Debugging for a Week Straight!
San Francisco State University
By: Amayrani Villegas, Abhi Wagh, Jocelyne Milke, Andre's Williams

Introduction

Hitting a wall while coding happens pretty often, regardless of how much experience or knowledge you have. Debugging, or trying to find those few errors in your code can be a frustrating process. But once you figure out exactly why your code isn't working, it can be extremely rewarding! Plus, you learn from your mistakes and become a smarter coder.

My group and I went through a similar situation, where our code for a project wasn't working at all. No matter how many times we messed around with it or the many different kinds of approaches we used to fix the errors, the code would return the same 1 or 2 errors. This continued for a week, until eventually we had a breakthrough! We realized we had to look at the code in chunks and zoom out to see the bigger picture.

Our project was based on a concept called machine learning, where you can train a computer to detect and analyze patterns in datasets. Inputs and expected outputs are provided to a machine learning model that is trained to analyze the data (inputs). Then, the model is tested against the expected outputs to measure how accurate its predictions are. In this way, the model is "learning" how to differentiate between different features of any given dataset. In our project, our data consisted of images. The model had to differentiate between pictures of waterfalls versus food.

Using Python in a Google Colab Notebook, we used the VGG16 model from the Keras library to accurately categorize our images. The VGG16 model is an example of a **Convolutional Neural Network (CNN)**, which is a type of machine learning typically used to analyze images. CNN's have multiple layers of processing, and computers are able to use CNN's to break images down into pixels and numbers. These pixels and associated numbers form a spatial feature map that allows computers to tell the differences and similarities between images.

Setting up the Python Coding Environment

We carried out our project using Python on a Google Colab notebook. It is possible to use other types of integrated development environments (IDE); however, we preferred Google Colab because you can create shared folders that all colleagues can have access to. Essentially, we defined our set of images and trained a VGG16 model to recognize them. First, we created a folder called FoodvsWaterfalls where we had two sub-folders, each representing one of our two classes, food and

waterfalls. Each group member uploaded images they had of either waterfalls or food from their cell phones. We then verified the format of the image was jpg or jpeg (ex: tacos.jpeg) and sorted them in their appropriate folder.

Setting up the Correct Folder Structure

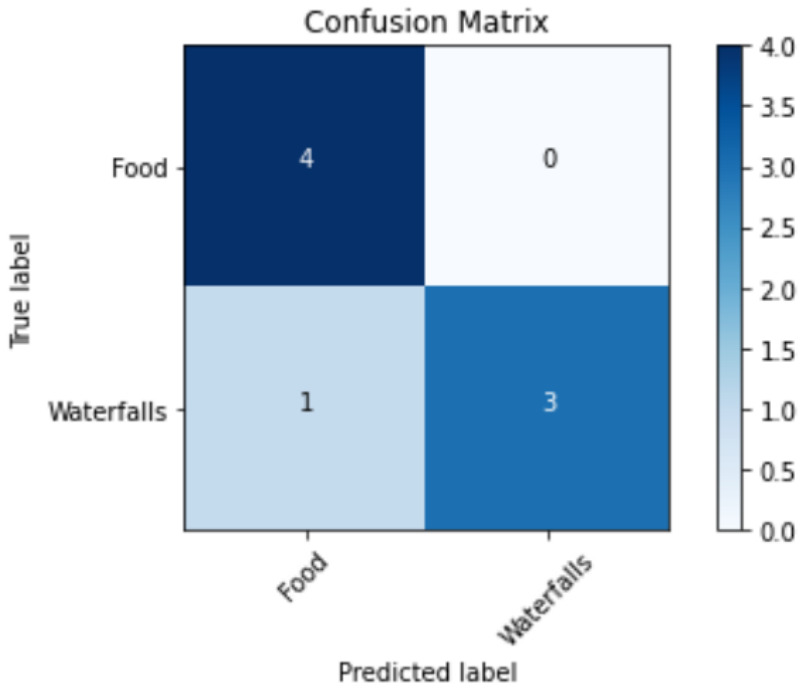
Next, we imported all packages needed for the project, mounted our Google Drive, and set our directory to our project folder. This would allow us to make changes to our Google Drive directly within our code. We then created a new folder using Python that we called FoodvsWaterfallstest2. To build a well performing model, it is essential to first train the model and then test it against data that comes from the same target. For this reason, we resized our images to ensure they were all the same size and copied over our images into a new folder and divided it into three subfolders called training, testing, and validation.

Creating a Machine Model

Now that our images were set up in our project folder (called FoodvsWaterfallstest2), we were able to move forward with the machine learning aspect of the project. We created a new Google notebook. We would like to extend a special thank you to Dr. Ilmi Yoon as this aspect of our project is possible thanks to their contribution. We first imported all the packages required and set our directory to the FoodvsWaterfallstest2 we created above. Before we created our model for our images, we downloaded a pretrained VGG16 model using the Keras library.

Measuring Accuracy

We trained our model to predict which images represented a waterfall versus food. After training it, we were able to compare the predictions the model made against the true, expected outputs of the data. We then obtained a confusion matrix, seen below. The confusion matrix shows how many times the model correctly classified an image compared to incorrectly. Then, we tested the accuracy of our model. It was calculated to be 87.5%, which tells us the model correctly classifies the images from our dataset 87.5% of the time when we run the code.

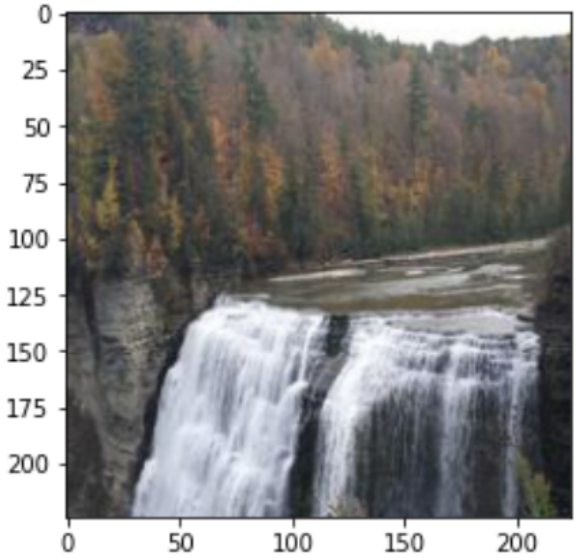


Labeling Our Images

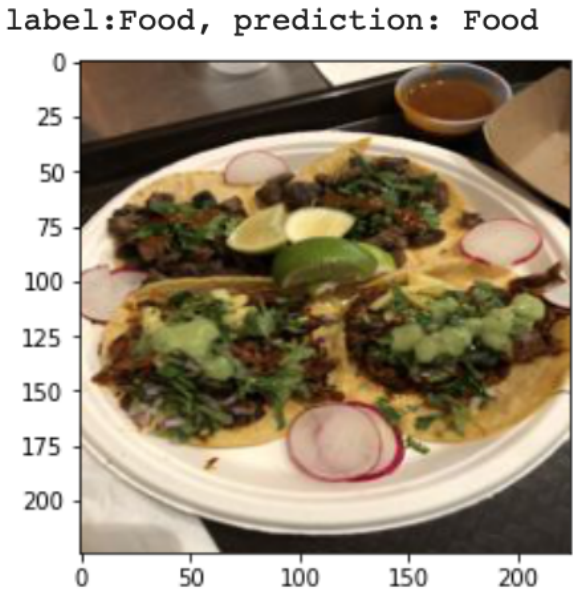
To finish off, we looked at our test images and created labels. We did this to organize our data and also evaluate how well the model predicted the categories. Below, you can see both incorrect and correct predictions made by our model.

Incorrect prediction :(

```
label:Waterfall, prediction: Food
```



Correct prediction :)



Overcoming Difficulties With Model Constructions

We originally planned to use photos of logs and flowers from a hiking trip for our machine learning model. We were splitting our images into their respective sub-folders, when we ran right into the dreaded ERROR message. We went through our code to figure out what was going on and found the first culprit: our source path was incorrect.

We changed the source path to match where the folders were located in our own Google Drive. The code continued to run until we got to the very last step-of course. Another error message popped up, telling us something was wrong with an image. So we combed through our folders, found a few duplicates, and removed them. Unfortunately, we got the same error message but with a different image identified. We then thought that something was wrong with all of the images, perhaps the fact that they all had different sizes. With the help of Dr. Pennings, we resized our images and tried again!

But the error message still stood...We tried one last thing: we changed all of the images to JPEG format and reuploaded them into a new shared folder. We ran the code from the start but got the same error. We even made a copy of the whole assignment and ran the code again (kind of like turning something on and off again). We ended up at the drawing board once more.

Ultimately, we ended up completely switching directions and using random pictures of food and waterfalls that we had on our phones instead. The code worked well with these images, and we were able to get nice results! This just goes to show that sometimes things don't work out how you want or expect them to, but that's completely okay! Working through the code helped us identify some debugging practices and also taught us to be flexible.

Resources

1. https://medium.com/@mygreatlearning/what-is-vgg16-introduction-to-vgg16-f2d63849f615

2. https://blog.paperspace.com/facial-recognition-using-deep-learning/
3. https://www.wgu.edu/blog/what-convolutional-neural-network2008.html#close

4. https://www.geeksforgeeks.org/ml-machine-learning/