License plate detection and recognition system

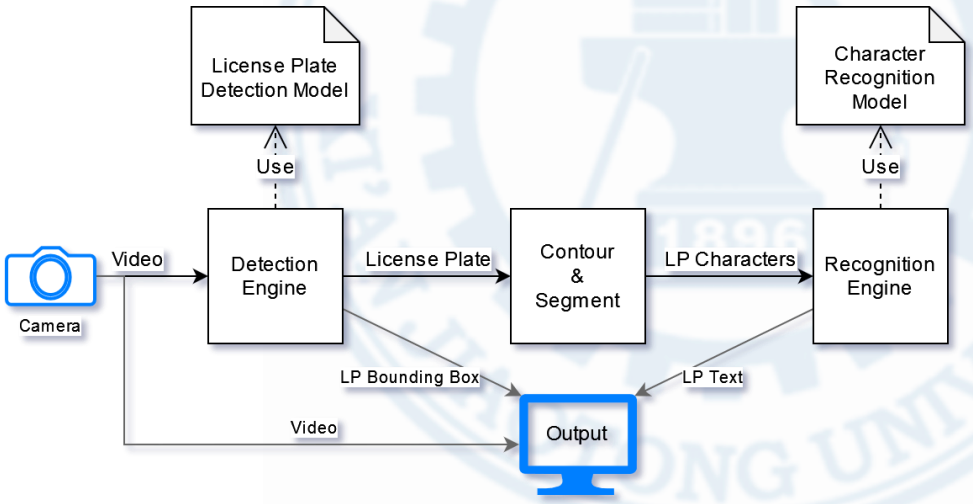
Damian Filo, 2022

1 Our Goals

1. Create a practical solution
2. Experiment with various approaches
3. Compare model architectures
4. Gain experience

30s
Our primary goal is to create a complete system with real-time detection and recognition. At the end of the presentation I will show you a demo of this function.
Also as is required for this class we will experiment with various architectures and approaches.
Then we can compare the architectures and arrive at a limited conclusion.
And as these goals are important, we cannot forget about the goal of gaining experience.

2 Project Outline



1m
We designed our project, so it is able to ingest real-time camera footage then detect and recognize license plates in that footage. Of course this won't be a perfect process, but we can get at least partial data most of the time and if the models were tuned for a specific deployment location, the detection and recognition accuracy could easily near 100% mark.
So let's break it down.
Camera footage flows to the Detection engine, which uses trained license plate detection model, it essentially looks for license plates in the footage and marks the region of the footage which is most likely a license plate.
This region containing license plate is processed, so it will be easier to do a character recognition on it. Additionally each detected character of the license plate is segmented out and each character can be then processed separately.
Recognition engine accepts these processed segments and tries to recognize each character. This Engine uses separately trained purpose-built character recognition model.
The data is then collected and written out as results.
In this presentation we will go through each part in more detail.

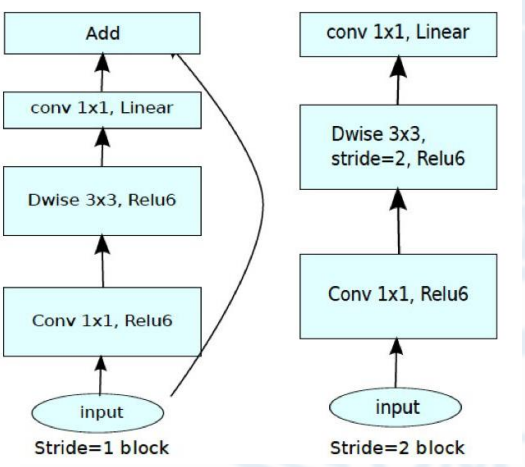
3 Detection Module – Characteristics

- Pre-trained detection model
  - Easy comparison
  - Speed essential
- Simple LP dataset
- SSD MobileNet v2

Model name	Speed (ms)	COCO mAP	Outputs
CenterNet HourGlass104 512x512	70	41.9	Boxes
CenterNet Resnet50 V1 FPN 512x512	27	31.2	Boxes
CenterNet Resnet50 V2 512x512	27	29.5	Boxes
EfficientDet D0 512x512	39	33.6	Boxes
EfficientDet D1 640x640	54	38.4	Boxes
SSD MobileNet v2 320x320	19	20.2	Boxes
SSD MobileNet V1 FPN 640x640	48	29.1	Boxes
SSD MobileNet V2 FPNLite 320x320	22	22.2	Boxes
SSD MobileNet V2 FPNLite 640x640	39	28.2	Boxes
Faster R-CNN ResNet50 V1 640x640	53	29.3	Boxes
Faster R-CNN ResNet101 V1 640x640	55	31.8	Boxes
Faster R-CNN ResNet152 V1 640x640	64	32.4	Boxes
Faster R-CNN Inception ResNet V2 640x640	206	37.7	Boxes

Let's continue with detection module.
The complexity for license plate detection from raw footage is quite high and our experiments with custom architectures didn't go well, as our team doesn't have as much experience to arrive to a meaningful conclusion that way. That's the reason why for quick development and respectable results with detection, we opted for the pre-trained model training approach. This offers us flexibility of choice from many professionally trained models with easily comparable characteristics. Additionally we don't even need to use large license plate datasets to get a really good performance out of our model. This is even more pronounced with our team's situation that we don't have an access to a fast compute GPU.
For our purposes the detection speed is essential as we would like to do a real-time recognition.
Our choice is a quite simple, a well-known model MobileNet v2 Single Shot Detector, which offers us fast training and detection as well.

3 Detection Module – MobileNet v2 SSD



The architecture of this model is really simple and that's the reason for its great speed, but it also doesn't lag much behind in accuracy metrics as well. We were able to train the model further with a simple dataset consisting of 400 cars.

3 Detection Module – Results – Success

Even with less pronounced, blending with background license plates, we can still observe a good prediction.

3 Detection Module – Results – Success

Even if the colors are a bit different, we can also get a good detection.

3 Detection Module – Results – Success

Also the accuracy with tilted license plates is quite good.

3 Detection Module – Results – Failure

It doesn’t perform well if the license plate is far away.

3 Detection Module – Results – Failure

Also there is a problem with license plates that are quite unique as you can see.

4 Recognition Module – Characteristics

- Well-known problem
- Custom architecture – training from scratch
- Problem with normalization
- Pre-processing

Optical character recognition is a well known machine learning use-case and also runs well on simpler architecture in comparison to license plate detection. Our approach is different for this problem, we decided to train our model from scratch on customized architecture and we could see really good results.
During our experimentation we were kind of stuck for a while on a particular problem, which seems to be a bug in Keras framework. The problem was with BatchNormalization layer, which caused that even with good training and testing accuracy we were getting bad results when using it with other data. We resolved this problem with removing these layers and the model still is able to achieve good performance.
In addition we employ Contouring and Segmentation techniques to prepare the license plates for easier character recognition.

4 Recognition Module – Data Preparation

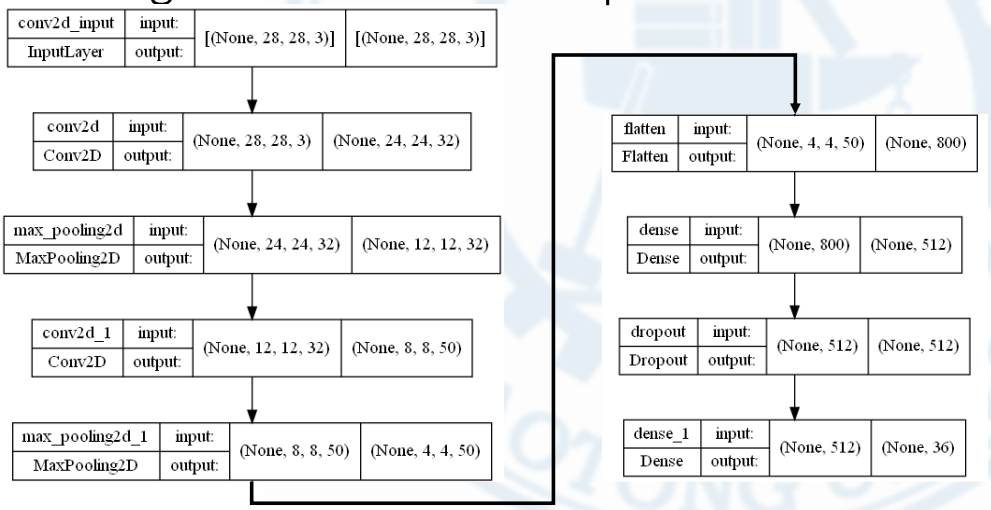
Crop

Contour

Segment

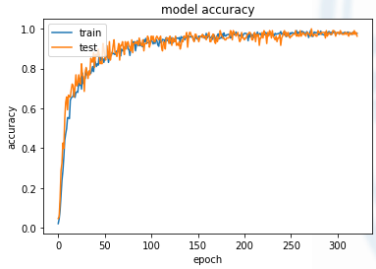
The pre-processing module consists of these steps and it is kind of required for the models we trained.

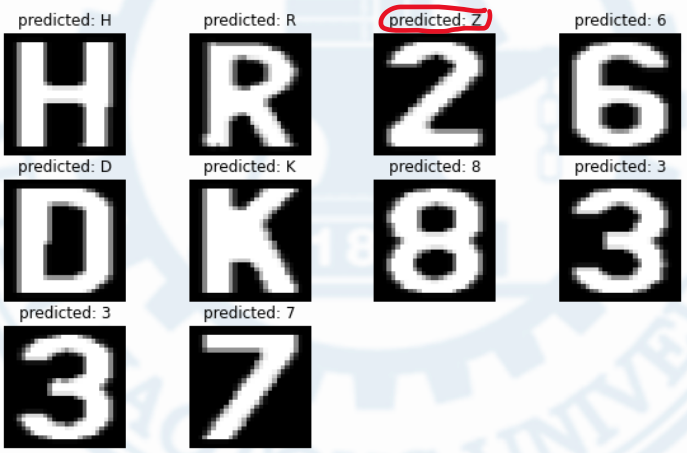
4 Recognition Module – Simple Architecture



This architecture represents a simple example. And consists of two blocks of convolution and pooling with fully connected network at the end. We also included a dropout layer, which helps the model accuracy in the real-world.

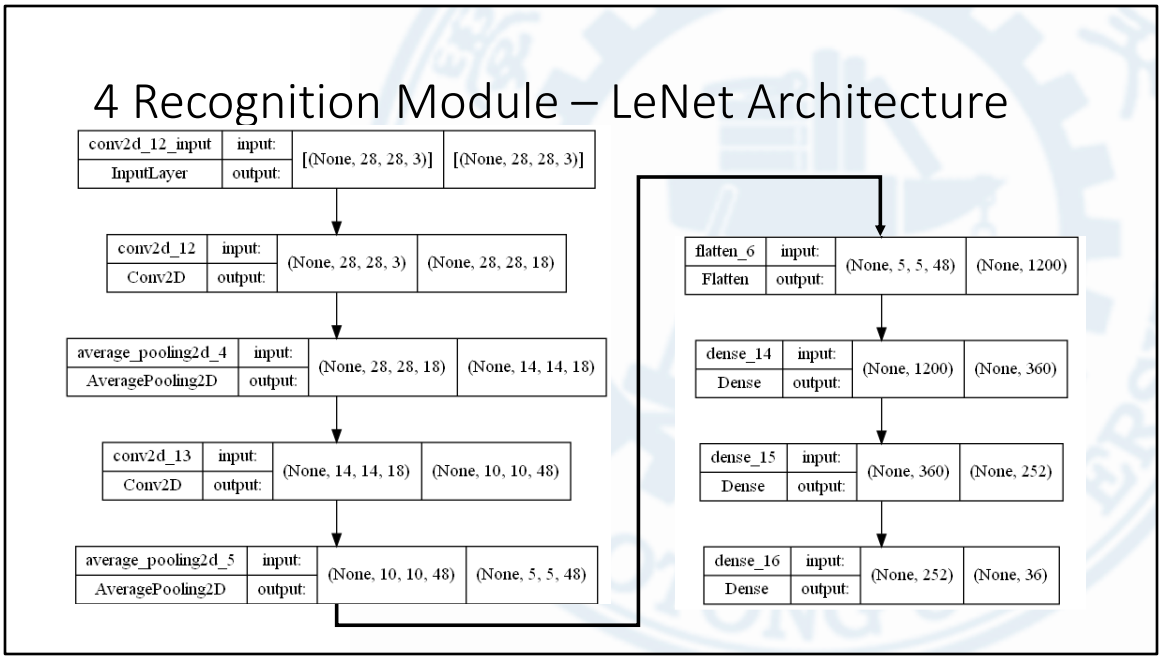
4 Recognition Module – Simple Architecture





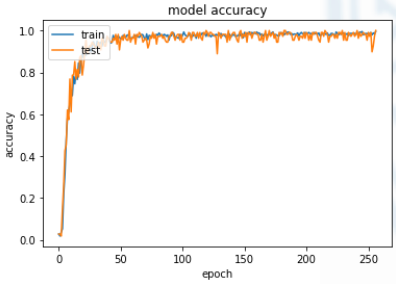
As we can see, the model was trained properly without overfitting, and the accuracy on our example is quite good. Everything is predicted correctly except the letter z which should have been a number 2.

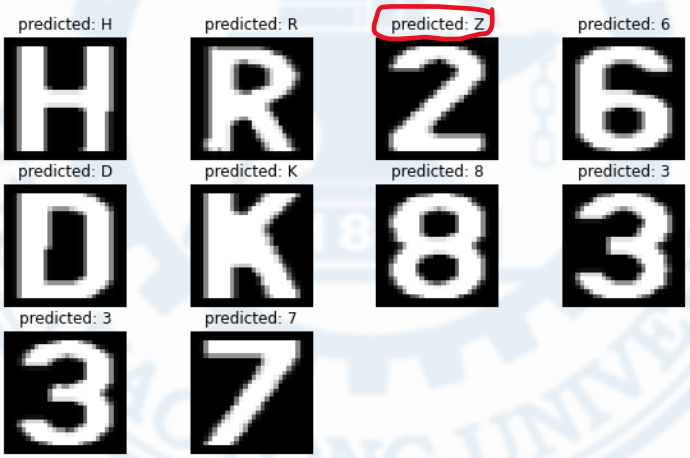
4 Recognition Module – LeNet Architecture



This architecture is inspired by LeNet and has similar composition to the previous one, but here we use sigmoid activation functions and our convolution layers have add borders around the samples.

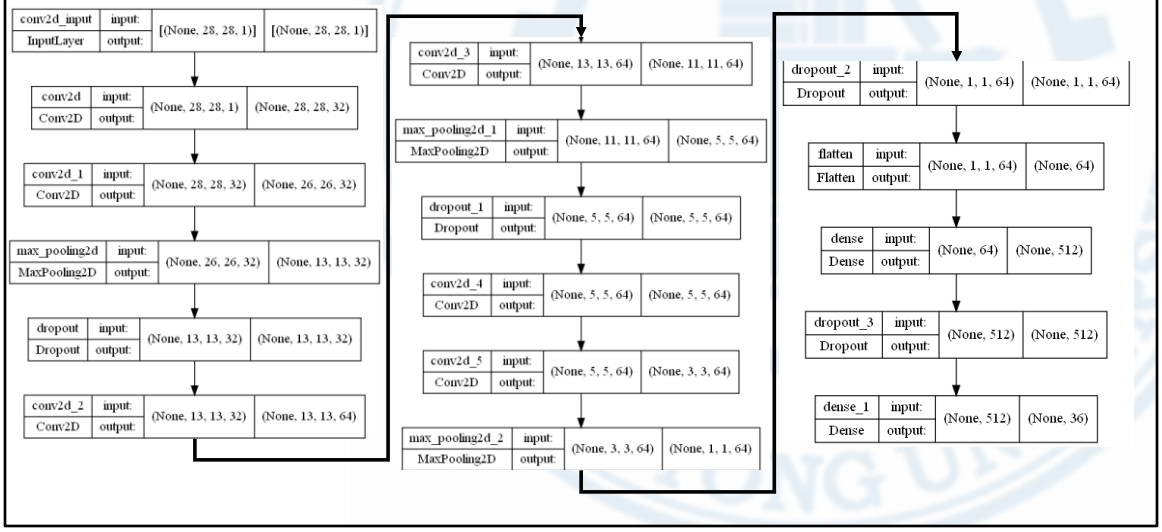
4 Recognition Module – LeNet Architecture





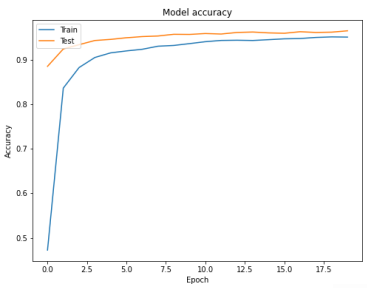
The learning progress was similar and the prediction on this example was the same as the previous architecture.

4 Recognition Module – Complex Architecture



For comparison purposes we decided to try also a more complex architecture, to see how it compares to the simpler ones.

4 Recognition Module – Complex Architecture



Prediction: RJ33CA395L

As we can see, the training progress is quite similar and with this more difficult example we can see that it still does a good job.

5 Real-Time Demo

Now is the time for a real-time demo.

6 Conclusion

- Practical implementation
- Use-case for each
- Difficult to debug
- Inventing wheel

1m
As you could see, we were successful in executing the practical implementation of our project. And as we have learnt through working on this project, every model architecture is good for something else. Our requirement limited the scope, but still we have much choice in selecting or designing our own architecture.
The main problem for beginners learning to use the Machine Learning tools is the difficulty to debug the problem if you don't have enough knowledge about the topic. We also experienced such difficulty but I think we as a team made a meaningful step forward.
As in real-world, in Machine Learning especially is often counter-productive to invent the wheel again. If there are resources made by professionals for our purpose available, we should use or get inspiration from them. There is no need to try develop something new, if we don't want to be researchers in the area.

Thank you for your attention!

Sources:

1) https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md
2) https://my.oschina.net/u/4067628/blog/3234607
3) https://github.com/nicknochnack