Team ENJ: Multi-Object Detection

By Eric Megrabov, Nathan Apolonio, Shiming Luo, Jonathan Lam

Please use this link to access the README and code for the project:
https://github.com/shl666/SSD300_PyTorch

Abstract:

We tackle the task of detecting small objects in a given image of a scene and determine how quickly and accurately we can do so with a Single Shot Detector model. We varied several parameters with the PASCAL VOC dataset and found that this model works well for a great deal of images, with proper detection of several objects. Ultimately, we found that this is a powerful tool for detecting items in scenes and expect to compare it against different models in the future to determine its efficiency as well as accuracy in the future.

Introduction: What is the targeted task? What are the challenges?

The advent of highly advanced deep learning architectures has vastly improved the performance and speed for object detection methods, allowing them to be developed from end to end and enable real-time cases. One of the most prominent deep learning approaches to object detection is the Single Shot Detector, or SSD, an end-to-end convolutional neural network that is capable of predicting bounding boxes and class probabilities. This network aims to solve the challenge of identifying the locations of several objections (and their frequency) in images, as opposed to exclusively identifying the class of each object in the image (i.e. image classification) but performs image classification at the final steps. Our project uses the PASCAL Visual Object Classification (PASCAL VOC) dataset for the 2012 challenge, which contain 20 different classes to be identified.

To evaluate an object detection method, the most common metric that is used is “mAP”, or “mean average precision”. “mAP” is computed by identifying the overlap between a predicted bounding box and its ground truth, then identifying their union. This procedure is repeated for each class and averaged over the different classes to obtain the mAP. The model we implemented achieves a mAP of 54.9%, which is comparable to the mAP achieved in the SSD paper (74.3% for input images of 300x300 and 76.9% for input images of 512x512, both of which are from the VOC2007 test set).

Method Description: algorithm, architecture, equations, etc.

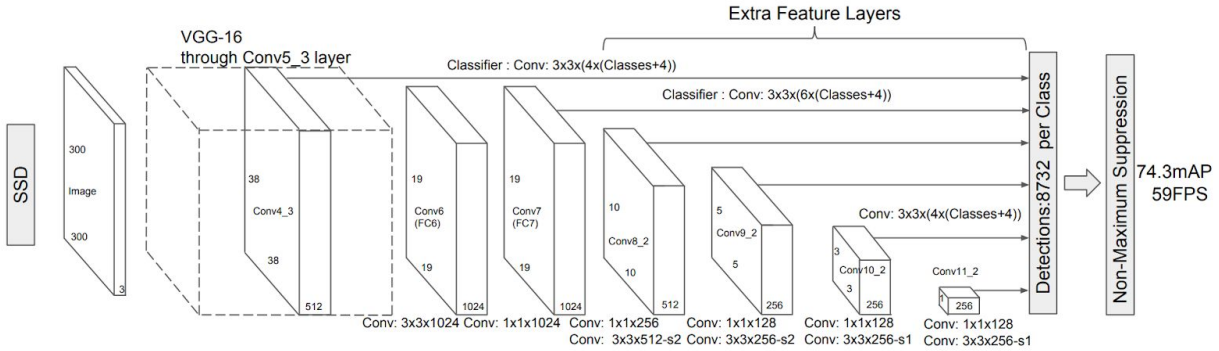


Figure 1: The Single Shot Detector Network Architecture

The SSD uses a convolutional neural network to develop a set of bounding boxes that identify and score the presence of certain object instances for a given box as input. The base network used is the VGG-16 network, but the paper references that other networks should be able to produce good results. Immediately following the VGG-16 base architecture is a set of convolutional feature layers that progressively decrease in size, allowing for predictions of detections at various scaling levels (these are referenced as "multi-scale feature maps for detection" in the SSD paper). Additionally, a set of convolutional filters are applied after a layer for producing detection predictions. Depending on the kernel, these outputs are either a prediction score for a category or a shape offset relative to the default box coordinates. Certain default bounding boxes are associated with a particular feature map so that the position for each box relative to the default box is predicted. The score per class is also computed after processing of each layer. In particular, for a given box, the class scores for each class are computed along with the 4 offsets relative to the default box shape. These 4 offsets are the width, height, center of the image. This is a total of (# of classes + 4)*(# of boxes) filtres applied around each location for a particular feature map, which produces (# of classes + 4)*(# of boxes)*(m*n feature map) outputs.

During the training procedure, the ground truth information for a particular image is assigned to specific outputs for the fixed set of detector outputs. Thus, training concerns the selection of default bounding boxes and scales for detection, as well as hard negative data mining and data augmentation. When selecting the set of bounding boxes to use for detection, these boxes should vary over location, aspect ratio, and scale. The default bounding boxes selected for SSD are chosen such that any image that overlaps with its ground truth counterpart has jaccard overlap greater than 0.5. The SSD paper asserts that this process allows the network to predict higher scores for multiple overlapping default boxes, as opposed to selecting the box with the highest overlap.

The default boxes are designed such that specific feature maps become sensitive for different scaling of objects. The paper uses the following formula for computing the scale for a default box:

$$s_k = s_{\min} + \frac{s_{\max} - s_{\min}}{m - 1}(k - 1), \quad k \in [1, m]$$

where s_min is 0.2 (the lowest layer has a scale of 0.2) and s_max is 0.9 (the highest layer has a scale of 0.9) and all layers in between are evenly spread out. The aspect ratios chosen in the paper are (1, 2, 3, ½, ⅓). The width and height for each box is computed as w = s_k√(aspect ratio) and h = s_k/√(aspect ratio). The center of each box is set to ((i + 0.5)/|f_k|,(j + 0.5)/|f_k|)) where |f_k| is the size of the k-th square feature map and i,j ∈ [0,|f_k|).

The loss function during training is as follows:

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g))$$

where L_conf(x,c) is the confidence loss over multiple classes confidences and is estimated using softmax loss and L_loc(x,l,g) is the localization loss for the matching strategy, i.e., a smooth L1 loss between the predicted box l and the ground truth parameters g. N is the number of default matching boxes and the weight term alpha is set to 1 by cross validation. The equations for L_conf(x,c) and L_loc(x,l,g) are

$$L_{conf}(x, c) = -\sum_{i \in Pos}^{N} x_{ij}^{p} log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^{k} \text{smooth}_{\text{L1}}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)$$

This process is immediately followed by a non-maximum suppression procedure that outputs the final detection decisions. When the confidence threshold is set to 0.01, most of the boxes can be filtered out. The authors of the paper used a confidence threshold of 0.01, nms with jaccard overlap of 0.45 per class and retained the top 200 detections per image, achieving over 70% mAP on a system with a Titan X and cuDNN v4 with Intel Xeon E5-2667v3@3.20 GHz and batch size of 8.

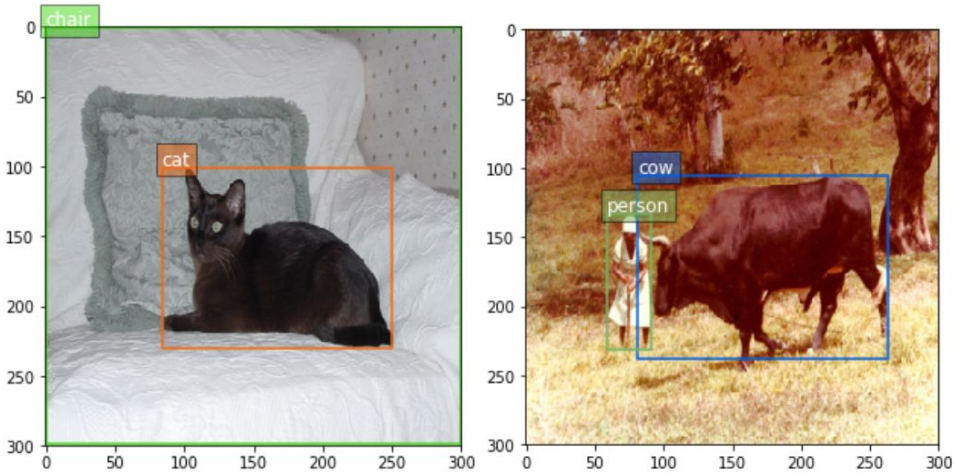
In summary, the SSD produces a score for the presence of each object category for each bounding box and, during training, will make proper adjustments such that the bounding boxes fit more appropriately to their designated object shapes. The set of various feature maps and filter sizes allows the network to detect objects of different sizes in images with varying resolutions. Non-maximum suppression is applied at the end for inference and results in the final detection outputs.

Experimental Setting: dataset, training parameters, validation and testing procedure (data split, evolution of loss with number of iterations etc.)

The dataset we chose to use for this project was the PASCAL Visual Object Classification dataset, which is often used for challenges surrounding segmentation, classification, and object detection, which is how we employed it. This dataset was finalized in 2012 with approximately 10,000 images used for training and validation, which makes up 50%. The other 50% was used for testing the images against the trained models that competitors developed for many of the different challenges proposed. This dataset consists of 20 categories, as can be seen below:

```
VOC2012_CLASSES = (  # always index 0
#     'background',
    'aeroplane', 'bicycle', 'bird', 'boat','bottle',
    'bus', 'car', 'cat', 'chair','cow',
    'diningtable', 'dog', 'horse', 'motorbike', 'person',
    'pottedplant','sheep', 'sofa', 'train', 'tvmonitor')
```

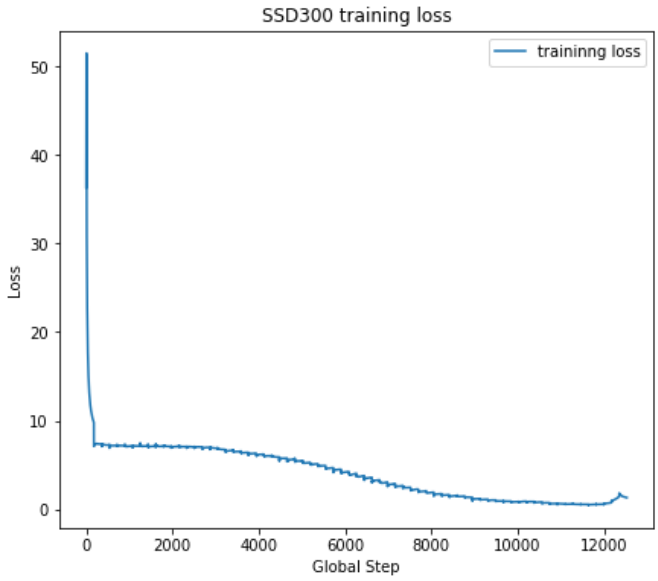
The goal is to detect these listed objects in an image that may or may not contain one or many of these listed items. Some ground truth images from the dataset can be found below.

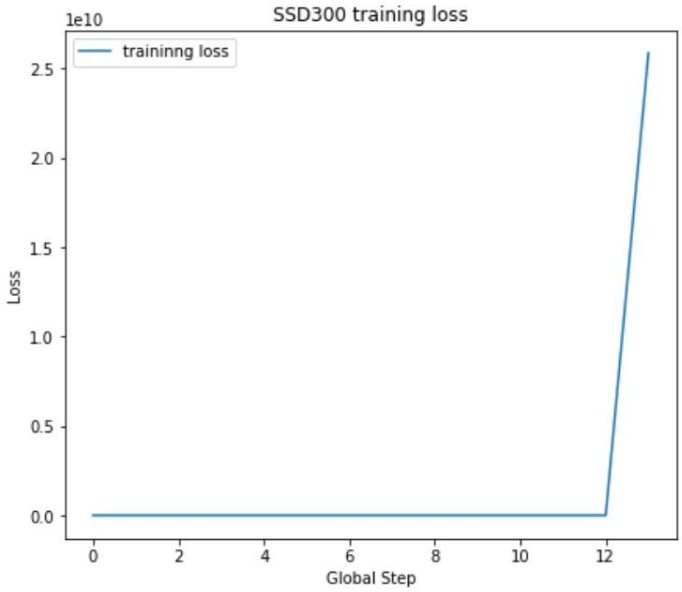


There are quite a few training parameters based on the structure of the single shot detector. In the VGG layers, there are 15 weights and 15 biases. In addition, there are 8 weights and 8 biases for the extra feature layers. Furthermore, there are 6 localization weights, 6 localization biases, 6 confidence weights, and 6 confidence biases. This yields a grand total of 70 learnable parameters.

We originally trained as the paper intended with a relatively small batch size of 8, but due to limitations with the GPU, we decided to use a batch size of 32 in order to perform faster training; Even so, a single epoch took approximately 30 minutes to run. We ran it for a total of 65 epochs to yield the result below. This was nearly two days of training the model continuously. As a result, we decided to calculate the loss of the training set in the form of global steps rather than epochs.

In addition, we performed our experiment with a couple of different optimizers; we tried both adam and stochastic gradient descent to see which one would run more quickly and which one would yield better results. The training loss with adam optimizer can be seen below on the left. As you can see, it converged nicely after only a few epochs and had slow improvement as time went on. The training loss with stochastic gradient descent is far worse, however, and is met with the gradient exploding problem. It can be seen to the right.





It is clear that at 12000 global steps (around 65 epochs), there was a slight jump in loss. Given this trend, we decided to stop training as it would no longer be beneficial to the model and would

hurt our prediction accuracy. Due to this jump in the loss, we decided to also turn down the learning rate from lambda = 1e-3 to lambda = 1e-4. This ended up yielding relatively good results on the validation set, as can be seen in the next section.

We determine if the result is acceptable and should be kept if the jaccard similarity between two bounding boxes is 50%, meaning that the bounding boxes have at least 50% overlap. In addition, if the same item is detected with different sized bounding boxes, we used non-maximal suppression in order to not display bounding boxes that would be redundant. It does not make sense to show multiple boxes around the same item since they do the same thing, so we just keep the largest bounding box. We tested the results with a series of different thresholds on both the training set and validation set to see what threshold values yielded the best results. Ultimately, we chose the recommended threshold from the Wei Liu's "SSD: Single Shot MultiBox Detector" (2016) of 50%.

Results: figures, tables, comparisons, successful cases and failures

Class	Training mAP	Validation mAP
aeroplane	0.963	0.736
bicycle	0.890	0.594
bird	0.928	0.555
boat	0.818	0.457
bottle	0.446	0.144
bus	0.952	0.715
car	0.818	0.528
cat	0.965	0.726
chair	0.847	0.382
cow	0.901	0.578
dining table	0.929	0.536
dog	0.960	0.642
horse	0.945	0.656
motorbike	0.948	0.671
person	0.736	0.471
potted plant	0.751	0.307
sheep	0.866	0.545
sofa	0.896	0.502
train	0.944	0.668
tv monitor	0.916	0.568
Average mAP	**0.871**	**0.549**

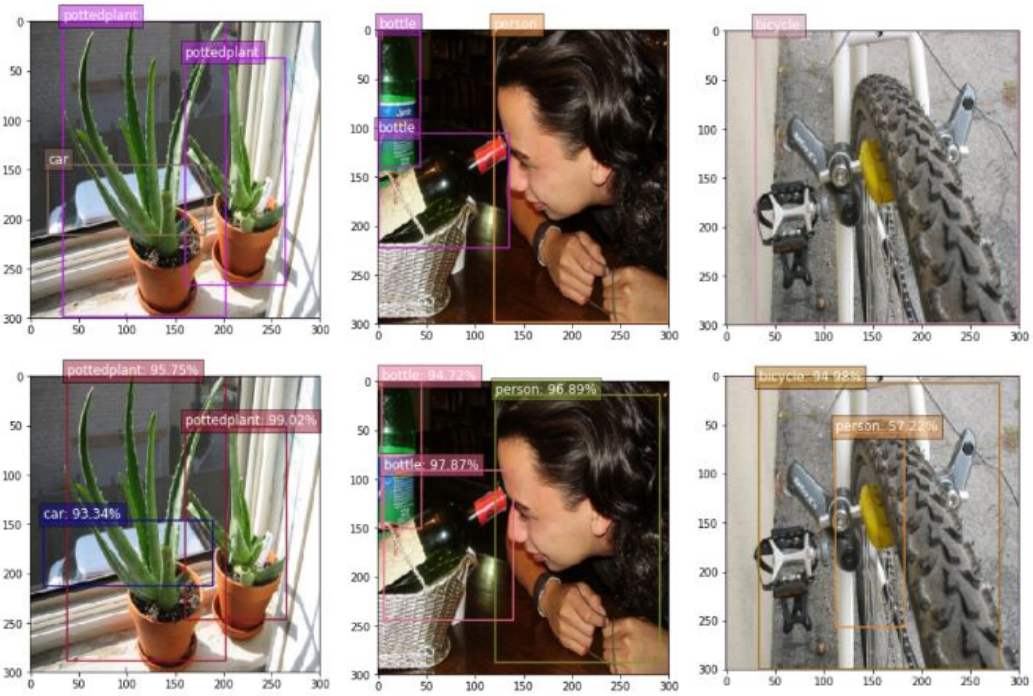
Resulting mAP Table

In the table above, it displays the overall mAP and the mAP of all classes in the training and validation set. The training average mAP is at 83.3% but the validation average mAP is only at 48.2%. This indicates some minor issues within our trained model.

Below are the results from our training set and validation set. For our training set, we can see that the labels and bounding boxes on the bottom images are fairly close to the ground truth image above them. But we see that the training set is still subjected to errors as we see in the bicycle picture.

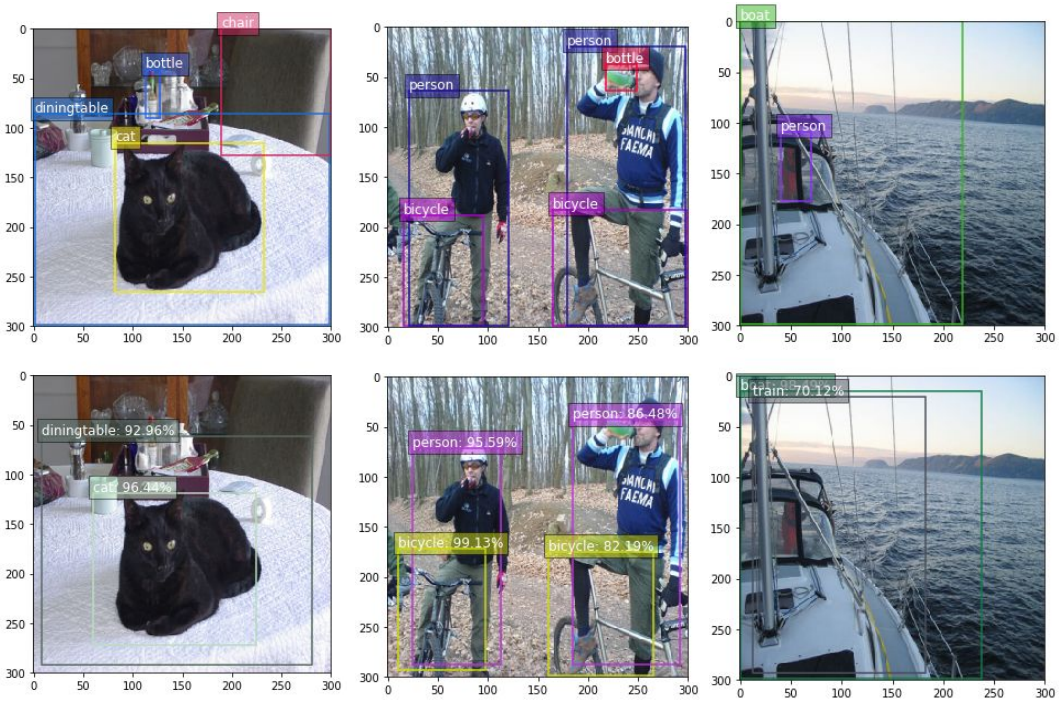
When evaluating our validation set, we tested the confidence threshold and decided to have a threshold of 0.7. The confidence threshold is used to determine what the model will predict for that image and any class that exceeds the threshold would be determined as a matching object in the image. When determining the threshold, we have to keep in mind to not make it too low or too high. If the threshold was too low, false positives start showing up in the picture as seen in the images predicted with a threshold of 0.25. However, if the threshold is too high, false negatives occur meaning that objects in the picture are being ignored. This can be seen in the images predicted with a threshold of 0.9 not labeling all the people and cars.

TRAINING RESULTS
THRESHOLD = 0.5



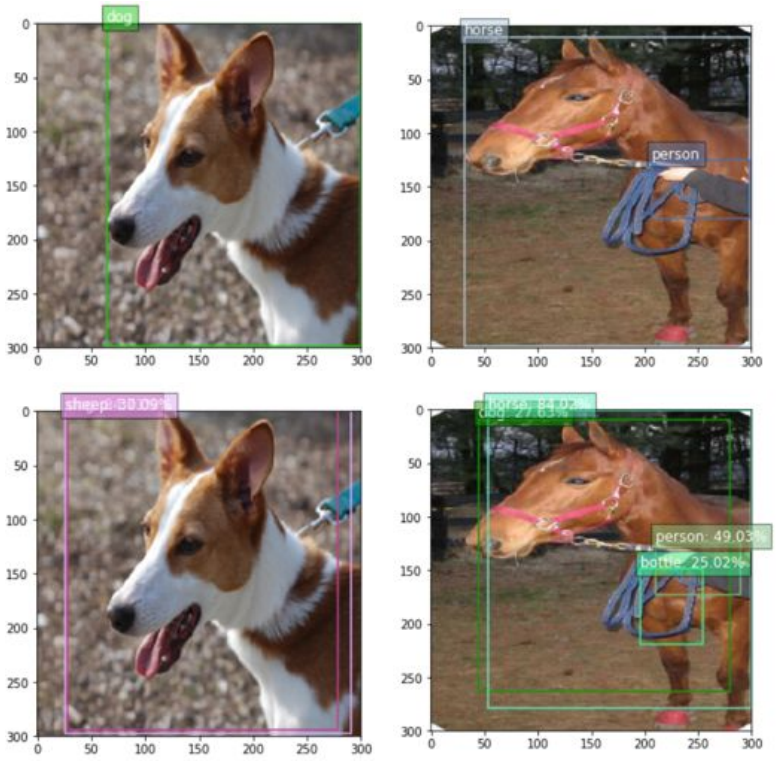
Training Set

VALIDATION RESULTS
THRESHOLD = 0.7



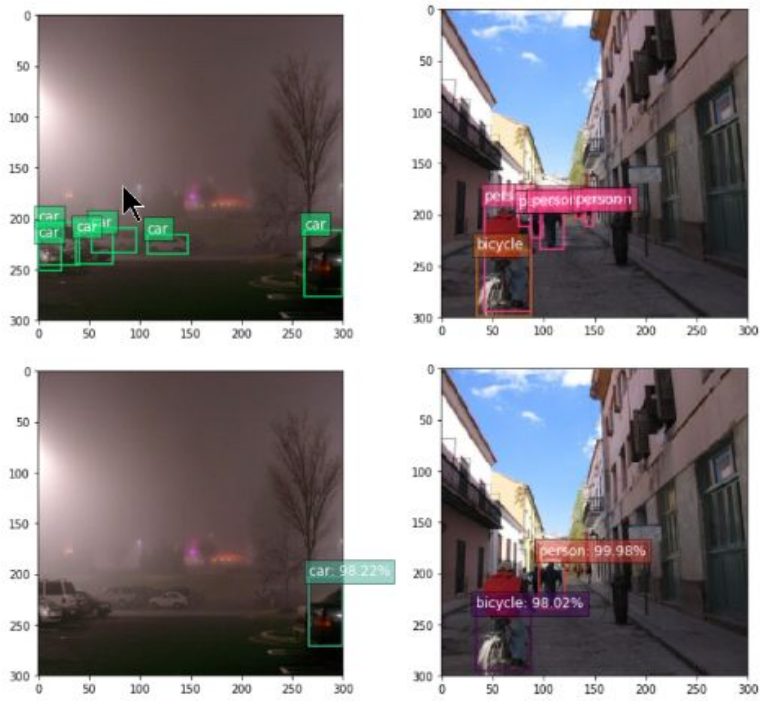
Current Prediction Model

Threshold = 0.25



False Positive Predictions

Threshold = 0.90



False Negative Predictions

Discussion: What did you learn? What were the difficulties? What could be improved?

Looking at the results of our SSD model, we can see some issues that it contains. While the SSD paper lists that their model achieves a mAP of 74.3% on the 2007 Pascal Voc dataset, our own model is sitting at a 54.9% mAP on the 2012 dataset. It highlights some of the issues that our model contains. Overfitting is an issue due to the big difference when comparing the 87.1% training mAP and the 54.9% validation mAP.

Some issues we do have is that the training is taking way longer than expected, causing us not to be able to re-run our training too much. We also experienced issues with upgrading our pytorch because it was causing GPU tasks to take longer than before. Due to these issues, we could not really try different changes to the SSD model since the runtime became an issue the closer we got to the deadline. At first, we did not think it was not much of an issue until we realized it was taking upwards of 30 minutes to run one epoch when training the model.

For future work, we have several ideas to help improve our model's accuracy. One suggested method was to use data augmentation to help improve robustness of our model. For data augmentation, there are several approaches outlined by the paper that training images could be randomly sampled. These include random sampling and different thresholds of the jaccard similarity. Theses were not explored because of time and computation constraints we had faced. Another interesting data augmentation technique is to feed certain transformed images that

improve the network's ability to learn invariant representations. Such transformations include rotation, upsampling and downsampling, and inversion performed at different scalings and for different aspect ratios. If we were able to use different forms of data augmentation, there is no doubt that it would have helped improve the bounding box confidence and prediction accuracy.

Another issue we can tackle that can improve our model with future work would be the fine-tuning of the confidence threshold. Currently, we are manually testing threshold values so the current mAP of the validation set is not the guaranteed best mAP value for our model; we just found a threshold value based on intuition as well as trial and error. At 0.5, there would be many false positives due to errors that would barely meet that requirement and a lower threshold would just introduce more false positives than correct labels. A threshold of 0.75 was tested along with 0.7, and we found that 0.7 held the best mAP we had. Thus, we decided to use that as our threshold. An improvement to our model would be to create a tuner for threshold to get the best mAP possible out of our validation set.

Another improvement for this project would be to create another neural network, like a fast R-CNN, and compare that performance to our SSD model for better comparisons. A proper comparison would be nice to see and help determine a better model to use for object detection, both in terms of efficiency as well as accuracy.

Bibliography:

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, Scott., Fu, C.Y., Berg, A.: SSD: Single Shot MultiBox Detector. In: ECCV. (2016)