Project: Open images 2019 – Visual Relationship

Kesar TN*
University of Central Florida
kesar@Knights.ucf.edu

Open Images dataset is a collection of around 9 million images and in this project, I concentrate on the visual relationship track. The objective of the project is to build a Deep learning model to predict the subject-object relation. It can be broadly divided into two sub problems. I train each model with precision on TensorFlow object detection API, using Faster RCNN Inception. I then create an ensemble network to make predictions of the visual relations. In this paper, I explain the architecture used, along with the approach for each track and finally visualize inferences from both the tracks.

Keywords: *Open Images, Visual Relationship, TensorFlow Object Detection API, Faster RCNN*

1 Introduction

The Open Images 2019 dataset can be broadly divided into is and non-is relationships for the visual relationship task. The is relations consist of an object in the image and it's attribute. For example, "Chair is wooden", "Cup is Plastic", "Jacket is Leather", etc. where for the first example the chair is the object and the attribute is wooden. In total, there are 57 object classes and 5 attributes namely Transparent, Plastic, Textile, Leather, Wooden. The non is relations consists of a Subject Object relation, along with a relational label, that is one of at, on, interacts with, holds, inside of and plays. Hence the labels will be as following, 'Boy interacts with Monkey', 'Dog on Motorcycle', 'Boy plays Drum' etc. In total, there are 287 different triplet relations in the training data. Along with one of 57 objects appearing as either the subject or the object. It was particularly difficult to train the non-is classes because of the model's behaviour of ignoring numerous bounding boxes in the same area, or in other words, non-max suppression. Even if I trained subjects and objects individually, It is extremely difficult to find out the relationship label. I use the second approach of training on all the triplet data. I train an ensemble model to obtain results for is and non is relations. There is a lot of pre-processing that is done to train the models. I use the pre-trained model, trained on open-images in order to fine tune it to the required constraints. The prediction is done on the bases of objects being detected, as in the model switches between is and non is based on the number of objects detected on the image. There were numerous attempts to use other techniques, using YOLO, Fastai etc. but the Tensorflow Object detection API is most effective approach since it's already trained on open-images. I explain my approach in the section Technical Description. The Visual Relation challenge is particularly hard because of the relationship between the objects. I use Tensorflow object detection API to do the problem statement, which was run on the UCF Newton Cluster. This kaggle competition provided the training and the testing data, along with the annotations.

2 Architectural Design

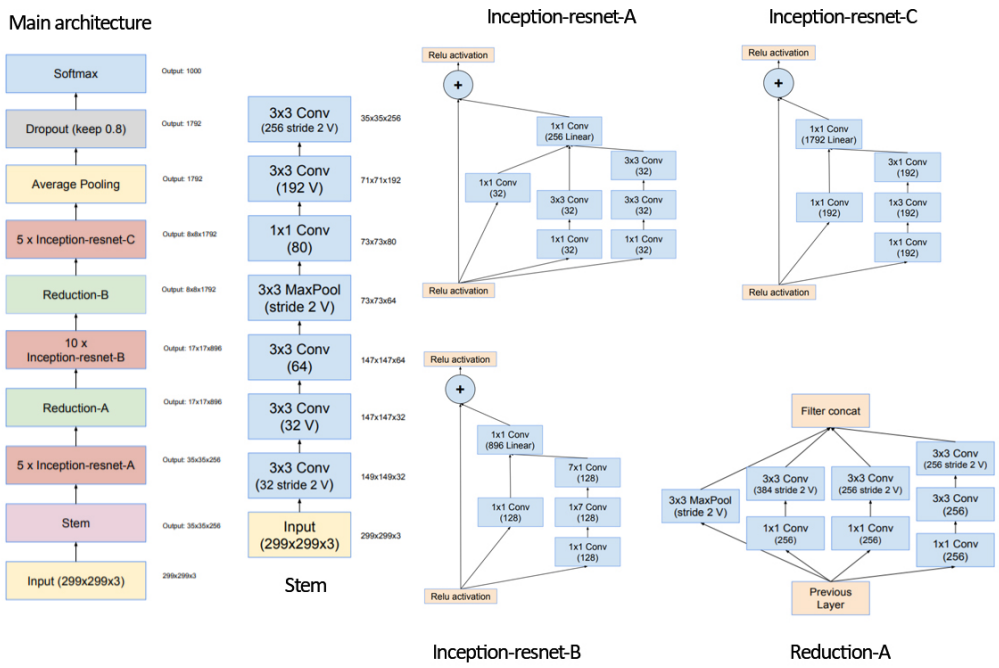
We divide the project into two different tasks i.e is and non-is relation both trained using TensorFlow Object Detection API using the pre-trained model `faster_rcnn_inception_resnet_v2_atrous_oidv4` from the model zoo which obtains 54% mAP@0.5[$\hat{4}$] in 425 ms.

Faster R-CNN introduced the Region Proposal Network (RPN) in their architecture. It means that it handles slow search selective algorithm with fast neural network. At the last layer of an initial CNN,

*Graduate student at University of Central Florida (https://github.com/kesaroid)

Preprint. Under review.

Figure 1: Faster RCNN Inception ResNet v2 architecture



a 3x3 sliding window moves across the feature map and maps it to a lower dimension (e.g. 256-d). For each sliding-window location, it generates multiple possible regions based on k fixed-ratio anchor boxes (default bounding boxes). Each region proposal consists of a) an "objectness" score for that region and b) 4 coordinates representing the bounding box of the region. In other words, we look at each location in our last feature map and consider k different boxes centered around it: a tall box, a wide box, a large box, etc. For each of those boxes, we output whether or not we think it contains an object, and what the coordinates for that box are. The 2k scores represent the softmax probability of each of the k bounding boxes being on "object." Notice that although the RPN outputs bounding box coordinates, it does not try to classify any potential objects: its sole job is still proposing object regions. If an anchor box has an "objectness" score above a certain threshold, that box's coordinates get passed forward as a region proposal. Once we have our region proposals, we feed them straight into what is essentially a Fast R-CNN. We add a pooling layer, some fully-connected layers, and finally a softmax classification layer and bounding box regressor. In a sense, Faster R-CNN = RPN + Fast R-CNN. Faster R-CNN may not be the simplest or fastest method for object detection, but it is still one of the best performing. Case in point, Tensorflow's Faster R-CNN with Inception ResNet is their slowest but most accurate model. I use this model to train my model to get the most accurate results.

As the latest generations of Inception models perform rapidly well, Combining Inception architecture with residual connections proves to be useful. We use cheap inception blocks followed by a 1x1 filter-expansion layer which is used for scaling up the dimensionality of the filter bank before the addition to match the depth of the input. Another small technical difference between our residual and non-residual Inception variants is that in the case of Inception-ResNet, we used batch-normalization only on top of the traditional layers, but not on top of the summations. By omitting the batch-normalization on top of those layers, the original authors were able to increase the overall number of Inception blocks substantially.

As we can see in Figure 1, The Faster-RCNN Resnet Inception block consists of various mini blocks including a stem. The first block or the stem is a linear block followed by 5 Inception-Resnet block A. We then have a bottleneck Inception Resnet layer called the reduction. While we only display Reduction-A, B is also exactly similar. This is followed by 10 Inception-resnet-B blocks, with

Reduction-B being used. It finally uses 5 Inception-Resnet C blocks before average pooling the layers and using SoftMax.

2.1 is-relation

We start with is relational elements that have an object attribute pair. These relations are straightforward like 'Chair is wooden.' The first approach I used took in all the 57 objects in the object section along with the 5 attributes. It is trained on a total of 62 classes = 57 + 5. It predicts both the Object and it's attribute. We calculate the final prediction based on the confidence of each prediction by multiplying the confidence of the object and attribute. But the problem with this approach was that in the non-max layer, It cancels out the second bounding box which denoted the attribute and only displayed the object category. Also because of the less number of attributes to be trained on, the model wasn't able to detect if it was Plastic, Transparent or Wooden. I would have to penalise the classes with lesser number of instances which would be called Negative hard-mining of classes. Instead I used another approach that suited better, since I was training an ensemble model.

We consider all the combinations of elements using the annotation and arrive at 42 different combinations in the train dataset and 41 total combinations in the test. For example, the combinations of all the objects to each of the attributes like 'Table is wooden', 'Chair is wooden' etc. This would be a many to one combinational matrix. I do this by finding all the unique pairs in the annotation csv and create a label_map.pbtxt for the same. I first read the entire data from the csv and left merge the dataframe using pandas. Once I obtain the classes and the attributes from their class id, I iterate through each of the image in the train/validation folder to retrieve the image dimensions. Using these image height and width, I multiply the normalised xmin, xmax, ymin, ymax and then convert them to int so that I get the pixel values of the bounding boxes. I then run an index check through all these bbox values so there is no pixel value beyond the image dimensions. I found a couple of images that had been annotated wrongly, when the annotator drags the bounding box region in the respective annotation tool. I add .jpg to all the filenames and save the csv locally, to create tfrecords. To create the tfrecord file, we use shards approach, which increases parallelism in the model, where the GPU will be able to load more number of images before exhausting resources. I then create tfrecords by using the csv file and call it train_is.tfrecord and val_is.tfrecord. By creating this approach, we ignore all the other combinations that might have not appeared in the test dataset. Using these 42 classes, we use the `faster_rcnn_inception_resnet_v2_atrous_oidv4` model and use it as a baseline. I use a batch-size of 1 with a very low learning rate of 5.99e-5 preserving the aspect ratio. This is trained for 200k iteration.

2.2 nonis-relation

For the non-is relations, the relations that need be predicted are 'Girl holds Microphone', 'Chair at table', 'Man interacts with Tennis racket' etc. This part of the problem is not just to predict both the subject-object pair but to also predict the relation between them. We first create another train and validation tfrecord. The first approach I tried was to consider the two bounding boxes of the classes seperately, that is considering the pair of bounding boxes as separate classes. Hence this subject-object relation will disappear. I had planned to get the bounding boxes of both the objects, and based on the IOU and the euclidean distance between the center of bounding boxes, describe the relation between the both of them. This could have either been done by using LightGBM or XGboost models according to previous kaggle grandmasters. I did not follow this method after realizing a few errors that I might face, I would have to create a different loss function using the confidences of the bounding boxes. In other words, before the bounding boxes in is relation had the same coordinates, where I could train the entire sentence with one class name. Hence I took the same approach as the is relation. I map all the values that are non-is onto a dataframe, and I consider the min-max of the 2 bounding boxes, creating a big bounding box around both objects. Now, using the subjectRelationship LabelObject, I create classes for each of these bounding boxes with respect to the images. This creates the training csv that will create the tfrecords. ...

3 Dataset

Open Images is a collaborative release of 9 million images annotated with image-level labels, object bounding boxes, object segmentation masks, and visual relationships. This uniquely large and diverse

Figure 2: Number of objects per image (left) and object area (right) for Open Images V5/V4 and other related datasets (training sets in all cases).

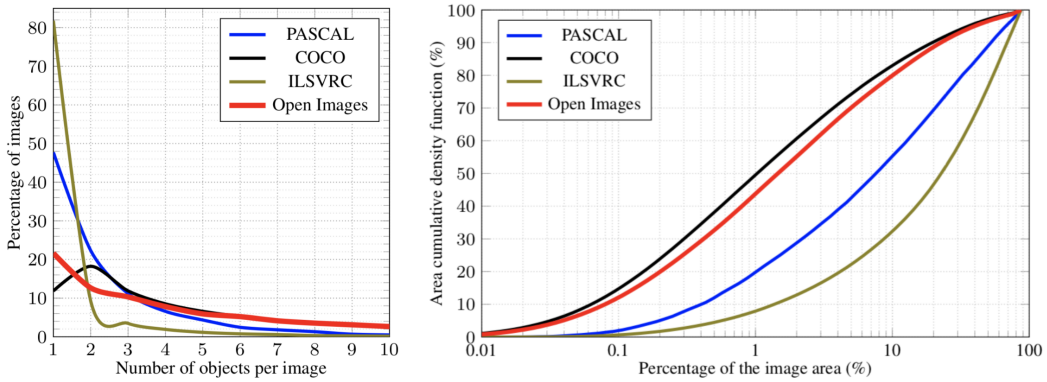
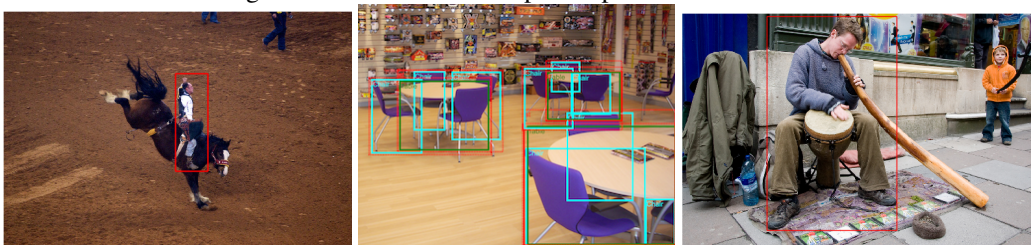


Figure 3: Visual Relationship example from the train data



dataset is designed to spur state of the art advances in analyzing and understanding images. It contains a total of 16M bounding boxes for 600 object classes on 1.9M images, making it the largest existing dataset with object location annotations. The boxes have been largely manually drawn by professional annotators to ensure accuracy and consistency.

In the annotation, a pair of objects connected by a relationship forms a triplet (e.g. "beer on table"). Visual attributes are also represented as triplets, where an object in connected with an attribute using the relationship is (e.g. "table is wooden", "handbag is made of leather" or "bench is wooden"). We initially selected 467 possible triplets based on existing bounding box annotations. The 329 of them that have at least one instance in the training split form the final set of visual relationships/attributes triplets. In total, we annotated 375K instances of these triplets on the training split, involving 57 different object classes and 5 attributes. These include both human-object relationships (e.g. "woman playing guitar", "man holding microphone") and object-object relationships (e.g. "beer on table", "dog inside car").

Annotations are exhaustive, meaning that for each image that can potentially contain a relationship triplet (i.e. contains the objects involved in that triplet), we provide annotations exhaustively listing all positive triplets instances in that image. For example, for "woman playing guitar" in an image, we list all pairs of ("woman","guitar") that are in the relationship "playing" in that image. All other pairs of (woman,guitar) in that image are reliable negative examples for the "playing" relationship.

- Object detection track for detecting bounding boxes around object instances, relaunched from 2018.
- Visual relationship detection track for detecting pairs of objects in particular relations, also relaunched from 2018.
- Instance segmentation track for segmenting masks of objects in images, brand new for 2019.

Google AI hopes that having a single dataset with unified annotations for image classification, object detection, visual relationship detection, and instance segmentation will stimulate progress towards genuine scene understanding.

Figure 4: is relation Tensorboard loss trends

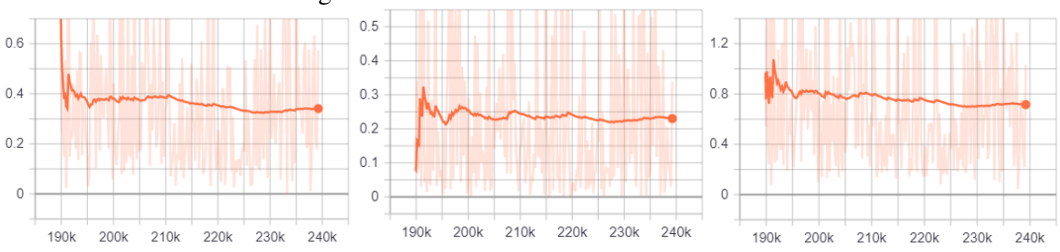
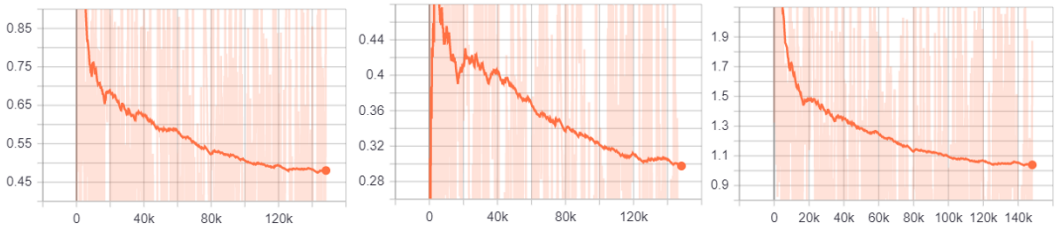


Figure 5: non-is relation Tensorboard loss trends



3.1 Visual Relationship

In this track of the Challenge, you are asked to detect pairs of objects and the relationships that connect them.

The training set contains 329 relationship triplets with 375k training samples. These include both human-object relationships (e.g. "woman playing guitar", "man holding microphone"), object-object relationships (e.g. "beer on table", "dog inside car"), and also considers object-attribute relationships (e.g."handbag is made of leather" and "bench is wooden").

4 Main Results

After training numerous models for various iterations, I create an ensemble network for the prediction of images for both is and non-is relation. I visualize the inference images, along with losses from the tensorboard. As we can see from the trends, The graph is a decreasing function by reaching 0.2600 for is relation. We can see the inferences in Figure 6. The second model I trained on was for the non-is relations. It consists of 287 classes in total. Hence, the results have subject and object pair with the relational label. Figure 7 describes the inferences for various test images.

5 Conclusion

The Visual Relationship task for the Open Images dataset is a complex problem statement. We are able to visualize the relation between objects by dividing the problem into is and non is training. We display the inferences and the results from the training and testing. The losses have been visualised along with the training accuracy.

The dataset was complicated with a lot of relations, Especially in the non is track. For the future work with more time, I would be using LightGBM or XGBoost to track the relations using the IOU and euclidean distance between the center of the bounding boxes. I intend to participate in the kaggle

Table 1: Comparing the is and non-is relations training metrics

Technique	loss	iterations
is relation	0.2600	239228
non-is relation	0.5718	148414

Figure 6: is relation visual inference

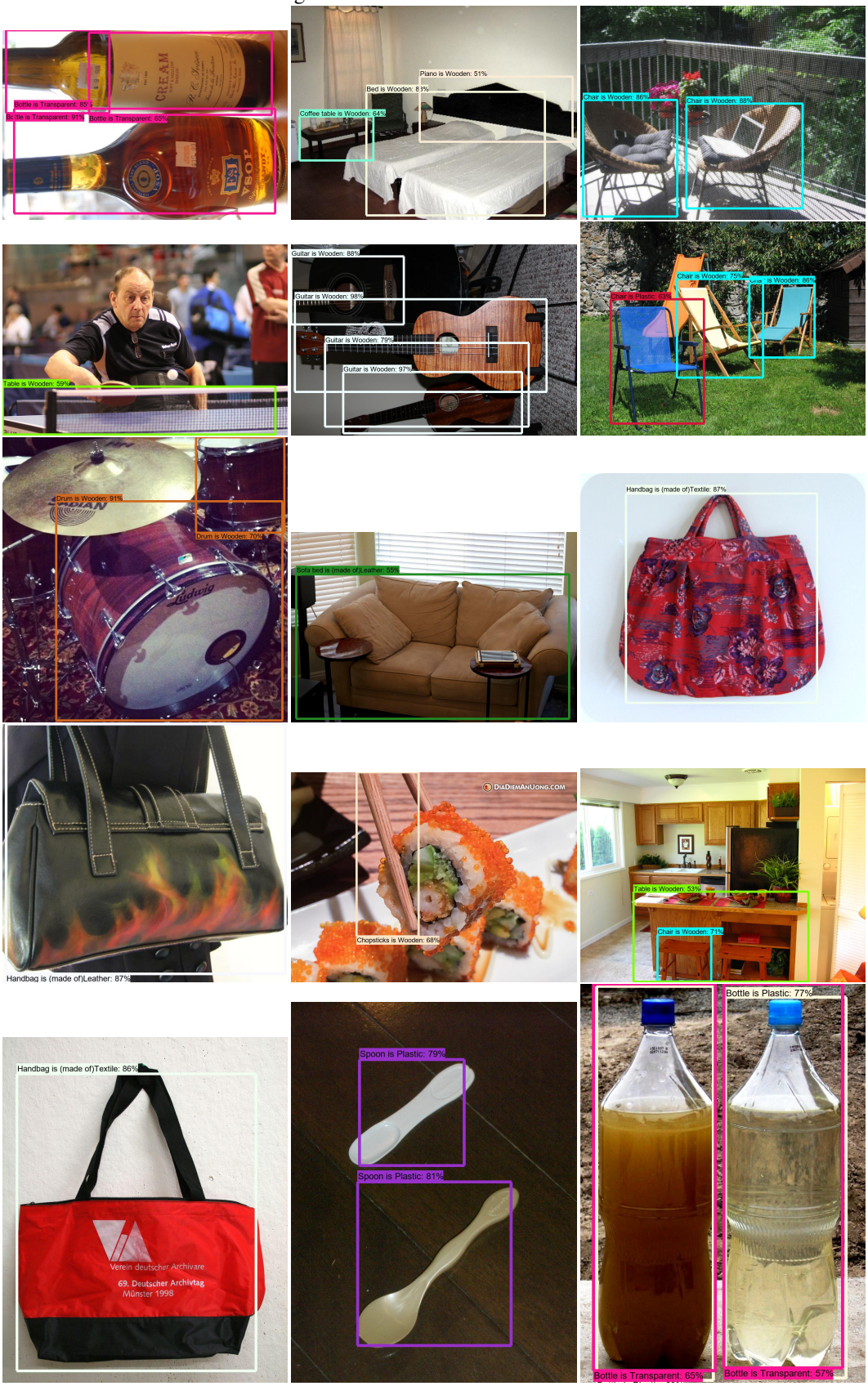
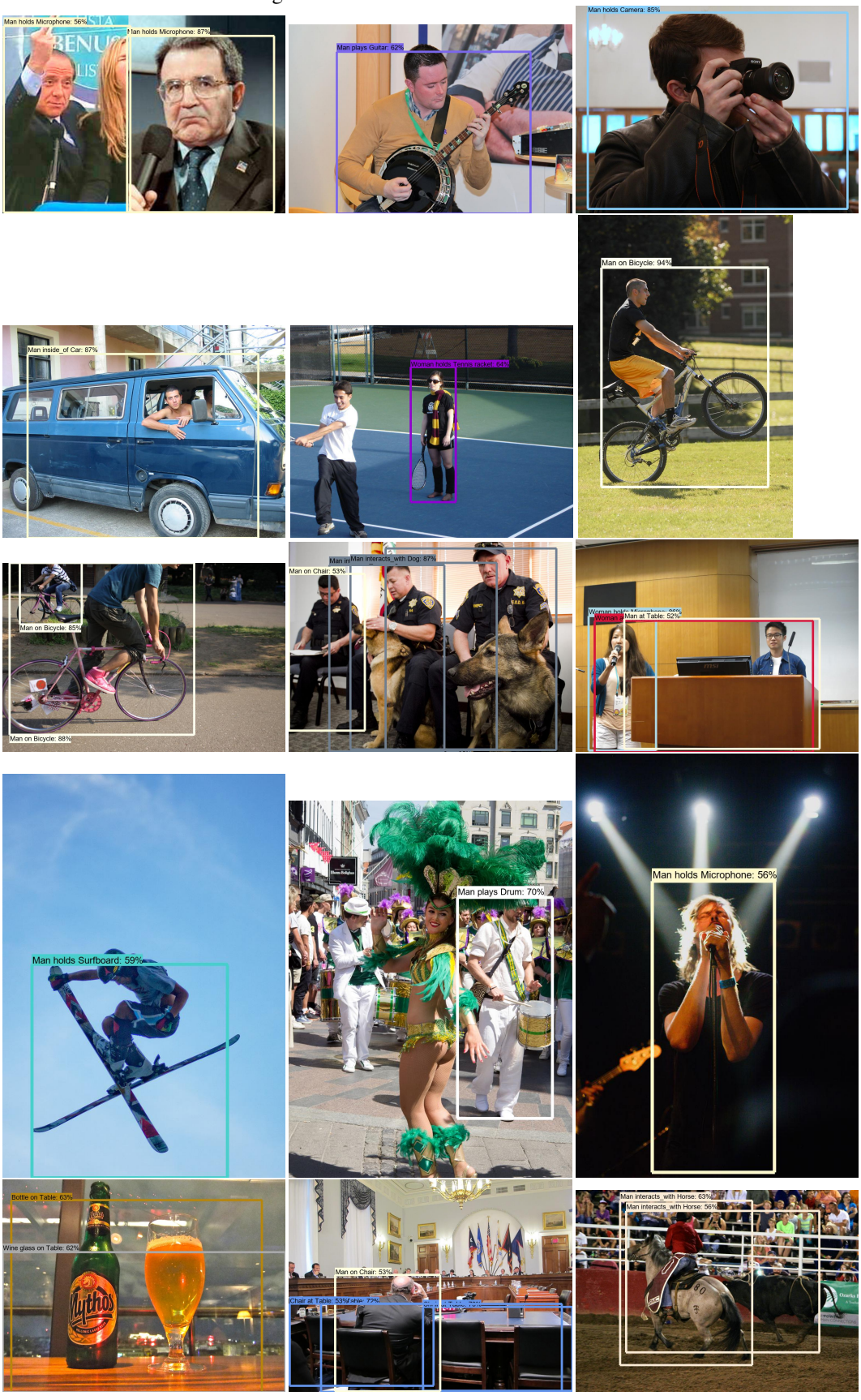


Figure 7: non-is relation visual inference



competition and improve on my results by the end of winter break. The inferences on the test set shall be done after applying to the kaggle competition as it was not available on the cluster.

References

[1] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi, (2016), Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning url=arXiv:1602.07261

[2] https://www.kaggle.com/c/open-images-2019-visual-relationship/discussion/94332latest-617243

[3] https://towardsdatascience.com/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9

[4] https://github.com/tensorflow/models/tree/master/research/object_detection