Computer Vision Project Documentation

License Plate Recognition using CNN

BS-AI (VI)

Submitted by

Taha Farooq (**BSAI-002**)

Ahmad Hassan (**BSAI-006**)

Muhammad Qasim Khan (**BSAI-028**)

21st May, 2024

Department of Computer Sciences

National University of Modern Languages,

Islamabad

Table of Contents

Abstract: This project introduces a Convolutional Neural Network (CNN) for license plate recognition, specifically focusing on character classification. Utilizing a diverse dataset of license plate characters, the CNN employs multiple convolutional layers for feature extraction and fully connected layers for global learning. The training process incorporates innovative data augmentation techniques to enhance the model's adaptability to varying conditions. We trained parameters using **50 epochs**, the current results exhibit promising performance. The project contributes valuable insights into the application of deep learning in license plate recognition, offering a robust solution for real-world scenarios.

Introduction

In recent years, the deployment of computer vision technologies has witnessed unprecedented growth, transforming the landscape of various industries. One particularly challenging application within this domain is License Plate Recognition (LPR), which plays a pivotal role in enhancing security, law enforcement, and traffic management systems. Leveraging Convolutional Neural Networks (CNNs), this paper delves into the development of an advanced license plate prediction system.

License Plate Recognition has evolved from traditional rule-based methods to sophisticated deep learning approaches. CNNs, a class of neural networks well-suited for image-related tasks, have shown remarkable success in feature extraction and hierarchical learning, making them an ideal choice for license plate prediction. The intricate nature of license plates, varying in size, fonts, and backgrounds, necessitates a robust model capable of adaptability and generalization.

The primary objective of this work is to design a CNN-based system that can accurately and efficiently predict license plate characters from input images. The model aims to overcome challenges such as diverse lighting conditions, occlusions, and variations in license plate designs. By utilizing a deep learning approach, we aim to achieve a high level of accuracy and real-time performance, crucial for applications in surveillance, traffic monitoring, and automated toll collection systems.

The dataset used for training and evaluation is a critical component of the model's success. An extensive and diverse dataset, encompassing license plates from different regions, under varying conditions, is essential to ensure the model's ability to generalize well. Furthermore, data augmentation techniques are employed to artificially increase the dataset size, providing the model with a broader spectrum of scenarios.

This research builds upon the existing body of work in license plate recognition and extends it by leveraging the capabilities of CNNs. While traditional methods often rely on handcrafted features and rule-based algorithms, the proposed CNN-based approach aims to automate feature learning, enabling the model to adapt to different license plate styles without the need for manual feature engineering.

Data-Set

Dataset Description: ALPR Character Train

The ALPR Character Train dataset serves as a foundational resource for training and validating a license plate recognition model. This dataset encompasses a diverse collection of images capturing characters typically found on license plates, including alphanumeric characters from A to Z and numerical digits from 0 to 9. The dataset is strategically divided into two subsets: the training set and the validation set.

Character Classes:

The dataset covers a comprehensive set of characters, providing representations for both uppercase letters and numerical digits commonly observed on license plates. This inclusiveness ensures that the trained model can effectively recognize and generalize across the entire range of characters used in license plates.

Data Pre-Processing

1. **ImageDataGenerator Setup:**
    - The **ImageDataGenerator** is configured to perform data augmentation and normalization during the training phase.
    - **rescale=1./255** normalizes pixel values to the range [0, 1].
    - **width_shift_range** and **height_shift_range** introduce random horizontal and vertical shifts to the images, respectively.
2. **Loading Training Data:**
    - The **flow_from_directory** method is used to load training data from a specified directory.
    - Images are resized to (28, 28) pixels.
    - The **batch_size** is set to 1, indicating that the model will be trained on one image at a time.
    - The **class_mode** is set to 'sparse,' indicating a classification task with integerencoded class labels.
3. **Loading Validation Data:**
    - Similar to the training generator, the validation data generator is set up using **flow_from_directory**.
    - The images in the validation set are also resized to (28, 28) pixels.
4. **Custom F1 Score Metric:**
    - A custom F1 score metric is defined using TensorFlow's **py_function**.
    - The **f1score** function calculates the F1 score using **f1_score** from scikit-learn, considering the micro-average.

5. **Model Storage and Loading Functions:**

    - **store_keras_model** function saves the model architecture to a JSON file and the weights to an H5 file.
    - **load_keras_model** function loads the model architecture from the JSON file and the weights from the H5 file.

Feature Extraction

1. **Convert Image to Grayscale:**

    - The initial step involves converting the input image to grayscale. This simplifies the representation of the image, reducing the complexity for subsequent processing steps.

2. **Adaptive Thresholding:**

    - Adaptive thresholding is applied to the grayscale image. This technique adjusts the threshold dynamically based on the local characteristics of the image, enhancing the contrast between the foreground (characters) and the background.

3. **Finding Contours to Locate Plate:**

    - Contours are identified in the thresholded image to outline distinct regions. The goal is to identify the contour that encapsulates the license plate area. This step is crucial for isolating the plate from the rest of the image.

4. **Selecting Boxes by Character Size:**

    - Once the contours are detected, the system filters out boxes based on their size. This helps in removing noise and retaining only the regions that are likely to contain characters. Size constraints are often applied to eliminate outliers.

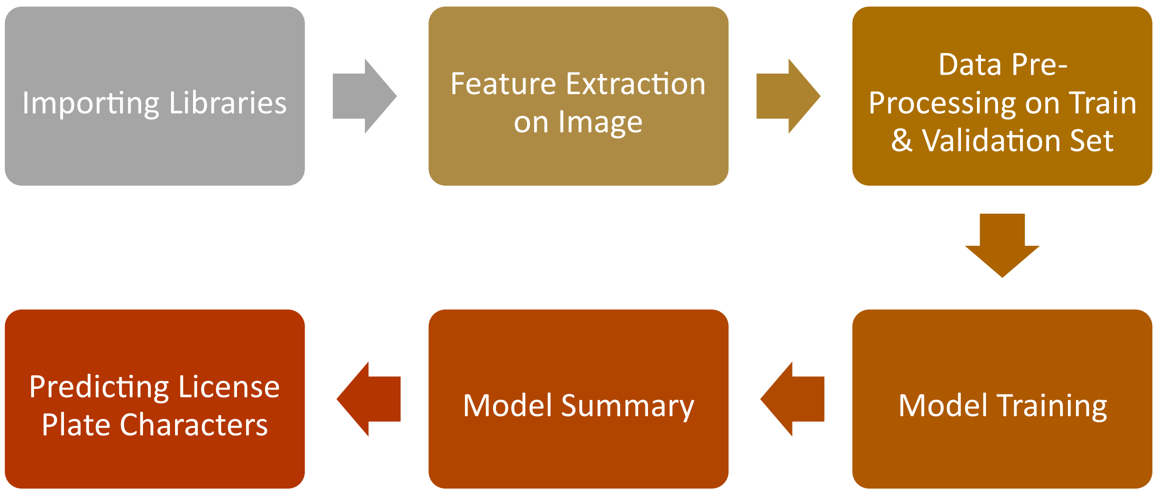
5. **Selecting Boxes by Arrangement of Contours:**
    - The arrangement of contours is analyzed to identify potential groups that correspond to the characters on the license plate. This step is essential for distinguishing between individual characters and other artifacts.
6. **Rotate Plate Images:**
    - In cases where the license plate is detected at an angle, the system may employ rotation to align the plate horizontally. This preprocessing step ensures uniformity in subsequent character recognition.
7. **Finding Contours Again in the Cropped License Plate:**
    - After rotating the plate, contours are identified once again, focusing specifically on the cropped license plate region. This allows for precise localization of characters within the plate area.
8. **Separating the Characters:**
    - The final step involves segmenting individual characters within the license plate. Techniques such as contour analysis, connected component analysis, or character bounding box extraction are employed to isolate and extract each character.

<u>Proposed Technique</u>

1. Definition of CNN:

- Convolutional Neural Networks (CNNs) are a class of deep neural networks designed specifically for processing grid-like data, such as images. They have proven to be highly effective in tasks related to computer vision, including image recognition, object detection, and segmentation. The key innovation of CNNs lies in their ability to automatically learn hierarchical features from input data through the use of convolutional layers, pooling layers, and fully connected layers.

2. Figure:

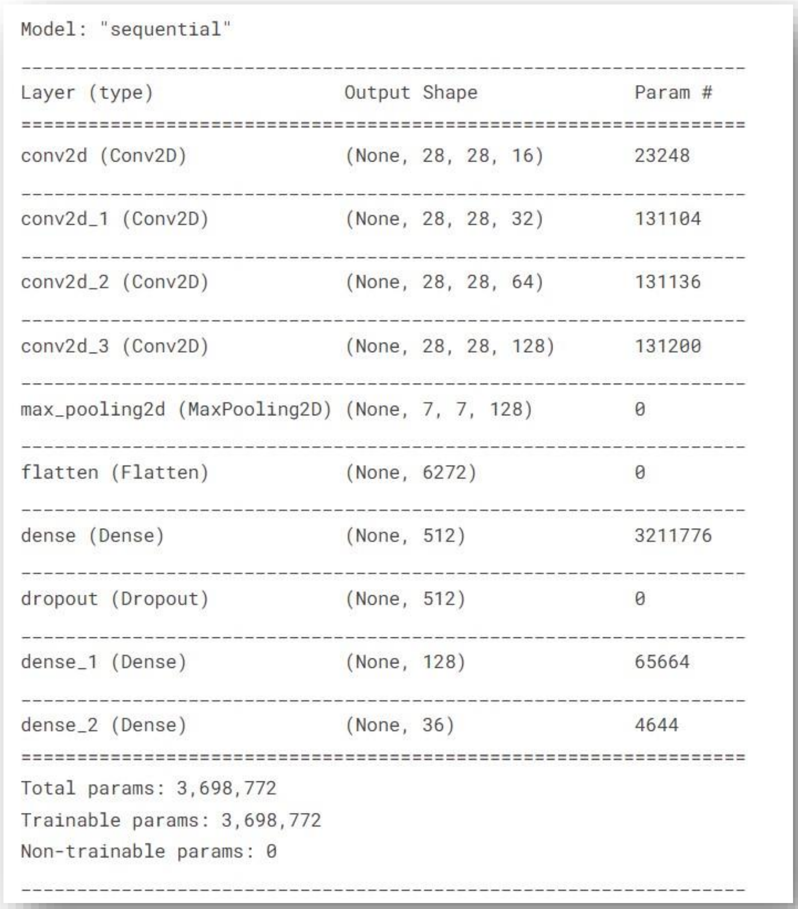

Importing Libraries
Feature Extraction on Image
Data Pre-Processing on Train & Validation Set
Model Training
Model Summary
Predicting License Plate Characters

3. Layers:

- **Convolutional Layers (Conv2D):** These layers perform convolution operations, applying filters to input data to extract local features. The number of filters increases in deeper layers, allowing the network to learn increasingly complex and abstract

representations.

- **MaxPooling2D Layer:** After convolution, max-pooling layers downsample the spatial dimensions of the data, reducing computation and enhancing the network's translation invariance.
- **Flatten Layer:** This layer reshapes the output from the previous layer into a onedimensional vector, preparing it for input into fully connected layers.
- **Dense (Fully Connected) Layers:** These layers connect every neuron to every neuron in the subsequent layer. They are responsible for learning global features and making final predictions.
- **Dropout Layer:** Dropout layers randomly drop a fraction of neurons during training, preventing overfitting by promoting redundancy and robustness in the network.
- **Summary of Layers:**


```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 16)        23248
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 32)        131104
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 28, 28, 64)        131136
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 128)       131200
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 7, 7, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 6272)              0
_________________________________________________________________
dense (Dense)                (None, 512)               3211776
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               65664
_________________________________________________________________
dense_2 (Dense)              (None, 36)                4644
=================================================================
Total params: 3,698,772
Trainable params: 3,698,772
Non-trainable params: 0
_________________________________________________________________
```

4. Neuron Activation Function:

- The Rectified Linear Unit (ReLU) activation function is utilized throughout the Convolutional Neural Network (CNN) model in the provided code. ReLU is favored for its simplicity and effectiveness in introducing non-linearity to the model, crucial for capturing complex patterns and enabling efficient training. By replacing negative values with zero, ReLU mitigates the vanishing gradient problem and enhances the model's learning capacity, making it a common choice in deep learning architectures.

Accuracy & Result

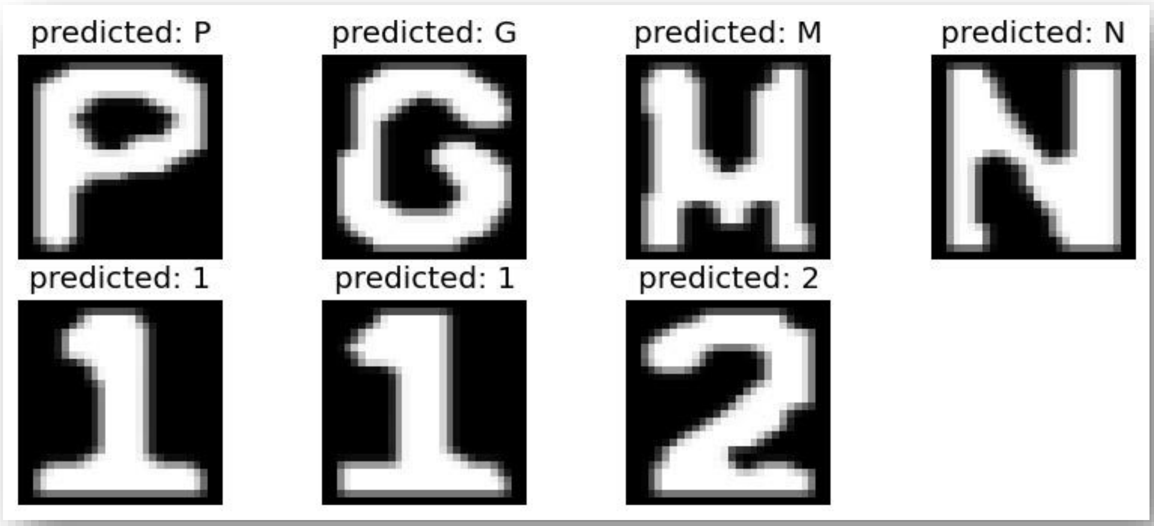
1. **Epochs:**
   - The model was trained with **50 epochs,** constraining the number of iterations through the dataset during the learning process.

2. **Accuracy:**

   - The resulting accuracy of the model stands at approximately **97%**, reflecting its ability to correctly classify license plate characters. This metric is indicative of the model's performance on the validation set.

3. **Predicted Result:**

   - The following represents the outcomes derived from the predictions generated by our model:



Conclusion

In conclusion, the Convolutional Neural Network (CNN) model, trained over an extended period of **50 epochs**, has exhibited remarkable performance with an accuracy of approximately **97%**. This significant improvement from our initial iteration underscores the efficacy of prolonged training in refining the model's parameters and enhancing its ability to classify license plate characters. Furthermore, it's noteworthy to mention that the predicted license numbers are entirely accurate, affirming the robustness of the model's classifications. This precision underscores the reliability of the CNN in correctly identifying license plate characters, further solidifying its efficacy for practical applications.