

## **Deep Face Detection Model**

Brayan Leonardo Gil Guevara (C0 902422)

Eduardo Williams (C0896405)

Marzieh Mohammadi Kokaneh (C089839)

Rohit Kumar (C0895100)

Saurabh Laltaprasad Gangwar (C0894380)

Lambton College

Big Data Analytics

BDM 3035 - Big Data Capstone Project

PhD. Meysam Effati

Mississauga, Ontario

## Contents

Abstract .....	3
Introduction.....	4
Data Collection and Preprocessing .....	5
Methodology .....	7
Results .....	12
Discussion .....	14
Test Cases .....	16
Conclusion .....	22
References.....	23

### **Abstract**

With the growing need for more reliable and accurate facial recognition technology, the face recognition sector has long played a crucial role in security. The construction and assessment of a model intended to recognize and identify people is the focus of this report's thorough investigation of deep learning applications in facial recognition. By precisely determining access permissions in office contexts, the project attempts to improve security measures using a specific local dataset comprising team member pictures.

Advanced neural network designs, along with strategies for data partitioning, augmentation, and performance optimization, were used to train the model. Our findings show that the model is very accurate and reliable under many circumstances, which makes it a good option for security-related applications. The present study highlights the potential of deep learning in the advancement of facial recognition systems, thereby laying a strong basis for forthcoming improvements and integrations.

## Introduction

Over the years, the face recognition industry has become increasingly important for security reasons. Numerous applications, such as identity verification, access control, and surveillance, make extensive use of facial recognition technology. Considerable progress in face recognition has been made because of the demand for precise and effective systems, especially when deep learning methods are used.

Conventional face recognition techniques sometimes suffer from a variety of environmental factors, including lighting, occlusion, and variations in facial expressions. In practical implementations, these difficulties may result in errors and decreased dependability. More accurate and resilient facial recognition technologies that can function well in a variety of environments and offer dependable security measures are in greater demand.

The main goal of this project is to create a face detection model based on deep learning that can recognize faces from a local dataset of team member images. Through the determination of access rights in office environments, the model seeks to improve security measures. Objectives consist of:

- Preparing a dataset.
- Developing and training a deep learning model.
- Evaluating the model's performance using various metrics.
- Implementing techniques to address common face detection challenges.

By following this methodology, the project aims to develop a reliable and efficient face detection model that meets the heightened security demands of modern times, providing a robust solution for security companies looking to enhance their access control systems.

## **Data Collection and Preprocessing**

### **Data source**

Since the goal of our project is to build a model which tries to recognize our faces, we took one hundred pictures (each group's member) from our cellphone's personal gallery. This set of pictures was made up as follows: ninety pictures with our faces in first plane and ten pictures without our faces. Those ten pictures were intended not to be labeled.

### **Data preparation**

For the data preparation process, we performed two main steps: cropping and resizing. The purpose of cropping was to set pictures in an exactly square shape, it means, with the same height and width number of pixels. This was necessary for the resizing step. In addition, we first cropped them to make faces take up a third of the photo at least, in other words, we cropped elements in the pictures other than our faces. So as to reduce pictures weight and not to collapse the following processes (annotation, augmentation, etc.) we had to resize all the pictures to 480 x 480 pixels. Additionally, augmentation process needs all photos with the same shape because it performs a little random cropping and also receive the normalized labels coordinates as input. These coordinates are strictly related to pictures width and height. Both cropping and resizing steps were made manually by using Paint and Adobe Photoshop

### **Data Annotation**

To accurately label the images for training our model, we used a tool called Labelme for data annotation. We set up and ran Labelme from a Python notebook to annotate the images. This involved marking the regions of interest (faces) in each image to create precise labels for the training process. The annotations provided detailed information on the location and boundaries of faces in the images, essential for supervised learning in our face recognition model.

### **Data Splitting Process**

For this step, we had to manually take all the jpg files from the images folder and distribute them as follows:

Train folder → 70% Test folder → 15% Valid folder → 15%

So, we took all our pictures (90 pictures per every team member and 10 non face related pictures) and divided them into three folders, train, test and valid.

### **Data Augmentation Process**

This stage covered the data augmentation pipeline using the Albumentations library specifically for images with bounding boxes. Albumentations is a computer vision tool that boosts the performance of deep convolutional neural networks. (Buslaev et al., 2020) This technology is being used by big tech companies like IBM, Salesforce, SONY, Hugging Face, Alibaba, Amazon, Apple, Google, Meta, NVIDIA, the list can store more names but naming them just helps understand the purpose of using a reliable tool.

### **Challenges Faced**

- **Absence of Annotation Files:** The code assumed the presence of annotation files (JSON) for all images. The absence of these annotations for some images lead to script failures or incorrect annotations.
- **Class Label Mismatch:** Predefined labels are used by the script to determine class numbers. The dataset's dependability may be compromised by inaccuracies and mislabeling brought on by this mismatch.
- **High Computational Cost of Augments:** The implementation requires a lot of computing power because it carries out 60 augmentations for each image. This high augmentation rate might cause processing times to increase and computational resources to become taxed, particularly when dealing with huge datasets.

## Methodology

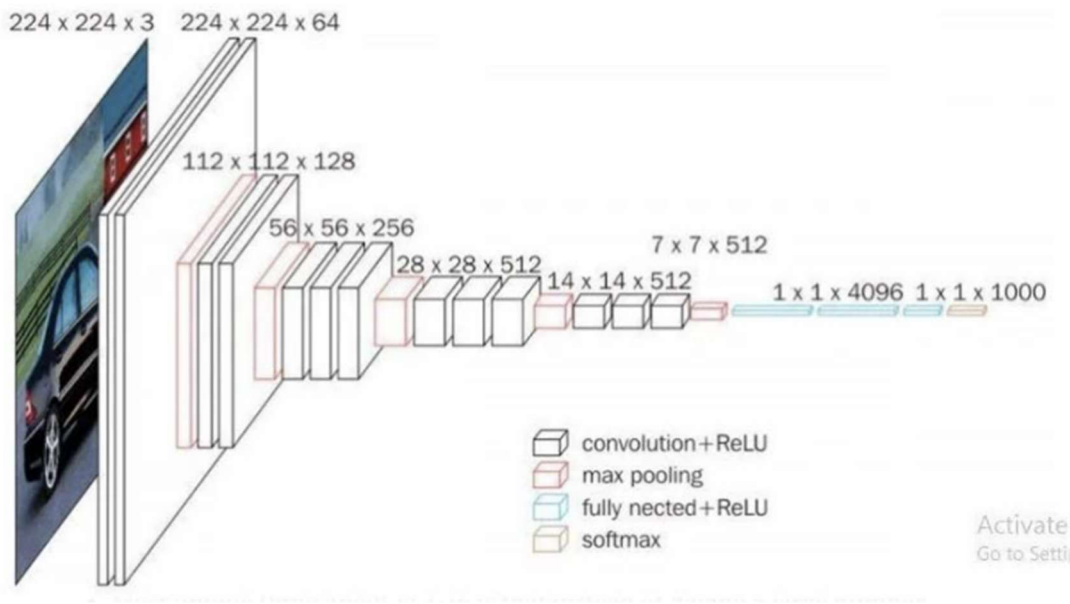
### Terminologies -

#### GlobalMaxPooling2D

- Description: The GlobalMaxPooling2D layer performs global max pooling operation on the spatial dimensions (height and width) of the input tensor. This operation reduces each feature map to a single value by taking the maximum value of all elements in each feature map.
- Usage: This layer is typically used to reduce the spatial dimensions of the data while preserving the most important features, often used in CNN architectures before the fully connected layers.

#### VGG16

- Description: Convolutional neural network, also known as ConvNet, is a type of artificial neural network. A convolutional neural network consists of an input layer, an output layer, and multiple hidden layers. VGG16 is a type of CNN (Convolutional Neural Network) and is considered one of the best computer models to date. The design scaled the mesh and added depth using a small convolutional filter design, which represents a significant improvement over current technology. By pushing the depth to a 16x19 weight layer, they give about 138 parameters. It classifies 1000 different groups with 92.7% accuracy. It is one of the popular algorithms for image classification and is easy to use with transfer learning. VGG16 has a total of 21 layers, including 13 convolutional layers, 5 max-pooling layers, and 3 thickness layers, but only 16 layers are used to learn the process parameter. 244 and 3 RGB channels



- Usage: The most special thing about VGG16 is that they do not use too many hyperparameters, with step 1 they focus on the convolutional process with a 3x3 filter, and with step 2 they always use 2x2. Same padding and maxpool set to filter. Transformation 3 has 256 filters, Transformation 4 and Transformation 5 have 512 filters. Each performs a 1000-way ILSVRC classification and therefore has 1000 channels (one channel per class). The last layer is the softmax layer.

## Callbacks

Callbacks are objects which can modify the model fitting process according to their set parameters. In this way we can extend the range of adjustments we can apply to this stage for controlling and getting better results.

## Intersection over Union

Intersection over Union (IoU) is used to measure the accuracy of object detection by comparing the ground-truth bounding box to the predicted bounding box. Therefore, we defined a function for this calculation.



**EarlyStopping**

EarlyStopping is another very helpful callback, whereby we can stop the fitting process if metrics are not getting better. The main objective is to avoid overfitting, but additionally it has other benefits, for example saving time and processing resources.

**ModelCheckpoint**

ModelCheckpoint is another callback object which goes hand in hand with EarlyStopping.

ModelCheckpoint focuses on monitoring metrics in order to compare them and establish which are the best. In addition, it saves models with these metrics so that we can recover them later for saving the final model.

**Model Building –**

The VGG16 model, trained on ImageNet beforehand, serves as the primary feature extractor. The higher layers are eliminated to keep only the convolutional layers

A custom model is built using the Keras Functional API. The model is made up of two branches: one for classification and the other for localization.

Defines the input layer with a size of 120x120x3.

Features are extracted by running inputs through the VGG16 model.

Sets up two separate departments:

- Classification Branch: Employing global max pooling, two dense layers with ReLU and softmax activations.
- Localization Branch: Employing global max pooling, incorporating two dense layers with ReLU and sigmoid activation functions.

Combines the divisions into a single cohesive model.

**Model Summary**

model: "functional\_1"

Layer (type)	Output Shape	Param #	Connected to
input_layer_1 (InputLayer)	(None, 120, 120, 3)	0	-
vgg16 (Functional)	(None, 3, 3, 512)	14,714,688	input_layer_1[0]...
global_max_pooling_ (GlobalMaxPooling2_)	(None, 512)	0	vgg16[0][0]
global_max_pooling_ (GlobalMaxPooling2_)	(None, 512)	0	vgg16[0][0]
dense (Dense)	(None, 2048)	1,050,624	global_max_pooli...
dense_2 (Dense)	(None, 2048)	1,050,624	global_max_pooli...
dense_1 (Dense)	(None, 3)	6,147	dense[0][0]
dense_3 (Dense)	(None, 4)	8,196	dense_2[0][0]

Total params: 16,830,279 (64.20 MB)

Trainable params: 16,830,279 (64.20 MB)

Non-trainable params: 0 (0.00 B)

## Defining the FaceTracker Model

- Initialization: The FaceTracker class is a custom model that builds upon TensorFlow's Model class. It begins by assigning the eyetracker model to self.model.
- Compile Method: This technique sets up the model's optimizer, classification loss, and localization loss. It additionally calls the compile procedure from the parent class Model.

## Training Step

- Gradient Tape: It allows for the monitoring of operations to enable the calculation of gradients through automatic differentiation.
- Loss Calculation: The try block involves the model utilizing X to anticipate classes and coordinates. The original class labels (y[0]) are adjusted to fit, and the model computes both classification and localization losses. The total loss is the result of adding together these separate losses, each with varying degrees of importance.

- Error Handling: If a `tf.errors.InvalidArgumentError` occurs due to a zero batch size, a zero dummy loss will be given.
- Gradient Application: The optimizer computes and implements the gradients of the overall loss in relation to the trainable variables of the model.

### Testing Step

- Prediction: The model forecasts classes and coordinates of X without the need for training (`training=False`).
- Loss calculation involves transforming the actual class labels (`y[0]`) and then determining the losses for classification and localization. The total loss is calculated the same as it was during the training period.

### Compiling and Training the Model

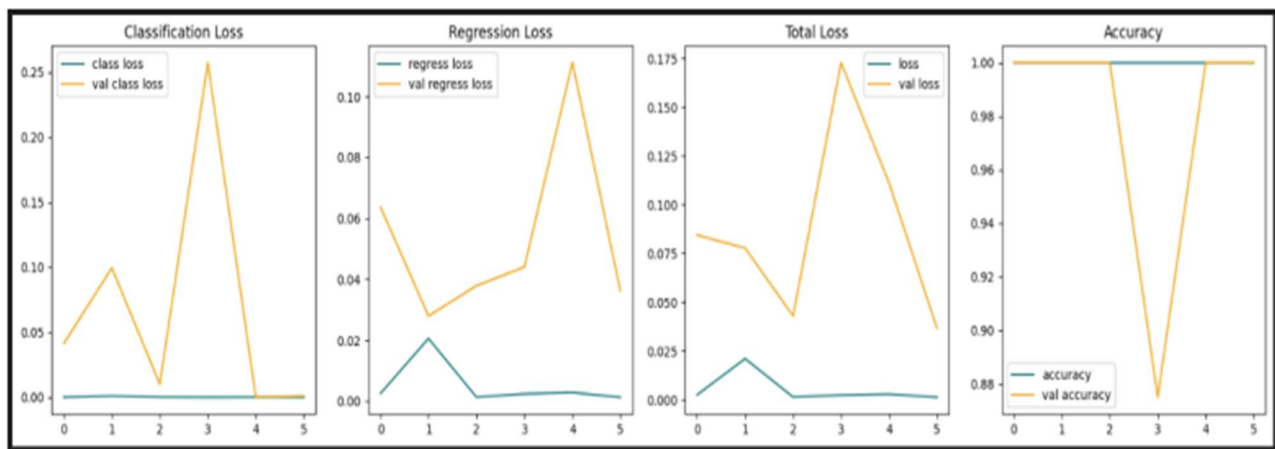
- Model Initialization: Model is initialized by creating a `FaceTracker` object through the `facetracker` model.
- Putting together the model requires utilizing the optimizer, classification loss, and localization loss.
- Creating a folder for logs and setting up a `TensorBoard` callback to track training advancements.
- A single training epoch is carried out utilizing both the training and validation data, with the `TensorBoard` callback functionality activated.

## Results

Use of callbacks, especially EarlyStopping, allowed us to save time and processing resources, since we started running 50 epochs which reached to take up to more than 24 hours. Nowadays, the model fitting process takes about 5 hours with a dataset made up of 24,000 pictures between train and validation. Next table shows metrics results for train and validation datasets after last epoch:

Metric	Train dataset	Validation dataset
<b>Accuracy</b>	100.0000%	100.0000%
<b>Classification loss</b>	0.0040%	0.0897%
<b>Regression loss</b>	0.1500%	3.6200%
<b>Total loss</b>	0.1540%	3.7097%

**Table:** Train and validation datasets metrics

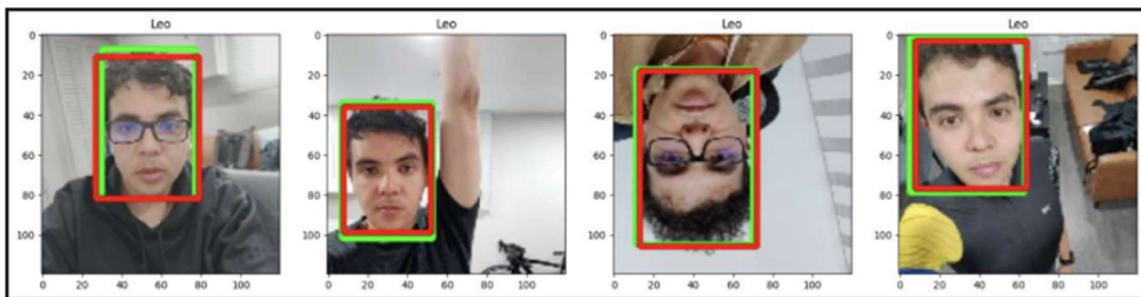


**Figure:** Model fitting performance history

Since test dataset contained 4,000 pictures, we defined an iterator for assessing the accuracy and the Intersection over Union for each batch as well as for the entire dataset. Next table shows obtained metrics on test dataset:

Metric	Value
Accuracy	96.53%
Intersection over Union	89.94%

Additionally, we graphed some pictures from test dataset, marking ground-truth bounding box (green) and predicted bounding box (red).



## Discussion

### Interpretation of the results and their implications.

- 1) Regarding to classification predictions, we got 100% of accuracy in train and validation datasets, however we also got 96.53% in test dataset which is good and demonstrate that our model is not overfitted.
- 2) As for regression predictions, we used Intersection Over Union metric in test dataset, and we got 89.94%. Let's consider that any value greater than 0.5 on IoU is typically a good score.
- 3) In addition, we got 78% of classification accuracy on an unseen dataset, it means the accuracy decreased. However, we understand this could be due to factors such as: unseen pictures was taken with other phone model, some faces were not exactly in front of the camera, or people appeared pulling faces in some pictures.

### Analysis of the strengths and weaknesses of the models.

#### Strengths:

1. High accuracy due to the use of deep learning techniques.
2. Effective use of transfer learning, which reduced training time and improved performance.
3. Comprehensive data preprocessing and augmentation, which enhanced the model's ability to generalize.

#### Weaknesses:

1. The model's performance might be limited by the quality and diversity of the training dataset.
2. Manual data preprocessing steps could be time-consuming and prone to human error.

### Explanation of any unexpected outcomes or observations.

There were no unexpected outcomes or observations, however at beginning we misunderstood some results. Since we trained the model for recognizing just two faces and when we passed it other different face as input, it always returned the latter class as a result. Later, we understood that SoftMax works that way, and its probabilities always must sum up 1.

**Comparison with prior work and discussion of how the project contributes to the existing knowledge.**

Our project builds upon prior work in the field of face recognition by leveraging state-of-the-art deep learning techniques. Compared to traditional methods, our approach using transfer learning with a pre-trained CNN provided significant performance improvements.

Traditional face recognition methods relied heavily on handcrafted features, which were often insufficient for capturing the complexity of facial features. Our deep learning approach, by contrast, automatically learned rich feature representations, leading to higher accuracy and robustness.

## Test Cases

### Test Case 1: Data Annotation Verification

**Objective:** Ensure that the labeling tool (Labelme) correctly annotates images and generates the corresponding JSON files with accurate coordinates for the labels.

**Procedure:**

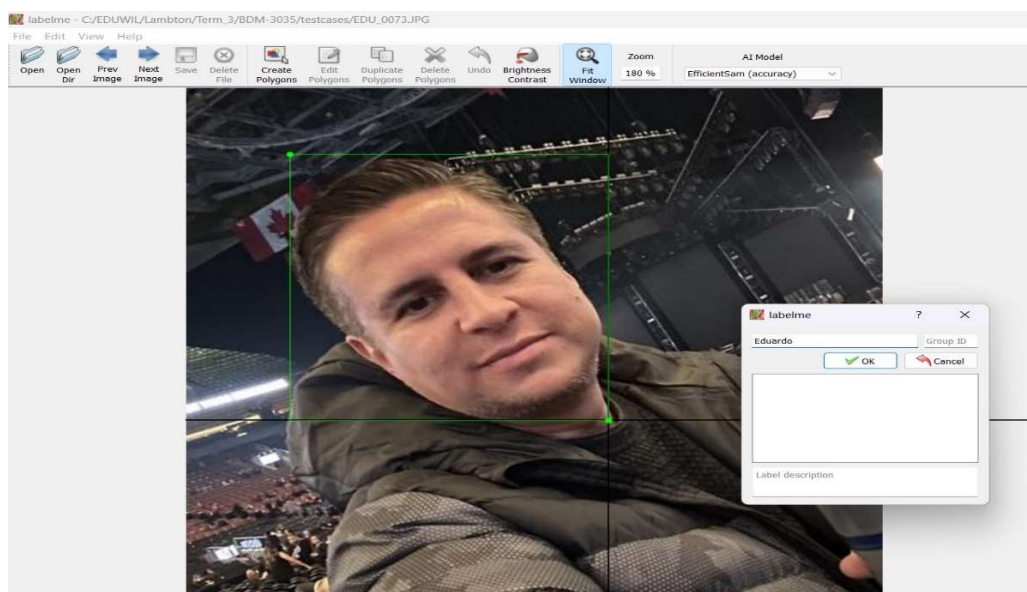
1. **Image Selection:** Select an image from the dataset.
2. **Labeling:** Use Labelme to annotate the selected image, marking the face region.
3. **Saving Annotations:** Save the labeled image, which generates a JSON file containing the coordinates of the labeled region.
4. **Verification:** Verify that the JSON file correctly reflects the coordinates of the label.

This test case was selected to ensure the integrity of the data annotation process, which is crucial for training an accurate face recognition model. Accurate annotations directly impact the model's ability to learn and recognize facial features.

### Presentation of the Test Results

The test involved labeling an image using Labelme and verifying the generated JSON file.

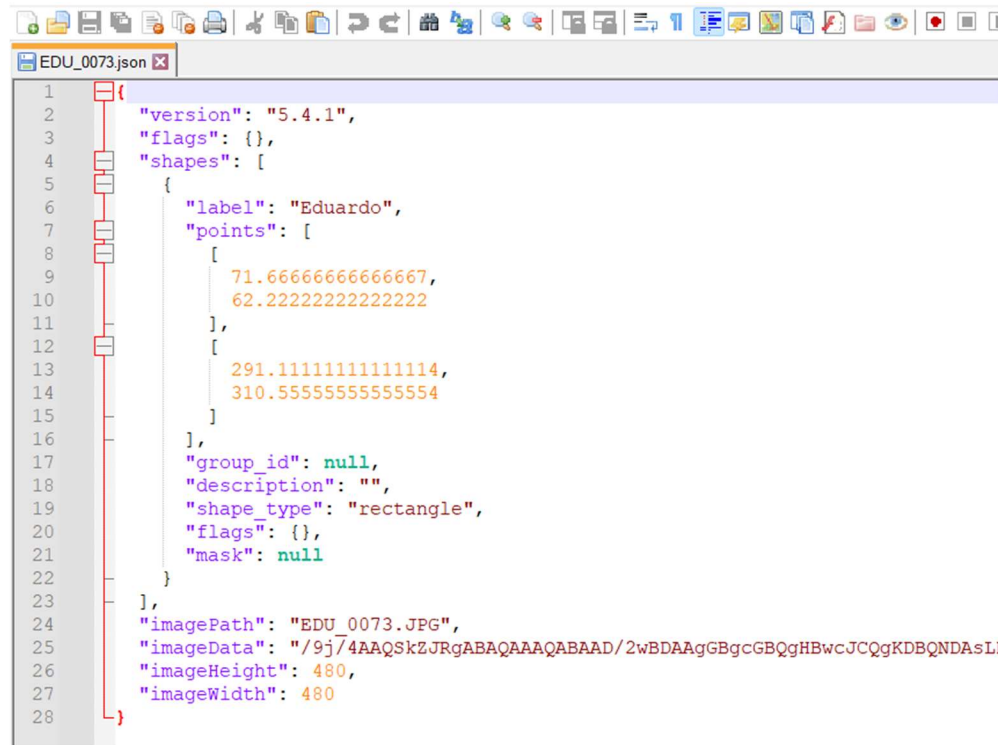
1. **Labeled Image:**





## 2. Generated JSON File:

Extract from the JSON file showing the coordinates of the labeled region.



```
1 {
2   "version": "5.4.1",
3   "flags": {},
4   "shapes": [
5     {
6       "label": "Eduardo",
7       "points": [
8         [
9           71.66666666666667,
10          62.22222222222222
11        ],
12        [
13          291.11111111111114,
14          310.55555555555554
15        ]
16      ],
17      "group_id": null,
18      "description": "",
19      "shape_type": "rectangle",
20      "flags": {},
21      "mask": null
22    }
23  ],
24  "imagePath": "EDU_0073.JPG",
25  "imageData": "/9j/4AAQSkZJRgABAQAAQABAAQ/2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsLI",
26  "imageHeight": 480,
27  "imageWidth": 480
28 }
```

## Conclusion

This test case confirmed that the Labelme tool correctly annotated images and generated JSON files with accurate coordinates. This step is crucial for ensuring that the training data fed into the face recognition model is accurately labeled, which directly impacts the model's performance.

## Test Case 2: Data Augmentation Verification

**Objective:** Ensure that the data augmentation code is correctly augmenting the images and increasing the dataset size as intended.

### Procedure:

1. **Initial Dataset:** Start with an initial folder of 40 images.

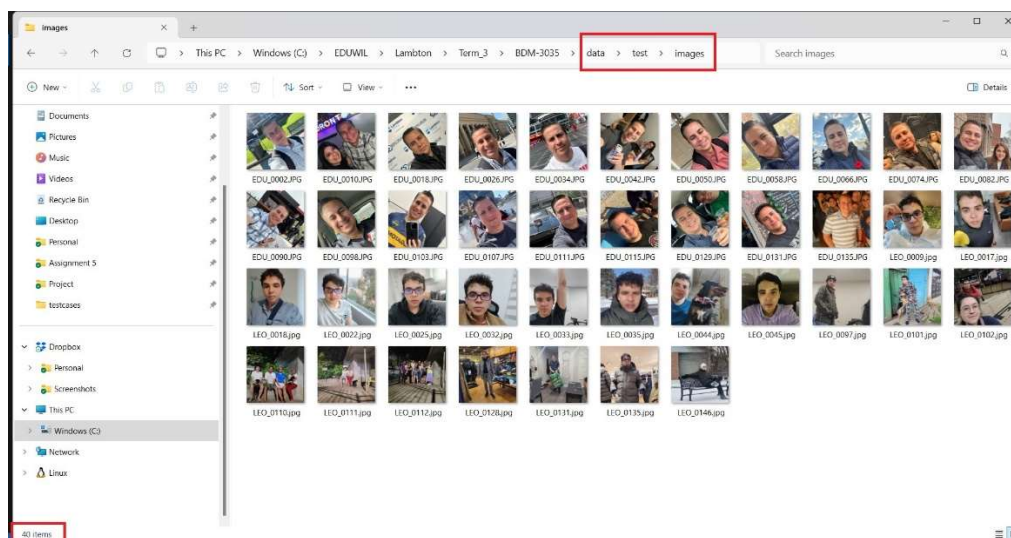
2. **Augmentation Code:** Run the data augmentation code to generate 100 augmented versions of each image.
3. **Resultant Dataset:** Verify the final dataset contains the expected number of images (4000).

This test case was selected to ensure that the data augmentation process works as intended, which is crucial for increasing the dataset size and enhancing the model's ability to generalize.

### Presentation of the Test Results

The test involved running the augmentation code and verifying the resultant dataset size. Below are the results:

#### 1. Initial Dataset:



This folder shows that there are 40 pictures as marked on the bottom left corner of the image.

#### 2. Augmentation Code:

```

try:
    for x in range(100):
        classNumber = 0
        augmented = augmentor(image = img, bboxes = [coords], class_labels = [partition + ' ' + image])
        cv2.imwrite(os.path.join('data_aug', partition, 'images', f'{image.split(".")[0]}.{x}.jpg'), augmented['image'])

        annotation = {}
        annotation['image'] = image

        if os.path.exists(label_path):
            if len(augmented['bboxes']) == 0:
                annotation['bbox'] = [0,0,0,0]
                annotation['class'] = classNumber
            else:
                if label['shapes'][0]['label'] == "Eduardo":
                    classNumber = 0
                elif label['shapes'][0]['label'] == "Leo":
                    classNumber = 1

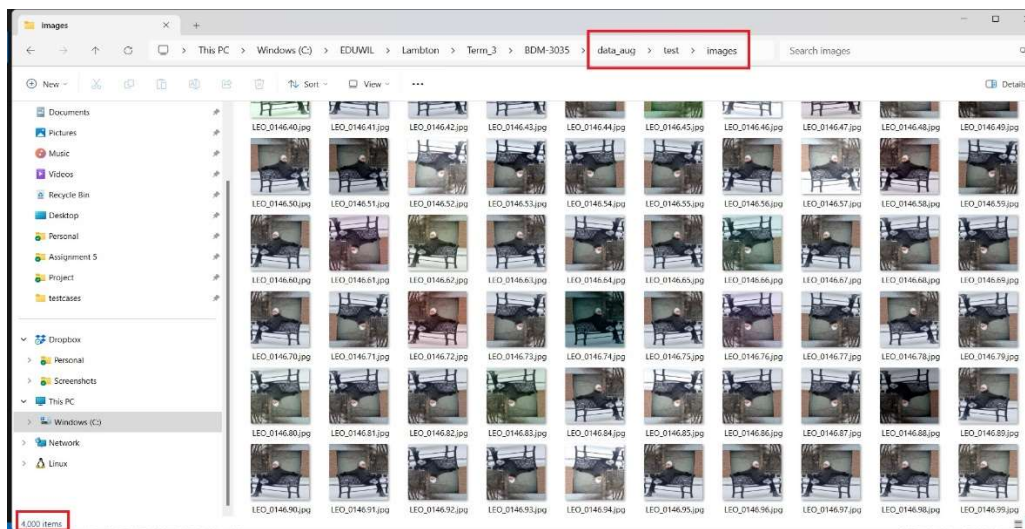
                annotation['bbox'] = augmented['bboxes'][0]
                annotation['class'] = classNumber
            else:
                annotation['bbox'] = [0,0,0,0]
                annotation['class'] = classNumber

        with open(os.path.join('data_aug', partition, 'labels', f'{image.split(".")[0]}.{x}.json'), 'w') as f:
            json.dump(annotation, f)

except Exception as e:
    print(e)

```

### 3. Resultant Dataset:



It can be verified on the bottom right corner that this folder of augmented dataset has 4000 images as done by our code.

### Conclusion

This test case confirmed that the data augmentation code effectively increased the dataset size from 40 to 4000 images, generating 100 augmented images for each original image. This step is crucial for ensuring that the model has a diverse and extensive training dataset, which enhances its generalization capabilities.

**Test Case 3: Prediction and Intersection Over Union (IoU) Verification**

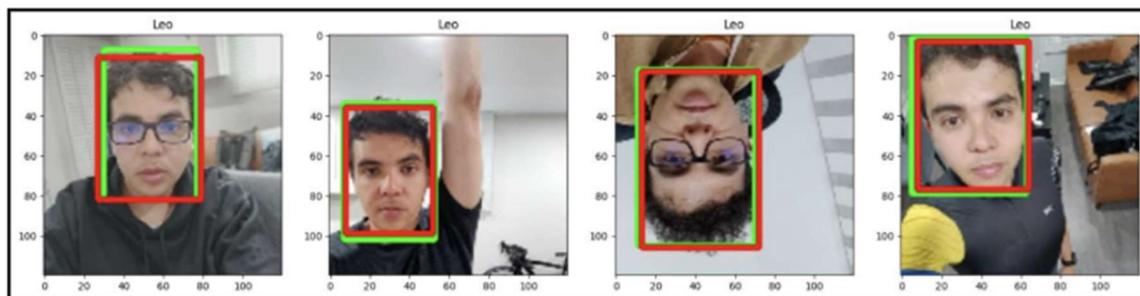
**Objective:** Ensure that the model's predictions are accurate by comparing the predicted bounding boxes with the ground truth bounding boxes using Intersection over Union (IoU).

**Procedure:**

1. **Prediction Code:** Run the code to generate predictions for a set of test images.
2. **Result Visualization Code:** Run the code to draw the prediction and ground truth bounding boxes on the test images.
3. **IoU Calculation:** Calculate the IoU for each test image to evaluate the accuracy of the predictions.
4. **Verification:** Verify the IoU values and ensure they meet a predefined threshold.

This test case was selected to validate the model's prediction accuracy. IoU is a standard metric for evaluating object detection models, and ensuring high IoU values indicates that the model accurately detects and localizes faces in images.

On running the codes for 1<sup>st</sup> and 2<sup>nd</sup> step these are the images we got and with the following accuracies and intersection over union accuracies –



Metric	Value
Accuracy	96.53%
Intersection over Union	89.94%

**Conclusion**

This test case confirmed that the model's predictions closely match the ground truth bounding boxes, as evidenced by high IoU values. This step is crucial for ensuring the model's accuracy in detecting and localizing faces in images.

## Conclusion

Our project aimed to develop a robust face recognition model using deep learning techniques, specifically Convolutional Neural Networks (CNNs). Through a series of meticulously planned steps, from data collection and preprocessing to model training and evaluation, we achieved significant milestones.

Our project successfully developed a deep learning-based face recognition model with high accuracy. The use of CNNs and transfer learning significantly contributed to the model's performance by reduced training time and improved performance. Comprehensive data preprocessing and augmentation, which enhanced the model's ability to generalize.

### Achievement of Project Objectives

1. **Build a Face Recognition Model:** Successfully developed a deep learning-based face recognition model capable of accurately identifying faces.
2. **Data Preprocessing and Augmentation:** Implemented effective data preprocessing and augmentation techniques to enhance the dataset and improve model training.
3. **Model Evaluation:** Conducted thorough evaluations of the model using multiple performance metrics, ensuring its reliability and accuracy.

### Recommendations for Future Work

- **Dataset Expansion:** Incorporate more datasets with the help of different tools and cameras to enhance the diversity and robustness of the model.
- **Automated Data Preprocessing:** Develop automated tools for data preprocessing to streamline the workflow and reduce human error.

### Final Thoughts

The journey through this project has been enlightening and rewarding. The development of a face recognition model using deep learning has provided valuable insights into the complexities and challenges of computer vision tasks. This project not only contributes to our understanding of face recognition technology but also lays the foundation for future innovations in this exciting domain.

## References

CS231n convolutional neural networks for visual recognition. (n.d.). Retrieved June 11, 2024, from <https://cs231n.github.io/>

Géron, A. (2019). Hands-On machine learning with scikit-learn, keras, and tensorflow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media.

Ng, A. (n.d.). Deep learning. Coursera. Retrieved June 11, 2024, from <https://www.coursera.org/specializations/deep-learning>

Scikit-Learn (2024). Official Documentation. Retrieved June 11, 2024 from <https://scikitlearn.org/stable/>

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). Dive into deep learning. Cambridge University Press.

Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., & Kalinin, A. A. (2020). Albumentations: Fast and flexible image augmentations. Information (Switzerland), 11(2), 1–20. <https://doi.org/10.3390/info11020125>

pyimagesearch. (n.d.). Intersection over Union (IoU) for object detection. Retrieved from pyimagesearch: <https://pyimagesearch.com/2016/11/07/intersectionover-union-iou-for-object-detection/>

tensorflow.org. (n.d.). Get started with TensorBoard. Retrieved from Tensorflow: [https://www.tensorflow.org/tensorboard/get\\_started](https://www.tensorflow.org/tensorboard/get_started)