

A group of cyclists is riding on a city street. The cyclists are wearing helmets and various colored jackets (yellow, blue, red). They are riding on a designated bike lane marked with white lines and a bicycle symbol. In the background, the United States Capitol building is visible, along with other city buildings and trees. The sky is overcast.

# Identifying Bike Lane Obstacles Using Deep Learning and Object Detection

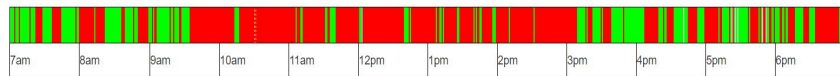
Brendan Freehart and Dan Bernstein  
Data Science DC  
March 14, 2019

# Parking Dirty Data Set

Is this bike lane blocked?



Parking Dirty Results - Crystal Dr. @ MidBlock B on 9/28/2016



Hover over a point on the graph above to see the photo for that minute.



10:31AM

Legend:

- BLOCKED (Red bar)
- CLEAR (Green bar)
- DATA NOT YET FINAL (Grey bar)

Analysis

|                                     |         |
|-------------------------------------|---------|
| Minutes of Data                     | 709     |
| Minutes Lane was Blocked            | 457     |
| Minutes Lane was Clear              | 245     |
| Minutes Data is Incomplete / Unsure | 7       |
| % of Minutes Blocked                | 64.46 % |

- > 6,000 tagged images from multiple Arlington County traffic cameras
- Perfect for automation using computer vision

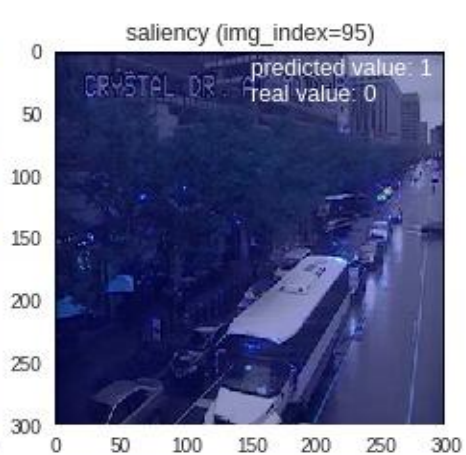
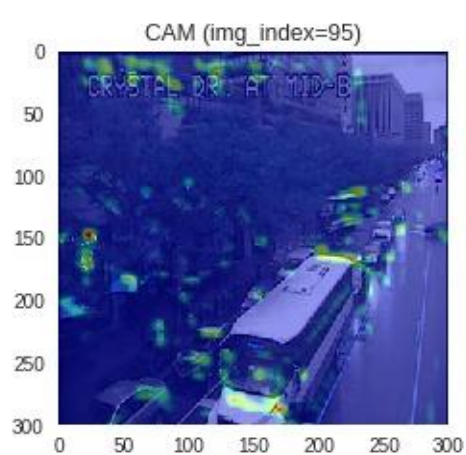
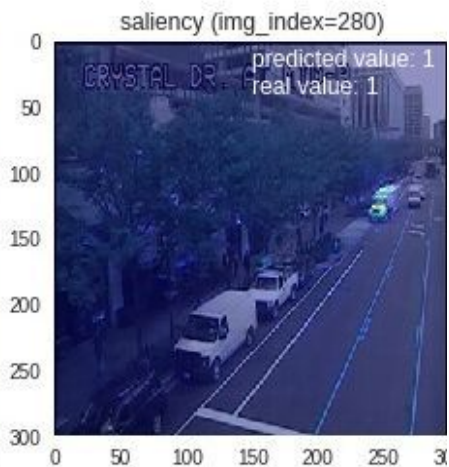
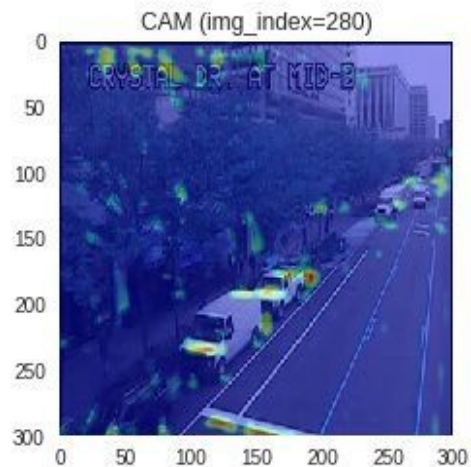
~85% test accuracy

# Image Classification Model

```
model = models.Sequential()  
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(300, 300, 3)))  
model.add(layers.MaxPooling2D((2, 2)))  
model.add(layers.Conv2D(64, (3, 3), activation='relu'))  
model.add(layers.MaxPooling2D((2, 2)))  
model.add(layers.Conv2D(128, (3, 3), activation='relu'))  
model.add(layers.MaxPooling2D((2, 2)))  
model.add(layers.Conv2D(128, (3, 3), activation='relu'))  
model.add(layers.MaxPooling2D((2, 2)))  
model.add(layers.Flatten())  
model.add(layers.Dense(128, activation='relu'))  
model.add(layers.Dense(1, activation='sigmoid'))  
  
## Compiler Includes Optimizer, and Learning Rate (LR), and Metrics  
model.compile(loss='binary_crossentropy',  
optimizer=optimizers.RMSprop(lr=1e-4), metrics=['acc'])
```



# Grad-CAM and Saliency Maps



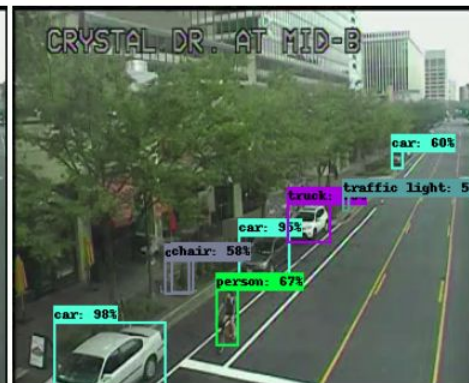
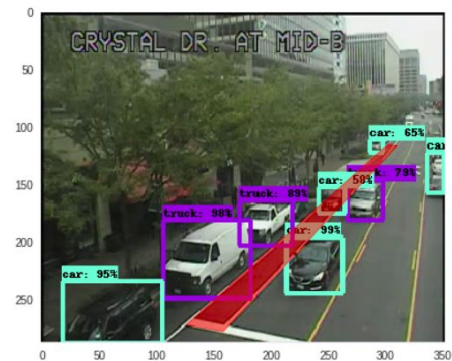
~81% accuracy

# Object Detection Model

- Faster RCNN algorithm
- Inception ResNet architecture
- Common Objects in Context (COCO) dataset

Sensitivity analysis for optimal overlap

| obstacle    | correct | correct_pct |
|-------------|---------|-------------|
| <chr>       | <dbl>   | <dbl>       |
| combined    | 3144    | 81.8        |
| 25%         | 3128    | 81.4        |
| 20%         | 3112    | 81          |
| 30%         | 3093    | 80.5        |
| centerPoint | 3084    | 80.3        |
| 35%         | 3069    | 79.9        |
| 15%         | 2969    | 77.3        |
| 40%         | 2961    | 77.1        |
| 10%         | 2736    | 71.2        |
| 45%         | 2721    | 70.8        |
| 50%         | 2431    | 63.3        |



# Next Steps

- Settle on a generalizable approach
- Create REST API
  - TensorFlow Serving
  - Flask
  - Heroku
- Streaming video with YOLO



[github.com/bfraiche/parkingdirty](https://github.com/bfraiche/parkingdirty)