



**MARMARA UNIVERSITY**

**FACULTY OF ENGINEERING**

**CSE4288**

**Introduction to Machine Learning**

**TERM PROJECT**

**Data Preprocessing and EDA**

**Group: 2**

## Table of Contents

1. Dataset .....	3
2. Data Cleaning and Preprocessing for YOLO .....	3
3. Data Preprocessing for Other Models, Image Embedding .....	4
4. Exploratory Data Analysis (EDA) .....	5
5. Feature Identification and Engineering .....	5
6. References .....	6

## 1. Dataset

For this project, we are using BDD100K dataset, which is a public dataset including images and videos taken from car cameras. This multimedia consists of daily traffic through the city: vehicles driving, pedestrians walking etc.

We are responsible for detecting the crosswalks in images. Therefore, only images are retrieved as the raw dataset. Currently, we have 70000 images in our raw dataset.

## 2. Data Cleaning and Preprocessing for YOLO

The dataset needs to be cleaned to begin preprocessing. In Section 1, it is mentioned that images consist of various daily vehicle and pedestrian traffic. However, we are only responsible with the images that includes crosswalk labelling for detecting the crosswalks. Moreover, among these images, there are images that are not labelled at all. Firstly, we cleaned images without labels. Then, we cleaned the images that do not include crosswalks.

After successfully cleaning the raw dataset, the second step is data preprocessing, where we need to process data in a way that can be the input for YOLOv8 training.

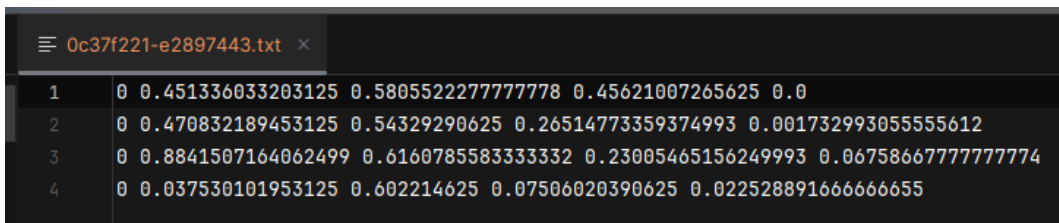
The crosswalk label includes the vertices of the 2d polygon, indicating the location of the object inside the image. In Figure 1, an example of crosswalk labelling is shown.

```
"category": "crosswalk",  
"poly2d": [  
  {  
    "vertices": [  
      [  
        490.49528,  
        419.568264  
      ],  
      [  
        576.406877,  
        416.571348  
      ]  
    ],  
    "types": "LL",  
    "closed": false  
  }  
]
```

**Figure 1:** Crosswalk labelling of an image

The labels are given as 2D polygons using vertices. However, YOLO's bounding boxes method is compatible with rectangular formed coordination. First thing to do is to convert these polygon vertices into rectangular form. This rectangle is defined by rightmost, leftmost, topmost and the bottommost points of the polygon.

After the conversion, the information about the image needed to be written in YOLO format, including the class ID, width and height of the crosswalk. In this project, the class ID of crosswalk is set as 0. Additionally, after calculating the width and height from the rectangular coordinates, these values should be normalized and recorded in txt file for each image. An example for YOLO format txt file is given in Figure 2.



1	0	0.451336033203125	0.5805522277777778	0.45621007265625	0.0
2	0	0.470832189453125	0.54329290625	0.26514773359374993	0.001732993055555612
3	0	0.8841507164062499	0.6160785583333332	0.23005465156249993	0.06758667777777774
4	0	0.037530101953125	0.602214625	0.07506020390625	0.022528891666666655

**Figure 2:** YOLO format txt file of training image

### 3. Data Preprocessing for Other Models, Image Embedding

The image embedding process began with dataset preprocessing, where the dataset was divided into training and validation sets, 80% of the images split for training and 20% for validation. The separation was made by saving the image names into two separate text files as train\_images.txt and val\_images.txt

For feature extraction, a pre-trained ResNet50 model from PyTorch was used. The model's fully connected layer was removed to use it as a feature extractor. Images were pre-processed by resizing them to 224x224 pixels and applying normalization based on ImageNet standards. The pre-processed images were then passed through the feature extractor, and the resulting feature vectors were flattened and saved as .npy files for both training and validation sets.

This process successfully converted images into feature vectors, enabling their use in machine learning tasks. With using a pre-trained ResNet50 model, we achieved efficient and reliable feature extraction. Missing images and processing errors were displayed during execution, providing clarity for troubleshooting.

## 4. Exploratory Data Analysis (EDA)

**Table 1:** Analysis of the raw dataset

<b>Total number of images in dataset</b>	70000
<b>Total number of images with crosswalk</b>	25637
<b>Average crosswalk number per image</b>	1
<b>Maximum crosswalk number per image</b>	5
<b>Minimum crosswalk number per image</b>	1
<b>Image Size</b>	1280x720

The dataset for this project includes 70,000 images, of which 25,637 contain crosswalk labels with an average of 1 crosswalk per image. Some images have up to 5 crosswalks, while others have just 1, highlighting the need for careful preprocessing and robust model design. Images with multiple crosswalks provide valuable training examples, but the variability in crosswalk density poses challenges for accurate detection in complex scenes. All images have a fixed size of 1280x720 pixels.

## 5. Feature Identification and Engineering

As mentioned in Section 1, raw dataset consists of various labels. In Figure 3, labels are described.

### Lane Categories

```
0: crosswalk
1: double other
2: double white
3: double yellow
4: road curb
5: single other
6: single white
7: single yellow
8: background
```

**Figure 3:** Lane Marking

The raw dataset includes labels for multiple categories, such as lanes, road curbs, and other markings. Since the objective of this project is to detect crosswalks, all other annotations were excluded. This filtering process ensured that only the "crosswalk" category (class ID 0) was included in the final dataset. This reduces noise and focuses the model on learning features specific to crosswalk detection.

## Data Preprocessing and EDA

As the dataset contains only one class, there is no direct issue with class imbalance. However, ensuring diverse examples of crosswalks such as different lighting conditions, angles, and environments was prioritized inside the dataset. This diversity helps the model generalize better during inference, particularly for unseen scenarios.

The dataset is divided into training set and validation set for both training and the evaluation of the model. 80% of the dataset is used for training and 20% of the dataset is used for validation. The images are shuffled to make sure that the dataset is split randomly.

About the cleaning and preprocessing of the dataset for training the YOLO model, images without crosswalk annotations were removed, as they do not contribute to the training process. Additionally, all bounding box coordinates were normalized to the image dimensions during preprocessing. This normalization step ensures that the model can handle any type of image without relying on the size of the image. Although we have a fixed size, this step was necessary for adaptation to the base YOLO model.

These steps ensure that the dataset is clean, relevant, and compatible with our model requirements and ready for development.

## 6. References

[1] Fisher Yu. 2024. “BDD100K Documentation”. Date Accessed: 25.11.2024.

(<https://doc.bdd100k.com/index.html>)

[2] Ultralytics, 2024. “Ultralytics YOLO Documentation”. Date Accessed: 25.11.2024.

(<https://docs.ultralytics.com/tr>)