Case Study: Deep learning for Vehicle License Plates detection

Francesco Pugliese, PhD

Table of contents

What is ANPR?

- Automatic number-plate recognition (ANPR) is the task of finding and recognizing license plates in images. It is commonly broken into **four** subtasks that form a **sequential pipeline**: vehicle detection, license plate detection, character segmentation and character recognition. For simplicity, the combination of the last two subtasks is referred as Optical Character Recognition(OCR).

- ANPR is used by police forces around the world for **law enforcement** purposes, including to check if a vehicle is registered or licensed.

- It is also used for electronic toll collection on pay-per-use roads and as a method of cataloguing the movements of traffic, for example by highways agencies.

- Systems commonly use **infrared** lighting to allow the camera to take the picture at any time of day or night. ANPR technology must take into account plate **variations** from place to place.

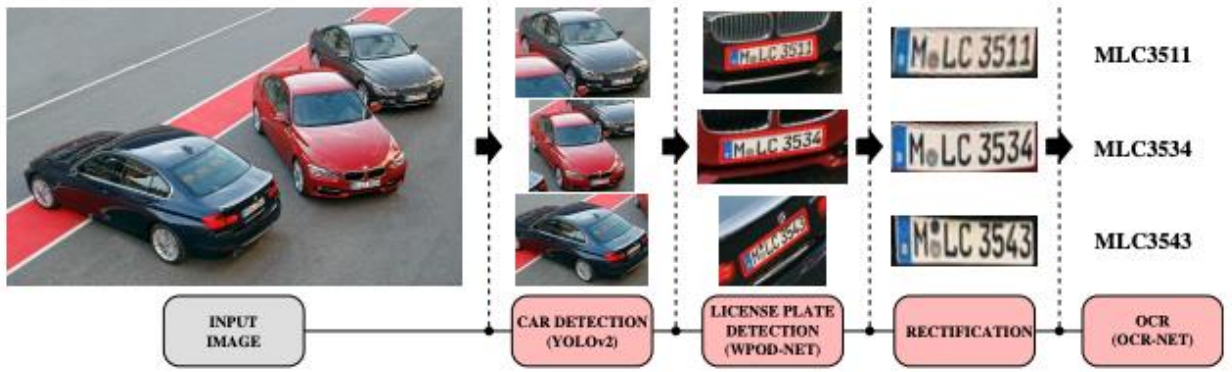
- 20 years ago, building a robust ANPR (ANPR) system could be considered as a **Master or PhD** level in the field of Computer Vision. Nowadays, the rise of Machine Learning (ML), especially with the formation of Neural Network and open-source ML libraries (Keras, Pytorch, MXNet, etc), has given us significant accessibility to develop and deploy these state-of-the-art programs.

ANPR workflow

- Given an input image, the first module detects vehicles in the scene. Within each detection region, the proposed Warped Planar Object Detection Network (**WPOD-NET**) searches for LPs and regresses one affine transformation per detection, allowing a rectification of the LP area to a rectangle resembling a frontal view. These positive and rectified detections are fed to an OCR Network for final character recognition.

- In summary, ANPR involves following steps –
  - Extracting License plates from the vehicle Images
  - Plate character segmentation from the extracted license plate
  - Train a Neural Network to predict segmented characters obtained from segmentation

ANPR workflow



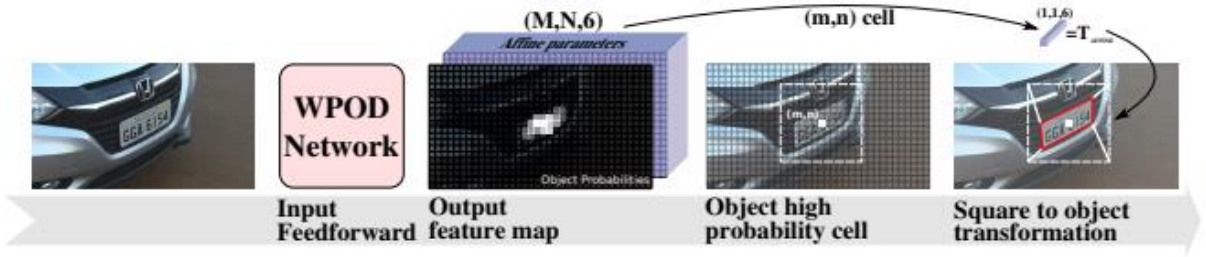
Workflow

Extracting License plate

Output for License plate extraction for a given vehicle image

Extracting License plate using WPOD-Net

- Initially, the network is fed by the resized output of the vehicle detection module.
- The feedforwarding results in an 8-channel feature map that encodes object/non-object probabilities and affine transformation parameters.
- To extract the warped LP, an imaginary square of fixed size is considered around the center of a cell (m, n).
- If the object probability for this cell is above a given detection threshold, part of the regressed parameters is used to build an affine matrix that transforms the fictional square into an LP region. Thus, we can easily unwarp the LP into a horizontally and vertically aligned object.

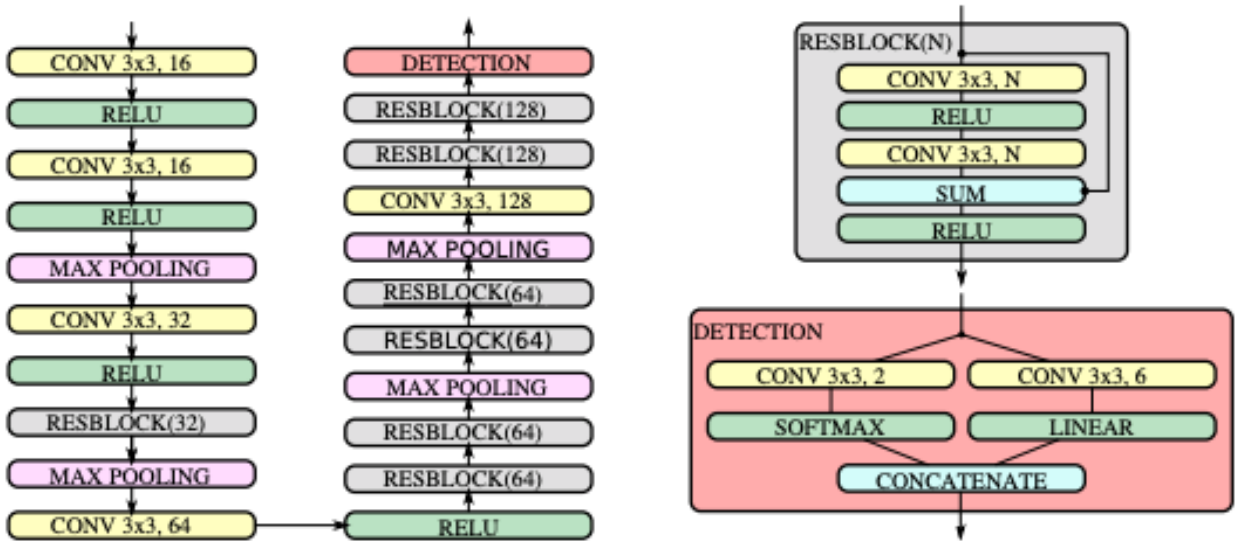


License plate detection process using WPOD-NET

WPOD-Net architecture

- The proposed architecture has a total of 21 convolutional layers, where 14 are inside residual blocks.
- The size of all convolutional filters is fixed in 3×3.
- ReLU activations are used throughout the entire network, except in the detection block.
- There are 4 max pooling layers of size 2×2 and stride 2 that reduces the input dimensionality by a factor of 16.
- Finally, the detection block has two parallel convolutional layers:
  (i) one for inferring the probability, activated by a softmax function, and
  (ii) another for regressing the affine parameters, without activation (or, equivalently, using the identity $F(x) = x$ as the activation function).

WPOD-Net architecture



WPOD-Net architecture

WPOD-Net notes

- Wpod-Net is a powerful tool since it was able to decently detect and extract license plates from **10** different countries. Nevertheless, this does not mean Wpod-Net is 100% accurate.

- A highly likely scenario is that the model can return some license plate look-alike objects (billboards, traffic signs, etc) along with the correct ones.

Plate character segmentation

- Segment key characters from License Plate (from the previous step) using Python and OpenCV.
- Each of those segmented character later will be passed through a Convolutional Neural Network (CNN) model and returned as digital letter.
- To begin with, we need to apply several processing techniques to reduce noise and emphasize key features of license characters.

Bounding boxes for characters in the extracted license plates

Image processing methods

The image processing methods we shall implement on our image are as follows:

- **Convert to 255 scale:** Extracted license image from Wpod-Net is interpreted as **0–1** scale, thus we need to convert it to 8-bit scale as standard image.

- **Convert to grayscale:** Color plays an **negligible** role to understand the license plate, thus we can remove it to optimize **computational** power.

- **Blur image:** Blur technique is performed to **remove** noise and irrelevant information.

- **Image thresholding:** We set a threshold so that any **smaller** pixel value than it would be converted to 255 and vice versa. This type of thresholding is called ***inverse binary thresholding*** .

- **Dilation:** This is a technique to increase the **white region** of the image. By implementing *dilation*, we want to enhance the white **contour** of each character.

Image processing methods

plate_image

gray

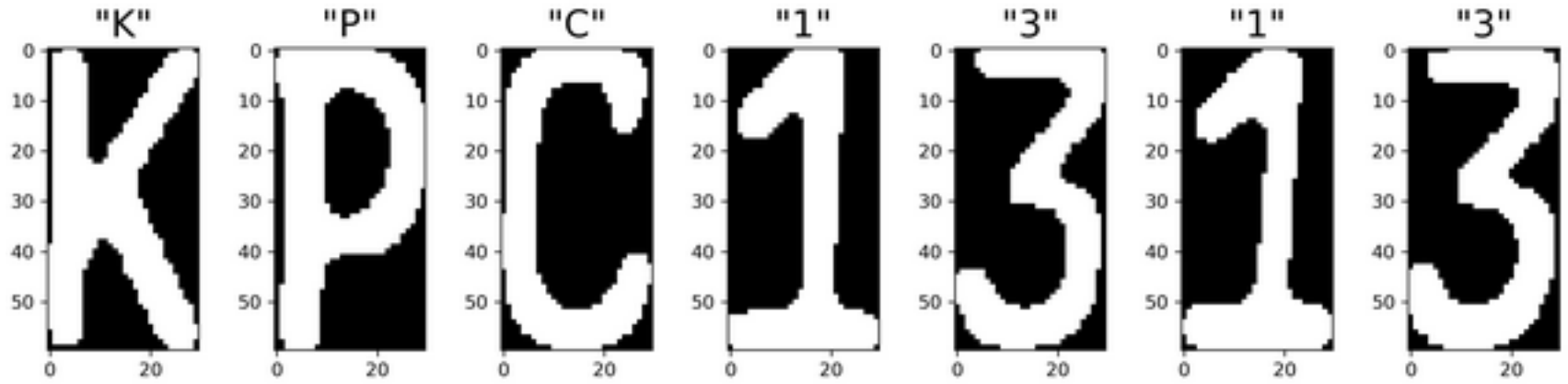
blur

binary

dilation

Image processing methods

Predict segmented characters



Classification of characters segmented using Deep learning

Predict segmented characters

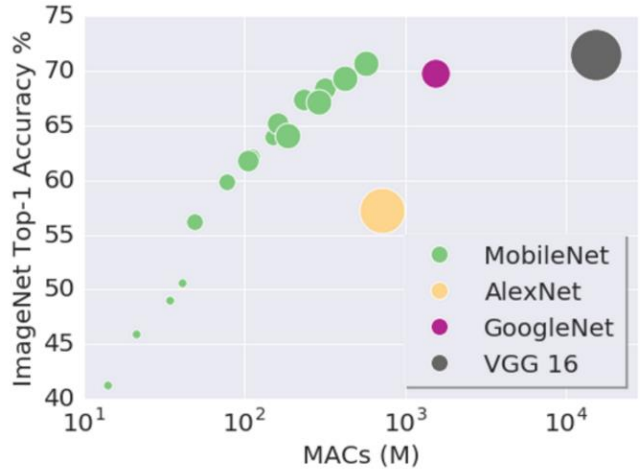
- To do this we need to train a Neural Network model which is capable of converting input images to digital letters.

- There are a wide range of prominent Neural Network architectures which you can select to train your model (e.g. ResNet, VGG, DenseNet, Inception, etc.), each of them has its own advantages and disadvantages .

- For this project, we would use MobileNets — a light weight deep learning network with noticeably good accuracy (source) as base model architecture

MobileNet

- For mobile and embedded vision applications.
- MobileNets are based on a streamlined architecture that uses depthwise separable convolutions to build light weight deep neural networks.
- Effective across a wide range of applications and use cases including object detection, finegrain classification, face attributes and large scale geo-localization.
- It is also very low maintenance thus performing quite well with high speed
- The speed and power consumption of the network is proportional to the number of MACs (Multiply-Accumulates) which is a measure of the number of fused Multiplication and Addition operations.

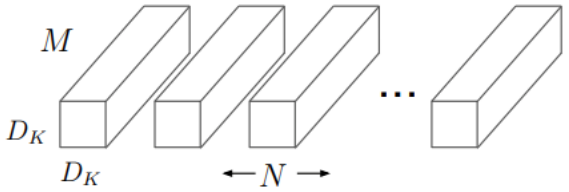


MobileNet Architecture

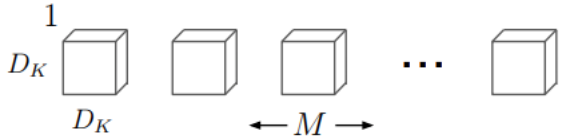
- It uses depthwise separable convolutions which basically means it performs a single convolution on each colour channel rather than combining all three and flattening it. This has the effect of filtering the input channels. Or as the authors of the paper explain clearly: " For MobileNets the depthwise convolution applies a single filter to each input channel.

- The pointwise convolution then applies a 1×1 convolution to combine the outputs the depthwise convolution. A standard convolution both filters and combines inputs into a new set of outputs in one step.

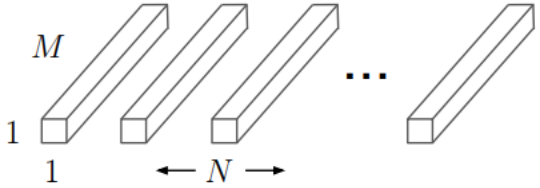
- The depthwise separable convolution splits this into two layers, a separate layer for filtering and a separate layer for combining. This factorization has the effect of drastically reducing computation and model size.



Standard convolutional filters



Depthwise convolutional filters



1*1 Convolutional filters called pointwise convolution in the Context of Depth wise Separable convolution

MobileNet architecture

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Commercial ANPR Applications

- Sighthound
- OpenALPR - is an official NVIDIA partner in the Metropolis platform2
- Amazon Rekognition - a general-purpose AI engine including a text detection and recognition module that can be used for LP recognition, as informed by the company.

References:

- ✓ https://en.wikipedia.org/wiki/Automatic_number-plate_recognition
- ✓ https://openaccess.thecvf.com/content_ECCV_2018/papers/Sergio_Silva_License_Plate_Detection_ECCV_2018_paper.pdf - Paper on License Plate Detection and Recognition
- ✓ https://arxiv.org/pdf/1704.04861.pdf - Paper about MobileNets
- ✓ https://towardsdatascience.com/transfer-learning-using-mobilenet-and-keras-c75daf7ff299

Francesco **Pugliese**