Project 3

CIS 365

Professor Moore

Deep Learning Flower Classification

Steven Bouwkamp

Matt Schuch

Andy Vuong

Techniques:

We originally started off making our own classifier. Our classifier had 15 layers, two of which were convolutional layers. We used the example from a Keras blog [1] as a guideline to how to create the layers. We then tested our own classifier on the 17 flowers dataset [2] and we got a surprising 70% accuracy. We obtained this value by not only using the 17 flowers dataset but also transforming each image to get more training images. Some of the transformations we used are horizontal flipping, zooming in, rescaling, and setting the fill mode to constant. However, that was as far as we got as we could not improve the model any further as the lack of data and the quality of the images prevented us from increasing the accuracy past 70%. As a result, we had to look elsewhere to improve the accuracy.

We decided to use a pre-trained model as they are trained with significantly more images and as a result, can detect features better than classifier that we made from scratch. We decided to use InceptionV3 as documentation of it was widely available, ease of use, and its size were beneficial to our goal. The InceptionV3 model had a significantly higher number of layers than the classifier we made, having around 300 layers compared to about 15 in the custom classifier.

After we got a high percentage, we decided to look another dataset that is like the 17 flowers dataset: the 102 flowers dataset [4]. Both are from the same site but we dismissed the 102 flowers dataset due to the images not being organized. However, we returned to the dataset after realizing the MatLab file had labels that organized the images so after writing a script that organizes the images based on flower class, we decided to use the 102 flowers dataset along with the 17 flowers dataset.

Parameters:

For each image, we did several transformations. The transformations we used were horizontal flip, zooming in, rescaling, and the type of fill mode. Rescaling is where we divide the possible RGB values, in this case, we divided the values by 255 which leaves us with the range of 0-1 with floats as possible RGB values (for regularization). The fill mode determines how Keras will fill in any black bars in the image, if there are any. In this case, we used the "constant" mode which essentially does nothing with the black bars as it fills the black bars with black bars.

Epoch was another parameter we experimented with. The number of epochs affected how long it took to train the pre-train model. We first started with a small number of epochs for the 17 flowers dataset and it yielded a pretty good accuracy. As we increased the epoch higher and higher, we got slightly better accuracy each time. At around 150 epochs, we obtained the best accuracy as any higher would make the model overfit the data set so we stopped at around 150 epochs for the training. The 102 flowers dataset was quite different as a low number of epochs

yielded a low accuracy of around 60-70%. However, when we increased the epoch to around 600, we obtained the best accuracy for the data set.

The number of layers we froze was another parameter we experimented with. The documentation had the first 172 layers frozen while the rest were kept as they were. This means that the layer does not change or affected by the data that passes through. The data is still affected however and we freeze the first layers to prevent overfitting from occurring. We experimented a bit with the number of layers frozen, such as freezing only 50 layers or 100 layers. However, we noticed that the accuracy was at its best at the given 172 layers frozen so we used that value for our model.

Results:

With our created classifier, we got a 70% accuracy which is quite impressive for a classifier created from scratch. Later, we switched to a pre-trained model as we could not improve the 70% accuracy. With the pre-trained models, we got some pretty good accuracies. For the 17 flowers dataset, we managed to obtain a 95% accuracy with 50 epochs and after increasing the epochs to around 150, we obtained the most optimal accuracy at around 98%. The 102 flowers dataset initially had a low accuracy when we used a small number of epochs, at around 60-70%. However, after steadily increased in the epochs to around 600, we obtained the best accuracy at around 95%.

Our conclusive results are a model with 98% accuracy on any of the flower types in the 17 flowers dataset and a model with 95% accuracy on any flower types in the 102 flowers dataset.

Problems/Difficulties Encountered:

Our first difficulties encountered happened during the setup phase of our project. We had some trouble with the TensorFlow updates, the Keras updates, and some cuDNN problems. The TensorFlow update caused some problems that were fixed with a nightly build so we had to download the fix. Some of us had cuDNN problems as if it was not installed correctly and the files were placed correctly, TensorFlow would not work. Keras had a huge update with their libraries right when we began work on the project so we had to update Keras accordingly so it matches up with the documentation.

Our other and main difficulty was finding a good dataset. The quality of our model and its accuracy hinges on the dataset. We had troubles finding a good leaf dataset so we eventually switched to flower data sets as we found some good ones from Oxford. While we did achieve good accuracies with both the 102 flowers dataset and the 17 flowers dataset, if we had more pictures, flower types, more variety of pictures, and better quality pictures than we would be able to train a model that would be much more accurate than what we currently have.

References:

[1] https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

[2]
http://www.robots.ox.ac.uk/~vgg/data/flowers/17/index.html

[3]
https://keras.io/applications/

[4]
http://www.robots.ox.ac.uk/~vgg/data/flowers/102/index.html

[5] General Convolutional Neural Network Source
http://cs231n.github.io/convolutional-networks/