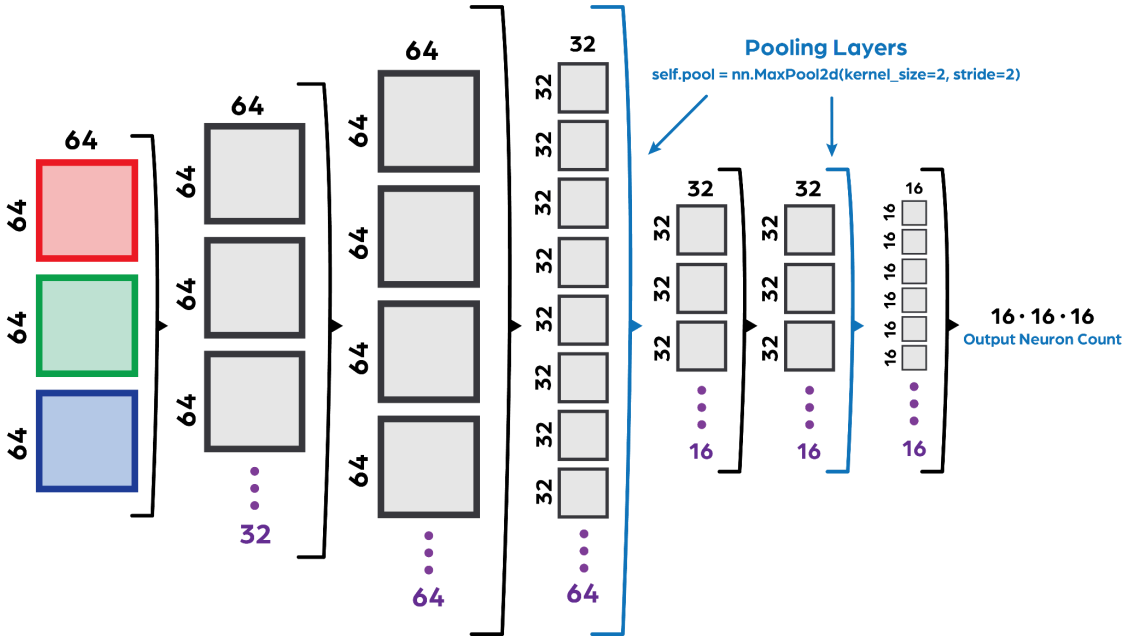
Model Report

This report outlines the architecture of our model and the changes we made to it in order to improve its accuracy. For more information on how our datasets were created and on the game itself, please refer to the README on Github.

Base Model Architecture:

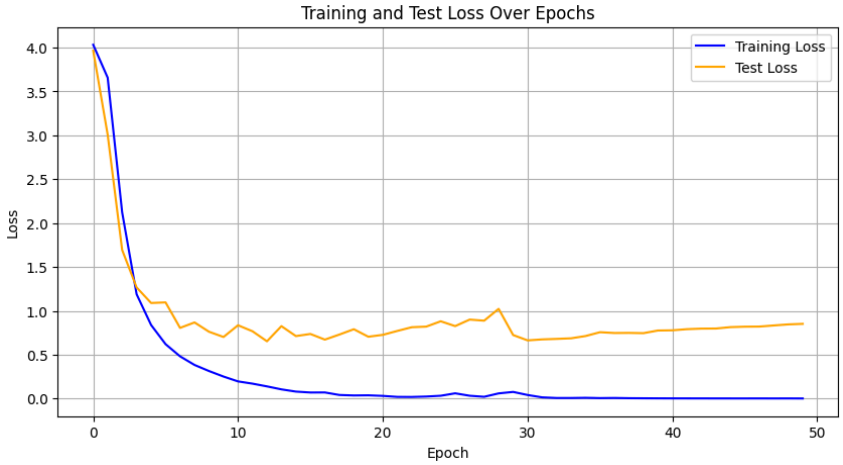


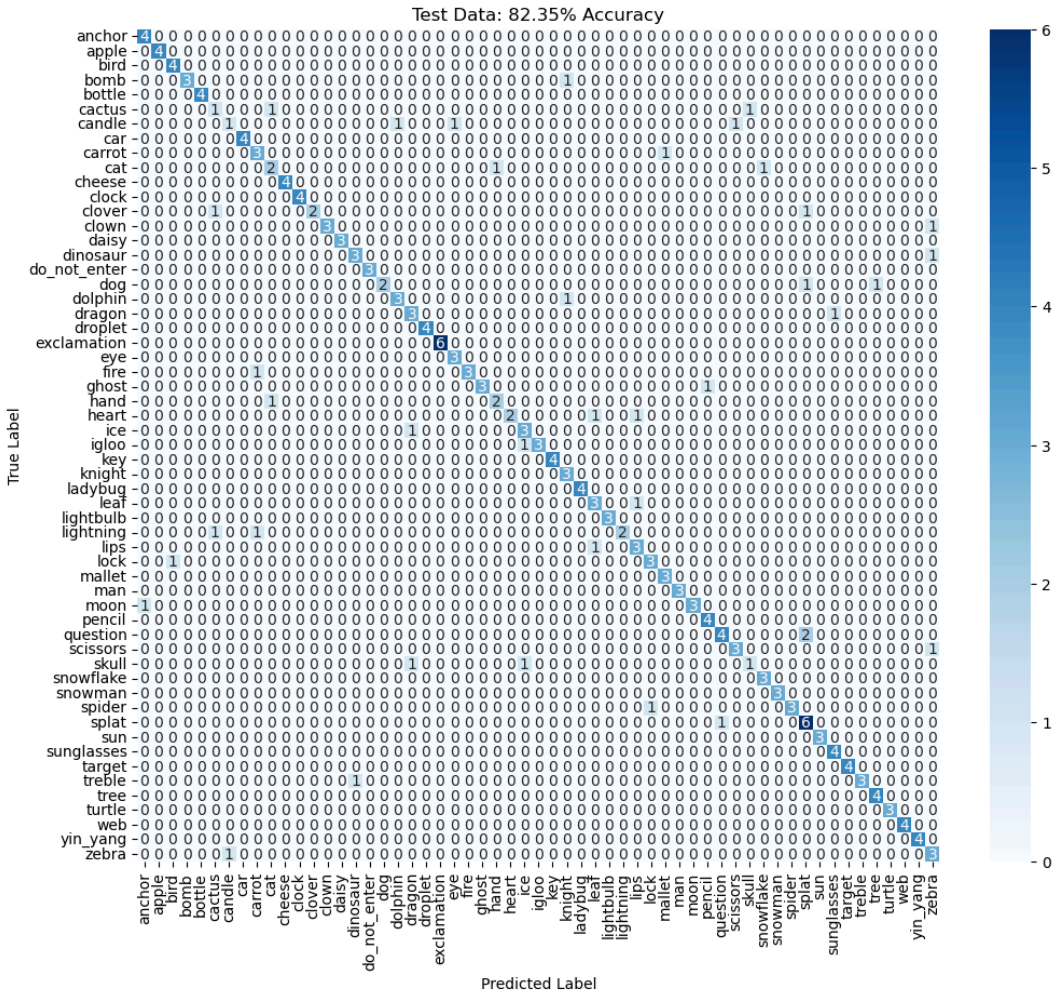
Our model architecture is inspired by the CNN Explainer by Georgia Tech and Oregon State. It takes in a 64x64 RGB image, then passes it through a series of convolutional and activation (in this case RELU) layers, as well as several pooling layers to shrink the inputs to the following layer. Each of our convolutional filters are 3x3 with a padding and stride of 1. Our final convolutional output is 16x16, which is then passed through a fully connected layer, resulting in the final output that categorizes the image.

In the iterations below, all input images are 64x64 unless otherwise specified.

Model Iterations:

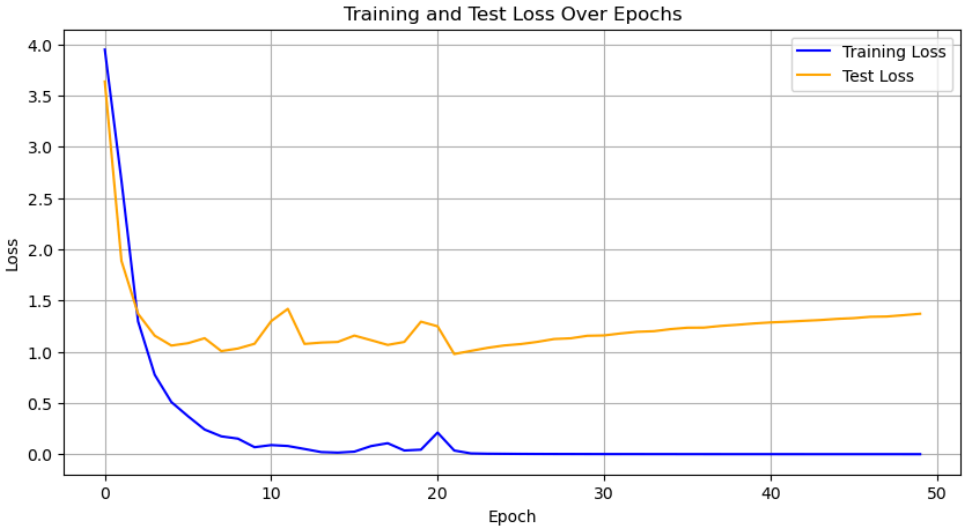
V1

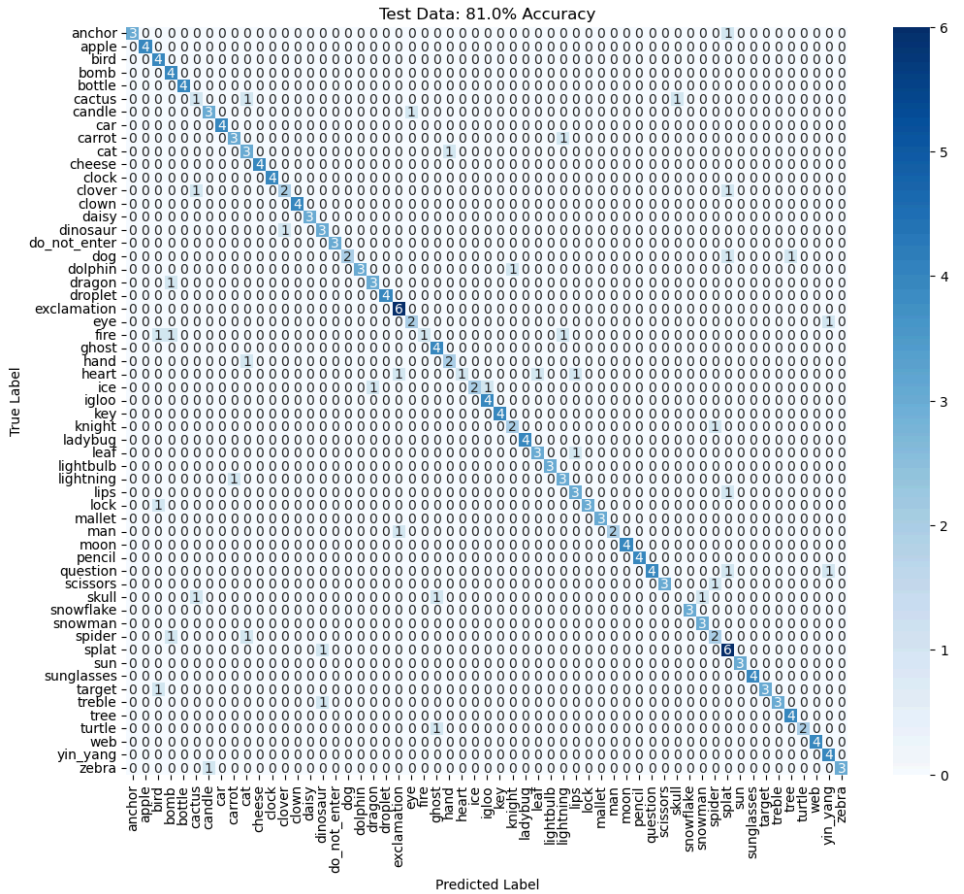




With the original iteration of our model, we were seeing some overfitting, evident by how the test loss curve diverges from the training loss curve.

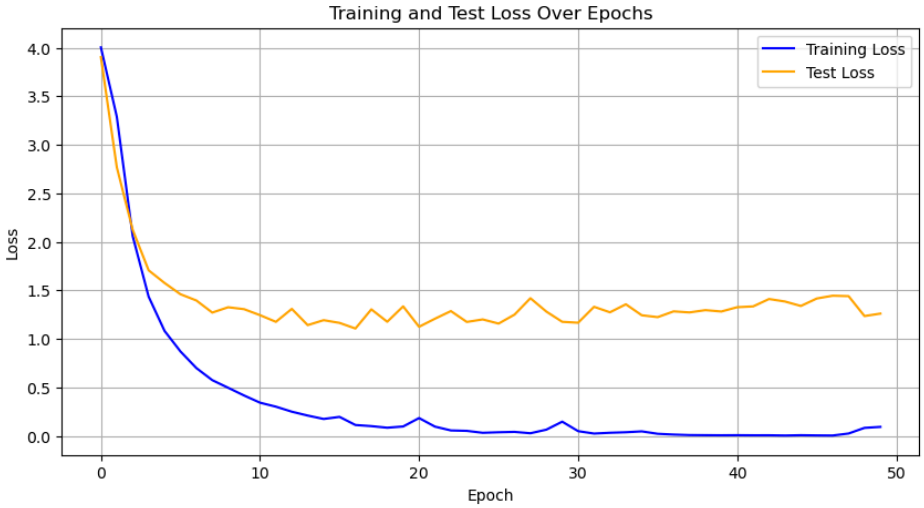
V2

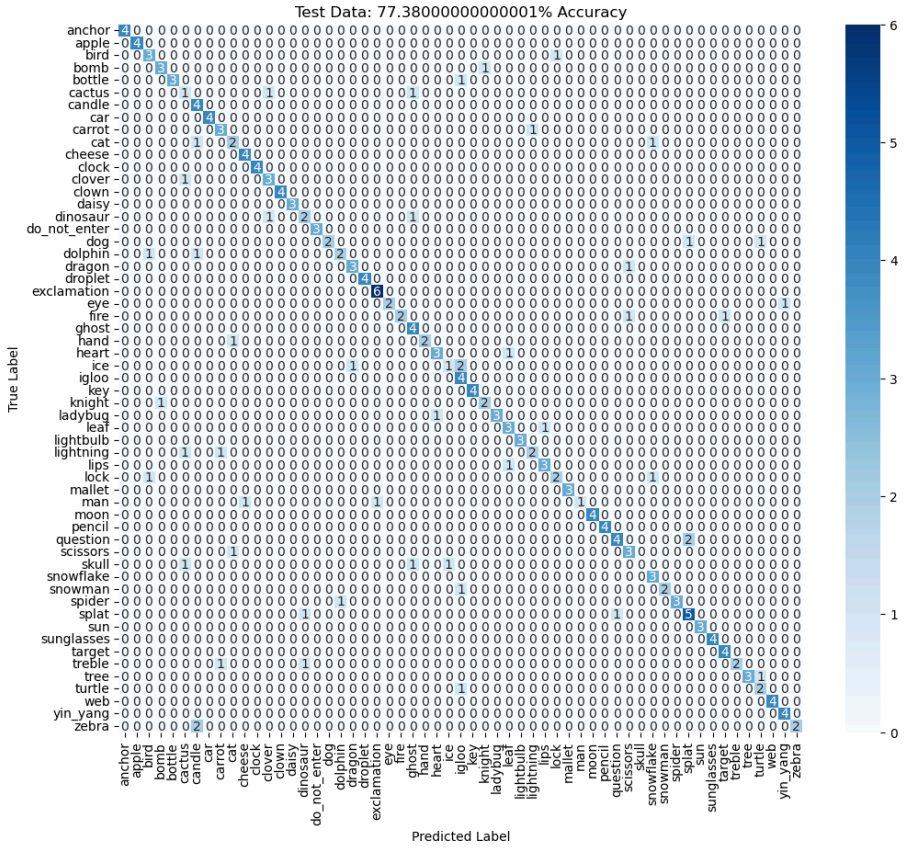




We are still seeing overfitting with the larger image sizes. This makes sense, as a larger image size would give us more data to overfit to. As increasing the image size didn't have a noticeable effect on our accuracy (it actually slightly decreased, with 81.36% for validation data and 81% for test data), and did increase training time, we decided to keep images at 64x64.

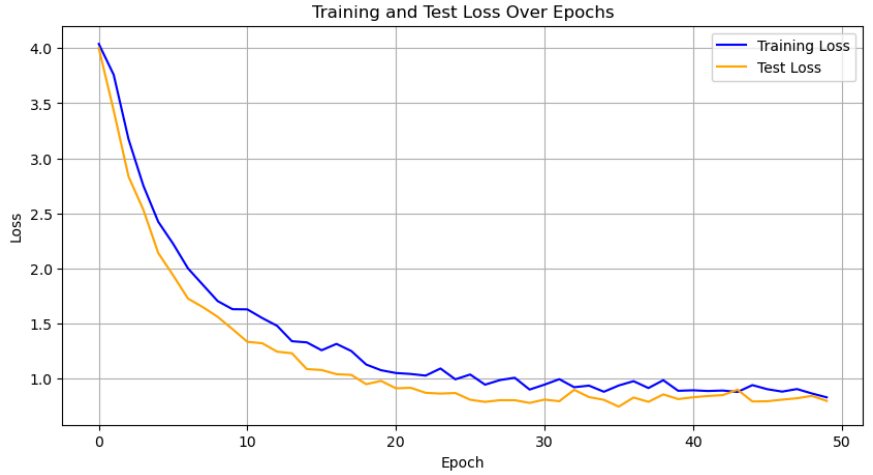
V3:

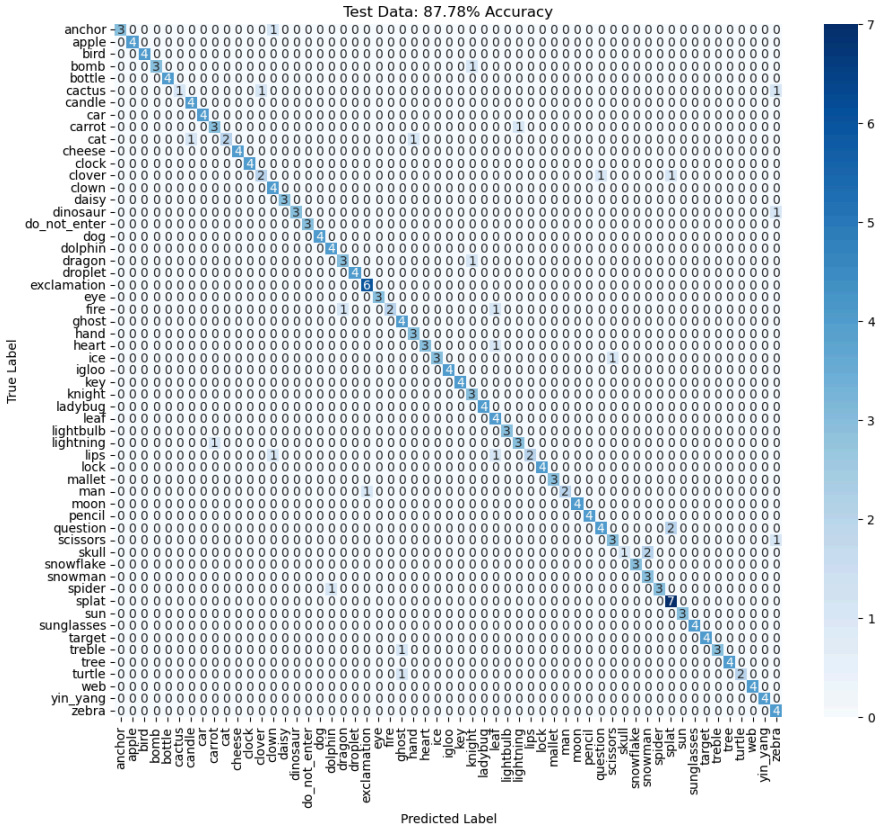




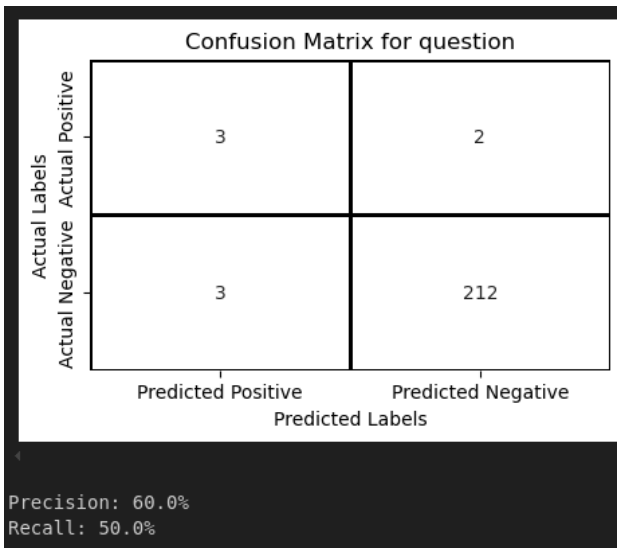
In our third model iteration, we added dropout to try and decrease overfitting. Dropout is a regularization technique, easily implementable with PyTorch, that temporarily removes random neurons, or “drops” them during the training process. This prevents the network from overfitting because it cannot become overly reliant on specific connections. We added dropout with p=0.1 (p is the probability of a neuron being dropped out) after each activation layer. This did seem to help slightly, but we still have a low accuracy, with an accuracy of 84.09% for the validation data and 77.38% for the test data.

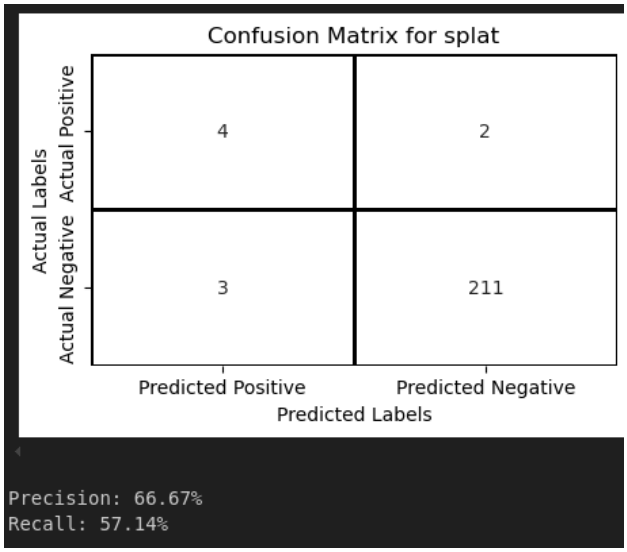
V4:





In our fourth and final iteration of the model, we added a larger amount of dropout (p=0.3) to the model after the fully-connected layer. This is because fully connected layers have a greater number of parameters, and are therefore more prone to overfitting. This seems to solve our overfitting issue, as the two loss curves in our training graph closely follow each other. Our overall accuracy has also improved, with an accuracy of 85.91% with the validation data and 87.78% for the test data.





For the final version of our model, the question icon and the splat icon were the two icons most prone to error. The confusion matrices were created from the validation data, which was less accurate than the test data. The question and splat symbols commonly get mixed up because instead of being interpreted as whole symbols, they are broken into separate components (because there are multiple closed contours in these symbols). These include green circles that are likely getting mistaken for each other.

Connecting to Learning Goals:

Overfitting was the main issue our model had, which we managed to solve with dropout. When changing the size of our input image in Version 2, we also reinforced our learning about how convolutional layers and pooling affect the size of the output image. Both of these steps helped us understand the affects changes in model architecture can have on the success of a model.

We believe that with more data, our model would have performed better. When we created our train/test/val split, we chose an 80%/10%/10% split. The size of this training set worked well, as we are able to successfully determine most icons on a Spot-It card. However, the size of our test and validation datasets are limited - we would be better able to determine the accuracy of the model with larger datasets. However, since we have mitigated over-fitting, the model's success on the test and validation datasets is a good indicator of model success.

Video Link:

https://youtu.be/_Oaybdc2RHk

Github Link:

https://github.com/amandalchang/spot_it_cnn