**Hninn Khin**
**DATS 6203 – Machine Learning II**
**Individual Report**

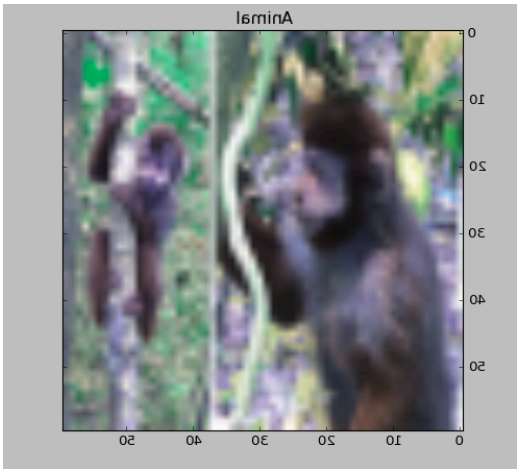
Our project dataset is called Google AI Open Images dataset. This dataset is available on Google Open Image Dataset V4 website. It was available for Kaggle object detection and segmentation competition. It consists of natural images that reflect everyday scene and provides contextual information. There are total 516 classes with total 9 million images. The original size of image is 1024x760. The type of objects in these datasets vary in shape, size and color across the same class. Since the raw dataset is very large, we only used a portion of it for our analysis. The aim of this project is to classify the different images in our dataset. The below figure shows the total 10 classes which we are using in this project.

The first thing that we did on the dataset was to resize all the images in one size into 60x60 and chose 3 channels images for training and testing purposes. The original size of image has a very high resolution of 1024x760 and also had some gray images in our dataset, we only considered colored images. After this preprocessing, we ended up with 34,471 images and since the data is huge, training the data in our own computer is not feasible, therefore we worked on GCP and AWS.

The shared works are looking for datasets, downloading the datasets, data preprocessing, data augmentation, creating frameworks, modifying the codes from online and interpreting the model performance and results. All three of us contributed to this project and there is a very thin line to describe who did what.

My part of contribution to this project is uploading our final dataset from my GCP cloud instance to Google Cloud storage and created the instruction and also started the final presentation, proofread the final project report, modifying the codes, read past research to understand our neural network framework, find appropriate code so that we can modify and implement it in our code.

During data argumentation process, our python did not run on cv2 and we kept getting error on cv2 packages and I modified the code using PIL packages. We didn't use this code in the final project code.



We found that implementing batch normalization improves our model from 35% accuracy to 50% accuracy.

Accuracy from the above network without batch normalization is 35% with 200 epochs, 2 epochs is 31.69% and with 500 epochs our accuracy is 37%. Loss functions provide more than just a static representation of how our model is performing – it shows how our algorithms fit data in the first place.

The following network is implemented with different optimizer that is Adagrad and without batch normalization. This network gives us very less accuracy. Accuracy from the below network using TensorFlow is 20.25%.

The accuracy from our small dataset is low and we think it could be because of the lack of data. It could also be because of an unfit model for this data. Probably, the biggest challenge we faced in this project is the size of the data. It severely limited what we could actually produce and massively slowed down the training and testing. This approach would have been interesting if we have more data for our 10 numbers of classes. Instead, we have each model with less training data, it would also be very interesting to see the comparison of one model trained on the full dataset and another model on the smaller data. Distribution of classes was very unequal in the dataset, some classes had 10 products in one category and some of them had thousands, which is not very good, so for better training on late stages we might have to use some data augmentation.