Annapoornima S

225229101

Lab12: Image corpus creation and Transfer Learning in CNN

1. Import the necessary libraries

In [1]:

```
import datetime
import keras
import pandas as pd
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D
```

2. Initialize some parameters

In [2]:

```
now = datetime.datetime.now  #get current time

batch_size = 128
num_classes = 5
epochs = 5

img_rows, img_cols = 28, 28
filters = 32
pool_size = 2
kernel_size = 3
```

3.Partition MINST dataset

In [4]:

```
#data, shuffled and split between train and test sets

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

In [5]:

```
#create 2 datasets: one with digits below 5 and one with 5 and above

x_train_lt5 = x_train[y_train < 5]
y_train_lt5 = y_train[y_train < 5]
x_test_lt5 = x_test[y_test < 5]
y_test_lt5 = y_test[y_test < 5]

x_train_gte5 = x_train[y_train >= 5]
y_train_gte5 = y_train[y_train >= 5] - 5
x_test_gte5 = x_test[y_test >= 5]
y_test_gte5 = y_test[y_test >= 5] - 5
```

In [6]:

```
x_train_lt5.shape
```

Out[6]:

```
(30596, 28, 28)
```

In [7]:

```
x_train_gte5.shape
```

Out[7]:

```
(29404, 28, 28)
```

4. Define the "feature" layers

In [8]:

```
feature_layers =[
    Conv2D(filters, kernel_size = (3,3), activation = 'relu',padding='same',input_shape
    Conv2D(filters, kernel_size = (3,3), activation = 'relu'),
    MaxPooling2D(pool_size=2),
    Dropout(0.25),
    Flatten()
]
```

5. Define the "classification" layers

In [9]:

```
classification_layers = [
    Dense(128, activation = 'relu'),
    Dropout(0.5),
    Dense(num_classes, activation = 'softmax')
]
```

6. Define a Sequential model

In [10]:

```
model = Sequential(feature_layers+classification_layers)
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 32)        320       
                                                                 
 conv2d_1 (Conv2D)           (None, 26, 26, 32)        9248      
                                                                 
 max_pooling2d (MaxPooling2  (None, 13, 13, 32)        0         
 D)                                                              
                                                                 
 dropout (Dropout)           (None, 13, 13, 32)        0         
                                                                 
 flatten (Flatten)           (None, 5408)              0         
                                                                 
 dense (Dense)               (None, 128)               692352    
                                                                 
 dropout_1 (Dropout)         (None, 128)               0         
                                                                 
 dense_1 (Dense)             (None, 5)                 645       
                                                                 
=================================================================
Total params: 702565 (2.68 MB)
Trainable params: 702565 (2.68 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

7. Create function train_model()

In [11]:

```
def train_model(model,train,test,num_classes):
    train[0] = train[0].reshape(train[0].shape[0],img_rows,img_cols,1)
    test[0] = test[0].reshape(test[0].shape[0],img_rows,img_cols,1)

    train[0] = train[0].astype('float32')
    test[0] = test[0].astype('float32')

    train[0] /= 255
    test[0] /= 255

    print(train[0].shape)
    print(test[0].shape)

    s = pd.Series(train[1])
    train[1] = pd.get_dummies(s)
    train[1] = train[1].values

    s = pd.Series(test[1])
    test[1] = pd.get_dummies(s)
    test[1] = test[1].values

    model.compile(optimizer='adadelta',loss='categorical_crossentropy',metrics=['accurac

    train_start_time=now()

    model.fit(train[0],train[1],epochs=epochs,verbose=2)

    print('\n')
    print('\n')

    print('Training time: %s' % (now() - train_start_time))

    score=model.evaluate(test[0],test[1],verbose=0)

    print('test loss ',score[0])
    print('test accuracy ',score[1])
```

8. Training on digits 5 to 9

In [14]:

```
train_model(model, [x_train_gte5, y_train_gte5], [x_test_gte5, y_test_gte5], num_classes
```

```
(29404, 28, 28, 1)
(4861, 28, 28, 1)
Epoch 1/5
919/919 - 76s - loss: 1.5799 - accuracy: 0.2967 - 76s/epoch - 82ms/step
Epoch 2/5
919/919 - 55s - loss: 1.5060 - accuracy: 0.4736 - 55s/epoch - 60ms/step
Epoch 3/5
919/919 - 68s - loss: 1.4131 - accuracy: 0.5893 - 68s/epoch - 74ms/step
Epoch 4/5
919/919 - 64s - loss: 1.2962 - accuracy: 0.6603 - 64s/epoch - 69ms/step
Epoch 5/5
919/919 - 62s - loss: 1.1590 - accuracy: 0.7052 - 62s/epoch - 67ms/step




Training time: 0:05:25.723386
test loss  1.022234559059143
test accuracy  0.8366591334342957
```

9. Freeze Feature Layers

In [15]:

```
#Freeze only the feature layers

for l in feature_layers:
  l.trainable = False
```

10. Print Summary

In [16]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 32)        320       
                                                                 
 conv2d_1 (Conv2D)           (None, 26, 26, 32)        9248      
                                                                 
 max_pooling2d (MaxPooling2  (None, 13, 13, 32)        0         
 D)                                                              
                                                                 
 dropout (Dropout)           (None, 13, 13, 32)        0         
                                                                 
 flatten (Flatten)           (None, 5408)              0         
                                                                 
 dense (Dense)               (None, 128)               692352    
                                                                 
 dropout_1 (Dropout)         (None, 128)               0         
                                                                 
 dense_1 (Dense)             (None, 5)                 645       
                                                                 
=================================================================
Total params: 702565 (2.68 MB)
Trainable params: 692997 (2.64 MB)
Non-trainable params: 9568 (37.38 KB)
_________________________________________________________________
```

11. Training for digits 0 to 4 based on digits 5 to 9

In [32]:

```
train_model(model, [x_train_lt5, y_train_lt5], [x_test_lt5, y_test_lt5], num_classes)
```

```
(30596, 28, 28, 1)
(5139, 28, 28, 1)
Epoch 1/5
957/957 - 27s - loss: 0.7033 - accuracy: 0.8567 - 27s/epoch - 28ms/step
Epoch 2/5
957/957 - 27s - loss: 0.6345 - accuracy: 0.8720 - 27s/epoch - 28ms/step
Epoch 3/5
957/957 - 27s - loss: 0.5740 - accuracy: 0.8836 - 27s/epoch - 28ms/step
Epoch 4/5
957/957 - 27s - loss: 0.5292 - accuracy: 0.8869 - 27s/epoch - 28ms/step
Epoch 5/5
957/957 - 28s - loss: 0.4898 - accuracy: 0.8945 - 28s/epoch - 29ms/step



Training time: 0:02:15.072956
test loss  0.3850436508655548
test accuracy  0.943568766117096
```

12. Reversing the training process

In [41]:

```
modelReverse = Sequential(feature_layers+classification_layers)
modelReverse.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 32)        320       
                                                                 
 conv2d_1 (Conv2D)           (None, 26, 26, 32)        9248      
                                                                 
 max_pooling2d (MaxPooling2  (None, 13, 13, 32)        0         
 D)                                                              
                                                                 
 dropout (Dropout)           (None, 13, 13, 32)        0         
                                                                 
 flatten (Flatten)           (None, 5408)              0         
                                                                 
 dense (Dense)               (None, 128)               692352    
                                                                 
 dropout_1 (Dropout)         (None, 128)               0         
                                                                 
 dense_1 (Dense)             (None, 5)                 645       
                                                                 
=================================================================
Total params: 702565 (2.68 MB)
Trainable params: 692997 (2.64 MB)
Non-trainable params: 9568 (37.38 KB)
_________________________________________________________________
```

In [44]:

```
train_model(modelReverse, [x_train_lt5, y_train_lt5], [x_test_lt5, y_test_lt5], num_clas
```

```
(30596, 28, 28, 1)
(5139, 28, 28, 1)
Epoch 1/5
957/957 - 27s - loss: 0.4572 - accuracy: 0.9025 - 27s/epoch - 29ms/step
Epoch 2/5
957/957 - 25s - loss: 0.4281 - accuracy: 0.9056 - 25s/epoch - 27ms/step
Epoch 3/5
957/957 - 25s - loss: 0.4068 - accuracy: 0.9081 - 25s/epoch - 27ms/step
Epoch 4/5
957/957 - 29s - loss: 0.3911 - accuracy: 0.9089 - 29s/epoch - 31ms/step
Epoch 5/5
957/957 - 32s - loss: 0.3704 - accuracy: 0.9136 - 32s/epoch - 33ms/step



Training time: 0:02:19.597069
test loss  0.27967116236686707
test accuracy  0.9527145624160767
```

In [45]:

```
#Freeze only the feature layers

for l in feature_layers:
  l.trainable = False
```

In [46]:

```
modelReverse.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 32)        320       
                                                                 
 conv2d_1 (Conv2D)           (None, 26, 26, 32)        9248      
                                                                 
 max_pooling2d (MaxPooling2  (None, 13, 13, 32)        0         
 D)                                                              
                                                                 
 dropout (Dropout)           (None, 13, 13, 32)        0         
                                                                 
 flatten (Flatten)           (None, 5408)              0         
                                                                 
 dense (Dense)               (None, 128)               692352    
                                                                 
 dropout_1 (Dropout)         (None, 128)               0         
                                                                 
 dense_1 (Dense)             (None, 5)                 645       
                                                                 
=================================================================
Total params: 702565 (2.68 MB)
Trainable params: 692997 (2.64 MB)
Non-trainable params: 9568 (37.38 KB)
_________________________________________________________________
```

In [ ]:

```
```