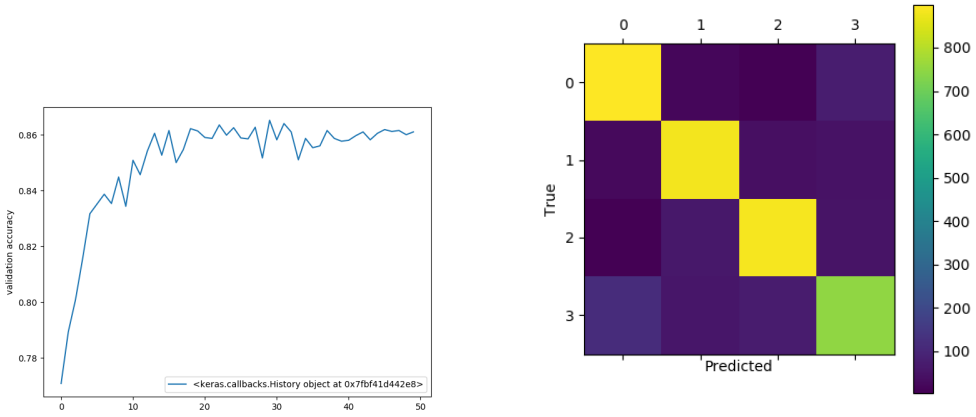
(a) (b)

Figure 3.19: ResNet-First Stage Result



(a) (b)

Figure 3.20: ResNet-Shoes Second Stage Result



(a) (b)

Figure 3.21: ResNet-Tops Second Stage Result

All in all, the first stage and second stage for ResNet offers more validation accuracy compared to stage 3. It is the same for VGG. All in all, the best two-stage model is ResNet, so the best model will use to analyze in the Real Data.

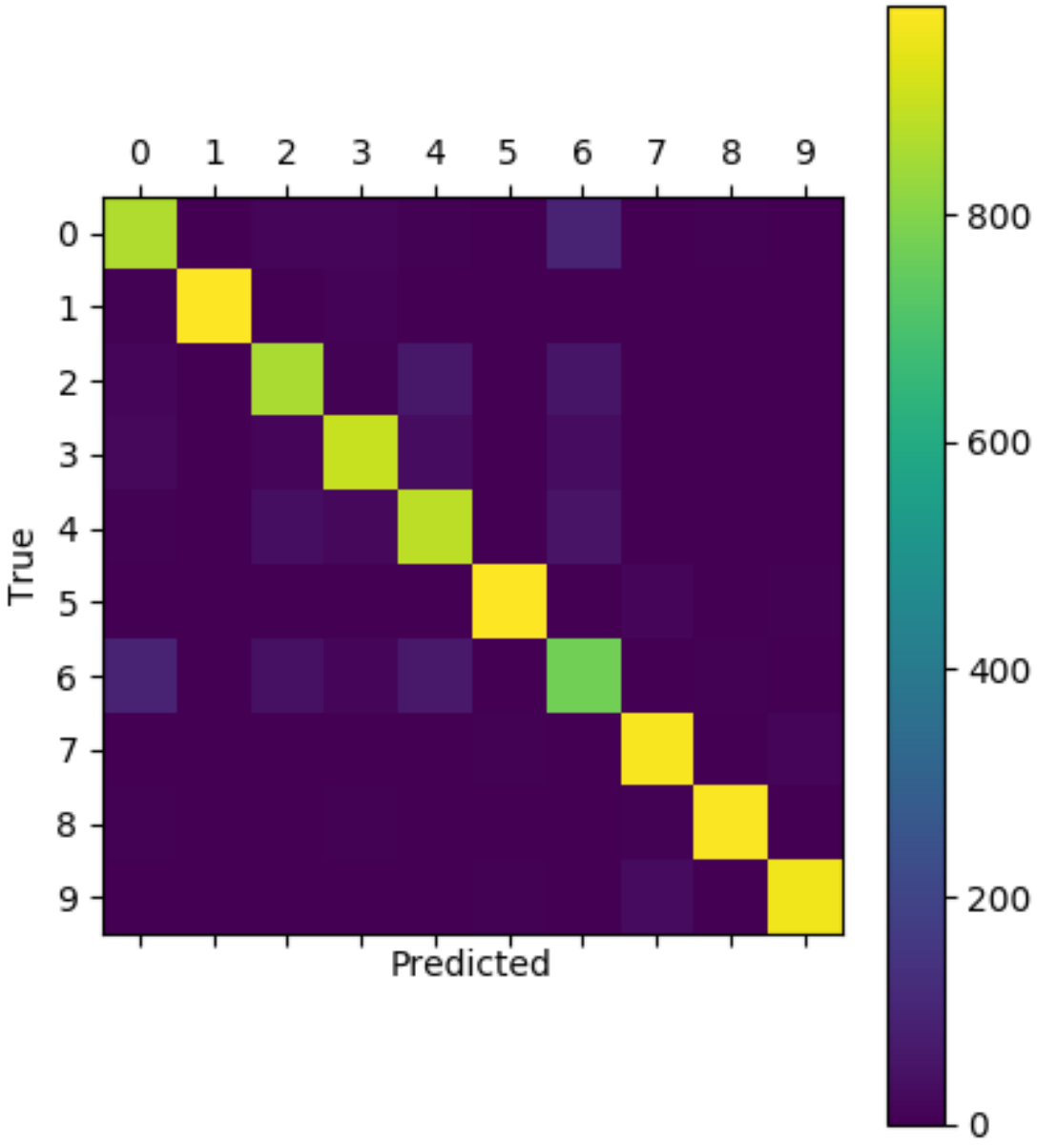


Figure 3.22: Final Two Stage Model Confusion Table

CHAPTER 4. MODEL APPLICATION

With the Kaggle data, the results of comparison now indicate that the ResNet model fares better than the rest of the models. Now in this chapter on testing the best model with real data, the two stage ResNet is used. Test data is set at 100, and the real data is loaded from the site imagenet.com. In order to make the data frame consistent, it has been re-sized to 224×224.

4.1 Data Preparation

Data preparation is the first step, and it involved data loading. Similar to the data loading operation used in the comparative analysis of the four CNNs, the operations called out are as follows. Firstly, the files, image size, classes, images per file and the data path are set out. The number of files initialized, the number of files is set as 3. The image size is set as 224. The number of classes is 3. The image of per file is 1000, and the number of images is calculated as the number of files multiplied by the images per file, which is the number of files * the image of per file. The data path is set as part of data preparation. In the loading of data, the images and the labels for the images are both loaded as the training data set. A random shuffle function is run.

Data normalization is part of the data preparation process. In the case of modern convolutional neural networks like ResNet, the batch normalization process is conducted as follows. The use of batch normalization leads to the speeding up of the training process, and furthermore, it improves the conditioning of the problem [26] [27]. It is in the general improvement of the conditioning of the problem that back propagation of the gradients is eased up. Total number of iterations used in the purpose of convergence is thus reduced overall. Program complexity will be reduced, and issues like time intensiveness because of a larger data set are reduced. Now, BN by itself is not that computationally intensive and the per iteration time is reduced. This is attributed to the

limited memory bandwidth of batch normalization. Two passes are computed with the input data. The first pass will basically compute the statistics of the batch being read. The second pass is the normalization of the output of the layer. In research works, it has been identified that the use of BN will reduce the training time to around 1/4 of the total training time of the ResNet-50 network [22] [23]. It has been of much use in the ImageNet classification because of how it reduces the training time. Now that convolutions are going to be optimized more, the use of BN will help reduce the run time even more. The data set was finally normalized. Normalization is necessary to ensure that all the data are on the same scale and this usually improves the performance. For the Fashion MNIST data set, the normalization for training data-set and the evaluation data set would be handled as below.

```
train_data = train_data/np.float32(255)
train_labels = train_labels.astype(np.int32)
eval_data = eval_data/np.float32(255)
eval_labels= eval_labels.astype(np.int32)
```

4.2 Training Data

In the training process, training data has to be loaded. The images set is initialized with numbers of images, image size, number of channels, and the labels set is initialized with number of images, type integer. Objects or images included in the training process are shirt, coat, pullover, trouser, sandal, boot and bag. The 'load training-data' is the function to be called when the CNN is trained [28] [16].

4.3 Best Model for Training and Testing (Results and Analysis)

The first step in testing is the initialization for Epochs, and the labels for the original run, the first stage, the test run for labels top, and labels shoe. Epochs value is set as 10, and the labels assigned for the images for all, are as follows, 0: "Shirt", 1:"Trouse", 2: "Pullover", 3: "Coat", 4:

"Sandal", 5: "Bag", 6: "Boot". For first stage, it is 0: "Top", 1: "Trouser", 3: "Shoe", 4: "Bag". For stage with just the label-top, the assignment is, 0: "Shirt", 1: "Pullover", 2: "Coat". For the label shoe, the assignment is 0: "Sandal", 1: "Boot". A series of functions are run for training data, using training labels, preparing the MNIST data for the first stage, preparing the data for second stage and so on. Initialization for the functions are the number of classes of first stage equal to 4, and the number of classes of second stage equal to 3 for top, and the number of classes of second stage equal to 2 for shoes, and the number of channels equal to the MNIST data, the number of classes equal to 7.

4.4 Modified Model for Each Stage

From the resultant data and analysis, it was necessary to consider whether the model had to be changed, such as whether the layer size had to be changed because of over-fitting issues. Now, the kernel size in the layer is what connects the input with output [29] [30]. A kernel size for the layer as defined in the above chapter is defined by size = n-inputs * n-outputs. A bias term is associated with the output as bis size = n-outputs. Now this term is much smaller than the kernel size and hence is not tracked. Now for the convolutional layer, as processing was conducted through the stages, a fully connected layer with pooling layers was first implemented, with the pixels as input and the channels as depth. With a sample of a squared convolutional layer with size k in theory, similar to what has been used in the current code, it could be said that the kernel size is $k^2 \times c^2$ and for a fully connected layer, the size is defined for the same and output as $(l^2 \times c)^2$with kernel size ratio defined as l^4/k^2. Operating with these image data and size preferences, it could be said that for small images in the range of 32x32(MNIST, CIFAR 10), the kernel size will already be huge, and it is in the range of 11x11. The factor here is 8,600. However, the 32x32 image is not a much realistic one. The more realistic one is the 224x224 image (ImageNet). The kernel size here is more appropriate at 3x3 and for this kernel size, the factor is 270,000,000. Computing ratio for both is considered with l^2 output pixels and computing ratio at l^2/k^2. The yield for the big image is therefore better at 5,500. Therefore, in research and in the current codes that was used in this work, one

could identify how the size of the convolutional layer had an effect on the number of weights that were used. It had the potential to drastically reduce the number of weights used in the network. It was inferred that the number of weights is dependent on kernel size therein and not as much on the input size. In the case of image processing, when the impact of number of weights on the kernel size is given more importance than the input size, then memory usage is reduced, and computations are improved generally. Now in the ResNet that was coded for and trained, it was identified that there were no issues of over-fitting and the current kernel size is considered as optimal for reducing memory usages and improving computations. Over-fitting is usually the drawback from having a large number of parameters. This means that when a machine learning algorithm learns a lot from the training data, then it loses the ability to generalize [22] [31]. This introduces the situation where the algorithm does not fare well in testing and does not scale for complex data. If issues of over-fitting are observed in CNN, a technique called dropout is used. Dropout is basically a form of regularization.

Dropout is a regularization technique for neural network models proposed by Srivastava, et al. in their 2014 paper 'Dropout: A Simple Way to Prevent Neural Networks from over-fitting' [32]. "Dropout is a technique where randomly selected neurons are ignored during training. They are "dropped-out" randomly. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass. As a neural network learns, neuron weights settle into their context within the network. Weights of neurons are tuned for specific features providing some specialization. Neighboring neurons become to rely on this specialization, which if taken too far can result in a fragile model too specialized to the training data. This reliant on context for a neuron during training is referred to complex co-adaptations" [32]. Therefore in dropout, when neurons are dropped out in the training process, then what happens is that other neurons tend to step in and they handle the representations of the dropped out or the missing neurons. This activity of stepping in is needed to make predictions for the missing neurons. This results in independent internal representations

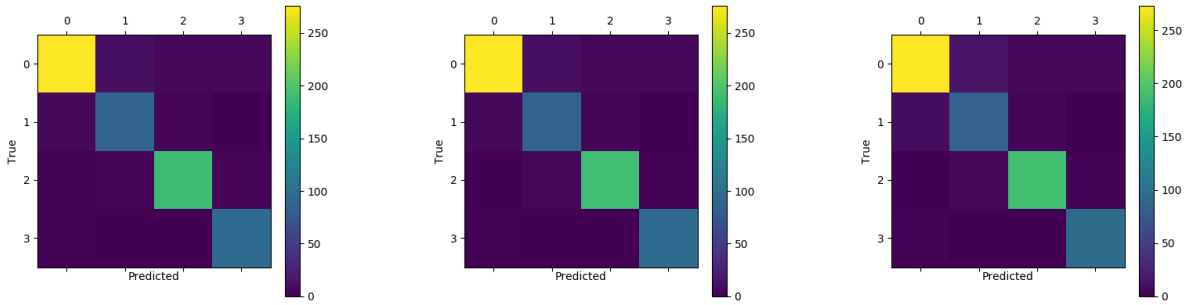
or new learning actions within the network. Instead of the machine learning everything from the training data leading to over-fitting, the drop out aspect will lead to multiple independent internal representations within the network keeping the training process alive. As a result of this action, the network is now less attuned or sensitive to the weights of neurons or test data, and therefore, the network becomes more capable in generalization. In the future, there are less likely changes for the network to over-fit with training data.

In the case of the ResNet operation, test data used in 100, and real data size is contained and optimized at 224*224. Therefore, there are no issues in size and over-fitting, and hence there is no reason to change the convolutional layer size or model. There are, however, some best practices as understood from existing works on CNN that could be applied in future work. For instance, in the case of convolutional neural network layers, the deeper the network the better. The width of the network is not of much importance compared to the depth. It is the wider networks that have one too many weights to contend with and hence lead to issues of over-fit. Comparatively, the deeper networks have features that make it more suitable for extraction and analysis. When all layers are made lighter, then the memory usage and computational speed performance is enhanced. As presented earlier, it is the kernel size that affects this.

4.4.1 Changing Pooling Layer For Each Stage

Any convolutional network will have a convolutional layer, a pooling layer and a fully connected layer, and these stacks form the CNN network (as discussed in the model representation of the earlier chapter). A pooling layer as introduced in ResNet architecture is a layer between successive convolutional layers in the architecture. The function of the pooling layer is to work on reducing spatial size of representation. There are always risks of over-fitting in the network and this is controlled by the presence of the pooling layer. The pooling layer essentially reduces the parameters and the computations within the network by operating independently on the depth slice of input [22] [31]. The MAX operation is used from the pooling layer in order to conduct spatial re-sizing.

Usually, a pooling layer filter is presented with size 2*2 and is applied with at least 2 down samples along width and height. All the MAX operations of the pooling layer will have a maximum of 4 numbers and the depth dimension would remain unchanged. The plotting of the confusion matrix is generated by the following steps of the code with the predicted values and true values for the matrix.There are three confusion matrix generated for the one pooling layer, the two pooling layer and the four pooling layer below. The accuracy of one pooling layer is 93.00%, the accuracy of Two pooling layer is 93.14%, and the accuracy of four pooling layer is 92.28%.



(a) (b) (c)

Figure 4.1: Changing pooling layer result from one to four layers.

Form the each confusion plot, we find when adding two pooling layer in the stage one, the accuracy will be highest. Therefore, we use two pooling layer model to the finally combine, and the combined confusion matrix is presented below, with true values and predicted values. The combined accuracy is 90.28%.

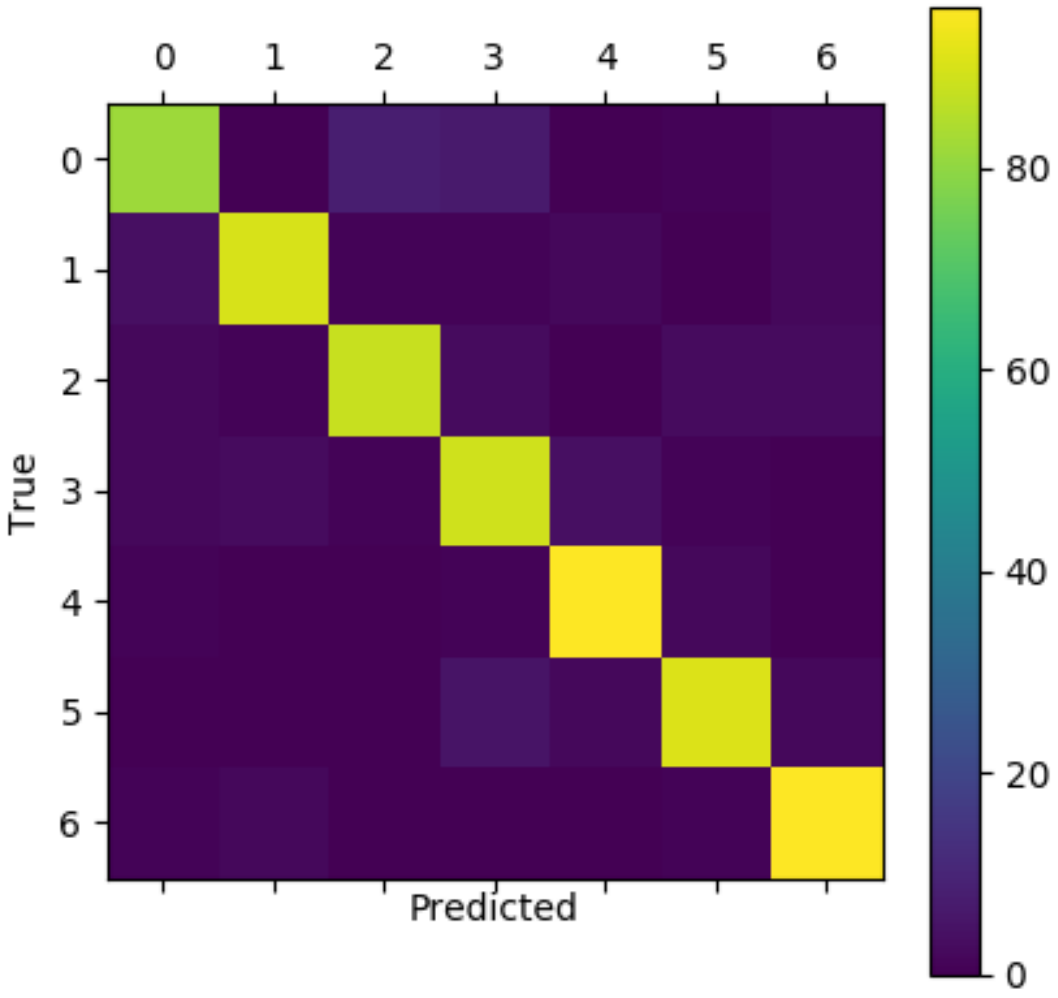


Figure 4.2: Two pooling layer of two-stage confusion table

CHAPTER 5. CONCLUSIONS

The aim of this work was to analyze different models of CNN to identify which of the CNN presents better accuracy in classification and identification. The models selected for evaluation are LeNet-5, AlexNet, VGG-16 and ResNet. Tensor flow has been used for creating the training process for the distance function. The first part of the work was focused on a comparative analysis of different CNNS for the same data set. Data set for the work was downloaded from the Kaggle website, which is an open source website for CNN data sets. The Fashion MNIST data set was downloaded, and CNN training and testing was carried out. Process steps like data preparation are presented in the work. It is identified that ResNet by far presents the most validation accuracy rate, as compared to LeNet-5, AlexNet, and VGG-16. In identifying the best CNN to use for this data set, the research work also presented the limitations of this research. Newer data sets and other real time fashion data sets will have to be trained before they can show similar results. Even then there could be issues of data standardization and these concerns were discussed as well. Data analysis with focus on the pooling layers, and a stage-based analysis was conducted as well. The second part of the work was focused on testing the best model. As the comparative analysis showed ResNet as being best model, the next part of the work was focused on testing the performance of the new best model. This was done by preparing the data, training the ResNet, and then using the trained model for testing. Discussion was focused on the modified models of each stage and changing the pooling layer.

Bibliography

[1] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, 2015, pp. 1135–1143.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[3] M. Hollemans, "Convolutional neural networks on the iphone with vggnet," *URl: http://machinethink. net/blog/convolutional-neural-networkson-the-iphone-with-vggnet*, 2016.

[4] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[5] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2012.

[6] A. R. Pope and D. G. Lowe, "Probabilistic models of appearance for 3-d object recognition," *International Journal of Computer Vision*, vol. 40, no. 2, pp. 149–167, 2000.

[7] D. G. Lowe, "Local feature view clustering for 3d object recognition," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1. IEEE, 2001, pp. I–I.

[8] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1717–1724.

[9] J. Le, "The 4 convolutional neural network models that can classify your fashion images," https://towardsdatascience.com/the-4-convolutional-neural-network-models-that-can-classify-your-fashion-images-9fe7f3e5399d, accessed October 6, 2018.

[10] A. Schindler, T. Lidy, S. Karner, and M. Hecker, "Fashion and apparel classification using convolutional neural networks," *arXiv preprint arXiv:1811.04374*, 2018.

[11] K. Mikolajczyk and C. Schmid, "An affine invariant interest point detector," in *European conference on computer vision*. Springer, 2002, pp. 128–142.

[12] Y. Bengio, P. Simard, P. Frasconi *et al.*, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[13] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4293–4302.

[14] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE transactions on neural networks*, vol. 8, no. 1, pp. 98–113, 1997.

[15] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[16] M. Brown and D. G. Lowe, "Invariant features from interest point groups." in *BMVC*, vol. 4, 2002.

[17] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *Computer vision and Image understanding*, vol. 106, no. 1, pp. 59–70, 2007.

[18] X. Han, Y. Zhong, L. Cao, and L. Zhang, "Pre-trained alexnet architecture with pyramid pooling and supervision for high spatial resolution remote sensing image scene classification," *Remote Sensing*, vol. 9, no. 8, p. 848, 2017.

[19] A. Dosovitskiy and T. Brox, "Generating images with perceptual similarity metrics based on deep networks," in *Advances in neural information processing systems*, 2016, pp. 658–666.

[20] Y. Tian, P. Luo, X. Wang, and X. Tang, "Deep learning strong parts for pedestrian detection," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1904–1912.

[21] S. Lu, Z. Lu, and Y.-D. Zhang, "Pathological brain detection based on alexnet and transfer learning," *Journal of computational science*, vol. 30, pp. 41–47, 2019.

[22] R. U. Khan, X. Zhang, and R. Kumar, "Analysis of resnet and googlenet models for malware detection," *Journal of Computer Virology and Hacking Techniques*, vol. 15, no. 1, pp. 29–37, 2019.

[23] Z. Wu, C. Shen, and A. Van Den Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *Pattern Recognition*, vol. 90, pp. 119–133, 2019.

[24] C. Shorten, "Introduction to resnets," https://towardsdatascience.com/introduction-to-resnets-c0a830a288a4, accessed July 24, 2018.

[25] Z. Lu, X. Jiang, and A. Kot, "Deep coupled resnet for low-resolution face recognition," *IEEE Signal Processing Letters*, vol. 25, no. 4, pp. 526–530, 2018.

[26] Y. Cui, F. Zhou, J. Wang, X. Liu, Y. Lin, and S. Belongie, "Kernel pooling for convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2921–2930.

[27] B. Li and Y. He, "An improved resnet based on the adjustable shortcut connections," *IEEE Access*, vol. 6, pp. 18 967–18 974, 2018.

[28] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[29] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, no. 10, pp. 1533–1545, 2014.

[30] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[31] M. Sun, Z. Song, X. Jiang, J. Pan, and Y. Pang, "Learning pooling for convolutional neural network," *Neurocomputing*, vol. 224, pp. 96–104, 2017.

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.