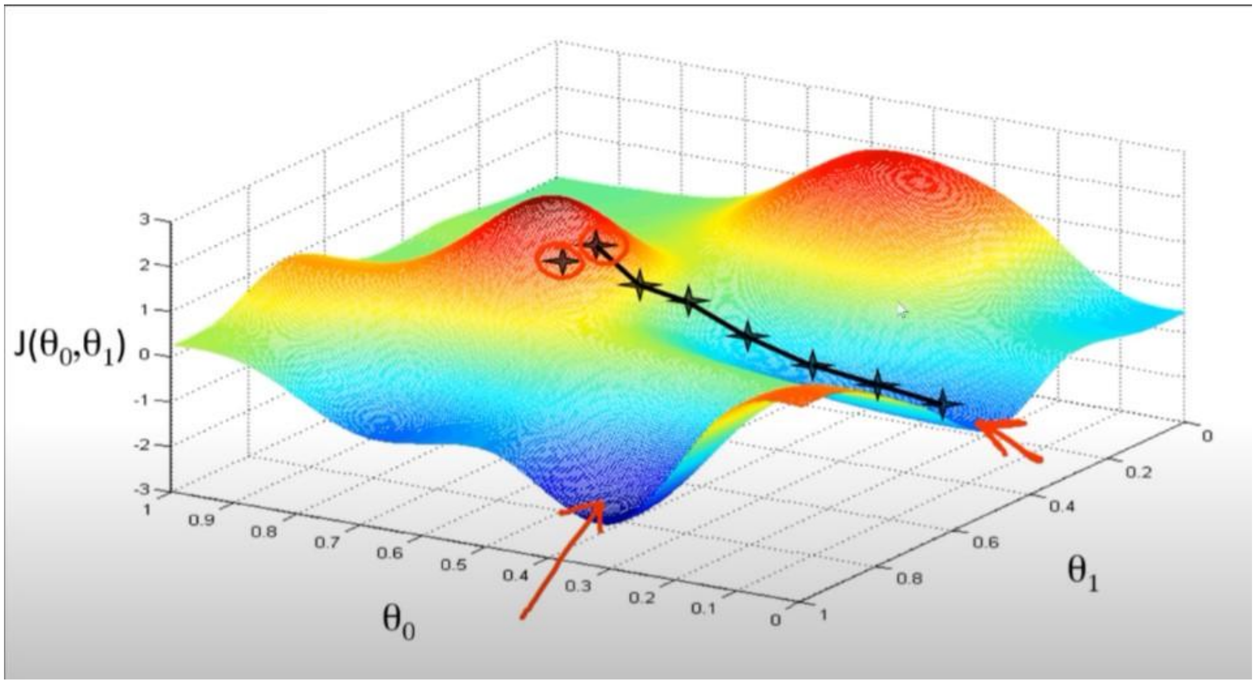
**Name: Jason Zhou**
**Mentor: Dr. Dongjin Song**
**Status Report #: 4**
**Time Spent on Research This Week: 2.5**
**Cumulative Time Spent on Research: 16**
**Miles Traveled to/from Mentor This Week:** 0
**Cumulative Miles Traveled to/from Mentor:** 0
==========================================================================

**Wednesday, September 29th, 2021: (1 hour)**

Just like the last week, I continued to watch the TensorFlow video course. This week, I dove into dense and convolutional neural networks. In general, all neural networks have activation and loss functions as well as gradient descents and backpropagation. The activation function is coded into every layer of a neural network and transforms a numeric value into a form that is easier to use. For example, if I say that 98 out of 151 students like cold weather, it would be easier to say that 64.9% of students like cold weather. Essentially we took large numbers like 98 and 151 and converted them into percentages (which are values between 0 and 1) to make them easier to understand. In the case of a computer, these transformed values make the data more useful.

Additionally, all neural networks have a loss function, which calculates a value based on how far away the actual output was from the expected output (accuracy). The higher the value, the more inaccurate the actual output was. As such, one should try to design a model that calculates a low value for the loss function (this would mean the accuracy of the model was high).



(This is a graph of a loss function. The function takes in two variables and outputs one variable. As such, the image shows a graph in the third dimension. The red areas represent places where the loss function is high, while the dark blue areas display places that have a low loss function)

As discussed in a previous status report, neural networks have node values, weights, and biases. At first, these values are completely randomized and then are fine-tuned as the model takes in more data. On this day, I learned that the process of fine-tuning values is actually done by pairing the loss function with a technique called gradient descent. If one looks at the above image as an example of a loss function, the gradient function simply calculates which direction the input values must be moved in order to reach the global minimum (blue area) of a loss function. In other words, it tells the model how to move from the red zone (or any zone that is not the global minimum) to the dark blue zone (global minimum). Once the gradient descent has done this, a technique called backpropagation is used to move backward through the neural network and update the numerical values (this is the process of fine-tuning values).

**Thursday, September 30th, 2021:**

Unfortunately, I had something at school that I could not miss, so had to email my mentor to cancel our zoom meeting for this week.

**Friday, October 1st, 2021: (1 hour)**

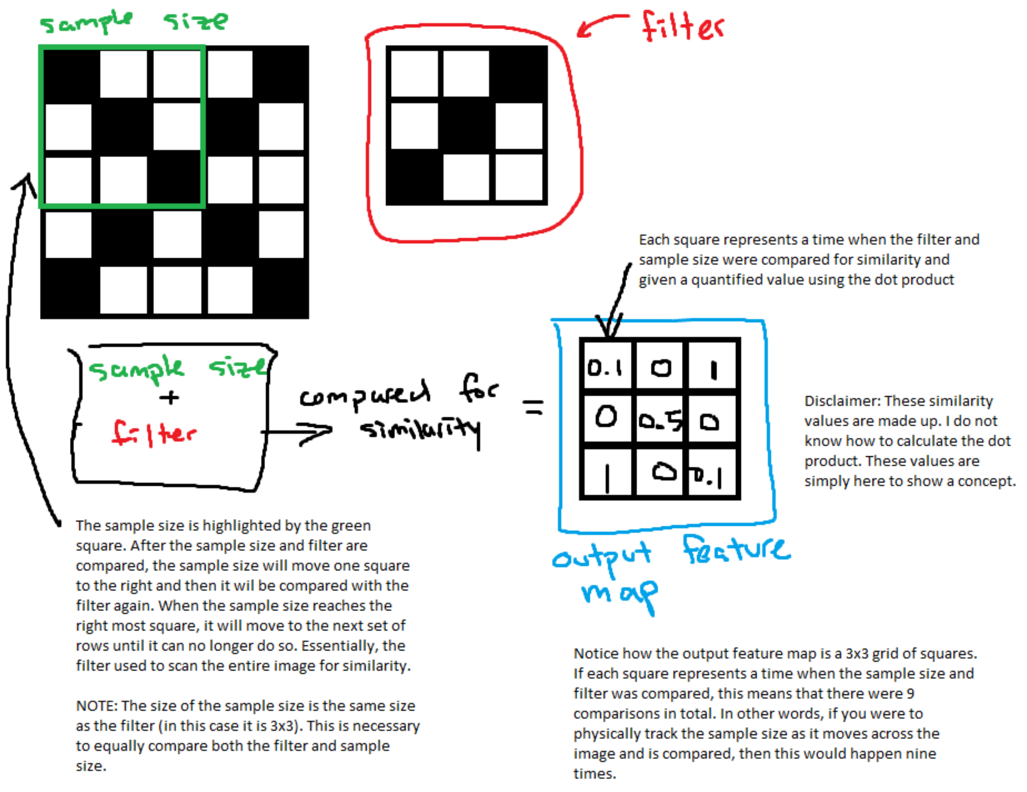
I spent the first 30 minutes reviewing concepts I had already learned (e.g linear regression and classification) and went over how to set them up with code.

Beyond review, I learned that convolutional networks are used for local learning, while dense neural networks are used for global learning. Essentially, a dense neural network (DNN) will memorize patterns and the positions they are in. For example, if I showed a DNN a picture of a dog, the model would memorize the patterns of the dog in that specific picture. However, if I were to flip or rotate the image, the DNN would not recognize the image as a dog.

In contrast, a convolutional neural network (CNN) focuses on understanding the underlying patterns behind an image. In the same dog example, even if the image were flipped, a CNN would still be able to identify the image as a dog because it has learned to recognize underlying patterns and not just memorize what something looks like. This is similar to students. A DNN would be like a student who simply memorizes math formulas but does not understand exactly how these formulas work. In contrast, a CNN would be like a student that knows how the formula works because he or she has learned the underlying principles behind it.

**Sunday, October 3rd, 2021: (0.5 hours)**

Although I only had time to study for 30 minutes, I spent this day learning more about convolutional neural networks (CNN). In a CNN, there are always “convolutional layers,” each of which has a number of certain sized filters. Filters are patterns that CNN’s look for in the data. For example, in an image, a filter could be a line, so the CNN would scan for any lines in the data (in this case, the data are images). When scanning the data, the model takes sample sizes of the data to compare with the filter. Then, a similarity value is calculated using the dot product (calculus). This quantified similarity is then placed in what is called an output feature map, which simply records the presence of certain patterns in a set of data.



Each square represents a time when the filter and sample size were compared for similarity and given a quantified value using the dot product

Disclaimer: These similarity values are made up. I do not know how to calculate the dot product. These values are simply here to show a concept.

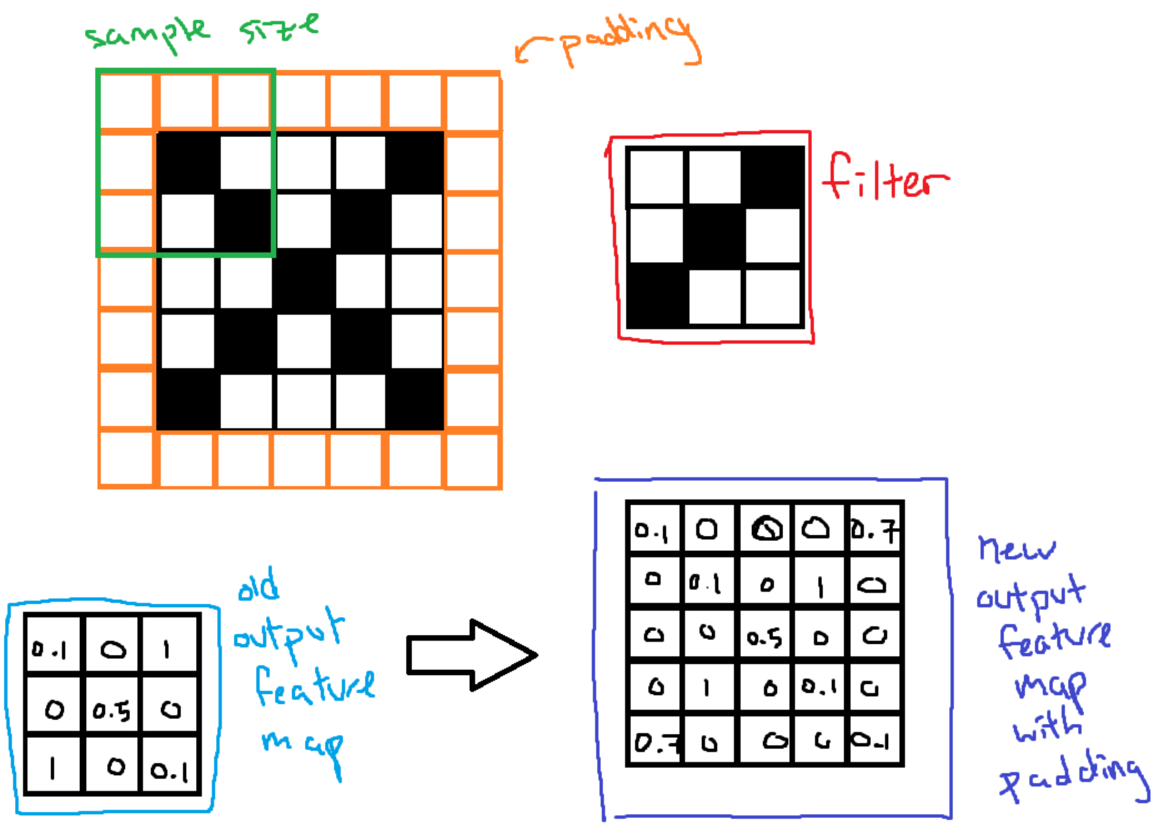
The sample size is highlighted by the green square. After the sample size and filter are compared, the sample size will move one square to the right and then it wil be compared with the filter again. When the sample size reaches the right most square, it will move to the next set of rows until it can no longer do so. Essentially, the filter used to scan the entire image for similarity.

NOTE: The size of the sample size is the same size as the filter (in this case it is 3x3). This is necessary to equally compare both the filter and sample size.

Notice how the output feature map is a 3x3 grid of squares. If each square represents a time when the sample size and filter was compared, this means that there were 9 comparisons in total. In other words, if you were to physically track the sample size as it moves across the image and is compared, then this would happen nine times.

(A visual representation of everything I was just said accompanied with explanations. Hopefully, this makes it easier to better understand what I am trying to present. For reference, the 5x5 square represents an image of an X, and the filter is scanning over the image for a diagonal line)

When scanning the data, there are certain techniques one can use called padding and pooling. Padding allows us to preserve the dimensionality of the data by adding extra sets of unimportant values to the data set. For example, in the above image, notice how the 5x5 image data was transformed into a 3x3 output feature map after being compared with a filter. Thus, when going from the image to the output feature, the data lost dimensionality (it became smaller) because it went from 5x5 to 3x3. If the original image data was padded, then the output feature map would retain the 5x5 dimensionality. As of right now, I am not sure why one would want to preserve the original dimensionality of the data. I will research this further in the coming weeks.



(A visual example of padding to supplement my explanation in the paragraph directly above. Because the padding has added extra space to the data, the filter has more space to compare for similarity. Thus, more space means more comparisons. More comparisons mean that the output feature map will increase, which is why it would now be 5x5 instead of the 3x3 dimensionality it had without padding. Additionally, just like the last image, keep in mind that all the values in both the old and new output feature maps are made up because I do not know how to calculate the values using the dot product yet.)

Besides padding, one can also use a technique called pooling on the data. Certain operations that fall under pooling include taking the minimum, maximum, or average of the data. For example, if one were taking the minimum to pool data, a sample size would be assigned within the output feature map (just like a sample size was assigned and moved around in the data when comparing for similarity.) Within the sample size, the minimum value would be recorded and placed inside a new output feature map. Then, a new sample size would be assigned. Similar to previous examples, each square in the new output feature map would correspond to a time when the minimum of the sample size was taken. This ultimately reduces the dimensionality (size) of the data while also extracting the important parts. For example, this would be similar to highlighting the passages of a text. Only the most important parts (or the pieces of data that one would like to see the most) are highlighted.

As of right now, this is all that I have learned. This week, I learned very complex concepts, which made it difficult to clearly illustrate what I had learned. To resolve this, I added images to supplement my explanations and hope that I was able to properly explain each concept. Overall, this was an exciting week, and I hope to learn even more in the future!

## References

freeCodeCamp.org. (2020, March 3). *TensorFlow 2.0 complete course - python neural networks for beginners tutorial* [Video]. *Youtube*. https://www.youtube.com/watch?v=tPYj3fFJGjk&t=6284s