**Practical Assignment**

**Objective: - Image Classification with Indoor Scenes Images**

Created by Intel for an image classification contest, this expansive image dataset contains approximately 25,000 images. Furthermore, the images are divided into the following categories: buildings, forest, glacier, mountain, sea, and street. The dataset has been divided into folders for training, testing, and prediction. The training folder includes around 14,000 images and the testing folder has around 3,000 images. Finally, the prediction folder includes around 7,000 images.

**Dataset Link: -**

[**https://www.kaggle.com/puneet6060/intel-image-classification/version/2**](https://www.kaggle.com/puneet6060/intel-image-classification/version/2)

**Task: -** Create a Web Application using Flask. Use the end user should be able to upload an image and get results with the prediction score. Use any CNN architecture launched after 2015.

**Deployment: -** Any Free Platform(Try to look out for free options.)

**Assignment Submission: -** Only submit the hosted app link.

**Classification of Intel image dataset with AlexNet**

## 1 - Packages

**import** numpy **as** np

**import** pickle

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** matplotlib.image **as** mpimg

**import** os

**import** cv2

**from** image\_data\_generator **import** image\_data\_generator\_train, image\_data\_generator\_valid, image\_data\_generator\_test

**import** tensorflow **as** tf

**from** tensorflow **import** keras

**from** keras **import** layers

**from** keras.preprocessing.image **import** ImageDataGenerator

**from** keras.models **import** Sequential

**from** keras.optimizers **import** Adam

**from** keras.layers **import** Dense, Conv2D, MaxPool2D, Activation, Flatten, Dropout

## 2 - Generating data for Training, Validation and Testing

*# Preprocessing and augmenting datasets*

train\_generator **=** image\_data\_generator\_train('Data/Train')

valid\_generator **=** image\_data\_generator\_valid('Data/Valid')

test\_generator **=** image\_data\_generator\_test('Data/Test')

## 3 - Creating the AlexNet model

*# Creating AlexNet network*

**def** model(input\_shape):

model **=** keras**.**Sequential()

*# Layer 1*

model**.**add(Conv2D(96, (11, 11), strides**=**4, input\_shape **=** input\_shape))

model**.**add(Activation('relu'))

model**.**add(MaxPool2D(pool\_size**=**(3, 3), strides**=**2))

*# Layer 2*

model**.**add(Conv2D(256, (5, 5), strides**=**1, padding**=**'same'))

model**.**add(Activation('relu'))

model**.**add(MaxPool2D(pool\_size**=**(3, 3), strides**=**2))

*# Layer 3, 4, and 5*

model**.**add(Conv2D(384, (3, 3), strides**=**1, padding**=**'same'))

model**.**add(Activation('relu'))

model**.**add(Conv2D(384, (3, 3), strides**=**1, padding**=**'same'))

model**.**add(Activation('relu'))

model**.**add(Conv2D(256, (3, 3), strides**=**1, padding**=**'same'))

model**.**add(Activation('relu'))

model**.**add(MaxPool2D(pool\_size**=**(3, 3), strides**=**2))

*# Layer 6*

model**.**add(Flatten())

model**.**add(Dense(4096))

model**.**add(Activation('relu'))

*# Layer 7*

model**.**add(Dropout(0.5))

model**.**add(Dense(4096))

model**.**add(Activation('relu'))

*# Layer 8*

model**.**add(Dropout(0.5))

model**.**add(Dense(6, activation**=**'softmax'))

opt **=** Adam(learning\_rate**=**0.0001)

model**.**compile(optimizer**=**opt,

loss**=**'categorical\_crossentropy',

metrics**=**['accuracy'])

**return** model

model **=** model((150, 150, 3))

model**.**summary()

## 4 - Training the model

*# Defining step sizes for training and validation*

STEP\_SIZE\_TRAIN**=**train\_generator**.**n**//**train\_generator**.**batch\_size

STEP\_SIZE\_VALID**=**valid\_generator**.**n**//**valid\_generator**.**batch\_size

*# Training the model*

history **=** model**.**fit(train\_generator, steps\_per\_epoch**=**STEP\_SIZE\_TRAIN,

validation\_data**=**valid\_generator, validation\_steps**=**STEP\_SIZE\_VALID, epochs**=**10)

*# Saving the model*

model**.**save('Pretrained\AlexNet\_model\_V2.h5')

## 5 - Making Predictions

*# Loading the trained model*

pretrained\_model **=** keras**.**models**.**load\_model('Pretrained/AlexNet\_model\_V2.h5')

*# Loading image directory*

img\_dir **=** "Data/Pred"

*# Preprocessing images*

pred\_generator **=** image\_data\_generator\_test(img\_dir)

*# Making predictions*

prediction **=** pretrained\_model**.**predict(pred\_generator, verbose**=**1)

predicted\_class\_indices **=** np**.**argmax(prediction, axis**=**1)

*# Classifying predictions*

labels **=** (train\_generator**.**class\_indices)

labels **=** dict((v,k) **for** k,v **in** labels**.**items())

predicted\_class\_indices

predictions **=** [labels[k] **for** k **in** predicted\_class\_indices]

print("Predictions: ")

*# Create a figure with 6 subplots*

fig, axs **=** plt**.**subplots(1, 4, figsize**=**(16, 10))

*# Plot the data in each subplot*

**for** image, i **in** zip(os**.**listdir(img\_dir**+**'/Pred\_folder'), range(len(os**.**listdir(img\_dir**+**'/Pred\_folder')))):

img **=** cv2**.**imread(os**.**path**.**join(img\_dir**+**'/Pred\_folder', image))

axs[i]**.**imshow(img)

axs[i]**.**set\_title(predictions[i])

*# Adjust the spacing between subplots*

plt**.**subplots\_adjust(wspace**=**0.3, hspace**=**0.3)

*# Show the plot*

plt**.**show()

## 6 - Analysing Results

*# Plotting loss and accuracy*

**import** warnings

warnings**.**filterwarnings('ignore')

df\_loss\_acc **=** pd**.**DataFrame(history**.**history)

df\_loss**=** df\_loss\_acc[['loss','val\_loss']]

df\_loss**.**rename(columns**=**{'loss':'train','val\_loss':'validation'},inplace**=True**)

df\_acc**=** df\_loss\_acc[['accuracy','val\_accuracy']]

df\_acc**.**rename(columns**=**{'accuracy':'train','val\_accuracy':'validation'},inplace**=True**)

df\_loss**.**plot(title**=**'Model loss',figsize**=**(12,8))**.**set(xlabel**=**'Epoch',ylabel**=**'Loss')

df\_acc**.**plot(title**=**'Model Accuracy',figsize**=**(12,8))**.**set(xlabel**=**'Epoch',ylabel**=**'Accuracy')