Importing The Needed Libraries

We imported the clip library for the model, the torch library for the device(cpu/gpu) related functions. An image processing library, pandas, numpy and sklearn for training and testing the dataset.

Firstly, we tested the image processing on an image. We simply loaded and displayed an image… Meow.

Secondly, we initialized the device, which is related to the hardware. So, we are selecting the device that will process and run the model.

Thirdly, we are reading the csv file and saving it.

Then we split the dataset into training and testing sets.

Using The CLIP Model

Firstly, we create a function that will return the embeddings for a given image. We transform the image, which is an image preprocessing step that ensures the image conforms to the model's requirements. The unsqueeze method changes the image format/dimensions to a batch format. The to(device) function moves the model to the specified device (cpu or gpu) to run the model.

Then we tested the the model on one image, by returning its embeddings. It worked!

Now that it worked, we got the embeddings for the entire training set and then the testing set.

Predicting Images With Fisher's Algorithm

Now that we have the embeddings, it is time to predict new unseen images using the model, with fisher's algorithm.

First, we filter the embeddings so that we have all dog embeddings and cat embeddings separately.

We calculate the mean for each, then the variance for each.

Then we calculate the weight vector that will classify the classes to either one side of the decision boundary or the other.

We predict the label of the test data using the calculated weight vector.

Now we will test the model's accuracy using classification accuracy metrics like accuracy score, precision, recall, f1-score and the confusion matrix.

We see that the accuracy results are very high. (around 0.99)

The heatmap representing the confusion matrix shows that the model got the majority of the data correct.

<u>Bonus</u>

We tried a very large C value and a very small C value, but after running the code and calculating the weights for both, we found that the accuracy was still very very high and did not change at all.