A Comprehensive CNN-Based System for Vehicle License Plate Detection, Character Segmentation, and Recognition

GMU ECE552 Introduction to Machine Learning and Artificial Intelligence, Spring 2024

Ethan Cheng
Department of Electrical and Computer Engineering
George Mason University
Fairfax, VA, USA
ycheng26@gmu.edu

***Abstract*—Vehicle license plate recognition (LPR) is a crucial component of intelligent transportation systems and plays a vital role in various applications, including traffic monitoring, law enforcement, and parking management. The system employs transfer learning from the EfficientNetB0 pre-trained model to efficiently detect and localize license plates in images. Subsequently, character segmentation and recognition are performed using OpenCV and another CNN model.**

Keywords—license plate recognition, convolutional neural network, computer vision, object detection, character recognition (*key words****)***

I. INTRODUCTION

LPR has gained significant attention in recent years due to its wide range of applications in ITS. Accurate and efficient LPR systems are essential for tasks such as automatic toll collection, parking access control, and traffic surveillance. Deep learning-based LPR techniques have emerged as a promising alternative, offering superior performance and robustness. CNNs, in particular, have demonstrated remarkable success in various image recognition tasks, including LPR. The first stage employs a CNN model based on the EfficientNetB0 pre-trained model to detect and localize license plates in images. This stage utilizes transfer learning to efficiently extract features from the input images and classify regions containing license plates. The second stage performs character segmentation to isolate individual characters from the detected license plate region. This stage utilizes a combination of image thresholding and contour detection to extract character regions. The final stage employs another CNN model to recognize the segmented characters. This stage utilizes a CNN model trained on a dataset of license plate characters to classify each character and generate the final license plate number.

II. BACKGROUND

The first stage involves training a Convolutional Neural Network (CNN) model for license plate detection. The model is based on EfficientNetB0[1][3], a pre-trained model known for its efficiency and accuracy. Transfer learning is employed, allowing the model to leverage the knowledge gained from the pre-training on ImageNet to better recognize license plates[1]. The dataset used for this stage is the keremberke/license-plate-object-detection dataset from Hugging Face[5], which provides a variety of images with annotated license plates. The primary function of this model is to identify the location of the license plate in an image and output a bounding box.

The second stage involves image processing and character segmentation. Once the license plate's location is identified and the image is cropped accordingly, OpenCV, a popular library for computer vision tasks, is used to segment the characters in the license plate[4]. The segmentation process involves converting the cropped image to grayscale, applying a binary threshold to separate the characters from the background, and finding contours in the binary image. Each contour is then bounded by a rectangle, and if the width and height of the rectangle are above a certain threshold, the corresponding character is extracted from the grayscale image.

The final stage involves training another CNN model to recognize the segmented characters[2]. Each segmented character is passed through this model, which has been trained to recognize alphanumeric characters. The output of this model is then combined in the correct order to give the final license plate number.

This project represents a comprehensive approach to license plate recognition, combining the power of CNNs, transfer learning, image processing, and character recognition. It demonstrates the potential of deep learning and computer vision in automating and improving the accuracy of license plate recognition, a task with significant real-world applications such as traffic monitoring, law enforcement, and automated toll collection.

III. PREPARE YOUR PAPER BEFORE STYLING

I used CNN for object detection and localization, which is the task of identifying the location of objects in images. I implemented a CNN model using the EfficientNetB0 architecture, which is a good model for image classification tasks. I adapted this model for object detection by adding three fully connected layers as hidden layers and a fully connected layer at the end to output four values representing the bounding box coordinates of the detected license plate.

$$bbox = (x_{min}, y_{min}, width, height)$$

For the loss function, I use the Intersection over Union (IoU) loss. IoU is a common metric for evaluating the accuracy of object detection models, as it measures the overlap between the predicted bounding box and the true bounding box. You've also implemented a custom metric that checks if the IoU score is greater than 0.5, which is a

common threshold for considering a detection to be successful.

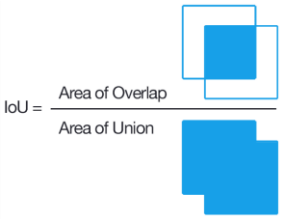

Fig. 1.

$$loss = 1 - IoU$$

The dataset has been divided into training set, validation set and test set. The model is trained on the training set, and the validation set is used to monitor the model's performance during training and adjust the learning rate. After training, you evaluate the model's performance on the test set by calculating the IoU score and the success rate (the proportion of predictions with an IoU score greater than 0.5). I also plot the training and validation loss and IoU score over time, which helps me monitor the model's learning process and diagnose issues such as overfitting or underfitting. Finally, you visualize the model's predictions on test images by drawing the predicted bounding boxes on the images. This allows you to qualitatively assess the model's performance and identify any issues with the predictions..

By segmenting the characters on a license plate and processing them individually, it lays the groundwork for subsequent character recognition using machine learning techniques. The main steps include image cropping, resizing, color space conversion, thresholding, contour detection, and image visualization.

The bounding box coordinates for the license plate are used to crop the original image to isolate the license plate. The cropped image is then resized to a standard size of 270x90 pixels using OpenCV's resize function. The resized image is converted from the BGR color space to grayscale. A binary inverse threshold is then applied to the grayscale image to create a binary image, which helps in segmenting the individual characters on the license plate. OpenCV's findContours function is used to detect the contours in the binary image. Each contour corresponds to a potential character on the license plate. For each detected contour, a bounding rectangle is computed. If the width and height of the bounding rectangle are greater than a certain threshold, the corresponding region in the grayscale image is extracted as a character image. The character images are sorted based on their y-coordinate and x-coordinate to ensure they are in the correct order as they appear on the license plate.

The effectiveness of the character segmentation process is evaluated visually by displaying each segmented character image using matplotlib's imshow function. This allows for a qualitative assessment of whether the correct regions of the image have been segmented as characters.

CNNs have been highly successful in image classification tasks due to their ability to automatically learn hierarchical representations from raw pixel data.

The images are loaded and resized to a standard size of 28x28 pixels. The labels are transformed into one-hot encoded format using LabelBinarizer, which is a common practice for multi-class classification tasks. The dataset is split into training, validation, and test sets. A common practice in machine learning to ensure that the model can generalize well to unseen data. A CNN model is set up using the Sequential API from Keras. The model architecture includes two convolutional layers, each followed by a max pooling layer, a batch normalization layer, a dropout layer for regularization, a flatten layer to convert the feature maps into a 1D tensor, and two dense layers. The final layer uses a softmax activation function, suitable for multi-class classification. The model is trained using the Adam optimizer and the categorical cross-entropy loss function, which is standard for multi-class classification tasks. The training process is performed on a GPU for faster computation.

The model's performance is evaluated on the test set, and the accuracy is reported. Additionally, the training and validation accuracy is plotted against the epochs to visualize the learning process.The effectiveness of the character recognition model is evaluated both quantitatively and qualitatively. Quantitative evaluation is done by calculating the accuracy of the model on the test set. Qualitative evaluation is done by visualizing the predictions of the model on random test images.

IV. EXPERIMENTAL RESULTS

The model was evaluated using a test dataset and produced promising results. The Intersection over Union (IoU) score, a common metric for evaluating the accuracy of object detection models, is approximately 0.68. This shows that there is some degree of overlap between the predicted and actual bounding boxes, indicating that the model is able to accurately identify the license plate's location in most cases. The loss value during the evaluation period is approximately 0.40. While this shows that there is room for improvement in minimizing the error rate, it is a good result considering the complexity of the task.

Table 1.

Metric	**Value**
Test IoU	0.6796
Test Loss	0.3972

a. Evaluation of license plate detection model

The model's predictions were also visually inspected by overlaying the predicted bounding boxes on the test images. As can be seen from the provided images, the model is able to correctly identify and locate the license plate, with the predicted bounding box closely matching the actual location of the license plate.

Fig. 2.

The next step is to segment single characters from the license plate for subsequent recognition. The provided image shows the result of this character segmentation process. Each segmented character is clearly isolated and displayed in grayscale, labeled "Character 1" through "Character 8." The segmentation process seems to be effective, with each character clearly isolated.

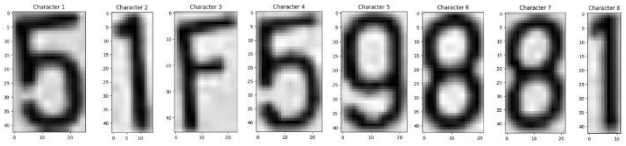

Fig. 3.

The final stage of your project involves character recognition, where each segmented character from the license plate is input to a trained CNN model for recognition. The model's predictions are then used to construct the license plate number.The model, which was trained to recognize segmented characters, achieved a test accuracy of 99.92% and a test loss of 0.0043. This high level of accuracy indicates that the model was able to correctly classify the vast majority of characters in the test set.

Table 2.

Metric	**Value**
Test Accuracy	0.9992
Test Loss	0.0043

a. Evaluation of character recognition model

For each character, the model outputs a probability distribution over the possible classes, and the class with the highest probability is selected as the predicted character. If the maximum probability is greater than 0.9, the corresponding character is added to the license plate number.

V. CONCLUSION

This project has been a profound learning journey, offering a deep understanding of the complexities involved in designing a model. It underscored the importance of not only the model architecture but also the significance of having an appropriate dataset for training. Initially, datasets from Kaggle were considered, but they only contained around 400 images, which was insufficient for training a robust CNN model. The search led to the discovery of a more suitable dataset on Hugging Face, which was instrumental in the successful training of the model.The teachings of Dr. Andrew on object detection were invaluable, particularly the concept of Intersection over Union (IoU). This concept inspired the training process for the License Plate Detection model, proving to be a crucial element in the successful completion of the project.

While the current project has achieved its objectives, there are several potential directions for future work.The current model is designed to detect a single license plate in an image. A possible improvement would be to modify the model to detect multiple license plates in an image. This would likely require a redesign of the model architecture.A long-term goal could be to design a model similar to YOLO (You Only Look Once), which is capable of performing both classification and localization simultaneously. This would be a challenging but interesting direction for future work.

VI. REFERENCES

[1] Tan M and Le Q 2019 Efficientnet: rethinking model scaling for convolutional neural networks Int. Conf. on Machine Learning (PMLR) pp 6105–14 (arXiv:1905.11946)

[2] S. Raj, Y. Gupta and R. Malhotra, "License Plate Recognition System using Yolov5 and CNN," 2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2022, pp. 372-377, doi: 10.1109/ICACCS54159.2022.9784966.

[3] Image classification via fine-tuning with EfficientNet (keras.io)

[4] Zhang, C., Tai, Y., Li, Q., Jiang, T., Mao, W., Dong, H. (2021). License Plate Recognition System Based on OpenCV. In: Jain, L.C., Kountchev, R., Shi, J. (eds) 3D Imaging Technologies—Multi-dimensional Signal Processing and Deep Learning. Smart Innovation, Systems and Technologies, vol 234. Springer, Singapore. https://doi.org/10.1007/978-981-16-3391-1_28.

[5] Keremberke. (2021). License Plate Object Detection Dataset. Hugging Face. Available at: https://huggingface.co/datasets/keremberke/license-plate-object-detection

[6] https://www.youtube.com/watch?v=J-OY4F-z7RA&ab_channel=DatumLearning

[7] Jelal. (2021) License Plate Digits Classification Dataset [Data set]. Kaggle. https://www.kaggle.com/datasets/aladdinss/license-plate-digits-classification-dataset