Parking space occupancy Detection using Convolutional Neural Network

Daneshwari Kankanwadi,[redacted] B.Tech-CSE, SE, JNU

Abstract—Other team member - Akanksha Marskole. Detecting an empty parking space in real time is a big challenge, especially in metropolitan areas. Real time parking occupancy detection is based on object detection, where objects are detected and classified. Hence, cameras at parking lots, taking real time images, can be used to detect either the number of empty parking spaces or to tell whether a particular parking space is occupied or not. This paper presents a Convolutional Neural Network based approach to classify a parking space as empty or occupied, on PKLot dataset (Bibliography [B-14]).

Index Terms—CNN, PKLot, Empty, Occupied, Parking space, accuracy, confusion matrix

1 Introduction

EMPTY PARKING SPACE IS THE NEED OF ANY CITIZEN AND THIS NEED HAS GROWN WITH THE INCREASE OF AUTOMOBILE INDUSTRY AND POPULATION PER METER-SQUARE.

Traffic on roads due to completely occupied parking lots is a common scenario. This not only leads to clashes and noise pollution, also results in waste of time per person. Thus, as a member of team with Akanksha Marskole, I have provided a CNN based architecture to classify whether a parking space is empty(vacant/unoccupied) or occupied. This paper presents a CNN model that is trained and tested on the segmented parking spaces from parking lot images.

2 Previous Work

Since their creation, CNNs have been an important architecture to solve image based problems. Convolutional Neural Networks(CNN), since their introduction, have provided an edge over classical Machine Learning algorithms. They are widely used for object detection and Computer Vision area. To detect parking space and also classify it is an important task that is required. Earlier, image processing (Bibliography [B-15]), CNN[1] and YOLO[2] based approach have been used to detect empty parking spaces. I have used PKLot (Bibliography [B-14]), a robust dataset of parking lot images.

2.1 About PKLot dataset

The PKLot (Bibliography [B-14]) dataset has both coloured parking lot images and segmented images of individual parking spaces. The segmented parking space images are coloured images with different sizes.

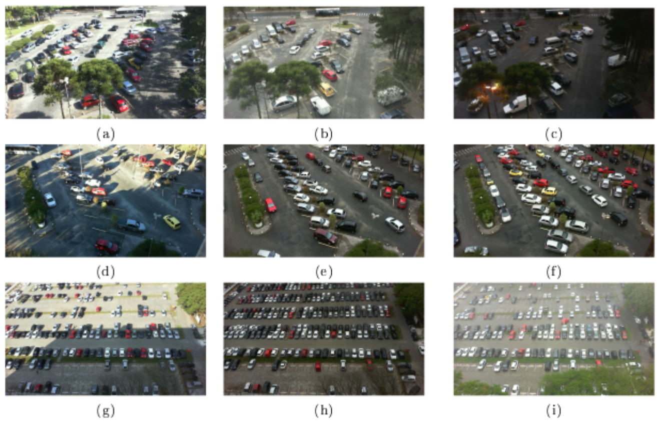


Fig. 1. UFPR04 parking lot images from PKLot dataset (a), (b), (c) are from UFPR04, (d), (e), (f) are from UFPR05 , (g), (h), (i) are from PUCPR parking lot of sunny, rainy and cloudy weather respectively where images are in order.

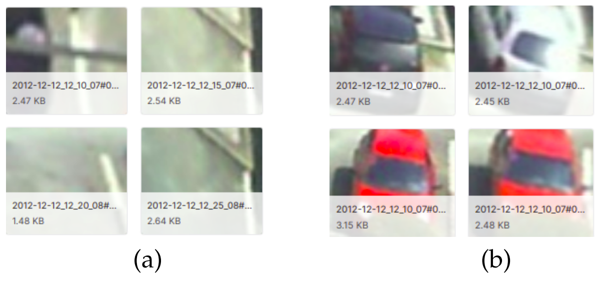


Fig. 2. Segmented parking spaces from UFPR04 parking lot, (a)Class Empty, (b) Class Occupied

3 Challenges

Some of the key challenges that might effect the working of model are -

1. The segmented images in PKLot dataset are of varied sizes. Thus they have to be reshaped but extending their sizes may add blurness and images after reshpaing might not be clear.

• *Daneshwari Kankanwadi is with the Department of Computer Science and Engineering, School of Engineering, Jawaharlal Nehru University, New Delhi, India*
mail to:Daneshwari Kankanwadi

Manuscript received December 12, 2020; revised December 14, 2020.

2. If colour of the car is similar to that of road then the model may misclassify the images.

3. Due to presence of shadow or weather as rainy or cloudy, then image may be classified as occupied due to dark pixel values.

4. The road may have cracks and if that crack is on empty parking space, then model may not recognise it as empty.

4 Methodology

Data Preparation- :

The PKLot dataset provides images of parking spaces and different parking lots. To feed these images in CNN model, the image pixels and labels have to be stored in a file that can be accessed when required. Thus, all the images were reshaped to size of 45x45 with their labels stored as, 0-Empty and 1-Occupied. The information of segmented parking space images is stored in ’ufpr04.csv’ and ’ufpr04_arr.csv’ files. The columns are- “parking_space_name”, “weather”, “labels”, “pixels”. Also, for more convenient, the information with the same format is stored in .npy files - ’ufpr04.npy’ and ’ufpr04_arr.npy’.

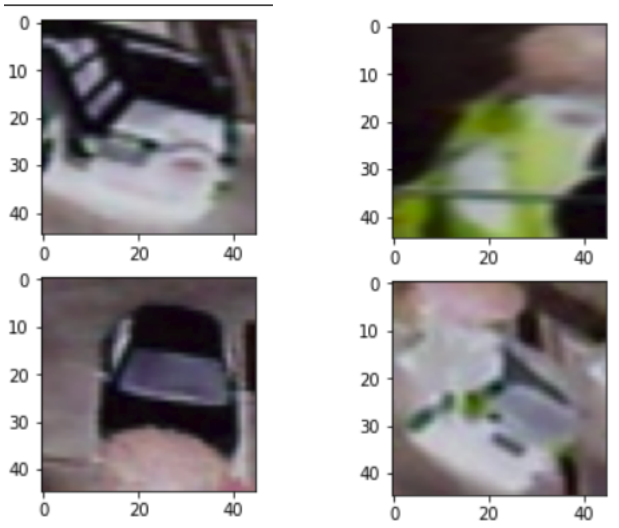


Fig. 3. Reshaped images to size 45x45

Before the pixels of images can be used, preprocessing is done to ensure that all pixel values are in the range [0,1]. Therefore, the image pixels were first zero-centered by dividing them by standard deviation and then normalising them. Now these pixels can be used to train the model.

CNN Model Creation: □

The images with shape (45,45,3), where the format is (IMG_HEIGHT, IMG_WIDTH, NUM_CHANNELS), are fed into the CNN model. The model has two convolutional layers with, first layer has 32 filters and second has 128 filters. The filter size in each convolutional layers is 3x3. The convolutional layer is followed by activation by ’relu’ activation function. Since there is no padding during convolution, the size of input and output from a single convolutional layer is not same. Maxpooling is used for pooling layer, where the kernel size is 2x2, which when applied to input gives the maximum of pixels under the filter i.e. region of 2x2. To avoid overfitting the model on the data, dropout layers with dropout rate value 0.5 is applied.

After all the convolutional layers, a flattened layer is applied so as to flatten the final three-dimensional output i.e. feature map to one-dimension so as to feed it in the neural network. The neural network has one hidden layer with 32 neurons and ’relu’ activation function is used for each neuron. The output layers has two neurons corresponding to each class and ’sigmoid’ activation function is used for each neuron, since the model is a binary classifier. The output is one-hot encoded i.e. each class has a bit that tells about whether the image belong to that class or not. The output look like for class 0(Empty) - [1 0] and for class 1(Occupied) - [0 1].

The arcitecture diagram with all the layers and the shape of their inputs and outputs is shown in Figure 4. □

Since the model is a binary classifier, hence loss function ’binary_crossentropy’ is used. ’Adam’ optimizer is used to optimize the loss function. Finally when the model is created, there are 370,082 trainable parameters and 320 non-trainable parameters. The trainable paramters are updated with every input sample, hence at the end we will have a trained model with updated weights.

5 Results

The CNN model was trained with number of samples being as shown in Table .1

Training	43521
Validation	14507
Test	14507
Total	72535

TABLE 1
Number of images in training, validation and testing datasets

As the model was trained and validated on training and validation data respectively, the change in the accuracy and loss values is plotted in the Figure 5.

The accuray obtained on training data was , and on validation data was . When the CNN model was tested for testing data, the confusion matrix obtained is as shown in Table 2

The confusion matrix is between target and predicted classes. Thus when according to the performance metrics in Table 3:

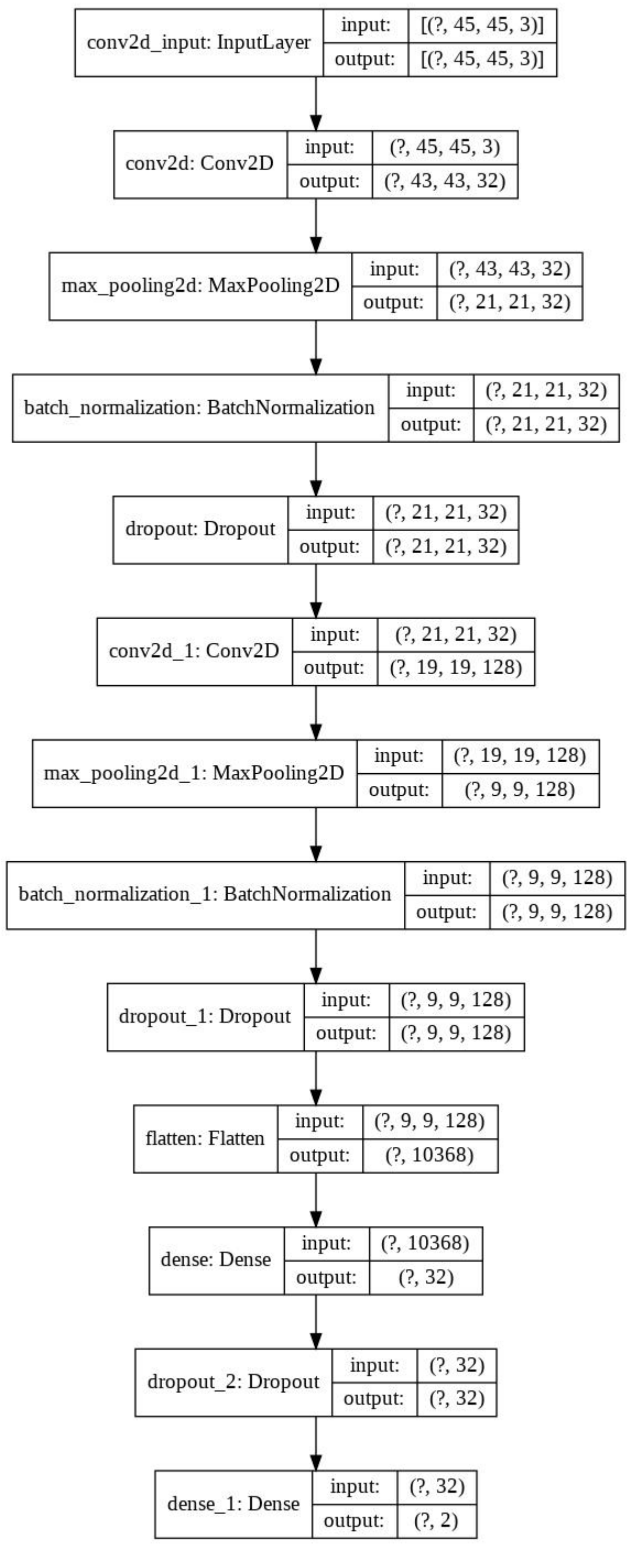

conv2d_input: InputLayer | input: [(?, 45, 45, 3)] | output: [(?, 45, 45, 3)]
conv2d: Conv2D | input: (?, 45, 45, 3) | output: (?, 43, 43, 32)
max_pooling2d: MaxPooling2D | input: (?, 43, 43, 32) | output: (?, 21, 21, 32)
batch_normalization: BatchNormalization | input: (?, 21, 21, 32) | output: (?, 21, 21, 32)
dropout: Dropout | input: (?, 21, 21, 32) | output: (?, 21, 21, 32)
conv2d_1: Conv2D | input: (?, 21, 21, 32) | output: (?, 19, 19, 128)
max_pooling2d_1: MaxPooling2D | input: (?, 19, 19, 128) | output: (?, 9, 9, 128)
batch_normalization_1: BatchNormalization | input: (?, 9, 9, 128) | output: (?, 9, 9, 128)
dropout_1: Dropout | input: (?, 9, 9, 128) | output: (?, 9, 9, 128)
flatten: Flatten | input: (?, 9, 9, 128) | output: (?, 10368)
dense: Dense | input: (?, 10368) | output: (?, 32)
dropout_2: Dropout | input: (?, 32) | output: (?, 32)
dense_1: Dense | input: (?, 32) | output: (?, 2)

Fig. 4. Architecture diagram of CNN mode*l*

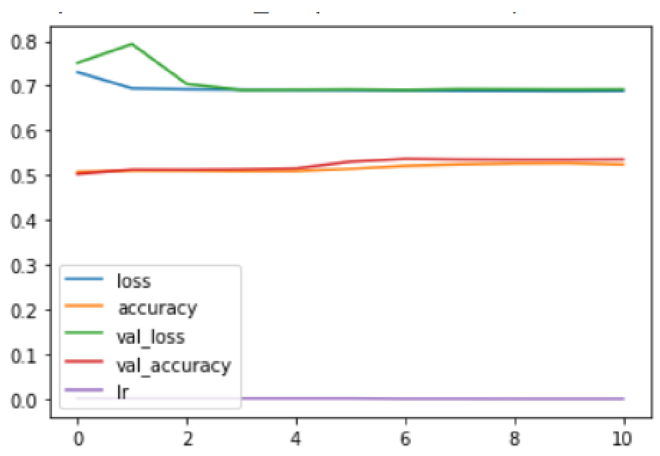
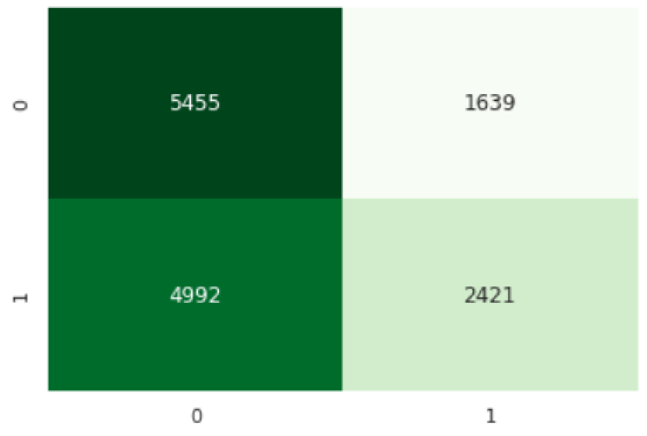


Fig. 5. Plot of the loss and accuarcy values on training and validation data

TABLE 2
Confusion matrix for binary class values



6 Conclusions

From the results above we conclude that weather has an important role during testing, since it may be the case that image that was occupied with grey car may be misclassified as empty, due to colour of road. Also while training, it may have not encountered all the images with white line classified as empty.

Also, class Occupied had a little number of images more than of Empty class. Thus it may have created a bias toward the class as Empty. Thus, proper augmentation can be used to cover this difference and performance of model can be improved.

Since, the model is created with only two convolutional layers, more such layers can be added so that model learns more number of different features. This would also improve the performance since feature map now would have more sharp features.

	Class 0 (Empty)	Class 1 (Occupied)
Precision	0.52	0.60
Recall	0.77	0.33
F1 score	0.62	0.42

TABLE 3
Precision, Recall, F1 score

7 Future Work

In my work, I have used the segmented images from a single parking lot- UFPR04 for classification. Further work can be done by using segmented parking space images from other parking lots for both training and testing or training on one parking lot and testing on other parking lots or training and testing on images from all the three parking lots together.

Also, parking lot images can be used for object detection, where the model would first detect the parking space and then classify it as empty or occupied, or the output can be a number of empty parking spaces. In addition to this, model may also be trained to create bounding boxes on all parking spaces and then classify them.

Link to code base -

Links to all my codes and other files.

1. Link to code repo - Final_CNN

2. Link to code file named "CNN_parking.ipynb". When clicked, this takes to a page where you have to click on 'Open with Google Colaboratory' provided there.

3. Link to dataset folder named "dataset". This folder has 'ufpr04', 'ufpr04_arr', 'UFPR04', 'ufpr04_cloudy', 'ufpr04_rainy' and 'ufpr04_sunny' folders. 'UFPR04' contains the images which have been stored, in form of pixel array in the files, in rest of the folders.

Out of these, I have used datasets 'ufpr04' and 'ufpr04_arr'. The .npy files from these have been used in my code.

4. Link to actual dataset - PKLot. This is the actual dataset as mentioned in Bibliography [B-15].

Bibliography

[B-1]https://pythonprogramming.net/loading-custom-data-deep-learning-python-tensorflow-keras/

[B-2]https://machinelearningmastery.com/one-hot-encoding-for-categorical-data/

[B-3] https://stackoverflow.com/questions/53317592/-reading-pascal-voc-annotations-in-python

[B-4]https://github.com/matterport/Mask_RCNN/

[B-5]https://towardsdatascience.com/smart-parking-an-application-of-ai-9a4af90b1de6

[B-6]https://towardsdatascience.com/parking-occupancy-detection-de25cc0966ef

[B-7]https://towardsdatascience.com/how-to-train-your-own-object-detector-with-tensorflows-object-detector-api-bec72ecfe1d9

[B-8]https://medium.com/@ageitgey/snagging-parking-spaces-with-mask-r-cnn-and-python-955f2231c400

[B-9] https://cs231n.github.io/neural-networks-2/

[B-10] https://www.tensorflow.org/tutorials/images/cnn

[B-11] https://www.analyticsvidhya.com/blog/2020/02/learn-image-classification-cnn-convolutional-neural-networks-3-datasets/

[B-12]https://machinelearningmastery.com/index-slice-reshape-numpy-arrays-machine-learning-python/

[B-13]https://medium.com/python-pandemonium/data-visualization-in-python-bar-graph-in-matplotlib-f1738602e9c4

[B-14] 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.

Link to PKLot dataset

[B-15] Parking Lot Occupancy Tracking Through Image Processing.

Link to paper

References

[1] G. Amato, F. Carrara, F. Falchi, C. Gennaro, C. Meghini, and C. Vairo, "Deep learning for decentralized parking lot occupancy detection," *Expert Systems with Applications*, vol. 72, pp. 327 – 334, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S095741741630598X

[2] X. Ding and R. Yang, "Vehicle and parking space detection based on improved YOLO network model," *Journal of Physics: Conference Series*, vol. 1325, p. 012084, oct 2019. [Online]. Available: https://doi.org/10.1088%2F1742-6596%2F1325%2F1%2F012084