**Autonomous Vehicles using Computer Vision, Machine Learning and Deep Learning**

**By: Ali Khodr Ali - . James Suresh - 250782695, Jasleen Bedi - and Rosario Sridish - 251369817**

Abstract— The project uses computer vision and machine learning methodologies to employ YOLOv8 for its real-time object detection, coupled with deep Convolutional Neural Networks (CNNs), Attentive CNN frameworks for enhanced visual attention, and the profound feature discernment of the VGG16 network, the system achieves remarkable precision. Further refined by Support Vector Machines (SVMs) for definitive classification and leveraging OpenCV for sophisticated image preprocessing, suite empowers self-driving cars with precise detection and interpretation of road signs, lanes, other vehicles, and pedestrians, ensuring secure road traversal. Keywords—Deep Learning, Computer Vision, YOLO, Transformers, R-CNN, SVM, Attentive CNN

# **Introduction**

Autonomous or self-driving systems process and interpret real-time data by integrating OpenCV with sophisticated deep learning architectures, including Convolutional Neural Networks (CNNs) and Attentive CNNs. By employing transfer learning, we enhance our models by incorporating pre-trained networks such as VGG16, which allows us to tailor them for efficient navigation in diverse vehicular environments. In addition, we incorporate advanced object detection frameworks, including YOLOv8 and support vector machines (SVMs), to enhance the precision of decision-making under dynamic road conditions.   
  
The objective of the project is to implement detection for road signs, lanes, vehicles, and pedestrians. A variety of methodologies are used in the training and implementation of the models, including YOLOv8, OpenCV, Support Vector Machines (SVMs), convolution neural networks, and Attentive CNNs. Our road sign detection system consists of CNN and attentive CNN. SVM and YOLO version 8 are utilized to detect both vehicles and pedestrians. Canny Edge Detection and other OpenCV libraries are utilized to implement lane detection.

* Detecting the whereabouts of other vehicles on the road is critical for maintaining safe distances and preventing collisions.
* Lane detection is a critical function that ensures the vehicle remains on its intended course and averts unintended deviations from lanes by precisely monitoring lane markings.
* Road Sign Detection: Maintaining adherence to traffic regulations and safeguarding passengers necessitates the ability to identify traffic signs such as stop signs, speed limits, and pedestrian crossings.

# **Datasets Used**

**KITTI DATASET**

The KITTI dataset will serve as the principal dataset utilized for object detection training [1]. There are 21 training sequences and 29 testing sequences in the Kitti Dataset. Although it comprises eight divisions, the evaluation focuses primarily on the car and pedestrian categories. Automobiles, vehicles, vans, motorcycles, pedestrians, cyclists, and motorists, as well as miscellaneous objects (those not falling into the categories), will be included in the classes.   
  
 We combined more images from other sources with this dataset to make an aggregated dataset of more than 142,000 images. This process took more effort than the effort taken to make the models themselves.

Many algorithms are used to retrieve and transform data. Annotation cropper classifies and crops images into distinct folders after analyzing KITTI's annotations. The data subsequently finds application in classification models. Annotations compatible with the YOLO module can be converted from KITTI's annotations using the annotation converter included in the other script.

**GTSRB DATASET**

Particularly for the task of traffic sign recognition, the German Traffic Sign Recognition Benchmark (GTSRB) dataset is a collection of images of traffic signs that is extensively utilized in machine learning and computer vision research. Its primary purpose was to facilitate the evaluation and development of algorithms that could distinguish traffic signs from images captured in the real world.   
  
Comprising more than 50,000 images spanning 43 categories, this compilation offers a wide-ranging assortment of traffic signs situated in diverse conditions, including varying illumination and occlusions, to closely resemble real-life situations. Serving as a standard, GTSRB significantly contributes to the improvement of traffic sign recognition algorithms' precision and robustness by resolving obstacles such as sign occlusions and variable environmental conditions.

**Own videos for testing**

We mounted a camera on a vehicle and filmed London, Ontario in the real world to evaluate the models we developed and ensure that they functioned in real-world situations. We gathered a variety of vehicle and pedestrian footage for implementation into our model.

# **Preprocessing**

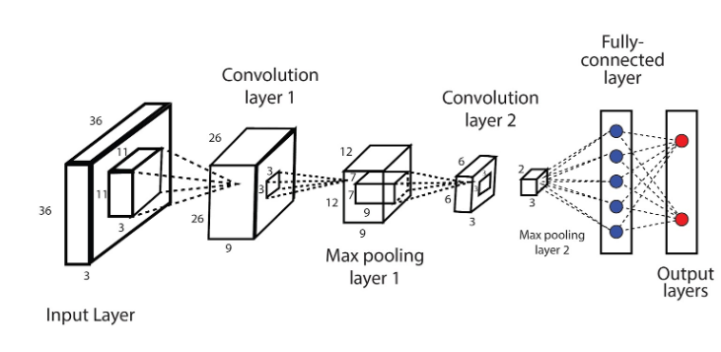
Images are recorded as pixel arrays, where each pixel has a value ranging from 0 to 255, representing the intensity of light. Higher values correspond to a closer proximity to white. By utilizing OpenCV, we transform colored images, which consist of three channels, into grayscale format to facilitate more straightforward processing. To detect edges, we utilize masks to accentuate white regions, thereby isolating road lanes by reducing non-white colors to zero in all channels.   
  
By utilizing OpenCV, we transform photos with color into grayscale, which enhances the efficiency of the processing. In order to detect edges, masks are utilized to segregate regions of high intensity values, thereby extracting the white color from the images. This method sets the values of colors that are not white to zero in all three-color channels, enabling us to concentrate exclusively on the white lanes in road photographs.   
  
Convolutions are employed to identify characteristics and edit images by utilizing a kernel matrix to implement filters that extract significant features while maintaining the spatial pixel connections. Feature detectors are used to analyze grayscale images, resulting in the creation of feature maps. These maps can be further improved by applying sharpening or blurring kernels, as necessary.   


Image processing techniques, such as Gaussian blur, box blur, and unsharp masking, are used to improve the quality of our images. The applied techniques encompass the use of Gaussian and box blurs to diminish noise and enhance the smoothness of textures. Additionally, unsharp masking is utilized to emphasize significant edges, such as those observed on road signs, lane markers, and barriers. OpenCV enables sophisticated technological features such as Canny edge detection for precise identification of lane boundaries and Haar cascades for efficient recognition of pedestrians and vehicles. OpenCV uses models to precisely classify and monitor things in real-time, effortlessly integrating with LIDAR and cameras to assess complex terrains. Normalized blurring kernels, such as Gaussian or box blur, reduce image noise and minimize intricate details, hence simplifying our data for input into following networks.   
  
 **Edge Detection and Gradient Calculation:**

Image is smoothened using a Gaussian filter with a specified spread. In this smoothed image, we calculate the gradient magnitude and direction for each pixel. [6] We then apply non-maximum suppression or thinning to eliminate pixels that are not the maximum along the gradient's direction. To retain strong edges and suppress noise, we perform hysteresis thresholding on the gradient values.

Sobel and Canny Edge detection is applied for edge to detection, finding high gradient areas and noise suppression. Sobel focuses on significant changes in pixel intensity. To refocus interest on a specific area, we do a region of interest masking. It focuses on crucial areas like road lanes and traffic signs, enhancing accuracy. For feature extraction, HOG (Histogram of oriented gradients) which captures distribution of gradient orientations in image. For image denoising histogram equalization is done redistributing pixel intensities. ROI extraction is done then, we apply bounding box extraction and ROI cropping. We detected straight lines, road markings and curves using Hough transform.

# **Methodology**

For this project, three important computer vision related tasks were studied:

**Lane Detection:**

The Lane detection model utilizes OpenCV's powerful computer vision capabilities to reliably detect and follow roadway lane markings in videos. The model effectively differentiates lanes from the surrounding environment by utilizing complex image processing techniques such as edge detection, color filters, and transformations.

**Traffic Sign Classification:**

**CNN:**

The convolutional neural network (CNN) is designed for picture classification, specifically focused on identifying traffic signals. The model begins with an input layer specifically tailored to accommodate the structure of the training images. Subsequently, the system employs convolutional layers with Rectified Linear Unit (ReLU) activation to extract distinctive characteristics from the images. This is followed by pooling layers to decrease their dimensionality. A flattening layer is used to reformat the data in a way that is suitable for completely connected dense layers. Dropout is used to mitigate overfitting during the training process. The script specifies 25 units with sigmoid activation. Next, the script establishes callbacks to store the model that performs the best during training. It also builds the model using a suitable loss function and optimizer. During the training process, it employs early stopping to avoid overfitting and trains the model in batches for a specified number of epochs, reserving a portion of the data for validation. The script logs the duration of the training process, displays a concise overview of the model, and assesses the model's precision using test data that has not been previously seen during the testing phase.

**Attentive CNN:**

This model begins with a structured input layer tailored to the shape of traffic sign images, followed by a series of convolutional layers, each equipped with 64 and 128 filters, respectively. [7] These layers are augmented with max pooling for spatial dimension reduction, batch normalization for training stability and efficiency, and dropout layers with a rate of 0.5 to mitigate overfitting. Unique to this architecture is the inclusion of a Multi-Head Attention layer after reshaping the convolved features, which enables the model to concurrently focus on various segments of the image, enhancing its ability to discern intricate patterns. A Global Average Pooling layer is used instead of a traditional flattening step to summarize detected features. The network further processes these through a dense layer with 256 units and a final output layer employing a SoftMax activation function, designed to distribute probabilities across the 43 distinct traffic sign classes. The compilation of this model uses the Adam optimizer and sparse categorical crossentropy, aiming for the best accuracy

**Vehicle and Object Detection and Classification:**

**SVM using Linear SVC:**

The Support Vector Machine (SVM) algorithm is a supervised learning method primarily used for classification and regression tasks. SVM is used to find the best hyperplane that maximizes the margin between different classes in the feature space. For linearly separable datasets, it finds a direct hyperplane, while for non-linearly separable data, it uses kernel functions to project the data into a higher-dimensional space where a linear separation becomes possible.

The algorithm uses the critical data points nearest to the hyperplane to define the decision boundary. SVM optimization aims to balance the maximization of the margin with the accurate classification of training data. Once trained, SVM classifies new instances based on their position relative to the hyperplane though its effectiveness is based on the choice of kernel type and regularization parameters to prevent overfitting or underfitting.

**YOLO:**

YOLO (You Only Look Once) is a CNN algorithm that processes an image in multiple steps using the neural network framework, YOLO frames object detection as a single regression problem, directly predicting bounding boxes and class probabilities from full images in one evaluation [4]. It uses a singleshot approach to detecting algorithms. It is a highly responsive, fast low latency approach to detecting objects. This approach enables YOLO ideal for applications that require fast and reliable object recognition. YOLO can be used for real-time detection of other vehicles, pedestrians, traffic signs, and various obstacles on the road. Its ability to process images swiftly and accurately ensures that autonomous vehicles can respond promptly to dynamic road conditions. For this project 2 versions of Yolo will be analyzed:- Yolov8-s(small) and Yolov8-n(nano).

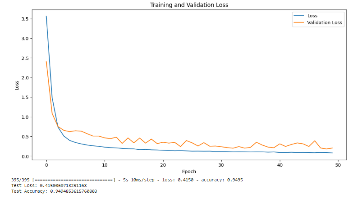
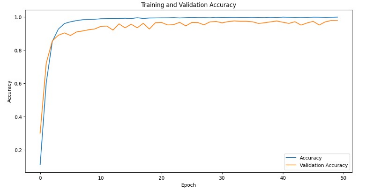
The nano model is less computationally intensive and has approximately 3.2 million parameters. Yolo small on the other hand features around 11.2 million parameters [5]. Larger models are not necessarily more robust, and in some cases, they may even perform worse in terms of robustness to real-world factors such as geometric changes, blur, noise, and lighting changes

# **Results**

**Traffic Sign Classification:**

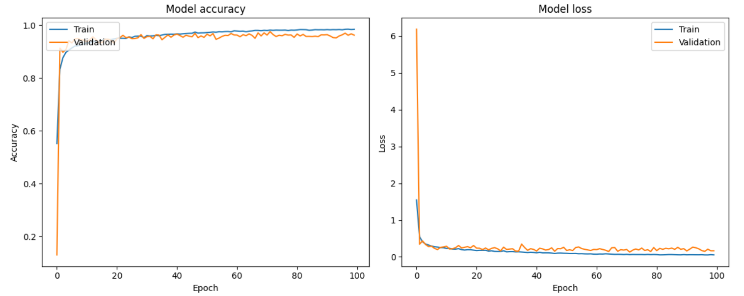
**VGG 16:**

VGG 16 classifies 43 types of traffic signs. Max-pooling, dropout, and convolutional layers are included to reduce overfitting and enhance generalization. After the model is trained over several epochs, it achieves a high-test accuracy of approximately 94.95%, along with approximately 91.56%, 92.20%, and 91.58% for precision, recall, and F1-score, respectively. By assisting in the identification of misclassifications, the confusion matrix supports focused enhancements. Ensuring dependable identification is crucial for the use of autonomous driving in this configuration.



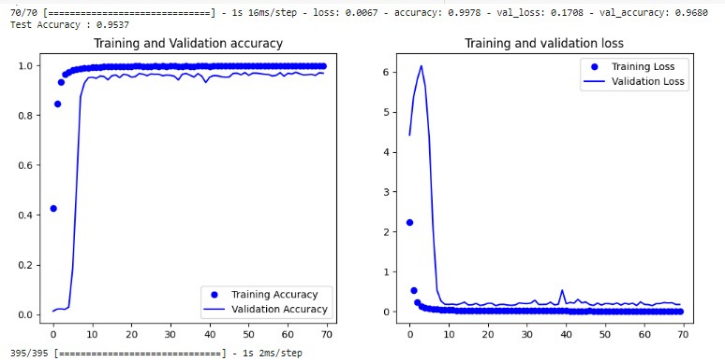
**Attentive CNN:**

The Attentive CNN achieved impressive accuracy after extensive training. Over 100 epochs, training accuracy reached a peak of 98% and remained stable, indicating effective learning and consistent performance. Validation accuracy closely mirrored training accuracy at around 96%, signifying the model's ability to perform well on unseen data. Both training and validation loss metrics displayed a rapid decrease followed by a plateau, suggesting the model's predictions stabilized early in training. The model's effectiveness is further confirmed by a low test loss (0.2304) and a high test accuracy (96.19%). These results demonstrate its robustness and capability to deliver accurate predictions on entirely new data. The Attentive CNN's performance on unseen data (test set) suggests it's well-suited for real-world applications. It can be either deployed immediately or fine-tuned for specific tasks.



**CNN:**

The CNN neural network accurately predicts traffic signs, correlating closely with actual labels, as evidenced by near-perfect accuracy in both training and validation sets and a test accuracy of 99.7%. There are two plots: one for accuracy and another for loss. The accuracy plot demonstrates rapid attainment of high precision, while the loss plot shows a sharp initial decline that stabilizes with ongoing training, indicating effective learning and application to new data.



This classification report includes precision, recall, f1-score, and support for each class, along with an overall model accuracy of 95.65%. Precision measures the percentage of correct positive identifications per class, while recall measures the accuracy of positive identifications. The F1-score is the harmonic mean of precision and recall, and support indicates the true instance count per class. The high values across these metrics demonstrate CNN's strong performance.

**Vehicle and Object Detection and Classification:**

**SVM using Linear SVC:**

For our experiment, we employed a Support Vector Classifier to distinguish between three main groups of road objects: automobiles, pedestrians, and traffic signs. The dataset consisted of 5,000 photos specifically selected for these categories for efficient model training and assessment.   
The Support Vector Classifier was trained after the images were modified, which is an effective method for addressing the challenging classification challenges caused by the diverse shapes and sizes of road items. The most effective hyperparameters, such as the cost parameter (C) and the kernel gamma, were identified using a thorough grid search with 5-fold cross-validation. The training set comprised 70% of the data, while the remaining 30% was allocated for model validation. The model exhibited a classification accuracy of 94% on the validation dataset. The precision for each category was remarkable, with automobiles achieving a 97% accuracy rate, traffic signs achieving a 95% accuracy rate, and pedestrians achieving a 90% accuracy rate. The variation in accuracy particular to each class emphasizes the difficulties encountered in pedestrian recognition, which are caused by the wide range of appearances and constantly changing shapes of pedestrians.

**YOLO**:

Both models were set to run for 100 epochs and had a patience value of 3. The training process was made to terminate early if the model fails to show any improvement after 3 epochs. The initial model (Yolov8n), which was smaller, took 39 epochs to finish training during a 0.55-hour duration. The Mean Average Precision achieved at 50% (MAP50) was approximately 95.26%, with a MAP50-95 of around 73.6% with a recall of 84.4%. The second model (Yolov8s), also known as the nano model, concluded its training after 65 epochs, which took around 1.025 hours. The final Mean Average Precision at 50 % (MAP50) obtained was around 93.5%, with a MAP50-95 of 83% with an overall recall 93.6%.

# **Conclusion**

When it comes to generalization and prediction accuracy, the CNN model performs better than the other two models under evaluation. With the lowest test loss of just 0.0087 and the best test accuracy of 99.73%, it can produce quite accurate predictions on fresh, untested data. Despite having the highest training accuracy (98.26%), Attentive CNN performs worse in test accuracy and loss, suggesting that it might not be as effective outside of the training context. Despite having a nearly flawless training accuracy of 99.99%, the VGG-16 exhibits indications of overfitting and falls short of its competitors in terms of test performance. Comparing attentive and VGG, the VGG16 model trained for 50 epochs yields slightly better test accuracy at approximately 95.8% compared to the attentive CNN's 96.2% accuracy after 100 epochs. However, the attentive CNN achieves a lower test loss of 0.2304, which suggests it might be generalizing slightly better than the VGG16 model. As a result, the CNN model is considered better due to its remarkable ability to combine learning and generalization.

Our unique Yolov8 small model yielded the most accurate results for detection. However, the system faced difficulties in identifying objects due to its primary concentration on detection. However, because it has a reasonably low latency in detecting objects, it should be considered as the main algorithm for object detection. All models showed high accuracy and precision in classification. However, because model maintenance is crucial, it is necessary to regularly update our models with fresh data in response to changing surroundings and other factors. It is important to consider the amount of time required for training while choosing the optimal model. Considering its comparatively low training time compared to other models, SVM will be chosen as the model for backup classification. The final system would incorporate a carefully calibrated Yolov8 tiny model for detection, combined with SVM and VGG classifiers.

**DATASET LINKS:**

KITTI Dataset

https://s3.eu-central-1.amazonaws.com/avg-kitti/data\_tracking\_image\_2.zip (Dataset)

https://s3.eu-central-1.amazonaws.com/avg-kitti/data\_tracking\_label\_2.zip (Training labels)

https://www.kaggle.com/datasets/stavanjoshi/road-signs-in-canada

https://www.kaggle.com/datasets/alincijov/self-driving-cars/code

Video dataset for testing pedestrians: https://www.kaggle.com/datasets/smeschke/pedestrian-dataset

GTSRB Road sign Dataset: https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign

Dataset for Canadian Road signs to test in the future: https://www.kaggle.com/datasets/stavanjoshi/road-signs-in-canada

**References**

1. <https://www.lakera.ai/blog/how-robust-are-pre-trained-object-detection-ml-models>
2. [The KITTI Vision Benchmark Suite (cvlibs.net)](https://www.cvlibs.net/datasets/kitti/index.php)
3. [YOLOv8 Based Object Detection for Self-driving Cars | IEEE Conference Publication | IEEE Xplore](https://ieeexplore.ieee.org/document/10441381)
4. [YOLO-Based Object Detection and Tracking for Autonomous Vehicles Using Edge Devices | SpringerLink](https://link.springer.com/chapter/10.1007/978-3-031-21065-5_25)
5. [YOLOv8 - Ultralytics YOLOv8 Docs](https://docs.ultralytics.com/models/yolov8/#performance-metrics)

[6] "Digital Image Processing," IEEE Transactions on Image Processing, vol. 15, no. 11, pp. 3421-3432, 2006

.