Modulation Classification

جامعة الإسكندرية

ALEXANDRIA
UNIVERSITY

Name 1: Abdelrahman Salem Mohamed

ID 1: 6309

Name 2: Reem Abdelhalim

ID 2: 6114

Name 3: Salma Ahmed

ID 3: 6126

Problem Statement

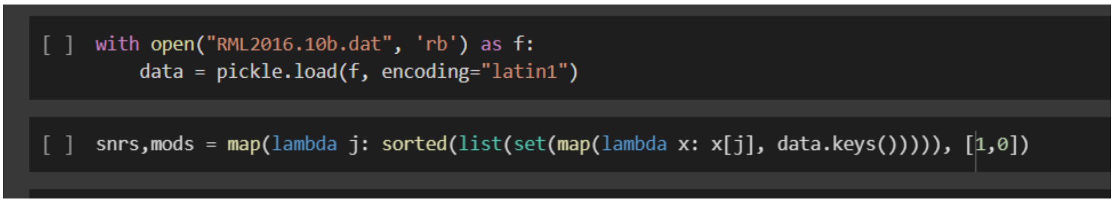
Classify the modulating signals that being generated by the transmitted and sent to the receiver then the receiver decodes it and de-modulate to extract the original signal

Source Code

- CNN
- LSTM
- RNN

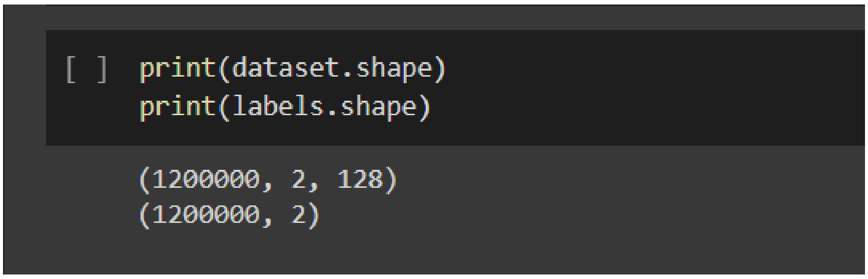
Procedures

1- Download and Upload the RML2016.10b



```
[ ] with open("RML2016.10b.dat", 'rb') as f:
        data = pickle.load(f, encoding="latin1")

[ ] snrs,mods = map(lambda j: sorted(list(set(map(lambda x: x[j], data.keys())))), [1,0])
```

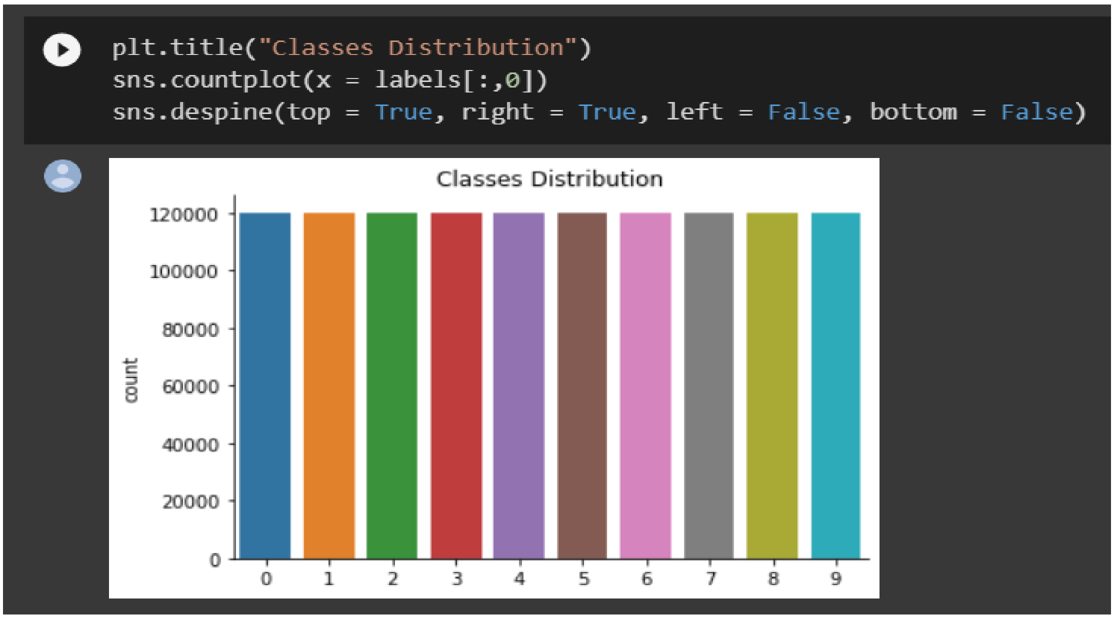
The dataset shape is (1200000,2,128)



```
[ ] print(dataset.shape)
    print(labels.shape)

    (1200000, 2, 128)
    (1200000, 2)
```

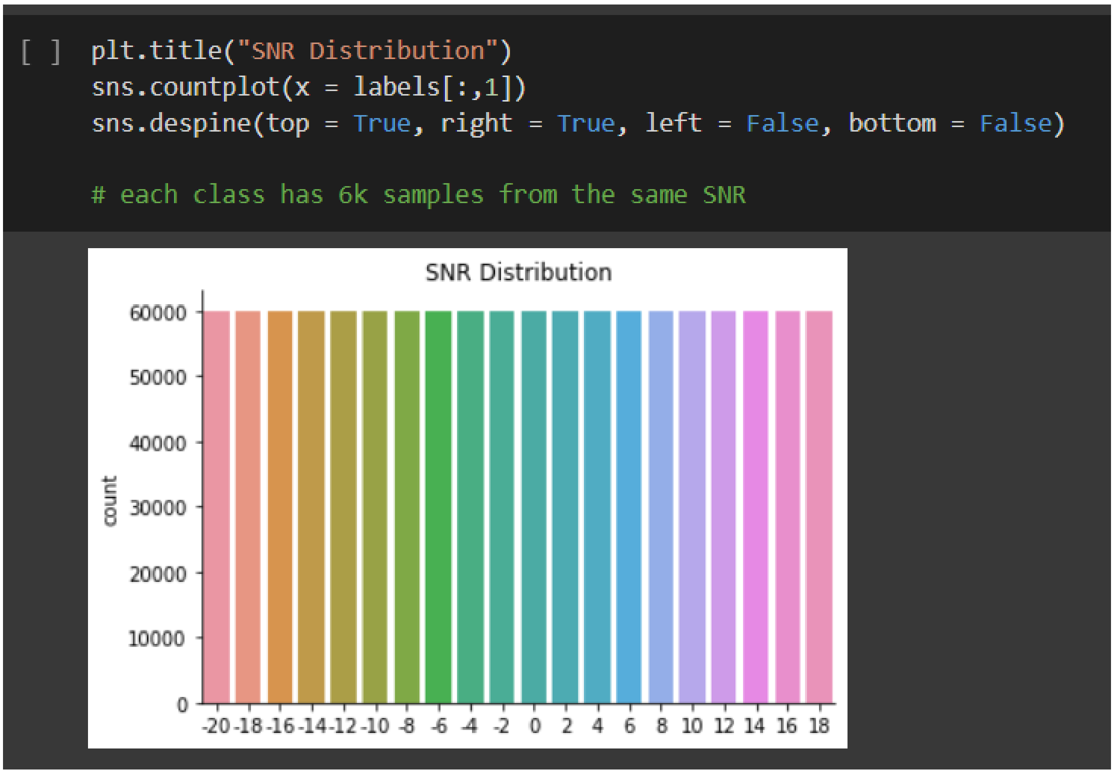
Classes Distribution

- Modulation Type



```
plt.title("Classes Distribution")
sns.countplot(x = labels[:,0])
sns.despine(top = True, right = True, left = False, bottom = False)
```

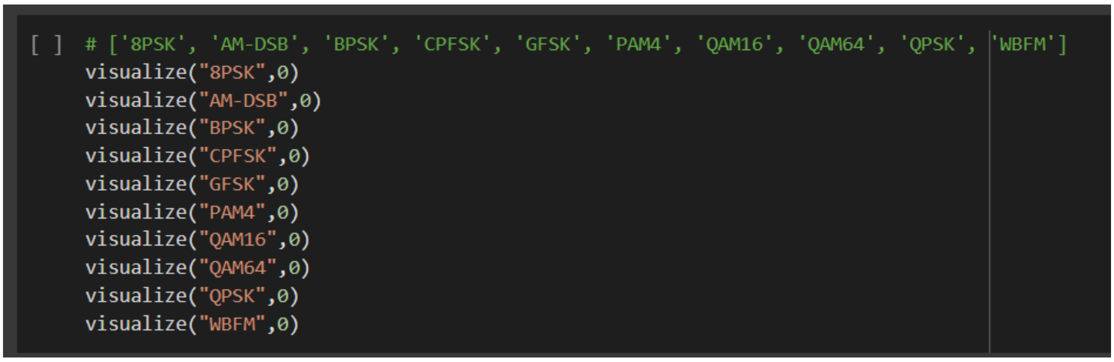
- Signal to Noise Ration (SNR)



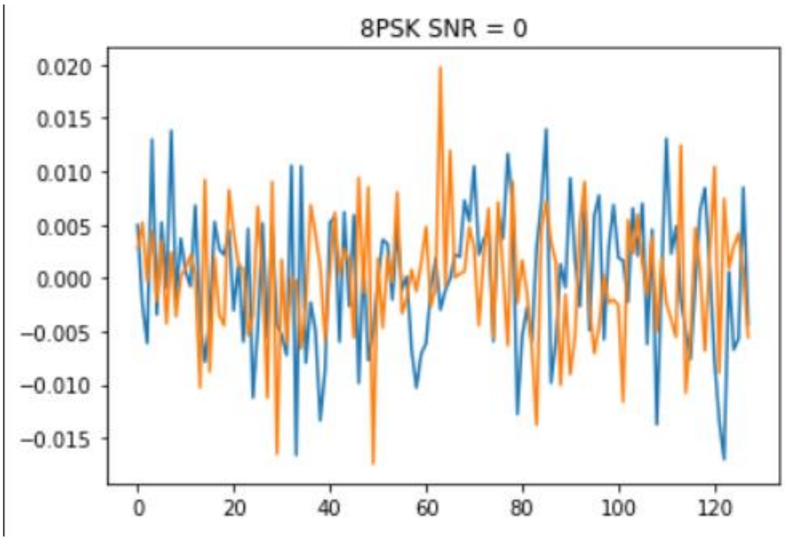
```
plt.title("SNR Distribution")
sns.countplot(x = labels[:,1])
sns.despine(top = True, right = True, left = False, bottom = False)

# each class has 6k samples from the same SNR
```

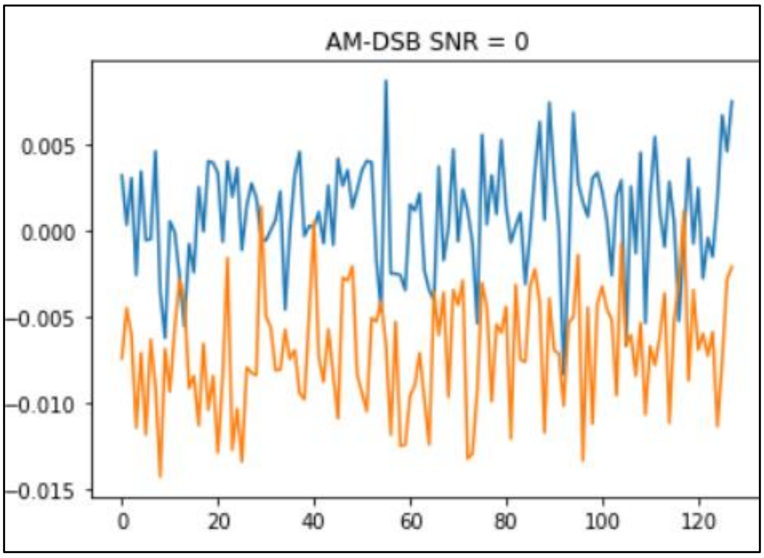
We have 10 different modulation type and 20 different SNR value

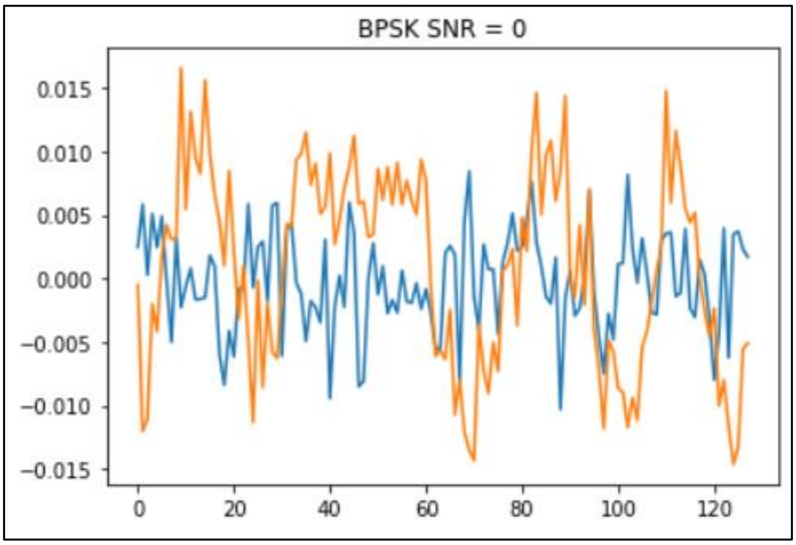
2- Explore Dataset Samples

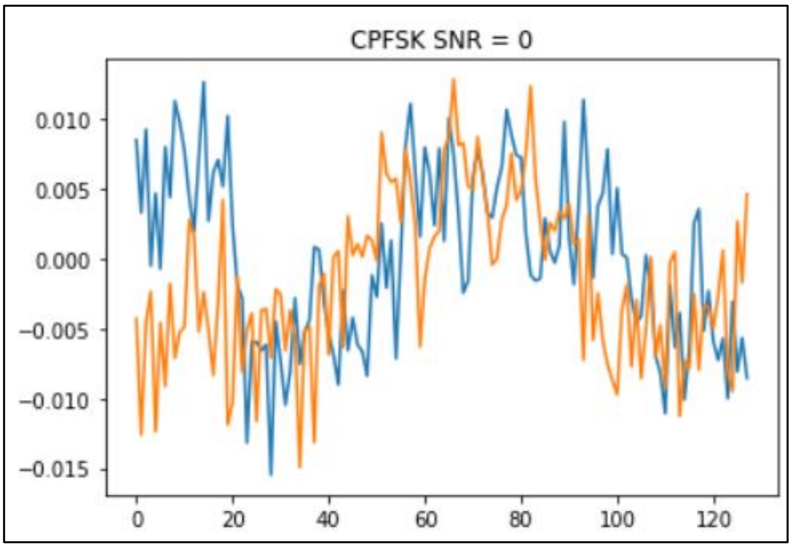


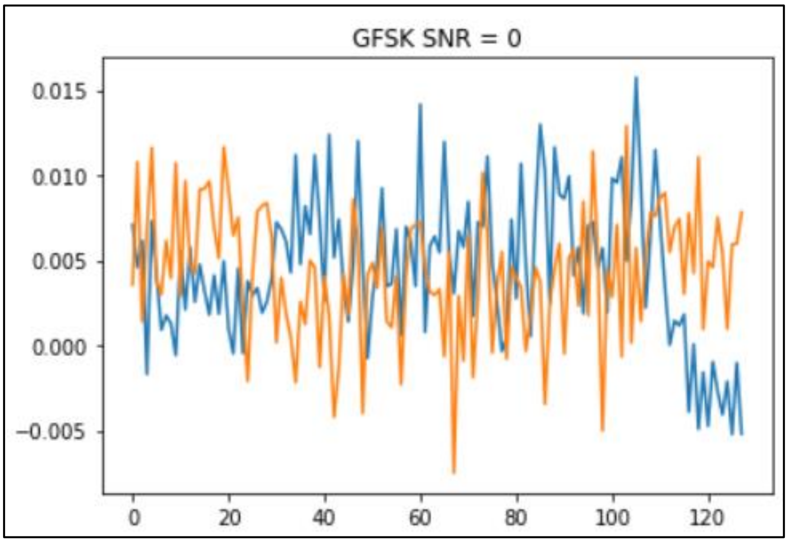
```
# ['8PSK', 'AM-DSB', 'BPSK', 'CPFSK', 'GFSK', 'PAM4', 'QAM16', 'QAM64', 'QPSK', 'WBFM']
visualize("8PSK",0)
visualize("AM-DSB",0)
visualize("BPSK",0)
visualize("CPFSK",0)
visualize("GFSK",0)
visualize("PAM4",0)
visualize("QAM16",0)
visualize("QAM64",0)
visualize("QPSK",0)
visualize("WBFM",0)
```

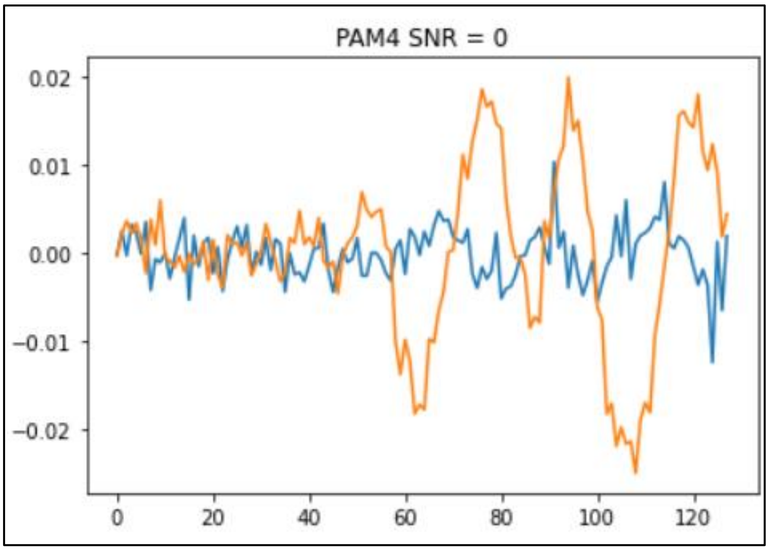


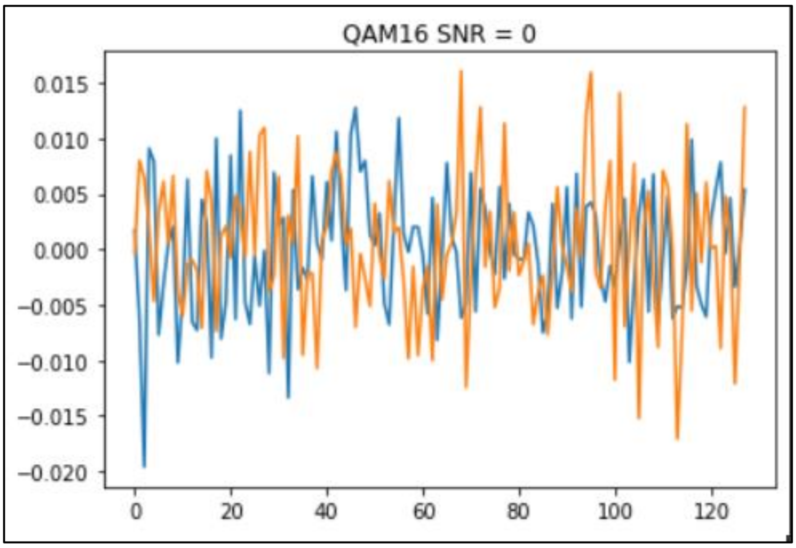

AM-DSB SNR = 0
0.005
0.000
−0.005
−0.010
−0.015
0
20
40
60
80
100
120

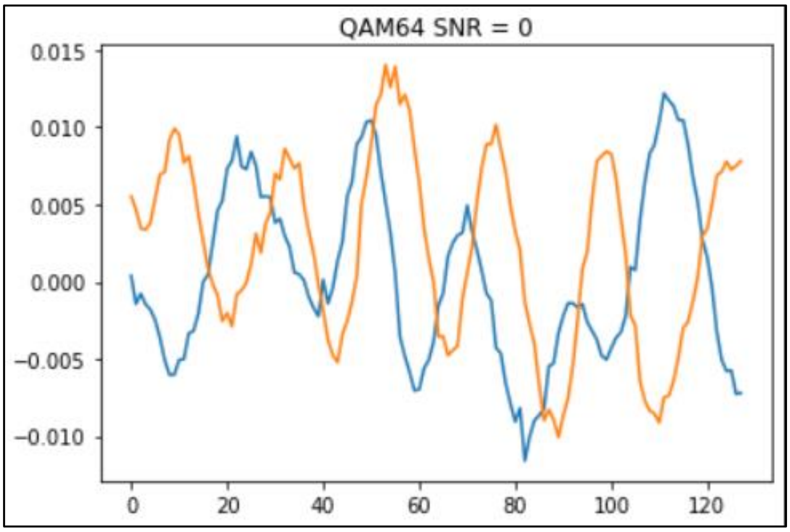

BPSK SNR = 0
0.015
0.010
0.005
0.000
−0.005
−0.010
−0.015
0
20
40
60
80
100
120

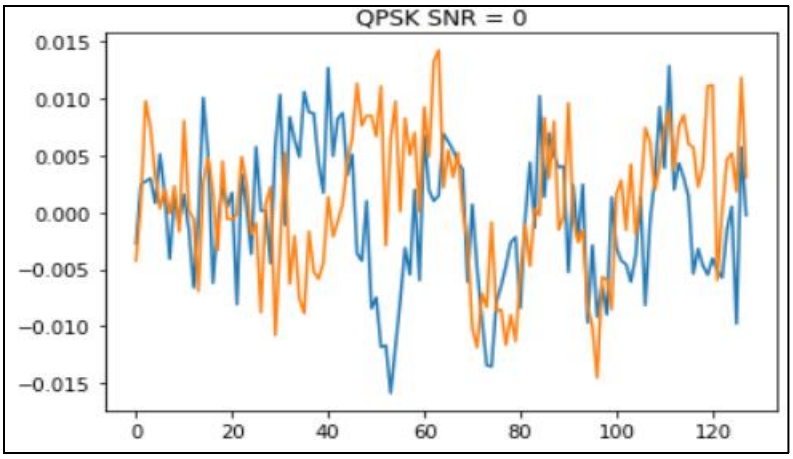

CPFSK SNR = 0
0.010
0.005
0.000
−0.005
−0.010
−0.015
0
20
40
60
80
100
120

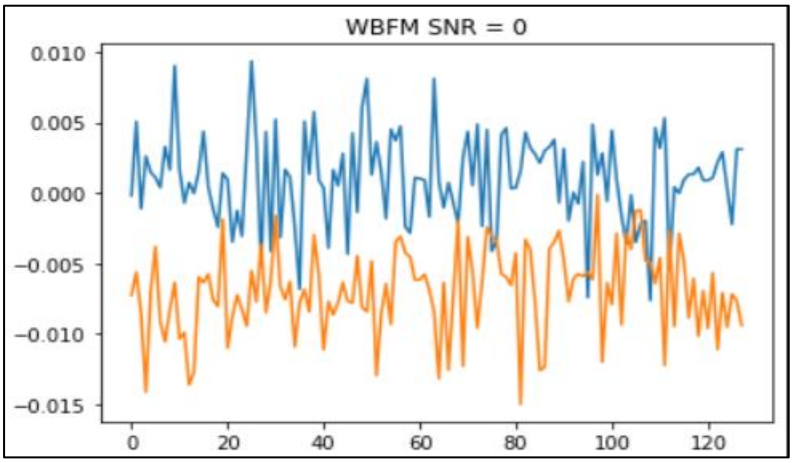

GFSK SNR = 0
0.015
0.010
0.005
0.000
−0.005
0
20
40
60
80
100
120


PAM4 SNR = 0
0.02
0.01
0.00
−0.01
−0.02
0
20
40
60
80
100
120

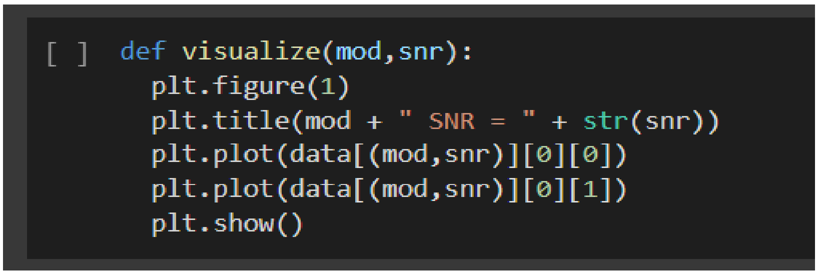

QAM16 SNR = 0
0.015
0.010
0.005
0.000
−0.005
−0.010
−0.015
−0.020
0
20
40
60
80
100
120


QAM64 SNR = 0
0.015
0.010
0.005
0.000
−0.005
−0.010
0
20
40
60
80
100
120


QPSK SNR = 0
0.015
0.010
0.005
0.000
−0.005
−0.010
−0.015
0
20
40
60
80
100
120


WBFM SNR = 0
0.010
0.005
0.000
−0.005
−0.010
−0.015
0
20
40
60
80
100
120

And here the visualize function

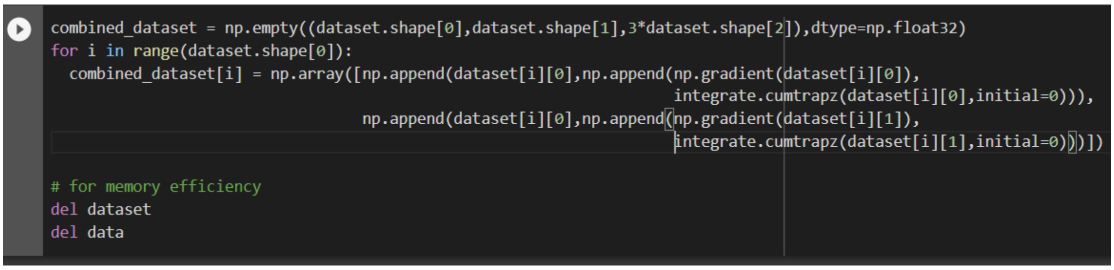


```
def visualize(mod,snr):
  plt.figure(1)
  plt.title(mod + " SNR = " + str(snr))
  plt.plot(data[(mod,snr)][0][0])
  plt.plot(data[(mod,snr)][0][1])
  plt.show()
```

3- Create Feature Space

After download and upload the dataset into our program we create the dataset that mainly consists of:

- The original signal in the time domain
- The integration of the signal w.r.t the time series using gradient method
- Differentiation of the signal w.r.t the time series using trapezoid method

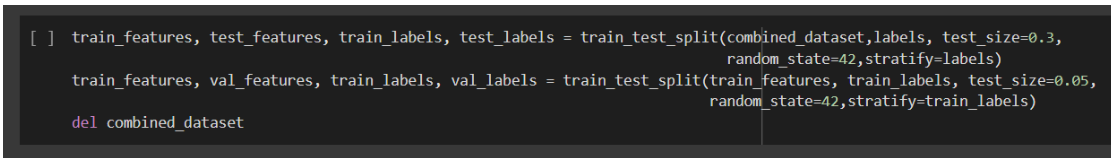


```
combined_dataset = np.empty((dataset.shape[0],dataset.shape[1],3*dataset.shape[2]),dtype=np.float32)
for i in range(dataset.shape[0]):
  combined_dataset[i] = np.array([np.append(dataset[i][0],np.append(np.gradient(dataset[i][0]),
                                                                    integrate.cumtrapz(dataset[i][0],initial=0))),
                                 np.append(dataset[i][0],np.append(np.gradient(dataset[i][1]),
                                                                    integrate.cumtrapz(dataset[i][1],initial=0)))])

# for memory efficiency
del dataset
del data
```

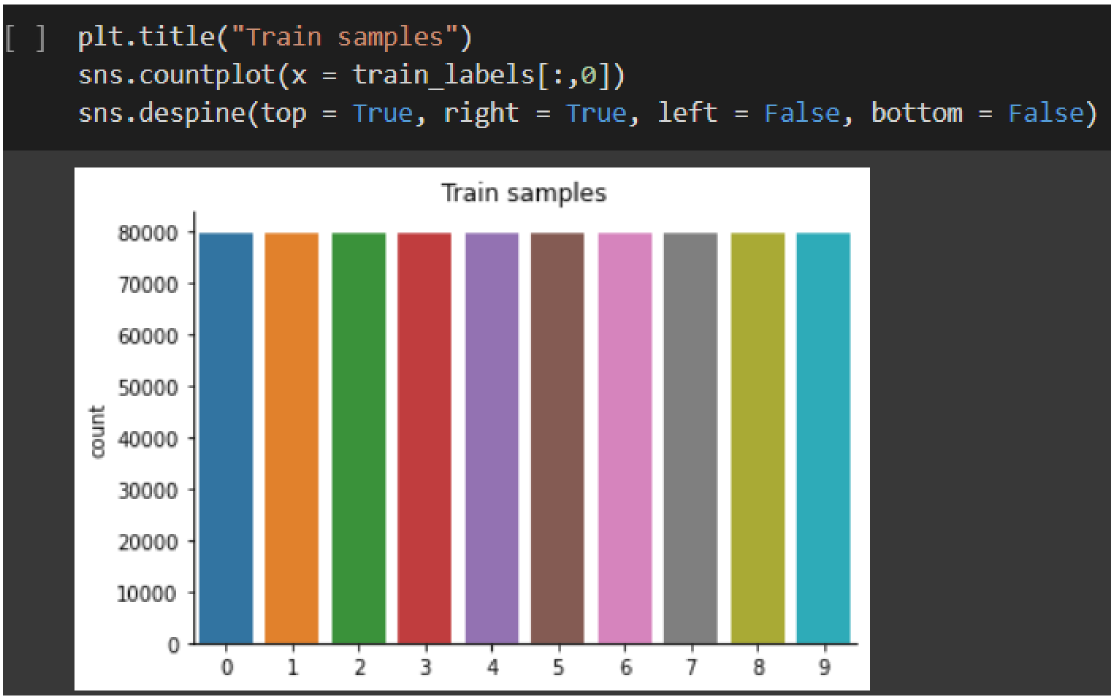
And combine all together at the end the shape will be (1200000,2,384)

4- Splitting the dataset

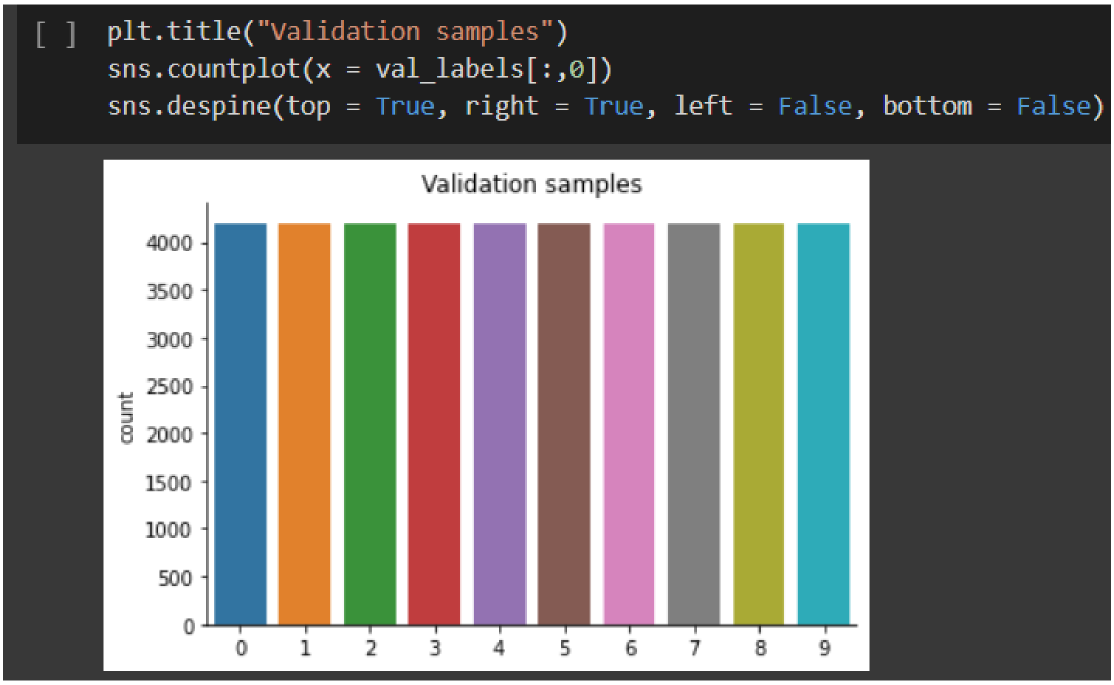
- Split the data into 70% for training/validation and 30% for testing.
- Use 5% of the training and validation dataset for validation.



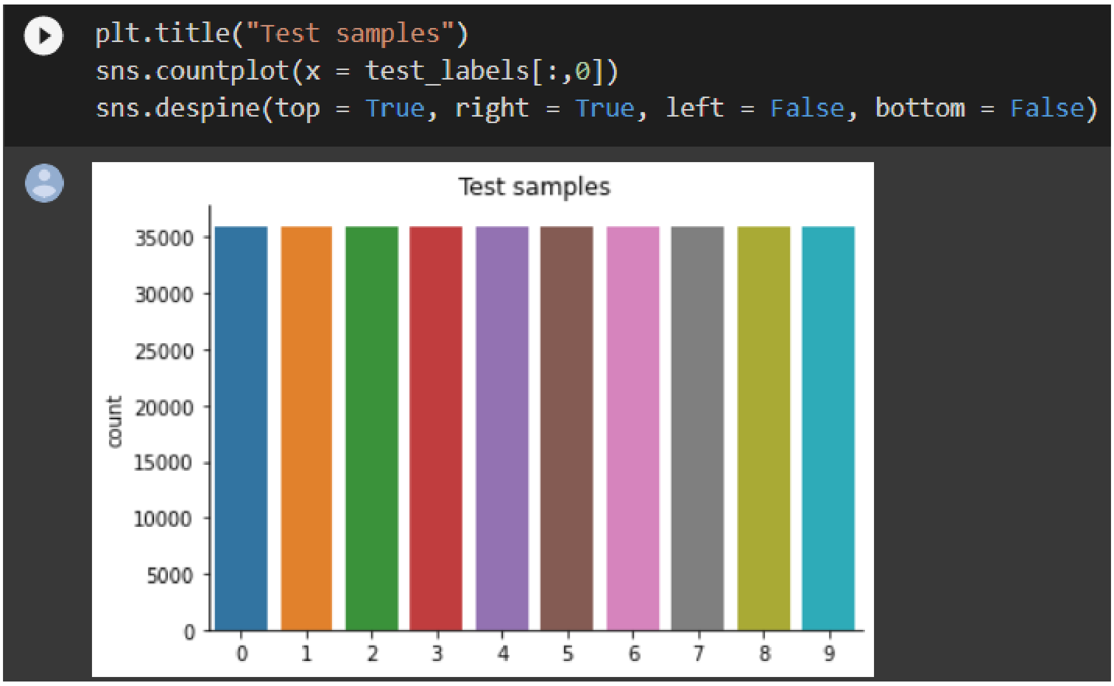
```
train_features, test_features, train_labels, test_labels = train_test_split(combined_dataset,labels, test_size=0.3,
                                                                            random_state=42,stratify=labels)
train_features, val_features, train_labels, val_labels = train_test_split(train_features, train_labels, test_size=0.05,
                                                                          random_state=42,stratify=train_labels)
del combined_dataset
```



```
[ ]  plt.title("Train samples")
     sns.countplot(x = train_labels[:,0])
     sns.despine(top = True, right = True, left = False, bottom = False)
```



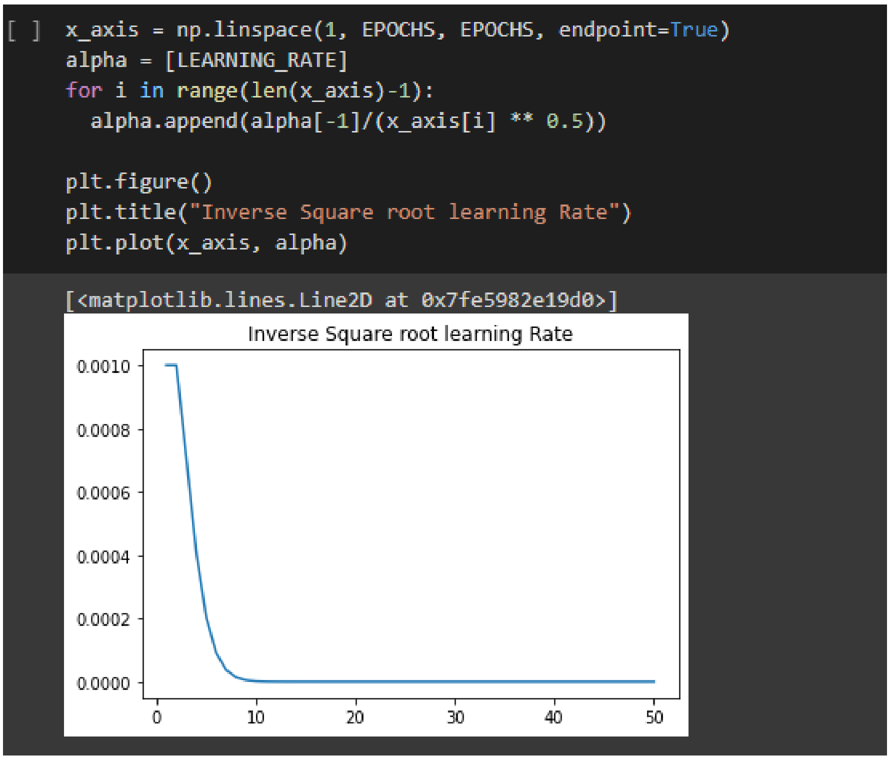
```
[ ]  plt.title("Validation samples")
     sns.countplot(x = val_labels[:,0])
     sns.despine(top = True, right = True, left = False, bottom = False)
```



```
plt.title("Test samples")
sns.countplot(x = test_labels[:,0])
sns.despine(top = True, right = True, left = False, bottom = False)
```

4- Building Models

Learning Rate: inverse square root has been applied to the learning rate on the validation dataset and select the best Learning rate depends on the accuracy



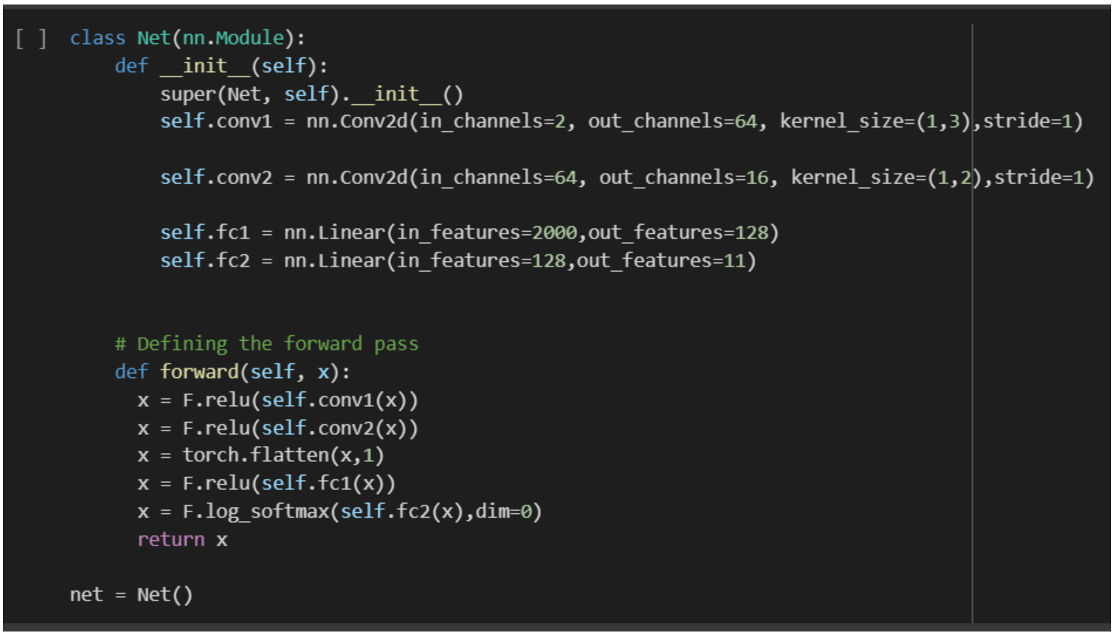
```
[ ] x_axis = np.linspace(1, EPOCHS, EPOCHS, endpoint=True)
    alpha = [LEARNING_RATE]
    for i in range(len(x_axis)-1):
      alpha.append(alpha[-1]/(x_axis[i] ** 0.5))

    plt.figure()
    plt.title("Inverse Square root learning Rate")
    plt.plot(x_axis, alpha)
```

```
[<matplotlib.lines.Line2D at 0x7fe5982e19d0>]
```

The best one was about 0.00939

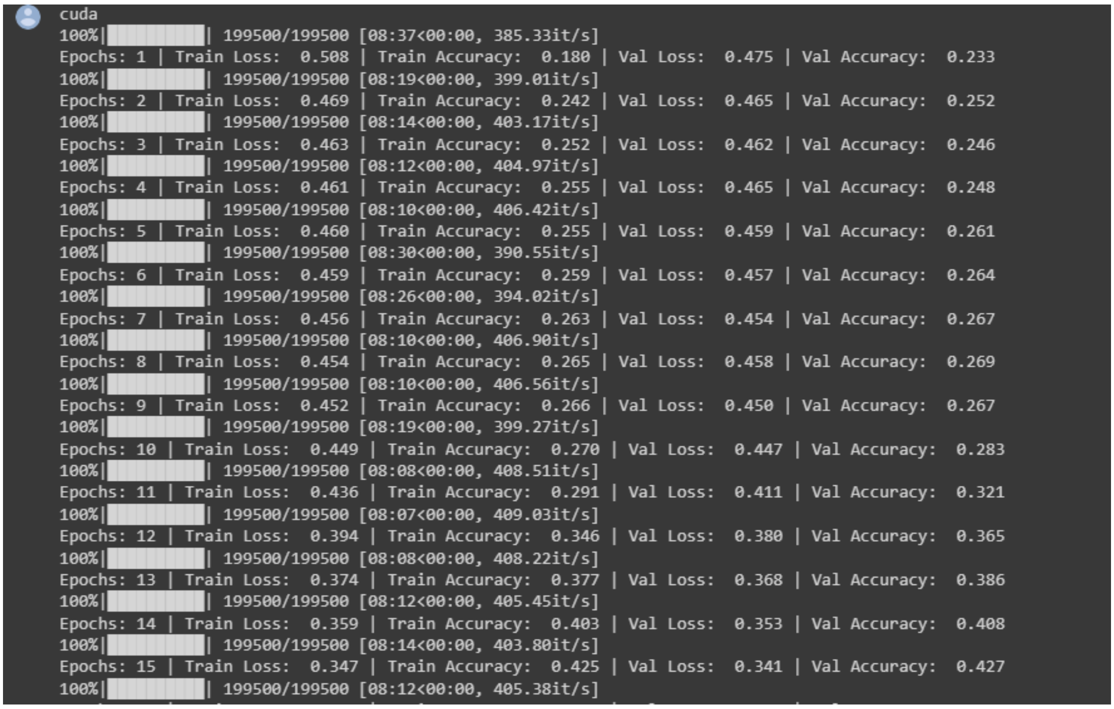
So, we select Learning Rate to be 0.001

1-CNN

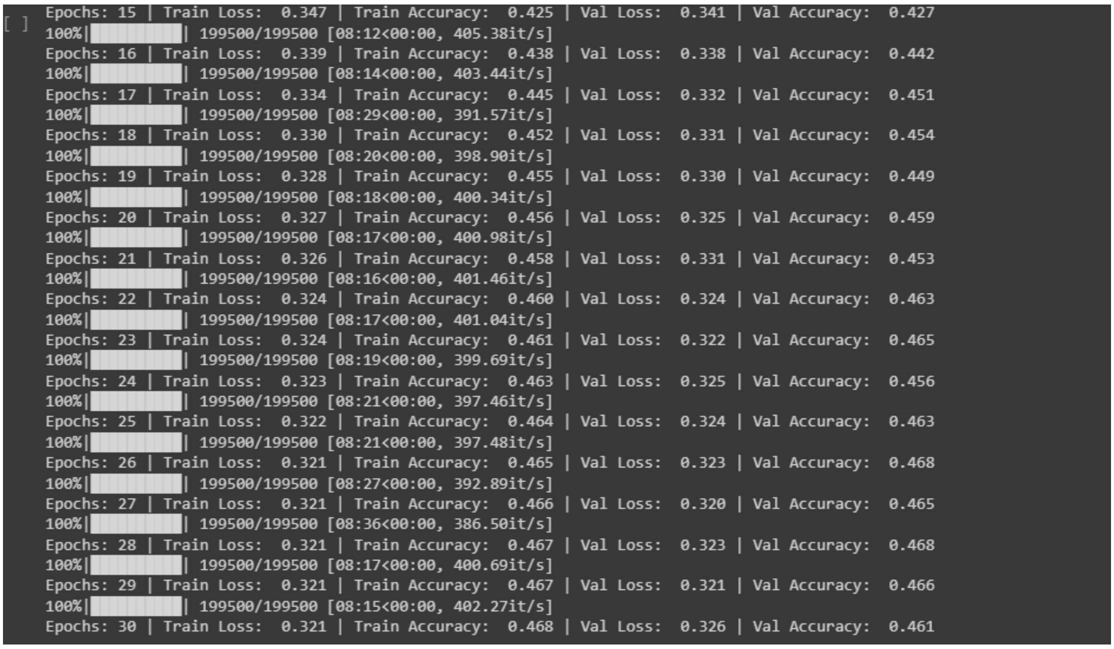


```
[ ] class Net(nn.Module):
        def __init__(self):
            super(Net, self).__init__()
            self.conv1 = nn.Conv2d(in_channels=2, out_channels=64, kernel_size=(1,3),stride=1)

            self.conv2 = nn.Conv2d(in_channels=64, out_channels=16, kernel_size=(1,2),stride=1)

            self.fc1 = nn.Linear(in_features=2000,out_features=128)
            self.fc2 = nn.Linear(in_features=128,out_features=11)


        # Defining the forward pass
        def forward(self, x):
          x = F.relu(self.conv1(x))
          x = F.relu(self.conv2(x))
          x = torch.flatten(x,1)
          x = F.relu(self.fc1(x))
          x = F.log_softmax(self.fc2(x),dim=0)
          return x

    net = Net()
```

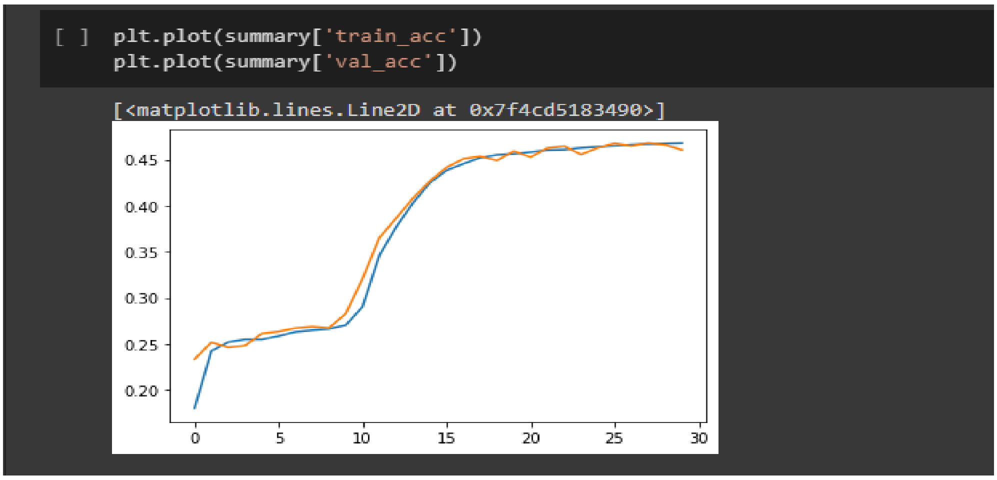
We used the architecture of the Neural Network in the assignment with 30 epochs and batch size = 8



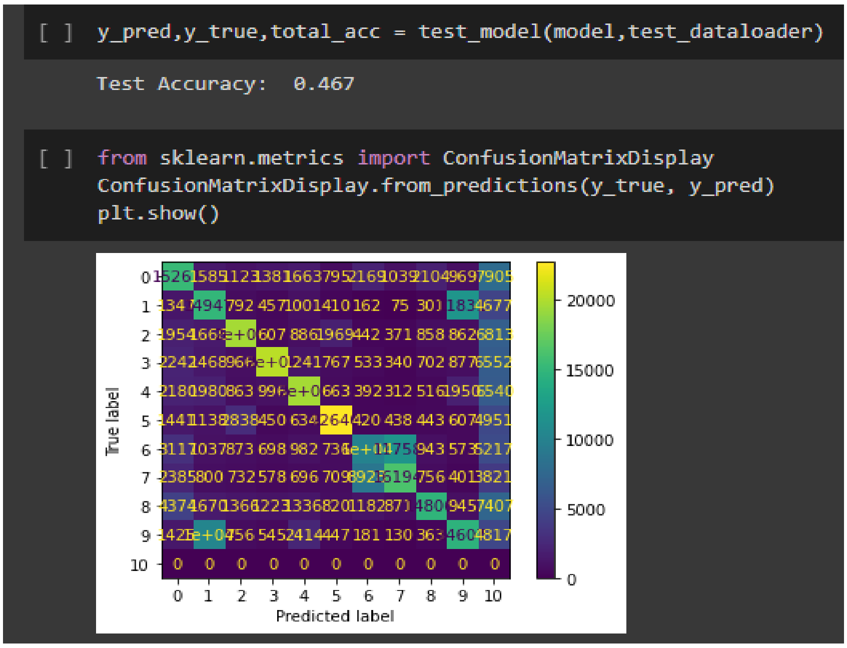
```
cuda
100%|██████████| 199500/199500 [08:37<00:00, 385.33it/s]
Epochs: 1 | Train Loss:  0.508 | Train Accuracy:  0.180 | Val Loss:  0.475 | Val Accuracy:  0.233
100%|██████████| 199500/199500 [08:19<00:00, 399.01it/s]
Epochs: 2 | Train Loss:  0.469 | Train Accuracy:  0.242 | Val Loss:  0.465 | Val Accuracy:  0.252
100%|██████████| 199500/199500 [08:14<00:00, 403.17it/s]
Epochs: 3 | Train Loss:  0.463 | Train Accuracy:  0.252 | Val Loss:  0.462 | Val Accuracy:  0.246
100%|██████████| 199500/199500 [08:12<00:00, 404.97it/s]
Epochs: 4 | Train Loss:  0.461 | Train Accuracy:  0.255 | Val Loss:  0.465 | Val Accuracy:  0.248
100%|██████████| 199500/199500 [08:10<00:00, 406.42it/s]
Epochs: 5 | Train Loss:  0.460 | Train Accuracy:  0.255 | Val Loss:  0.459 | Val Accuracy:  0.261
100%|██████████| 199500/199500 [08:30<00:00, 390.55it/s]
Epochs: 6 | Train Loss:  0.459 | Train Accuracy:  0.259 | Val Loss:  0.457 | Val Accuracy:  0.264
100%|██████████| 199500/199500 [08:26<00:00, 394.02it/s]
Epochs: 7 | Train Loss:  0.456 | Train Accuracy:  0.263 | Val Loss:  0.454 | Val Accuracy:  0.267
100%|██████████| 199500/199500 [08:10<00:00, 406.90it/s]
Epochs: 8 | Train Loss:  0.454 | Train Accuracy:  0.265 | Val Loss:  0.458 | Val Accuracy:  0.269
100%|██████████| 199500/199500 [08:10<00:00, 406.56it/s]
Epochs: 9 | Train Loss:  0.452 | Train Accuracy:  0.266 | Val Loss:  0.450 | Val Accuracy:  0.267
100%|██████████| 199500/199500 [08:19<00:00, 399.27it/s]
Epochs: 10 | Train Loss:  0.449 | Train Accuracy:  0.270 | Val Loss:  0.447 | Val Accuracy:  0.283
100%|██████████| 199500/199500 [08:08<00:00, 408.51it/s]
Epochs: 11 | Train Loss:  0.436 | Train Accuracy:  0.291 | Val Loss:  0.411 | Val Accuracy:  0.321
100%|██████████| 199500/199500 [08:07<00:00, 409.03it/s]
Epochs: 12 | Train Loss:  0.394 | Train Accuracy:  0.346 | Val Loss:  0.380 | Val Accuracy:  0.365
100%|██████████| 199500/199500 [08:08<00:00, 408.22it/s]
Epochs: 13 | Train Loss:  0.374 | Train Accuracy:  0.377 | Val Loss:  0.368 | Val Accuracy:  0.386
100%|██████████| 199500/199500 [08:12<00:00, 405.45it/s]
Epochs: 14 | Train Loss:  0.359 | Train Accuracy:  0.403 | Val Loss:  0.353 | Val Accuracy:  0.408
100%|██████████| 199500/199500 [08:14<00:00, 403.80it/s]
Epochs: 15 | Train Loss:  0.347 | Train Accuracy:  0.425 | Val Loss:  0.341 | Val Accuracy:  0.427
100%|██████████| 199500/199500 [08:12<00:00, 405.38it/s]
```



```
[ ] Epochs: 15 | Train Loss:  0.347 | Train Accuracy:  0.425 | Val Loss:  0.341 | Val Accuracy:  0.427
    100%|██████████| 199500/199500 [08:12<00:00, 405.38it/s]
    Epochs: 16 | Train Loss:  0.339 | Train Accuracy:  0.438 | Val Loss:  0.338 | Val Accuracy:  0.442
    100%|██████████| 199500/199500 [08:14<00:00, 403.44it/s]
    Epochs: 17 | Train Loss:  0.334 | Train Accuracy:  0.445 | Val Loss:  0.332 | Val Accuracy:  0.451
    100%|██████████| 199500/199500 [08:29<00:00, 391.57it/s]
    Epochs: 18 | Train Loss:  0.330 | Train Accuracy:  0.452 | Val Loss:  0.331 | Val Accuracy:  0.454
    100%|██████████| 199500/199500 [08:20<00:00, 398.90it/s]
    Epochs: 19 | Train Loss:  0.328 | Train Accuracy:  0.455 | Val Loss:  0.330 | Val Accuracy:  0.449
    100%|██████████| 199500/199500 [08:18<00:00, 400.34it/s]
    Epochs: 20 | Train Loss:  0.327 | Train Accuracy:  0.456 | Val Loss:  0.325 | Val Accuracy:  0.459
    100%|██████████| 199500/199500 [08:17<00:00, 400.98it/s]
    Epochs: 21 | Train Loss:  0.326 | Train Accuracy:  0.458 | Val Loss:  0.331 | Val Accuracy:  0.453
    100%|██████████| 199500/199500 [08:16<00:00, 401.46it/s]
    Epochs: 22 | Train Loss:  0.324 | Train Accuracy:  0.460 | Val Loss:  0.324 | Val Accuracy:  0.463
    100%|██████████| 199500/199500 [08:17<00:00, 401.04it/s]
    Epochs: 23 | Train Loss:  0.324 | Train Accuracy:  0.461 | Val Loss:  0.322 | Val Accuracy:  0.465
    100%|██████████| 199500/199500 [08:19<00:00, 399.69it/s]
    Epochs: 24 | Train Loss:  0.323 | Train Accuracy:  0.463 | Val Loss:  0.325 | Val Accuracy:  0.456
    100%|██████████| 199500/199500 [08:21<00:00, 397.46it/s]
    Epochs: 25 | Train Loss:  0.322 | Train Accuracy:  0.464 | Val Loss:  0.324 | Val Accuracy:  0.463
    100%|██████████| 199500/199500 [08:21<00:00, 397.48it/s]
    Epochs: 26 | Train Loss:  0.321 | Train Accuracy:  0.465 | Val Loss:  0.323 | Val Accuracy:  0.468
    100%|██████████| 199500/199500 [08:27<00:00, 392.89it/s]
    Epochs: 27 | Train Loss:  0.321 | Train Accuracy:  0.466 | Val Loss:  0.320 | Val Accuracy:  0.465
    100%|██████████| 199500/199500 [08:36<00:00, 386.50it/s]
    Epochs: 28 | Train Loss:  0.321 | Train Accuracy:  0.467 | Val Loss:  0.323 | Val Accuracy:  0.468
    100%|██████████| 199500/199500 [08:17<00:00, 400.69it/s]
    Epochs: 29 | Train Loss:  0.321 | Train Accuracy:  0.467 | Val Loss:  0.321 | Val Accuracy:  0.466
    100%|██████████| 199500/199500 [08:15<00:00, 402.27it/s]
    Epochs: 30 | Train Loss:  0.321 | Train Accuracy:  0.468 | Val Loss:  0.326 | Val Accuracy:  0.461
```



```
[ ] plt.plot(summary['train_acc'])
    plt.plot(summary['val_acc'])

    [<matplotlib.lines.Line2D at 0x7f4cd5183490>]
```

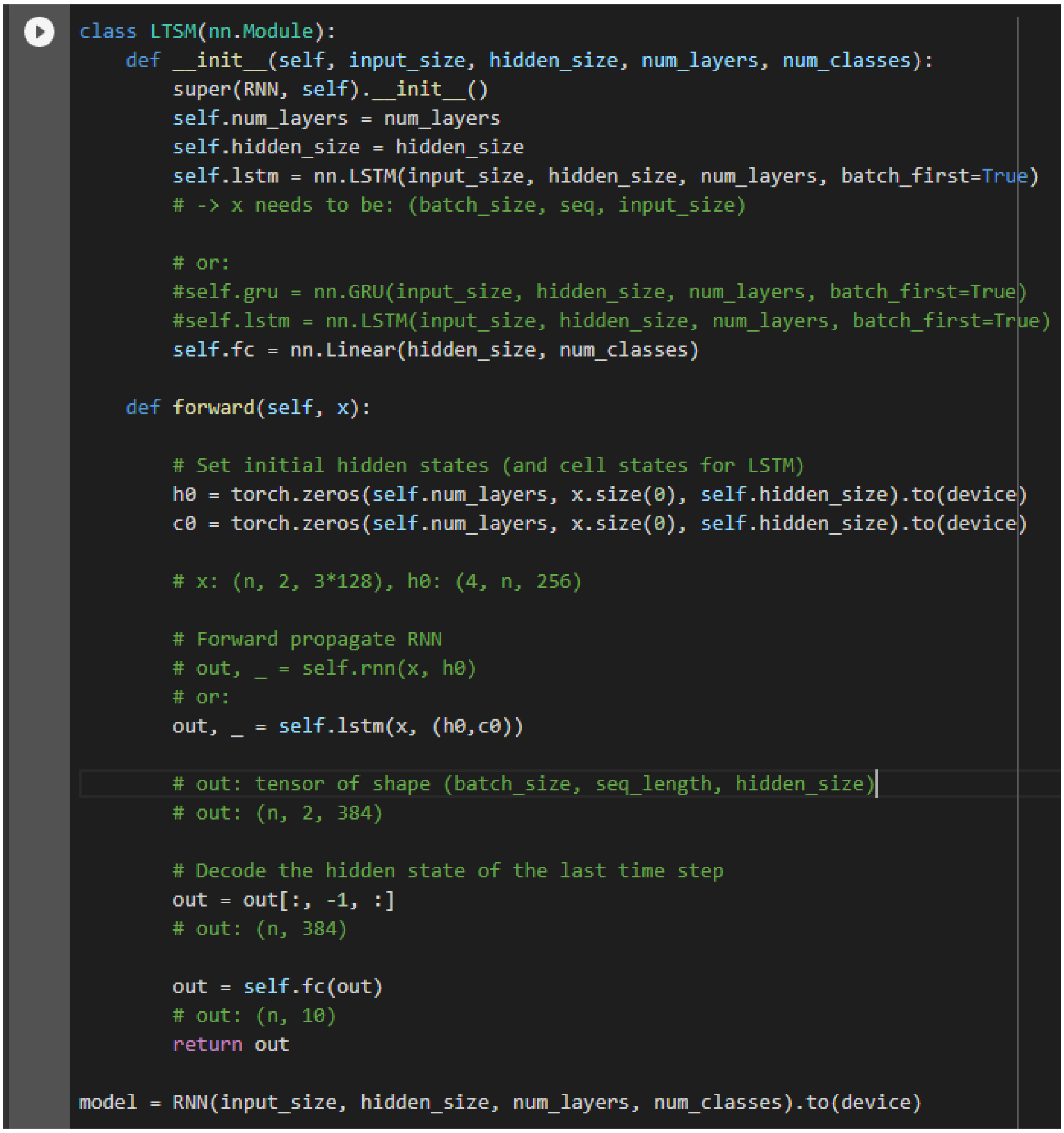


```
[ ] y_pred,y_true,total_acc = test_model(model,test_dataloader)

    Test Accuracy:  0.467
```

```
[ ] from sklearn.metrics import ConfusionMatrixDisplay
    ConfusionMatrixDisplay.from_predictions(y_true, y_pred)
    plt.show()
```

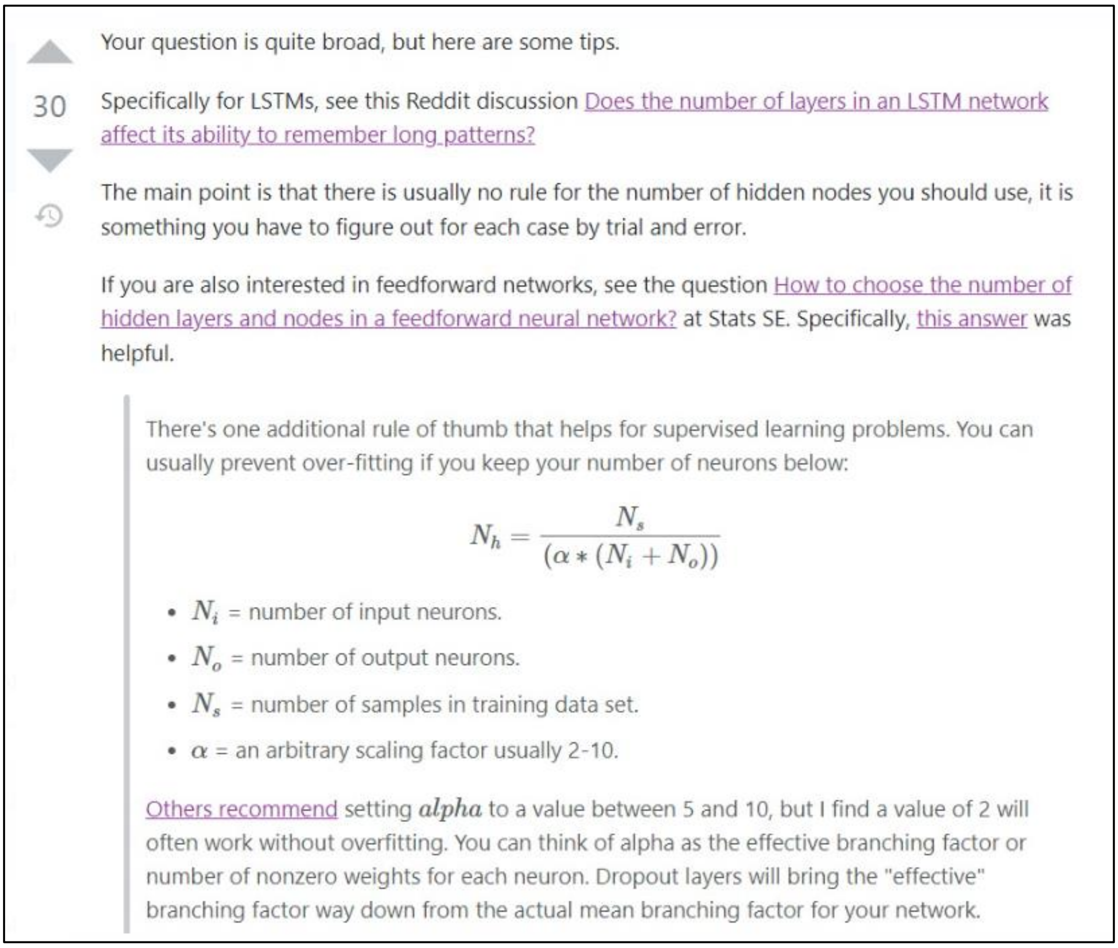
2- LSTM

With 4 hidden layers and each layer has 256 neurons



```python
class LTSM(nn.Module):
    def __init__(self, input_size, hidden_size, num_layers, num_classes):
        super(RNN, self).__init__()
        self.num_layers = num_layers
        self.hidden_size = hidden_size
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers, batch_first=True)
        # -> x needs to be: (batch_size, seq, input_size)

        # or:
        #self.gru = nn.GRU(input_size, hidden_size, num_layers, batch_first=True)
        #self.lstm = nn.LSTM(input_size, hidden_size, num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_size, num_classes)

    def forward(self, x):

        # Set initial hidden states (and cell states for LSTM)
        h0 = torch.zeros(self.num_layers, x.size(0), self.hidden_size).to(device)
        c0 = torch.zeros(self.num_layers, x.size(0), self.hidden_size).to(device)

        # x: (n, 2, 3*128), h0: (4, n, 256)

        # Forward propagate RNN
        # out, _ = self.rnn(x, h0)
        # or:
        out, _ = self.lstm(x, (h0,c0))

        # out: tensor of shape (batch_size, seq_length, hidden_size)
        # out: (n, 2, 384)

        # Decode the hidden state of the last time step
        out = out[:, -1, :]
        # out: (n, 384)

        out = self.fc(out)
        # out: (n, 10)
        return out

model = RNN(input_size, hidden_size, num_layers, num_classes).to(device)
```

We just search for how to pick the number of layers and layer size and we just figure out that the recommended combination is:



30

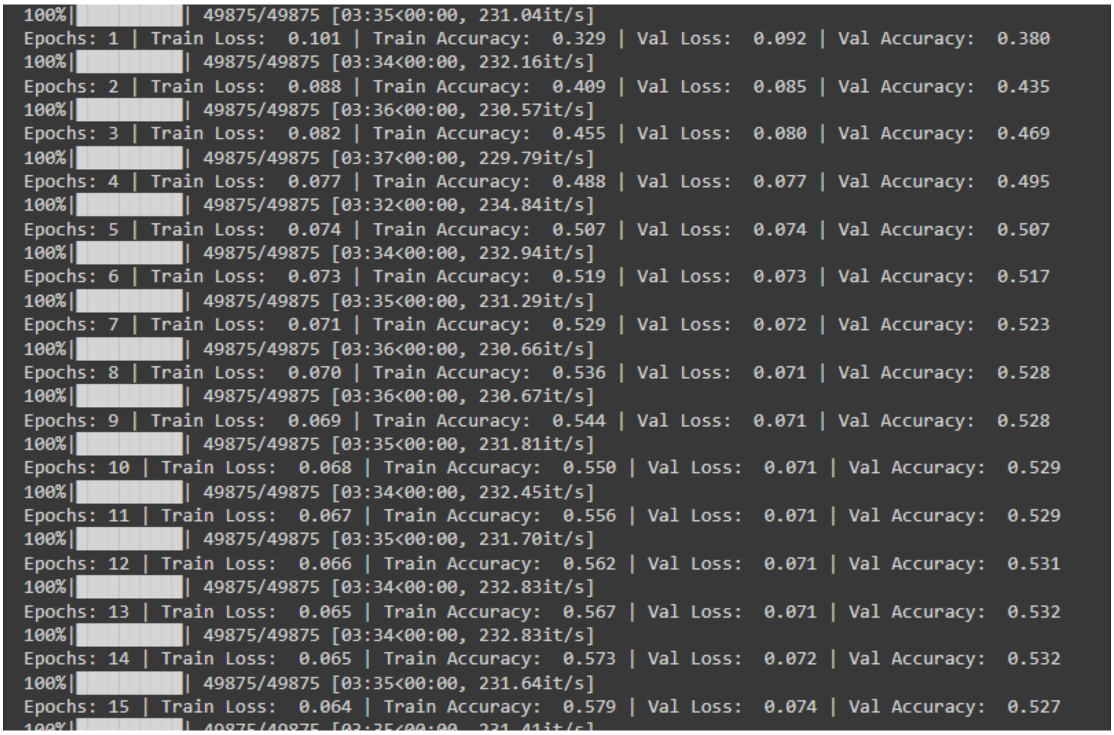
Your question is quite broad, but here are some tips.

Specifically for LSTMs, see this Reddit discussion Does the number of layers in an LSTM network affect its ability to remember long patterns?

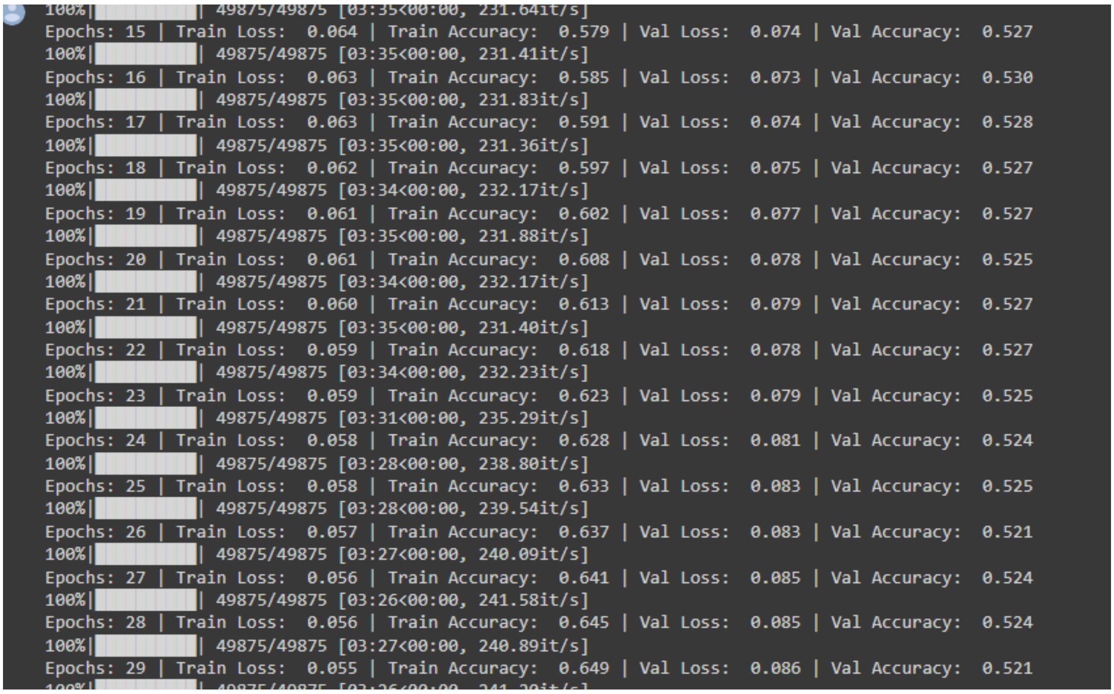
The main point is that there is usually no rule for the number of hidden nodes you should use, it is something you have to figure out for each case by trial and error.

If you are also interested in feedforward networks, see the question How to choose the number of hidden layers and nodes in a feedforward neural network? at Stats SE. Specifically, this answer was helpful.

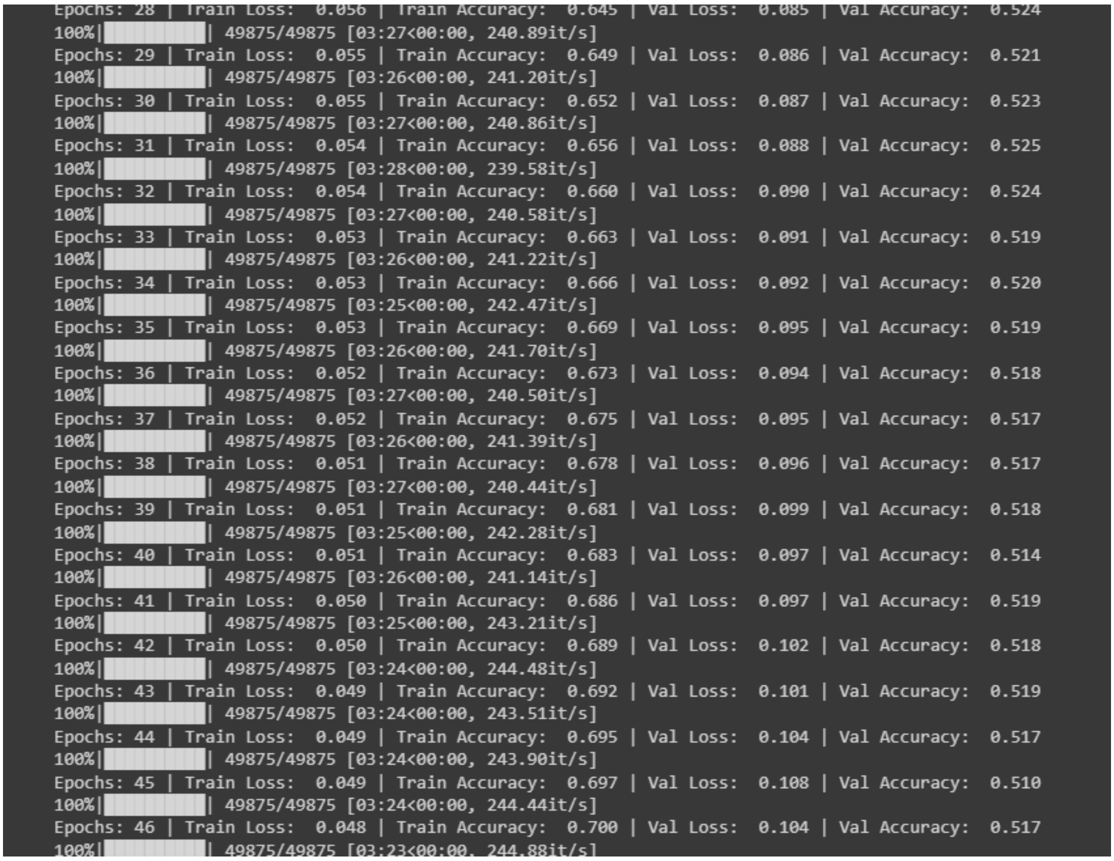
> There's one additional rule of thumb that helps for supervised learning problems. You can usually prevent over-fitting if you keep your number of neurons below:
>
> $$N_h = \frac{N_s}{(\alpha * (N_i + N_o))}$$
>
> - N_i = number of input neurons.
> - N_o = number of output neurons.
> - N_s = number of samples in training data set.
> - α = an arbitrary scaling factor usually 2-10.
>
> Others recommend setting $alpha$ to a value between 5 and 10, but I find a value of 2 will often work without overfitting. You can think of alpha as the effective branching factor or number of nonzero weights for each neuron. Dropout layers will bring the "effective" branching factor way down from the actual mean branching factor for your network.



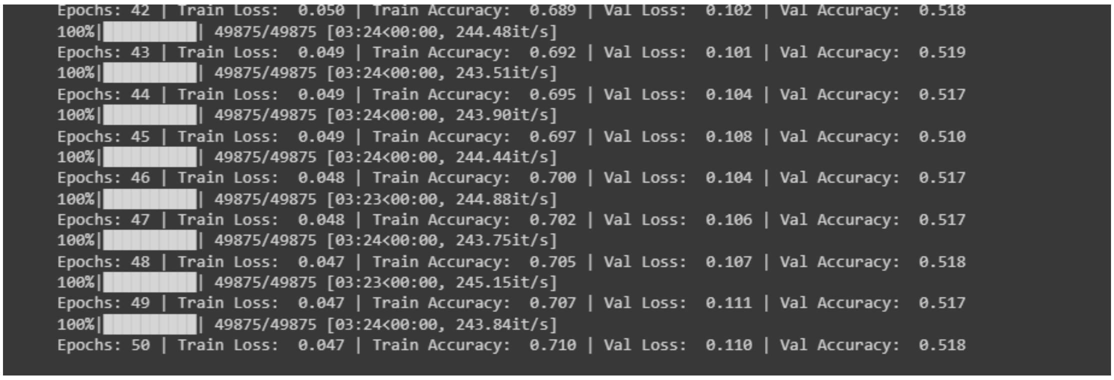
```
100%|██████████| 49875/49875 [03:35<00:00, 231.04it/s]
Epochs: 1 | Train Loss:  0.101 | Train Accuracy:  0.329 | Val Loss:  0.092 | Val Accuracy:  0.380
100%|██████████| 49875/49875 [03:34<00:00, 232.16it/s]
Epochs: 2 | Train Loss:  0.088 | Train Accuracy:  0.409 | Val Loss:  0.085 | Val Accuracy:  0.435
100%|██████████| 49875/49875 [03:36<00:00, 230.57it/s]
Epochs: 3 | Train Loss:  0.082 | Train Accuracy:  0.455 | Val Loss:  0.080 | Val Accuracy:  0.469
100%|██████████| 49875/49875 [03:37<00:00, 229.79it/s]
Epochs: 4 | Train Loss:  0.077 | Train Accuracy:  0.488 | Val Loss:  0.077 | Val Accuracy:  0.495
100%|██████████| 49875/49875 [03:32<00:00, 234.84it/s]
Epochs: 5 | Train Loss:  0.074 | Train Accuracy:  0.507 | Val Loss:  0.074 | Val Accuracy:  0.507
100%|██████████| 49875/49875 [03:34<00:00, 232.94it/s]
Epochs: 6 | Train Loss:  0.073 | Train Accuracy:  0.519 | Val Loss:  0.073 | Val Accuracy:  0.517
100%|██████████| 49875/49875 [03:35<00:00, 231.29it/s]
Epochs: 7 | Train Loss:  0.071 | Train Accuracy:  0.529 | Val Loss:  0.072 | Val Accuracy:  0.523
100%|██████████| 49875/49875 [03:36<00:00, 230.66it/s]
Epochs: 8 | Train Loss:  0.070 | Train Accuracy:  0.536 | Val Loss:  0.071 | Val Accuracy:  0.528
100%|██████████| 49875/49875 [03:36<00:00, 230.67it/s]
Epochs: 9 | Train Loss:  0.069 | Train Accuracy:  0.544 | Val Loss:  0.071 | Val Accuracy:  0.528
100%|██████████| 49875/49875 [03:35<00:00, 231.81it/s]
Epochs: 10 | Train Loss:  0.068 | Train Accuracy:  0.550 | Val Loss:  0.071 | Val Accuracy:  0.529
100%|██████████| 49875/49875 [03:34<00:00, 232.45it/s]
Epochs: 11 | Train Loss:  0.067 | Train Accuracy:  0.556 | Val Loss:  0.071 | Val Accuracy:  0.529
100%|██████████| 49875/49875 [03:35<00:00, 231.70it/s]
Epochs: 12 | Train Loss:  0.066 | Train Accuracy:  0.562 | Val Loss:  0.071 | Val Accuracy:  0.531
100%|██████████| 49875/49875 [03:34<00:00, 232.83it/s]
Epochs: 13 | Train Loss:  0.065 | Train Accuracy:  0.567 | Val Loss:  0.071 | Val Accuracy:  0.532
100%|██████████| 49875/49875 [03:34<00:00, 232.83it/s]
Epochs: 14 | Train Loss:  0.065 | Train Accuracy:  0.573 | Val Loss:  0.072 | Val Accuracy:  0.532
100%|██████████| 49875/49875 [03:35<00:00, 231.64it/s]
Epochs: 15 | Train Loss:  0.064 | Train Accuracy:  0.579 | Val Loss:  0.074 | Val Accuracy:  0.527
100%|██████████| 49875/49875 [03:35<00:00, 231.41it/s]
```



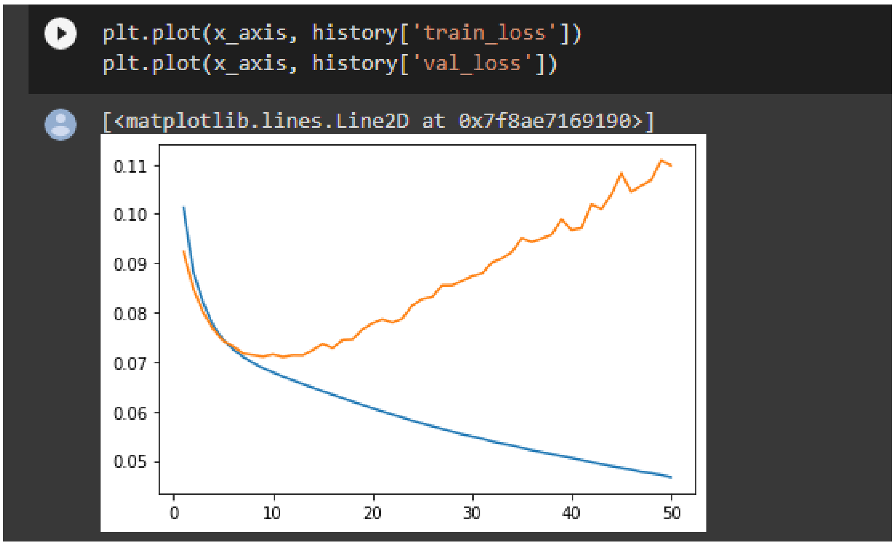
```
100%|██████████| 49875/49875 [03:35<00:00, 231.64it/s]
Epochs: 15 | Train Loss:  0.064 | Train Accuracy:  0.579 | Val Loss:  0.074 | Val Accuracy:  0.527
100%|██████████| 49875/49875 [03:35<00:00, 231.41it/s]
Epochs: 16 | Train Loss:  0.063 | Train Accuracy:  0.585 | Val Loss:  0.073 | Val Accuracy:  0.530
100%|██████████| 49875/49875 [03:35<00:00, 231.83it/s]
Epochs: 17 | Train Loss:  0.063 | Train Accuracy:  0.591 | Val Loss:  0.074 | Val Accuracy:  0.528
100%|██████████| 49875/49875 [03:35<00:00, 231.36it/s]
Epochs: 18 | Train Loss:  0.062 | Train Accuracy:  0.597 | Val Loss:  0.075 | Val Accuracy:  0.527
100%|██████████| 49875/49875 [03:34<00:00, 232.17it/s]
Epochs: 19 | Train Loss:  0.061 | Train Accuracy:  0.602 | Val Loss:  0.077 | Val Accuracy:  0.527
100%|██████████| 49875/49875 [03:35<00:00, 231.88it/s]
Epochs: 20 | Train Loss:  0.061 | Train Accuracy:  0.608 | Val Loss:  0.078 | Val Accuracy:  0.525
100%|██████████| 49875/49875 [03:34<00:00, 232.17it/s]
Epochs: 21 | Train Loss:  0.060 | Train Accuracy:  0.613 | Val Loss:  0.079 | Val Accuracy:  0.527
100%|██████████| 49875/49875 [03:35<00:00, 231.40it/s]
Epochs: 22 | Train Loss:  0.059 | Train Accuracy:  0.618 | Val Loss:  0.078 | Val Accuracy:  0.527
100%|██████████| 49875/49875 [03:34<00:00, 232.23it/s]
Epochs: 23 | Train Loss:  0.059 | Train Accuracy:  0.623 | Val Loss:  0.079 | Val Accuracy:  0.525
100%|██████████| 49875/49875 [03:31<00:00, 235.29it/s]
Epochs: 24 | Train Loss:  0.058 | Train Accuracy:  0.628 | Val Loss:  0.081 | Val Accuracy:  0.524
100%|██████████| 49875/49875 [03:28<00:00, 238.80it/s]
Epochs: 25 | Train Loss:  0.058 | Train Accuracy:  0.633 | Val Loss:  0.083 | Val Accuracy:  0.525
100%|██████████| 49875/49875 [03:28<00:00, 239.54it/s]
Epochs: 26 | Train Loss:  0.057 | Train Accuracy:  0.637 | Val Loss:  0.083 | Val Accuracy:  0.521
100%|██████████| 49875/49875 [03:27<00:00, 240.09it/s]
Epochs: 27 | Train Loss:  0.056 | Train Accuracy:  0.641 | Val Loss:  0.085 | Val Accuracy:  0.524
100%|██████████| 49875/49875 [03:26<00:00, 241.58it/s]
Epochs: 28 | Train Loss:  0.056 | Train Accuracy:  0.645 | Val Loss:  0.085 | Val Accuracy:  0.524
100%|██████████| 49875/49875 [03:27<00:00, 240.89it/s]
Epochs: 29 | Train Loss:  0.055 | Train Accuracy:  0.649 | Val Loss:  0.086 | Val Accuracy:  0.521
100%|██████████| 49875/49875 [03:26<00:00, 241.20it/s]
```



```
Epochs: 28 | Train Loss:  0.056 | Train Accuracy:  0.645 | Val Loss:  0.085 | Val Accuracy:  0.524
100%|██████████| 49875/49875 [03:27<00:00, 240.89it/s]
Epochs: 29 | Train Loss:  0.055 | Train Accuracy:  0.649 | Val Loss:  0.086 | Val Accuracy:  0.521
100%|██████████| 49875/49875 [03:26<00:00, 241.20it/s]
Epochs: 30 | Train Loss:  0.055 | Train Accuracy:  0.652 | Val Loss:  0.087 | Val Accuracy:  0.523
100%|██████████| 49875/49875 [03:27<00:00, 240.86it/s]
Epochs: 31 | Train Loss:  0.054 | Train Accuracy:  0.656 | Val Loss:  0.088 | Val Accuracy:  0.525
100%|██████████| 49875/49875 [03:28<00:00, 239.58it/s]
Epochs: 32 | Train Loss:  0.054 | Train Accuracy:  0.660 | Val Loss:  0.090 | Val Accuracy:  0.524
100%|██████████| 49875/49875 [03:27<00:00, 240.58it/s]
Epochs: 33 | Train Loss:  0.053 | Train Accuracy:  0.663 | Val Loss:  0.091 | Val Accuracy:  0.519
100%|██████████| 49875/49875 [03:26<00:00, 241.22it/s]
Epochs: 34 | Train Loss:  0.053 | Train Accuracy:  0.666 | Val Loss:  0.092 | Val Accuracy:  0.520
100%|██████████| 49875/49875 [03:25<00:00, 242.47it/s]
Epochs: 35 | Train Loss:  0.053 | Train Accuracy:  0.669 | Val Loss:  0.095 | Val Accuracy:  0.519
100%|██████████| 49875/49875 [03:26<00:00, 241.70it/s]
Epochs: 36 | Train Loss:  0.052 | Train Accuracy:  0.673 | Val Loss:  0.094 | Val Accuracy:  0.518
100%|██████████| 49875/49875 [03:27<00:00, 240.50it/s]
Epochs: 37 | Train Loss:  0.052 | Train Accuracy:  0.675 | Val Loss:  0.095 | Val Accuracy:  0.517
100%|██████████| 49875/49875 [03:26<00:00, 241.39it/s]
Epochs: 38 | Train Loss:  0.051 | Train Accuracy:  0.678 | Val Loss:  0.096 | Val Accuracy:  0.517
100%|██████████| 49875/49875 [03:27<00:00, 240.44it/s]
Epochs: 39 | Train Loss:  0.051 | Train Accuracy:  0.681 | Val Loss:  0.099 | Val Accuracy:  0.518
100%|██████████| 49875/49875 [03:25<00:00, 242.28it/s]
Epochs: 40 | Train Loss:  0.051 | Train Accuracy:  0.683 | Val Loss:  0.097 | Val Accuracy:  0.514
100%|██████████| 49875/49875 [03:26<00:00, 241.14it/s]
Epochs: 41 | Train Loss:  0.050 | Train Accuracy:  0.686 | Val Loss:  0.097 | Val Accuracy:  0.519
100%|██████████| 49875/49875 [03:25<00:00, 243.21it/s]
Epochs: 42 | Train Loss:  0.050 | Train Accuracy:  0.689 | Val Loss:  0.102 | Val Accuracy:  0.518
100%|██████████| 49875/49875 [03:24<00:00, 244.48it/s]
Epochs: 43 | Train Loss:  0.049 | Train Accuracy:  0.692 | Val Loss:  0.101 | Val Accuracy:  0.519
100%|██████████| 49875/49875 [03:24<00:00, 243.51it/s]
Epochs: 44 | Train Loss:  0.049 | Train Accuracy:  0.695 | Val Loss:  0.104 | Val Accuracy:  0.517
100%|██████████| 49875/49875 [03:24<00:00, 243.90it/s]
Epochs: 45 | Train Loss:  0.049 | Train Accuracy:  0.697 | Val Loss:  0.108 | Val Accuracy:  0.510
100%|██████████| 49875/49875 [03:24<00:00, 244.44it/s]
Epochs: 46 | Train Loss:  0.048 | Train Accuracy:  0.700 | Val Loss:  0.104 | Val Accuracy:  0.517
100%|██████████| 49875/49875 [03:23<00:00, 244.88it/s]
```

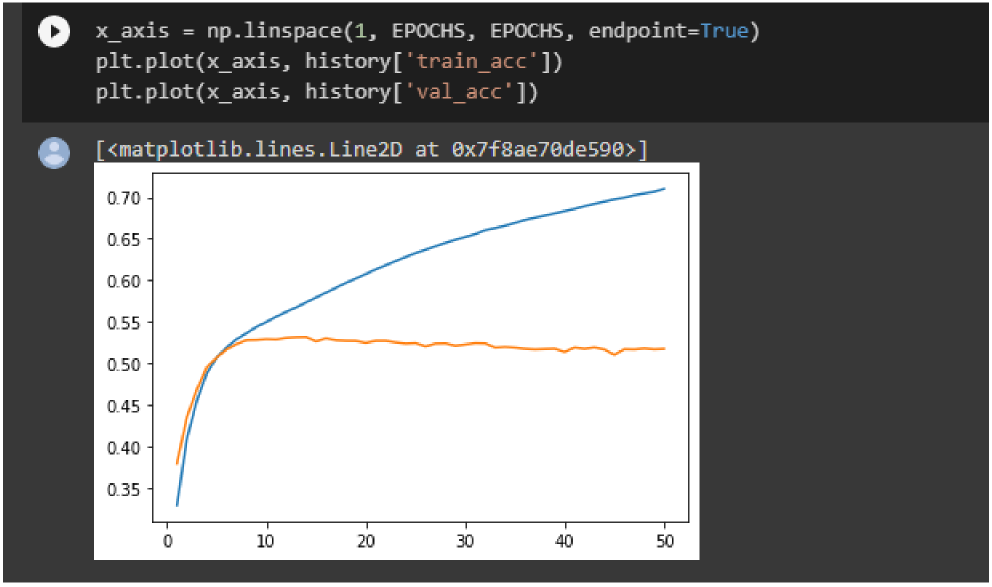


```
Epochs: 42 | Train Loss:  0.050 | Train Accuracy:  0.689 | Val Loss:  0.102 | Val Accuracy:  0.518
100%|██████████| 49875/49875 [03:24<00:00, 244.48it/s]
Epochs: 43 | Train Loss:  0.049 | Train Accuracy:  0.692 | Val Loss:  0.101 | Val Accuracy:  0.519
100%|██████████| 49875/49875 [03:24<00:00, 243.51it/s]
Epochs: 44 | Train Loss:  0.049 | Train Accuracy:  0.695 | Val Loss:  0.104 | Val Accuracy:  0.517
100%|██████████| 49875/49875 [03:24<00:00, 243.90it/s]
Epochs: 45 | Train Loss:  0.049 | Train Accuracy:  0.697 | Val Loss:  0.108 | Val Accuracy:  0.510
100%|██████████| 49875/49875 [03:24<00:00, 244.44it/s]
Epochs: 46 | Train Loss:  0.048 | Train Accuracy:  0.700 | Val Loss:  0.104 | Val Accuracy:  0.517
100%|██████████| 49875/49875 [03:23<00:00, 244.88it/s]
Epochs: 47 | Train Loss:  0.048 | Train Accuracy:  0.702 | Val Loss:  0.106 | Val Accuracy:  0.517
100%|██████████| 49875/49875 [03:24<00:00, 243.75it/s]
Epochs: 48 | Train Loss:  0.047 | Train Accuracy:  0.705 | Val Loss:  0.107 | Val Accuracy:  0.518
100%|██████████| 49875/49875 [03:23<00:00, 245.15it/s]
Epochs: 49 | Train Loss:  0.047 | Train Accuracy:  0.707 | Val Loss:  0.111 | Val Accuracy:  0.517
100%|██████████| 49875/49875 [03:24<00:00, 243.84it/s]
Epochs: 50 | Train Loss:  0.047 | Train Accuracy:  0.710 | Val Loss:  0.110 | Val Accuracy:  0.518
```



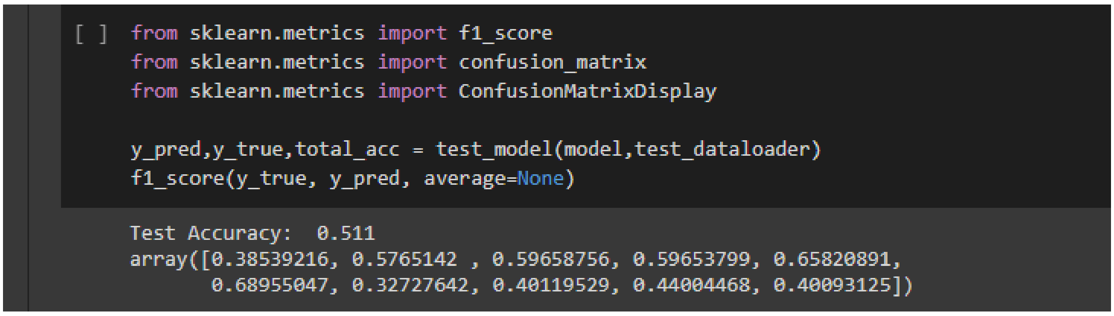
```
plt.plot(x_axis, history['train_loss'])
plt.plot(x_axis, history['val_loss'])
```

```
[<matplotlib.lines.Line2D at 0x7f8ae7169190>]
```



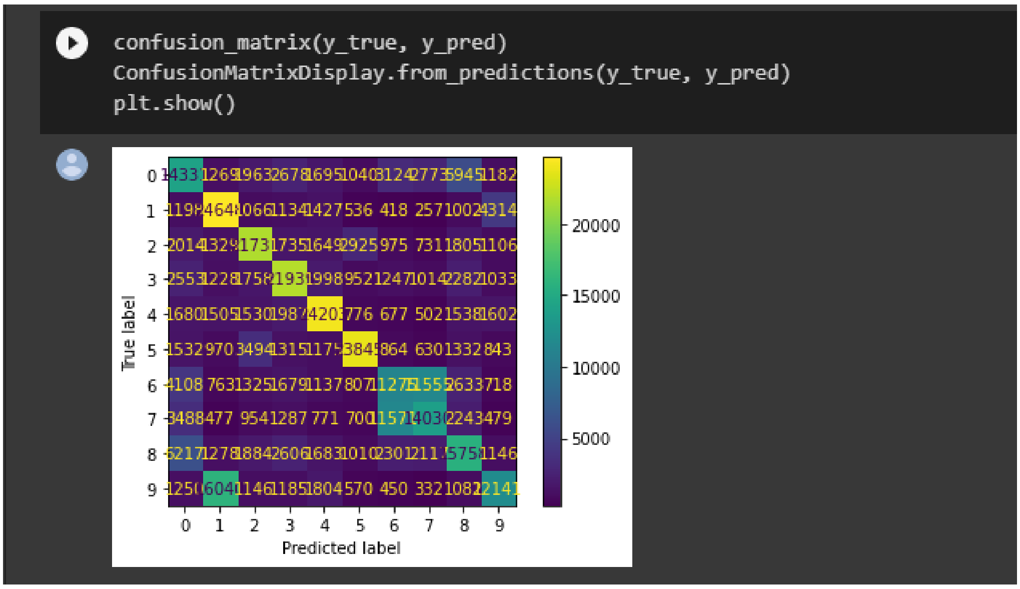
```
x_axis = np.linspace(1, EPOCHS, EPOCHS, endpoint=True)
plt.plot(x_axis, history['train_acc'])
plt.plot(x_axis, history['val_acc'])
```

```
[<matplotlib.lines.Line2D at 0x7f8ae70de590>]
```



```
[ ]  from sklearn.metrics import f1_score
     from sklearn.metrics import confusion_matrix
     from sklearn.metrics import ConfusionMatrixDisplay

     y_pred,y_true,total_acc = test_model(model,test_dataloader)
     f1_score(y_true, y_pred, average=None)
```

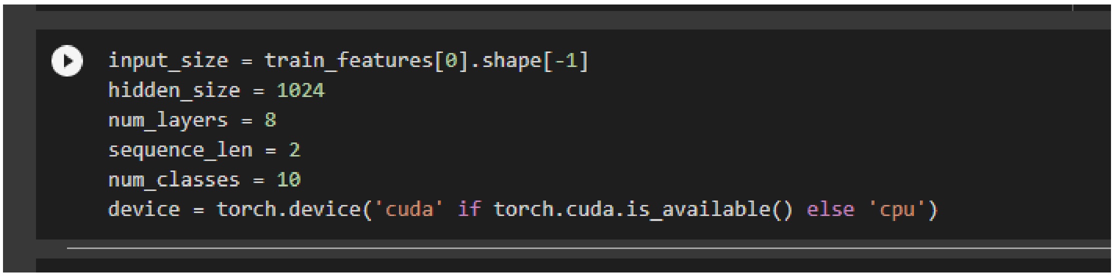
```
Test Accuracy:  0.511
array([0.38539216, 0.5765142 , 0.59658756, 0.59653799, 0.65820891,
       0.68955047, 0.32727642, 0.40119529, 0.44004468, 0.40093125])
```



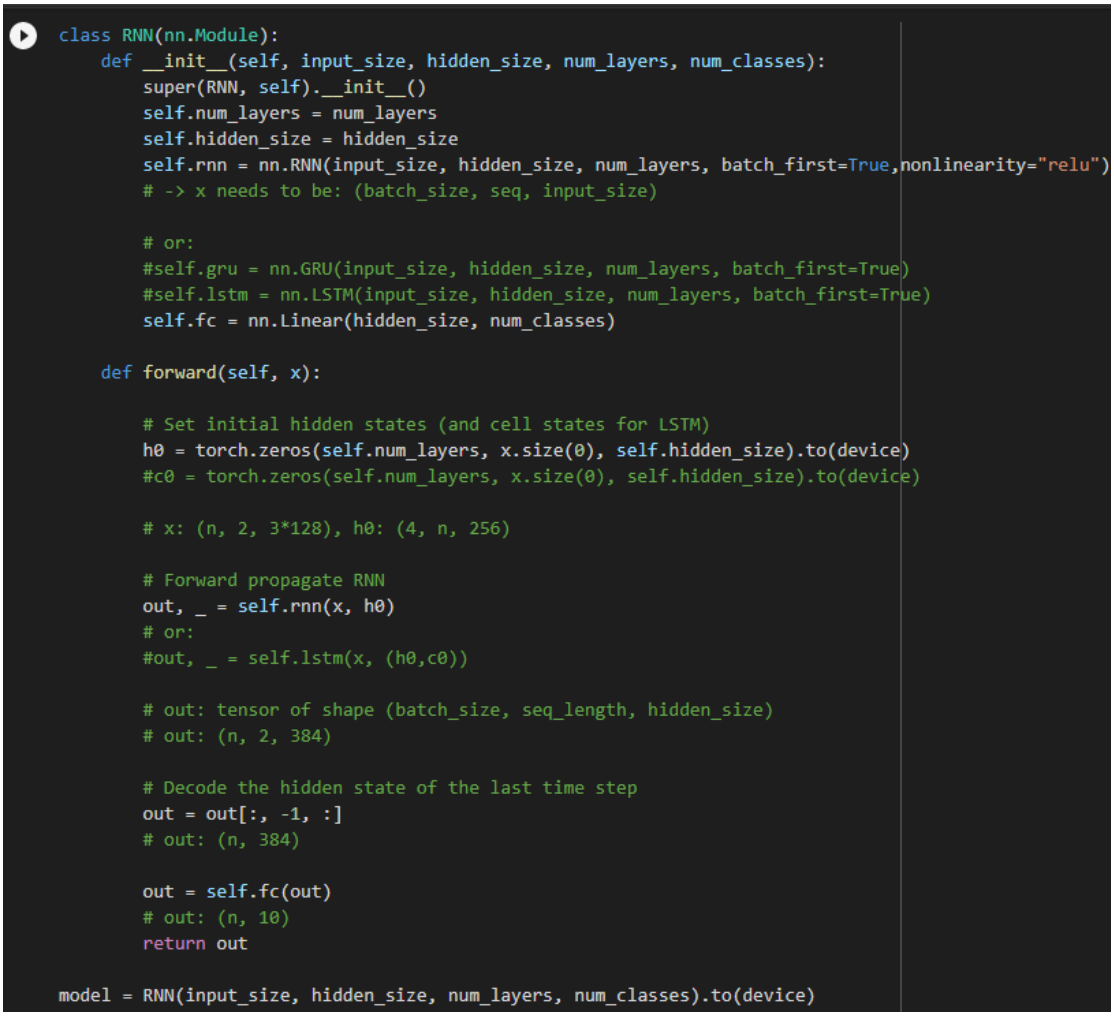
```
confusion_matrix(y_true, y_pred)
ConfusionMatrixDisplay.from_predictions(y_true, y_pred)
plt.show()
```

3- RNN

We have used 8 hidden layers and 1024 neurons each based on several trials

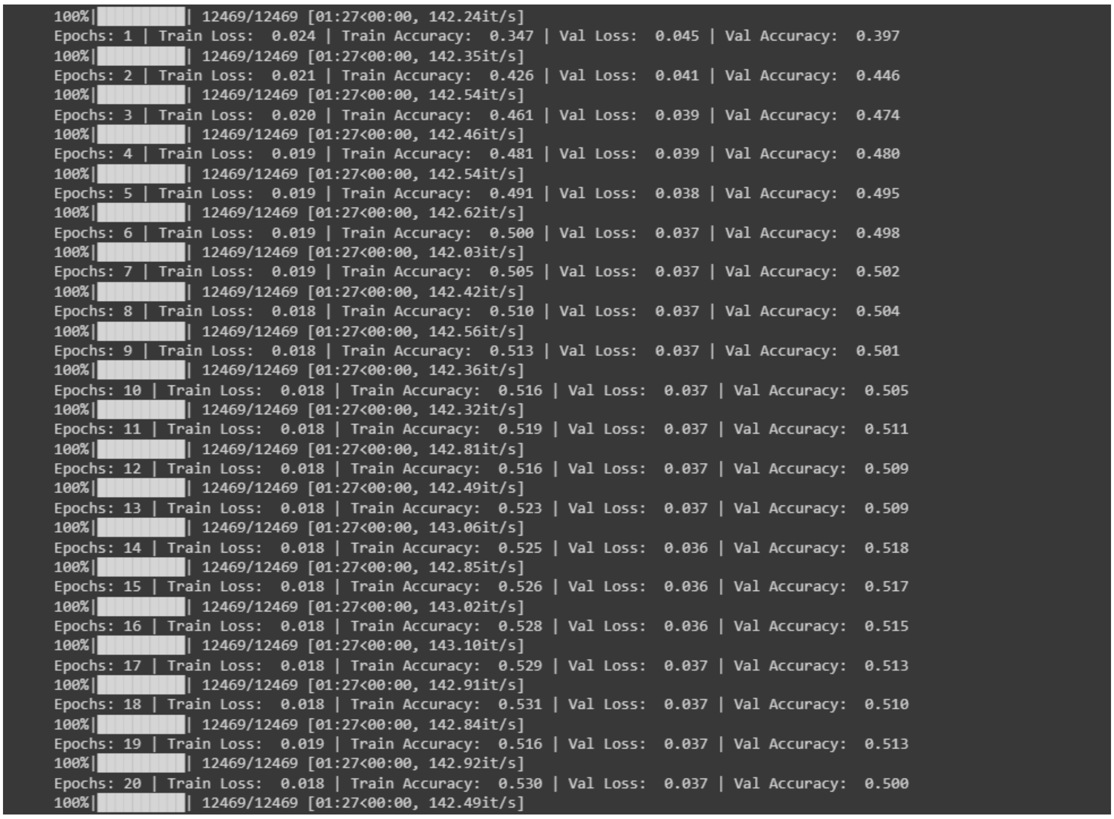


```
input_size = train_features[0].shape[-1]
hidden_size = 1024
num_layers = 8
sequence_len = 2
num_classes = 10
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

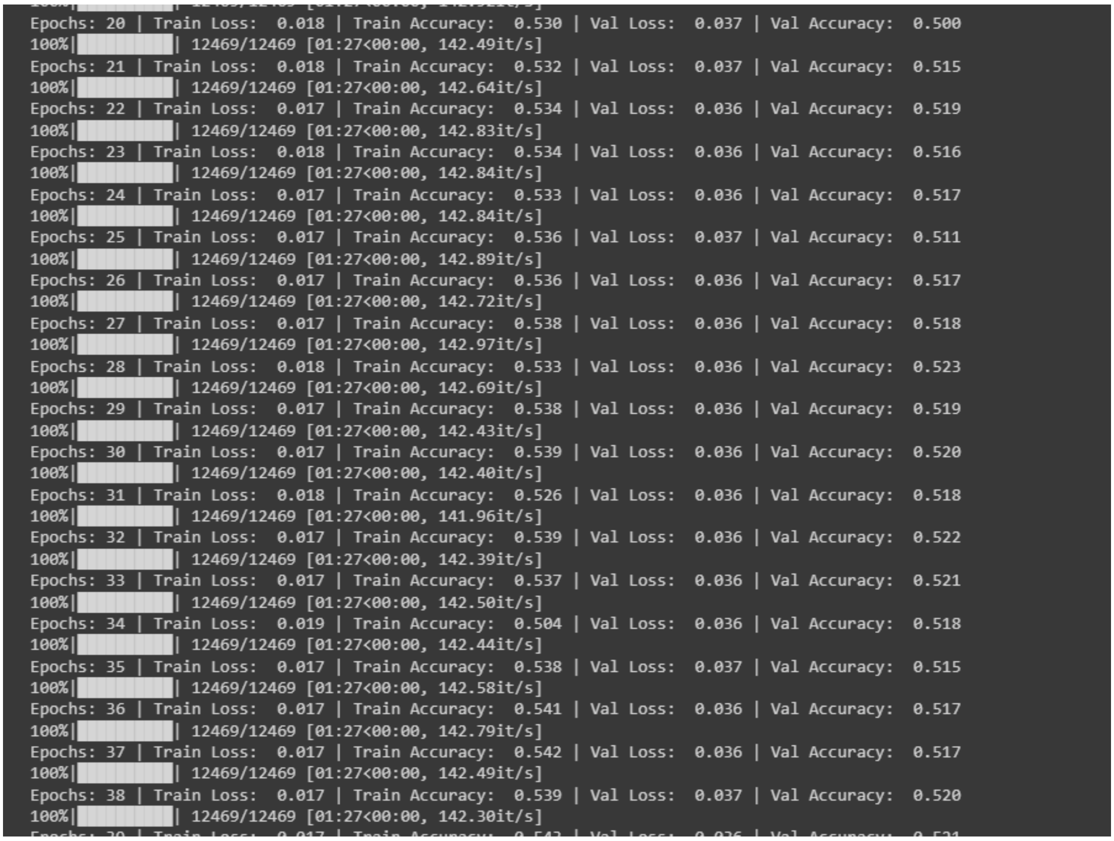


```
class RNN(nn.Module):
    def __init__(self, input_size, hidden_size, num_layers, num_classes):
        super(RNN, self).__init__()
        self.num_layers = num_layers
        self.hidden_size = hidden_size
        self.rnn = nn.RNN(input_size, hidden_size, num_layers, batch_first=True,nonlinearity="relu")
        # -> x needs to be: (batch_size, seq, input_size)

        # or:
        #self.gru = nn.GRU(input_size, hidden_size, num_layers, batch_first=True)
        #self.lstm = nn.LSTM(input_size, hidden_size, num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_size, num_classes)

    def forward(self, x):

        # Set initial hidden states (and cell states for LSTM)
        h0 = torch.zeros(self.num_layers, x.size(0), self.hidden_size).to(device)
        #c0 = torch.zeros(self.num_layers, x.size(0), self.hidden_size).to(device)

        # x: (n, 2, 3*128), h0: (4, n, 256)

        # Forward propagate RNN
        out, _ = self.rnn(x, h0)
        # or:
        #out, _ = self.lstm(x, (h0,c0))

        # out: tensor of shape (batch_size, seq_length, hidden_size)
        # out: (n, 2, 384)

        # Decode the hidden state of the last time step
        out = out[:, -1, :]
        # out: (n, 384)

        out = self.fc(out)
        # out: (n, 10)
        return out

model = RNN(input_size, hidden_size, num_layers, num_classes).to(device)
```

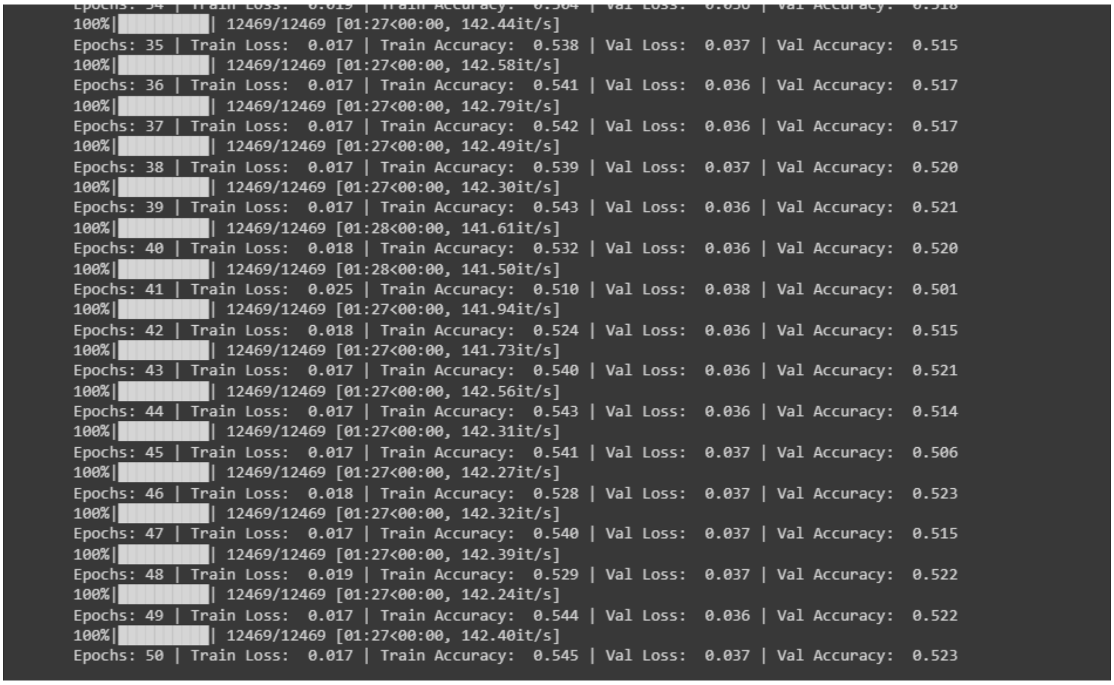
50 epochs with same learning rate



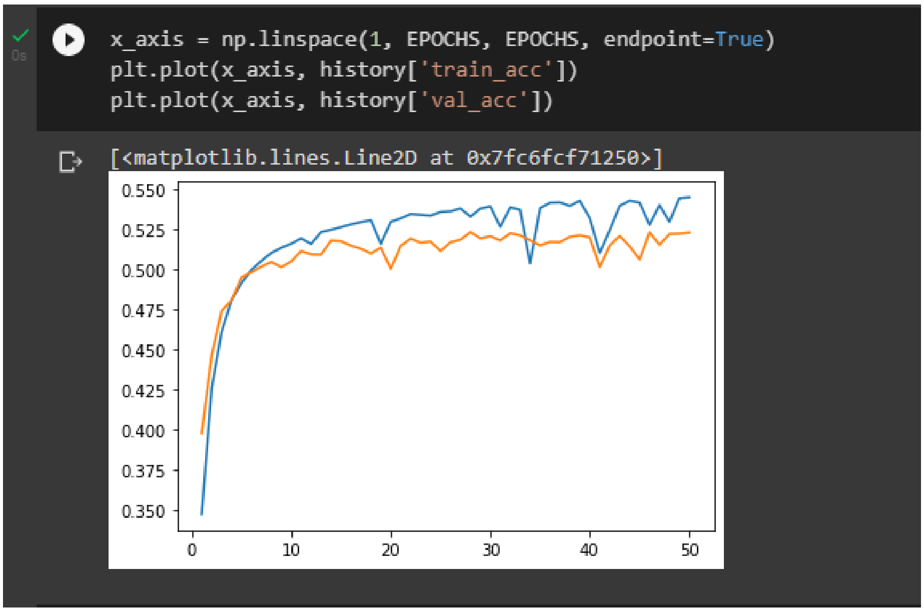
```
    100%|██████████| 12469/12469 [01:27<00:00, 142.24it/s]
    Epochs: 1 | Train Loss:  0.024 | Train Accuracy:  0.347 | Val Loss:  0.045 | Val Accuracy:  0.397
    100%|██████████| 12469/12469 [01:27<00:00, 142.35it/s]
    Epochs: 2 | Train Loss:  0.021 | Train Accuracy:  0.426 | Val Loss:  0.041 | Val Accuracy:  0.446
    100%|██████████| 12469/12469 [01:27<00:00, 142.54it/s]
    Epochs: 3 | Train Loss:  0.020 | Train Accuracy:  0.461 | Val Loss:  0.039 | Val Accuracy:  0.474
    100%|██████████| 12469/12469 [01:27<00:00, 142.46it/s]
    Epochs: 4 | Train Loss:  0.019 | Train Accuracy:  0.481 | Val Loss:  0.039 | Val Accuracy:  0.480
    100%|██████████| 12469/12469 [01:27<00:00, 142.54it/s]
    Epochs: 5 | Train Loss:  0.019 | Train Accuracy:  0.491 | Val Loss:  0.038 | Val Accuracy:  0.495
    100%|██████████| 12469/12469 [01:27<00:00, 142.62it/s]
    Epochs: 6 | Train Loss:  0.019 | Train Accuracy:  0.500 | Val Loss:  0.037 | Val Accuracy:  0.498
    100%|██████████| 12469/12469 [01:27<00:00, 142.03it/s]
    Epochs: 7 | Train Loss:  0.019 | Train Accuracy:  0.505 | Val Loss:  0.037 | Val Accuracy:  0.502
    100%|██████████| 12469/12469 [01:27<00:00, 142.42it/s]
    Epochs: 8 | Train Loss:  0.018 | Train Accuracy:  0.510 | Val Loss:  0.037 | Val Accuracy:  0.504
    100%|██████████| 12469/12469 [01:27<00:00, 142.56it/s]
    Epochs: 9 | Train Loss:  0.018 | Train Accuracy:  0.513 | Val Loss:  0.037 | Val Accuracy:  0.501
    100%|██████████| 12469/12469 [01:27<00:00, 142.36it/s]
    Epochs: 10 | Train Loss:  0.018 | Train Accuracy:  0.516 | Val Loss:  0.037 | Val Accuracy:  0.505
    100%|██████████| 12469/12469 [01:27<00:00, 142.32it/s]
    Epochs: 11 | Train Loss:  0.018 | Train Accuracy:  0.519 | Val Loss:  0.037 | Val Accuracy:  0.511
    100%|██████████| 12469/12469 [01:27<00:00, 142.81it/s]
    Epochs: 12 | Train Loss:  0.018 | Train Accuracy:  0.516 | Val Loss:  0.037 | Val Accuracy:  0.509
    100%|██████████| 12469/12469 [01:27<00:00, 142.49it/s]
    Epochs: 13 | Train Loss:  0.018 | Train Accuracy:  0.523 | Val Loss:  0.037 | Val Accuracy:  0.509
    100%|██████████| 12469/12469 [01:27<00:00, 143.06it/s]
    Epochs: 14 | Train Loss:  0.018 | Train Accuracy:  0.525 | Val Loss:  0.036 | Val Accuracy:  0.518
    100%|██████████| 12469/12469 [01:27<00:00, 142.85it/s]
    Epochs: 15 | Train Loss:  0.018 | Train Accuracy:  0.526 | Val Loss:  0.036 | Val Accuracy:  0.517
    100%|██████████| 12469/12469 [01:27<00:00, 143.02it/s]
    Epochs: 16 | Train Loss:  0.018 | Train Accuracy:  0.528 | Val Loss:  0.036 | Val Accuracy:  0.515
    100%|██████████| 12469/12469 [01:27<00:00, 143.10it/s]
    Epochs: 17 | Train Loss:  0.018 | Train Accuracy:  0.529 | Val Loss:  0.037 | Val Accuracy:  0.513
    100%|██████████| 12469/12469 [01:27<00:00, 142.91it/s]
    Epochs: 18 | Train Loss:  0.018 | Train Accuracy:  0.531 | Val Loss:  0.037 | Val Accuracy:  0.510
    100%|██████████| 12469/12469 [01:27<00:00, 142.84it/s]
    Epochs: 19 | Train Loss:  0.019 | Train Accuracy:  0.516 | Val Loss:  0.037 | Val Accuracy:  0.513
    100%|██████████| 12469/12469 [01:27<00:00, 142.92it/s]
    Epochs: 20 | Train Loss:  0.018 | Train Accuracy:  0.530 | Val Loss:  0.037 | Val Accuracy:  0.500
    100%|██████████| 12469/12469 [01:27<00:00, 142.49it/s]
```



```
  100%|██████████| 12469/12469 [01:27<00:00, 142.92it/s]
  Epochs: 20 | Train Loss:  0.018 | Train Accuracy:  0.530 | Val Loss:  0.037 | Val Accuracy:  0.500
  100%|██████████| 12469/12469 [01:27<00:00, 142.49it/s]
  Epochs: 21 | Train Loss:  0.018 | Train Accuracy:  0.532 | Val Loss:  0.037 | Val Accuracy:  0.515
  100%|██████████| 12469/12469 [01:27<00:00, 142.64it/s]
  Epochs: 22 | Train Loss:  0.017 | Train Accuracy:  0.534 | Val Loss:  0.036 | Val Accuracy:  0.519
  100%|██████████| 12469/12469 [01:27<00:00, 142.83it/s]
  Epochs: 23 | Train Loss:  0.018 | Train Accuracy:  0.534 | Val Loss:  0.036 | Val Accuracy:  0.516
  100%|██████████| 12469/12469 [01:27<00:00, 142.84it/s]
  Epochs: 24 | Train Loss:  0.017 | Train Accuracy:  0.533 | Val Loss:  0.036 | Val Accuracy:  0.517
  100%|██████████| 12469/12469 [01:27<00:00, 142.84it/s]
  Epochs: 25 | Train Loss:  0.017 | Train Accuracy:  0.536 | Val Loss:  0.037 | Val Accuracy:  0.511
  100%|██████████| 12469/12469 [01:27<00:00, 142.89it/s]
  Epochs: 26 | Train Loss:  0.017 | Train Accuracy:  0.536 | Val Loss:  0.036 | Val Accuracy:  0.517
  100%|██████████| 12469/12469 [01:27<00:00, 142.72it/s]
  Epochs: 27 | Train Loss:  0.017 | Train Accuracy:  0.538 | Val Loss:  0.036 | Val Accuracy:  0.518
  100%|██████████| 12469/12469 [01:27<00:00, 142.97it/s]
  Epochs: 28 | Train Loss:  0.018 | Train Accuracy:  0.533 | Val Loss:  0.036 | Val Accuracy:  0.523
  100%|██████████| 12469/12469 [01:27<00:00, 142.69it/s]
  Epochs: 29 | Train Loss:  0.017 | Train Accuracy:  0.538 | Val Loss:  0.036 | Val Accuracy:  0.519
  100%|██████████| 12469/12469 [01:27<00:00, 142.43it/s]
  Epochs: 30 | Train Loss:  0.017 | Train Accuracy:  0.539 | Val Loss:  0.036 | Val Accuracy:  0.520
  100%|██████████| 12469/12469 [01:27<00:00, 142.40it/s]
  Epochs: 31 | Train Loss:  0.018 | Train Accuracy:  0.526 | Val Loss:  0.036 | Val Accuracy:  0.518
  100%|██████████| 12469/12469 [01:27<00:00, 141.96it/s]
  Epochs: 32 | Train Loss:  0.017 | Train Accuracy:  0.539 | Val Loss:  0.036 | Val Accuracy:  0.522
  100%|██████████| 12469/12469 [01:27<00:00, 142.39it/s]
  Epochs: 33 | Train Loss:  0.017 | Train Accuracy:  0.537 | Val Loss:  0.036 | Val Accuracy:  0.521
  100%|██████████| 12469/12469 [01:27<00:00, 142.50it/s]
  Epochs: 34 | Train Loss:  0.019 | Train Accuracy:  0.504 | Val Loss:  0.036 | Val Accuracy:  0.518
  100%|██████████| 12469/12469 [01:27<00:00, 142.44it/s]
  Epochs: 35 | Train Loss:  0.017 | Train Accuracy:  0.538 | Val Loss:  0.037 | Val Accuracy:  0.515
  100%|██████████| 12469/12469 [01:27<00:00, 142.58it/s]
  Epochs: 36 | Train Loss:  0.017 | Train Accuracy:  0.541 | Val Loss:  0.036 | Val Accuracy:  0.517
  100%|██████████| 12469/12469 [01:27<00:00, 142.79it/s]
  Epochs: 37 | Train Loss:  0.017 | Train Accuracy:  0.542 | Val Loss:  0.036 | Val Accuracy:  0.517
  100%|██████████| 12469/12469 [01:27<00:00, 142.49it/s]
  Epochs: 38 | Train Loss:  0.017 | Train Accuracy:  0.539 | Val Loss:  0.037 | Val Accuracy:  0.520
  100%|██████████| 12469/12469 [01:27<00:00, 142.30it/s]
  Epochs: 39 | Train Loss:  0.017 | Train Accuracy:  0.542 | Val Loss:  0.036 | Val Accuracy:  0.521
```

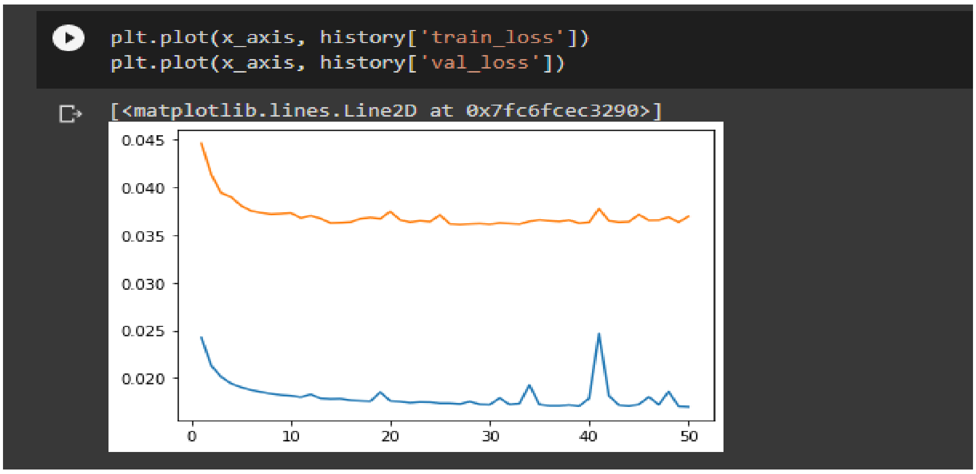


```
Epochs: 34 | Train Loss:  0.019 | Train Accuracy:  0.504 | Val Loss:  0.036 | Val Accuracy:  0.518
100%|██████████| 12469/12469 [01:27<00:00, 142.44it/s]
Epochs: 35 | Train Loss:  0.017 | Train Accuracy:  0.538 | Val Loss:  0.037 | Val Accuracy:  0.515
100%|██████████| 12469/12469 [01:27<00:00, 142.58it/s]
Epochs: 36 | Train Loss:  0.017 | Train Accuracy:  0.541 | Val Loss:  0.036 | Val Accuracy:  0.517
100%|██████████| 12469/12469 [01:27<00:00, 142.79it/s]
Epochs: 37 | Train Loss:  0.017 | Train Accuracy:  0.542 | Val Loss:  0.036 | Val Accuracy:  0.517
100%|██████████| 12469/12469 [01:27<00:00, 142.49it/s]
Epochs: 38 | Train Loss:  0.017 | Train Accuracy:  0.539 | Val Loss:  0.037 | Val Accuracy:  0.520
100%|██████████| 12469/12469 [01:27<00:00, 142.30it/s]
Epochs: 39 | Train Loss:  0.017 | Train Accuracy:  0.543 | Val Loss:  0.036 | Val Accuracy:  0.521
100%|██████████| 12469/12469 [01:28<00:00, 141.61it/s]
Epochs: 40 | Train Loss:  0.018 | Train Accuracy:  0.532 | Val Loss:  0.036 | Val Accuracy:  0.520
100%|██████████| 12469/12469 [01:28<00:00, 141.50it/s]
Epochs: 41 | Train Loss:  0.025 | Train Accuracy:  0.510 | Val Loss:  0.038 | Val Accuracy:  0.501
100%|██████████| 12469/12469 [01:27<00:00, 141.94it/s]
Epochs: 42 | Train Loss:  0.018 | Train Accuracy:  0.524 | Val Loss:  0.036 | Val Accuracy:  0.515
100%|██████████| 12469/12469 [01:27<00:00, 141.73it/s]
Epochs: 43 | Train Loss:  0.017 | Train Accuracy:  0.540 | Val Loss:  0.036 | Val Accuracy:  0.521
100%|██████████| 12469/12469 [01:27<00:00, 142.56it/s]
Epochs: 44 | Train Loss:  0.017 | Train Accuracy:  0.543 | Val Loss:  0.036 | Val Accuracy:  0.514
100%|██████████| 12469/12469 [01:27<00:00, 142.31it/s]
Epochs: 45 | Train Loss:  0.017 | Train Accuracy:  0.541 | Val Loss:  0.037 | Val Accuracy:  0.506
100%|██████████| 12469/12469 [01:27<00:00, 142.27it/s]
Epochs: 46 | Train Loss:  0.018 | Train Accuracy:  0.528 | Val Loss:  0.037 | Val Accuracy:  0.523
100%|██████████| 12469/12469 [01:27<00:00, 142.32it/s]
Epochs: 47 | Train Loss:  0.017 | Train Accuracy:  0.540 | Val Loss:  0.037 | Val Accuracy:  0.515
100%|██████████| 12469/12469 [01:27<00:00, 142.39it/s]
Epochs: 48 | Train Loss:  0.019 | Train Accuracy:  0.529 | Val Loss:  0.037 | Val Accuracy:  0.522
100%|██████████| 12469/12469 [01:27<00:00, 142.24it/s]
Epochs: 49 | Train Loss:  0.017 | Train Accuracy:  0.544 | Val Loss:  0.036 | Val Accuracy:  0.522
100%|██████████| 12469/12469 [01:27<00:00, 142.40it/s]
Epochs: 50 | Train Loss:  0.017 | Train Accuracy:  0.545 | Val Loss:  0.037 | Val Accuracy:  0.523
```



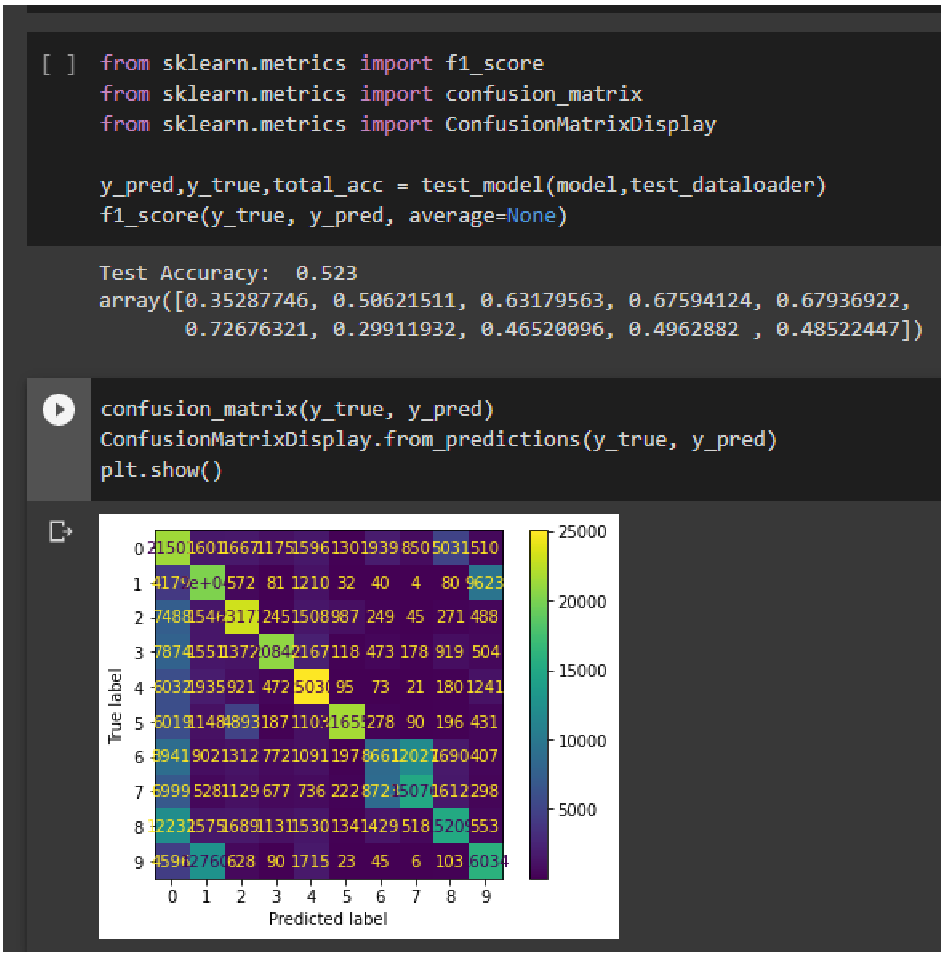
```
x_axis = np.linspace(1, EPOCHS, EPOCHS, endpoint=True)
plt.plot(x_axis, history['train_acc'])
plt.plot(x_axis, history['val_acc'])
```

```
[<matplotlib.lines.Line2D at 0x7fc6fcf71250>]
```



```
plt.plot(x_axis, history['train_loss'])
plt.plot(x_axis, history['val_loss'])
```

```
[<matplotlib.lines.Line2D at 0x7fc6fcec3290>]
```



```
[ ] from sklearn.metrics import f1_score
    from sklearn.metrics import confusion_matrix
    from sklearn.metrics import ConfusionMatrixDisplay

    y_pred,y_true,total_acc = test_model(model,test_dataloader)
    f1_score(y_true, y_pred, average=None)
```

```
Test Accuracy:  0.523
array([0.35287746, 0.50621511, 0.63179563, 0.67594124, 0.67936922,
       0.72676321, 0.29911932, 0.46520096, 0.4962882 , 0.48522447])
```

```
confusion_matrix(y_true, y_pred)
ConfusionMatrixDisplay.from_predictions(y_true, y_pred)
plt.show()
```

Results

Model	Accuracy	Most Confusing Class
CNN	46.7%	QAM-16
LTSM	51.1%	BSK-8
RNN	52.3%	QAM-16