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

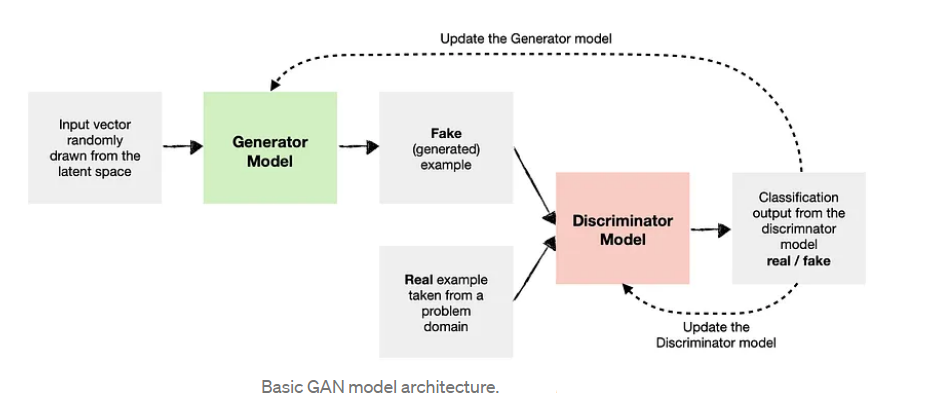
For Odd Roll Number Fashion MNIST

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.

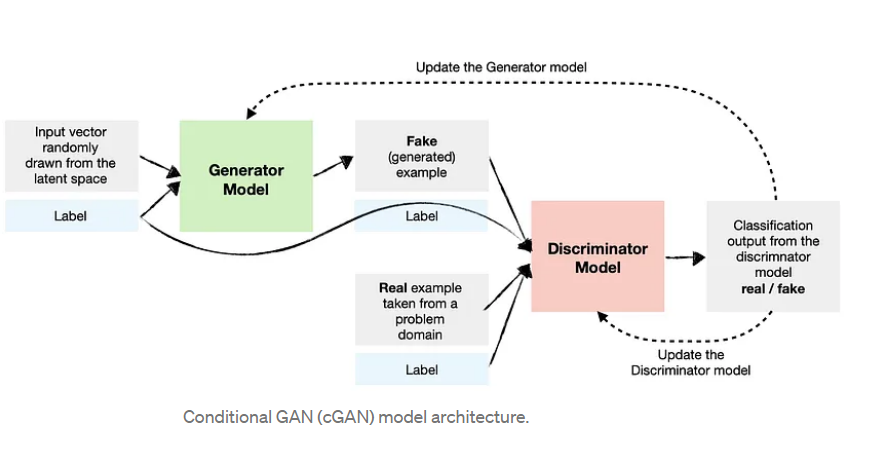


The generator network generates fake data, while the discriminator network tries to differentiate between real and fake data. The two networks are trained simultaneously, with the generator trying to generate increasingly realistic data, and the discriminator getting better at distinguishing between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

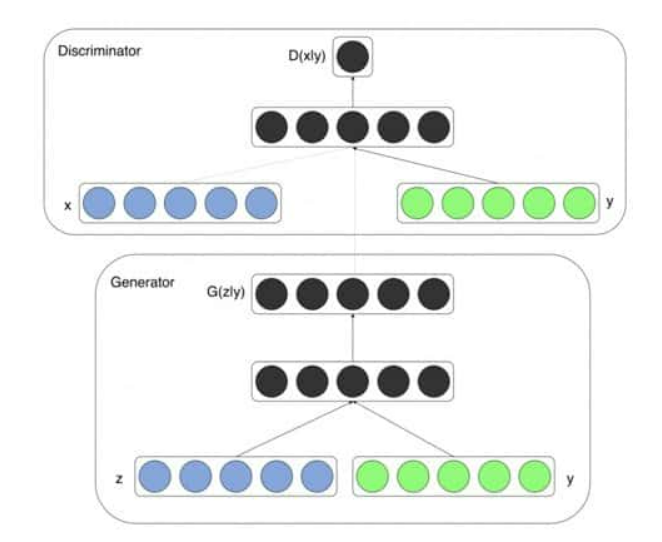
Despite their challenges, GANs have proven to be a powerful tool in the field of machine learning and have led to significant advancements in generating realistic and high-quality data.

## **Conditional Deep Convolutional Generative Adversarial Network (cDCGAN):**



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.

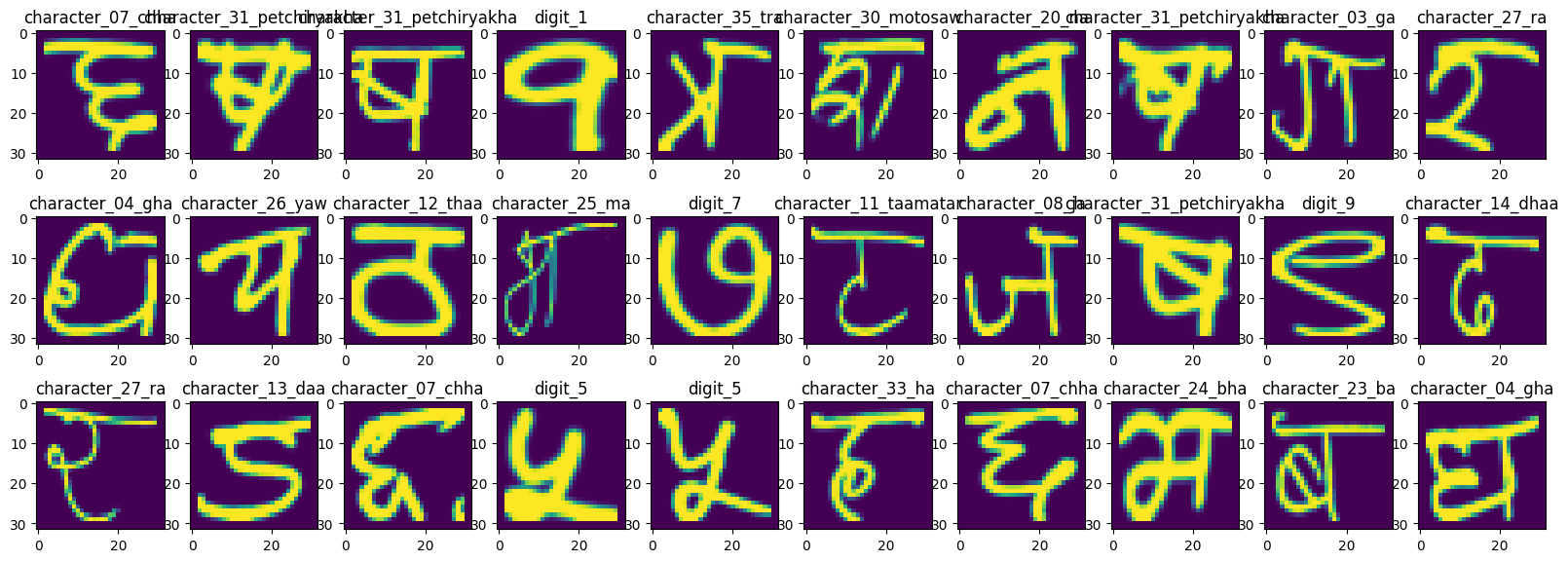


The discriminator network also takes in the same conditional vector and uses it to classify whether the input image is real or fake. The conditional vector serves as a form of supervision that helps guide the generator network to produce images that are more consistent with the specified attributes.

# 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

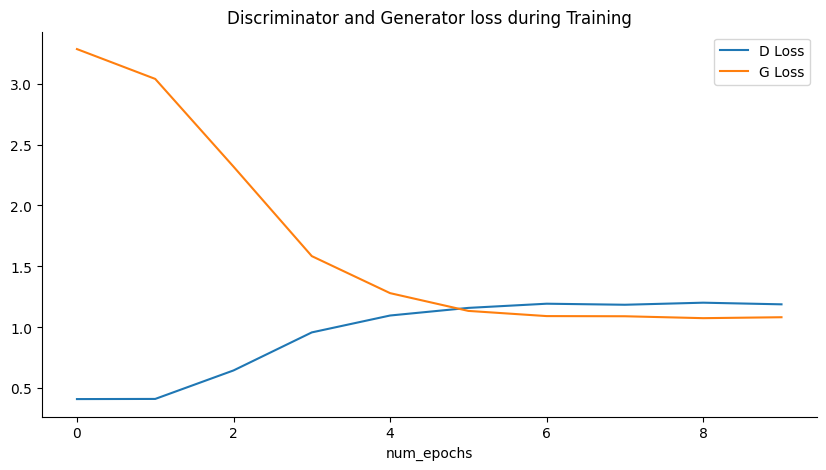
Epoch 7/10 Discriminator Loss 1.194 Generator Loss 1.092 D(x) 0.586 D(G(x)) 0.368

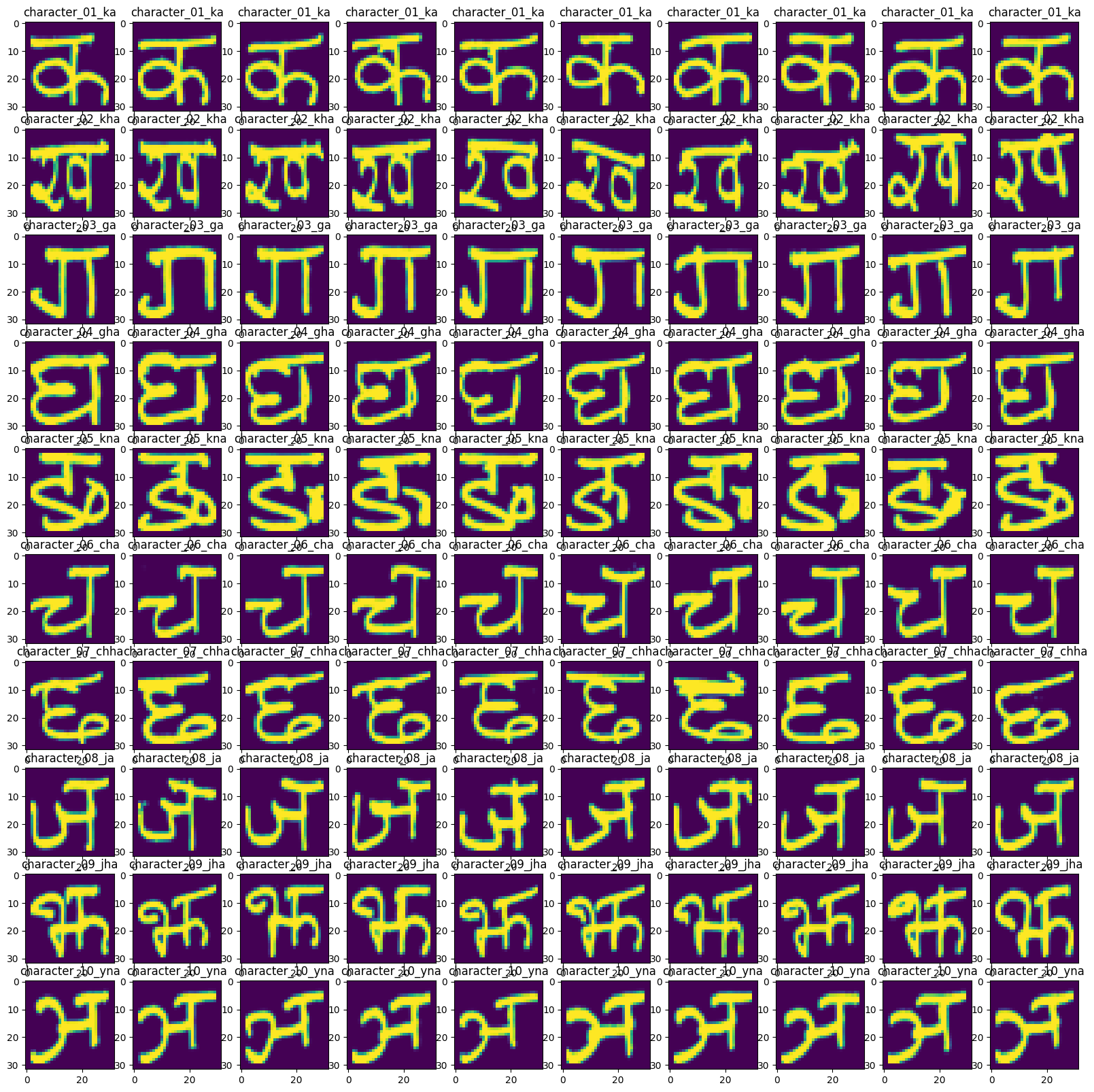
Epoch 8/10 Discriminator Loss 1.185 Generator Loss 1.091 D(x) 0.589 D(G(x)) 0.366

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:**

****

****

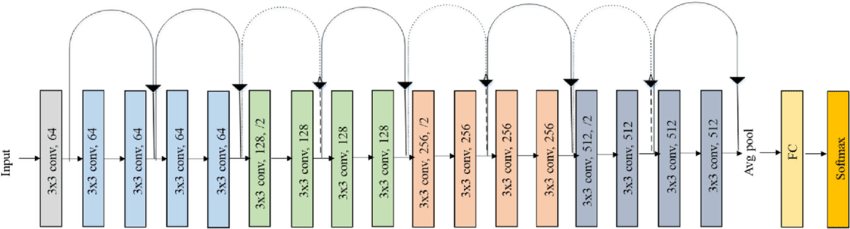
**B. Train a ResNet18 classifier on the given dataset and treat the generated samples as test dataset and report following [20 Marks]**

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,

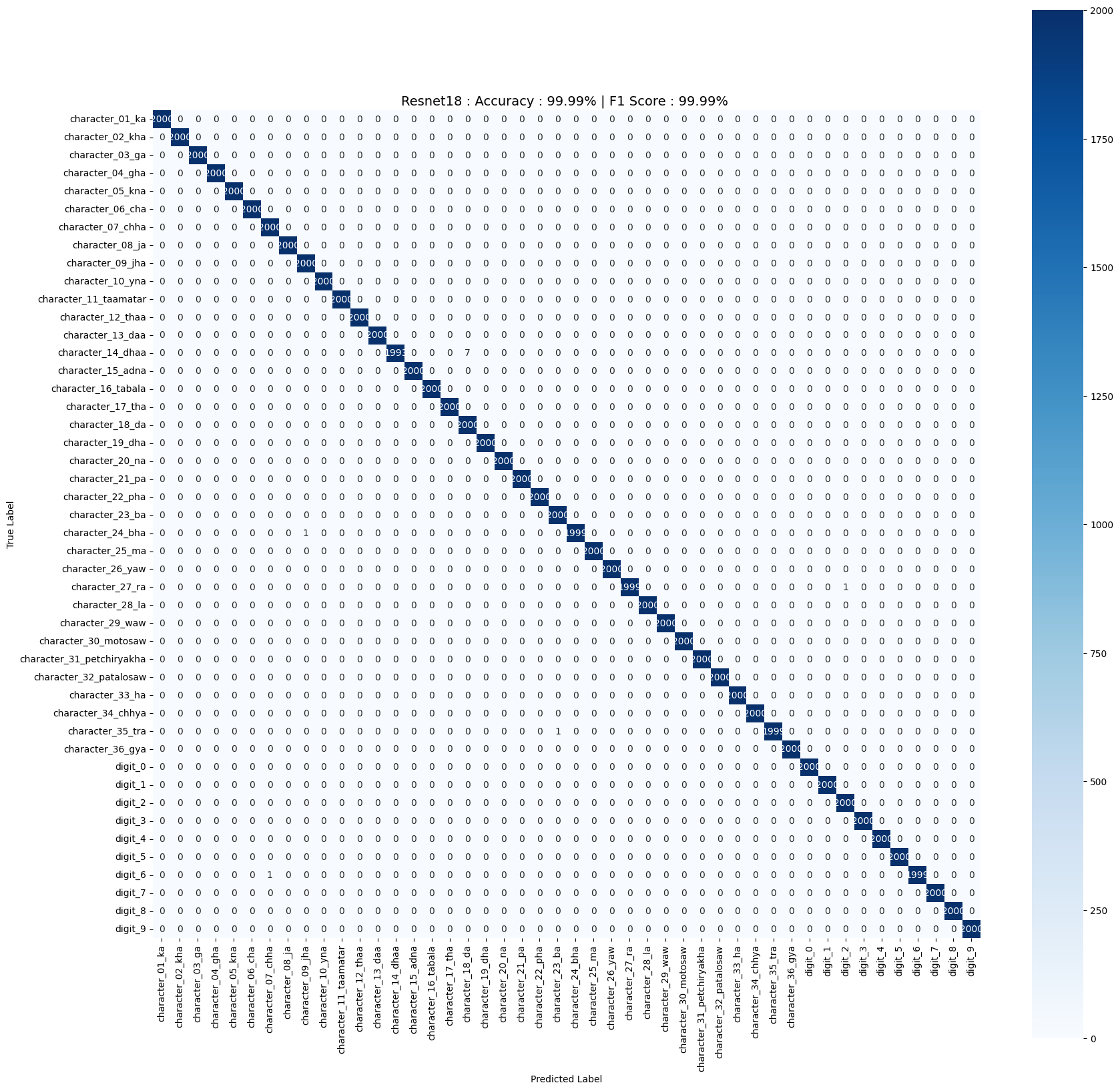
## **Training loss and accuracy curve:**

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

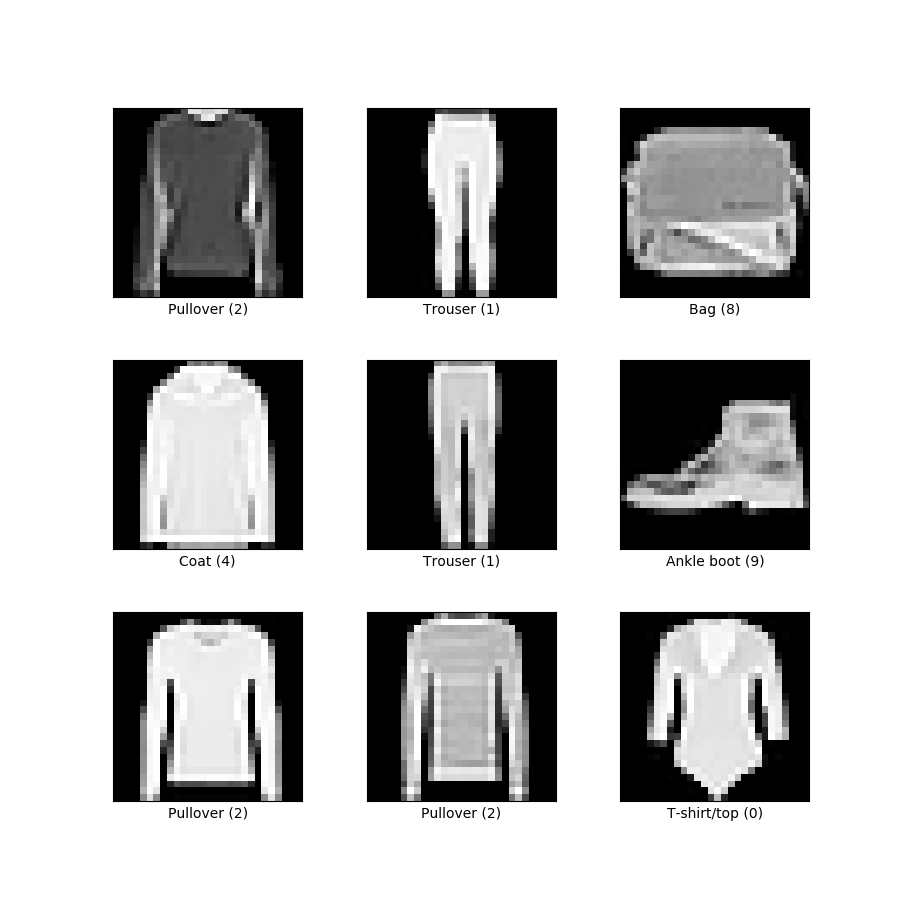
# Solution:

## 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:**

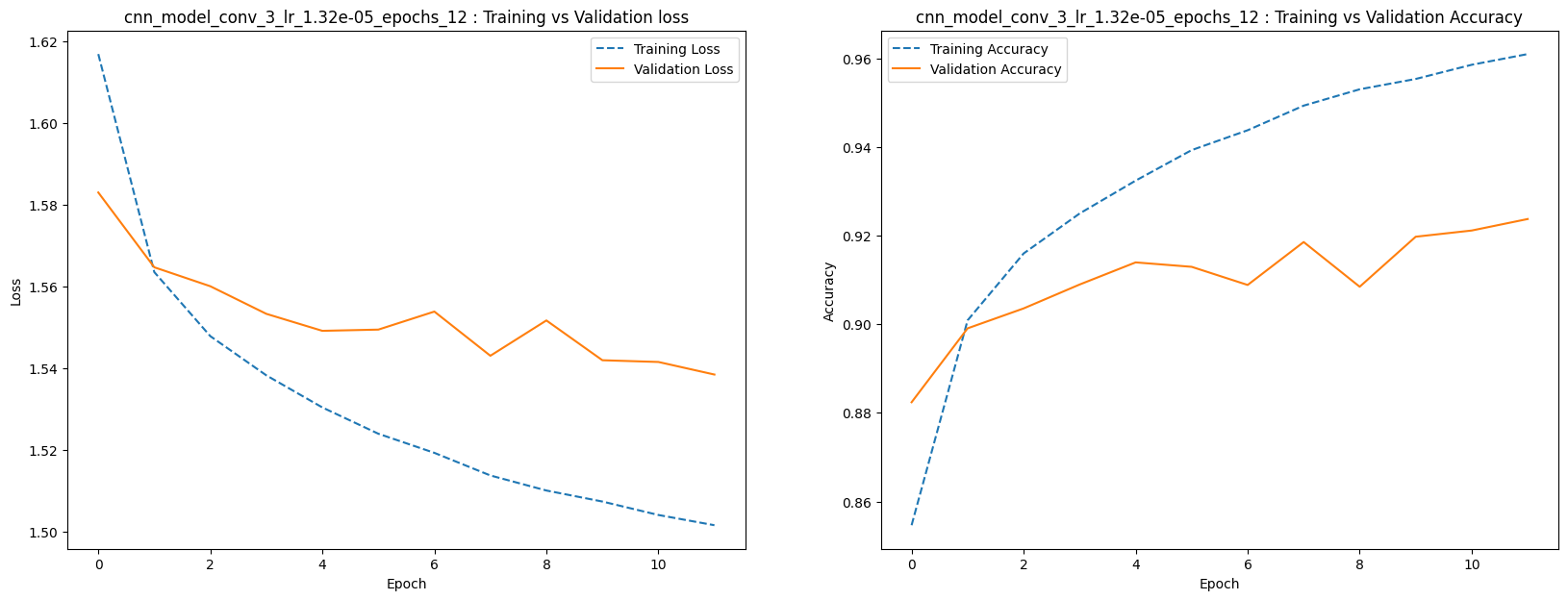
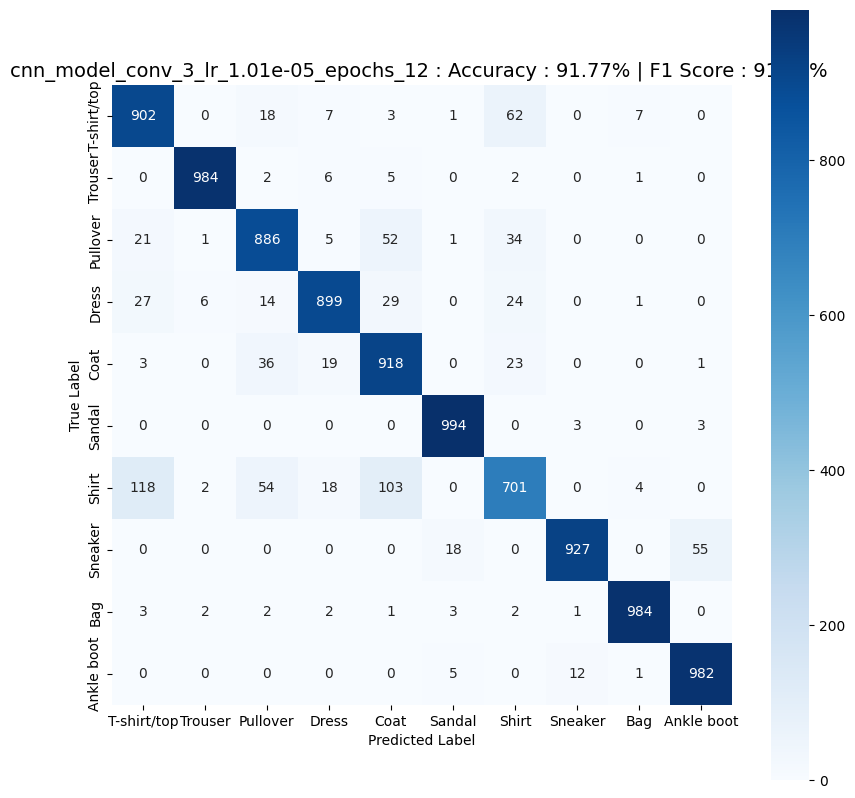
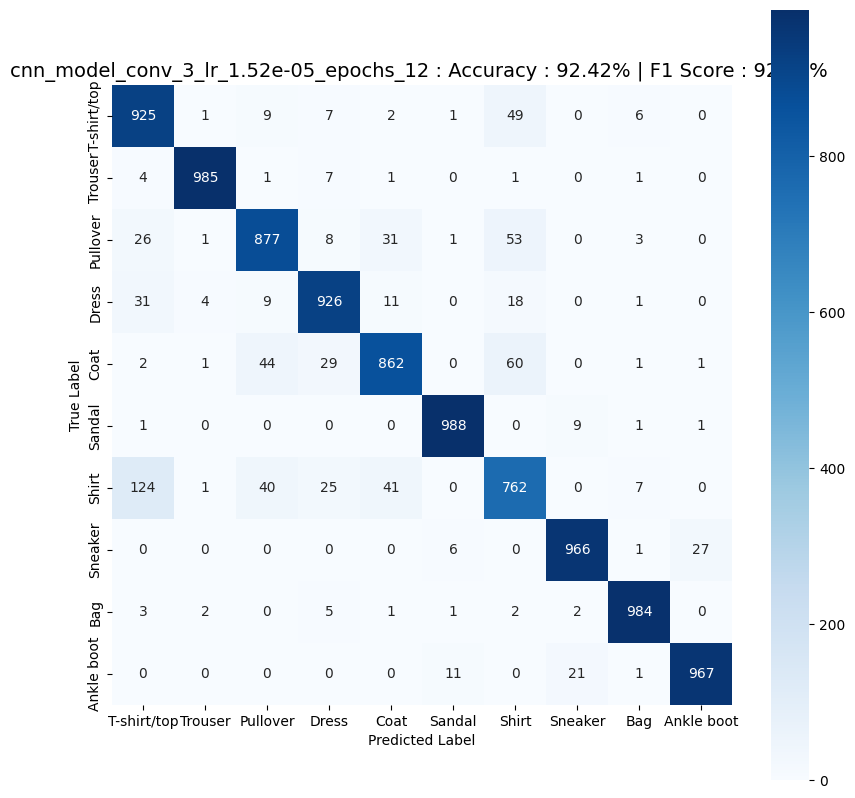
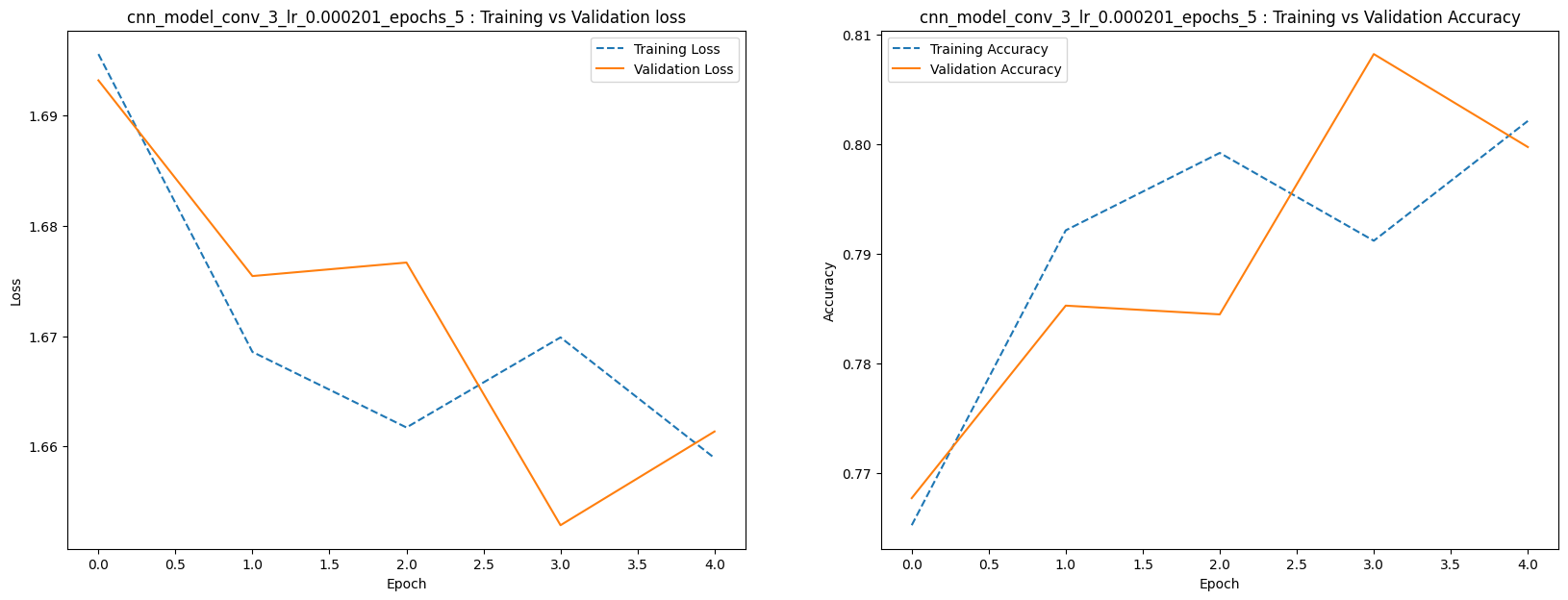
|  |  |
| --- | --- |
| Hyper Parameter | Value |
| No of Convolution Layer | 3 |
| Number of Epochs | 12 |
| Learning Rate | 1.3190817110540418e-05 |

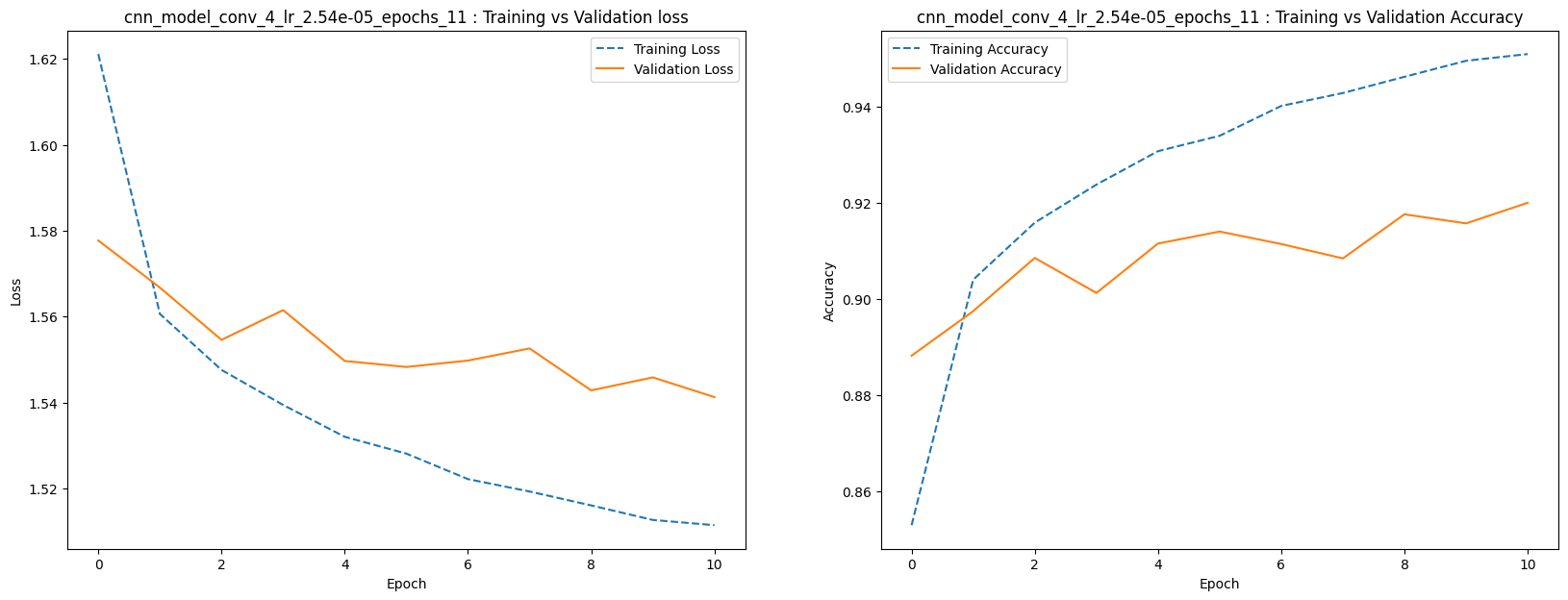
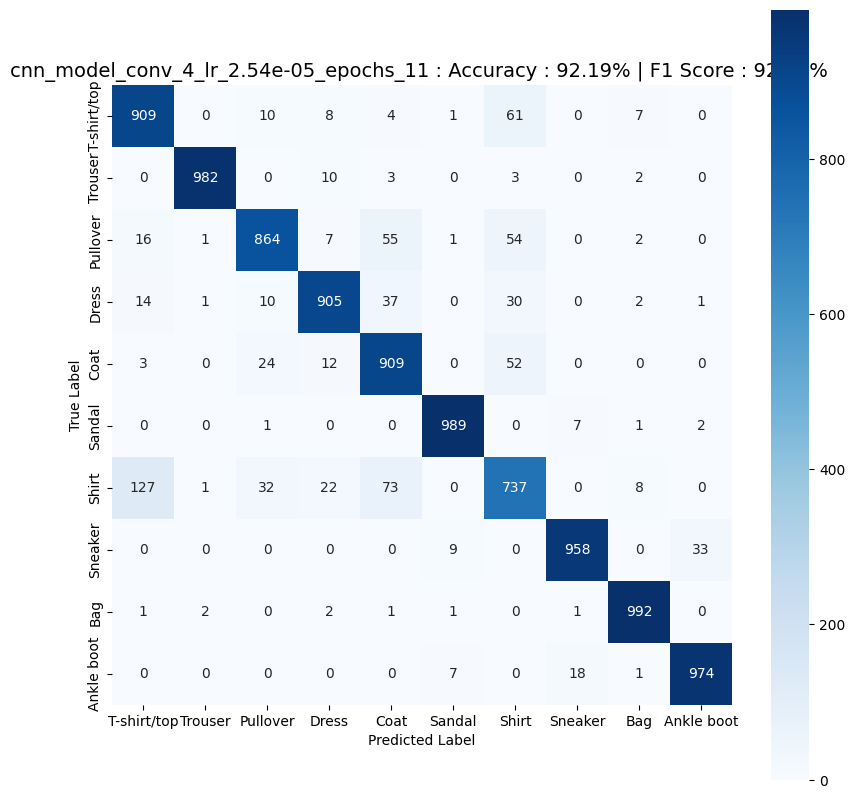
Complete run log included with files

# Details of Best Model found:

# 

# Some other good trials:





### References:

1. Lectures given by Respected Professors
2. Sessions given by knowledgeable TAs
3. Papers:
   1. “Conditional Generative Adversarial Nets”, By Mehdi Mirza and Simon Osindero

[arXiv:1411.1784v1 [cs.LG] 6 Nov 2014](https://arxiv.org/pdf/1411.1784.pdf)

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