Finding Empty Parking Spaces Using parking lot images

Will Ochowicz (wro258), Udaya Tenneti (ut352) & Charu Paliwal (cp38939)
Masters, Computer Science
The University of Texas at Austin

Abstract

Finding a parking spot for your vehicle has become a difficult task in big cities. Most of the time, there are vacant spots, but the drivers do not have information about availability, so they spend lots of time searching. Here, we are trying to establish whether parking images can be processed to estimate the empty parking spots in open lots like shopping mall parking lots or university parking lots. To test the efficacy of this idea, we build convolutional neural network models to predict empty parking spaces on a pre-existing data set of parking images. We built a binary classification model on individual parking spots to identify if the spot is empty or full and achieved 99.91% accuracy in three epochs on segmented images. Using the same model on images of the whole parking lot led to 94.5% accuracy. We then tested classifying images of the whole parking lot using a CNN multi-class classifier resulting in 72.08% accuracy. We tried improving upon this model by running a larger multi-class classifier and a regression neural network, both of which underperformed the original classifier.

1. Introduction

Searching for parking can be a major issue for those traveling to or within a city. Effectively allocating parking is a problem that researchers continue to look into. However, all systems designed to allocate parking also require knowledge of where parking is available in order to effectively route consumers to spaces. Using Computer Vision (CV) to determine whether parking spaces are available is a potential tool to provide reliable input data to models trying to allocate parking.

In this paper, we will follow the methodology from the paper "Parking Space Classification using Convolutional Neural Networks" by Cazamias and Marek from Stanford (hereafter referred to as the Stanford paper) as a reproduction effort and inspiration for further testing ideas. The paper describes two methods:

1) Binary classification on each individual marked parking space in the picture

2) A neural net that labels an image in a class indicating the number of available parking spaces in the picture (multi-class classification with [0,1,…19+] labels)

We will attempt to replicate both approaches. We will also try three of the Stanford paper's suggestions for improvement, first by using the binary classifier to estimate the number of available spots in a whole-lot image, second by using MSE instead of softmax to convert the counter net from multi-class classification to regression for actually predicting number of available parking spots, and third by training a multi-class classification with 40 labels instead of 20 labels.

2. Background

Our research is divided into two phases. In the first phase, we will attempt to verify the findings of the Stanford paper. The Stanford Paper explores two methods of using data from the PKLot dataset. The first method runs a binary classifier on the already segmented images and classifies the images as either occupied or empty. The second method runs a multi-class classifier on the whole-lot images and classifies the images with one of the labels [0 empty spaces, 19+ empty spaces].

In the second phase of this study, we will make three attempts to enhance the whole-lot classifier from the Stanford research. First, we use the binary classifier to label empty spaces in the whole-lot images. Second, we train a multi-class classifier model using a regression loss function instead of a negative log-likelihood loss function. Finally, we retrain the multi-class classifier using labels for up to 40 empty spaces.

3. Material & Data Sources

We used the dataset of parking images available on Kaggle. It has12,417 images (1280X720) captured from two different parking lots in different weather conditions and from different angles. The data is split up into two sections: PKLot and PKSegmented. PKLot are images of the full parking lot from 3 different angles (PUCPR, UFPR04, UFPR05) under 3 different conditions (Sunny, Cloudy, Rainy) on various days. PKSegmented is PKLot images deconstructed into its individual parking spot components, i.e. one PKLot image is segmented into individual images of parking spots. The data is labeled in both sections with whether a parking spot is occupied or unoccupied, as well as the coordinates of the parking spot.

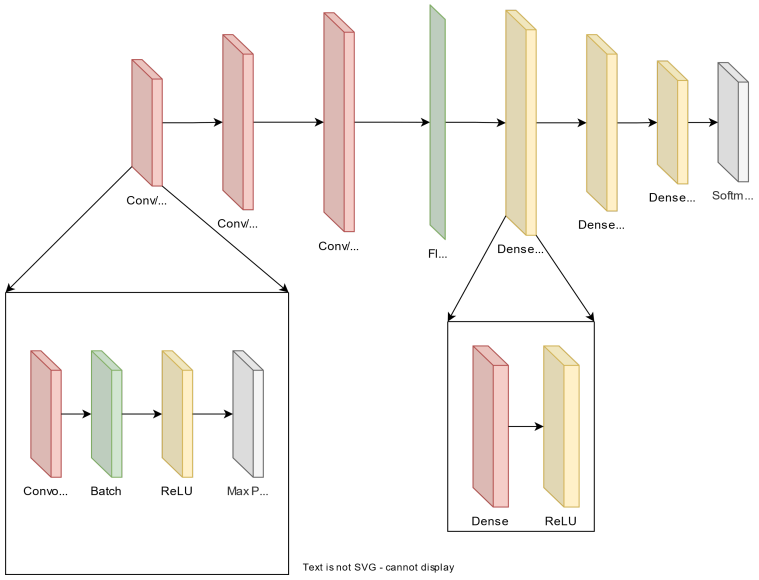
In the case of the PKLot images, the data was not evenly distributed in terms of multiclass labeling. 57% of the images had 19 spaces or more empty. 14% of the images had 0 spaces empty. In the case of PKSegmented images, the images were roughly evenly divided into empty and occupied.

As for labeling, for PKLot images, gold labels were assigned as such: the number of empty spaces if it was less than 19, otherwise 19. This gives us 20 classes [0,1,...19]. In the case of running a neural network regression, the number of empty spaces were used as the gold labels.

PKSegmented images are created from sections of the whole-lot images. Each of the whole-lot angles has had visible parking spaces outlined by the researchers who created the PKLot dataset. The PKSegmented images come from extracting the bounded parking spaces and manipulating the extracted image to a 48x64 rotated image consisting of just the parking space. More details on how the segmented images are extracted is available in the methods section.

4. Methods

The network starts with 3 groups of layers with 10, 20, and 30 channel outputs respectively. Each group consists of a convolution layer, then a batch normalization layer, then a ReLU activation layer, and finally a max pooling layer. Our convolutional layers uses a kernel size of 5, with a stride of 1 and padding of 1. A flattening layer comes next, then three sets of fully connected layers and a ReLU layer, going from 20 channels, to 10 channels, to the final dense layer output respectively. In the case of classifiers, the last channel is a log-softmax channel. Each of our models (binary classifier, multi-class classifier, regression) is identical except for the number of channels in the final dense layer and regression did not use a softmax. Each picture is identified using a softmax layer as either a full parking space or an empty parking space in the output (binary classification). As for loss functions, we used negative log-likelihood in our classifiers and mean-squared error for regression. The optimizer we used was Adam.



Given the uneven distribution of the PKLot images, in order to properly train and test our neural network, a Stratified k-Fold cross-validation was employed. A Stratified k-Fold splits the data into k folds where each fold holds roughly the space percentage of labels. This allows for training to be more consistent and

for the model to not train on specific subsections of the data. Additionally, the images were resized to 256x128 and normalized to (0,1).

In order to process the whole-lot images for use in the binary classifier we created a transform to convert the polygons bounding of the parking spaces into the correct size and shape for the classifier. First, we loaded the image and an associated xml file which contained the bounding boxes for each parking space. Next, we looped over the spaces in the xml file, calculated a boundary for each space, and extracted that section of the whole image. We then rotated the image so that the parking space was horizontal, cropped extraneous pixels, and resized the image to be 48x64. Example images of the process are displayed below.

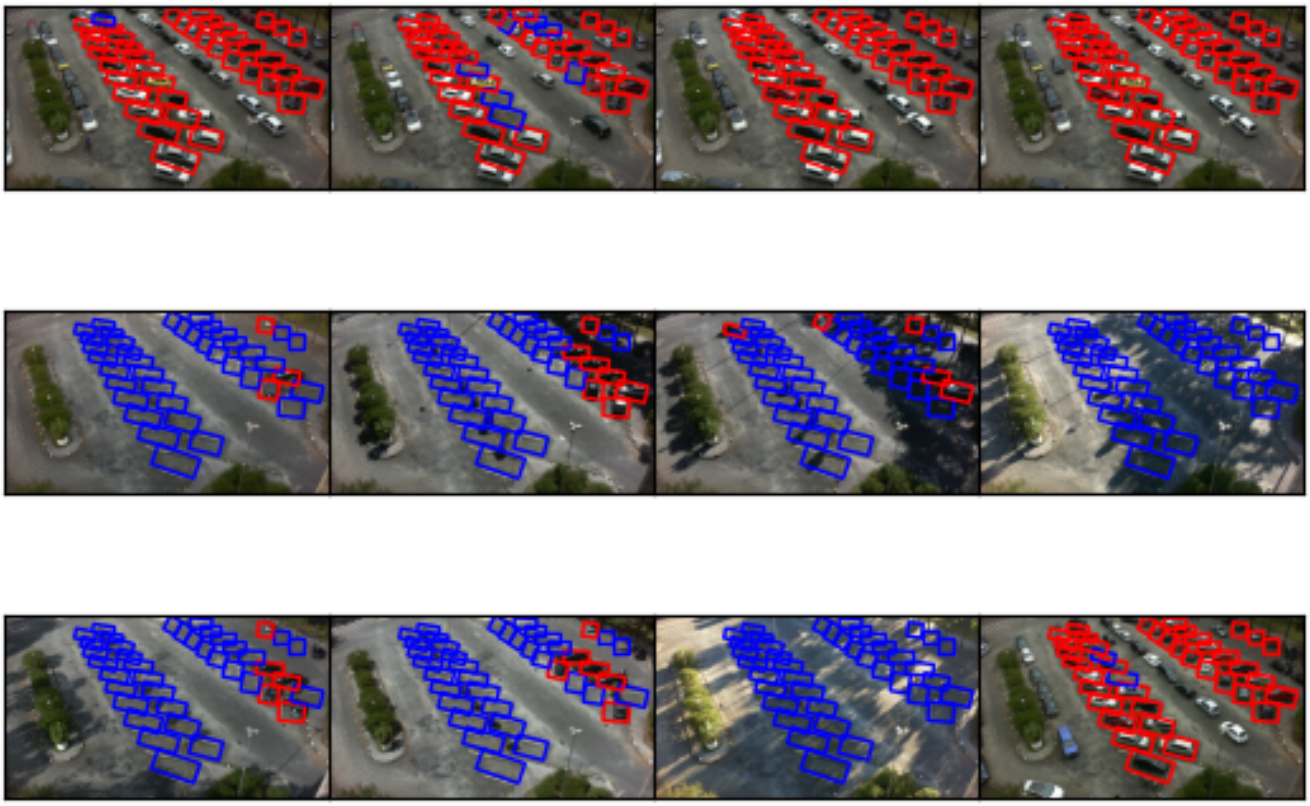


Figure 4A. Examples of whole lot images with empty parking spaces highlighted in blue and occupied spaces highlighted in red

Figure 4B. A parking space

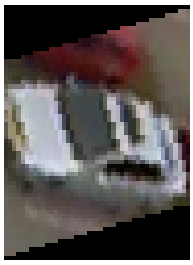


Figure 4C. Rotated

Figure 4D. Cropped

Figure 4E. Resized

5. Results

A. Binary Classification

The first attempt was made to achieve the Stanford results testing multiple hyperparameters. In trying to replicate the Stanford results, we recreated their CNN exactly. However, because they do not specify hyper-parameters such as number of epochs or learning rate, we had to determine the optimal value for those. We tested three learning rates for the binary classifier: $1*10^{-3}$, $1*10^{-5}$, and $1*10^{-7}$. Our heuristic for choosing the number of epochs was to cut-off the number of epochs once testing accuracy began to decrease. We found that the optimal learning rate was $1*10^{-5}$, and the optimal number of epochs was three. Although the epoch count is low for an image classification task, the relatively small image size (48 pixels*64 pixels) and the low number of classes (occupied vs. not occupied) meant that we achieved 99.91% accuracy on the validation set for binary classification after just three epochs. For this test we ran the classifier on all of the segmented images, with an 80/20 train/test split, and validated that the model generalized well by running 5-fold cross validation on the model with using a learning rate of $1*10^{-5}$ and 3 epochs. The average accuracy over 5-fold validation was 99.8%, indicating that the model generalizes well.

Table #. Accuracy of model over number of epochs and learning rate

Learning Rate	$1*10^{-3}$	$1*10^{-5}$	$1*10^{-7}$
Epoch 1	97.52%	99.87%	99.31%
Epoch 2	97.11%	99.87%	99.48%
Epoch 3	51.41%	99.91%	99.56%
Epoch 4	51.41%	99.91%	99.61%
Epoch 5	51.41%	99.90%	99.64%

Table #. Average loss of model over number of epochs and learning rate

Learning Rate	$1*10^{-3}$	$1*10^{-5}$	$1*10^{-7}$
Epoch 1	0.3381	0.3146	0.3215
Epoch 2	0.3421	0.3146	0.3191
Epoch 3	0.7991	0.3141	0.3181
Epoch 4	0.7991	0.3141	0.3176
Epoch 5	0.7991	0.3142	0.3172

The accuracy and loss with a learning rate of $1*10^{-3}$ significantly decreased after 2 epochs. It is not clear why this happened, but it may be because the model became stuck in a bad state.

In the Stanford paper, they are able to achieve accuracy of 99.97% after just one epoch. With a learning rate of $1*10^{-5}$, we achieved an accuracy of 99.87% after just one epoch, up to 99.91% after three epochs. This is reasonably close to the results obtained by Stanford. Variations between our accuracy and the Stanford paper's accuracy could be explained by slight variations in the testing and training datasets, and also by variations in the hyperparameters.

B. PKLot: Stratified k-fold on all angles/weather

The first scenario we ran was a stratified k-fold on 90% of the full data set with 10% removed to be used for validation. On average, the accuracy of the CNN multi-class classifier for this scenario was 72.08%. However, accuracy here is measured as whether the classification was exact. If we look at the errors, we can measure the model's distribution of predictions. The errors are as follows -

Mean Error -0.83

Mean-Square Error 10.44

Std. Dev of Error 3.12

95% Confidence Interval of Error [-0.95, -0.72]

As you can see in Figure 5a below, the errors were normally distributed around 0, which is promising. However, in Figure 5b, you can see the confusion matrix showing the CNN mostly guesses at the extremes. Given we didn't know the specifics of the hyper-parameters nor the testing methodologies of the Stanford paper, we were not able to achieve their 80% accuracy under proper testing conditions. However, when training on the full data set and then randomly selecting a section to test/validate rather than using stratified k-fold, we were able to achieve their 80% accuracy. However, we felt this was not an appropriate methodology for testing.

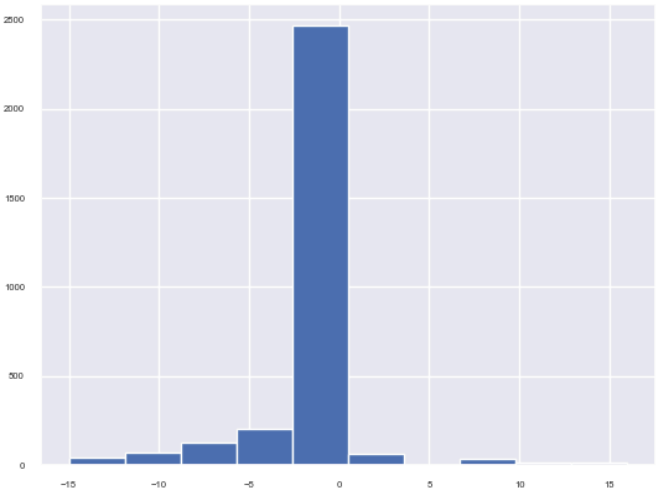


Figure 5a: errors (pred - truth) of CNN on all data using stratified k-fold

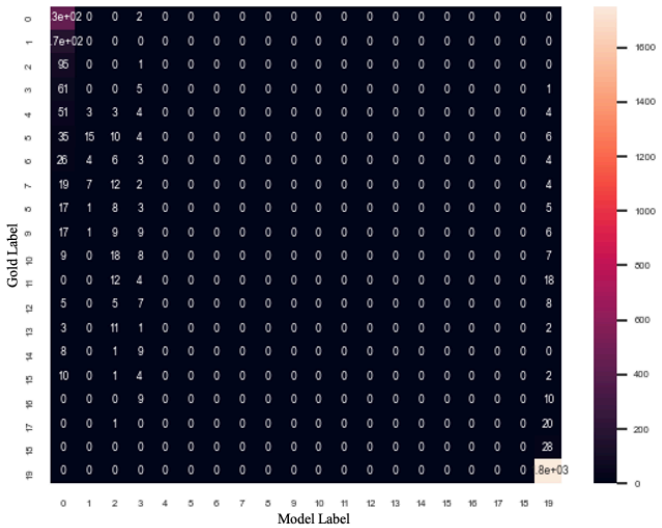


Figure 5b: confusion matrix for CNN stratified k-fold

C. PKLot: Training on subset of data

We also tried training on subsets of the data and testing on the rest of the data, such as training on a specific weather condition and testing on the other conditions, or training on a specific angle and testing on the other angles. The results are below in Figure 5c and 5d. The CNN handles the different weather conditions much better than the different angles. This shows the CNN is much more susceptible to a change in angle of the image rather than the shade or color of the image. The results are lower in general than the CNN for the full data, but some of the subsets of data are very small which may contributes to this. We also observe that the model doesn't overfit, as training on one subset and testing on the same subset doesn't have higher accuracy than the Stratified k-fold CNN on the full data set.

		Test			
		Rainy	**Cloudy**	**Sunny**	**All Others**
Train	Rainy	67.88%	71.01%	61.29%	65.03%
	Cloudy	59.17%	66.97%	54.75%	55.49%
	Sunny	66.77%	75.99%	68.57%	73.74%

Figure 5c: Train vs test accuracy on weather conditions

		Test			
		PUCPR	**UFPR04**	**UFPR05**	**All Others**
Train	PUCPR	79.24%	52.49%	40.73%	46.35%
	UFPR04	60.04%	71.49%	17.36%	38.89%
	UFPR05	65.99%	52.97%	64.35%	59.83%

Figure 5d: Train vs test accuracy on angle

D. Whole-lot classification using binary classifier

After reasonably replicating the results from the Stanford paper, we next attempted to improve on their whole-lot classifier. Because of the high accuracy of the binary classifier, it stands to reason that an accurate estimate of the number of empty spaces in an entire parking lot could be reached by running the binary classifier on pre-segmented sections of the parking lot. This approach has the advantage of increased accuracy on each parking space, but has the disadvantage that it requires both a static camera and manual labeling of the individual parking spaces prior to determining the number of empty spaces. In the PKLot dataset, each of the images is taken from a static camera and have already been labeled.

Our process for running the binary classifier over the whole-lot image was as follows: First, we trained the model on segmented images from either one or two of the parking lot views. This model was a binary

classifier trained exactly as described in section 5.A with a learning rate of $1*10^{-5}$ and 3 epochs. After training the classifier, we tested the classifier on the remaining parking lot views.

Each of the extracted parking space images was then put through the classifier. We measured the overall accuracy (percent of segmented spaces classified correctly), and the mean squared error ((# of empty spaces estimated - # of empty spaces labeled)^2). Results are displayed below-

Trained on\Tested on	Accuracy	Mean Squared Error
Trained on PUCPR & UFPR05 Tested on UFPR04	90.01%	20.61
Trained on PUCPR & UFPR04 Tested on UFPR05	96.64%	7.31
Trained on UFPR04 & UFPR05 Tested on PUCPR	96.85%	14.55

Figure 5e: Results of binary classifier from training on two sets and testing on the remaining set

In general, using the binary classifier scored significantly better than the whole-lot classifier using either multiclass classification or regression as the loss functions. The binary classifier also performed much better when the model was trained on two angles instead of just one, no matter which two angles they were.

E. PKLot: Regression Neural Network

To try and improve upon the multi-class model, we tried running a regression neural network instead of a classifier. Here, we left everything the same but had the model output just one number, and used a mean-squared loss function for backward calculation of the network. The results were mixed. One on hand, the mean error was -0.19 and mean-squared error 25.87. However, even when considering whether the model predicted within five classes of the gold label, the accuracy was only 40%. The issue with a regression in this case is that regression can output non-integer numbers, which makes no sense in terms of parking spaces. In the figure below, we can see that even though most of the errors are around 0, there is a large bias of underestimating the number of empty parking spaces. We still kept the ReLU activation function in the last layer since we wanted to make sure that the network only outputs positive numbers. But clearly this problem is better defined as a classification problem or an object detection counter problem.

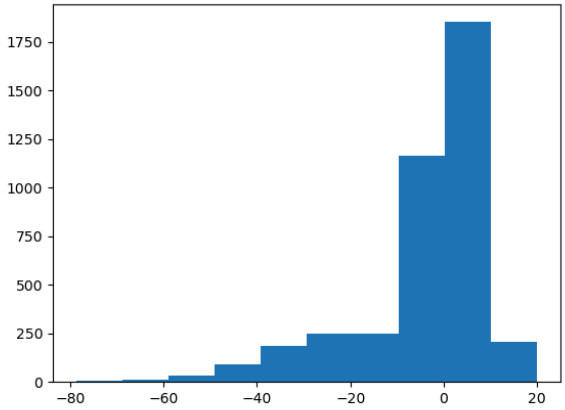


Figure 5f: Errors distribution of regression neural network

F. PKLot: Different Label Multi-Class Classification

Since 57% of the images were under one class, the 19+ label, we decided to try running a larger multi-class classification with 40 labels instead. You can see the distribution of the labels in figure 5g. After running a stratified k-fold CNN, we got results as -

Accuracy 51%

Mean Error -3.19

Mean-Square Error 58.39

Std. Dev of Error 6.94

95% Confidence Interval of Error [-3.44, -2.94]

These results are significantly worse than the 20 classes classifer. We also tried dropping the 19+ label and just running an 18 class classification on the images (only images with 18 empty parking spaces or less). The accuracy was 35%, a mean error of -3.8, mean of predictions of 0.36 and a standard deviation of predictions of 1.02. This shows that this model only learned to guess 0 empty spaces.

Fig 5g: Distribution of classes for 40 labels

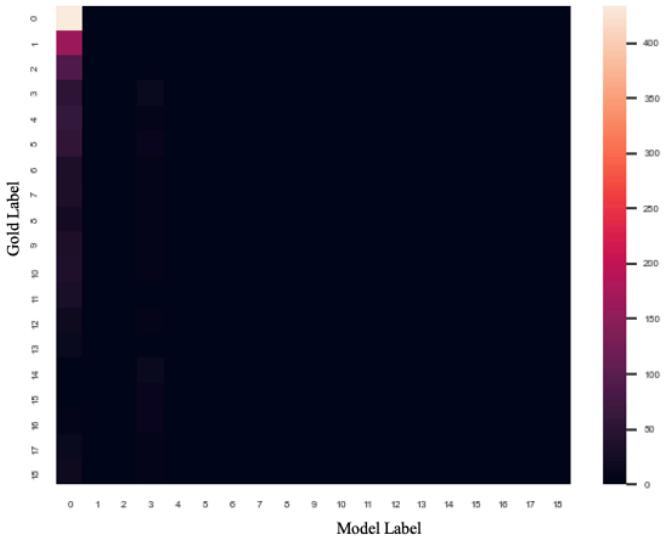


Fig 5h: Confusion matrix on 18 label classification

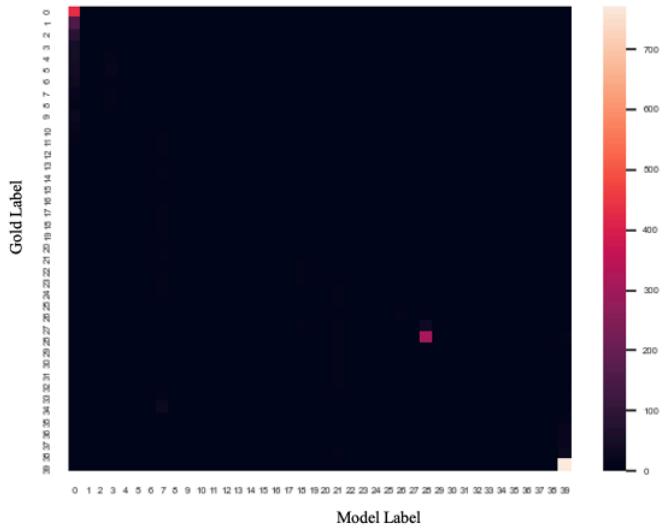


Fig 5i: Confusion matrix on 40 label classification

6. Conclusion

In conclusion, we have successfully demonstrated that computer vision with a single camera and Convolutional Neural Networks has a success rate that is comparable to earlier standard approaches. We were successful in reproducing the Stanford paper results for binary classification. In comparison to the Stanford study's accuracy of 99.97%, we successfully replicated the binary classification results and achieved 99.91% accuracy. Next, we tried to improve on Stanford's whole-lot classifier, given that the binary classifier has successfully produced high accuracy on pre-segmented pictures, it may also perform well with whole-lot images. The accuracy achieved in this instance was 94.5%, which clearly demonstrates that the binary classifier performed better than the whole-lot classifier.

The multi-class classification on entire images of the parking lot achieved an accuracy of 72.08% compared to the Stanford paper reported accuracy of 80%. The differences can be attributed to methodology of training and testing data, in which we used stratified k-fold to split the data. When training on the full set, we were able to achieve 80% accuracy. Further analysis of the multi-class classifier included training on subsets of the data, such as specific camera angles and weather conditions. Both showed the model performs just as well regardless of weather conditions, but suffers when training on one angle and testing on another.

To explore whether we could improve the model, we tried using a regression neural network and increasing/decreasing the class size for the multi-class classification. All results showed lower accuracy than the original 20 label classifier. As a result, we can conclude that the model performs best at predicting when the parking lot is either generally full or generally empty.

A potential avenue for improvement would be to use object detection or segmentation models to identify parking spaces without requiring hand labeling the spaces. Since binary classification on individual spaces is very accurate, automatically identified spaces could be used to quickly operationalize image feed of a parking lot.

Another improvement would be a larger variety of parking lot images. Many of the images had similar states such as completely full or completely empty which led to the model leaning towards picking labels at the extremes. With more images of parking lots in states outside of completely empty or completely full, the model would learn and improve accuracy across all labels.

7. References

- https://people.bu.edu/cgc/Published/TITSprint_09_13.pdf
- http://cs231n.stanford.edu/reports/2016/pdfs/280_Report.pdf
- https://www.inf.ufpr.br/lesoliveira/download/ESWA2015.pdf
- https://arxiv.org/pdf/2106.07228.pdf
- https://www.kaggle.com/datasets/ipythonx/pklotdbs