

Chihuahua vs Muffin

Over the past week, I've experienced both frustration and small victories while working on this project. The goal of my neural network is to differentiate between a chihuahua and a blueberry muffin. Although this task sounds straightforward, it turned out to be quite complicated. Many steps that appeared easy were much more difficult in practice. I started by using Google Colab but ran into trouble inspecting the data. When I encountered the error "No such file or directory: 'data'" while loading the data. I decided to switch from Colab to a Jupyter notebook.

My first challenge was setting up the environment. I repeatedly faced `ModuleNotFoundError` for 'torch' and 'autogluon', which was frustrating because I thought everything was installed. I tried pip installing the missing packages several times, hoping it would resolve the issue. Eventually, I realized that my Jupyter Notebook was connected to a different Python environment. After switching the kernel to the correct environment, the imports finally worked.

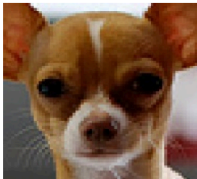
The second biggest challenge I faced was importing the `chihuahua-vs-muffin.git` repository. Each time I used the command you suggested, "`!git clone...`" in my Jupyter notebook, I received a syntax error. I tried reinstalling various components to resolve the issue, but ultimately, I returned to Google Colab. That switch turned out to be the key to resolving my original problem. The "No such file or directory" message popped up because I hadn't cloned the repository before running the code.

The final challenge happened while I was setting up the neural network. I was using the `MyModule()` class instead of the `MySkynet()` class, which was the source of the error I was getting. With everything fixed, I could finally see my model's predictions.

First, I set the number of epochs to six to prevent overfitting, keeping this parameter low. I observed that increasing the image size reduced accuracy, so I chose to use a size of 244. I added an input layer to the network architecture, but this didn't significantly impact performance. Switching the optimizer from SGD to Adam greatly improved my model's accuracy. Finally, I adjusted the learning rate to help the model learn faster, given the limited number of epochs.

This was my highest accuracy output

100% Chi, 0% Muff



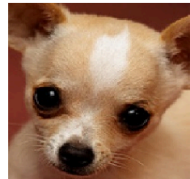
100% Chi, 0% Muff



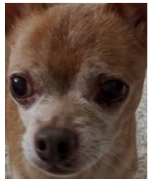
100% Chi, 0% Muff



100% Chi, 0% Muff



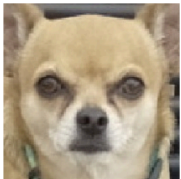
100% Chi, 0% Muff



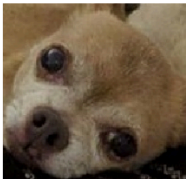
100% Chi, 0% Muff



100% Chi, 0% Muff



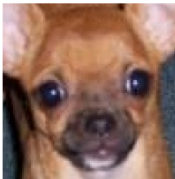
100% Chi, 0% Muff



100% Chi, 0% Muff



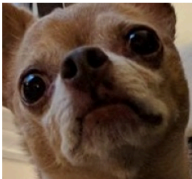
100% Chi, 0% Muff



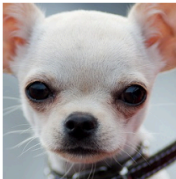
100% Chi, 0% Muff



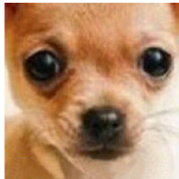
100% Chi, 0% Muff



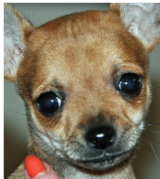
100% Chi, 0% Muff



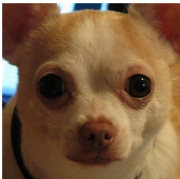
100% Chi, 0% Muff



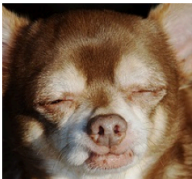
57% Chi, 43% Muff



100% Chi, 0% Muff



100% Chi, 0% Muff



100% Chi, 0% Muff



0% Chi, 100% Muff



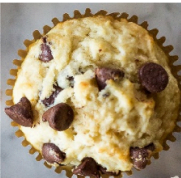
100% Chi, 0% Muff



0% Chi, 100% Muff



0% Chi, 100% Muff



0% Chi, 100% Muff



6% Chi, 94% Muff



1% Chi, 99% Muff



0% Chi, 100% Muff



0% Chi, 100% Muff



0% Chi, 100% Muff



92% Chi, 8% Muff



34% Chi, 66% Muff

