DL-Ops Assignment-4 Report

- by D22CS051

Task: Implement DCGAN and use Optuna for hyperparameter search.

Note: All the files are uploaded in the zipped folder, so there is no need to train the model again.

Objectives:

Question 1.

- Train a Conditional Deep Convolutional Generative Adversarial Network (cDCGAN) on Dataset. (You may use this for Reference) [25 Marks]
  - Generate 50 Samples of each class from your trained generator. [5 Marks]
- Train a ResNet18 classifier on the given dataset and treat the generated samples as test dataset and report following [20 Marks]
  - F1 Score for each class
  - Confusion matrix

Procedure:

Checking Available GPU

- It checks if GPU is available or not by running !nvidia-smi.

Importing Requirements

- This part imports the required libraries to run the code, including:
  - torch, torchvision, torchmetrics, torchattacks: Libraries for deep learning and PyTorch-specific tools for metrics and adversarial attacks.
  - matplotlib, numpy, pandas: Libraries for data visualization and manipulation.
  - sklearn: A library for machine learning that includes tools for data preprocessing and evaluation.

Device Agnostic code

- The variable device is set to 'cuda' if a GPU is available, otherwise it is set to 'cpu'.

Getting Dataset

- The dataset is downloaded from Google Drive by using !gdown command.
- The downloaded file is read using pandas.read_csv.

- The dataset is then processed and transformed to be used in the deep learning model by:
    - Splitting the dataset into input data (X) and labels (y).
    - Reshaping the input data to images of size 32x32.
    - Converting the labels to integer values from 0 to number of classes.
    - Splitting the dataset into train and test sets.
    - Converting the train and test sets to PyTorch's Dataset format.
    - Converting the Dataset format to Dataloader format with batch size of 32.

Hyperparameters

- workers: Number of workers for dataloader
- batch_size: Batch size during training
- image_size: Spatial size of training images. All images will be resized to this size using a transformer.
- nc: Number of channels in the training images. For color images, this is 3
- nz: Size of z latent vector (i.e. size of generator input)
- ngf: Size of feature maps in generator
- ndf: Size of feature maps in discriminator
- num_epochs: Number of training epochs
- lr: Learning rate for optimizers
- beta1: Beta1 hyperparameter for Adam optimizers
- ngpu: Number of GPU

Weight Initialization

- A function called weights_init is defined to initialize the weights of the neural network.
- This function is used to initialize the weights of the convolution layers with a normal distribution with mean 0 and standard deviation 0.02, and the weights of the batch normalization layers with a normal distribution with mean 1 and standard deviation 0.02. The biases are initialized to 0.

Generator

- A Generator class is defined which takes ngpu as an argument.
- The generator takes a random noise vector of size nz as input and outputs an image of size image_size.
- The main module of the generator consists of several ConvTranspose2d layers, which are used to upscale the random noise vector into an image.
- nn.BatchNorm2d is used for batch normalization after each convolution layer except the last one.
- nn.ReLU is used as an activation function after each batch normalization layer except the last one.
- The output layer uses nn.Tanh as the activation function.

Discriminator

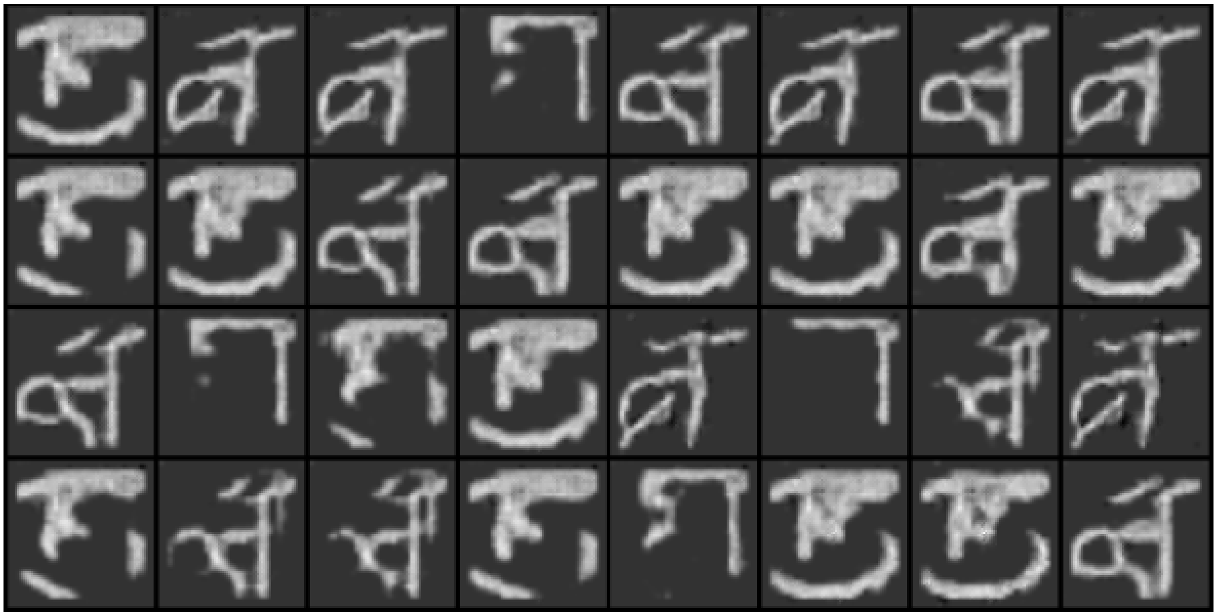
- A Discriminator class is defined which takes ngpu as an argument.
- The discriminator takes an image of size image_size as input and outputs a scalar value representing the probability of the input being real or fake.
- The main module of the discriminator consists of several Conv2d layers, which are used to downscale the input image into a scalar value.
- nn.BatchNorm2d is used for batch normalization after each convolution layer except the first one.
- nn.LeakyReLU is used as an activation function after each batch normalization layer except the last one.
- The output layer uses nn.Sigmoid as the activation function.

Models

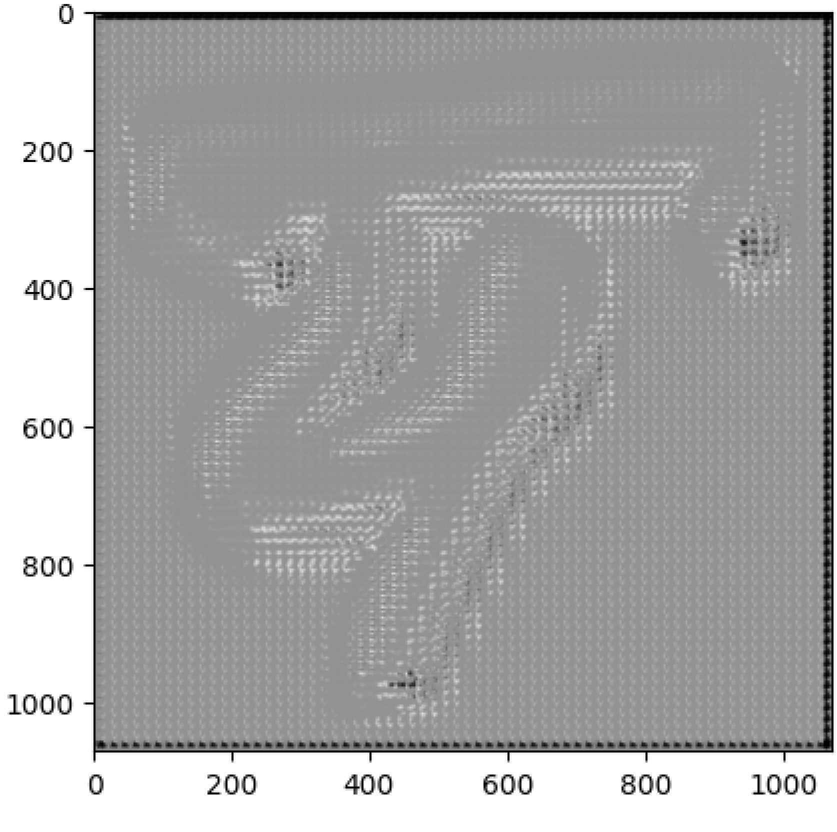
- The generator and discriminator models are created using the Generator and Discriminator classes, respectively.
- If the device is cuda and the number of GPUs is greater than 1, the models are parallelized using nn.DataParallel.
- Training a Generative Adversarial Network (GAN) to generate fake images that are similar to a given dataset. Here are the steps that this code performs:
- Define the loss function for the GAN, which is Binary Cross Entropy (BCE) loss.
- Create a batch of latent vectors that will be used to visualize the progression of the generator.
- Establish the convention for real and fake labels during training. The real label is set to 1, and the fake label is set to 0.
- Setup the Adam optimizer for both the generator (G) and discriminator (D) networks.
- Train the GAN for a number of epochs. For each epoch, loop over each batch in the dataloader.
- Update the discriminator network:
- Train with an all-real batch: calculate loss on all-real batch and update the discriminator network
- Train with an all-fake batch: generate a batch of latent vectors, generate fake images with the generator, calculate loss on the all-fake batch and update the discriminator network.
- Update the generator network: generate another batch of latent vectors, generate fake images with the generator, calculate loss and update the generator network.
- Output the training statistics.
- Save the trained models to files.
- Generate a grid of fake images and save them as a video.
- Load the trained models from files.
- Resizes the images in the fake tensor to 64 x 64 using the resize method in torchvision.transforms.functional.
- Displays the resized image using matplotlib.pyplot.imshow.
- Creates a dataset of generated images using a while loop that generates images from a generator network and adds them to the dataset until the dataset contains 2,300 images (2300 being the number of images needed for testing).

- Concatenates the generated dataset into a single dataset using torch.utils.data.ConcatDataset.
- Creates a data loader for the generated dataset using torch.utils.data.DataLoader.
- Defines a function called training_step that performs a training step for one epoch (one iteration of the training data).
- Defines a function called testing_step that performs a testing step for one epoch (one iteration of the testing data).
- Performs the training and testing steps in a loop for a specified number of epochs. During each epoch, the script prints the loss and accuracy for the training and testing steps.
- The train function which takes in the necessary parameters and performs the training loop for the specified number of epochs. It uses the training_step and testing_step functions to perform the training and testing steps respectively. The training and testing losses and accuracies are saved in a dictionary and returned at the end of each epoch. The best model is saved using the torch.save function.
- The plot_graph function which takes in the training and testing losses and accuracies and plots them using matplotlib.
- Initializing the ResNet18 model with the desired number of output classes and loading the weights.
- Initializing the loss and accuracy functions.
- Initializing the optimizer with a specified learning rate.
- Training the ResNet18 model for the desired number of epochs.

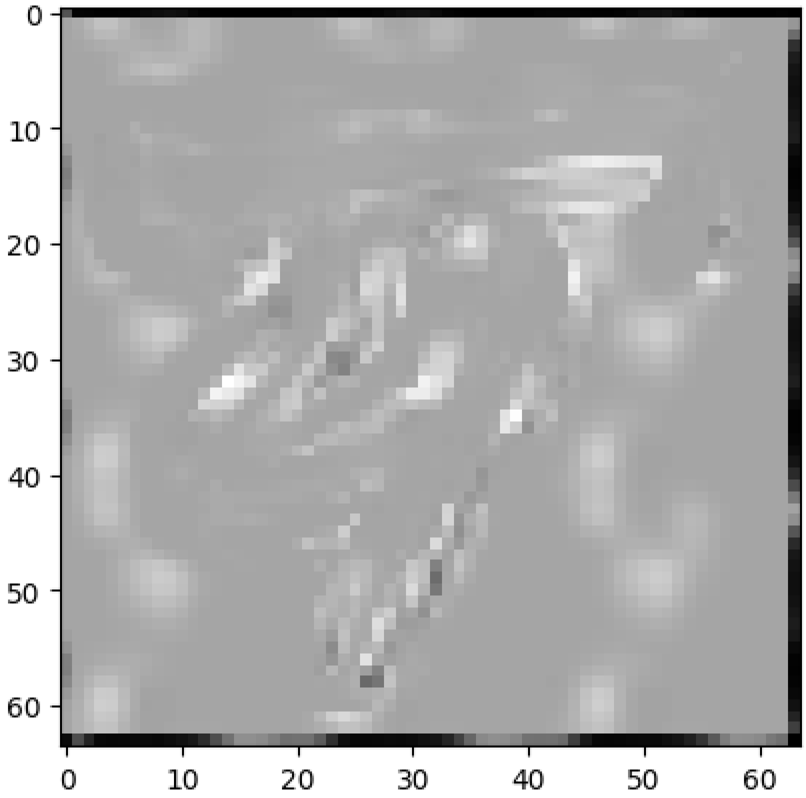
Results:
For 64 epochs of DCGAN



Generated images

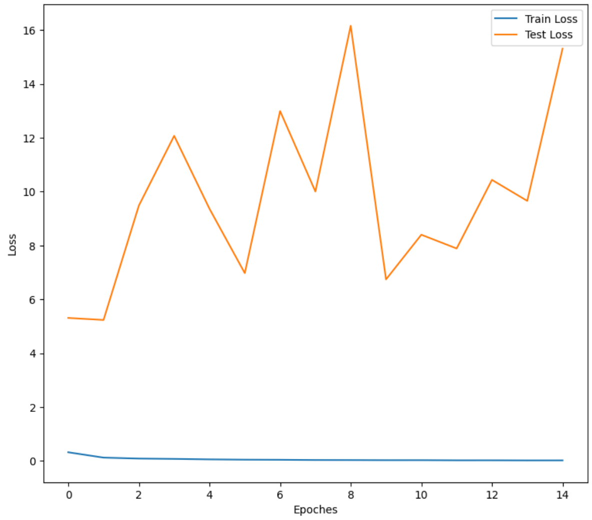


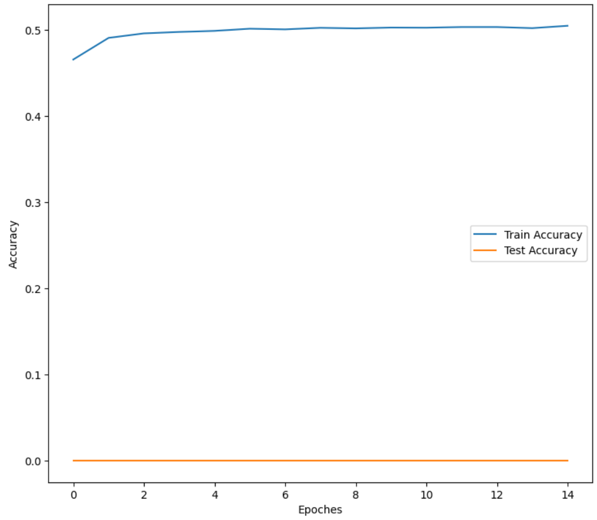
Solo-generated image in 1172x1172 size



Solo-generated image in 64x64 size

Train on Resnet18

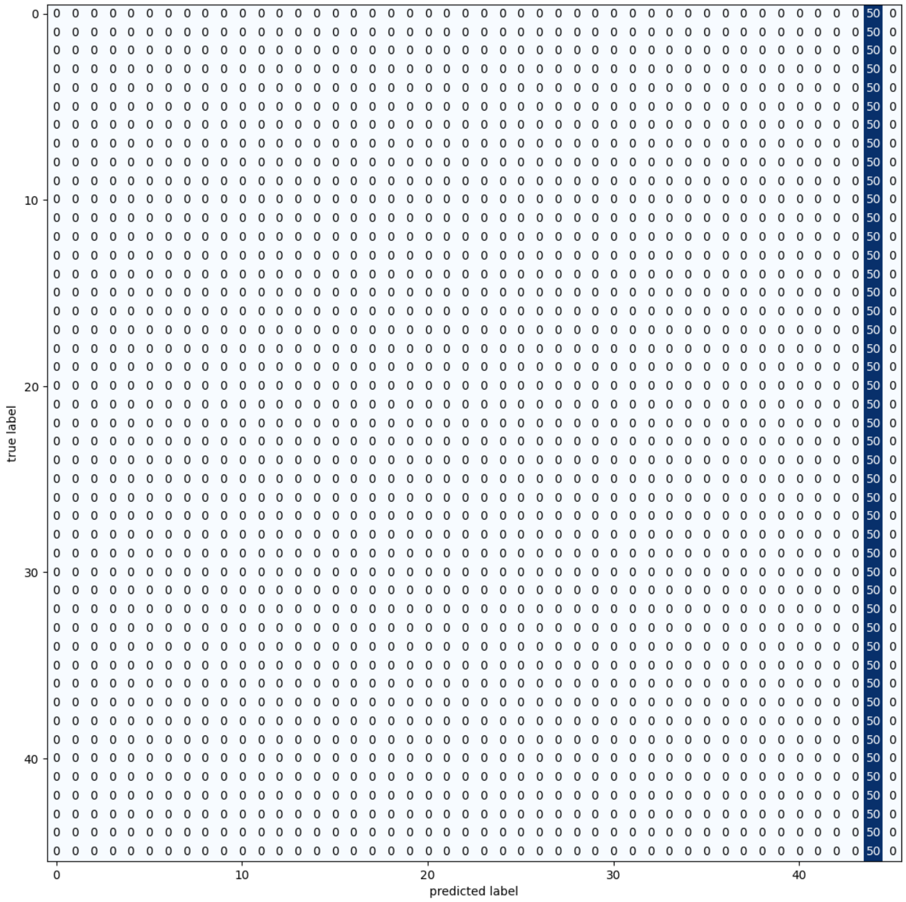




Execution time: 770.1250406219988 seconds.

F1 Score: 0.0009

Confusion matrix:



	0										10										20										30										40					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0

Observations:

- It gives poor performance on generated images.
- But I train it on 10 epochs and 64 epochs respectively on DCGAN. However 64 epochs DCGAN generates give good performance compared to 10 epochs DCGAN.
- Also model which train on more datasets give good performance compared to less train datasets.
- Therefore forbetter Generated image we need to train DCGAN for mode number of epochs as possible.

Objectives:

Question 2.

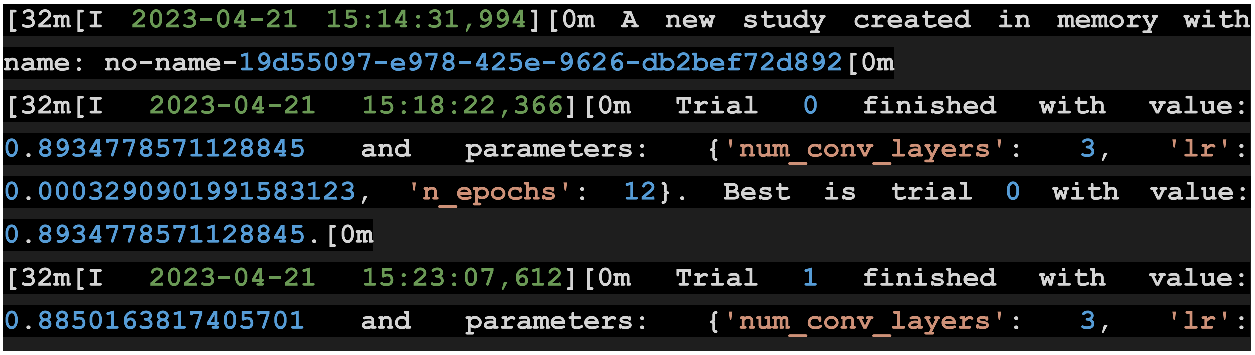
- Train a CNN based classification model and perform Optimized Hyperparameter Tuning using Optuna Library on the below-mentioned dataset. Perform 100 trials.
- Hyperparameters should be
  - 1) No of Convolution Layers 3 to 6
  - 2) Number of Epochs 10 to 50
  - 3) Learning rate 0.0001 to 0.1
- Report the observations and the best trial. Report how many trials were pruned
- For Even Roll Number MNIST
- For Odd Roll Number Fashion MNIST

Procedure:

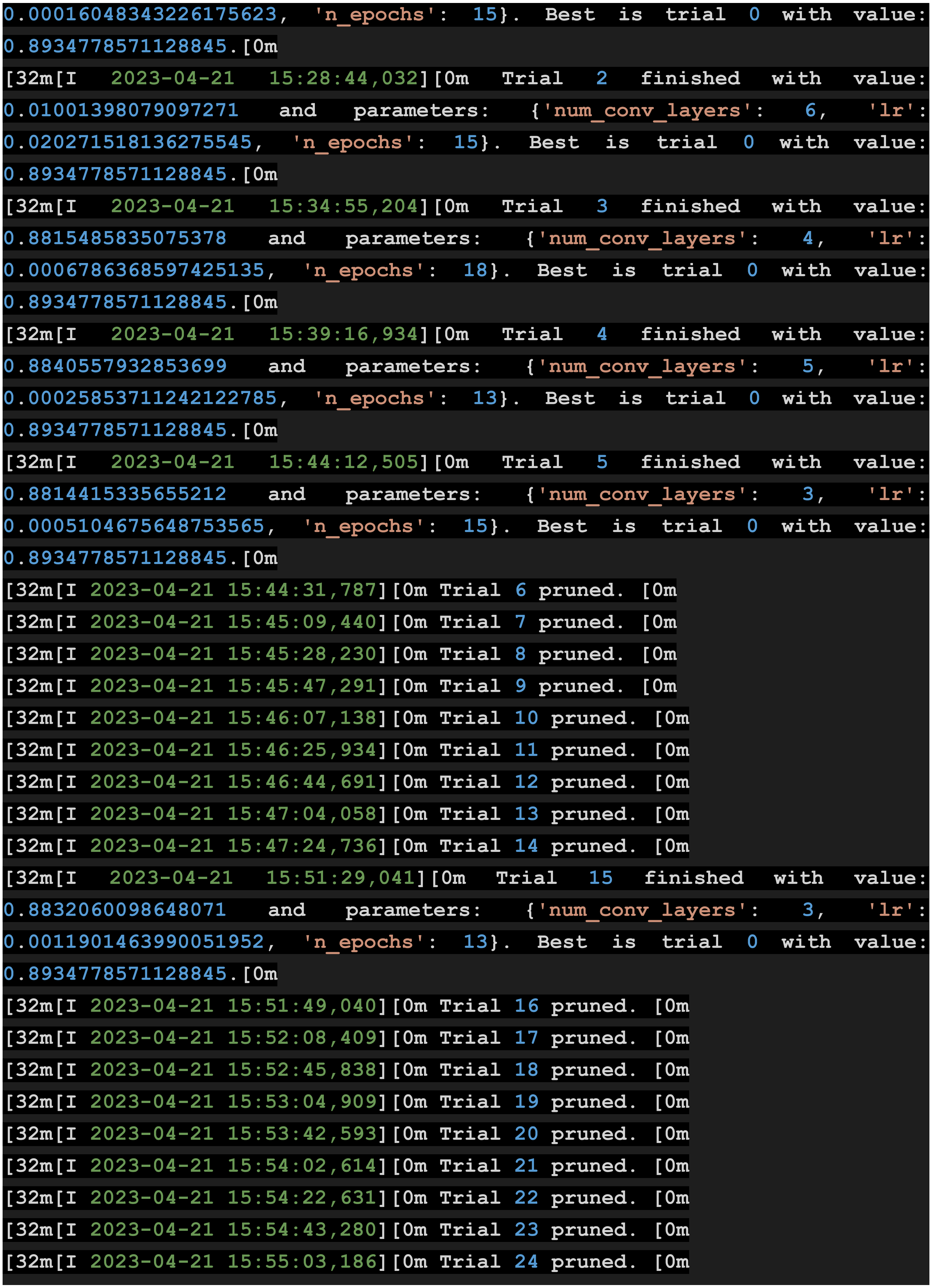
- Installs the optuna package.
- Imports the required libraries including PyTorch, Matplotlib, NumPy, TorchMetrics, TorchInfo, TorchAttacks, and Optuna.
- Checks the availability of GPU and sets the device accordingly.
- Downloads the FashionMNIST dataset and converts it into PyTorch DataLoader for training and testing.
- Defines the hyperparameters for the DataLoader, such as batch size and number of workers.
- Plots a random subset of images from the training dataset using Matplotlib.
- Prints the number of batches in the train and test dataloaders.
- Defines the class_names variable to be equal to the classes present in the train_dataset.
- Defines the index_cls variable to be equal to the mapping between class names and class indices.
- Defines a plot_loss_curves function that plots the training and testing loss and accuracy curves of a model.
- Defines two functions training_step and testing_step for performing one epoch of training and testing, respectively.
- The training_step function performs the training step for one epoch. It takes as input a PyTorch model object, a training data loader, a loss function object, an accuracy function from TorchMetrics, an optimizer function object, a device (either CPU or GPU), and an optional profiler from PyTorch. The function puts the model in training mode and loops over the batches in the data loader. For each batch, it performs the forward pass, calculates the loss, sets the optimizer's gradients to zero, backpropagates the loss, and updates the optimizer's parameters. The function returns the average training loss and accuracy for the epoch.

- The testing_step function performs the testing step for one epoch. It takes as input a PyTorch model object, a testing data loader, a loss function object, an accuracy function from TorchMetrics, and a device (either CPU or GPU). The function puts the model in evaluation mode and loops over the batches in the data loader. For each batch, it performs the forward pass and calculates the loss and accuracy. The function returns the average testing loss and accuracy for the epoch.
- Define the CNN model that will be used for training. The model is defined as a class called CustomCNN, which is a subclass of the nn.Module class in PyTorch. The constructor of the class initializes the parameters of the model, including the number of convolutional layers, the number of filters in each layer, the number of neurons in the fully connected layers, and the dropout rates. The forward method of the class implements the forward pass of the CNN.
- Define a function called objective that is used as the objective function for the Optuna study. The function takes a trial object as an argument, which is used to sample hyperparameters for the CNN model. The function returns the accuracy of the model on the test set.
- Create an instance of the Optuna.Study class, which is used to perform hyperparameter optimization.
- Use the study.optimize method to run the hyperparameter optimization. The objective function is used as the objective function, and the n_trials parameter specifies the number of trials to run.
- Print out the results of the study, including the number of finished trials, the number of pruned trials, and the number of complete trials. Also, print out the best trial.
- Note that the code also defines some helper functions for training and testing the CNN model. These functions are used within the objective function to train and test the model.

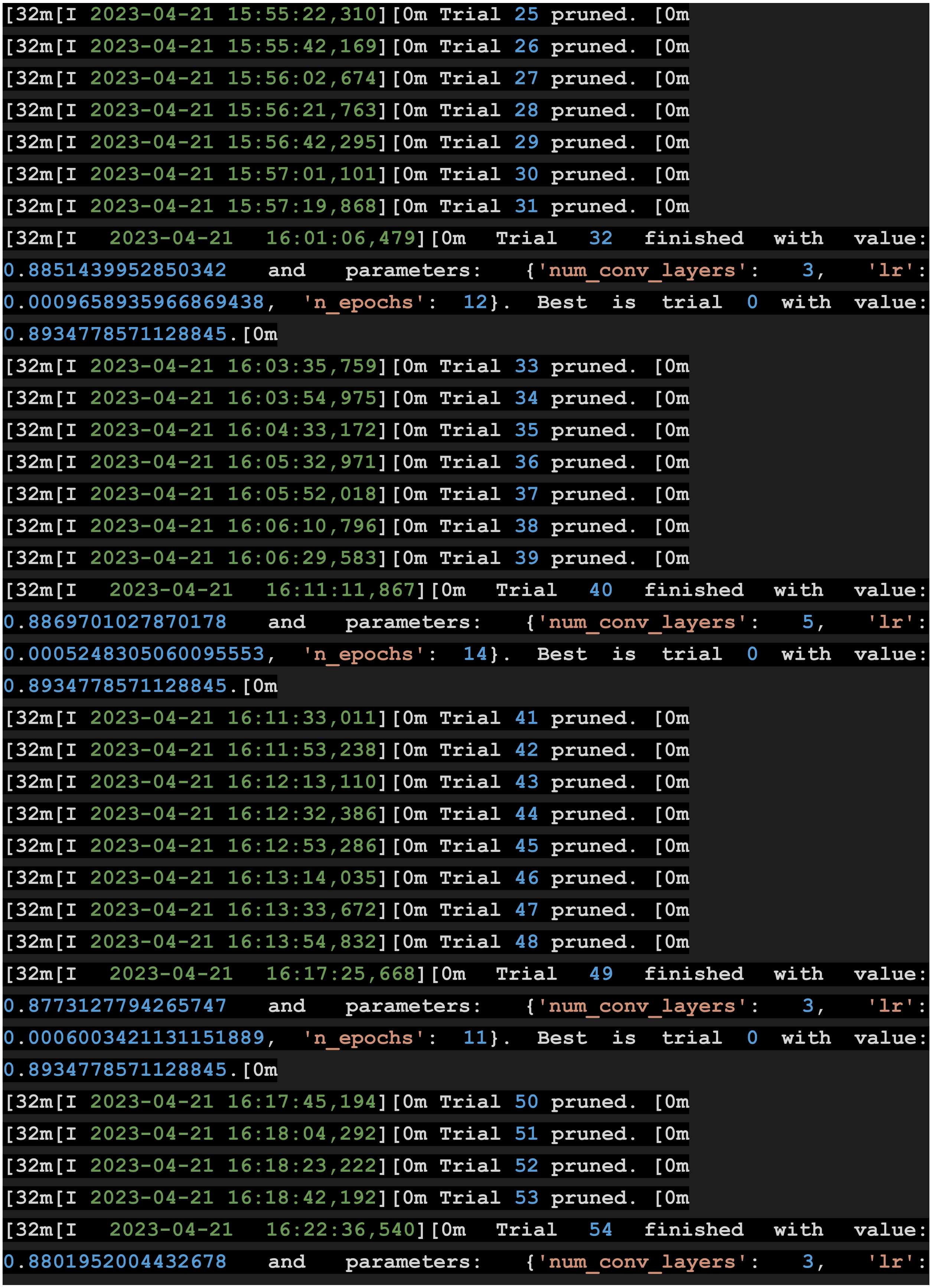
Results:



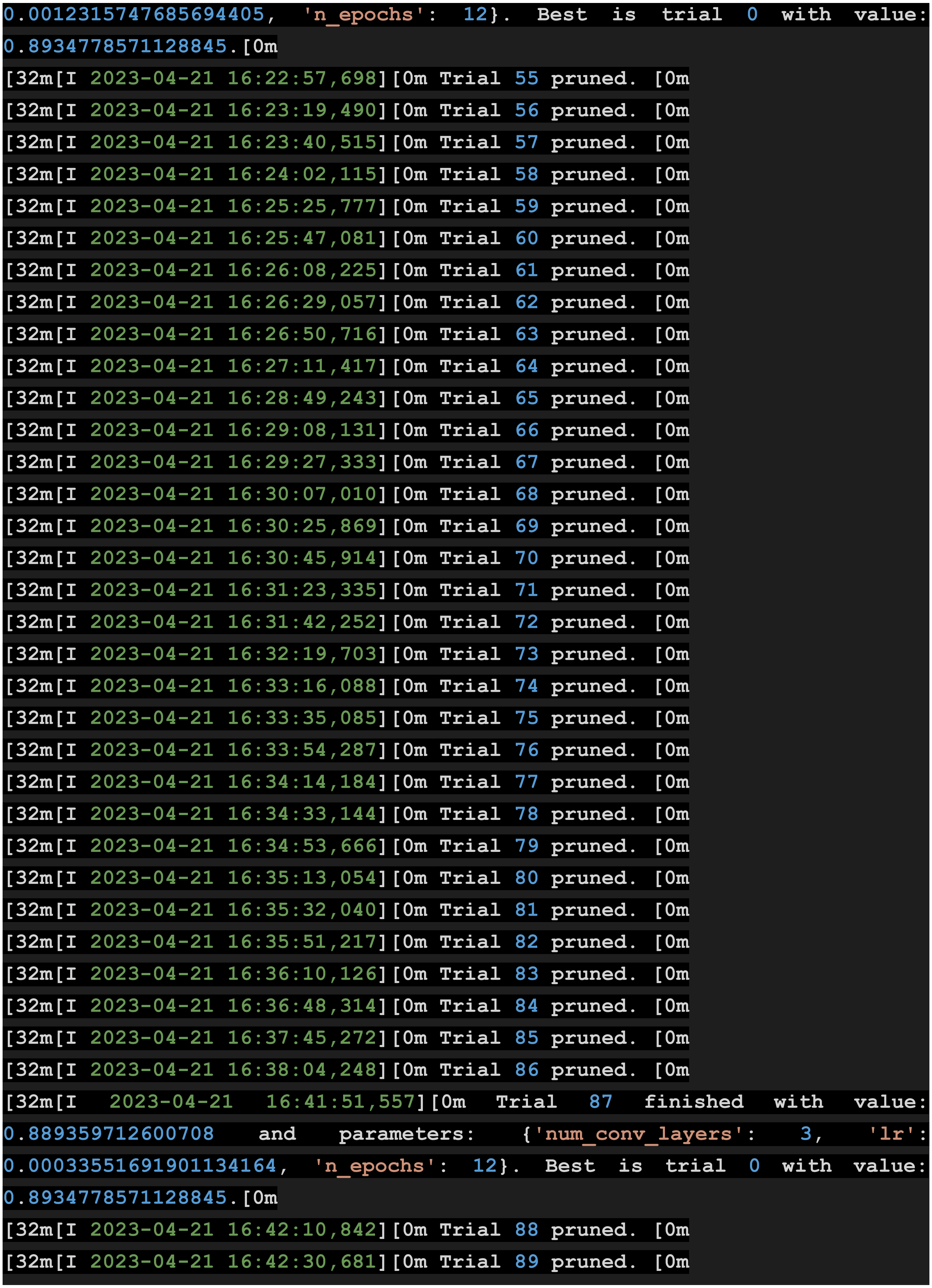
```
[32m[I 2023-04-21 15:14:31,994][0m A new study created in memory with
name: no-name-19d55097-e978-425e-9626-db2bef72d892[0m
[32m[I 2023-04-21 15:18:22,366][0m Trial 0 finished with value:
0.8934778571128845 and parameters: {'num_conv_layers': 3, 'lr':
0.0003290901991583123, 'n_epochs': 12}. Best is trial 0 with value:
0.8934778571128845.[0m
[32m[I 2023-04-21 15:23:07,612][0m Trial 1 finished with value:
0.8850163817405701 and parameters: {'num_conv_layers': 3, 'lr':
```



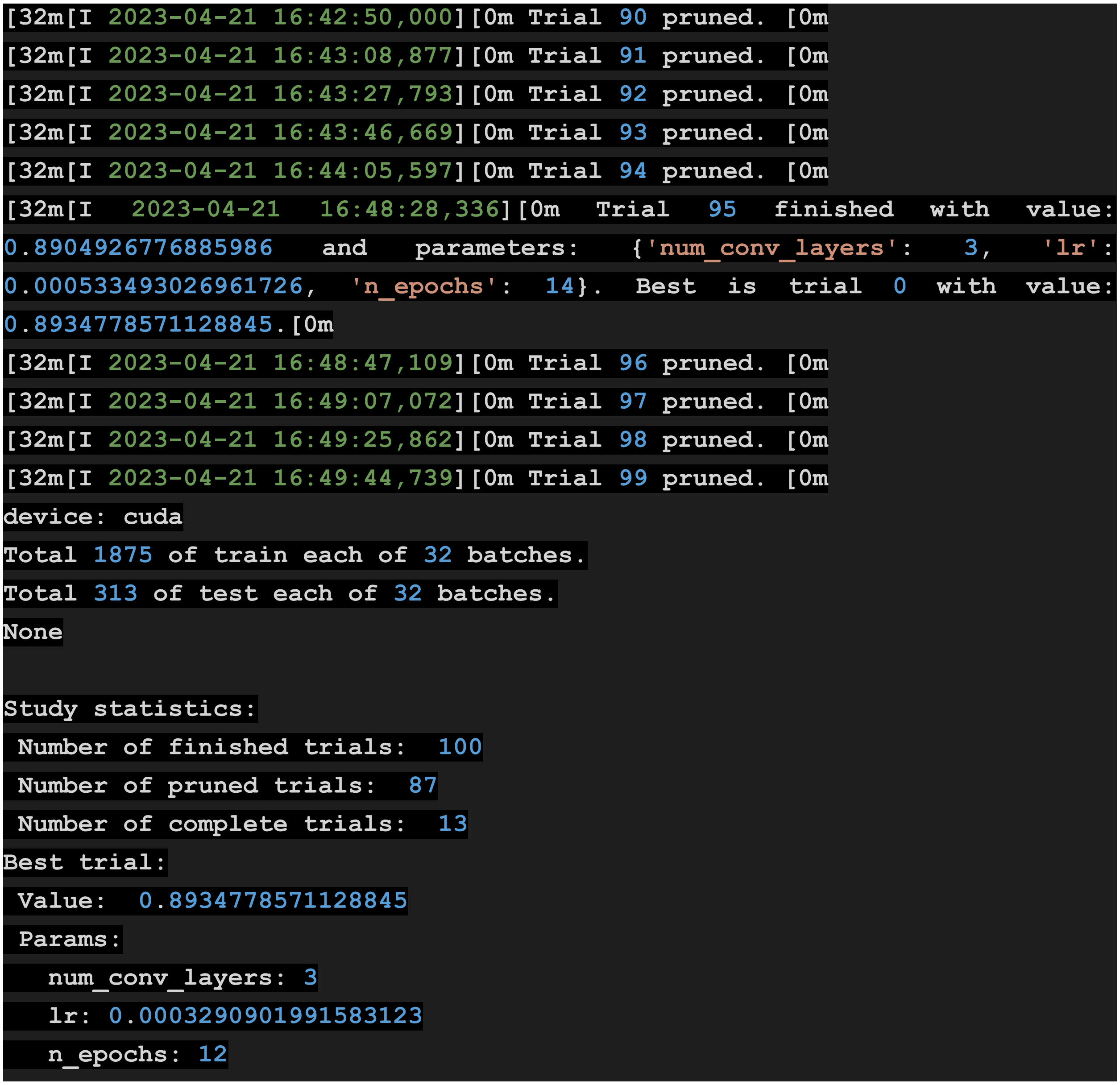
```
0.00016048343226175623, 'n_epochs': 15}. Best is trial 0 with value:
0.8934778571128845.[0m
[32m[I 2023-04-21 15:28:44,032][0m Trial 2 finished with value:
0.01001398079097271 and parameters: {'num_conv_layers': 6, 'lr':
0.020271518136275545, 'n_epochs': 15}. Best is trial 0 with value:
0.8934778571128845.[0m
[32m[I 2023-04-21 15:34:55,204][0m Trial 3 finished with value:
0.8815485835075378 and parameters: {'num_conv_layers': 4, 'lr':
0.0006786368597425135, 'n_epochs': 18}. Best is trial 0 with value:
0.8934778571128845.[0m
[32m[I 2023-04-21 15:39:16,934][0m Trial 4 finished with value:
0.8840557932853699 and parameters: {'num_conv_layers': 5, 'lr':
0.00025853711242122785, 'n_epochs': 13}. Best is trial 0 with value:
0.8934778571128845.[0m
[32m[I 2023-04-21 15:44:12,505][0m Trial 5 finished with value:
0.8814415335655212 and parameters: {'num_conv_layers': 3, 'lr':
0.0005104675648753565, 'n_epochs': 15}. Best is trial 0 with value:
0.8934778571128845.[0m
[32m[I 2023-04-21 15:44:31,787][0m Trial 6 pruned. [0m
[32m[I 2023-04-21 15:45:09,440][0m Trial 7 pruned. [0m
[32m[I 2023-04-21 15:45:28,230][0m Trial 8 pruned. [0m
[32m[I 2023-04-21 15:45:47,291][0m Trial 9 pruned. [0m
[32m[I 2023-04-21 15:46:07,138][0m Trial 10 pruned. [0m
[32m[I 2023-04-21 15:46:25,934][0m Trial 11 pruned. [0m
[32m[I 2023-04-21 15:46:44,691][0m Trial 12 pruned. [0m
[32m[I 2023-04-21 15:47:04,058][0m Trial 13 pruned. [0m
[32m[I 2023-04-21 15:47:24,736][0m Trial 14 pruned. [0m
[32m[I 2023-04-21 15:51:29,041][0m Trial 15 finished with value:
0.8832060098648071 and parameters: {'num_conv_layers': 3, 'lr':
0.0011901463990051952, 'n_epochs': 13}. Best is trial 0 with value:
0.8934778571128845.[0m
[32m[I 2023-04-21 15:51:49,040][0m Trial 16 pruned. [0m
[32m[I 2023-04-21 15:52:08,409][0m Trial 17 pruned. [0m
[32m[I 2023-04-21 15:52:45,838][0m Trial 18 pruned. [0m
[32m[I 2023-04-21 15:53:04,909][0m Trial 19 pruned. [0m
[32m[I 2023-04-21 15:53:42,593][0m Trial 20 pruned. [0m
[32m[I 2023-04-21 15:54:02,614][0m Trial 21 pruned. [0m
[32m[I 2023-04-21 15:54:22,631][0m Trial 22 pruned. [0m
[32m[I 2023-04-21 15:54:43,280][0m Trial 23 pruned. [0m
[32m[I 2023-04-21 15:55:03,186][0m Trial 24 pruned. [0m
```



```
[32m[I 2023-04-21 15:55:22,310][0m Trial 25 pruned. [0m
[32m[I 2023-04-21 15:55:42,169][0m Trial 26 pruned. [0m
[32m[I 2023-04-21 15:56:02,674][0m Trial 27 pruned. [0m
[32m[I 2023-04-21 15:56:21,763][0m Trial 28 pruned. [0m
[32m[I 2023-04-21 15:56:42,295][0m Trial 29 pruned. [0m
[32m[I 2023-04-21 15:57:01,101][0m Trial 30 pruned. [0m
[32m[I 2023-04-21 15:57:19,868][0m Trial 31 pruned. [0m
[32m[I 2023-04-21 16:01:06,479][0m Trial 32 finished with value:
0.8851439952850342 and parameters: {'num_conv_layers': 3, 'lr':
0.0009658935966869438, 'n_epochs': 12}. Best is trial 0 with value:
0.8934778571128845.[0m
[32m[I 2023-04-21 16:03:35,759][0m Trial 33 pruned. [0m
[32m[I 2023-04-21 16:03:54,975][0m Trial 34 pruned. [0m
[32m[I 2023-04-21 16:04:33,172][0m Trial 35 pruned. [0m
[32m[I 2023-04-21 16:05:32,971][0m Trial 36 pruned. [0m
[32m[I 2023-04-21 16:05:52,018][0m Trial 37 pruned. [0m
[32m[I 2023-04-21 16:06:10,796][0m Trial 38 pruned. [0m
[32m[I 2023-04-21 16:06:29,583][0m Trial 39 pruned. [0m
[32m[I 2023-04-21 16:11:11,867][0m Trial 40 finished with value:
0.8869701027870178 and parameters: {'num_conv_layers': 5, 'lr':
0.0005248305060095553, 'n_epochs': 14}. Best is trial 0 with value:
0.8934778571128845.[0m
[32m[I 2023-04-21 16:11:33,011][0m Trial 41 pruned. [0m
[32m[I 2023-04-21 16:11:53,238][0m Trial 42 pruned. [0m
[32m[I 2023-04-21 16:12:13,110][0m Trial 43 pruned. [0m
[32m[I 2023-04-21 16:12:32,386][0m Trial 44 pruned. [0m
[32m[I 2023-04-21 16:12:53,286][0m Trial 45 pruned. [0m
[32m[I 2023-04-21 16:13:14,035][0m Trial 46 pruned. [0m
[32m[I 2023-04-21 16:13:33,672][0m Trial 47 pruned. [0m
[32m[I 2023-04-21 16:13:54,832][0m Trial 48 pruned. [0m
[32m[I 2023-04-21 16:17:25,668][0m Trial 49 finished with value:
0.8773127794265747 and parameters: {'num_conv_layers': 3, 'lr':
0.0006003421131151889, 'n_epochs': 11}. Best is trial 0 with value:
0.8934778571128845.[0m
[32m[I 2023-04-21 16:17:45,194][0m Trial 50 pruned. [0m
[32m[I 2023-04-21 16:18:04,292][0m Trial 51 pruned. [0m
[32m[I 2023-04-21 16:18:23,222][0m Trial 52 pruned. [0m
[32m[I 2023-04-21 16:18:42,192][0m Trial 53 pruned. [0m
[32m[I 2023-04-21 16:22:36,540][0m Trial 54 finished with value:
0.8801952004432678 and parameters: {'num_conv_layers': 3, 'lr':
```



```
0.0012315747685694405, 'n_epochs': 12}. Best is trial 0 with value:
0.8934778571128845.[0m
[32m[I 2023-04-21 16:22:57,698][0m Trial 55 pruned. [0m
[32m[I 2023-04-21 16:23:19,490][0m Trial 56 pruned. [0m
[32m[I 2023-04-21 16:23:40,515][0m Trial 57 pruned. [0m
[32m[I 2023-04-21 16:24:02,115][0m Trial 58 pruned. [0m
[32m[I 2023-04-21 16:25:25,777][0m Trial 59 pruned. [0m
[32m[I 2023-04-21 16:25:47,081][0m Trial 60 pruned. [0m
[32m[I 2023-04-21 16:26:08,225][0m Trial 61 pruned. [0m
[32m[I 2023-04-21 16:26:29,057][0m Trial 62 pruned. [0m
[32m[I 2023-04-21 16:26:50,716][0m Trial 63 pruned. [0m
[32m[I 2023-04-21 16:27:11,417][0m Trial 64 pruned. [0m
[32m[I 2023-04-21 16:28:49,243][0m Trial 65 pruned. [0m
[32m[I 2023-04-21 16:29:08,131][0m Trial 66 pruned. [0m
[32m[I 2023-04-21 16:29:27,333][0m Trial 67 pruned. [0m
[32m[I 2023-04-21 16:30:07,010][0m Trial 68 pruned. [0m
[32m[I 2023-04-21 16:30:25,869][0m Trial 69 pruned. [0m
[32m[I 2023-04-21 16:30:45,914][0m Trial 70 pruned. [0m
[32m[I 2023-04-21 16:31:23,335][0m Trial 71 pruned. [0m
[32m[I 2023-04-21 16:31:42,252][0m Trial 72 pruned. [0m
[32m[I 2023-04-21 16:32:19,703][0m Trial 73 pruned. [0m
[32m[I 2023-04-21 16:33:16,088][0m Trial 74 pruned. [0m
[32m[I 2023-04-21 16:33:35,085][0m Trial 75 pruned. [0m
[32m[I 2023-04-21 16:33:54,287][0m Trial 76 pruned. [0m
[32m[I 2023-04-21 16:34:14,184][0m Trial 77 pruned. [0m
[32m[I 2023-04-21 16:34:33,144][0m Trial 78 pruned. [0m
[32m[I 2023-04-21 16:34:53,666][0m Trial 79 pruned. [0m
[32m[I 2023-04-21 16:35:13,054][0m Trial 80 pruned. [0m
[32m[I 2023-04-21 16:35:32,040][0m Trial 81 pruned. [0m
[32m[I 2023-04-21 16:35:51,217][0m Trial 82 pruned. [0m
[32m[I 2023-04-21 16:36:10,126][0m Trial 83 pruned. [0m
[32m[I 2023-04-21 16:36:48,314][0m Trial 84 pruned. [0m
[32m[I 2023-04-21 16:37:45,272][0m Trial 85 pruned. [0m
[32m[I 2023-04-21 16:38:04,248][0m Trial 86 pruned. [0m
[32m[I 2023-04-21 16:41:51,557][0m Trial 87 finished with value:
0.889359712600708 and parameters: {'num_conv_layers': 3, 'lr':
0.00033551691901134164, 'n_epochs': 12}. Best is trial 0 with value:
0.8934778571128845.[0m
[32m[I 2023-04-21 16:42:10,842][0m Trial 88 pruned. [0m
[32m[I 2023-04-21 16:42:30,681][0m Trial 89 pruned. [0m
```



```
[32m[I 2023-04-21 16:42:50,000][0m Trial 90 pruned. [0m
[32m[I 2023-04-21 16:43:08,877][0m Trial 91 pruned. [0m
[32m[I 2023-04-21 16:43:27,793][0m Trial 92 pruned. [0m
[32m[I 2023-04-21 16:43:46,669][0m Trial 93 pruned. [0m
[32m[I 2023-04-21 16:44:05,597][0m Trial 94 pruned. [0m
[32m[I  2023-04-21  16:48:28,336][0m  Trial  95  finished  with  value:
0.8904926776885986  and  parameters:  {'num_conv_layers':  3,  'lr':
0.000533493026961726,  'n_epochs':  14}.  Best  is  trial  0  with  value:
0.8934778571128845.[0m
[32m[I 2023-04-21 16:48:47,109][0m Trial 96 pruned. [0m
[32m[I 2023-04-21 16:49:07,072][0m Trial 97 pruned. [0m
[32m[I 2023-04-21 16:49:25,862][0m Trial 98 pruned. [0m
[32m[I 2023-04-21 16:49:44,739][0m Trial 99 pruned. [0m
device: cuda
Total 1875 of train each of 32 batches.
Total 313 of test each of 32 batches.
None

Study statistics:
 Number of finished trials:  100
 Number of pruned trials:  87
 Number of complete trials:  13
Best trial:
 Value:  0.8934778571128845
 Params:
   num_conv_layers: 3
   lr: 0.0003290901991583123
   n_epochs: 12
```

Observation:

- It take less time than traditional methods because it pruned many of epochs with result are not improve
- For 100 hyperparameter combination it take just 2 hours but traditional method take more than 4 hours
- Also give best outcomes compared to the old school method.

Refrences:

- https://www.kaggle.com/code/just4jcgeorge/dcgan-fashion-mnist-pytorch/notebook
- https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html
- https://github.com/elena-ecn/optuna-optimization-for-PyTorch-CNN
- https://www.analyticsvidhya.com/blog/2020/11/hyperparameter-tuning-using-optuna/
- Also from DR. Anush Lectures