Traffic Optimization Using Machine Learning

MAKING INTERSECTIONS SMARTER TO REDUCE DRIVER FRUSTRATION

JOE KESSLER – PRINCIPAL SOFTWARE DEVELOPER

Heavy Traffic Sucks

- Very frustrating at times
  - Leads to road rage
  - Makes us late to the dog groomer
- Ever get “all the red lights”? It really does matter
  - Travel time
  - Sense of victimhood
  - Harder on vehicles
- Increases fuel consumption and therefore cost

Reasons for Annoying Traffic

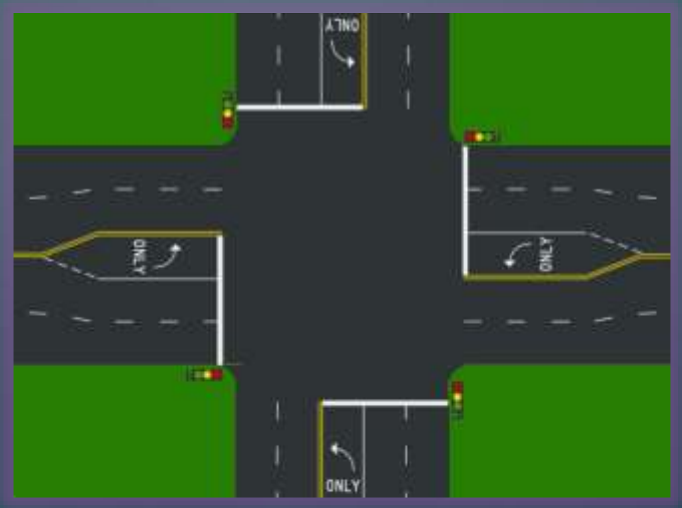
- Volume, obviously
- Overly simple traffic control signals
  - Often sensor based with timer component
  - Sometimes only timers
- Rubber banding
  - Getting "out of sync" with lights causes stop-and-go

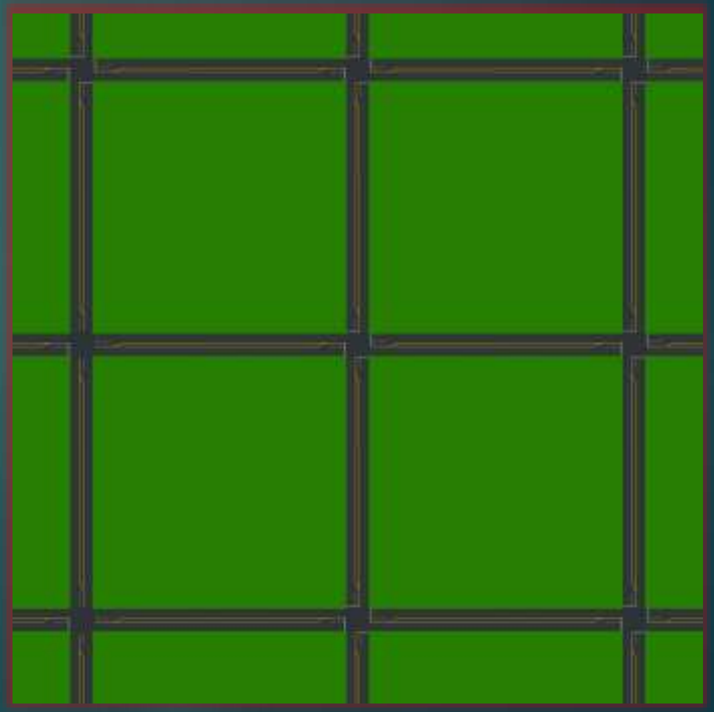
What Can We Do to Help?

- Make intersections smarter
  - Control traffic more like a person would
  - Incorporate “common sense”
- Concepts
  - Avoid unnecessary red lights
  - Retain traffic momentum where possible
  - Reduce rubber banding

Simulated Neighborhood

- Simple "city" grid interconnected by lights
- Each agent/component with independent logic
  - Intersections (Stock)
  - Intersections (Enhanced/ML)
  - Cars
  - Left turn lanes
  - Sensors
  - Cameras
  - Etc…





Simulation Characteristics

- The environment simulates
  - Intersection logic
  - Typical car/driver behavior and reactions
  - Traffic volume at a variety of times,
  - Sensors at each stop line, including distinct left lane sensors
  - Speed limits
  - Etc…
- **Virtual cameras to provide images of oncoming traffic**

Driver Behavior

- Self-contained agents (disconnected from intersection logic, etc)
- All follow same rules - there is no road rage!
- Have a “GPS” telling them the route at start of travel
  - That is, travel directions are not fixed
- Can turn left or right at intersections
- Max speed is 40 MPH for entire town but doesn’t have to be

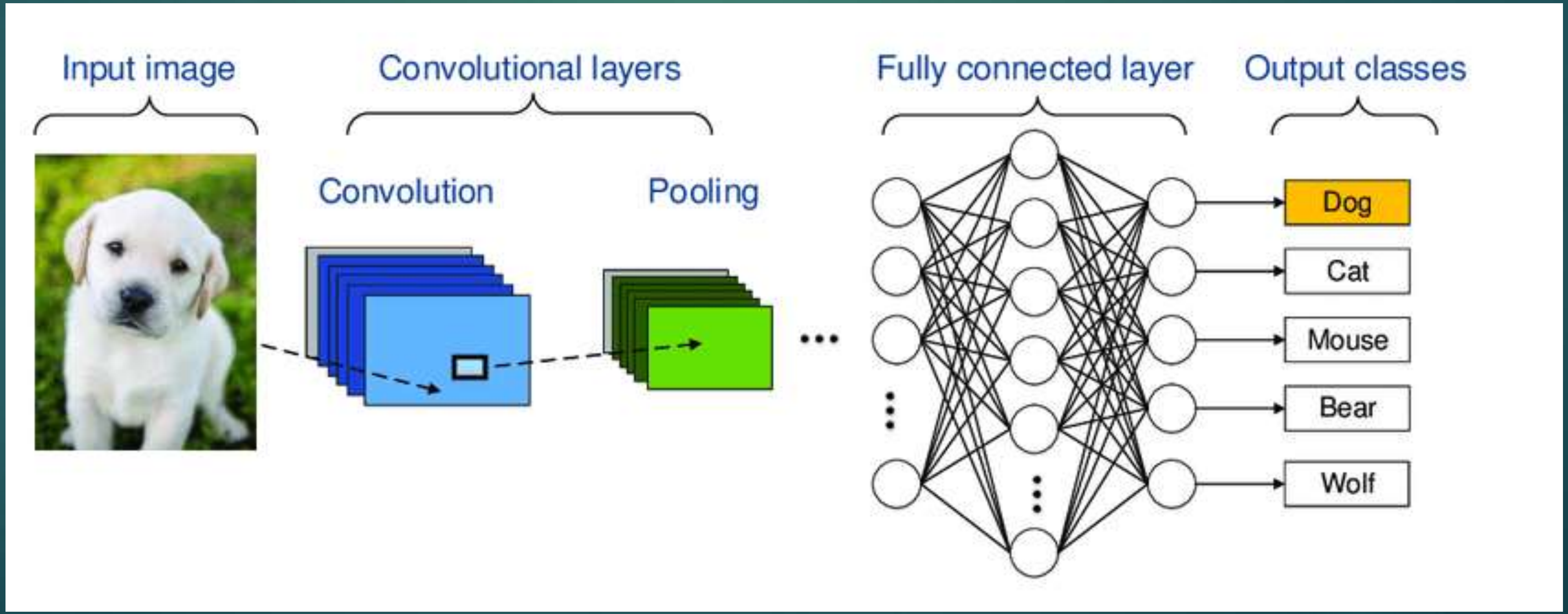
Technical Pieces

- Simulator (Python)
- CNN/ML model training (Python/Keras/TensorFlow)
- Training Data Generator (C#/WinForms)
- Traffic Visualizer (C#/WinForms)
- Traffic Camera WS (C#/ASP.NET)

- Runtime hardware
  - 12^{th} Gen Intel Core i7-12700K
  - 128GB DDR4-3600
  - Nvidia RTX 3090ti (24GB onboard)

CNN Construction

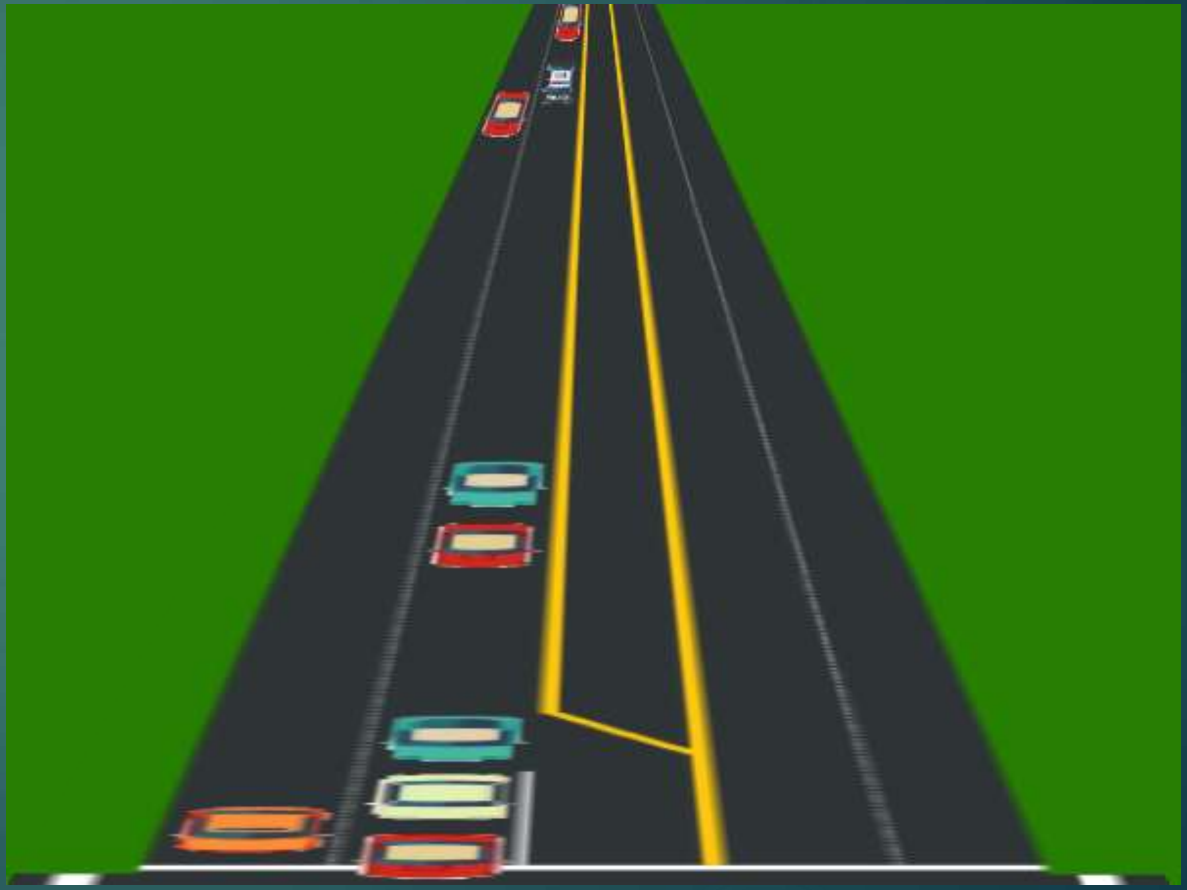
- CNN learns how to translate an image into usable output
- Can classify (dog, cat, etc) or regress (produce values)
- In our case, we want to count/locate cars



What Does CNN Buy Us?

- The CNN can process this image and understand
  - How many cars
  - Rough idea of distances
  - Like having sensors on the entire road segment
- Combined with stop line sensors
  - We get a decent idea of situation
- From there all that is required is some simple logic

(Sample CameraWS image)



Generating Data via Random Sampling

- Training a neural network requires lots of data

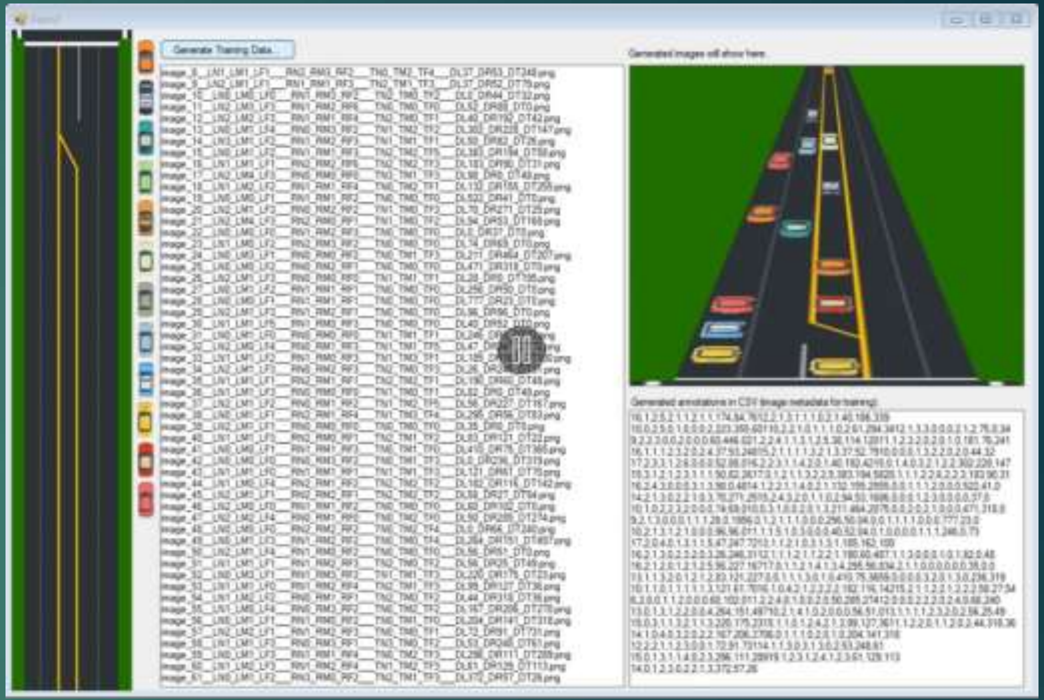
- Generate annotated sample for a neural net
  - Create image from random traffic data (interpretation -> image)
  - Annotate automatically

- Neural net learns to perform inverse (image -> interpretation)

CNN Training Data Generator

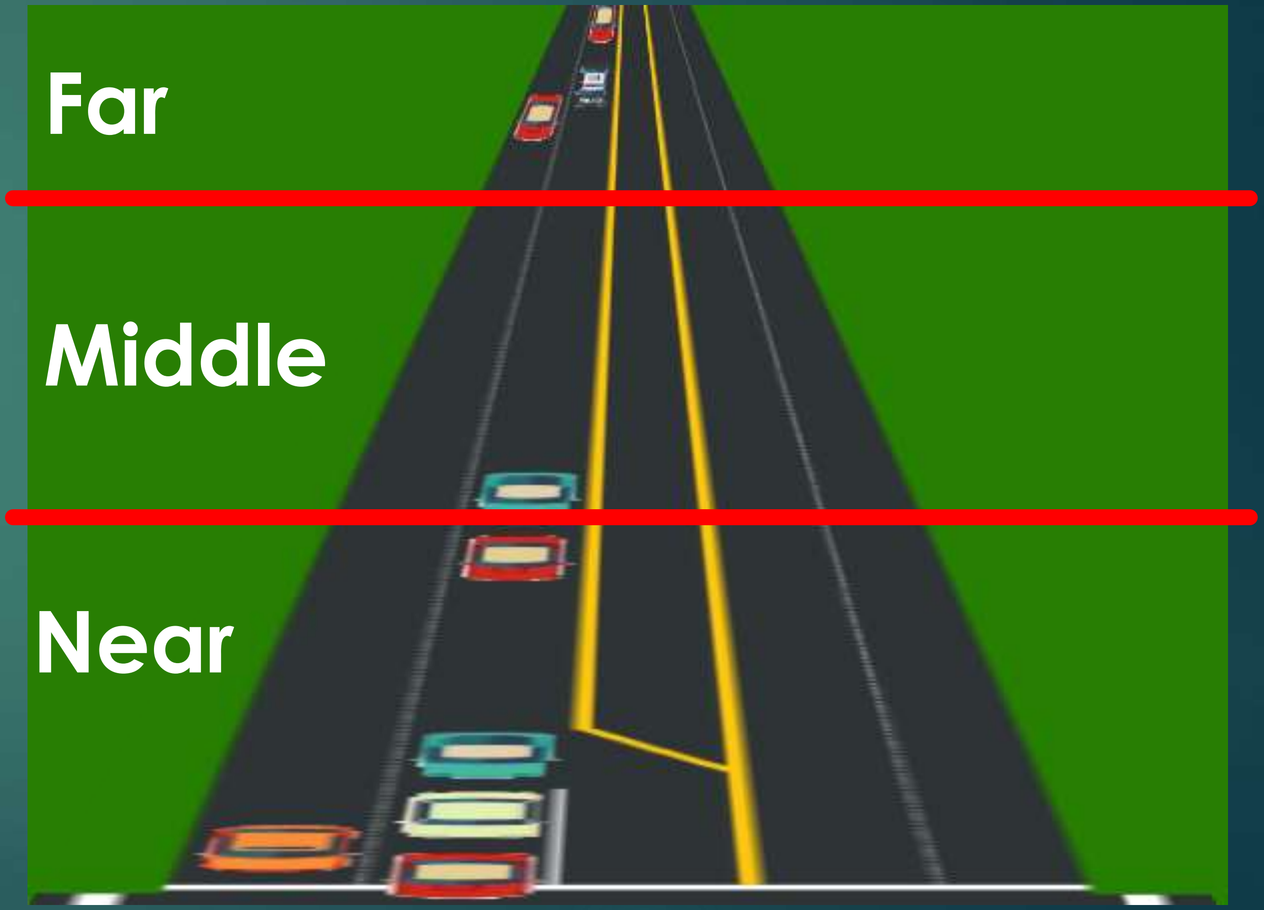
- Generates and annotates sample traffic images

- For training CNN to understand camera frame captures



Distance Zones

- The CNN can see rough distance of cars via “zones”
  - A single frame is processed by the CNN and assigned to these zones
  - Zones exist for each of three lanes (therefore 9 regions)
- CNN understands perspective
- Actual intersection logic is extremely simple



CNN Model in Keras (Python)

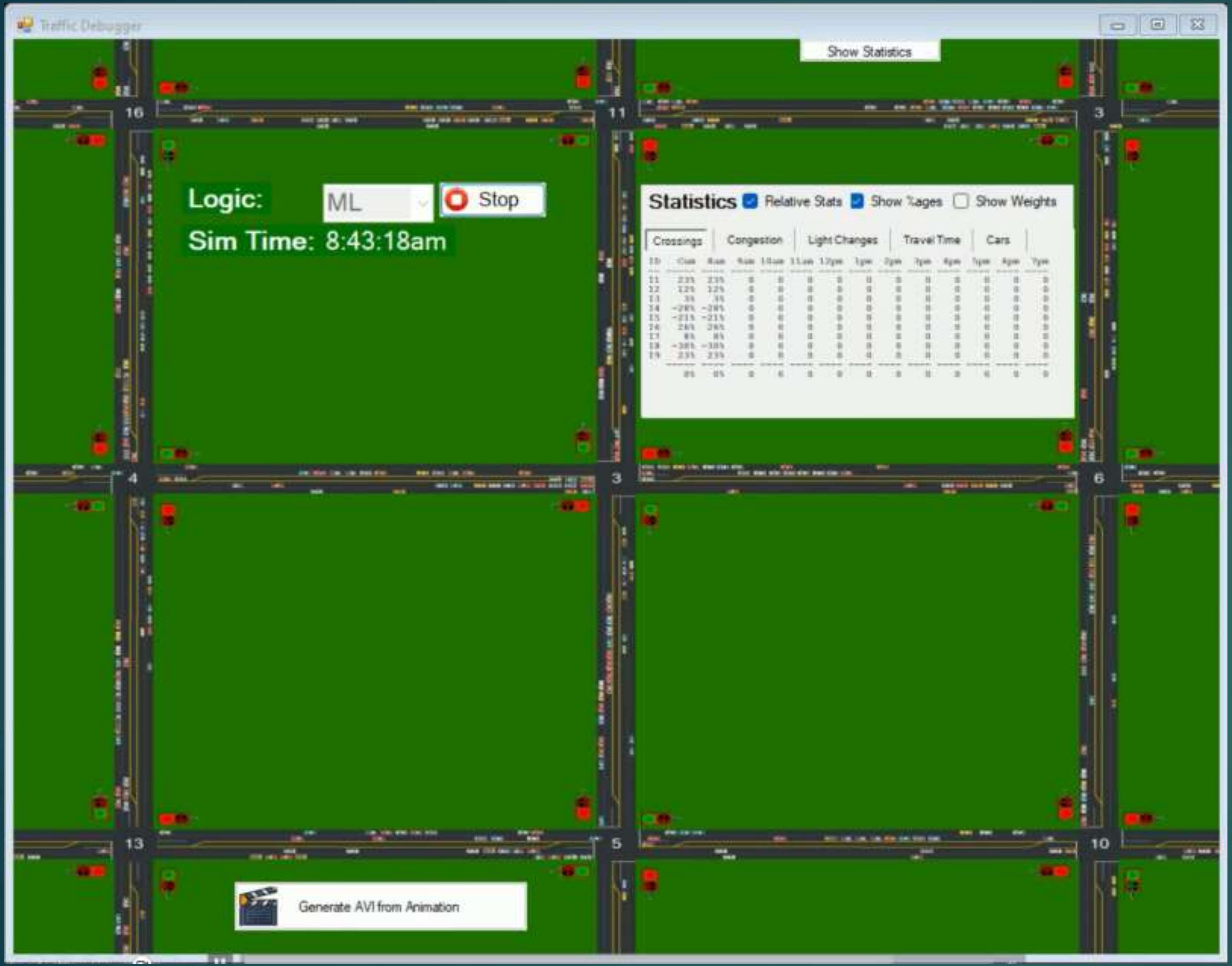
Training on Roughly 20,000 Sample Images

```
# Load up the training data.
data, labels = myutils.load_training_data(outputPath)

# Partition the data into training and testing splits using 75% of
# the data for training and the remaining 25% for testing
(trainX, testX, trainY, testY) = train_test_split(data, labels, test_size=0.25, random_state=42)

# Set up the CNN model.  Sometimes layers/settings are trial and error.
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1), activation='relu',
input_shape=imageShape))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(32, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(9, activation='sigmoid’)) # Trains on 9 “zones” on 3 lanes.


# Train the model.  Yoiks!
model.compile(optimizer='rmsprop', loss='mean_squared_error', metrics=['accuracy'])
model.fit(trainX, trainY, batch_size=1, epochs=250, verbose=1)
```

Applying the CNN in Simulator

Pretty easy once we have all the pieces!

```
# This code is going images in each direction from the perspective of the
# intersection.  It will then use the CNN model to decode it into usable data.
model_cnn = model_functions.load_camera_model(self.global_ctx, False)

# Capture the images in each oncoming direction via CameraWS.  This accesses
# four cameras mounted at the intersection.
predictions_by_oncoming_direction = { 'north': None, 'south': None, 'east': None, 'west': None };

images_by_oncoming_direction = capture_image(myconstants.camera_ws_url)

# Convert each image to usable values.
for oncoming_direction in images_by_oncoming_direction:
    image_for_this_direction = images_by_oncoming_direction[oncoming_direction]

    # Prediction (camera image -> data) from the trained CNN model.
    predicted_data_from_image = model_cnn.predict(direction, verbose = 0)

    # Wrap it up nicely.
    traffic_scenario = TrafficScenario(self, predicted_frustration_level, road_lanes, self.global_ctx)
    predictions_by_oncoming_direction[orientation] = traffic_scenario

return predictions_by_oncoming_direction
```


Traffic Debugger
Show Statistics
16
11
3
Logic:
ML
Stop
Sim Time: 8:43:18am
Statistics
Relative Stats
Show %ages
Show Weights
Crossings
Congestion
Light Changes
Travel Time
Cars
4
3
6
13
5
10
Generate AVI from Animation

(Zoomed)


1

Results – Delta % from Stock

Crossings | Congestion | Light Changes | Travel Time | Cars

Crossings

ID	Cum	8am	9am	10am	11am	12pm	1pm	2pm	3pm	4pm	5pm	6pm	7pm
I1	0%	3%	-4%	-4%	-15%	-9%	0%	1%	18%	-8%	23%	1%	3%
I2	7%	5%	-15%	34%	10%	22%	4%	-2%	-1%	25%	2%	5%	9%
I3	13%	-4%	5%	13%	27%	36%	10%	16%	57%	-1%	37%	5%	-14%
I4	-15%	-4%	-9%	-26%	-13%	-32%	-14%	-34%	-9%	-19%	-14%	1%	-12%
I5	-6%	-9%	-4%	-3%	-27%	-14%	14%	-22%	-1%	-26%	-9%	3%	23%
I6	-7%	-12%	1%	-21%	-29%	15%	-35%	3%	0%	4%	-3%	-3%	-5%
I7	3%	-8%	29%	-6%	38%	3%	-12%	21%	-2%	10%	7%	-12%	-6%
I8	-6%	12%	-11%	-21%	-13%	-13%	26%	-17%	-2%	-25%	-12%	16%	-10%
I9	10%	-7%	12%	18%	25%	4%	13%	14%	23%	-3%	23%	-7%	13%
	0%	-2%	0%	-3%	-2%	-1%	-1%	-3%	6%	-6%	4%	1%	0%

Crossings | Congestion | Light Changes | Travel Time | Cars

Light Changes

ID	Cum	8am	9am	10am	11am	12pm	1pm	2pm	3pm	4pm	5pm	6pm	7pm
I1	-44%	-42%	-39%	-50%	-36%	-61%	-31%	-50%	-35%	-42%	-50%	-44%	-44%
I2	-42%	-48%	-44%	-46%	-53%	-36%	-22%	-24%	-61%	-50%	-35%	-36%	-50%
I3	-51%	-43%	-38%	-56%	-46%	-52%	-53%	-40%	-58%	-60%	-64%	-50%	-54%
I4	-35%	-40%	-48%	-42%	-21%	-46%	-25%	-26%	-33%	-50%	-26%	-42%	-18%
I5	-38%	-42%	-33%	-30%	-30%	-56%	-46%	-39%	-39%	-29%	-41%	-42%	-20%
I6	-34%	-31%	-55%	-32%	-14%	-33%	-42%	-33%	-26%	-39%	-42%	-30%	-30%
I7	-43%	-54%	-36%	-50%	-48%	-35%	-37%	-41%	-20%	-55%	-50%	-38%	-46%
I8	-39%	-53%	-42%	-18%	-48%	-30%	-25%	-40%	-40%	-44%	-35%	-48%	-46%
I9	-43%	-53%	-42%	-53%	-34%	-48%	-35%	-40%	-46%	-38%	-48%	-44%	-32%
	-41%	-45%	-42%	-42%	-37%	-45%	-35%	-37%	-41%	-46%	-44%	-42%	-38%

Crossings | Congestion | Light Changes | Travel Time | Cars

Congestion

ID	Cum	8am	9am	10am	11am	12pm	1pm	2pm	3pm	4pm	5pm	6pm	7pm
I1	0%	-10%	-31%	-21%	0%	33%	15%	0%	-13%	-10%	-14%	10%	15%
I2	-3%	-30%	-11%	-6%	4%	-15%	0%	-22%	-10%	16%	-6%	-5%	3%
I3	5%	-29%	-22%	21%	14%	0%	-25%	-50%	0%	60%	0%	0%	0%
I4	-15%	-30%	-7%	-32%	-8%	-3%	-13%	4%	-8%	-30%	-20%	-7%	-27%
I5	-22%	-34%	-15%	-23%	-28%	-26%	-6%	-4%	-15%	-39%	-32%	-30%	-27%
I6	-11%	-16%	-13%	10%	-19%	-8%	-28%	-18%	-13%	-19%	-20%	-15%	-11%
I7	0%	0%	10%	-11%	5%	-28%	-23%	-21%	-5%	0%	0%	6%	16%
I8	-15%	-10%	-14%	-27%	-25%	-16%	-5%	-22%	-12%	-31%	-27%	-25%	-3%
I9	-9%	-5%	3%	-8%	-11%	0%	-14%	-7%	-13%	-34%	-14%	-15%	-21%
	-14%	-24%	-15%	-16%	-14%	-18%	-17%	-22%	-19%	-19%	-17%	-13%	-11%

Crossings | Congestion | Light Changes | Travel Time | Cars

Travel Time

8am	9am	10am	11am	12pm	1pm	2pm	3pm	4pm	5pm	6pm	7pm
-11%	-17%	-11%	-8%	-13%	-16%	-13%	-13%	-17%	-20%	-15%	-11%

Results - Interpretation

- Same throughput since cars on the road are same (0% diff)
- Fewer signal changes (by almost 50%!)
    - Drivers more likely to maintain momentum as the ML changes states at better times
- Lower congestion (-14% overall)
- Lower travel times (-11% overall)