DL lab: 7

AIM: Write a program to implement autoencoder, Lenet5, VGG16 and alexnet on MNIST dataset. Compare the results in terms of training time and accuracy (both training and testing).

1. Auto encoder

In [ ]:
```
from keras.layers import Dense,Conv2D,MaxPooling2D,UpSampling2D
from keras import Input, Model
from keras.datasets import mnist
import numpy as np
import keras
```

In [ ]:
```
encoding_dim = 64
input_img = Input(shape=(784,))

# encoded representation of input
encoded = Dense(encoding_dim, activation='relu')(input_img)

# decoded representation of code
decoded = Dense(784, activation='sigmoid')(encoded)

# Model which take input image and shows decoded images
autoencoder = Model(input_img, decoded)
```

In [ ]:
```
# This model shows encoded images
encoder = Model(input_img, encoded)

# Creating a decoder model
encoded_input = Input(shape=(encoding_dim,))

# last layer of the autoencoder model
decoder_layer = autoencoder.layers[-1]

# decoder model
decoder = Model(encoded_input, decoder_layer(encoded_input))
```

In [ ]:
```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.
x_train = x_train.reshape((len(x_train), np.prod(x_train.shape[1:])))
x_test = x_test.reshape((len(x_test), np.prod(x_test.shape[1:])))
print(x_train.shape)
print(x_test.shape)
```

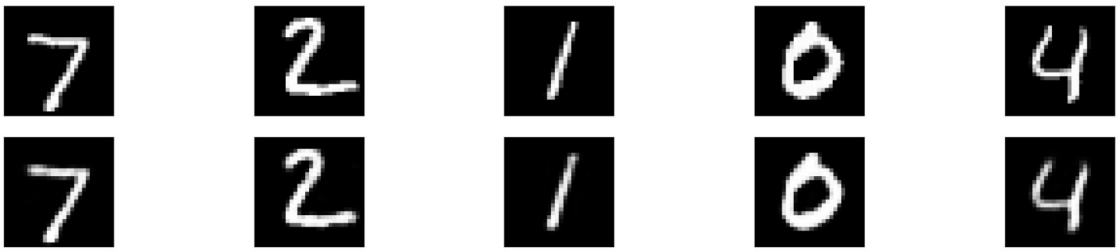
```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-data
sets/mnist.npz
11490434/11490434 [==============================] - 2s 0us/step
(60000, 784)
(10000, 784)
```

In [ ]:
```
%%time

autoencoder.compile(optimizer='adam', loss='binary_crossentropy')

autoencoder.fit(x_train, x_train,epochs=15, batch_size=256, validation_data=(x
encoded_img = encoder.predict(x_test)
decoded_img = decoder.predict(encoded_img)
```

```
Epoch 1/15
235/235 [==============================] - 7s 9ms/step - loss: 0.2459 - val_l
oss: 0.1621
Epoch 2/15
235/235 [==============================] - 2s 7ms/step - loss: 0.1438 - val_l
oss: 0.1274
Epoch 3/15
235/235 [==============================] - 1s 6ms/step - loss: 0.1188 - val_l
oss: 0.1092
Epoch 4/15
235/235 [==============================] - 2s 8ms/step - loss: 0.1046 - val_l
oss: 0.0982
Epoch 5/15
235/235 [==============================] - 2s 7ms/step - loss: 0.0955 - val_l
oss: 0.0908
Epoch 6/15
235/235 [==============================] - 2s 7ms/step - loss: 0.0891 - val_l
oss: 0.0855
Epoch 7/15
235/235 [==============================] - 2s 9ms/step - loss: 0.0846 - val_l
oss: 0.0819
Epoch 8/15
235/235 [==============================] - 2s 9ms/step - loss: 0.0814 - val_l
oss: 0.0793
Epoch 9/15
235/235 [==============================] - 2s 7ms/step - loss: 0.0793 - val_l
oss: 0.0776
Epoch 10/15
235/235 [==============================] - 2s 7ms/step - loss: 0.0778 - val_l
oss: 0.0765
Epoch 11/15
235/235 [==============================] - 2s 8ms/step - loss: 0.0768 - val_l
oss: 0.0756
Epoch 12/15
235/235 [==============================] - 2s 7ms/step - loss: 0.0761 - val_l
oss: 0.0750
Epoch 13/15
235/235 [==============================] - 2s 7ms/step - loss: 0.0755 - val_l
oss: 0.0746
Epoch 14/15
235/235 [==============================] - 1s 6ms/step - loss: 0.0751 - val_l
oss: 0.0743
Epoch 15/15
235/235 [==============================] - 1s 6ms/step - loss: 0.0748 - val_l
oss: 0.0741
313/313 [==============================] - 1s 2ms/step
313/313 [==============================] - 1s 2ms/step
CPU times: user 24.5 s, sys: 2.27 s, total: 26.8 s
Wall time: 44.6 s
```

In [ ]:
```
import matplotlib.pyplot as plt
```

In [ ]:
```
plt.figure(figsize=(20, 4))
for i in range(5):
    # Display original
    ax = plt.subplot(2, 5, i + 1)
    plt.imshow(x_test[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    # Display reconstruction
    ax = plt.subplot(2, 5, i + 1 + 5)
    plt.imshow(decoded_img[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
plt.show()
```



2. Lenet5

In [ ]:
```
import keras.layers as layers
!pip3 install visualkeras
import visualkeras
import time
```

In [ ]:
```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
print('X_train shape', x_train.shape, 'X_test shape', x_test.shape)
```

```
X_train shape (60000, 28, 28) X_test shape (10000, 28, 28)
```

In [ ]:
```
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.
y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)
print('X_train shape', x_train.shape, 'X_test shape', x_test.shape)
```

```
X_train shape (60000, 28, 28) X_test shape (10000, 28, 28)
```

In [ ]:
```
model = keras.Sequential()
model.add(layers.Conv2D(filters=6, kernel_size=(3, 3), activation='relu',input
model.add(layers.AveragePooling2D())
model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model.add(layers.AveragePooling2D())
model.add(layers.Flatten())
model.add(layers.Dense(units=120, activation='relu'))
model.add(layers.Dense(units=84, activation='relu'))
model.add(layers.Dense(units=10, activation = 'softmax'))
model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.opti
```

In [ ]:
```
startTime = time.time()
model.fit(x_train,y_train,epochs=10)
endTime = time.time()
```

```
Epoch 1/10
1875/1875 [==============================] - 11s 3ms/step - loss: 0.2290 - ac
curacy: 0.9296
Epoch 2/10
1875/1875 [==============================] - 12s 6ms/step - loss: 0.0736 - ac
curacy: 0.9782
Epoch 3/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0518 - ac
curacy: 0.9843
Epoch 4/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0403 - acc
uracy: 0.9874
Epoch 5/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0320 - acc
uracy: 0.9900
Epoch 6/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0268 - acc
uracy: 0.9914
Epoch 7/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0233 - acc
uracy: 0.9926
Epoch 8/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0187 - acc
uracy: 0.9939
Epoch 9/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0172 - acc
uracy: 0.9945
Epoch 10/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0143 - acc
uracy: 0.9952
```

In [ ]:
```
models = {}
```

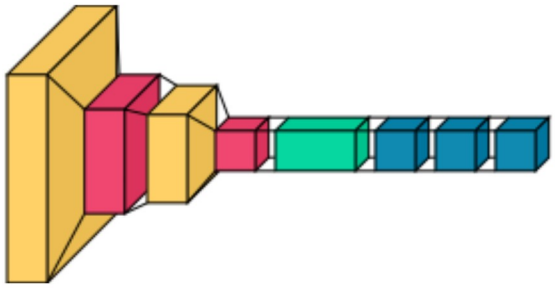
In [ ]:
```
def addToDict(model_name, model):
  models[model_name] = {}

  print("on traning data: ")
  train = model.evaluate(x_train,y_train)
  models[model_name]["train"] = round(train[1]*100,4)

  print("on testing data: ")
  test = model.evaluate(x_test,y_test)
  models[model_name]["test"] = round(test[1]*100, 4)

  models[model_name]["time"] = (endTime - startTime)

  print("data: ")
  print(models[model_name])
```

In [ ]:
```
addToDict("Lenet-5",model)
```

```
on traning data:
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0089 - acc
uracy: 0.9973
on testing data:
313/313 [==============================] - 1s 3ms/step - loss: 0.0286 - accur
acy: 0.9908
data:
{'train': 99.7333, 'test': 99.08, 'time': 82.7493040561676}
```

In [ ]:
```
visualkeras.layered_view(model)
```

Out[ ]:



3. VGG16

In [ ]:
```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPool2D , Flatten
```

In [ ]:
```
model = keras.Sequential()
model.add(layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu',inpu
model.add(layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))

model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))

model.add(layers.Conv2D(filters=128, kernel_size=(3, 3),padding="same", activa
model.add(layers.Conv2D(filters=128, kernel_size=(3, 3),padding="same", activa

model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))

model.add(layers.Conv2D(filters=256, kernel_size=(3, 3),padding="same", activa
model.add(layers.Conv2D(filters=256, kernel_size=(3, 3),padding="same", activa
model.add(layers.Conv2D(filters=256, kernel_size=(3, 3),padding="same", activa

model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))

model.add(layers.Conv2D(filters=512, kernel_size=(3, 3),padding="same", activa
model.add(layers.Conv2D(filters=512, kernel_size=(3, 3),padding="same", activa
model.add(layers.Conv2D(filters=512, kernel_size=(3, 3),padding="same", activa

model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))

model.add(layers.Conv2D(filters=512, kernel_size=(3, 3),padding="same", activa
model.add(layers.Conv2D(filters=512, kernel_size=(3, 3),padding="same", activa
model.add(layers.Conv2D(filters=512, kernel_size=(3, 3),padding="same", activa

model.add(MaxPool2D(pool_size=(2,2),strides=(2,2), padding='same'))

model.add(layers.Flatten())

model.add(layers.Dense(units=120, activation='relu'))
model.add(layers.Dense(units=84, activation='relu'))
model.add(layers.Dense(units=10, activation = 'softmax'))
model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.opti
```

In [ ]:

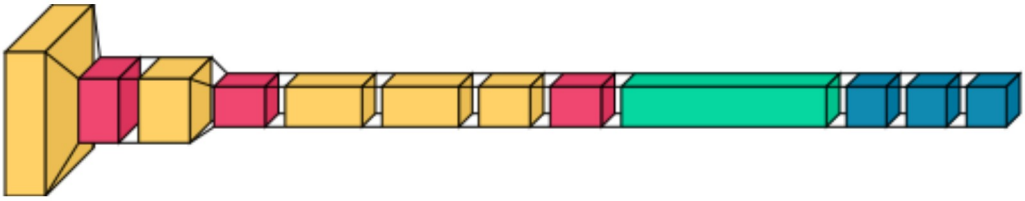
```
startTime = time.time()
model.fit(x_train,y_train,epochs=100)
endTime = time.time()
```

```
Epoch 85/100
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0224 - ac
curacy: 0.9969
Epoch 86/100
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0585 - ac
curacy: 0.9947
Epoch 87/100
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0126 - ac
curacy: 0.9979
Epoch 88/100
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0680 - ac
curacy: 0.9938
Epoch 89/100
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0233 - ac
curacy: 0.9965
Epoch 90/100
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0251 - ac
curacy: 0.9959
Epoch 91/100
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0185 - ac
curacy: 0.9976
Epoch 92/100
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0387 - ac
curacy: 0.9948
Epoch 93/100
1875/1875 [==============================] - 16s 8ms/step - loss: 0.0250 - ac
curacy: 0.9955
Epoch 94/100
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0232 - ac
curacy: 0.9964
Epoch 95/100
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0230 - ac
curacy: 0.9964
Epoch 96/100
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0609 - ac
curacy: 0.9913
Epoch 97/100
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0429 - ac
curacy: 0.9932
Epoch 98/100
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0955 - ac
curacy: 0.9825
Epoch 99/100
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0383 - ac
curacy: 0.9945
Epoch 100/100
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0567 - ac
curacy: 0.9926
```

In [ ]:
```
addToDict("VGG16",model)
visualkeras.layered_view(model)
```

```
on traning data:
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0261 - acc
uracy: 0.9941
on testing data:
313/313 [==============================] - 1s 5ms/step - loss: 0.1247 - accur
acy: 0.9870
data:
{'train': 99.4067, 'test': 98.7, 'time': 1582.2733597755432}
```

Out[ ]:



4. Alex net

In [ ]:

```
model = keras.Sequential()
model.add(layers.Conv2D(filters=96, kernel_size=(11, 11), activation='relu',in

model.add(MaxPool2D(pool_size=(3,3),strides=(2,2)))

model.add(layers.Conv2D(filters=256, kernel_size=(5, 5),padding="same", activa

model.add(MaxPool2D(pool_size=(3,3),strides=(2,2)))

model.add(layers.Conv2D(filters=384, kernel_size=(3, 3),padding="same", activa
model.add(layers.Conv2D(filters=384, kernel_size=(3, 3),padding="same", activa
model.add(layers.Conv2D(filters=256, kernel_size=(3, 3),padding="same", activa

model.add(MaxPool2D(pool_size=(3,3),strides=(2,2), padding='same'))

model.add(layers.Flatten())

model.add(layers.Dense(units=120, activation='relu'))
model.add(layers.Dense(units=84, activation='relu'))
model.add(layers.Dense(units=10, activation = 'softmax'))
model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.opti
```

In [ ]:

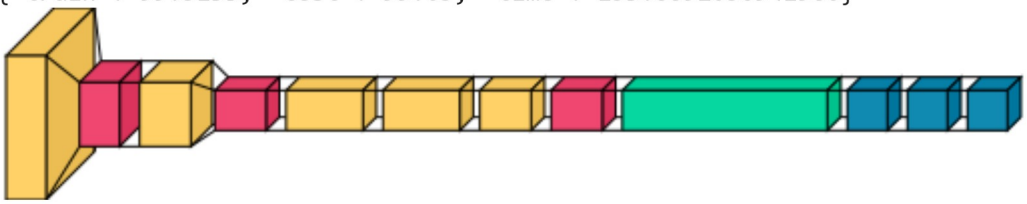
```
startTime = time.time()
model.fit(x_train,y_train,epochs=10)
endTime = time.time()
```

```
Epoch 1/10
1875/1875 [==============================] - 16s 8ms/step - loss: 0.2692 - ac
curacy: 0.9103
Epoch 2/10
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0782 - ac
curacy: 0.9814
Epoch 3/10
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0595 - ac
curacy: 0.9857
Epoch 4/10
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0480 - ac
curacy: 0.9883
Epoch 5/10
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0430 - ac
curacy: 0.9900
Epoch 6/10
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0359 - ac
curacy: 0.9919
Epoch 7/10
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0367 - ac
curacy: 0.9918
Epoch 8/10
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0348 - ac
curacy: 0.9922
Epoch 9/10
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0332 - ac
curacy: 0.9927
Epoch 10/10
1875/1875 [==============================] - 15s 8ms/step - loss: 0.0277 - ac
curacy: 0.9937
```

In [ ]:
```
addToDict("Alex-net",model)
visualkeras.layered_view(model)
```

```
on traning data:
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0201 - acc
uracy: 0.9951
on testing data:
313/313 [==============================] - 2s 6ms/step - loss: 0.0456 - accur
acy: 0.9905
data:
{'train': 99.5133, 'test': 99.05, 'time': 153.8892056941986}
```

Out[ ]:



comparison:

In [ ]:
```
import pandas as pd
```

In [ ]:
```
df = pd.DataFrame(models)
df
```

Out[ ]:

	Lenet-5	VGG16	Alex-net
train	99.733300	99.40670	99.513300
test	99.080000	98.70000	99.050000
time	82.749304	1582.27336	153.889206

In [ ]: