Hninn Khin
DATS 6203 – Machine Learning II
Individual Report
12/04/2018

Image Classification on Google AI Open Image V4

The name of the dataset for this image classification project is called Google AI Open Image dataset. This dataset is available on Google Open Image Dataset V4 website. It was also available for Kaggle object detection and segmentation competition. It consists of natural images that reflect everyday scene and provides contextual information. There are 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 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, interpreting the model performance, results, editing group report and presentation. 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 are modifying the codes, uploading our final dataset from my GCP cloud instance to Google Cloud storage and created the instruction README.code.md on how to download our data with wget and unzip the data on ubuntu Linux and run our code for the project including Github repository Readme.md and editing and proofreading the final project report, reading past research paper and articles to understand our neural network framework and model, find appropriate reference code so that we can modify and implement it in our project.

During data argumentation process, our python did not run on cv2 previously and we kept getting error on cv2 packages and I modified the code using PIL packages.

However, our cv2 worked again later in the project and we ended up not using the code that wrote using PIL to resize our images and create the array in the final project code. I shared the code that I wrote in my 'mywork.py' Following is before (1024X760) and after(60X60) resizing with 3 channels.

We used the convolutional neural network on TensorFlow and also tested with Keras for our project because the main idea of the project is to be able to classify the images. CNN has been used in image classification, object detection and other applications in previous researches. A common choice for the structure of our CNN consists of layers of convolutional and max pooling layers which are then combined in a deep layer. It has size of the filter, the kernel size, strides, padding and activation function. For max pooling layer, the parameters are kernel size and strides. The functions applied in this project are RELU and SoftMax. In this case, we are using a max pooling layer which is a common choice for convolutional neural networks. We have also used other commonly used layers that are flatten and dropout layers. Dropout layers are used to prevent overfitting.

We found that implementing batch normalization improves our model from 35% accuracy to **50% in accuracy**. Accuracy of the 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 network is

implemented with different optimizer such as Adagrad and without batch normalization, the accuracy is 20.25% using TensorFlow.

Frameworks		Accuracy	Layers	Kernel size	Epochs
Tensorflow	No Batch Normalization	35%	4	5x5	200
Keras	No Batch Normalization	40%	4	5x5	20
Keras	With Batch Normalization	45%	6	5x5	20
Keras	K-fold cross validation	50%	6	5x5	20

In conclusion, the accuracy from our small dataset is low and we think it could be because of the lack of 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.