Detection of empty parking spot

Deep learning project

Raphaël Chekroun

École normale supérieure - PSL University

Introduction

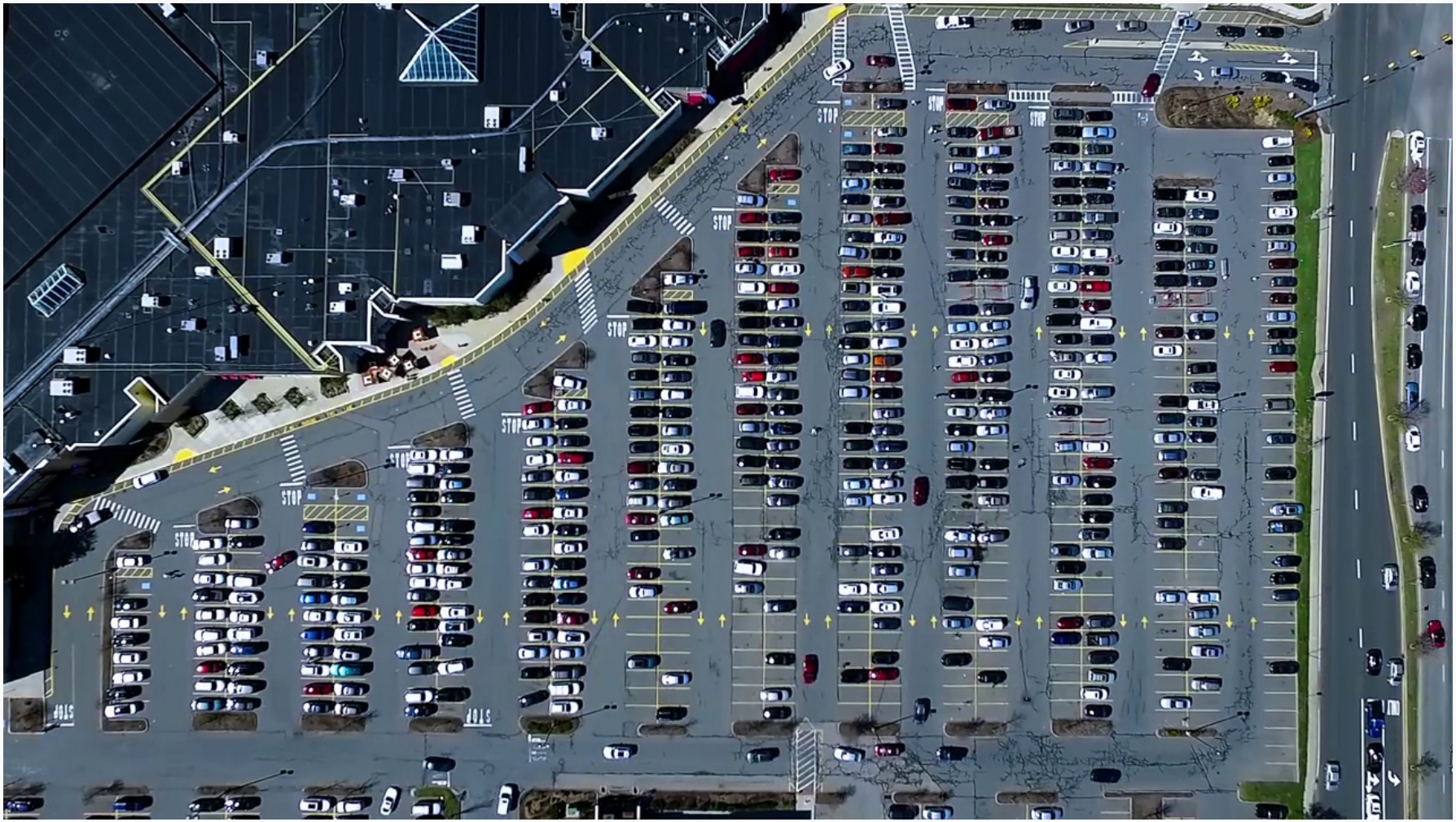
Introduction

$\rightarrow$ Goal: Use NN to predict on a video of a parking the number of available spots and their locations.

$\rightarrow$ This work is divided in two part:

1. Implement a NN able to recognize if a given parking spot is occupied or empty,
2. Detect the delimitations of parking lanes and parking spots from an image.

An overlook of the result



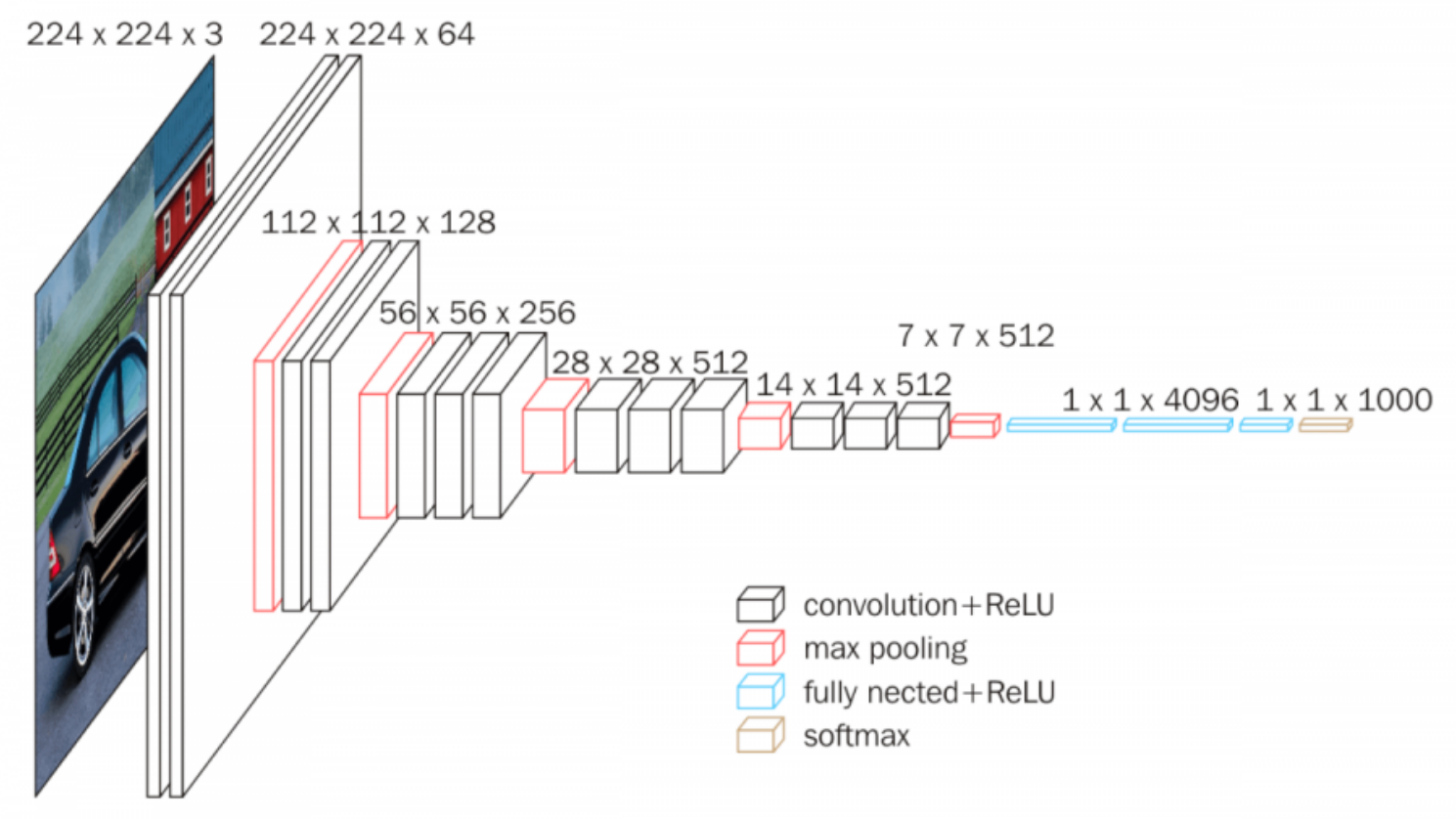
An overlook of the result

The design of the neural network

A choice of NN architecture

$\rightarrow$ I made some research to find that many of problems of this type were solved using a VGG16 neural network architecture, as it would be the more efficient.

$\rightarrow$ The VGG16 architecture:



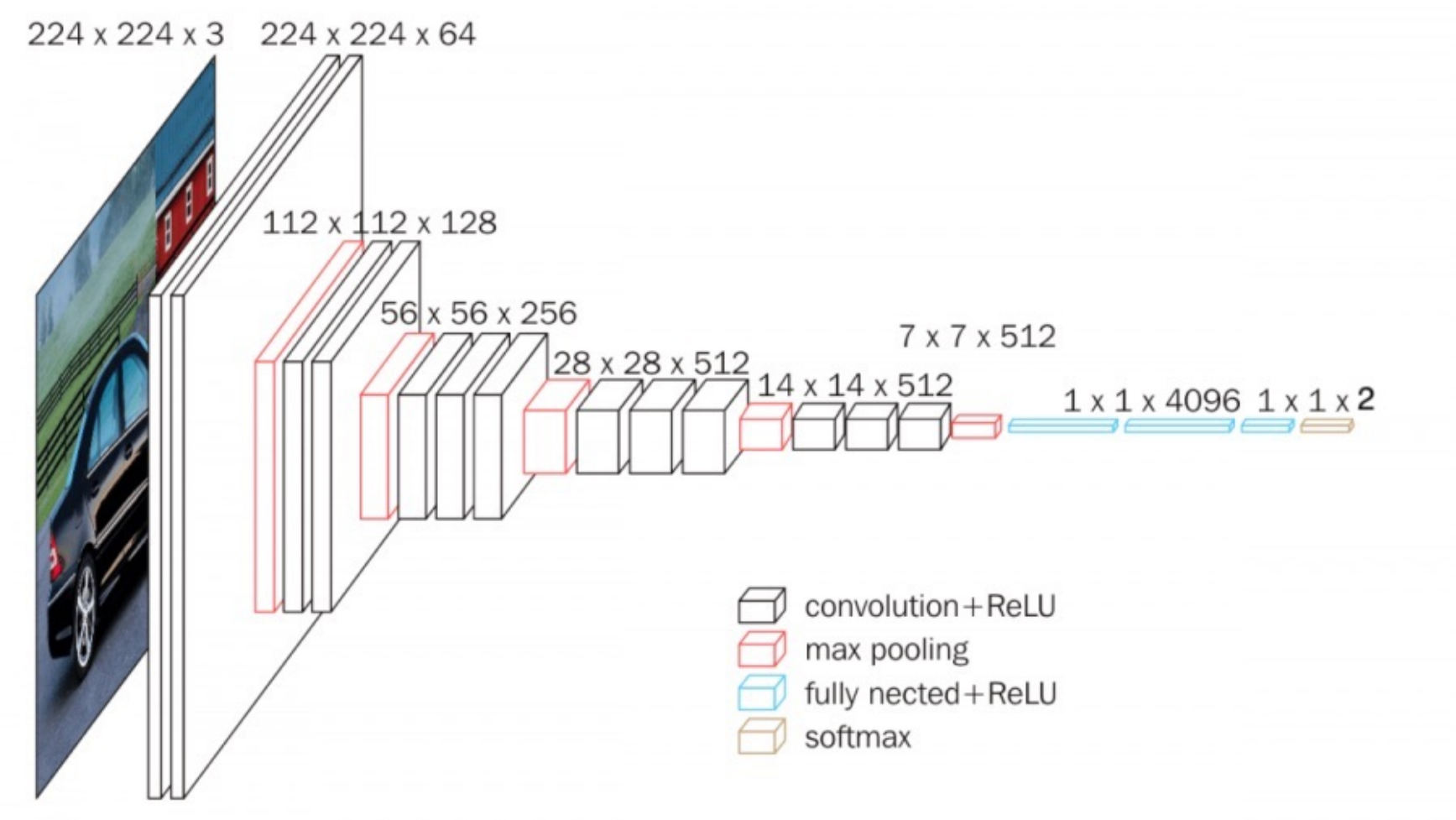
Transfer learning on a VGG16 architecture

→ I implemented everything with PyTorch, using a pre-trained VGG16.

→ As VGG16 is designed to work with 1000 different class I had to remove its last layer to replace it with a new layer with only 2 outputs: 'empty' or 'occupied'.

→ I trained the last layer for 50 epochs on Google Colab.

The final NN architecture



Information on the dataset

My dataset is composed of cropped screenshots of different photos, as I didn't find one adapted to my situation.

Information on the dataset

My dataset is composed of cropped screenshots of different photos, as I didn't find one adapted to my situation.

$\rightarrow$ Training set:

- 285 photos of car seen from the sky
- 96 photos of empty parking spot seen from the sky

Information on the dataset

My dataset is composed of cropped screenshots of different photos, as I didn't find one adapted to my situation.

$\rightarrow$ Training set:

- 285 photos of car seen from the sky
- 96 photos of empty parking spot seen from the sky

$\rightarrow$ Testing set:

- 126 photos of car seen from the sky
- 38 photos of empty parking spot seen from the sky

Efficiency of my NN

→ After training, I have a very high accuracy of 0.9878:

Critics on the model

$\rightarrow$ In reality it's weaker than that: this accuracy is only valide on the very little training set I have

Critics on the model

$\rightarrow$ In reality it's weaker than that: this accuracy is only valide on the very little training set I have

$\rightarrow$ I experimentally measured an accuracy of approximatly 0.86, with fake occupied and never any fake empty

Critics on the model

$\rightarrow$ In reality it's weaker than that: this accuracy is only valide on the very little training set I have

$\rightarrow$ I experimentally measured an accuracy of approximatly 0.86, with fake occupied and never any fake empty

$\rightarrow$ To fix this, I should find a bigger dataset of making mine bigger with other screenshots

Critics on the model

$\rightarrow$ In reality it's weaker than that: this accuracy is only valide on the very little training set I have

$\rightarrow$ I experimentally measured an accuracy of approximatly 0.86, with fake occupied and never any fake empty

$\rightarrow$ To fix this, I should find a bigger dataset of making mine bigger with other screenshots

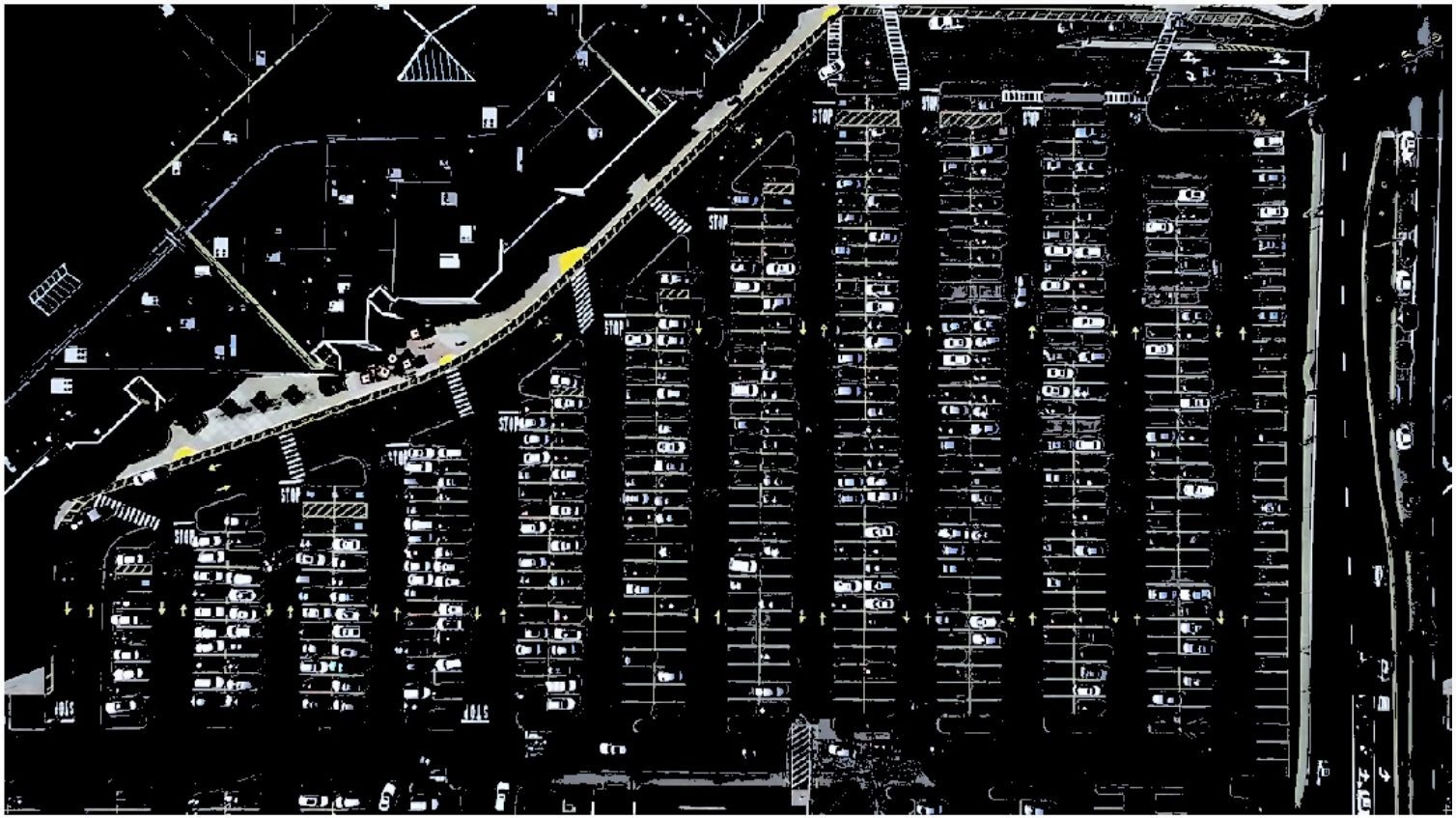
$\rightarrow$ Particularly, I should take screenshots of parking spots where the floor is cracked or stained

Detect the parking spots

Steps to detect the parking spot

$\rightarrow$ I started by myslef until I found a work on github doing the same thing. Our strategy was the same, and is very common:

1. Apply a yellow and white mask to eliminate all useless information:



Steps to detect the parking spot

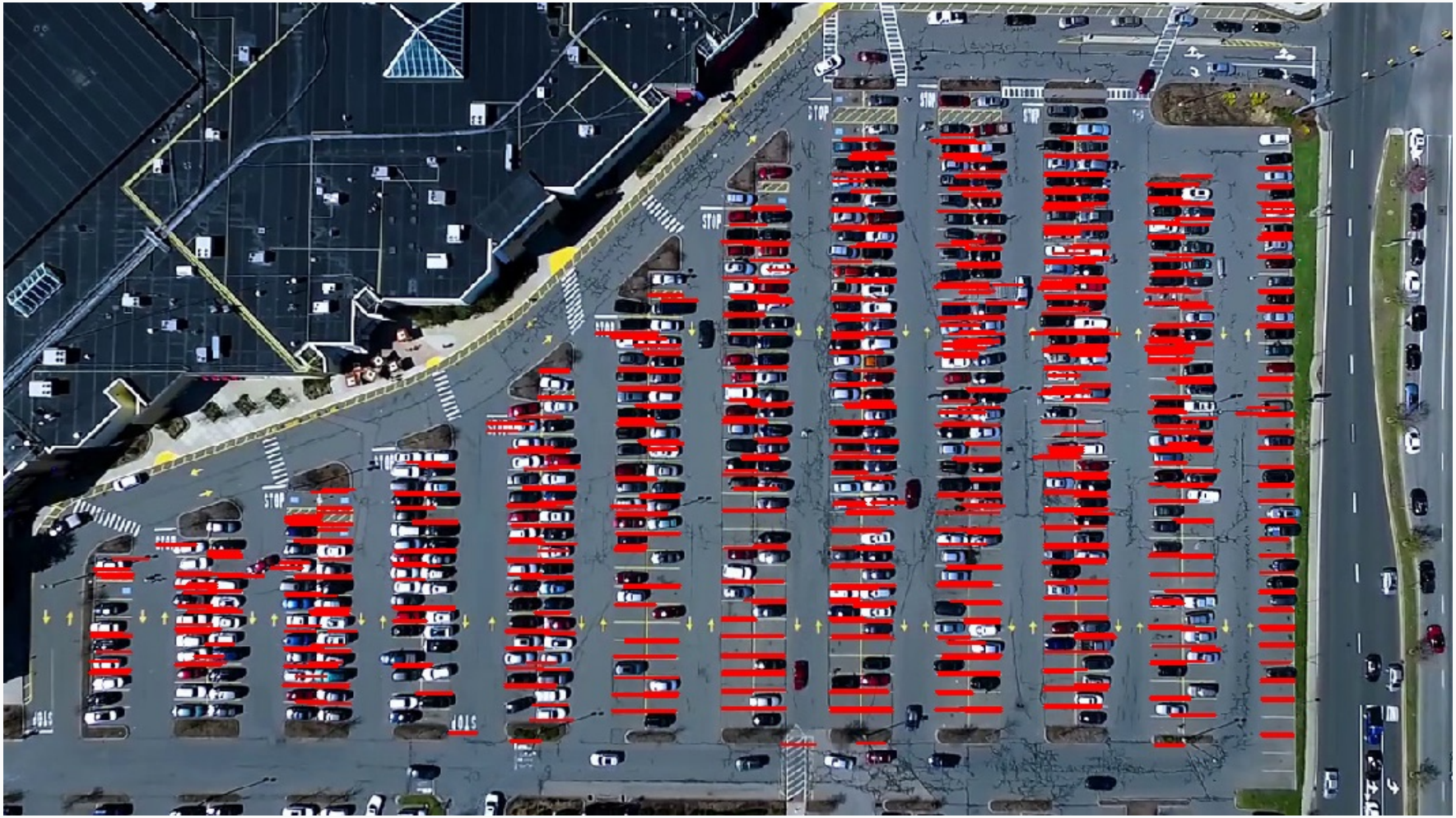
2. Turn the image in grey shades and apply a canny edge

Steps to detect the parking spot

3. Manually crop the parking shape to avoid useless information

Steps to detect the parking spot

4. Search for the lines in the image: hough lines algorithm

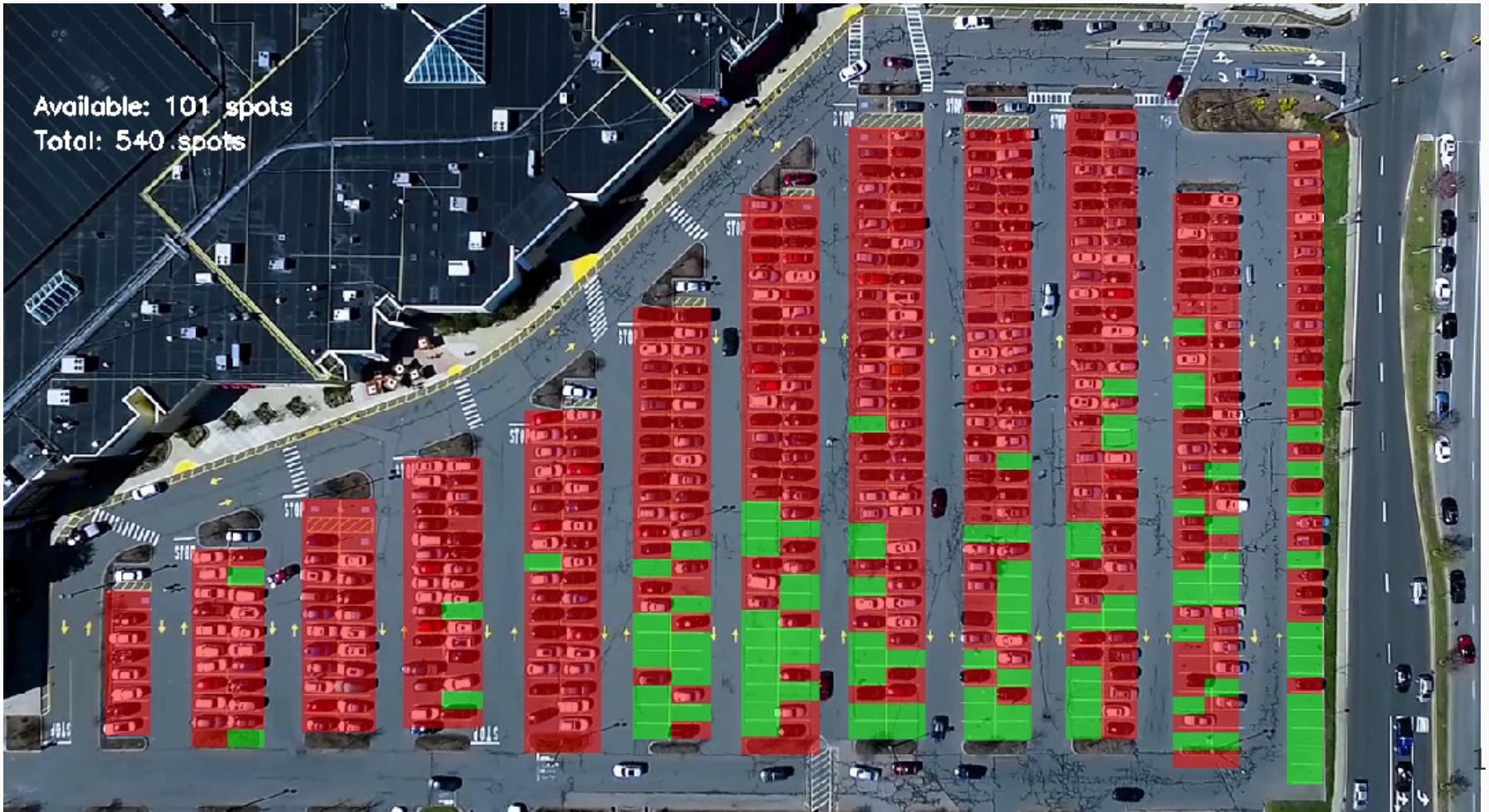


Steps to detect the parking spot

5. Search for clusters of hough lines: it'll be the parking lane

Find the available spots

We just avec to apply the NN designed in the first part of the presentation to every parking spot and draw the rectangles in red if there's a car in it, in green otherwise.



Conclusion

Conclusion

$\rightarrow$ It was a pretty challenging project overall

Conclusion

→ It was a pretty challenging project overall

→ I found every ressources I needed online:

Conclusion

$\rightarrow$ It was a pretty challenging project overall

$\rightarrow$ I found every ressources I needed online:

- About the neural network and transfer learning: a tuto on kaggle, which was not really working out of the box but I made it compatible with my project,

Conclusion

$\rightarrow$ It was a pretty challenging project overall

$\rightarrow$ I found every ressources I needed online:

- About the neural network and transfer learning: a tuto on kaggle, which was not really working out of the box but I made it compatible with my project,
- About the parking spot detection, I got freely inspired by the open-source work of Priyanka Dwivedi on GitHub. I found the step to follow by my own and reading articles, but its implementation was really useful; I even used some of here function, s.t. one to print high quality images or one for the hough-line clustering.