Parking Lot Vacancy Detection Using Deep Learning for Low-end Hardware

Hao Liu, Sigurd Totland, and Yen-Lin Wu

https://github.com/wuyenlin/norpark_project

Introduction

Parking lot vacancy detection is a use case where deep learning based computer vision techniques can play a huge role. Traditional solutions for vacancy detection are typically costly as a single sensor is needed for every parking space. Computer vision techniques however can accomplish the same task at a much lower cost as only a handful of cameras are needed.

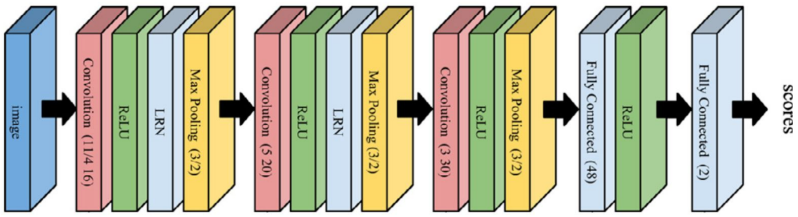
In this project we have reproduced parts of the paper by Amato et. al. (2016). In particular, the paper introduces a lighter version of AlexNet, capable of running on low-end devices like a Raspberry Pi. They also introduce a new dataset CNRPark and perform experiments between this and a baseline dataset PKLot. We referred to this website for details regarding the paper and downloading the dataset.

Paper Contribution

The CNRPark dataset developed by the authors has the parking lot images cropped into patches, each corresponding to a parking space in the parking lot. Each patch is then labeled as either 0 for vacant and 1 for occupied. The dataset is further divided into Sunny, Rainy and Overcast splits.

These patches are scaled to 224x224 and fed through a binary image classifier for classification. The network is named mAlexNet as it is a smaller version of the classic AlexNet that is able to run on lower-end hardware. The structure of the mAlexNet has been shown in the

below picture. Compared with the AlexNet, they only use 3 convolutional layers and 2 fully-connected layers to reduce the training time for lower-end hardware. In addition, the loss function in the mAlexNet is a cross-entropy loss function that matches with that of an AlexNet.



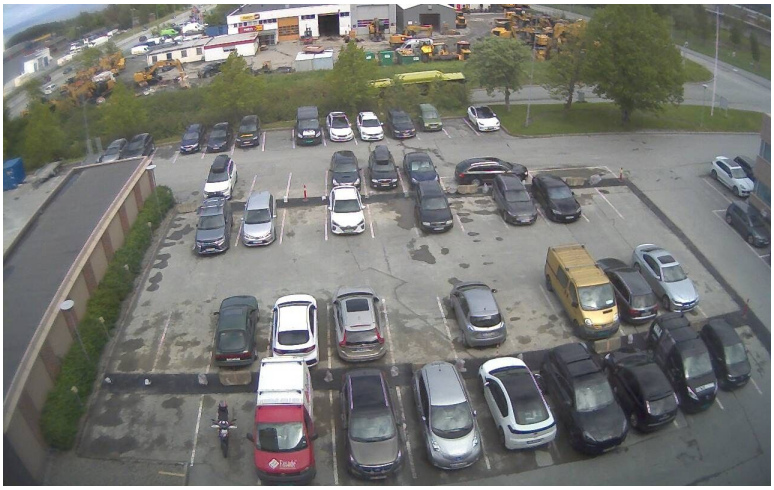
Our contribution

1. Code rewrite

The original code from the author is written in Python2 and Caffe. We rewrote the code in Python3 and Pytorch to examine whether different frameworks would produce the same results. The rewrite consisted of reimplementing the mAlexNet in Pytorch, as well as implementing code for reproducing the results of Table 2 and Figure 5 from the paper. In addition, a dataloader for their datasets had to be rewritten in Python3. This dataloader reads labels from a text file that contains the image locations and their corresponding labels.
Some indispensable information like the number of epochs and the loss functions used had to be found by going through the caffe implementations from the authors, as this information was omitted in the paper.

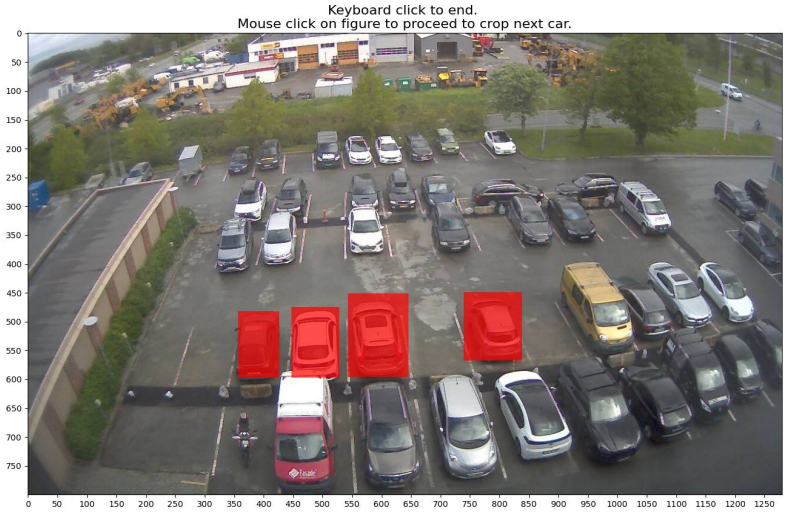
2. Image Segmentation Toolbox

We also created our own test set by downloading images of a surveillance camera in a parking lot at Trondheim, Norway from Inescam. The frame is taken every 15 minutes for 2 consecutive days and the segmented images are labelled manually in the same fashion as CNRPark. There are 46 segmented images per frame and it results in a total number of 4140 testing images. The dataset (in zip file) can be downloaded from here.



To enable rapid labeling of the dataset, we created our own open source toolbox to do the segmentation and labeling task. It has been made public with an MIT license and can be found here. It allows user inputs to manually draw bounding boxes around the parking spaces, as shown below. The input is then saved as a mask file and can be loaded to crop the image into smaller segments. The labeling tool then reads the segments one by one for the user to manually label them by hitting 1 (for occupied) and 0 (for empty) on keyboard. The segmentation file path and the input are logged into a text file, which will be ready to use once the labeling is done.

Our labeling and segmentation toolbox can be used in conjunction with our code by anyone wishing to implement their own parking lot vacancy detection system, i.e. they can create and label their own training dataset to adapt the mAlexNet model to their specific domain.



Reproduction Results

1. Performance among PKLot and CNRPark dataset

The reproduction of Table 2 compares the accuracy from paper, Caffe code, and Pytorch code. It is clear that the result from rerunning caffe code is really close to that from the paper. As for the result of running pytorch code, we notice the most significant differences in the case when

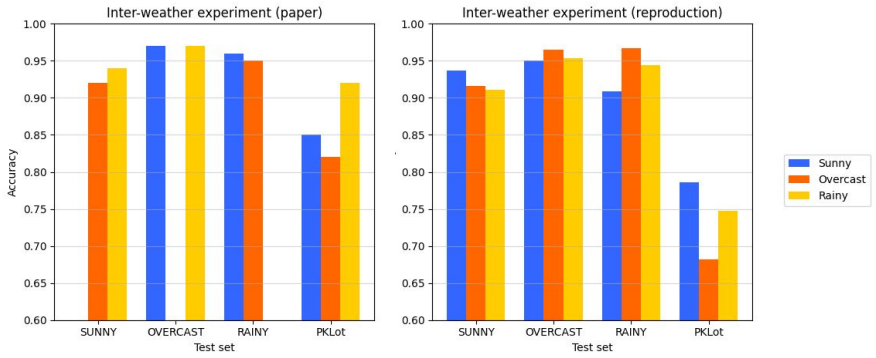
1) trained on UFPR04 and test on UFPR05
2) trained on UFPR05 and test on UFPR04

To further investigate the reason for this, we performed an experiment (see section 4) to see how big the variance is at each epoch.

Test set	Paper Accuracy	Caffe	Pytorch
Train on UFPR04			
UFPR04	0.9954	0.9980	0.9500
UFPR05	0.9329	0.9523	0.8000
PUC	0.9827	0.9916	0.9200
Train on UFPR05			
UFPR04	0.9369	0.9392	0.7400
UFPR05	0.9949	0.9996	0.9700
PUC	0.9272	0.9365	0.8900
Train on PUC			
UFPR04	0.9803	0.9860	0.9400
UFPR05	0.9600	0.9632	0.9300
PUC	0.9990	0.9990	0.9800
Train on CNRParkOdd			
CNRParkEven	0.9013	0.9395	0.8500
Train on CNRParkEven			
CNRParkOdd	0.9071	0.8968	0.8500

2. Inter-weather experiment

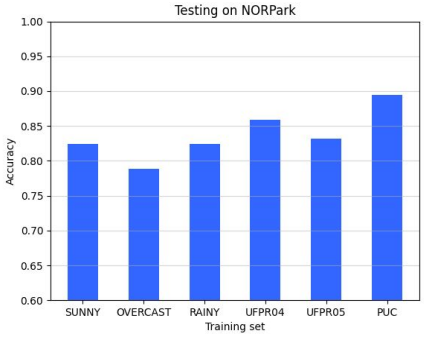
The figure below shows the results of the inter-weather experiment, the left side being the result from paper and the right side being the reproduced one. We can see significantly worse performance when testing on PKLot no matter the weather.



	Sunny	Overcast	Rainy	PKLot
Sunny	0.937/X	0.950/0.970	0.909/0.960	0.786/0.850
Overcast	0.916/0.920	0.965/X	0.967/0.950	0.682/0.820
Rainy	0.911/0.940	0.954/0.970	0.944/X	0.747/0.920

3. Testing on NORPark

Below shows the results of testing accuracies on NORPark. We used the models that trained on Sunny, Overcast, Rainy, UFPR04, UFPR05, and PUC to test our dataset and obtained an accuracy as high as 89%. The accuracies using models trained on CNRPark were lower because the number of images were less than that in any subsets of PKLot. According to Table 1 in the paper, there are 695,899 images in PKLot and 144,965 in CNRPark-EXT. We then infer this to be the main reason why models trained on PKLot have better generalization than trained on CNRPark.



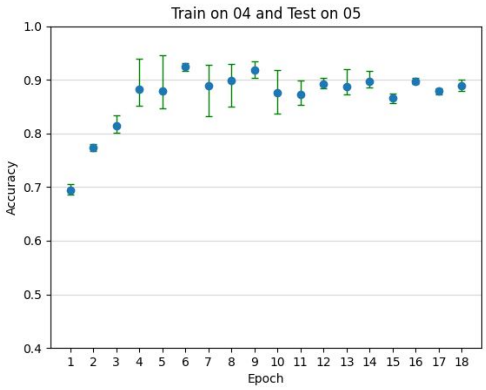
Training set	Accuracy
SUNNY	0.8240
OVERCAST	0.7850
RAINY	0.8280
UFPR04	0.8610
UFPR05	0.8350
PUC	0.8930

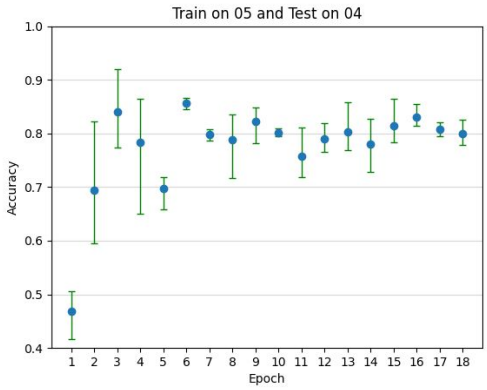
Discussion

1. Why did the results vary from paper?

Variances (Pytorch)

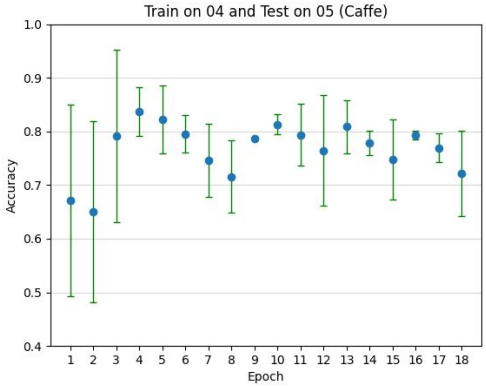
As the 2 cases discussed previously that have significantly lower accuracy compared to the paper, we decided to run the training process for 3 times at each epoch. Below are the 2 figures that show the accuracy versus epoch. Blue dot stands for the mean accuracy over 3 times of training, and the green error bar extends to the lowest and highest accuracy obtained at that epoch. We can see from both results that epoch = 6 yields the highest mean accuracy. And the second result shows greater variance than the first. The reasons are discussed in the next bullet point.





Variances (Caffe)

We have also done the same running author's code. However, each epoch only contains 2 times of training (presumably due to the setup of Caffe), and the result is shown as follows. Epoch = 4 has the highest mean accuracy, whereas epoch = 16 shows the smallest variance.



2. Author response

After contacting the author regarding the result difference (discussed in results 1), we received responses explaining that their reported accuracy in the paper was **NOT** the final accuracy after training for 18 epochs, but rather the highest accuracy throughout the training. To justify it, we can read the highest accuracy from the figure "*Train on 04 Test on 05*" where we have 0.95 at epoch =5 and 0.92 at epoch=3, and the corresponding accuracy in the table in Results 1 shows 0.9329.

Moreover, the authors told us that they in fact trained on a higher resolution version of the CNRPark dataset, than the one that is published on their website. This version has image patches larger than 244px, which explains why their implementation scaled every patch to this size. They had to publish the lower resolution version due to privacy concerns, however in doing this, their paper reproducibility is significantly hurt.

Conclusion

We reproduced part of the results from the paper using their code and our own re-written code. The reproduced results were slightly different from the paper, and we suppose this might be caused by high variance in the experiments and because the authors reported only their highest achieved accuracies, rather than averaged values. We also created our own testing dataset, NORPark, and manually labeled them and tested on the dataset. The accuracy was as high as 89%, implying our dataset has similar quality compared to PKLot. The test set was made with a custom-built labeling and segmentation toolbox which we have made open source.

References

Amato, G., Carrara, F., Falchi, F., Gennaro, C., Meghini, C., & Vairo, C. (2017). Deep learning for decentralized parking lot occupancy detection. Expert Systems with Applications, 72, 327-334.

Almeida, P., Oliveira, L. S., Silva Jr, E., Britto Jr, A., Koerich, A., PKLot – A robust dataset for parking lot classification, Expert Systems with Applications, 42(11):4937-4949, 2015.