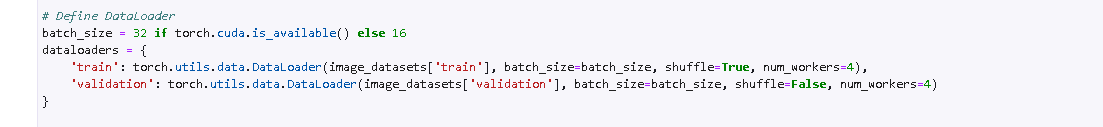
Project Muffin or Chihuahua

This report is about a simple yet informational AI that differentiates between a chihuahua and a muffin and presents a percentage for the accuracy for the particular image. For this report, it will need me to detail all the steps, decisions, and theoretical concepts behind the development, training, and the evaluation of this NN model and be able to perform this classification tack effectively. pytorch is a deep learning framework that provides tools and modules for building and training neural networks. torchvision is a package used for computer vision tasks. Matplotlib is mainly used for data visualization. PIL is a library for opening, manipulating, and saving many different images file formats, lastly glob, it is used to help in retrieving files/pathnames matching a specified pattern.

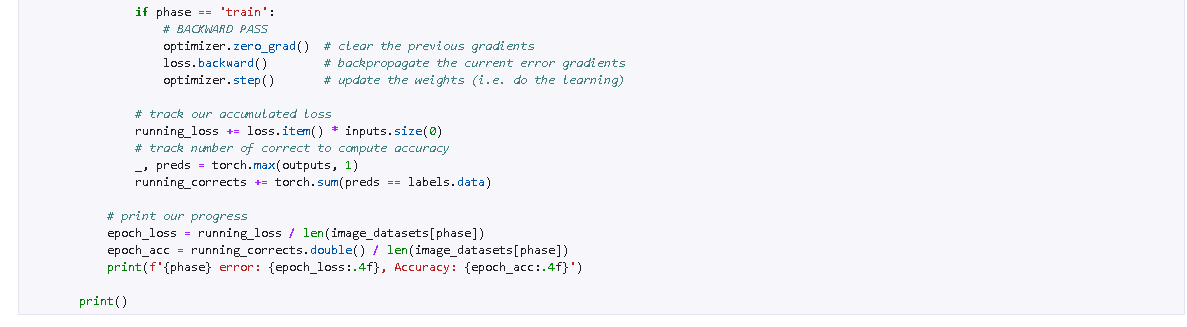
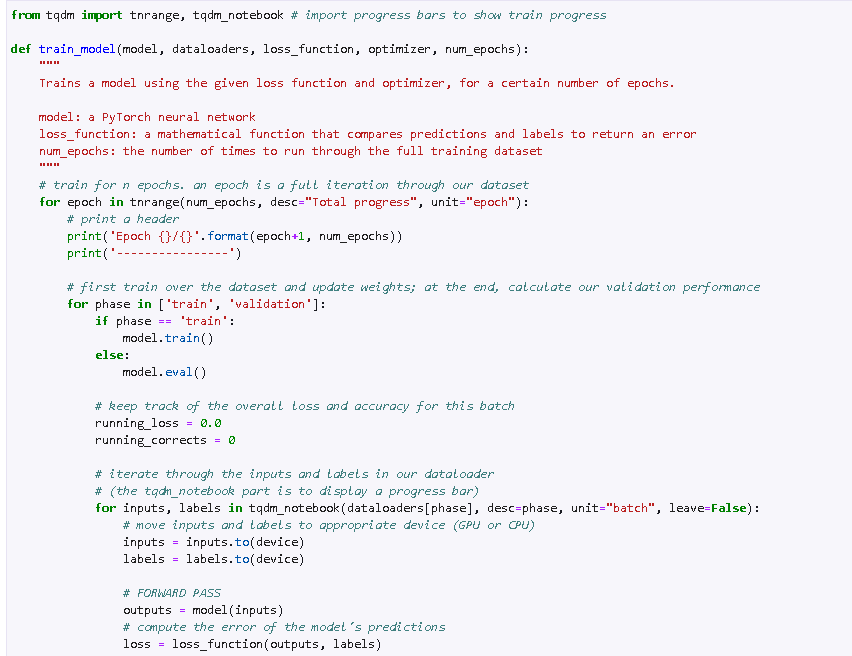
Pytorch allows a good support for GPU acceleration. It also allows us to define and train neural network models efficiently. torchvision simplifies the process of loading and preprocessing image data for model training and have a good amount of pre-trained models that can be used for transfer learning, it could be easier to perform as it is a simple model. Matplotlib is important for monitoring model training progress through plots of loss and accuracy metrics and allowing us to perform debugging potential issues. PIL carries the handling of image data, live loading images from file paths, resizing, and converting them to proper forms. Glob is used to dynamically retrieve image file paths, which is convenient when dealing with large datasets organized in directories based on class labels.

As for the data, it comes from a dataset specifically created for the project of classifying images into categories: chihuahua or muffin. In this case, there is a folder that contains all the files required to test and train of them. for the structure, it is under “train” and “validation”. with subfolders representing the two classes, “chihuahua” and “muffin” and the pictures can be further characterized with their own number. The First transformation is resizing as it is required for all the images to be feed to the NN model. Third is normalization, this helps in stabilizing the training process by bringing input data to a common scale and improving convergence during optimization. random rotation introduces augmentation that helps in increasing the diversity of training samples, preventing overfitting and improving the model’s ability to learn invariant features.





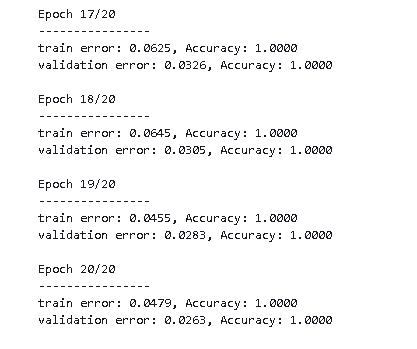
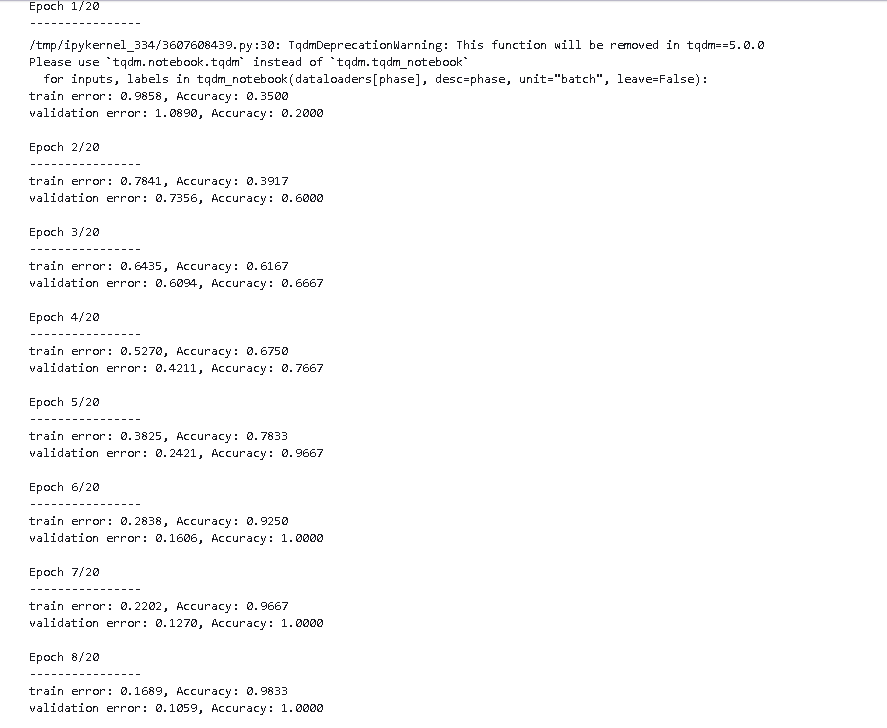
I have decided to go through the resnet model for the image classification task due to its proven effectiveness from previous assignments, resnet introduces skip connections, allowing the training of very deep networks without encountering significant degradation in performance. residual blocks in resnet allow for the reusability of features learned at different layers and at the end promotes efficient extraction and enables the network to focus on learning information. This model has convolutional layers, residual blocks, pooling layers, fully connected layer, and output layer.



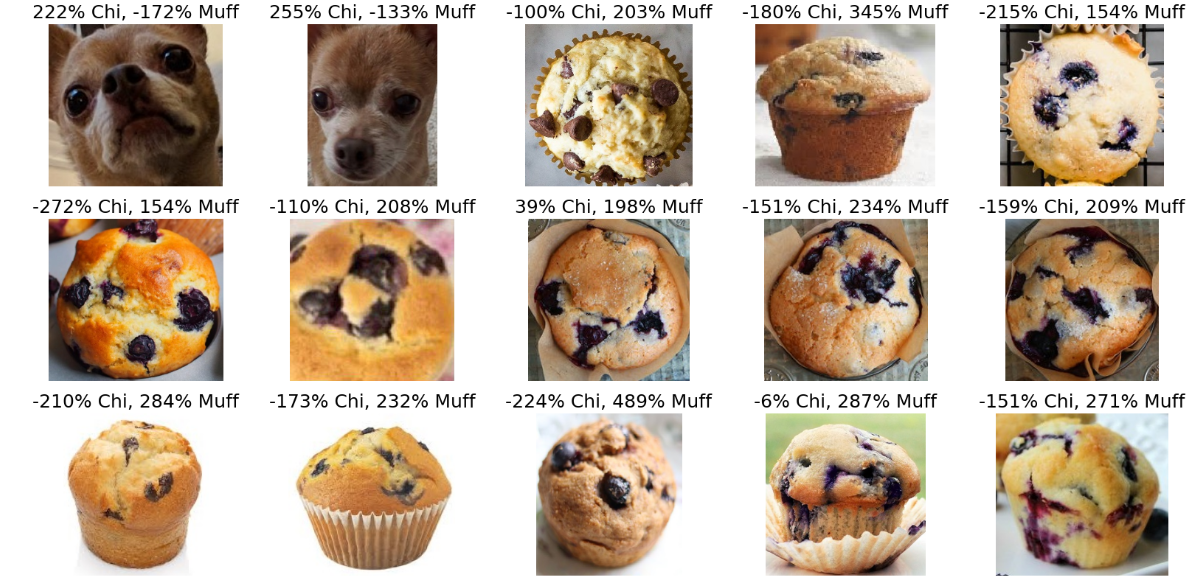
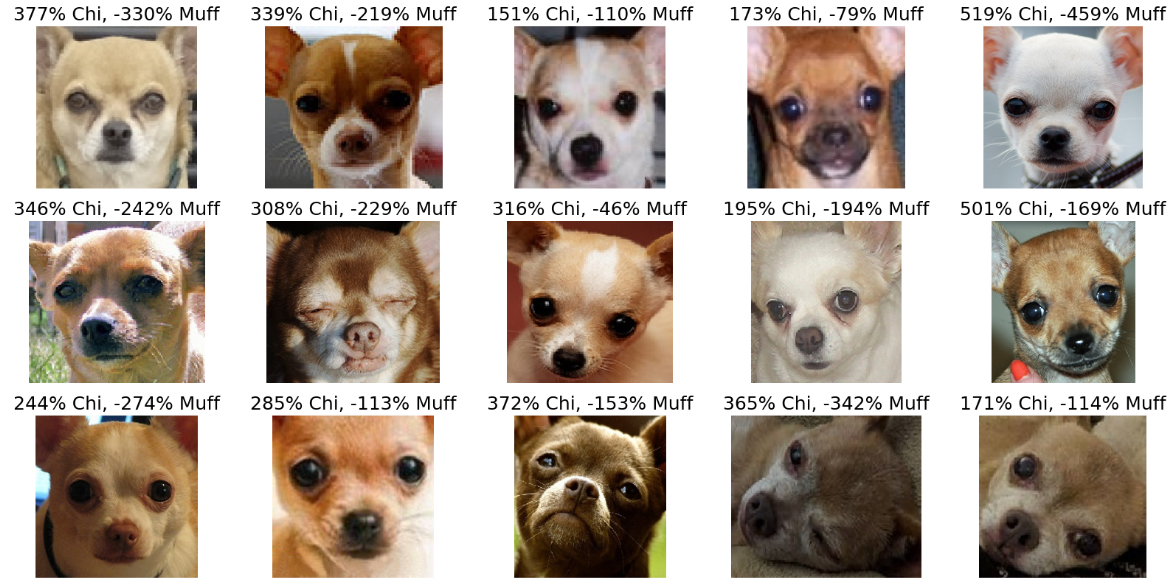
For the loss function in resnet model, I used Cross-entropy loss function, it is widely used in classification tasks, including image classification, where the goal is to minimize the difference between predicted probabilities and actual class labels. Because of this loss function, the model learns to make more accurate predictions by adjusting its weights and biases to minimize the loss. For the optimizer, I went for the Adam optimizer, it adjusts the learning rate dynamically based on the gradient magnitudes of parameters, allowing faster convergence and improved training stability. For the learning rate, I set it to 0.001, which is a good starting point for deep learning tasks. The role of optimizer is quite important by updating the model’s parameters based on the gradients computed during the backend. epochs refer to a one complete cycle through the entire training dataset during the training, and it usually depends on various factors like the dataset size. For this case, I decided to go for 20 epochs, as it is important for me to monitor the loss and accuracy during training to avoid overfitting. 20 epochs is a good balance between giving the model sufficient training time and preventing it from memorizing the training data too much.



There are multiple key metrics for this model, accuracy is the proportion of correctly classified images out of the total number of images, it is the general number for the model’s performance. the precision is the amount of true positives predictions to the total amount of actual positives.



There were multiple changes to the model itself, it focusses on architectural changes, hyperparameter tuning, and additional data augmentation techniques to make it unique and faster with every change. For the architectural changes. I experimented with a different architecture such as using ResNet model (In this case was using a ResNet-18) to program and obtain a more complex features and improve accuracy and regarding the complexity of ResNet-I first played around the expanded version of ResNet and tweaked some changes but at the end decided to go for preprogrammed form of ResNet, and I also added dropout layers to prevent overfitting and improve generalization. I also played around with hyperparameters by fine-tuning them for the specific model such as the learning rate, batch size, and the optimizer parameters. At the end, I also played with data augmentation like random rotation to provide diverse examples for the model to learn from and enhancing its use of data to variations in the dataset. Most of this is from post-optimization where I first did the main basic training done by the professor and then worked on trying to make it better before this report, for the main value concerns were accuracy and precision, with a good amount of epochs (20 as the main but took around 17) to get to a 100% for the accuracy and precision, because of this, there was a significant improvement in correctly classifying images and reduction in false positive predictions in a low amount of epochs. Similar to precision, the recall also had gotten better, it shows the model’s improved ability to identify actual positive instances. With these results, it indicates the new effectiveness in classifying chihuahua and muffin images more accurately and reliably. These are also the final results for the project with the changes that occurred to it (but the main issue that had risen was the percentages for the pictures, even though it was 100% accurate for all of them, all of them had values greater or lower than 100% to identify them, it could be due to the tweaks in the project that potentially be fixed if there is a restriction of how high the percentages can go)



For the theoretical concepts, first up is the neural networks, they are computational models based on the structure and functioning of biological neural networks in the human brain. They consist of layers that have organized inner parts like nodes. And there are three main layers in a neural network are input layers, hidden layers, and output layers; input layers are where the data is fed into the model, output layers are the middle layers that process the data, and output layers are which that produces the final output or prediction. As for learning for a neural network, it is called training, where they create their internal parameters based on input-output pairs to minimize the error and loss. Secondly, is convolutional layer. It is designed for processing spatial data such as images. With learnable filters or kernels, they apply convolution operations to input data, filters like extracting features such as edges, textures, and patterns. The reason why CNNs is quite handy is from the leverage parameter sharing (similar features will be placed through different spatial locations) and spatial recognition of layers (early layers detect simple features and deeper layers combine it to make it complex), considering CNNs good for image recognition tasks. Thirdly, the activation function’s purpose is to create non-linearities into the neural networks, allows them to learn complex relationships and make non-linear transformation on the input data. There are common activation functions like ReLU (Rectified Linear Unit), sigmoid, tanh, and softmax. For example, ReLU replaces negative values with zero, introducing a non-linear element crucial for learning intricate patterns in data. Without them, the neural networks would only be able to model and train in a linear fashion, limiting the capacity to solve complex solutions. Lastly, Backpropagation. It relates to the iterative process of updating the model’s parameters by “propagating” the error backwards through the network because whenever training, when there is an incorrect result, it is important for the model to recognize and minimize the error, it also involves with other model parameters like gradient descent (I think) to reduce the loss and at the end, allows the neural networks to learn from data and improve their predictions over time.

In conclusion, this project was aimed to understand how a typical model is ran and try to find ways to make it better, by the classification between chihuahua and muffins using a deep learning approach, like the CNNs. The performance was fairly good with a 100% accuracy after roughly 20 epochs after the alterations which wasn’t the case for the starting model. And with the end goal, it had a good balance of correctly identifying chihuahuas and muffins will minimizing false predictions. There can be multiple improvements like architecture enhancement, hyperparameter tuning, data augmentation, ensemble methods and transfer learning. I could go for more complex networks or utilizing pre-trained models like ResNet or VGG and this also relates to the transfer learning to potentially boost performance, especially with limited training data. Fine-tuning the hyperparameters like learning rate, batch size, and optimizer parameters which could be beneficial for the overall model performance. Increasing the diversity of advanced data augmentation would help for the model generalization to handle variations in input images. ensemble methods could also be a good option to boost the performance from reducing variance and bias. There is always a room for improvement one way or the other, some from simple method and some from changing the whole model to fit it. It is always nice to see the cute face of a chihuahua confused with a tasty delight.

