Home Assignment – 5

Prepared By: Birva Patel (1111092)

Professor: Dr. Yimin Yang

- **(4 Points) run the pretrained DCNN from Section 2 with the data from Section 1, take the average accuracy from 3 runs (test and training).**

  According to my student-id : last digit = 2 → Scene-15 dataset and Caltech101
  Second Last digit = 9 → Alexnet

- Alexnet pretrained is not avilable as vgg16, resnet and inception so I used Alexnet weights available for keras and put them to train the network.
- I also used data augmentation method to increase the accuracy of the model.
- I tried using data shuffling, but it was making process more slow and tidious. And most importantly, result difference was not that much. So I put that part in comment.
- As mentioned in the assignment, I used 30 images per class from caltech101 and 100 images from each class in scene 15.
- For 3 run, I simply call fit function 3 times in my model to get average accuracy.

Code:

- **Pretrained ALEXNET with SCENE-15**

```
from keras.preprocessing.image import ImageDataGenerator

import numpy as np

from sklearn.preprocessing import LabelBinarizer

import cv2

from keras.optimizers import Adam,SGD

import os,keras

import time

########################################## Preprocessing of dataset
####################################

# load the dataset

path = '15-Scene' #Add the path to the dataset

categories = sorted(os.listdir(path))

ncategories = len(categories)
```

```
# Dividing First 30 images as Training and rest as testing
data_train = []
labels_train = []
data_test = []
labels_test = []

for i, category in enumerate(categories):
   counter = 0;
         #np.random.shuffle(categories)
   for f in os.listdir(path + "/" + category):
      ext = os.path.splitext(f)[1]
      fullpath = os.path.join(path + "/" + category, f)
      label = fullpath.split(os.path.sep)[-2]
      image = cv2.imread(fullpath)
      image = cv2.resize(image, (224, 224))
      counter = counter + 1
      if (counter <= 100):
         data_train.append(image)
         labels_train.append(label)
      else:
         data_test.append(image)
         labels_test.append(label)

print ('First 30 images per class as Training and rest as testing')
#Normalization
x_train = np.array(data_train, dtype="float") / 255.0
x_test = np.array(data_test, dtype="float") / 255.0
del data_train
del data_test
```

```
# LabelBinarizer
lb = LabelBinarizer()
y_train = lb.fit_transform(labels_train)
y_test = lb.fit_transform(labels_test)
del labels_train
del labels_test
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
############################################## Model Implementation
############################################################
# Data Augmentation
gen = ImageDataGenerator(width_shift_range=.2,
                         height_shift_range=.2,
                        zoom_range=0.2)

test_gen = ImageDataGenerator()
train_generator = gen.flow(x_train, y_train, batch_size=64)
test_generator = test_gen.flow(x_test, y_test, batch_size=64)

# Alexnet model with 5 convolution layer, 2 fully connected layer and 1 output layer.
model = keras.models.Sequential([
    keras.layers.Conv2D(filters=96, kernel_size=(11,11), strides=(4,4), activation='relu',
input_shape=(224,224,3)),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(3,3), strides=(2,2)),
    keras.layers.Conv2D(filters=256, kernel_size=(5,5), strides=(1,1), activation='relu',
padding="same"),
    keras.layers.BatchNormalization(),
```

```
    keras.layers.MaxPool2D(pool_size=(3,3), strides=(2,2)),
    keras.layers.Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2D(filters=384, kernel_size=(1,1), strides=(1,1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2D(filters=256, kernel_size=(1,1), strides=(1,1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(3,3), strides=(2,2)),
    keras.layers.Flatten(),
    keras.layers.Dense(4096, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(4096, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(15, activation='softmax')
])


start = time.time()
model.load_weights("15-alexnet_weights.h5")
sgd = keras.optimizers.SGD(lr=0.01,momentum=0.9)
# compile the model to make a use of categorical cross-entropy loss function and sgd optimizer to get high accuracy.
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
##### Making 3 Run of the model to testify it correctly.
itr1 = model.fit(train_generator,verbose=1,epochs=10,shuffle=True)
itr2 = model.fit(train_generator,verbose=1, epochs=10,shuffle=True)
itr3 = model.fit(train_generator,verbose=1, epochs=10,shuffle=True)
################################################## Results
##################################################
```

```
score = model.evaluate(test_generator)
stop = time.time()
print(f"Training accuracy time with Alexnet pretrained model with Scene-15: {stop - start}s")
print('Test accuracy:', (score[1]*100))
```

- **Pretrained ALEXNET with CALTECH 101**

```
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
from sklearn.preprocessing import LabelBinarizer
import cv2
from keras.optimizers import Adam,SGD
import os,keras
import time
######################################### Preprocessing of dataset
###################################
# load the dataset
path = '15-Scene' #Add the path to the dataset
categories = sorted(os.listdir(path))
ncategories = len(categories)

# Dividing First 30 images as Training and rest as testing
data_train = []
labels_train = []
data_test = []
labels_test = []

for i, category in enumerate(categories):
    counter = 0;
        #np.random.shuffle(categories)
    for f in os.listdir(path + "/" + category):
        ext = os.path.splitext(f)[1]
```

```
        fullpath = os.path.join(path + "/" + category, f)
        label = fullpath.split(os.path.sep)[-2]
        image = cv2.imread(fullpath)
        image = cv2.resize(image, (224, 224))
        counter = counter + 1
        if (counter <= 100):
            data_train.append(image)
            labels_train.append(label)
        else:
            data_test.append(image)
            labels_test.append(label)

print ('First 30 images per class as Training and rest as testing')
#Normalization
x_train = np.array(data_train, dtype="float") / 255.0
x_test = np.array(data_test, dtype="float") / 255.0
del data_train
del data_test
# LabelBinarizer
lb = LabelBinarizer()
y_train = lb.fit_transform(labels_train)
y_test = lb.fit_transform(labels_test)
del labels_train
del labels_test
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
############################################## Model Implementation
############################################################
```

```
# Data Augmentation
gen = ImageDataGenerator(width_shift_range=.2,
                         height_shift_range=.2,
                        zoom_range=0.2)

test_gen = ImageDataGenerator()
train_generator = gen.flow(x_train, y_train, batch_size=64)
test_generator = test_gen.flow(x_test, y_test, batch_size=64)

# Alexnet model with 5 convolution layer, 2 fully connected layer and 1 output layer.
model = keras.models.Sequential([
    keras.layers.Conv2D(filters=96, kernel_size=(11,11), strides=(4,4), activation='relu', input_shape=(224,224,3)),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(3,3), strides=(2,2)),
    keras.layers.Conv2D(filters=256, kernel_size=(5,5), strides=(1,1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(3,3), strides=(2,2)),
    keras.layers.Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2D(filters=384, kernel_size=(1,1), strides=(1,1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2D(filters=256, kernel_size=(1,1), strides=(1,1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(3,3), strides=(2,2)),
    keras.layers.Flatten(),
    keras.layers.Dense(4096, activation='relu'),
```

```
    keras.layers.Dropout(0.5),
    keras.layers.Dense(4096, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(15, activation='softmax')
])


start = time.time()
model.load_weights("15-alexnet_weights.h5")
sgd = keras.optimizers.SGD(lr=0.01,momentum=0.9)
# compile the model to make a use of categorical cross-entropy loss function and sgd optimizer to get high accuracy.
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
##### Making 3 Run of the model to testify it correctly.
itr1 = model.fit(train_generator,verbose=1,epochs=10,shuffle=True)
itr2 = model.fit(train_generator,verbose=1, epochs=10,shuffle=True)
itr3 = model.fit(train_generator,verbose=1, epochs=10,shuffle=True)
########################################################## Results ##################################################
score = model.evaluate(test_generator)
stop = time.time()
print(f"Training accuracy time with Alexnet pretrained model with Scene-15: {stop - start}s")
print('Test accuracy:', (score[1]*100))
```

- **(4 points) run the DCNN from Section 2 with the data from Section 1 from scratch, take the average accuracy from 1 runs (test and training)**

ANS:

- I am using Scene-15 and caltech 101 on alexnet model. Data set is large and input image size for alexnet is (224,224,3) which makes it difficult to get more accurate classification.
- I tried using random shuffling but same as pretrained network it was not making much difference and taking much time so I just put comment on that part.
- Also I used 30 images per class for caltech and 100 images per class in scene-15.

CODE:

- **ALEXNET with CALTECH 101 from Scratch**

```
from keras.preprocessing.image import ImageDataGenerator

import numpy as np

from sklearn.preprocessing import LabelBinarizer

import cv2

from keras.optimizers import Adam,SGD

import os,keras

import time

########################################  Preprocessing of data
########################################

# load the dataset

path = 'Caltech101'

categories = sorted(os.listdir(path))

ncategories = len(categories)


# Dividing First 30 images are considered as Training and rest as testing

data_train = []

labels_train = []

data_test = []

labels_test = []

for i, category in enumerate(categories):

    counter = 0;
```

```
            #np.random.shuffle(categories)
    for f in os.listdir(path + "/" + category):
        ext = os.path.splitext(f)[1]
        fullpath = os.path.join(path + "/" + category, f)
        label = fullpath.split(os.path.sep)[-2]
        image = cv2.imread(fullpath)
        image = cv2.resize(image, (227, 227))
        counter = counter + 1
        if (counter <= 30):
            data_train.append(image)
            labels_train.append(label)
        else:
            data_test.append(image)
            labels_test.append(label)

print ('First 100 images per class as Training and rest as testing')
#Normalization
x_train = np.array(data_train, dtype="float") / 255.0
x_test = np.array(data_test, dtype="float") / 255.0
del data_train
del data_test
#LabelBinarizer
lb = LabelBinarizer()
y_train = lb.fit_transform(labels_train)
y_test = lb.fit_transform(labels_test)
del labels_train
del labels_test
print("Data Splitted")
print(x_train.shape)
```

```
print(y_train.shape)

print(x_test.shape)

print(y_test.shape)

############################################## Model Implementation
#################################################################

# Data Augmentation

gen = ImageDataGenerator(width_shift_range=.2, height_shift_range=.2, zoom_range=0.2)


test_gen = ImageDataGenerator()

train_generator = gen.flow(x_train, y_train, batch_size=64)

test_generator = test_gen.flow(x_test, y_test, batch_size=64)


# Alexnet model with 5 convolution layer, 2 fully connected layer and 1 output layer.

model = keras.models.Sequential([

   keras.layers.Conv2D(filters=96, kernel_size=(11,11), strides=(4,4), activation='relu',
input_shape=(227,227,3)),

   keras.layers.BatchNormalization(),

   keras.layers.MaxPool2D(pool_size=(3,3), strides=(2,2)),

   keras.layers.Conv2D(filters=256, kernel_size=(5,5), strides=(1,1), activation='relu',
padding="same"),

   keras.layers.BatchNormalization(),

   keras.layers.MaxPool2D(pool_size=(3,3), strides=(2,2)),

   keras.layers.Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), activation='relu',
padding="same"),

   keras.layers.BatchNormalization(),

   keras.layers.Conv2D(filters=384, kernel_size=(1,1), strides=(1,1), activation='relu',
padding="same"),

   keras.layers.BatchNormalization(),

   keras.layers.Conv2D(filters=256, kernel_size=(1,1), strides=(1,1), activation='relu',
padding="same"),

   keras.layers.BatchNormalization(),
```

```
    keras.layers.MaxPool2D(pool_size=(3,3), strides=(2,2)),
    keras.layers.Flatten(),
    keras.layers.Dense(4096, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(4096, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(102, activation='softmax')
])
start = time.time()
adam = keras.optimizers.Adam(lr=0.01)
sgd = keras.optimizers.SGD(lr=0.01,momentum=0.9)
# compile the model to make a use of categorical cross-entropy loss function and sgd optimizer to get high accuracy.
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
model.fit(train_generator,verbose=1,epochs=150,shuffle=True)


########################################################## Results ################################################
score = model.evaluate(test_generator)
stop = time.time()
print(f"Training accuracy time with Alexnet model from scratch with Caltech101: {stop - start}s")
print('Test accuracy:', (score[1]*100))
```

- **ALEXNET with SCENE-15 From Scratch:**

```
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
from sklearn.preprocessing import LabelBinarizer
import cv2
from keras.optimizers import Adam,SGD
import os,keras
```

```
import time

########################################  Preprocessing of data
########################################

# load the dataset

path = '15-Scene'

categories = sorted(os.listdir(path))

ncategories = len(categories)


# Dividing First 100 images are considered as Training and rest as testing

data_train = []

labels_train = []

data_test = []

labels_test = []


for i, category in enumerate(categories):

   counter = 0;

         #np.random.shuffle(categories)

   for f in os.listdir(path + "/" + category):

      ext = os.path.splitext(f)[1]

      fullpath = os.path.join(path + "/" + category, f)

      label = fullpath.split(os.path.sep)[-2]

      image = cv2.imread(fullpath)

      image = cv2.resize(image, (224, 224))

      counter = counter + 1

      if (counter <= 100):

         data_train.append(image)

         labels_train.append(label)

      else:

         data_test.append(image)

         labels_test.append(label)
```

```
print ('First 100 images per class are considered as Training and rest as testing')
# Normalizing the data
x_train = np.array(data_train, dtype="float") / 255.0
x_test = np.array(data_test, dtype="float") / 255.0
del data_train
del data_test
# LabelBinarizer
lb = LabelBinarizer()
y_train = lb.fit_transform(labels_train)
y_test = lb.fit_transform(labels_test)
del labels_train
del labels_test
print("Data Splitted")
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
############################################## Model Implementation
############################################################
# Data augmentation
gen = ImageDataGenerator(width_shift_range=.2, # image augmentation using kearas
imagedatagenerator function
                         height_shift_range=.2,
                        zoom_range=0.2)

test_gen = ImageDataGenerator()
train_generator = gen.flow(x_train, y_train, batch_size=64)
test_generator = test_gen.flow(x_test, y_test, batch_size=64)
# Alexnet model with 5 convolution layer, 2 fully connected layer and 1 output layer.
```

```
model = keras.models.Sequential([
    keras.layers.Conv2D(filters=96, kernel_size=(11,11), strides=(4,4), activation='relu', input_shape=(224,224,3)),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(3,3), strides=(2,2)),
    keras.layers.Conv2D(filters=256, kernel_size=(5,5), strides=(1,1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(3,3), strides=(2,2)),
    keras.layers.Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2D(filters=384, kernel_size=(1,1), strides=(1,1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2D(filters=256, kernel_size=(1,1), strides=(1,1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(3,3), strides=(2,2)),
    keras.layers.Flatten(),
    keras.layers.Dense(4096, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(4096, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(15, activation='softmax')
])

start = time.time()
adam = keras.optimizers.Adam(lr=0.01)
sgd = keras.optimizers.SGD(lr=0.01,momentum=0.9)
```

```
# compile the model to make a use of categorical cross-entropy loss function and sgd optimizer to get high accuracy.

model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

# fitiing the model.

model.fit(train_generator,verbose=1,epochs=150,shuffle=True)

######################################################## Results ################################################

score = model.evaluate(test_generator)

stop = time.time()

print(f"Training accuracy time with Alexnet model from scratch with Scene-15 : {stop - start}s")

print('Test accuracy:', (score[1]*100))
```

- **(2 points) compare the performance gab and training time between the above two conditions. Explain why you obtained such results, and give a brief discussion about it.**

Ans:

- As we know the performance of the transfer learning is much more accurate and faster than the training from the scratch.
- Because In transfer learning instead of training all the layers of the model we lock some of the layers and use those trained weights in the locked layers to extract particular features from our data.
- Using transfer learning I am getting high accuracy in less training time with less epoch whereas in training from scratch it gives less accuracy with so much time and epoch.
- Mainly for scene 15,
- Here are my results:

Scene-15 from scratch with 1 run of 150 epoch

Training time : 1631.322796344757s

Test accuracy : 56.8844199180603

Caltech 101 from scratch with 1 run of 150 epoch

Training time : 3870.557685259752s

Test accuracy: 68.44207060337067

Scene-15 (pretrained) with 3 run of 10 epoch

Training time : 986.7564839674884s

Test accuracy : 79.61473822593689

Caltech 101(pretrained) with 3 run of 10 epoch

Training time : 1284.57584878368s

Test accuracy: 86.43878269195557