Name: Gaurav Kedia

ROll No: 39

Batch: E-2

Aim: Write a program to implement feed forward neural network, alexnet, lnet5 and VGG16 on MNIST Dataset. Compare the result in terms of time and accuracy

```
import pandas as pd
import numpy as np
```

Autoencoder

```
from tensorflow import keras
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.datasets import mnist
import numpy as np

(x_train, _), (x_test, _) = mnist.load_data()
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.
x_train = np.reshape(x_train, (len(x_train), np.prod(x_train.shape[1:])))
x_test = np.reshape(x_test, (len(x_test), np.prod(x_test.shape[1:])))

input_img = Input(shape=(784,))
encoded1 = Dense(128, activation='relu')(input_img)
encoded2 = Dense(64, activation='relu')(encoded1)
decoded1 = Dense(128, activation='relu')(encoded2)
decoded2 = Dense(784, activation='sigmoid')(decoded1)
autoencoder = Model(inputs=input_img, outputs=decoded2)

autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
autoencoder.fit(x_train, x_train,
                epochs=5,
                batch_size=256,
                shuffle=True,
                validation_data=(x_test, x_test))
```

```
Epoch 1/5
235/235 [==============================] - 6s 21ms/step - loss: 0.2227 - val_loss: 0.1431
Epoch 2/5
235/235 [==============================] - 4s 17ms/step - loss: 0.1263 - val_loss: 0.1121
Epoch 3/5
235/235 [==============================] - 5s 21ms/step - loss: 0.1076 - val_loss: 0.1014
Epoch 4/5
235/235 [==============================] - 4s 19ms/step - loss: 0.1000 - val_loss: 0.0960
Epoch 5/5
235/235 [==============================] - 4s 17ms/step - loss: 0.0956 - val_loss: 0.0925
<keras.callbacks.History at 0x7fdcd8645ee0>
```

AlexNet

```
from tensorflow import keras
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.datasets import mnist
import numpy as np

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

```
input_img = Input(shape=(28, 28, 1))
x = Conv2D(96, (11, 11), strides=(4, 4), padding='same', activation='relu')(input_img)
x = MaxPooling2D((3, 3), strides=(2, 2))(x)
x = Conv2D(256, (5, 5), padding='same', activation='relu')(x)
x = MaxPooling2D((3, 3), strides=(2, 2))(x)
x = Conv2D(384, (3, 3), padding='same', activation='relu')(x)
x = Conv2D(384, (3, 3), padding='same', activation='relu')(x)
x = Conv2D(256, (3, 3), padding='same', activation='relu')(x)
x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = Flatten()(x)
x = Dense(4096, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(4096, activation='relu')(x)
x = Dropout(0.5)(x)
output = Dense(num_classes, activation='softmax')(x)
alexnet = Model(inputs=input_img, outputs=output)

alexnet.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
alexnet.fit(x_train, y_train,
            batch_size=128,
            epochs=1,
            verbose=1,
            validation_data=(x_test, y_test))

score = alexnet.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
469/469 [==============================] - 889s 2s/step - loss: 0.6750 - accuracy: 0.7452 - val_loss: 0.0974 - val_accuracy: 0.9724
Test loss: 0.0973912701010704
Test accuracy: 0.9724000096321106
```

LeNet-5

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Conv2D, AveragePooling2D, Flatten

(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape(x_train.shape[0], 28, 28, 1).astype('float32') / 255.0
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1).astype('float32') / 255.0


num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Conv2D(6, kernel_size=(5, 5), strides=(1, 1), activation='relu', input_shape=(28, 28, 1)))
model.add(AveragePooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(16, kernel_size=(5, 5), strides=(1, 1), activation='relu'))
model.add(AveragePooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Flatten())
model.add(Dense(120, activation='relu'))
model.add(Dense(84, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adam(), metrics=['accuracy'])

model.fit(x_train, y_train, batch_size=128, epochs=1, verbose=1, validation_data=(x_test, y_test))

score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
469/469 [==============================] - 35s 73ms/step - loss: 0.3892 - accuracy: 0.8872 - val_loss: 0.1112 - val_accuracy: 0.966
Test loss: 0.11123871803283691
Test accuracy: 0.9660000205039978
```

VGG-16

```
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.utils import to_categorical

# Load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Normalize pixel values to be between 0 and 1
x_train = x_train.astype("float32") / 255.0
x_test = x_test.astype("float32") / 255.0

# Convert labels to one-hot encoding
y_train = to_categorical(y_train, num_classes=10)
y_test = to_categorical(y_test, num_classes=10)

# Reshape images to 4D tensor (batch_size, height, width, channels)
x_train = x_train.reshape((-1, 28, 28, 1))
x_test = x_test.reshape((-1, 28, 28, 1))

# Define VGG16 model
model = Sequential()
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', input_shape=(28, 28, 1)))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(10, activation='softmax'))

# Compile VGG16 model
sgd = SGD(lr=0.01, momentum=0.9, nesterov=True)
model.compile(optimizer=sgd, loss='categorical_crossentropy', metrics=['accuracy'])

# Train VGG16 model
model.fit(x_train, y_train, batch_size=128, epochs=1, validation_split=0.2)

# Evaluate VGG16 model on test set
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizer
375/375 [==============================] - 597s 2s/step - loss: 0.4828 - accuracy: 0.8635 - val_loss: 0.0746 - val_accuracy: 0.9767
Test loss: 0.05981737747788429
Test accuracy: 0.9805999994277954
```