**FASHION IMAGE CLASSIFICATION**

**Report submitted to**

**Techno India University, West Bengal**

**for the partial fulfillment**

**Of**

**Bachelor of Technology (B. Tech.)**

**degree in**

**Computer Science & Engineering**

**By**

**MD ASIF KHAN - 161001001026**

**SOURADEEP KUMAR BASU - 161001001025**

**RAMANTI SHAW – 161001001030**

**RAJDEEP DE – 171001011002**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING TECHNO INDIA UNIVERSITY, WEST BENGAL, SALT LAKE, KOLKATA – 700091, INDIA**

June 2020

CERTIFICATE

This is to certify that the Dissertation Report entitled, “FASHION IMAGE CLASSIFICATION” submitted by “MD ASIF KHAN, SOURADEEP KUMAR BASU, RAMANTI SHAW & RAJDEEP DE” to Techno India University, Kolkata, India, is a record of Bonafede Project work carried out by them under my supervision and guidance and is worthy of consideration for the award of the degree of Bachelor of Technology (B.Tech) in Computer science & Engineering.

**Approved By:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Supervisor(s)

Date:

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

HOD, CSE, Techno India University

Date:

**ACKNOWLEDGEMENT**

We would first like to thank our thesis supervisor Dr. Jayanta Poray, Associate Professor (Teacher in Charge), CSE Dept, TIU. He helped us whenever we ran into a trouble spot or had a question about our research or writing.

We take this opportunity to express gratitude to all of the Department faculty members for their help and support.

We also thank our parents for the unceasing encouragement, support and attention and also sense of gratitude to one and all, who directly or indirectly, have bestowed their hand in this thesis.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

MD ASIF KHAN

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SOURADEEP KUMAR BASU

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

RAMANTI SHAW

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

RAJDEEP DE

**TABLE OF CONTENTS**

**ABSTRACT** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .2

**CHAPTER 1. INTRODUCTION** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

**CHAPTER 2. BACKGROUND** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

**CHAPTER 3. MODEL EXPLORATION** . . . . . . . . . . . . . . . . . . . . . . . . . . . .6

3.1 Kaggle Data . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

3.2 Data Preparation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .7

3.3 Training Data . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

3.4 Models for Training and Testing . . . . . . . . . . . . . . . . . . . . . . . .8

3.4.1 3-Layer CNN . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

3.4.2 VGG-19 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12

3.5 Activation Function, Optimizer & Loss function. . . . . . . . . . . . . . . . . . . 14

3.5.1 RELU. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14

3.5.2 SoftMax. . . . . . . . . . . . . . . . . . . . . . . . . . . … … …14

3.5.3 Loss Function. . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

3.5.4 Adam Optimizer. . . . . . . . . . . . . . . . . . . . . .. . 15

Data Flow Diagram. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17

3.6 Results And Analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18

**CHAPTER 4. MODEL APPLICATION** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24

4.1 Data Preparation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24

4.2 Training Data . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24

4.3 Best Model for Training and Testing (Results and Analysis) . . . . . . . . 25

**CHAPTER 5. CONCLUSIONS** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27 **BIBLIOGRAPHY** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28

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 images, associated with a label from 10 classes. In this report, the accuracy of Some popular CNN models that are VGG-19, 3-layer CNN for classifying MNIST-fashion data revealed that 3-layer CNN was the best suited for the selected dataset. The training process has been coded with Keras with Tensor flow backend. 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). It is applied to image processing tasks, human action recognition, fashion image classification and medical image classification.

The aim of this work is to analyze a few different models of CNN namely, the VGG-19 and 3-layer CNN. Analysis is carried out with a fashion dataset from the Kaggle website. A training process for this process coded with TensorFlow. The results of the CNN models are then analyzed and evaluated to identify the best model. 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.

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. 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". 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.

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 × 5 × n, here ‘n’ refers to the number of convolutions in the CNN.

Here 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 categories, 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 Bewakoof, Farfetch, Zalando 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 image processing with DNN are that of the varying styles, the textures, the shapes and colors. 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. 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 differentiating persons from products, gender.

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. It is a data set from Zalando originally and each of the gray scale images has the 10 classes.

3.1 Kaggle Data

Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255. The training and test data sets have 785 columns. The first column consists of the class labels (see above), and represents the article of clothing. The rest of the columns contain the pixel-values of the associated image.

* To locate a pixel on the image, suppose that we have decomposed x as x = i \* 28 + j, where i and j are integers between 0 and 27. The pixel is located on row i and column j of a 28 x 28 matrix.
* For example, pixel31 indicates the pixel that is in the fourth column from the left, and the second row from the top, as in the ascii-diagram below.

**Labels**

Each training and test example is assigned to one of the following labels:

* 0 T-shirt/top
* 1 Trouser
* 2 Pullover
* 3 Dress
* 4 Coat
* 5 Sandal
* 6 Shirt
* 7 Sneaker
* 8 Bag
* 9 Ankle boot

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, Keras, Numpy, Matplotlib.pylot, Seaborn.
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 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 8 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 (60000, 785) and (10000, 785).
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. For instance, in the VGG 19, 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.

3.4 Models for Training and Testing

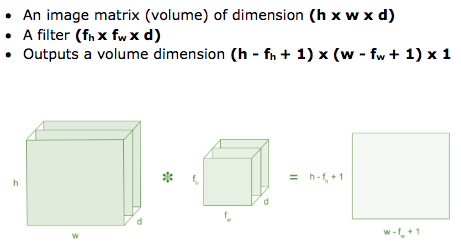
**3.4.1 3-Layer CNN**

A **Convolutional Neural Network (ConvNet/CNN)** is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics. Here we use 3-layer CNN model, it has 3 layers and it is based on Fully Connected CNN.



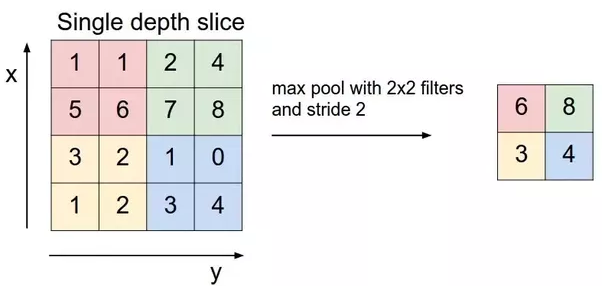
**Convolution Layer**

Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.



**Pooling Layer**

Pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling also called subsampling or down sampling which reduces the dimensionality of each map but retains important information. Max pooling takes the largest element from the rectified feature map.



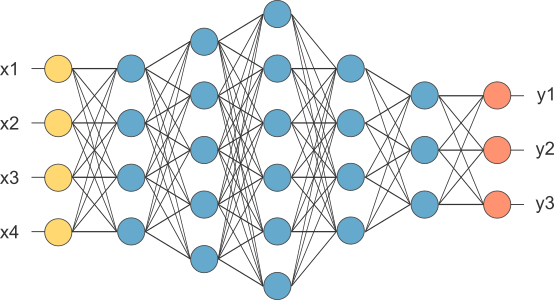
**Flatten Layer**

2D convolution layers processing 2D data (for example, images) usually output a tridimensional tensor, with the dimensions being the image resolution (minus the filter size -1) and the number of filters. If you use a layer with N filters of size s on WxH images, the result will be of dimension **(W-(s-1), H - (s-1), N).** You have **N** images of resolution **(W-(s-1), H - (s-1)).** This structure is important if you want to chain convolution layers together or with other layers that perform a spatial treatment (pooling, upscaling, etc.).But if you want to do classification, in the last steps of your network, you usually want to use fully connected layers that don’t take any structure (spatial or otherwise) into account for processing, so you just want the output of the last convolution layers to be considered as a large piece of unstructured data.

That’s what “flattening” means. It breaks the spatial structure of the data and transforms your tridimensional **(W-(s-1), H - (s-1), N)** tensor into a vector of size **(W-(s-1))x(H - (s-1))x N**.

**Fully Connected Layer**

The layer we call as FC layer, we flattened our matrix into vector and feed it into a fully connected layer like a neural network.



In the above diagram, the feature map matrix will be converted as vector (x1, x2, x3, …). With the fully connected layers, we combined these features together to create a model. Finally, we have an activation function such as SoftMax or sigmoid to classify the outputs.

**The Class Definition for CNN**

**from keras.models import Sequential**

**from keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout**

**cnn\_model = Sequential()**

**cnn\_model.add(Conv2D(Filters,Kernel\_size=’’, input\_shape = , activation =’’))**

**cnn\_model.add(MaxPooling2D(pool\_size = ))**

**cnn\_model.add(Dropout=)**

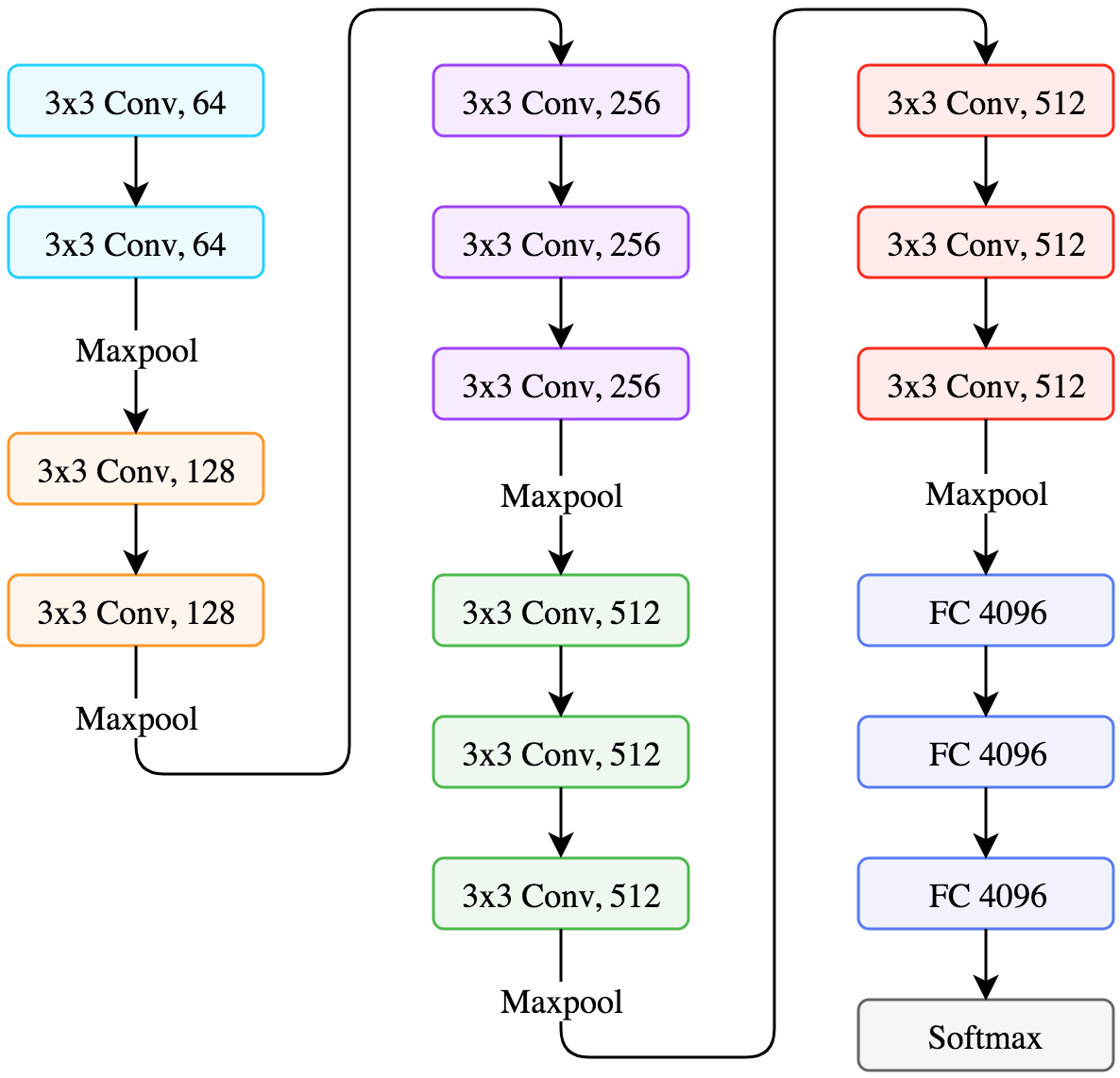
**cnn\_model.add(Flatten ())**

**cnn\_model.add(Dense(units=, activation =''))**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| **Sequential** | A Sequential model is appropriate for**a plain stack of layers**where each layer has**exactly one input tensor and one output tensor.** |
| **filters** | It is the numbers of filters that convolutional layers will learn from. It is an integer value and also determines the number of output filters in the convolution. |
| **Kernel\_size** | This parameter determines the dimensions of the kernel. Common dimensions include 1×1, 3×3, 5×5, and 7×7 which can be passed as (1, 1), (3, 3), (5, 5), or (7, 7) tuples. |
| **input\_shape** | It is the input shape of the image with channels like (28, 28, 1). |
| **activation** | The activation parameter to the Conv2D class is simply a convenience parameter which allows you to supply a string, which specifies the name of the activation function you want to apply after performing the convolution. |
| **Pool\_size** | This parameter is used in max pooling  Layer. It needs the poolsize and it takes integer values. Pool size like (2,2) or (3,3) passed as tuples. |
| **Dropout** | A Simple Way to Prevent Neural Networks from Overfitting. It takes float value we take it between 0.2 to .3. |

**3.4.2 VGG-19**

VGG is a convolutional neural network from researchers at Oxford’s Visual Geometry Group, hence the name VGG. It was the runner up of the ImageNet classification challenge with 7.3% error rate. ImageNet is the most comprehensive hand-annotated visual dataset, and they hold competitions every year where researchers from all around the world compete. All the famous CNN architectures make their debut at that competition. Among the best performing CNN models, VGG is remarkable for its simplicity. Let’s take a look at its architecture.



VGG is a 19-layer neural net, not counting the maxpool layers and the SoftMax at the end.

**The Class Definition for VGG-19**

**from keras.applications import VGG19**

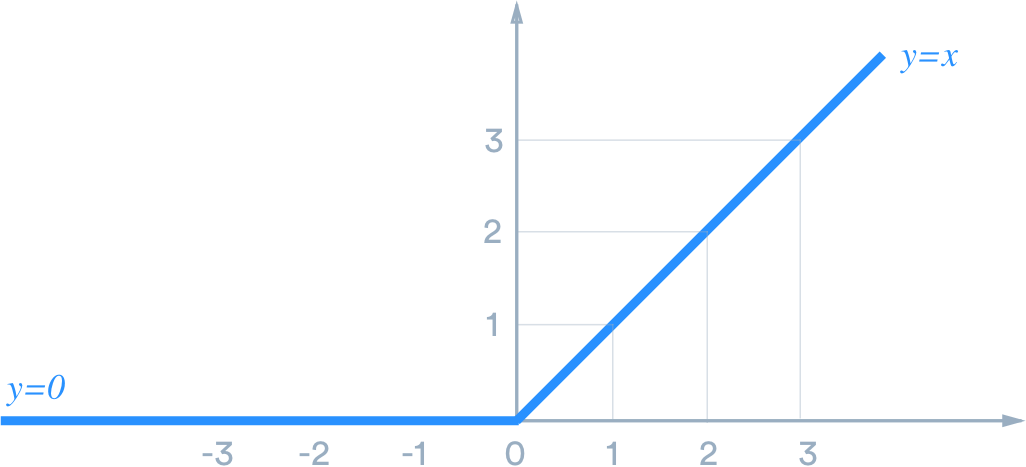
**vgg19 = VGG19(weights='', include\_top=, input\_shape =, classes = )**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| **weights** | weights specify the weight checkpoint from which to initialize the model. |
| **include\_top** | It refers to including (or not) the densely connected classifier on top of the network. By default, this densely connected classifier corresponds to the 1,000 classes from ImageNet. It takes Boolean values like True or False. |
| **input\_shape** | It is the input shape of the image with channels like (48, 48, 3). |
| **classes** | Optional number of classes to classify images. |

3.5 Activation Functions, Optimizer and Loss Function

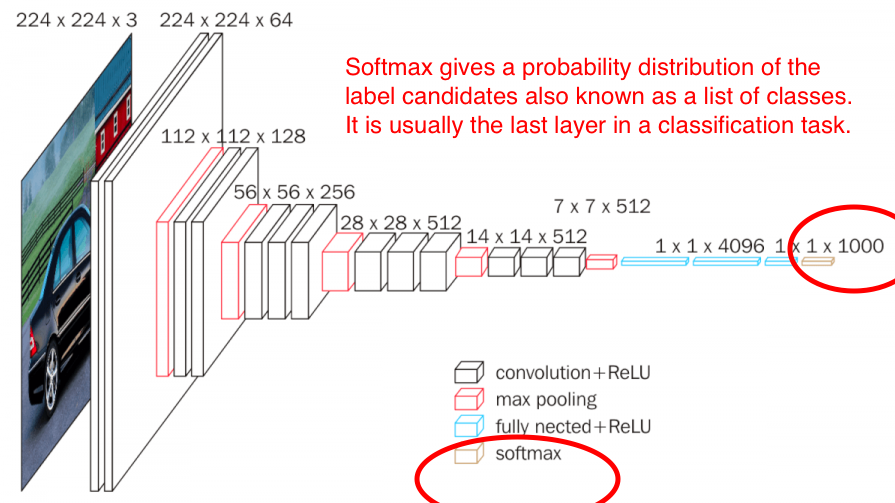
**3.5.1 ReLU (Rectified Linear Unit)**

ReLU stands for rectified linear unit, and is a type of activation function. Mathematically, it is defined as y = max (0, x). ReLU is the most commonly used activation function in neural networks, especially in CNNs. ReLU is linear (identity) for all positive values, and zero for all negative values.



**3.5.2 SoftMax**

SoftMax turn logits (numeric output of the last linear layer of a multi-class classification neural network) into probabilities by take the exponents of each output and then normalize each number by the sum of those exponents so the entire output vector adds up to one — all probabilities should add up to one. Cross entropy loss is usually the loss function for such a multi-class classification problem. SoftMax is frequently appended to the last layer of an image classification network.



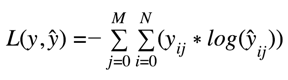
The Softmax function allows us to express our inputs as a discrete probability distribution. Mathematically, this is defined as follows:

**for I = 1, 2 ……**

* Negative inputs will be converted into nonnegative values, thanks to the exponential function.
* Each input will be in the interval (0,1).
* As the *denominator* in each Softmax computation is the same, the values become proportional to each other, which makes sure that together they sum to 1.

**3.5.3 Loss Function**

When doing multi-class classification, categorical cross entropy loss is used a lot. Here in our model we use sparse categorical cross entropy as loss function, the only difference between sparse categorical cross entropy and categorical cross entropy is the format of true labels. When we have a single-label, multi-class classification problem, the labels are mutually exclusive for each data, meaning each data entry can only belong to one class.



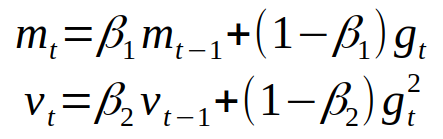
where ŷ is the predicted value. Categorical crossentropy will compare the distribution of the predictions (the activations in the output layer, one for each class) with the true distribution, where the probability of the true class is set to 1 and 0 for the other classes. To put it in a different way, the true class is represented as a one-hot encoded vector, and the closer the model’s outputs are to that vector, the lower the loss.

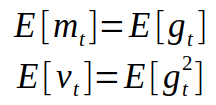
**3.5.4 Adam Optimizer**

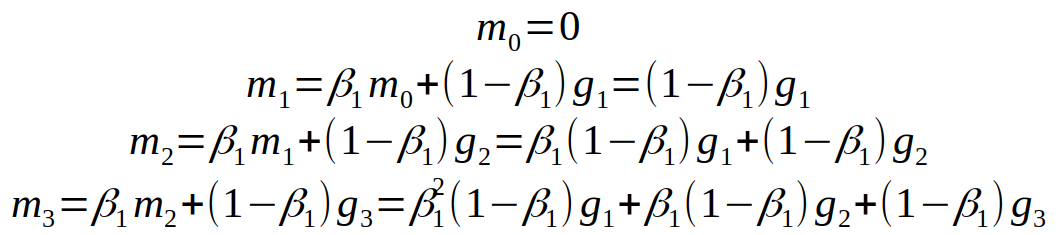
Adam (short for Adaptive Moment Estimation) is an update to the RMS Prop optimizer. In this optimization algorithm, running averages of both the gradients and the second moments of the gradients are used. Given parameters  {\displaystyle w^{(t)}}and a loss function  {\displaystyle L^{(t)}}LL,where {\displaystyle t}t indexes the current training iteration (indexed at {\displaystyle 0}0), Adam's parameter update is given by:

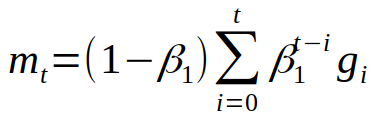


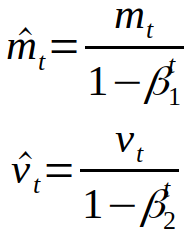
Where m = Moment and X = Random variable

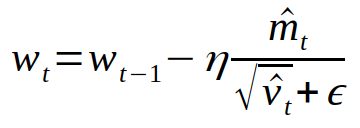






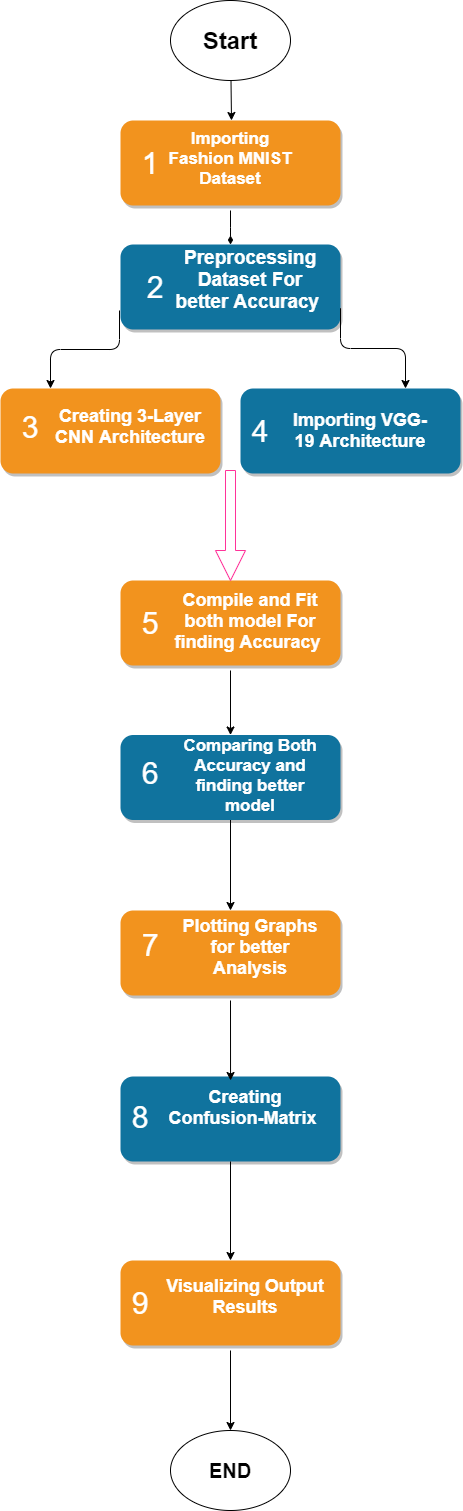






Where w is model weights, eta (look like the letter n) is the step size (it can depend on iteration). And that’s it, that’s the update rule for Adam.

Data Flow Diagram



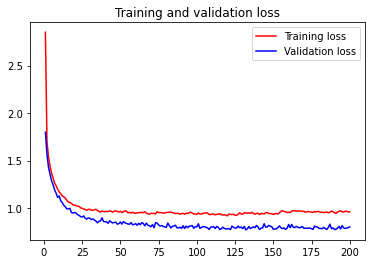
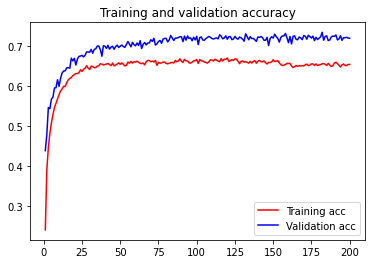
3.6 Result and Analysis

The results for each of the CNN are compared and analyzed in this section.

**For VGG-19**, the input and the training and prediction data set are as follows. It can be observed that there are issues in prediction 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 VGG-19. The code for VGG-19 includes just passed 4 arguments to the constructor:

* weights specify the weight checkpoint from which to initialize the model.
* include\_top refers to including (or not) the densely connected classifier on top of the network. By default, this densely connected classifier corresponds to the 1,000 classes from ImageNet. Because I intend to use my own densely connected classifier (with only 10 classes), I don’t need to include it.
* input\_shape: optional shape tuple, only to be specified if include\_top is False.
* classes: optional number of classes to classify images into, only to be specified if include\_top is True, and if no weights argument is specified.

****

****

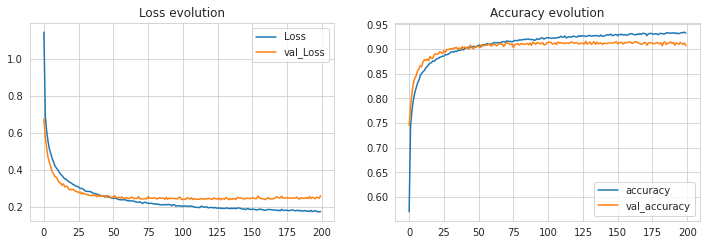
Although the lines of loss and accuracy for both training and validation follow similar trends, there're some space between their values.

**For 3-Layer CNN**, the input and the training and prediction data set are as follows. It can be observed that there are issues in prediction for coat and for pullover etc. Most of the images for Shirt, boot, bag and pant are categorized correctly, but some other images were wrongly identified.



This CNN takes as input tensors of shape *(image\_height, image\_width, image\_channels)*. In this case, I configure the CNN to process inputs of size *(28, 28, 1)*, which is the format of the FashionMNIST images. I do this by passing the argument *input\_shape= (28, 28, 1)* to the first layer.

* The 1st layer is a *Conv2D* layer for the **convolution** operation that extracts features from the input images by sliding a convolution filter over the input to produce a feature map. Here I choose feature map with size 3 x 3.
* The 2nd layer is a *MaxPooling2D* layer for the **max-pooling** operation that reduces the dimensionality of each feature, which helps shorten training time and reduce number of parameters. Here I choose the pooling window with size 2 x 2.
* To combat overfitting, I add a *Dropout* layer as the 3rd layer, a powerful regularization technique. **Dropout** is the method used to reduce overfitting. It forces the model to learn multiple independent representations of the same data by randomly disabling neurons in the learning phase. In this model, dropout will randomly disable 20% of the outputs.
* I repeat these steps to add more hidden layers: 2 *Conv2D* layers, 1 *MaxPooling2D* layers, and 2 *Dropout* layers.
* The next step is to feed the last output tensor into a stack of *Dense* layers, otherwise known as **fully-connected** layers. These densely connected classifiers process vectors, which are 1D, whereas the current output is a 3D tensor. Thus, I need to **flatten** the 3D outputs to 1D, and then add 2 *Dense* layers on top.
* I add another *Dropout* layer between these 2 *Dense* layers to disable 30% of the outputs.
* I do a 10-way classification (as there are 10 classes of fashion images), using a final layer with 10 outputs and a SoftMax activation. **SoftMax** activation enables me to calculate the output based on the probabilities. Each class is assigned a probability and the class with the maximum probability is the model’s output for the input.

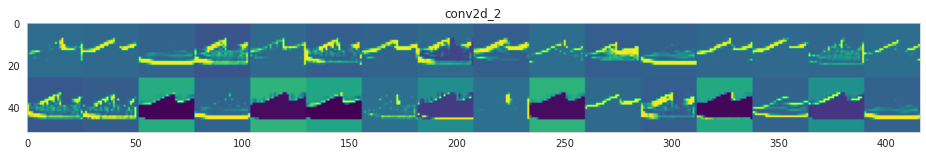


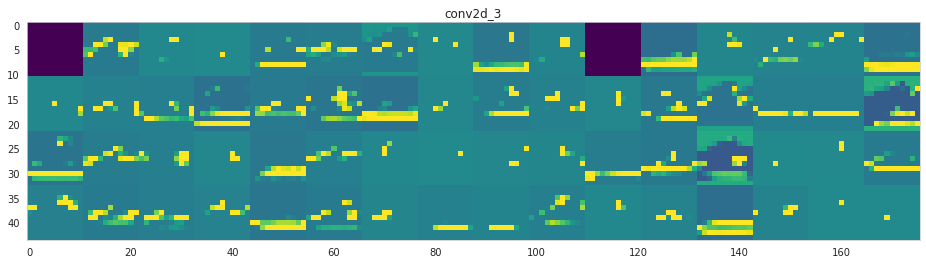
We see the lines of loss and validation loss and accuracy and validation accuracy both follow similar trends, there're small space between values. The plots look decent training curve just follow validation curve.

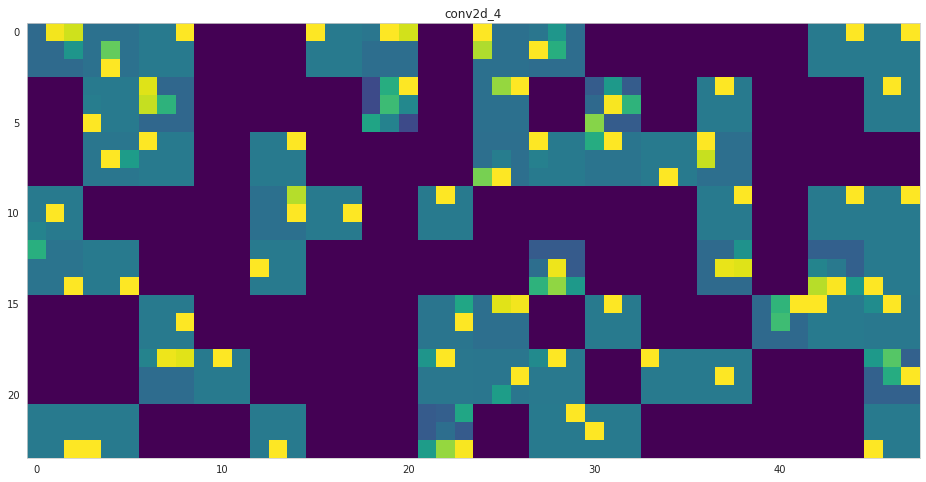
In compiling I choose **Sparse\_categorical\_crossentropy** as the loss function (which is relevant for multiclass, single-label classification problem) and **Adam** optimizer.

* The cross-entropy loss calculates the error rate between the predicted value and the original value. Categorical is used because there are 10 classes to predict from. If there were 2 classes, I would have used binary\_crossentropy.
* The Adam optimizer is an improvement over SGD (Stochastic Gradient Descent). The optimizer is responsible for updating the weights of the neurons via backpropagation. It calculates the derivative of the loss function with respect to each weight and subtracts it from the weight. That is how a neural network learns.
* 163,850 parameters are available to be trained.
* The output of the *Conv2D* and *MaxPooling2D* layers are 3D tensors of shape *(height, width, channels)*.
* The number of channels is controlled by the 1st argument passed to the *Conv2D* layer (32).
* The (1, 1, 128) outputs from the 5th *Dropout* layer are flattened into vectors of shape (128,) before going through 2 *Dense* layers.
* Train the model with batch size of 512 and 200 epochs on both training and validation data.

This is the visualization of how convolution layers process an image and how they classify them,







Here are both models test accuracy and test loss,

|  |  |  |
| --- | --- | --- |
| **Name** | **Test Accuracy** | **Test Loss** |
| VGG-19 | **73.79** | **0.757** |
| 3-Layer CNN | **91.42** | **0.247** |

We see that accuracy rate of VGG-19 is lower so 3-layer CNN best model for the dataset.

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.  F1 Score is the weighted average of Precision and Recall.

**precision recall f1-score support**

**Class 0 0.63 0.70 0.66 1000**

**Class 1 0.94 0.92 0.93 1000**

**Class 2 0.57 0.59 0.58 1000**

**Class 3 0.64 0.84 0.73 1000**

**Class 4 0.61 0.59 0.60 1000**

**Class 5 0.91 0.84 0.88 1000**

**Class 6 0.44 0.31 0.36 1000**

**Class 7 0.82 0.84 0.83 1000**

**Class 8 0.90 0.82 0.86 1000**

**Class 9 0.84 0.89 0.86 1000**

**accuracy 0.73 10000**

**macro avg 0.73 0.73 0.73 10000**

**weighted avg 0.73 0.73 0.73 10000**

In above this precision and recall table is for VGG-19 we see that the model underperforms for Class 2 and 6. It lacks precision for class 0 and 4, additionally.

**precision recall f1-score support**

**Class 0 0.91 0.83 0.87 1000**

**Class 1 0.99 0.99 0.99 1000**

**Class 2 0.91 0.81 0.85 1000**

**Class 3 0.91 0.93 0.92 1000**

**Class 4 0.86 0.82 0.84 1000**

**Class 5 0.99 0.97 0.98 1000**

**Class 6 0.68 0.84 0.75 1000**

**Class 7 0.95 0.96 0.95 1000**

**Class 8 0.99 0.98 0.98 1000**

**Class 9 0.95 0.96 0.96 1000**

**accuracy 0.91 10000**

**macro avg 0.91 0.91 0.91 10000**

**weighted avg 0.91 0.91 0.91 10000**

In above this precision and recall table is for 3-Layer CNN we see that the model underperforms for class 6 in terms of both precision and recall. For class 0, the classifier is slightly lacking precision; whereas for class 2 and 4, it is slightly lacking recall.

CHAPTER 4. MODEL APPLICATION

With the Kaggle data, the results of comparison now indicate that the 3-layer CNN model far better than the rest of the models. Now in this chapter on testing the best model with real data, 3-layer CNN is used.

4.1 Data Preparation

First split the original training data (60,000 images) into 80% training (48,000 images) and 20% validation (12000 images) optimize the classifier, while keeping the test data (10,000 images) to finally evaluate the accuracy of the model on the data it has never seen. This helps to see whether we over-fitting on the training data and whether we should lower the learning rate and train for more epochs if validation accuracy is higher than training accuracy or stop over-training if training accuracy shift higher than the validation.

**X\_train, X\_validate, y\_train, y\_validate = train\_test\_split (X\_train, y\_train, test\_size = 0.2, random\_state = 12345)**

After loading and splitting the data, I preprocess them by reshaping them into the shape the network expects and scaling them so that all values are in the [0, 1] interval. Previously, for instance, the training data were stored in an array of shape (60000, 28, 28) of type uint8 with values in the [0, 255] interval. I transform it into a float32 array of shape (60000, 28 \* 28) with values between 0 and 1.

**X\_train = X\_train.reshape(X\_train.shape[0], \*(28, 28, 1))**

**X\_test = X\_test.reshape(X\_test.shape[0], \*(28, 28, 1))**

**X\_validate = X\_validate.reshape(X\_validate.shape[0], \*(28, 28, 1))**

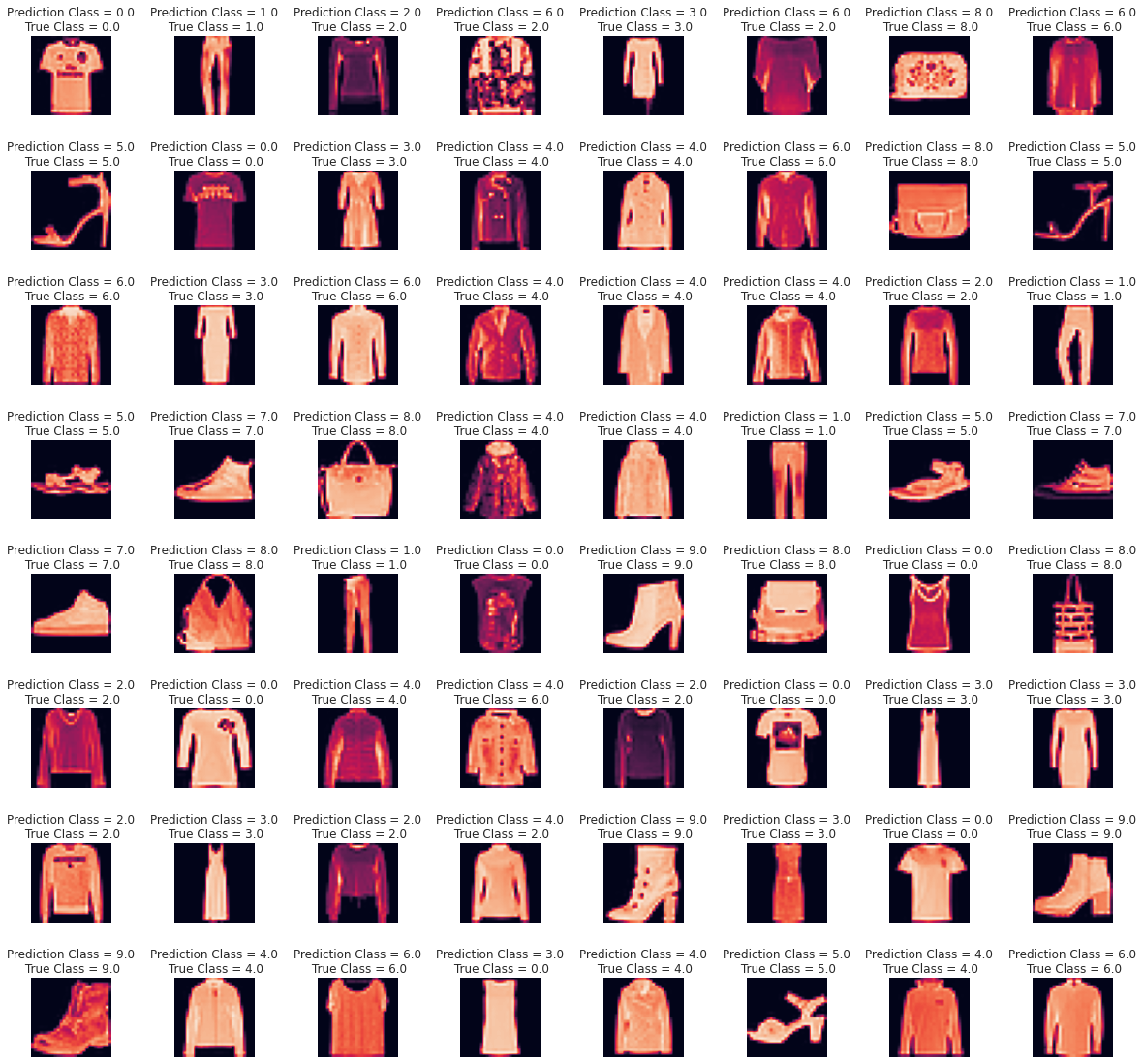
4.2 Training Data

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.

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

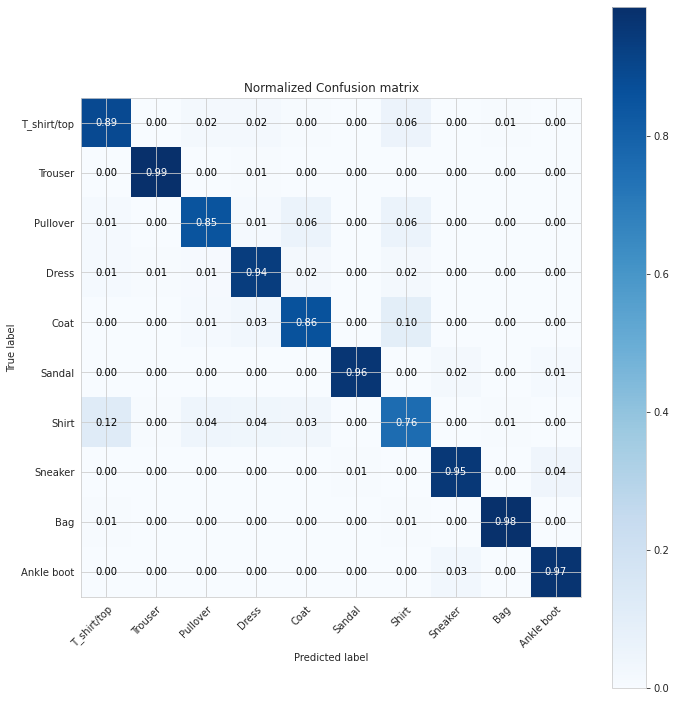
The first step in testing is the initialization for Epochs, and the labels for the original run, the first stage, the test run for labels top, and labels shoe. Epochs value is set as 200, and the labels assigned for the images for all, are as follows, 0: “Shirt”, 1: “Trouser”, 2: “Pullover”, 3: “Dress”, 4: “Coat”, 5: “Sandal”, 6: “Shirt”, 7: “Sneaker”, 8: “Bag”, 9: “Ankle-Boot”.

After that we train the model using CNN and the output is shown below,



We achieve 91.42% test accuracy so it is best model for the dataset.

Let’s see the Normalized Confusion Matrix,



For better visualization of confusion matrix, we call a function ‘plot\_confusion\_matrix’ is been used. Now by calling above function we will visualize a normalized confusion matrix to see the percentage of correct and incorrect classification by the model. We see in the above confusion matrix, our model has given the highest accuracy of 99% in recognizing Trouser, 98% in recognizing Bags. The model has given lowest of 76% in recognizing shirts. It has given more then 90% accuracy and more then 85% accuracy in recognizing apparels of 9 classes.

The level of accuracy in recognizing objects is definitely high.

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 VGG-19 and 3-Layer CNN. 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 3-Layer CNN by far presents the most validation accuracy rate, as compared to VGG-19. 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 3\_Layer CNN 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 CNN, and then using the trained model for testing.

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. 35

[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. 36

[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.