# Fruit Classification Using Convolutional Neural Network

Ryan Kinsey – Individual Report

The George Washington University

## Introduction

For this project we created and implemented a Convolutional Neural Network (CNN) on a dataset of fruit images. There was a total of 15,506 training images and 5,195 testing images from 33 different image classes. The CNN was implemented in Python using Pytorch as the framework and GCP. We treated this project as a research opportunity and the breakdown of responsibilities are below.

- **Ryan Kinsey**
    - Code the data loader section
    - Help code the visualizations for loss/accuracy
    - Take the lead on the Tech Report
    - Literature review of similar work
    - Based on research, make suggestions to our network to boost accuracy
    - Draft the Introduction, Dataset, CNN sections in the report
    - Contribute to the Experimental Setup and Results sections in the report
    - Create figures for the Report

- **Eric Goldman**
    - Significant contributions to the code
    - Code the confusion matrix section
    - Code the section to visualize fruits in the minibatches
    - Experiment with a 'subset' dataset to get our model running properly
    - Identify fruits that were misclassified
    - Take the lead on the PowerPoint presentation
    - Draft most of the presentation

- **Henry Tappa**
    - Code the Conv Net architecture
    - Figure out the proper sizes for input/output to the layers
    - Set the code up to run on GPU (cuda)
    - Code the train model section
    - Code the test model section
    - Help code the visualization for loss/accuracy
    - Drafted the conclusion section of the Tech Report

## Individual work

As you will see in the "Introduction" and "Convolutional Neural Network" sections of the Technical Report much literature review was done. I spent a great deal of time researching previous similar work. This research was done with the goal of guiding our project with empirical evidence-based reasoning.

For example, at the beginning of the project we had debated whether or not to grey-scale the images or keep them color. I insisted on keeping the images color, but since we had no prior experience with color image processing some research needed to be done to understand the differences. The main thing we needed to understand was the RGB color channels that we had not previously had experience with (Udofia, 2018). Additionally, I had debated adding a dropout layer to the architecture, but through research came across the idea of BatchNorm2d. I was able to find out that by applying normalization to the mini-batched we could potentially reduce the need for a dropout layer (Ioffe and Szgedy, 2015). Lastly, I took time to understand why a non-linear activation function was typically applied to CNN's. Essentially what I learned was that by changing all negative vales to 0 it can protect against the vanishing/exploding gradient problem (Xu et al., 2015).

For the coding aspect, I was tasked with creating a data load section. I spent some time looking at the Pytorch source code to better understand the ImageFolder.() and Data.Loader() functions. I also came across some code that normalized the image data while transforming to tensor. It is important to normalize the data because when multiplying weights and adding bias values we want the gradients to behave well. In other words, if the image is not normalized there could be situations where the gradient calculations get crazy and unmanageable.

## Results

When our 2-layer CNN returned an accuracy of just 97%, I suggested that we add a 3rd layer. My thinking behind this was that the underlying function of the data was too complex for the 2-layer CNN. Adding a 3rd layer would allow for a more robust model and our accuracy increased immediately to 99%.

Another task I took on was analyzing the confusion matrices. This required some indexing to try and pinpoint the fruits that had high levels of misclassifications. I created the below figures to represent our findings.

| | Apple Red 1 | Cherry | Grape | Kiwi | Quince |
|---|---|---|---|---|---|
| Apple Red 1 | **164** | 0 | 0 | 0 | 0 |
| Cherry | 0 | **164** | 0 | 0 | 0 |
| Grape | 0 | 0 | **164** | 0 | 0 |
| Kiwi | 0 | 0 | 0 | **156** | 0 |
| Quince | 0 | 0 | 0 | 0 | **166** |

**Figure 3.** Confusion matrix for fruit_data_subset

| | Apple Red 1 | Apple Red Yellow | Braeburn | Cherry | Golden 3 | Granny Smith | Nectarine | Peach |
|---|---|---|---|---|---|---|---|---|
| Apple Red 2 | | | **10** | | | | | |
| Braeburn | | | | | | | **36** | |
| Nectarine | | **19** | | | | | | **13** |
| Peach | **17** | | | | | | | |
| Pear | | | | | **26** | **4** | | |
| Plum | | | | **11** | | | | |

**Figure 4.** Number of Misclassifications, 2-Layer Network

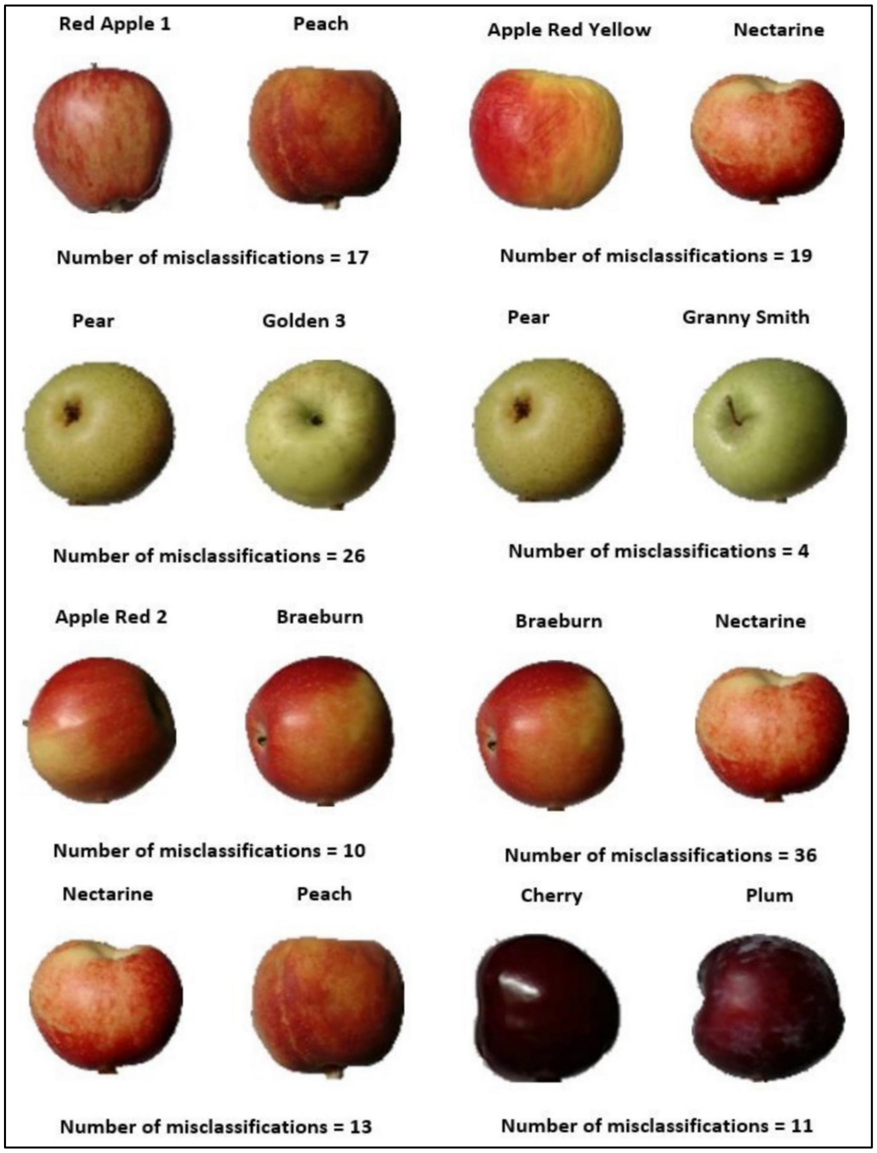
Red Apple 1
Peach
Apple Red Yellow
Nectarine
Number of misclassifications = 17
Number of misclassifications = 19
Pear
Golden 3
Pear
Granny Smith
Number of misclassifications = 26
Number of misclassifications = 4
Apple Red 2
Braeburn
Braeburn
Nectarine
Number of misclassifications = 10
Number of misclassifications = 36
Nectarine
Peach
Cherry
Plum
Number of misclassifications = 13
Number of misclassifications = 11

**Figure 5.** Fruit Misclassifications, 2-Layer Network

Above, figure 3 was produced using the confusion matrix from our 'subset' dataset. When we were first starting we chose to limit our dataset to 5 classes until we got a well running model. Relatively easily we were able to achieve 100% accuracy on these 5 classes of fruits that visually appear very different.

After we had our model running well we ran it on the overall dataset to get figure 4. These were the misclassification for the 2-layer model. Figure 5 is a visual representation of those misclassifications. As you can see, many of them are very hard to distinguish from one another. It was after this that I suggested adding a 3rd layer to boost accuracy and Henry quickly added a 3rd layer and we achieved 99% accuracy.

Henry and I worked together to visualize the loss and accuracy vs. epochs. Given the way our training and testing for-loops were written in the code, the initial output was not intuitive. It took me some time, but I figured out that the loss and accuracy data are plotted per mini-batch training. The way our dataset was split using minibatches we ended up with about 2,300 mini batches that were then split up into iterations per epoch. Knowing this I was able to reformat the axes of our plots with the below code. It's a convoluted way to visualize the loss and accuracy, but the end result represents the model's learning as it advances in epochs. Below are these plots.

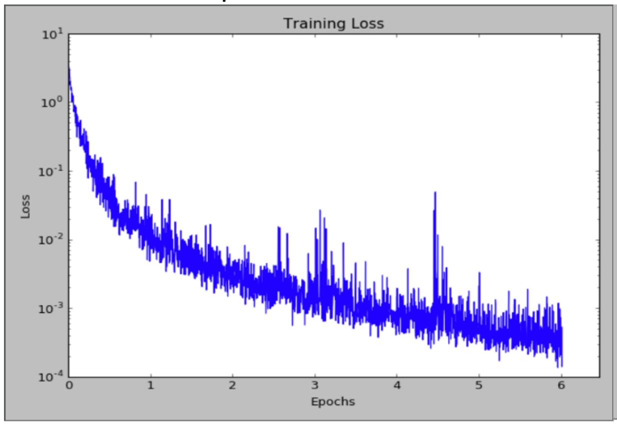

**Figure 6.** Cross-Entropy Loss, 2-Layer Network

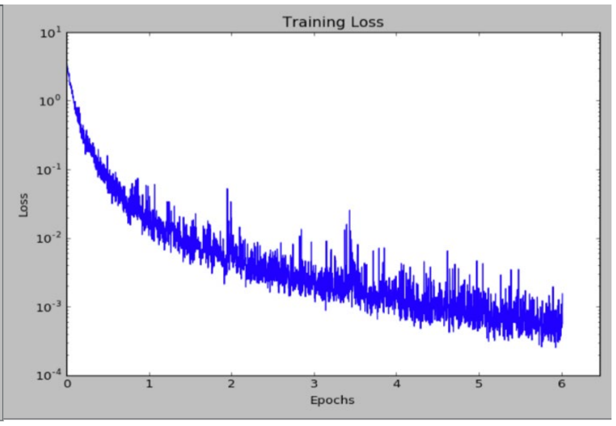

**Figure 7.** Cross-Entropy Loss, 3-Layer Network

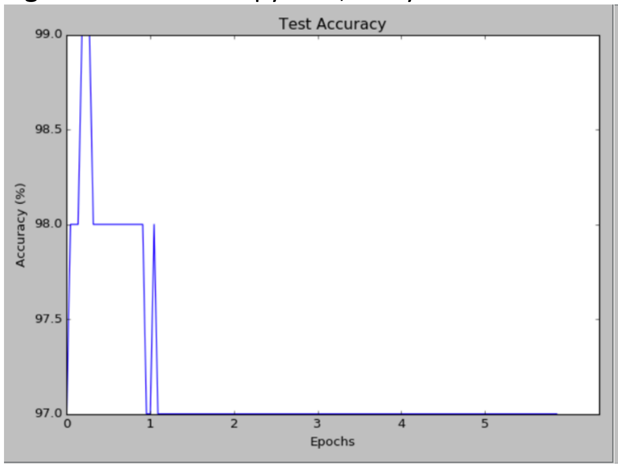

**Figure 8.** Test Accuracy, 2-Layer Network

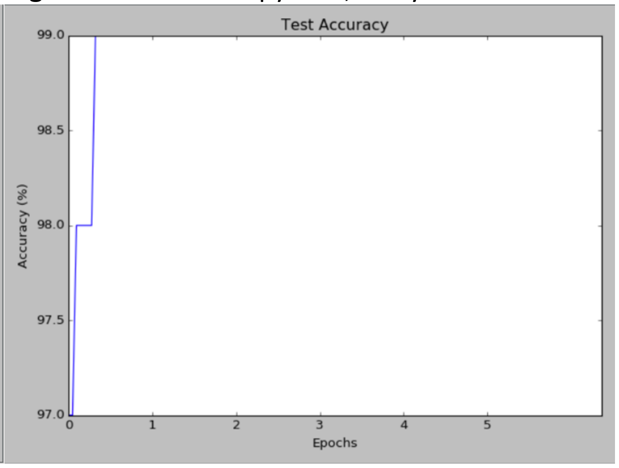

**Figure 9.** Test Accuracy, 3-Layer Network

## Summary

We set out to create a convolutional neural network using fruit data that could reliably classify different fruit types. We cited a few different situations where software of this nature could be used, but ultimately our project was for learning purposes. At the core of it, we wanted to create a 2D CNN using color images. This was something new we hadn't had experience with and we chose to use Pytorch since this dataset had very little prior research using Pytorch.

Whether it was theoretical or coding, I would say that the research that I preformed was the most valuable experience I will take from this project. I found myself really studying source code to understand exactly what they would do. Additionally, I found myself researching different parameters of the model. I remember spending a good deal of time to try and figure out which optimizer and loss function would be most appropriate. I found that many people have their opinions, but general consensus is that CrossEntropy and ADAM are the most appropriate for a network of this nature.

One thing I would improve about this project was the framework we chose. I really wanted to explore the kernels and feature maps, but I found that Pytorch does not make it easy to do so. We chose to use Pytorch since very few people had used it for this dataset. All things considered; I think using tensorflow would have been better so that we could take advantage of tensorboard.

My code contributions calculate as following:

(2 – 7) / (2 + 14) x 100 = **31.25**

## References

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

Udofia, U. (2018). [online] Available at: https://www.researchgate.net/publication/325803364_A_Study_on_CNN_Transfer_Learning_for_Image_Classification [Accessed 17 Apr. 2019].

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].

Vanderhorst, D. (2018). CBP intercepts prohibited Mexican fruit [online] Available at: https://www.thepacker.com/article/cbp-intercepts-prohibited-mexican-fruit [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.

Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. [online] arXiv.org. Available at: https://arxiv.org/abs/1502.03167v3 [Accessed 16 Apr. 2019].

Xu, B., Wang, N., Chen, T. and Li, M. (2015). Empirical Evaluation of Rectified Activations in Convolutional Network. [online] arXiv.org. Available at: https://arxiv.org/abs/1505.00853 [Accessed 17 Apr. 2019].

Yamashita, R., Nishio, M., Do, R.K.G. et al. Insights Imaging (2018) 9: 611. https://doi.org/10.1007/s13244-018-0639-9

Kingma, D. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. [online] arXiv.org. Available at: https://arxiv.org/abs/1412.6980 [Accessed 19 Apr. 2019].