Name: Biki Paul
Stream: CSIT
Year: 2nd
Section: E
Class Roll Number: 17
Enrollment Number: 12019009023028
Registration Number: 304201900900871
Paper Name: Artificial Intelligence & Machine Learning Advanced Laboratory

Assignment-8

In [1]:
```
import tensorflow as tf
```

In [2]:
```
import keras
```

```
Using TensorFlow backend.
```

In [3]:
```
import numpy as np
```

In [4]:
```
print(np.__version__)
```

```
1.19.2
```

In [5]:
```
print(tf.__version__)
```

```
2.1.0
```

In [6]:
```
from sklearn.datasets import load_digits
```

In [7]:
```
import matplotlib.pyplot as plt
```

In [8]:
```
digits = load_digits()
```

In [9]:
```
image = digits.data[55]
```

In [10]:
```
print(image)
```

```
[ 0.  0.  2. 14. 15.  5.  0.  0.  0.  0. 10. 16. 16. 15.  1.  0.  0.  3.
 16. 10. 10. 16.  4.  0.  0.  5. 16.  0.  0. 14.  6.  0.  0.  5. 16.  6.
  0. 12.  7.  0.  0.  1. 15. 13.  4. 13.  6.  0.  0.  0. 11. 16. 16. 15.
  0.  0.  0.  0.  2. 11. 13.  4.  0.  0.]
```

In [11]:
```
image.shape
```

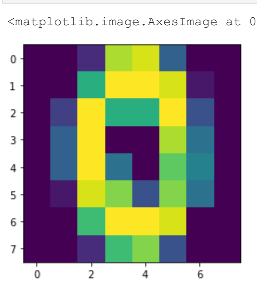
Out[11]:
```
(64,)
```

In [12]:
```
np.reshape(image, (8, 8))
```

Out[12]:
```
array([[ 0.,  0.,  2., 14., 15.,  5.,  0.,  0.],
       [ 0.,  0., 10., 16., 16., 15.,  1.,  0.],
       [ 0.,  3., 16., 10., 10., 16.,  4.,  0.],
       [ 0.,  5., 16.,  0.,  0., 14.,  6.,  0.],
       [ 0.,  5., 16.,  6.,  0., 12.,  7.,  0.],
       [ 0.,  1., 15., 13.,  4., 13.,  6.,  0.],
       [ 0.,  0., 11., 16., 16., 15.,  0.,  0.],
       [ 0.,  0.,  2., 11., 13.,  4.,  0.,  0.]])
```

In [13]:
```
plt.imshow(np.reshape(image, (8, 8)))
```

Out[13]:
```
<matplotlib.image.AxesImage at 0x12c63e2b080>
```



In [14]:
```
digits.target[55]
```

Out[14]:
```
0
```

In [15]:
```
from sklearn.model_selection import train_test_split
```

In [16]:
```
x_train, x_test, y_train, y_test = train_test_split(digits.data, digits.target, test_s
```

In [17]:
```
x_train.shape
```

Out[17]:
```
(1437, 64)
```

In [18]:
```
x_test.shape
```

Out[18]:
```
(360, 64)
```

In [19]:
```
x_train = x_train.reshape(1437, 8, 8, 1)
```

In [20]:
```
x_test = x_test.reshape(360, 8, 8, 1)
```

In [21]:
```
y_train.shape
```

Out[21]:
```
(1437,)
```

In [22]:
```
y_train
```

Out[22]:
```
array([3, 0, 9, ..., 2, 2, 9])
```

In [23]:
```
from keras.utils import to_categorical
```

In [24]:
```
y_train = to_categorical(y_train)
```

In [25]:
```
y_test = to_categorical(y_test)
```

In [26]:
```
y_train.shape
```

Out[26]:
```
(1437, 10)
```

In [27]:
```
y_train
```

Out[27]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 1.],
       ...,
       [0., 0., 1., ..., 0., 0., 0.],
       [0., 0., 1., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 1.]], dtype=float32)
```

In [28]:
```
from keras.models import Sequential
```

In [29]:
```
from keras.layers import Dense, Conv2D, Flatten, MaxPooling2D
```

In [30]:
```
model = Sequential()
```

In [31]:
```
model = Sequential([Conv2D(8, 3, input_shape =(8, 8, 1)), MaxPooling2D(pool_size = 2),
```

In [32]:
```
model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accu
```

In [33]:
```
model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=10)
```

```
Train on 1437 samples, validate on 360 samples
Epoch 1/10
1437/1437 [==============================] - 0s 222us/step - loss: 4.5690 - accuracy:
0.1371 - val_loss: 2.8735 - val_accuracy: 0.2528
Epoch 2/10
1437/1437 [==============================] - 0s 93us/step - loss: 2.2989 - accuracy:
0.3459 - val_loss: 1.6267 - val_accuracy: 0.4583
Epoch 3/10
1437/1437 [==============================] - 0s 102us/step - loss: 1.3768 - accuracy:
0.5532 - val_loss: 0.9890 - val_accuracy: 0.6667
Epoch 4/10
1437/1437 [==============================] - 0s 148us/step - loss: 0.9056 - accuracy:
0.7084 - val_loss: 0.6829 - val_accuracy: 0.8000
Epoch 5/10
1437/1437 [==============================] - 0s 158us/step - loss: 0.6596 - accuracy:
0.7954 - val_loss: 0.4984 - val_accuracy: 0.8611
Epoch 6/10
1437/1437 [==============================] - 0s 151us/step - loss: 0.5151 - accuracy:
0.8372 - val_loss: 0.3902 - val_accuracy: 0.9000
Epoch 7/10
1437/1437 [==============================] - 0s 119us/step - loss: 0.4351 - accuracy:
0.8678 - val_loss: 0.3373 - val_accuracy: 0.9139
Epoch 8/10
1437/1437 [==============================] - 0s 155us/step - loss: 0.3689 - accuracy:
0.8887 - val_loss: 0.2888 - val_accuracy: 0.9361
Epoch 9/10
1437/1437 [==============================] - 0s 139us/step - loss: 0.3213 - accuracy:
0.9061 - val_loss: 0.2548 - val_accuracy: 0.9444
Epoch 10/10
1437/1437 [==============================] - 0s 117us/step - loss: 0.2879 - accuracy:
0.9179 - val_loss: 0.2306 - val_accuracy: 0.9417
```

Out[33]:
```
<keras.callbacks.callbacks.History at 0x12c5ac00ac8>
```

In [34]:
```
model.predict(x_test[0:1])
```

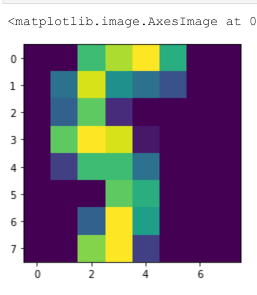
Out[34]:
```
array([[8.8836707e-04, 1.1965218e-04, 7.5979577e-04, 2.0590072e-05,
        4.2586671e-05, 9.9022466e-01, 6.2972135e-03, 5.6099315e-04,
        1.0860979e-03, 5.9629507e-08]], dtype=float32)
```

In [35]:
```
y_test[0:1]
```

Out[35]:
```
array([[0., 0., 0., 0., 0., 1., 0., 0., 0., 0.]], dtype=float32)
```

In [36]:
```
plt.imshow(np.reshape(x_test[0], (8, 8)))
```

Out[36]:
```
<matplotlib.image.AxesImage at 0x12c654560b8>
```



In [37]:
```
model.predict(x_test[100:101])
```

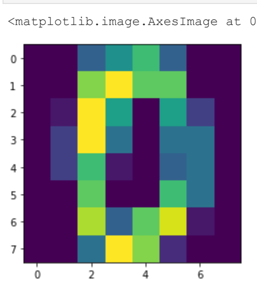
Out[37]:
```
array([[7.9917586e-01, 4.1413645e-04, 1.0691656e-02, 2.3537922e-04,
        1.1571106e-05, 1.8215738e-01, 4.1056718e-03, 2.1214911e-03,
        3.1991811e-05, 1.0549608e-03]], dtype=float32)
```

In [38]:
```
y_test[100]
```

Out[38]:
```
array([1., 0., 0., 0., 0., 0., 0., 0., 0., 0.], dtype=float32)
```

In [39]:
```
plt.imshow(np.reshape(x_test[100], (8, 8)))
```

Out[39]:
```
<matplotlib.image.AxesImage at 0x12c654bdbe0>
```

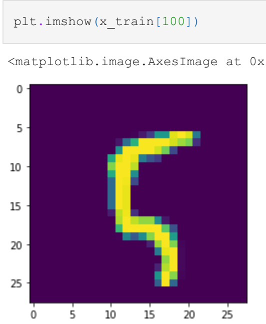


In [42]:
```
from keras.datasets import mnist
```

In [43]:
```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

In [44]:
```
plt.imshow(x_train[100])
```

Out[44]:
```
<matplotlib.image.AxesImage at 0x12c65599860>
```



In [45]:
```
from numpy import array
from scipy.linalg import svd
A = array([[1, 2], [3 ,4], [5, 6]])
print(A)
# svd
U, s, VT = svd(A)
print(U)
print(s)
print(VT)
```

```
[[1 2]
 [3 4]
 [5 6]]
[[-0.2298477   0.88346102  0.40824829]
 [-0.52474482  0.24078249 -0.81649658]
 [-0.81964194 -0.40189603  0.40824829]]
[9.52551809 0.51430058]
[[-0.61962948 -0.78489445]
 [-0.78489445  0.61962948]]
```

In [46]:
```
from numpy import array
from numpy import diag
from numpy import dot
from numpy import zeros
from scipy.linalg import svd
# define a matrix
A = array([[1, 2], [3 ,4], [5, 6]])
print(A)
# Singular-value decomposition
U, s, VT = svd(A)
# create m*n Sigma matrix
Sigma = zeros((A.shape[0], A.shape[1]))
# populate Sigma with m*n diagonal matrix
Sigma[:A.shape[1], :A.shape[1]] = diag(s)
B = U.dot(Sigma.dot(VT))
print(B)
```

```
[[1 2]
 [3 4]
 [5 6]]
[[1. 2.]
 [3. 4.]
 [5. 6.]]
```

In [45]:
```
from numpy import array
from numpy import diag
from numpy import dot
from numpy import zeros
from scipy.linalg import svd
# define a matrix
A = array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(A)
# Singular-value decomposition
U, s, VT = svd(A)
#create m*n Sigma matrix
Sigma = diag(s)
# reconstruct matrix
B = U.dot(Sigma.dot(VT))
print(B)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
[[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]
```

In [47]:
```
from numpy import array
from scipy.linalg import pinv
# define a matrix
A = array([
    [0.1, 0.2],
    [0.3, 0.4],
    [0.5, 0.6],
    [0.7, 0.8]])
print(A)
# calculate psudoinverse
B = pinv(A)
print(B)
```

```
[[0.1 0.2]
 [0.3 0.4]
 [0.5 0.6]
 [0.7 0.8]]
[[-1.00000000e+01 -5.00000000e+00  1.31262171e-14  5.00000000e+00]
 [ 8.50000000e+00  4.50000000e+00  5.00000000e-01 -3.50000000e+00]]
```

In [48]:
```
from numpy import diag
from numpy import zeros
# define a matrix
A = array([
    [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    [11, 12, 13, 14, 15, 16, 17, 18, 19, 20],
    [21, 22, 23, 24, 25, 26, 27, 28, 29, 30]])
print(A)
# Singular-value decomposition
U, s, VT = svd(A)
# create m*n Sigma matrix
Sigma = zeros((A.shape[0], A.shape[1]))
# populate Sigma with m*n diagonal matrix
Sigma[:A.shape[0], :A.shape[0]] = diag(s)
# select
n_elements = 2
Sigma = Sigma[:, :n_elements]
VT = VT[:n_elements, :]
# reconstruct
B = U.dot(Sigma.dot(VT))
print(B)
# transfrom
T = U.dot(Sigma)
print(T)
T = A.dot(VT.T)
print(T)
```

```
[[ 1  2  3  4  5  6  7  8  9 10]
 [11 12 13 14 15 16 17 18 19 20]
 [21 22 23 24 25 26 27 28 29 30]]
[[ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]
 [11. 12. 13. 14. 15. 16. 17. 18. 19. 20.]
 [21. 22. 23. 24. 25. 26. 27. 28. 29. 30.]]
[[-18.52157747  -6.47697214]
 [-49.81310011  -1.91182038]
 [-81.10462276   2.65333138]]
[[-18.52157747  -6.47697214]
 [-49.81310011  -1.91182038]
 [-81.10462276   2.65333138]]
```

In [49]:
```
from numpy import array
from sklearn.decomposition import TruncatedSVD
# define array
A = array([
    [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    [11, 12, 13, 14, 15, 16, 17, 18, 19, 20],
    [21, 22, 23, 24, 25, 26, 27, 28, 29, 30]])
print(A)
# svd
svd = TruncatedSVD(n_components = 2)
svd.fit(A)
result = svd.transform(A)
print(result)
```

```
[[ 1  2  3  4  5  6  7  8  9 10]
 [11 12 13 14 15 16 17 18 19 20]
 [21 22 23 24 25 26 27 28 29 30]]
[[18.52157747  6.47697214]
 [49.81310011  1.91182038]
 [81.10462276 -2.65333138]]
```

In [50]:
```
from pandas import read_csv
from sklearn.decomposition import PCA
path = r'C:\Users\user\OneDrive\Desktop\Cvsvfile\pima-indians-diabetes.csv'
names = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI',
dataframe = read_csv(path, names = names)
array = dataframe.values
x = array[:,0:8]
y = array[:,8]
pca = PCA(n_components = 5)
fit = pca.fit(x)
print(("Explained Variance: %s") % (fit.explained_variance_ratio_))
print(fit.components_)
```

```
Explained Variance: [0.88854663 0.06159078 0.02579012 0.01308614 0.00744094]
[[-2.02176587e-03  9.78115765e-02  1.60930503e-02  6.07566861e-02
   9.93110844e-01  1.40108085e-02  5.37167919e-04 -3.56474430e-03]
 [-2.26488861e-02 -9.72210040e-01 -1.41909330e-01  5.78614699e-02
   9.46266913e-02 -4.69729766e-02 -8.16804621e-04 -1.40168181e-01]
 [-2.24649003e-02  1.43428710e-01 -9.22467192e-01 -3.07013055e-01
   2.09773019e-02 -1.32444542e-01 -6.39983017e-04 -1.25454310e-01]
 [-4.90459604e-02  1.19830016e-01 -2.62742788e-01  8.84369380e-01
  -6.55503615e-02  1.92801728e-01  2.69908637e-03 -3.01024330e-01]
 [ 1.51612874e-01 -8.79407680e-02 -2.32165009e-01  2.59973487e-01
  -1.72312241e-04  2.14744823e-02  1.64080684e-03  9.20504903e-01]]
```

In [51]:
```
from sklearn.datasets import load_digits
from sklearn.decomposition import IncrementalPCA
X, _ = load_digits(return_X_y = True)
transformer = IncrementalPCA(n_components = 10, batch_size = 100)
transformer.partial_fit(X[:100, :])
X_transformed = transformer.fit_transform(X)
X_transformed.shape
```

Out[51]:
```
(1797, 10)
```

In [52]:
```
from sklearn.datasets import load_digits
from sklearn.decomposition import KernelPCA
X, _ = load_digits(return_X_y = True)
transformer = KernelPCA(n_components = 10, kernel = 'sigmoid')
X_transformed = transformer.fit_transform(X)
X_transformed.shape
```

Out[52]:
```
(1797, 10)
```

In [ ]: