LAB 1

Sakshi Jain, Sai Kiran Baghavathula

San Jose State University

DATA 255 - Fall24

Dr. Simon Shim

Fall 2024

10/30/2024

1. **Differences between Object Detection, Image Classification and Image Segmentation**

Image Classification: An entire image may be viewed as a single unit, one that is accompanied by a single category, through the process of Image Classification. By simply looking at the image, the algorithm is expected to determine which class out of many it belongs to. If a model is presented with a photograph, for example, it would state that the photo is either a cat or a dog.

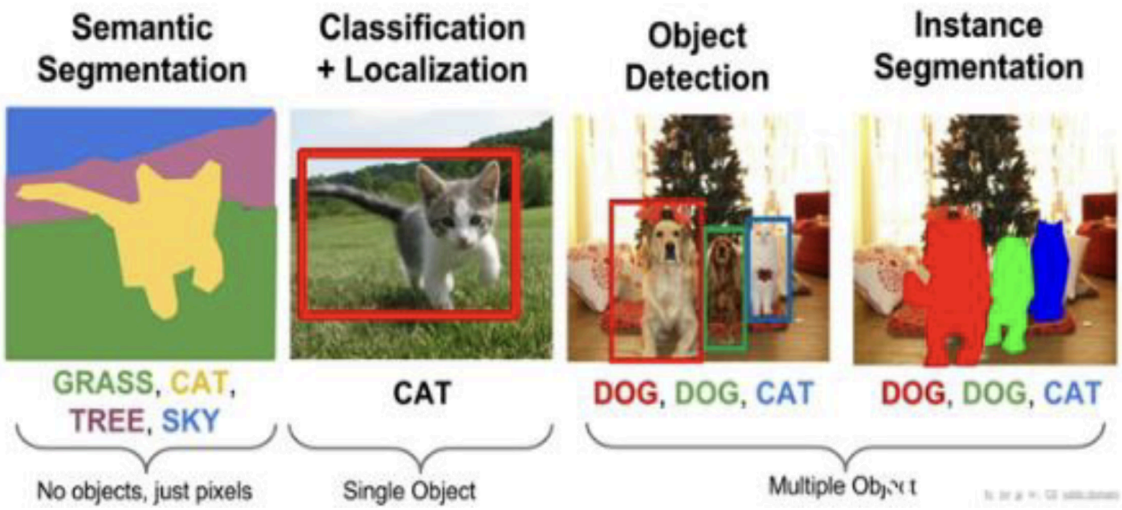
Object Detection: While the focus of image classification is to scan through an image to classify it, object detection focuses on classifying and localizing multiple images in a single image. It not only predicts what each class of detectors in the picture does but also draws a rectangular box on every single object. For example, in an urban area photograph, detection of all the boxes that are found would include detecting traffic lights, pedestrians as well as cars.

Image Segmentation: goes a step further by assigning a specific group of pixels that convey specific information to every pixel present in the scanned image. However, there are 2 major types:

1. ***Semantic Segmentation***: The Pixels in this scenario have been separated into several categories or classes but the classes are not determined based on individual objects which makes for a clearer view. When viewing the 'tree' class, for instance, all the pixels are likewise grouped under that term.
2. ***Instance Segmentatio***n: This approach involves even more detail as different classes (>1) are taken into account. In the scenario where one views a crowd, every individual will be tracked down and outfitted with a unique label.

Key Differences

1. Classification delivers a relatively coarse analysis (one label per image), detection deals mainly with spatial features within the image, and segmentation deals with the best granularity trait of classifying objects at the pixel level.
2. Output complexity escalates from a single class label produced by classification to detection which produces class labels and location entities, to segmentation which produces a class label for each pixel.
3. General identification tasks can easily make use of classification. Detection finds application in self-driving cars and cameras, segmentation is widely used in fields where the accurate location of cut is necessary like the case of X-ray images.



Reference:
https://sjsu.instructure.com/courses/1595177/files/folder/Lectures?preview=78149159

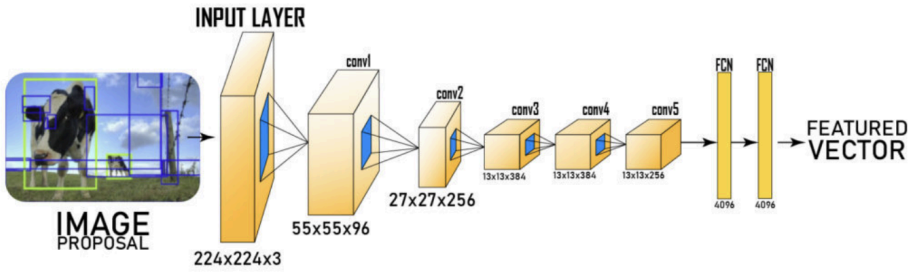
2. **Architectures and Differences of R-CNN, Fast R-CNN, and Faster R-CNN**

R-CNN (Regions with Convolutional Neural Networks):
***Architecture*:**

1. ***Region Proposal:*** Applies selective search to create approximately 2000 region proposals containing possible objects.
2. ***Region CNN***: Each Region proposal is altered to fit a certain size and these are fed individually through the CNN model for feature learning.
3. ***Classification:*** The learned features are sent for processing into different linear SVMs which are class-specific to identify the position of the object.
4. ***Bounding Box Regression***: The coordinates which define the boundaries of the objects are refined.

Drawbacks:

1. Costs a lot of resources as it requires a computation of CNN for each region.
2. Takes quite a long to prepare, or train, and even longer to conduct inferences.
3. There is a multi-stage training process which is more cumbersome.



Reference: https://www.geeksforgeeks.org/r-cnn-region-based-cnns/

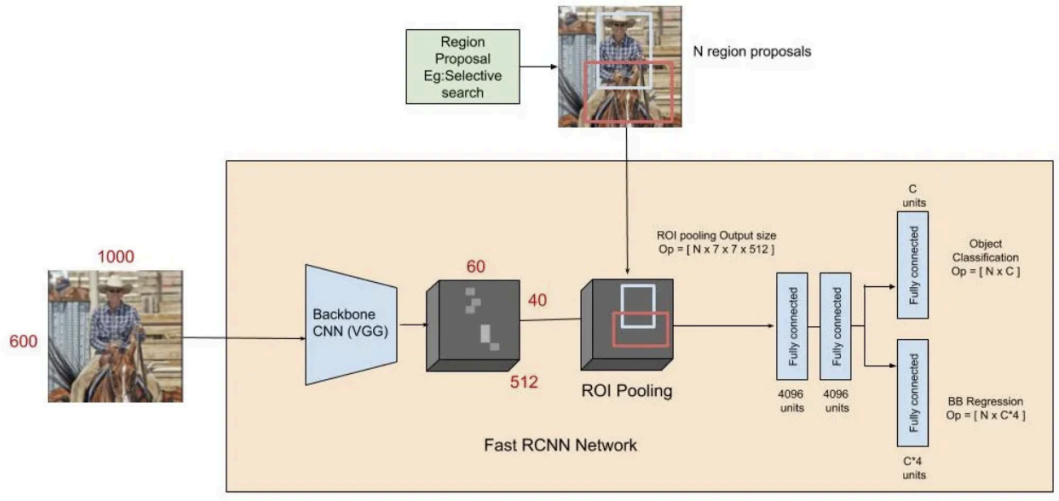
***Fast R-CNN*:**

Improvements:

1. ***Single Forward Pass***: It only takes one pass for the picture through a CNN and a convolutional feature map of it nearly stretches out all over the image.
2. ***Region of Interest (RoI) Pooling***: Area proposals are first aligned on the feature map, and a RoI pooling layer produces fixed-size feature vectors for every area within the proposal.
3. ***Unified Network***: The elements are passed through the fully connected layers where both classification and the bounding box regression take place.

Advantages:

1. Training and inference speed is significantly improved.
2. All processes are combined in one-step training.
3. Storage demand is reduced because features do not have to be stored for each area.



References:https://towardsdatascience.com/fast-r-cnn-for-object-detection-a-technical-summary-a0ff94faa022

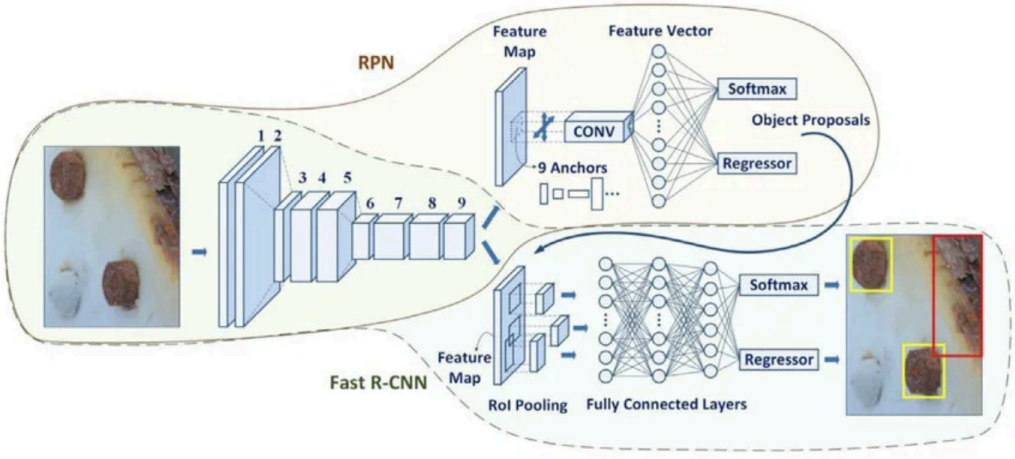
Faster R-CNN:

Innovations:

1. ***Region Proposal Network (RPN)***: This paper proposes a small network that gets convolutional layers from the detection network to propose possible areas.
2. ***Shared Computation***: The RPN and the detection network both share convolutional layers in the base.
3. ***Unified Framework***: Region proposal generation is merged with object detection whereby an integrated trainable network can be used.

Benefits:

1. Eliminates the use of external region proposal techniques such as selective search for region proposals.
2. Further improve speed and efficiency.
3. Because the entire structure of both the region proposals and detection networks are optimized the accuracy increases as well.



Reference: https://towardsdatascience.com/faster-rcnn-object-detection-f865e5ed7fc4

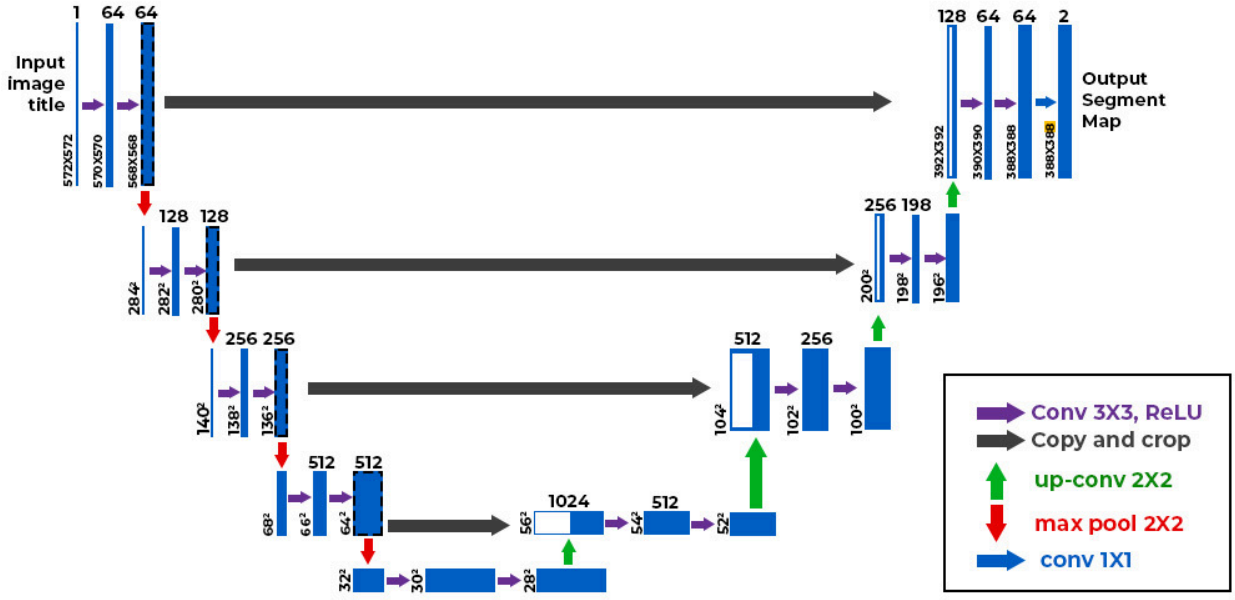
Differences:
R-CNN: It receives region proposals from outside the network and applies the CNN to every segment independently.
Fast R-CNN: All images are inputted only a single time, uses RoI pooling but does not eliminate the external region proposals.
Faster R-CNN: Synthesis of region proposal generation along with the detection network for effective and fast object detection.

3. **U-net**

U-Net is such a convolutional neural network architecture that is advantageous in image segmentation in medical images specifically with speed and accuracy.
Architecture Overview:

1. ***Symmetrical U-Shape***: Comprised of a contracting (downsampling) path and an expansive (upsampling) path making it appear like a U in shape.
2. ***Contracting Path (Encoder)***: The goal here is to capture context and then extract features from an image. This is done through the sequential application of two 3x3 convolutions (with ReLU activation) followed by downsampling using a 2x2 max pooling. To allow the hub to learn sophisticated features, the amount of feature channels increases by two during each stage of downsampling.

3. ***Expansive Path (Decoder)***: This facilitates accurate position at a feature level by upsampling features that were passed through the encoder path that contained high-level features. To achieve this objective, each step of upsampling starts with up-convolution of the feature map which drastically reduces the number of feature channels by half and ends with a 2x2 convolution. Those features were then connected using skip connections after being shrunk from the contracting path. This was proceeded by two more 3x3 convolutions (with ReLU activation).
4. ***Final Layer:*** This leads to the generation of a segmentation map through a 1x1 convolution, which projects each component feature vector to the number of classes to be output.



References: https://www.geeksforgeeks.org/u-net-architecture-explained/

Key Features:

1. ***Skip Connections:***
    a. Connections between the encoder and decoder pathways at corresponding levels.
    b. Maintain the spatial information that was lost after the downsampling.
    c. Make the network learn both the overall structure and the specifics.
2. ***Fully Convolutiona***l: No fully connected layers; accept images of varying dimensions.
3. ***Data Efficiency***:
    a. Optimized for scenarios with minimal training.
    b. Such employs a wide range of data manipulations to increase strength.

Applications:

1. Microscope images for cell segmentation purposes.

2. Outline highlighting of the tumors in medical imaging.
3. Satellite Pictures of the Earth.
4. Understanding various scenes in a self-driving car.

Advantages:

1. This makes it possible to perform segmentation with great accuracy even when training images are very few.
2. Thanks to its structure, training, and inference are speedier in comparison.
3. Can also be used for various segmentation problems that are not limited to biomedical imaging.

4. **Widely Used Metrics in the Object Detection Industry**

1. ***Mean Average Precision (mAP)***:
    a. ***Average Precision (AP):*** The measure is represented by the area under the precision-recall graph for one class. It takes into account both recall and precision at multiple levels of confidence.
    b. ***Mean Average Precision (mAP):*** The measure defines the average of all the measures classified as AP. The measure is an indication of the general effectiveness of the object detection system.
    c. ***IoU Thresholds:*** AP is frequently recorded at 0.5 and higher coverage levels or averaged out over multiple levels of coverage.
2. ***Intersection over Union (IoU):*** There is an overlap between the estimated bounding box and the true location in the bounding box. We can establish the formula for IoU as follows: IoU = (Area of Overlap) / (Area of Union). Predetermines whether, by a certain threshold concerning IoU, a predicted bounding box is accepted as true positive (e. g. Iou>0.5) hence setting a limit for a true positive box.
3. ***Precision and Recall:*** Simply states that the average precision measures the accuracy of the detector whereas recall is the ability to find all items of relevance.
    a. ***Precision***:
        i. Is the ratio of true positive detections and the number of total positive predictions made
        ii. $Precision = \frac{True\ Positive}{True\ positive + False\ Positive}$
        iii. High precision means a low rate of false positives in detection.
    b. ***Recall***:
        i. This is the ratio of true positive detections and the number of all positives that were to be detected.
        ii. $Recall = \frac{True\ Positive}{True\ positive + False\ Negative}$
        iii. High recall means a low rate of false negatives in the outcomes.

5. **Explanation of Non-Maximum Suppression**

Non-Maximum Suppression (NMS) NMS is an algorithm that deletes less confident bounding boxes surrounding the same object so that each instance of an object is targeted once in the prediction output. When a set number of bounding boxes surround the same object that are all blue or all red NMS will strike out the less certain ones and delete them out. By removing the boxes that were inaccurate or loosely defined NMS improves the detector's overall performance.

How It Works: The output of an object detection model in particular its bounding boxes and one or multiple confidence scores are defined for every single class. Bounded boxes with low-confidence scores (that is, a score that is less than or equal to the cutoff set by the user) are deleted since they will be low-confidence detections remaining bounding boxes are ranked according to their confidence scores in descending order.

1. ***Selection Process:***
    a. Set up an empty list that will contain the final bounding boxes.
    b. Loop the sorted list:
        i. Select the box with the maximum confidence score and put it into the final list.
        ii. Compute IoU: Compute the IoU for the selected box with all the maximum bounding boxes.
        iii. Suppression: IoUs that exceed the threshold that has been set (for example, 0.5) to an already selected box have to be deleted as it is likely that these are the same boxes.
    c. Until all boxes have been processed, repeat the step above.
2. ***Final Output:*** The remaining boxes after NMS have been processed are say one object each and hence contain the final detection of each object.

Key Considerations:

1. ***IoU Threshold:*** Tuning the IoU threshold is a critical parameter as it prevents too many boxes from being suppressed (and risks the chance that close objects are missed) and too many boxes from being retained (and causing duplication of detections).
2. ***Class-wise NMS***: To avoid the suppression of boxes across different classes, non-maximum suppression (NMS) is performed independently on each of the classes of the object.

Variants of NMS:

1. ***Soft-NMS:*** Soft-NMS decreases the confidence of boxes that overlap the bounding box rather than removing it completely; this is done based on the degree of IOU overlap. Assists in situations where the objects are placed near each other, allowing for a greater number of detections to be maintained.Reduces the number of detections in an attempt to facilitate further processing.Increased the accuracy of the detector by suppressing several

redundant detections of the identical object which could result in multiple false positives. Implemented in the majority of the current object detection systems i.e. Faster R-CNN, YOLO, SSD, and applied as a regular post-processing stage.

6. **Road Sign Dataset:**

The structure of the dataset is:

There are 4 files inside the folder "Road_Sign_Detection_4:

1. train:
    a. Images: There are multiple images inside which are used to train the model.
    b. Labels: labels of the images.
2. Valid:
    a. Images: multiple images are used to evaluate the performance of the model while training.
    b. Labels: labels of the images
3. Test: multiple images are used to evaluate the performance of the model after training the model.

7. **Data Transformation and Augmentation Steps**

Data Transformation:

Data transformation procedures come into play during plasticity as it is vital to preprocess the data to retain uniformity and enhance the training process.

The procedures that were carried out include:

1. ***Image Resizing:*** The images were modified to a universal size of 640*640 pixels. The reason is to ensure that all input received by the model is of the same size thereby making it easier to avoid complications caused by different sizes of data inputs and wasting time in the design of the neural network.
2. ***Box Adjustment***: As the images were being resized, the bounding boxes for the images were also enlarged to fit the new image size.
3. ***Normalization:*** Although it does not appear in the coding, there is normalization of pixel intensity to certain ranges like from '0' to '1' during the tensor conversion of an image which is suitable for neural network applications.

8. **Plot Random Images from the Datasets**

Through visualization of the datasets, one can appreciate the data characteristics and also check the integrity of the data preprocessing. The presentation of images together with the ground truths was made using a function display_images_with_bboxes.

Based on the information on the labels and their bounding boxes, this is what the function probably saw in each of the image sets:

Validation Images:

1. There are labels "Dangerous right turn", “Speed limit 100 kph”, “Speed limit 80 kph” and “Stop Sign”.
2. This function was able to identify the objects in the images and draw the boxes that are in the images for the object and all the labels are placed in the boxes to avoid confusion.

Test Images:

1. The test images show several instances of detection of a Speed limit of 20 Kph.
2. This demonstrates how bounding boxes of large size are consistently placed over every sign reading “20 Kph Speed Limit” across several pictures of the same sign.

Train Images:

1. There are also found labels for road signs depicted in this set, for example, “One-way road”, “Speed limit 20 kph”, “Speed limit 120 kph”, and “Crosswalk”.
2. Class labels are visible above bounding boxes that have been placed over these captured objects.

The code selected a limited number of images from every data loader (val_loader, test_loader, and train_loader), and presented them with the corresponding labels and bounding boxes. The traffic sign encircled by the box indicates that the model has supposedly mastered the technique of identifying different types of signs in different datasets and their corresponding areas.

9. **Model Implementation from Scratch**

The designing of the object detection is achieved using convolutional neural networks which were designed from scratch and are referred to as basic_object_detector.

Model Architecture:

The architecture consists of:

Convolutional Layer 1:

Input channels: 3 (RGB),
Output channels: 16,
Kernel size: 3,
Padding: 1

Convolutional Layer 2:

Input channels: 16,
Output channels: 32,
Kernel size: 3,
Padding: 1

Convolutional Layer 3:

input channels: 32,
output channels: 64,
kernel size: 3,
padding: 1.

Fully Connected Layer 1:

input features: 64*80*80,
output features of 512

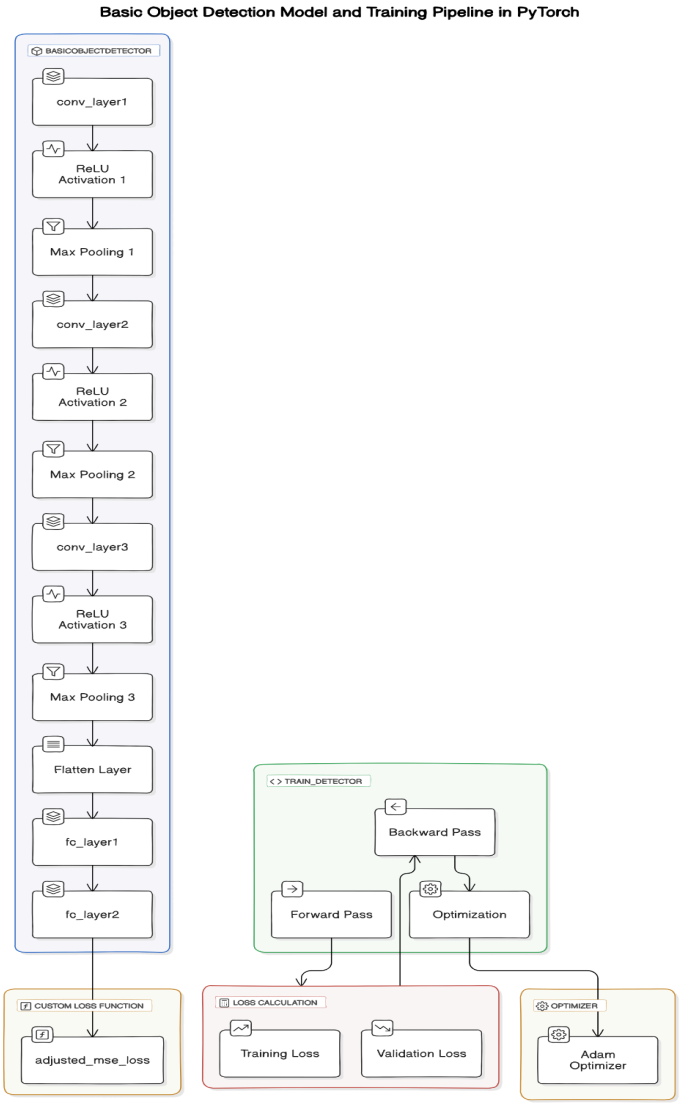
Fully Connected Layer 2:

input features = 512,
output features = 4

Activation Function: ReLU Activation is used after each convolution and fully connected layer.
Max Pooling: Used after each convolution layer with a kernel size of 2.
Loss Function: To accommodate batches in which some images may lack a bounding box, the Adjusted MSE loss function was employed.
Optimizer: Adam optimizer is implemented with a learning rate of 0.001.
Epochs: The model is trained for 50 epochs.

Observations:

Training Loss: Loss decreases progressively which shows the model is capable and can learn.
Validation Loss: Shows an initial decline, but after a while, it becomes very volatile. This could mean an indication of overfitting, upon reaching particular epochs.



10. **Compute IOU and Display Predicted Images**

IOU: A mean IoU of 0.587 is observed which implies that a mean of 58.7% of the predicted bounding boxes in a model's output overlap with the ground truth bounding boxes for the test set.

Accuracy: The model gave an accuracy of 0.787 which suggests that 78.7% of the predictions output has an IOU that is greater than or equal to the threshold level which is (0.25). This implies that in about 79% of the cases, the model predictions are valid according to established criteria.

Observations:

1. Predicted bounding boxes (Red Boxes) usually cover surface ground bounding boxes (Green Boxes). But sometimes this is not the case.

2. Based on the values of the IoU metrics and detection accuracy such a moderate performance was observed.
3. Perhaps some of the predictions were approximately in a very small window of the actual bounding box which could lead to a reduction of the IoU.

Below are the predicted images by the custom model

11, 12. **Use pre-trained models such as YOLOv8 for object detection and print the IOUExperiment with pre-trained models and show the IOU of the test data set. Show tables and graphs of how the results change**

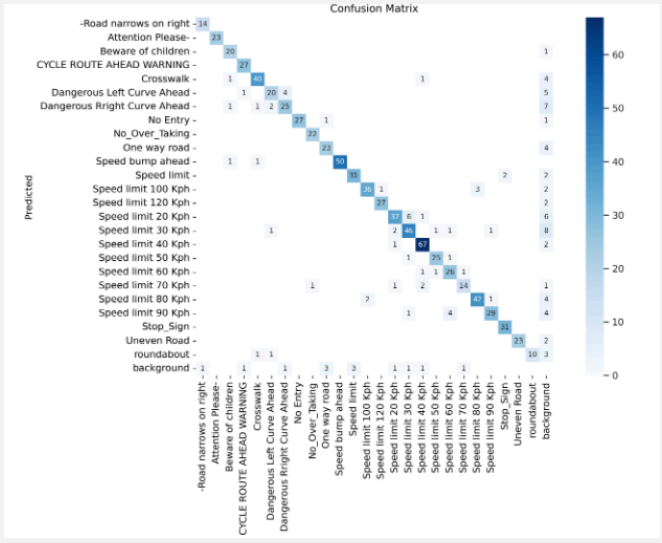
Experimented with different models of yolo. When taking the larger models because the size of the dataset is smaller the models are getting overfitted and not performing better on the unseen images. Achieved promising results using a smaller model called yolov8n. Compared to yolov10l, yolov8x the yolov8s model performed better but not as good as yolov8n (as per the kaggle competition). This report includes the top 2 models comparison and these two models are performed well in the experiments with different models.

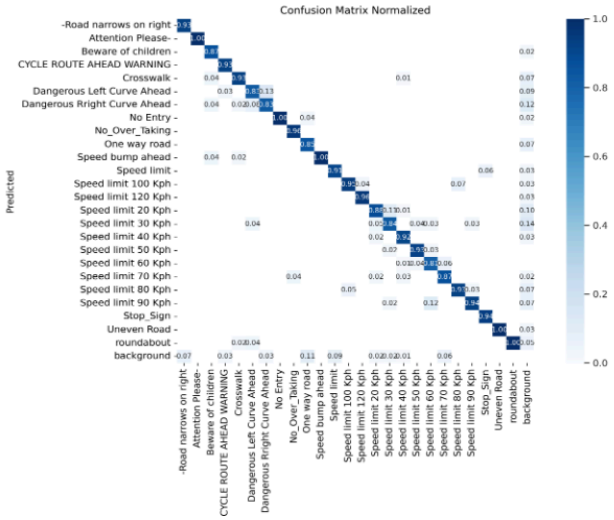
Below are the parameters used for yolo8n, which gave a good result.

model.train(data='/content/Road_Sign_Detection/data.yaml', epochs=30, batch = -1, optimizer= 'auto', amp=True)

1. Number of epochs was set to 30. That means the model will go through the entire dataset 30 times.
2. The batch size is given as -1, which means allowing the model to select the batch size based on memory and resources.
3. The optimizer is set to auto. It allows the model to choose the appropriate optimizer based on the library defaults.
4. To speed up the training process enable Automatic Mixed Precision (amp=True).

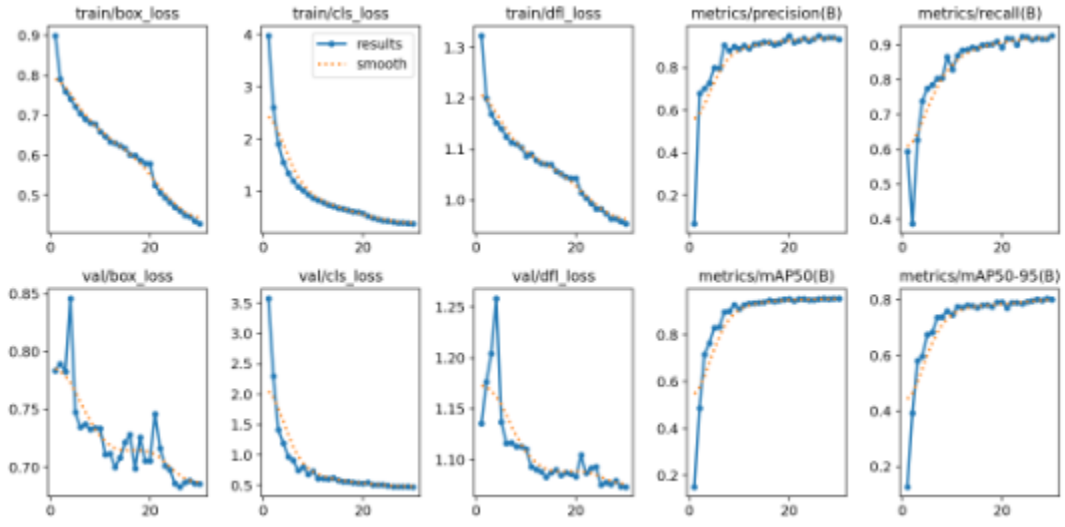
Confusion matrix of yolov8n model.





The confusion matrix and the normalized confusion matrix as presented for the YOLOv8n model confirm that the model can detect several road sign classes, as most categories have good diagonal values. The normalized matrix which factorizes the frequency of each class shows the percentage practice level or degree of the model in different classes, the higher the value closer to 1 the higher the efficient predictions. Yet there are some inconsistencies in the predictions which should be expected, this is especially true of the similar speed limits and the visually similar road signs making these areas prospects for further refinement or additional training data. In summary, the YOLOv8n model met all expectations but should be optimized for particular signs with a closely related visual appearance.

Yolov8n model performance



The training visualizations of this module are very helpful and encouraging as they show the model performance over different training epochs. The train/box_loss, train/cls_loss, and

train/dfl_loss graphs depict a general continuous decreasing trend which implies that learning was indeed taking place and errors were being minimized and corrected over training. Following the trends, the val/box_loss, val/cls_loss, and val/dfl_loss are also on a downward trend albeit with minor spikes, thus indicating good learning but there is room for improvement, especially on unseen data. It is worth noting the strong values seen in the metrics/precision(B) and metrics/recall(B) graphs, whereby recall is in the range of near 1 which indicates detecting true positives with high accuracy. Finally, the last performance graphs, the metrics/mAP50(B) and metrics/mAP50-95(B) also depict increasing mean average precision which suggests that the model performs reliably with multiple IoUs.

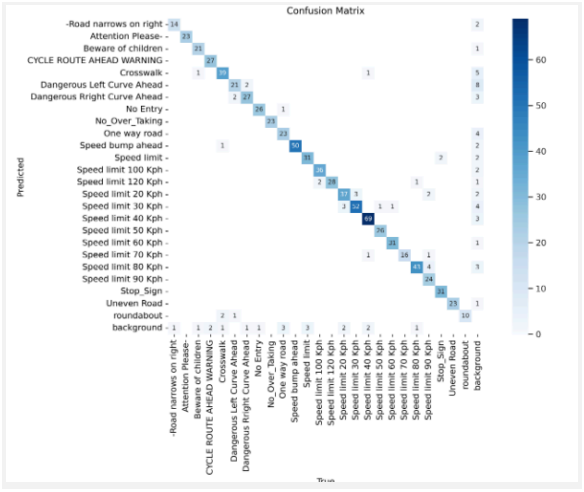
As the ground truth of the test dataset is not available. Calculated IOU on the validation dataset. YOLOv8n's overall mean Intersection over Union (IoU) of 0.832 is impressive as it shows strong congruence between predicted bounding boxes and those that encompass the target objects. The score is therefore useful in showing that the model can localize the objects with a considerable degree of accuracy, further making the model suitable for applications that demand accurate object localization.

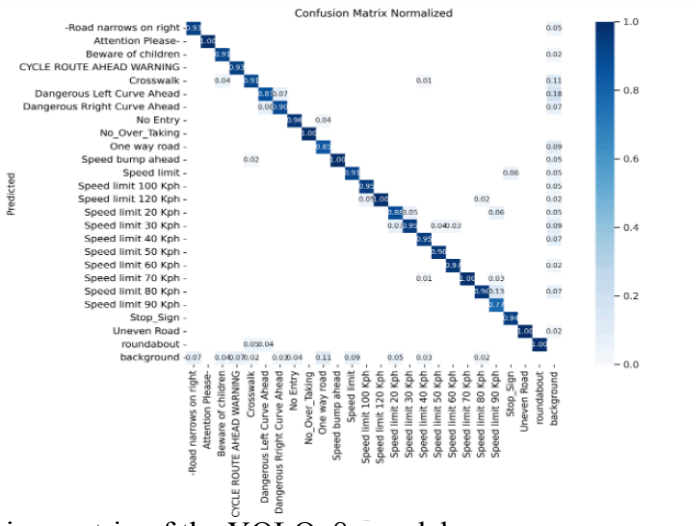
Below are the parameters used for yplov8s

model.train(data='/content/Road_Sign_Detection/data.yaml', epochs=20, batch = -1, optimizer= 'auto', amp=True

1. Number of epochs was set to 30. That means the model will go through the entire dataset 30 times.
2. The batch size is given as -1, which means allowing the model to select the batch size based on memory and resources.
3. The optimizer is set to auto. It actually allows the model to choose the appropriate optimizer based on the library defaults.
4. To speed up the training process enable Automatic Mixed Precision (amp=True).

Confusion matrix of yolov8s model.

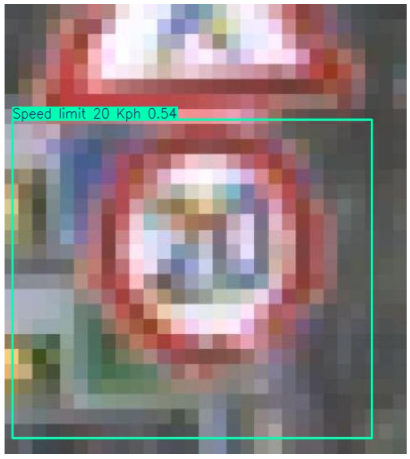


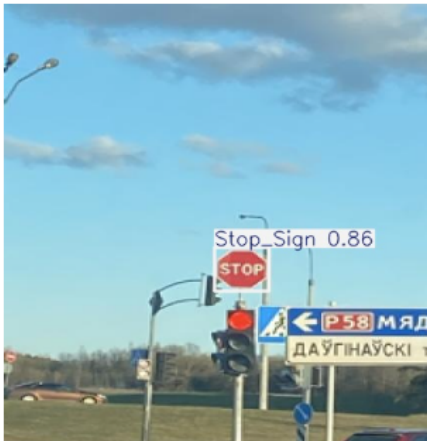


The normalized confusion matrix and the standard confusion matrix of the YOLOv8s model provide classification performance results for individual road sign classes. In the normalized matrix, high values of the diagonal elements clustered around 1 indicate correct predictions for many classes which, in this instance, means that the model gets these signs right most of the time. Some off-diagonal values show the number of errors made due to having a class too similar in appearance to the other classes, such as different speed limit signs or other caution signs. As a whole however, the YOLOv8s model does perform at a decent level but does make mistakes in differentiating between signs that look similar to each other which could benefit from more tuning or more training images of the corresponding classes.

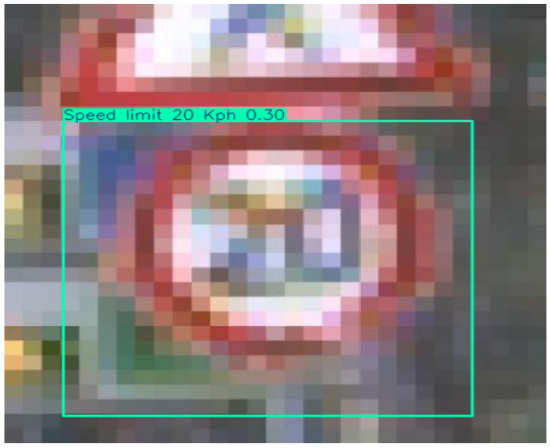
The YOLOv8s model showcased an average IoU of 0.859 which was the highest in all matches indicating higher precision and accuracy in the localization and alignment between the fitted bounding box and the ground truth box. This implies that the model is capable of detecting and localizing the objects quite accurately. Considering the challenges posed for object detection tasks, the model has performed quite well. However, the score (0.88079) in the Kaggle competition is higher with the 8n model than the 8s model.

Below are the few predicted images by the yolov8n model





Below are the few predicted images by the yolov8s model



Comparison of yolov8n and yolov8s architectures

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6

Reference: https://github.com/autogyro/yolo-V8

Size (pixels): In both models, the input size is fixed at 640x640 pixels during training and inference thereby maintaining the same image resolution.
mAP (Mean Average Precision) val 50-95:The MAp score range from 0.5 to 0.95 iou score is of each model, hence indicating their accuracy. YOLOv8n had an mAP of 37.3 whereas YOLOv8s

managed to obtain a more positive mAP score of 44.9 which indicates why YOLOv8s performs better in object detection on higher IoU threshold volumes.

Speed (CPU ONNX): This is the inferencing metric whereby, a CPU was used in running the ONNX in a graph on the Nvidia A100 GPU for inference. The YOLOv8n model makes a running time of 80.4 ms in each inference while that of the YOLOv8s is 128.4 ms making this model slower than the former one. YOLOv8n is considerably more performant on CPU execution targets.
Speed (A100 TensorRT): This is for evaluating the inference running time on an NVIDIA A100 GPU leveraging TensorRT optimizations. YOLOv8s takes 1.20 ms, YOLOv8n leaves no time at all and simply takes 0.99 ms to run. This disparity is slightly less due to the YOLOv8n being faster however the overall tradeoff on speed is accuracy.
Parameters (M): The parameter count is a basic indication of how complex a model is. YOLOv8n had 3.2 million parameters out of which YOLOv8s had 11.2 million thus making it larger and more complex than the more sophisticated YOLOv8s version.

Floating point operations (FLOPs) serve as an indicator of the complexity of sound systems calculations. It is understood that the layer given attention in YOLOv8n has 8.7 billion FLOPs while that of YOLOv8s has 28.6 billion, thereby making the latter much more demanding in computational resources.

Comparison of yolov8n and yolov8s model on Road Sign Detection Dataset.

Model	Parameters	IOU (validation set)	Kaggle Competition Score	Complexity of Model
yolov8n	epochs=30, batch = -1, optimizer= 'auto', amp=True	0.832	0.88079	3.2 million parameters
yolov8s	epochs=30, batch = -1, optimizer= 'auto'	0.859	0.85365	28.6 million parameters

13. **Report on the model and IOU results:**

CNN Multi-Layer Perceptron model called BasicObjectDetector is chosen for the implementation with the following architecture:

Convolutional Layers:

1. ***Conv Layer 1***: input 3 channels (RGB), 16 filters (3 × 3), and padding= 1.
2. ***Conv Layer 2***: input 16 channels, 32 filters (3 × 3), padding =1
3. ***Conv Layer 3:*** input 32 channels, 64 filters of size 3 × 3, padding of 1.
4. ***Pooling Layers***: Max Pooling with kernel size 2 × 2 applied after every convolutional layer halves spatial dimensions such layers as Max Pooling layer

Fully Connected Layers:

1. ***FC Layer 1***: The output from the convolutional layers is flattened and linked with 512 neurons.
2. ***FC Layer 2***: 512 neurons are linked with 4 output neurons, which represent the x-min, y-min, x-max, and y-max coordinates which correspond to the bounding box coordinates.

Activation Functions:

1. ReLU (Rectified Linear Unit) – this is a linear combination that is connected to each of the convolutional and fully connected layers except the final output layer.

Justification for the Choice of the Model:

The BasicObjectDetector was selected with the following justifications:

1. Straightforward components of architecture help to grasp the basic principles of object detection with CNNs.
2. Less complicated models are straightforward to implement and debug over complicated ones.
3. The construction of the model from scratch reinforces knowledge of the different layers, activation functions, and data flow through the neural network.
4. Establishes a baseline to weigh the performance against pending further developments or more advanced models.
5. Applicable in scenarios where the computing power is low and so many experiments and iterations can be done quickly.
6. Facilitates stepwise improvements in performance by adding and changing sites to the adverts.

IoU Results:

The evaluation of the model was performed using the Intersection over Union (IoU) metric on the test dataset. A benchmark is established about an IoU threshold of 0.25 to say that detections were successful.

Evaluation Metrics:

1. Average IoU: 0.587, measures the extent to which the boxes predicted by the model overlap with the boxes that are found in the ground truths. This implies that there is moderate precision in the localization of the objects detected within predicted bounding boxes.
2. Detection Accuracy: 78.7%, this means approximately 78.7% of the predictions made were able to meet the IoU threshold; of 0.25. This suggests the model manages to detect the presence of objects but most probably lacks precision in the localization of the objects.

Observations:

1. The learning curve demonstrated appreciable improvement, with the training loss diminishing throughout the fifty epochs of training.
2. The validation loss took a downward trend although this may have been affected by overtraining on the training set alone or variability within the data.
3. Around the final epochs, the model loss began to drop consistently and plateau with changes in loss.
4. The closeness of the training and validation loss/confusion matrices hints at some degree of overtraining.
5. The bounding boxes predicted by the model fit well with the objects in the images.
6. The average bounding box had a moderate precision, in that the contours of the object were not well fitted by the BBs.
7. Across the test, IoU scores were maintained as similarities were observed in the datasets tested.

Hyperparameters:

1. Learning Rate: 0.001, Learning norm ascribed for the Adam optimizer, this Sets the pace for learning without compromising the robustness of the network.
2. Optimizer: Adam, Effective to employ for this function as the model was trained on sparse gradients which varies, also it adjusts its learning rate on its own which is beneficial.
3. Number of Epochs: 50, Sufficient duration for the model to learn the parameters was provided while avoiding prolonged epochs which are likely to increase the chances of overfitting the data.
4. Batch Size: 4, Lowering the batch size accelerates the achievement of stable and generalized gradients.
5. Loss Function: Adjusted Mean Squared Error (MSE), a Custom loss function designed for images with different and variable bounding boxes per image.Aimed at reducing the gap between predicted and ground truth bounding box coordinates.

Comparison of yolov8n and custom model

Feature	YOLOv8n Model	Custom Model (BasicObjectDetector)
Architecture	Pretrained YoLov8n architecture modeled for object detection domain	Customized structure with 3 convolutional layers and two fully connected layers
Training Command	model.train(data='/content/Road_Sign_Detection/data.yaml', epochs=30, batch=-1, optimizer='auto', amp=True)	Uses a custom application of the train-detector function with fifty epochs
Overall Mean IoU	0.832 (valid set) Kaggle score: 88079	0.587
Loss Function	Built-in YOLO loss functions for object detection	Custom adjusted_mse loss, which computes the mean squared error of the prediction in comparison with the ground truth as a loss
Epochs	30	50
Optimizer	Automatically determined	Adam optimizer with learning rate = 0.001
Inference Speed	Fast	Slow

Parameter Complexity	High	Low
Computational Load	High FLOPs	Lower FLOPs since the architecture was made for simple tasks with lower degrees of complexity
Performance	More accuracy since the system will be more general in unseen data	Lower accuracy, with a higher risk of misclassifications
Suitability	Ideal for advanced object detection task requirements with high accuracy	basic or lower-level object detection tasks where high accuracy, and generalizability is not very important.