Chloe Tu
ITAI 1378
September 17, 2025

Chihuahua or Muffin Workshop Reflective Journal

Summary of the Workshop

This workshop provided a hands-on introduction to image classification using deep learning. I learned how to build a machine learning model to sort pictures into groups. My main goal was to create a simple neural network that could tell if a picture showed a chihuahua or a muffin. I started by getting everything set up in Google Colab. I copied a project from GitHub that had the code and pictures I needed. The main way I taught the computer was by building a neural network in PyTorch. I showed it pictures, and it guessed what they were. Then, I calculated how wrong its guesses were using something called a loss function. Based on that error, I used an optimizer to change the network's settings so it would do better next time. This learning process is called backpropagation.

Key Concepts Learned

I learned many important ideas about image classification. This is the process of teaching a computer to sort images. I worked with neural networks, which are like layers of connected nodes that help the computer process information. I used PyTorch to build a simple network and its different layers, like linear layers with activation functions such as ReLU and Softmax. I also learned how to get image data ready for the network. This meant using transforms to resize the pictures and make them the right format for the network to understand. The network learns from its mistakes during training. A loss function, like nn.CrossEntropyLoss, tells the network how wrong its guesses are. Then an optimizer, such as Adam or SGD, adjusts the network to make it more accurate.

Challenges Encountered and How They Were Overcame

Getting the model to be perfect at telling chihuahuas and muffins apart was a real challenge. When I first trained the basic model, it wasn't 100% accurate. I tried changing some settings, like how many times the model looked at all the pictures (epochs), how fast it learned (learning rate), and using different learning methods (optimizers like Adam and Adagrad). I did this in code step 1 and later in steps 5.1, 5.2, and 5.3. These changes helped a little sometimes, but the accuracy still wasn't perfect. I thought maybe the network wasn't smart enough, so in code step 2, I changed how it was built. I added another layer and made the existing layers bigger so it could learn more complex things. I also realized the pictures themselves could be a problem, so in code step 3, I made the pictures bigger and added ways to slightly change the training pictures, like flipping them or changing their colors a bit. When I trained the model again with these changes (code steps 4 and 5), the accuracy on new pictures got much better, reaching up to 93.33%. Even though I kept trying different settings in later steps, reaching 100% accuracy was still hard. It showed me that making models perfect is a common difficulty.

Insights Gained

I learned that building a good machine learning model is about more than just its design. How you get the data ready and set up the learning process matters just as much. I saw that changing small things like the learning rate or the number of training times was not enough to make a big difference. The real improvement came from making the model smarter by adding more layers and from making the training pictures more varied. This showed me that the model's structure and the quality of the data are very important for good results. I also saw that even when my improved model was perfect at classifying the pictures it learned from, it sometimes still made mistakes on new pictures. This taught me why testing a separate set of pictures is so important. It taught me how hard it is to build a model that will always work perfectly in the real world.

Potential Real-World Applications

The methods I used to tell chihuahuas and muffins apart have many uses in the real world. Sorting images is used in important things like helping doctors find diseases in medical scans, in self-driving cars to spot things on the road, in security cameras to recognize people, and even in phone apps that can identify plants or animals from a photo. Being able to train computers to look at pictures and sort them is a basic part of many technologies we use every day.

Personal Reflections

Thinking about classifying chihuahuas and muffins was a deeply interesting and personal part of this experience. At first, it seemed like a funny and simple idea. But as I thought about it more, it revealed how tricky it can be for computers to understand pictures. Chihuahuas and muffins can look similar sometimes, especially if the picture isn't very clear or the lighting is different. They might both be small and have a similar round shape. This is why building a computer model that can tell them apart perfectly is much harder than one might think. The model must learn to see very small, subtle differences. It needs to look for specific features like the presence of eyes or a nose, which you would see on a chihuahua, or the bumpy, uneven top and little dark spots from berries on a muffin. The fact that my model did not easily achieve 100% accuracy on the validation pictures, even after I tried to make the network smarter and added more varied training pictures, really showed me how difficult this visual task is for a machine. It made me realize how truly amazing humans are at just knowing the difference right away, no matter how a picture is framed or lit. This seemingly simple example truly made me think about all the complex steps involved in teaching computers to "see" and understand images the way I do.

Reference

- patitimoner. "GitHub - Patitimoner/Workshop-Chihuahua-Vs-Muffin: Workshop 1, Simple Deep Learning Neural Network Training." GitHub, 2023, https://github.com/patitimoner/workshop-chihuahua-vs-muffin. Accessed 17 Sept. 2025.
- "Torch.optim — PyTorch 2.7 Documentation." Pytorch.org, 2024, docs.pytorch.org/docs/stable/optim.html. Accessed 17 Sept. 2025.
- IBM. "What Is a Neural Network?" IBM, 6 Oct. 2021, www.ibm.com/think/topics/neural-networks. Accessed 17 Sept. 2025.
- Bergmann, Dave, and Cole Stryker. "What Is Backpropagation? | IBM." Ibm.com, 2 July 2024, www.ibm.com/think/topics/backpropagation. Accessed 17 Sept. 2025.
- patitimoner/workshop-chihuahua-vs-muffin. Retrieved from https://github.com/patitimoner/workshop-chihuahua-vs-muffin. Accessed 17 Sept. 2025.
- GeeksforGeeks. "PyTorch Loss Functions." GeeksforGeeks, 11 Oct. 2023, www.geeksforgeeks.org/deep-learning/pytorch-loss-functions/.Accessed 17 Sept. 2025.
- GeeksforGeeks. "PyTorch Loss Functions." GeeksforGeeks, 11 Oct. 2023, www.geeksforgeeks.org/deep-learning/pytorch-loss-functions/. Accessed 17 Sept. 2025.
- "Torch.optim — PyTorch 2.7 Documentation." Pytorch.org, 2024, docs.pytorch.org/docs/stable/optim.html. Accessed 17 Sept. 2025.
- TensorFlow. "Data Augmentation | TensorFlow Core." TensorFlow, www.tensorflow.org/tutorials/images/data_augmentation. Accessed 17 Sept. 2025.