Applications of Machine Learning to the Multiclassification of Images

Anouksha Bansal [*]

January 5, 2022

Abstract

Classification of clothing finds wide application in fashion industry, social media, e-commerce, forensics, criminal law etc. and thus requires improved accuracy in classifying them. In this project we review Machine Learning (ML) concepts and work on various methods of Convoluted Neural Networks. We apply them to a data set of images to create a model that classifies the images as different clothing items. The Hyper-parameter Lambda (λ) is used in a Regularization function. The model loss reduces with the addition of the Hyper-parameter to the model. However, upon increasing the value of Lambda there is no significant improvement on the model. Hence, we can conclude that Lambda = 0.0 is the best model with minimum loss.

1 Introduction

Machine Learning is an application which has the ability to automatically improve and learn without the need to be explicitly programmed. ML mainly focuses on the advancement in programming which can readily access data and routinely employ the data for self-development and learning. ML algorithms use statistics to find patterns within huge amounts of raw data. The primary step in the learning process is for the program to observe large data sets and find relevant patterns to provide superior decision-making possibilities on the basis of examples provided in the future prospect. The main agenda for the computer program to be able to work automatically and make appropriate adjustments without human assistance and intervention.

In this paper, the dataset in image form has been acquired from the Fashion-MNIST database. Originally, MNIST (Modified National Institute of Standards and Technology) is a large database of handwritten digits that has been widely researched upon. Fashion-MNIST is also a large database consisting of clothing articles images. The images have been standardized to 28×28 pixel size.

[*]Advised by: Guillermo Goldsztein of the Georgia Institute of Technology

Previously, using neural networks algorithms, researchers have been successful in achieving “close to human performance” on the MNIST database.

Fashion-MNIST is a large dataset consisting of 70,000 clothes images, which is further branched into approximately 10,000 examples in test sets and the remaining 60,000 in the training set. Each example has been associated with one of the labels of the 10 classes and is a standard 28 $\times$ 28 grayscale image. Each labeled class represents individual clothing types such as shirts, shoes, dresses, etc.

Supervised ML algorithms can apply previously learnt behavior to newer data sets using classified-labeled examples to predict future possibilities. The process is to begin with the analysis of a known database where the ML algorithm is able to produce an “inferred function” to make predictions of the output values. The ML program, after sufficient training, is able to provide predicted targets for new inputs. The ML algorithm also has the ability to compare the output predictions to the intended output and is, therefore, able to find errors.

Classification is the procedure of predicting the various categories of a dataset. Binary-classification, as the name suggests, is the method of classifying the input data into two classes, whereas multi-classification is the method of classifying the input data into more than two classes. In our dataset we have 10 classes each representing an article of clothing. Classification is an example of supervised learning. Predictive modeling is defined as estimating a mapping function (f) from the input variable (X) into discrete output variables (y).

Section 2, Neural Networks, highlights the use of Convolutional Neural Networks (CNN) for image classification, binary cross entropy, and parameters used to minimize error. Section 3 covers the related literature being reviewed followed by the methodology adopted to create the model in section 4. The subsequent sections cover the results discussion (section 5), conclusion (section 6) and references.

2 Neural Networks

Neural Networks (NN) algorithms are based on the neuron structure of the human brain. NN algorithms are modelled to recognize patterns of large amounts of data. The data can be in the form of images, text, signals, time series or any other real-world source. Source Data is then translated and fed through pattern recognition algorithms which processes it and groups them based on similar patterns. Output of the data are the labels of the similar groups. Output labels are in numerical vector form. A NN is comprised of a series of layers with multiple nodes. Nodes in each layer are interconnected where the First layer (Input layer) is the raw data that is used to collect the input patterns and the Last layer (output layer) comprises of the mapped labels outputs. In between the First and Last layer there are series of many hidden layers. The whole process is analogous to the way neurons in the brain works, taking signal from a previous layer and passing it along to the next successive layer. The hidden layers in a NN algorithm work as an inter-connected network where the nth layer is the output of the $(n-1)^{\text{th}}$ layer and is subsequently the input for the $(n+1)^{\text{th}}$ layer.

CNN is a type of neural network that is particularly popular in the image processing realm. In our project we have used CNN for image classification. A CNN is also known as a Deep Learning algorithm which is able to take an input image, recognize its patterns (based on its features, size, aspects, type, etc.) and classify it into an output Class. The algorithms are used to identify differences in the images based on their known features. Pre-processing for CNN may be much lower compared to non-CNN classification algorithms. A CNN is able to apply relevant filters to an image and capture its Spatial and Temporal features. The CNN framework breaks down the complexity of images into a simple and easy to process format, without losing any features, and makes good predictions.

When working on ML problems, the training models are optimized by minimizing the loss function. A good model is determined by its low loss function. CNN models use the Cross-Entropy loss function to optimize its classification.

For a random variable x, with $p(x)$ probability distribution function, Entropy loss function is defined as in equation 1:

$$E(X) = \begin{cases} -\int_x px^{\int p(x) \log_p x} & \text{if } x \text{ is continuous} \\ \sum_x^x p(x) \log_p x & \text{if } x \text{ is discrete} \end{cases}$$

Entropy is the uncertainty level of a random variable x. A higher value of entropy defines the higher uncertainty of the probability distribution and a smaller value defines lesser uncertainty.

The Cross Entropy Loss Function is also called logarithmic loss or log loss and is defined in equation 2, where t_i is the true label and p_i is the SoftMax probability of the i^{th} class, as follows:

$$C(x) = \sum_{i=1}^{n} t_i \log p_i, \text{ for } n \text{ classes}$$

One Hot Encoding of true labels are required for Categorical cross-entropy loss calculation.

Higher complexity models are capable of fitting the data better although sometimes this causes overfitting. Resulting in a higher loss on the testing set. The goal is to make a tradeoff between how well the data fits while minimizing the loss. We used the Hyperparameter Lambda (λ) in a Regularization function. This hyperparameter controls the weight of the penalty. If Lambda (λ) is increased complexity of the model will contribute to higher cost. Which also means that a lower Lambda (λ) biases towards less complex models.

3 Related Work

ML techniques are being widely explored by researchers these days for a plethora of diverse applications. This technique has been applied on various aspects of automation and networking [Boutaba, R et.al. 2018]. It also applies in the diagnosis of lung cancer and COVID-19 using a multi class deep learning model from x-ray and CT scan imagery [Dina M. Ibrahim et.al. 2021]. In another claim, the ML technique is explored for the classification and detection of breast cancer [Sharma S et.al. 2020]. The application of ML techniques is not only limited to health care and automation, but is relevant in diverse fields like e-commerce, criminal investigations, and more. Internet of Things (IoT) and Artificial Intelligence (AI), along with ML Deep Learning, have also been proposed to classify recycled clothing [Sun-Kuk Noh, 2021]. Researchers have made an attempt to recognize people based on their attire, utilizing deep learning concepts to classify clothing images [B. Marianna et.al. 2018]. Likewise, the fashion industry requires classification of clothing images [L. Brian and J. Karthik, 2015]. Significant work has also been done utilizing the concepts of deep learning to retrieve clothing in a huge corpus [K. Lin et.al. 2015]. Machine learning concepts and advancements can substantially improve accuracy in classifying clothing and enhance user experience across social media, fashion world, forensics, and much more.

4 Methodology

Figure 1 depicts the flow of model being generated to classify images of clothing items using machine learning. Fashion-MNIST dataset has been used to test validate the algorithm. In our project, we have created a CNN model with RELU activation function. The steps include feature learning, followed by classification. The input layer comprises of 28 × 28 pixel images, and the output layer is the classifier with 10 possible classes and SoftMax activation function. Model loss is calculated as the Categorical Cross Entropy loss.

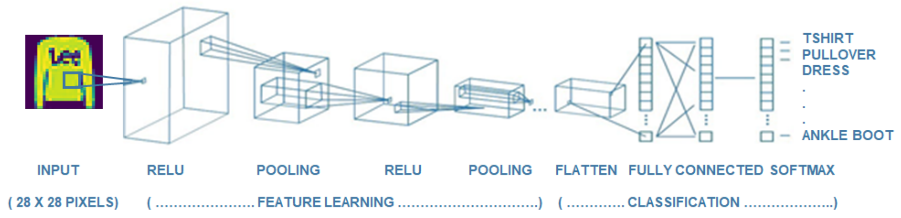

TSHIRT
PULLOVER
DRESS
ANKLE BOOT
INPUT
RELU
POOLING
RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
(28 X 28 PIXELS)
(...................... FEATURE LEARNING ....)
(............ CLASSIFICATION ....)

Figure 1: Block diagram of the model generated to classify images of clothing items

The Fashion-MNIST database has been divided into 60 thousand training sets and 10 thousand testing sets. Each example has been associated with one of the 10 classes' labels and are a standard 28 × 28 grayscale image. Each labeled class represents individual clothing types such as shoes, sandals, shirts, pullovers, dresses, etc. Below is the mapping of 0-9 integers to the class labels:

- 0: Ankle Boot

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

Figure 2 is the sample of the F-MNIST database. Each input image is a 28 × 28 pixel size square.

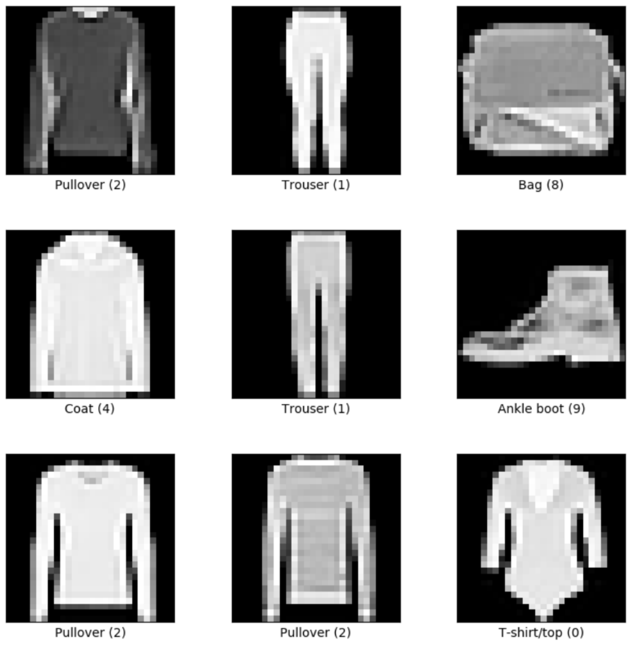


Figure 2: Sample data from Fashion-MNIST database.
Source: `https://www.tensorflow.org/datasets/catalog/fashion_mnist`

The steps followed for classification are as under:

- Neural Network (NN) model is applied to the data

- Testing of model on validation set
- Modified model including the hyper-parameter λ on the RELU activation function layer
- Testing of model for different values of λ

Results obtained are discussed in the subsequent section.

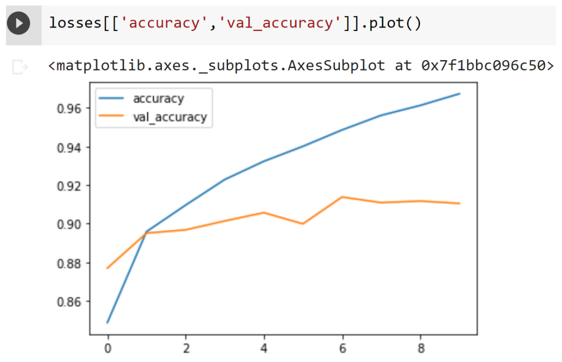
5 Result and Discussion

The results of the validation set model giving the loss accuracy for different hyper-parameters are represented in Table 1.

Hyper-parameter	Loss	Accuracy
none	0.3162	0.9125
0.00	0.2827	0.8996
0.0002	0.3397	0.8975
0.0004	0.3723	0.8932
0.00045	0.3957	0.8794
0.0005	0.3827	0.8881
0.0006	0.3911	0.8869

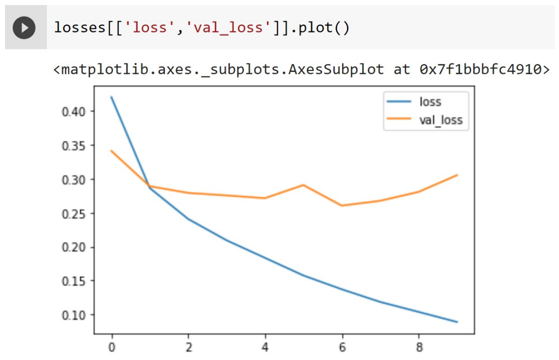
Table 1: Results of the ML model on the validation set

The model with no hyperparameter had an accuracy of over 95% and a loss of less than 10%. However, when tested on the validation set, the validation accuracy reduced to 90% with a loss of 32%. We see that the model is overfitting the data and, hence, it performed low on the validation set.



```
losses[['accuracy','val_accuracy']].plot()
<matplotlib.axes._subplots.AxesSubplot at 0x7f1bbc096c50>
```

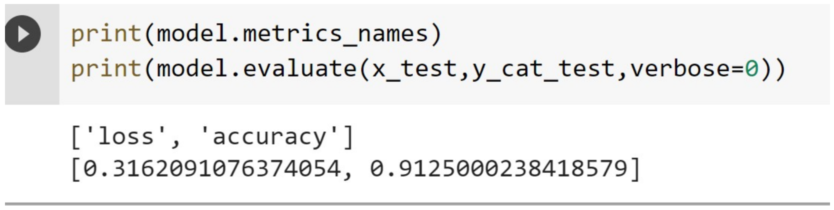
Figure 3: Plot of training set accuracy vs validation set accuracy for no hyperparameter



```
losses[['loss','val_loss']].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1bbbfc4910>
```

Figure 4: Plot of training set loss vs validation set loss for no hyperparameter



```
print(model.metrics_names)
print(model.evaluate(x_test,y_cat_test,verbose=0))
```

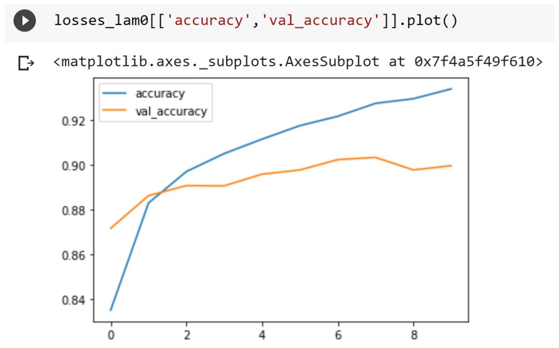
```
['loss', 'accuracy']
[0.3162091076374054, 0.9125000238418579]
```

Figure 5: Metrics for loss and accuracy on validation set for no hyperparameter

We modified the model to include the hyperparameter λ on the RELU activation function layer. We tested the model for different values of λ, as follows:

$\lambda = 0.0$:

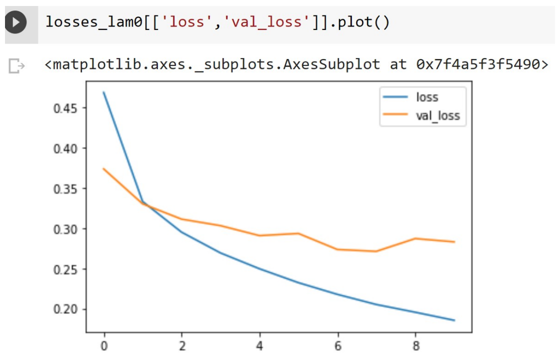
Training accuracy measured at 93%, with a loss of 20%. Validation accuracy about 90% with a loss of 28%. This was an improvement from the original model.



```
losses_lam0[['accuracy','val_accuracy']].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4a5f49f610>
```

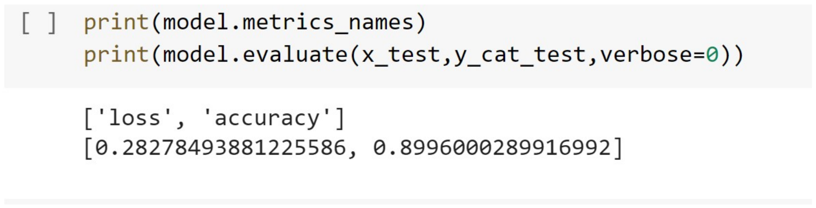
Figure 6: Plot of training set accuracy vs validation set accuracy for hyperparameter $\lambda = 0.0$



```
losses_lam0[['loss','val_loss']].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4a5f3f5490>
```

Figure 7: Plot of training set loss vs validation set loss for hyperparameter λ = 0.0



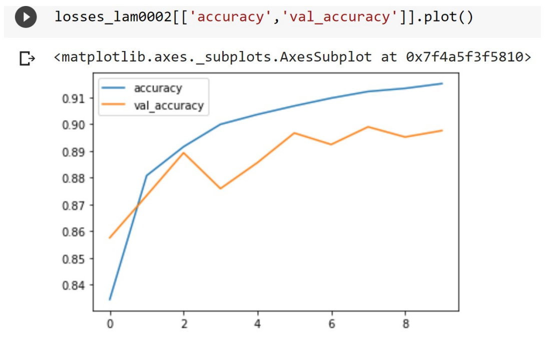
```
[ ] print(model.metrics_names)
    print(model.evaluate(x_test,y_cat_test,verbose=0))
```

```
['loss', 'accuracy']
[0.28278493881225586, 0.8996000289916992]
```

Figure 8: Metrics for loss and accuracy on validation set for hyperparameter λ = 0.0

We increased values of the Hyperparameter and following are the results:
λ = 0.0002:

Training accuracy was 92% with a loss of 29%. Validation accuracy was about 90% with a loss of 34%



```
losses_lam0002[['accuracy','val_accuracy']].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4a5f3f5810>
```

Figure 9: Plot of training set accuracy vs validation set accuracy for hyperparameter λ = 0.0002

```
[ ]  losses_lam0002[['loss','val_loss']].plot()

     <matplotlib.axes._subplots.AxesSubplot at 0x7f4a5c58de10>
```

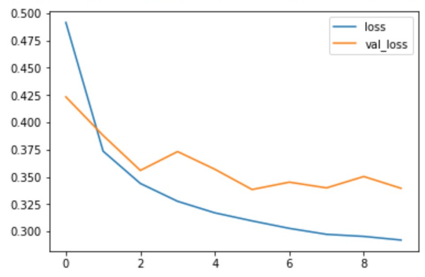


Figure 10: Plot of training set loss vs validation set loss for hyperparameter $\lambda = 0.0002$

```
[ ]  print(model.metrics_names)
     print(model.evaluate(x_test,y_cat_test,verbose=0))

     ['loss', 'accuracy']
     [0.33957138657569885, 0.897599995136261]
```

Figure 11: Metrics for loss and accuracy on validation set for hyperparameter $\lambda = 0.0002$

$\lambda = 0.0004$:

Training accuracy was 91% with a loss of 33%. Validation accuracy was about 89% with a loss of 37%.

```
losses_lam0004[['accuracy','val_accuracy']].plot()

<matplotlib.axes._subplots.AxesSubplot at 0x7f4a57ecf890>
```

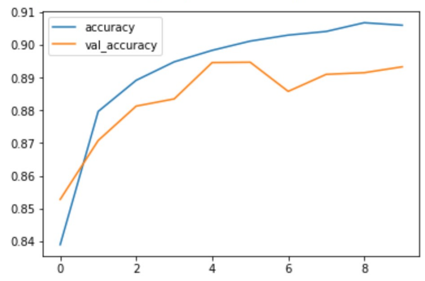


Figure 12: Plot of training set accuracy vs validation set accuracy for hyperparameter $\lambda = 0.0004$

```
[ ] losses_lam0004[['loss','val_loss']].plot()

<matplotlib.axes._subplots.AxesSubplot at 0x7f4a6859e110>
```

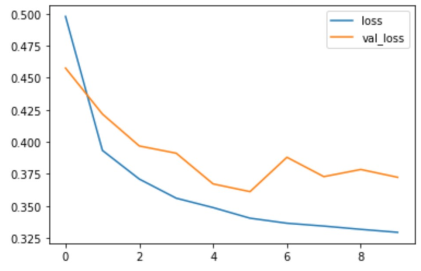


Figure 13: Plot of training set loss vs validation set loss for hyperparameter $\lambda = 0.0004$

```
[ ] print(model.metrics_names)
    print(model.evaluate(x_test,y_cat_test,verbose=0))

    ['loss', 'accuracy']
    [0.37239134311676025, 0.8932999968528748]
```

Figure 14: Metrics for loss and accuracy on validation set for hyperparameter $\lambda = 0.0004$

$\lambda = 0.0005$:

Training accuracy was 90% with a loss of 34%. Validation accuracy was about 89% with a loss of 38%.

```
losses_lam0005[['accuracy','val_accuracy']].plot()

<matplotlib.axes._subplots.AxesSubplot at 0x7f4b42709690>
```

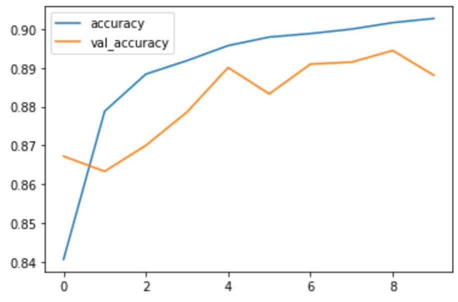


Figure 15: Plot of training set accuracy vs validation set accuracy for hyperparameter $\lambda = 0.0005$

```
losses_lam0005[['loss','val_loss']].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4b42709510>
```

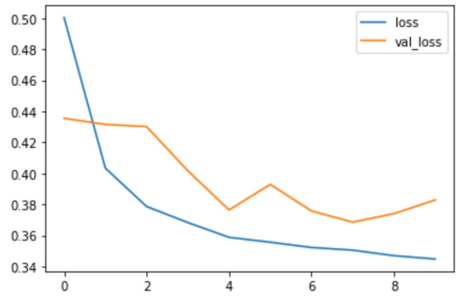


Figure 16: Plot of training set loss vs validation set loss for hyperparameter λ = 0.0005

```
print(model.metrics_names)
print(model.evaluate(x_test,y_cat_test,verbose=0))
```

```
['loss', 'accuracy']
[0.38279515504837036, 0.8881000280380249]
```

Figure 17: Metrics for loss and accuracy on validation set for hyperparameter $\lambda = 0.0005$

$\lambda = 0.0006$:

Training accuracy was 90% with a loss of 35%. Validation accuracy was about 89% with a loss of 39%.

```
losses_lam0006[['accuracy','val_accuracy']].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4a54477b10>
```

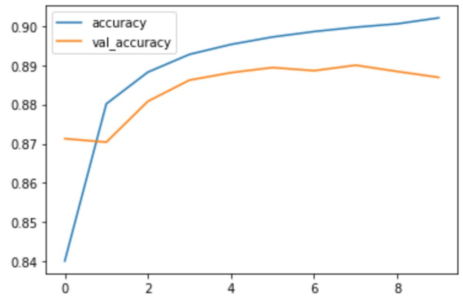


Figure 18: Plot of training set accuracy vs validation set accuracy for hyperparameter $\lambda = 0.0006$

```
losses_lam0006[['loss','val_loss']].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4a55439110>
```

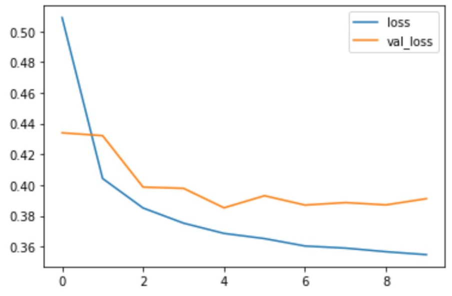


Figure 19: Plot of training set loss vs validation set loss for hyperparameter λ = 0.0006

```
print(model.metrics_names)
print(model.evaluate(x_test,y_cat_test,verbose=0))
```

```
['loss', 'accuracy']
[0.391184538602829, 0.886900007724762]
```

Figure 20: Metrics for loss and accuracy on validation set for hyperparameter λ = 0.0006

Hyper-parameter	Loss	Accuracy
none	0.3162	0.9125
0.00	**0.2827**	**0.8996**
0.0002	0.3397	0.8975
0.0004	0.3723	0.8932
0.00045	0.3957	0.8794
0.0005	0.3827	0.8881
0.0006	0.3911	0.8869

Table 2: Results of the model with hyper-parameter λ

In our data set, we notice that the initial model is overfitting the data. However, when the hyperparameter λ is applied to the model, the loss reduces without bringing the accuracy down too much.

6 Conclusions

The concept of Convoluted Neural Network has been applied to different clothing images, leading to the creation of a classification model. We noticed that the model loss reduces with the addition of the hyper-parameter lambda (λ) in the regularization function. However, upon increasing the value of lambda, there is no significant improvement on the model. Hence, we can conclude that $\lambda = 0.0$ is the best model with the minimum loss. It is evident that the developed model in this work for classification of clothing has reduced loss and can find utility across varied industries and applications.

References

[BK15] L. Brian and J. Karthik. Convolutional neural networks for fashion classification and object detection. *http://cs231n.stanford.edu/*, 2015.

[BMvEE18] G. Zeno B. Marianna and v. E. Erwin. Clothing identification via deep learning: forensic applications. *Forensic Sciences Research, vol. 3, no. 3, pp. 219–229*, 2018.

[Bou18] Salahuddin M.A. Limam N. et al. Boutaba, R. A comprehensive survey on ml for networking: evolution, applications and research opportunities. *J Internet Serv Appl 9, 16*, 2018.

[DMI21] Amany M. Sarhan Dina M. Ibrahim, Nada M. Elshennawy. Deep-chest: Multi-classification deep learning model for diagnosing covid-19, pneumonia, and lung cancer chest diseases. *Computers in Biology and Medicine, Volume 132, 2021, 104348, ISSN 0010-4825*, 2021.

[KLC15] K.-H. Liu J.-H. Hsiao K. Lin, H.-F. Yang and C.-S. Chen. Rapid clothing retrieval via deep learning of binary codes and hierarchical search. *in Proceedings of ICMR'15, Shanghai, China*, 2015.

[Noh21] Sun-Kuk Noh. Recycled clothing classification system using intelligent iot and deep learning with alexnet. *Computational Intelligence and Neuroscience, vol. 2021, Article ID 5544784*, 2021.

[SS20] Mehra R. Sharma S. Conventional ml and deep learning approach for multi-classification of breast cancer histopathology images-a comparative insight. *Journal of Digital Imaging*, 2020.