License Plate Detection and Recognition Executive Report

Introduction:
The problem we decided to solve is license plate number recognition in harsh weather conditions. Now this problem has obviously been solved, so to make it our own we wanted to see if we could detect license plate numbers within difficult weather conditions, specifically snow. Detecting license plate numbers can have many real world applications such as highway ticketing as cameras can easily send bills to houses where the license plate is registered. However, what if it is a rainy or snowy day, will the license plate have difficulty being read? This problem is twofold in terms of machine learning as we first needed to detect a license plate on a car using the pre-trained object detection model Yolov8. Then in order to actually read the plate we had to use a text recognition model called Easyocr. So to summarize this we relied on two models *Yolov8* and *Easyocr*.

1. The *Yolov8* Object recognition model to find the location of the license plate on a sample photo or video (data point).
2. The *Easyocr* model recognizes the license plate located by Yolov8.

Background information:
YOLOv8 is a cutting-edge machine-learning algorithm developed for real-time object detection, as described in a recent research paper. The algorithm relies on a convolutional neural network (CNN) architecture, specifically designed to detect and locate objects in an image with high accuracy and speed.
This latest version of YOLO leverages the strengths of its predecessors, while addressing the limitations that were identified in earlier versions. A significant advancement is incorporating an efficient backbone network called EfficientNeXt, which strikes a balance between accuracy and efficiency. The algorithm also uses various modules, such as feature pyramid networks, spatial attention modules, and channel attention modules, to further improve the system's accuracy.
.

EasyOCR has received significant attention in recent research. It boasts high accuracy and speed in recognizing and extracting text from a diverse range of input sources, such as scanned images, photos, and videos.
The architecture of EasyOCR is based on a deep neural network specifically designed for text recognition tasks. It consists of multiple Convolutional Neural Networks (CNNs) layers that identify and extract the features of the input text, followed by a sequence of Recurrent Neural Networks (RNNs) layers that process the extracted features to generate the recognized text.
One of the standout features of EasyOCR is its ability to recognize text in various languages, including Latin-based languages like English, Spanish, and French, as well as Chinese, Japanese, and Korean. The algorithm uses multiple language models that are trained to recognize each

language. To enhance the accuracy and robustness of the system, EasyOCR incorporates several techniques, such as data augmentation, image normalization, and ensemble learning. The algorithm is also highly versatile, able to handle different text sizes, fonts, and orientations.

Objectives:
The previous version of plate recognition was developed with the two models mentioned above using clear images with little variation in weather conditions. We want to extend the applicable scenarios of the models by modifying the data points with filters, in this case we applied snow filters to 1/4th of the data. In real world scenarios, license plates can be contaminated by different weather conditions. By incorporating weather into the training data points, we can improve the accuracy of the models during sub-ideal conditions.

Methodology:

Gathering Data:
Our first line of duty was downloading a dataset from online. We downloaded a dataset of car images that already split images into a test, training, and validation set. There are over 500 data points in total. Each image also has the correct license plate coordinates written in an annotation file in order to compare the ground truth label to the model prediction during training. The dataset was downloaded from Roboflow.

Modifying Data:
After downloading the data, we leveraged Chatgpt to generate codes that run through all the data images and apply a snow filter to a quarter of the images. We still wanted a majority to be clear images so that the model can still recognize regular conditions and thought 25% was a good percentage of images to apply a snowy filter. After we edited the images we trained the model using pre-existing code from the Yolov8 model on the new data that included snowy images.

Understanding the Yolov8 model:
**At first we thought we could just implement the model and have no issues, but it became apparent that we actually had to understand Yolo and the pre existing model to make any sense of what we were doing. This was our first issue, we could not understand what was happening and decided to deep dive into the code and really see its inner workings. This is what we found.

This pre-existing Yolo training script trains an object detection model on a given dataset (in this step it is only detecting license plates not reading the numbers). It loads the model architecture from a specified model file, loads the dataset using the provided data configuration file

(data.yaml), and trains the model using the specified hyperparameters and optimization techniques.

During training, the script performs data augmentation, such as random cropping and flipping, to help the model generalize to new data. It also calculates the loss using the predicted bounding boxes and class probabilities and backpropagating the gradients to update the model weights.

At the end of each epoch, the script evaluates the model's performance on the validation set by calculating the mean average precision (mAP) and other metrics such as precision, recall, and F1 score. The script outputs the training and validation loss, mAP, and other metrics after each epoch. Below is a more specific look at the metrics:

For Training:

Bounding box loss (bbox loss) is a loss function used to evaluate how well the predicted bounding box locations of the license plates match the ground truth bounding box locations. The box loss is computed using the difference between predicted and ground truth bounding box coordinates.

Classification loss (cls loss) is a loss function used to evaluate how well the predicted class labels match the ground truth class labels. It calculates based on if there is a license plate detected or not. The class loss is computed using cross-entropy loss between predicted class probabilities and ground truth class labels.

For validation:

Mean Average Precision (mAP) is a metric used to evaluate how well an object detection model can locate and classify objects in an image. It is calculated by measuring how many objects are correctly detected (precision) and how many are missed (recall) at different levels of confidence. The mAP is a summary statistic that takes into account the performance of the model across all confidence levels. A higher mAP indicates better performance.

**At first we were stumped because the model provided training loss but not validation loss, so we thought something was going wrong. However, through some research we realized why. Typical validation loss was not calculated by the model but rather mean average precision was used to evaluate how the model performed on the validation set. This is because forms of validation loss, specifically box loss, are used to optimize the model during training, and are used to update the model's weights to better detect objects in the training data.

However during validation, the goal is to evaluate the performance of the model on new, unseen data. Calculating box loss during validation would not provide any benefit to optimizing the

model, and would only slow down the evaluation process. Instead, during validation, metrics such as mean average precision (mAP) are often used to evaluate the performance of the model in object detection.

Object Detection(Yolov8) Results:

After training(we trained for 120 epochs which took about 1 hour), the training and validation metrics were graphed and saved. The model performed very well and even with the snowy conditions managed to detect a high percentage of license plates correctly. The training losses (box loss and class loss) showed smooth curves that significantly decreased over time, while the validation set metrics (maP) showed a significant increase in precision over time. We did not write the training code, but became very familiar with all parts of it and were able to adapt it to our own custom snowy dataset.

Easyocr and Plate Recognition:
By making some tweaks to the Yolov8 model, we were able to use Easyocr to read the license plates that we previously detected with Yolov8. Easyocr is a widely used text recognition tool.

How it works:

Easyocr works by taking an input image (in our case the bounded box license plate generated by Yolo) and preprocessing it to improve the quality of the image. Then, it uses a pre-trained deep learning model to recognize the characters and words in the image. Finally, it post-processes the recognized text to improve its accuracy.

To test its accuracy, this time we actually had to write our own code. We simply wanted to test if prediction was correct or not, so we downloaded a new data set of 20 images. We then added annotation files for each image with the correct plate number. Using chatgpt we then wrote code which would loop through the dataset and predict the plate and then compare it to the label. The results would then be plotted in a bar graph of number correct versus number incorrect.

Results of Easyocr
Here is where things weren't too hot. Easyocr struggled to recognize plate numbers even in clear conditions. The model did well only on images that were super straightforward and the numbers were very clear. The model predicted only 12% accuracy. We tried various models to improve accuracy, but nothing worked. It seems this is just a difficult problem to solve. Again Easyocr has been pre trained on millions of data points and we simply couldn't make it better. We didn't want to use an easy dataset to just get good results either and wanted to get a true reality of what a car image might look like on the highway. We thought it was fair to indicate our shortcomings but also acknowledge the difficulty as this is one of the best models currently available and it still struggles. It is able to get a lot of letters correct but it gets tripped up when the lighting isn't perfect or the angle of the image is a little off.

Final Takeaway
So essentially our model performed very well on object detection(Yolov8) even in snowy conditions, but had a difficult time with text recognition even in clear conditions. There are many examples of images that work, but in the grand scheme of things the plate number recognition just wasn't strong enough to be considered good. Again, even before we edited the images Easyocr struggled, so it wasn't the snow that affected the results.

Setbacks/Difficulties

Our main setbacks came in testing and developing the dataset for Easyocr. There was no way to verify results, so we had to come up with a solution with the help of chatgpt and create a custom dataset with labels. Also in the same sense we spent a lot of time trying to figure out why the model wasn't working well and trying to improve it, but our efforts just weren't going anywhere. We are glad, however, with our work as we learned a lot and are proud of our solution to test the Easyocr results.

Also, as mentioned a little earlier, one important takeaway for us was the importance of understanding the model you're working with. Anybody can implement somebody else's work but truly understanding it can expand the potential possibilities and help better process the results.

Opportunities for the Future
Undoubtedly for the future we want to find a better way to read license plate images from different angles with different levels of lights. Even before thinking of weather conditions, this is something that needs to be conquered first and a problem we encountered that we never thought we would. Like we said, straightforward images are easy, but the minute they start getting a little shifted,blurred,or dark the model does not perform well. Once we do that hopefully we can move on to thinking about reading license plates in weather conditions. We are excited with what we learned and the possibilities for the future!

Disclaimer:
Yolov8 and Easyocr are models pre-trained and developed by Ultralytics and JaidedAI. ChatGPT is responsible for generating lines of code to fix errors in the code, finding github commands, and colab commands, as well as generating segments of code for use. Also credit to Muhammad Moin for his tutorial and Github repository on this topic. We based our custom model on his work and data, and were able to implement Easyocr using his tutorial. We also used his github heavily to structure our own github and colab notebook.