Fruits Classification Using CNN: Individual Final Report – Henry Tappa

Introduction

Image classification tasks using Convolutional Neural Networks (CNN) have been an active area of research in recent years (Li et al., 2017). They take advantage of the unique way computers interpret images by using kernels to identify key components of an image. These kernels are convolved over individual images to obtain feature maps of the specific features that appear in the image (Saxena & Rarr, V. 2019).

We buit a fruit classification network for this project using a 2-dimentional CNN architecture. The CNN was built in Python using Pytorch as the framework and was run in the cloud using a Google Cloud Platform instance configured with 1 GPU. Potential applications of this network include automating Customs and Border Protections (CBP) processes or for use in the agricultural industry (Vibhute and K. Bodhe, 2012).

The data set that we used was the “Fruits 360 dataset,” a Kaggle dataset with over 20,000 images of 33 types of fruit (Kaggle.com, 2017). The images are full-color, 100x100 pixels and treated as such in the network. The data, as structured, was split into a training set with 15,506 images and a testing set with 5,195 images.

Project Development and Individual Work

I started out building the CNN network, as well as the training and testing loops, using my previous homework and exams as reference points, as well as the online documentation for Pytorch. Using `nn.Module`, I created a class object with two `nn.Sequential` layers. Initially, I used `nn.Conv2d`, `nn.ReLU`, and `nn.MaxPool2d` within these layers. I knew that the input channels for the network had to be “3” since we were using color images, and the output features of the network had to be equal to the number of classes. As far as all of the other intermediate values in the network, including kernel size and padding, I found that the best way to determine what these values should be was through trial and error. After the other team members input the code to load and preprocess the data, we ran the code on a subset of the data with just five classes (2,432 training images and 814 test images) to see how the algorithm would work on this data. We found the accuracy to be about 95%. Another team member recommended adding a `nn.BatchNorm2d` module to the network in order to normalize the mini-batches. After I added `nn.BatchNorm2d` to our model, the accuracy rose to 100%.

We then decided to test this algorithm on the full data set, with 33 classes. This performed better than we expected, with 97% accuracy. Throughout this process, I was constantly trying new combinations of values in the CNN network to improve on this and found that the best combination was starting with 3 input channels, 32 output channels, a kernel size of 5, and padding of 2. In order to see where our misclassifications were, I input code to build a confusion matrix. After defining a blank confusion matrix, I added code within the testing loop to add label values and predicted values to the confusion matrix. After printing this confusion matrix and finding it to be too large to fit within the confines of the terminal, I decided to instead convert the matrix to a dataframe and transfer it to a .csv file.

I wanted to visualize the loss from our training loop as it iterated over each epoch, so I created a blank list for loss values, and added code to the training loop that appended the loss values for each batch to this list. I also wanted to create a similar plot for accuracy over each epoch, so I included code in the testing loop that calculated the accuracy and added it to an accuracy list. I was having difficulty getting the Matplotlib plots to display correctly, but fortunately one of the other team members was able to fix the code so they would display properly. These plots are displayed below in Figure 1 and Figure 2.

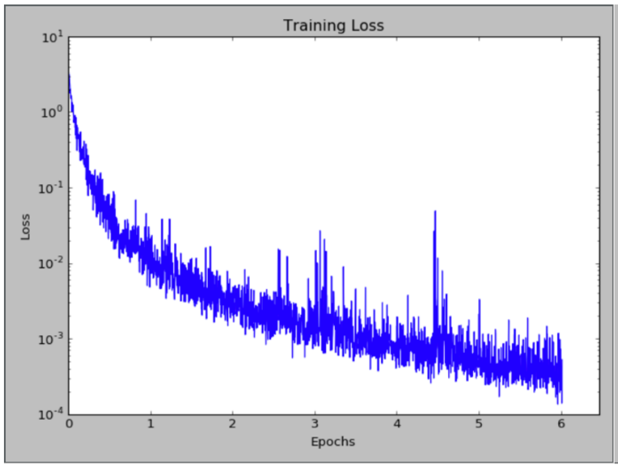

Figure 1. Cross-Entropy Loss, 2-Layer Network

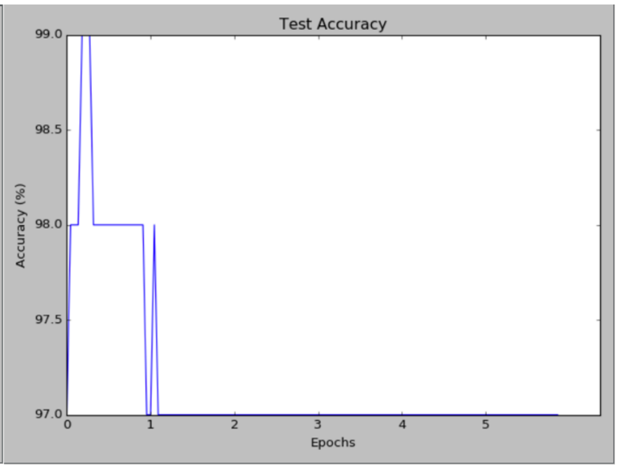

Figure 2. Test Accuracy, 2-Layer Network

We then decided to add a third layer to our network to see if this would increase accuracy. I added a third sequential layer to the model and adjusted the channel values appropriately. This made quite an improvement, bringing accuracy to 99%. Once all the models were running properly, I split the initial python file into three separate files: `fruits_cnn_subset.py`, `fruits_cnn_all_2layer.py`, and `fruits_cnn_all_3layer.py`, and updated the `readme.md` file to reflect this and ensure that the code could be easily understood.

Results

The first model, which used a 2-layer network on a subset of the data with 5 classes, reached 100% accuracy after just 6 epochs. The second model, which used the same 2-layer network described earlier on the whole dataset with 33 classes and the same number of epochs, had an accuracy of 97%.

Increasing the epochs did not substantially change the accuracy of this model. Therefore, as mentioned previously, we added a third layer to see if training the network even further could have a significant effect on the accuracy. The new 3-layer model had an accuracy of 99%, and the only misclassifications were between Braeburn and Nectarine, with 33 misclassifications. Though the model did learn to correct itself with all of the fruits that were misclassified in the 2-layer network, it still had trouble distinguishing between these two particular fruits.

Summary

The CNN model that we developed for this project was able to classify the test set from the Fruits 360 dataset of 5,195 images with 99% accuracy, which we determine to be a very successful classification rate. This was an incredible team effort through which we learned a lot about CNNs and how to collaborate effectively on future machine learning projects. For further research, we could investigate the misclassifications for “Braeburn” and “Nectarine” and determine how these could be mitigated. The images in the Fruits 360 dataset were specifically processed for the purpose of machine learning, so further research could also involve testing the network on raw, unprocessed photographs of fruit, and making further adjustments based on the results of these classifications.

Works Cited

W. Li, G. Wu, F. Zhang and Q. Du, "Hyperspectral Image Classification Using Deep Pixel-Pair Features," in *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 2, pp. 844-853, Feb. 2017. doi: 10.1109/TGRS.2016.2616355

Saxena, A. & Rarr, V. (2019). Convolutional Neural Networks (CNNs): An Illustrated Explanation - XRDS. [online] XRDS. Available at: https://blog.xrds.acm.org/2016/06/convolutional-neural-networks-cnns-illustrated-explanation/ [Accessed 20 Apr. 2019].

Kaggle.com. (2017). Fruits 360 dataset. [online] Available at: https://www.kaggle.com/moltean/fruits/version/2 [Accessed 6 Apr. 2019].

Vibhute, A. and K. Bodhe, S. (2012). Applications of Image Processing in Agriculture: A Survey. International Journal of Computer Applications, 52(2), pp.34-40.