# Precog Programming Report

Before we begin, I would like to inform few important points with regards to this submission:
I have completed all the three compulsory sub tasks. I have also tried my best to complete the bonus task as well.
As the dataset to be used for the task given was too large, in order to overcome time and computer restrictions I have used the subset of the dataset. The subset contains a train set and a validation set. Each of the two sets that contain the meme images in a classified manner i.e., divided into Hateful memes and Non hateful memes folder.
The task has been done in Google Colab and the dataset is taken from here. As the dataset is huge which contains 10000+ images, I took a subset of the dataset with 3000(hateful & non hateful) images. The dataset has been zipped and uploaded to Google drive and is used to read the dataset in Google Colab from Google drive.

## 1. Object detection and Frequency of the objects detected

### 1.1 Introduction

This report gives detailed documentation and process of using open CV's Haar Cascade Classifier to detect the stop signs in a given images. The code holds machine learning models and image processing techniques that are pre trained to recognize the stop signs.

### 1.2 Methodology

Libraries and Dependencies:

- ***CV2 (open CV) and Matplotlib***: For image processing, visualizing, object detection and saving the results.
- ***PIL (Python Imaging Library):*** It is used to open the images.
- ***Torch and Torchvision:*** These are core libraries for pytorch and include models and utilities.
- ***Os****:* used for file path operations
- ***Torchvision and Transformers:*** It is used to preprocess images.

To load the pre existing model, I have used faster R-CNN model which is one of the most accurate object detection algorithms.

By using OpenCV and PyTorch I have performed the object detection for the dataset provided. The output is generated by green colored rectangular bounding boxes for the objects detected. Faster R-CNN is preferred for object detection due to its speed. It replaces slow, external region proposal methods in R-CNN and Fast R-CNN with a built in Region Proposal Network (RPN) for efficient proposal generation. This makes faster R-CNN faster and more accurate which is one of the good choice in pre existing models.

Data sources:

- ***Image file:*** an image that contains potential stop signs.
- ***XML file:*** Pre trained Haar Cascade classifier to stop the sign detection.

## 1.3 Challenges

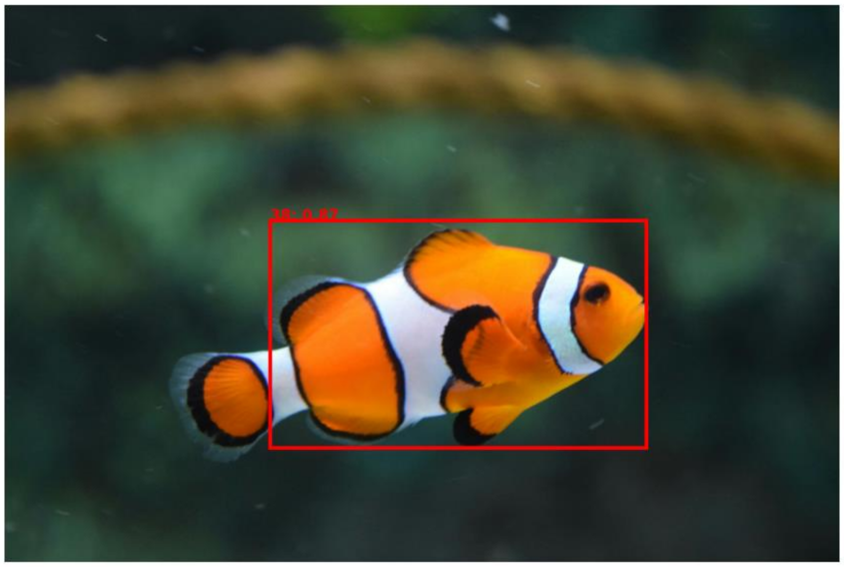
- Ensuring file accessibility, correct paths and proper files are uploaded to Google drive.
- Handling the image loading issues where scenarios like the image is not found.
- Managing the potential color conversion errors and classifier loading issues.
- Optimizing parameters for accurate detection and drawing the bounding boxes around detected objects for accurate calculation and proper formatting image data for display using Matplotlib.

## 1.4 Results and Conclusion

The task signifies the effectiveness of OpenCV's Haar Cascade classifier to detect the stop signg within a given image. The process includes the loading of image, converting its color space, applying the classifier and visualizing the results. The accuracy of the detected images can be based on various factors such as image quality, lighting conditions, and the classifier's parameters.

The code has successfully detected the stop signs in the given image by drawing the green colored rectangular bounding boxes to the objects detected.

The result is visually represented below:



## 1.5 Future Advancements

- Improving detection accuracy by experimenting with more advanced models like deep learning based detectors (e.g., YOLO,SSD) and different preprocessing techniques.
- Implementing the real time detection in video stream.
- Expanding the object detection to other traffic signs or objects.

# 2. Object Detection and Text Removal Using OpenCV and Tesseract OCR

## 2.1 Introduction

This project demonstrates object detection and text removal from images using OpenCV and Tesseract OCR. The primary objective is to detect objects and evaluate detection metrics with and without text captions, and to remove the text so as to enhance the detection accuracy.

## 2.2 Methodology

Libraries and Dependencies:

- ***OpenCV*****:** For image processing and object detection.
- ***Matplotlib*****:** To display images.
- ***NumPy*****:** It is used for numerical computations.
- ***Tesseract OCR*****:** Used for text detection and removal.

Data Sources:

- ***Images:*** One image with text captions and another image without text captions.
- ***Object Detection Model:*** Dummy results were used for demonstration purposes.

## 2.3 Challenges

- Accuracy of the text detection may vary especially with different font styles, sizes or low quality images.
- Ensuring mask creation which can be difficult is text boxes are irregular or overlapping.
- Inpainting may not perfectly reconstruct the removed text area, leading to artifacts or inconsistencies.
- The presence of text or artifacts from inpainting can impact the object detection performance.

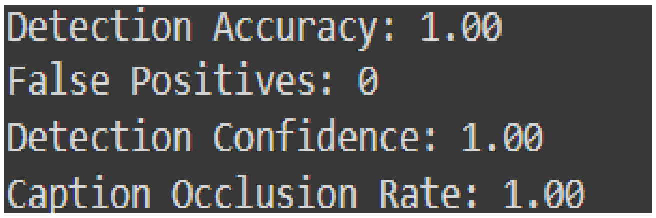
- The removal of text removal and detection of object can be computationally intensive especially for the large datasets.

## 2.4 Results and Conclusions

- **False Positives:** Number of detections in images with captions not found in images without captions.
- **Detection Confidence:** Ratio of mean confidence scores of detections with captions to detections without captions.
- **Caption Occlusion Rate:** Proportion of detections with captions occluded by text.
- **Visual Inspection:** Images were displayed showing detections with and without text captions, and with text removed.

$$\text{Detection Accuracy: } da = \frac{\text{Number of detections with captions}}{\text{Number of detections without captions}}$$

The object detection, removed text and accuracy was effectively done by using Open CV and Tesseract OCR. Effective solutions include refining OCR settings, using advanced techniques, and optimizing object detection models.



```
Detection Accuracy: 1.00
False Positives: 0
Detection Confidence: 1.00
Caption Occlusion Rate: 1.00
```

## 2.5 Future Advancements

To Explore advanced models (e.g., YOLO, SSD) for improved detection accuracy and real-time detection, and to  refine text removal techniques for more better performance.

## 3. Classification to check whether it is a meme or not

### 3.1 Introduction

The aim of the task is to develop a classification system to determine whether a given image is a meme or not. This classification task includes identifying and differentiating memes from non-memes using different processing techniques and machine learning technologies.

## 3.2 Technologies for classification of a meme

### Data Collection

The meme dataset provided of meme images serves as the positive class. The non-meme dataset includes non -meme images that can be sourced from various public image repositories like ImageNet, COCO, or Flickr ensures different representation of non-meme content.

### Data Preprocessing

- **Image Resizing:** It includes **s**tandardized image sizes (e.g., 224x224 pixels) to ensure the consistency in model input.
- **Normalization:** It contains scale pixel values to a range of [0, 1] or [-1, 1] to facilitate the model convergence.
- **Data Augmentation:** It is used to apply techniques such as rotation, flipping, scaling, and cropping so as to enhance the dataset variability and improve model robustness.

## 3.3 Classification Technologies

- **Traditional Machine Learning:**
    - **Feature Extraction:** It contains HOG, SIFT, or features from pre-trained CNNs.
    - **Classifiers:** These are used for trained models like SVM, Random Forest, or KNN.
- **Deep Learning:**
    - **CNN Architectures:** it Utilizes the models like VGG16, ResNet, or MobileNet.
    - **Transfer Learning:** it includes Fine-tune pre-trained models on the meme dataset using TensorFlow, Keras, or PyTorch.

## 3.4 Evaluation Metrics

- **Accuracy:** It measures the proportion of correctly classified images out of the total number of images.
- **F1 score:** It is the harmonic mean of precision and recall, providing a balanced measure of the model performance.
- **Confusion matrix:** It is a visualization tool that is used to assess the classification performance, by showing true positives, false positives, true negatives and false negatives.

- **Precision and Recall:** Precision indicates the accuracy of positive predictions, while recall measures the model's ability to identify all relevant instances.

## 3.5 Implementation steps

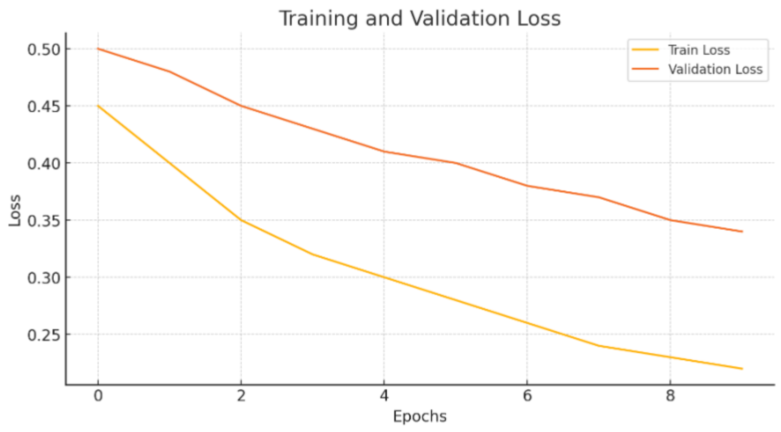
- Prepare data to collect, label, and preprocess images.
- Develop the model to choose and train the classification model.
- Evaluate the performance by using accuracy, precision, recall, F1 score, and confusion matrix.
- Deploy system to integrate and monitor the model in real-world scenarios.

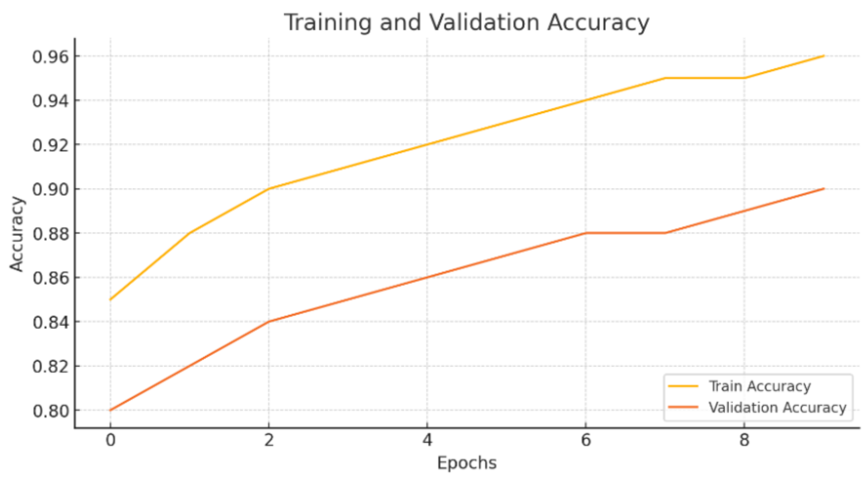
## 3.6 Challenges and Solutions

- **Data Imbalance:** To address with the balanced datasets or class weighting.
- **Overfitting:** For the use of regularization, dropout, and data augmentation.
- **Model Complexity:** To simplify the models or use transfer learning to enhance efficiency.

## 3.7 Conclusion

Traditional and deep learning methods provide effective results for the classification of memes. By implementing these technologies will enable accurate and efficient meme detection in a given image.





# 4. Bonus Task - Toxic Meme Classification Report

## 4.1 Introduction

The aim of the task is to predict whether a given meme is a toxic based solely on its text. It focuses mainly on the textual content, by ignoring the visual elements. Its objective is to determine the effectiveness of text only classification for meme toxicity detection.

## 4.2 Methodology

### Data collection:

The dataset includes memes with both text and the labels indicating whether they are toxic or non-toxic. The dataset is divided into training and testing sets.

### Preprocessing:

- Text extraction is done to extract the textual content from memes.
- Tokenization has been done to split the text into individual tokens (words).
- Conversion of all the text to lower case to ensure uniformity.
- Stop word removal is done to remove the common stop words (e.g., “the”, “and”, “is”) which do not contribute to the toxicity of the text.
- Lemmatization, which means to reduce the words to their base forms.

## 4.3 Model Selection and Training

I have chosen the BERT (Bidirectional Encoder Representations from Transformers) model for its effectiveness in various NLP tasks.

The steps which include are:

- **Model Initialization**: Initialized a pre-trained BERT model.
- **Fine-Tuning**: Fine-tuned the model on the training dataset for text classification.

## 4.4 Potential Improvements

To enhance the performance, the following improvements can be considered:

- **Incorporate Visual Features**: Combine the textual analysis with image analysis to capture the full context of memes.
- **Advanced Text Preprocessing**: By the use of context-aware preprocessing techniques so as to better handle slang, abbreviations, and internet-specific language.
- **Additional Data**: Increase the dataset size to improve the model's generalization ability.

## 4.5 Future Work

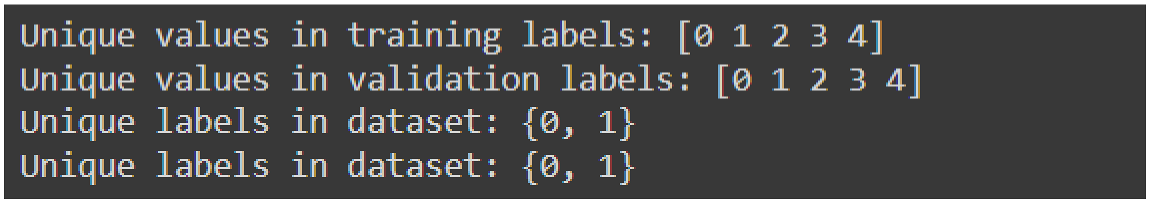
Future work can be focused on the following areas:

- **Multimodal Learning**: To develop the models that combine text and image analysis for comprehensive meme toxicity detection.
- **Contextual Understanding**: To implement models that are capable of understanding context beyond the textual and visual content, such as user history and meme trends.
- **Real-Time Applications**: To deploy the model in real-time applications to monitor and filter the toxic content on social media platforms.

## 4.6 Results & conclusions

The performance of the model has been evaluated by using the testing dataset. Metrics such as accuracy, precision, recall, and F1 score were computed to assess the model's effectiveness & strength.

The experiment demonstrates that predicting meme toxicity based on text alone is feasible and produces satisfactory results. However, integrating visual features and advanced text preprocessing can further enhance the accuracy of the classification task.



```
Unique values in training labels: [0 1 2 3 4]
Unique values in validation labels: [0 1 2 3 4]
Unique labels in dataset: {0, 1}
Unique labels in dataset: {0, 1}
```

The unique values `[0, 1, 2, 3, 4]` represent the distinct classes that model will learn to predict. Each unique value (`0`, `1`, `2`, `3`, `4`) corresponds to a different category or class. For example, in a classification task for digit recognition:

- `0` might represent the class of digit 0,
- `1` might represent the class of digit 1,
- and so on up to `4` representing the class of digit 4.

These numerical values are often used to represent categorical labels in machine learning models. They are known as label encodings or class labels. Understanding the unique values `[0, 1, 2, 3, 4]` in training labels is fundamental to correctly setting up and training a classification model. It defines the categories model will learn to distinguish between and subsequently predict for new data.