Summary and experiments of a license plate recognition neural network model.

Selected Topics in Artificial Intelligence. Professor Gissel Velarde.

Ignacio del Río *
Universidad Privada Boliviana

Erick Martínez *
Universidad Privada Boliviana

Cristian Paz *
Universidad Privada Boliviana

***Abstract*—The main parts of a license plate recognition deep convolutional neural network [1] will be described, such as the model, evaluation, related work, etc.**

I. Introduction

Automatic license plate recognition is a quite challenging and important task, as it is applied in a lot of ambits, and it must adapt itself to different environments and maintain the highest level of accuracy possible. The LPRNET algorithm is based on a deep convolutional neural network, which showed to be very efficient when performing plate recognition, in addition to not requiring many sources at the time of the execution [1].

Datasets

- Chinese plates dataset: dataset of 11696 pictures of real Chinese plates, with different lighting, resolution, angle and blurring [2]. The Chinese plates contain numbers, letters and a Chinese symbol, which represents the plates's province.
- Turkish plates dataset: dataset of 100000 pictures of synthetic Turkish plates, with the same size, lighting, resolution, angle and blurring [4]. The Turkish plates only contain numbers and letters.

II. Related Work

Next, some methods and functions applied in similar projects will be briefly described:

Character Segmentation & Classification

The plate's characters must be segmented, and then classified, and it will be the network's input [1].

Long Short Term Memory (LSTM)

RNN that can store important information from a sequence for a longest period.

Recurrent Neural Network (RNN)

Neural Network capable of processing sequences of images, sounds, etc., which a conventional neural network can´t do.

Spatial Transformer Network (STN)

Network that can change the spatial characteristics of a feature (size, inclination, etc) for different purposes.

Generative Adversarial Network (GAN)

Network capable of generating new elements based on what it learned, and distinguish if the element is original or generated.

III. Model

Basically the design of the model consists of 5 parts:

- Location network with Spatial Transformer Layer (optional).
- Light-weight convolutional neural network (backbone).
- Per-position character classification head.
- Character probabilities for further sequence decoding.
- Post-filtering procedure.

Regarding the procedure, the architecture LocNet was used to estimate the optimal transformation parameters. Then, the backbone network architecture takes a raw RGB image as input and calculates spatially distributed characteristics. The CTC loss method is applied for misaligned input and output sequences. On the other hand, to improve performance, the predecoder feature map was expanded.For the post filtering, a task-oriented language model was used. Finally, all the experiments were done with TensorFlow with the Adam optimizer with a batch size of 32, a learning rate of 0.001 and a gradient noise of 0.001 as in [1].

IV. Proposed Experiments

We performed four experiments:

- Experiment 1 (Exp. 1) consists in running the original implementation [2] of the paper's model [1], using the original chinese plates dataset. We will use the values: batches = 100, epochs = 10.
- Experiment 2 (Exp. 2) consists in running the original implementation [2] of the paper's model [1], using the original chinese plates dataset. We will use the values: batches = 100, epochs = 50.
- Experiment 3 (Exp. 3) consists in running the implementation of a turkish plates recognition neural network [3]. We will use the values: epochs = 50, steps = 312.
- Experiment 4 (Exp. 4) consists in running the implementation of a turkish plates recognition neural

network [3]. We will use the values: epochs = 10, steps = 50.

V. Model Evaluation

For experiments 1 and 2, we will evaluate the results according to the **train cost** obtained value, since we could not resolve to obtain the accuracy.

For experiment 3, we will evaluate the results according to the **accuracy's** value obtained.

VI. Actual Results

After executing the experiments, these are the final results:

	Exp. 1	**Exp. 2**	**Exp. 3**	**Exp. 4**
Initial Train Cost	26.585	24.148	2.5166	3.5184
Final Train Cost	2.242	0	1.0405	1.8273
Accuracy	x	x	0.8087	0.6259

Table 1: Results obtained in the experiments with a Chinese plates dataset [2] and a synthetic Turkish plates dataset [4].

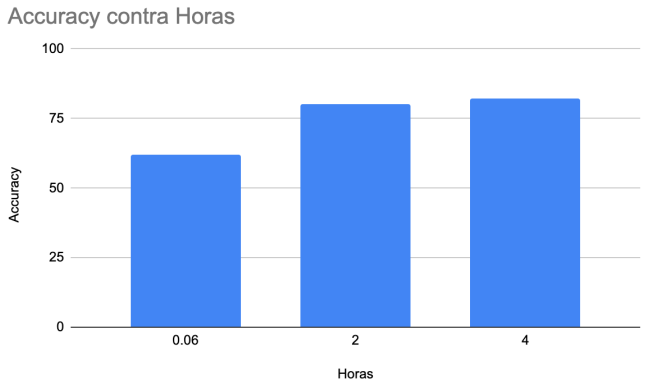

Fig. 1: Accuracy as a function of training time in Exp. 3, Exp. 4 and original implementation of [3].

VII. Conclusions

- At some point in experiment 2, the train cost became 0 before the execution ended, so we think it is not necessary to train the model that much if it won't drop better results (it can't, the minimum train cost is always 0).
- In experiment 3, the accuracy improves very slowly at the end, so we think it is not necessary to train the model that much, because it takes a lot of time and resources for such an insignificant improvement.

VIII. References

[1] Sergey Zherzdev, Alexey Gruzdev, 27 Jun 2018, LPRNET: License Plate Recognition via Deep Neural Network, https://arxiv.org/pdf/1806.10447.

[2] Shiyang Zhang, 03 Jan 2019, Plate_Recognition-LPRnet, https://github.com/lyl8213/Plate_Recognition-LPRnet.

[3] David Johnson, 21 Jul 2020, Plate Reader, https://www.kaggle.com/davidjj999/plate-reader/notebook.

[4] Tolga Üstünkök, 03 Sep 2019, Synthetic Turkish License Plates, https://www.kaggle.com/tustunkok/synthetic-turkish-license-plates

IX. Contribution

* All authors have contributed equally.