DL-Ops

Assignment 4 - Report

Debonil Ghosh
Roll No: M21AIE225
Executive MTech
Artificial Intelligence
Indian Institute of Technology, Jodhpur

Problem Statement:

Question 1 [50 marks]

Train a Conditional Deep Convolutional Generative Adversarial Network (cDCGAN) on Dataset. (You may use this for Reference) [25 Marks]

A. Generate 50 Samples of each class from your trained generator. [5 Marks]

B. Train a ResNet18 classifier on the given dataset and treat the generated samples as test dataset and report following [20 Marks]

1. F1 Score for each class
2. Confusion matrix

Reference: https://learnopencv.com/conditional-gan-cgan-in-pytorch-and-tensorflow/

Questions 2 [50 marks]

Train a CNN based classification model and perform Optimized Hyperparameter Tuning using Optuna Library on the below-mentioned dataset. Perform 100 trials.

Hyperparameters should be

1) No of Convolution Layers 3 to 6
2) Number of Epochs 10 to 50
3) Learning rate 0.0001 to 0.1

Report the observations and the best trial. Report how many trials were pruned

For Even Roll Number MNIST

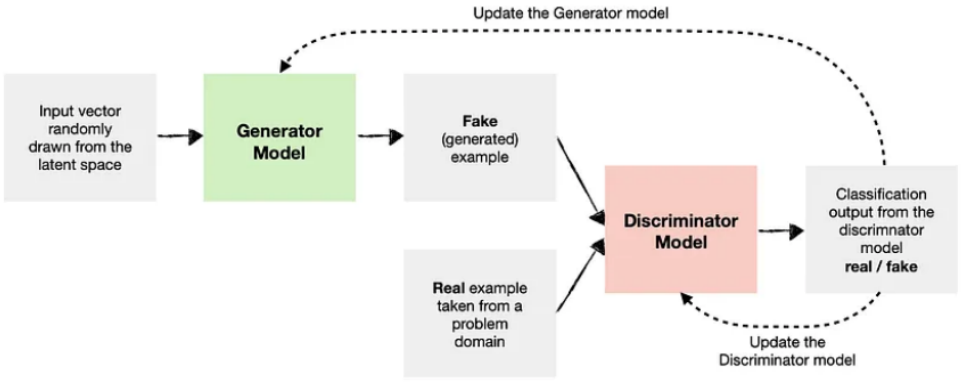
For Odd Roll Number Fashion MNIST

Reference: https://github.com/elena-ecn/optuna-optimization-for-PyTorch-CNN

Question 1

Generative Adversarial Network (GAN):

Generative Adversarial Network (GAN) is a type of artificial neural network architecture that involves two neural networks, a generator and a discriminator, working in tandem to generate synthetic data that is similar to a given dataset.



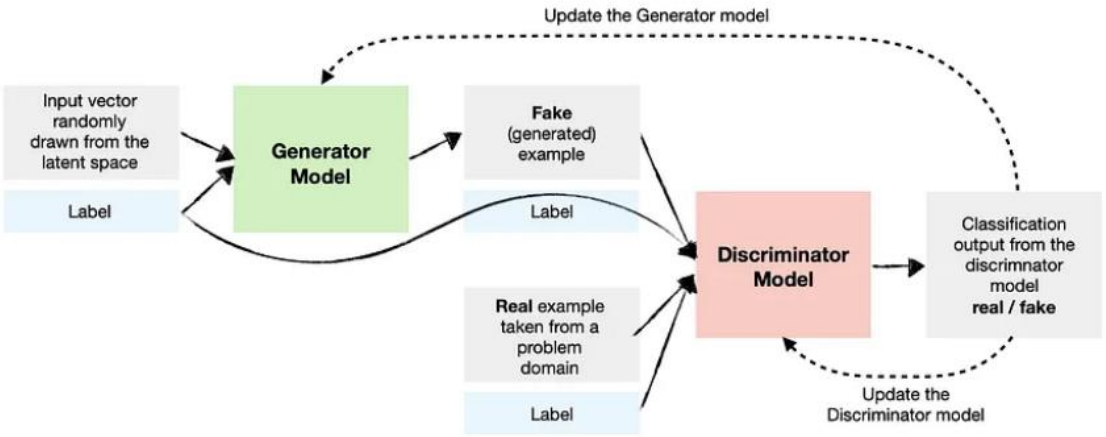
Basic GAN model architecture.

The generator network generates fake data, while the discriminator network tries to differentiate between real and fake data. The two networks are trained simultaneously, with the generator trying to generate increasingly realistic data, and the discriminator getting better at distinguishing between real and fake data.

GANs have a wide range of applications, including image and video generation, data augmentation, and data synthesis for training other machine learning models. However, GANs can be difficult to train and require careful tuning of hyperparameters to achieve good results.

Despite their challenges, GANs have proven to be a powerful tool in the field of machine learning and have led to significant advancements in generating realistic and high-quality data.

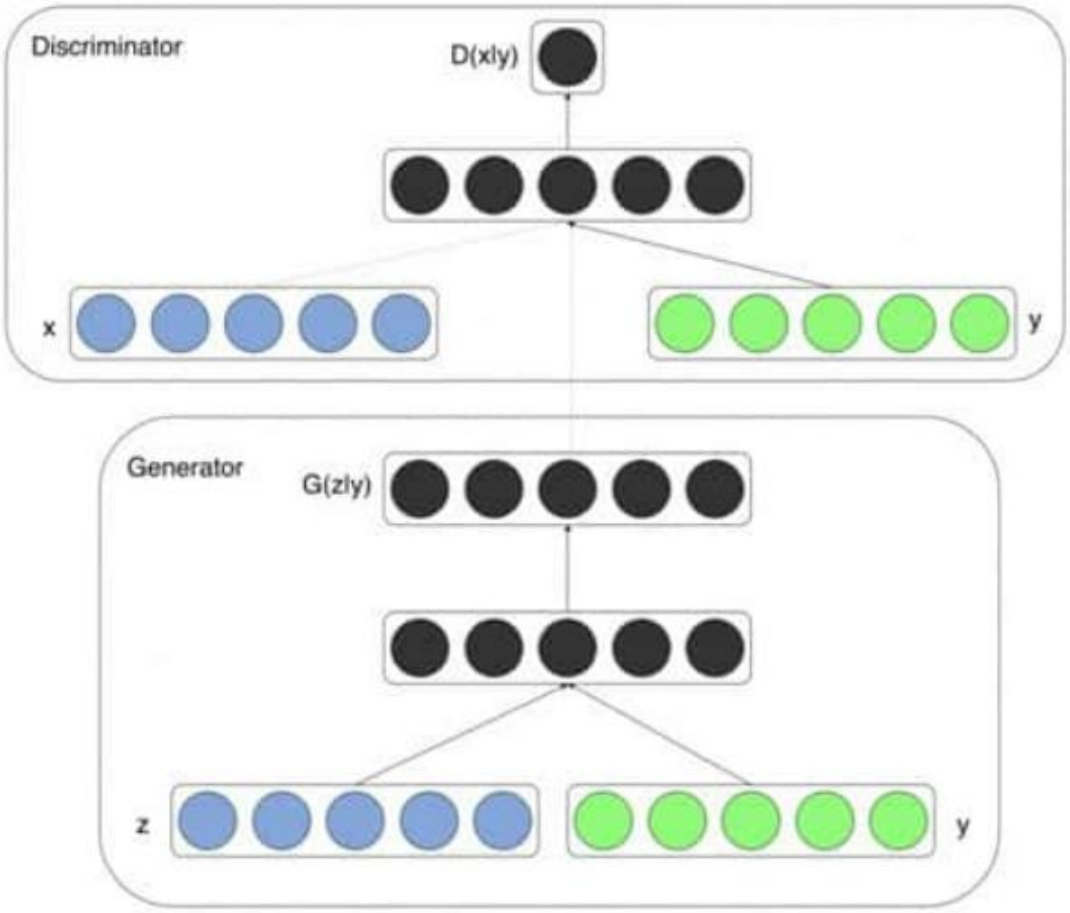
Conditional Deep Convolutional Generative Adversarial Network (cDCGAN):



Conditional GAN (cGAN) model architecture.

A Conditional Deep Convolutional Generative Adversarial Network (cDCGAN) is a type of neural network architecture used for image synthesis tasks. It is an extension of the Deep Convolutional Generative Adversarial Network (DCGAN), which is a type of GAN that uses convolutional neural networks (CNNs) for both the generator and discriminator.

In cDCGAN, the generator network takes in not only a random noise vector as input but also a conditional vector that represents some additional information, such as class labels or attributes of the images to be generated. The conditional vector is concatenated with the noise vector and then fed through the generator network to produce the synthetic image.

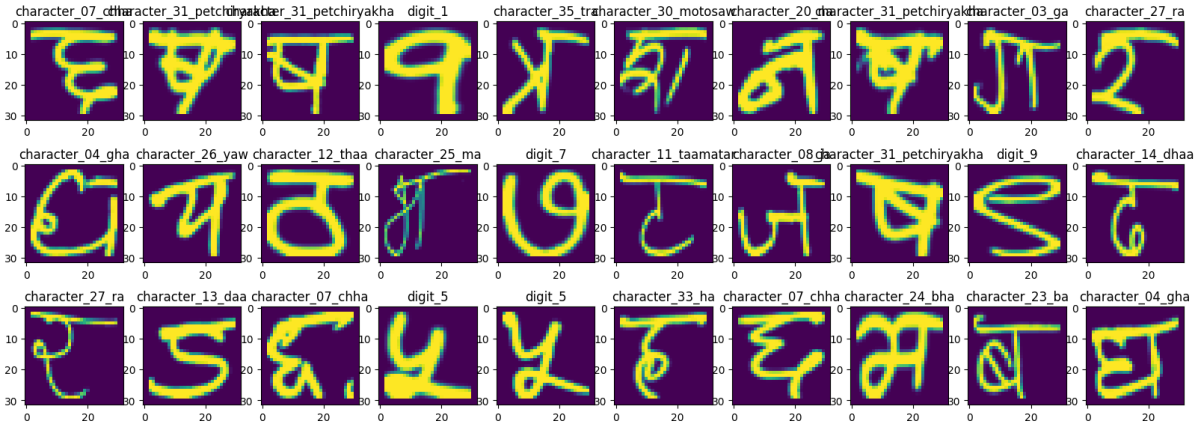


The discriminator network also takes in the same conditional vector and uses it to classify whether the input image is real or fake. The conditional vector serves as a form of supervision that helps guide the generator network to produce images that are more consistent with the specified attributes.

By incorporating conditional information into the GAN architecture, cDCGAN can generate images with greater control and specificity. It has been used for a wide range of applications, including image synthesis, style transfer, and data augmentation.

Given Dataset:

The CIFAR-10 dataset consists of **92000 32x32 grayscale images** in **46 classes**, with 2000 images per class. Dataset finally split into **73600 training** images and **18400 test images**.



Torch Transformations Applied:

- Normalize to (0.5, 0.5, 0.5), (0.5, 0.5, 0.5)
- Transform to tensor

Using default PyTorch data-loader function with the dataset path ‘./data’ and used above mentioned transforms for pre-processing.

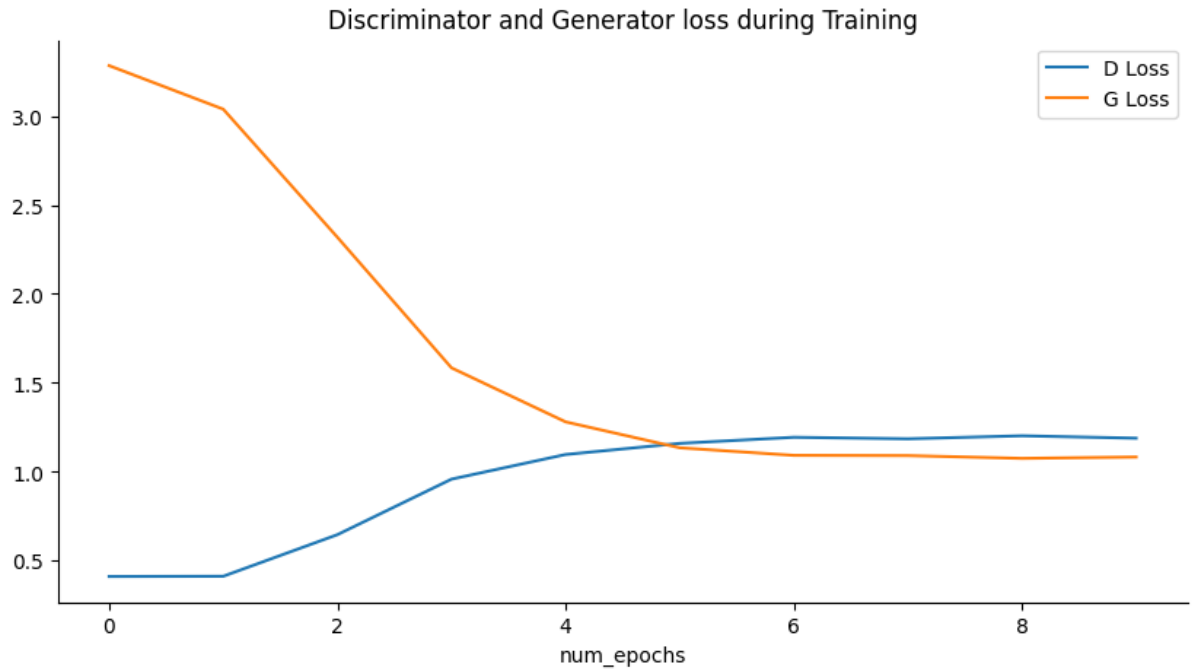
Hyper Parameters Used for cDCGAN training:

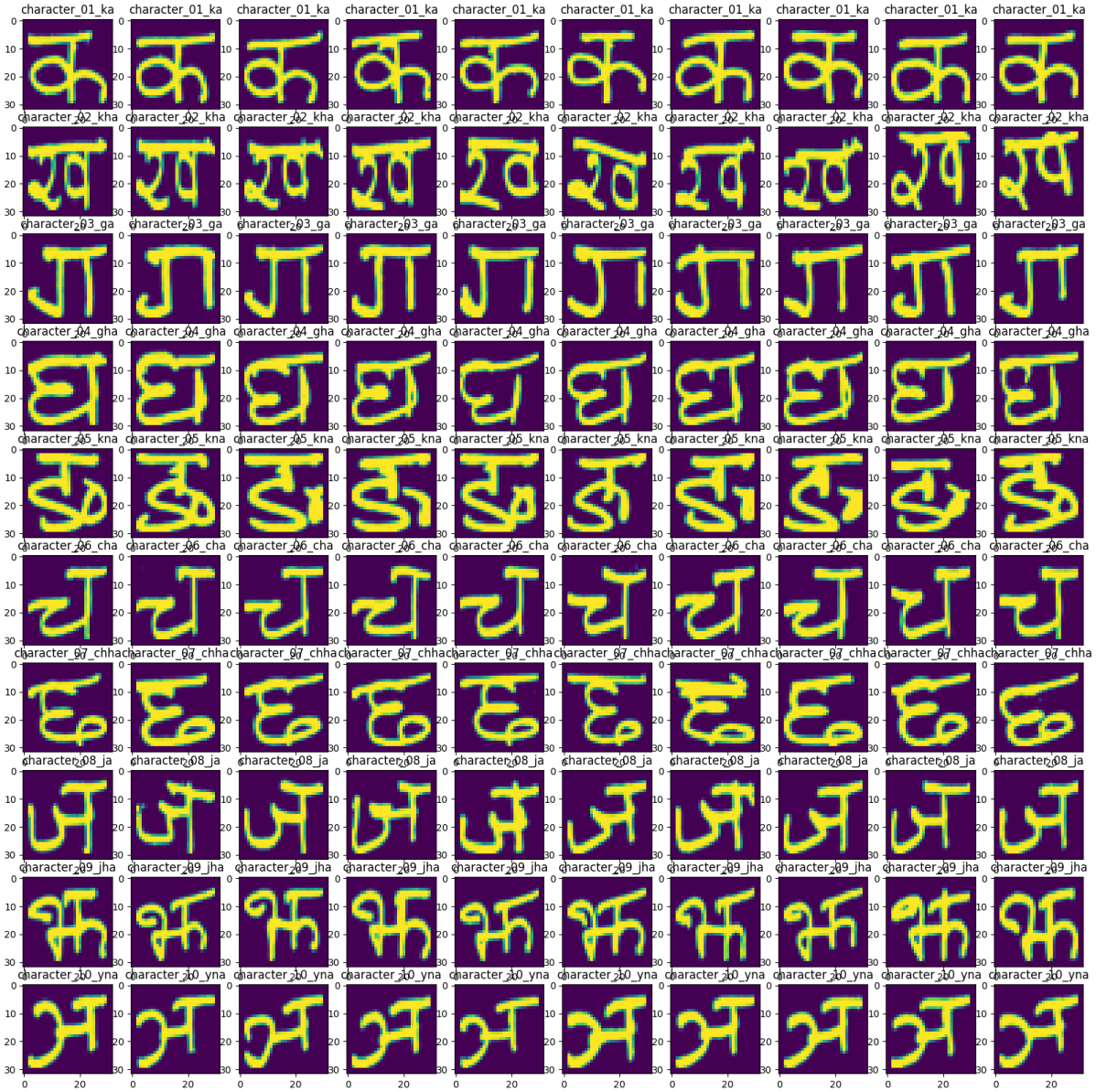
Hyper Parameter	Value
Latent Size	100
Number of Epochs	10
Batch Size	128
Loss Function	BinaryCrossEntropyLoss
Optimizer	Adam
Learning Rate	2e-4
Adam Beta	0.5

Training Logs:

```
Epoch 1/10 Discriminator Loss 0.411 Generator Loss 3.284 D(x) 0.844
D(G(x)) 0.070
 Epoch 2/10 Discriminator Loss 0.412 Generator Loss 3.039 D(x) 0.840
D(G(x)) 0.071
 Epoch 3/10 Discriminator Loss 0.646 Generator Loss 2.319 D(x) 0.769
D(G(x)) 0.159
 Epoch 4/10 Discriminator Loss 0.958 Generator Loss 1.584 D(x) 0.672
D(G(x)) 0.267
 Epoch 5/10 Discriminator Loss 1.097 Generator Loss 1.280 D(x) 0.624
D(G(x)) 0.325
 Epoch 6/10 Discriminator Loss 1.159 Generator Loss 1.135 D(x) 0.599
D(G(x)) 0.357
 Epoch 7/10 Discriminator Loss 1.194 Generator Loss 1.092 D(x) 0.586
D(G(x)) 0.368
 Epoch 8/10 Discriminator Loss 1.185 Generator Loss 1.091 D(x) 0.589
D(G(x)) 0.366
 Epoch 9/10 Discriminator Loss 1.202 Generator Loss 1.075 D(x) 0.582
D(G(x)) 0.370
 Epoch 10/10 Discriminator Loss 1.189 Generator Loss 1.083 D(x) 0.585
D(G(x)) 0.367
```

Discriminator vs Generator Loss:




character_01_ka
character_02_kha
character_03_ga
character_04_gha
character_05_kna
character_06_cha
character_07_chha
character_08_ja
character_09_jha
character_10_yna

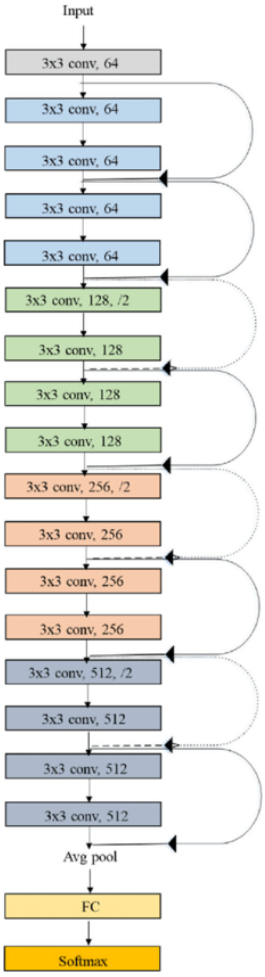
B. Train a ResNet18 classifier on the given dataset and treat the generated samples as test dataset and report following [20 Marks]

1. **F1 Score for each class**
2. **Confusion matrix**

Hyper Parameters Used:

Hyper Parameter	Value
Learning Rate	1e-5
Number of Epochs	10
Batch Size	64
Loss Function	CrossEntropyLoss
Optimizer	Adam

Original RESNET18 Architecture:



As per problem statement, initial and final layer of the architecture has been modified.

ResNet(

(conv1): Conv2d(1, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)

(bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu): ReLU(inplace=True)

.

.

.

(avgpool): AdaptiveAvgPool2d(output_size=(1, 1))

(fc): Linear(in_features=512, out_features=46, bias=True)

)

Model Training details

Logs:

Epoch: 1 (0m 58s) Training Loss: 2.827,Test Loss: 1.724, Training acc: 0.295,Test acc: 0.551,

Epoch: 2 (1m 57s)Training Loss: 1.171,Test Loss: 0.734, Training acc: 0.687,Test acc: 0.797,

Epoch: 3 (2m 57s)Training Loss: 0.568,Test Loss: 0.410, Training acc: 0.840,Test acc: 0.884,

Epoch: 4 (3m 58s)Training Loss: 0.348,Test Loss: 0.265, Training acc: 0.900,Test acc: 0.925,

Epoch: 5 (5m 2s)Training Loss: 0.238,Test Loss: 0.184, Training acc: 0.931,Test acc: 0.948,

Epoch: 6 (6m 4s)Training Loss: 0.172,Test Loss: 0.135, Training acc: 0.950,Test acc: 0.961,

Epoch: 7 (7m 8s)Training Loss: 0.131,Test Loss: 0.104, Training acc: 0.962,Test acc: 0.971,

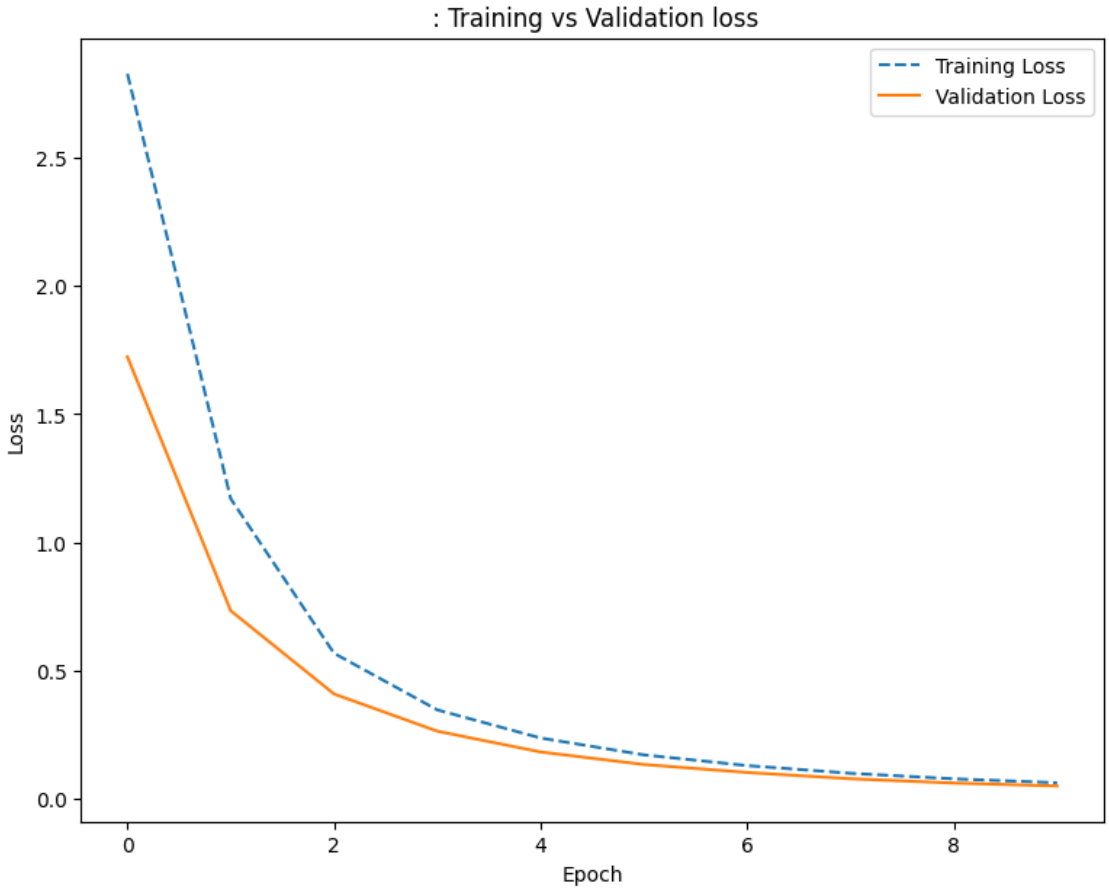
Epoch: 8 (8m 13s)Training Loss: 0.101,Test Loss: 0.079, Training acc: 0.971,Test acc: 0.978,

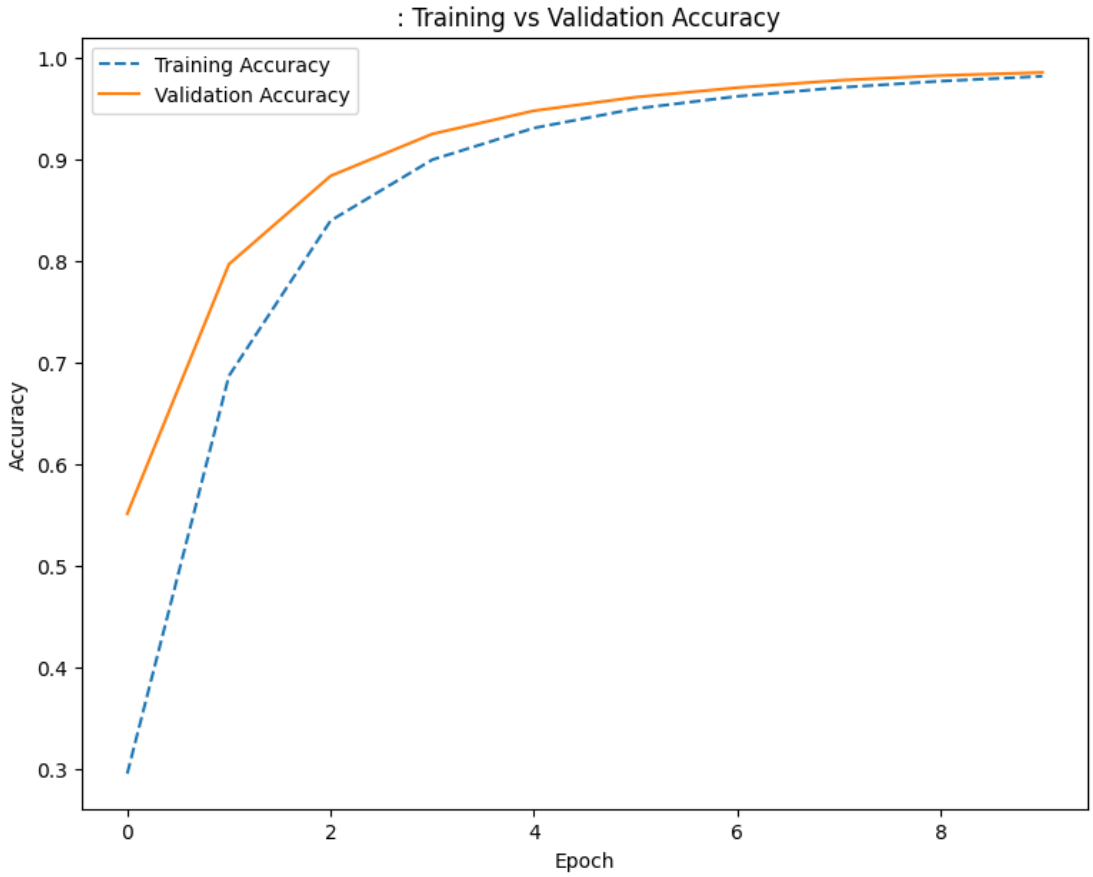
Epoch: 9 (9m 19s)Training Loss: 0.079,Test Loss: 0.062, Training acc: 0.977,Test acc: 0.982,

Epoch: 10 (10m 24s)Training Loss: 0.064,Test Loss: 0.051, Training acc: 0.982,Test acc: 0.985,

Training completed in 10m 24sTraining Loss: 0.064,Test Loss: 0.051, Training acc: 0.98,Test acc: 0.99,

Training loss and accuracy curve:



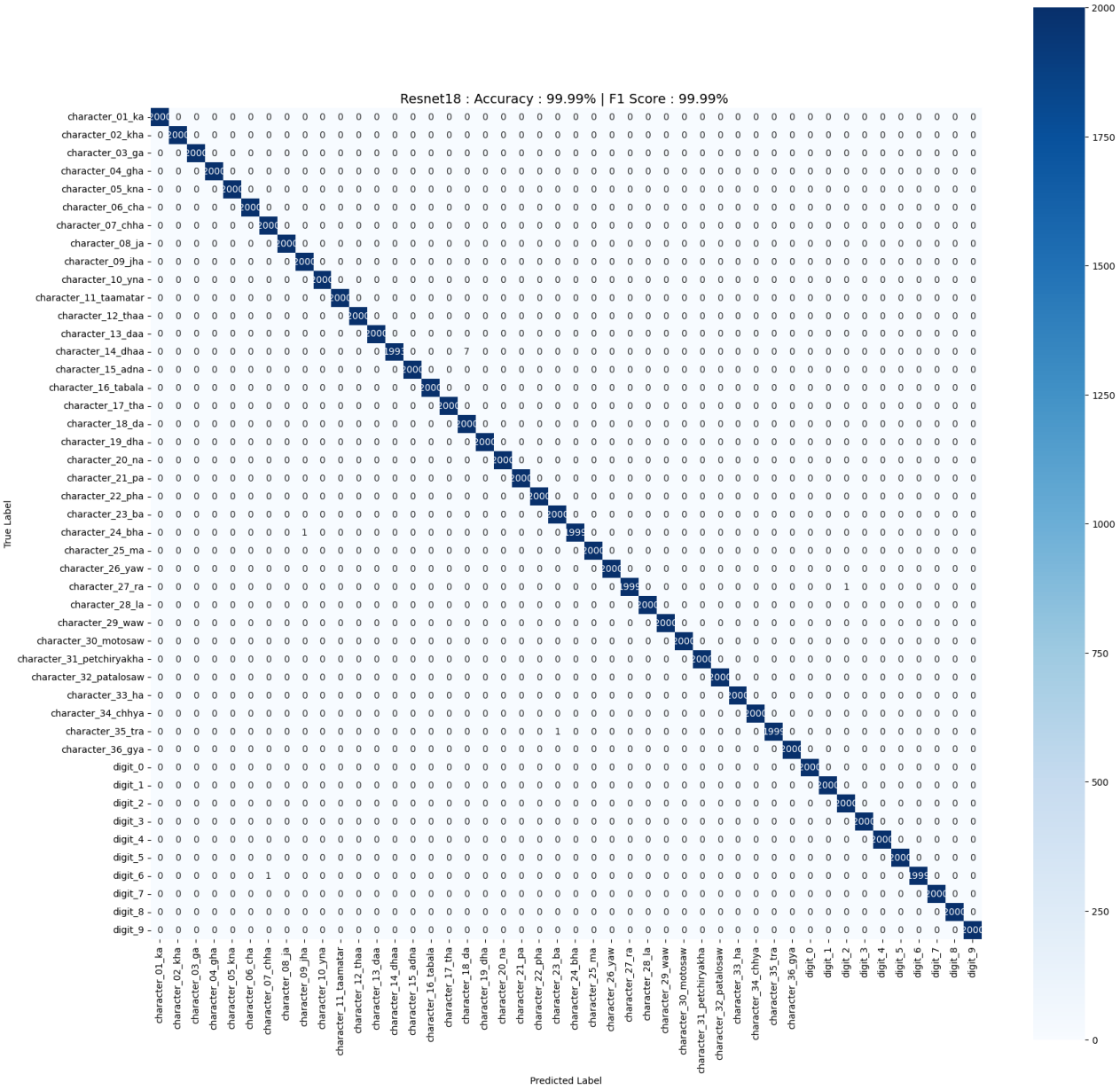


Model Evaluation Summary:

Accuracy: 99.988%

F1 Score: 99.988%

Confusion Matrix:



F1 Score for each class:

Class	F1 Score
character_01_ka	1
character_02_kha	1
character_03_ga	1

character_04_gha	1
character_05_kna	1
character_06_cha	1
character_07_chha	0.999750062
character_08_ja	1
character_09_jha	0.999750062
character_10_yna	1
character_11_taamatar	1
character_12_thaa	1
character_13_daa	1
character_14_dhaa	0.998246932
character_15_adna	1
character_16_tabala	1
character_17_tha	1
character_18_da	0.998253057
character_19_dha	1
character_20_na	1
character_21_pa	1
character_22_pha	1
character_23_ba	0.999750062
character_24_bha	0.999749937
character_25_ma	1
character_26_yaw	1
character_27_ra	0.999749937
character_28_la	1
character_29_waw	1
character_30_motosaw	1
character_31_petchiryakha	1
character_32_patalosaw	1
character_33_ha	1
character_34_chhya	1
character_35_tra	0.999749937
character_36_gya	1
digit_0	1
digit_1	1
digit_2	0.999750062
digit_3	1
digit_4	1
digit_5	1
digit_6	0.999749937
digit_7	1
digit_8	1
digit_9	1

Question 2[50 marks]

Train a CNN based classification model and perform Optimized Hyperparameter Tuning using Optuna Library on the below-mentioned dataset. Perform 100 trials.

Hyperparameters should be

1) No of Convolution Layers 3 to 6
2) Number of Epochs 10 to 50
3) Learning rate 0.0001 to 0.1

Report the observations and the best trial. Report how many trials were pruned

For Even Roll Number MNIST

For Odd Roll Number Fashion MNIST

Solution:

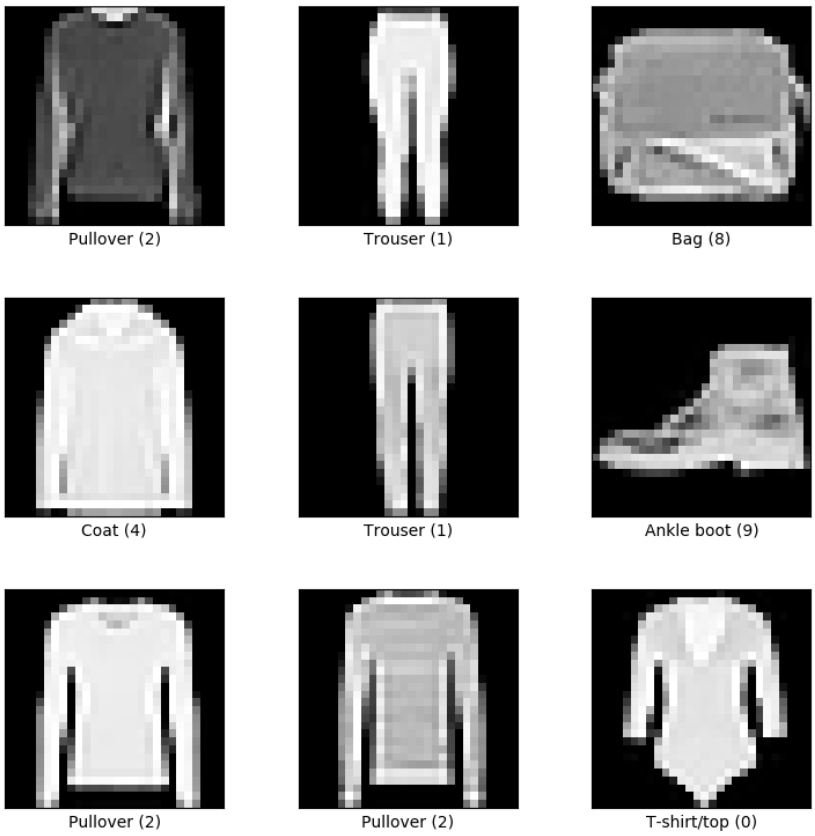
Optuna

Optuna is an automatic hyperparameter optimization software framework, particularly designed for machine learning.

As Roll number M21AIE225 is odd, assigned Dataset is FashionMNIST

FashionMNIST Dataset:

The FashionMNIST dataset consists of **60000 28x8 grayscale images** in **10 classes**, with 6000 images per class. There are **50000 training** images and **10000 test images**.



Torch Transformations Applied:

- Normalize to (0.5, 0.5, 0.5), (0.5, 0.5, 0.5)
- Transform to tensor

Using default PyTorch data-loader function with the dataset path ‘./data’ and used above mentioned transforms for pre-processing.

Hyper Parameters Used for training:

Hyper Parameter	Value
No of Convolution Layer	3 - 6
Number of Epochs	10 - 50
Batch Size	128
Loss Function	CrossEntropyLoss
Optimizer	Adam
Learning Rate	1e-5 – 0.1

Final result found after Optuna optimization:

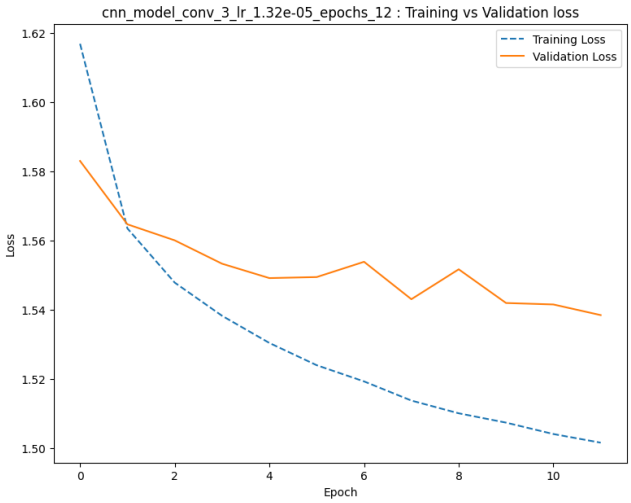
Hyper Parameter	Value
Number of finished trials	100
Number of pruned trials	85
Number of complete trials	15
Best Accuracy Score	**0.9261**

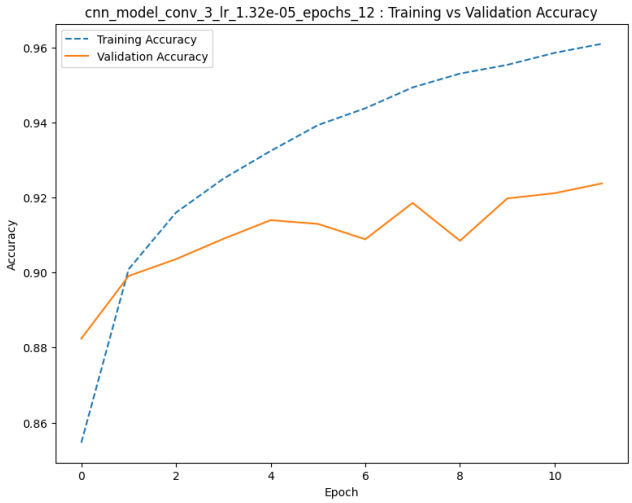
Best Hyper Parameters found:

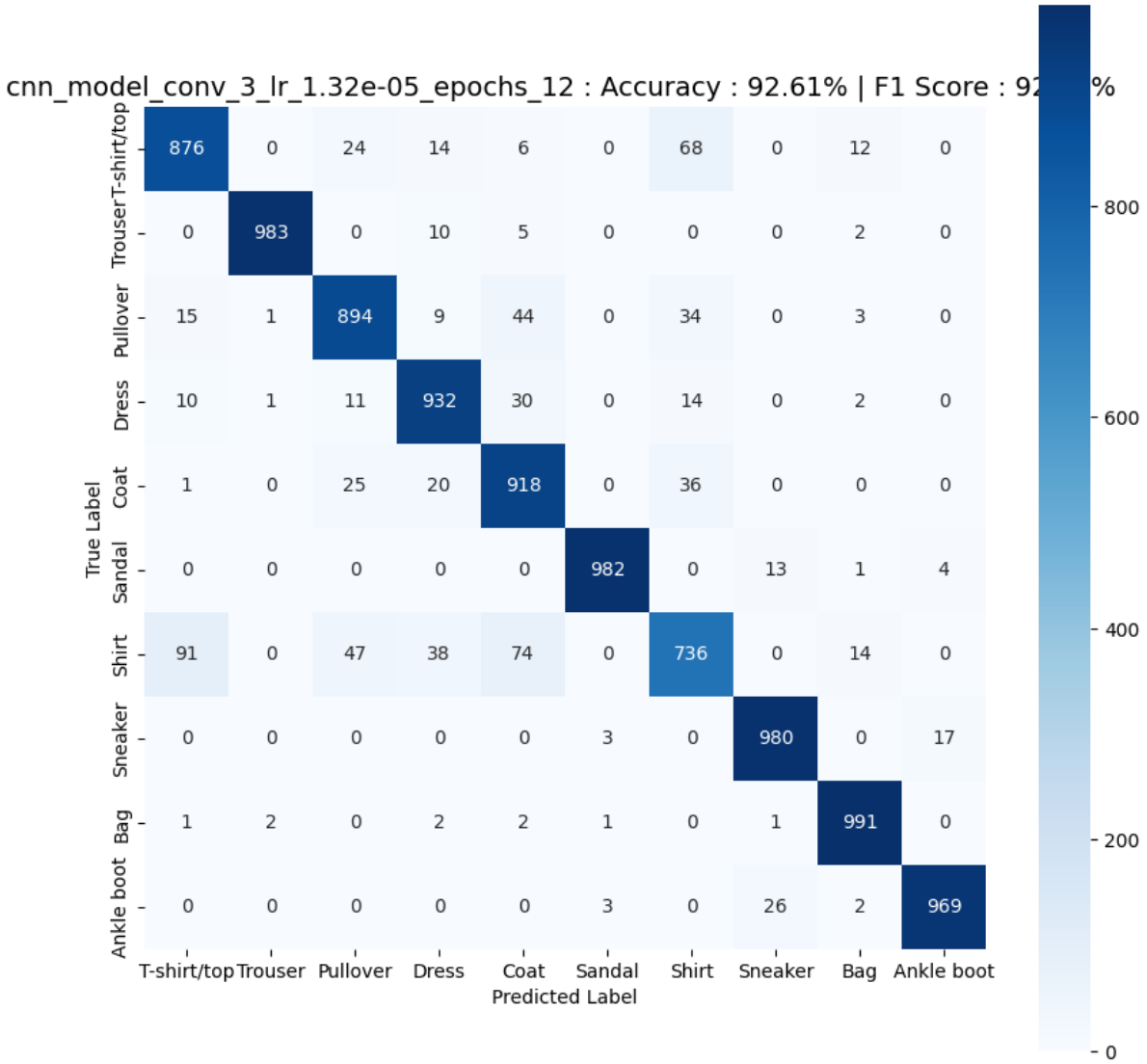
Hyper Parameter	Value
No of Convolution Layer	**3**
Number of Epochs	**12**
Learning Rate	**1.3190817110540418e-05**

Complete run log included with files

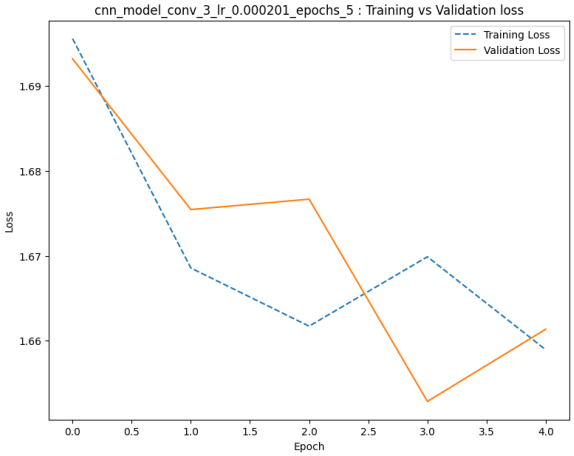
Details of Best Model found:

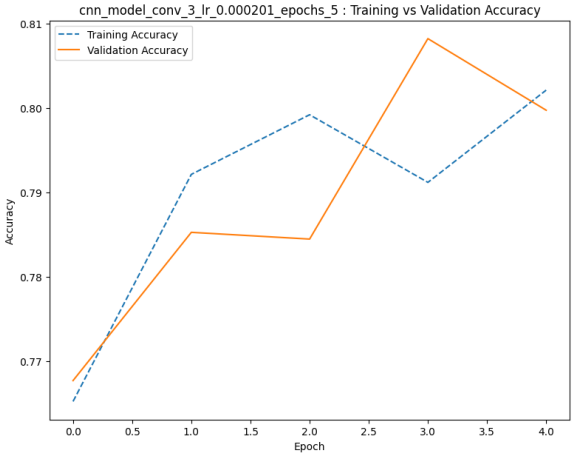


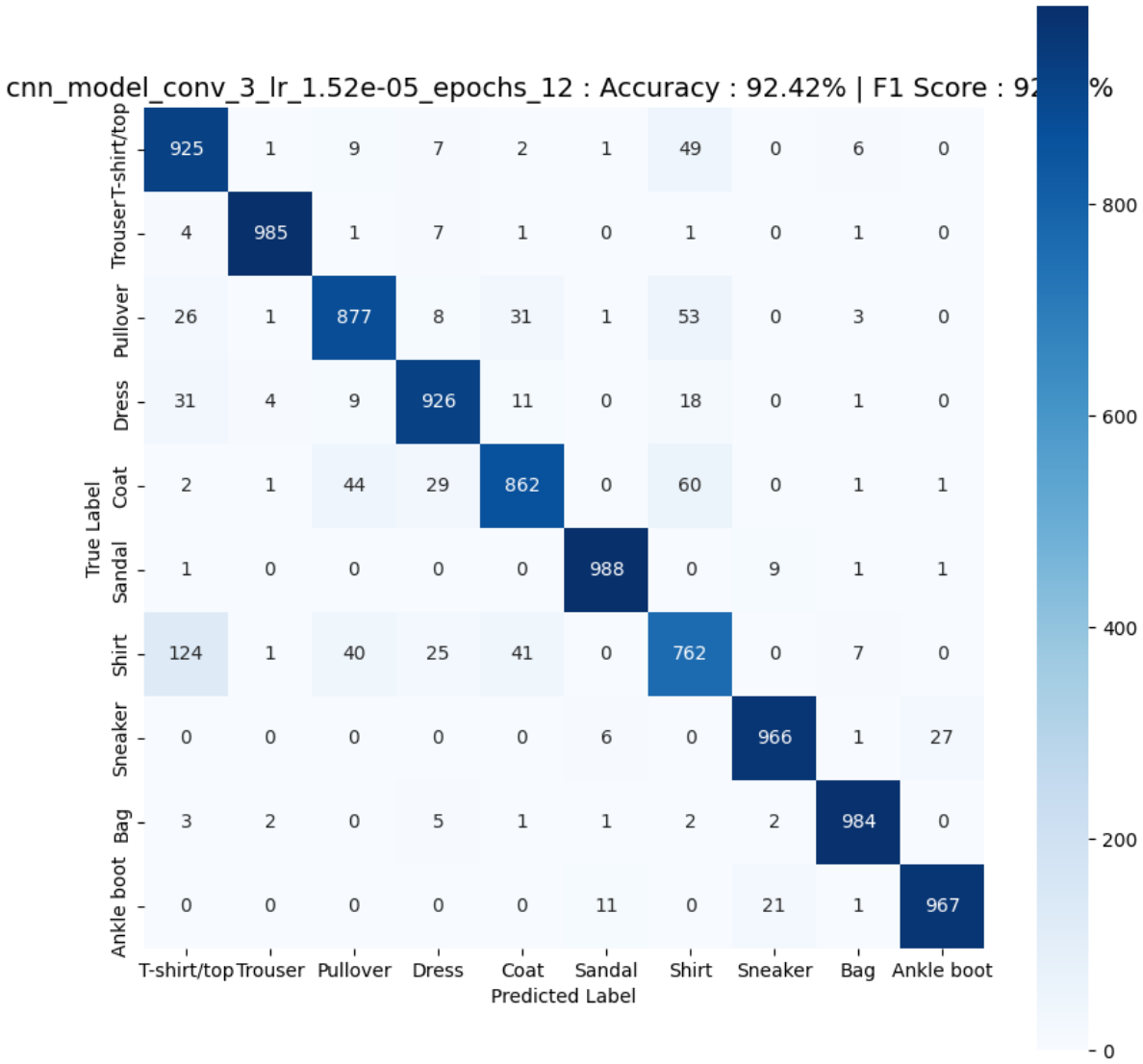


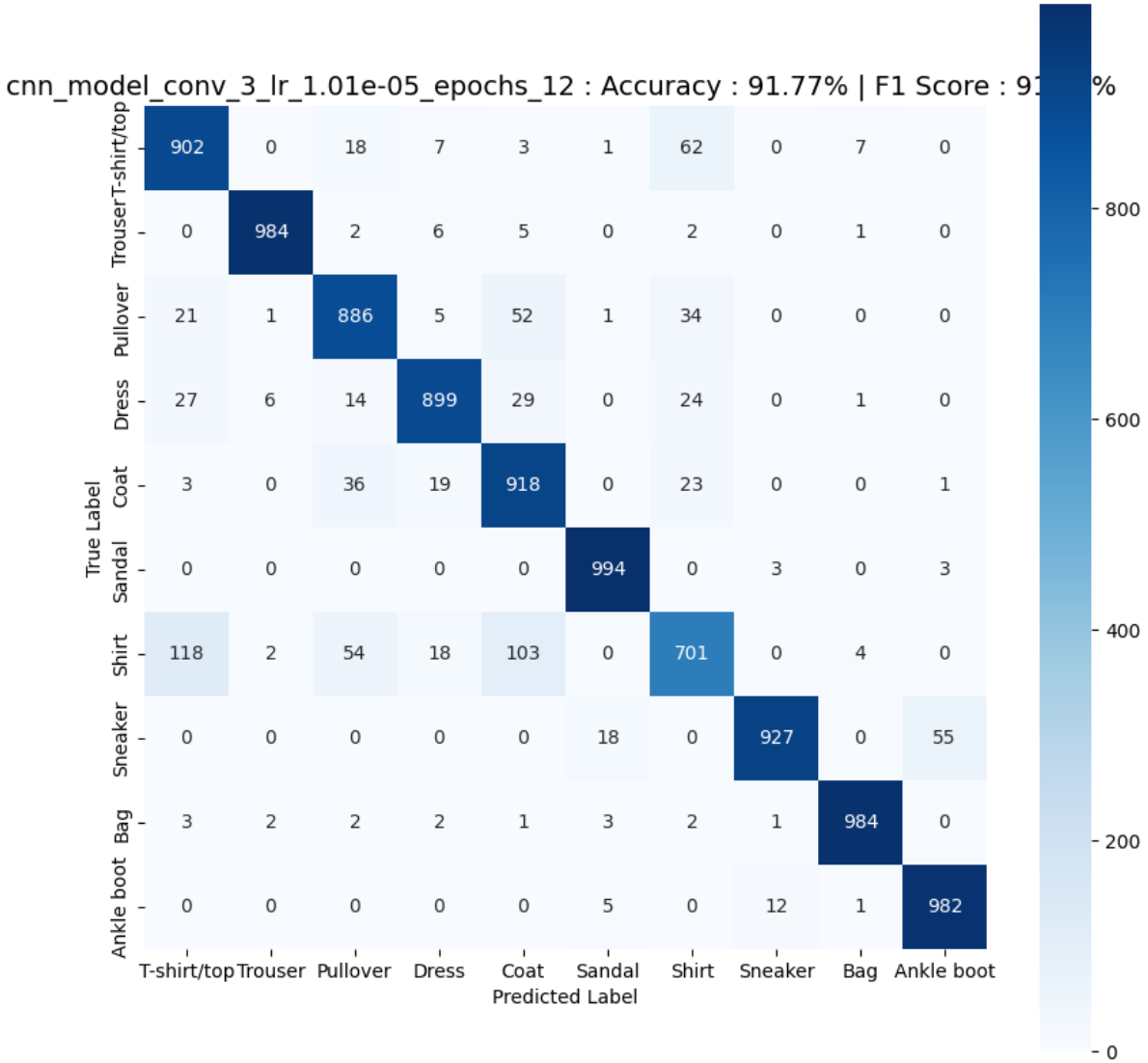


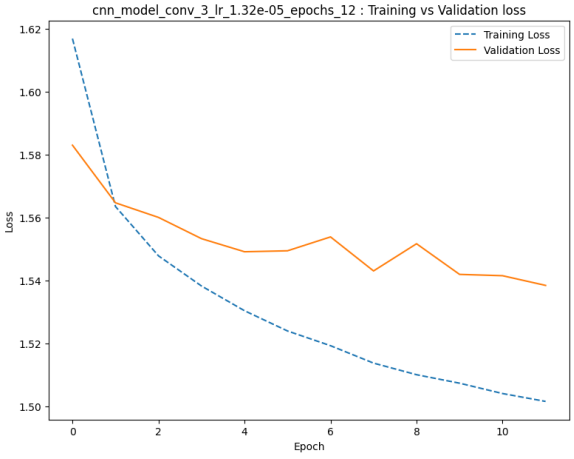
Some other good trials:

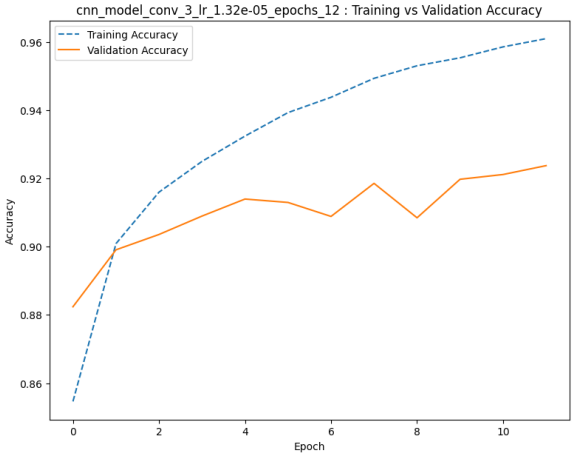


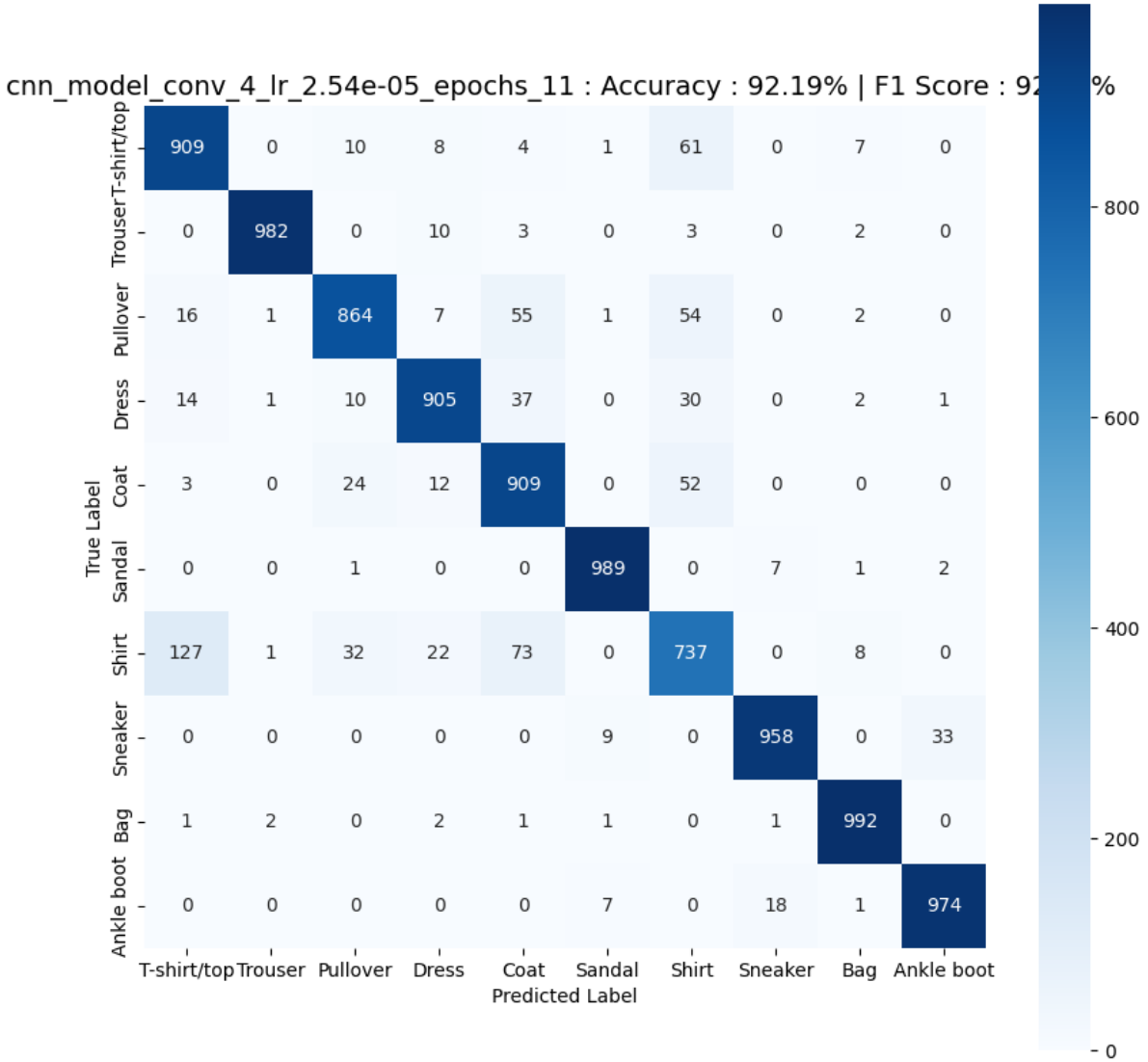


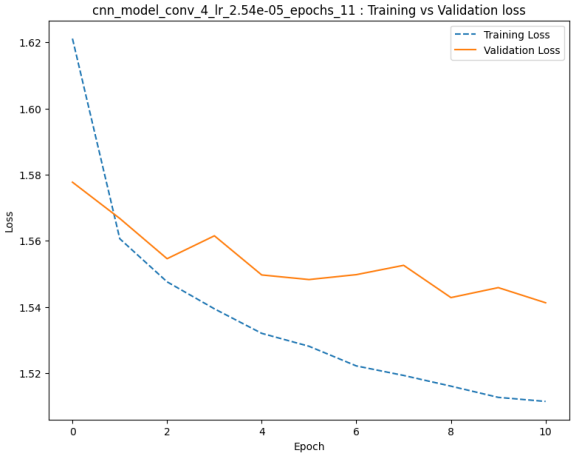


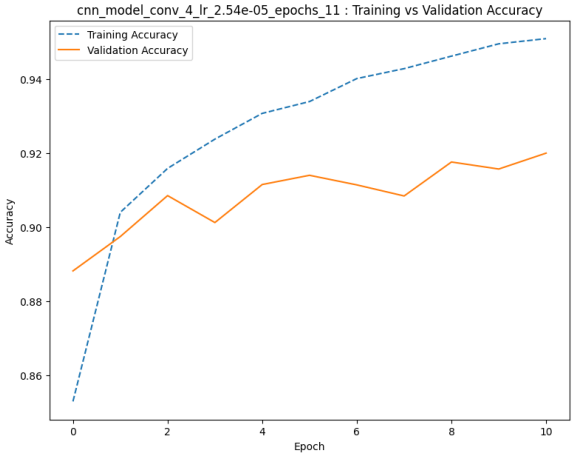

cnn_model_conv_3_lr_1.01e-05_epochs_12 : Accuracy : 91.77% | F1 Score : 91
True Label
T-shirt/top 902 0 18 7 3 1 62 0 7 0
Trouser 0 984 2 6 5 0 2 0 1 0
Pullover 21 1 886 5 52 1 34 0 0 0
Dress 27 6 14 899 29 0 24 0 1 0
Coat 3 0 36 19 918 0 23 0 0 1
Sandal 0 0 0 0 0 994 0 3 0 3
Shirt 118 2 54 18 103 0 701 0 4 0
Sneaker 0 0 0 0 0 18 0 927 0 55
Bag 3 2 2 2 1 3 2 1 984 0
Ankle boot 0 0 0 0 0 5 0 12 1 982
T-shirt/top Trouser Pullover Dress Coat Sandal Shirt Sneaker Bag Ankle boot
Predicted Label
800
600
400
200
0


cnn_model_conv_3_lr_1.32e-05_epochs_12 : Training vs Validation loss
Training Loss
Validation Loss
Loss
Epoch


cnn_model_conv_3_lr_1.32e-05_epochs_12 : Training vs Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy
Epoch


cnn_model_conv_4_lr_2.54e-05_epochs_11 : Accuracy : 92.19% | F1 Score : 92
True Label
Predicted Label
T-shirt/top
Trouser
Pullover
Dress
Coat
Sandal
Shirt
Sneaker
Bag
Ankle boot
909 0 10 8 4 1 61 0 7 0
0 982 0 10 3 0 3 0 2 0
16 1 864 7 55 1 54 0 2 0
14 1 10 905 37 0 30 0 2 1
3 0 24 12 909 0 52 0 0 0
0 0 1 0 0 989 0 7 1 2
127 1 32 22 73 0 737 0 8 0
0 0 0 0 0 9 0 958 0 33
1 2 0 2 1 1 0 1 992 0
0 0 0 0 0 7 0 18 1 974
800
600
400
200
0


cnn_model_conv_4_lr_2.54e-05_epochs_11 : Training vs Validation loss
Training Loss
Validation Loss
Loss
Epoch


cnn_model_conv_4_lr_2.54e-05_epochs_11 : Training vs Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy
Epoch

References:

1. Lectures given by Respected Professors
2. Sessions given by knowledgeable TAs
3. Papers:
    a. “Conditional Generative Adversarial Nets”, By Mehdi Mirza and Simon Osindero arXiv:1411.1784v1 [cs.LG] 6 Nov 2014

4. Blogs:
    a. https://towardsdatascience.com/cgan-conditional-generative-adversarial-network-how-to-gain-control-over-gan-outputs-b30620bd0cc8
    b. https://towardsdatascience.com/using-conditional-deep-convolutional-gans-to-generate-custom-faces-from-text-descriptions-e18cc7b8821
    c. https://learnopencv.com/conditional-gan-cgan-in-pytorch-and-tensorflow/
    d. And many other