A

MAJOR PROJECT

On

Detecting Disguised Faces With Transfer Learning

(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

Abhinav Joel T (187R1A05C1)

Daphedari Kishan Prasad (187R1A05D5)

Adabala Taraka Rama Venkata Sai Hanuman (187R1A05C2)

Under the Guidance of

Dr. A. PRABHU

(Associate Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi) Recognized Under Section 2(f) & 12(B) of the UGC Act.1956,

Kandlakoya (V), Medchal Road, Hyderabad-501401.

2018-2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled " **Detecting Disguised Faces With Transfer Learning "** being submitted by **ABHINAV JOEL T(187R1A05C1) DAPHEDARI KISHAN PRASAD(187R1A05D5),ADABALA TARAKA RAMA VENKATA SAI HANUMAN(187R1A05C2)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by him/her under our guidance and supervision during the year 2021-22.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Dr. A. Prabhu
Associate Professor
INTERNAL GUIDE

Dr. A. Raji Reddy
DIRECTOR

Dr. K. Srujan Raju
Head Of The Department

EXTERNAL EXAMNIER

Submitted for viva voice Examination held on _____

ACKNOWLEGDEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide **Dr. A. Prabhu**, Associate Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark. We also take this opportunity to express a deep sense of gratitude to Project Review Committee (PRC) **Mr. J. Narasimha Rao, Dr. T. S. Mastan Rao, Mr. A. Uday Kiran, Mr. A. Kiran Kumar, Mrs. G. Latha** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju**, Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy**, Director for being cooperative throughout the course of this project. We also express our sincere gratitude to **Sri. Ch. Gopal Reddy**, Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, We would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

Adabala Taraka Rama Venkata Sai Hanuman (187R1A05C2)

Daphedari Kishan Prasad (187R1A05D5)

Abhinav Joel T (187R1A05C1)

ABSTRACT

In general, a human being has the memory power due to which he/she will be able to remember whatever they have seen but as the technology is increasing the advancement is increasing in such a way that now computer is also able to recognize theX faces from its memory but in order to differentiate them, we need more advancement which leads to the development of Machine learning. Machine learning concepts developed by Arthur Manuel. There have been many techniques used over the past decade to determine the identity of a person's face, such as Eigenfaces and Principal Component Analysis (PCA), to Convolutional Neural Networks (CNN) to ensure the ability to recognize faces has become further and further. An approach to machine learning called transfer learning involves creating a model of the first training task, then testing it using the model. The difference between transfer learning and traditional machine learning is that translation involves using a pre-trained model in order to start a secondary task using the initial model. It is expected that this paper will contribute to the field of image classification by using Machine Learning algorithms to solve the problem. Transfer learning significantly improves the performance of VGG models Based on these results, we conclude that VGG Models are the best choice for recognizing faces using ImageNet weight.

LIST OF FIGURES/TABLES

FIGURE NO	FIGURE NAME	PAGE NO
Figure 3.1	Project Architecture	6
Figure 3.3	Use Case Diagram	8
Figure 3.4	Sequence Diagram	9
Figure 3.5	Activity Diagram	10
Figure 3.6	Class Diagram	11

LIST OF SCREENSHOTS

SCREENSHOT NO	SCREENSHOT NAME	PAGE NO
Screenshot 5.1	Output-1	27
Screenshot 5.2	Output-2	27
Screenshot 5.3	Output-3	28
Screenshot 5.4	Output-4	28

TABLE OF CONTENTS

ABSTRACT i

LIST OF FIGURES ii
LIST OF SCREENSHOTS iii

1. INTRODUCTION 1

1.1 PROJECT SCOPE 1

1.2 PROJECT PURPOSE 1

1.3 PROJECT FEATURES 1

2. SYSTEM ANALYSIS 2

2.1 PROBLEM DEFINITION 2

2.2 EXISTING SYSTEM 2

2.2.1 LIMITATIONS OF EXISTING SYSTEM 3

2.3 PROPOSED SYSTEM 3

2.3.1 ADVANTAGES OF PROPOSED SYSTEM 3

2.4 FEASIBILITY STUDY 4

2.4.1 ECONOMIC FEASIBILITY 4

2.4.2 TECHNICAL FEASIBILITY 4

2.4.3 SOCIAL FEASIBILITY 5

2.5 HARDWARE & SOFTWARE REQUIREMENTS 5

2.5.1 HARDWARE REQUIREMENTS 5

2.5.2 SOFTWARE REQUIREMENTS 5

3. ARCHITECTURE 6

3.1 PROJECT ARCHITECTURE 6

3.2 MODULES DESCRIPTION 6

3.2.1 USER 6

3.2.2 VGG16 7

3.2.3 ResNet50 7

3.2.4 InceptionV3 7

3.3 USECASE DIAGRAM 8

3.4 SEQUENCE DIAGRAM 9

3.5 ACTIVITY DIAGRAM 10

3.6 CLASS DIAGRAM 11

4.IMPLEMENTATION 18

4.1 SAMPLE CODE 19

5.SCREEN SHOTS 27

6.TESTING 29

6.1 INTRODUCTION TO TESTING 29

6.2 TYPES OF TESTING 29

6.2.1 UNIT TESTING 29

6.2.2 INTEGRATION TESTING 29

6.2.3 FUNCTIONAL TESTING 30

6.3 TEST CASES 31

7.CONCLUSION & FUTURE SCOPE 32

7.1 PROJECT CONCLUSION 32

7.2 FUTURE SCOPE 32

8.REFERENCES 33

8.1 GITHUB REPOSITORY LINK 33

8.2 REFERENCES 33

9.JOURNAL

1. INTRODUCTION

1.INTRODUCTION

1.1 PROJECT SCOPE

In past decade, numerous approaches method to identifying person's faces which is Eigenfaces and Principal Component Analysis (PCA), to Convolutional Neural Networks (CNN) which then after that, the ability to recognize face became higher and higher. Transfer learning is an approach used in machine learning where the first training task produces a model, then we do the second test using the model of the first training task. Transfer learning differs from traditional machine learning because it involves using a pre-trained model as a springboard to start a secondary task.

1.2 PROJECT PURPOSE

We compare some popular Pre-Trained CNN Model Architecture provided by Keras which is an opensource neural network library written in Python. The architecture we used is VGG16, VGG19, ResNet50, ResNet152 v2, InceptionV3 and Inception-ResNet V2. From this research, we expected to see the best Pretrained Architecture model with the highest level of accuracy, and the lowest cost function in the optimal hyperparameter state.

1.3 PROJECT FEATURES

We introduce a deep learning framework to first detect 14 facial key-points which are then utilized to perform disguised face identification. Since the training of deep learning architectures relies on large annotated datasets, two annotated facial key-points datasets are introduced. The effectiveness of the facial key point detection framework is presented for each key point. The superiority of the key-point detection framework is also demonstrated by a comparison with other deep networks. The effectiveness of classification performance is also demonstrated by comparison with the state of-the-art face disguise classification methods.

2.SYSTEM ANALYSIS

SYSTEM ANALYSIS

System Analysis is the important phase in the system development process. The System is studied to the minute details and analyzed. Analysis is the process of finding the best solution to the problem. System analysis is the process by which we learn about the existing problems, define objects and requirements, and evaluate the solutions. It is the way of thinking about the organization and the problem it involves, a set of technologies that helps in solving these problems. Feasibility study plays an important role in system analysis which gives the target for design and development.

2.1 PROBLEM DEFINITION

From this research, we expected to see the best Pretrained Architecture model with the highest level of accuracy, and the lowest cost function in the optimal hyperparameter state. Dataset, which is a data set of 75 pictures of a person's face using a disguised tool like a bandana, masker, fake moustache, fake beard, glasses, etcetera. Disguised face identification (DFI) is an extremely challenging problem due to the numerous variations that can be introduced using different disguises. We introduce a deep learning framework to first detect 14 facial key-points which are then utilized to perform disguised face identification.

2.2 EXISTING SYSTEM

In the Existing system disguise face detection uses PCA approach. The PCA algorithm has an improved recognition rate for face images with large variations in lighting direction and facial expression and the face images are divided into smaller sub-images. The PCA approach is applied to each of these sub-images. Since some of the local facial features of an individual do not vary even when the pose, lighting direction and facial expression vary. The accuracy of the conventional PCA method and modular PCA method are evaluated under the conditions of varying expression, illumination and pose using standard face databases.

2.2.1 LIMITATIONS OF EXISTING SYSTEM

- Finding the eigenvectors and eigenvalues are time consuming on PCA. The size and location of each face image must remain similar PCA (Eigenface) approach maps features to principal subspaces that contain most energy.
- When a face-detection algorithm finds a face in an image or in a still from a video capture, the relative size of that face compared with the enrolled image size affects how well the face will be recognized.
- **Algorithm**: Principal Component Analysis (PCA).

2.3 PROPOSED SYSTEM

In the proposed system, we compare some popular Pre-Trained CNN Model Architecture provided by Keras which is an opensource neural network library written in Python. The architecture we used is VGG16, VGG19, ResNet50 and InceptionV3. Then, we divide into two parts: using the vector to train the classifier model, and evaluating the accuracy and cost function of the classifier model from this research, we expected to see the best Pretrained Architecture model with the highest level of accuracy, and the lowest cost function in the optimal hyperparameter state. In transfer learning, the knowledge of an already trained machine learning model is applied to a different but related problem. For example, if you trained a simple classifier to predict whether an image contains a backpack, you could use the knowledge that the model gained during its training to recognize other objects like sunglasses.

2.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

- With transfer learning a solid machine learning model can be built with comparatively little training data because the model is already pre-trained.
- This is especially valuable in natural language processing because mostly expert knowledge is required to create large labeled datasets.
- Additionally, training time is reduced because it can sometimes take days or even weeks to train a deep neural network from scratch on a complex task.

Algorithm: CNN with VGG16, VGG19, ResNet50 and Inceptionv3.

2.4 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. Three key considerations involved in the feasibility analysis are

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

2.4.1 ECONOMIC FEASIBILITY

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during preliminary investigation:

- The costs conduct a full system investigation.
- The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.

  Since the system is developed as part of project work, there is no manual cost to spend for the proposed system. Also, all the resources are already available

2.4.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.4.3 BEHAVIORAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system**.**

2.5 HARDWARE & SOFTWARE REQUIREMENTS

2.5.1 HARDWARE REQUIREMENTS:

Hardware interfaces specifies the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

- System : i9 Processor
- Hard Disk : 500 GB
- Input Devices : Keyboard, Mouse
- Ram : 8GB

2.5.2 SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements.

- Operating system : Windows 10/ MacOS
- Coding Language : Python, Django
- Tool : PyCharm CE

3. ARCHITECTURE

3. ARCHITECTURE

3.1 PROJECT ARCITECTURE

The project architecture shows the entire structure of the bot.

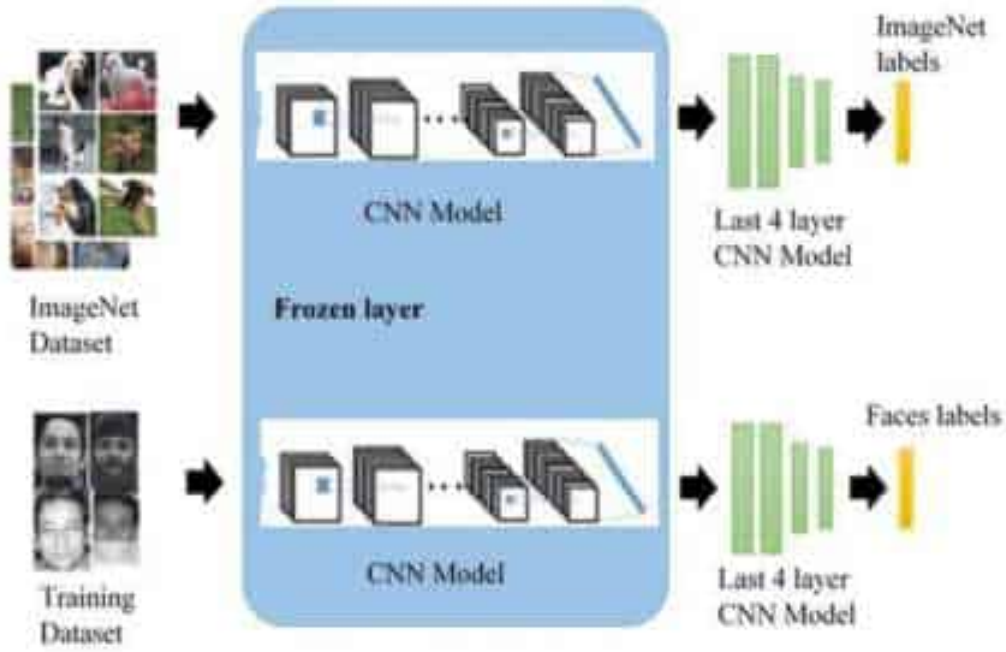


Figure 3.1 Project Architrcture

3.2 MODULES DESCRIPTION

3.2.1 USER

The User can collect the images from web surfing. in the data folder training, testing and validation images are available. These are classified into 3 classes those are disguise, mask and scarf. Now user has to create the objects of Vgg16 and Vgg19 and generate models weight filed call .h5 file. This h5 file contain all the weights of our models. At the time of generating h5 file the los function, loss accuracy, accuracy and test accuracy will be displayed in the graph and execution time. User can test the object detection from a video file or system camera. To process this user required high configuration system. User can test images by help of generated models weights files.

3.2.2 VGG16

The Vgg16 architecture comes from the VGG group, Oxford. VGG was made to improve from the AlexNet architecture by replacing large kernel filters (11 and 5 in the first and second convolutional layers) with some 3x3 kernel filters. With a given receptive field, small-sized kernels that are stacked are better than large-size kernels, because several non-linear layers increase the depth of the network which makes it possible to learn more complex feature.

3.2.3 ResNet50

In accordance with what has been discussed so far, namely, to improve accuracy in the network must increase the depth of the layer, as long as it can keep over-fitting. However, increasing the deep network does not work by simply adding layers. Deep networks are difficult to practice because of the problem of vanishing gradients, where gradients are repropagated to the previous layer, repeated repetition can make the gradient very small. As a result, as the network grows, the performance becomes saturated or even begins to degrade quickly. the basic idea of ResNet (Residual Network) is to introduce what is called an "identity shortcut connection" that passes through one or more layer.

3.2.4 InceptionV3

VGG achieved phenomenal accuracy in the ImageNet dataset, but its use requires high computation, even though it uses a GPU (Graphic Processing Unit). This has become inefficient due to the large width of the convolutional layer used. GoogLeNet builds on the idea that most activations in deep networks are not needed (zero value) or excessive because of the correlation between them. Therefore, the most efficient deep network architecture will have sparse connections between activations, which implies that all 512 output channels will not have connections between each other. GoogLeNet designed a module called the Inception module which numbered roughly like a thin CNN with a solid construction. Because only a small fraction of the neurons is effective as mentioned previously, the width/number of convolutional filters of the kernel size is kept small. This module also uses convolution of various sizes to capture details at various scales.

3.3 USE CASE DIAGRAM

A use case diagram purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

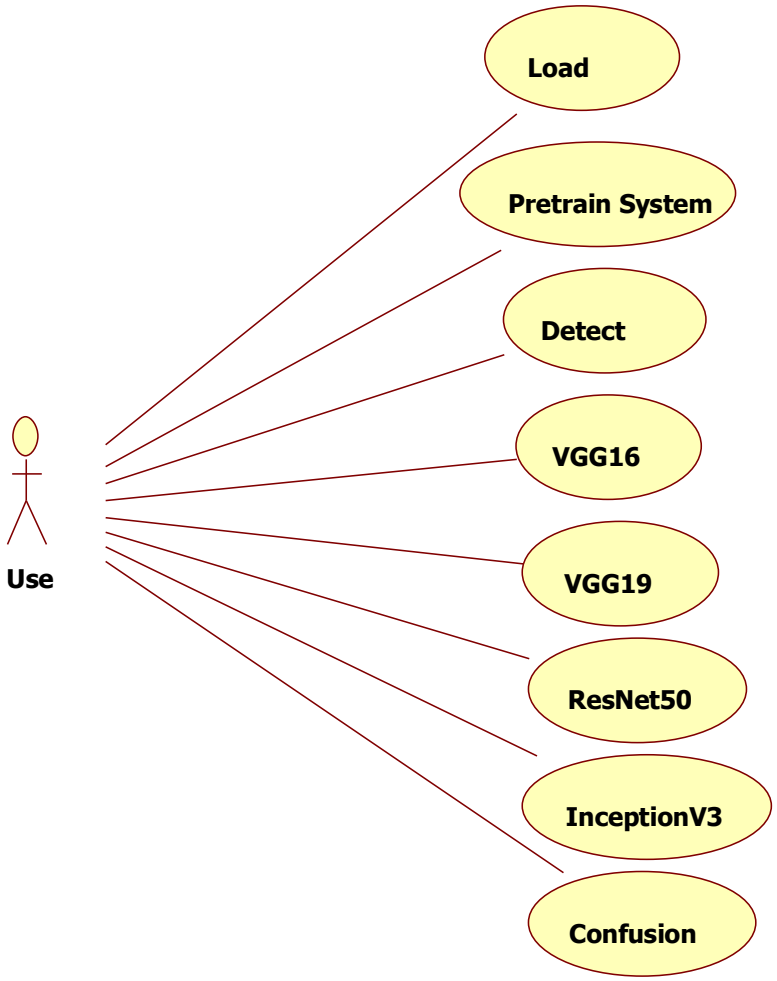


Figure 3.3: Use Case Diagram for how the system functions

3.4 SEQUENCE DIAGRAM

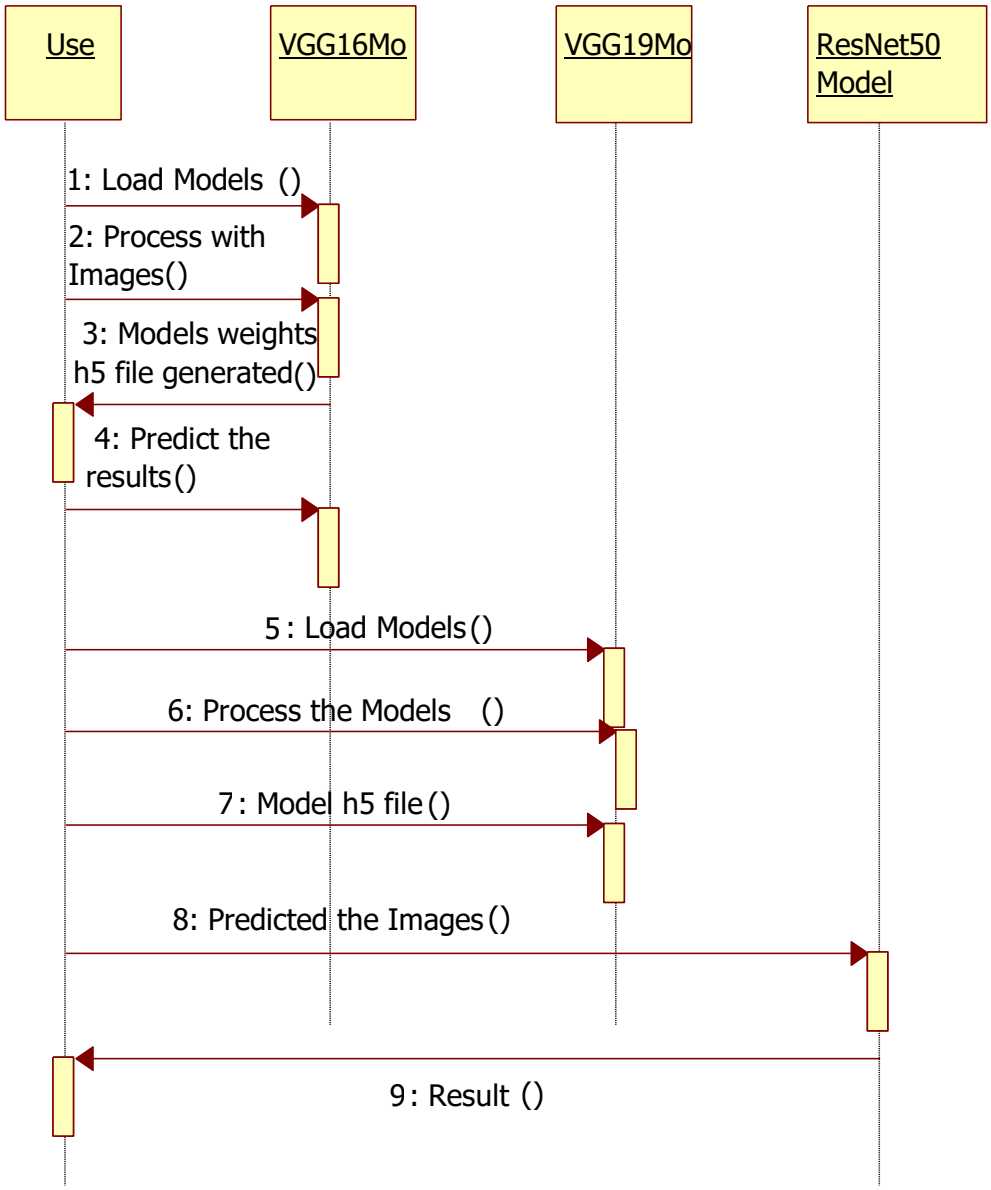


Figure 3.4: Sequence Diagram for how processes operate with one another and in what order

3.5 ACTIVITY DIAGRAM

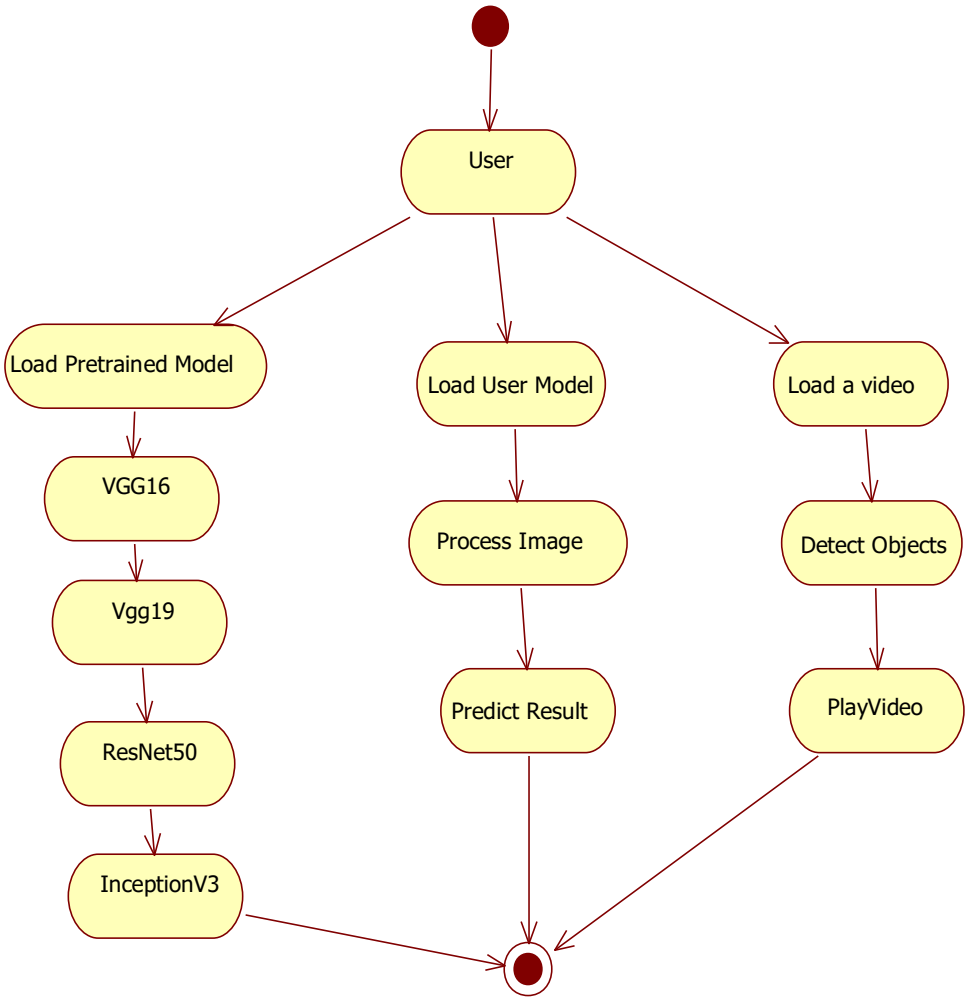


Figure 3.5: Activity Diagram for step-by-step workflows of components in a system

3.6 CLASS DIAGRAM

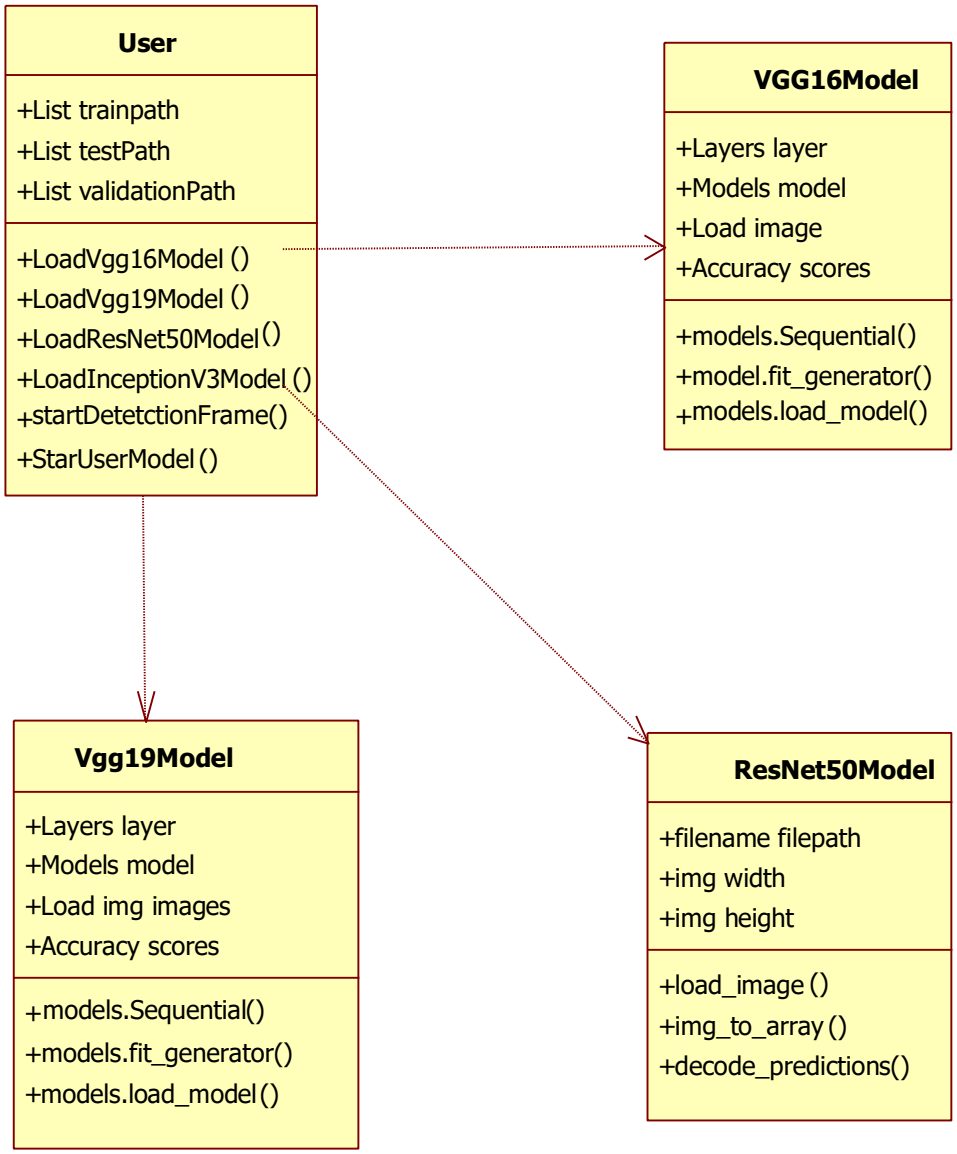


Figure 3.6: Class Diagram for structure of a system by showing the system's relationships among the classes

4.IMPLEMENTATION

4.IMPLEMENTATON

4.1 SAMPLE CODE

1-RunFirstVideoDetection.py

```
from imageai.Detection import VideoObjectDetection
import os
execution_path = os.getcwd()
detector = VideoObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath(os.path.join(execution_path, "yolo.h5"))
detector.loadModel()
video_path=detector.detectObjectsFromVideo(input_file_path=os.path.join(executio
n_path,"traffic-mini.mp4"),
output_file_path=os.path.join(execution_path,"traffic_mini_detected_1"),frames_per
_second=29,log_progress=True)
print(video_path)
```

2-RunFirstCameraDetection.py

```
from imageai.Detection import VideoObjectDetection
import os
import cv2
execution_path = os.getcwd()
camera = cv2.VideoCapture(0)
detector = VideoObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath(os.path.join(execution_path, "yolo.h5"))
detector.loadModel()
video_path = detector.detectObjectsFromVideo(camera_input=camera,
output_file_path=os.path.join(execution_path,"camera_detected_1"),
frames_per_second=29,log_progress=True)
print(video_path)
```

3-RunTestVGG16Val.py

```
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow
from numpy.random import seed
seed(1337)
tensorflow.random.set_seed(42)
from tensorflow.python.keras.applications import vgg16
from tensorflow.python.keras.applications.vgg16 import preprocess_input
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator, load_img
from tensorflow.python.keras.callbacks import ModelCheckpoint
from tensorflow.python.keras import layers, models, Model, optimizers
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from plot_conf import plot_confusion_matrix
train_data_dir = "data/train"
val_data_dir = "data/val"
test_data_dir = "data/test"
category_names = sorted(os.listdir('data/train'))
nb_categories = len(category_names)
img_pr_cat = []
for category in category_names:
    folder = 'data/train' + '/' + category
    img_pr_cat.append(len(os.listdir(folder)))
sns.barplot(y=category_names, x=img_pr_cat).set_title("Number of training images per category:")
for subdir, dirs, files in os.walk('data/train'):
    for file in files:
        img_file = subdir + '/' + file
        image = load_img(img_file)
        plt.figure()
```

```
        plt.title(subdir)
 plt.imshow(image)
        break
img_height, img_width = 224,224
conv_base = vgg16.VGG16(weights='imagenet', include_top=False, pooling='max',
input_shape = (img_width, img_height, 3))
for layer in conv_base.layers:
    print(layer, layer.trainable)
model = models.Sequential()
model.add(conv_base)
model.add(layers.Dense(nb_categories, activation='softmax'))
model.summary()
#Number of images to load at each iteration
batch_size = 32
# only rescaling
train_datagen =  ImageDataGenerator(rescale=1./255)
test_datagen =  ImageDataGenerator( rescale=1./255)
# these are generators for train/test data that will read pictures #found in the defined
subfolders of 'data/'
print('Total number of images for "training":')
train_generator = train_datagen.flow_from_directory(train_data_dir,
target_size = (img_height, img_width),
batch_size = batch_size,
class_mode = "categorical")
print('Total number of images for "validation":')
val_generator = test_datagen.flow_from_directory(
val_data_dir,
target_size = (img_height, img_width),
batch_size = batch_size,
class_mode = "categorical",
shuffle=False)
print('Total number of images for "testing":')
test_generator = test_datagen.flow_from_directory(test_data_dir,
target_size = (img_height, img_width),
```

```
batch_size = batch_size,
class_mode = "categorical",
shuffle=False)
learning_rate = 5e-5
epochs = 2
checkpoint =ModelCheckpoint("vgg16_classifier.h5", monitor = 'val_acc', verbose=1, save_best_only=True, save_weights_only=False, mode='auto', period=1)
model.compile(loss="categorical_crossentropy",optimizer=tensorflow.optimizers.Adam(lr=learning_rate, clipnorm = 1.), metrics = ['acc'])
history = model.fit_generator(train_generator,  epochs=epochs, shuffle=True,
 validation_data=val_generator,callbacks=[checkpoint])
model = models.load_model("vgg16_classifier.h5")
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1,len(acc)+1)
plt.figure()
plt.plot(epochs, acc, 'b', label = 'Training accuracy')
plt.plot(epochs, val_acc, 'r', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend()
plt.savefig('Accuracy.jpg')
plt.figure()
plt.plot(epochs, loss, 'b', label = 'Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.savefig('VGG16Loss.jpg')
Y_pred = model.predict_generator(test_generator)
y_pred = np.argmax(Y_pred, axis=1)
cm = confusion_matrix(test_generator.classes, y_pred)
plot_confusion_matrix(cm, classes = category_names, title='Confusion Matrix', normalize=False, figname = 'VGG16_Confusion_matrix_concrete.jpg')
```

```
accuracy = accuracy_score(test_generator.classes, y_pred)
print("Accuracy in test set: %0.1f%% " % (accuracy * 100))
conv_base = vgg16.VGG16(weights='imagenet', include_top=False, pooling='max', input_shape = (img_width, img_height, 3))
#for layer in conv_base.layers[:-13]:
#    layer.trainable = False
model = models.Sequential()
model.add(conv_base)
model.add(layers.Dense(nb_categories, activation='softmax'))
train_datagen= ImageDataGenerator(rescale=1./255,rotation_range=10,zoom_range=0.1,
width_shift_range=0.1,height_shift_range=0.1, horizontal_flip=False,
 brightness_range = (0.9,1.1),fill_mode='nearest' )
# this is a generator that will read pictures found in
# subfolers of 'data/train', and indefinitely generate
# batches of augmented image data
train_generator = train_datagen.flow_from_directory(
train_data_dir,
target_size = (img_height, img_width),
batch_size = batch_size,
#save_to_dir='augm_images',
save_prefix='aug',
save_format='jpg',
class_mode = "categorical")
learning_rate = 5e-5
epochs = 2
checkpoint = ModelCheckpoint("vgg16_classifier_augm.h5", monitor='val_acc', verbose=1, save_best_only=True, save_weights_only=False, mode='auto', period=1)
model.compile(loss="categorical_crossentropy", optimizer=optimizers.Adam(lr=learning_rate, clipnorm=1.), metrics = ['acc'])
history = model.fit_generator(train_generator, epochs=epochs,
shuffle=True,validation_data=test_generator,callbacks=[checkpoint] )
model = models.load_model("vgg16_classifier_augm.h5")
acc = history.history['acc']
```

```
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1,len(acc)+1)
plt.figure()
plt.plot(epochs, acc, 'b', label = 'Training accuracy')
plt.plot(epochs, val_acc, 'r', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend()
plt.savefig('Vgg16Accuracy_Augmented.jpg')
plt.figure()
plt.plot(epochs, loss, 'b', label = 'Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.savefig('VGG16Loss_Augmented.jpg')
Y_pred = model.predict_generator(test_generator)
y_pred = np.argmax(Y_pred, axis=1)
cm_aug = confusion_matrix(test_generator.classes, y_pred)
plot_confusion_matrix(cm_aug, classes = category_names, title='Confusion Matrix',
normalize=False, figname = 'Vgg16Confusion_matrix_Augm.jpg')
accuracy = accuracy_score(test_generator.classes, y_pred)
print("Accuracy in test set: %0.1f%% " % (accuracy * 100))
'''
test_subset_data_dir = "data/test_subset"
test_subset_generator = test_datagen.flow_from_directory(
test_subset_data_dir,
batch_size = batch_size,
target_size = (img_height, img_width),
class_mode = "categorical",
shuffle=False)
Y_pred = model.predict_generator(test_subset_generator)
y_pred = np.argmax(Y_pred, axis=1)
```

```
img_nr = 0
for subdir, dirs, files in os.walk('data/test_subset'):
    for file in files:
        img_file = subdir + '/' + file
        image = load_img(img_file,target_size=(img_height,img_width))
        pred_emotion = category_names[y_pred[img_nr]]
        real_emotion = category_names[test_subset_generator.classes[img_nr]]
        plt.figure()
        plt.title('Predicted: ' + pred_emotion + '\n' + 'Actual:      ' + real_emotion)
        plt.imshow(image)
        img_nr = img_nr +1
'''
```

4-RunTestVGG19Val.py

```
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow
from numpy.random import seed
seed(1337)
tensorflow.random.set_seed(42)
from tensorflow.python.keras.applications import vgg19
from tensorflow.python.keras.applications.vgg19 import preprocess_input
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator, load_img
from tensorflow.python.keras.callbacks import ModelCheckpoint
from tensorflow.python.keras import layers, models, Model, optimizers
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from plot_conf import plot_confusion_matrix
train_data_dir = "data/train"
val_data_dir = "data/val"
test_data_dir = "data/test"
```

```
category_names = sorted(os.listdir('data/train'))
nb_categories = len(category_names)
img_pr_cat = []
for category in category_names:
    folder = 'data/train' + '/' + category
    img_pr_cat.append(len(os.listdir(folder)))
sns.barplot(y=category_names, x=img_pr_cat).set_title("Number of training images per category:")
for subdir, dirs, files in os.walk('data/train'):
    for file in files:
        img_file = subdir + '/' + file
        image = load_img(img_file)
        plt.figure()
        plt.title(subdir)
        plt.imshow(image)
        break
img_height, img_width = 224,224
conv_base = vgg19.VGG19(weights='imagenet', include_top=False, pooling='max', input_shape = (img_width, img_height, 3))
for layer in conv_base.layers:
    print(layer, layer.trainable)
model = models.Sequential()
model.add(conv_base)
model.add(layers.Dense(nb_categories, activation='softmax'))
model.summary()
#Number of images to load at each iteration
batch_size = 32
# only rescaling
train_datagen =  ImageDataGenerator( rescale=1./255)
test_datagen =  ImageDataGenerator( rescale=1./255)
# these are generators for train/test data that will read pictures #found in the defined subfolders of 'data/'
print('Total number of images for "training":')
train_generator = train_datagen.flow_from_directory(
```

```
train_data_dir,
target_size = (img_height, img_width),
batch_size = batch_size,
class_mode = "categorical")
print('Total number of images for "validation":')
val_generator = test_datagen.flow_from_directory(
val_data_dir,
target_size = (img_height, img_width),
batch_size = batch_size,
class_mode = "categorical",
shuffle=False)
print('Total number of images for "testing":')
test_generator = test_datagen.flow_from_directory(
test_data_dir,
target_size = (img_height, img_width),
batch_size = batch_size,
class_mode = "categorical",
shuffle=False)
learning_rate = 5e-5
epochs = 2
checkpoint = ModelCheckpoint("vgg19_classifier.h5",monitor ='val_acc', verbose=1,
save_best_only=True, save_weights_only=False, mode='auto', period=1)
model.compile(loss="categorical_crossentropy",
optimizer=optimizers.Adam(lr=learning_rate, clipnorm = 1.), metrics = ['acc'])
history = model.fit_generator(train_generator, epochs=epochs, shuffle=True,
validation_data=val_generator,callbacks=[checkpoint] )
model = models.load_model("vgg19_classifier.h5")
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1,len(acc)+1)
plt.figure()
plt.plot(epochs, acc, 'b', label = 'Training accuracy')
```

```
plt.plot(epochs, val_acc, 'r', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend()
plt.savefig('Accuracy.jpg')
plt.figure()
plt.plot(epochs, loss, 'b', label = 'Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.savefig('VGG19Loss.jpg')
Y_pred = model.predict_generator(test_generator)
y_pred = np.argmax(Y_pred, axis=1)
cm = confusion_matrix(test_generator.classes, y_pred)
plot_confusion_matrix(cm, classes = category_names, title='Confusion Matrix', normalize=False, figname = 'VGG19_Confusion_matrix_concrete.jpg
accuracy = accuracy_score(test_generator.classes, y_pred)
print("Accuracy in test set: %0.1f%% " % (accuracy * 100))
conv_base = vgg19.VGG19(weights='imagenet', include_top=False, pooling='max', input_shape = (img_width, img_height, 3))
#for layer in conv_base.layers[:-13]:
#    layer.trainable = False
model = models.Sequential()
model.add(conv_base)
model.add(layers.Dense(nb_categories, activation='softmax'))
train_datagen = ImageDataGenerator( rescale=1./255,  rotation_range=10,
zoom_range=0.1, width_shift_range=0.1, height_shift_range=0.1,
horizontal_flip=False,brightness_range = (0.9,1.1),fill_mode='nearest'  )
# this is a generator that will read pictures found in
# subfolers of 'data/train', and indefinitely generate
# batches of augmented image data
train_generator = train_datagen.flow_from_directory(
train_data_dir,
target_size = (img_height, img_width),
batch_size = batch_size,
```

```
#save_to_dir='augm_images',
save_prefix='aug',
save_format='jpg',
class_mode = "categorical")
learning_rate = 5e-5
epochs = 2
checkpoint = ModelCheckpoint("vgg19_classifier_augm.h5", monitor='val_acc', verbose=1, save_best_only=True, save_weights_only=False, mode='auto', period=1)
model.compile(loss="categorical_crossentropy", optimizer=optimizers.Adam(lr=learning_rate, clipnorm=1.), metrics = ['acc'])
history=model.fit_generator(train_generator,epochs=epochs,shuffle=True, validation_data=test_generator, callbacks=[checkpoint])
model = models.load_model("vgg19_classifier_augm.h5")
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1,len(acc)+1)
plt.figure()
plt.plot(epochs, acc, 'b', label = 'Training accuracy')
plt.plot(epochs, val_acc, 'r', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend()
plt.savefig('Vgg19Accuracy_Augmented.jpg')
plt.figure()
plt.plot(epochs, loss, 'b', label = 'Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.savefig('VGG19Loss_Augmented.jpg')
Y_pred = model.predict_generator(test_generator)
y_pred = np.argmax(Y_pred, axis=1)
cm_aug = confusion_matrix(test_generator.classes, y_pred)
```

```
plot_confusion_matrix(cm_aug, classes = category_names, title='Confusion Matrix',
normalize=False, figname = 'Vgg19Confusion_matrix_Augm.jpg')
accuracy = accuracy_score(test_generator.classes, y_pred)
print("Accuracy in test set: %0.1f%% " % (accuracy * 100))
'''
test_subset_data_dir = "data/test_subset"
test_subset_generator = test_datagen.flow_from_directory(
test_subset_data_dir,
batch_size = batch_size,
target_size = (img_height, img_width),
class_mode = "categorical",
shuffle=False)
Y_pred = model.predict_generator(test_subset_generator)
y_pred = np.argmax(Y_pred, axis=1)
img_nr = 0
for subdir, dirs, files in os.walk('data/test_subset'):
    for file in files:
        img_file = subdir + '/' + file
        image = load_img(img_file,target_size=(img_height,img_width))
        pred_emotion = category_names[y_pred[img_nr]]
        real_emotion = category_names[test_subset_generator.classes[img_nr]]
        plt.figure()
        plt.title('Predicted: ' + pred_emotion + '\n' + 'Actual:      ' + real_emotion)
        plt.imshow(image)
        img_nr = img_nr +1
'''
```

5-ResNet50-Code.py

```
import PIL
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.applications.imagenet_utils import decode_predictions
import matplotlib.pyplot as plt
```

```
import numpy as np
from keras.applications.resnet50 import ResNet50
from keras.applications import resnet50
filename = '371.jpg'
original = load_img(filename, target_size = (224, 224))
print('PIL image size',original.size)
plt.imshow(original)
plt.show()
#convert the PIL image to a numpy array
numpy_image = img_to_array(original)
plt.imshow(np.uint8(numpy_image))
print('numpy array size',numpy_image.shape)
# Convert the image / images into batch format
image_batch = np.expand_dims(numpy_image, axis = 0)
print('image batch size', image_batch.shape)
processed_image = resnet50.preprocess_input(image_batch.copy())
# create resnet model
resnet_model = resnet50.ResNet50(weights = 'imagenet')
# get the predicted probabilities for each class
predictions = resnet_model.predict(processed_image)
# convert the probabilities to class labels
label = decode_predictions(predictions)
print(label)
```

6-InceptionV3-Code.py

```
from keras.applications.inception_v3 import InceptionV3
# load model
model = InceptionV3()
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.applications.vgg16 import preprocess_input
from keras.applications.vgg16 import decode_predictions
# load an image from file
```

```
image = load_img('371.jpg', target_size=(299, 299))
# convert the image pixels to a numpy array
image = img_to_array(image)
# reshape data for the model
image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
# prepare the image for the VGG model
image = preprocess_input(image)
# predict the probability across all output classes
yhat = model.predict(image)
# convert the probabilities to class labels
label = decode_predictions(yhat)
# retrieve the most likely result, e.g. highest probability
label = label[0][0]
# print the classification
print('%s (%.2f%%)' % (label[1], label[2]*100))
```

7-TestOwnModels.py

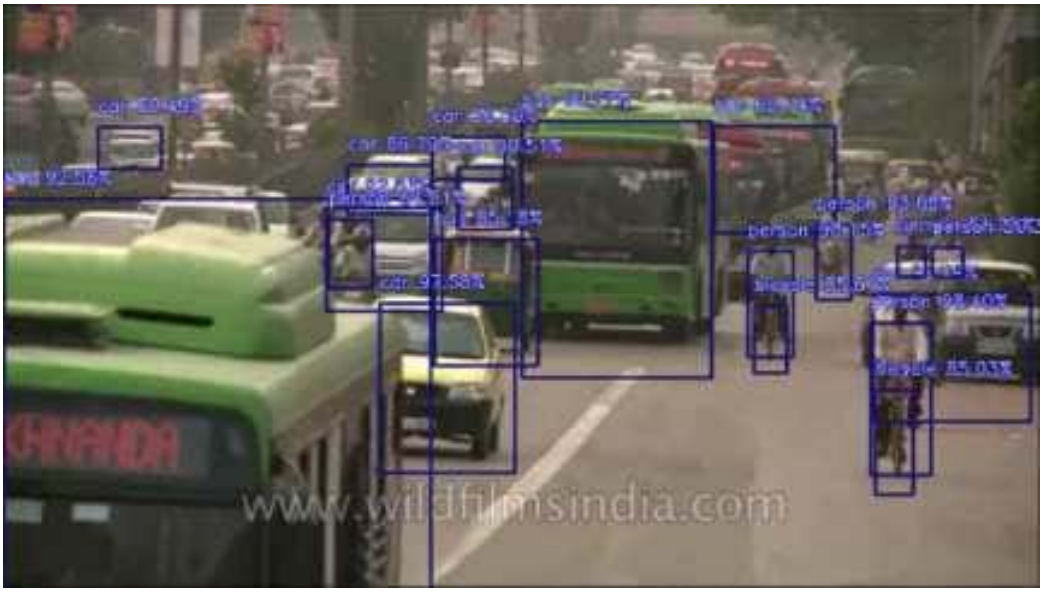
```
import PIL
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.applications.imagenet_utils import decode_predictions
from tensorflow.python.keras.models import load_model
import matplotlib.pyplot as plt
import numpy as np
from keras.applications.resnet50 import ResNet50
from keras.applications import resnet50
filename = '43.jpg'
original = load_img(filename, target_size = (224, 224))
print('PIL image size',original.size)
plt.imshow(original)
plt.show()
#convert the PIL image to a numpy array
numpy_image = img_to_array(original)
```

```
plt.imshow(np.uint8(numpy_image))
print('numpy array size',numpy_image.shape)
# Convert the image / images into batch format
image_batch = np.expand_dims(numpy_image, axis = 0)
print('image batch size', image_batch.shape)
processed_image = resnet50.preprocess_input(image_batch.copy())
modelpath = 'vgg19_classifier.h5'
# create resnet model
resnet_model = load_model(modelpath) #resnet50.ResNet50(weights = 'imagenet')
# get the predicted probabilities for each class
predictions = resnet_model.predict(processed_image)
categories = ["disguise", "mask","scarf"]
print(categories[int(predictions[0][0])])
# convert the probabilities to class labels
#label = decode_predictions(predictions)
#print(label)
```

5.SCREENSHOTS

5.SCREENSHOTS

5.1 OUTPUT - 1



Screenshot 5.1: Output-1

5.2 OUTPUT – 2

Screenshot 5.2: Ouput-2

5.3 OUTPUT-3

Screenshot 5.3: Output-3

5.4 OUTPUT-4

Screenshot 5.4: Output-4

6.TESTING

6. TESTING

6.1 INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

6.2 TYPES OF TESTING

6.2.1 UNITTESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.2.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals

Functional testing is centered on the following items:

Valid Input	:	Identified classes of valid input must be accepted.
Invalid Input	:	Identified classes of invalid input must be rejected.
Functions	:	Identified functions must be exercised.
Output	:	Identified classes of application outputs must be exercised.
Systems/Procedures	:	Interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identifying Business process flows; data fields, predefined processes.

6.3 TEST CASES

S.no	Test Case	Excepted Result	Result	Remarks(IF Fails)
1.	User Has to locate the image path	User can keep training images, testing images and validation images under data folder	Pass	If images not available then failed
2.	Start the system cameras or pre video files	Detect the objects from given video files	Pass	CNN Model required
3.	Detected object play again the video	All frames playing again.	Pass	Video file .avi format required
4.	Create object of VGG16 Model	VGG16 Model object creates and process.	Pass	Vgg16 has to install in the system
5.	Generate the h5 weights files	Loaded Model .H5 file will generate and stores in system drive	Pass	If model not trained the failed
6.	Load VGG19 Model	VGG19 Model loaded	Pass	Vgg16 Model ahs to Load
7.	Load the ResNet50 Models	ResNet50 Model loaded and object created	Pass	ResNet has to install.
8.	Load InceptionV3 model	Inception V3 Model has to load	Pass	First Inception V3 Model has to load
9.	Test user images	User has to test its own images	Pass	To test created models weight required
10.	Confusion Matrix generated	Confusion Matrix for Vgg16 and Vgg19 Models	Pass	If model not trained the failed

7. CONCLUSION

7. CONCLUSION & FUTURE SCOPE

7.1 PROJECT CONCLUSION

In this project, we propose a comparison of six popular CNN Models to recognizing the disguised person's face using "Recognizing Disguised Faces" datasets, and the findings are how Transfer Learning be used in Face Verification problem. In training result shows that the VGG model has balance accuracy of training and validation, and the other side, the ResNet152 v2 Model has a better accuracy than VGG in train set. But in the test result shows, that VGG model is the highest performance than other CNN Models. We then conclude that ImageNet weight can be used for Transfer Learning to Recognize face using VGG Model. The success of this Convolutional Neural Network is also the main reason why Deep Learning CNN has been such a hot topic in recent years.

7.2 FUTURE SCOPE

As a future research direction, we plan to encode and incorporate the concept of familiarity in automatic algorithms which may improve the performance. Further, we also believe that the study of how disguising individual facial parts affect representations of faces might lead to better solutions to mitigate these variations.

8.BIBLIOGRAPHY

8.BIBLIOGRAPHY

8.1 GITHUB REPOSITORY LINK

https://github.com/AbhinavJoel/major-Project

https://github.com/hanuman-adabala/Major-project-batch6

https://github.com/KishanPrasadD/Detecting-Disguised-Faces-With-Transfer-Learning.git

8.2 REFERENCES

[1] A. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," IBM Journal of Research and Development, vol. 3, no. 3, p. 210–229, 1959.

[2] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, p. pp. 586–591, 1991.

[3] J. Yang, D. Zhang, A. F. Frangi and J. Yang, "Two dimensional PCA : a new approach to appearance-based face representation and recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 1, p. 131–137, 2004.

[4] Y. Taigman, M. Yang, M. Ranzato and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," CVPR, p. 1701–1708, 2014..

[5] J. West, D. Ventura and S. Warnick, "A Theoretical Foundation for Inductive Transfer," Spring Research Presentation, 2007.

[6] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama and a. T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in ACM MM, 2014.

[7] D. Tomé, F. Monti, L. Baroffio, L. Bondi, M. Tagliasacchi and S. Tubaro, "Deep Convolutional Neural Networks for pedestrian detection," Signal Processing: Image Communication, vol. 47, pp. 482-489, 2016.

[8] Z. Deng, H. Sun, S. Zhou, J. Zhao and H. Z. Lin Lei, "Multi-scale object detection in remote sensing imagery with convolutional neural networks," ISPRS Journal of Photogrammetry and Remote Sensing, vol. 145, no. Part A, pp. 3-22, 2018.

9. JOURNAL

International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 06 | June - 2022 Impact Factor: 7.185 ISSN: 2582-3930

Detecting Disguised Faces with Transfer Learning

Dr. A. Prabhu[1], Daphedari Kishan Prasad[2], Adabala Taraka Rama Venkata Sai Hanuman[3]and Abhinav Joel T[4]

[1] *Associate Professor, Department of Computer Science and Engineering, CMR Technical Campus, Medchal, Telangana, India.*

[2] *UG Student, Department of Computer Science and Engineering, CMR Technical Campus, Medchal, Telangana, India.*

[3] *UG Student, Department of Computer Science and Engineering, CMR Technical Campus, Medchal, Telangana, India.*

[4] *UG Student, Department of Computer Science and Engineering, CMR Technical Campus, Medchal, Telangana, India.*

ABSTRACT:

In general, a human being has the memory power due to which he/she will be able to remember whatever they have seen but as the technology is increasing the advancement is increasing in such a way that now computer is also able to recognize theX faces from its memory but in order to differentiate them, we need more advancement which leads to the development of Machine learning. Machine learning concepts developed by Arthur Manuel. There have been many techniques used over the past decade to determine the identity of a person's face, such as Eigenfaces and Principal Component Analysis (PCA), to Convolutional Neural Networks (CNN) to ensure the ability to recognize faces has become further and further. An approach to machine learning called transfer learning involves creating a model of the first training task, then testing it using the model. The difference between transfer learning and traditional machine learning is that translation involves using a pre-trained model in order to start a secondary task using the initial model. It is expected that this paper will contribute to the field of image classification by using Machine Learning algorithms to solve the problem. Transfer learning significantly improves the performance of VGG models Based on these results, we conclude that VGG Models are the best choice for recognizing faces using ImageNet weights

Key words: *Image, Memory, Machine Learning, Techniques, CNN models, Face Recognition, Deep Learning, Transfer Learnings*

1-INTRODUCTION:

Computer systems use algorithms and statistical models to automate task performance without being explicitly programmed through machine learning (ML). Almost every application we use daily uses learning algorithms. Google works so well because a learning algorithm is trained to rank web pages every time someone searches the internet with the help of a search engine. A variety of applications can be performed with these algorithms, including data mining, image processing, predictive analytics, etc. Its main benefit is that algorithms can learn to interpret data automatically once they learn what to do with it. An overview of machine learning algorithms and their future prospects is presented in this paper where we are using the transfer learning for recognizing face in disguise. Several methods for identifying people's faces have been developed over the last decade, including Eigenfaces and Principal Component Analysis (PCA), as well as Convolutional Neural Networks (CNN), and the capacity to recognize faces has improved dramatically. Transfer learning is a machine learning technique in which the first training problem produces a model, and then the second test is performed using the model from the first training assignment.[1] Transfer learning varies from typical machine learning in that it includes employing a pre-trained model to launch a secondary task. Using Keras, an open-source neural network library written in Python, we compare a few pre-trained CNN architectures. A total of six architectures were used: VGG16, VGG19, ResNet50, ResNet152 v2, InceptionV3, and Inception-ResNet V2. Based on the results of this research, we expected to see the best Pretrained Architecture model with the highest degree of accuracy, as well as the most cost-effective function within the optimal hyperparameters. First, we construct a deep learning framework for detecting 14 facial key-points, and then we utilize those points to identify disguised faces based on these key points. Because deep learning architectures require large annotated datasets for training, two annotated face key-points datasets are presented. For each key point, the efficiency of the facial main point detection method based is demonstrated. A comparison of the important detection method based with other deep networks demonstrates its advantages. When compared to state-of-the-art face disguise classification technique,

the efficiency of classification performance is also proved.

2- System Analysis:

The critical phase of the system design is system analysis. The system is thoroughly investigated and assessed. The process of determining the optimum solution to a problem is known as analysis. The process of learning about existing problems, defining variables and requirements, and evaluating solutions is known as system analysis. It is a style of thinking about an organization and the difficulties it faces, as well as a combination of technologies that aid in the resolution of these issues. Feasibility studies are crucial in system analysis because they provide an objective for development and design.[2][1] We expected the finest Pretrained Architecture model with highest level of accuracy and the lowest cost function in the ideal hyperparameter state to emerge from this study. A data collection of 75 photographs of a person's face taken with a disguised tool such as a bandana, masker, fake moustache, fake beard, spectacles, and so on. Due to the countless changes that can be created using various disguises, disguised face identification (DFI) is an incredibly difficult subject. We provide a deep learning system that first detects 14 facial key-points before performing disguised face detection.

3- Existing system:

The PCA method is used in the present structure to recognize disguised faces. Face photos with considerable fluctuations in light direction and facial expression are better recognized by the PCA algorithm, which divides the face images into smaller sub-images. Each of these inter - and intra is subjected to the PCA method. Because certain of an individual's local facial features do not change regardless of stance, lighting direction, or facial emotion. Using typical face databases, the accuracy of a conventional PCA approach and the modular PCA method is assessed underneath the conditions of varied expression, illumination, and position. • PCA takes a long time to find the eigenvectors and eigenvalues. Each face image's size and location must be consistent. The PCA (Eigenface) technique maps features to major subspaces that contain the most energy. When a facial expression algorithm identifies a face in an image or an even now from a video recording, the face's relative size compared to the entire picture size influences how effectively the face is recognized. Principal Component Analysis (PCA) is an algorithm (PCA).

4- Proposed system:

We compare the common Pre-Trained CNN Model Architectures given by Keras, an opensource neural network toolkit written in Python, in the suggested system. VGG16, VGG19, ResNet50, and InceptionV3 were the architectures we employed. Then we split it into two parts: using the vector to train the classifier model and evaluating the accuracy and cost function of the classifier model. We expected seeing the best Pre - trained models Architecture model with the highest level of accuracy and the lowest cost function in the optimal hyperparameter state in this research. Transfer learning is when a machine learning model that has already been trained is used to solve a separate but related problem. For instance, if you trained a basic algorithm to classify whether a picture contains a bag, you'd be able to predict whether the image contains a backpack may apply the knowledge obtained during the model's training to distinguish additional objects, such as sunglasses.[3] Because the model has been pre-trained, a good machine learning model may be generated with relatively little training data via transfer learning.

• This is especially useful in natural language processing, where huge labelled datasets require a lot of expert knowledge.

• In addition, training time is lowered because training a deep convolutional neural network from start on a complex job can take days or weeks.

5- Requirements:

The logical properties of each interface between both the software system and the system's hardware components are defined by hardware interfaces. Some hardware requirements are listed below.

• Processor: i9

• Hard Drive: 500 GB

• Input Devices: Keyboard, Mouse

The logical properties of each interface and software component of the system are specified in Software Requirements. Some software requirements are listed here.

• Operating Systems: Windows 10/MacOS

• Programming Languages: Python, Django

• PyCharm CE is the programme you'll be using.

6- Proposed Methods:

Convolutional Neural Network (ConvNet):

Convolutional Neural Networks (CNN), often known as ConvNet, are a type of Artificial Neural Network (ANN) that is currently thought to be the best technique for solving object recognition and digit detection problems. [5]There are various models being built in a deep neural network like CNN, but we will only focus on three alternative model architectures in this research.

1) AlexNet is a website that connects people.

This architecture, which was created in 2012, is the first neural network that can accurately categories some objects in the Imagenet database, compared to standard approaches that existed before AlexNet. this network consists of 5 convolutional layers following 3 fully linked layers.

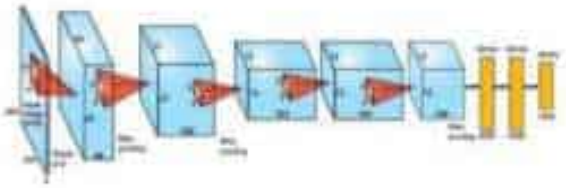


FIG -1 Network architecture of AlexNet.

2) VGG16

The VGG group, Oxford, designed this architecture in 2013. By substituting huge kernel filters (11 and 5 during the first and second convolutional layers, respectively) with some [5] 3x3 kernel filters, VGG was able to improve on the AlexNet design. Small-sized kernel that are layered are better than large-sized kernels for a given receptive field because numerous non-linear layers improve the depth of the network, allowing it to learn more complicated features.

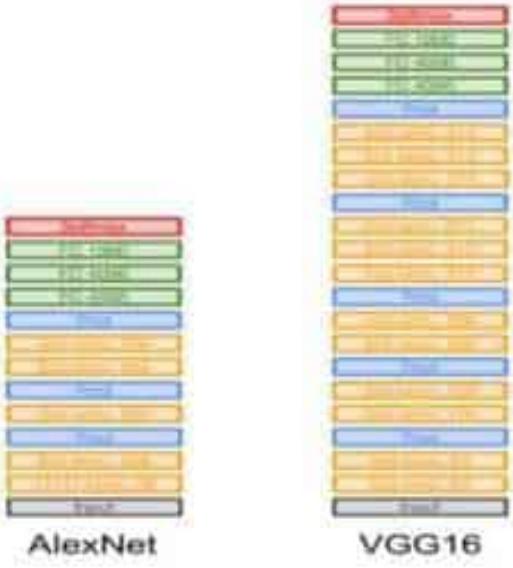


FIG – 2 Network architecture of VGG16.

3) ResNet

According to what has been mentioned thus far, in order to improve network accuracy, the layer depth must be increased as long as the network can continue to over-fit. However, just adding layers will not increase the depth of the network. Deep networks are challenging to implement because of the issue of vanishing gradients, which occurs when gradients are repeated to the preceding layer, resulting in a very small gradient.[5] As a consequence, as the network expands, performance gets saturated, if not rapidly degraded. ResNet (Residual Network) was founded in 2015 with the goal of introducing a "identity shortcut link" that passes via one or more levels.

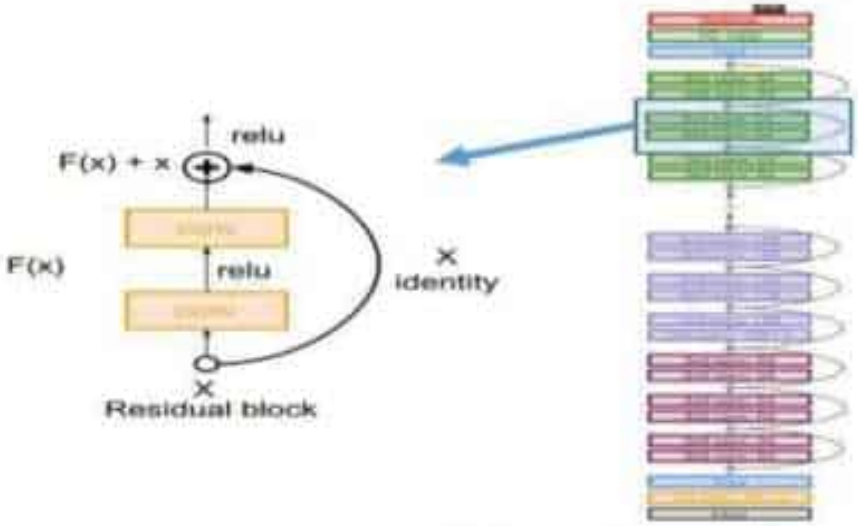
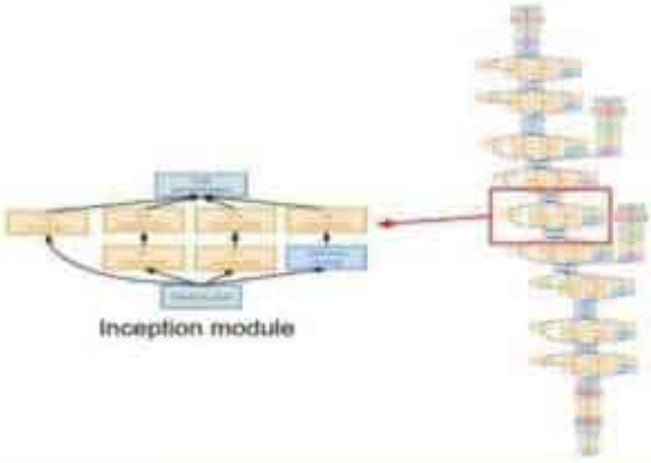


FIG -3 Network architecture of ResNet.

4) Inception V3

VGG attained exceptional precision in the ImageNet dataset, however its application, despite the GPU, necessitates a lot of work (Graphic Processing Unit). Due to the wide breadth of the convolution layers utilised, this has become inefficient. GoogLeNet is based on the premise that most deep network activations are either unnecessary (zero value) or excess due to the association between them. As a result, the most effective deep network architecture will have few connections among activations, meaning that none of the 512 output channels will be connected.[5] GoogLeNet created the Inception module, which was numbered in the manner of a thin CNN with a solid architecture. As previously stated, only a small fraction of a neurons are effective, hence the size of convolutional filters is limited.

FIG-4 Network architecture of GoogLeNet/Inception.



7- Architecture:

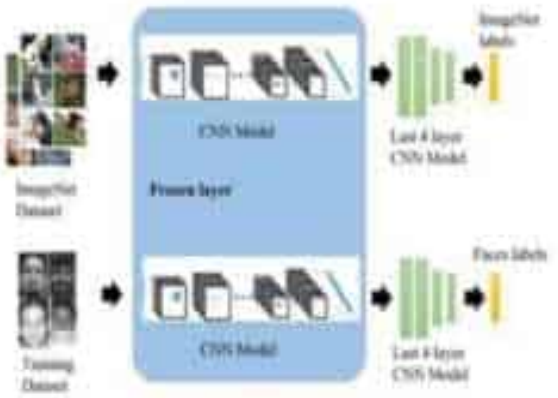


FIG- 5 ARCHITECURE REPRESENTATION

We set up a simple Sequence Model Architecture for the base layer, and then added the input layer to the VGG16, VGG19, ResNet50, ResNet152 v2, InceptionV3, and Inception-ResNet V2 Pre-trained Models we used in this study. Then, to our input model, we apply a pre-trained weight "ImageNet" that could be used for Transfer Learning. On our first attempt, we use VGG16 and proceed in the same manner with our other Input File.[6] We use 30 epochs for all trained models in each setup. And, with the exception of the InceptionV3 Model, each model receives a separate parameter modification after unfreezing the last four layers.

CNN Model	Freezing all layer (parameter)	Unfreezing last 4 layer (parameter)	CNN Model	Loss Value	Accuracy (%)	Validation Loss Value	Validation accuracy (%)
VGG16	8,406,507	15,545,931	VGG16	3.41	20.49	3.14	45.24
VGG19	8,406,507	15,545,931	VGG19	3.91	11.93	3.81	26.19
ResNet50	102,838,347	103,893,067	ResNet50	1.74	53.6	4.66	1.6
ResNet152 v2	102,838,347	103,893,067	ResNet152 v2	1.57	61.0	6.16	1.6
Inception v3	52,506,009	52,506,009	Inception v3	3.51	16.3	3.43	3.9
Inception-ResNet v2	100,741,195	103,937,611	Inception-ResNetv2	4.01	7.3	4.15	1.6

TABLE -I TRAINABLE PARAMETER OF CNN MODEL

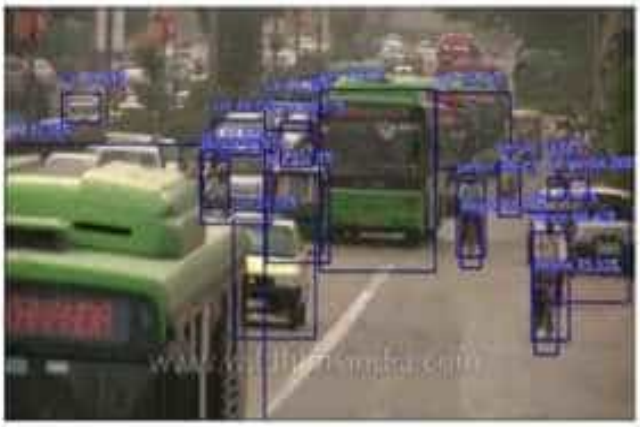
To avoid overfitting the model, we utilize a data pretreatment approach before training it. The technique entails downsizing all images to 224x224 pixels, as well as flipping and rotating them.

FIG -6 Output image after preprocessing

For training, we employ two setups: Freezing all layers in the CNN Model (setup 1) & Unfreezing the last four layers (setup 2). (setup 2). We are using the weight from the prior training to Cnn Architecture, which in our case comes from Keras trained in the ImageNet dataset, which has a lot of annotated images. The setting 2 means that we froze all layers but the last four, and then train the last four to improve accuracy on the dataset we used. This technique is known as Transfer Learning. In the CNN Model, all layers are frozen (Setup 1). We want to know the initial given impact for applying pre-trained weight on CNN Model, which is ImageNet, in this training configuration. In the CNN Model, unfreeze the final four layers (setup 2)In this training scheme, we first freeze all levels but the last four to our Pre-Trained CNN Model, which implies we only train the CNN Model's last four layers. We apply this to the entire Model, regardless of layer type. Then we have to re-fit our model.

8- Results:

8.1 - OUTPUT - 1



Screenshot 8.1: Output-1

8.2 OUTPUT – 2

Screenshot 8.2: Ouput-2

8.3 OUTPUT-3

Screenshot 8.3: Output-3

8.4 OUTPUT-4

Screenshot 8.4: Output-4

9- Testing and Test case used:

Unit testing is designed to ensure that each path of a business process performs accurately. Integration tests demonstrate that although the components were individually satisfactory[8], as shown by successfully unit testing, the combination of components is correct and consistent.

Functional testing is centered on the following items:

Valid Input : Identified classes of valid input must be accepted.

Output : Identified classes of application outputs must be exercised.

S.no	Test Case	Excepted Result	Result	Remarks(IF Fails)
1.	User Has to locate the image path	User can keep training images, testing images and validation images under data folder	Pass	If images not available then failed
2.	Start the system cameras or pre video files	Detect the objects from given video files	Pass	CNN Model required
3.	Detected object play again the video	All frames playing again.	Pass	Video file .avi format required
4.	Create object of VGG16 Model	VGG16 Model object creates and process.	Pass	Vgg16 has to install in the system
5.	Generate the h5 weights files	Loaded Model .H5 file will generate and stores in system drive	Pass	If model not trained the failed
6.	Load VGG19 Model	VGG19 Model loaded	Pass	Vgg16 Model ahs to Load
7.	Load the ResNet50 Models	ResNet50 Model loaded and object created	Pass	ResNet has to install.
8.	Load InceptionV3 model	Inception V3 Model has to load	Pass	First Inception V3 Model has to load
9.	Test user images	User has to test its own images	Pass	To test created models weight required
10.	Confusion Matrix generated	Confusion Matrix for Vgg16 and Vgg19 Models	Pass	If model not trained the failed

10- Conclusion and Future Scope:

We provide a comparison of six common CNN Models for detecting a disguised person's face using the "Recognizing Disguised Faces" datasets in this project, and the outcomes show how Transfer Learning may be employed in the Face Verification problem. The VGG model has a balanced accuracy of training and validation in the training set, whereas the ResNet152 v2 model has a greater accuracy than VGG in the train set.The Convolutional Neural Network's success is also one of the reasons why Deep Learning CNN has become such a popular topic in recent years. We intend to encode and include the idea of familiarity in automated algorithms as a future research topic, which may increase performance. Furthermore, we anticipate that studying how masking unique facial features affects face presentations may lead to improved strategies to attenuate these differences.

Reference:

[1] A. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," IBM Journal of Research and Development, vol. 3, no. 3, p. 210–229, 1959.

[2] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, p. pp. 586–591, 1991.

[3] J. Yang, D. Zhang, A. F. Frangi and J. Yang, "Two dimensional PCA : a new approach to appearance-based face representation and recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 1, p. 131–137, 2004.

[4] Y. Taigman, M. Yang, M. Ranzato and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," CVPR, p. 1701–1708, 2014.,

[5] J. West, D. Ventura and S. Warnick, "A Theoretical Foundation for Inductive Transfer," Spring Research Presentation, 2007.

[6] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama and a. T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in ACM MM, 2014.

[7] D. Tomé, F. Monti, L. Baroffio, L. Bondi, M. Tagliasacchi and S. Tubaro, "Deep Convolutional Neural Networks for pedestrian detection," Signal Processing: Image Communication, vol. 47, pp. 482-489, 2016.

[8] Z. Deng, H. Sun, S. Zhou, J. Zhao and H. Z. Lin Lei, "Multi-scale object detection in remote sensing imagery with convolutional neural networks," ISPRS Journal of Photogrammetry and Remote Sensing, vol. 145, no. Part A, pp. 3-22, 2018.

ISSN: 2582-3930

International Journal of Scientific Research in Engineering and Management

is hereby awarding this certificate to

Adabala Taraka Rama Venkata Sai Hanuman

in recognition the publication of manuscript entitled

Detecting Disguised Faces with Transfer Learning

published in Ijsrem Journal Volume 06 Issue 06, June 2022

Editor in Chief
E-mail: editor@ijsrem.com

ISSN: 2582-3930

International Journal of Scientific Research in Engineering and Management

is hereby awarding this certificate to

Abhinav Joel T

in recognition the publication of manuscript entitled

Detecting Disguised Faces with Transfer Learning

published in Ijsrem Journal Volume 06 Issue 06, June 2022

www.ijsrem.com

Editor in Chief
E-mail: editor@ijsrem.com

ISSN: 2582-3930

International Journal of Scientific Research in Engineering and Management

is hereby awarding this certificate to

Daphedari Kishan Prasad

in recognition the publication of manuscript entitled

Detecting Disguised Faces with Transfer Learning

published in Ijsrem Journal Volume 06 Issue 06, June 2022

www.ijsrem.com

Editor in Chief
E-mail: editor@ijsrem.com