Car License Plate Recognition

Shauna Blomgren, Hanna Koo, Yash Girish Nadge, Brian Ramos-Lopez

December 14, 2024

1 Problem introduction

In our project, our objective is to utilize machine learning and computer vision techniques to create a model that can correctly identify a car license plate from an image or video. We explore different techniques in object identification, and compare our model to technology that has been deployed in the real world.

2 Motivation and problem significance

Manually extracting and identifying license plates can be very tedious and time consuming, especially given the vast amount of footage captured by surveillance cameras at intersections and the many vehicles that appear in each frame. However, this information is extremely valuable for various applications, such as monitoring traffic, locating stolen vehicles, tracking suspects, identifying unregistered cars, managing tolls, overseeing parking, and more. Developing a machine capable of reading all license plates in a video or image, in partnership with police resources, such as stolen or missing vehicle databases, can help optimize the allocation of police resources and improve public safety. Therefore, we are motivated by the way this technology can improve the quality of life of many citizens in urban areas.

3 Methodology

3.1 Dataset

We start with the following Kaggle data set: `https://www.kaggle.com/datasets/andrewmvd/car-plate-detection/data`. This dataset contains 433 images with bounding box annotations of the car license plates within the image. Annotations are provided in the PASCAL VOC format. This means that for each image, there is also an XML file included which contains metadata about the image, along with the coordinates for the bounding box for each license plate.

Figure 1: Cars0.png

3.2 Model architectures

We use the following process to identify a license plate: extract an image from the video, use a machine learning model to identify the plate region, and use OCR to read the characters on the plate. In order to identify the plate region, we test two different kinds of models: a simple CNN as control, and three YOLOv5 models, which is a newer adaptation of YOLO, a model known for it's real-time performance.

The CNN uses a simple architecture, with two convolutional layers, a pooling layer, and 2 fully connected layers:

```
def __init__(self):
    super(SimpleCNN, self).__init__()
    # Two convolutional layers + pooling
    self.conv1 = nn.Conv2d(3, 16, kernel_size=3)
    self.pool = nn.MaxPool2d(2, 2)
    self.conv2 = nn.Conv2d(16, 32, kernel_size=3)

    self.fc1 = nn.Linear(32*54*54, 128)
    self.fc2 = nn.Linear(128, 4)  # output: [class, x_center, y_center, width, height]
```

We train the model with SmoothL1 loss function, and Adam optimizer for 30 epochs.

We train the YOLOv5 model three different ways:

1. RGB images only
2. Black and white preprocessed images only
3. Black and white images on a model pre-trained on RGB images

For the preprocessed images, we first turn the image to greyscale, then use adaptive thresholding in order to create more clear contrast in images that have uneven lighting. The YOLO models are also trained for 30 epochs.

Figure 2: Image preprocessing steps

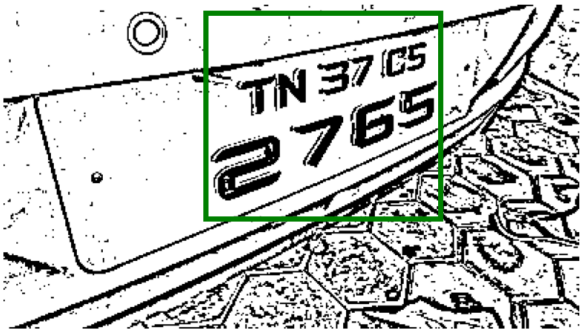

Figure 3: Region of interest highlighted on preprocessed image

EasyOCR is the OCR library we use in order to identify the characters on the plate after detecting it in the image. License plate recognition is a common application for OCR, and is known to be highly accurate. EasyOCR's architecture combines CNNs and RNNs with LSTM. The CNN first extracts features, and the RNN is used to process these features sequentially. The Long short-term memory layers in the RNN help with character predictions through patterns seen over a large number of input images. We hypothesize that the OCR model might benefit from the black and white preprocessed images, and perform more accurately on those compared to the RGB images.

4 Experimental results

4.1 Analysis and comparison of models

We show the loss, testing accuracy, and training accuracy curves for the CNN:

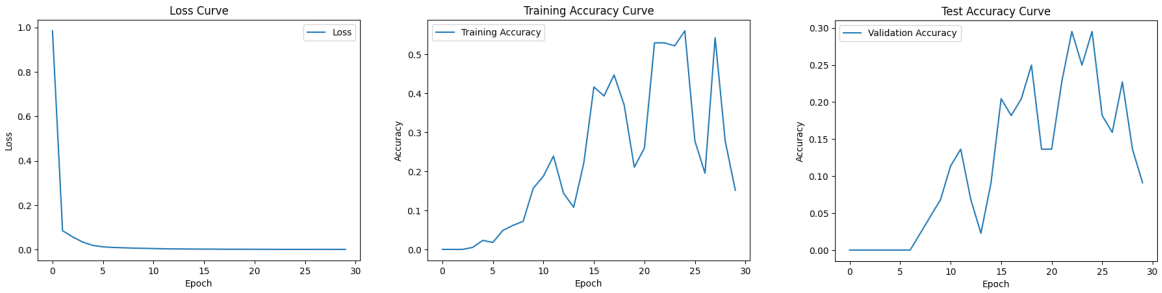

Figure 4: Simple CNN statistics

Although the loss decreases over the epochs, the training and test accuracy do not reach very high. These statistics prove that a simple CNN is not suitable for the task at hand.

Now, we look at the F1 curve and Precision-recall curve for the YOLOv5 models. These statistics are from a validation set mixed with both RGB and BW images. First is the Black and white model:

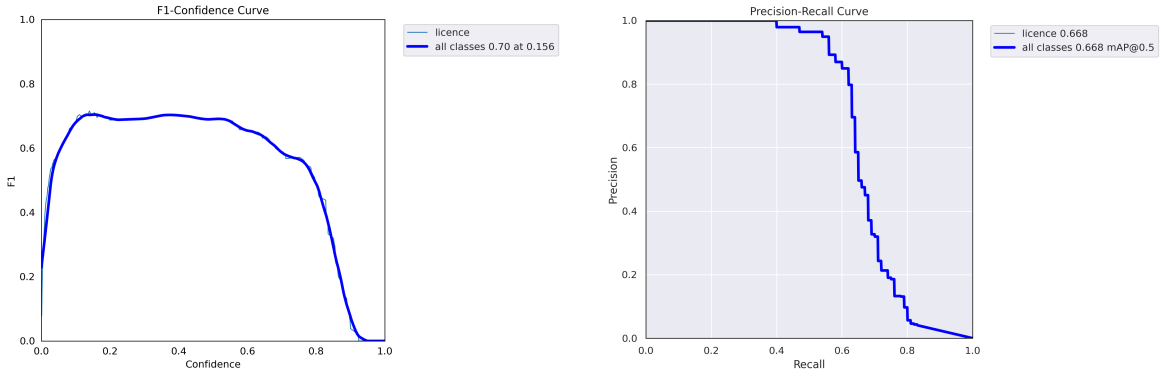

Figure 5: F1 and PR curve for BW YOLOv5

We can see that the preprocessed images have an okay performance, with a peak F1 score of around 0.7 from confidence thresholds 0.1-0.5, a slight dropoff from 0.6-0.8, and a steep dropoff after. The PR curve is also just okay, with a 0.668 peak at threshold 0.5.

Next, we look at the RGB images:

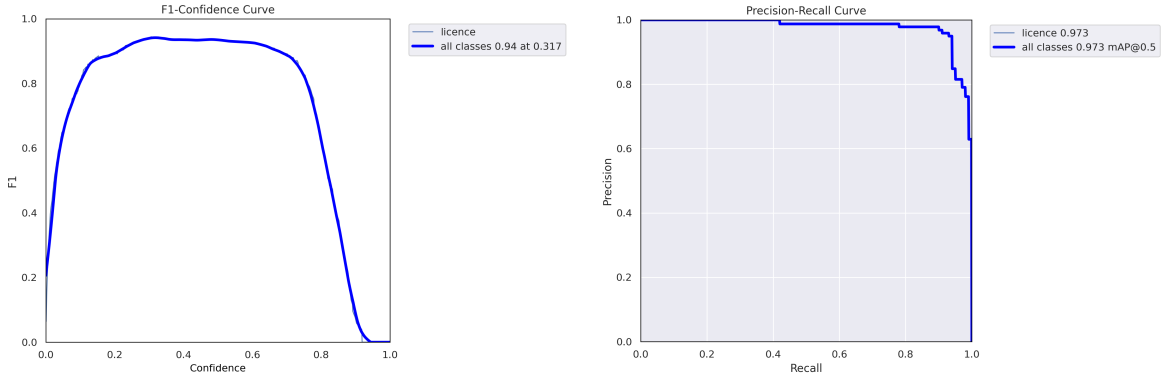

Figure 6: F1 and PR curve for RGB YOLOv5

We immediately see better results, where the F1 curve sees a max score of 0.94 at threshold 0.317, and stays consistent around this range until around 0.75. The precision curve is also very close to the top-right corner, and has a high peak of 0.912. This goes against what we hypothesized earlier when talking about the data preprocessing, which we will analyze later in the report.

Finally, we look at the pre-trained YOLO model trained again on the BW images:

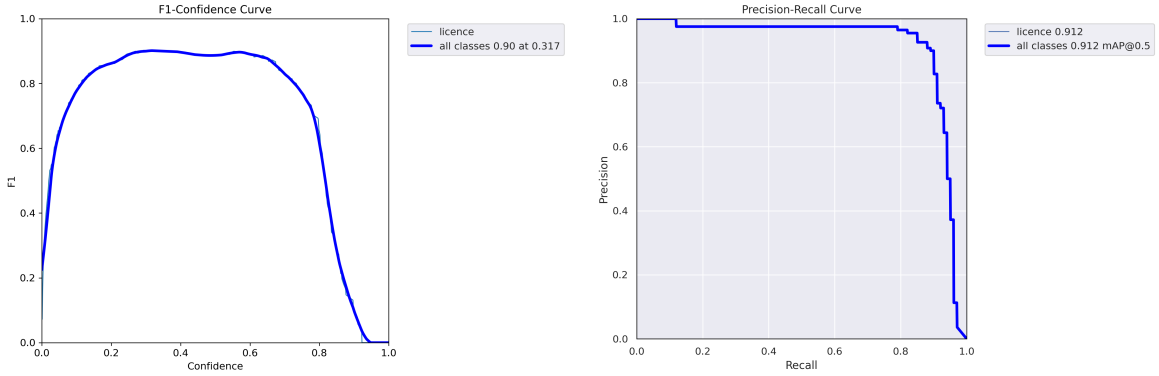

Figure 7: F1 and PR curve for Mixed YOLOv5

The stats are still much better than the BW only model, but slightly lower than the RGB only model. This may be for a couple reasons. Perhaps the BW images are not emphasizing the features that we are looking for, and the model is being skewed away from the optimal weights. Or, the model has been trained too much and is beginning to overfit. Another reason could be that YOLO works best on colored images, depending on how it handles the images in its own pipeline. Although it could be a combination of all of these, when considering the lower scores of the BW only model, the first reason seems to be the biggest. Even though we had preprocessed the data in an attempt to highlight the important features, perhaps we had done the opposite, and hidden them instead.

Because of these statistics, we decided to use the weights of the best performing epoch from the RGB model for our input into the OCR.

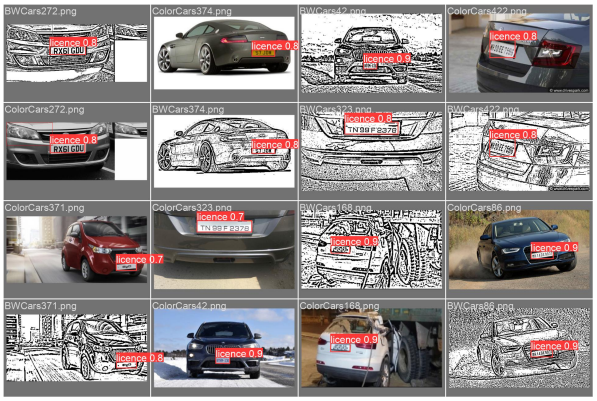

Figure 8: Example validation batch from the YOLO models

4.2 Contextualization of PR tradeoff

In our graphs, we see that the last two YOLO graphs maintain a good balance between precison and recall. When thinking about which statistic is more important within the context of our problem, we can consider whether to prioritize correct identification, or quantity of plates recognized. If our model has a low precision, it would mean that an object in an image that is not a license plate is identified as one. If our model has a low recall, it would mean that a plate that is in the image is not identified at all. These could both have poor consequences, and which statistic is prioritized would be dependent on how the technology is used.

4.3 Transfer learning for YOLOv5

We see how the YOLO model performs better than the CNN. Alongside the obvious reason that the YOLOv5 is a model optimized for object detection, we also can see better performance due to how we use transfer learning for the YOLOv5 models. Because we are importing a model with pre-trained weights, when we train again on our dataset, the model is able to converge much faster. We attempted to utilize transfer learning in a different way with the third iteration of the YOLO model, but were unsuccessful. In theory, training another time with the BW images would help us fine-tune even further and achieve even better accuracy, but this unfortunately did not happen in practice.

4.4 Potential shortcomings of EasyOCR

Our application does not achieve perfect accuracy when reading the plates, and here we attempt to explain why that may be. Although EasyOCR is very effective at character recognition, license plates in specific might not see the full benefit of the EasyOCR architecture. Because license plates are usually a random assortment of characters and do not form words, there are not as many patterns for the RNN part of the model to analyze compared to other applications for character recognition, like language processing. However, license plates are still a sequence of characters, albeit random, and EasyOCR will benefit from that when trying to identify them in an image.

5 Real-world comparison

To see how our academic project compares to a real-world application of ALPR (Automatic number place recognition), we look at an open-source technology OpenALPR. The OpenALPR pipeline is as follows:

1. Detection: Uses LBP (Local Binary Pattern) to identify multiple possible regions on the image
2. Binarization: preprocesses the image to make it black and white with the Wolf-Jolien and Sauovola method

3. Character analysis: Attempts to find character-sized blobs in each region, throwing out the region if it cannot
4. Plate edges: Finds the precise edges of the license plate within the detected plate region
5. Deskew: Corrects the size and orientation of the plate
6. Character segmentation: Attempts to break up the characters on the plate and cleans up the individual character boxes
7. OCR: Compute possible characters and their confidences for each character
8. Postprocessing: Creates a top N list of best possible plate letter combinations, and optionally compares the plates to regional templates if the image location is known

We notice that our model is much simpler than OpenALPR, but it still manages to resemble the basic framework of their pipeline. We preprocess images, detect the region, and use OCR methods to get the possible license plate. However, there are some interesting differences in implementation.

Our project has a focus on machine learning, so we use neural networks for both the plate region detection and the character detection. However, OpenALPR uses LBP for the region detection, which does not inherently use machine learning. In fact, it does not use any machine learning until the OCR step. This might allude towards how computer vision and ALPR have been around for much longer than machine learning has been popular, and techniques for object detection had already been established before models like YOLO. The techniques for the detection step of OpenALPR have not been updated since its creation a decade ago, so the accuracy for OpenALPR likely does not suffer without the use of YOLO. It could also mean that license plate detection is a simple enough problem where machine learning is unnecessary, and only provides more overhead, slowing down the technology. The source code for OpenALPR shows that they use Tesseract as their OCR library. This means that they use an RNN with LSTM, just like we do with EasyOCR. However, they separate each individual character before inputting it into the network, which would increase their accuracy. If we did the same, we would remove the issue of having our network having to separate the characters along with identifying them, which would give the network an easier job of focusing on just identifying one character.

Finally, the other preprocessing steps that OpenALPR uses, like removing plate edges, fixing the orientation, and cleaning up each character all contribute to the tecnology's success. We also attempt some data preprocessing, but OpenALPR gives us some insight onto how we may not have done enough preprocessing for it to be completely effective. Perhaps the augmentation we did contributed more noise to the image, which instead decreased accuracy, and caused the model to fit to unwanted features. If we were to change how we preprocess the data in the future, we would want to add the steps that we were missing from the OpenALPR pipeline, and this would definitely help us achieve higher accuracy.

6 Conclusion

Some insights that we gained from this project are:

1. Data augmentation must be done correctly in order to be helpful
2. There are many different techniques and implementations to get to the same result, each having their own tradeoffs
3. YOLO is a very effective model for object detection, seen by its high accuracy when compared to a simple CNN
4. Transfer learning is a good method to get quicker and better results

7 Group contributions

Shauna: Accessed, annotated, partitioned, and preprocessed dataset, worked on YOLO models
Hanna: Wrote report, researched OpenALPR, made and presented slides
Yash: Worked on OCR and application
Brian: Worked on YOLO models, CNN, and model evaluation

8 Sources

https://www.kaggle.com/datasets/andrewmvd/car-plate-detection/data
https://medium.com/@mohamed5elyousfi/-277e9c685578
https://github.com/JaidedAI/EasyOCR
https://github.com/openalpr