Korean License Plate Recognition Framework using Faster R-CNN and EasyOCR

Yeongjun Go, Minseo Kim, Hyunsoo Kim, and Saehee Eom
Department of Interdisciplinary Major in AI, Seoul National University, Seoul 08826 Korea
saehee99@snu.ac.kr

Abstract

We propose the framework for detecting car plate numbers in a sequence of two models - an Object Detection (OD) model for car plate localization, and an Optical Character Recognition (OCR) model for reading the characters on the plate. We have encountered several obstacles during tackling this task; obtaining the dataset, finding the optimal model, and testing the performance were formidable challenges that demanded rigorous problem-solving and iterative refinement. After thorough performance testing across various models, we ultimately selected "Faster R-CNN" for OD and "EasyOCR" for OCR. Faster R-CNN tested mean IoU of 0.72, and EasyOCR tested accuracy of 87.60%. The final merged model, which has been tested with the dataset of 107 car images - which shows the collaborative efforts of our four teammates, each one of them contributing 20+ images. It detected 87 car numbers perfectly out of 107 car photos, demonstrating its efficacy of handling data with variation, which can be encountered in real-world scenarios. Having successfully built the two-step model structure, we anticipate developing this framework into an advanced one for a multiple of applications in the Intelligent Transportation Systems (ITS) field, ranging from traffic control to urban planning, and also for personal uses such as parking management systems.

Teammates

We are Team No. 2, which has four group members:
(2021-17457) Yeongjun Go / (2021-13425) Minseo Kim
(2021-10112) Hyunsoo Kim / (2019-14505) Saehee Eom

1. Introduction

1.1. About Our Topic

Detecting car license plate numbers is a task critical for building an Intelligent Transportation System (ITS). By detecting the car plate or numbers, we are able to track the individual vehicles, which can be needed from administrative agencies for law enforcement - tracking a vehicle related to crime - or for security purposes to block any vehicles from entering certain prohibited areas, or for traffic analysis which can also lead to development of urban planning or organizing of transportation systems.

License plate and number detections may seem simple because of the fact that the plates are rigid and intra-class appearance consistency is high. But due to various factors, this task has been the subject of numerous studies and, for our team as well, represents a highly challenging endeavor. These challenges include diverse lighting conditions, such as reflections, shadows, and illuminations, as well as unpredictable weather conditions. Additionally, factors like the high velocity of cars, resulting in blurry images, and potential obstructions further contribute to the complexity of the task.

In this work, we have tried to solve this challenging task specifically focused on Korean car plates. This was to simplify the task, since the design of car plates exhibit significant variation. For example, we have noticed that in the United States car plates varied in its design based on the registration area. Car plates in Egypt had Arabic numbers and letters. We have concluded that simplifying the task to the domain in which we have familiarity in would be better than having to generalize a model to a unscalably various cases in terms of efficiency and the expected performance of the model.

Figure 1. Left: US Car Plates of diverse background design, Right: Egyptian Car Plate with Arabic Numbers

1.2. Background Information

Korean car plates have changed its design over the years, and there exists a variety of characters or colors used in the latest version of the design system. Since most of the vehicles we see now are at least from the 2000s, we examined the last 3 versions of designs. The designs are shown in Figure 3.

From 2004 to 2006, only green license plates were used. Vehicles of this version has 7 characters on the plates, separated into two lines. The first line starts with two numbers based on the category of the vehicle, and ends with a Korean Character. The second line had 4 numbers.

From 2006 to 2019, only white plates were used, and the double lined characters were aligned to be a single line. Vehicles of this version also had 7 characters on the plates.

The latest version, applied from 2019 to today, has added one more number to the first two numbers. Thus vehicles of this version have 8 characters on the plates. Depending on its uses or categories, car plates may vary in its color: Battery Electric Vehicles(BEV) and Fuel Cell Electric Vehicles(FCEV) are blue, corporate owned vehicles are light green, and so on.

Having analyzed the training data, we have concluded that the data was imbalanced in terms of the versions. Vehicles with green background were hard to find. Thus we decided to remove any car plate data having been applied the version older than the one revised in 2006.

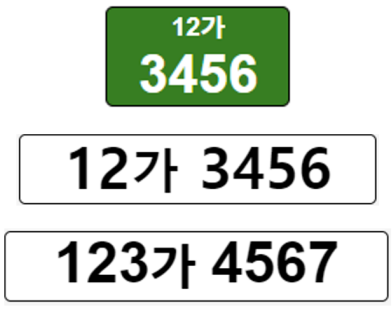


Figure 2. The 3 latest versions of Korean License Plate designs
Topmost: 2004-2006, Middle: 2006-2019, Bottom: 2019-present

1.3. Related Works

1.3.1 The Evolution of R-CNN models

The traditional process of object detection follows three steps: region selection, feature extraction, and classification. Previously, object detection was performed using features such as Haar-like features, models such as HOG and classification models such as SVM, AdaBoost, and DPM. However, these classical methods have had disadvantages - low accuracy and huge computational costs, leading to long execution time. Accordingly, object detection method using Neural Networks were proposed.

R-CNN follows the process of region proposal, feature extract, bounding box regression and classification. [5] Region proposal is finding an area where an object is likely to exist, feature extract is extracting feature for the image, and bounding box regression and classification are finding the location of an object and classifies the class. First, areas in which object exists is proposed through selective search, and crop and resize to make it the same size. This passes through the CNN to extract features with fixed length vectors. Through Classification and regression, bounding boxes are found.

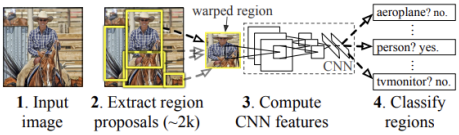


Figure 3. The Procedure of R-CNN Models

R-CNN takes a long time because many regions are found through selective search and each performs CNN. To solve this problem, Fast R-CNN was proposed. [4] Unlike R-CNN that searches features by performing CNN for each region, Fast R-CNN creates a feature map through CNN from the original image and projects the regions into the feature map (RoI projection). Through this, feature vectors can be created just by performing CNN once. These feature vectors are converted into fixed-length vectors through RoI pooling, instead of cropping and resizing like R-CNN. After that, classification and regression are performed through the FC layer.

Faster R-CNN is a model that improves Fast R-CNN's selective search process to run quickly on GPUs. [8] In Faster R-CNN, feature map goes through the region proposal network to find the region. Extracted regions are projected into the feature map and RoI pooling is performed. The region proposition network is a method of moving a sliding window and comparing the anchor box to the ground truth box for each grid cell. Through this method, selective search process can be implemented quicker.

1.3.2 Deep Learning Based OCR Techniques

Text images have information in the order of texts, which correspond to unsegmented data because it is expensive or difficult to separate from the image. One of the main characteristics of such unsegmented data is that unsegmented sub-data forms a sequence. Thus, researchers came up with a method to use CNN and RNN together, and this model is called CRNN. As a feature extractor, a network such as CNN-based ResNet can be used to extract feature from text

images, and the extracted features are converted into a sequential features through map-to-sequence, and inputs of various lengths are put into a processable RNN. Bidirectional LSTM was used because RNN requires wide information to recognize letters from features and requires not only the previous information but also the latter information. Through this, the result of each step is converted into a letter in a transcription layer. [9] CRNN can handle Unsegmented Data using CTC (Connectionist Temporary Classification). CTC is used to process data with different lengths of input and output sequences and decode using Blank tokens to obtain the probability of label sequences in model outputs. Through this, it is possible to obtain a text string by distinguishing the same texts and removing duplicate texts. Edit distance is used to determine the similarity between the predicted word and the actual correct answer. [6] OCR is very difficult in a general image due to the irregular direction or the curved arrangement of texts. By applying Thin Plate Spline (TPS) Transformation, the input image can be transformed to fit the word area so that it is well recognized. After defining the control point, TPS can estimate the change in all locations by iterating the change in the measurement direction when the points are moved to a specific location, thereby creating a change in the overall pixel. [10] Through the spatial transformer network (STN), a network that predicts how much the control point should move can be attached to the front end of the recognition model to fit the image in the forward direction, and the feature can be extracted, which is possible because the TPS operation is a differentiable operation. [7] Recently, Attention and Transformer have begun to be used in the recognition model. [1, 11] Through these preceding papers, this project conceived a Text Recognition model through STN, CNN, and Transformer.

2. Tasks

2.1. Data

Initially, our search focused on datasets containing raw images of vehicles with visible car plates. paired with label data that included both bounding box information and the corresponding character and number strings on the car plates. However, we encountered into a problem of scarcity of datasets meeting these specific criteria. The majority of available datasets consisted of either raw car images paired with bounding box annotations, or cropped car plate images paired with recognized characters. It was almost impossible to find any Korean datasets.

To tackle this problem and successfully move onto the next step, we have concluded that attaining data for OD and OCR individually would be the only solution. Because we wanted to focus on Korean car plates, we had to get Korean car plate data for the OCR models. The location of car plates on the vehicles had little to no difference, so for OD the only important criteria was to obtain data with lables of high accuracy.

The dataset for OD was the Car License Plate Detection Dataset from Kaggle. It had 433 images with bounding box annotations of the car license plates within the image. It was originally from car images that were uploaded by the photographer Gunnar Bjarki on Unsplash, a stock photography website. Although the size of the dataset is not big, it had the most accurate annotations and had the most data from what we have searched from.

The owner of the dataset has uploaded 443 images in .png forms with 443 .xml files for annotations, so we had to preprocess to obtain a single tabular file for the labled data. We used jupyter notebook to do so, and finally got the .csv file. We split the data into training and validation data of size 347 and 86, respectively.

For OCR we needed to use Korean license plate images paired with the annotations of the numbers and characters on them. Thus we searched for Korean dataset platforms, and were able to attain Vehicle Category / Model Year / License Plate Detection dataset from AI Hub a dataset platform ran by Ministry of Science and ICT (MSIT) and National Information Society Agency (NIA). This data was implemented by LAON PEOPLE Inc., and they preprocessed the 2,189 hours worth data they got from surveillance cameras (CCTVs) and additional cameras they had installed in Bucheon-si, Gyeonggi-do. Specifically for the license plate number data, they extracted images from videos and cropped the images to leave only the license plates.

We first had 100,000 images in .png form with 100,000 .json files, and also because we needed to extract certain forms of license plates (only the forms in the last 2 versions of plates) and remove the rest, and there were some annotations with errors, we also had to preprocess for the OCR data. We used jupyter notebook and from the original data of size 100,000, we were able to obtain 61924 images. We split this data into training and validation data of size 55102, and 6822, respectively.

2.2. Model Structure

We have split our task into two parts: Object Detection and Optical Character Recognition. OCR models include Text detection step, which can be regarded as a specific task of Object Detection. But because there could possibly be characters somewhere outside the plates (see the left image in Figure 4), we tried to remove these by applying Object Detection first. By cropping the images close to the car plates, we would be able to minimize the chances of having the irrelevant characters detected. Thus for our OCR model we focused on implementing Text recognition.

Figure 4. These images show why OD needs to be applied first; The left image can lead to detecting 'PORSCHE Cayenne 49너 8528' but the right image is cropped close to the car plate, without the irrelevant characters.

2.2.1 Object Detection

For object detection, we used Faster R-CNN model. Faster R-CNN make anchor boxes with different sizes and ratios. There are three size and three ratios, so nine boxes are created in each grid cell. Convolution layer make feature maps from original image. Region proposal network(RPN) receive feature maps and get class score and bounding box regressor for anchors. And proposal layer extract region proposals using anchor boxes and class scores, bounding box regressor by non maximum suppression and class score thresholding. ROI pooling is doing max pooling with feature maps and samples from region proposals. Finally fc layer get feature vectors and train classifier and bounding box regressor with multi task loss. The structure of Faster R-CNN model is shown in Figure 5.

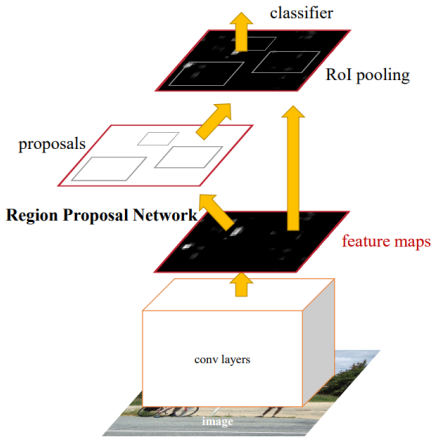


Figure 5. The structure of Faster R-CNN

We also tried implementing non-deep learning systems using traditional Computer Vision methodologies. We first tried a model of combination of known traditional Computer Vision approaches. This model did not need training, so the lack of data was not an issue for this approach. It starts off with Blackhat morphological operation: it enhances black characters on light backgrounds. Next, these several steps were intended to detect large contours from the images, where there could be possibly license plates located in. First, the model fills up small holes in the images - it smooths out the noises using a kernel. Next, binary thresholding is applied using Otsu's method. Then the image intensity is obtained as gradient magnitude, and apply Gaussian Blur on it. Now we are able to group the contours. Get each contour and its size, sort by the sizes, and pick the largest ones - they become the candidates of the car plates, Now lastly pick the contour based on its location in the image - if it is on the outer edges of the images, it is likely that the contour is part of the background. We also tried histogram of oriented gradients(HOG) which is an approach of traditional object detection. We extract gradient from original images. And we express the image as histograms of gradient angle. Next we normalized and represent to vector. The vector become descriptor of HOG. We can calculate similarity between descriptor of original image and template image by inner product. Finally we can detect location which have high degree of similarity.

In conclusion, we used pretrained Faster R-CNN model provided by torchvision. This model is pretrained by COCO dataset with ResNet-50, a CNN structure with 50 layers. Faster R-CNN has different classes that represents diverse objects. Since our purpose is to detect car license plates, we designated two classes for our task: license plates and background.

2.2.2 OCR

We conceived an OCR model based on deep learning to perform text recognition. For the first time, we made the STN-CNN-Transformer model, but it did not perform as well as expected. So we made another model using Vision Transformer, but it did not perform well, too. Therefore we thought other methods except for the deep learning model and tried to perform text recognition through the Bag-of-Words approach. We tried to learn the visual vocabulary through the image of each letter unit and classify new letter image with its frequency of the visual words. But we could not execute BoW because there was a problem that it was very difficult to detect letter unit image in practice. Hence, we decided to perform OCR using the existing OCR library, and tried to use the Pororo OCR library, which has better performance than other libraries. However the Pororo library only runs on torch 1.6.0 version, which cannot be available for now. Then we finally used the EasyOCR library. EasyOCR, an open source python library, is an OCR pre-trained model that supports more than 80 languages. Figure 6 shows the framework of EasyOCR divided into text detection and text recognition. The text detection part use a CRAFT model, and the text recognition part use a CRNN model consist of ResNet-LSTM-CTC ar-

chitecture. [2] EasyOCR can obtain the recognized text, the coordinates of the 4 points of the bounding box, and the probability of the recognized text as the execution result.

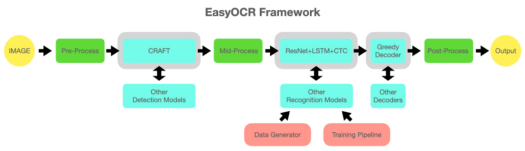


Figure 6. EasyOCR framework.

We designed a model to construct a word list with characters (numbers and Hangul words) that may appear on license plates and return the output corresponding to the indices of the word list. After preprocessing the cropped image through object detection, EasyOCR is performed to recognize license plate number. Then we performed post-processing for those OCR results in order to get them as the indices of the word list.

At the beginning of this project, we had read some prior papers to implement OCR model based on deep learning. Taking inspiration from the preceding papers, we have implemented an encoder-decoder architecture, which is similar to CRNN, that uses a transformer decoder instead of an RNN decoder. [9] The overall model architecture is shown in Figure 7.

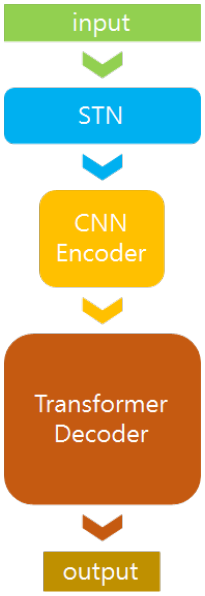


Figure 7. CNN encoder-Transformer decoder model architecture.

Our model consists of a CNN encoder including STN block and a Transformer decoder. Forward propagation is made as follows: (1) resize the license plate image to a gray scale image with dimensions of (1,128, 256). (2) the misaligned image is aligned as it passes through the spatial transformer network block. (3) The aligned images return the feature maps as they pass through the CNN encoder based on Resnet34. (4) In the transformer decoder, the license plate number prediction of length 8 (ex. 1237├5678) is returned through self-attention between output and cross-attention between output and spatial feature maps. (5) In the training process, a word probability tensor of (8, 55) (55 is length of the word list.) is returned and loss is calculated using Focal Loss in order to make the loss computation more sensitive to less frequent cases.

We made another model for the license plate OCR by slightly changing the architecture of the Vision Transformer. In a model of Google Research Brain Team's paper, we increased the number of class tokens to eight, the length of the license plate. Each class token returns each character of license plate. [3] The overall model architecture is shown in Figure 8. The datasets and training details are the same as the previous model.

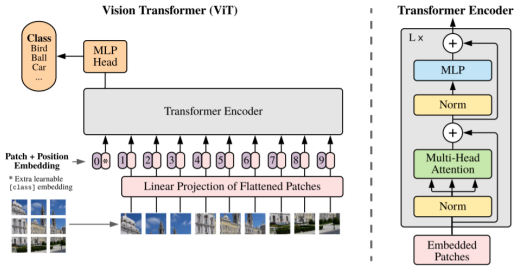


Figure 8. Vision Transformer architecture. [3]

3. Results

3.1. Object Detection

To test the performance of OD models, we conducted experiments with car license plate detection dataset from Kaggle that we have split previously for validation. We chose Interaction of Union(IoU) of detected bounding box and ground truth as the evaluation metric . From the 86 test images, we calculated the area of the correct bounding box and also the predicted bounding box. IoU is obtained by dividing the intersection of these areas by the union of them. Thus for each image we get a single number, and for each model we were able to obtain the mean of these 86 IoUs. The result is shown in Table 1 and Figure 9. The traditional model is marked with the name 'Trad'.

Our Faster R-CNN model showed 0.72 IoU. The detected bounding boxes were located near or on the license plates as we can see in the example images.

We also evaluated the traditional vision model and HOG model with the same data. They showed 0.1 and 0.05 mean

IoU. We found out that traditional models can't detect license plate well.

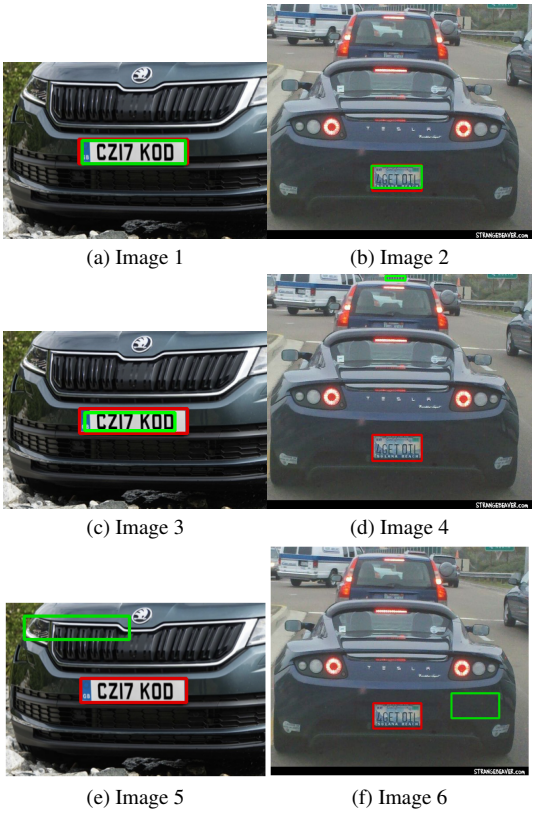
Despite their recognized high performance, traditional methods proved less effective in our context. We examined several potential factors that would be the underlying causes of this outcome.

The traditional model is designed to identify dark characters on a bright background. However, our Kaggle dataset includes images with both dark numbers on bright backgrounds and bright numbers on dark backgrounds. Consequently, this approach falls short in detecting all license plates.

As for the HOG method, it calculates the similarity between the shape of an object and a template image by analyzing gradients. In the case of car license plates, while the border lines may exhibit similar shapes, the individual numbers inside the plate vary significantly. This results in suboptimal performance, as the HOG method struggles to effectively detect license plates.

Model	IoU	Execution Time (s)
Faster R-CNN	0.72	12
Trad	0.10	15
HOG	0.05	314

Table 1. Comparison of Model Performances. We have tested with the Kaggle dataset, and used mean IoU of the images for evaluation.



(a) Image 1 (b) Image 2

(c) Image 3 (d) Image 4

(e) Image 5 (f) Image 6

Figure 9. Example Test Results from 3 models for OD
The red box is the ground truth and the green box is the predicted bounding box.
Image 1, 2: Faster R-CNN, Image 3, 4: Trad, Image 5, 6: HOG

3.2. OCR

3.2.1 EasyOCR

Since EasyOCR is a pre-trained model, training is not required. After filtering according to the license plate type in AI Hub's validation dataset, 6,822 license plate images were obtained. For each image, the input data was obtained in the form of a tensor by resizing it to (1, 128, 256), and the accuracy was measured by performing EasyOCR. In the text obtained by EasyOCR, the string may be separated, the order may be changed, or characters that are not in the word list may be incorrectly detected. Therefore, through post-processing, only the characters on the word list were saved in order by sorting based on the first point of the bounding box, and then changed to the word list index and compared with the label. Label is also pre-processed with the word list index. The evaluation method was evaluated by dividing into two criteria: string unit matching accuracy and character unit matching accuracy. In the string unit matching accuracy, the accuracy was obtained by dividing the count that completely matches the prediction output and the true label by the total number of data. Character unit matching accuracy was calculated by counting the number of characters that matched the prediction output and the true label, and the Longest Common Subsequence (LCS) algorithm was used to avoid problems such as frame shift. The total number matched through LCS was divided into the total number of data and label length 8. As a result of testing the validation dataset, the string unit matching accuracy was 41.07%, the character unit matching accuracy was 87.60%, and each execution time took 166.54s and 139.33s. The results are in Table 2. In addition, when testing with padding images based on the largest image instead of resizing the image in pre-processing, string unit matching accuracy was 43.71%, and character unit matching accuracy was 87.29%.

Matching Criteria	Accuracy	Execution Time (s)
String matching	0.4107	166.54
Character matching	0.8760	139.33

Table 2. OCR accuracy and execution time using EasyOCR.

There was no satisfactory accuracy for string unit matching accuracy, but the reason for this is that the data of the

AI Hub is included in the periphery because only the license plate is not cropped, and pre-processing such as sharpening and hyperparameter tuning for resize is insufficient.

3.2.2 Our OCR Model

The deep learning-based OCR model that we made ourselves faced various obstacles. The biggest problem was the lack of resources to train the model. Google Colab was used for model training because there was no GPU. However, due to Colab's RAM limitations, loading a large amount of image data for training causes a "cuda out of memory" error. To prevent this, a very small batch size and a small number of model parameters must be selected. Since the GPU provided by Colab is not performing well, a small batch size limits the number of epochs because of a significant increase in training time.

Figure 10. License plate text's darkness is different. Left image has black text color and right image has white text color.

If you look at Figure 10, you can see that depending on the type of car plate, the text part is dark in left image and bright in right image. Also, train images contain useless parts around the car plates. So we tried to crop these train images closer to the car plate using our Faster R-CNN model, but it was not perfect, as in Figure 11. Considering its high performance in detecting the car plates from images of cars, we assumed that the bad performance arises from trying to detect the car plates from an already-cropped images (which were not given in the training step).

Figure 11. Bad cases for cropping license plate.

Eventually, despite of our great effort, we failed to train our models.

3.3. Merging Object Detection and OCR

The OD-OCR merge model was tested through 107 car photos each taken by the teammates. Cropped images are

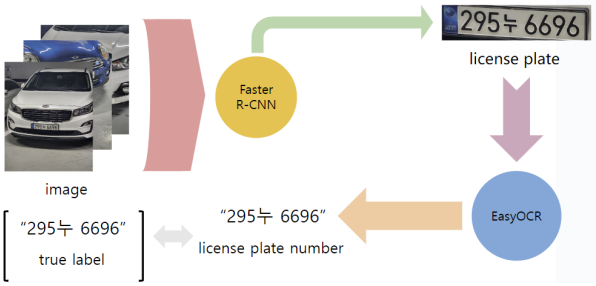


Figure 12. Overall OD-OCR merge model pipeline.

generated through Faster R-CNN from test images and apply preprocessing. In the preprocessing process, gray scale conversion, resizing with (128, 512), and sharpening with strength = 1.3 were performed.

−1	−1	−1
−1	9	−1
−1	−1	−1

Figure 13. A sharpening filter.

After that, the recognized text is obtained through EasyOCR, and the output is generated as a word list index through post-processing as shown in 3.2.1. The evaluation method was also measured as string unit matching accuracy and character unit matching accuracy, respectively, as in 3.2.1. The results of 107 test images of the OD-OCR merge model showed that 87 car numbers completely matched labels.

	In presentation	
	accuracy	execution time (s)
string matching	0.7570	4.02
character matching	0.9650	6.06
	In report	
	accuracy	execution time (s)
string matching	0.8131	5.12
character matching	0.9755	5.21

Table 3. Test accuracy and execution time using merge model.

At the time of presentation, the string unit matching accuracy was 75.70% and the character unit matching accuracy was 96.50%, but during the report period, the sharpening method was further improved to further increase the accuracy. When resized to (128, 256), the string unit matching accuracy was measured very low, about 20%, and as a result of performing hyperparameter tuning on the resize,

the accuracy could be greatly improved. It is thought that the recognition will be performed well if a size suitable for each image is applied to 20 images with the current prediction output different from the label. It is expected that this problem can be further improved through methods such as building a pyramid for various resizes in the future.

4. Conclusion

4.1. Discussion

We have successfully implemented a robust framework for Korean license plate detection, overcoming the challenges associated with limited data to achieve high-performance results. The development of an end-to-end model involved solving diverse problems, possibly stemming from the scarcity of suitable datasets. However, our efforts in addressing these challenges through numerous iterations have yielded a framework that demonstrates exceptional performance rates. We believe that the hours of discussions as a team, with the individual dedication from all the teammates made this possible.

With our license plate detection system, we can look forward to applying this to various purposes.

Governments and Public Institutions can use plate detection for building Intelligent Transportation Systems(ITS). An ITS is an system that is implemented to apply advanced, cutting-edge technologies to the field of public transportation, having the objective to develop the transportation network in a way that the users are safer and more convenient.

In the context of ITS development, our framework can be used for in various domains, including traffic management, law enforcement, and urban planning. Precise quantification of traffic through the detection of car plates is possible, enabling informed communication from the public institution to the vehicle owner.

For legal institutions, our framework provides a means to monitor vehicular movements. This extends to keeping track of stolen vehicles and those implicated in criminal acts.

Beyond law enforcement, our framework contributes significantly to urban planning and transportation system organization. Through the systematic detection of car plates, we gain insights into the detailed ebb and flow of vehicular traffic.

Furthermore, individuals can leverage this system for parking management systems. While license plate detection is already a prevalent feature in parking solutions, the current process requires human intervention for the finalization of parking toll payments. Through the implementation of an integrated system, the prospect of automated toll collection and waivers becomes feasible.

For stronger security, individuals may find value in controlling access to residential areas. Our framework can be utilized to restrict outsider entry, providing a means to log data on individuals who have entered or exited within specific timeframes.

4.2. Limitations and Further Topics

We have achieved a commendable level of accuracy and speed in the detection of license plates - but in a narrow scope. Our emphasis has been on particular categories of car plates, and the model we've developed performs exceptionally well in this targeted context. Looking ahead, we aim to extend the applicability of our model to older versions of license plates or those from diverse regions globally.

For these expanded research tasks, the prerequisites include an extensive dataset characterized by precise annotations and a balance across classes, along with substantial computational resources. This approach ensures the adaptability and generalizationability of our model as it evolves to cover a broader range of license plate variations and geographical considerations.

Exploring the real-time application of our model to images and videos can also be an intriguing next step. This task includes the construction of pipelines to integrate data from cameras, implementing robust data engineering processes to handle high-volume data, and optimizing the model's architecture to ensure its compactness without compromising the desired output. The transition to real-time applications is an integration of data acquisition, processing, and model development, and this integrated system can be applied to practical, real-world scenarios.

References

[1] Jeonghun Baek, Geewook Kim, Junyeop Lee, Sungrae Park, Dongyoon Han, Sangdoo Yun, Seong Joon Oh, and Hwalsuk Lee. What is wrong with scene text recognition model comparisons? dataset and model analysis, 2019. 3

[2] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. Character region awareness for text detection, 2019. 5

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. 5

[4] Ross Girshick. Fast r-cnn, 2015. 2

[5] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2014. 2

[6] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 369–376, New York, NY, USA, 2006. Association for Computing Machinery. 3

[7] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks, 2016. 3

[8] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016. 2

[9] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition, 2015. 3, 5

[10] Baoguang Shi, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. Robust scene text recognition with automatic rectification, 2016. 3

[11] Lu Yang, Fan Dang, Peng Wang, Hui Li, Zhen Li, and Yanning Zhang. A holistic representation guided attention network for scene text recognition, 2021. 3