Research Assignment 2:

Artificial Intelligence and Deep Learning

Tyrone Brown

Abstract

Image classification is a common task for Neural Network models. Accomplishing this task can be done so in several ways, however. This research explored both Dense Neural Networks and Convolutional Neural Networks and compared how the two performed in a variety of experiments that looked at different parameters and architectures. Specifically, our classification models used the MNIST fashion data to categorize images from an e-commerce catalog. The CNN performed the best, although experiencing more overfitting when doing equivalent comparisons to DNN (holding epochs constant). However, adjusting for this training time still allowed us to achieve an accuracy of 91.19% using a CNN with 2 Conv/MaxPool layers.

Introduction

Zalando is one of Europe's largest e-commerce companies providing clothing for men, women, and children. To accommodate developing a recommendation system, we have been tasked with creating an image classification model that can decipher each catalog item as 1 of 10 clothing categories. The purpose of this research is to explore alternative neural network structures and evaluate their differences in performance in image classification. The primary models that were compared were Dense Neural Nets and Convolutional Neural Nets. The goal is to choose the best algorithm and architecture for classifying images from an online clothing catalog.

Literature Review

Dense Neural Networks and Convolutional Neural Networks were previously evaluated in the article "Dense or Convolutional Neural Network" (Hue, 2020). The findings were in line with the findings of this research, in that CNN models often outperform their DNN counterparts when it comes to image classification.

Methodology

Methodology Overview

For this research, Dense Neural Net models and Convolutional Neural Net models were analyzed and compared with varying architectures for each. In image classification, a DNN simply takes all pixels of an image, flattens them into a single array, and uses each pixel as an input feature in a fully connected network. CNNs also utilize this dense structure, but only after additional convolution and pooling layers. The purpose of these additional layers is to extract certain visible features from an image, namely textures and edges. Once these features have been learned, these are flattened and input into the dense structure.

Implementation and Programming

Figure 1 shows the libraries that were used in performing all experimentation steps.

Data Preparation, Exploration and Visualization

The dataset used for training and testing the models used in this research is the MNIST fashion dataset. This dataset comes with and was obtained from TensorFlow's library. The data loads into a training set consisting of 60,000 28x28 images and a testing set consisting of 10,000 28x28 images. Each image in both sets has an associated label of what class the image shows (10 clothing categories). Since each feature (pixel) has values from 0-255, we must first normalize the dataset to values between 0-1 in preparing for model training. All clothing labels were converted to categorical arrays (1x10) by using TensorFlow's to_categorical() function.

Results

Key Observations

DNN versus CNN. The CNN models were easily the better performers when comparing to the DNN models. The best DNN model achieved an accuracy of 89.18% while CNN maxed out at 91.46%. However, the difference in training time between the 2 was exceptionally large as each of the models took 50.65 and 845.45 seconds to train, respectively (on 15 epochs).

Dropout Layers. Overall, the addition of a dropout layer with 20% dropout rate had virtually no impact (or a slightly negative impact) on all equivalent models. See Figure 6 for the comparison of our best model with and without the dropout layer.

Stride. Increasing the strides for the Convolution layers and Pooling layers, individually, each had negative affects on the top structure's performance but significantly improved training time.

Batch Size. As the batch sizes were increased, holding all else constant from our best model, the performance also decreased. Ranging from 100 to 10,000 for batch size, accuracy drops from 91.46% to 84.04%

Experiment Results

93 models were evaluated as a part of this research. See Figures 2-5 for all results.

DNN | 2 Hidden Layers of Equal Sizes. This experiment consisted of a DNN with 784 input nodes, 2 hidden layers and 10 output nodes that correspond to the 10 clothing categories. 6 models were built using this architecture, varying the number of nodes in each of the hidden layers. The best performance, an accuracy of 89.59%, was observed when each hidden layer consisted of 320 nodes. An additional set of models was created including a dropout layer. The best model of that set achieved an accuracy of 89.18% with hidden layer size of 320.

DNN | 3 Hidden Layers of Equal Sizes. This experiment consisted of a DNN with 784 input nodes and 3 hidden layers. 6 models were built using this architecture, varying the number of nodes in each of the hidden layers. The best performance, an accuracy of 88.98%, was observed when each hidden layer consisted of 320 nodes. An additional set of models was created including a dropout layer. The best model of that set achieved an accuracy of 88.91% with hidden layer size of 320.

DNN | 3 Hidden Layers of Descending Sizes. This experiment consisted of a DNN with 784 input nodes and 3 hidden layers of descending sizes. 5 models were built with the layer size being cut by 50% in each layer. Another 6 models were built with the first 2 layers having the same size, and third layer being half the size. The best performance, an accuracy of 89.11%, was observed when the hidden layers were of sizes 320, 160 then 80. An additional set of models was created including a dropout layer. The best model of that set achieved an accuracy of 88.92% with hidden layer sizes of 320, 320 and 160.

CNN | 2 Conv/Pooling Layers. This experiment consisted of a CNN with 2 filter layers, 2 convolution layers and 1 hidden dense layer. 6 models were built with varying filter sizes. The best performance of all experiments, an accuracy of 91.46%, was observed with filter counts of 32 and 64. An additional set of models was created including a dropout layer. The best model of that set achieved an accuracy of 91.39% with filter counts of 32 and 64. Although this was the best performing model holding all else constant (including epochs), Figure 7 shows that the model appears to overfit after around 5 epochs. To remedy this, an additional model was created stopping at 6 epochs; the accuracy drops slightly to 91.19%, however the time to train is reduced from 845 seconds to 378 seconds.

CNN | 2 Conv/Pooling Layers | Filter Visualization. Using the model above, we can extract and visualize the filter outputs and their related pooling outputs. All models had the most issues with deciphering between shirts and t-shirts. Figure 8 shows the features detected for a “shirt”, which even to the human eye isn’t different from a “t-shirt/top”. The highlighted features in Figure 8 show the networks detection of a few key edges that one would determine to be of a t-shirt by looking at the full image, filtered image as well as the pooled image. Figure 9 shows just how often our model confuses these categories.

CNN | 3 Conv/Pooling Layers. This experiment consisted of a CNN with 3 filter layers, 3 convolution layers and 1 hidden dense layer. 6 models were built with varying filter sizes. The best performance, an accuracy of 89.17%, was observed with filter counts of 32, 64 and 128. An additional set of models was created including a dropout layer. The best model of that set achieved an accuracy of 89.58% with filter counts of 32, 64 and 128.

CNN | 2 Conv/Pooling Layers, 1 Dense Hidden Layer, Varying Sizes. For this experiment, and all to follow, the base architecture used was of that which performed the best above; 2 Conv/Pooling Layers of 32 and 64 filters (Exp3_32_Model). 7 models were created by varying the size of the hidden dense layer. The best performance, an accuracy of 91.31%, was observed when the hidden layer contained 320 nodes.

CNN | 2 Conv/Pooling Layers, 2 Dense Hidden Layers, Varying Sizes. 8 models were created by varying the size of the 2 hidden dense layers. The best performance, an accuracy of 91.24%, was observed when the hidden layers each contained 1,024 nodes.

CNN | 2 Conv/Pooling Layers, 1 Dense Hidden Layer, Varying Stride. 2 models were created by independently increasing the Convolution stride size and the Pooling stride size. The best performance, an accuracy of 88.53%, was observed when increasing Pooling stride to 3x3.

CNN | 2 Conv/Pooling Layers, 1 Dense Hidden Layer, Varying Batch. 4 models were created by varying the batch size. The best performance, an accuracy of 91.35%, was observed when batch size was set to 500.

Conclusions

After evaluating all models and the impacts of various structures/parameters, the recommendation to management for our image classification model would be a CNN with 2 Convolution/Filter layers, 32 and 64 filters, with just 1 hidden layer. As observed, the CNN required a significant amount of time using this architecture and 15 epochs, however this can be remedied by early stopping at 6 epochs. Doing so reduced our accuracy from 91.46% to 91.19% but reduced the time to train by more than half. Furthermore, an additional recommendation should be made to re-consider the groupings of some of the clothing, specifically t-shirts and shirts and discussed in the results.

References

Hue, A. (2020, July 18). Dense or Convolutional‐Part‐1. Retrieved October 18, 2020, from https://medium.com/analytics-vidhya/dense-or-convolutional-part-1-c75c59c5b4ad

Appendix

Figure 1

Python Libraries Used for Research

```python
import datetime
import time
from packaging import version
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.ensemble import RandomForestClassifier

from collections import Counter
import numpy as np
import pandas as pd
from numpy.random import seed

# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow.keras.utils import to_categorical
from tensorflow import keras
from tensorflow.keras import models, layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D, BatchNormalization
from tensorflow.keras.layers import Dropout, Flatten, Input, Dense
from tensorflow.keras.datasets import fashion_mnist
```

Figure 2

Model Results (DNN with no Dropout)

Description	Model	Seed	Dense Layers	Conv/Pool Layers	Dense Nodes (DNN)	Filters (DNN)	Hidden Activation	Dropout Rate	Epochs	Optimizer	Time	Test Accuracy
DNN 2 Hidden Layers (Equal Sizes)	Exp1_10_Model	5	2		[10, 10]		relu	0	15	adam	15.29668593	85.29%
	Exp1_20_Model	5	2		[20, 20]		relu	0	15	adam	16.47711015	86.65%
	Exp1_40_Model	5	2		[40, 40]		relu	0	15	adam	16.4238801	87.35%
	Exp1_80_Model	5	2		[80, 80]		relu	0	15	adam	18.70116425	88.29%
	Exp1_160_Model	5	2		[160, 160]		relu	0	15	adam	27.33928823	88.89%
	Exp1_320_Model	5	2		[320, 320]		relu	0	15	adam	47.26520896	89.59%
DNN 3 Hidden Layers (Equal Sizes)	Exp2_10_Model	5	3		[10, 10, 10]		relu	0	15	adam	16.01287293	83.54%
	Exp2_20_Model	5	3		[20, 20, 20]		relu	0	15	adam	15.48161101	86.50%
	Exp2_40_Model	5	3		[40, 40, 40]		relu	0	15	adam	17.31635427	87.15%
	Exp2_80_Model	5	3		[80, 80, 80]		relu	0	15	adam	20.74799466	88.66%
	Exp2_160_Model	5	3		[160, 160, 160]		relu	0	15	adam	35.40010905	88.42%
	Exp2_320_Model	5	3		[320, 320, 320]		relu	0	15	adam	67.27881598	88.98%
DNN 3 Hidden Layers (N -> N/2 -> N/4)	Exp2b_10_Model	5	3		[10, 5.0, 2.5]		relu	0	15	adam	15.32803893	76.80%
	Exp2b_20_Model	5	3		[20, 10.0, 5.0]		relu	0	15	adam	15.36213803	85.11%
	Exp2b_40_Model	5	3		[40, 20.0, 10.0]		relu	0	15	adam	17.18824458	87.34%
	Exp2b_80_Model	5	3		[80, 40.0, 20.0]		relu	0	15	adam	19.02289939	87.75%
	Exp2b_160_Model	5	3		[160, 80.0, 40.0]		relu	0	15	adam	26.02824378	88.40%
	Exp2b_320_Model	5	3		[320, 160.0, 80.0]		relu	0	15	adam	41.3033731	89.11%
DNN 3 Hidden Layers (N -> N -> N/2)	Exp2c_10_Model	5	3		[10, 10, 5.0]		relu	0	15	adam	15.31327081	83.17%
	Exp2c_20_Model	5	3		[20, 20, 10.0]		relu	0	15	adam	16.0131073	86.17%
	Exp2c_40_Model	5	3		[40, 40, 20.0]		relu	0	15	adam	16.52262378	87.49%
	Exp2c_80_Model	5	3		[80, 80, 40.0]		relu	0	15	adam	20.26115251	88.56%
	Exp2c_160_Model	5	3		[160, 160, 80.0]		relu	0	15	adam	31.37456059	88.20%
	Exp2c_320_Model	5	3		[320, 320, 160.0]		relu	0	15	adam	56.61020875	88.55%

Figure 3

Model Results (DNN with 20% Dropout)

Description	Model	Seed	Dense Layers	Conv/Pool Layers	Dense Nodes (DNN)	Filters (DNN)	Hidden Activation	Dropout Rate	Epochs	Optimizer	Time	Test Accuracy
DNN 2 Hidden Layers (Equal Sizes) Dropout = 20%	Exp1_10_Model_Dropout	5	2		[10, 10]		relu	0.2	15	adam	15.56873512	82.66%
	Exp1_20_Model_Dropout	5	2		[20, 20]		relu	0.2	15	adam	15.62679148	86.00%
	Exp1_40_Model_Dropout	5	2		[40, 40]		relu	0.2	15	adam	16.91031718	86.90%
	Exp1_80_Model_Dropout	5	2		[80, 80]		relu	0.2	15	adam	20.23085928	88.31%
	Exp1_160_Model_Dropout	5	2		[160, 160]		relu	0.2	15	adam	30.29851651	88.87%
	Exp1_320_Model_Dropout	5	2		[320, 320]		relu	0.2	15	adam	50.65452194	89.18%
DNN 3 Hidden Layers (Equal Sizes) Dropout = 20%	Exp2_10_Model_Dropout	5	3		[10, 10, 10]		relu	0.2	15	adam	16.59723496	83.01%
	Exp2_20_Model_Dropout	5	3		[20, 20, 20]		relu	0.2	15	adam	16.54930925	86.01%
	Exp2_40_Model_Dropout	5	3		[40, 40, 40]		relu	0.2	15	adam	17.91833448	87.13%
	Exp2_80_Model_Dropout	5	3		[80, 80, 80]		relu	0.2	15	adam	22.73384333	88.47%
	Exp2_160_Model_Dropout	5	3		[160, 160, 160]		relu	0.2	15	adam	43.03747773	88.02%
	Exp2_320_Model_Dropout	5	3		[320, 320, 320]		relu	0.2	15	adam	66.77269745	88.91%
DNN 3 Hidden Layers (N -> N/2 -> N/4) Dropout = 20%	Exp2b_10_Model_Dropout	5	3		[10, 5.0, 2.5]		relu	0.2	15	adam	15.17047	68.28%
	Exp2b_20_Model_Dropout	5	3		[20, 10.0, 5.0]		relu	0.2	15	adam	16.47280979	82.94%
	Exp2b_40_Model_Dropout	5	3		[40, 20.0, 10.0]		relu	0.2	15	adam	17.43917489	86.66%
	Exp2b_80_Model_Dropout	5	3		[80, 40.0, 20.0]		relu	0.2	15	adam	19.45000625	87.91%
	Exp2b_160_Model_Dropout	5	3		[160, 80.0, 40.0]		relu	0.2	15	adam	26.99340796	88.22%
	Exp2b_320_Model_Dropout	5	3		[320, 160.0, 80.0]		relu	0.2	15	adam	43.30580592	88.52%
DNN 3 Hidden Layers (N -> N -> N/2) Dropout = 20%	Exp2c_10_Model_Dropout	5	3		[10, 10, 5.0]		relu	0.2	15	adam	15.83522916	79.61%
	Exp2c_20_Model_Dropout	5	3		[20, 20, 10.0]		relu	0.2	15	adam	16.44111013	85.59%
	Exp2c_40_Model_Dropout	5	3		[40, 40, 20.0]		relu	0.2	15	adam	17.76067472	86.91%
	Exp2c_80_Model_Dropout	5	3		[80, 80, 40.0]		relu	0.2	15	adam	20.63178802	87.71%
	Exp2c_160_Model_Dropout	5	3		[160, 160, 80.0]		relu	0.2	15	adam	35.15743375	88.89%
	Exp2c_320_Model_Dropout	5	3		[320, 320, 160.0]		relu	0.2	15	adam	55.9388299	88.92%

Figure 4

Model Results (CNN with 2 and 3 Conv/Pool Layers, with and without 20% Dropout)

Description	Model	Seed	Dense Layers	Conv/Pool Layers	Dense Nodes (DNN)	Filters (DNN)	Hidden Activation	Dropout Rate	Epochs	Optimizer	Time	Test Accuracy
CNN 2 Conv/Pool Layers (N -> N*2)	Exp3_1_Model	5	1	2	[1024]	[1, 2]	relu	0	15	adam	233.647383	84.42%
	Exp3_2_Model	5	1	2	[1024]	[2, 4]	relu	0	15	adam	249.508512	87.75%
	Exp3_4_Model	5	1	2	[1024]	[4, 8]	relu	0	15	adam	287.1167579	89.51%
	Exp3_8_Model	5	1	2	[1024]	[8, 16]	relu	0	15	adam	343.6225519	90.16%
	Exp3_16_Model	5	1	2	[1024]	[16, 32]	relu	0	15	adam	486.4139123	90.67%
	Exp3_32_Model	5	1	2	[1024]	[32, 64]	relu	0	15	adam	845.4483314	91.46%
	Exp3_32_Model_6Epochs	5	1	2	[1024]	[32, 64]	relu	0	15	adam	377.8962166	91.19%
CNN 3 Conv/Pool Layers (N -> N*2 -> N*4)	Exp4_1_Model	5	1	3	[1024]	[1, 2, 4]	relu	0	15	adam	239.2184708	70.29%
	Exp4_2_Model	5	1	3	[1024]	[2, 4, 8]	relu	0	15	adam	249.5887611	78.55%
	Exp4_4_Model	5	1	3	[1024]	[4, 8, 16]	relu	0	15	adam	270.7562928	83.87%
	Exp4_8_Model	5	1	3	[1024]	[8, 16, 32]	relu	0	15	adam	309.9524767	86.86%
	Exp4_16_Model	5	1	3	[1024]	[16, 32, 64]	relu	0	15	adam	406.7143426	88.80%
	Exp4_32_Model	5	1	3	[1024]	[32, 64, 128]	relu	0	15	adam	709.9085112	89.17%
CNN 2 Conv/Pool Layers (N -> N*2) Dropout = 20%	Exp3_1_Model_Dropout	5	1	2	[1024]	[1, 2]	relu	0.2	15	adam	250.9958901	84.38%
	Exp3_2_Model_Dropout	5	1	2	[1024]	[2, 4]	relu	0.2	15	adam	265.6077387	87.64%
	Exp3_4_Model_Dropout	5	1	2	[1024]	[4, 8]	relu	0.2	15	adam	296.4236913	89.20%
	Exp3_8_Model_Dropout	5	1	2	[1024]	[8, 16]	relu	0.2	15	adam	357.4352975	90.20%
	Exp3_16_Model_Dropout	5	1	2	[1024]	[16, 32]	relu	0.2	15	adam	493.9962835	90.39%
	Exp3_32_Model_Dropout	5	1	2	[1024]	[32, 64]	relu	0.2	15	adam	865.3213038	91.39%
CNN 3 Conv/Pool Layers (N -> N*2 -> N*4) Dropout = 20%	Exp4_1_Model_Dropout	5	1	3	[1024]	[1, 2, 4]	relu	0.2	15	adam	252.7605612	70.39%
	Exp4_2_Model_Dropout	5	1	3	[1024]	[2, 4, 8]	relu	0.2	15	adam	270.1563165	78.48%
	Exp4_4_Model_Dropout	5	1	3	[1024]	[4, 8, 16]	relu	0.2	15	adam	287.952244	84.41%
	Exp4_8_Model_Dropout	5	1	3	[1024]	[8, 16, 32]	relu	0.2	15	adam	324.2131732	86.47%
	Exp4_16_Model_Dropout	5	1	3	[1024]	[16, 32, 64]	relu	0.2	15	adam	419.0056968	88.49%
	Exp4_32_Model_Dropout	5	1	3	[1024]	[32, 64, 128]	relu	0.2	15	adam	721.3594909	89.58%

Figure 5

Model Results (Further experiments using best performing model)

Description	Model	Seed	Dense Layers	Conv/Pool Layers	Dense Nodes (DNN)	Filters (DNN)	Hidden Activation	Dropout Rate	Epochs	Optimizer	Time	Test Accuracy
CNN 2 Conv/Pool Layers (32 -> 64) 1 Dense Layer	exp_denseSize_10	5	1	2	[10]	[32, 64]	relu	0	15	adam	572.1769197	88.59%
	exp_denseSize_20	5	1	2	[20]	[32, 64]	relu	0	15	adam	566.5048504	90.39%
	exp_denseSize_40	5	1	2	[40]	[32, 64]	relu	0	15	adam	567.6819127	90.40%
	exp_denseSize_80	5	1	2	[80]	[32, 64]	relu	0	15	adam	574.7383423	90.85%
	exp_denseSize_160	5	1	2	[160]	[32, 64]	relu	0	15	adam	586.201231	90.98%
	exp_denseSize_320	5	1	2	[320]	[32, 64]	relu	0	15	adam	615.1177037	91.31%
	exp_denseSize_640	5	1	2	[640]	[32, 64]	relu	0	15	adam	684.2308655	91.16%
CNN 2 Conv/Pool Layers (32 -> 64) 2 Dense Layers	exp_denseSize2_10	5	2	2	[10, 10]	[32, 64]	relu	0	15	adam	583.4081497	89.43%
	exp_denseSize2_20	5	2	2	[20, 20]	[32, 64]	relu	0	15	adam	589.3459959	89.72%
	exp_denseSize2_40	5	2	2	[40, 40]	[32, 64]	relu	0	15	adam	591.6683769	90.65%
	exp_denseSize2_80	5	2	2	[80, 80]	[32, 64]	relu	0	15	adam	596.1150713	90.88%
	exp_denseSize2_160	5	2	2	[160, 160]	[32, 64]	relu	0	15	adam	618.4023604	91.03%
	exp_denseSize2_320	5	2	2	[320, 320]	[32, 64]	relu	0	15	adam	670.068656	91.22%
	exp_denseSize2_640	5	2	2	[640, 640]	[32, 64]	relu	0	15	adam	788.29092	90.71%
	addDenseLayer	5	2	2	[1024, 1024]	[32, 64]	relu	0	15	adam	1056.44575	91.24%
Best Exp3 Increased Conv. Stride Size (2,2)	increaseConvStride	5	1	2	[1024]	[32, 64]	relu	0	15	adam	129.132843	86.44%
Best Exp3 Increased Pool Stride Size (3,3)	increasePoolStride	5	1	2	[1024]	[32, 64]	relu	0	15	adam	392.0053823	88.53%
Best Exp3 Varying Batch Sizes	exp_batchSize_500	5	1	2	[1024]	[32, 64]	relu	0	15	adam	719.1245253	91.35%
	exp_batchSize_1000	5	1	2	[1024]	[32, 64]	relu	0	15	adam	692.4851878	90.64%
	exp_batchSize_5000	5	1	2	[1024]	[32, 64]	relu	0	15	adam	678.9370654	86.61%
	exp_batchSize_10000	5	1	2	[1024]	[32, 64]	relu	0	15	adam	672.1767962	84.04%

Figure 6

Test Accuracy vs. Number of Filters in First Convolution Layer (from model Exp3_32_Model)

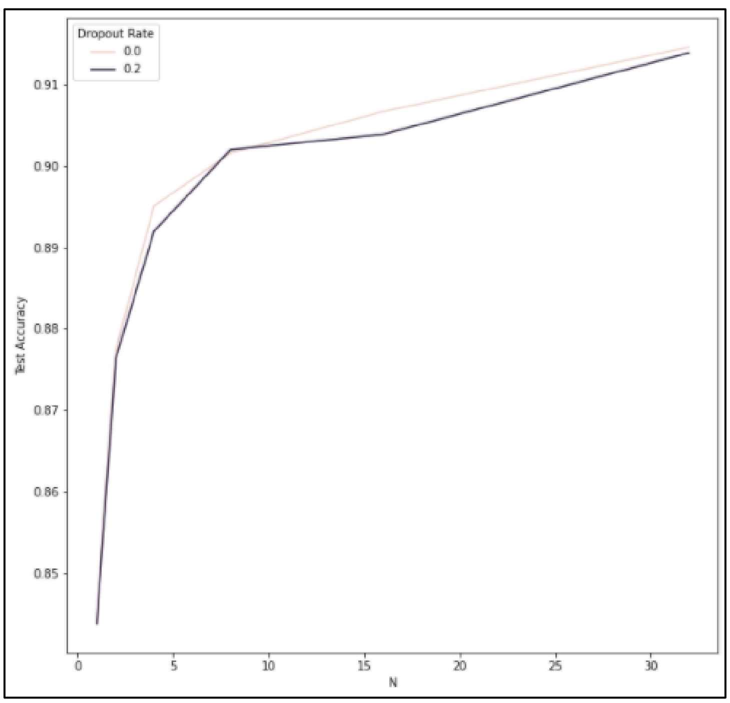

Figure 7

Exp3_32_Model Loss and Accuracy Trends

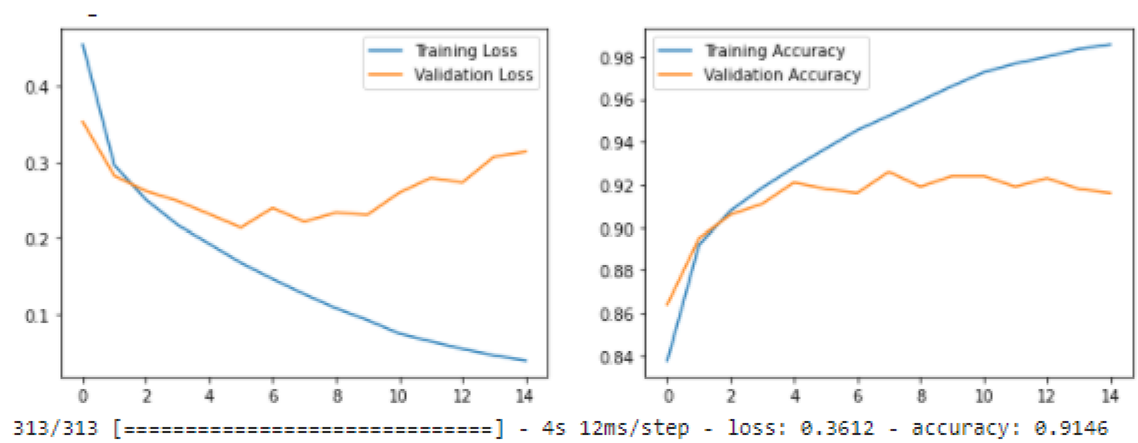

Figure 8

Exp3_32_Model Features for a "Shirt"

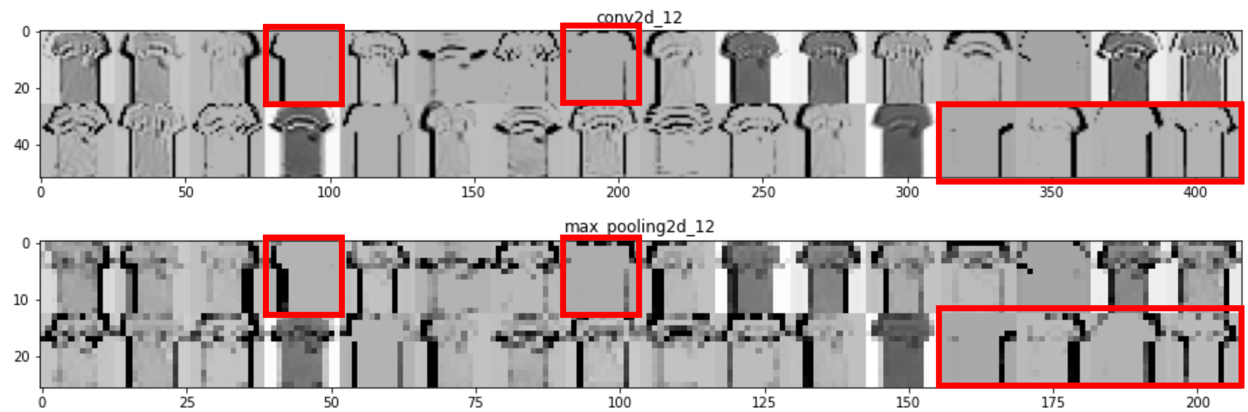

Figure 9

Exp3_32_Model Confusion Matrix