For the task of objection detection to correctly locate and identify the pool, for the ease of business implementation, we have opted for the torch implementations of mobilenet,which is known have faster implementation and training speed than any other detection models under that torch.vision library. Generally for a detection model since the development of faster R-CNN, the first chuck of the model comes as the feature extraction, implemented by the CNN architectures. Passing the image containing the object of pool to the CNN architectures will produce the feature map. Such CNN architectures are often referred as the backbone layers for feature mapping. Many pre-trained feature backbones are available for implementation. Examples are ResNet, VGG, and Densenet. Typically, those backbone models utilizes the convolutional 2D layers, some of which are followed by the MaxPooling operation. The advantage of CNN layers is that it allows the extraction of location invariant features by each filter. Many Convolutional layers contain more than one features filter, as a result of which, it will produce a feature map that is relatively small in size but has greater depth. The another chunk of the objective detection model, operated on the feature map is the regional proposal network. The idea is to for location of the feature map propose several different anchor ratio boxes that may or may not contain the object one is interested. All these proposal boxes are then label based on the overlapping IOU metrics, each box then has its objective scores and label scores. The procedure is simply done by using the CNN layers. The first CNN layer has the kernel of 3\*3, which through padding will yield to the same size of feature map. The resulting feature map is then simultaneously operated by two CNN layers, one is for producing the regression scores for each box coordinates, and another is for producing the objective scores. This regional proposal network is followed by other operations, as discussed in the paper Faster-RCNN. However, in the implementation, we are not very much concerned with the internal structure.

Another data pre-processing needed is to calculate the rectangular box coordinates from the polygon vectors. We then reformulate the question as finding the smallest bounding rectangular for a set of points. This implementation is through the Convexhull from the spicy library. The data loader is implemented using the Pytorch.dataloder class. In the training loop, to avoid the overfitting, the weight decay mechanism is implemented. After 10 training epochs, the following prediction for one testing image is obtained.

