Michael Ryvin

First Model: Neural Network

Training and Testing Data:

For this model, the training data consisted of pixels from the image provided. In order to minimize the number of pixels, the image was first made into a lower resolution by a factor of 6. After this, the x values were made up of rgb values of the pixels in the image. Pixels were only included if they were in the bottom ⅔ of the image, other than the white square. This was primarily done for ease of coding as well as lowering the amount of data that would need to be processed by the model. The y values were gathered with a similar manner. However, the y values corresponded to the rgb values of the pixels directly above the x values.

The testing data would start off with the rows immediately above and below the white square. However, as the model took effect, the testing data came from the new rows that were created.

Structure of the Data:

The x values are structured as an array of rgb values. There is a single array, and each value in the array is a three value array. These three values correspond to red, green, and blue. The y values were structured in a similar manner, except as three different arrays. There was an array with every red value, an array with every green value, and an array with every blue value. The x array was used to make models for each of the corresponding y arrays.

Data Preprocessing:

The data was viewed as numerical. By default, rgb values are put in a range of 0 to 255. However, this would make it harder to run a neural network. As such, I divided the values by 255 so that each color value was on a range between 0 and 1. in order to display the image this would be altered later, but for the purposes of the model the numbers had to be kept between 0 and 1.

Model, Models, Modeling:

The model used is a neural network model, used to create a numerical regression. However, rather than using a single neural network, the model uses a combination of three neural networks. A separate neural network is conducted for each rgb value. Each model is identical, with it using the numerical rgb values to predict either r, g, or b values of the pixels directly above or below.

Bootstrapping:

My model included bootstrapping, as the pixels not at the top or bottom of the white square ended up having their colors based on newly filled pixels. The issue with using bootstrapping was that there was no way to prevent the regression models from eventually reaching infinitely high or low values. As a result, in order to combat this, I decided to perform the model both going up and down the white square, so as to reduce the inaccuracy of the model.

Measuring Loss:

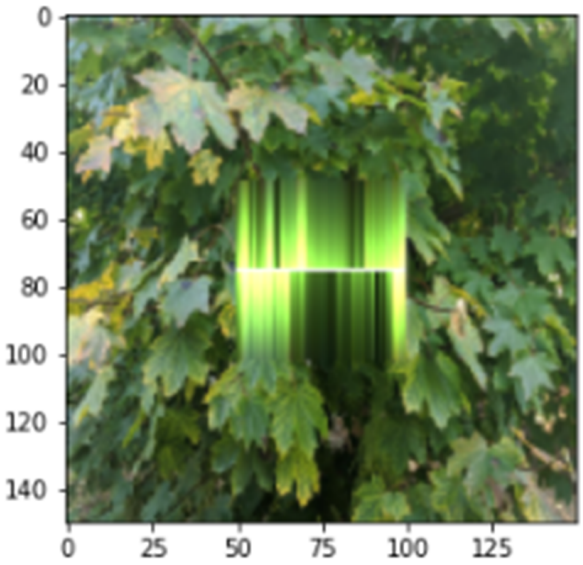
The error of the model is computed as the neural network is conducted. The error is the difference between the tanh used in the neural network and the value of the specific y value. As the weights change over time, ideally the error decreases. In theory, a lower error should correspond to a more accurate model in terms of colors. However, given the regression model used, it did not necessarily produce a much better result.

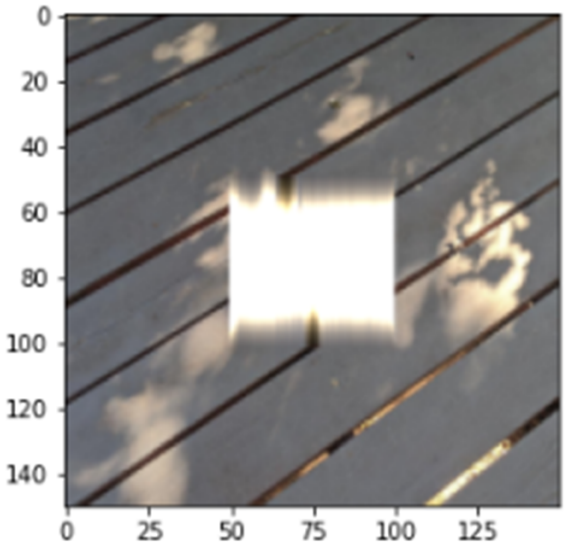
Training:

I do not believe that overfitting is much of an issue in the model. Based on how the model was made and what training data was used, the model should not necessarily overfit. However, it is still not a great model as the testing data does not fully capture the surroundings of the square.

Assessing your Models:

The way to determine how well the model performed is simply by looking at it visually from a human perspective. The more that the filled-in square blends into the background, the better the model is. My model produced the following results for wood and leaves.





As can be seen, by the results, the model is only able to extrapolate from the values directly above and below the square, producing inaccurate results. Likewise, the model has a tendency to move the color values to be very high or very low, thus causing the colors toward the middle of the square to be even more inaccurate. This is very visible in the wood image, as the model causes all the color values to be very high and produce a white color.

Second Model: Decision Tree

Training and Testing Data:

For this model, the training data consisted of pixels from the image provided. In order to minimize the number of pixels, the image was first made into a lower resolution by a factor of 10. After this, the x values were made up of rgb values of the pixels in the image. Pixels were only included if they were in the bottom ⅔ of the image, other than the white square. This was primarily done for ease of coding as well as lowering the amount of data that would need to be processed by the model. The y values were gathered in a similar manner. However, the y values corresponded to the rgb values of the pixels directly above the x values. The testing data would start off with the rows immediately above the white square. However, as the model took effect, the testing data came from the new rows that were created.

Structure of the Data:

The x values are structured as an array of rgb values. There is a single array, and each value in the array is a three value array. These three values correspond to red, green, and blue. The y values were structured in a similar manner, except as three different arrays. There was an array with every red value, an array with every green value, and an array with every blue value. The x array was used to make models for each of the corresponding y arrays.

Data Preprocessing:

The data was viewed as numerical. By default, rgb values are put in a range of 0 to 255. However, this would make it harder to run a neural network. As such, I divided the values by 255 so that each color value was on a range between 0 and 1. in order to display the image this would be altered later, but for the purposes of the model the numbers had to be kept between 0 and 1.

Model, Models, Modeling:

The model used is a numerical-based decision tree. However, rather than using a single decision tree, the model uses a combination of three decision trees. A separate decision tree is

conducted for each rgb value. Each model is identical, with it using the numerical rgb values to predict either r, g, or b values of the pixels directly above or below. In each branch of the decision tree, it is first determined whether r, g, or b would be used as the best separator for the tree. Once this is determined, the model decides where is the best place to separate the tree, and does so based on that r, g, or b value. The average r, g, or b value is then found in that branch, depending on whether I am looking for the r, g, or b value in that specific tree.

Bootstrapping:

My model included bootstrapping, as the pixels not at the top of the white square ended up having their colors based on newly filled pixels. The issue with using bootstrapping was that there was that the model ended up simply creating vertical lines of essentially the same colors. As a result, the model did not perform perfectly, but it did fill in the vaguely correct colors in the square.

Measuring Loss:

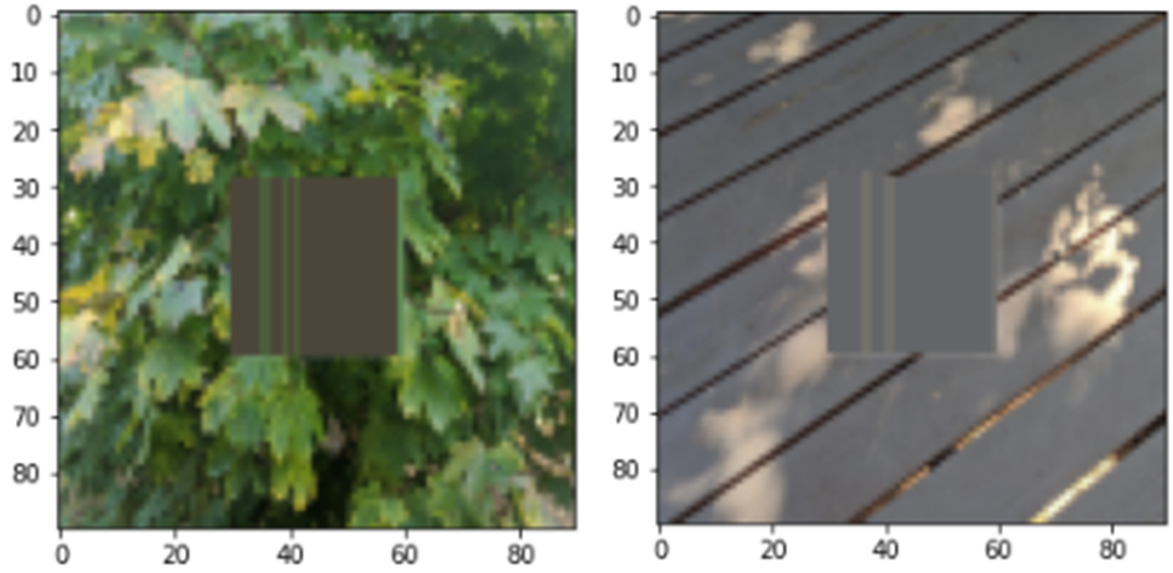
The error of the model is computed by determining how inaccurate the prediction is for a given pixel compared to the actual colors that a pixel would have. This is done for r, g, and b values separately as three different models are made. As a result, there are different errors for the r value, the g value, and the b value.

Training:

I do not believe that overfitting is much of an issue in the model. Based on how the model was made and what training data was used, the model should not necessarily overfit. However, it is still not a great model as the testing data does not fully capture the surroundings of the square.

Assessing your Models:

The way to determine how well the model performed is simply by looking at it visually from a human perspective. The more that the filled-in square blends into the background, the better the model is. My model produced the following results for wood and leaves.



As can be seen, by the results, the model is only able to extrapolate from the values directly above the square, producing inaccurate results. Likewise, the model for some reason produced far more inaccurate results on the leaves than on the wood in terms of the color. This may be due to the input space affecting the model being produced, or perhaps the tree should have gone deeper than it did. Perhaps if a more powerful computer were used, then a deeper tree would be possible, thus producing better results.

Bonus 3: Birthday Dog