CSE 574 Fall 2019 Project 2

Using Neural Networks for Classification of Fashion-MNIST Images

Srisai Karthik Neelamraju
UBID - 50316785
University at Buffalo
Buffalo, NY 14260
neelamra@buffalo.edu

Abstract

Extracting features from images and processing them to identify an image holds significant importance in vision-related tasks. Using neural networks, however, these tasks can be efficiently tackled. One such task is classifying images of fashion apparel into several categories. In this project, three deep learning models with various architectures were implemented to address this problem. One network had one-hidden layer, whereas the second had multiple dense hidden layers. The third architecture was based on using convolutional layers. Different hyperparameters for each of these three architectures were tuned and the best combination of parameters was then chosen to train each of these networks. Observing their performances on the test set revealed that highest accuracy was achieved by the convolutional neural network, whereas the single-layer neural network performed the worst. These results further demonstrated the ability of deep convolutional neural networks in recognizing exact features of images.

1 Introduction

Deep learning is widely used these days for solving complex machine learning tasks in areas such as computer vision and natural language processing. Artificial neural networks, which are arrangements of neurons in several layers similar to a human brain, are central to any deep learning model. Each neuron in a neural network essentially performs a computation on its input – most typically multiplying the input with a set of weights and then activating the input using functions like sigmoid. These activations are then passed to all the connected neurons in the network. The difference in the actual and expected outputs is used to train the weights of each neuron in the network. In this way, a neural network learns a mapping from the input data to the output. Deep neural networks with many layers can be helpful in realizing extremely complex non-linear functions. The aim of this project is to analyze the performance of different neural networks on classifying images into several (ten) categories. These networks are modeled such that their output layers give the probabilities of the input belonging each of the ten classes using a SoftMax function [1]. The loss function for this problem is defined by categorical cross entropy, whose value is highest when predicted and actual outputs are complementary. The architectures implemented are detailed herewith.

2 Dataset

For this project, the Fashion-MNIST dataset that contains images of fashion apparel was used. This dataset, as described in the dataset description [2], was initially developed to replace the famous MNIST digits dataset since the latter does not adequately represent all the complexities possessed by a modern-day computer vision task. The training set consists of 60,000 grayscale images of size 28 × 28 each. Hence, there are 784 pixels (features) that describe each training instance. Associated with each training instance is an integer that describes its class. There are exactly ten classes of images in the training set: T-shirt/top (0), Trouser (1), Pullover (2), Dress (3), Coat (4), Sandal (5), Shirt (6), Sneaker (7), Bag (8) and Ankle Boot (9). Integers from 0 to 9 are used as labels to denote these classes in that order as shown. On the other hand, the test set consists of 10,000 images. In the test set, there are exactly 1,000 images belonging to each of the ten possible classes. This was also used as a validation set for tuning the best possible hyperparameters.

3 Preprocessing

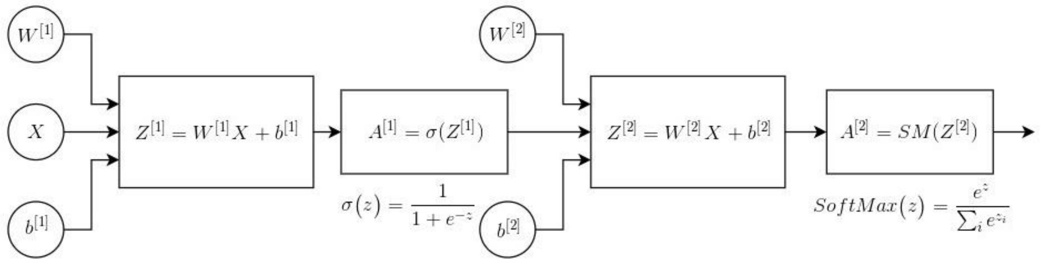
As described previously, the training and test data of the Fashion-MNIST dataset consists of 28 × 28 grayscale images. Hence, each image has 784 pixels with each pixel being an integer between 0 and 255. To improve the training performance of the single layer neural network, these pixel values were subjected to standard scaling using Scikit-learn's preprocessing module, i.e. they were normalized to have zero mean and unit variance. Besides, for the multi-layer and the convolutional neural networks, these pixel values were normalized to the range [0, 1] by dividing each of them with 255. Hence, essentially, min-max scaling was performed on the data sets for the second and the third implementations.

4 Architecture

To classify the images into one of the 10 classes, three different approaches were considered. Firstly, a single layer neural network that uses mini-batch gradient descent was employed. Later, a multi-layer fully connected deep neural network and a convolutional neural network implemented using Keras were used. The respective neural network architectures and their hyperparameters were finalized based on their performance on the validation set. The chosen architectures and best set of parameters used to train these are described in this section.

4.1 Neural Network with One Hidden Layer

In this part, a neural network with 128 units in the hidden layer was designed from scratch. All the layers in this architecture were fully-connected. Sigmoid function was used as the activation for the hidden layer, whereas SoftMax was the activation for the output layer. If X denotes the training data with instances along the columns and ($W^{[l]}$, $b^{[l]}$, $A^{[l]}$) denote the weights, biases and activations respectively of layer l, then the figure below represents the computation graph of the forward propagation involved in the implementation -



$W^{[1]}$
X
$b^{[1]}$
$Z^{[1]} = W^{[1]}X + b^{[1]}$
$A^{[1]} = \sigma(Z^{[1]})$
$\sigma(z) = \frac{1}{1+e^{-z}}$
$W^{[2]}$
$b^{[2]}$
$Z^{[2]} = W^{[2]}X + b^{[2]}$
$A^{[2]} = SM(Z^{[2]})$
$SoftMax(z) = \frac{e^{z}}{\sum_i e^{z_i}}$

Figure 1: Forward Propagation in one hidden layer Neural Network

The loss function for this problem was calculated using the expression,

$$L(A^{[2]}, Y) = -\frac{1}{n_{train}} \sum_{i=1}^{n_{train}} Y^{(i)} \log A^{[2](i)}$$

This loss function is also known as the categorical cross entropy and used widely for multi-class classification problems. Gradients with respect to weights and biases were found by backpropagation and mini-batch gradient descent was used to update them, i.e. only a subset of training instances was considered for each step of gradient update. A mini-batch size of 32 and a learning rate of 0.03 was observed to be performing best on the validation set. The gradients were computed by backpropagating the output $A^{[2]}$ as follows (for $dB^{[2]}$ and $dB^{[1]}$, the summations are defined over each row since the training instances are placed along the columns of the training data X; '×' in $dZ^{[1]}$ represents element-wise product) -

$$dZ^{[2]} = A^{[2]} - Y$$

$$dW^{[2]} = \frac{1}{n_{train}} dZ^{[2]} A^{[1]T}$$

$$db^{[2]} = \frac{1}{n_{train}} \sum dZ^{[2]}$$

$$dA^{[1]} = W^{[2]T} dZ^{[2]}$$

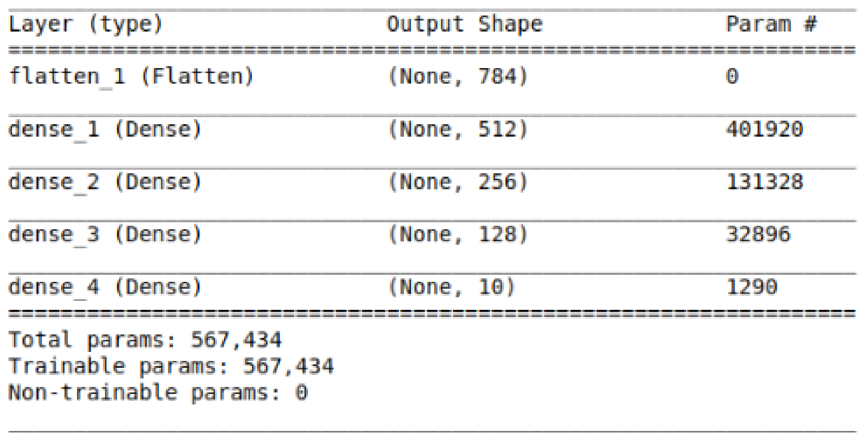
$$dZ^{[1]} = dA^{[1]} \times \left[A^{[1]} - (A^{[1]})^2\right]$$

$$dW^{[1]} = \frac{1}{n_{train}} dZ^{[1]} X^T$$

$$db^{[1]} = \frac{1}{n_{train}} \sum dZ^{[1]}$$

4.2 Neural Network with Multiple Hidden Layers

In this part, a neural network with an input layer, three hidden layers and an output layer was designed using Keras sequential model [3]. All the layers in this architecture were dense [4]. The 28 × 28 input to the first layer was flattened to form a vector of length 784. In the subsequent layers, there are 512, 256, 128 and 10 units respectively as described in figure 2.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               401920
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_3 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_4 (Dense)              (None, 10)                1290
=================================================================
Total params: 567,434
Trainable params: 567,434
Non-trainable params: 0
_________________________________________________________________
```

Figure 2: Architecture of the multi-layer neural network implemented using Keras

The first three layers used Rectified Linear Unit (ReLU) as the activation function, whereas the last layer used SoftMax activation. The closed form of ReLU activation is -

$$ReLU(z) = \begin{cases} z, \ if \ z > 0 \\ 0, \ otherwise \end{cases}$$

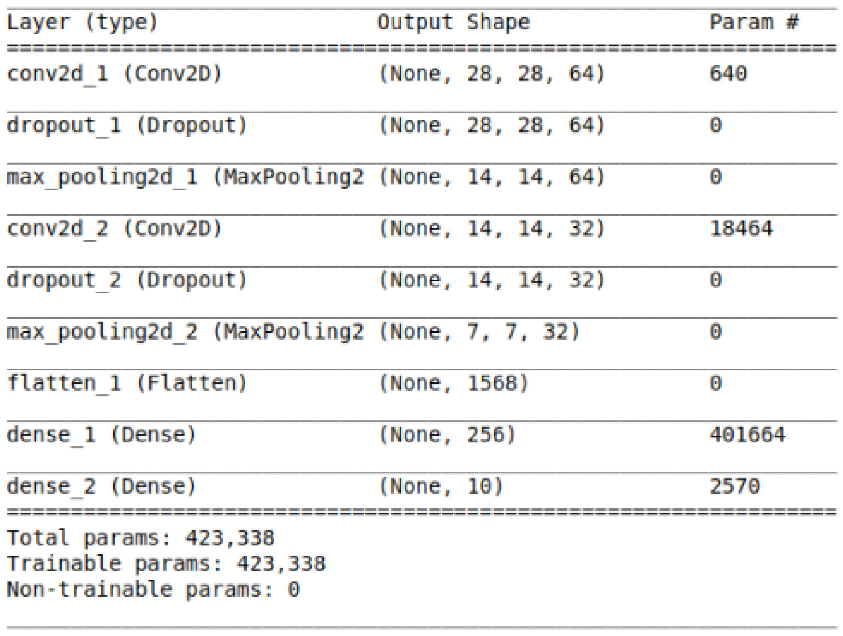
Once again, categorical cross entropy was the loss function, but this time Adam optimizer [5] was used instead of conventional gradient descent. Adam is an adaptive optimizer that enables faster convergence of deep neural networks. Upon trying various combinations of learning parameter, for this architecture, a mini-batch size of 256 and a learning rate of 0.0001 was found to be the giving low validation loss and high validation accuracy.

4.3 Convolutional Neural Network

In this part, a convolutional neural network with two convolutional layers each followed by one pooling layer, and two dense layers was implemented in the order shown in Figure 3.

The first convolutional layer uses 64 three-dimensional filters with same padding, i.e. the dimension of the input remains unchanged despite the convolution operation. Similarly, the second convolutional layer also uses same padding but with 32 three-dimensional filters. A max pooling layer is placed after each of the convolutional layers so as to achieve a reduction in the dimensions of the input. All the layers in this architecture have ReLU activation function, apart from the output layer which uses SoftMax activation.

By adding dropout layers to the sequential model after each convolutional layer, dropout regularization was performed as it was observed that the model was overfitting the training data after very few epochs and thus, giving high validation set loss.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 64)        640
_________________________________________________________________
dropout_1 (Dropout)          (None, 28, 28, 64)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 32)        18464
_________________________________________________________________
dropout_2 (Dropout)          (None, 14, 14, 32)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1568)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               401664
_________________________________________________________________
dense_2 (Dense)              (None, 10)                2570
=================================================================
Total params: 423,338
Trainable params: 423,338
Non-trainable params: 0
_________________________________________________________________
```

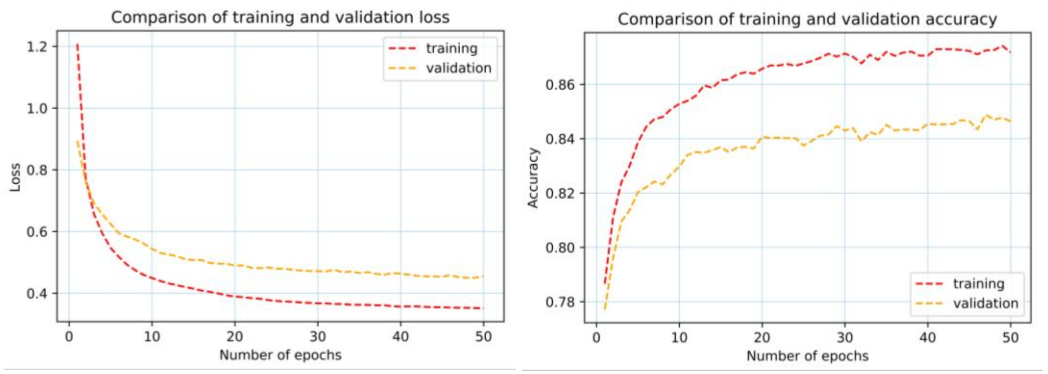
Figure 3: Architecture of the convolutional neural network implemented using Keras

Yet again, categorical cross entropy was the loss function and Adam was the optimizer used for optimizing the weights and the biases of each layer. However, for this convolutional model, a mini-batch size of 64 as opposed to 256 for the multi-layer neural network and a learning rate of 0.001 was found to be the best possible combination.

5 Results and Inferences

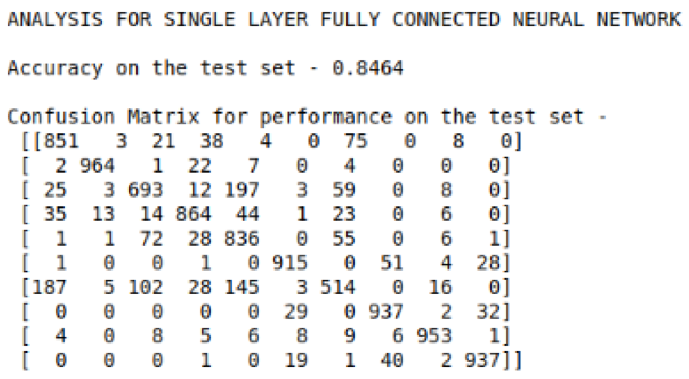
After tuning all the hyperparameters of the learning algorithms, the performance of each of the aforementioned three models on the training and validation sets was observed.

Figures 4(a) and 4(b) respectively compare the training and validation losses and accuracies after 50 epochs for the first neural network. The training was not extended beyond 50 epochs as further training of the network yielded little increase in the validation set accuracy.



(a) (b)

Figure 4: Analyzing performance of the single hidden layer neural network; comparing training and validation (a) loss and (b) accuracy

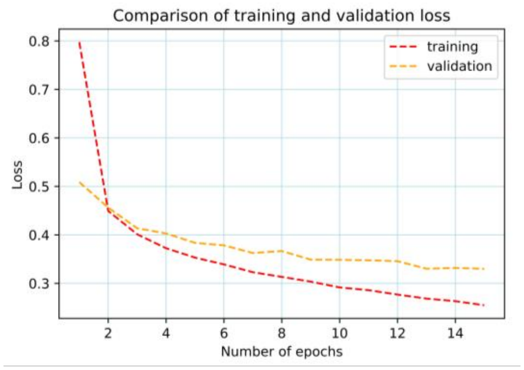


```
ANALYSIS FOR SINGLE LAYER FULLY CONNECTED NEURAL NETWORK

Accuracy on the test set - 0.8464

Confusion Matrix for performance on the test set -
 [[851   3  21  38   4   0  75   0   8   0]
 [  2 964   1  22   7   0   4   0   0   0]
 [ 25   3 693  12 197   3  59   0   8   0]
 [ 35  13  14 864  44   1  23   0   6   0]
 [  1   1  72  28 836   0  55   0   6   1]
 [  1   0   0   1   0 915   0  51   4  28]
 [187   5 102  28 145   3 514   0  16   0]
 [  0   0   0   0   0  29   0 937   2  32]
 [  4   0   8   5   6   8   9   6 953   1]
 [  0   0   0   1   0  19   1  40   2 937]]
```

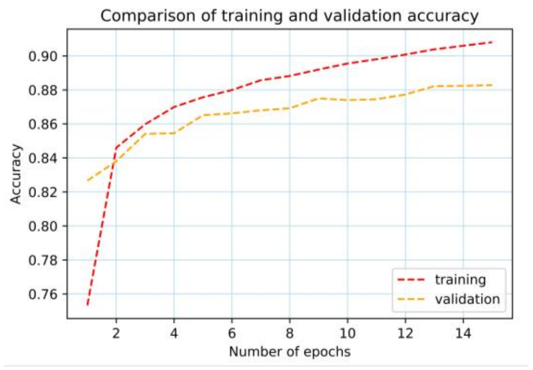
Figure 5: Evaluation metrics on the test set for the single hidden layer neural network

It was observed that the performance of this model on the test set was satisfactory with an accuracy of about 84%. However, the confusion matrix shown in figure 5 suggested that as many as 197 images of pullovers (class 2) were wrongly classified as coats (4). Similarly, images of t-shirts (0), pullovers (2), coats (4) and shirts (6) had high errors in classification among others. Only 514 out of 1,000 (~ 51%) images of shirts were correctly classified. A similar trend was also observed for the classes sandal (5), sneaker (7) and ankle boot (9), even though the performance on these classes wasn't as poor as on the former. These metrics suggested that a better architecture was required to address this classification efficiently.

Going ahead to the second neural network, figures 6(a) and 6(b) respectively compare its training and validation losses and accuracies after 15 epochs. Since it was observed that further training of the model often resulted in overfitting, early stopping was used to halt the training after 15 epochs, i.e. before the model started to overfit.

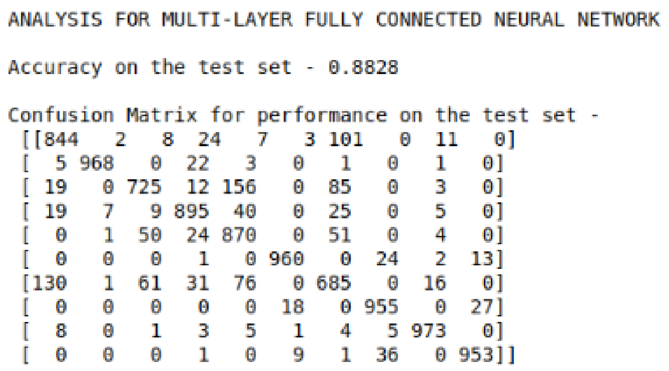


(a)



(b)

Figure 6: Analyzing the performance of fully connected multi-layer neural network; comparing training and validation (a) loss and (b) accuracy



```
ANALYSIS FOR MULTI-LAYER FULLY CONNECTED NEURAL NETWORK

Accuracy on the test set - 0.8828

Confusion Matrix for performance on the test set -
 [[844   2   8  24   7   3 101   0  11   0]
 [  5 968   0  22   3   0   1   0   1   0]
 [ 19   0 725  12 156   0  85   0   3   0]
 [ 19   7   9 895  40   0  25   0   5   0]
 [  0   1  50  24 870   0  51   0   4   0]
 [  0   0   0   1   0 960   0  24   2  13]
 [130   1  61  31  76   0 685   0  16   0]
 [  0   0   0   0   0  18   0 955   0  27]
 [  8   0   1   3   5   1   4   5 973   0]
 [  0   0   0   1   0   9   1  36   0 953]]
```

Figure 7: Evaluation metrics on the test set for the multi-layer neural network

It was observed that the model made decent predictions on the test set with an accuracy of nearly 88%, which was relatively better than what a one hidden layer neural network had achieved. Besides, the confusion matrix in figure 7 showed that images belonging to classes trouser (1), dress (3), sandal (5), sneaker (7), bag (8) and ankle boot (9) have been identified more correctly than others. Even though a significant improvement was made in classifying the shirts, the accuracy of the model on that class was once again the least. Figure 8 displays two wrongly classified images. On the other hand, the accuracy on the class bag (8) was the highest as 973 out of 1,000 (~ 97%) images were rightly classified. In sum, even this model was not capable of differentiating the nuances in the images of t-shirts, pullovers, coats and shirts whose image pixels follow pretty much similar distributions.

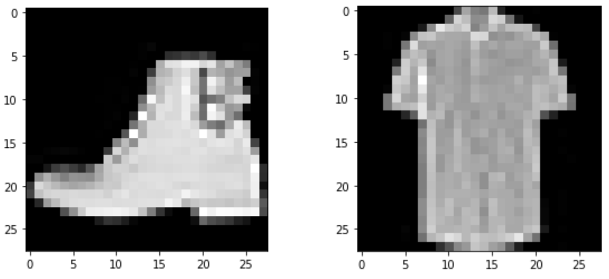
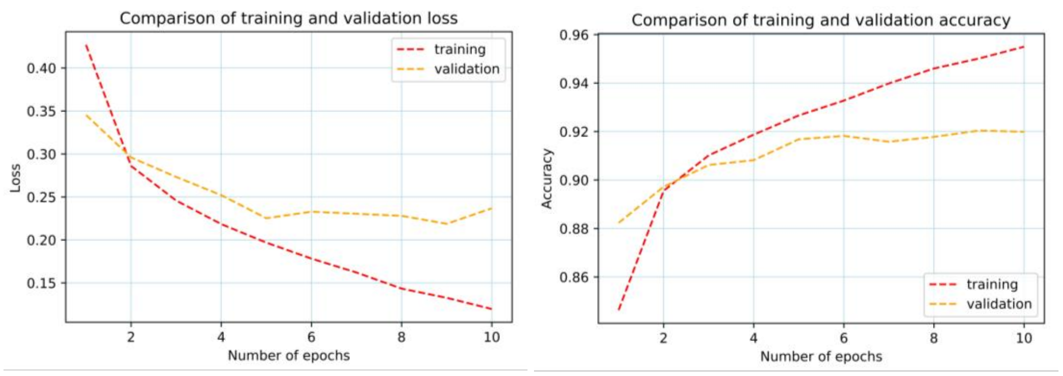


Figure 8: Examples of an ankle boot and a shirt image that were wrongly classified into classes sneaker and T-shirt respectively

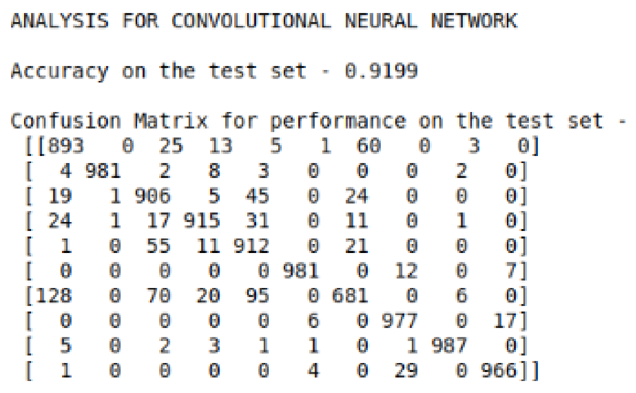
Finally, moving on to the third neural network, figures 9(a) and 9(b) respectively compare its training and the validation losses and accuracies after 10 epochs. It was observed that overfitting the training set was a real concern while training this model and consequently, dropout regularization was achieved by adding two dropout layers with dropout rate 0.20. This means that 20% of the input nodes were dropped (inactivated) while performing the forward propagation. Consequently, the model no longer suffered from overfitting.

It was observed that the convolutional model made good predictions on the test set with an accuracy of nearly 92%, which was much better than what both of the earlier neural networks had achieved. Besides, the confusion matrix in Figure 10 showed that images belonging to classes 1, 3, 5, 7, 8 and 9 have once again been identified more correctly than others whereas those corresponding to class 6 had least accuracy. Howver, unlike previous models, this model had 91% accuracy on classifying the images of the class coat (4). Even though the performance of the networks in tasks 2 and 3 was much similar in rightly identifying shirts and t-shirts, the number of coats and pullovers identified has increased this time. This shows that the convolutional network was able to successfully find exact features that differentiate coats and pullovers from the shirts.



(a) (b)

Figure 9: Analyzing performance of the convolutional neural network; comparing training and validation (a) loss and (b) accuracy



```
ANALYSIS FOR CONVOLUTIONAL NEURAL NETWORK

Accuracy on the test set - 0.9199

Confusion Matrix for performance on the test set -
 [[893   0  25  13   5   1  60   0   3   0]
 [  4 981   2   8   3   0   0   0   2   0]
 [ 19   1 906   5  45   0  24   0   0   0]
 [ 24   1  17 915  31   0  11   0   1   0]
 [  1   0  55  11 912   0  21   0   0   0]
 [  0   0   0   0   0 981   0  12   0   7]
 [128   0  70  20  95   0 681   0   6   0]
 [  0   0   0   0   0   6   0 977   0  17]
 [  5   0   2   3   1   1   0   1 987   0]
 [  1   0   0   0   0   4   0  29   0 966]]
```

Figure 10: Evaluation metrics on the test set for the convolutional neural network

In conclusion, by analyzing the confusion matrices on the test set for these three neural networks, an important observation can be made. The performance was consistently good on images belonging to several classes and similarly, consistently bad on images belonging to some others. The images belonging to the classes like trouser, dress and bag (on which these models performed the best) were unique as there was no other class with similar images. However, the images of the classes {t-shirt, pullover, coat, shirt} and {sandal, sneaker, ankle boot} were much alike. As a result, the pixel patterns for images of these types were very similar as shown in figure 11. This was the primary reason for having so many mismatches when the models were analyzed.

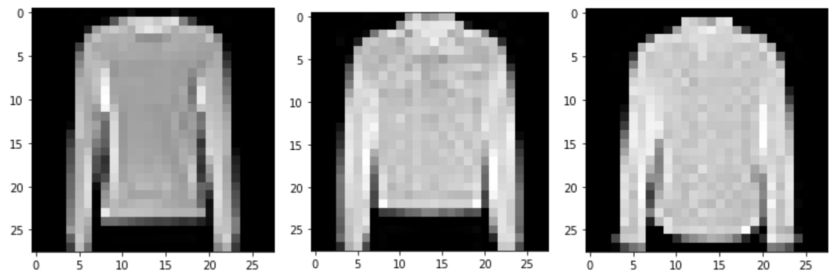


Figure 11: Very similar images of pullover, coat and shirt

However, it was observed that the convolutional model was able to successfully tackle this problem to an extent. This model was able to differentiate pullovers, coats, sandals, sneakers and ankleboots. This was because the convolutional network has capabilities of exhibiting much complex non-linearities and thus understands the nuances in these classes better. On the flip side, it still performed very moderately on identifying shirts and t-shirts. These results suggested that none of the three neural networks had the complexity required to identify features that can correctly classify images belonging to these classes. Consequently, none of them was able to efficiently differentiate these two classes. Having said that, very deep convolutional layers might be the best choice to further boost the test set accuracy.

6 Summary

In this project, three deep learning models that used different neural network architectures - single-layer, fully connected multi-layer and convolutional - were designed to classify images from Fashion-MNIST dataset. It contained 60,000 training and 10,000 test images. Mini-batch gradient descent was used to optimize the first network whereas Adam optimization was used for the latter two. Hyperparameters like the number of hidden layers, number of hidden units in each layer, learning rate, mini-batch size and number of epochs were tuned for each of these three neural networks and they were then trained on the dataset. Later, their performance on the testing set was analyzed. While the neural network with one hidden layer gave an accuracy of about 84%, the multi-layer neural network performed better with an accuracy of 88%. However, it was the convolutional neural network that performed the best among the three models, giving a test set accuracy of 92%. This further demonstrated the efficiency of convolutional networks in correctly identifying the features of an image and classifying it. Besides, it was also observed that much deeper networks were required to increase the accuracy on the training set beyond what was achieved.

References

[1] Bishop, C.M., 2006. *Pattern Recognition and Machine Learning*. Springer.

[2] https://research.zalando.com/welcome/mission/research-projects/fashion-mnist/

[3] https://keras.io/models/sequential/

[4] https://keras.io/layers/core/

[5] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).

[5] CSE 574 Slides by Prof. Sargur Srihari

[6] Andrew Ng’s ‘Deep Learning by deeplearning.ai’ on Coursera