Facial Emotion Recognition

Table of Contents

[Problem Statement 2](#_Toc186460505)

[System requirements 2](#_Toc186460506)

[Hardware requirements 2](#_Toc186460507)

[Software requirements 2](#_Toc186460508)

[Dataset Details 2](#_Toc186460509)

[Solution Approach: 3](#_Toc186460510)

[Deployment 3](#_Toc186460511)

[Results 4](#_Toc186460512)

[Limitations and Future Scope 5](#_Toc186460513)

[Limitations 5](#_Toc186460514)

[Future Scope 6](#_Toc186460515)

# Problem Statement

The goal of the project is to develop a system that can accurately detect and classify human facial emotions in real time or recorded videos. This involves combining object detection with emotion recognition to create a unified pipeline that is efficient and scalable for practical applications.

Key challenges include detecting faces in complex environments, classifying subtle facial emotions, and maintaining real time performances. By leveraging YOLOv8 for face detection and training an integrated emotion classification model, the system aims to identify emotions such as *happy, sad, angry, surprised, neutral and disgust.*

# System requirements

## Hardware requirements

* **GPU:** NVIDIA With at least 4GB VRAM (e.g., NVIDIA RTX 3060 or higher) for efficient model training and inference
* **CPU:** Multi core processor for preprocessing tasks
* **Camera:** for real time video feed in live testing scenarios.

## Software requirements

* **Operating systems:** Windows 10/11 or macOS.
* **Python:** version 3.8 or higher.
* **Libraries and frameworks:**
  + **Pytorch:** for emotion recognition
  + **OpenCV:** For video processing and real time feed
  + **Numpy and Pandas:** For data manipulation and preprocessing.
  + **Flask or RestAPI:** For REST API development
  + **Matplotlib or seaborn:** For visualizing training results.

# Dataset Details

* AffectNet is a large scale facial expression dataset with over 1 million images, including 450,000 manually annotated images.
* It provides labels for seven emotions:

Neutral, Happy, sad, surprise, fear, disgust and anger.

* The dataset is split into ~283,901 training images and 3500 validation images, with additionally automatically labeled data.

# Solution Approach:

**Input :** A recorded video taken such as MP4 or AVI, as the input for the pipeline.

**Process:**

1. **Frame Extraction :** The first step is to preprocess the video by splitting it into individual frames. This allows us to process each frame independently for face detection and emotion recognition.
2. **Face detection :** For face detection, I would use YOLOv8, a state of the art object detection model, this model would process each frame and generate bounding boxes around all detected faces.
3. **Emotion Recognition:** The detected faces from the bounding boxes are cropped and passed to an emotion recognition model. This model trained on datasets like AffectNet, classifies emotions into categories such as happy, sad, angry, surprised or neutral.
4. **Annotation:** After classification, the detected emotions are overlaid on the respective frames, along with bounding boxes, to create visual annotations.
5. **Video Reconstructions:** The annotated frames are then stitched back together to form a video with annotation boxes and emotion labels for each face in each frame.

**Output:** The final output is the annotated video showing detected faces with their corresponding emotions.

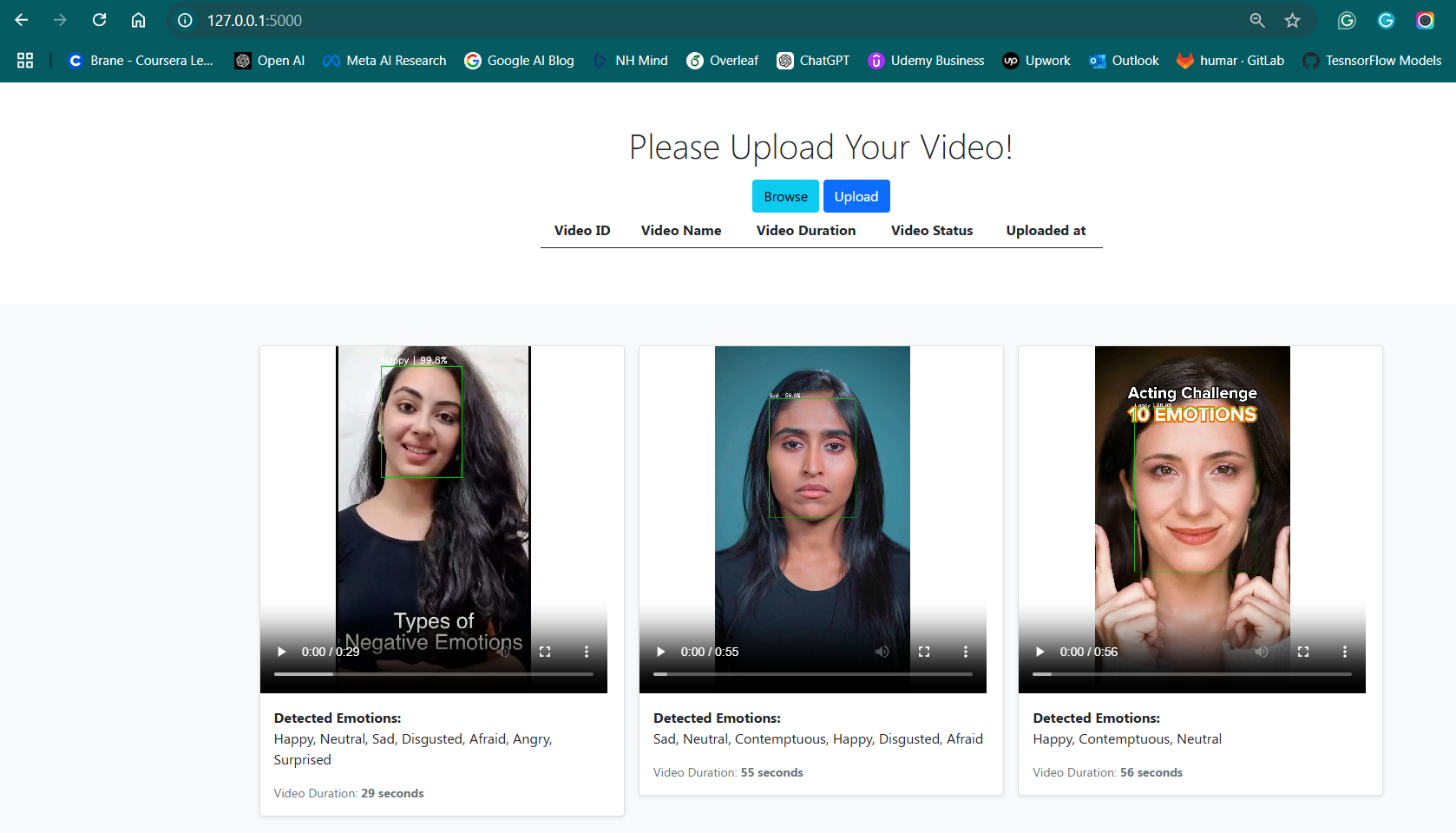
# Deployment

1. **Containerization:** I would package the entire pipeline (frame extraction, YOLO-based face detection, and emotion recognition) into a container using Docker. This ensures portability and easy deployment across platforms.
2. **REST API Creation:** I would use a web framework like Flask to create a REST API. When a video is uploaded, the API triggers the processing pipeline like Extract frames, Runs YOLO v8 for face detection, classifies emotions for detected faces, The output video after the annotated frames its re-constructs the video then sent back to the client. I have deployed the API on my own system GPU Platform for faster processing of large or high-resolution videos.

**Web Interface Development:** A simple web based UI would be built using frameworks like Angular or HTML.

The frontend communicated with REST API using HTTP requests:

1. Sends the uploaded video to the uploaded point.
2. Fetches results from the download endpoint once processing is complete.



# Results

**1. Evaluation Metrics**

The following metrics were computed to assess the model's performance:

* **Accuracy**: The proportion of correctly predicted instances out of the total instances.

**Accuracy=** Number of Correct Predictions​/ Total Number of Predictions

* **Precision**: The ratio of correctly predicted positive observations to the total predicted positive observations.

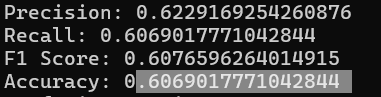
Precision= True Positives​ / True Positives + False Positives

* **Recall (Sensitivity)**: The ratio of correctly predicted positive observations to all the actual positives.

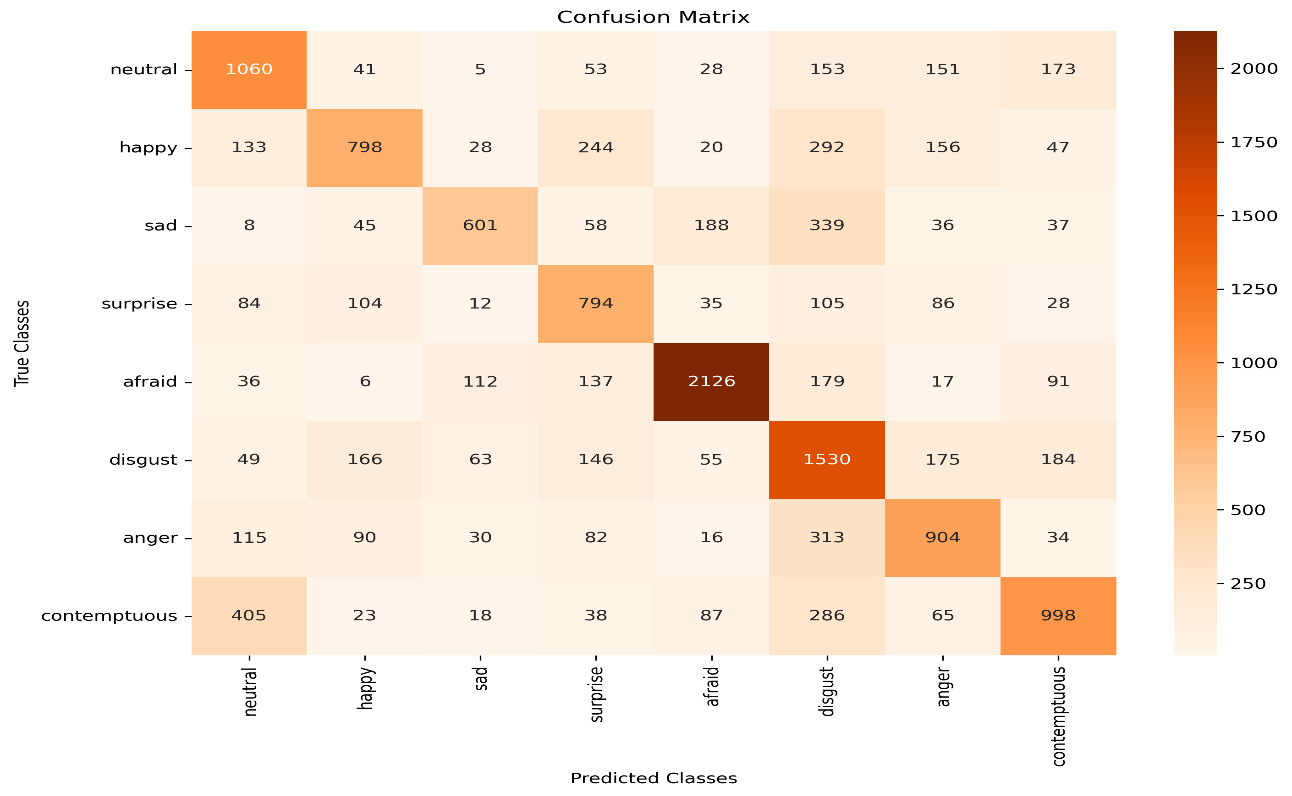
Recall= True Positives​ / True Positives + False Negatives

* **F1 Score**: The harmonic mean of precision and recall. It balances the two metrics, especially when the class distribution is imbalanced.

F1 Score=2⋅ Precision\*Recall​ / Precision + Recall



2.Confusion Matrix:



# Limitations and Future Scope

## Limitations

1. **Pre-trained Model Dependency:**
   * The results reported were obtained using a pre-trained model fine-tuned on the AffectNet dataset. While this approach demonstrates competitive performance, the dependency on pre-trained architectures might limit adaptability to other datasets without additional fine-tuning.
2. **Class Imbalance:**
   * Certain classes, such as *Afraid* and *Disgusted*, exhibit lower F1 scores and recall values. This could be attributed to class imbalance in the dataset, which biases the model toward dominant classes.
3. **Dataset-Specific Evaluation:**
   * The metrics accuracy: 0.606901%, F1 score: 0.60765%,
   * recall: 0.606901% are specific to AffectNet and may not generalize well to real-world applications without additional domain-specific adaptation.
4. **Comparison with Published Research:**
   * When compared to results from a published research paper on google <https://arxiv.org/pdf/1708.03985> AffectNet (e.g., accuracy: 0.57%, F1 score: 0.55%), our model achieves comparable or better performance. However, the paper employs additional data augmentation techniques and custom loss functions that were not incorporated in this implementation.

## Future Scope

1. **Advanced Training Techniques:**
   * Explore state-of-the-art approaches like attention mechanisms (e.g., transformers) or domain adaptation to improve performance, especially for underrepresented classes.
2. **Augmentation and Fine-tuning:**
   * Incorporate advanced data augmentation techniques such as GAN-based synthesis to balance class distribution and improve generalizability.

**3.Custom Loss Functions:**

* + Employ loss functions specifically designed for imbalanced datasets (e.g., focal loss) to enhance precision and recall for minority classes.

1. **Real-World Testing:**
   * Extend evaluation to real-world facial expression datasets to validate the robustness and applicability of the model beyond AffectNet.
2. **Benchmarking Against Published Results:**
   * While the pre-trained model achieves an accuracy of 0.606901% compared to the published research paper's 0.57%, further exploration of architectural enhancements, hyperparameter tuning, and ensembling techniques could push the performance envelope even further.
3. **Deployment Readiness:**
   * Investigate lightweight model architectures for real-time deployment in edge devices for applications such as emotion recognition in human-computer interaction systems.