Parking Occupancy Detection from CCTV Images

Jule Valendo Halim -1425567

GEOM90038 - Advanced Imaging

Parking Occupancy Detection from CCTV Images

Contents

Introduction

Recent developments in data has undergone significant changes from data processing into machine learning due to increased volumes of readily available data (Khan and Al-Habsi (2020)). The introduction of image-based neural network architecture has allowed the field of computer vision to flourish. Computer vision leverages the power of machine learning to derive meaningful information from visual data, which allows them to take actions and recommendations when they detect issues. Powerful advances in machine learning such as the widely popular transformer model that Chat-GPT is based on has also been adapted to be trained on visual information (El-Nouby et al. (2021)). Figure 1 provides a representation of a neural network, which consists of multiple layers. The input layer is the initial data, and hidden layers calculate the weights of each of these data and propagates them forward to other hidden layers. Finally, the output layer provides probabilities of the desired output (e.g., some classification label).

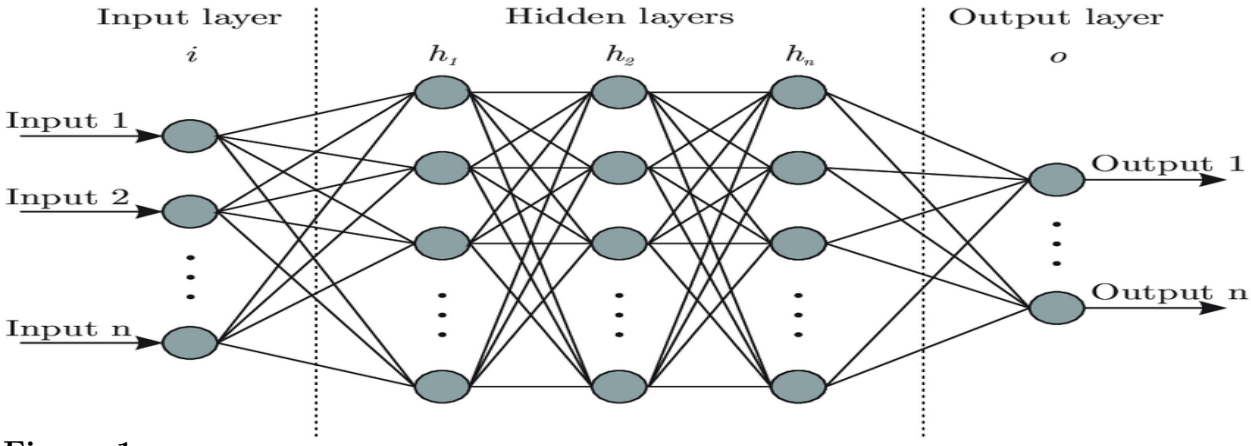

Figure 1

Visual Representation of Layers That Makes Up a Neural Network (Shukla (2019))

Figure 2 provides a simplified neural network architecture for visual data. The visual data that is placed into the input layer depends on the specific architecture and design. In the case of figure 2, a section of the image is placed into the input layer. It is then propagated forward towards additional layers before finally obtaining the results of the output layer, which in this case, attempts to classify the image into four possible species.

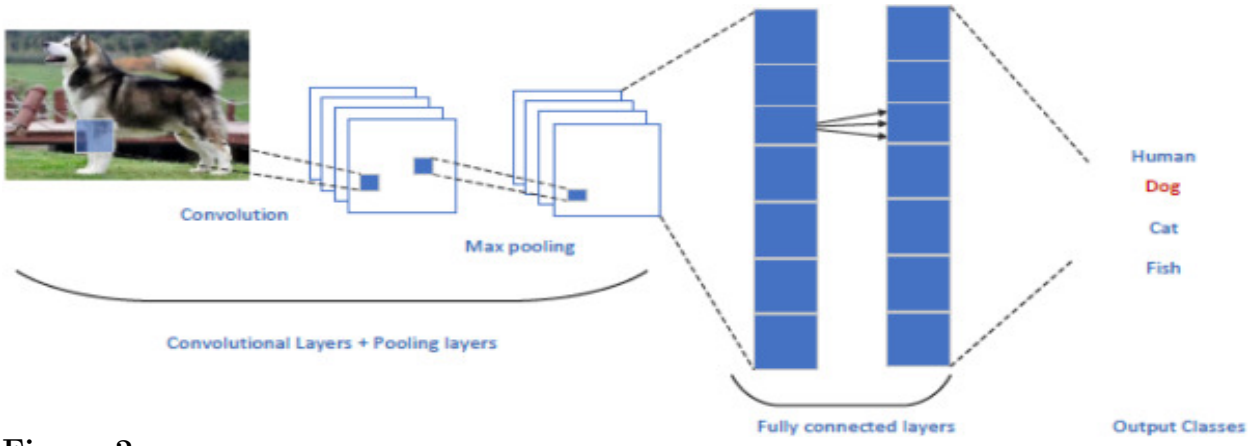

Figure 2

Sample Architecture of Machine Learning Being Used in Computer Vision (Chai et al. (2021))

The use of computer vision has been applied to wide range of different fields. For example, Esteva et al. (2021) discussed the way medical imaging applications in multiple medical areas such as pathology and dermatology has enhanced the level of care provided. Another field in which machine learning has enhanced computer vision is in the automation and digitization of fruit quality measuring and maintainence (Rathnayake P et al. (2022)). Figure 3 shows how computer vision can be used to detect cars and humans in images. These detections are generated post-training, where multiple images are provided for training the neural network. The trained neural network can then be given new images or even real time video to detect what they were trained to. However, as seen in figure 3, such neural networks are not perfect, and can misclassify or not detect parts of the image that they are supposed to (e.g., the model did not successfully classify one of the humans walking).

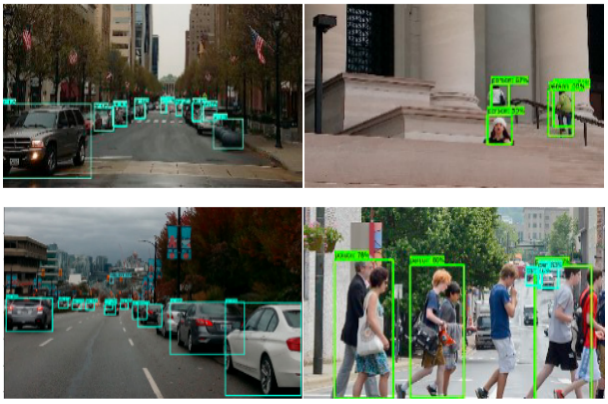

Figure 3

Sample Image Detection Results of Cars and People Using Deep Learning and RCNN Models(Khan and Al-Habsi (2020))

Faster Recurrent Neural Network models (FasterRCNN model) are a form of neural network that can detect objects in images. The FasterRCNN model is based off Ren et al. (2016). It consists of two main components, the Region Proposal Network (RPN) and the FastRCNN. RPNs are a convolutional neural network, which are a form of neural network such as those described above, but are specialized in taking three dimensional data (such as images) as inputs. RPNs are further specialized to handle different schemas of different images. Figure 4 shows the different schemas that the RPN can handle, namely differing scales, differing filter sizes, and multiple references of the same image.

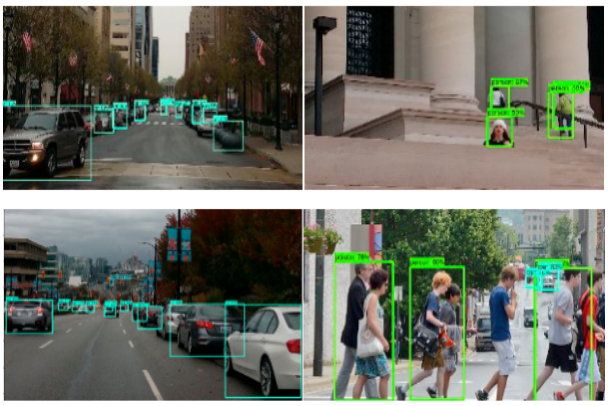

Figure 4

Different Schemas that RPNs Are Adapted To(Ren et al. (2016))

The other component, the FastRCNN (Girshick (2015)) is a neural network that has been shown to be much faster than other image detection neural networks. The main innovation that causes this increased detection speed is the use of of Regions of Interest (RoI). The RoI is defined with a four-tuple (r,c,h,w). The r and c specifies the top-left corner, while the h and w defines its height and width respectively. These tuples are then max-pooled to convert features inside regions of interest into a small feature map with fixed spatial size (e.g., height and width of 6 x 7). Figure 5 shows the overall architecture of the fastRCNN. As seen in this figure, the image is projected into an RoI format before being pooled and trained on a neural network.

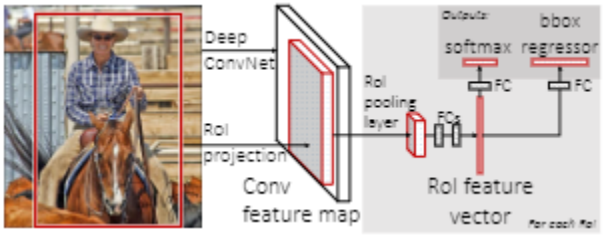

Figure 5

The FastRCNN Architecture, Including the Conversion of Images into ROI(Girshick (2015))

The FasterRCNN architecture has layers that provide output towards the RPN. This is then used to create a proposal, which basically proposes what sort of schema the image is using (see figure 4). The image that is passed through the RPN is also used to create feature maps, which create is then passed to the the RoI pooling before being passed to a classifier. In this architecture, the RPN is performing the 'attention' task of FasterRCNN. Attention allows the model to incorporate sequential information into the model (e.g., in a picture of a man on a horse, the model is able to remember that the section of the image with a hat is in the upper-middle area of the whole image). Figure 6 shows the entirety of the FasterRCNN architecture.

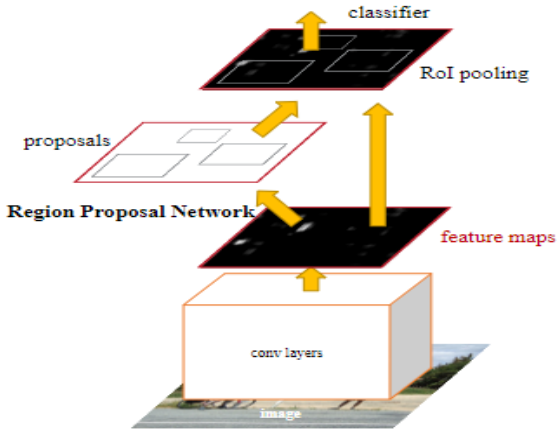

Figure 6

The FasterRCNN Architecture, Showing the Components of RPN and FastRCNN RoI (Ren et al. (2016))

The other model used in this report is the Resnet50 CNN model (He et al. (2015)). The Resnet50 CNN model's building block consists of a series of weight layers that have an relu activation function (an activation function converts the weights provided by the layer mapped to an output). Relu takes the maximum of two values and returns an output depending on the input (e.g., a relu that is max(0,x) and has a binary classification task to classify 0 and 1 would classify an output as 1 if x is greater than 0, and classify it as 0 if it is less than 0). Figure 7 shows the building block layer of the Resnet50. The Resnet50 CNN consists of multiple of these layers along with other kinds of layers, one of which is the max-pool layer. This increased depth of representation is vital for visual recognition tasks, and this model is popular in many visual recognition tasks.

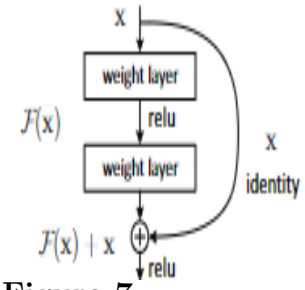

Figure 7

The Resnet50 CNN Building Block (Layer)(He et al. (2015))

In this report, I aim to use and investigate a pre-trained FasterRCNN model and Resnet50 CNN model(trained on the PKPlot dataset) to detect cars and delineate parking spaces in a picture of the Barry Street parking lot. The software used will be MATLAB 2020a, along with additional toolboxes such as the deep learning toolbox. The methodology of this begins with a visualization of the dataset, before using a pre-trained Resnet50 CNN classifier to create a classifier. Afterwards, parking slot delineation is done using FasterRCNN. The dataset for this step involves images of the parking slot and bounding boxes, which identifies the space in which an object is identified (see figure 3, where cars and human areas and identified using bounding boxes).

These results will then be evaluated by calculating recall and precision. These metrics are calculated as

$$\text{Precision} = \frac{TP}{TP + FP} \quad \text{where } TP = \text{True Positives, } FP = \text{False Positives}$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad \text{where } TP = \text{True Positives, } FN = \text{False Negatives}$$

Afterwards, the object performance will be post-processed to improve the resulting classification. Visual inspection of the resulting bounding boxes will also be compared to the ground truth of where bounding boxes should be. Additional information is found in the methods and results section. The entirety of the code is found in appendix D. However, relevant parts of the code will be written throughout the report.

Methods and Results

Visualizing Dataset

This step involves loading in the PKPlot and Barry Street annotated images. These annotations are bounding boxes that are manually set. Figure 8 shows the annotated images of the PKPlot (left) and Barry Street (right) datasets. The PKPlot dataset consists of over 695,000+ parking space images. For this report, only a segment of the total dataset was used.

Figure 8

Images of Annotated from the PKPlot Dataset (left) and Barry Street Dataset (right)

Figure 9 shows a sample empty slot and occupied slot found in the annotated PKPlot datasets.

Figure 9

Sample Image of Empty (left) and Occupied (right) Slots of PKPlot, Sampled From 1500 Empty Spaces and 1500 Occupied Spaces

Creating Car Detector

Re-Training the Resnet50 CNN Model on PKPlot Dataset

To create the car detector, the pre-trained Resnet50 CNN was used. The model was set to train on 70% of the dataset, and 30% was kept as the validation set. The classication layer of the model with a different classification layer that predicts two classes, occupied and empty parking spaces. An additional augmented dataset was also created, which involves two changes, rotating and changing the scales of the original image. This would allow the model to be trained on a larger dataset as well as being able to generalize to unseen car images (e.g., if the same car was rotated, the model can still identify it). The model was then re-trained with the new models. Figure 10 shows the loss and accuracy of the model over 20 epochs. The loss and accuracy converge at around 5-8 epochs, with the model obtaining near perfect accuracy and nearly no loss on both the validation and test data.

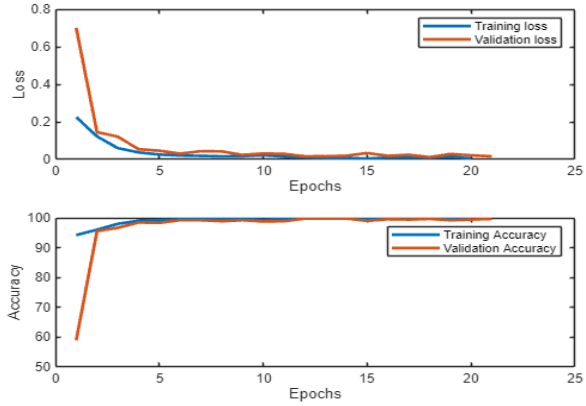

Figure 10

Loss (up) and Accuracy (down) on Validation and Training Datasets, Re-Trained Model on PKPlot

Testing the Model on Barry Street Dataset

The same model was then used to identify the same labels (occupied and empty parking spaces) on the Barry Street dataset. Figure 11 shows the confusion matrix of the re-trained model on the Barry Street dataset. It was able to correctly identify the empty

and occupied spaces with high accuracy, correctly identifying 99.2% of the parking spaces (20.4% were correctly identified as empty and 78.8% were correctly identified as occupied). 0.8% of the predictions were incorrect. The model appears to be able to predict occupied areas more accurately than empty areas (100% accuracy in classifying occupied targets and 96.5% accuracy is classifying empty spaces).

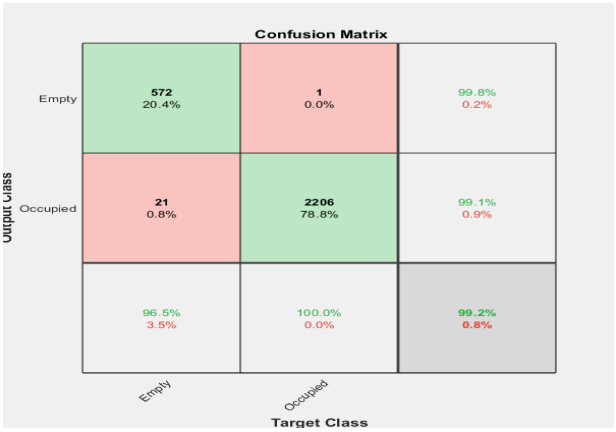

Figure 11

Confusion Matrix of Re-Trained Resnet50 CNN Model on Barry Street Data

Figure 12 shows the visualizations of the 21 incorrectly predicted areas, along with their target scores. A target score of greater than 0.5 for each label indicates that it classifies that space as that label (e.g., an occupied score of 0.6763 means that the model predicted that the model is 67.63% confident that the area is occupied, and the reverse is also true for empty scores). As seen from the figure, occupied scores are generally much higher, meaning that the model was very confident in their wrong predictions of an area being occupied, compared to empty scores, which only has a empty score of 0.58, which does not show high confidence in that prediction.

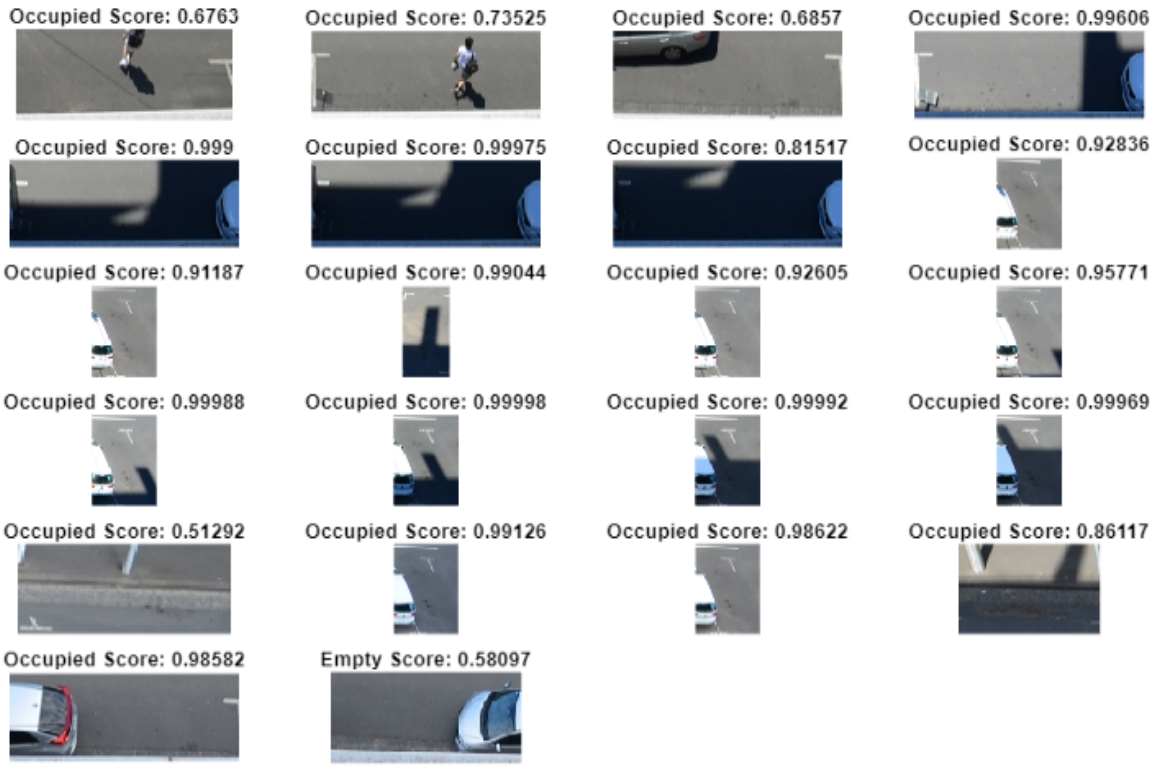

Figure 12

Visualization of 21 (0.8% of the total dataset) wrong predictions, along with their target scores

Automatic Delineation of Parking Spaces

Re-Training the FasterRCNN on PKPlot

This section uses the FasterRCNN pre-trained model that was trained on detection of cars in highways from a mounted camera on a car. It has been shown to be able to predict cars in highways well, but on still CCTV images, the performance is questionable. Similar to the re-training done on the Restnet50 CNN, the FasterRCNN is re-trained on the PKPlot dataset to identify cars and parking spaces. The initial model's performance is shown in figure 13. The graphs plot the model performance over iterations. The first graph (from top left) shows the training loss, which is close to 0. The second and third graphs

show the results of the RPN layer, both accuracy and RMSE. The last two graphs show the overall model's accuracy and RMSE.

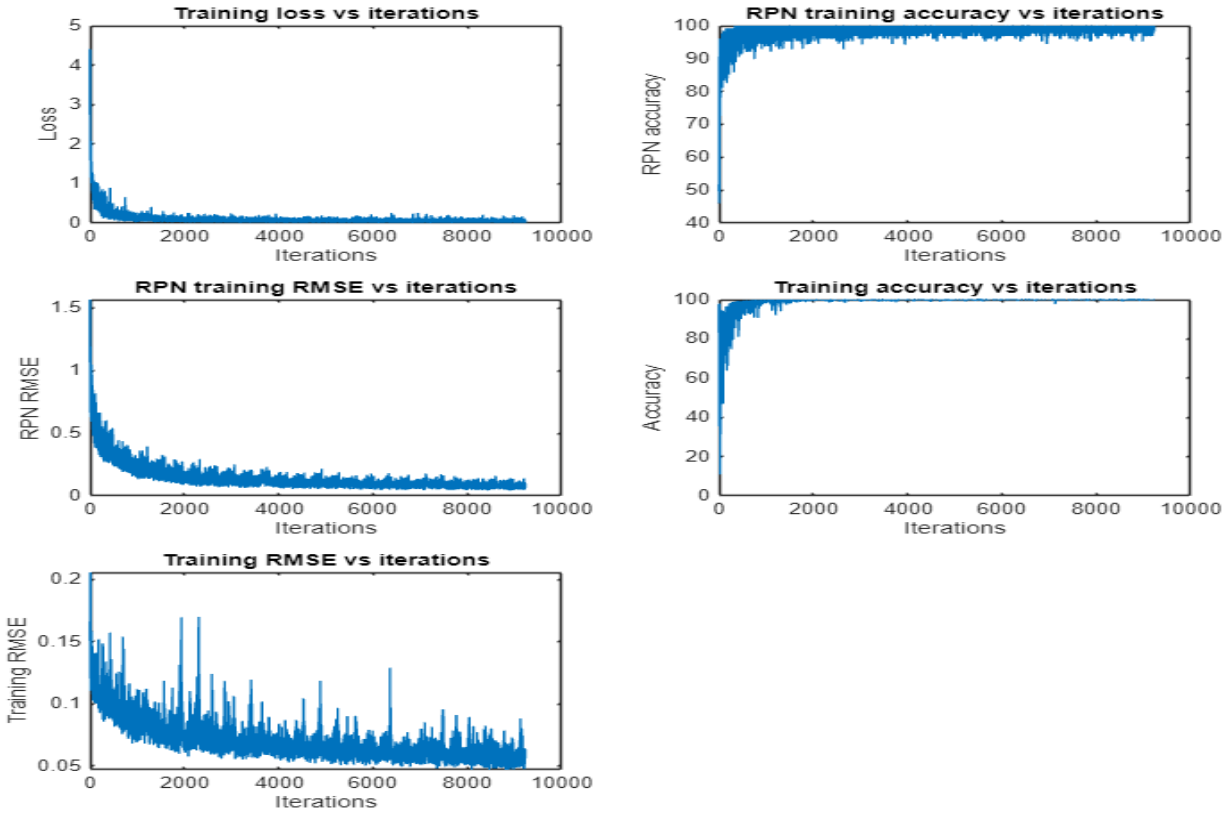

Figure 13

Results of Re-Training FasterRCNN on PKPlot

Visualizing Re-Trained Model Predictions

To visualize the prediction results of the model, the predicted areas in which the model classifies as having a car is shown in figure 14. The yellow bounding boxes have the confidence scores as well (similar to the scores in 12) The prediction confidence are generally quite high, with most correctly identified parking spaces having a confidence of above 0.9, with one exception of the car at the bottom right. However, this car is also cut off and so could contribute to the reduced confidence. Nevertheless, the model was not able to identify all the cars, even in areas where the cars are fully in frame. The car was also unable to correctly identify any empty parking spaces in the image.

Figure 14

FasterRCNN Predictions on the Barry Street

Improving Model Prediction

To improve the model to correctly identify more cars, the model was further trained on more frames from the Barry Street CCTV. The predicted bounding boxes on all the frames were clustered and drawn. Figure 15 shows the clusters represented as scatter points and bounding boxes.

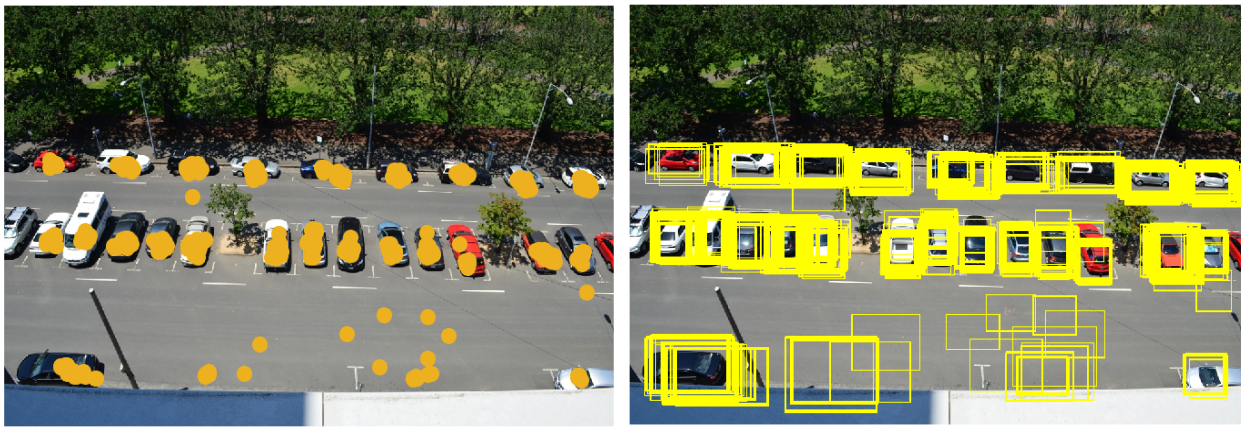

Figure 15

Predicted bounding box clusters of all frames, shown as scatter points (left) and boxes (right)

The bounding boxes (figure 15 (right)) then had their classification scores averaged and plotted. Figure 16 shows the resulting averaged cluster and the corresponding classification scores.

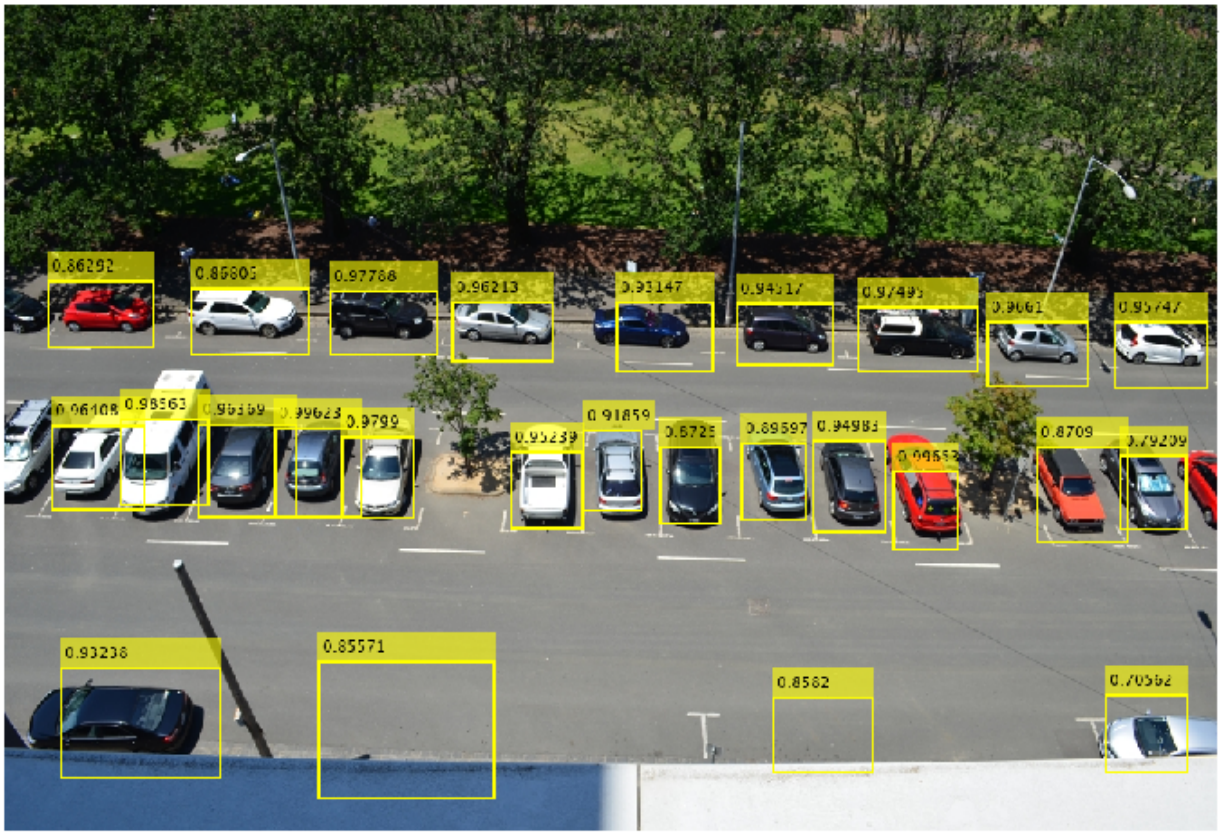

Figure 16

Averaged Bounding Boxes and the Corresponding Classification Scores

Evaluation

Evaluation Steps and Code

To perform evaluation on the resulting averaged predictions, the recall and precisions are calculated. The predicted data is structured as a dataset where each row consists of [x coordinates, y coordinates, length, width] of the bounding box. A ground truth dataset that contains the same structure as the predictions were given, and is used for the evaluation calculations described above. The evaluation steps and corresponding code are:

1. Displaying the means of the bounding boxes described in figure 16.

```
% x(1), y(2), length(3), width(4)
disp(classifiedMean)
```

The resulting table displayed is shown in appendix A, along with additional mean calculations for the width and length.

2. Adjusting the x and y positions of the bounding boxes in (1) by a shift of 141 pixels along the x axis and 58 pixels on the y axis.

```
adjustedClassifiedMean=classifiedMean
adjustedClassifiedMean(:,1)=adjustedClassifiedMean(:,1)-141
adjustedClassifiedMean(:,2)=adjustedClassifiedMean(:,2)-58
```

The resulting table of adjusted bounding box X and Y coordinates are shown in appendix B.

3. Create and input the table in a format for the 'evaluateDetectionPrecision' function.

```
detectionResults = table('Size',[1,2], ...
'VariableTypes',{'cell','cell'}, ...
'VariableNames', {'Boxes','Scores'})
detectionResults.Boxes{1} = adjustedClassifiedMean;
detectionResults.Scores{1} = classifiedScoreMean;
disp(detectionResults)
```

4. Load the ground truth data and create a table similar to step (3)

```
load("FinalCode\GroundTruthBarryStreet.mat")
groundTruthData = table('Size',[1,1], ...
'VariableTypes',{'cell'}, ...
'VariableNames', {'Boxes'})
groundTruthData.Boxes{1} = ParkingSlots
```

5. Obtain the precision and recall of adjusted bounding boxes and plot the results.

```
[averagePrecision, recall, precision] =
    evaluateDetectionPrecision(detectionResults, groundTruthData)
figure;
plot(recall, precision,'-x');
xlabel('Recall');
ylabel('Precision');
title(['Average Precision = ' num2str(averagePrecision) ]);
grid on;
```

Evaluation Results

Table 17 shows the resulting precision and recall calculations on the adjusted X and Y positions.

Bounding Box	Precision	Recall
1	1	0
2	0	0
3	0.5	0.035714
4	0.333333	0.035714
5	0.5	0.071429
6	0.6	0.107143
7	0.666667	0.142857
8	0.714286	0.178571
9	0.625	0.178571
10	0.555556	0.178571
11	0.6	0.214286
12	0.636364	0.25
13	0.666667	0.285714
14	0.692308	0.321429
15	0.714286	0.357143
16	0.733333	0.392857
17	0.75	0.428571
18	0.705882	0.428571
19	0.666667	0.428571
20	0.631579	0.428571
21	0.65	0.464286
22	0.666667	0.5
23	0.681818	0.535714
24	0.652174	0.535714
25	0.625	0.535714
26	0.6	0.535714
27	0.615385	0.571429
Average Precision	0.370992	N/A

Figure 17

Table of Precision, Average Precision, and Recall of Adjusted Bounding Box Predictions

Figure 18 shows the resulting plot of precision and recall. The average precision is 0.37099.

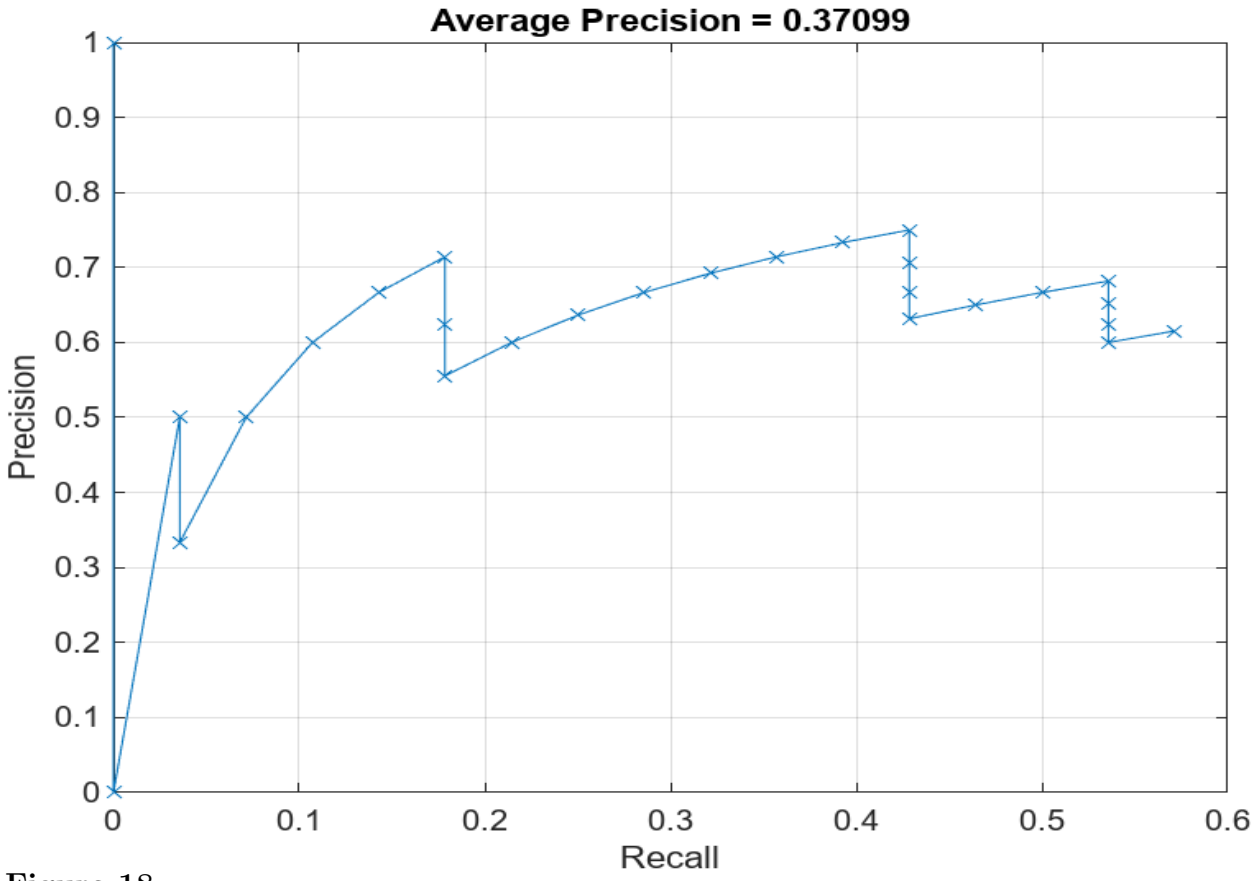

Figure 18

Plotted Recall vs Precision

To obtain a visualization of the errors in the predictions, the ground truth bounding boxes are plotted agains the adjusted predictions.

```
figure;
xlim([0, 1100]);
ylim([0, 650]);
set(gca, 'YDir', 'reverse')
hold on;

for i = 1:size(adjustedClassifiedMean,1)
    rectangle('Position', adjustedClassifiedMean(i,:), 'EdgeColor',
     'black', 'LineWidth', 1);
```

```
end

for i = 1:size(ParkingSlots,1)
    rectangle('Position', ParkingSlots(i,:), 'EdgeColor', 'r',
    'LineWidth', 1);
end
hold off;
```

Figure 19 shows the plotted bounding boxes of the predicted bounding boxes (black) and the ground truth boxes (red). As this figure shows, the predicted areas are very different from the ground truth boxes. To improve this, post-processing improvements will be done in the next section to improve the precision and recall of the predictions by adjusting the predicted bounding boxes to more accurately match the ground truth.

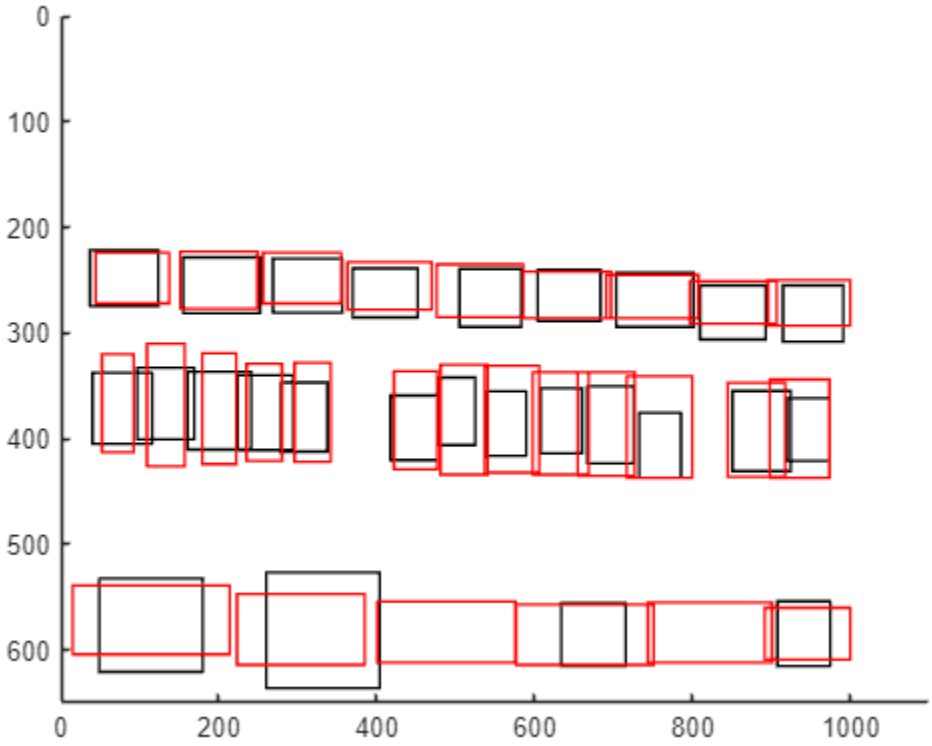

Figure 19

Plotted Bounding Boxes of Unimproved Predictions (black) Plotted Against Ground Truth Bounding Boxes (red)

Post-Processing Improvements

Plotting Bounding Box Sizes

To visualize the different bounding box sizes, the predicted bounding boxes are plotted. To ensure that the orientation of each bounding boxes are the same, the aspect ratios of each box was calculated. If it was greater than 1, the x and y coordinates are swapped, along with their widths and lengths. This effectively orients each box to be 'vertically' oriented (the length is shorter than the width for each box). The code to plot the boxes and the swaps are shown below.

```
figure;
hold on;
axis equal;
xlim([0, 200]);
ylim([0, 200]);
xlabel('Width of parking slots in pixels');
ylabel('Length of parking slots in pixels');
set(gca, 'XTick', [], 'YTick', []);

%create a copy of adjustedClassifiedMeans before rotation for next task
unrotatedClassifiedMeans=adjustedClassifiedMean

for i = 1:size(adjustedClassifiedMean, 1)
    box =adjustedClassifiedMean(i, :);
    aspectRatio = adjustedClassifiedMean(i,4) /adjustedClassifiedMean(i,3);
    if aspectRatio<1
        temp = adjustedClassifiedMean(i,3);
        adjustedClassifiedMean(i,3) =adjustedClassifiedMean(i,4);
        adjustedClassifiedMean(i,4) = temp;

        temp2=adjustedClassifiedMean(i,1);
        adjustedClassifiedMean(i,1)=adjustedClassifiedMean(i,2);
        adjustedClassifiedMean(i,2)=temp2;
        rectangle('Position', [50, 0, adjustedClassifiedMean(i,3),
         adjustedClassifiedMean(i,4)], 'EdgeColor', 'black');
    else
```

```
        rectangle('Position', [50, 0, adjustedClassifiedMean(i,3),
         adjustedClassifiedMean(i,4)], 'EdgeColor', 'black');
    end

end

hold off;
```

Figure 20 shows the resulting bounding box plots. As seen from this figure, there are a few bounding boxes that are significantly larger than the others. Observing figure 19, these two appears to be the two large boxes on the bottom left of the image.

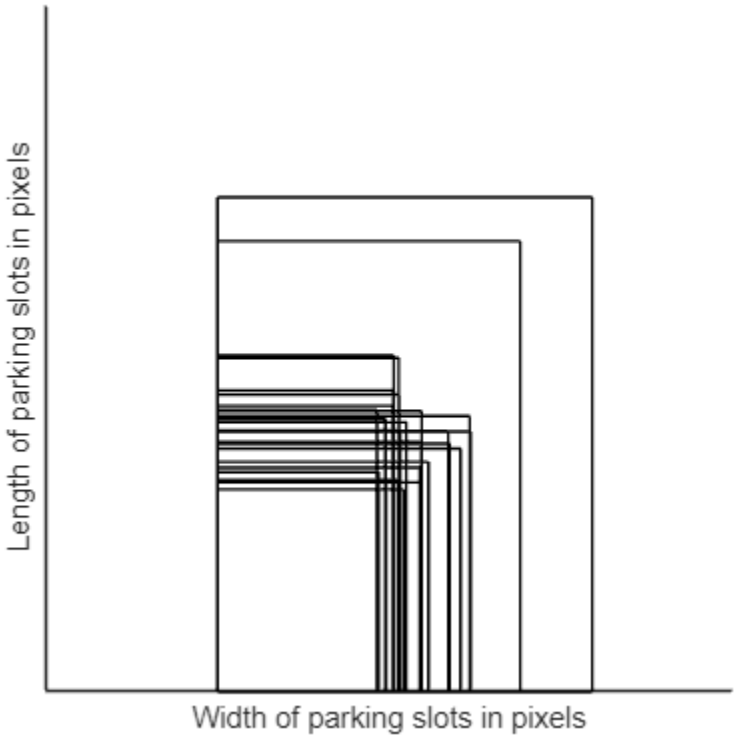

Figure 20

Plotted Bounding Boxes With Vertical Orientation

The average of these box widths and lengths are calculated to be used to post-process the predicted bounding boxes to improve the predictions.

```
averageLength = mean(adjustedClassifiedMean(:,4));
averageWidth = mean(adjustedClassifiedMean(:,3));
disp(averageLength)
disp(averageWidth)
```

The resulting average length is calculated as 79.8080 pixels and an average width of 60.3352 pixels. The screenshot of the output is in appendix C. These lengths will be used to adjust the predicted bounding box sizes in the next section.

Post-Process to Improve Predicting Bounding Box

To improve predicted bounding boxes, a few assumptions was made.

1. The width and length of each parking slot is constant, thus the average length and width calculated above will be used as the width and length of each parking space.

2. The car will take 80% of the parking space, so the detections, or the predicted bounding box length, will be extended by 25%.

The code used is based on the pseudocode provided.

```
newBoxes = zeros(size(unrotatedClassifiedMeans));
aspectMore1=zeros(size(unrotatedClassifiedMeans));
aspectLess1=zeros(size(unrotatedClassifiedMeans));

% x,y, width, length
for i = 1:length(unrotatedClassifiedMeans)
    box = unrotatedClassifiedMeans(i, :);
    aspectRatio = box(3) / box(4);
    if aspectRatio > (averageLength/averageWidth) % for horizontally longer
        deltaX = (box(3) - (averageLength * 1.25));
        deltaY = (box(4) - averageWidth);
```

```
        newBoxes(i, 1) = box(1) + (deltaX / 2);
        newBoxes(i, 2) = box(2) + (deltaY / 2);
        newBoxes(i, 3) = averageLength*1.25;
        newBoxes(i, 4) = averageWidth;

        aspectMore1(i, 1) = box(1) + (deltaX / 2);
        aspectMore1(i, 2) = box(2) + (deltaY / 2);
        aspectMore1(i, 3) = averageLength*1.25 ;
        aspectMore1(i, 4) = averageWidth;
    else % for vertically longer boxes
        deltaX = (box(3) - averageWidth);
        deltaY = (box(4) - (averageLength * 1.25));
        newBoxes(i, 1) = box(1) + (deltaX / 2);
        newBoxes(i, 2) = box(2) + (deltaY / 2);
        newBoxes(i, 3) = averageWidth;
        newBoxes(i, 4) = averageLength*1.25;

        aspectLess1(i, 1) = box(1) + (deltaX / 2);
        aspectLess1(i, 2) = box(2) + (deltaY / 2);
        aspectLess1(i, 3) = averageWidth;
        aspectLess1(i, 4) = averageLength*1.25;
    end
end
```

The resulting new boxes were then plotted, with predicted boxes with an aspect ratio greater than 1 being colored green, aspect ratios being less than 1 colored cyan, and the ground truth bounding boxes being colored red. Figure 21 shows this plot.

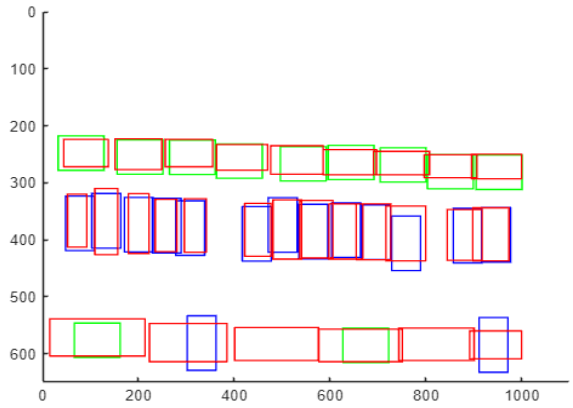

Figure 21

Predicted Bounding Boxes (blue and green) Plotted With Ground Truth (red)

A second plot was created without differing colors for different aspect ratios. Figure 22 shows this plot.

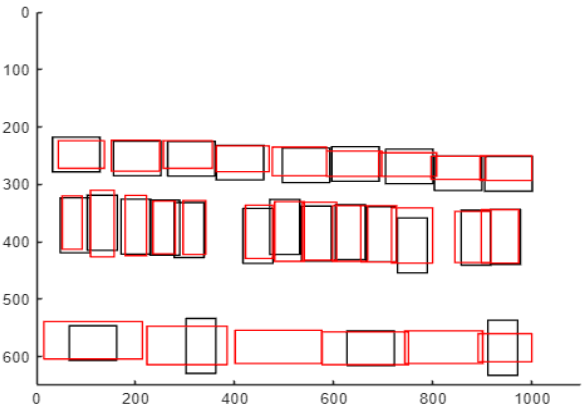

Figure 22

Predicted Bounding Boxes (black) Plotted With Ground Truth (red) Without Differentiation of Aspect Ratios

```
figure;
xlim([0, 1100]);
ylim([0, 650]);
set(gca, 'YDir', 'reverse')
hold on;
```

```
for i = 1:size(newBoxes,1)
    rectangle('Position', aspectMore1(i,:), 'EdgeColor',
     'g', 'LineWidth', 1);
end

for i = 1:size(newBoxes,1)
    rectangle('Position', aspectLess1(i,:), 'EdgeColor',
     'blue', 'LineWidth', 1);
end

for i = 1:size(ParkingSlots,1)
    rectangle('Position', ParkingSlots(i,:), 'EdgeColor',
     'r', 'LineWidth', 1);
end
hold off;
```

The recall and precisions of the new post-processed boxes were then plotted in a similar way to the recall and precision plot of the initial predicted bounding boxes. Table 23 shows the resulting precisions, recall, and average precisions of the post-processed predicted bounding boxes.

Bounding Box	Precision	Recall
1	1	0
2	1	0.035714
3	1	0.071429
4	1	0.107143
5	1	0.142857
6	1	0.178571
7	1	0.214286
8	1	0.25
9	1	0.285714
10	1	0.321429
11	1	0.357143
12	1	0.392857
13	1	0.428571
14	1	0.464286
15	1	0.5
16	0.933333	0.5
17	0.9375	0.535714
18	0.941176	0.571429
19	0.944444	0.607143
20	0.947368	0.642857
21	0.95	0.678571
22	0.952381	0.714286
23	0.954545	0.75
24	0.956522	0.785714
25	0.916667	0.785714
26	0.92	0.821429
27	0.884615	0.821429
Average Precision	0.803712	N/A

Figure 23

Table of Precision, Average Precision, and Recall of Adjusted and Post-Processed Bounding Box Predictions

After the precisions and recalls were calculated, they are plotted using the following code.

```
newDetectionResults = detectionResults;
newDetectionResults.Boxes{1} = newBoxes;

[newAveragePrecision, newRecall, newPrecision] = evaluateDetectionPrecision
(newDetectionResults, groundTruthData);

figure;
plot(newRecall, newPrecision,'-x');
xlabel('Recall');
ylabel('Precision');
title(['Average Precision = ' num2str(newAveragePrecision) ]);
grid on;
```

The resulting plot is shown in figure 24.

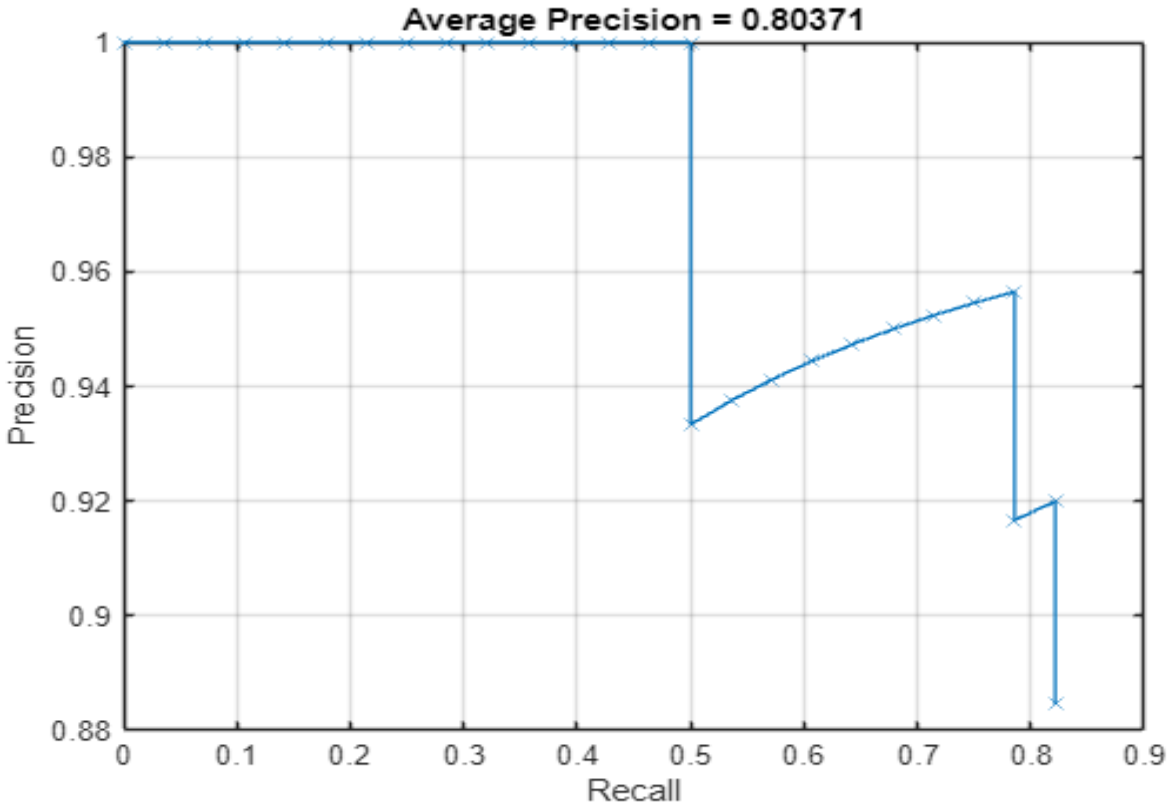

Figure 24

Plotted Recall vs Precision for Post-Processed Bounding Boxes

Discussion

Accuracy, Precision, and Recall Evaluation

Car Detection With Resnet50 CNN Model

The accuracy of the car detector model with the Resnet50 CNN model is highly accurate, as shown in figure 10, showing that the model has high accuracy and low loss on both the training and development dataset. Furthermore, it is also shown to be able to generalize well to unseen datasets, as seen in the high accuracy in figure 11.

Parking Spot Delineation With FasterRCNN Model

As seen in figure 13, the accuracy in the training was quite high, with RPN and the overall model obtaining near perfect accuracy over a few thousand iterations. However, as shown in figure 14, it is not able to generalize very well towards unseen datasets, where it was only able to detect 17 of the 28 annotated parking spaces. Adding on more data by providing various other CCTV images of Barry Street did enhance the accuracy (see figure 15), where the model was able to detect 26 of the 28 spaces.

While detection accuracy after providing the model with more data is better, the quality of the bounding boxes are not very accurate to the ground truth (see figure 19). The resulting average precision was approximately 37%, meaning that of all the instances the model predicted as a parking spot, about 37% had correct bounding box predictions. The recall also tend to range from 3.5% to 57%, which means that about 3.5% to 57% of the specific bounding box were successfully detected as parking spots, while missing the remaining percentages for that specific bounding box. As the recall increases, the general trend is that there is an increase in precision as well, with only some points having decreased precision.

To improve the precision and recall of each bounding boxes, the improvements were made. The results will be discussed in the following section.

Improvement Based on Assumptions

The average precision decreased by about 43%, which shows that the post-processed bounding boxes more accurately fit the annotated ground truth. The general trend of recall vs precision is also present in the post-processed plot (see figure 24).

To further investigate how the predicted boxes compare to the ground truth, figures 21 and 22 was created. In contrast to figure 19, the post-processed boxes more closely resemble the ground truths. However, seeing figure 21 shows that not all the parking spot orientations were identified. For example, two of the blue boxes on the bottom are vertically oriented, but the ground truth was that it should be a horizontal parking spot. This results in these predicted boxes having increased deviation from the ground truth. The bottom left parking spot was also much smaller than the others in the same row, which could be attributed to the image being cut off at that point, as seen in figure 14.

For the vertically oriented predicted boxes, it correctly predicted all the the vertical parking spaces, although it did misclassify two horizontal boxes as vertical as mentioned above. However, for some of the boxes, there is a slight offset in the x and y coordinates, which causes increased errors.

Overall, the improvement through post-processing increaases the quality of predictions, through correctly identifiying orientations and adjusting the predicted bounding box sizes to be more accurate. However, there are still some problems with the assumptions. For example, the bottom row of Barry Street has a much larger parking spot thatn other areas, which does cause increased errors when only depending on average length and width. Furthermore, the intial predictions are also influential in identifying the correct aspect ratio of each bounding box (e.g., if the predicted width and length had an incorrect aspect ratio, the preprocessed boxes would also have incorrect aspect ratios). Nevertheless, using the assumptions and post-processing based on these assumptions does increase the average precision of the predictions.

Challenges and Shortcomings

One challenge in this report is correctly identifying aspect ratios. As stated above, due to the way the predictions were used in post-processing, the aspect ratios for some bounding boxes were not correctly identified. Generalizability is a core aspect in model training, and if a model was overfitted on certain datasets, they might not generalize well to unseen images. As such, although the assumptions and post-processing did help to improve some evaluation metrics, further post-processing could run the risk of overfitting. For example, if each bounding box was slowly shifted into its correct space and orientation by checking it against its corresponding ground truth box. This would greatly improve the metrics, but would also be an issue of overfitting.

Another challenge is interpreting the provided pseudocode as it was quite difficult to visualize the change. However, once it was understood that it was meant to account for horizontal and vertical orientations of parking, the resulting transformations were easier to visualize.

Scopes of Improvement

To improve the report, the trained model could include training on the parking space orientations as an output. By directly training the output class of orientation itself, this could improve accuracy in post-processing. This would be in contrast to what this report has done, which is to calculate the aspect ratios based on predicted bounding boxes, which could cause errors to propagate. For instance, as described above, vertically oriented ground truth boxes are sometimes identified as horizontally oriented, which is an example of the error propagation (e.g., because the bounding box was more horizontal, the post-processing step also assumes it is horizontal regardless of the ground truth).

Another improvement could be to have more sophisticated assumptions. The provided assumptions are quite general, but as discussed above, does not apply to all parking spaces. Certain additional assumptions such as middle is always vertically oriented

or the bottom row is always wider could further improve the evaluation metrics. However, this may not always be desired as it could cause overfitting. However, if the goal was to obtain accurate predictions for one image to be used for some other purpose, this could be beneficial.

One final improvement is that there could be more sophisticated models, in which post-processed models could be fed back into the model to increase the generalizability of the model. However, designing such a model is outside the scope of this report and is a large project to undertake by itself.

Conclusions and Future Directions

In conclusion, I have successfully investigated how neural networks and machine learning are applied in the field of computer vision. I have used pre-trained Restnet50 CNN and FasterRCNN models to perform car detection and delineation respectively. Furthermore, the performance of these models have been evaluated and visualized. After training, the FasterRCNN model was further fine-tuned in two ways. First was by providing it more data and averaging the predicted bounding boxes. The second was post-processing by using certain assumptions based on averaging length and width of all the bounding boxes. The results of post-processing was that the average precision decreased by about 43%, and visual inspection of the plotted bounding boxes as seen in figure 21 and 22 shows that there was an improvement in the predicted bounding box areas to be closer to the ground truth. However, there are still issues in the predictions of bounding boxes due to two main points, that the assumption does not hold for all parking spaces and that errors are propagated from initial predictions due to the calculations of aspect ratios.

Future work can build on this report by implementing some of the suggestions in the scope of improvement section. While some of these are difficult and would require a large amount of work, such as the creation of a more sophisitcated model, some others can greatly improve the evaluation metrics, either by increasing assumptions or attempting to overfit the predictions based on the ground truth. However, these should be carefully considered on whether or not the risk of overfitting is suitable for the purposes of the desired study aim.

Nevertheless, this report has implemented all the steps provided in the specification and has achieved the aim of improving predictions provided by the provided models using various procedures.

References

Chai, J., Zeng, H., Li, A., & Ngai, E. W. (2021). Deep learning in computer vision: A critical review of emerging techniques and application scenarios. *Machine Learning with Applications*, *6*, 100134. https://doi.org/https://doi.org/10.1016/j.mlwa.2021.100134

El-Nouby, A., Neverova, N., Laptev, I., & Jégou, H. (2021). Training vision transformers for image retrieval.

Esteva, A., Chou, K., Yeung, S., Naik, N., Madani, A., Mottaghi, A., Liu, Y., Topol, E., Dean, J., & Socher, R. (2021). Deep learning-enabled medical computer vision. *npj Digital Medicine*, *4*, 1–10. https://doi.org/https://doi.org/10.1038/s41746-020-00376-2

Girshick, R. (2015). Fast r-cnn. *2015 IEEE International Conference on Computer Vision (ICCV)*, 1440–1448. https://doi.org/10.1109/ICCV.2015.169

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition.

Khan, A. I., & Al-Habsi, S. (2020). Machine learning in computer vision [International Conference on Computational Intelligence and Data Science]. *Procedia Computer Science*, *167*, 1444–1451. https://doi.org/https://doi.org/10.1016/j.procs.2020.03.355

Rathnayake P, W., D, G. D., Punchihewa G, A., Anjana G, N., Suriya Kumari, P. K., & Samarakoon, U. (2022). Certimart: Use computer vision to digitize and automate supermarket with fruit quality measuring and maintaining. *2022 4th International Conference on Advancements in Computing (ICAC)*, 36–41. https://doi.org/10.1109/ICAC57685.2022.10025119

Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster r-cnn: Towards real-time object detection with region proposal networks.

Shukla, L. (2019). Designing your neural networks. *Towards Data Science*. https://towardsdatascience.com/designing-your-neural-networks-a5e4617027ed

Appendix

Appendix A - Table of the averaged bounding box means along with the average width and length.

Bounding Box	X Coordinate	Y Coordinate	Width(pixels)	Length(pixels)
1	558.5679651	417.1488864	59.2047558	61.02432155
2	364.6099463	397.8601191	70.86074114	70.83589295
3	844.6466823	300.9213637	98.01382723	51.35044642
4	950.7499573	312.8576748	83.30618791	51.14232658
5	1056.078143	312.946587	76.18511131	53.17958154
6	295.8858469	286.3368602	97.23646982	52.8677588
7	510.8260085	296.736689	82.10477614	46.52019309
8	745.3094125	297.8267577	79.47927644	49.01446097
9	645.94278	297.1745139	78.51497201	55.04155829
10	301.7773088	394.7287189	80.30609606	73.55901443
11	419.8101246	404.7791311	58.96754556	65.33890048
12	409.7346948	287.5226484	87.84018917	51.32338297
13	177.8481516	279.504489	86.57217649	52.93628292
14	180.8841018	395.6830783	75.92977213	67.24473521
15	748.6458317	410.2037168	52.60987676	61.4943903
16	189.4892875	590.2971429	131.2677248	88.27477253
17	1060.972933	419.8334653	54.28303066	58.95344486
18	619.0400256	399.9973782	46.91044448	64.02006383
19	681.5340922	413.2001524	49.18197295	61.04869702
20	991.9946958	412.86345	73.75539478	75.79877977
21	400.9065423	584.642299	144.0338792	109.2483771
22	1048.985918	611.7449403	66.98693741	61.52769688
23	807.8181278	408.4250569	59.42851196	72.8044273
24	774.9789102	613.5086241	81.78166498	59.51536058
25	237.6583541	390.8823923	72.33478954	67.72445576
26	873.6361372	433.4169055	53.0676195	61.77040696
Average	N/A	N/A	76.92937478	63.21383573

Figure 25

Original Unadjusted Bounding Box Means

Appendix B - Table of adjusted bounding box means along with the average width and length

Bounding Box	X Coordinate	Y Coordinate	Width(pixels)	Length(pixels)
1	417.5679651	359.1488864	59.2047558	61.02432155
2	223.6099463	339.8601191	70.86074114	70.83589295
3	703.6466823	242.9213637	98.01382723	51.35044642
4	809.7499573	254.8576748	83.30618791	51.14232658
5	915.0781426	254.946587	76.18511131	53.17958154
6	154.8858469	228.3368602	97.23646982	52.8677588
7	369.8260085	238.736689	82.10477614	46.52019309
8	604.3094125	239.8267577	79.47927644	49.01446097
9	504.94278	239.1745139	78.51497201	55.04155829
10	160.7773088	336.7287189	80.30609606	73.55901443
11	278.8101246	346.7791311	58.96754556	65.33890048
12	268.7346948	229.5226484	87.84018917	51.32338297
13	36.84815156	221.504489	86.57217649	52.93628292
14	39.88410176	337.6830783	75.92977213	67.24473521
15	607.6458317	352.2037168	52.60987676	61.4943903
16	48.48928755	532.2971429	131.2677248	88.27477253
17	919.9729329	361.8334653	54.28303066	58.95344486
18	478.0400256	341.9973782	46.91044448	64.02006383
19	540.5340922	355.2001524	49.18197295	61.04869702
20	850.9946958	354.86345	73.75539478	75.79877977
21	259.9065423	526.642299	144.0338792	109.2483771
22	907.9859182	553.7449403	66.98693741	61.52769688
23	666.8181278	350.4250569	59.42851196	72.8044273
24	633.9789102	555.5086241	81.78166498	59.51536058
25	96.65835409	332.8823923	72.33478954	67.72445576
26	732.6361372	375.4169055	53.0676195	61.77040696
Average	N/A	N/A	76.92937478	63.21383573

Figure 26

Adjusted Bounding Box Means

Appendix C - Screenshot of the average width and length of rotated and adjusted predicted bounding boxes

```
averageLength = mean(adjustedClassifiedMean(:,4));
averageWidth = mean(adjustedClassifiedMean(:,3));
disp(averageLength)
```

```
    79.8080
```

```
disp(averageWidth)
```

```
    60.3352
```

Figure 27

Adjusted Predicted Bounding Boxes Width and Length

Appendix D - Complete Code Used For Tasks 4 and 5 of the Assignment

```
% Task 4

% x(1), y(2), length(3), width(4)
disp(classifiedMean)
adjustedClassifiedMean=classifiedMean
adjustedClassifiedMean(:,1)=adjustedClassifiedMean(:,1)-141
adjustedClassifiedMean(:,2)=adjustedClassifiedMean(:,2)-58

detectionResults = table('Size',[1,2], ...
    'VariableTypes',{'cell','cell'}, ...
    'VariableNames', {'Boxes','Scores'})
detectionResults.Boxes{1} = adjustedClassifiedMean;
detectionResults.Scores{1} = classifiedScoreMean;
disp(detectionResults)
load("FinalCode\GroundTruthBarryStreet.mat")
groundTruthData = table('Size',[1,1], ...
    'VariableTypes',{'cell'}, ...
    'VariableNames', {'Boxes'})
groundTruthData.Boxes{1} = ParkingSlots
[averagePrecision, recall, precision] =
evaluateDetectionPrecision(detectionResults, groundTruthData)
figure;
plot(recall, precision,'-x');
xlabel('Recall');
ylabel('Precision');
```

```
title(['Average Precision = ' num2str(averagePrecision) ]);
grid on;
figure;
xlim([0, 1100]);
ylim([0, 650]);
set(gca, 'YDir', 'reverse')
hold on;

for i = 1:size(adjustedClassifiedMean,1)
    rectangle('Position', adjustedClassifiedMean(i,:),
    'EdgeColor', 'black', 'LineWidth', 1);
end

for i = 1:size(ParkingSlots,1)
    rectangle('Position', ParkingSlots(i,:), 'EdgeColor',
    'r', 'LineWidth', 1);
end
hold off;


%Task 5

figure;
hold on;
axis equal;
xlim([0, 200]);
ylim([0, 200]);
```

```
xlabel('Width of parking slots in pixels');
ylabel('Length of parking slots in pixels');
set(gca, 'XTick', [], 'YTick', []);

%create a copy of adjustedClassifiedMeans before rotation for next task
unrotatedClassifiedMeans=adjustedClassifiedMean

for i = 1:size(adjustedClassifiedMean, 1)
    box =adjustedClassifiedMean(i, :);
    aspectRatio = adjustedClassifiedMean(i,4)
    /adjustedClassifiedMean(i,3);
    if aspectRatio<1
        temp = adjustedClassifiedMean(i,3);
        adjustedClassifiedMean(i,3) =adjustedClassifiedMean(i,4);
        adjustedClassifiedMean(i,4) = temp;

        temp2=adjustedClassifiedMean(i,1);
        adjustedClassifiedMean(i,1)=adjustedClassifiedMean(i,2);
        adjustedClassifiedMean(i,2)=temp2;
        rectangle('Position', [50, 0, adjustedClassifiedMean(i,3),
        adjustedClassifiedMean(i,4)], 'EdgeColor', 'black');
    else
        rectangle('Position', [50, 0, adjustedClassifiedMean(i,3),
        adjustedClassifiedMean(i,4)], 'EdgeColor', 'black');
    end

end
```

```
hold off;
averageLength = mean(adjustedClassifiedMean(:,4));
averageWidth = mean(adjustedClassifiedMean(:,3));
disp(averageLength)
disp(averageWidth)

newBoxes = zeros(size(unrotatedClassifiedMeans));
aspectMore1=zeros(size(unrotatedClassifiedMeans));
aspectLess1=zeros(size(unrotatedClassifiedMeans));

% x,y, width, length
for i = 1:length(unrotatedClassifiedMeans)
    box = unrotatedClassifiedMeans(i, :);
    aspectRatio = box(3) / box(4);
    if aspectRatio > (averageLength/averageWidth) % for vertically longer
        deltaX = (box(3) - (averageLength * 1.25));
        deltaY = (box(4) - averageWidth);
        newBoxes(i, 1) = box(1) + (deltaX / 2);
        newBoxes(i, 2) = box(2) + (deltaY / 2);
        newBoxes(i, 3) = averageLength*1.25;
        newBoxes(i, 4) = averageWidth;

        aspectMore1(i, 1) = box(1) + (deltaX / 2);
        aspectMore1(i, 2) = box(2) + (deltaY / 2);
        aspectMore1(i, 3) = averageLength*1.25 ;
        aspectMore1(i, 4) = averageWidth;
```

```
    else % for horizontally longer boxes
        deltaX = (box(3) - averageWidth);
        deltaY = (box(4) - (averageLength * 1.25));
        newBoxes(i, 1) = box(1) + (deltaX / 2);
        newBoxes(i, 2) = box(2) + (deltaY / 2);
        newBoxes(i, 3) = averageWidth;
        newBoxes(i, 4) = averageLength*1.25;

        aspectLess1(i, 1) = box(1) + (deltaX / 2);
        aspectLess1(i, 2) = box(2) + (deltaY / 2);
        aspectLess1(i, 3) = averageWidth;
        aspectLess1(i, 4) = averageLength*1.25;
    end
end

figure;
xlim([0, 1100]);
ylim([0, 650]);
set(gca, 'YDir', 'reverse')
hold on;


for i = 1:size(newBoxes,1)
    rectangle('Position', aspectMore1(i,:), 'EdgeColor',
     'g', 'LineWidth', 1);
end
```

```
for i = 1:size(newBoxes,1)
    rectangle('Position', aspectLess1(i,:), 'EdgeColor',
     'blue', 'LineWidth', 1);
end

for i = 1:size(ParkingSlots,1)
    rectangle('Position', ParkingSlots(i,:), 'EdgeColor',
    'r', 'LineWidth', 1);
end
hold off;

figure;
xlim([0, 1100]);
ylim([0, 650]);
set(gca, 'YDir', 'reverse')
hold on;


for i = 1:size(newBoxes,1)
    rectangle('Position', newBoxes(i,:), 'EdgeColor',
     'black', 'LineWidth', 1);
end

for i = 1:size(ParkingSlots,1)
    rectangle('Position', ParkingSlots(i,:), 'EdgeColor',
    'r', 'LineWidth', 1);
end
```

```
hold off;


newDetectionResults = detectionResults;
newDetectionResults.Boxes{1} = newBoxes;

[newAveragePrecision, newRecall, newPrecision] =
 evaluateDetectionPrecision(newDetectionResults, groundTruthData);

figure;
plot(newRecall, newPrecision,'-x');
xlabel('Recall');
ylabel('Precision');
title(['Average Precision = ' num2str(newAveragePrecision) ]);
grid on;
```