Letter Recognition

License Plate Recognition

Andrew Snedeker, Catherine Tapia, Christian Saliski

CS301 Section 103

Professor Monogioudis

November 28, 2021

Abstract

The problem we have investigated is character recognition and computer vision. This project focuses on number plate recognition. The data set comes from kaggle and it includes a small number of images of numbers from 0-9 and letters from the English alphabet. The use of deep learning made it possible to train our convnets with the amount of data available. The convnets include Conv2D activated with relu and MaxPooling2D which increases the capacity of our network. Additionally, after the size of each image is reduced we apply the flatten layer to make them one dimensional. In an attempt to avoid overfitting we utilized the Dropout layer and since our goal is to classify these images we also used Dense layers to obtain our output. After training the data and testing it our results show that after using deep learning we are able to recognize letters. The classification of numbers proved to be more difficult, and the model we used did not accurately detect them. Training multiple models to rectify this problem was not feasible due to the GPU limitations imposed by Google when using Colab. In addition we have compared our trained data results to EasyOCR which offers optical character recognition. The output from EasyOCR was accurate and read the license plate with no errors.

Introduction

Computer vision allows machines to obtain desired information from images, videos or visual content. Not only does it enable a system to acquire the data, but also identify and classify it. This can be a very powerful tool since the accuracy rate can be extremely high. Computer vision works by acquiring the data, processing it, and understanding the data. This project aims to resolve character recognition in number plates. Our goal is to train a CNN with images of numbers and letters so that it can recognize number plates with a reasonable amount of accuracy. Certainly, character recognition is really important and especially using it on number plates as it is usually used for law enforcement purposes. For instance, finding a stolen car, checking if a vehicle is registered or obtaining other beneficial information can be some of the advantages this feature offers.

After applying multiple techniques and methods to train our model our results showed high accuracy. We decided to test our results and compare them using EasyOCR. In the end our model was able to detect letters a-z; however, it did fail to recognize digits. We chose to compare them using this library since we were familiar with using segmentation, resizing, and filtering on the validation set. EasyOCR showed a great capacity to recognize both numbers and letters. Enhancing some of the techniques we utilize can potentially improve our model so that it shows the ability to recognize numbers.

Related Work

There are a lot of examples related to computer vision and number plate recognition. Certainly, there were more examples about number and letter recognition or using libraries like EasyOCR to obtain the desired results. EasyOCR is a great tool and even though it has a high accuracy result, it still failed to recognize some characters in the number plate when we first used it. After some research, we were able to find some work showing how data is trained from scratch and the implementation of CNN for image recognition. Most of the work related to our project was found in kaggle and the approaches used were similar to the ones implemented in this project. Similarly, we found some more advanced work that used other methods like feature extraction model and Backpropagation Neural Network to achieve the same results with a higher performance.

Data: https://www.kaggle.com/kdnishanth/characterrecognitionfromnumberplate/

The data chosen for this project can be found at kaggle and it's made up of images of numbers and letters. It's separated into training, testing and validation, which all together contain over 30,000 images. Each number and letter is in a subfolder that contains multiple images of each one. Since we are handling JPEG content, we have to pre-process the data accordingly so that we can feed it to our network. In order to do so we have to read each file, decode from JPEG to RGB grids of pixels, convert into floating point tensors and rescale pixel values. With the help of Keras ImageDataGenerator it was possible to generate batches of tensor image data.

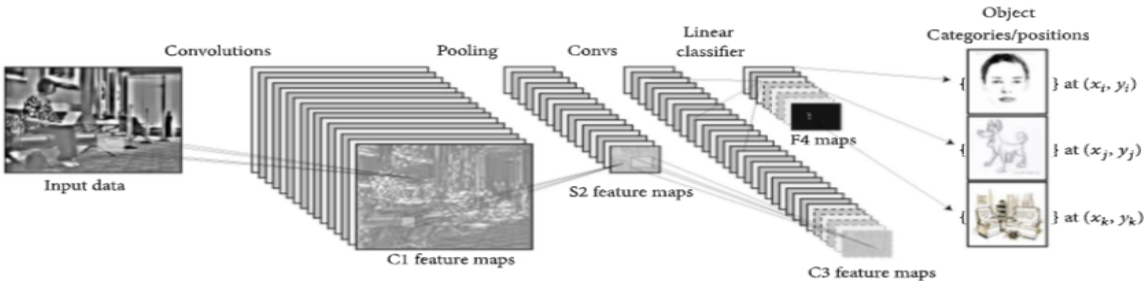
Additionally, since we are working with small datasets we can encounter overfitting as we trained our data. To resolve this problem we have generated more training data from our existing data. Data augmentation solves this issue by applying several transformations that provide our model with a large number of different images. ImageDataGenarator takes parameters that allow us to make these transformations. We were able to apply shear_range and zoom_range. Each one of these transformations is applied randomly to each image to make the necessary changes so that our model can encounter the maximum amount of variation of each image. We also applied validation_split, which split 20% of our training set and allocated it to the validation set.

Method

Deep learning offers one of the most important plans to resolve problems that are related to computer vision. "It allows computational models of multiple processing layers to learn and represent data with multiple levels of abstraction mimicking how the brain perceives and understands multimodal information, thus implicitly capturing intricate structures of large‐scale data." (Voulodimos et al., 2018) The reason behind the creation of neural networks is to replicate the human brain's capacity of recognition. The project aims to recognize number plates so the method we decided to implement is Convolutional Neural Networks. This approach is known to have great results in pattern recognition.

Figure 1

Example architecture of a CNN for a computer vision task (object detection).

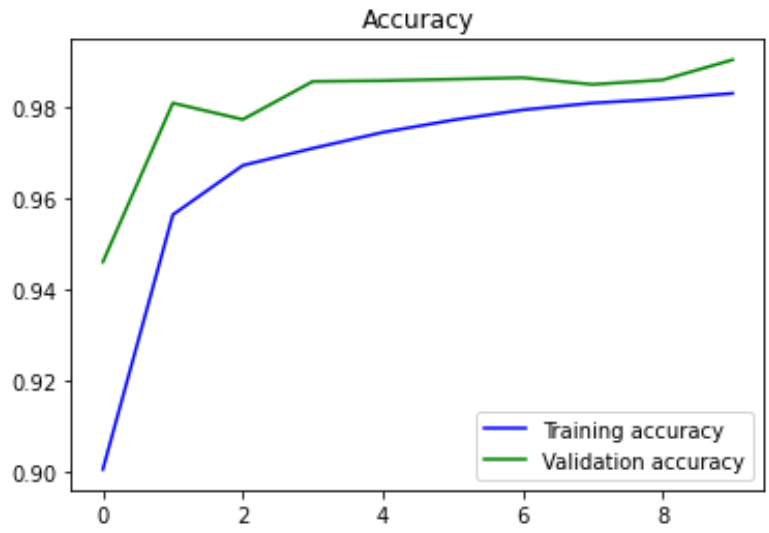


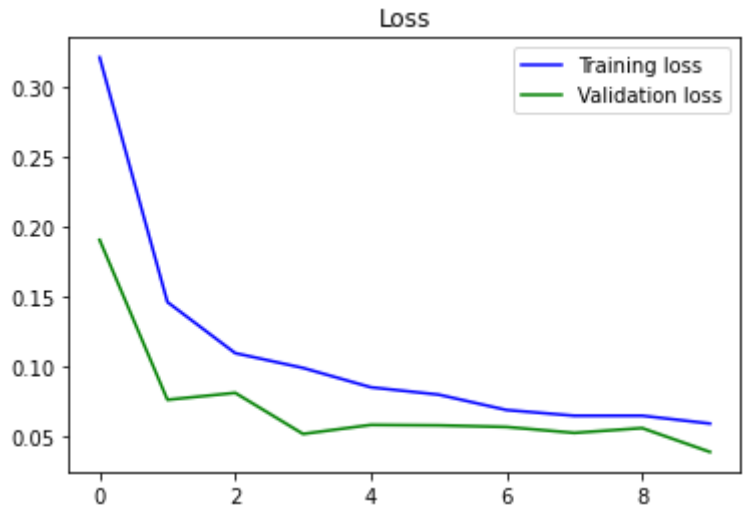
A CNN is made up of three main neural layers that include convolutional, pooling and fully connected. The convolutional layer applies a filter to the images and maps each feature depending on their location and strength. Then to reduce the size of the feature maps, we apply the pooling layer. This strategy leads to some information loss that is beneficial for the network as it does less computational projections which also decreases the chance of overfitting. Also, we encounter the fully connected layer that takes care of keeping layers linked. "Fully connected layers eventually convert the 2D feature maps into a 1D feature vector" (Voulodimos et al., 2018). During this layer all the techniques applied are finally flattened and classified into our output. The reason behind choosing this approach is that a CNN uses kernels that are trained to detect specific features. Also, the techniques that were previously mentioned facilitate the learning process and are meant to provide a higher accuracy rate. We encounter other approaches like RNN that also have a similar structure; however it is mostly used in sequence, sentiment, and video classification.

	Convolutional neural network (CNN)	Recurrent neural network (RNN)
ARCHITECTURE	Feed-forward neural networks using filters and pooling	Recurring network that feeds the results back into the network
INPUT/OUTPUT	The size of the input and the resulting output are fixed (i.e., receives images of fixed size and outputs them to the appropriate category along with the confidence level of its prediction)	The size of the input and the resulting output may vary (i.e., receives different text and output translations—the resulting sentences can have more or fewer words)
IDEAL USAGE SCENARIO	Spatial data (such as images)	Temporal/sequential data (such as text or video)
USE CASES	Image recognition and classification, face detection, medical analysis, drug discovery and image analysis	Text translation, natural language processing, language translation, entity extraction, conversational intelligence, sentiment analysis, speech analysis

Experiment

Developing a CNN to create and train our model proved to be very difficult for beginner data science students like ourselves. We used an open source CNN that was developed for this specific problem in order to develop and train our own model. The CNN we used and modified can be found here: https://www.kaggle.com/sweta88/notebook0e193408a2. As we trained our model, we experimented with changing the hyperparameter to compare the results. However, we did encounter some problems due to the amount of time it took to train the data set. Also, we had limited use of the GPUs Colab offers. To solve our optimization problem, we decided to use adam in the compilation step and categorical cross entropy as our loss to handle the multiple classes. Similarly, to evaluate our model we used Keras' accuracy metrics and we were able to achieve great accuracy, but as we tested our model our results were not as accurate.





When it came to detecting numbers our model failed most of our test cases and did not recognize the numbers that we ended up segmenting and extracting from the test image of a license plate. In order for the image of the license plate to be recognized by both EasyOCR and our model, we needed to perform multiple transformations. First we converted it from BGR to grey, and we modified it further to perform edge detection. From that point we needed to continue transforming the image to find the contours and apply masking so that the plate could be properly cropped. After it was cropped it was further transformed and segmented using two open source functions from: MobileNet V2 Implementation. This allowed us to extract the individual characters and feed them into our model to determine the accuracy of it. Most of the public work we came across and chose as reference did have similar techniques and great results. The steps and hyperparameters used were much larger and their training models were able to recognize most if not all of the images during the testing process. This showed the correlation between the use of these techniques and having great results.

Conclusion

Computer vision is the field that enables computers to classify and interpret visual inputs. It has been able to develop and improve its results with the help of deep learning. This project showed us the relationship between them and how to apply these techniques. We focused on understanding how convolutional neural networks work and following each procedure to achieve character recognition. The objective was to train our model so that it can identify the characters

on a number plate. We decided to use CNN because we were working with images and needed to solve a classification problem. Additionally, the open source material that was available on kaggle, youtube, stack overflow, and other websites showed implementations that used CNNs.

We were able to use Keras, a deep learning python library to implement our CNN model. Creating the CNN proved to be very difficult, and we used an open source CNN that was posted on kaggle that addressed this specific problem. Initially we believed that the network went through too many epochs, and we reduced the size from 35 to 10 to avoid overfitting. Both of these proved to be inaccurate when it came to detecting numbers, however they were much more accurate when it came to letter detection. After adjusting the steps per epoch and number of epochs, our problem persisted. We implemented other CNNs that were discovered from open-source material and they too proved to be ineffective. After training our data our results showed a high accuracy, but after testing it we faced problems identifying digits. We determined that it was possible that it got stuck at some local minimum which is why our accuracy was high during the training, but we experienced some conflict as we tested our data. Furthermore, we compared our results and using EasyOCR showed that we were able to identify all characters easily. After the comparison we concluded that we would need to refine our training methods and neural network to improve the outcome.

References

https://www.hindawi.com/journals/cin/2018/7068349/

https://www.kaggle.com/kdnishanth/characterrecognitionfromnumberplate/

https://www.kaggle.com/nainikagaur/mobilenet-v2-implementation

https://www.kaggle.com/sweta88/notebook0e193408a2

https://keras.io/api/callbacks/reduce_lr_on_plateau/

https://keras.io/api/models/sequential/

https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D

https://www.tensorflow.org/api_docs/python/tf/keras/layers/MaxPool2D

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Flatten

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Activation

https://www.tensorflow.org/api_docs/python/tf/keras/Sequential

https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense

https://www.tensorflow.org/api_docs/python/tf/keras/layers/BatchNormalization

https://keras.io/api/layers/activations/#softmax-function

https://www.applause.com/blog/training-data-validation-data-vs-test-data

https://medium.com/analytics-vidhya/understanding-image-augmentation-using-keras-tensorflow-a6341669d9ca

https://searchenterpriseai.techtarget.com/feature/CNN-vs-RNN-How-they-differ-and-where-they-overlap

https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5

https://stackoverflow.com/questions/44747343/keras-input-explanation-input-shape-units-batch-size-dim-etc

https://www.youtube.com/watch?v=NApYP_5wlKY