RESCUE SHALL PASS: EMERGENCY VEHICLE DETECTION PROGRESS REPORT

Haşim Zafer Çiçek
21990629

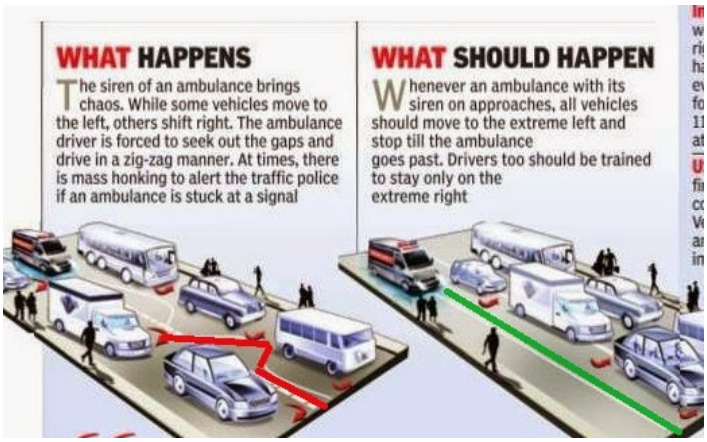
Enes Yavuz
21989712

Mehmet Giray Nacakcı
21989009

June 17, 2023

1 INTRODUCTION

In crowded cities with narrow roads and long waiting times at traffic lights, emergency vehicles have to use flashing lights and loud sirens to alert other vehicles around in order to claim priority. And it depends on other drivers to manipulate the traffic flow (such as driving through a red light) to help emergency vehicles pass through as quickly as possible, and it also sometimes creates chaos.



If traffic controller systems (such as traffic lights) could detect emergency vehicles, and act accordingly; emergency vehicles could get the priority they need, without bringing chaos to the flow of the traffic, and thus reach their destination as quickly as possible. Our aim is to develop a Computer Vision model to provide this detection solution.

Our model should Detect vehicles in photos and (binary) classify them as:

Emergency (police, ambulance, fire truck, and so on)

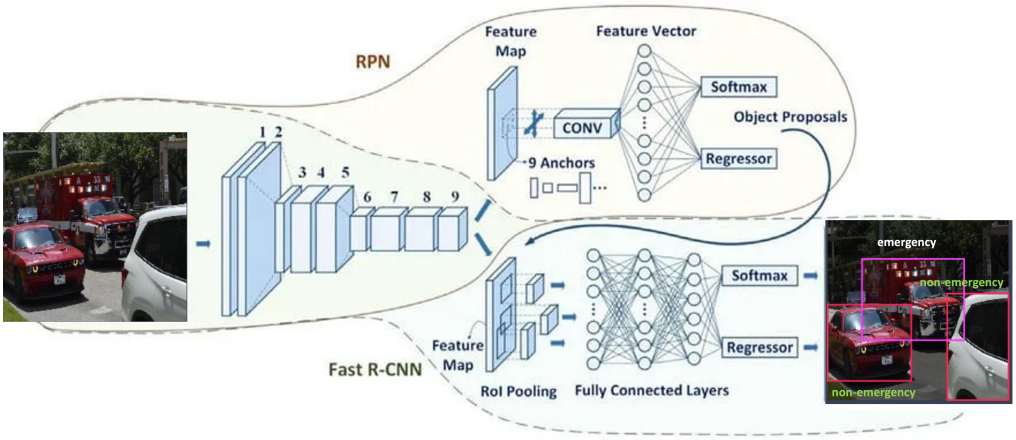
versus

Non-Emergency (car, bus, truck, and so on).

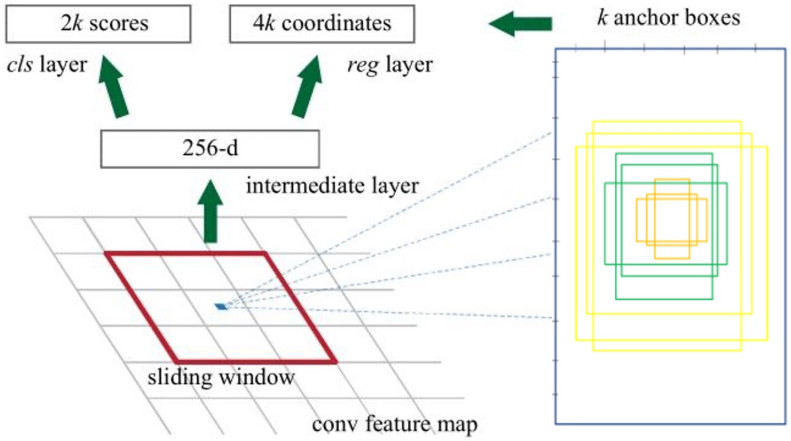
2 METHOD

In our Progress Report, we reviewed emergency vehicles or any detection methods and decided on CNN based methods, since they are domain-independent and transferable, robust, state-of-the-art in Computer Vision. Specifically, we decided on Faster-R-CNN, which provided highest accuracies in shortest time, among the models discussed.

Faster R-CNN (Region-based Convolutional Neural Network) is a popular object detection algorithm that combines Region Proposal Network and Fast-R-CNN; which help with object localization, and efficient computation, respectively.



Region Proposal Network: For each grid of the feature map: K different anchor boxes of different scale and aspect ratios are proposed, regressed into 4D vector defining bounding boxes, and object versus non-object classification probabilities.



Since Faster-R-CNN produces Region Proposals (of any object), then classifies objects, our previously mentioned "binary" classification should actually be 3-class classification:

0. Non-Vehicle (background, another object, false positive region proposal)

1. Emergency Vehicle

2. Non-Emergency Vehicle

```
FasterRCNN Architecture Summary:

Fasterrcnn_mobilenet_v3_large_fpn (100 layers):

    Backbone with Feature Pyramid Network:
        ...
        InvertedResidual blocks
        FeaturePyramidNetwork

    RegionProposalNetwork:

        classification_logits: Conv2d(256, 15)
        boundingBox_predictor: Conv2d(256, 60)

    RoIHeads:
        ...

        box_predictor: FastRCNNPredictor
            classification_score:  Linear(in=1024, out=3)
            boundingBox_predictor: Linear(in=1024, out=12)
```

Since we recently have experience with it from our Lab assignments, we used PyTorch deep learning library. Training process was Transfer Learning. We imported a pre-trained Faster-R-CNN model from PyTorch library, froze all layers, replaced box-predictor as the one seen above and fine-tuned box-predictor.

our code and trained models: `https://drive.google.com/drive/folders/1eeOegaLwydif3pxjH7aKCdAYwbzAH4fH?usp=sharing`

3 Experimental Settings

3.1 Dataset:

`https://universe.roboflow.com/mariem-7lymx/emergency-ixgcc`

This dataset consists of 2200 photos of vehicles, in traffic or parked, taken from different angles, annotated with bounding boxes. These photos contain around 1150 Emergency vehicles, and 1750 Non-Emergency vehicles.

We retrieved the dataset with the YOLOv8 download option, which serves class labels and bounding boxes in txt files. We used the default partition of 1550 train, 440 validation, 225 test images.

We applied normalization, as well as Random Horizontal Flip as Augmentation.

3.2 Development Environment:

We have worked on Jupyter Notebook format. Since CPU power in local development was too weak (30 minutes per training epoch); we have collaborated on and trained our models on Google Colab with Nvidia A100 (state of the art) GPU (15 seconds per training epoch). We saved our models onto the disk, thus can use later (for real-time demo) without training.

3.3 Performance Evaluation:

For the RPN (region proposal network) part of the model, we have provided Localization Precision and Recall, which was calculated based on the confusion matrix (True Positives, False Positives, True Negatives, False Negatives) by Intersection-over-Union (of actual vs predicted bounding boxes).

For the 3-class (Emergency, Non-Emergency, Non-Vehicle) Classification part of the model, we measured Accuracy, and drew a Confusion matrix for Emergency and Non-emergency classes. Non-vehicle class did not appear to be in the output results, eliminated by the network.

We also measured runtimes for training and testing, which we optimized by choosing CPU, GPU, worker-thread-count and such runtime environment parameters.

4 Experimental Results

The Loss we plotted is the calculated as the average of each batch of the following:

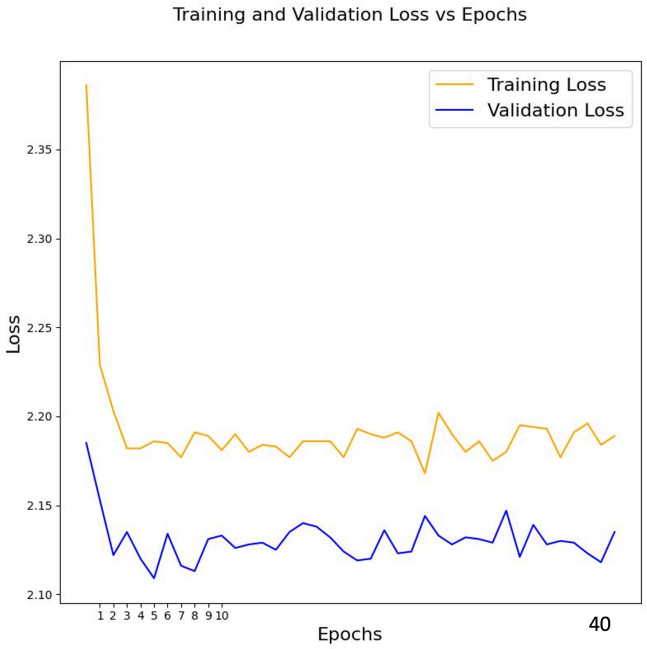
batch-loss = loss-classifier(neg-log-likelihood) + loss-objectness + loss-rpn-box-reg + loss-box-reg

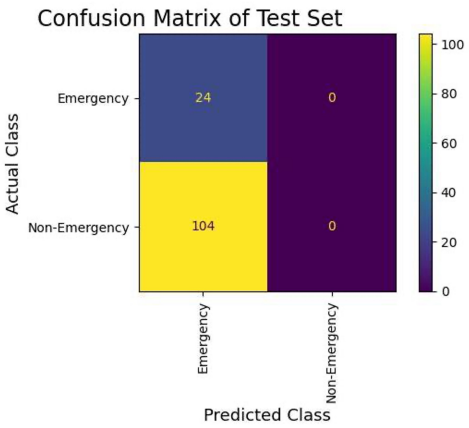
Since it was causing an increase in loss, we did not resize images any smaller than 224x224, which is the original size.

4.1 model 1.

Training: epoch = 40. Learning-rate=0.001. optimizer=scohastic-gradient-descent. batch-size=25. runtime=750 seconds.

Test: runtime=5 seconds. Actual Vehicle : 285 Predicted Vehicle : 655 Bounding box Precision: 0.15 Recall: 0.65 ; classification ACCURACY: 0.19

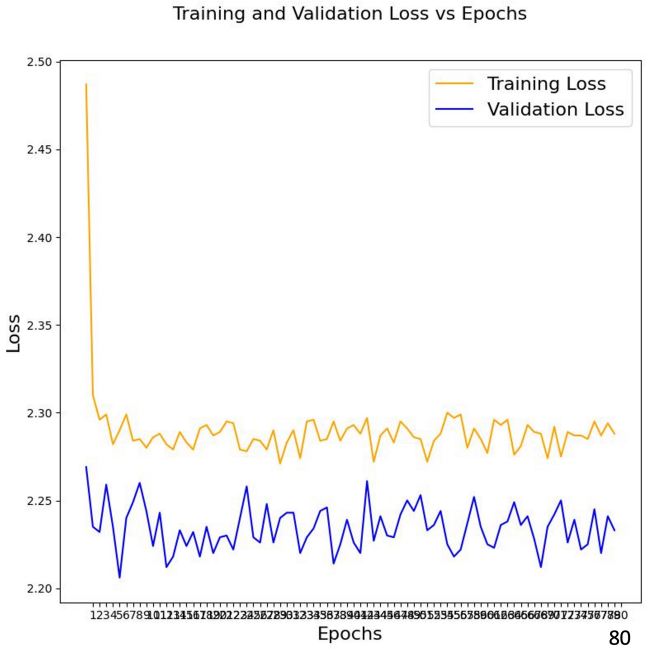


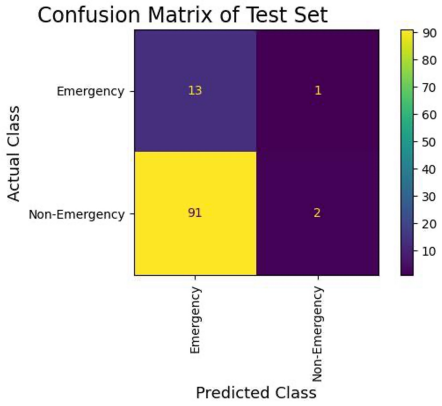


4.2 model 2.

Training: epoch = 80. Learning-rate=0.0001. optimizer=scohastic-gradient-descent. batch-size=25. runtime=1500 seconds.

Test: runtime=5 seconds. Actual Vehicle : 285 Predicted Vehicle : 1479 Bounding box Precision: 0.06 Recall: 0.65 ; classification ACCURACY: 0.14

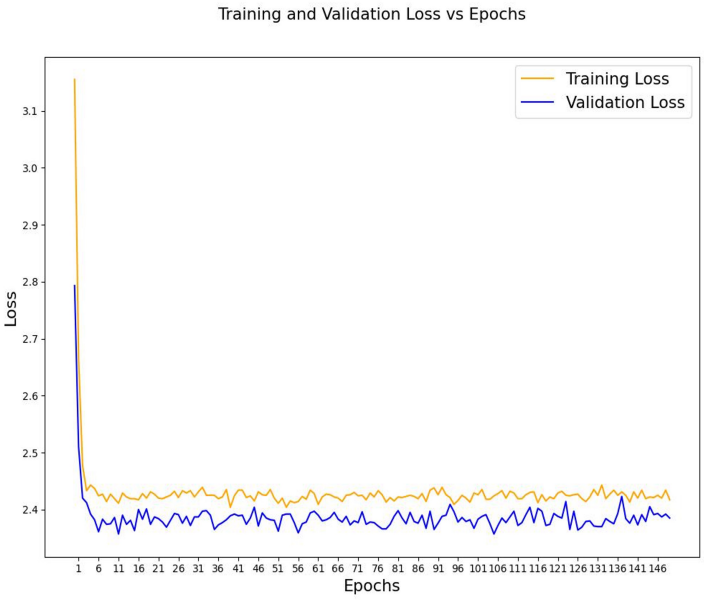


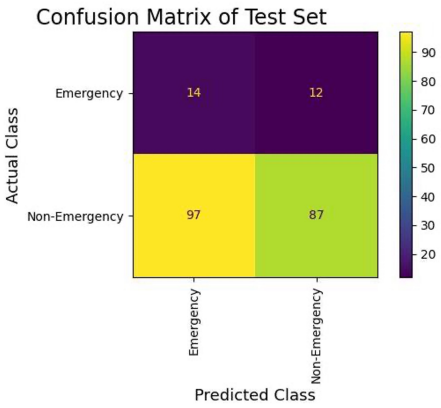


4.3 MODEL 3.

Training: epoch = 150. Learning-rate=0.00001. optimizer=scohastic-gradient-descent. batch-size=25. runtime=2900 seconds.

Test: runtime=5 seconds. Actual Vehicle : 285 Predicted Vehicle : 6127 Bounding box Precision: 0.03 Recall: 0.99 ; classification ACCURACY: 0.48

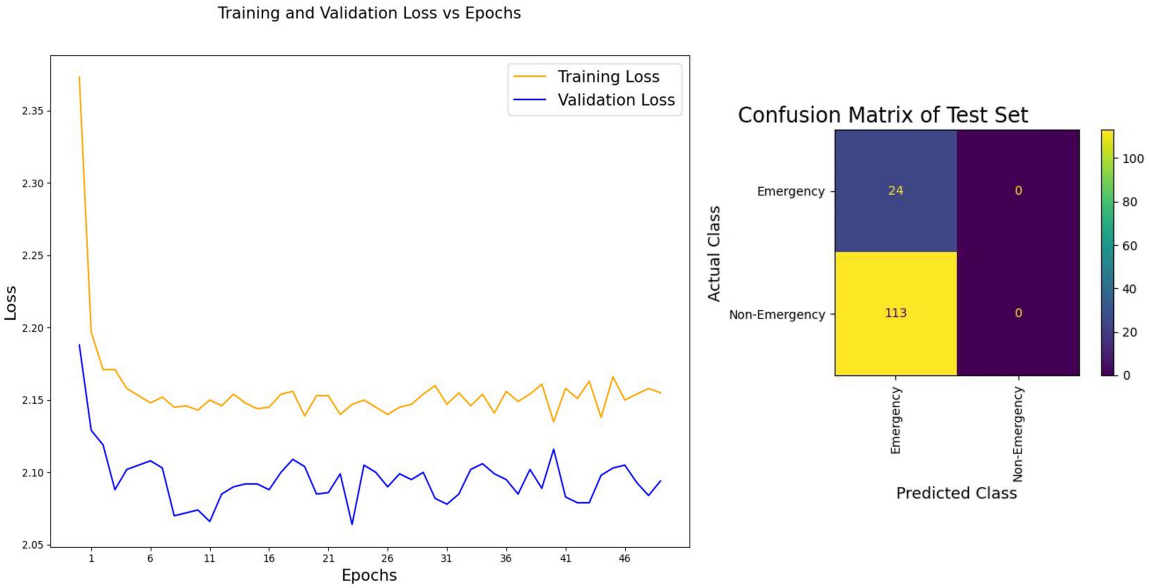




4.4 MODEL 4.

Training: epoch = 50. Learning-rate=0.001. optimizer=Adaptive Movement Estimation. batch-size=75. runtime=1000 seconds.

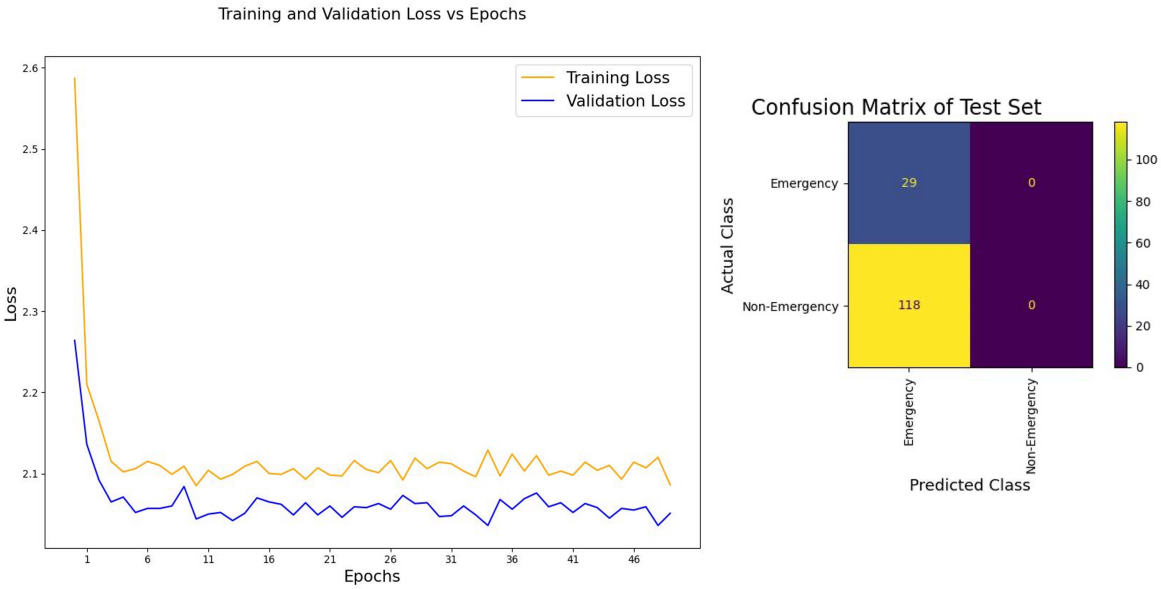
Test: runtime=5 seconds. Actual Vehicle : 285 Predicted Vehicle : 514 Bounding box Precision: 0.21 Recall: 0.69 ; classification ACCURACY: 0.18



4.5 MODEL 5.

Training: epoch = 50. Learning-rate=0.01. optimizer=Adaptive Movement Estimation. batch-size=100. runtime=1000 seconds. Reaching the smallest loss among the 5 models: 2.03.

Test: runtime=5 seconds. Actual Vehicle : 285 Predicted Vehicle : 406 Bounding box Precision: 0.3 Recall: 0.7 ; classification ACCURACY: 0.2



5 DISCUSSIONS

Fine-tuning a state of the art model, we were definitely waiting accuracies close to 0.90, and more homogenous confusion-matrices, yet;

There is a **major bug**. Since our dataset ground-truth labels were 0 and 1, we should have mapped them to 1 and 2 before training, since pytorch FasterRCNN uses 0 as background class. Thus, we accidentally trained emergency vehicles as background; non-emergency vehicles as emergency, and no vehicles as non-emergency, thus non-emergency prediction count is always zero.

We will try to fix this bug before our presentation.

In the presentation, we will also run a demo and present visual results, how the detection looks like, annotated on example pictures.

REFERENCES

All References are taken from our Progress Report:

`https://drive.google.com/drive/folders/1eeOegaLwydif3pxjH7aKCdAYwbzAH4fH?usp=sharing`

Figures: Ambulance in Introduction section

`https://www.emergencymedicinekenya.org/ambulances/`

`https://www.sampspeak.in/2015/03/give-way-to-ambulance-take-left-wait.html`

Figure: Region Proposals, Figure: RPN

`https://arxiv.org/abs/1506.01497`

Figure: Faster-R-CNN

`https://kaanugurluoglu123.medium.com/nesne-tanma-algoritmas-faster-r-cnn-nedir-1738f0cca8b7`