|  |  |
| --- | --- |
| **Ex no : 3** | **Hyper Parameter tuning in Multilayer Perceptron** |
| **Date :** |

**Aim**

To implement multi-layer network for MNIST dataset with different hyper parameters and visualizing the comparison

**Basic Theory of Hyper Parameters.**

* + Hyperparameters are parameters whose values control the learning process and determine the values of model parameters that a learning algorithm ends up learning
  + The prefix ‘hyper\_’ suggests that they are ‘top-level’ parameters that control the learning process and the model parameters that result from it.
  + The hyperparameters that were used during training are not part of this model
  + Parameters considered as a hyper parameters in this program are

1. Choosing activation function
2. Choosing no of epochs
3. Choosing validation split
4. Choosing the optimizer
5. Choosing learning rate

**PROCEDURE AND CODE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Scenario 1** | **Scenario 2** | **Scenario 3** | **Scenario 4** |
| Activation function (Only for Hidden Layers) | Sigmoid | Relu | Tanh | Sigmoid |
| No of Epochs | 24 | 25 | 21 | 23 |
| Validation Split | 0.3 | 0.2 | 0.1 | 0.2 |
| Optimizer | Adagrad | Adam | Adadelta | Adam |
| Learning Rate | 0.001 | 0.001 | 0.001 | 0.001 |

**Code for Scenario 1**

import tensorflow

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense,Flatten

X\_train,y\_train),(X\_test,y\_test)= keras.datasets.mnist.load\_data()

X\_test.shape

X\_train.shape

y\_train.shape

y\_test.shape

y\_train

import matplotlib.pyplot as plt

plt.imshow(X\_train[2])

X\_train = X\_train/255

X\_test = X\_test/255

X\_train[0]

model = Sequential()

model.add(Flatten(input\_shape=(28,28)))

model.add(Dense(128,activation='Sigmoid'))

model.add(Dense(32,activation=' Sigmoid '))

model.add(Dense(10,activation='softmax'))

model.summary()

model.compile(loss='sparse\_categorical\_crossentropy', optimizer=' Adagrad',metrics=['accuracy'])

history\_scenario1=model.fit(X\_train,y\_train,epochs=24, validation\_split=0.3)

y\_prob = model.predict(X\_test)

y\_pred = y\_prob.argmax(axis=1)

from sklearn.metrics import accuracy\_score

accuracy\_score(y\_test,y\_pred)

plt.plot(history\_scenario1.history\_scenario1['loss'])

plt.plot(history\_scenario1.history\_scenario1['val\_loss'])

plt.plot(history\_scenario1.history\_scenario1['accuracy'])

plt.plot(history\_scenario1.history\_scenario1['val\_accuracy'])

**Code for Scenario 2**

import tensorflow

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense,Flatten

X\_train,y\_train),(X\_test,y\_test)= keras.datasets.mnist.load\_data()

X\_test.shape

X\_train.shape

y\_train.shape

y\_test.shape

y\_train

import matplotlib.pyplot as plt

plt.imshow(X\_train[2])

X\_train = X\_train/255

X\_test = X\_test/255

X\_train[0]

model = Sequential()

model.add(Flatten(input\_shape=(28,28)))

model.add(Dense(128,activation='Relu'))

model.add(Dense(32,activation='Relu'))

model.add(Dense(10,activation='softmax'))

model.summary()

model.compile(loss='sparse\_categorical\_crossentropy', optimizer='Adam',metrics=['accuracy'])

history\_scenario1=model.fit(X\_train,y\_train,epochs=25, validation\_split=0.2)

y\_prob = model.predict(X\_test)

y\_pred = y\_prob.argmax(axis=1)

from sklearn.metrics import accuracy\_score

accuracy\_score(y\_test,y\_pred)

plt.plot(history\_scenario1.history\_scenario1['loss'])

plt.plot(history\_scenario1.history\_scenario1['val\_loss'])

plt.plot(history\_scenario1.history\_scenario1['accuracy'])

plt.plot(history\_scenario1.history\_scenario1['val\_accuracy'])

**Code for Scenario 3**

import tensorflow

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense,Flatten

X\_train,y\_train),(X\_test,y\_test)= keras.datasets.mnist.load\_data()

X\_test.shape

X\_train.shape

y\_train.shape

y\_test.shape

y\_train

import matplotlib.pyplot as plt

plt.imshow(X\_train[2])

X\_train = X\_train/255

X\_test = X\_test/255

X\_train[0]

model = Sequential()

model.add(Flatten(input\_shape=(28,28)))

model.add(Dense(128,activation='Tanh'))

model.add(Dense(32,activation='Tanh'))

model.add(Dense(10,activation='softmax'))

model.summary()

model.compile(loss='sparse\_categorical\_crossentropy', optimizer='Adadelta',metrics=['accuracy'])

history\_scenario1=model.fit(X\_train,y\_train,epochs=21, validation\_split=0.1)

y\_prob = model.predict(X\_test)

y\_pred = y\_prob.argmax(axis=1)

from sklearn.metrics import accuracy\_score

accuracy\_score(y\_test,y\_pred)

plt.plot(history\_scenario1.history\_scenario1['loss'])

plt.plot(history\_scenario1.history\_scenario1['val\_loss'])

plt.plot(history\_scenario1.history\_scenario1['accuracy'])

plt.plot(history\_scenario1.history\_scenario1['val\_accuracy'])

**Code for Scenario 4**

import tensorflow

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense,Flatten

X\_train,y\_train),(X\_test,y\_test)= keras.datasets.mnist.load\_data()

X\_test.shape

X\_train.shape

y\_train.shape

y\_test.shape

y\_train

import matplotlib.pyplot as plt

plt.imshow(X\_train[2])

X\_train = X\_train/255

X\_test = X\_test/255

X\_train[0]

model = Sequential()

model.add(Flatten(input\_shape=(28,28)))

model.add(Dense(128,activation='Sigmoid'))

model.add(Dense(32,activation=' Sigmoid '))

model.add(Dense(10,activation='softmax'))

model.summary()

model.compile(loss='sparse\_categorical\_crossentropy', optimizer=' Adam',metrics=['accuracy'])

history\_scenario1=model.fit(X\_train,y\_train,epochs=23, validation\_split=0.2)

y\_prob = model.predict(X\_test)

y\_pred = y\_prob.argmax(axis=1)

from sklearn.metrics import accuracy\_score

accuracy\_score(y\_test,y\_pred)

plt.plot(history\_scenario1.history\_scenario1['loss'])

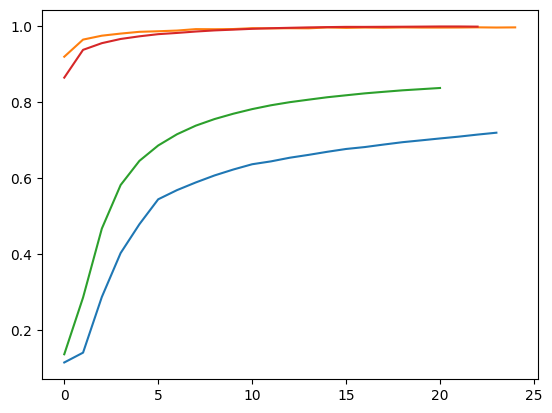
plt.plot(history\_scenario1.history\_scenario1['val\_loss'])

plt.plot(history\_scenario1.history\_scenario1['accuracy'])

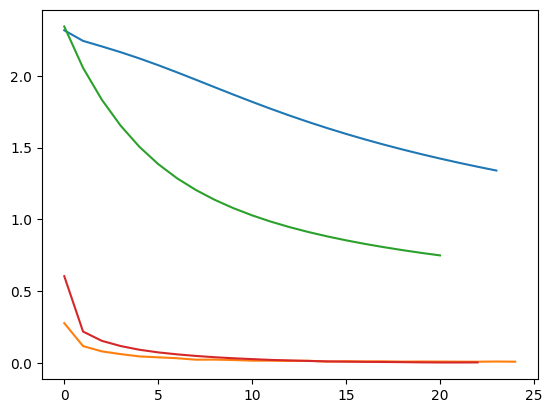
plt.plot(history\_scenario1.history\_scenario1['val\_accuracy'])

**OUTPUT**

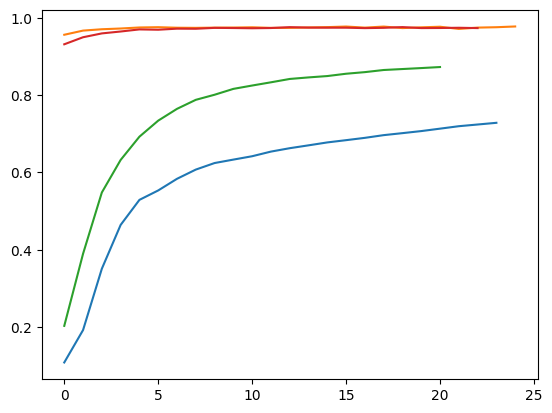
Training Accuracy

****

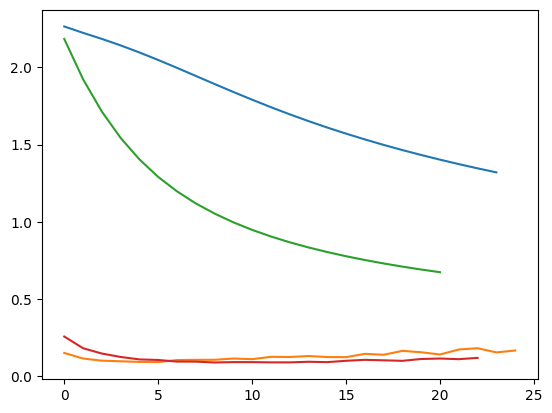
Training Loss

****

Validation Accuracy

****

Validation Loss

****

**RESULT**

Thus implementation of multi-layer network for MNIST dataset with different hyper parameters and visualizing the comparison has been carried out successfully.