Configuration Manual

MSc Research Project

Data Analytics

Asim Arif Ibushe

Student ID: x21172218

School of Computing

National College of Ireland

Supervisor: Mr. Bharat Agarwal

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	…Asim Ibushe……………………………………………………………………………..
Student ID:	…x21172218……………………………………………………………………….……….
Programme:	…MSc in Data Analytics……………………… **Year:** ………2023…………
Module:	…Research Project..…………………………………………………………….………..
Lecturer:	…Mr. Bharat Agarwal……..…………………………………………………….…………
Submission Due Date:	…25/04/2024……………………………………………………………………….……….
Project Title:	…………Document Image Classification using Convolutional Neural Network and Transfer Learning Technique……………………….………………
Word Count:	……………643……………… **Page Count:** …………9…………….…….…………….

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……Asim Ibushe……………………………………………………………………..

Date: ……25/04/2024………………………………………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Asim Ibushe
Student ID: x21172218

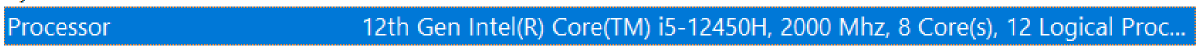
1 Introduction:

This configuration manual describes the specification of all the software and hardware used in our research project. Each of the hardware and software versions, rating, platform and other technical information is provided in this manual.

2 System Configuration:

2.1 Hardware configuration:

Personal Windows machine is used for all the implementation of the research. Specification of the machine is mentioned in the table below.



Processor 12th Gen Intel(R) Core(TM) i5-12450H, 2000 Mhz, 8 Core(s), 12 Logical Proc...

Fig.1 Processor and Core

Google Colab Pro configuration for processing: P100 GPU 16 GB

2.2 Software configuration:

Different software and dependencies which were used in this research all of these are listed in below table along with the versions.

Anaconda and Jupyter notebook:

Anaconda 3 includes a free web tool called Jupyter Notebook that lets you create documents with code, equations, graphics, and tests. The Jupyter notebook was used for the whole implementation process, which included all visualizations, model development, training, testing, and assessment.

Google Colab Pro:
Utilizing Google Colab Pro with a P100 GPU 16 GB resulted in a runtime of approximately 1500 seconds, or roughly 29 to 30 minutes of execution time

Python Libraries:

Several python libraries were used in the research for various tasks. This library information is given in below table.

Library	Version
OpenCV	4.6.0
Numpy	1.22.4
Scikit learn	1.4.0
matplotlib	3.8.0
pandas	2.1.1
Tensorflow	V2.16.1
Keras	3.3.1

Table. 1

3 Research Implementation Code:

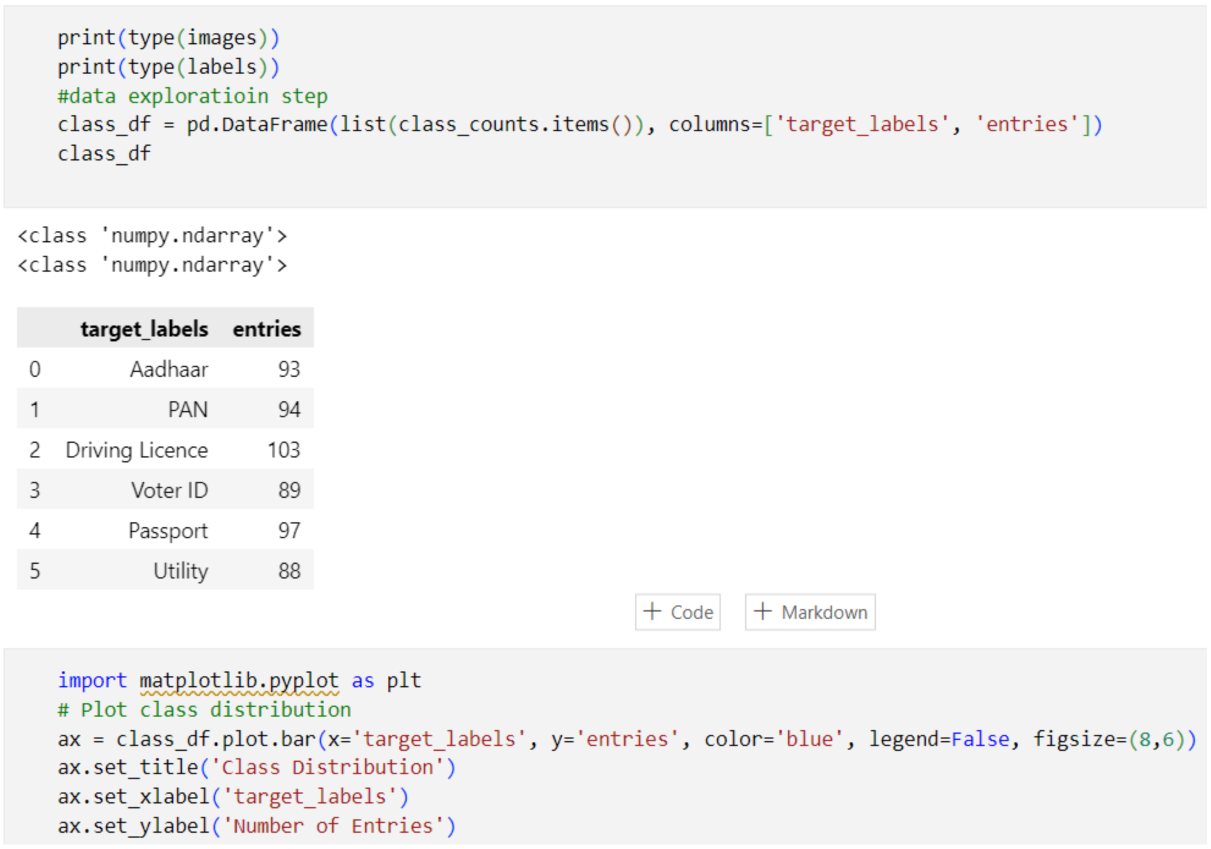
3.1 Dataset 1 Loading and Pre-processing:

```
dataset_folder=r"I:\NCI_Sem3\Document_Dataset"
#initializing all target variables
class_labels = ["Aadhaar", "PAN", "Driving Licence", "Voter ID", "Passport", "Utility"]
```

```
def preprocess_images(folder_path):
    images = []
    labels = []
    class_count = {}

    for i, label in enumerate(class_labels):
        class_count[label] = 0
        folder = os.path.join(folder_path, label)
        for filename in os.listdir(folder):
            img_path = os.path.join(folder, filename)
            img = cv2.imread(img_path)
            if img is not None and img.size != 0:
                img = cv2.resize(img, (224, 224))
                img = img / 255.0
                images.append(img)
                labels.append(i)
                class_count[label] += 1

    return np.array(images), np.array(labels), class_count
```

```
# Call the function to preprocess images and get class counts
images, labels,class_counts= preprocess_images(dataset_folder)
```

Fig. 2: Image Dataset Directory Traversal and Pre-processing module



```
print(type(images))
print(type(labels))
#data exploratioin step
class_df = pd.DataFrame(list(class_counts.items()), columns=['target_labels', 'entries'])
class_df
```

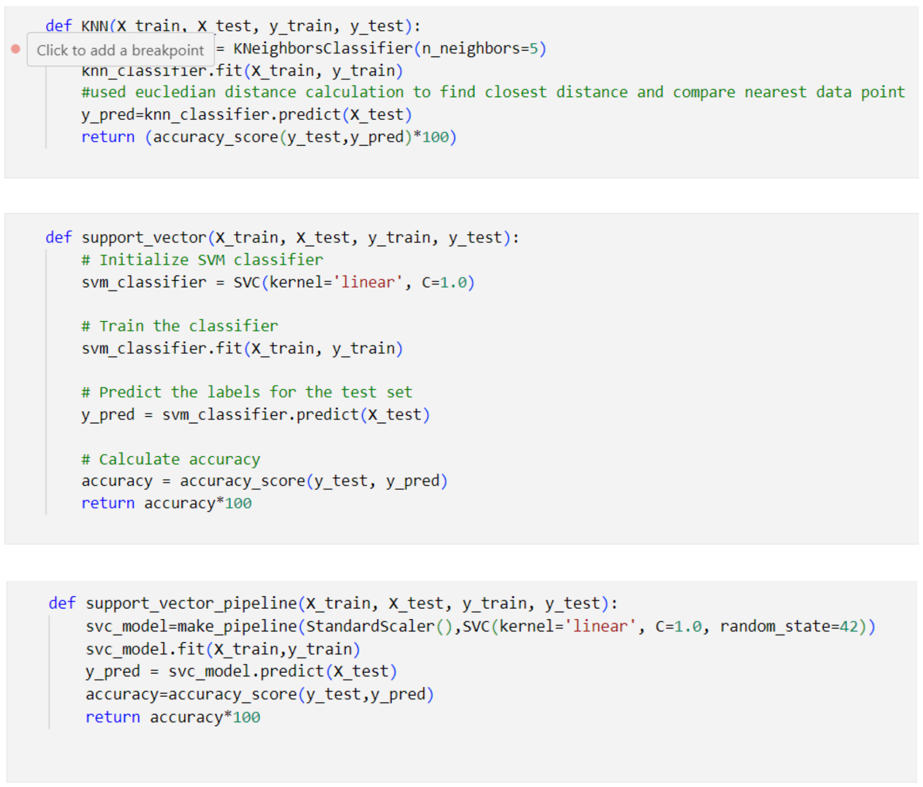
```
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

	target_labels	entries
0	Aadhaar	93
1	PAN	94
2	Driving Licence	103
3	Voter ID	89
4	Passport	97
5	Utility	88

```
import matplotlib.pyplot as plt
# Plot class distribution
ax = class_df.plot.bar(x='target_labels', y='entries', color='blue', legend=False, figsize=(8,6))
ax.set_title('Class Distribution')
ax.set_xlabel('target_labels')
ax.set_ylabel('Number of Entries')
```

Fig. 3: Image dataset exploration using matplotlib

Fig 2 and Fig 3 shows the python code for dataset loading and data preprocessing and exploring class distribution.

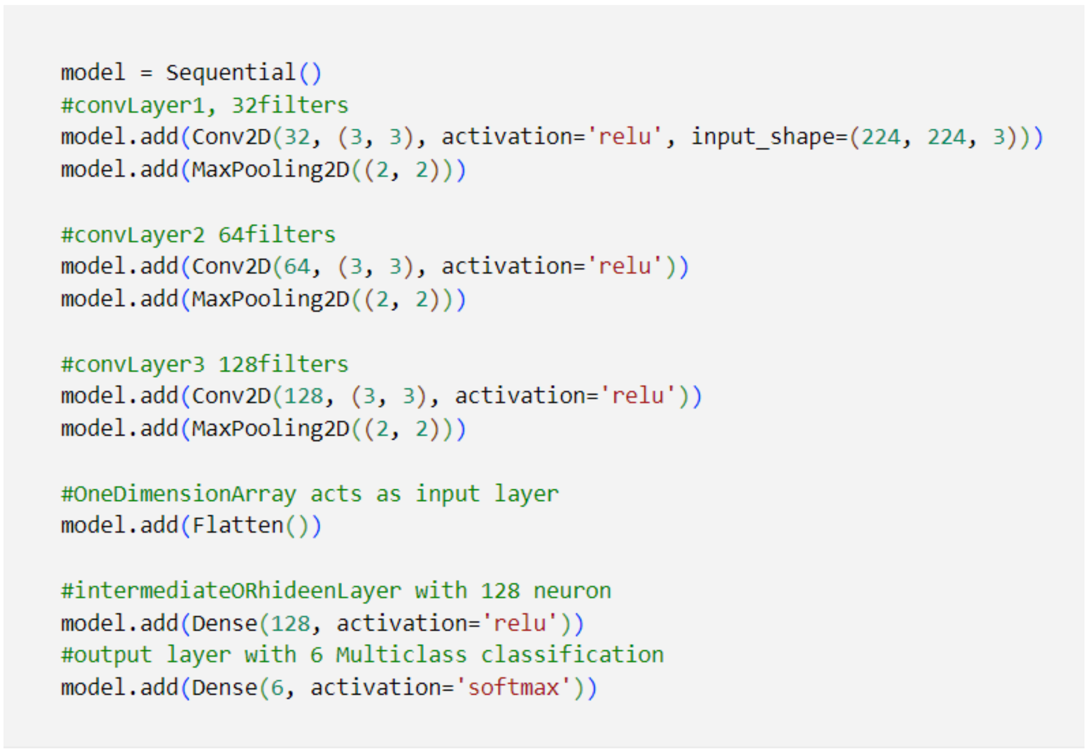


```
def KNN(X_train, X_test, y_train, y_test):
                   = KNeighborsClassifier(n_neighbors=5)
    knn_classifier.fit(X_train, y_train)
    #used eucledian distance calculation to find closest distance and compare nearest data point
    y_pred=knn_classifier.predict(X_test)
    return (accuracy_score(y_test,y_pred)*100)
```

```
def support_vector(X_train, X_test, y_train, y_test):
    # Initialize SVM classifier
    svm_classifier = SVC(kernel='linear', C=1.0)

    # Train the classifier
    svm_classifier.fit(X_train, y_train)

    # Predict the labels for the test set
    y_pred = svm_classifier.predict(X_test)

    # Calculate accuracy
    accuracy = accuracy_score(y_test, y_pred)
    return accuracy*100
```

```
def support_vector_pipeline(X_train, X_test, y_train, y_test):
    svc_model=make_pipeline(StandardScaler(),SVC(kernel='linear', C=1.0, random_state=42))
    svc_model.fit(X_train,y_train)
    y_pred = svc_model.predict(X_test)
    accuracy=accuracy_score(y_test,y_pred)
    return accuracy*100
```

Fig. 4: Traditional ML Implementation

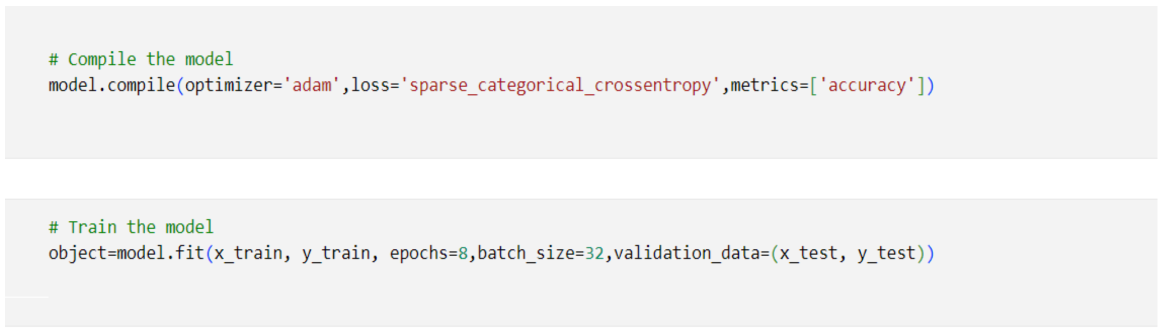


```
model = Sequential()
#convLayer1, 32filters
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)))
model.add(MaxPooling2D((2, 2)))

#convLayer2 64filters
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

#convLayer3 128filters
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

#OneDimensionArray acts as input layer
model.add(Flatten())

#intermediateORhideenLayer with 128 neuron
model.add(Dense(128, activation='relu'))
#output layer with 6 Multiclass classification
model.add(Dense(6, activation='softmax'))
```

Fig. 5: Designing Convolutional Architecture for Dataset 1

Fig. 5 shows how CNN architecture is designed with filter and pooling for feature extraction which helps for dimensionality reduction and image normalization.

3.2 Dataset 1 Model training step:

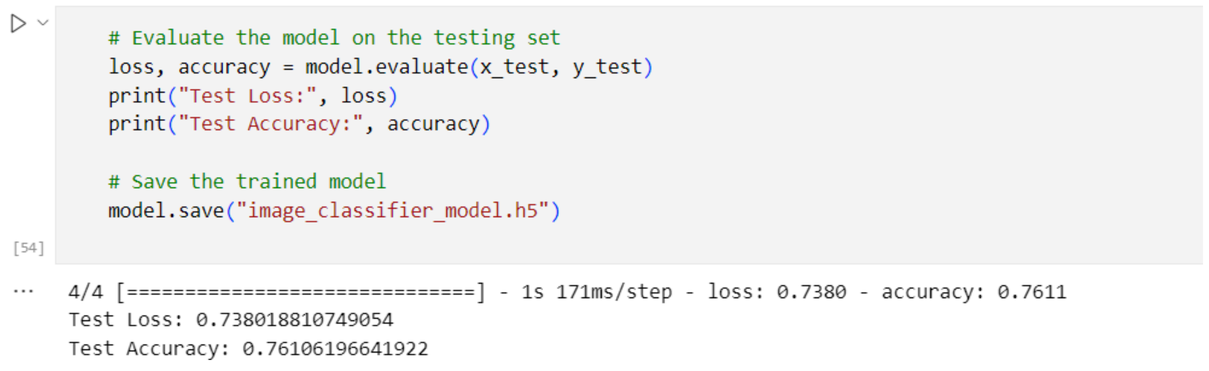


```
# Compile the model
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

```
# Train the model
object=model.fit(x_train, y_train, epochs=8,batch_size=32,validation_data=(x_test, y_test))
```

Fig. 6: Training Phase and deciding hyper-parameter epoch value which optimizer to use.

Fig. 6 shows the code written for compiling and model training phase, where both training and validation dataset are passed as an argument for self-feature extraction and weight tunning.



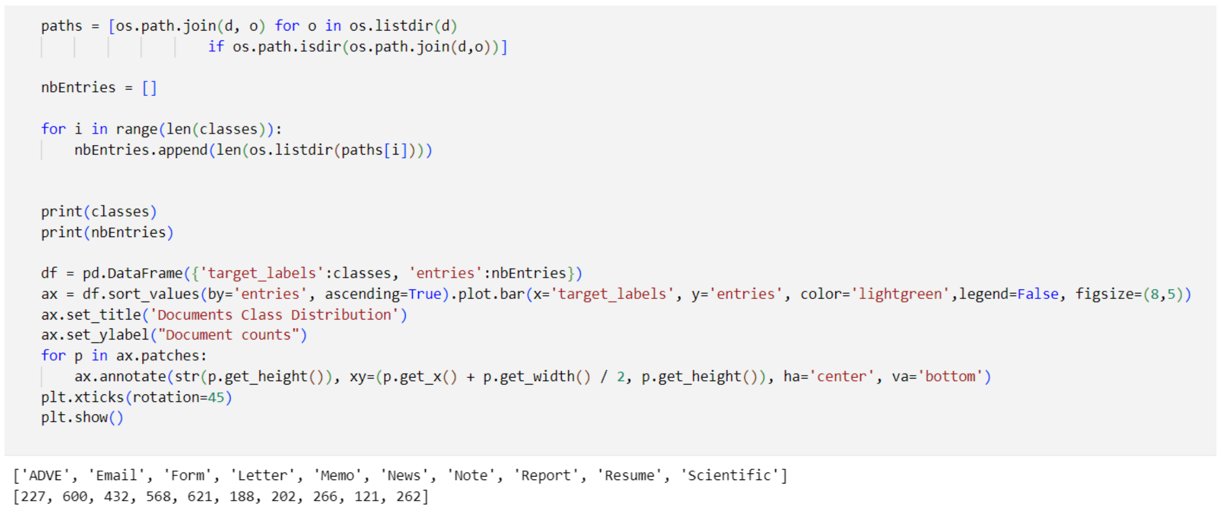
```
# Evaluate the model on the testing set
loss, accuracy = model.evaluate(x_test, y_test)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)

# Save the trained model
model.save("image_classifier_model.h5")
```

```
[54]
```

```
4/4 [==============================] - 1s 171ms/step - loss: 0.7380 - accuracy: 0.7611
Test Loss: 0.738018810749054
Test Accuracy: 0.76106196641922
```

Fig. 7: Model validation and save network architecture for future inheritance.

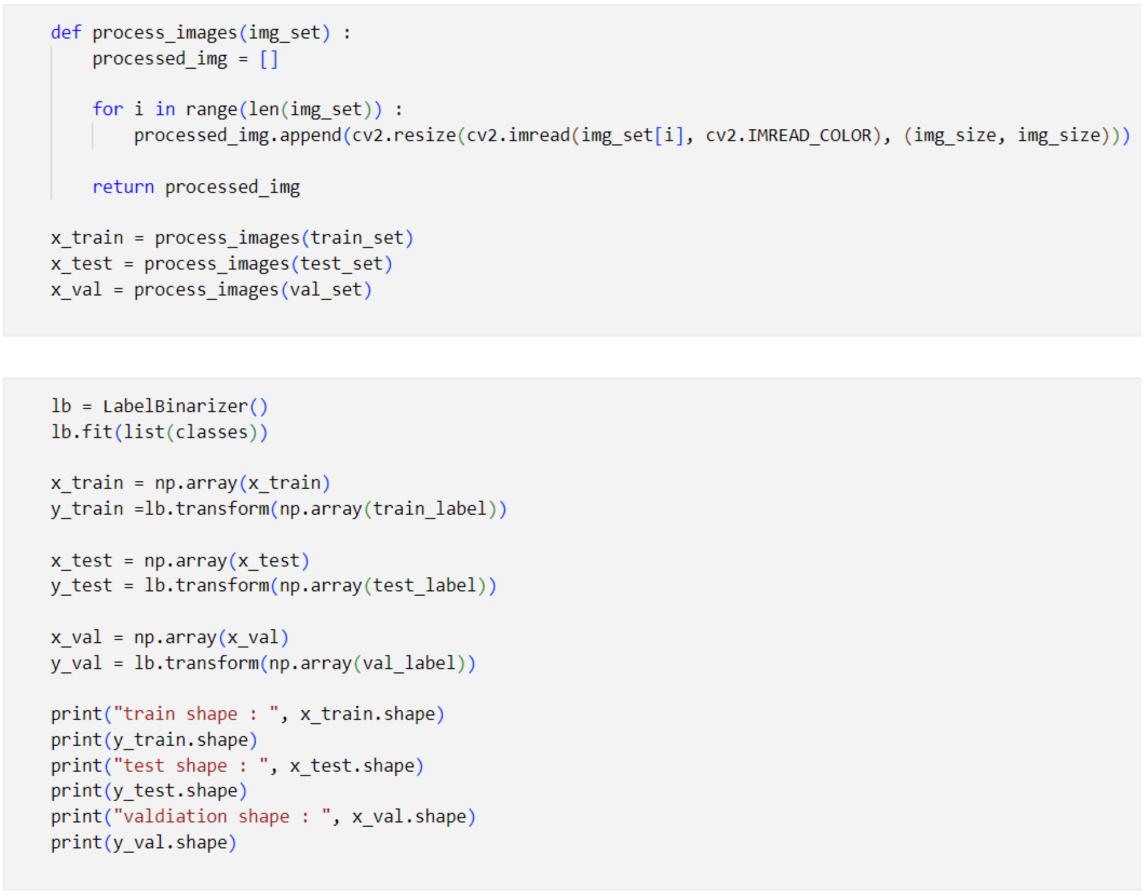
3.3 Dataset 2 Loding and Exploratory data analysis:



```
paths = [os.path.join(d, o) for o in os.listdir(d)
                    if os.path.isdir(os.path.join(d,o))]

nbEntries = []

for i in range(len(classes)):
    nbEntries.append(len(os.listdir(paths[i])))


print(classes)
print(nbEntries)

df = pd.DataFrame({'target_labels':classes, 'entries':nbEntries})
ax = df.sort_values(by='entries', ascending=True).plot.bar(x='target_labels', y='entries', color='lightgreen',legend=False, figsize=(8,5))
ax.set_title('Documents Class Distribution')
ax.set_ylabel("Document counts")
for p in ax.patches:
    ax.annotate(str(p.get_height()), xy=(p.get_x() + p.get_width() / 2, p.get_height()), ha='center', va='bottom')
plt.xticks(rotation=45)
plt.show()
```

```
['ADVE', 'Email', 'Form', 'Letter', 'Memo', 'News', 'Note', 'Report', 'Resume', 'Scientific']
[227, 600, 432, 568, 621, 188, 202, 266, 121, 262]
```

Fig. 8   Importing all images for dataset2, exploring class distribution

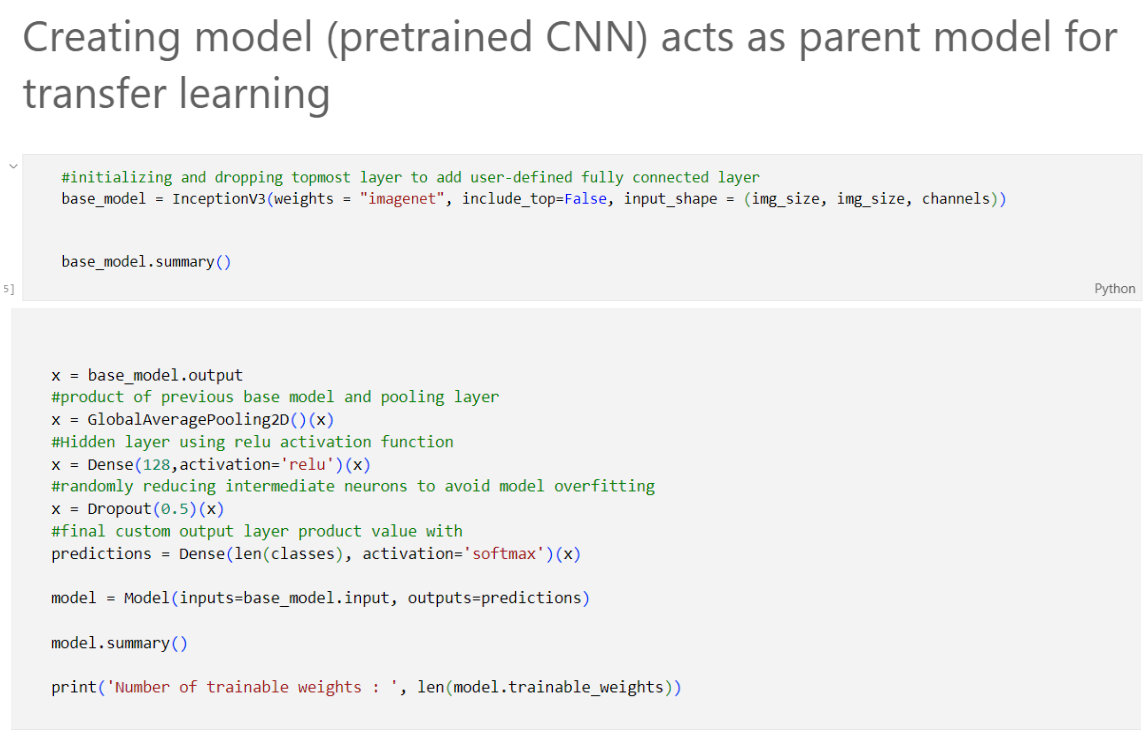


```python
def process_images(img_set) :
    processed_img = []

    for i in range(len(img_set)) :
        processed_img.append(cv2.resize(cv2.imread(img_set[i], cv2.IMREAD_COLOR), (img_size, img_size)))

    return processed_img

x_train = process_images(train_set)
x_test = process_images(test_set)
x_val = process_images(val_set)
```

```python
lb = LabelBinarizer()
lb.fit(list(classes))

x_train = np.array(x_train)
y_train =lb.transform(np.array(train_label))

x_test = np.array(x_test)
y_test = lb.transform(np.array(test_label))

x_val = np.array(x_val)
y_val = lb.transform(np.array(val_label))

print("train shape : ", x_train.shape)
print(y_train.shape)
print("test shape : ", x_test.shape)
print(y_test.shape)
print("valdiation shape : ", x_val.shape)
print(y_val.shape)
```

Fig. 9 Fit Transformation and Image preprocessing before training phase

3.4 Dataset 2: Transfer learning implementation



Creating model (pretrained CNN) acts as parent model for transfer learning

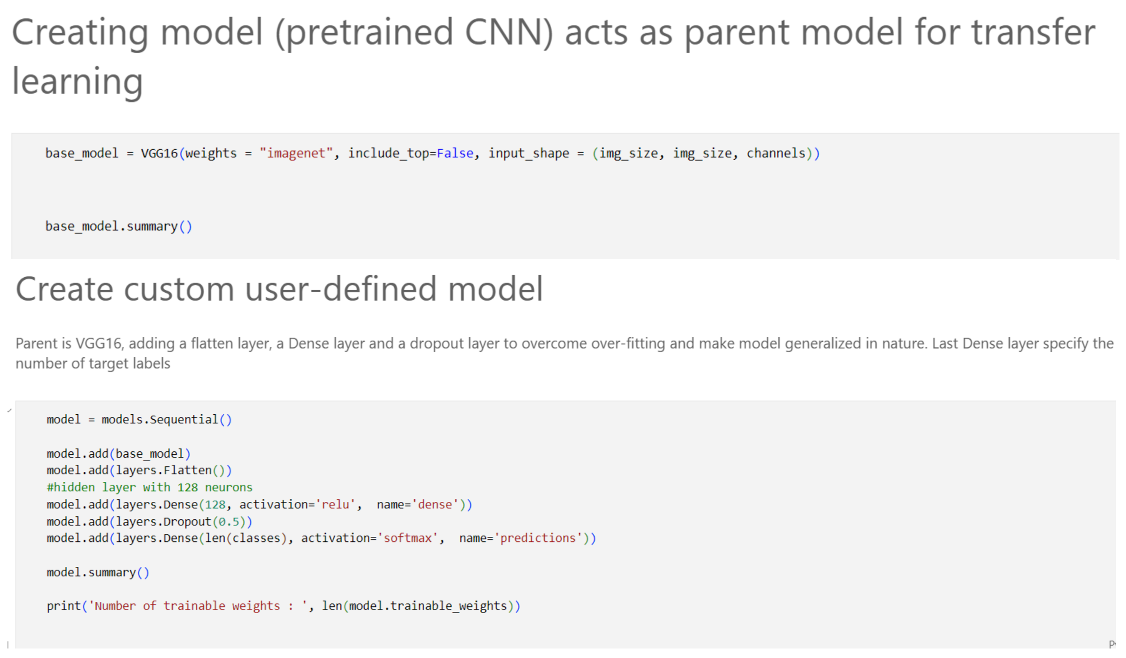
```
#initializing and dropping topmost layer to add user-defined fully connected layer
base_model = InceptionV3(weights = "imagenet", include_top=False, input_shape = (img_size, img_size, channels))


base_model.summary()
```

5]

Python

```
x = base_model.output
#product of previous base model and pooling layer
x = GlobalAveragePooling2D()(x)
#Hidden layer using relu activation function
x = Dense(128,activation='relu')(x)
#randomly reducing intermediate neurons to avoid model overfitting
x = Dropout(0.5)(x)
#final custom output layer product value with
predictions = Dense(len(classes), activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)

model.summary()

print('Number of trainable weights : ', len(model.trainable_weights))
```

Fig. 10 InceptionV3 implementation using ImageNet dataset weights



Creating model (pretrained CNN) acts as parent model for transfer learning

```
base_model = VGG16(weights = "imagenet", include_top=False, input_shape = (img_size, img_size, channels))


base_model.summary()
```

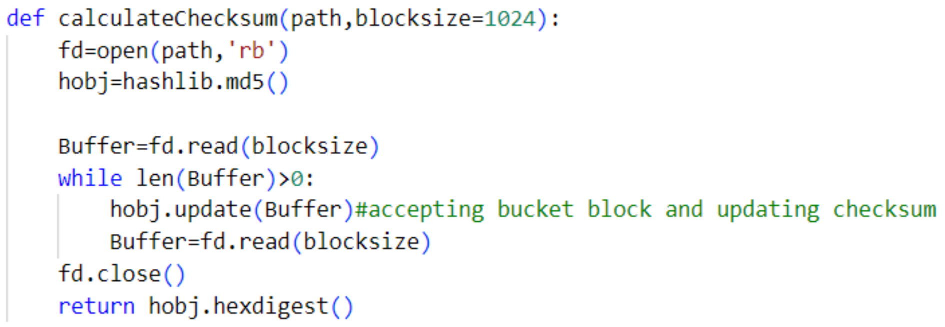
Create custom user-defined model

Parent is VGG16, adding a flatten layer, a Dense layer and a dropout layer to overcome over-fitting and make model generalized in nature. Last Dense layer specify the number of target labels

```
model = models.Sequential()

model.add(base_model)
model.add(layers.Flatten())
#hidden layer with 128 neurons
model.add(layers.Dense(128, activation='relu',  name='dense'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(len(classes), activation='softmax',  name='predictions'))

model.summary()

print('Number of trainable weights : ', len(model.trainable_weights))
```

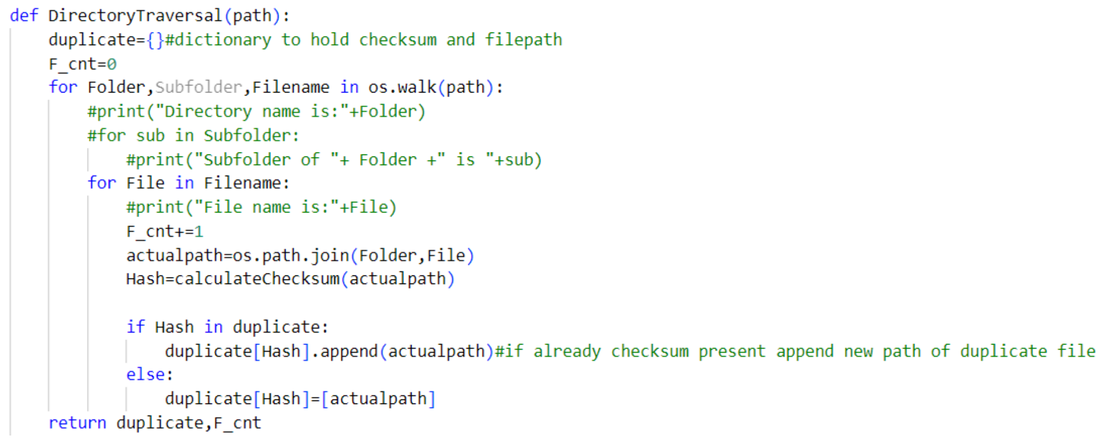
Fig. 11 VGG16 implementation using ImageNet dataset weights

3.5 Python Script for Image Data De-duplication



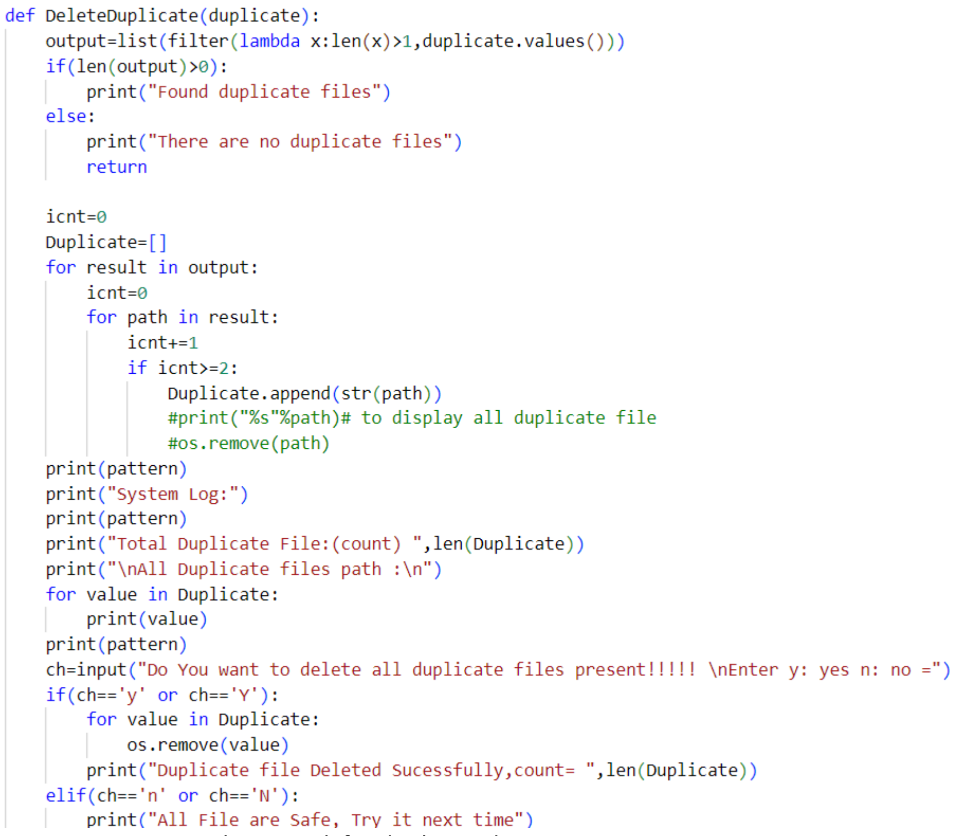
```python
def calculateChecksum(path,blocksize=1024):
    fd=open(path,'rb')
    hobj=hashlib.md5()

    Buffer=fd.read(blocksize)
    while len(Buffer)>0:
        hobj.update(Buffer)#accepting bucket block and updating checksum
        Buffer=fd.read(blocksize)
    fd.close()
    return hobj.hexdigest()
```

Fig. 11 message digest calculating hash value for each document image



```python
def DirectoryTraversal(path):
    duplicate={}#dictionary to hold checksum and filepath
    F_cnt=0
    for Folder,Subfolder,Filename in os.walk(path):
        #print("Directory name is:"+Folder)
        #for sub in Subfolder:
            #print("Subfolder of "+ Folder +" is "+sub)
        for File in Filename:
            #print("File name is:"+File)
            F_cnt+=1
            actualpath=os.path.join(Folder,File)
            Hash=calculateChecksum(actualpath)

            if Hash in duplicate:
                duplicate[Hash].append(actualpath)#if already checksum present append new path of duplicate file
            else:
                duplicate[Hash]=[actualpath]
    return duplicate,F_cnt
```

Fig. 12 Directory Traversal



```python
def DeleteDuplicate(duplicate):
    output=list(filter(lambda x:len(x)>1,duplicate.values()))
    if(len(output)>0):
        print("Found duplicate files")
    else:
        print("There are no duplicate files")
        return

    icnt=0
    Duplicate=[]
    for result in output:
        icnt=0
        for path in result:
            icnt+=1
            if icnt>=2:
                Duplicate.append(str(path))
                #print("%s"%path)# to display all duplicate file
                #os.remove(path)
    print(pattern)
    print("System Log:")
    print(pattern)
    print("Total Duplicate File:(count) ",len(Duplicate))
    print("\nAll Duplicate files path :\n")
    for value in Duplicate:
        print(value)
    print(pattern)
    ch=input("Do You want to delete all duplicate files present!!!!! \nEnter y: yes n: no =")
    if(ch=='y' or ch=='Y'):
        for value in Duplicate:
            os.remove(value)
        print("Duplicate file Deleted Sucessfully,count= ",len(Duplicate))
    elif(ch=='n' or ch=='N'):
        print("All File are Safe, Try it next time")
```

Fig. 11 Dictionary to keep track for duplicate document image