Saima Sano

ITAI-1378

Patricia McManus

**Workshop-Chihuahua-vs-Muffin**

This workshop was highly engaging, primarily because I delved into the code with greater detail, aided by the strategically placed question marks throughout the script. The workshop utilized a neural network architecture featuring two hidden layers to facilitate machine learning. It transformed images linearly before applying activation functions, such as ReLU, to introduce non-linearity into the model. Additionally, the workshop explored various parameters, including batch size, learning rate, and the number of epochs, to optimize the dataset and minimize the loss function. The key concepts I acquired through this lab include:

**Logistic Regression:**

The first thing I learned is that to train the neural network, we must work on logistic regression. To classify the image, we must combine the simplicial complexes of geometry, where point, line, and other high dimensional shapes exists and statistical probability to find best function to minimize the error and train the ML to better image processing.

**Neural Network:**

Neural Network is a family of functions which have many parametrized functions. We have to minimize the loss or error; therefore, we find the best function available from NN to support our project.

**Linear Separator:**

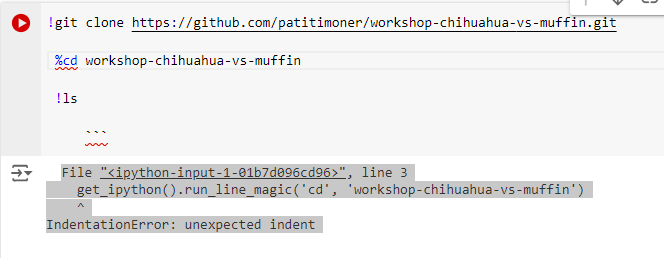
In this lab, we used linear separator for this data to make machine learn the difference between chihuahua and muffin. The “nn.Linear” is linear layer in the neural network, which is used to construct feedforward neural networks in PyTorch. We have 4 layers in this lab, the two hidden layer applies linear transformation to the input it receives.

**Activation Function:**

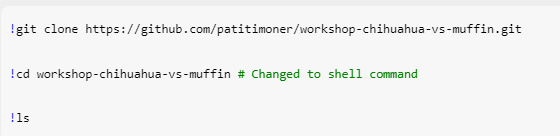
In this lab we are using Activation function ReLU (Rectified Linear Unit). The image from real-world is not linear, the linear transformation done by “nn.Linear” is not enough to capture the complex patterns in the data. By introducing ReLu activation function, we are introducing non-linearity into the model. This allow NN to learn non-linear mapping from input to the output. This also mitigate the vanishing gradient problem by maintaining a gradient of 1 for positive inputs. This property facilitates faster convergence during training, allowing networks to learn more effectively (Glorot & Bengio, 2010).

**Challenges I have encountered:**

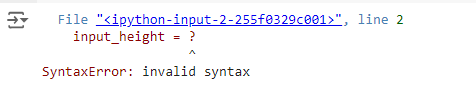
When I was cloning the notebook, I got this error.

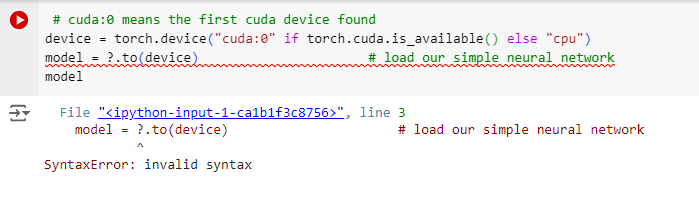


I asked Google Colab to explain the error and provide ways to fix it. It suggested the command “%cd” is IPython magic commands and my notebook is in regular Python script. Therefore, if I want to clone the file, I have to change “%cd” to “!cd”. this feature of Colab resolved the issue for me quickly, so I was able to move to next step.



The next challenge was to fill the missing weight and height in the code. In define image height and width, the input value was question mark. At first, I was not able to understand what this was asking. I asked ChatGPT and it guided me to look at my image dataset to find out the height and width of the image. My curiosity also led me to ImageNet dataset. The reason the images are set to 224x224 size because it offers balance between computational efficiency and sufficient image resolution for detailed feature extraction in classification task. (Krizhevsky et al., 2012).





This was the third blank in this code. Here, I must hold an instance of neural network class to variable “model”. In the previous code, we have created an instance “MySkynet” which is a custom neural network class. It is a feedforward neural network built using PyTorch’s “nn.Module” class. “nn.Module” serves as the foundation for constructing neural networks. Any custom model or layer is created by subclassing nn.Module, enabling users to build complex neural networks while inheriting essential features like parameter registration and forward pass definition (Paszke et al., 2019). It has 4 layer, input layer (self.layer\_0), two hidden layers (self.layer\_1 and self.layer\_2) and output layer (self.layer\_3)

The next challenging task was

A screenshot of a computer code

Description automatically generated

So, I was not able to handle this error because I did not what was wrong. When I clone the workshop\_1, it did not clone the data folder, du to which I was getting this error. I asked my group member but was not able to figure out. Then Alisha and me started working on this together and I clone the file again and match the output. This time, I did not change “%cd” to “!cd” to this, I just made sure there was no indentation and this time the folder data got clone.

After that when it came to get the batch\_szie, I played with values. At first, I got the warning for using 64 as batch\_szie but then I decrease it to 16. I also increased the number of Epochs to get the train accuracy to 0.9877. I changed the learning parameter to 0.01 to train the model better. With each change I noted how the number on top of image have changed.

A collage of different dogs

Description automatically generated

**Conclusion:**

The workshop was both enjoyable and overwhelming as I navigated the challenges of determining the optimal values for the missing parameters. However, once I began to see results, I thoroughly appreciated the opportunity to experiment with the learning rate and the number of epochs. This experience deepened my understanding of how neural networks learn through trial and error, akin to the process of teaching young children. Just as we expose children to various concepts and reinforce their learning through repetition, neural networks similarly refine their performance through iterative adjustments.

**References**

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems, 32*, 8026-8037.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *In Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)* (pp. 249-256).

College of Saint Benedict and Saint John’s University Digital Commons. (2021, June 3). *Geometry in the age of artificial intelligence and big data* [Video]. YouTube. https://www.youtube.com/watch?v=QBWcWv3hXxo