CADT Anniversary 10

Cambodia Academy of Digital Technology

Master of Computer Science

Traffic Sign Recognition

Prof Dr. Chamnongthai Kosin & Dr. Kong Phutphalla

Present by: SIM Daro

Table of Contents

Abstract

Abstract

Traffic Sign Recognition (**TSR**) is a critical component in modern driver-assistance systems, enabling real-time identification and interpretation of traffic signs to ensure road safety. This technology helps drivers by recognizing various traffic signs, which is essential for informed decision-making on the road.

Problem statements:

- The Drivers may not remember the meanings of all traffic signs, causing confusion
- Misinterpreting traffic signs can result in fines or legal issues.
- New drivers or visitors may find it hard to quickly learn the meanings of different signs.
- In busy areas, drivers may miss important signs because of too much information.
- Drivers often need to understand signs quickly, which can be challenging.

The proposed solution uses various machine learning approach, combined with YOLO model for object detection, to develop an accurate (**TSR**) system. The system is designed to detect and classify traffic signs from video inputs in real-time, addressing the outlined problems by providing drivers with timely and reliable sign recognition, thereby enhancing road safety.

Task Breakdown

Planning & Task Breakdown

No	Tasks	Time	Responsibility
1	Defining Project, Scope, and Requirement	1 days	Team
2	Data Preparation - Data Collection - Data Preprocessing - Data Normalization - Data Splitting	2 weeks	Team
3	Modelling, Evaluation, Inference	6 days	Individual
4	Model Initialization for Traffic Sign Detection	1 week	Individual
5	Traffic Sign Detection and Confidence Validation		Individual
6	Traffic Sign Recognition and Post-Processing		Individual

Dataset

Dataset

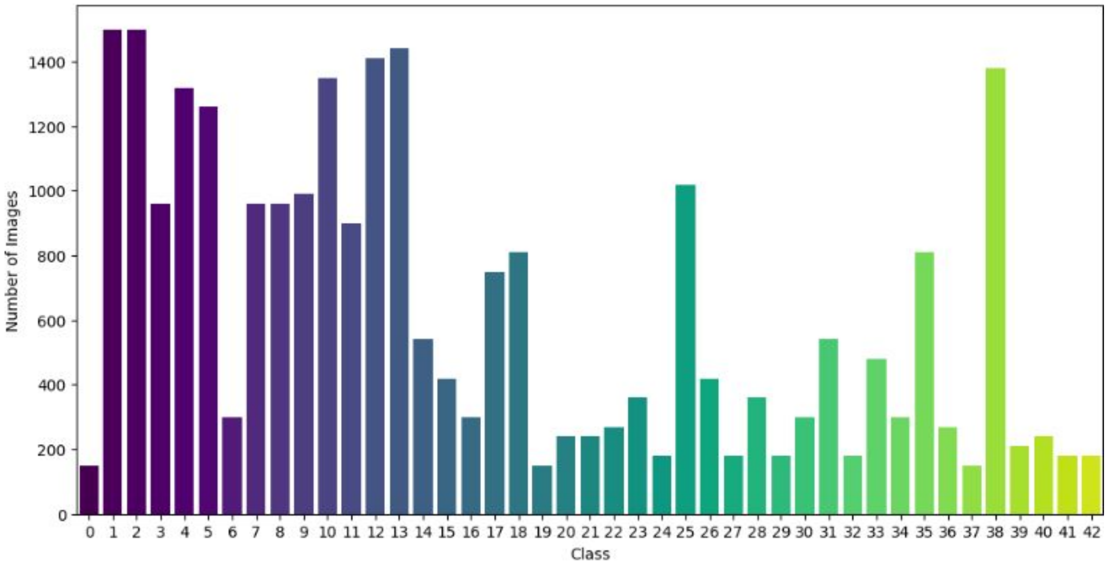
Kaggle

Kaggle lets us search for and download datasets easily. The large community also provides valuable learning resources

German Data

The German Traffic Sign dataset is a multi-class classification challenge with 43 classes and over 50,000 realistic images.

Dataset


Number of Images
0
200
400
600
800
1000
1200
1400
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42
Class

Dataset

Data Splitting

Training

Used to build and train the model (39209 Records)

Testing

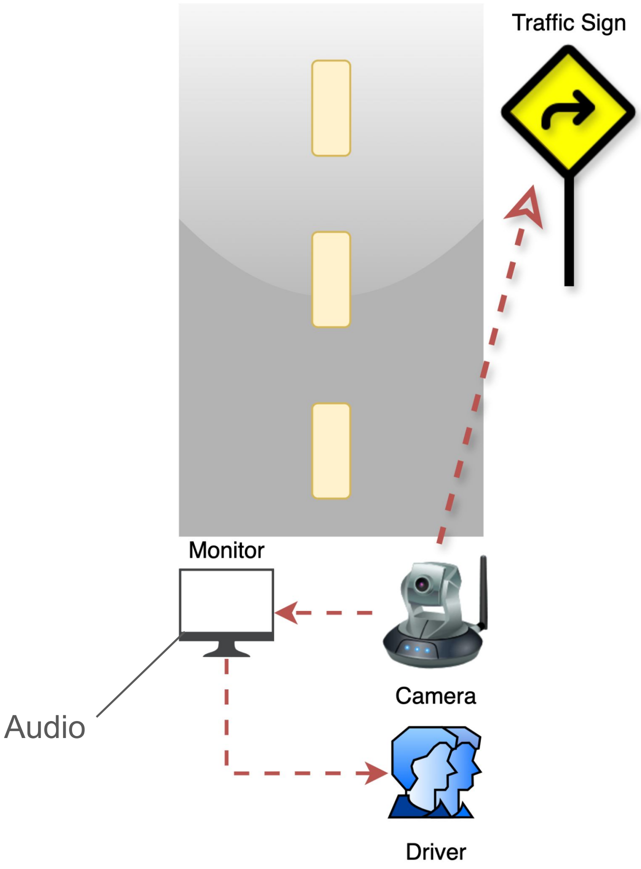
Used to evaluate model performance (12630 Records)

Implementation

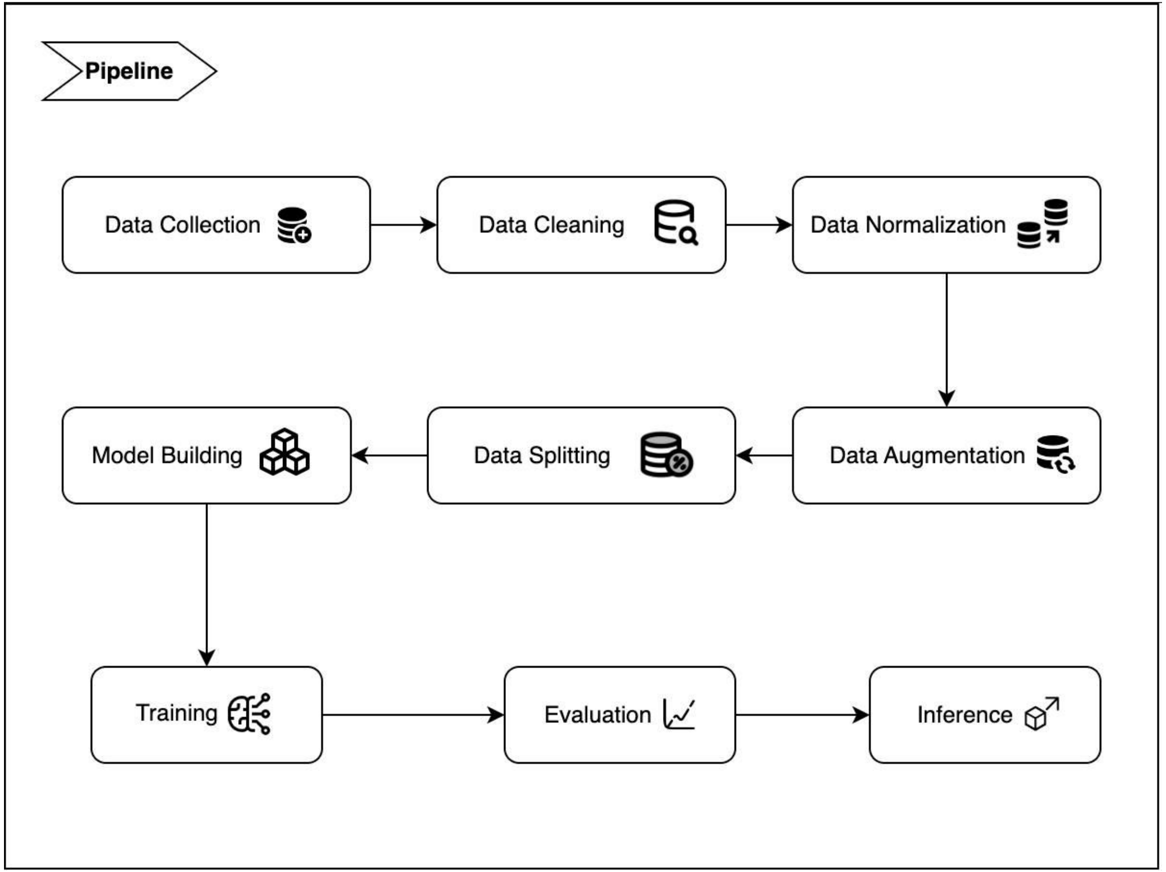
Implementation

Hardware Design

- The camera captures images and processes them to detect and recognize traffic signs in real-time.
- Upon detection, a digital display provides visual alerts to drivers and pedestrians.
- This alert system enhances road safety by ensuring clear communication of traffic information.
- The design aims to improve awareness and prevent accidents through timely visual cues.



Implementation



Algorithm Pseudocode

DataLoader

INITIALIZE GTSRB data loader with:

- `dataset_path`
- `input_size` = (224, 224)
- `batch_size` = 64
- `num_workers` = 4
- `train` = True

DEFINE data transformation sequence:

- **RESIZE** images to `input_size`
- **CONVERT** images to tensors
- **NORMALIZE** tensors using ImageNet mean and std values

LOAD GTSRB dataset:

- **IF** `train` is True, **USE** training split
- **ELSE USE** test split
- **APPLY** transformations
- **DOWNLOAD** dataset if not available locally

INITIALIZE data loader with:

- `dataset`
- `batch_size`
- `shuffle` = True
- `num_workers`

RETURN `dataset`, `data_loader`

Model Initialization

LOAD pre-trained MobileNetV3 Small model

MODIFY final classification layer to match the number of classes in GTSRB dataset:

- **SET** `num_classes` = number of unique classes in training dataset
- **REPLACE** last layer with a new linear layer:
    - `in_features` = original layer input features
    - `out_features` = `num_classes`

MOVE model to GPU if available, otherwise use CPU

Algorithm Pseudocode

Model Training

FUNCTION train_model with parameters: `model, train_loader, criterion, optimizer, num_epochs` = 10

SET model to training mode

FOR each `epoch` from 1 to `num_epochs`:

- **SET** `running_loss` = 0.0
- **INITIALIZE** lists `all_labels` and `all_preds`
- **FOR** each batch in `train_loader`:
  - **MOVE** `inputs` and `labels` to device (GPU/CPU)
  - **ZERO** the gradients of the optimizer
  - **FORWARD PASS**:
    - **COMPUTE** `outputs` from `model(inputs)`
    - **COMPUTE** `loss` using `criterion(outputs, labels)`
  - **BACKWARD PASS**:
    - **CALCULATE** gradients via `loss.backward()`
    - **UPDATE** model parameters via `optimizer.step()`
  - **DETERMINE** predictions:
    - **EXTRACT** predicted labels from `outputs`
    - **APPEND** true `labels` to `all_labels`
    - **APPEND** `predicted` labels to `all_preds`
  - **ACCUMULATE** `running_loss` with `loss.item()`
- **COMPUTE** average `epoch_loss` = `running_loss` / length of `train_loader`
- **PRINT** `epoch` and `epoch_loss`

- **PRINT** final classification report using `all_labels` and `all_preds`

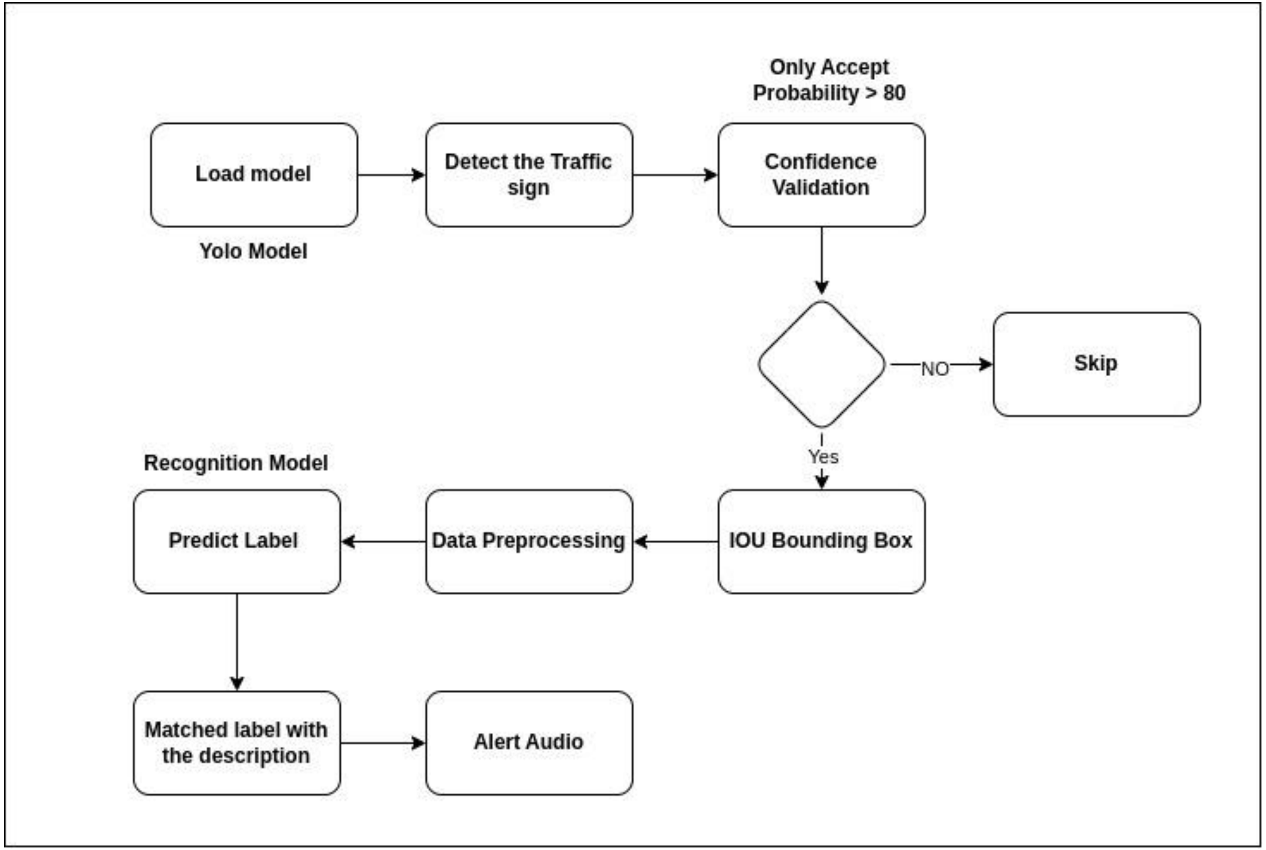
Algorithm Pseudocode

Model Evaluation

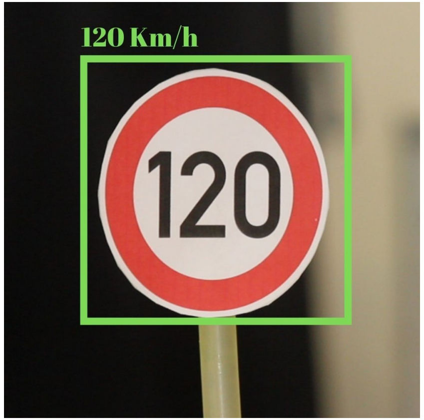
FUNCTION evaluate_model with parameters: `model, data_loader`

- **SET** model to evaluation mode
- **INITIALIZE** lists `all_labels` and `all_preds`
- **WITH** no gradient calculation:
  - **FOR** each batch in `data_loader`:
    - **MOVE** `inputs` and `labels` to device
    - **COMPUTE** `outputs` from `model(inputs)`
    - **EXTRACT** predicted labels
    - **APPEND** true `labels` to `all_labels`
    - **APPEND** `predicted` labels to `all_preds`
  - **CALCULATE** `accuracy` as the percentage of correctly predicted labels
  - **PRINT** `accuracy`

Implementation



Algorithm Pseudocode




```
DEFINE CLASS TrafficSignRecognitionModel
    INITIALIZE with model_path
    LOAD model from model_path
    SET model to evaluation mode

    FUNCTION predict(image_tensor)
        DISABLE gradient computation
        FORWARD pass image_tensor through model
        EXTRACT predicted label from model output
        RETURN predicted label
    END FUNCTION
END CLASS
```

1

```
DEFINE CLASS DBBox
    INITIALIZE with xmin, ymin, xmax, ymax, label
    SET xmin, ymin, xmax, ymax, label as attributes
END CLASS
```

2

```
DEFINE CLASS TrafficSignDetector
    FUNCTION __init__(model_path,
target_class_id=11, iou_threshold=0.5)
        LOAD YOLO model from model_path
        SET target_class_id to 11 (default)
        SET iou_threshold to 0.5 (default)
        INITIALIZE empty list to store saved bounding
boxes
    END FUNCTION
END CLASS
```

3

```
FUNCTION InferenceImage with parameter : image
    CONVERT image to a temporary path
    SAVE image to temporary path

    CALL TrafficSignDetector from Yolov10
    RETURN Detected result DBBox(image)
END FUNCTION
```

4

Algorithm Pseudocode

Data Preprocessing

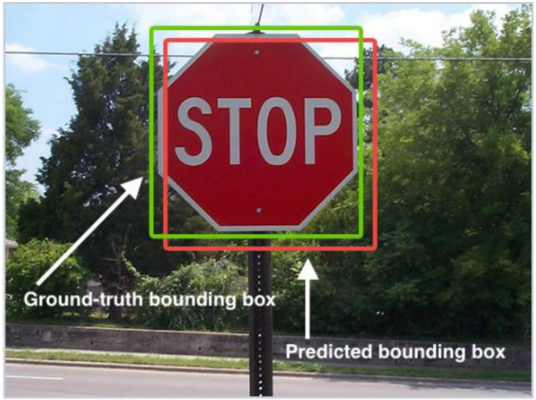
```
DEFINE CLASS Data_Preprocessing
    INITIALIZE with input size (default 224x224)
    SETUP transformation pipeline with resize, tensor conversion, and normalization

    FUNCTION preprocess(frame, bbox)
        EXTRACT xmin, ymin, xmax, ymax from bbox
        CROP image from frame using bounding box
        CONVERT cropped image to RGB format
        APPLY transformations to image
        RETURN transformed image as tensor with batch dimension
    END FUNCTION
END CLASS

FUNCTION save_preprocessed_image(frame, bbox, output_path)
    INITIALIZE Preprocessor
    CALL preprocess on preprocessor with frame and bbox to get tensor
    SAVE tensor to output_path
END FUNCTION
```

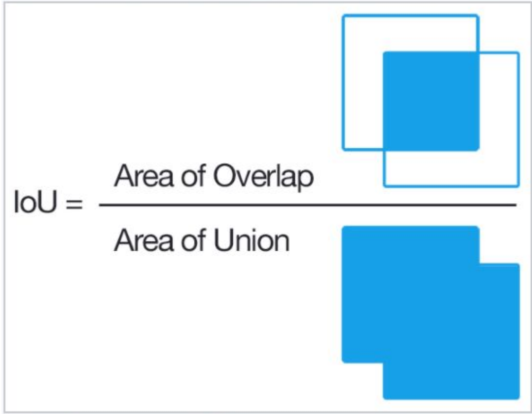
Algorithm Pseudocode

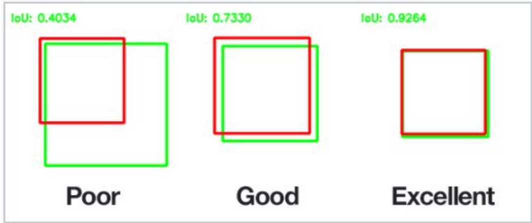
Intersection Over Union(IOU) Computation

In video traffic sign detection, IoU helps make sure the model doesn’t detect the same sign in multiple frames. If the sign is already detected in one frame and a new detection in the next frame overlaps too much, IoU helps ignore the repeated detection.

```
FUNCTION compute_iou(bbox1, bbox2)
    EXTRACT coordinates x1, y1, x2, y2 from bbox1
    EXTRACT coordinates x1_2, y1_2, x2_2, y2_2 from bbox2

    CALCULATE intersection coordinates xA, yA, xB, yB
    CALCULATE intersection area (interArea)

    CALCULATE area of bbox1 (boxAArea)
    CALCULATE area of bbox2 (boxBArea)

    CALCULATE IoU as interArea divided by sum of areas minus interArea
    RETURN IoU
END FUNCTION
```







Algorithm Pseudocode

Processing Input Image and Predicting label

Image from internet

Preprocessed Image 224x224

```
tensor([[ 43.0000,  31.0000,  53.0000],
        [  3.0000,   7.0000,  -6.2000],
        [ -4.2000,  -4.3000, -92.0000]])
```

Tensor Value

```
FUNCTION PROCESS_IMAGE_FROM_URL with parameters: image_url, output_dir, recognition_model_path

    INITIALIZE Traffic Sign Recognition Model
    INITIALIZE Preprocessor

    DOWNLOAD image from image_url
    CONVERT downloaded image to a frame

    PERFORM inference on the frame to obtain detection results
    CREATE an empty list called output to store bounding boxes

    IF the result contains predictions THEN
        OPEN a file in the output directory to write the results

        FOR each prediction in the result's predictions DO
            EXTRACT xmin, ymin, xmax, ymax coordinates from the prediction
            EXTRACT confidence from the prediction

            IF confidence is greater than or equal to the threshold THEN
                CROP the detected traffic sign from the frame
                SAVE the cropped image to the output directory

                INITIALIZE a bounding box using the extracted coordinates
                PREPROCESS the frame and bounding box to obtain a tensor
                SAVE the tensor to the output directory

                PREDICT the label using the preprocessed tensor
                WRITE the detected bounding box and predicted label to the results file
                APPEND the bounding box and label to the output list
            ELSE
                CONTINUE to the next prediction
            END IF
        END FOR
    ELSE
        RETURN "No traffic signs detected."
    END IF

    RENDER the image with bounding boxes using the frame, output, and output_dir

END FUNCTION
```

Algorithm Pseudocode

From predicted label to Audio, Audio Preparation

Classes: 43 classes
Features: Classid, Label, Description

ClassId	Label	Description
0	Speed limit (20km/h)	Indicates a maximum speed of 20 kilometers per...
1	Speed limit (30km/h)	Indicates a maximum speed of 30 kilometers per...
2	Speed limit (50km/h)	Indicates a maximum speed of 50 kilometers per...
3	Speed limit (60km/h)	Indicates a maximum speed of 60 kilometers per...
4	Speed limit (70km/h)	Indicates a maximum speed of 70 kilometers per...
5	Speed limit (80km/h)	Indicates a maximum speed of 80 kilometers per...
6	End of speed limit (80km/h)	Indicates the end of the 80 kilometers per hou...
7	Speed limit (100km/h)	Indicates a maximum speed of 100 kilometers pe...
8	Speed limit (120km/h)	Indicates a maximum speed of 120 kilometers pe...
9	No overtaking	Prohibits overtaking other vehicles.

Initialize and setup:

1. SET `csv_file` to the name of the CSV file containing labels ("Labels.csv").
2. READ the CSV file into a DataFrame called `data`.
3. SET `output_dir` to the directory where MP3 files will be saved ("descriptions_mp3").
4. CREATE the output directory if it does not already exist.

Process each row in the CSV:

5. FOR each row in the DataFrame: - EXTRACT `class_id` from the 'ClassId' column. - EXTRACT `description` from the 'Description' column.
6.
7. - CREATE a text-to-speech object `tts` using `description` and English language settings.
8. - SET `mp3_filename` to the path combining `output_dir` and `class_id` with ".mp3" extension.
9. - SAVE the speech as an MP3 file using `tts.save()` with `mp3_filename`.

END

```
(base) daro@daro-Aspire-A515-56:~/Documents/Computer-vision/Traffic-sign-recognition-application/app/descriptions_mp3$ ls
0.mp3   11.mp3  13.mp3  15.mp3  17.mp3  19.mp3  20.mp3  22.mp3  24.mp3  26.mp3  28.mp3  2.mp3   31.mp3  33.mp3  35.mp3  37.mp3  39.mp3  40.mp3  42.mp3  5.mp3  7.mp3  9.mp3
10.mp3  12.mp3  14.mp3  16.mp3  18.mp3  1.mp3   21.mp3  23.mp3  25.mp3  27.mp3  29.mp3  30.mp3  32.mp3  34.mp3  36.mp3  38.mp3  3.mp3   41.mp3  4.mp3   6.mp3  8.mp3
```

Algorithm Pseudocode

From predicted label to Audio (Conts.)

Matched Label With the Audio File:

1. DEFINE function `play_audio_for_label` with `predicted_label` as the input parameter.

Check the predicted label:

2. IF `predicted_label` is equal to 2 THEN - SET `mp3_filename` to "descriptions_mp3/2.mp3". - CALL `os.system` to execute the command to play `mp3_filename`. 3. END IF

End of function: 4. END function `play_audio_for_label`.

```
(base) darc@daro-Aspire-A515-56:~/Documents/Computer-vision/Traffic-sign-recognition-application/app/descriptions_mp3$ ls
0.mp3   11.mp3  13.mp3  15.mp3  17.mp3  19.mp3  20.mp3  22.mp3  24.mp3  26.mp3  28.mp3  2.mp3   31.mp3  33.mp3  35.mp3  37.mp3  39.mp3  40.mp3  42.mp3  5.mp3  7.mp3  9.mp3
10.mp3  12.mp3  14.mp3  16.mp3  18.mp3  1.mp3   21.mp3  23.mp3  25.mp3  27.mp3  29.mp3  30.mp3  32.mp3  34.mp3  36.mp3  38.mp3  3.mp3   41.mp3  4.mp3   6.mp3  8.mp3
```

Technology and Tools

python™

Flask
web development,
one drop at a time

ultralytics

Google
Text-to-Speech

OpenCV

Result

Result

Model	Accuracy	Epoch	Learning Rate	Total Params	Training Time On GPU T100
Mobilenet-v3-small	98%	10	0.001	1,561,931	5.10 Minutes

MobileNetV3-Small: Key Information

- **Developed by:** Google Research
- **Architecture:** Optimized for mobile and edge devices, balancing speed and accuracy.
- **Pre-trained Dataset:** Typically trained on ImageNet for classification tasks.
- **Input Size:** Best input size is 224x224 pixels.
- **Activation Function:** hard swish activation .
- **Attention Mechanism:** Integrates Squeeze-and-Excitation (SE) blocks for better feature representation.
- **Usage:** Ideal for resource-constrained environments like mobile apps and embedded devices.

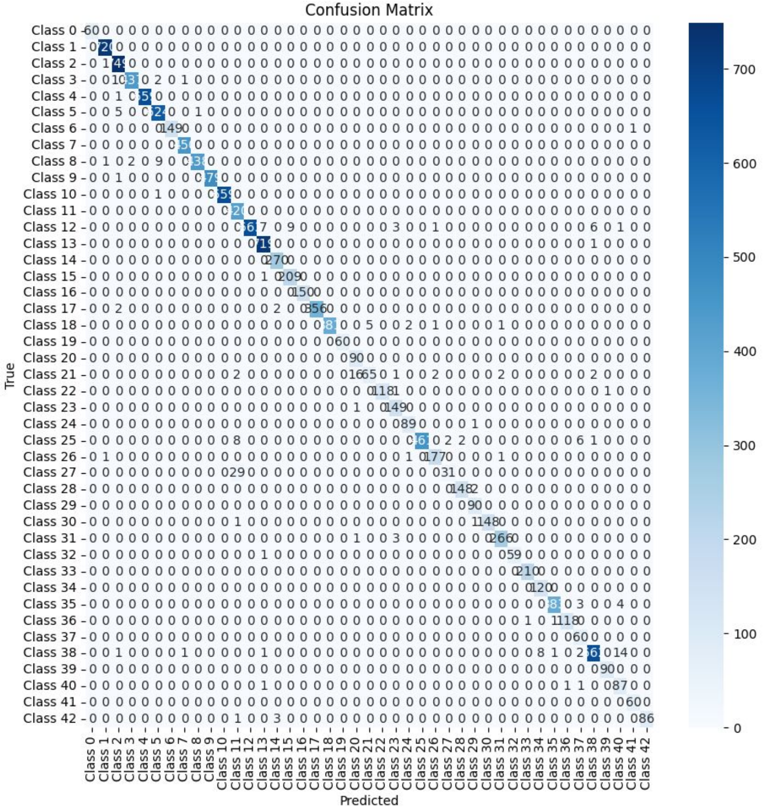
Paper: https://arxiv.org/pdf/1905.02244

Result

Confusion Matrix: A table that compares actual and predicted labels in a classification task.

Why we use it for classification problem:

- **Identifies Misclassifications:** Shows where the model is making errors.
- **Class Performance:** Evaluates accuracy for each class individually.
- **Model Improvement:** Helps in diagnosing and improving model performance.



Conclusion

Conclusion

Successfully developed a traffic sign recognition system utilizing AI and computer vision techniques.

Challenges Encountered:

- **Data Quality**: Data quality, which required careful preprocessing and augmentation.
- **Knowledge Gaps**: Spent significant time learning and testing to bridge knowledge gaps in model optimization and real-time processing.
- **Real-Time Implementation**: Particularly issues related to varying brightness and environmental conditions.

THANKS

Do you have any questions?