CSE5ML ASSIGNMENT 1

In submitting your work, you are consenting that it may be copied and transmitted by the university for the detection of plagiarism.

“This is my own work. I have not copied any of it from anyone else.”

Name: Maninderpreet Singh Puri

Student number: 20494381

**Python Code**

NN 1

import tensorflow as tf

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import Dropout

from keras.layers import Flatten

from keras.constraints import maxnorm

from keras.optimizers import SGD

from keras.layers.convolutional import Conv2D

from keras.layers.convolutional import AveragePooling2D

from keras.utils import np\_utils

# load data

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

# normalize inputs from 0-255 to 0.0-1.0

x\_train = x\_train.astype('float32')

x\_test = x\_test.astype('float32')

x\_train = x\_train.reshape(x\_train.shape[0], 28, 28, 1).astype('float32')

x\_test = x\_test.reshape(x\_test.shape[0], 28, 28, 1).astype('float32')

x\_train = x\_train / 255.0

x\_test = x\_test / 255.0

# Encode the outputs

y\_train = np\_utils.to\_categorical(y\_train) #Converts a class vector (integers) to binary class matrix.

y\_test = np\_utils.to\_categorical(y\_test)

num\_classes = y\_test.shape[1]

# Build the model

model = Sequential()

model.add(Conv2D(32, (3, 3), input\_shape=(28, 28, 1), activation='relu'))

model.add(Conv2D(32, (3, 3), activation='relu'))

model.add(AveragePooling2D())

model.add(Flatten())

model.add(Dense(512, activation='tanh'))

model.add(Dense(1024, activation='sigmoid'))

model.add(Dropout(0.2))

model.add(Dense(num\_classes, activation='softmax'))

# Compile model

epochs = 5

lrate = 0.0012

decay = lrate/epochs

sgd = SGD(lr=lrate, momentum=0.7, decay=decay) #Stochastic gradient descent optimizer

model.compile(loss='categorical\_crossentropy', optimizer=sgd, metrics=['accuracy'])

model.summary()

# Fit the model

model.fit(x\_train, y\_train, validation\_data=(x\_test, y\_test), epochs=epochs, batch\_size=60, verbose=1)

# Final evaluation of the model

scores = model.evaluate(x\_test, y\_test, verbose=0)

print("Accuracy: %.2f%%" % (scores[1]\*100))

NN 2

import tensorflow as tf

import matplotlib.pyplot as plt

import numpy as np

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import Dropout

from keras.layers import Flatten

from keras.constraints import maxnorm

from keras.optimizers import SGD

from keras.layers.convolutional import Conv2D

from keras.layers.convolutional import AveragePooling2D

from keras.utils import np\_utils

# load data

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

# normalize inputs from 0-255 to 0.0-1.0

x\_train = x\_train.astype('float32')

x\_test = x\_test.astype('float32')

x\_train = x\_train.reshape(x\_train.shape[0], 28, 28, 1).astype('float32')

x\_test = x\_test.reshape(x\_test.shape[0], 28, 28, 1).astype('float32')

x\_train = x\_train / 255.0

x\_test = x\_test / 255.0

# Encode the outputs

y\_train = np\_utils.to\_categorical(y\_train) #Converts a class vector (integers) to binary class matrix.

y\_test = np\_utils.to\_categorical(y\_test)

num\_classes = y\_test.shape[1]

# Build the model

model = Sequential()

model.add(Conv2D(32, (3, 3), input\_shape=(28, 28, 1), activation='relu'))

model.add(Conv2D(32, (3, 3), activation='relu'))

model.add(AveragePooling2D())

model.add(Flatten())

model.add(Dense(512, activation='relu'))

model.add(Dense(1024, activation='tanh'))

model.add(Dense(512, activation='sigmoid'))

model.add(Dropout(0.2))

model.add(Dense(num\_classes, activation='softmax'))

# Compile model

epochs = 5

lrate = 0.0012

decay = lrate/epochs

sgd = SGD(lr=lrate, momentum=0.7, decay=decay) #Stochastic gradient descent optimizer

model.compile(loss='categorical\_crossentropy', optimizer=sgd, metrics=['accuracy'])

model.summary()

# Fit the model

model.fit(x\_train, y\_train, validation\_data=(x\_test, y\_test), epochs=epochs, batch\_size=60, verbose=1)

# Final evaluation of the model

scores = model.evaluate(x\_test, y\_test, verbose=0)

print("Accuracy: %.2f%%" % (scores[1]\*100))

NN 3

import tensorflow as tf

import matplotlib.pyplot as plt

import numpy as np

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import Dropout

from keras.layers import Flatten

from keras.constraints import maxnorm

from keras.optimizers import SGD

from keras.layers.convolutional import Conv2D

from keras.layers.convolutional import MaxPooling2D

from keras.utils import np\_utils

# load data

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

# normalize inputs from 0-255 to 0.0-1.0

x\_train = x\_train.astype('float32')

x\_test = x\_test.astype('float32')

x\_train = x\_train.reshape(x\_train.shape[0], 28, 28, 1).astype('float32')

x\_test = x\_test.reshape(x\_test.shape[0], 28, 28, 1).astype('float32')

x\_train = x\_train / 255.0

x\_test = x\_test / 255.0

# Encode the outputs

y\_train = np\_utils.to\_categorical(y\_train) #Converts a class vector (integers) to binary class matrix.

y\_test = np\_utils.to\_categorical(y\_test)

num\_classes = y\_test.shape[1]

# Build the model

model = Sequential()

model.add(Conv2D(32, (3, 3), input\_shape=(28, 28, 1), activation='relu'))

model.add(Conv2D(32, (3, 3), activation='relu'))

model.add(MaxPooling2D())

model.add(Flatten())

model.add(Dense(512, activation='relu'))

model.add(Dropout(0.2))

model.add(Dense(num\_classes, activation='softmax'))

# Compile model

epochs = 5

lrate = 0.0012

decay = lrate/epochs

sgd = SGD(lr=lrate, momentum=0.7, decay=decay) #Stochastic gradient descent optimizer

model.compile(loss='categorical\_crossentropy', optimizer=sgd, metrics=['accuracy'])

model.summary()

# Fit the model

model.fit(x\_train, y\_train, validation\_data=(x\_test, y\_test), epochs=epochs, batch\_size=60, verbose=1)

# Final evaluation of the model

scores = model.evaluate(x\_test, y\_test, verbose=0)

print("Accuracy: %.2f%%" % (scores[1]\*100))