Christina Saidy

Report: Object Detection Model

The Oxford-IIIT Pet Dataset contains 37 different classes that include different breeds of cats and dogs. One of the first challenges I faced with this dataset was figuring out how to correctly parse the res files that have the bounding box coordinates and match them with their corresponding images. Initially, my dataset returned images with wrong bounding box coordinates because the matching wasn't done properly. This went unnoticed until I trained my first model on mismatched data, and the results were horrible. Eventually, the approach I went with was to extract the breed and image information from the filename in the xml files and use that to get the bounding boxes to the correct images. This approach made sure that each image was paired with the right label and bounding box.

For the modeling part, I tried two different approaches. The first was transfer learning using Resnet50 as a base model. Resnet50 is a popular backbone for object detection tasks because of its good feature extraction capabilities. Implementing this was pretty straightforward. I replaced the output layers ("heads") of Resnet50 with two separate layers: one for predicting the class (the breed) and another for predicting the bounding box coordinates.

The second model was a custom CNN built from scratch, which loosely follows the structure of VGG. It has six convolutional layers, each followed by batch normalization, ReLU activation, and max pooling. After these layers, there's a fully connected layer that splits into two heads, one for classification and one for bounding box regression.

Both models were trained using the same process. I split the dataset into training and validation sets. For the classification part, I used cross-entropy loss, and for the bounding box coordinates, I used mean squared error (MSE) loss. The optimizer used was Adam, which updated the model parameters during training. I also used TensorBoard to log and plot the losses and accuracy during both training and validation, which was very helpful to monitor how the models were improving over time. The model that performed best (meaning it had the lowest validation loss) was saved to be tested on.

When testing and visualizing the output, the Restnet50 model performed quite well. It was almost always able to guess the correct breed. Intersection over union scores was used to test the models as well. Resnet50 also had decent IOU scores. However, the custom CNN did not perform as expected. Its predictions were wrong about half of the time. I tried different things to improve the custom CNN, like adding or removing convolutional layers, increasing the number of epochs, and adjusting the learning rate. But so far, the model has not shown much improvement and still struggles to generalize well to unseen data

As expected, using transfer learning is better effective and time efficient. It is definitely easier to obtain better results using a pre trained model. The most shocking part was the time it takes to train a model from scratch.