License Plate Detection and Text Recognition

Feb 17 – Jun 4 2020

Presented by:

Lorenzo Lamberti
GreenWaves Technologies

Project Overview

GREENWAVES TECHNOLOGIES

Project Goal:

Application of Optical Character Recognition (OCR) techniques to **implement license plate recognition on GAP.**

extraction

京QC8152

Project Flow:

1. **Study of related works**:

    find a suitable algorithm for embedded devices

2. **Implementation** of the algorithm (python code)

3. **Evaluation of performances:**

    accuracy and reliability of model's predictions

4. **Optimization** of the model to satisfy GAP's constrains

Introduction to OCR

Optical Character Recognition

Goal: recognize text in images and transcribe it .

OCR categories:
- **In the wild** (unpredictable scenario)
- **controlled environment** (fixed setup)

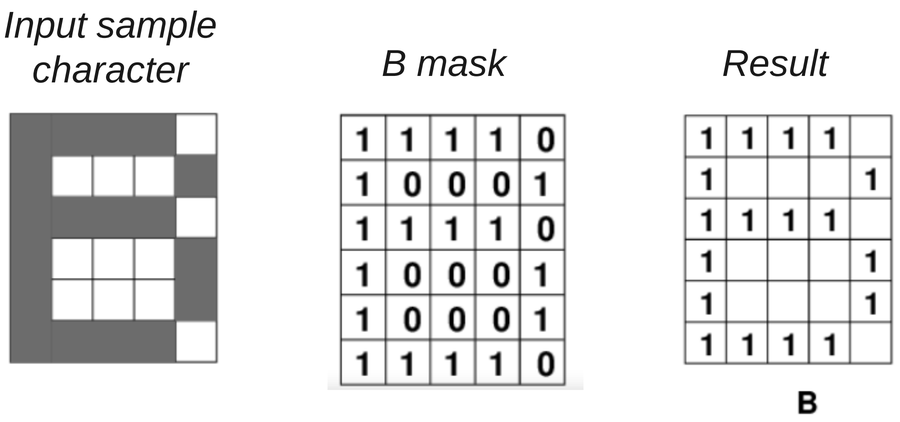
License plate recognition is a sub-category of OCR in the wild

OCR in the wild is the hardest task and not completely solved

Challenges OCR in the wild

- **Artifacts**: light variations and reflections.
- **Weather conditions:** snow and rain.
- **Text is sparse**: text can be randomly located in the image.
- **Font size not constant:** depends on the object distance.
- **Variability of fonts** (including special characters e.g. logograms for China)

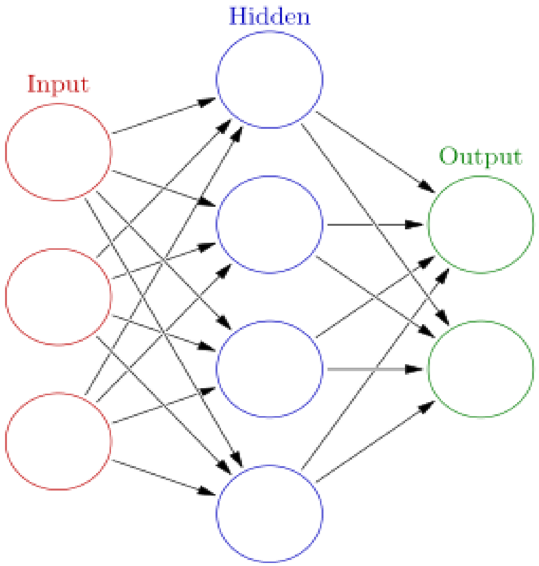
OCR Strategies



Traditional approaches:

Classic computer vision techniques (Tesseract)
(preprocessing + template/pattern matching)

- Based on a-priori assumptions
- Poor generalization on new scenarios: bad results in the wild



New approaches:

Based on **Deep learning** (Convolutional Neural Networks)

- Algorithm automatically learns features from given istances
- Good generalization if a large dataset is provided

State of the art

License plate Recognition

Standard Deep Learning approach to License Plate OCR:

The problem is tackled in **2 steps**:
(They are treated separately)

1. **Text Detection:**
   Localization of the full words

2. **Text Recognition:**
   Reconstruction of the word character by character

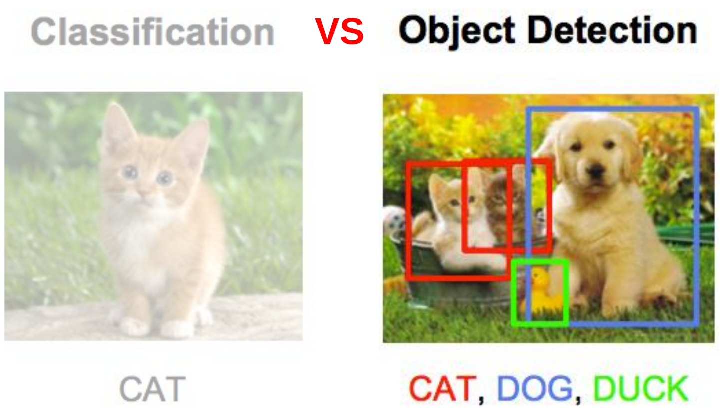
<Sichuan>A57237

Segmentation-free methods

A B C 1 2 3 4

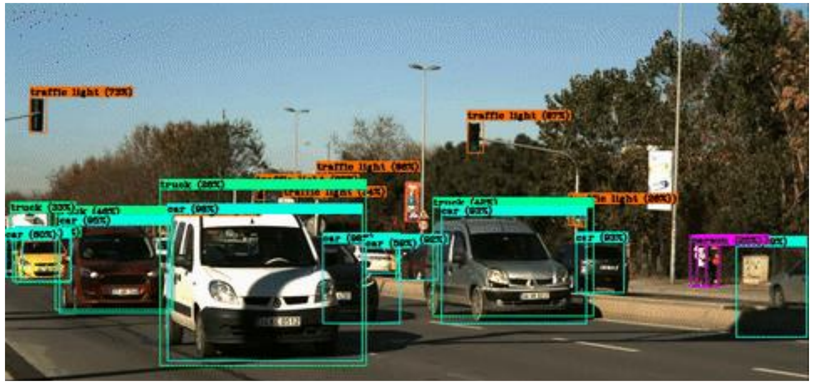
Segmentation methods

Step 1: License Plate Detection

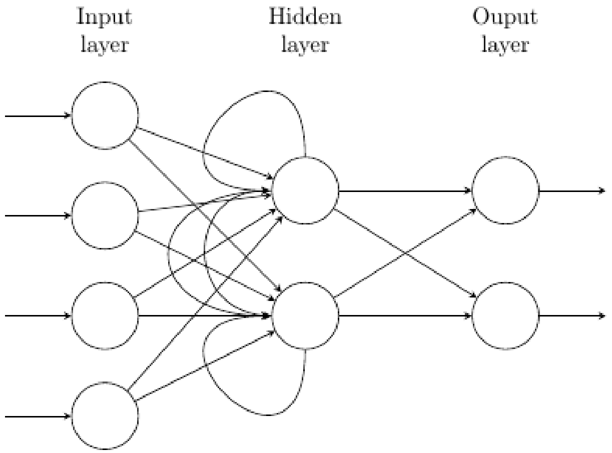


Object Detection algorithms:
(based on CNNs, convolutional neural networks)

- Single Shot Detector (**SSD**)
- You Only Look Once (YOLO)
- Region Based CNNs (R-CNN)



Step 2: License Plate Recognition

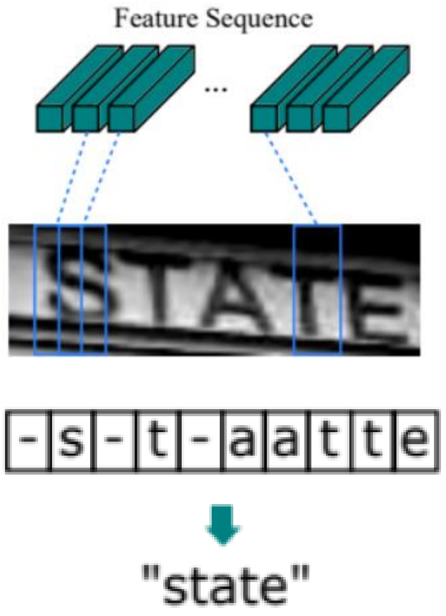


Recurrent Neural Networks (RNNs):

- LSTM
- BiLSTM
- CRNN

It is a type of network in which inside loops are allowed: **memory effects**.

It models temporal/spatial sequences.



It's the most common approach for text recognition
but Gap doesn't support RNNs at the moment

Step 2: License Plate Recognition

Convolutional Neural Networks Only:

- **Character segmentation with object detectors**:
  Too redundant.
  Need training data with segmented characters annotation: very rare

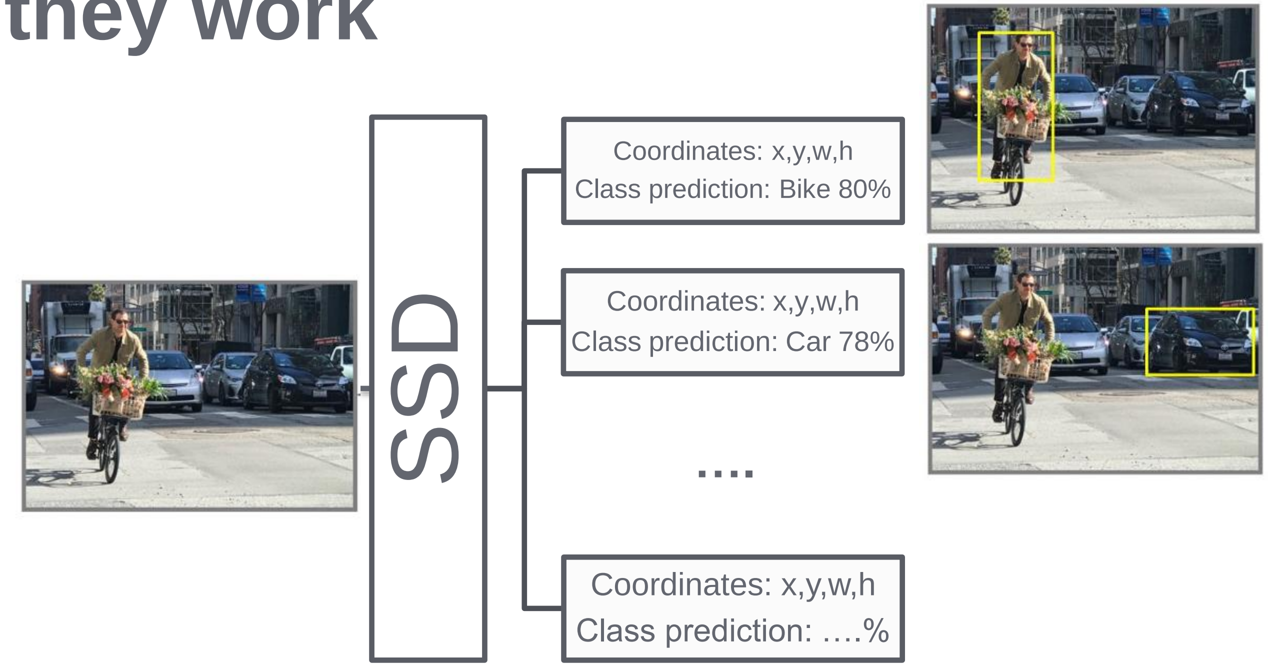
- **Holistic CNNs:**
  8 branches of the CNN search for different positions in the image.
  One branch specializes for detecting different characters in the plate.
  Code not publicly available

- **LPRNet:**
  End-to-end method without preliminary character segmentation. Straight from pixels to char predictions

Approaches selected: how they work

1) SSD: Single-Shot detector

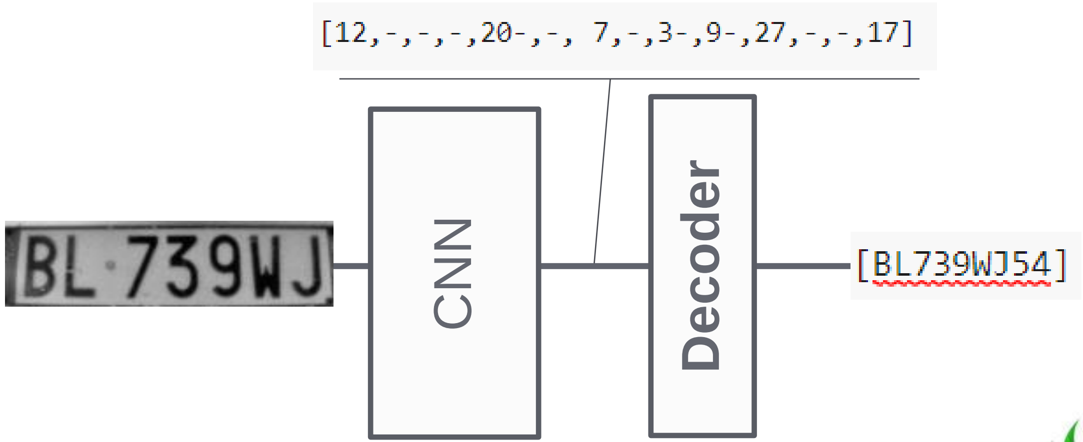
- A fully Convolutional Neural Network
- Single step: simultaneous bounding box regression and classification
- One bbox and class associated to each object



2) **LPRNet for character recognition:**

Convolutional Neural network + Decoder

1. CNN predicts probability of characters and blank spaces
2. Decoder: associates probabilities to the right characters and removes double predictions



Approaches selected: Why

1° Step: SSD (object detection)

Code available in official Tensorflow obj detection API
Lightweight models available (Mobilenet)
Flexible: easy to switch architecture & a lot of pretrained models to try.
Big dataset ready to use.

2° Step: LPRNet (character recognition)

Code available (Intel - tensoflow)
Etremely lightweitght: 1.7 Mb
Fast: inference time 3ms on a Nvidia GTX 1080
Promises an accuracy of over 95%
Only a CNN: doesn’t use RNNs (not supported by gap)

License Plate Detection

Character Recognition

<Henan>K0F755

GAP Constraints:

- **Tensorflow** framework: we must provide a .tflite model
- **Memory: 8Mb** (must fit in L3 memory)
- **Real-Time Execution: ~ 1 FPS**
- Good accuracy (of course)

General design flow for ML algorithms

1. **Dataset search:** a lot data to train the algorithm

2. **Model selection:** satisfy GAP's constrains

3. Optimization: **quantization aware training**

4. **Validation:** test performances of real data

   I acquired a little validation dataset (70 images) with GAP Himax camera

Validation dataset: GAP’s Himax camera

Low resolution grayscale images (QVGA: 320x240)

20 Italian license plates

50 Chinese license plates

Acquired at 5 distances (0.3m, 1m, 2m, 3m, 4m)

0.3 m → 1 m → 2 m → 3 m → 4 m

Evaluation Metrics

1° Step: License Plate Detection

mAP

AP
average
precision

$$AP = \frac{\text{Correct predictions}}{\text{Total n° License Plates}}$$

How many plates were correctly located

IoU:
Intersection
over union

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

How well the prediction is overlapping to the LP

2° Step: Text Recognition: $\textbf{Accuracy} = \frac{\text{Correct license plates}}{\text{Total n° License Plates}}$

A license plate is correct if all the characters are correctly recognized
One mistake = wrong recognition

1 B A 3 1 0 2
Correct!

1 B A 2 1 0 2
Wrong!

Design Flow on GAP

Step 1: License Plate Detection

GREENWAVES TECHNOLOGIES

SSD - Object Detection

Design flow of the license plate detector:

1. **Dataset search:** open source Google OpenImages

   7k high-res images

2. Find the right **architecture** (accuracy/memory tradeoff)

   Resnet: 270Mb (too big)
   MobilenetV1-V2: 22,7Mb ; 18,5Mb (still too big)
   SSDlite MobilenetV2: 13,3 Mb

3. Optimization: **quantization aware training**

   float32bit → int8bit (4x memory save)
   13,3 Mb → 3,2Mb

4. **Validation** on images acquired on gap

   Not a problem transfering learning from hi-res images to Himax QVGA

Open Images dataset

My validation dataset (Himax pictures)

Training Experiments

Bigger images = better accuracy:

- SSD 512x512 images: 45.4%
- SSD 320x240 images (QVGA): 42.0%

We are forced to use QVGA format (320x240).
But a camera with higher resolution improves accuracy.

Shrinking down the model's size:

Architectures

- SSD MobilenetV1 (22Mb) 39.6%
- SSD MobilenetV2 (18Mb) 42.0%
- SSDlite MobilenetV2 (13.3Mb) 39.8%

We want to fit in 8 MB

Failed approaches:

- Grayscale training: 38.3%
- Frozen backbone training: 17.3%

Validation images are grayscale.
Transform training images to grayscale too

Start from a pretrained model and just
finetune last layer of the network

Optimization: quantization aware training

- SSD SSDliteV2 (13.3Mb) 39.8%
- SSD SSDliteV2 **8bit quant** (3.2Mb) 38.9%

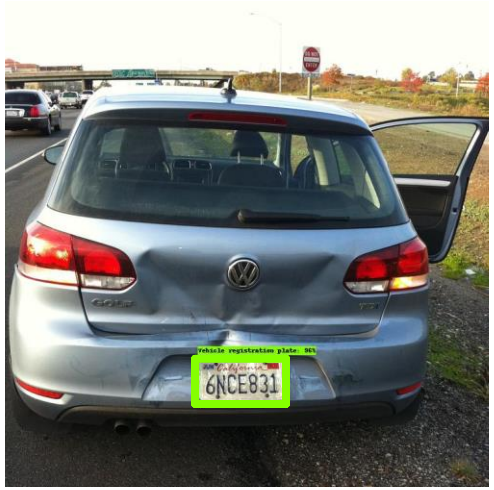
Final model

GREENWAVES TECHNOLOGIES

Final Detection Accuracy

Model's final mAP score: 38.9 %

- SSDlite MobilenetV2 architecture
- 3.2Mb
- 8 bit quantized
- Training on QVGA resolution images (320x240)



Result Comparison:

- Nice accuracy, same algorithm performs 22% on COCO.
- But here we are detecting a single class (easier problem)

Reference benchmark : standard COCO dataset

Algorithm	Architecture	Size	mAP on COCO
YOLOv2	Darknet 19	(60Mb)	21%
YOLOv3	Darknet 53	(190Mb)	33%
SSD	Mobilenet v1	(73Mb)	18%
SSD	Mobilenet v2	(138Mb)	22%
SSD	**SSDlitev2**	**(48Mb)**	**22%**
SSD	Resnet 50	(350Mb)	35%
SSD	Resnet 101	(610Mb)	40%

Detection distance

After 2 meters the characters are not readable anymore. But the algoritm still works.

0.3 m → 1 m → 2 m → 3 m → 4 m

Design Flow on GAP

Step 2: Character Recognition

LPRNet – License Plate Recognition Network

Design flow for character recognition:

1. Dataset search:

Synthetic Chinese License Plates (270k images)
CCPD: real Chinese License Plates (200k images)
Czech Low-Quality License Plates (185k images)

2. Architecture manipulation:

original size: 6.8Mb
replaced unsupported operations: 9.6Mb
Depth reduction fully connected layers : 4 Mb

3. Optimization: **quantization aware training**

float32bit → int8bit (4x memory save)
4 Mb → 1.0Mb

4. **Validation** on images acquired on gap

Himax QVGA grayscale images

Training Datasets for char recognition

Synthetic Chinese | CCPD Chinese | Czech dataset

GAP Validation image

Accuracy on benchmark datasets

Synthetic Chinese License Plates: 97.7 %

CCPD: real Chinese License Plates: 99.1%

Czech Low-Quality License Plates: 99.2%

Accuracy after quantization and depth reduction

Training and testing on Synthetic chinese license plates

Original network (6.8Mb):	98.1 %	optimization
Replace unsupported operations (9.6Mb):	97.5% ↓	
Depth reduction fully connected layers (4 Mb):	97.6% —	
Quantization aware training (1.0Mb):	97.7% —	

There is no accuracy drop while optimizing: brought down the model size from 9.6 to 1 Mb

Maximum Recognition Distance

QVGA (320x240) low quality images: from 2 meters the characters are barely readable.

The algorithm is reliable up to 1m of distance.

0.3 m	1 m	2 m	3 m
Accuracy: 100%	Accuracy: 100%	Accuracy: 36.4% Accur-1 : 54.5%	Accuracy: 0%

GREENWAVES TECHNOLOGIES

Good resistance to tilting and blur

- Tilting Effects

- Blurred images

Final Pipeline

Unification of the 2 steps

GREENWAVES TECHNOLOGIES

Final Unified Pipeline

Unification of the 2 steps:

1. SSD: License plate detection (3.2Mb)
2. LPRNet: Text Recognition (1Mb)

Final LP model size : 3.2 Mb + 1Mb = **4.2 Mb** **OK!**

Gap memory constrain: **8Mb**
So it can fit within L3 memory of GAP

Input Image

1 Step: SSD
for plate detection

2 Step: LPRNet
for text recognition

Final Pipeline Speed: does it run in real-time?

Inference estimation

			Worst case 1 MAC/Cycle		best case 8 MAC/Cycle
1° Step (SSD):	**540M** MACs	→	**3.6 sec at 150MHz**	→	**0.45 sec at 150MHz**
2° Step (LPRNet):	**172M** MACs	→	**1.8 sec at 150MHz**	→	**0.14 sec at 150MHz**
Steps Combined:	**712M** MACs	→	**5.4 sec** at 150MHz	→	**0.49 sec** at 150MHz

Real-Time applications: ~ 1 FPS

Our Speed:

0.49 sec < Inference Speed < 5.4sec
0.2 FPS < Inference Speed < 2 FPS

GREENWAVES TECHNOLOGIES

Conclusion

Conclusion

2 Steps approach for License Plate OCR:

1. License Plate Detection: SSD (Single Shot Detector)
2. License Plate Text Recognition: LPRNet

Works on Himax QVGA grayscale images:

- Detects License plates up to **4m** distance. → **mAP score** = 38.9%
- Recognize Text up to **1m** distance. → **Accuracy** = 97.7 - 99.2%

Final LP model size : 3.2 Mb + 1Mb = 4.2 Mb → Fits in the **L3 RAM memory** of gap: < 8Mb

Inference speed (expected on GAP): 0.49 sec < time < 5.4sec → Could run in **Real-Time**: ~ 1 FPS

Thank you!

Questions?

Thanks for helping:

- Manuele Rusci
- Marco Fariselli
- Francesco Paci
- Ahmad Bijar

GREENWAVES TECHNOLOGIES