Journal Report 13

12/16/19-12/20/19
Odin Woitek
Computer Systems Research Lab
Period 1, White

Daily Log

Monday December 16

Did more research on keras models always predicting the same class. Found that a common issue is forgetting to normalize data.

Tuesday December 17

I looked very carefully at my input and output data, thinking I forgot to normalize it. However, all values were between 0 and 1, and the y values were hot encoded. I looked closer at my softmax function, which I knew was appropriate for this problem with the single-choice categorical y values, and thought that perhaps I could address the issue by seeing the output data from that layer. As it turned out, there is little to no difference in the softmax output when different images are fed into the model.

Thursday December 19

I had some trouble here trying to figure out how to fix my problem. It could be that my data is bad, but there have been similar projects that use the IAM database with image patches that seem to work. I tried to make my network as simple as possible, which did not change much. When I added a batch normalization layer after the convolution layer in my shortened network, I noticed the softmax output was more evenly distributed, but it was still predicting roughly the same probabilities between images. Oddly, I found that if I put batch normalization after the activation function of a convolution layer and made the optimizer SGD with a clipvalue of 0.5, I would get balanced although consistent probabilities and would get my normal 27% accuracy on the test set. Using adam instead would yield terrible accuracy on the test set.

Monday January 6

Implemented the simplified network containing only one convolutional layer with the batch normalization and trained for 20 epochs, which evaluated at 31% accuracy on the test set, meaning it no longer only predicted the value to be 0 for each image.

Tuesday January 7

Implemented confusion matrix. Had an issue printing it, since the tensorflow gpu version I have doesn't have the Session() attribute, but I was able to get around this using tf.compat.v1.Session(). I then ran into another error while trying to print, which I got past by disabling eager execution.

Thursday January 9

Implemented confusion matrix for the model output. I ran into a few issues with the dimensions of the numpy arrays I was putting into the confusion matrix, but after using np.argmax() on each of them, I got the two arrays I intended to compare. Once I got this working, I had an issue where the confusion matrix wasn't printing cleanly, splitting each row onto two lines, which made it difficult to read. It turns out, this is because numpy limits the amount of characters per line to 75 by default, so I just changed the max linewidth, which fixed the issue. Now that I had what I needed, I added my previous layers back on to the model and after training for 20 epochs, got an accuracy of 91% on the test set.

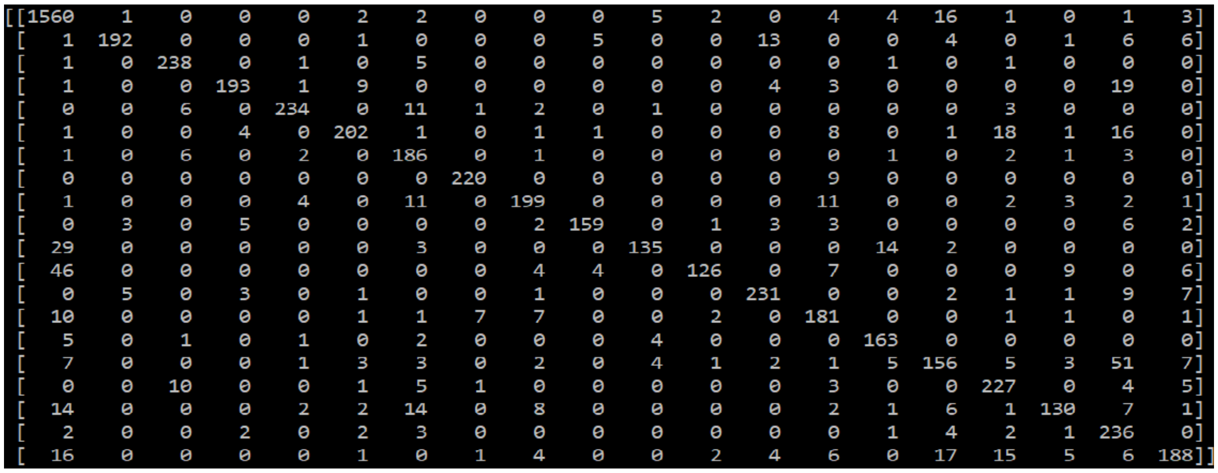
Timeline

Date	Goal	Met
December 2-6	Improve accuracy to 0.3	Yes, after implementing Batch Normalization, accuracy exceeded 0.3 with an extremely simple network and little training
December 9-13	Successfully overfit model on small dataset of 50 images	Yes, with Batch Normalization Layer, model no longer only predicted author 0
December 16-January 10	Train a model that can predict author identity with at least 0.7 accuracy on the test set	Yes, after implementing Batch Normalization to my previous model, I got an accuracy of 0.87 on the test set after 20 epochs.
January 13-17	Successfully save model after training on zoidberg	
January 20-24	Improve accuracy on the test set to 0.9	

Reflection

The week leading up to winter break, I was having a lot of trouble with my model, as its accuracy would never go beyond 27%. The winter goal I had set for the end of the week was to reach 70% accuracy, so being completely unable to improve from here was a huge issue that had stuck through each and every model I have created so far. I decided to create an extremely simple network with only one convolution layer so I could more easily find a solution to my problem. After playing around and researching the effects of my optimizer, activation functions, learning rate, and so on, I finally noticed that the network was predicting a value of zero regardless of what image was being fed into the network. In order to solve this, I tried putting in a Batch Normalization layer after the convolution layer, which led to an increase in accuracy, so the model was no longer only predicting zero. For the rest of the week, I worked on implementing a confusion matrix so I could have a better understanding of my model's predictions. Once I finally had it working, I saw that most predictions were still zero, but not all of them were, which was an improvement. After putting my other layers back on to the network, I was able to get my accuracy up to 87% on the test set.

The confusion matrix I had from the model with 87% accuracy:



```
[[1560    1    0    0    0    2    2    0    0    0    5    2    0    4    4   16    1    0    1    3]
 [   1  192    0    0    0    1    0    0    0    5    0    0   13    0    0    4    0    1    6    6]
 [   1    0  238    0    1    0    5    0    0    0    0    0    0    0    1    0    1    0    0    0]
 [   1    0    0  193    1    9    0    0    0    0    0    0    4    3    0    0    0    0   19    0]
 [   0    0    6    0  234    0   11    1    2    0    1    0    0    0    0    0    3    0    0    0]
 [   1    0    0    4    0  202    1    0    1    1    0    0    0    8    0    1   18    1   16    0]
 [   1    0    6    0    2    0  186    0    1    0    0    0    0    0    1    0    2    1    3    0]
 [   0    0    0    0    0    0    0  220    0    0    0    0    0    9    0    0    0    0    0    0]
 [   1    0    0    0    4    0   11    0  199    0    0    0    0   11    0    0    2    3    2    1]
 [   0    3    0    5    0    0    0    0    2  159    0    1    3    3    0    0    0    0    6    2]
 [  29    0    0    0    0    0    3    0    0    0  135    0    0    0   14    2    0    0    0    0]
 [  46    0    0    0    0    0    0    0    4    4    0  126    0    7    0    0    0    9    0    6]
 [   0    5    0    3    0    1    0    0    1    0    0    0  231    0    0    2    1    1    9    7]
 [  10    0    0    0    0    1    1    7    7    0    0    2    0  181    0    0    1    1    0    1]
 [   5    0    1    0    1    0    2    0    0    0    4    0    0    0  163    0    0    0    0    0]
 [   7    0    0    0    1    3    3    0    2    0    4    1    2    1    5  156    5    3   51    7]
 [   0    0   10    0    0    1    5    1    0    0    0    0    0    3    0    0  227    0    4    5]
 [  14    0    0    0    2    2   14    0    8    0    0    0    0    2    1    6    1  130    7    1]
 [   2    0    0    2    0    2    3    0    0    0    0    0    0    0    1    4    2    1  236    0]
 [  16    0    0    0    0    1    0    1    4    0    0    2    4    6    0   17   15    5    6  188]]
```