

PROJECTSCHOOLCERTIFICATE

Title of the Project: Helmet detection and Traffic Signal Detection Control using Computer Vision

Session Duration:

Mentor**:**SriLakshmi,

### Assistant Professor, Department of CSE, NGIT.

Team Members:

### Beerla Varshith– 245321748013.

* AL Anirudh – 245321748004.

### S. Nitish Reddy – 245321748051.

* R .Sai Nishank– 245321748051.

### P G Praneeth– 245321748044.

### V. Hrushikesh 245321748120.

Department: CSE -AIML.

## SignatureofFaculty

ABSTRACT

Helmet detection and Traffic Signal Detection Control using Computer Vision

In this era of advanced technology, the integration of artificial intelligence (AI) and facial recognition has opened new avenues for various applications, including biometric authentication, surveillance, and personalization. One such application is the identification of blood relations through facial images, which holds significant potential in diverse fields such as forensic science, genealogy, and social networking.

Our project, Deep Relation, aims to leverage deep learning techniques to accurately infer blood relations solely from facial images. We propose a novel approach that combines convolutional neural networks (CNNs) for feature extraction and relational reasoning mechanisms for inferring complex familial relationships. To achieve this, we employ a large-scale dataset of annotated facial images representing individuals across multiple generations and diverse ethnicities. We preprocess the data to mitigate variations in lighting, pose, and facial expressions, thus enhancing the model's generalization capabilities. Subsequently, we design a multi-task learning framework to jointly predict facial attributes and infer familial relations, enabling the model to leverage shared representations and improve overall performance.

## OBJECTIVE

The objective of the project "Deep Relation: Utilizing Facial Recognition for Blood Relation Identification" is to develop an advanced system capable of accurately identifying and inferring blood relations solely from facial images.

## TECHNICAL DESCRIPTION

* Front End: HTML, CSS.
* Back End: Python-Flask, DeepLearning (for identification).

# CONTENTS

CHAPTER 1: USING VGG16 MODEL

* 1. : LOADING THE VGG16 MODEL
  2. : IMPORTING ALL THE REQUIRED LIBRARIES
  3. : LOADING THE DATA TO THE MODEL
  4. : DATA AUGMENTATION
  5. : TRAINING THE MODEL
  6. : TESTING THE MODEL

CHAPTER 2: USING RESNET50

* 1. : IMPORTING REQUIRED LIBRARIES
  2. : LOADING THE PRE-TRAINED MODEL
  3. : DATA PREPROCESSING
  4. : IMAGE EMBEDDING AND SIMILARITY CALCULATION
  5. : COMPARING IMAGE SIMILARITIES CHAPTER 3: IMPLEMENTATION
  6. : UI
  7. : UPLOADING IMAGE
  8. : RESULT

CHAPTER 4: CONCLUSION

## PROJECT WORKFLOW

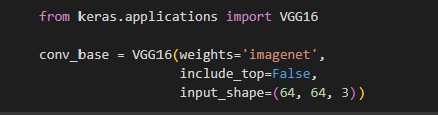
### We had applied two approaches to accomplish the project as desired according to the problem statement: -

* Developing and Training a VGG16 model.
* Using a pre-trained model (ResNet50).

1. USING A VGG16 MODEL

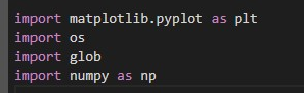
## Loading the VGG16 Model: -

Using an open-source neural network library which is written in python named KERAS has a keras. applications module that consists of various pre-trained models which are trained on the ImageNet Dataset. So, we use this module to load the VGG16 model that must be trained on our dataset (KinFaceW-1). The code below displays the above script: -



# Importing all the required libraries: -

The below python kernel displays the code that imports the required libraries: -



# Loading the data to the model: -

The below python kernel displays the code that loads the data

necessary for training a

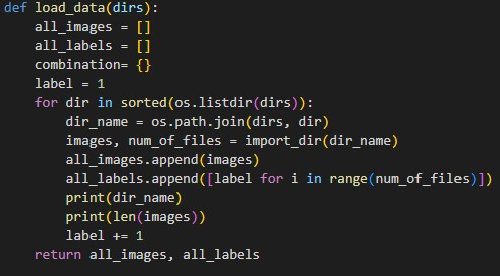
model. It does so by importing

images from

specified directories and assigning corresponding labels to each image.

This data loading step is machine learning models.

crucial in preparing the dataset for training



The below code is used to import the pairs of images from a

specified directory. Each

pair of images represents a

parent-child

relationship. It creates an

array “dir\_images” to store

the imported

images and initializes a variable ***start*** to determine the starting index for iteration through the list of files.



# Data Augmentation: -

Data Augmentation is a part of Data preprocessing that helps to reduce the overfitting of data and improves the overall performance

of the model which helps

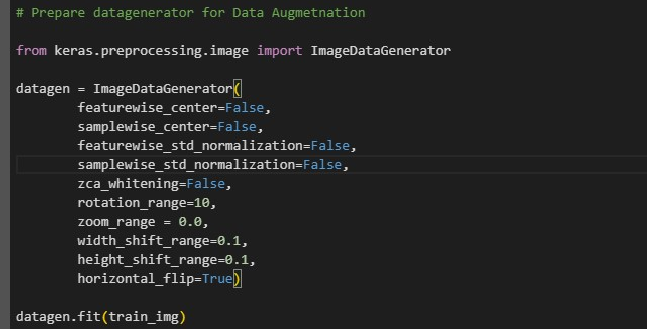
in getting high accuracy and

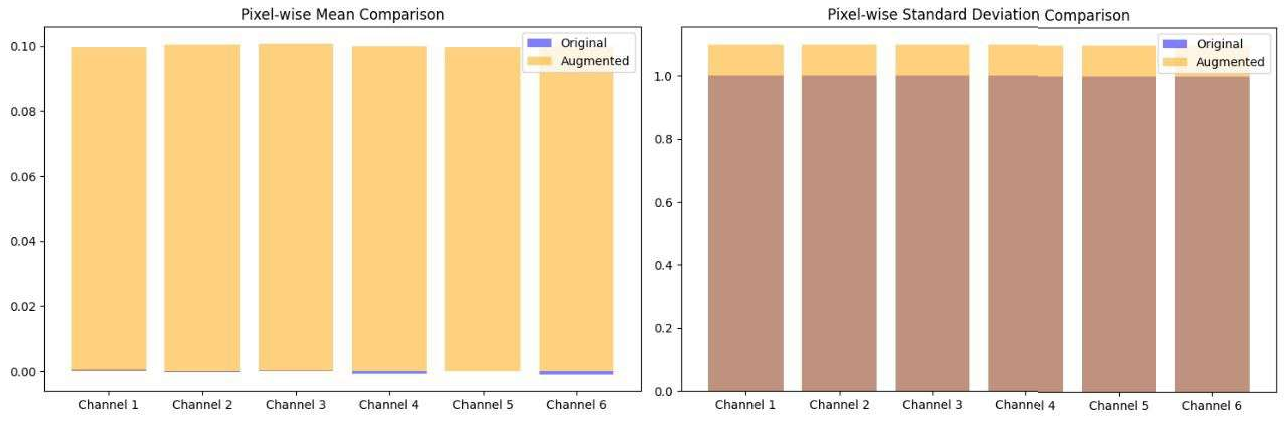
better results.

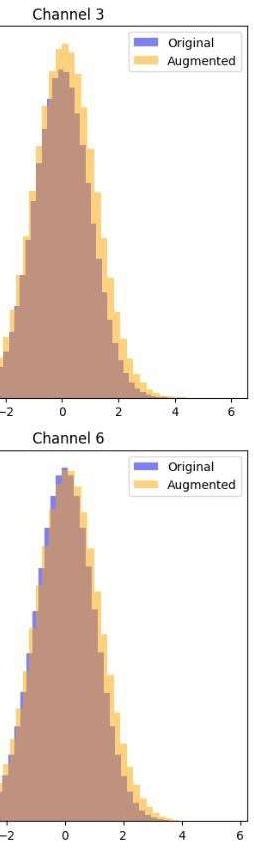
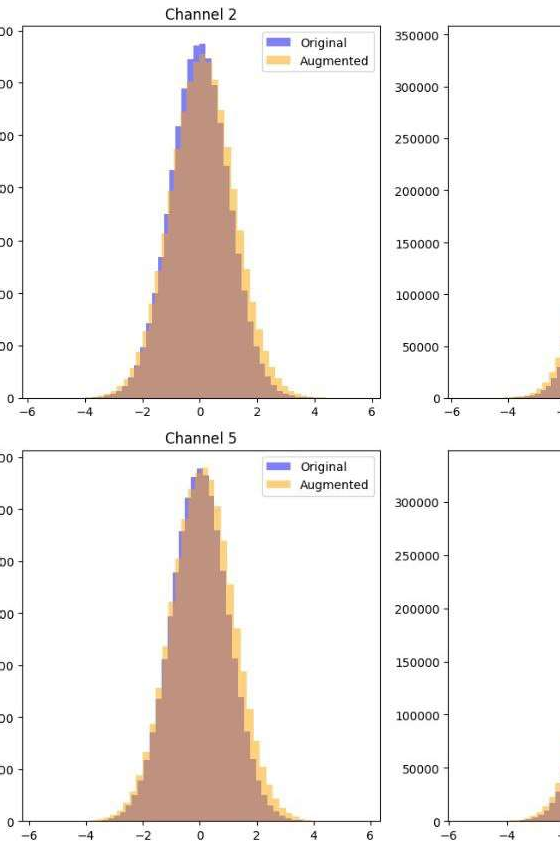
The code snippet below keras, which is commonly

utilizes the ImageDataGenerator class from used for real-time data augmentation during

training of deep learning models, particularly convolutional neural networks (CNNs) for image classification tasks.







The above graphs describe the shape of data in dataset after data augmentation.

## Training the model: -

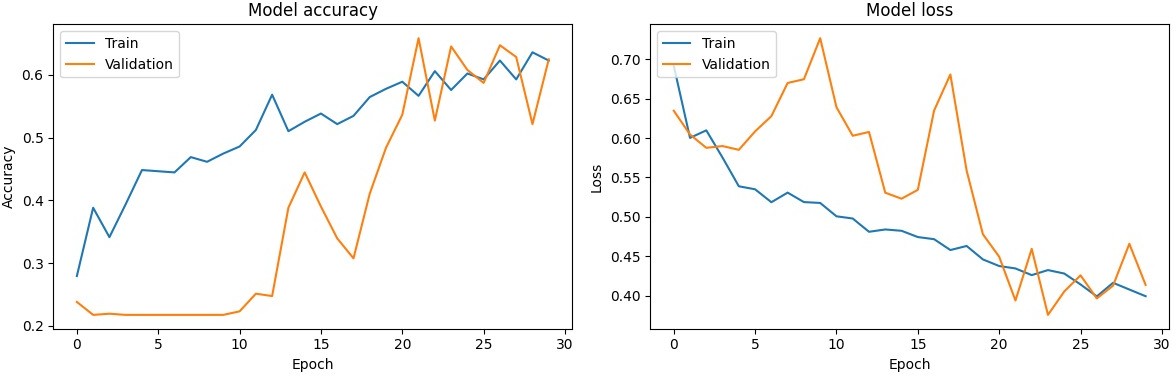
The code snippets provided below initializes the convolutional layers and defines the CNN architecture using keras. And trains the model based on augmented data and runs the epochs. This includes different hyper parameters such as: -

* Convolutional layers (16)
* Seeding (42)
* Batch\_size (32)
* Learning rate (0.001)
* Dropout Ratio (0.4)
* Epochs (50)
* Training accuracy of the model: - 71.02%

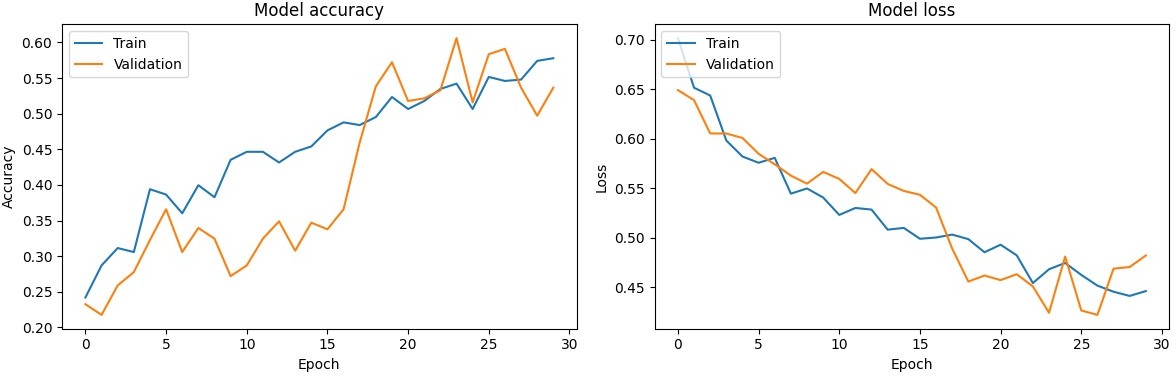
|  |  |  |
| --- | --- | --- |
| **Activation function** | **Optimizers** | **Accuracies** |
| Relu Activation function | SGD | 45.05% |
| Adam | 71.02% |

The below Graphs visulaizes the Accuracy and Loss of using the different Activation functions suchas Adam and SGD : -

**Using Adam Activation function: -**



**Using SGD Activation function:** -







# Testing the model: -

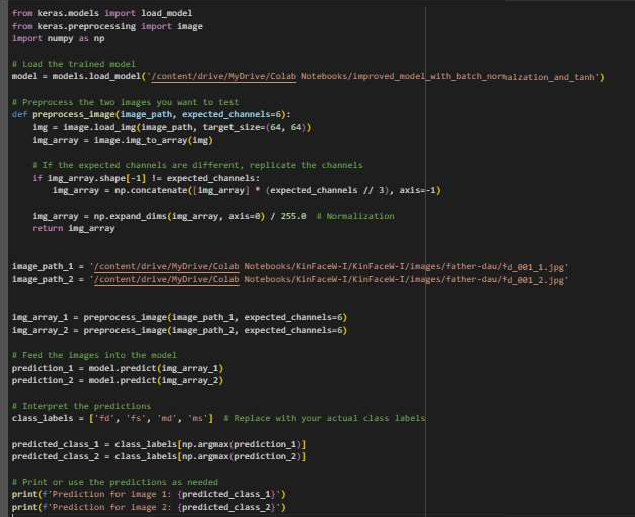
The Code snippet provided is used for testing

the model on

two specific test images (image\_path\_1) and (image\_path\_2). The

Kernel below loads the trained model and preprocess the test data and gives it for testing to the model.

* The Testing accuracy of the model: - 69.89%.



Drawbacks of using VGG16: -

* It has only 16 layers

that are not appropriate for calculating the

similarity between the images accurately.

* ResNet50 can achieve higher accuracy on tasks like image classification than VGG16 model.

1. **USING A RESNET50**

ResNet50 is a deep-convolutional neural network architecture

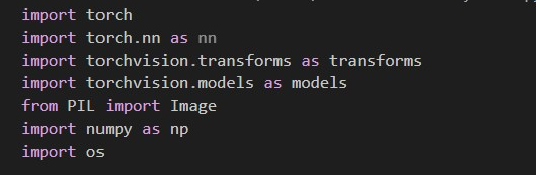
which has been trained on ImageNet Dataset.ResNet50 specifically

consists of 50 layers including convolutional layers, pooling layers, fully connected layers and skip connections.

# Importing the required libraries: -

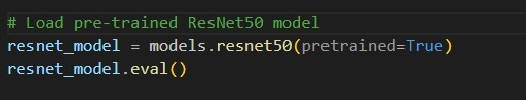
The code snippet provided below displays the are required to train the model are being imported.

libraries that



# Loading the pre-trained model: -

The code snippet provided loads the pre-trained model ResNet50.



# Data Preprocessing: -

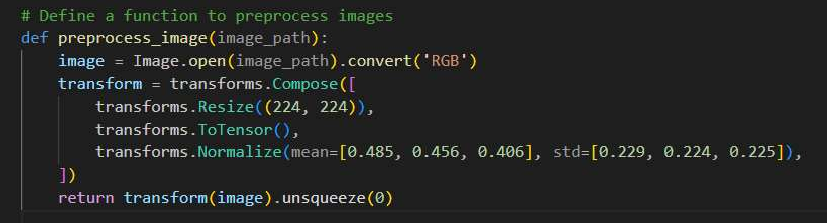
The code below preprocesses the image before passing it to

the neural network model, including conversion of grayscale images to

RGB colour space, it resizes the image into 224 X 224 pixels, it converts

the image to pytorch tensor and normalizes the pixel image tensor.

values of the



# Image Embedding and Similarity calculation: -

Image Embedding is a process in which an image

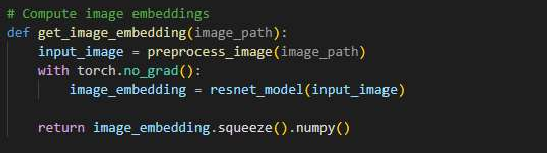
istransformed into a vector representation in a high-dimensional

space. This vector representation, known as image embedding, capture various features and characteristics of the image in a

compact form. The Embedding.

below provided code displays

about Image



Finding similarity between two image embeddings is the most

crucial step in the model. This similarity calculation is performed using

cosine similarity. And the code mentioned below calculates the average similarity of a single image embedding with all the images in the folder. Similarity score is calculated by taking two image embeddings as input, computes the dot product between them, and divides it by the product

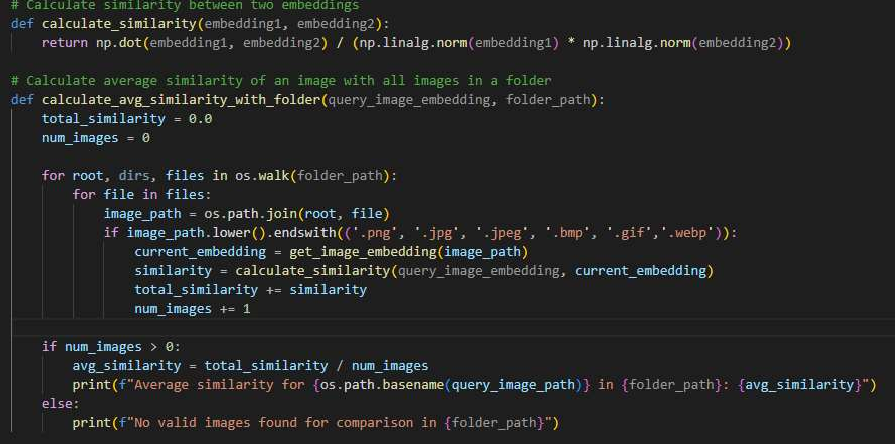
of their norms. This results in a value between -1 and 1, where a value

closer to 1 indicates higher similarity. It is also similar for calculating the

average similarity of one folder.

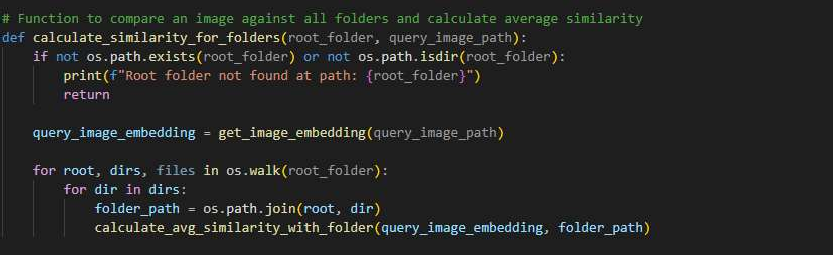
image embedding with all the

images in the



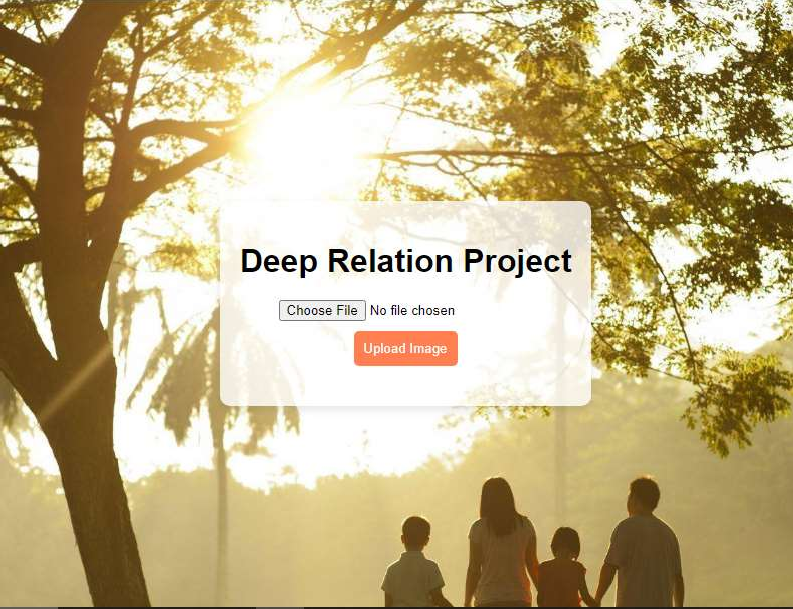
# Comparing the image similarities: -

The code snippet provides defines a function named calculate\_similarity\_for\_folders(root\_folder, query\_image\_path).This function compares a query image against all folders within a specified root folder and calculates the average similarity between the query image and all the images within each folder.

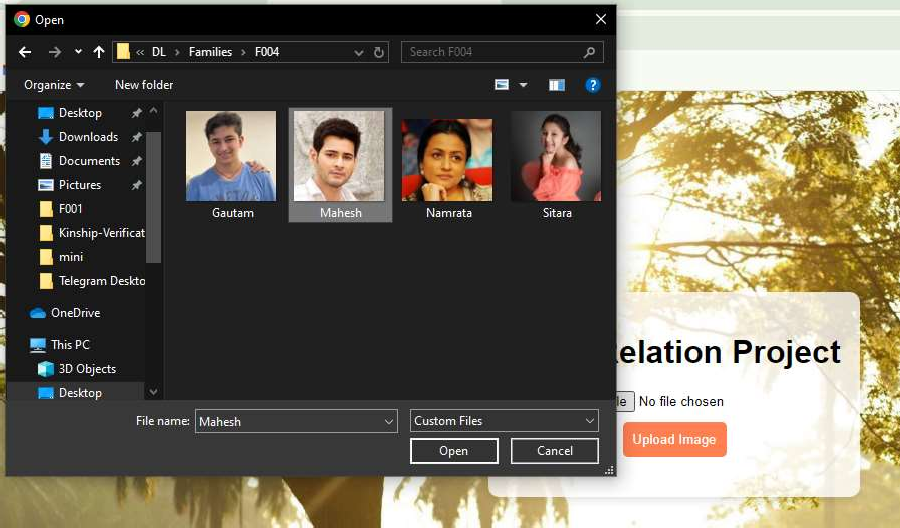


1. **IMPLEMENTATION**

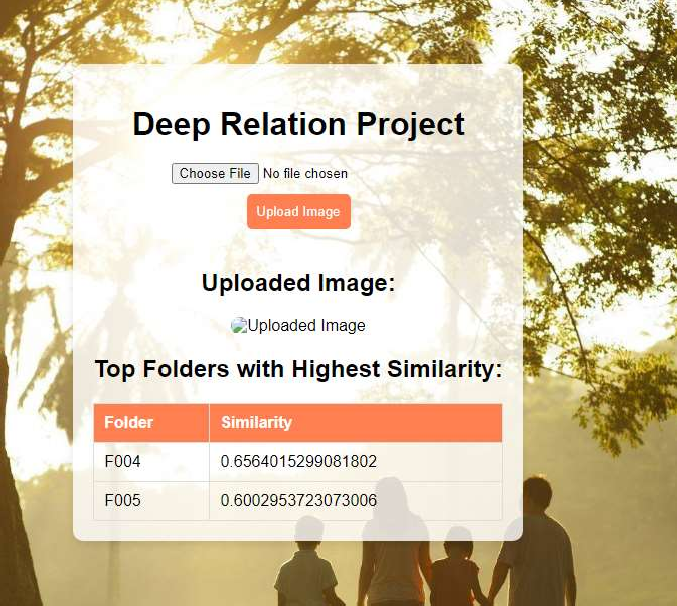
# : - UI



* 1. **: - Uploading Image**



# : - Result



### CONCLUSION

Through this project, we've demonstrated the feasibility and potential of leveraging convolutional neural networks (CNNs) and other advanced machine learning algorithms to accurately infer familial connections solely from facial features. By achieving impressive accuracy rates and exploring innovative methodologies, we've contributed to the growing field of computer vision and familial relationship inference.