MACHINE LEARNING GLASSES RESULTS LOG

KIRK SWANSON

1. 11/26/2017

(1) Merged recent updates to GitHub
(2) Downloaded images from a while ago. This first one is batch size 80, learning rate 0.001, beta 0.01, dropout 0.5, including the recent change that each batch is chosen from a newly randomly chuffled metadata:

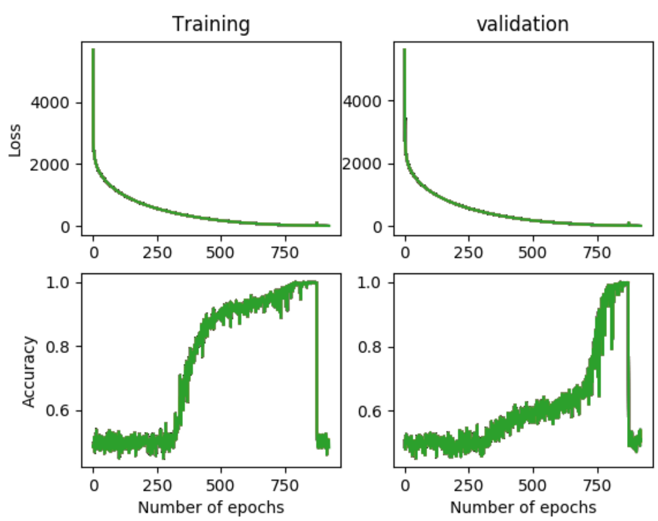

FIGURE 1. Batch Size = 80, Learning Rate = 1e-3, Beta = 0.01, Dropout = 0.5

(3) This run had an optimal validation accuracy of 99.7635 and an optimal test accuracy of 99.7297. I also tested this multiple times, running the SAME set of hyperparameters, to see what is going on with that weird spike. Does it keep showing up? What is going on? Here are more examples, all run at the same hyperparameters as above:

Date: January 9, 2018.

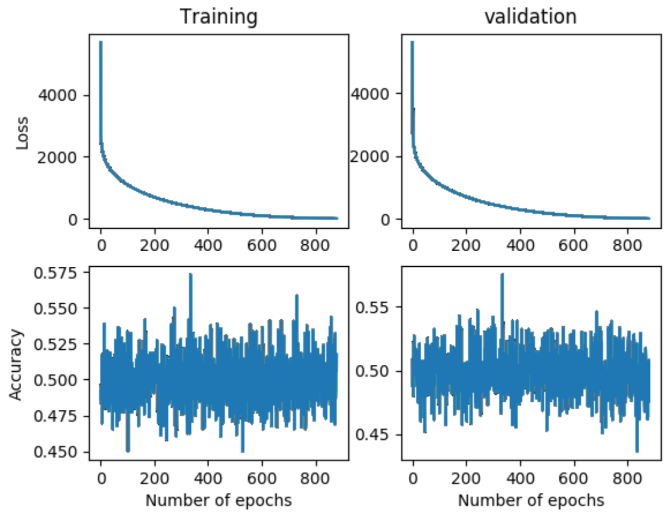

FIGURE 2. Batch Size = 80, Learning Rate = 1e-3, Beta = 0.01, Dropout = 0.5

(4) This run had an optimal validation accuracy of 47.0608, optimal test accuracy of 47.6689, and final validation accuracy of 49.223, test accuracy of 50.0338

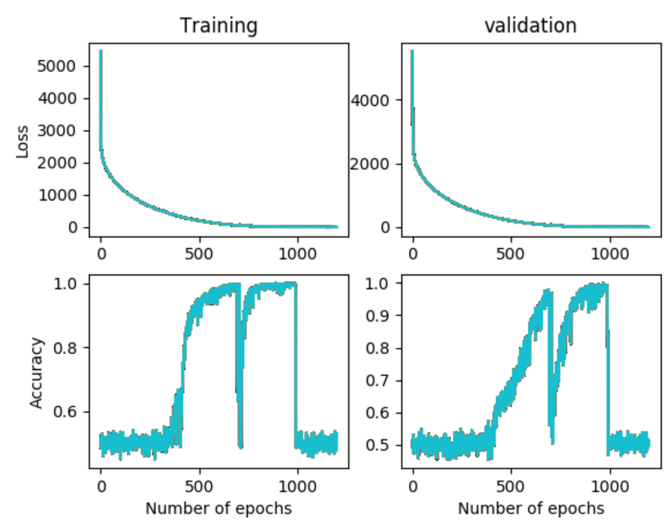

FIGURE 3. Batch Size = 80, Learning Rate = 1e-3, Beta = 0.01, Dropout = 0.5

(5) This run had an optimal validation accuracy of 99.4595, optimal test accuracy of 99.3243, and final validation accuracy of 50.777, test accuracy of 49.9662.

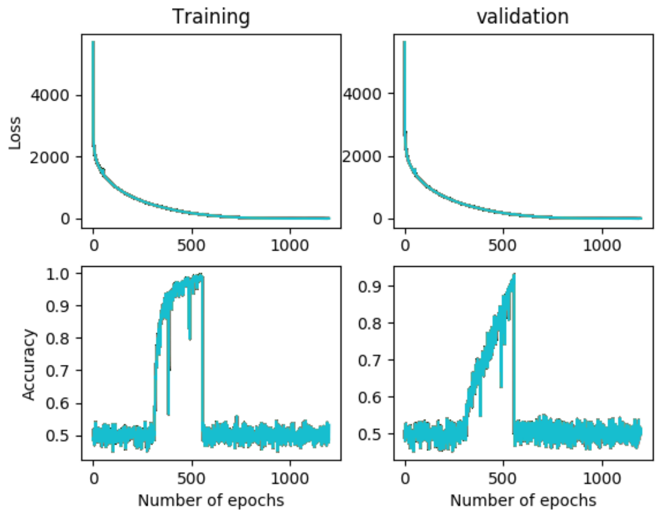

FIGURE 4. Batch Size = 80, Learning Rate = 1e-3, Beta = 0.01, Dropout = 0.5

(6) This run had an optimal validation accuracy of 992.5, optimal test accuracy of 91.7568, and final validation accuracy of 49.223, test accuracy of 50.0338.

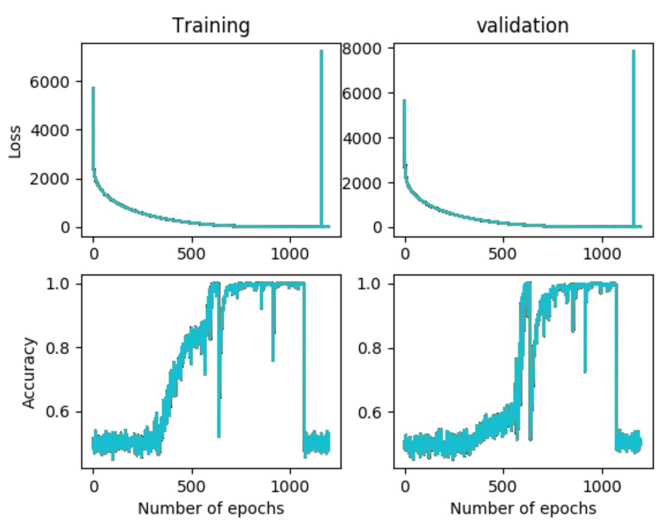

FIGURE 5. Batch Size = 80, Learning Rate = 1e-3, Beta = 0.01, Dropout = 0.5

(7) This run had an optimal validation accuracy of 99.8987, optimal test accuracy of 99.8649, and final validation accuracy of 50.777, test accuracy of 49.9662.

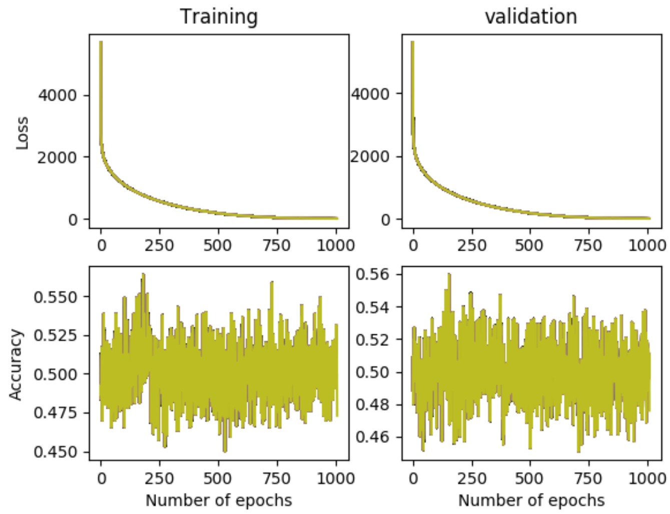

FIGURE 6. Batch Size = 80, Learning Rate = 1e-3, Beta = 0.01, Dropout = 0.5

(8) This run had an optimal validation accuracy of 55.4054, optimal test accuracy of 54.223, and final validation accuracy of 49,223, test accuracy of 50.0338.
(9) The following are images of runs on the dataset that consists of glasses and liquids that are each 12 time steps away from the assumed glass transition temperature of 0.21.

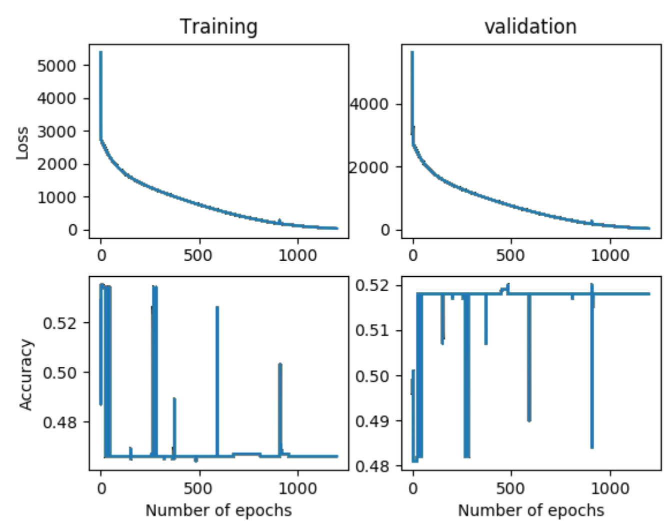

FIGURE 7. Batch Size = 100, Learning Rate = 1e-4, Beta = 0.01, Dropout = 0.5

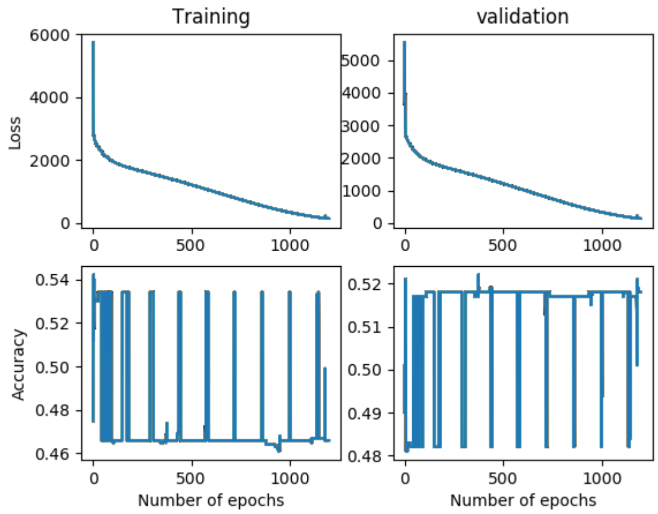

FIGURE 8. Batch Size = 10, Learning Rate = 1e-4, Beta = 0.01, Dropout = 0.5

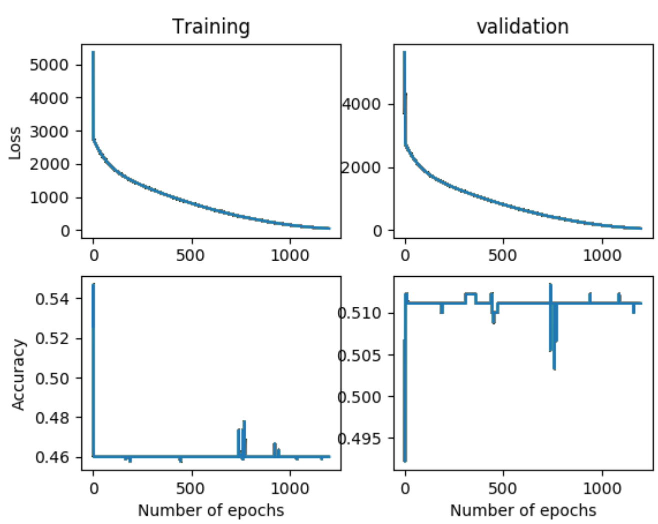

FIGURE 9. Batch Size = 150, Learning Rate = 1e-4, Beta = 0.01, Dropout = 0.5

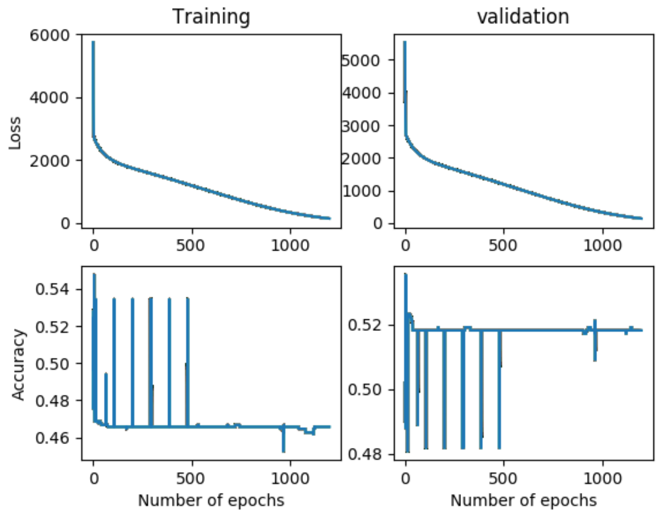

FIGURE 10. Batch Size = 15, Learning Rate = 1e-4, Beta = 0.01, Dropout = 0.5

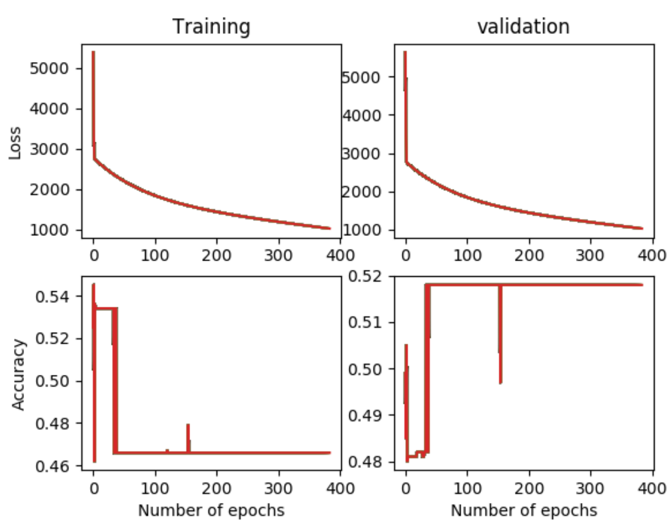

FIGURE 11. Batch Size = 200, Learning Rate = 1e-4, Beta = 0.01, Dropout = 0.5

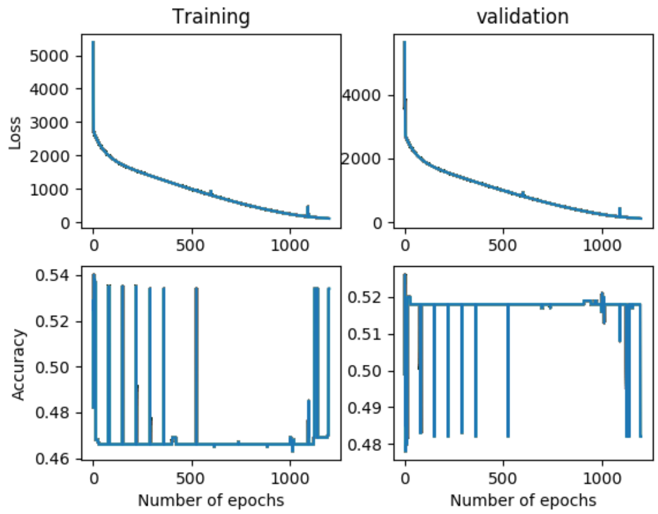

FIGURE 12. Batch Size = 20, Learning Rate = 1e-4, Beta = 0.01, Dropout = 0.5

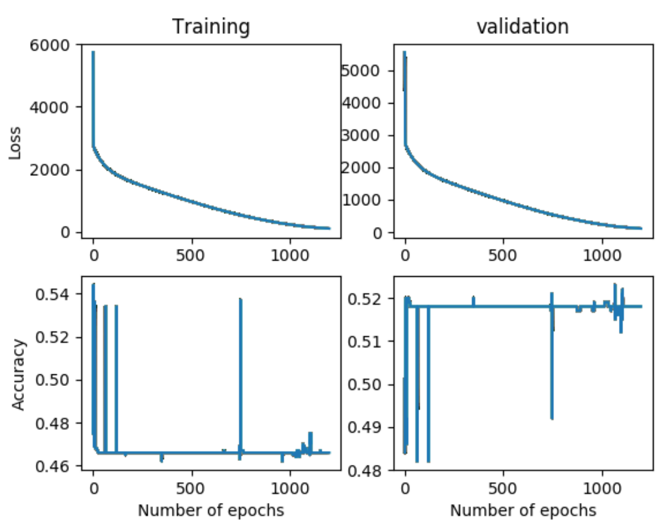

FIGURE 13. Batch Size = 25, Learning Rate = 1e-4, Beta = 0.01, Dropout = 0.5

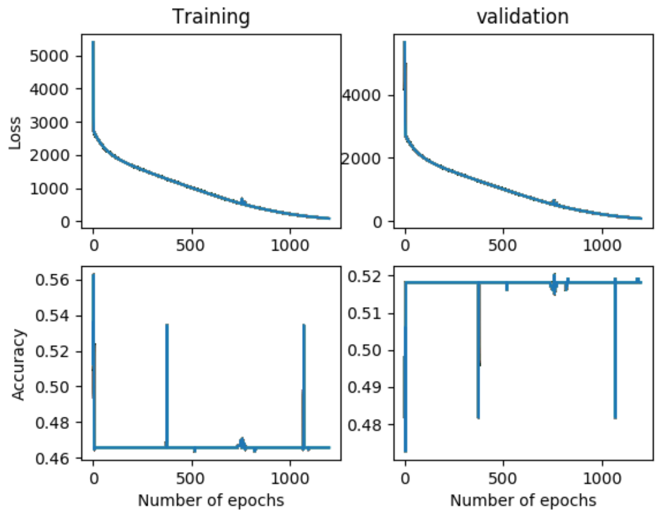

FIGURE 14. Batch Size = 30, Learning Rate = 1e-4, Beta = 0.01, Dropout = 0.5

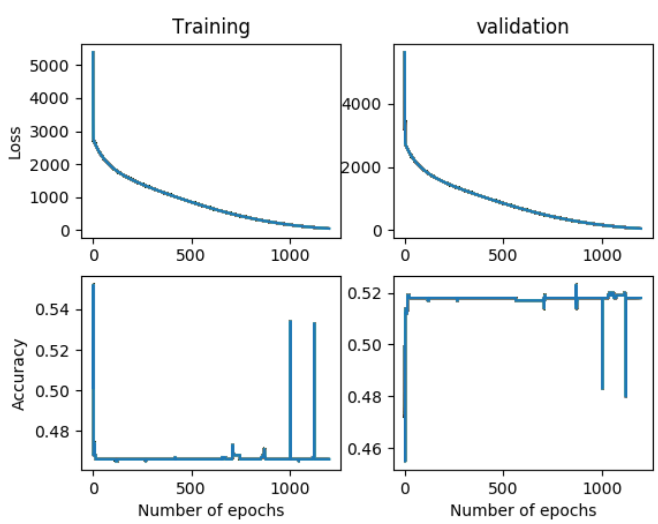

FIGURE 15. Batch Size = 50, Learning Rate = 1e-4, Beta = 0.01, Dropout = 0.5

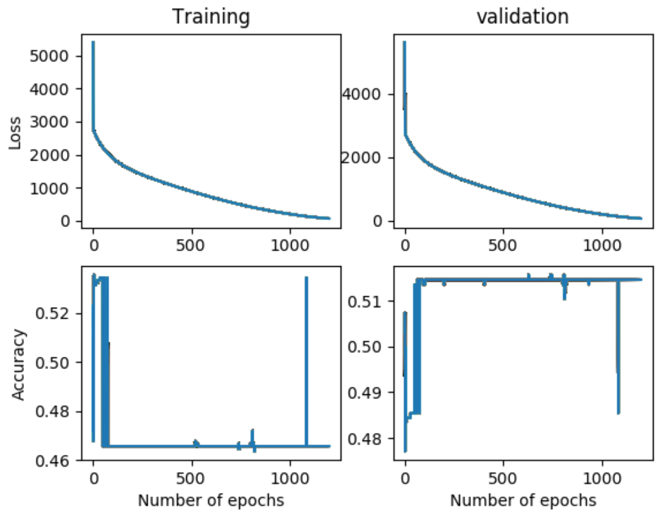

FIGURE 16. Batch Size = 80, Learning Rate = 1e-4, Beta = 0.01, Dropout = 0.5

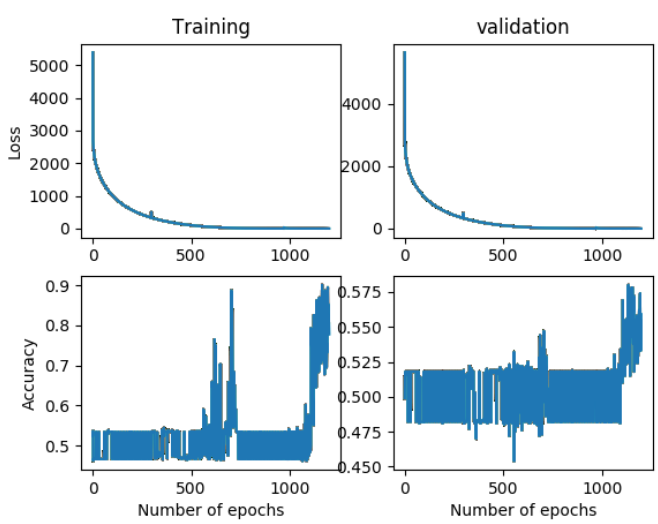

FIGURE 17. Batch Size = 100, Learning Rate = 1e-3, Beta = 0.01, Dropout = 0.5

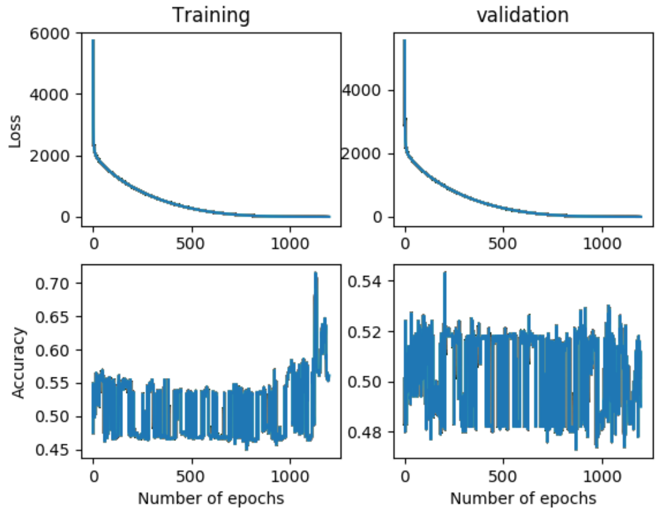

FIGURE 18. Batch Size = 10, Learning Rate = 1e-3, Beta = 0.01, Dropout = 0.5

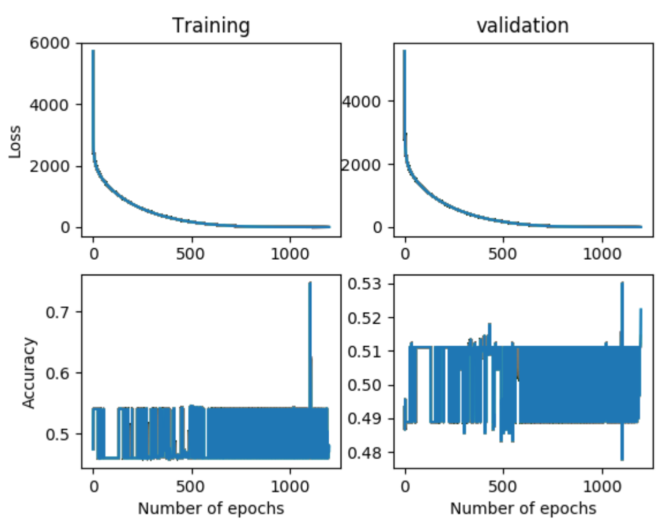

FIGURE 19. Batch Size = 150, Learning Rate = 1e-3, Beta = 0.01, Dropout = 0.5

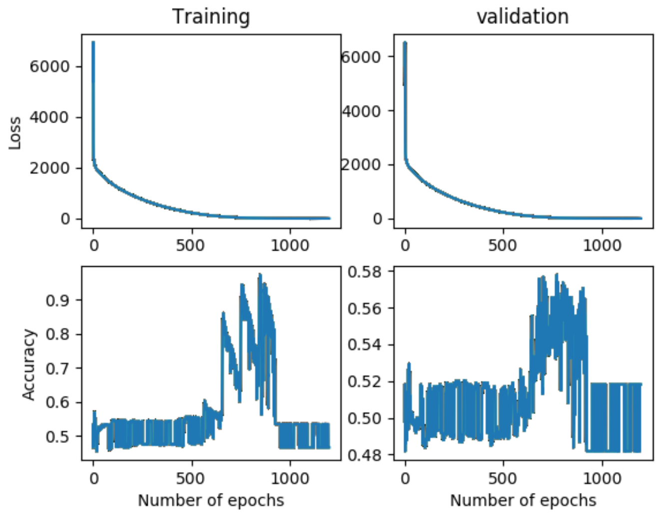

FIGURE 20. Batch Size = 15, Learning Rate = 1e-3, Beta = 0.01, Dropout = 0.5

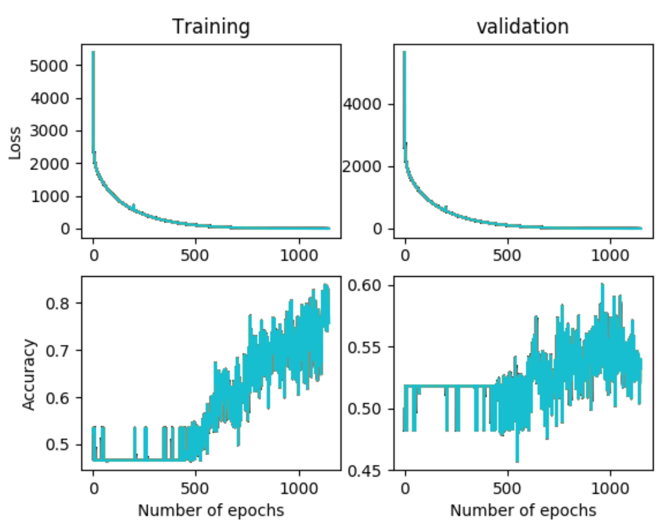

FIGURE 21. Batch Size = 200, Learning Rate = 1e-3, Beta = 0.01, Dropout = 0.1

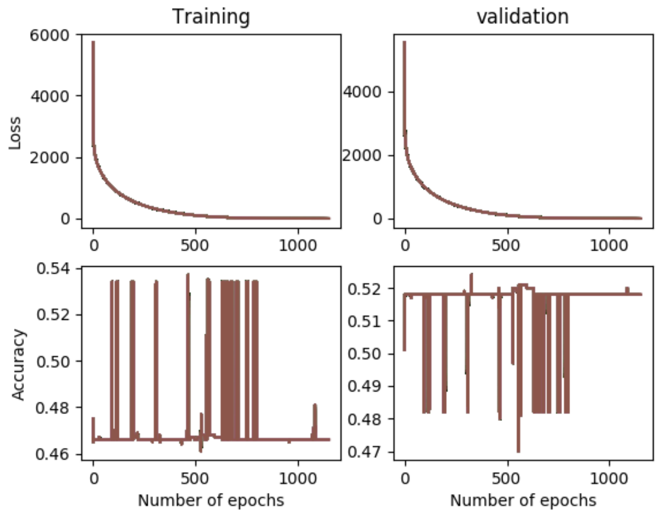

FIGURE 22. Batch Size = 200, Learning Rate = 1e-3, Beta = 0.01, Dropout = 0.2

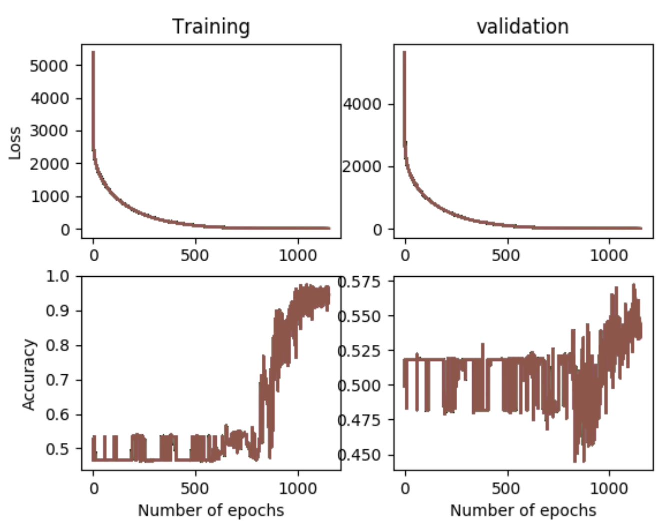

FIGURE 23. Batch Size = 200, Learning Rate = 1e-3, Beta = 0.01, Dropout = 0.3

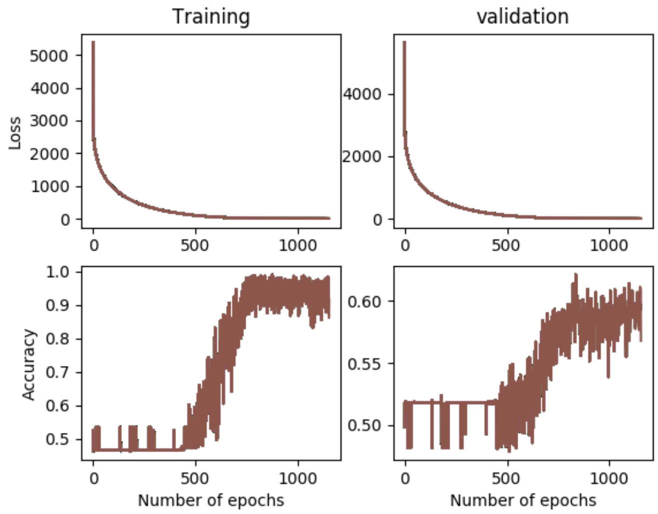

FIGURE 24. Batch Size = 200, Learning Rate = 1e-3, Beta = 0.01, Dropout = 0.4

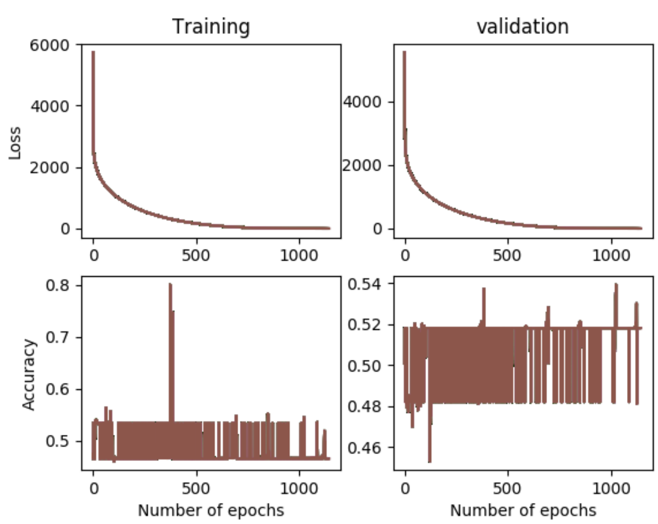

FIGURE 25. Batch Size = 200, Learning Rate = 1e-3, Beta = 0.01, Dropout = 0.5

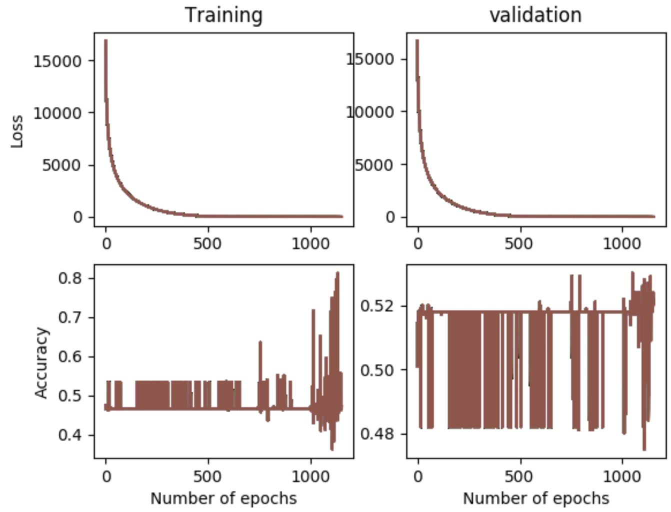

FIGURE 26. Batch Size = 200, Learning Rate = 1e-3, Beta = 0.01, Dropout = 0.3

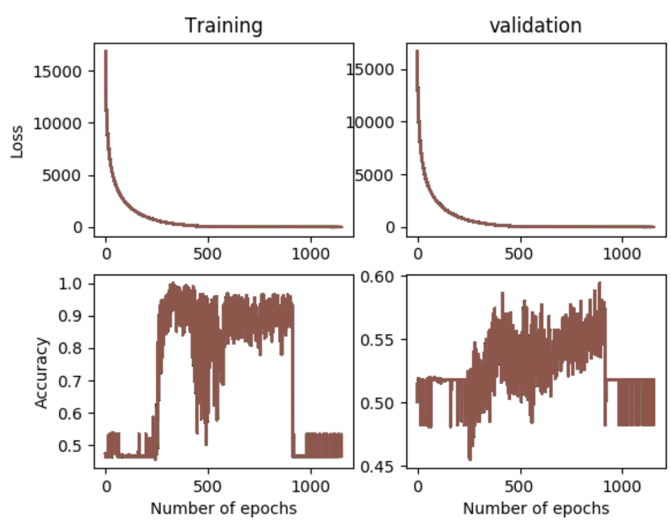

FIGURE 27. Batch Size = 200, Learning Rate = 1e-3, Beta = 0.01, Dropout = 0.4

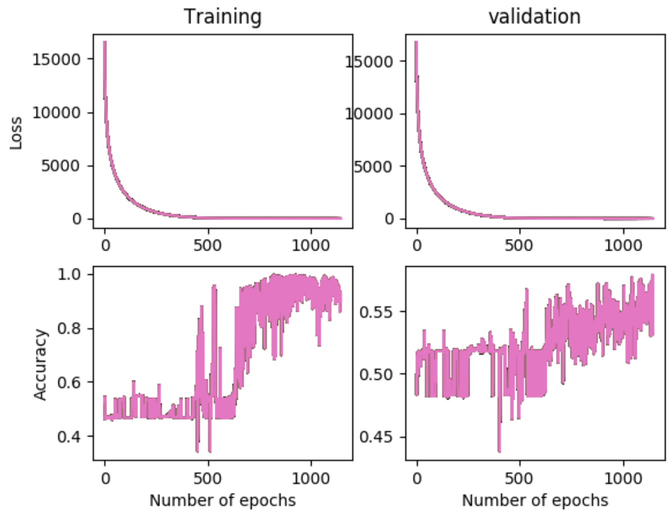

FIGURE 28. Batch Size = 200, Learning Rate = 1e-3, Beta = 0.01, Dropout = 0.4

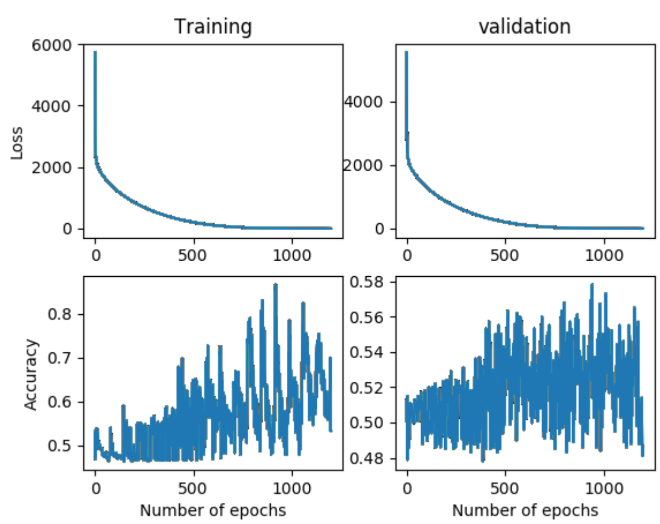

FIGURE 29. Batch Size = 20, Learning Rate = 1e-3, Beta = 0.01, Dropout = 0.5

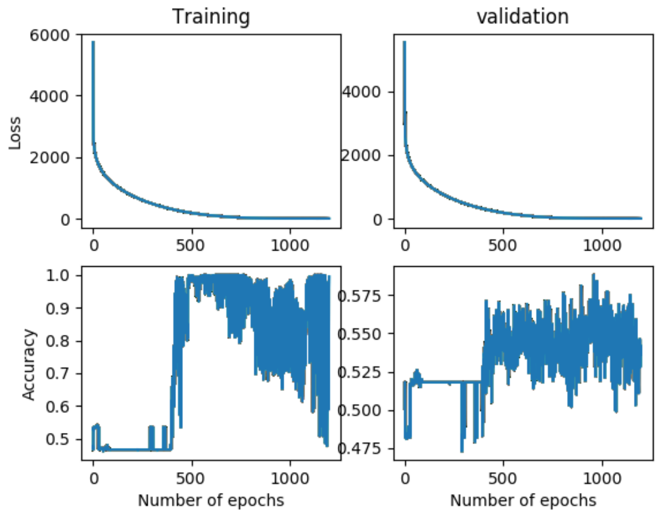

FIGURE 30. Batch Size = 250, Learning Rate = 1e-3, Beta = 0.01, Dropout = 0.5

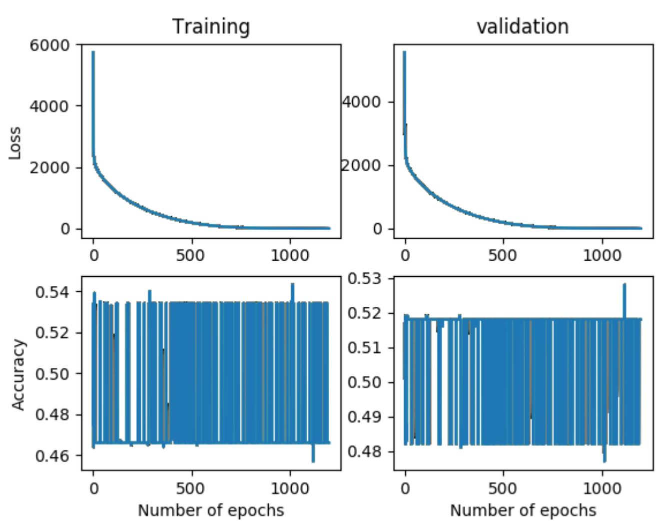

FIGURE 31. Batch Size = 25, Learning Rate = 1e-3, Beta = 0.01, Dropout = 0.5

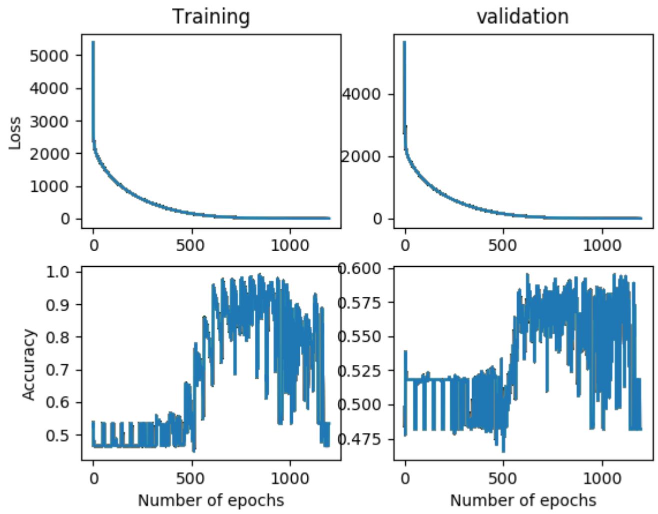

FIGURE 32. Batch Size = 30, Learning Rate = 1e-3, Beta = 0.01, Dropout = 0.5

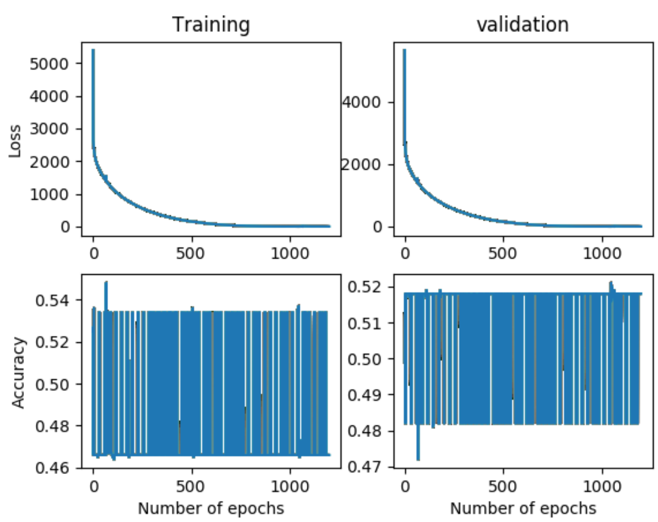

FIGURE 33. Batch Size = 50, Learning Rate = 1e-3, Beta = 0.01, Dropout = 0.5

(10) Visualizations.

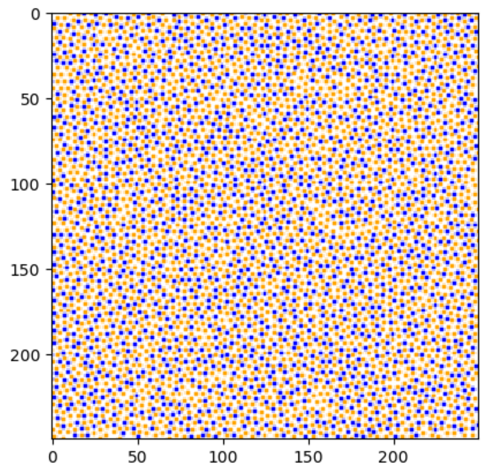

FIGURE 34. Glass original

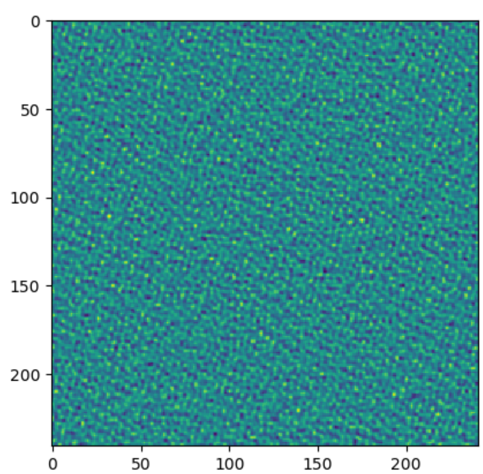

FIGURE 35. Glass channel 1

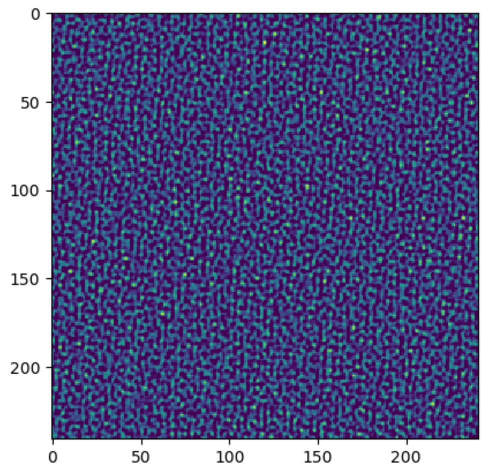

FIGURE 36. Glass channel 2

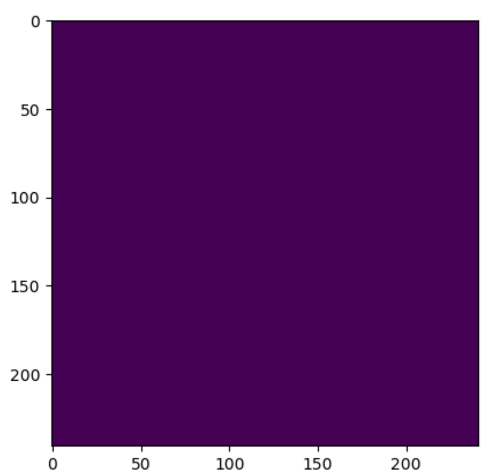

FIGURE 37. Glass channel 3

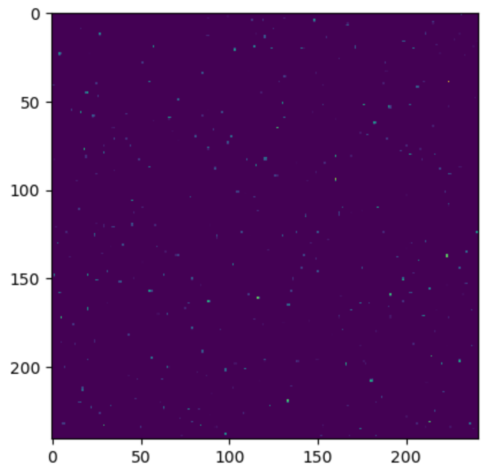

FIGURE 38. Glass channel 4

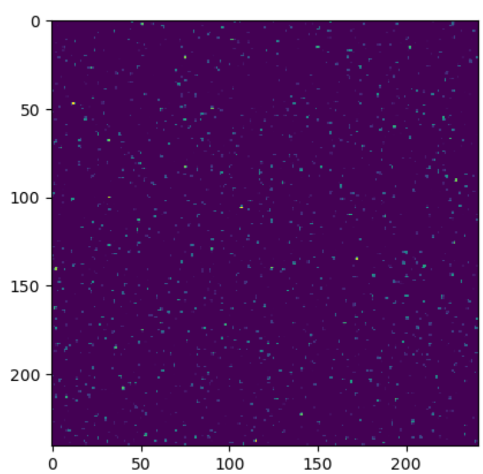

FIGURE 39. Glass channel 5

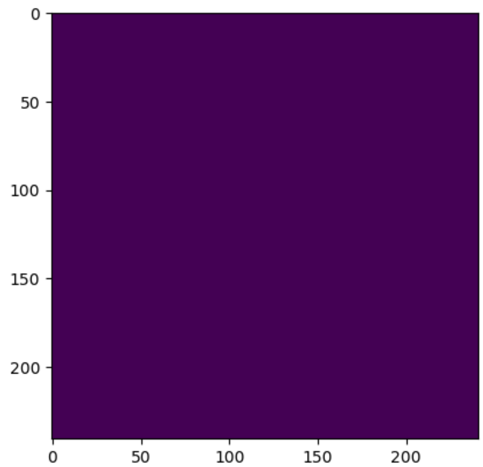

FIGURE 40. Glass channel 6

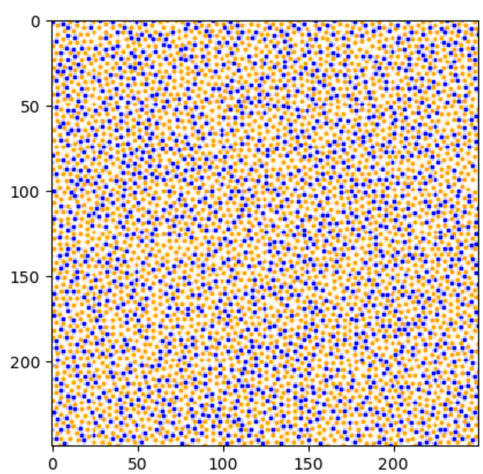

FIGURE 41. Liquid original

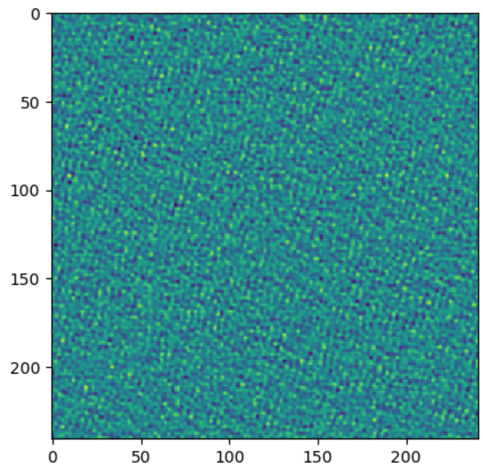

FIGURE 42. Liquid channel 1

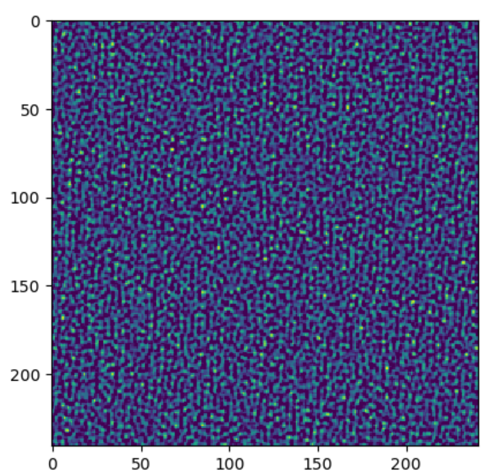

FIGURE 43. Liquid channel 2

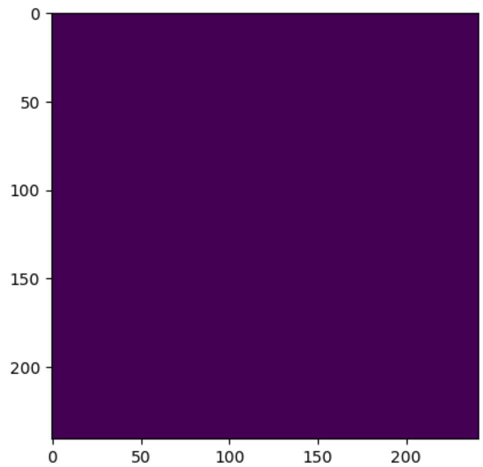

FIGURE 44. Liquid channel 3

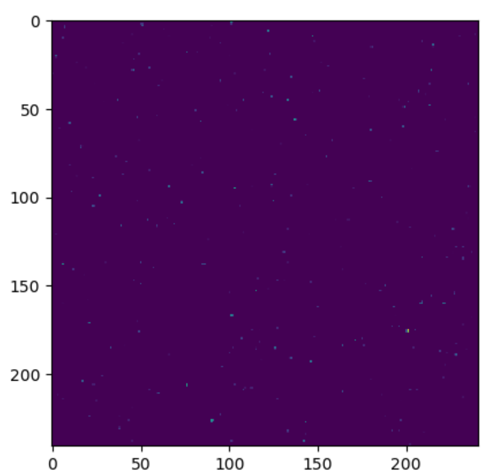

FIGURE 45. Liquid channel 4

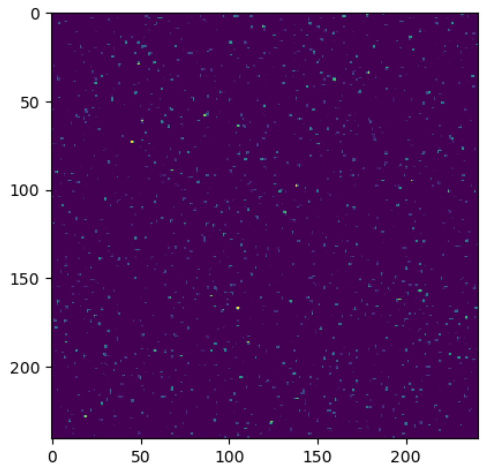

FIGURE 46. Liquid channel 5

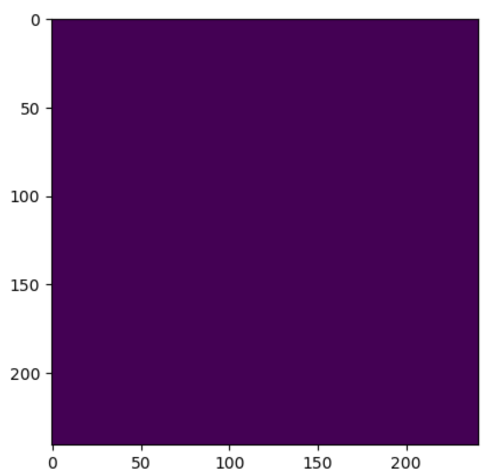

FIGURE 47. Liquid channel 6

2. 12/13/2017

(1) Updated python scripts and merged with GitHub - the recent additions include streamlining the learning rate schedule and data augmentation options. The following is for a standard run on the original endpoint data:

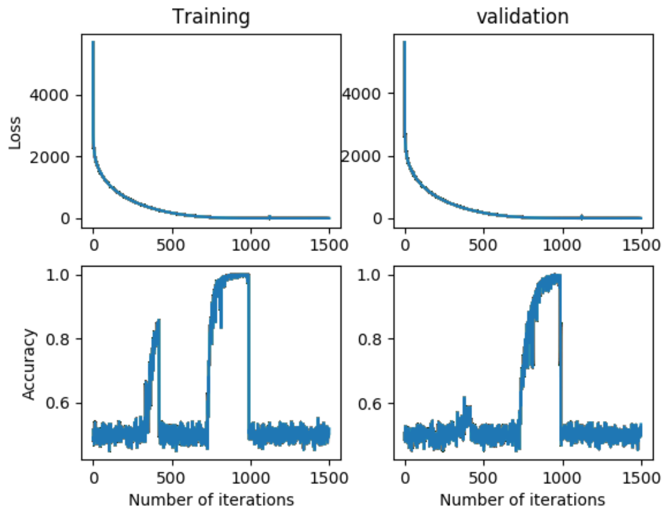

FIGURE 48. Initial Learning Rate = 1e-3, Final Learning Rate = 1e-3, Schedule Length = 550 iterations, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

(2) I then, twice, set the final learning rate to be smaller, at 1e-4:

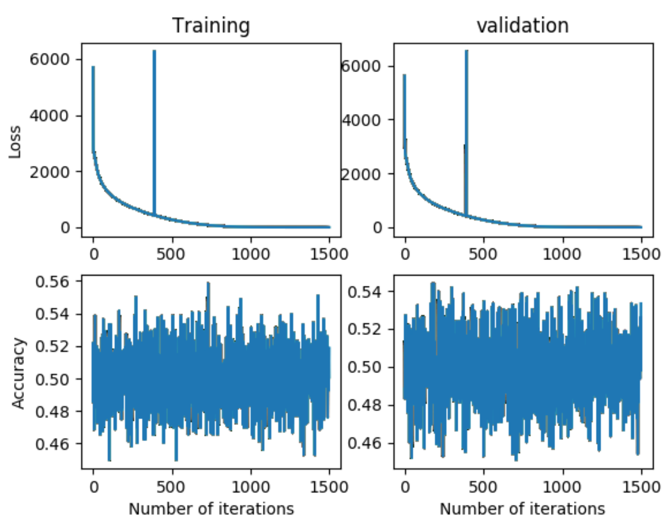

FIGURE 49. Initial Learning Rate = 1e-3, Final Learning Rate = 1e-3, Schedule Length = 550 iterations, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

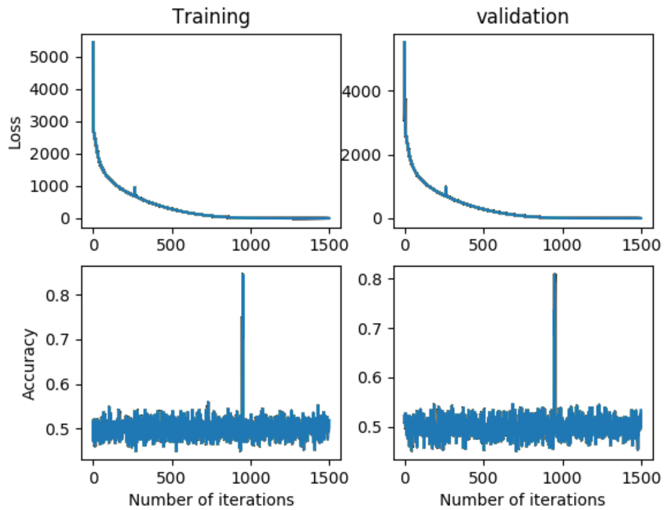

FIGURE 50. Initial Learning Rate = 1e-3, Final Learning Rate = 1e-3, Schedule Length = 550 iterations, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

(3) What is going on here? It seems as though setting the learning rate a magnitude lower really screws things up...
(4) Note: one iteration goes through 10 batches, so if the batch size is 80, then one iteration goes through 800 examples. The training set is 70 percent of the total 20,000 images, so the training set is 14,000 images. Thus, one epoch is 17.5 iterations, or approximately 18 iterations. That means that 1500 iterations is about 83 epochs. Question: how many epochs is typically needed for a binary classification problem? And how long does this typically take?
(5) I also tested the usual hyperparameters on the original endpoint data and turned on the data augmentation, to see if there would be any issues. I got the following training runs:

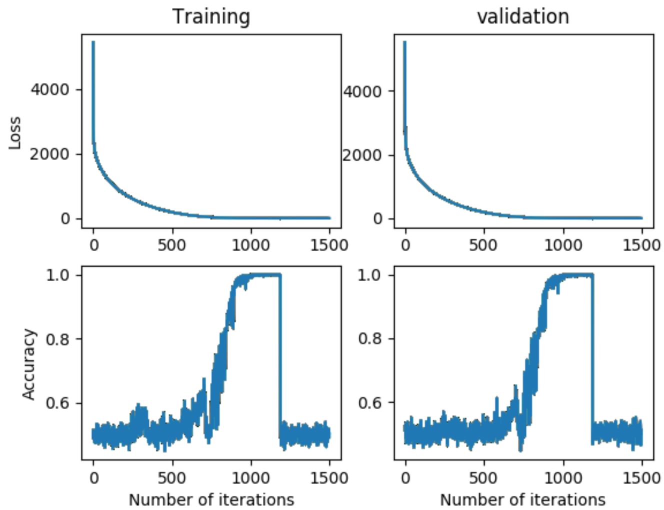

FIGURE 51. Initial Learning Rate = NA, Final Learning Rate = 1e-3, Schedule Length = NA, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

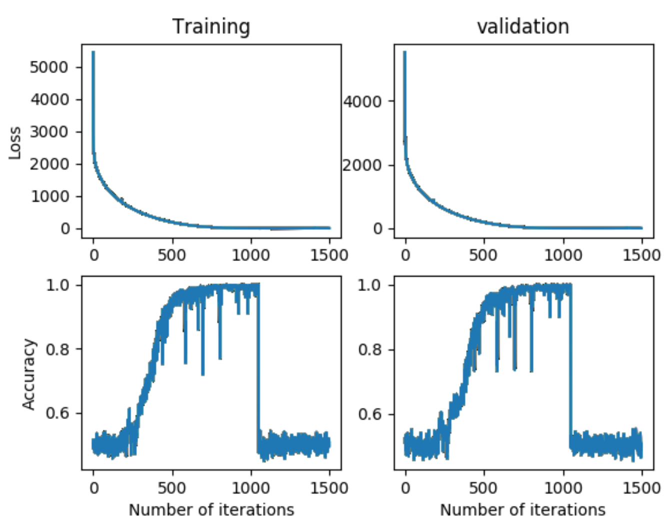

FIGURE 52. Initial Learning Rate = NA, Final Learning Rate = 1e-3, Schedule Length = NA, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

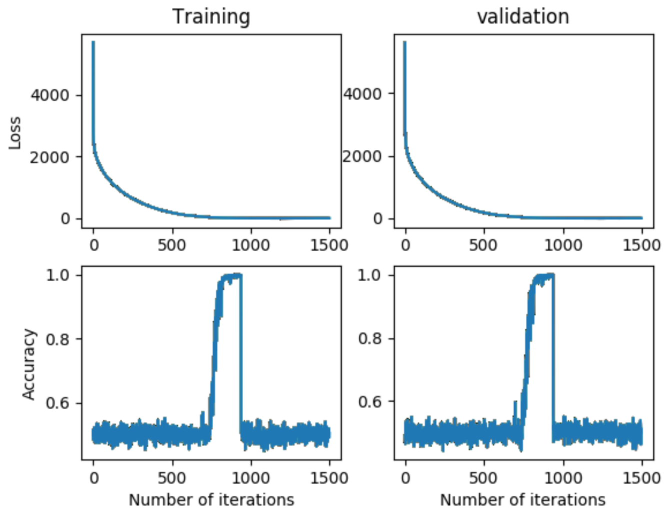

FIGURE 53. Initial Learning Rate = NA, Final Learning Rate = 1e-3, Schedule Length = NA, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

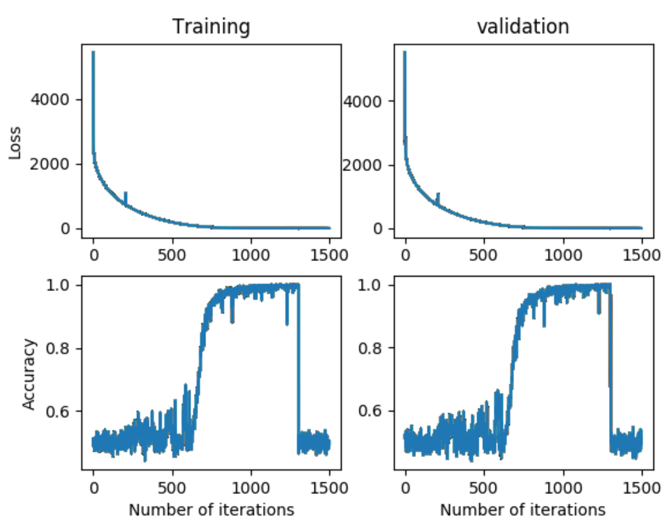

FIGURE 54. Initial Learning Rate = NA, Final Learning Rate = 1e-3, Schedule Length = NA, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

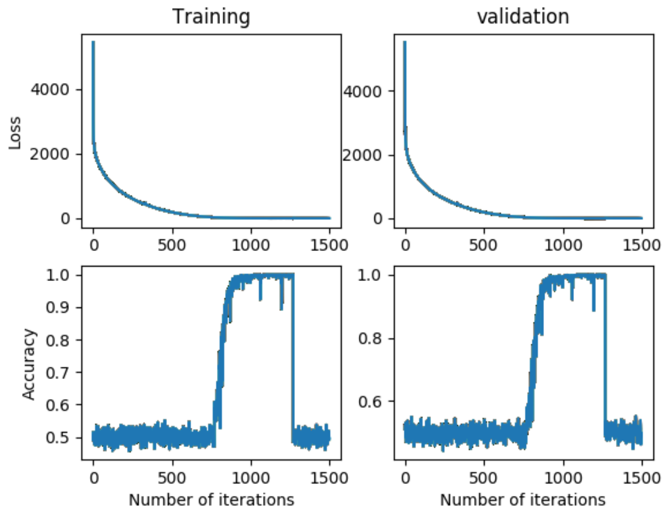

FIGURE 55. Initial Learning Rate = NA, Final Learning Rate = 1e-3, Schedule Length = NA, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

(6) I should double check that the dataset augmentation is actually working.

3. 12/22/2017

(1) I am now going to run a series of training runs that will explore the relationship between the learning rate schedule and accuracy. First, let's try the threshold idea. Then, we will try the standard exponential decay idea. Let's try the following. When the accuracy gets to 90 percent, let's decrease the learning rate by an order of magnitude. Let's just run that simple test, say, 8 times, and see what happens!
(2) Initial learning rate = , Final learning rate = ,

4. 12/24/2017

(1) We ran a series of training runs that used a learning rate threshold. The learning rate starts at the initial learning rate. The learning rate changes to the final learning rate whenever the validation accuracy is above 90 percent. This allows the learning rate to go back to the initial learning rate if the accuracy drops below 90 percent. Here are plots showing six tests. These plots demonstrate that the sharp drops in accuracy were, indeed, the result of a learning rate that was too large towards the end of training, since there are no sharp drops in accuracy here:

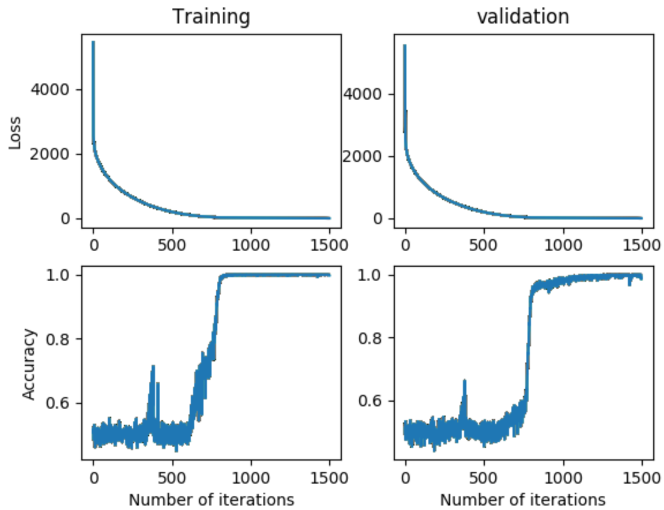

FIGURE 56. Initial Learning Rate = 1e-3, Final Learning Rate = 1e-4, Schedule Length = NA, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

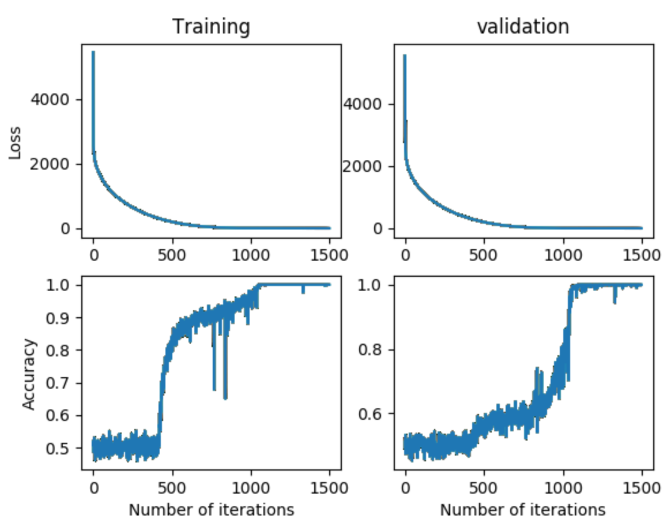

FIGURE 57. Initial Learning Rate = 1e-3, Final Learning Rate = 1e-4, Schedule Length = NA, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

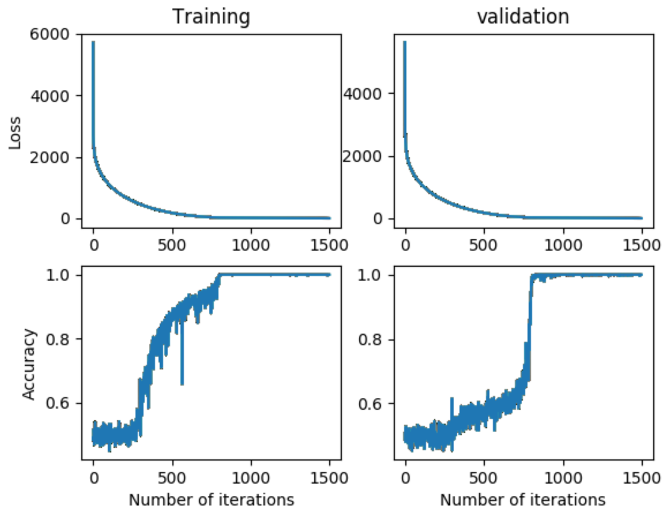

FIGURE 58. Initial Learning Rate = 1e-3, Final Learning Rate = 1e-4, Schedule Length = NA, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

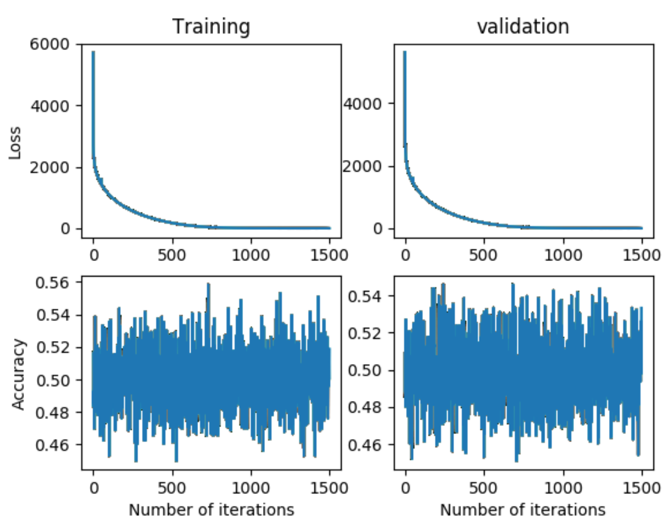

FIGURE 59. Initial Learning Rate = 1e-3, Final Learning Rate = 1e-4, Schedule Length = NA, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

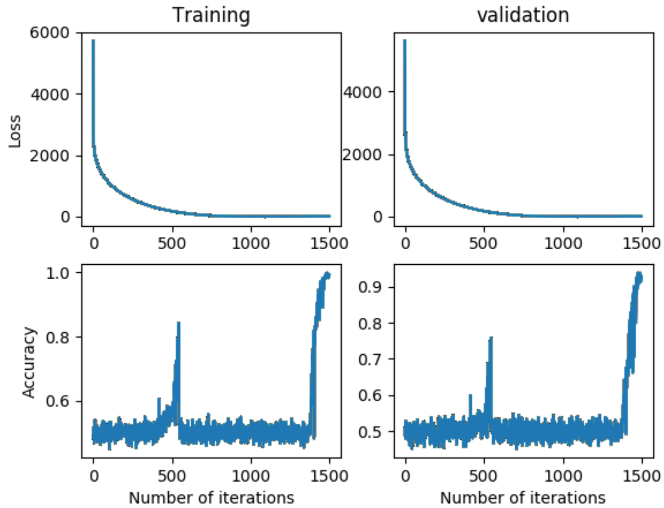

FIGURE 60. Initial Learning Rate = 1e-3, Final Learning Rate = 1e-4, Schedule Length = NA, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

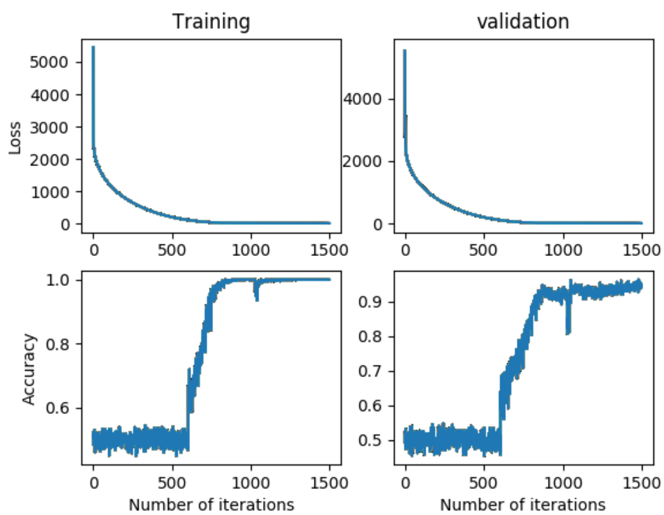

FIGURE 61. Initial Learning Rate = 1e-3, Final Learning Rate = 1e-4, Schedule Length = NA, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

4.1. **12/29/2017.**

(1) Ran another series of runs that used a similar learning rate threshold just to check that my GitHub had been updated appropriately. This time used only 1e-3 to 1e-4. In one example, the accuracy crashed. Perhaps evidences that smaller learning rates are needed closer to the minimum. For the sake of completeness, these extra runs are reported below:

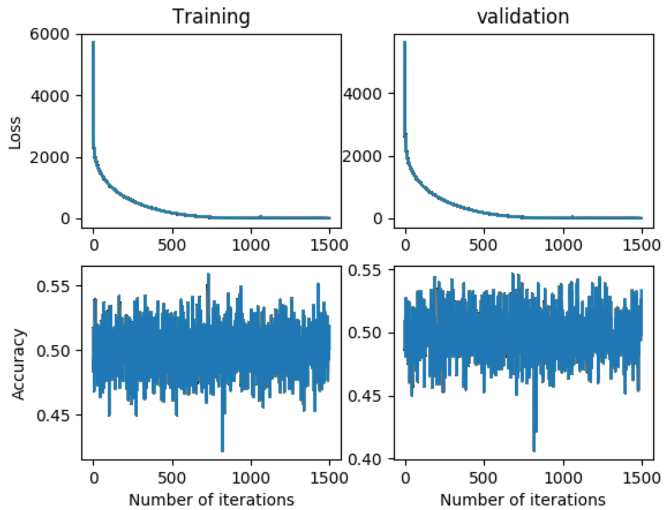

FIGURE 62. Initial Learning Rate = 1e-3, Final Learning Rate = 1e-4, Schedule Length = NA, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

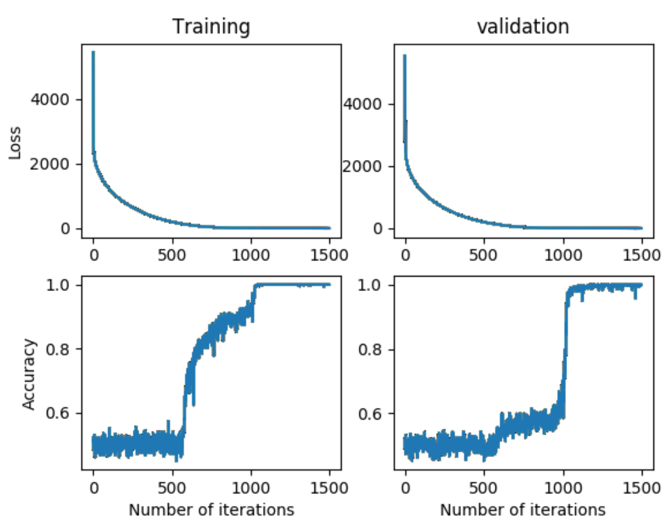

FIGURE 63. Initial Learning Rate = 1e-3, Final Learning Rate = 1e-4, Schedule Length = NA, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

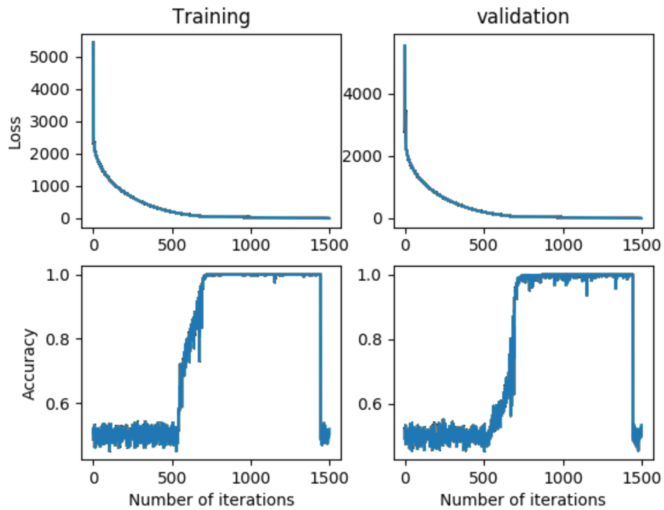

FIGURE 64. Initial Learning Rate = 1e-3, Final Learning Rate = 1e-4, Schedule Length = NA, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

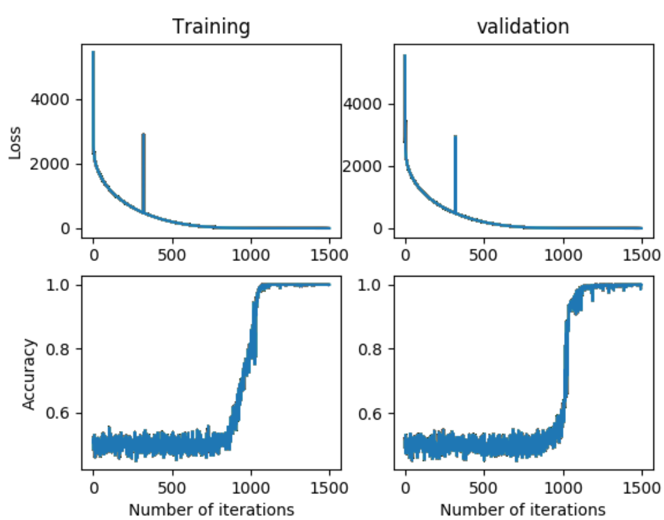

FIGURE 65. Initial Learning Rate = 1e-3, Final Learning Rate = 1e-4, Schedule Length = NA, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

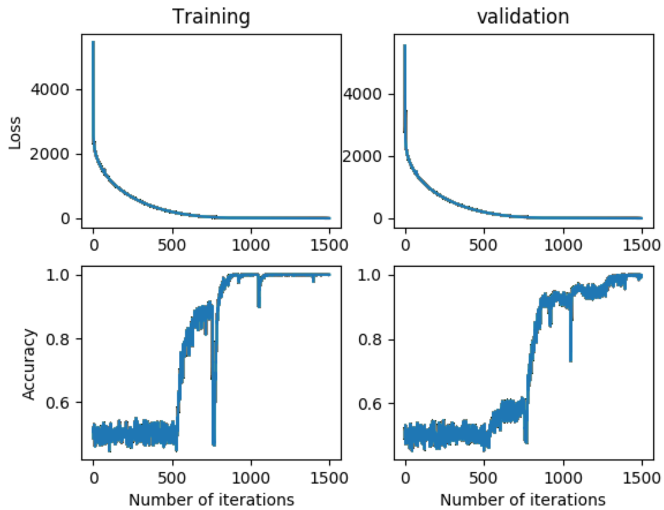

FIGURE 66. Initial Learning Rate = 1e-3, Final Learning Rate = 1e-4, Schedule Length = NA, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

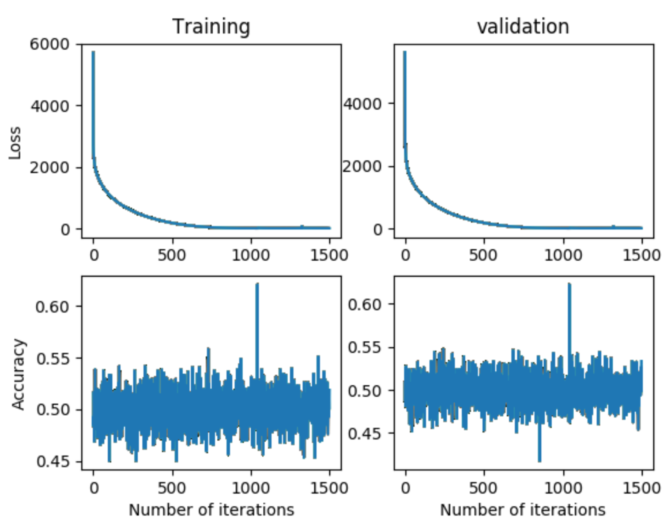

FIGURE 67. Initial Learning Rate = 1e-3, Final Learning Rate = 1e-4, Schedule Length = NA, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

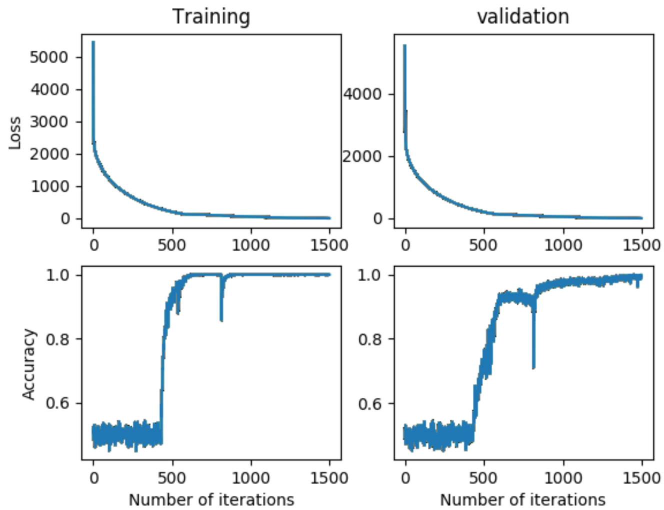

FIGURE 68. Initial Learning Rate = 1e-3, Final Learning Rate = 1e-4, Schedule Length = NA, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

4.2. **12/30/2017.**

(1) Tested the data augmentation scheme manually. Printed out a series of images and their respective augmented versions that are given when generator.next is invoked, respectively with the data aug option set to false and with the data aug option set to true.

4.3. **1/1/2018.**

(1) Added main_glassliquid_restart.py, a file that can run training starting from a previous saved model. Tested it. Now, let's try to train again on the dataset that consists of glasses and liquids that are each 12 time steps away from the assumed glass transition temperature of 0.21. Previously, the best parameters were batch size 200, learning rate 1e-3, beta 0.01, dropout 0.4. Now, we are going to do a series of simulations using the same parameters, except, first, we are going to include dataset augmentation, and second, we are going to run the simluation at least twice as long. Then, once we get a sense for how these train, we will add in a learning rate threshold step, or even multiple threshold steps if necessary!

4.4. **1/4/2018.**

(1) Realized that I had run on the original endpoint data. So, copied over the folder called "data_12steps" into the current machine-learning-glasses directory, removed the current metadata folder, and created a new metadata folder with this data. Then, set up new training runs again.

5. 1/9/2018

(1) Collected data from the above-mentioned simulations. The training curves are as follows:

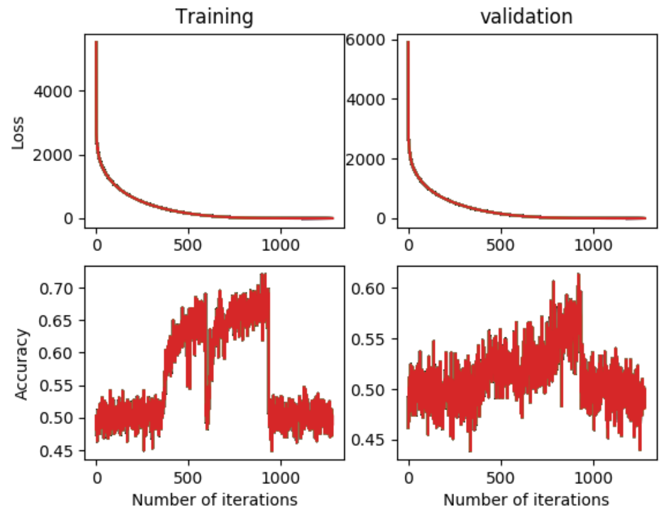

FIGURE 69. Initial Learning Rate = 1e-3, Final Learning Rate = 1e-4, Schedule Length = NA, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

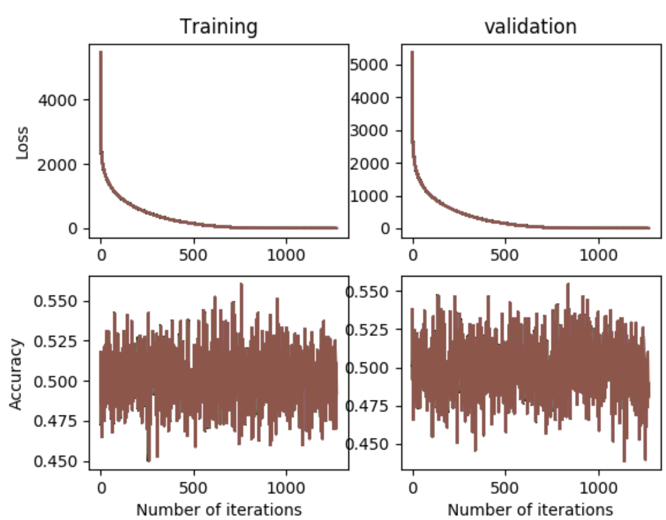

FIGURE 70. Initial Learning Rate = 1e-3, Final Learning Rate = 1e-4, Schedule Length = NA, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

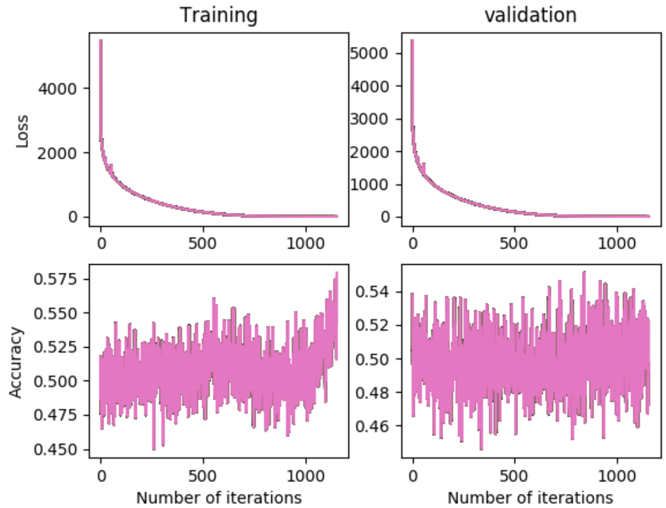

FIGURE 71. Initial Learning Rate = 1e-3, Final Learning Rate = 1e-4, Schedule Length = NA, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

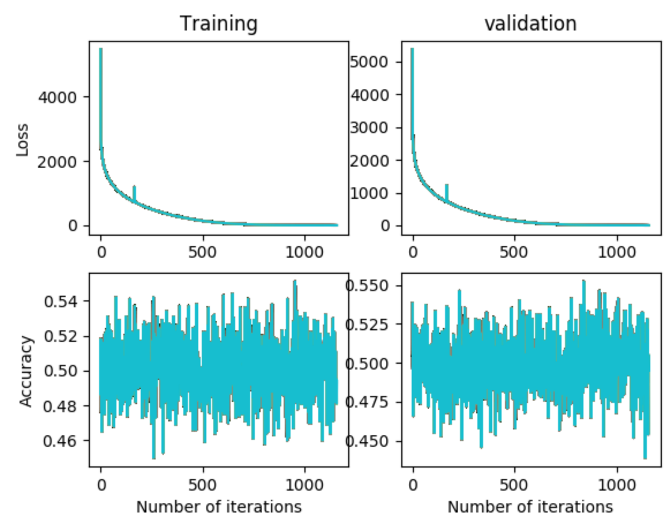

FIGURE 72. Initial Learning Rate = 1e-3, Final Learning Rate = 1e-4, Schedule Length = NA, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

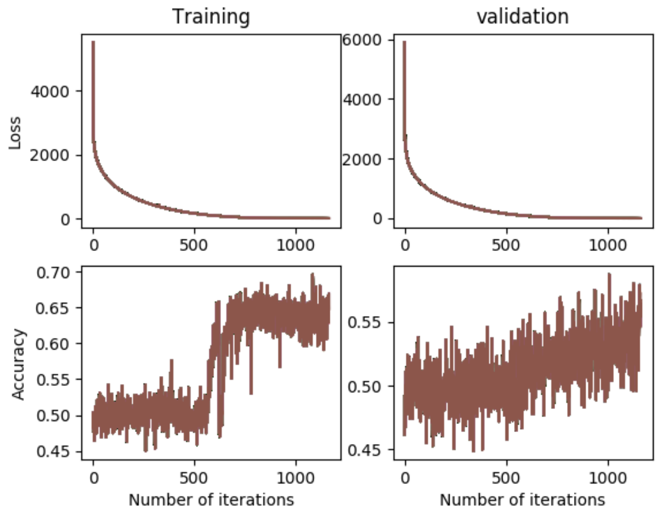

FIGURE 73. Initial Learning Rate = 1e-3, Final Learning Rate = 1e-4, Schedule Length = NA, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

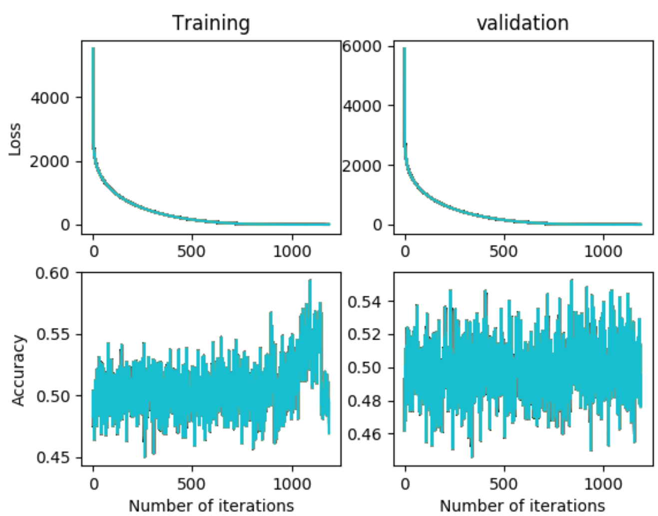

FIGURE 74. Initial Learning Rate = 1e-3, Final Learning Rate = 1e-4, Schedule Length = NA, Batch Size = 80, L2 beta = 0.01, Dropout = 0.5

(2) The problem is that my allotted time (36 hours) was not enough, so the simulations terminated early. My program, however, was only set to save the current model at the end of the iterations. So, the current model was not saved, only the best model was saved. So, I went back into the main code and changed it so that the current model is updated and saved at every iteration, in addition to the best model being updated when necessary. I also included restart capability into the main code (using parsing options), so that there is not a separate restart file.

(3) I am now running 7 new simulations. The first is just a re-do of version1 step 0. The others take version5 step 0 and start from the saved best model as a starting point, using the new restart capabilities of the main code. The first two simulations from this starting point (version5 step2 and version5 step2 trial2) use the original parameters. The next two (version5 step2 trial3 and version5 step2 trial4 use eta_initial=1e-4 and eta_final=1e-5), and the final two (version5 step2 trial5 and version5 step2 trial6) use eta_intial=1e-5 and eta_final=1e-6. We will see how these 7 new runs turn out.

References

[R] D. Reid, I. Lyubimov, M. D. Ediger, J. J. de Pablo, *Age and structure of a model vapour-deposited glass*, Nature Communications, 2016.

[QM] J. Gilmer, S. Schoenholz, P. Riley, O. Vinyals, and G. Dahl, *Neural Message Passing for Quantum Chemistry*, www.arxiv.org, 2017.

[GC] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, *Molecular Graph Convolutions: Moving Beyond Fingerprints*, www.arxiv.org, 2017.

[P] Q. Wei, R. G. Melko, and Jeff Z. Y. Chen, *Identifying polymer states by machine learning*, www.arxiv.org, 2017.

E-mail address: `swansonk1@uchicago.edu`

INSTITUTE FOR MOLECULAR ENGINEERING, UNIVERSITY OF CHICAGO, 5640 S ELLIS AVE, CHICAGO, IL 60637