Parking lot detection in order to automate the detection of the number of cars parked in a parking space
Deep Learning Final Report - Group 17

G. Dimitropoulos
4727657
G.Dimitropoulos-1@student.tudelft.nl

M. Manousogiannis
4727517
M.Manousogiannis@student.tudelft.nl

A. Stavroulakis
4747933
A.Stavroulakis@student.tudelft.nl

S. Tsoni
4747941
S.Tsoni@student.tudelft.nl

L.J.R. Weijs
4503813
L.J.R.Weijs@student.tudelft.nl

Abstract

In this paper, we present the final version of our project "Parking lot detection in order to automate the detection of the number of cars parked in a parking space". By using a pre-trained VGG16-net in combination with three fully connected layers we achieve an accuracy of 97% on overall. In this research we also investigate on the influence of training on different parking lots than testing and also considering different weather scenarios. Also we look into ways to make the system of car detection truly automatic by automating the manual pre-processing step of parking lot marking.

1. Introduction

This research builds upon [1], by adding a deep neural network at the back of the proposed network architecture. Our proposed deep convolutional neural network (CNN) tries to classify a parking spot image as available or occupied, based on the existence or absence of a car in this place. Our baseline model is a pre-trained VGG-16 neural network, where the newly added three fully connected layers at the end are optimized with the use of stochastic gradient descent (SGD) and different usages of our training data. For the training and evaluation of our model, we used the PKLot dataset [6], which consists of 12,417 images and 695,899 samples of parking places that were labeled manually by the authors of the aforementioned work. The images are from three different parking lots and are grouped by date and weather condition under which the image was taken.

2. Problem statement

Finding a parking space has been a problem for everyone who uses a car. Even in organized parking lots sometimes it is difficult to find a space, let alone knowing if one's car fits in that space. A common approach to that issue is to use sensors in order to identify if a spot is occupied or not, however, this creates problems such as maintenance of the hardware and initial investment costs. Another approach is to use a camera and a deep learning architecture to keep track of the availability of the parking spaces. We found this research field extremely useful because not only we will get familiar with the deep learning algorithms, but also we will implement something that can be used in real life. From a more technical approach, we would like to examine how accurate a CNN could be to successfully identify empty parking spaces, as well as how a number of different factors can influence its performance. For instance, it would be very interesting to examine how well our deep CNN approach can generalize under different weather conditions with less or more light or how much the performance is affected when our evaluation is done using images from different parking lots than the ones it was trained. The PKlot dataset provides us with the necessary data samples in order to achieve this kind of comparison, and if needed to try to find ways to minimize performance reduction. Finally, another possible research question that could be examined is whether the built network can successfully not only identify if a parking spot is occupied or not, but is also able to count all the available parking spots in a large image of a whole organized parking territory but first pre-processing possible parking places.

3. Scientific contribution

The contribution of this paper is thrice folded,

- Use a pre-trained general VGGNet, state-of-the-art in image classification combined by several fully connected layers to predict whether a parking lot is full or empty.
- Show extensively the performance of the complete deep CNN architecture proposed, by considering qualitative and quantative analysis.
- Investigate how different weather conditions affect the performance of the network.
- Investigate how the system would generalize when tested in different parking lots than the one trained with.
- Optimize the network to reduce the classification error and overfitting/underfitting.
- Investigating on ways to replace the manual preprocessing step of selecting parking places up front by deep learning methods.

4. Technical approach

As mentioned before, in this work we propose a system which employs deep CNN's [5] for detection of vacant and occupied parking slots as such an approach achieves a state-of-the-art performance since 2012 [4]. A pipeline of our algorithm can be depicted in the figure 1. The data, which are needed for training a network like this, should consist of a plethora of images which are captured in different angles and lighting conditions. The main reason behind this strategy is that a diversity in data facilitates the capability of a network to be generalized in a sound manner in new unseen data. A description of our dataset and a technical analysis of our work is given below.

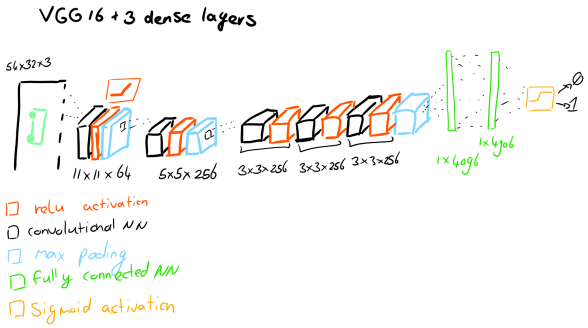

Figure 1: Pipeline and architecture of our proposed deep CNN model.

4.1. Dataset

The dataset [2] which we are going to use is the PKLot dataset and can be found in `http://www.inf.ufpr.br/lesoliveira/download/pklot-readme.pdf`. This dataset has 12,417 images which result in 695,899 samples of parking spots that were labeled manually by the authors of the aforementioned work. In order for occlusions to be minimized, the cameras were positioned at a high altitude. Every space in the images was labeled as free or occupied. To label in that way and to crop the images, the authors of that work developed a tool which is able to save information of each space in an XML file. The process of labeling is also shown in Figure 2. In the later section 6 we will discuss several methods on possibilities of how to automate this manual process.

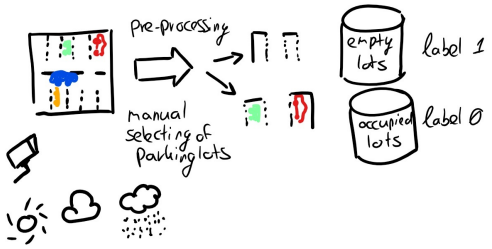

Figure 2: Captures are retrieved from a CCTV camera on a parking lot, for three different weather conditions. Each lot is manually drawn and put in the empty lot dataset or the occupied dataset.

In addition, each parking spot patch was rotated horizontally or vertically depending on their original rotation angle in order for the rotation variability to be removed. The images were obtained under different weather conditions (overcast, sunny, and rainy periods) as this procedure facilitates the capturing of sequences of images showing high variability in terms of illumination occasioned by a variation in the weather conditions. Finally, images were taken from different parking spots presenting distinct features, in a month period (30 days) within a five minutes time interval and cameras were put at several heights. During the implementation of our algorithm, a split of the dataset in 50% training and 50% testing was employed.

4.2. CNN Architecture

Due to time limitations and based on the fact that CNN's need a lot of time to be trained, we decided to use a pre-trained model and only to fine-tune the last fully connected layers of the model. More specifically, we used the pre-trained convolutional layers from VGGNet-F. We chose this version of VGGNet [10] as it is able to work with images of dimension 54x32. Another reason we use this VGGNet is

that is used throughout many computer vision research with great generality, although it can classify to a 1000 different classes building our own car detection would be more time consuming and also probably have a lower accuracy compared to this state-of-the-art model.These convolutional layers were pre-trained in ImageNet [3]. Furthermore, based on the fact that our dataset is not extremely large we thought that it is not a good idea to fine-tune the CNN from scratch due to overfitting concerns. Instead, since we are trying to test a relatively similar scenario, we decided that it was better to use Transfer learning techniques and just fine-tune the last fully connected layers of the network. Beside this, since our data are associated with cars, we thought that they are similar to one of the categories of ImageNet that contains images from different cars. Thus, we expect higher-level features in the CNN to be relevant to our dataset as well.
Based on all the above reasons, we decided to follow the Transfer Learning technique [7].

More specifically, we froze all the layers of our CNN and only fine-tuned the last three fully connected layers. After splitting our training set into two different sets (i.e. training-validation set) we were able to choose our hyperparameters through performing cross-validation. Namely, the hyperparameters we ended up can be depicted in Table 1 and we performed this procedure for 10 epochs. It is worth mentioning here, that we used a small learning rate for the CNN weights that are being fine-tuned as we expected the CNN weights to be relatively good, so we did not want to distort them too quickly.
An overview of our architecture in combination with the shape of the output and the number of parameters of each layer can be depicted in figure 3.

Hyperparameters	Value
Learning Rate	10^{-3}
Batch Size	128
Weight Decay	5×10^{-3}
Optimizer	SGD with Nesterov
Momentum of Nesterov	0.99
Dropout of the 3 FC Layers	0.2

Table 1: Values of Hyperparameter

Layer (type)	Output Shape	Param #
conv1 (Conv2D)	(None, 11, 6, 64)	23296
relu1 (Activation)	(None, 11, 6, 64)	0
norm1 (LocalResponseNormaliz	(None, 11, 6, 64)	0
pool1 (MaxPooling2D)	(None, 6, 3, 64)	0
conv2 (Conv2D)	(None, 6, 3, 256)	409856
relu2 (Activation)	(None, 6, 3, 256)	0
norm2 (LocalResponseNormaliz	(None, 6, 3, 256)	0
pool2 (MaxPooling2D)	(None, 3, 2, 256)	0
conv3 (Conv2D)	(None, 3, 2, 256)	590080
relu3 (Activation)	(None, 3, 2, 256)	0
conv4 (Conv2D)	(None, 3, 2, 256)	590080
relu4 (Activation)	(None, 3, 2, 256)	0
conv5 (Conv2D)	(None, 3, 2, 256)	590080
relu5 (Activation)	(None, 3, 2, 256)	0
pool5 (MaxPooling2D)	(None, 2, 1, 256)	0
flatten_1 (Flatten)	(None, 512)	0
dropout6 (Dropout)	(None, 512)	0
fc6 (Dense)	(None, 4096)	2101248
dropout7 (Dropout)	(None, 4096)	0
fc7 (Dense)	(None, 4096)	16781312
dropout8 (Dropout)	(None, 4096)	0
predictions (Dense)	(None, 1)	4097

Total params: 21,090,049
Trainable params: 18,886,657
Non-trainable params: 2,203,392

Figure 3: VGGNet-F Architecture

To give an intuition and a better understanding of Figure 3, we provide an example of how the number of parameters in each layer is calculated. In the first CNN layer, we have 64 filters of dimensions [11x11], 3 channels and a stride of 4. Thus the total number of parameters in this layer is given by the equation $(11 \times 11 \times 3 + 1) \times 64 = 23296$. Note that the 1 in the previous equation accounts for the bias term. Pooling and normalization layers do not have any parameters. In a similar manner, the number of parameters in each layer can be calculated. Finally, it is worth mentioning that the vast majority of parameters (i.e. almost 18M for a total of about 21M) are due to the fully connected layers.
The final output of our CNN model is going to be a probability vector that includes the probabilities that our input image belongs to the class 0 (i.e. vacant parking spot) or class 1 (i.e. occupied parking spot). An evaluation of our results follows in the next section.

5. Results

In order to be able to evaluate our results, we performed both a qualitative and a quantitative analysis. As far as the qualitative analysis is being considered, plots like confusion matrices, ROC curves, precision-recall curves and figures of our test error were investigated. On the other hand, in quantitative analysis, several metrics like accuracy, precision, recall and F1 measure have been examined in order to obtain a thorough view of our results. Furthermore, we implemented Saliency Maps [9] in order to be able to gain an

insight in which regions in the image our trained net pays most attention to give a specific class.

5.1. Qualitative Analysis

Below we are present the experimental Confusion matrix result in Table **??**. The below analysis gives a very good insight, as it is visible how accurate our deep network is in identifying classes of both occupied and free parking places. As we can notice, both false positives and false negatives values are quite low compared to true positives and true negatives. This fact can also be visualized in the ROC curve and the precision-recall graph presented in Figure 4 and 5 respectively.

Label / **Prediction**	Empty	Occupied
Empty	126024	666
Occupied	3758	86201

Table 2: Confusion matrix of the best performing general model.

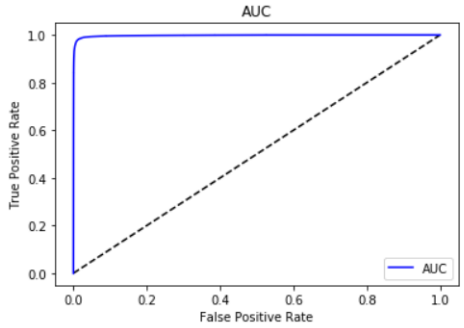

Figure 4: ROC curve of the AUC parking lot dataset.

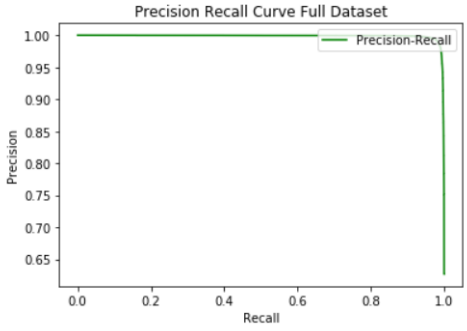

Figure 5: Precision Recall curve on the full dataset.

5.2. Quantitative Analysis

Regarding our Quantitative Analysis, we will demonstrate four basic metrics named Accuracy, Precision, Recall and F1 score. The values of each metric are calculated as the average of the 10 epochs we run in our experiment and shown in Table 3.

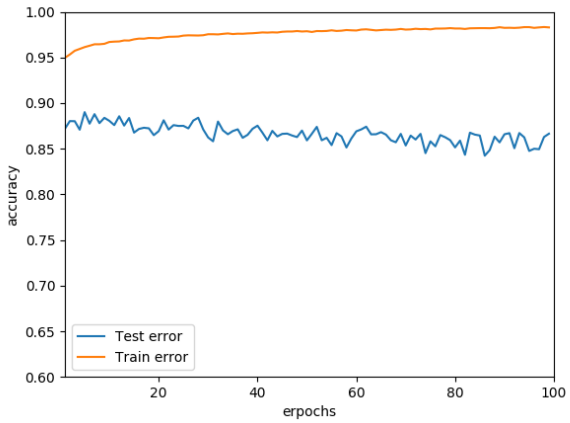

Figure 6: Accuracy over the amount of epochs,Train set: UFPR04 Test set: UFPR05.

Metric	**Value**
Accuracy	0.97
F1 score	0.98
Recall	0.99
Precision	0.97

Table 3: Qualitative metrics of the general deep CNN architecture proposed by us.

As can be noticed, the results are quite high with very few erroneously classified objects. Of course, this was not a surprise, as we are dealing with a binary classification task which is meant to be an easy task for a deep convolutional neural network.

In the graph 6 we can see a visualization of the train and test error throughout 100 epochs, for training and testing the network with different datasets.

5.3. Saliency Maps

In our analysis, we have also included a Saliency Map of one sample of our dataset. In the first image, the reader can see the original parking lot which is occupied by a car. When our network tries to classify this lot as free or occupied, it is focusing on some 'features' of the image that it considers more important in order to make a correct prediction. As can be seen, in the saliency map, the color in the center of the image is more intense, indicating that this is the most crucial region, as this is obviously the location that the car exists.

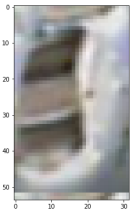

Figure 7: pre-saliency

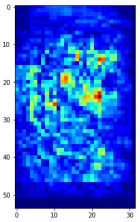

Figure 8: post-saliency

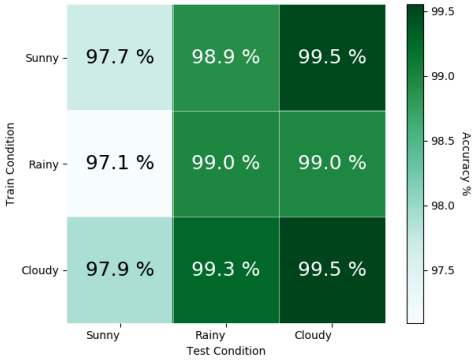

Figure 9: Heatmap with the accuracy percentage for training on different weather conditions than testing on.

5.4. Robustness

One of the interesting research questions we would like to answer, is how would our network perform and generalize when tested with data from different weather conditions and different parking lots than the ones it was trained with.

For that purpose we conducted to sets of experiments. In the first one we test and train with different weather conditions. The first heatmap in Figure 9 presents how the network performed for all the combinations. An interesting observation is that the best performing combinations contain the Cloudy images as test set.

Secondly we wanted to test the generization of the network for completely different datasets. We used three parking lots different from each other, namely PUC, UFR04 and UFR05. In figure 10 we can see the performance of the network for all the different combinations train and test set. As expected, when the network is trained and tested with the same dataset we get the best results. However also the results for the generalization of the network are pretty promising. The worst performance is obtained when training when the PUC dataset and testing with the other two.

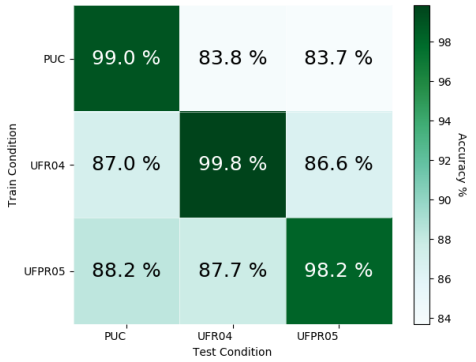

Figure 10: Heatmap with the accuracy percentage for training on different parking lots than testing on.

6. Automatic Pre-processing

In this section we will go deeper into making the pre-processing step, of manual selecting where the parking spots are, fully automatic. Please see the proposed workflow for the pre-processing step in Figure 11. Instead of directly looking at parking spaces, we look at where humans park, which is indirectly a proxy of a parking space. If in the images over time, the same bounding box is detected by a car recognizer we stated that, that specific bounding box is a parking space, with minimal overlap with other parking spaces. We consider three different angles on how to solve for this problem in the following subsections.

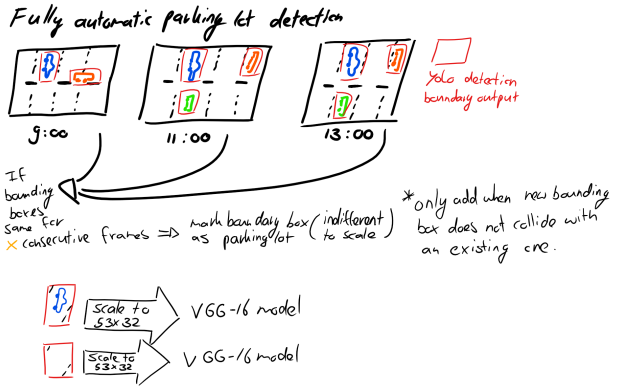

Figure 11: Pre-processing step to detect parking spaces automatically.

6.1. HOG with sliding window

This method has actually the least to do with deep learning but as a valid benchmark we wanted to investigate in such a handcrafted method further. First the histograms of oriented gradients (HOG) features are extracted for the manual hand labeled occupied and empty parking lots, as

shown by Figure 12. When you have a general HOG pattern, you can search across the parking lot to detect cars and propose bounding boxes. As final step the bounding boxes are merged if they are close to each other into a final prediction.

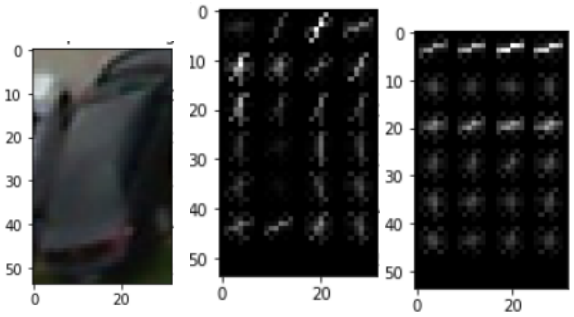

Figure 12: On the left a orginal car picture with in the middle the general HOG for an occupied lot and on the right a HOG feature of an empty parking lot created over all parking lots in the dataset.

The searching of bounding boxes by a sliding window however was not very successful because even when half a car was seen by the window it would draw a potential box, that is why in Figure 13 you can see that everything is proposed as a box which will result in the picture on the bottom of that Figure where there is just one car.

For this method further investigation needs to be done on the size of the sliding window compared to the size of the cars in the picture. But because this will introduce another extra manual solution to the problem at hand we would advice to read into the next sections.

6.2. YOLO

In our opinions the most promising angle is the use of the You Only Look Once (YOLO) model from Redmon et al. (2016) [8]. This model consists of 24 convolutional layers and 2 fully connected layers and is trained on the ImageNet Dataset and, therefore is more than general for our use case. When opting for the pre-trained model version 3 of that same paper, however, the model was only able to detect cars which were close enough to the camera. This result can be seen in Figure 14. In the other parking spaces it was not even able to detect any car.

Retraining the whole YOLO model seems to be a more appropriate option, but it makes a bit less sense to have first a model for detecting cars in the whole picture of the parking lot and then again when we try to classify the marked bounding boxes. Due to time limitations we were not able to re-train the whole YOLO model, but we found another proposing research which also published the weights of the YOLO model which seems to able to tackle

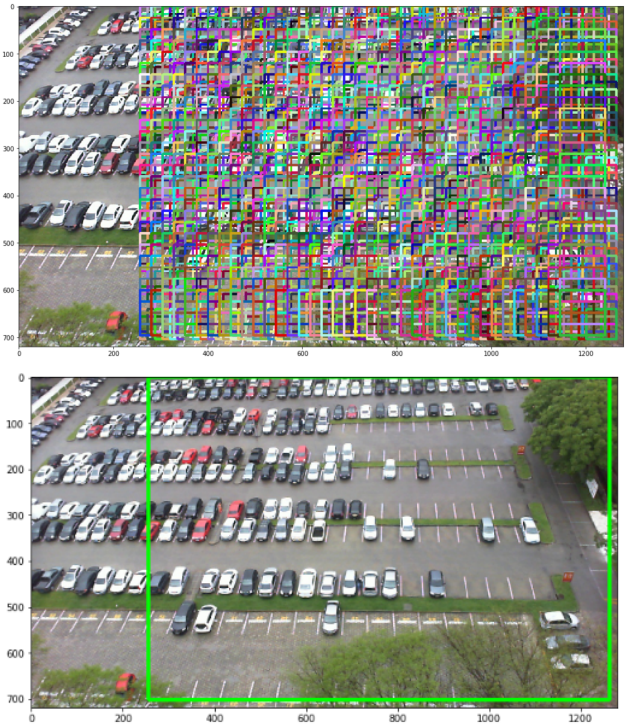

Figure 13: On the top you can see the proposed bounding boxes by HOG feature comparison and on the bottom picture you can see the end result for a prediction of where a car is located. The left of both pictures has been left out of consideration just for visualization purposes.

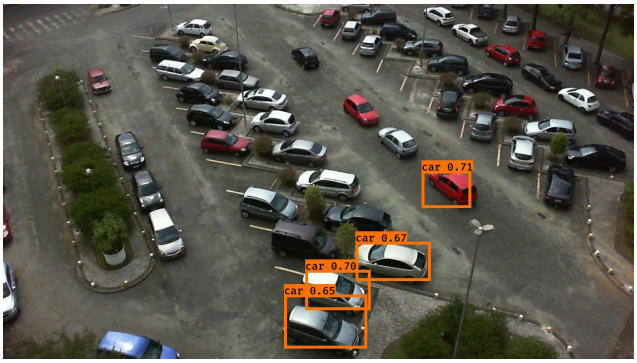

Figure 14: Result of the YOLO model on one of the three parking lot data.

the problem of small objects stated before. This repository can be found at `https://github.com/jekhor/aerial-cars-dataset`, unfortunately we also found out this pre-trained model to late for us to actually test it but it shows the most promise.

6.3. YOLT

Another direction than retraining the YOLO model is by looking at another architecture based on the previously mentioned YOLO, namely the You Only Look Twice (YOLT) model next stated by Adam van Etten (2016) [11]. This model is specialized in recognizing vehicles in satellite imagery, this is done by providing more pre-processing to the input images: upscaling via a sliding window and augmenting the input data for several sizes. Also the architecture is a bit different at the end, they propose that the final convolutional layer has a more dense grid. This model seems to be better in overall accuracy mainly in our opinion because of the right data set used. Again this was not feasible to implement in the short run because the compute engine in the cloud was busy doing the robustness checks and the model was constructed on certain GPU requirements.

7. Conclusion and discussion

On an high level, using a pre-trained model, optimized in another field of computer vision for a part of our complete architecture, is a good choice. This transfer learning, transferring knowledge from another objective space, has improved the calculation time significantly because we only needed to train the three latest fully connected layers. We would advice people to always look for state-of-the-art models before wasting time constructing an overfitted model.

We can definitely say that we found more ways to make this parking lot detection completely automatic by trying one of the aforementioned methods in Section 6, and further research should specifically focus on retraining the fast YOLO network for this use case. We can conclude that the automatic segmentation of the parking lot is a really challenging problem that requires a lot of work. Unfortunately we did not manage to segment the parking spots by our selves (with good results), but we knew from the beginning that this might be an extremely difficult extension for this project.

In this research, we investigated a lot how the network works and how we can optimize it. Thus we tuned the hyperparameters using cross validation and investigated the generalization. The generalization of the network was one of the most interesting and important things, since in real life conditions it is important for a parking lot detector not only to work under specific circumstances. Thus we used a dataset that provided us images with different weather conditions and different parking spaces. The network performs with minimum 97.1% accuracy for all the combinations of different weather conditions and maximum 99.5%. On the other hand the generalization on the different parking lots was not that easy and the network performed with minimum 83.7% and highest 99%. From these two cases we can conclude that the generalization is satisfactory and the model could be used in real life conditions.

References

[1] J. Cazamias and M. Marek. Parking space classification using convolutional neural networks. 2016.

[2] P. R. de Almeida, L. S. Oliveira, A. S. Britto, E. J. Silva, and A. L. Koerich. Pklot – a robust dataset for parking lot classification. *Expert Systems with Applications*, 42(11):4937 – 4949, 2015.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.

[6] P. Lmeida, L. S. Oliveira, E. Silva Jr, A. Britto Jr, and A. Koerich. Parking lot database. https://web.inf.ufpr.br/vri/databases/parking-lot-database/. Accessed: 2018-05-10.

[7] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. *CoRR*, abs/1403.6382, 2014.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[9] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013.

[10] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[11] A. van Etten. You only look twice. https://medium.com/the-downlinq/you-only-look-twice-multi-scale-object-detection-in-satellite-imagery-with-convolutional-neural-38dad1cf7571. Accessed: 2018-05-21.