Group 8 has decided to leverage the Cars Overhead With Context (COWC) dataset to learn more about object detection, especially for very large images. The data set contains more than 30,000 annotated cars across 7 different cities. The images in the data set are all very-high resolution satellite images at approximately 0.3 meter resolution. One of the primary challenges faced in working with high resolution satellite imagery is managing the size of the data. This will likely be achieved by subsetting the images to more manageable sections and processing in batches.

The images contain approximately 30,000 vehicles with annotated boundaries. Most previous examples utilizing this dataset have assumed that vehicles are roughly rectangular (usually 3m by 2m). Instead, we will attempt to draw the boundaries for each object as the vehicles vary in size and orientation. An extension to the COWC dataset has been released that further classifies the objects as cars, trucks and SUVs. These additional classes will be explored in the modeling process. Finally, past modelling efforts on this dataset have excluded the black and white images from analysis. This project will propose converting all seven images to black and white to test the ability to use all available data.

There appear to be three overarching architectures for analyzing images of this size: You Only Look Once (YOLO), RetinaNet and Region-Based Convolutional Neural Network (RCNN). We will compare the performance of YOLO and RetinaNet by analyzing their speed and accuracy. RCNN will be excluded as processing time is known to be substantially longer than the other two. Both YOLO and RetinaNet have existing Keras/Tensorflow implementations that we can leverage.

For object detection and boundary detection there are some commonly accepted error measures. First, there are the false positive/negative rates for the objects identified. Second, the Jaccard Index calculates what percent of the area bounded by the algorithm's boundaries is also bounded by the real boundaries thereby punishing the model for identifying the presence of a vehicle, but drawing the wrong boundaries. Finally, if we implement a vehicle classifier, we will also use false negative/positive rates to measure its accuracy of the particular classes.

The proposed timeline is to spend the first 2-3 weeks developing the code and preparing the data for implementing the object detection model. Upon successfully training the model, the next 2-3 weeks will be spent fine tuning results and further implementing the vehicle classifier. The final week will be spent annotating the code for intelligible explanation of the core processes and finally, preparing the report and presentation.