APRML : Assignment 2

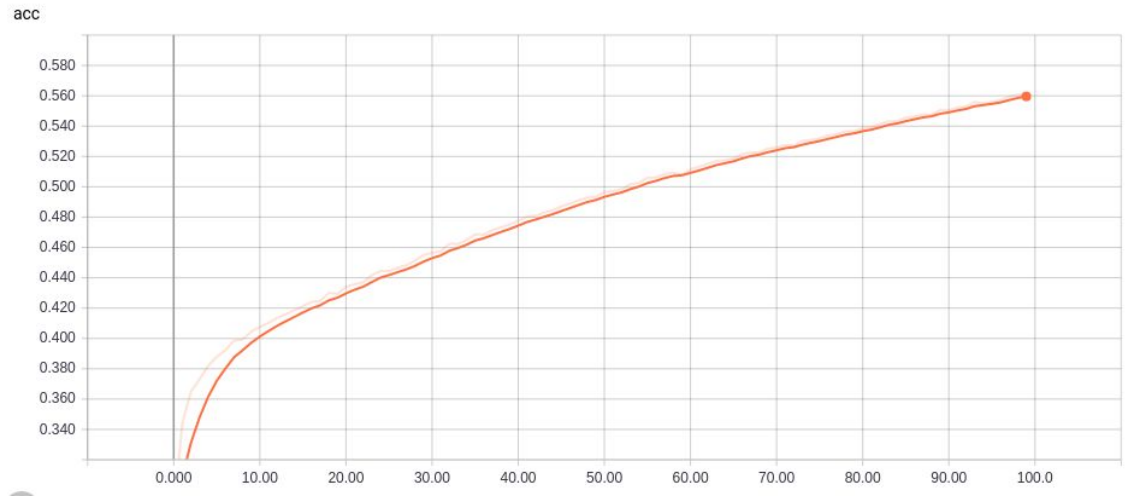
Anshuman Suri: 2014021

Note:

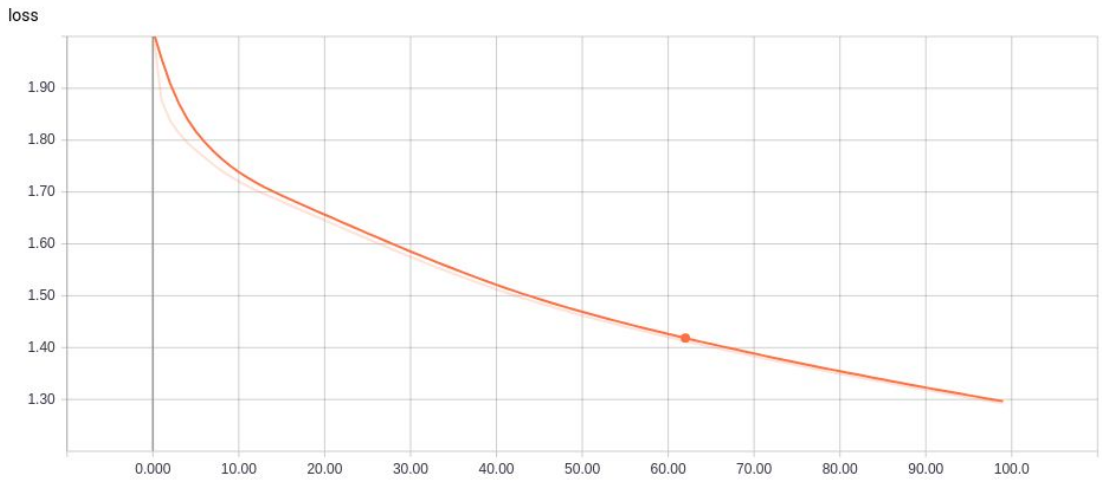
- Curing overfitting was not given emphasis, as it wasn't the objective of the assignment. Using data augmentation is known to help, and would've boosted testing accuracies for all the models in the assignment.
- Grid search was not employed, as resources + time was limited (someone kept resetting the GPU every few hours, which caused a lot of problems). All of the experiments have been run with the same optimizer (*RMSProp, lr=1e-4,wd=1e-6)*. When not specified, the batch size is 16 and dropout is 0. Max pooling has been used after every block (wherever possible).

1. **Testing accuracy: 50.57%, Loss: 1.42**
   The model's learning rate was kept low (as it was overfitting), because of which both testing and training accuracies are low. The loss seems to be converging on both test and train data, which is a good sign, The gradients, however, seem have the vanishing gradient problem, as there is little to no variation in weights in the first layer (as gradient is low)

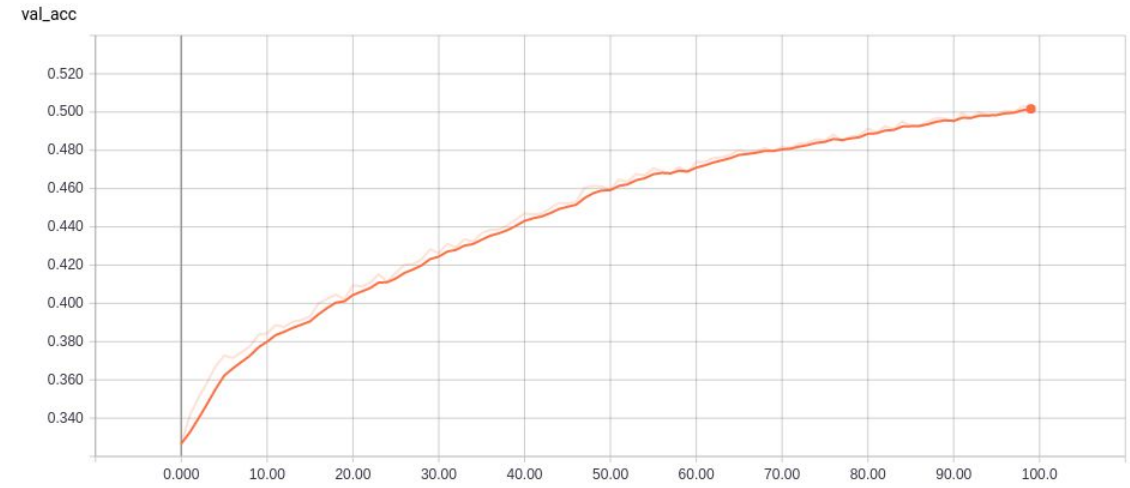
   Training accuracy:



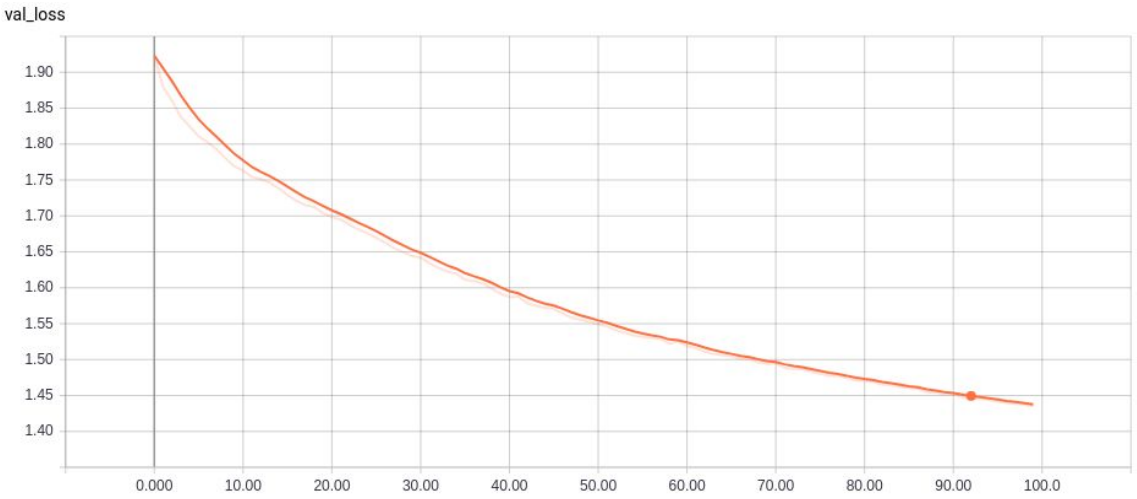
   Training loss:



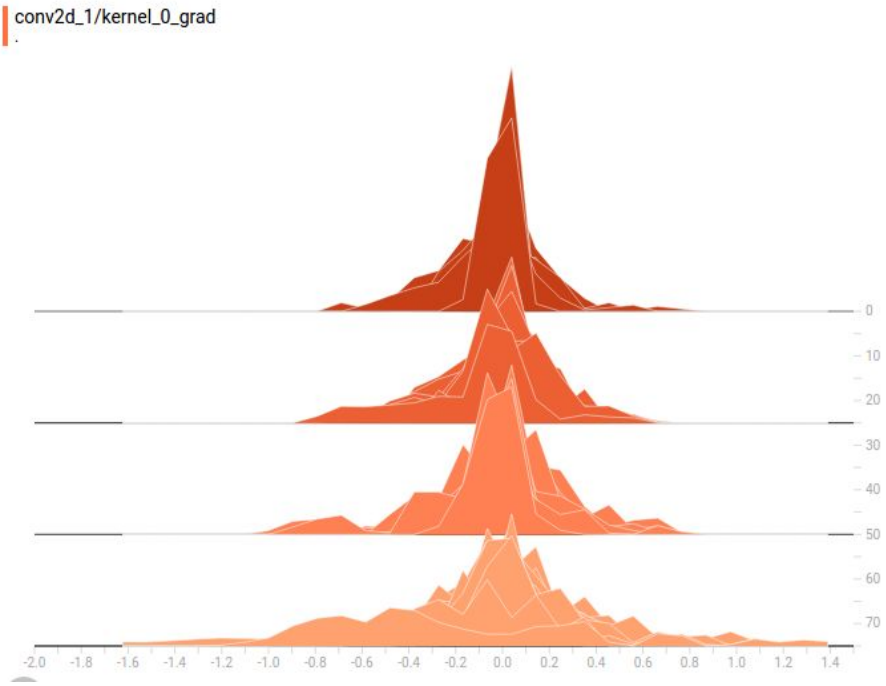
Validation accuracy:



Validation loss:



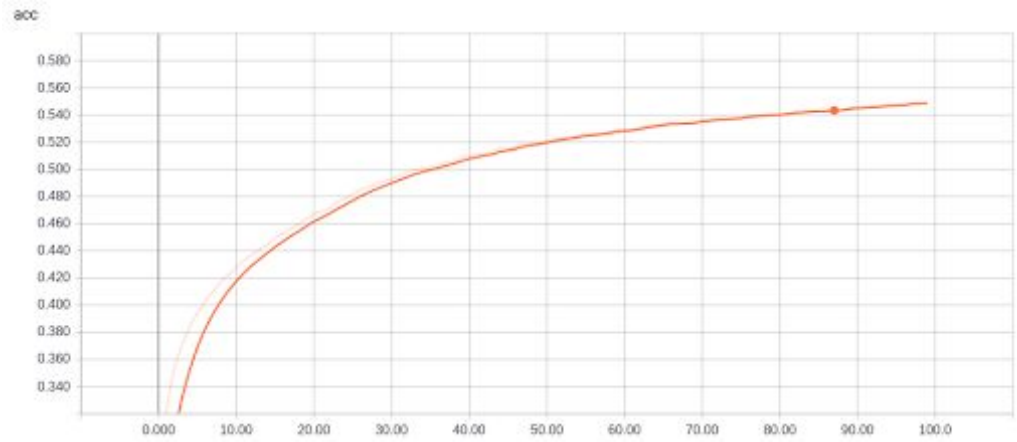
Gradient for kernel in first layer:



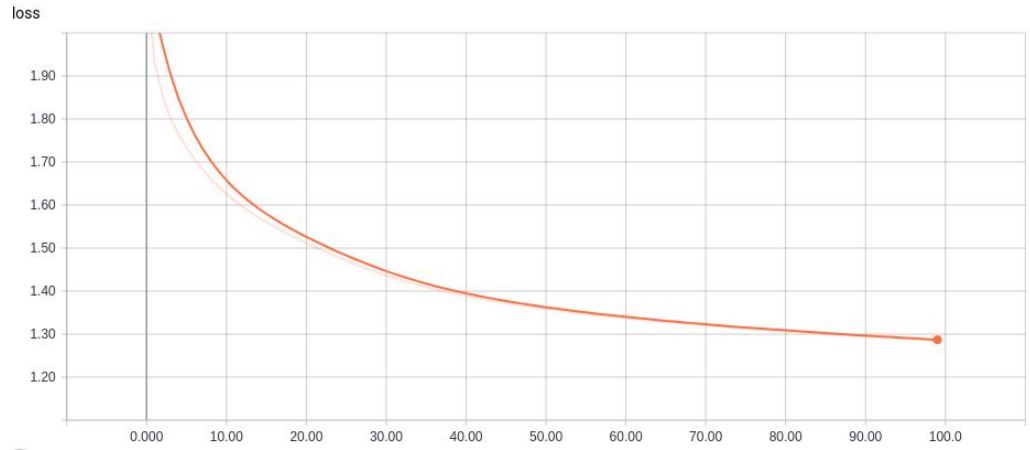
2. Out of all the given activation functions, *Maxout* seems to perform best, with a testing accuracy that is significantly better than the other four activation functions. All of them still face the gradient diminishing problem, as is visible from their gradients.

    a. **Testing accuracy: 52.44%, Loss: 1.38**

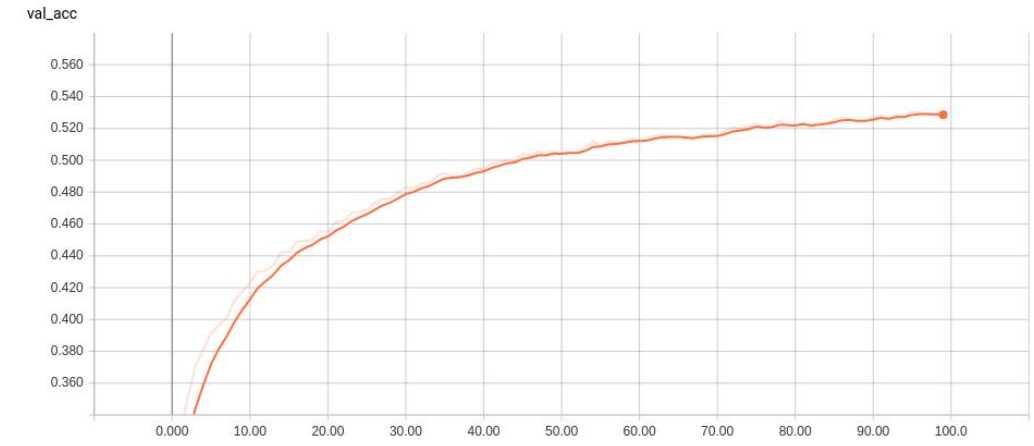
    Training accuracy:



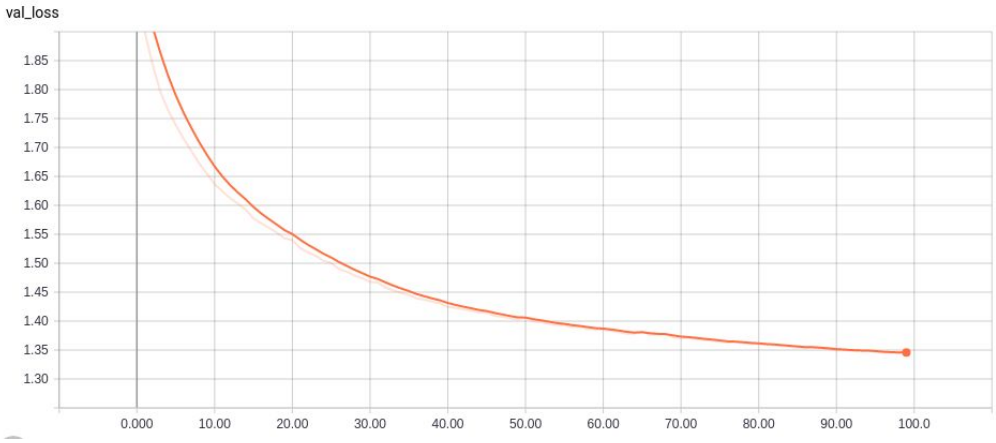
    Training loss:



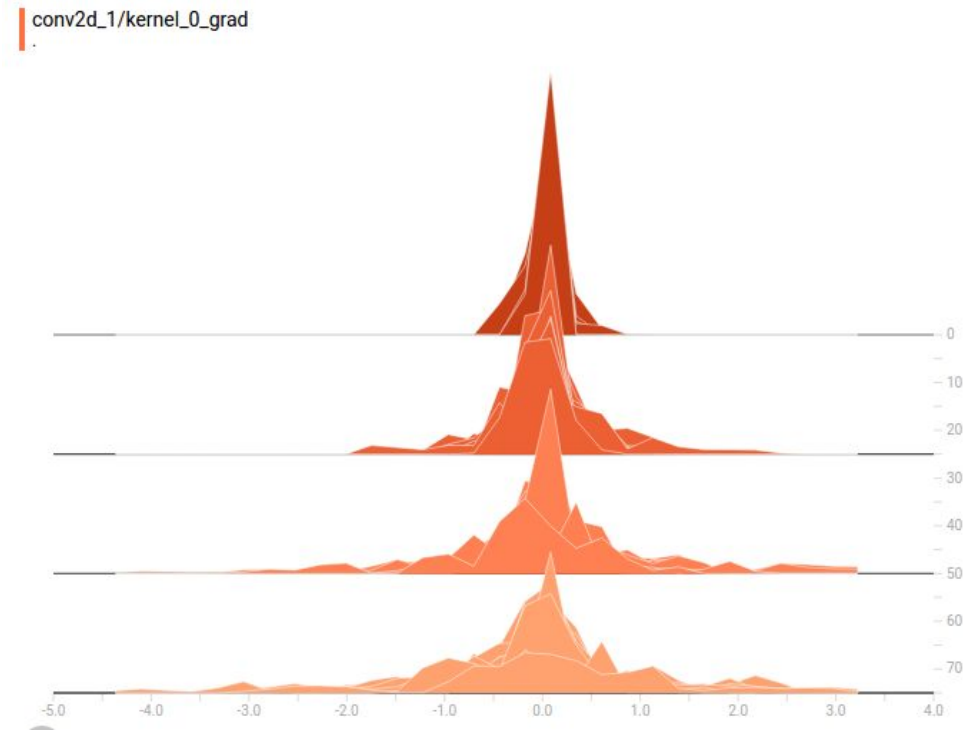
    Validation accuracy:



    Validation loss:

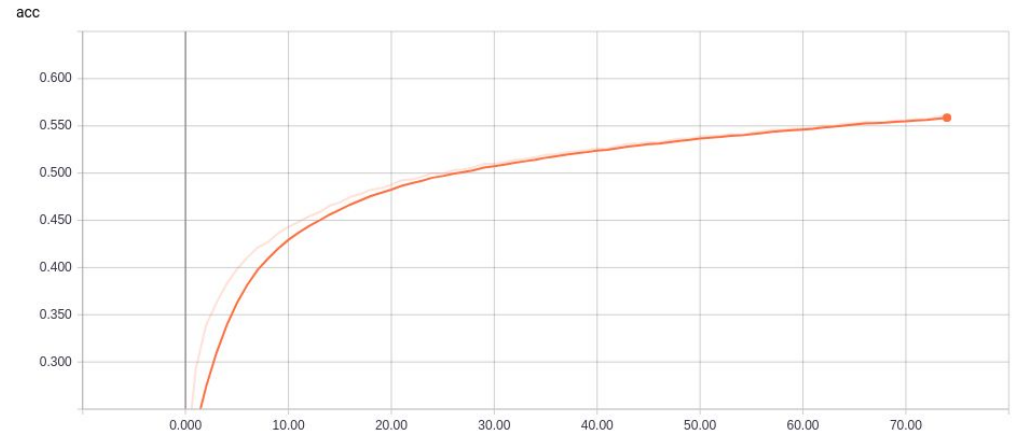


Gradient for kernel in first layer:

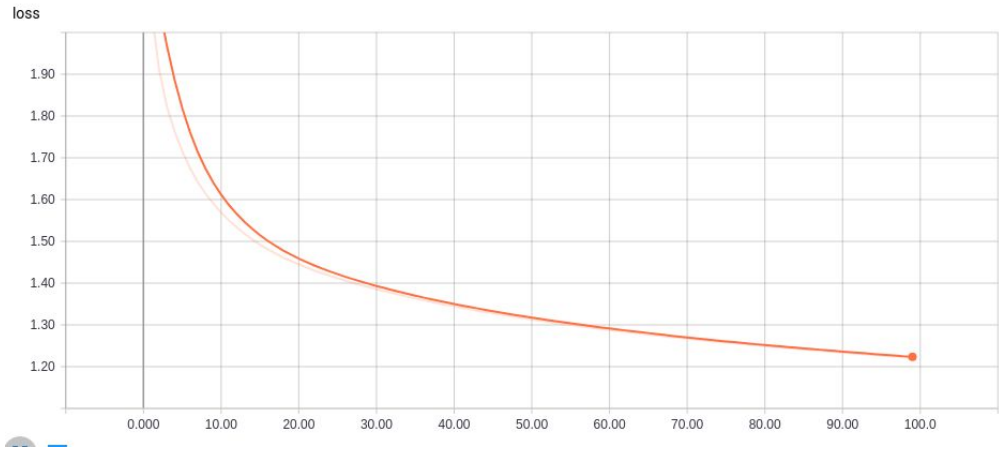


b. **Testing accuracy: 54.49%, Loss: 1.28**

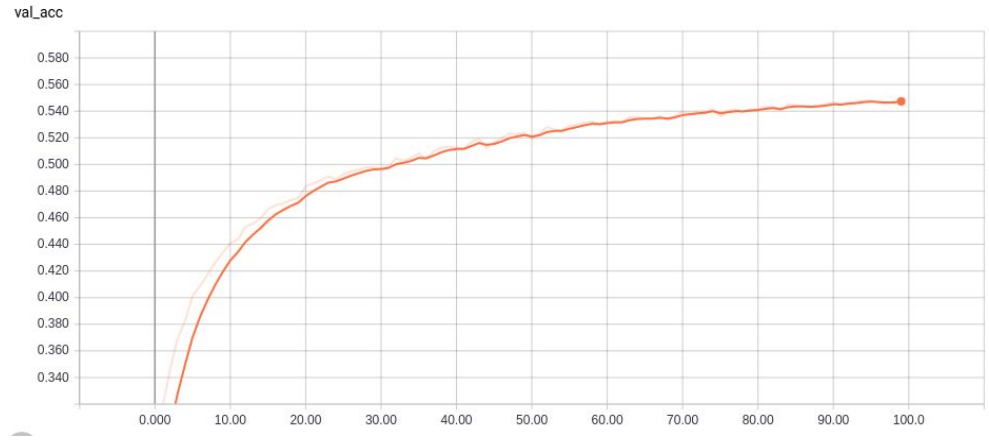
Training accuracy:



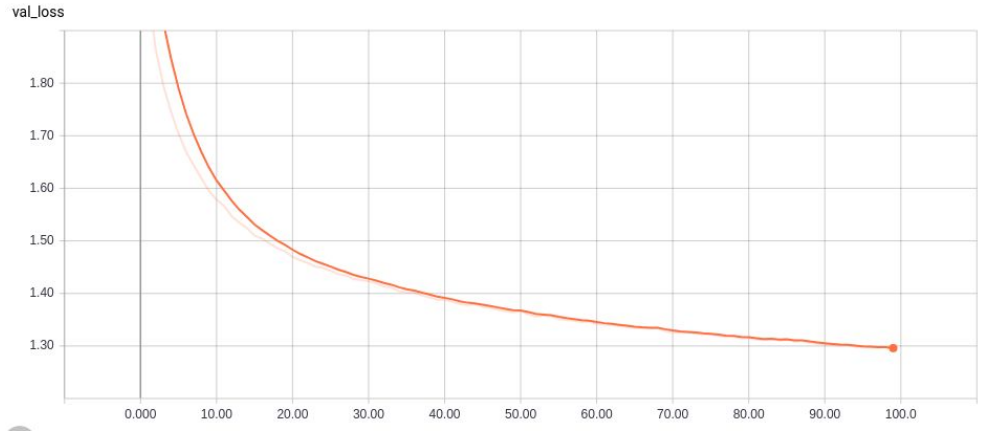
Training loss:



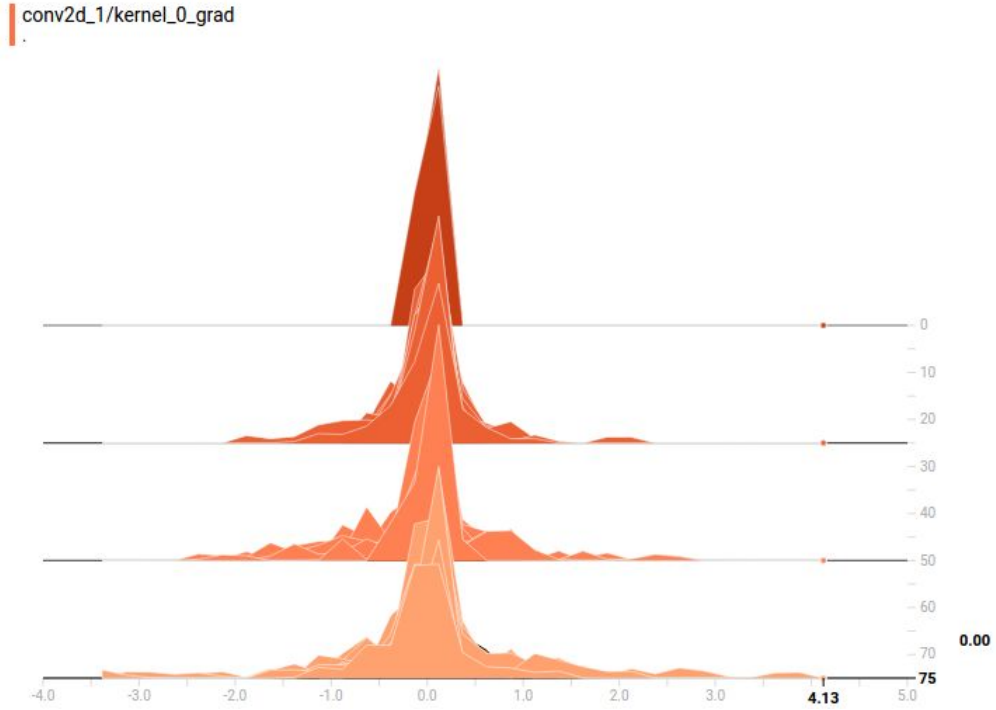
Validation accuracy:



Validation loss:

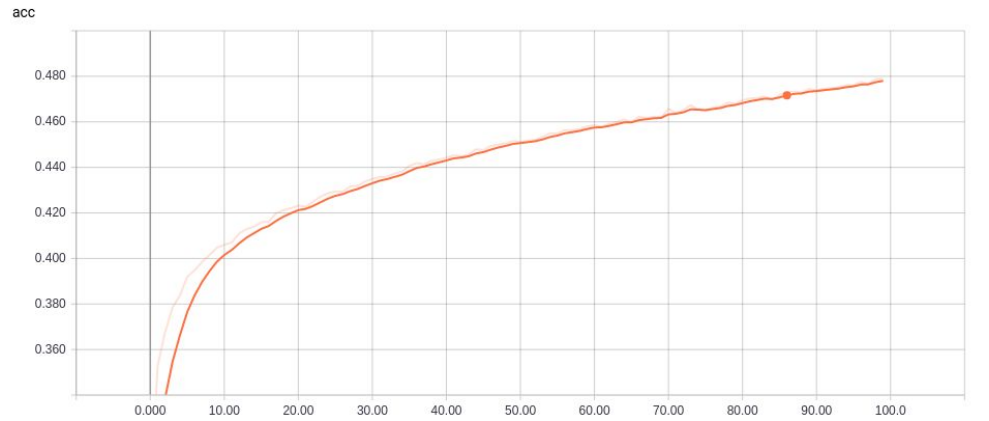


Gradient for kernel in first layer:

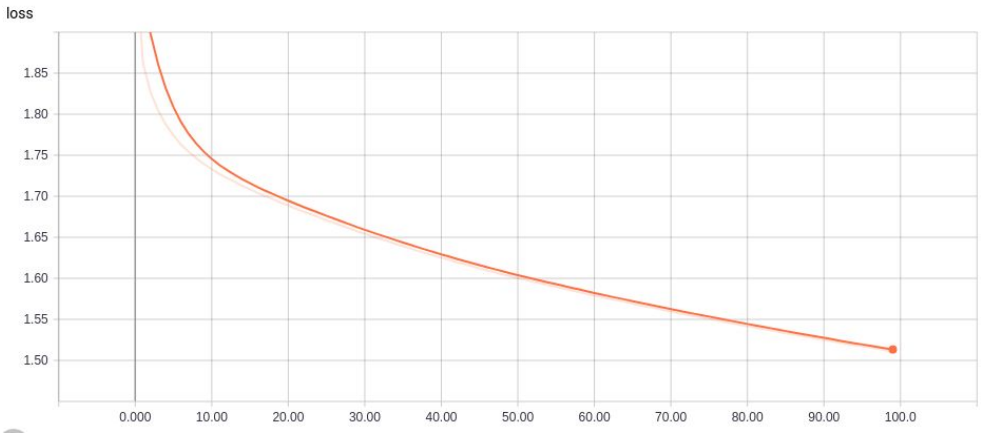


c. **Testing accuracy: 44.42%, Loss: 1.58**

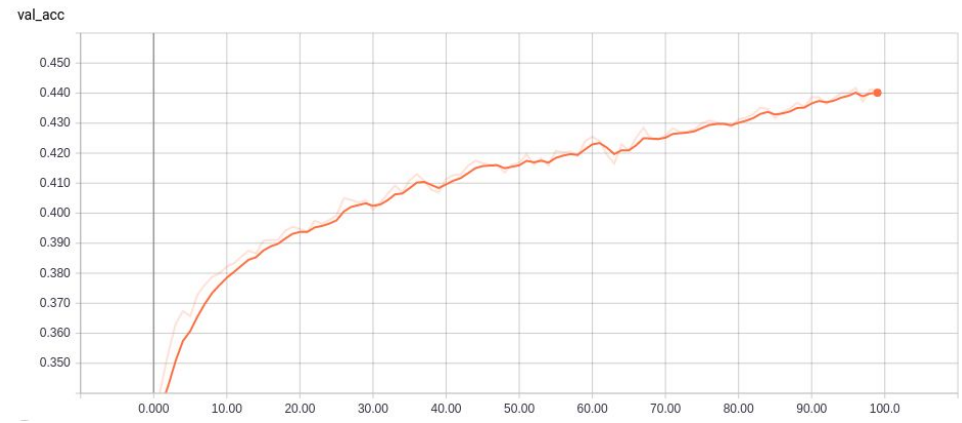
Training accuracy:



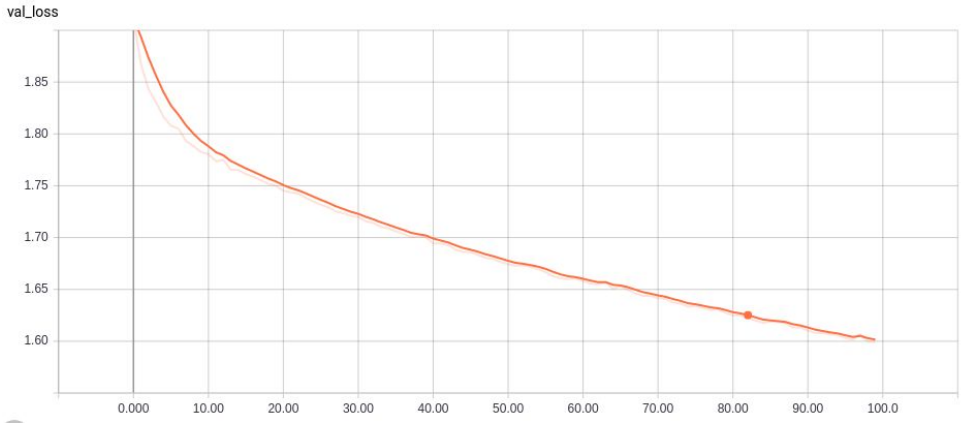
Training loss:



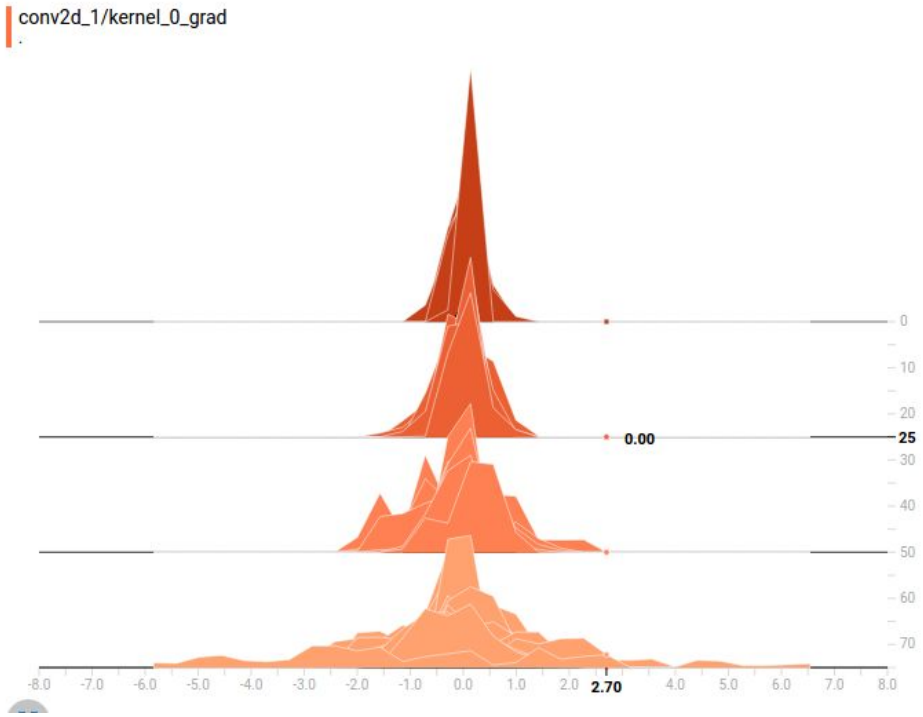
Validation accuracy:



Validation loss:

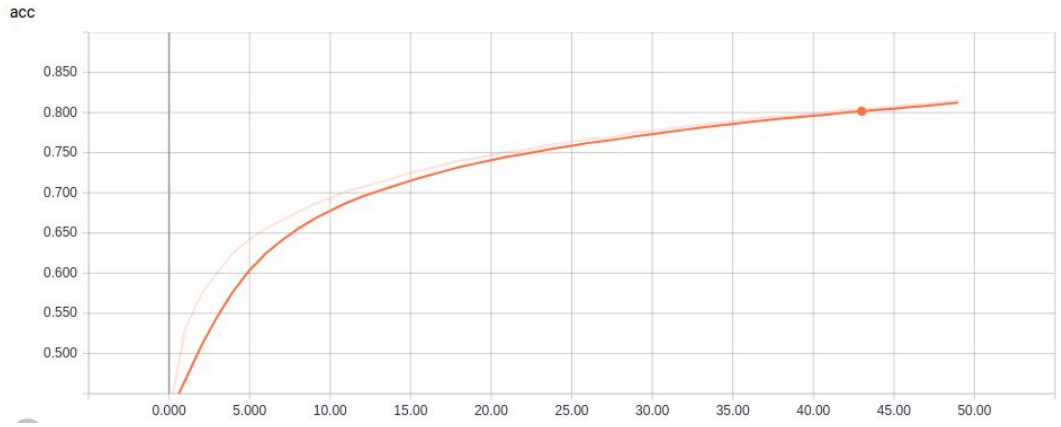


Gradient for kernel in first layer:

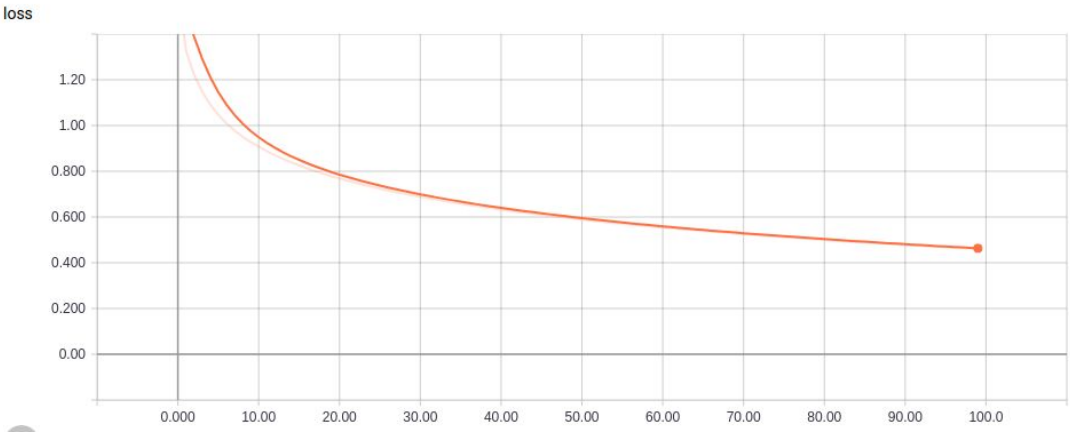


d. **Testing accuracy: 65.45%, Loss: 1.03**

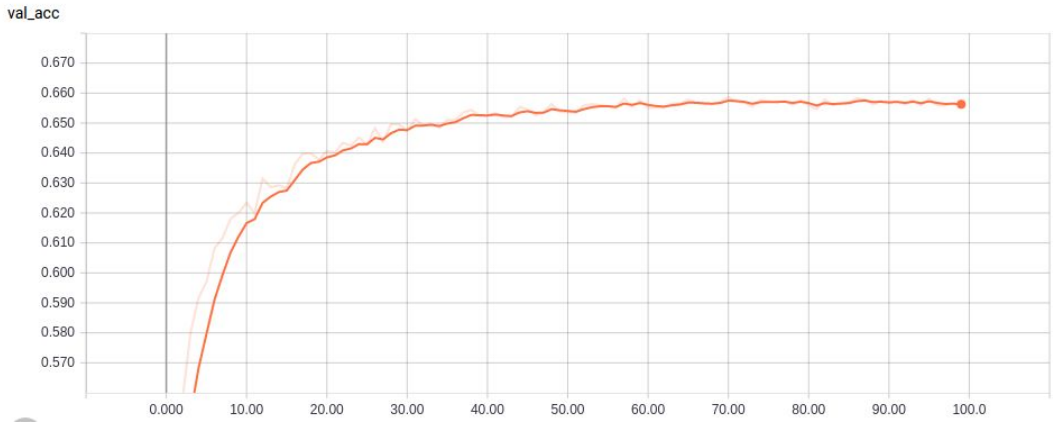
Training accuracy:



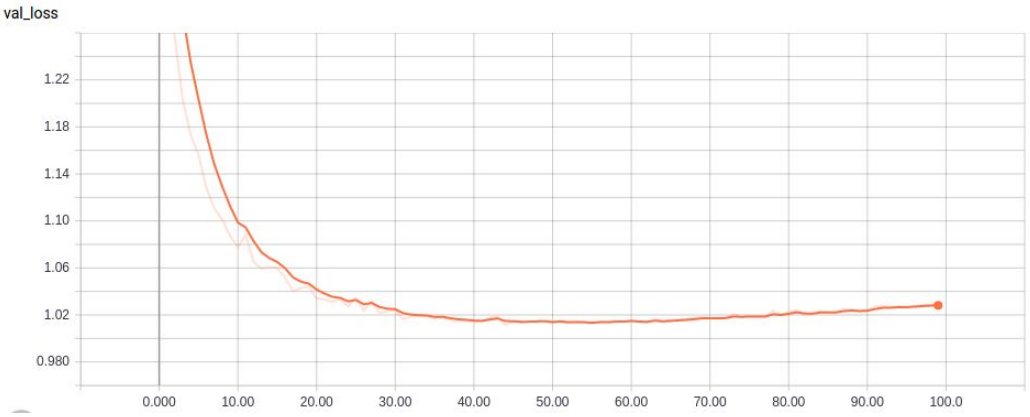
Training loss:



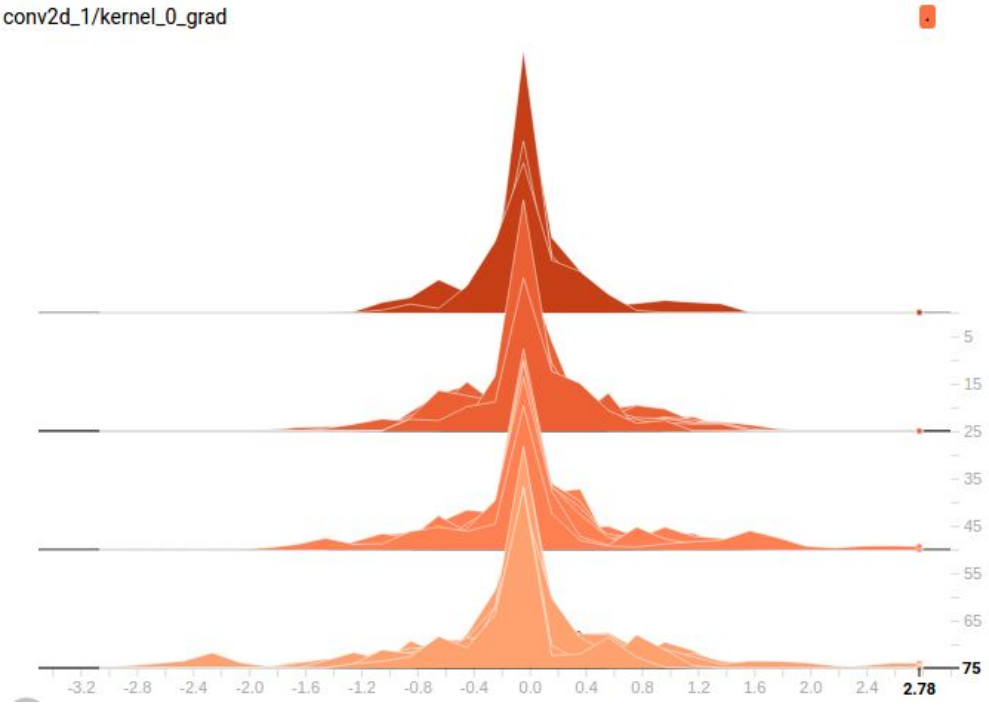
Validation accuracy:



Validation error:



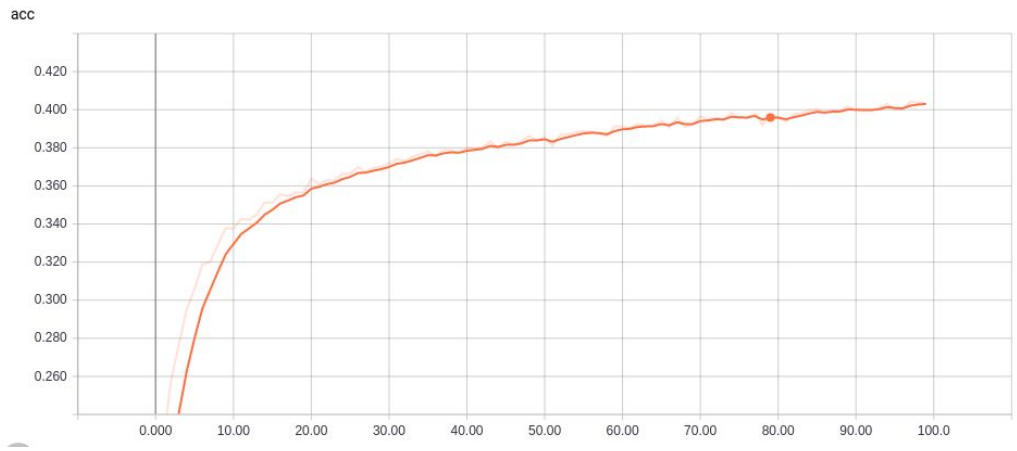
Gradient for kernel in first layer:



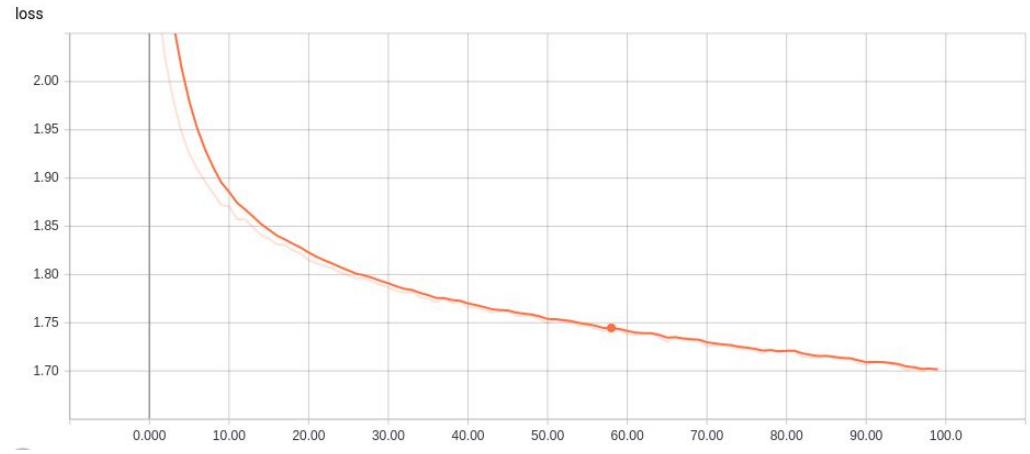
3. Using dropout is usually associated with helping against overfitting, which it does in the given case. However, an interesting observation is that the gradient diminishing problem seems to be slightly better, though far from solved. This can be attributed to the fact that because of fewer neurons contributing to the gradient while back propagating, some of the gradients which are closer to zero may be skipped out, leading to better gradients.

    a. **Testing accuracy: 41.94%, Loss: 1.67**

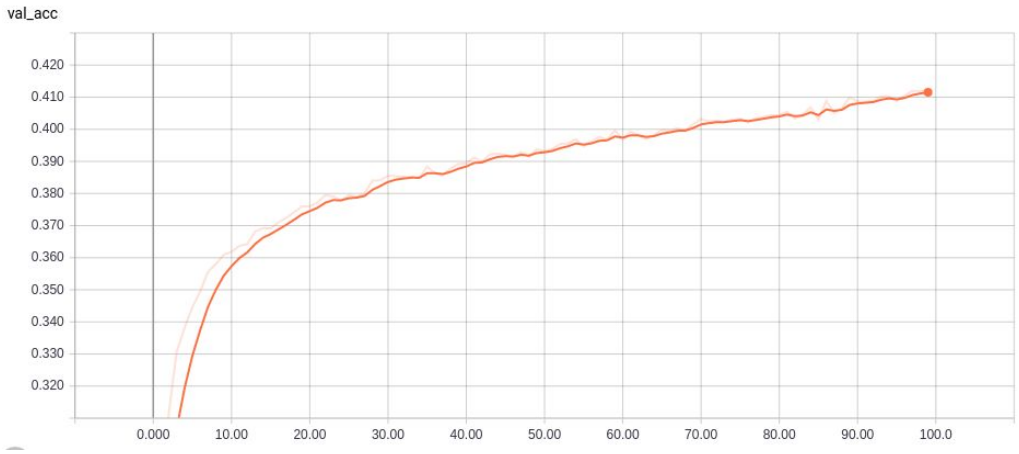
    Training accuracy:



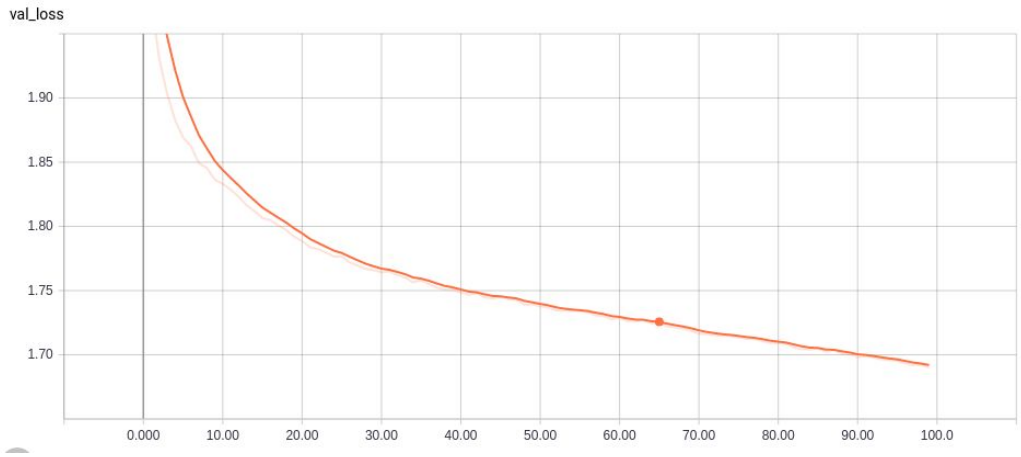
Training loss:



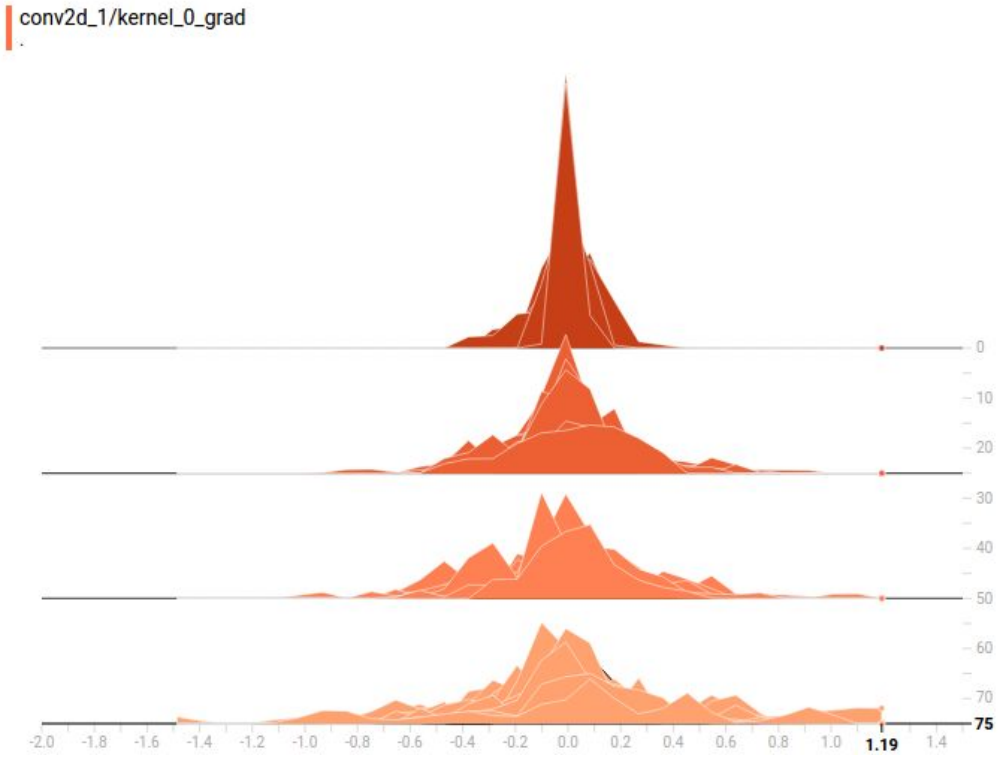
Validation accuracy:



Validation loss:

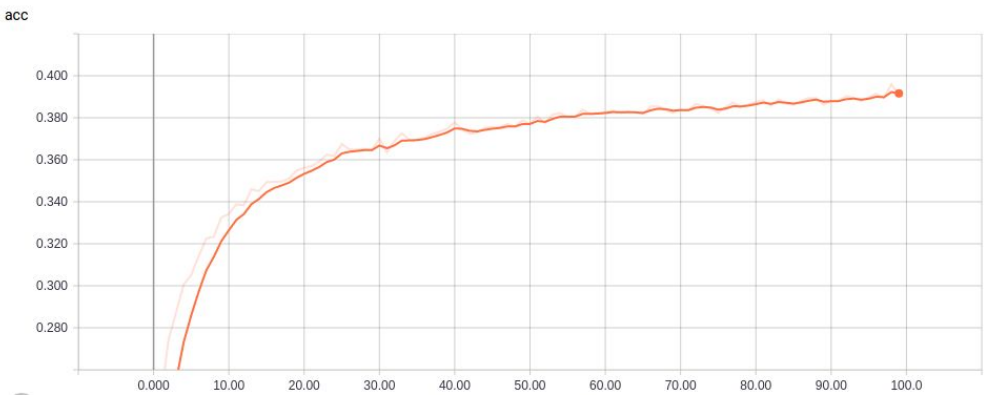


Gradient for kernel in first layer:

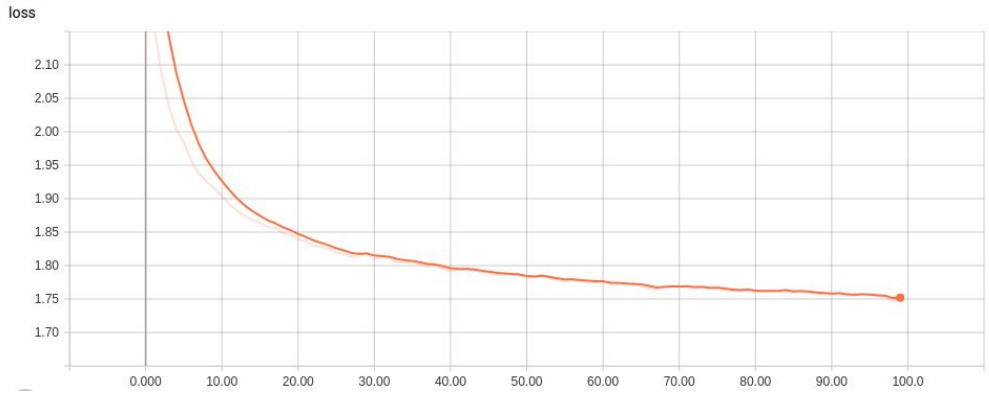


b. **Testing accuracy: 41.36%, Loss: 1.7**

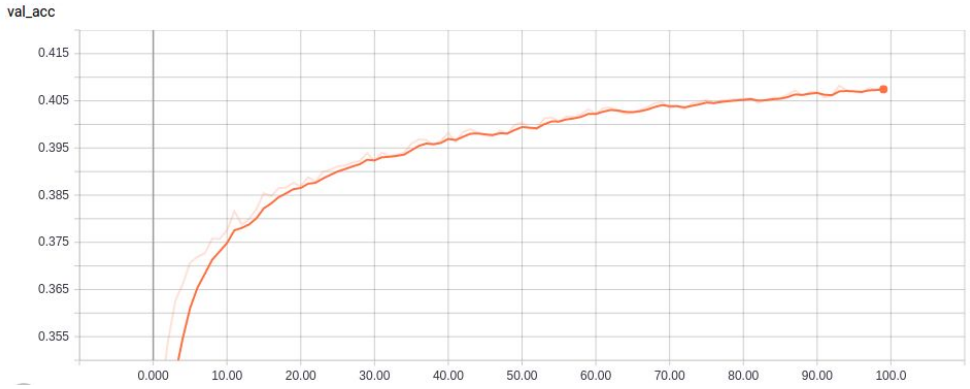
Training accuracy:



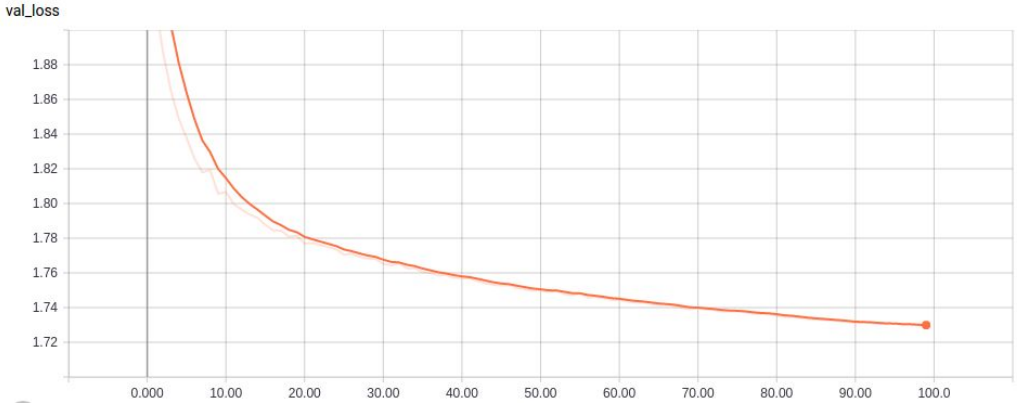
Training loss:



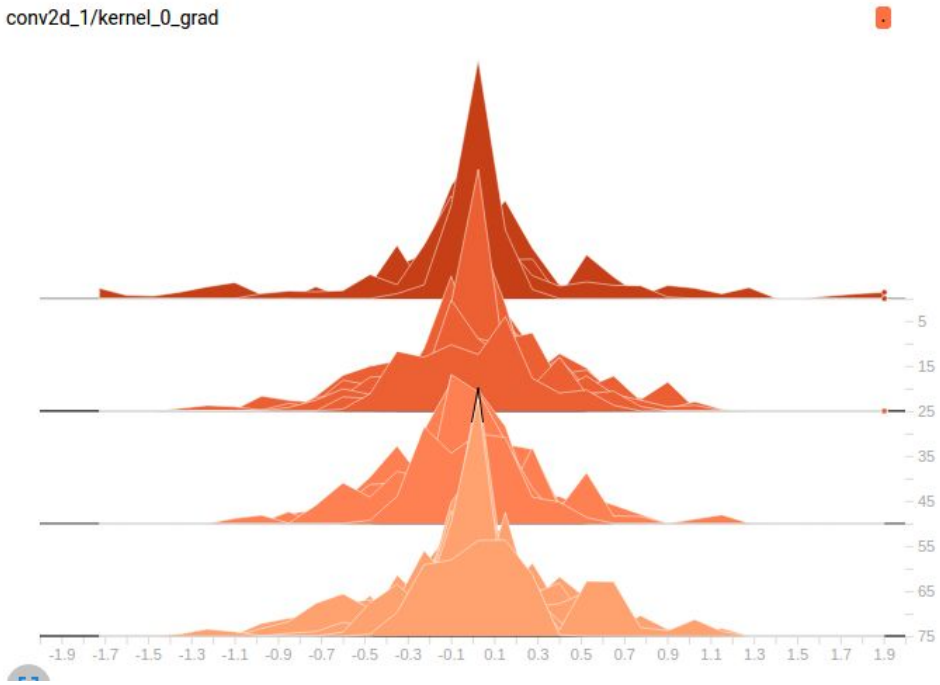
Validation accuracy:



Validation loss:

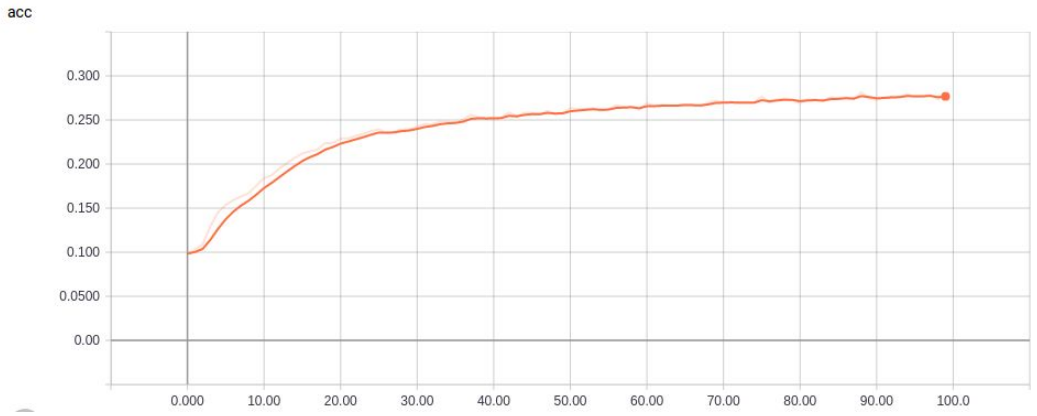


Gradient for kernel in first layer:

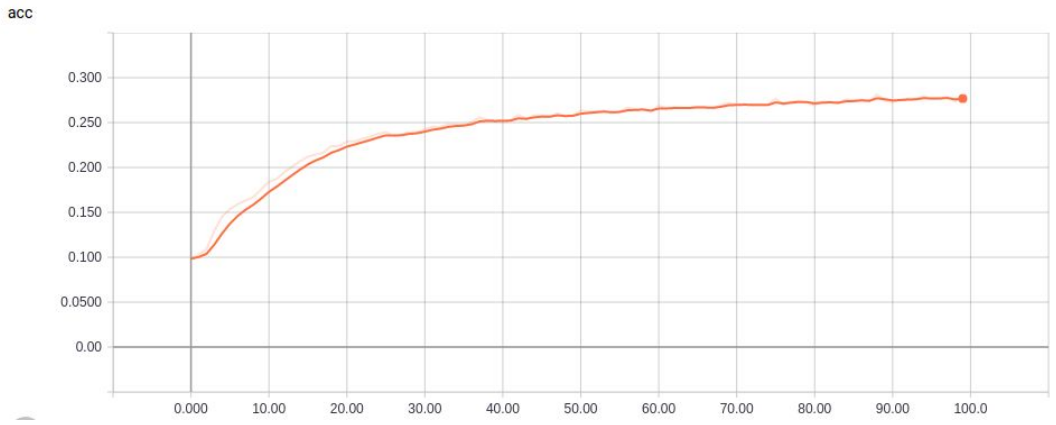


c. **Testing accuracy:36.46%, Loss: 1.94**

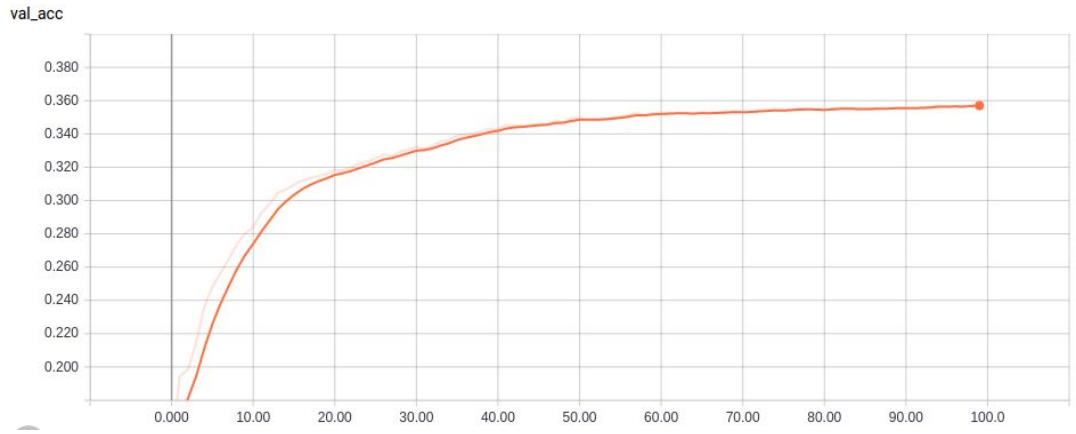
Training accuracy:



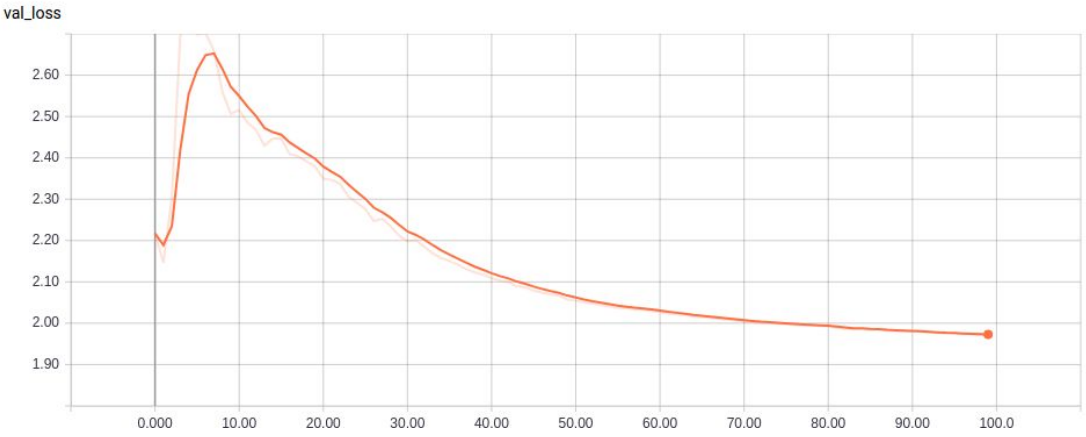
Training loss:



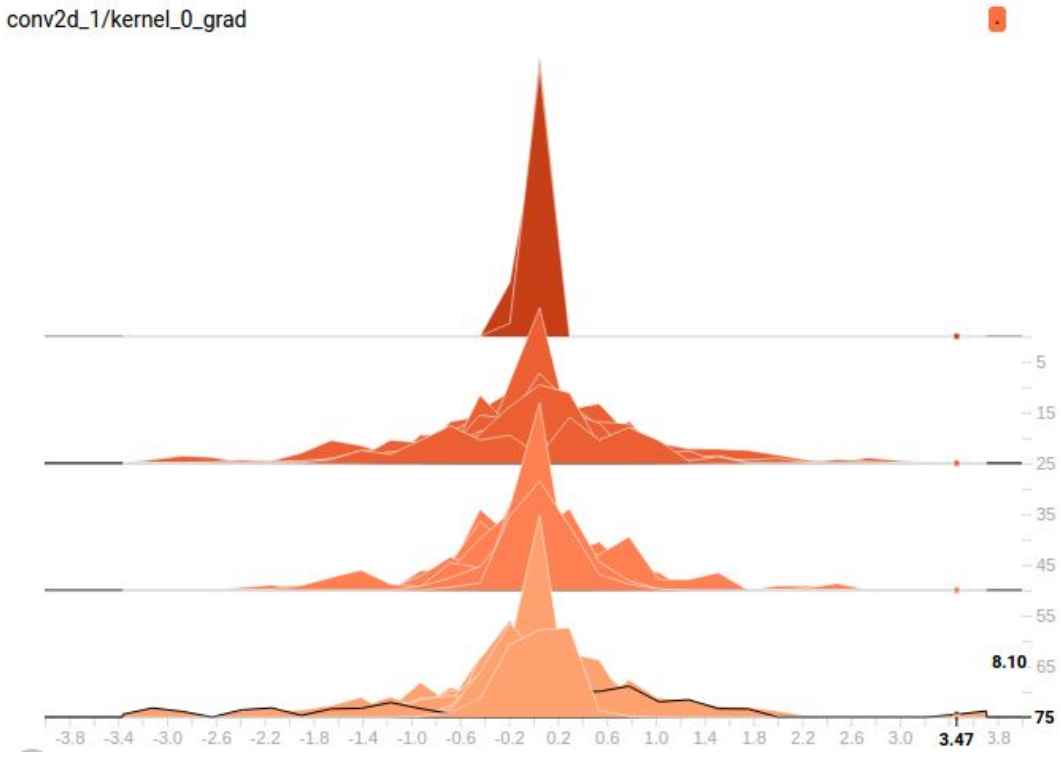
Validation accuracy:



Validation loss:



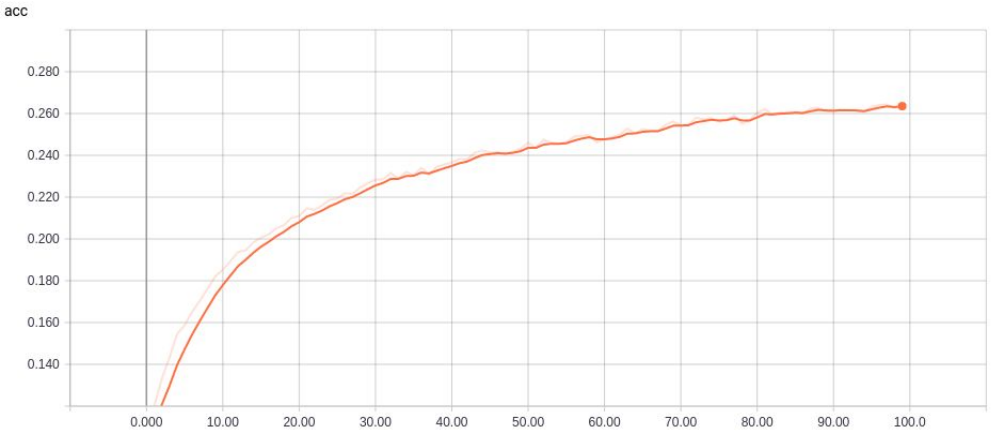
Gradient for kernel in first layer:



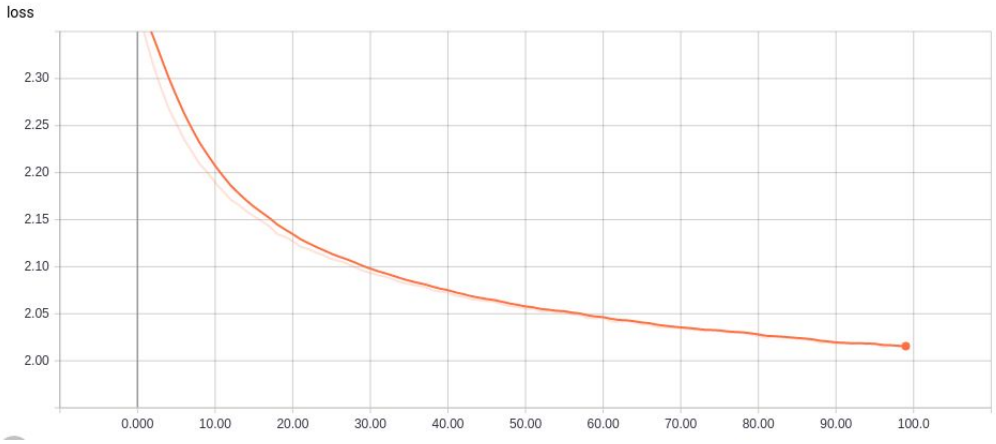
4. Increasing the batch size makes the model horribly worse. This can be explained by the fact that the batch size and thus cannot capture variance in data, leading to too much of generalization. Because of this, the gradients are not very meaningful. This, however, should not be interpreted as a possible remedy for the gradient pronlem.

    a. **Testing accuracy: 16.7%, Loss: 2.26**

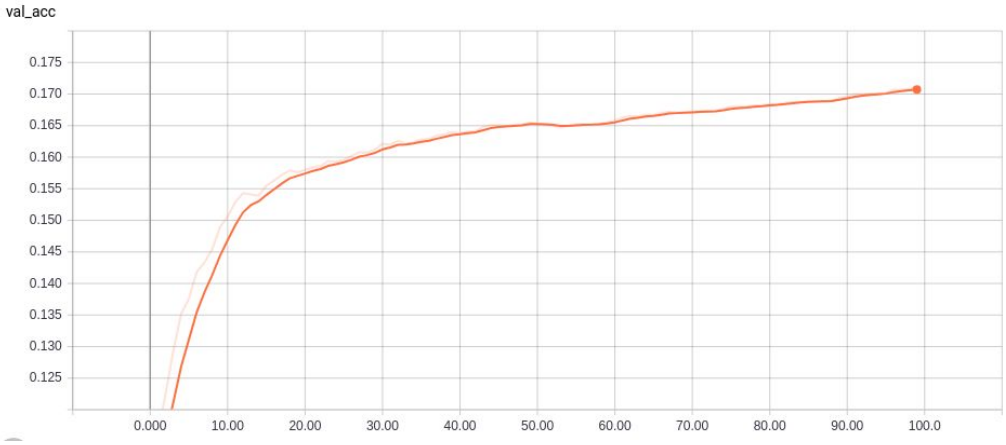
    Training accuracy:



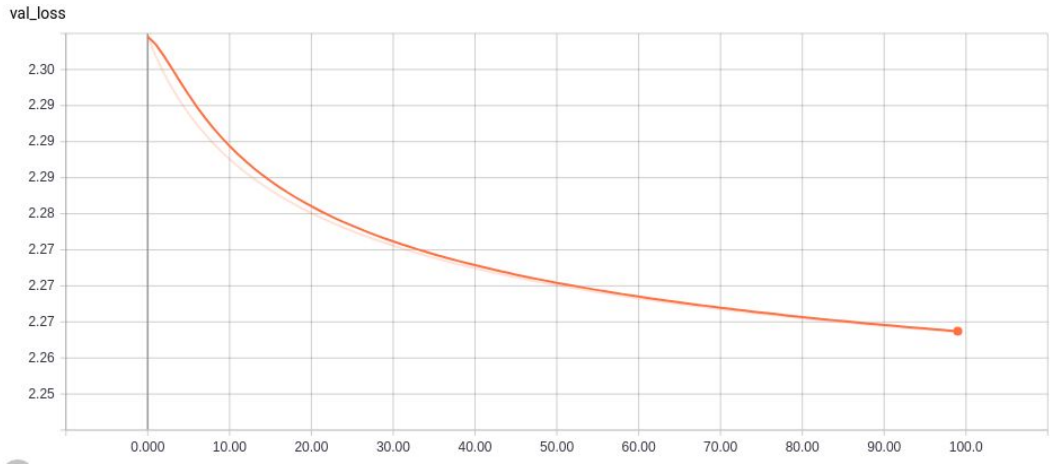
    Training loss:



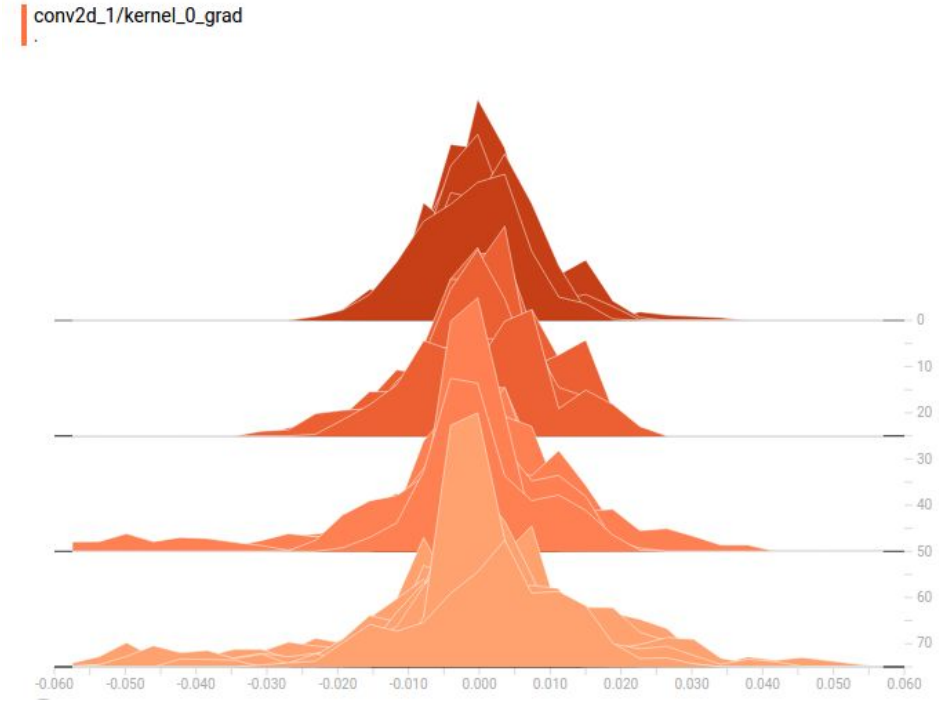
    Validation accuracy:



    Validation loss:

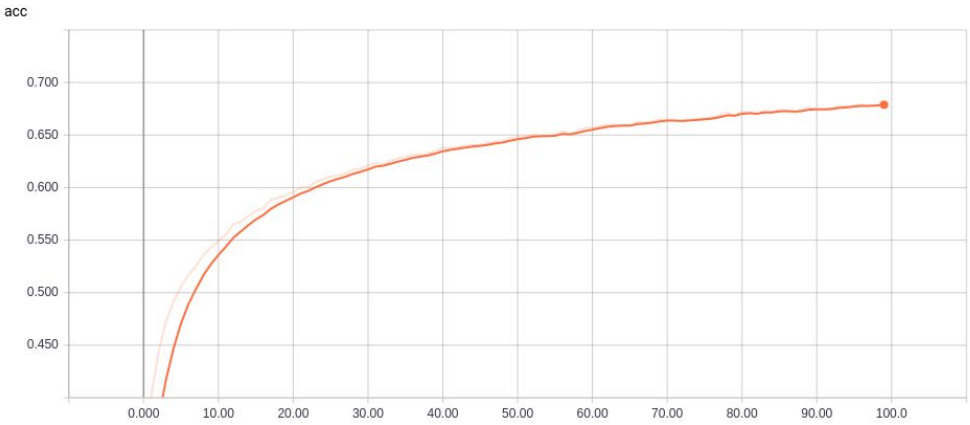


Gradient for kernel in first layer:

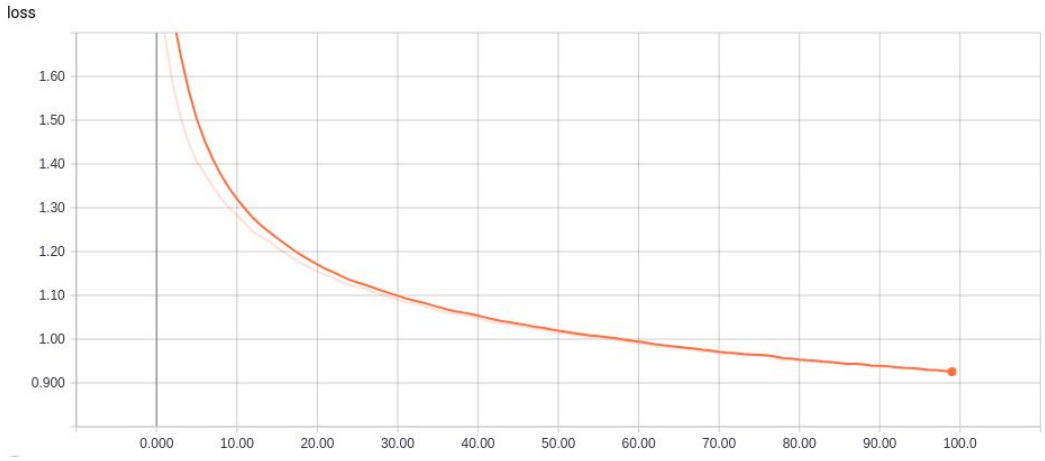


b. **Testing accuracy: 10%, Loss 3.41**

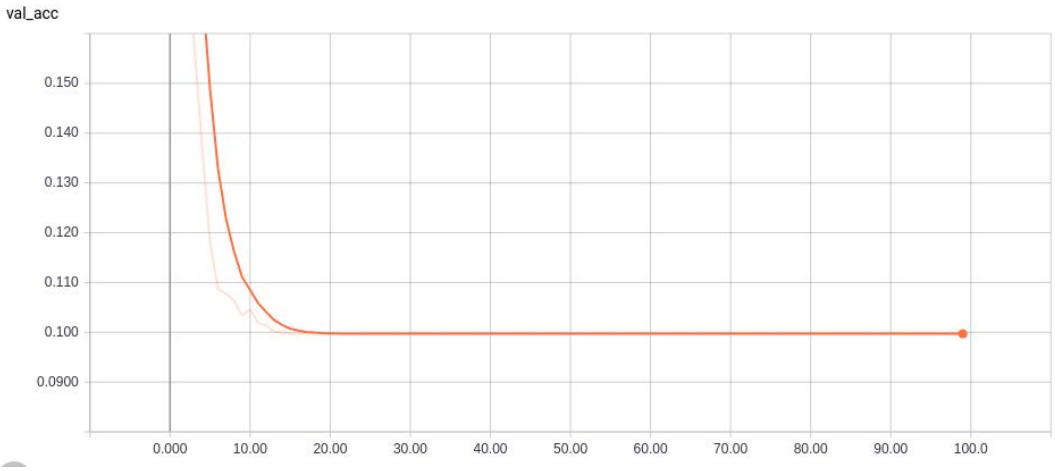
Training accuracy:



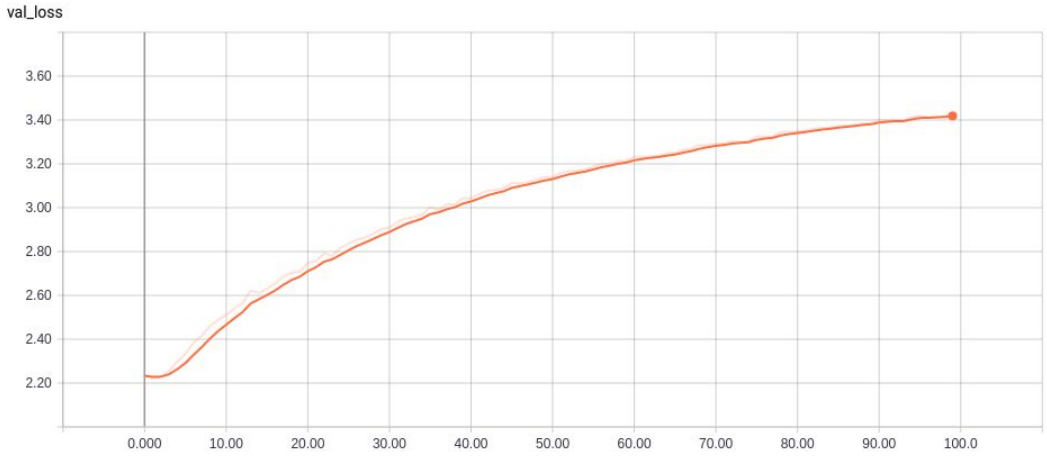
Training loss:



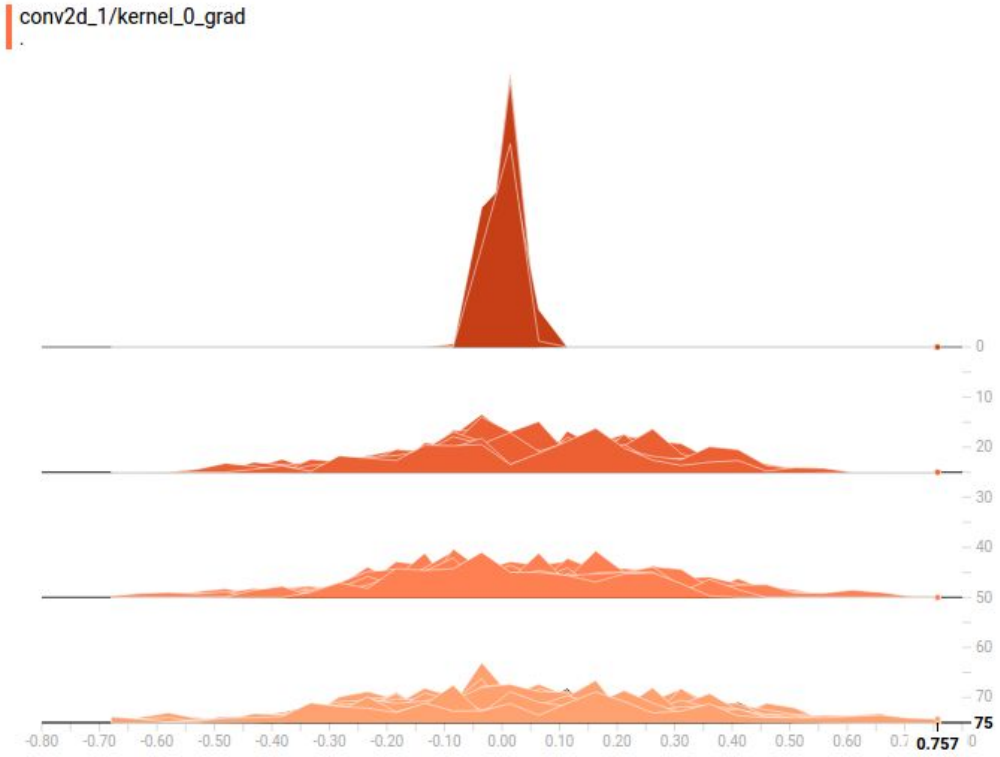
Validation accuracy:



Validation loss:

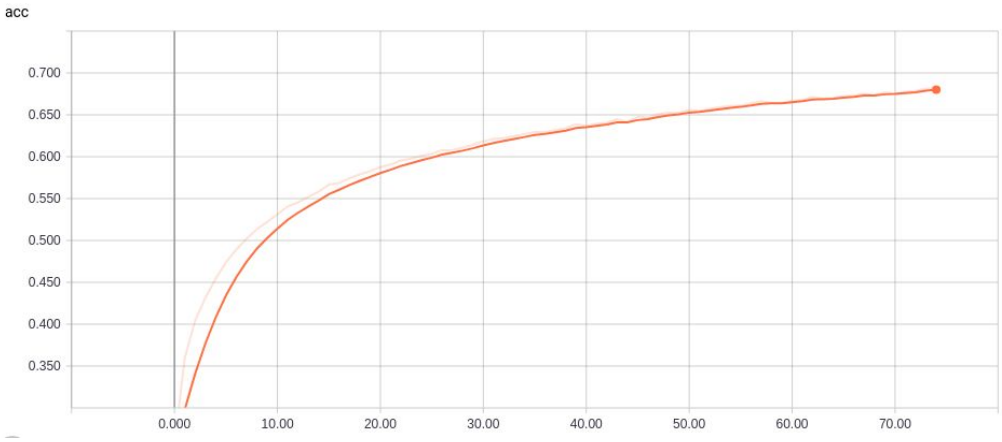


Gradient for kernel in first layer:

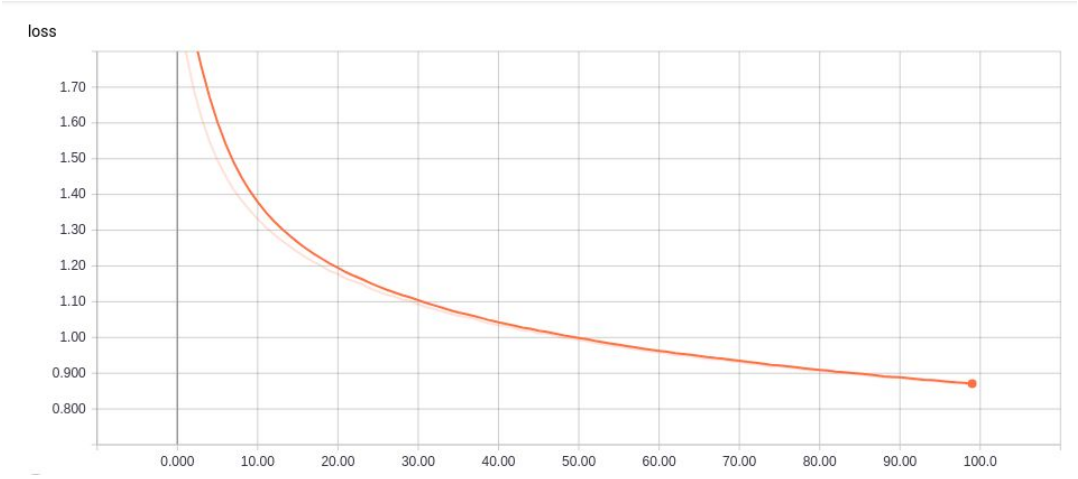


c. **Testing accuracy: 10%, Loss: 3.57**

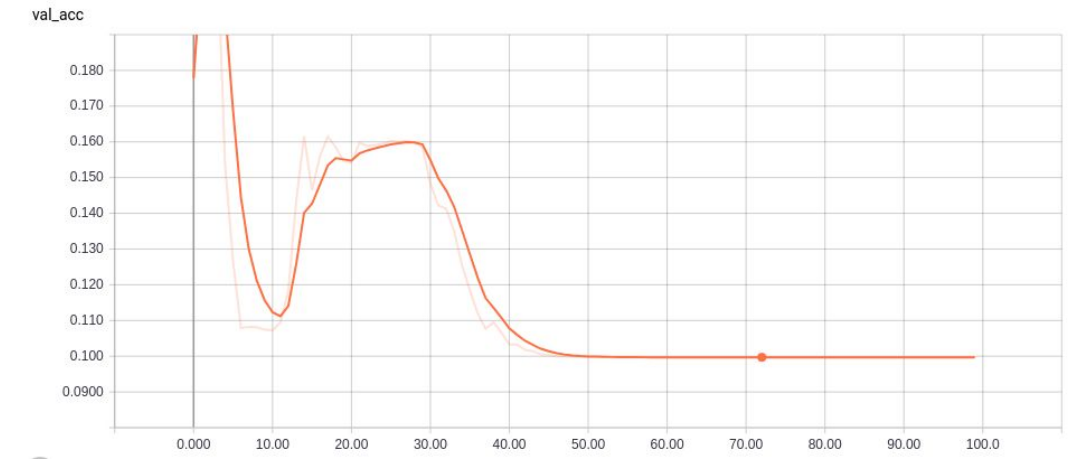
Training accuracy:



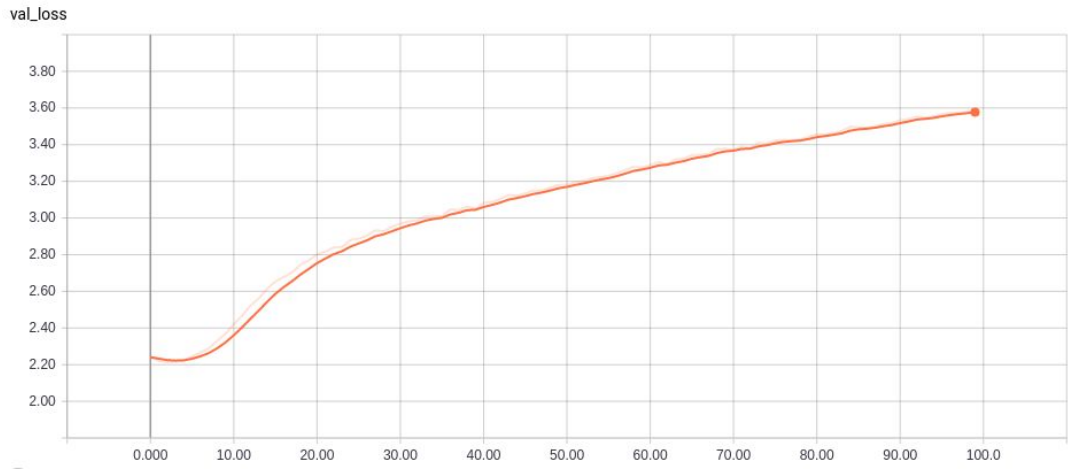
Training loss:



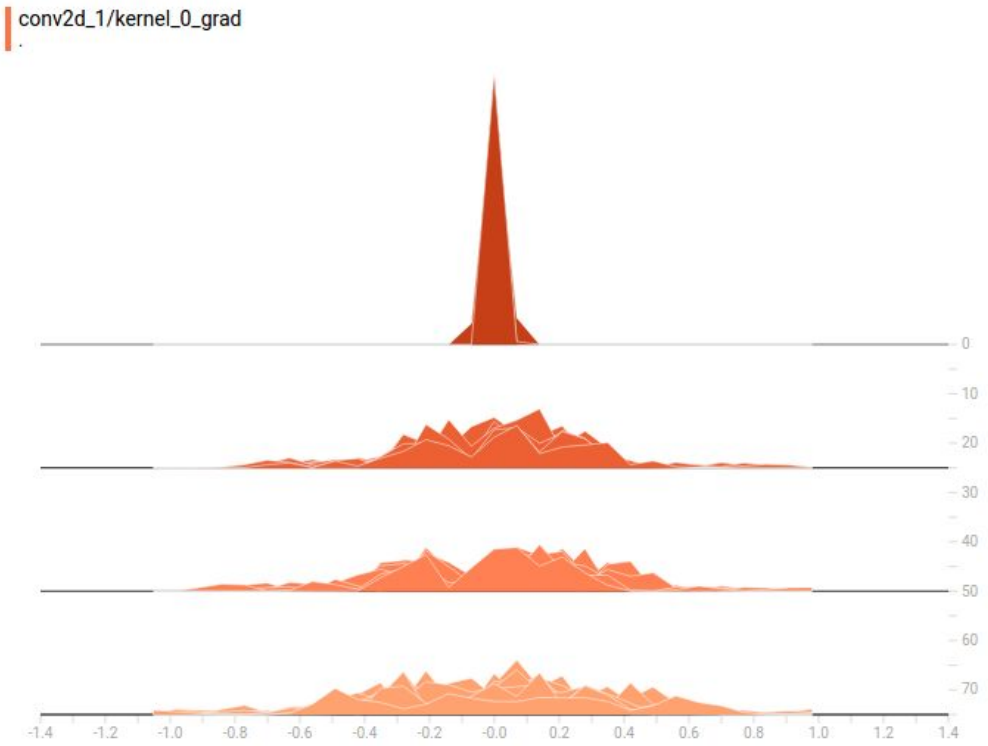
Validation accuracy:



Validation loss:

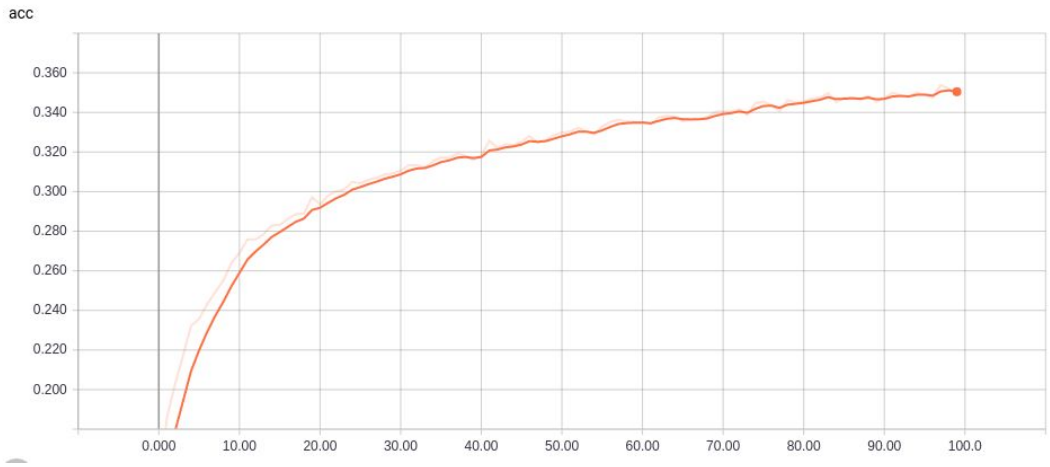


Gradient for kernel in first layer:

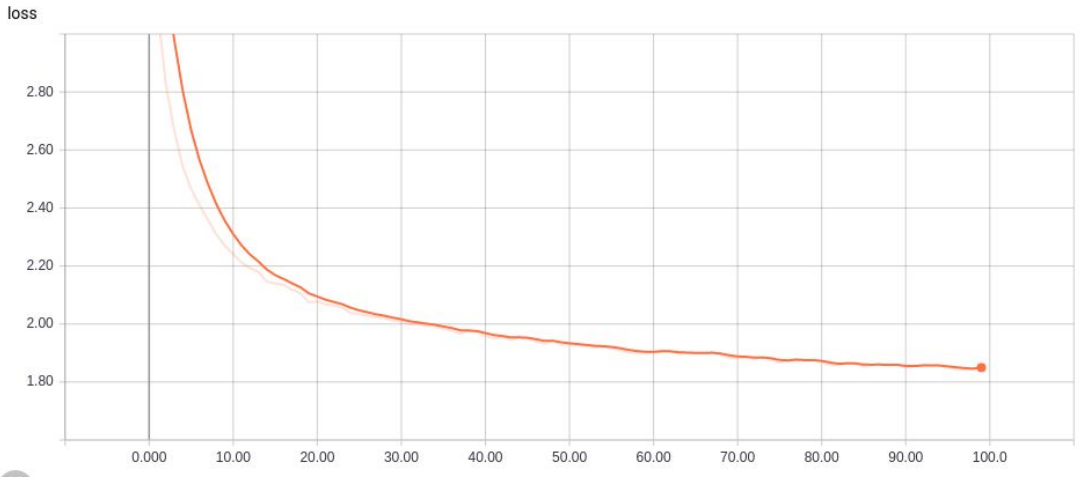


5. Using a ResNet (as expected) solves the gradient problem. The projection block solves is way more elegantly, with the gradients spread out smoothly. The losses are converging for both testing and training, and thus would finally get better if run for more iterations.
    a. **Testing accuracy: 29.68%, Loss 1.99**

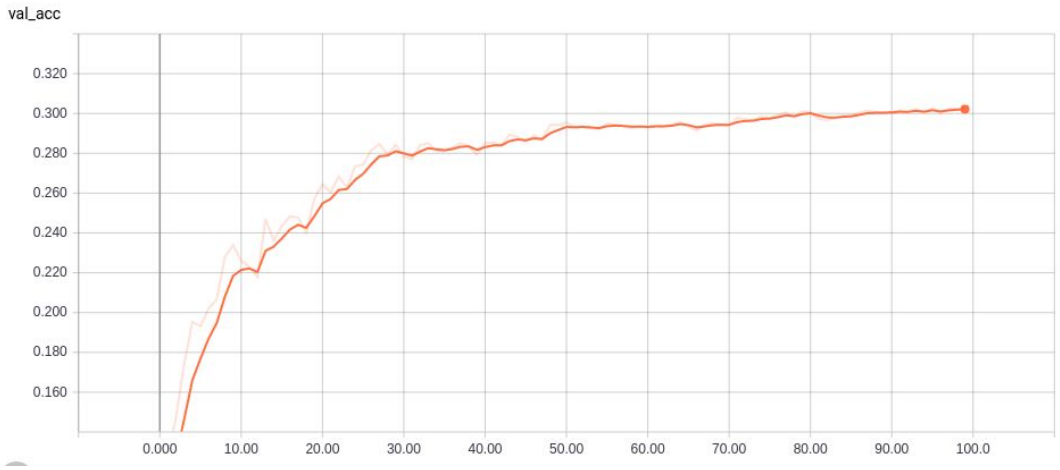
    Training accuracy:



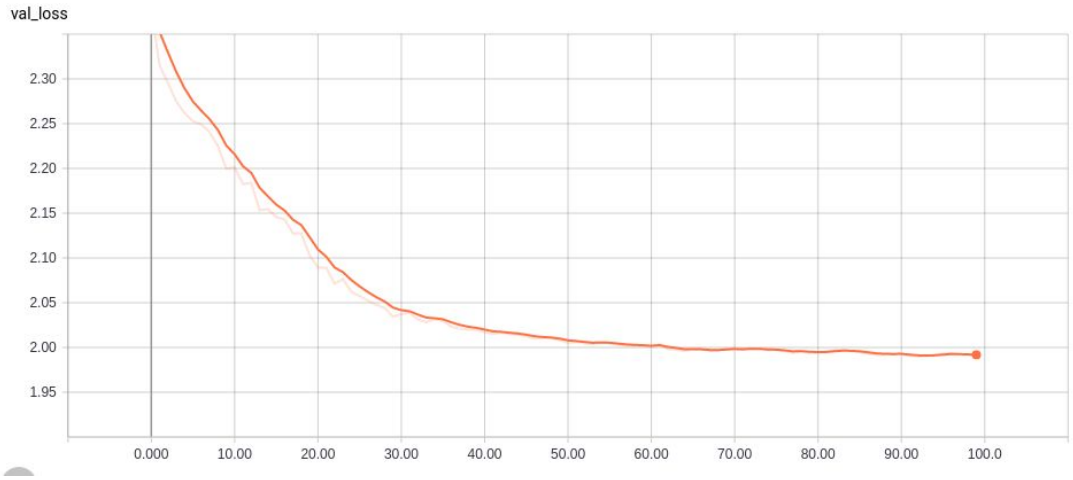
    Training loss:



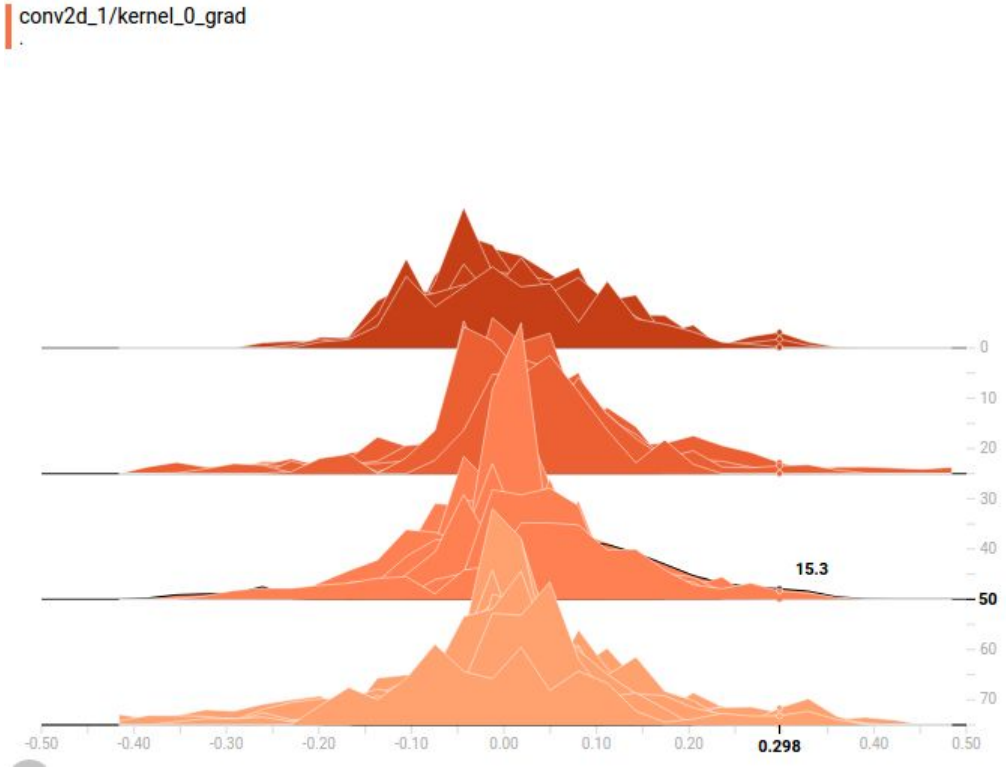
Validation accuracy:



Validation loss:

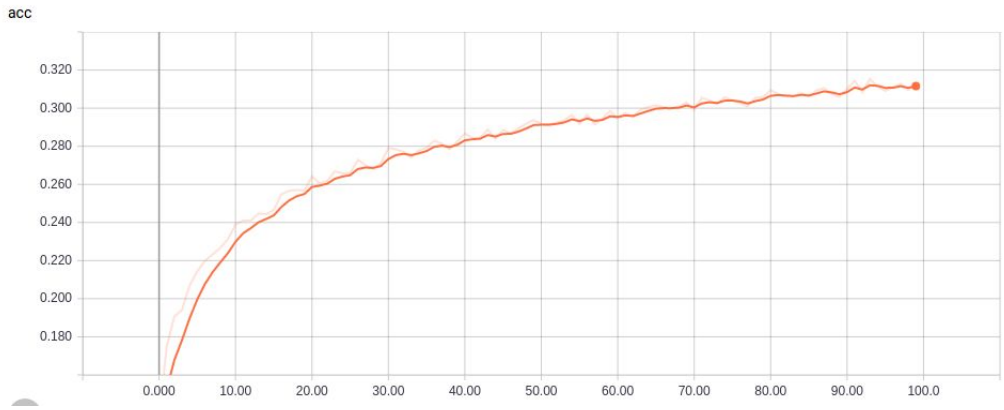


Gradient for kernel in first layer:

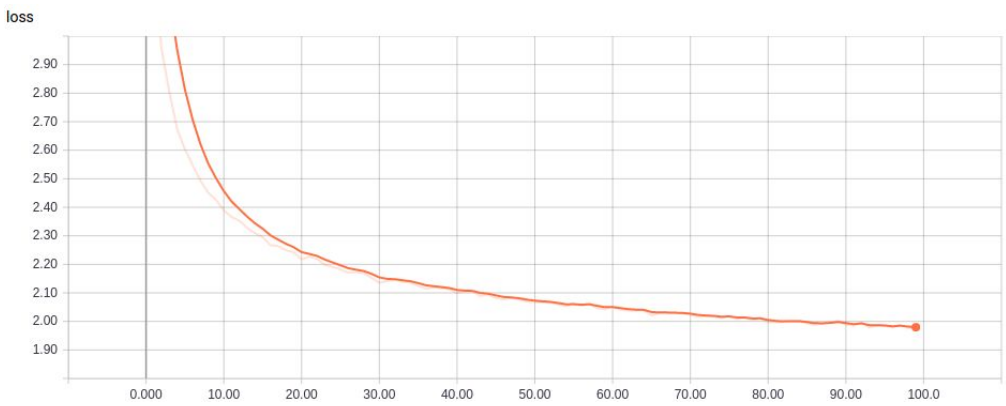


b. **Testing accuracy: 23.01%, Loss 2.04**

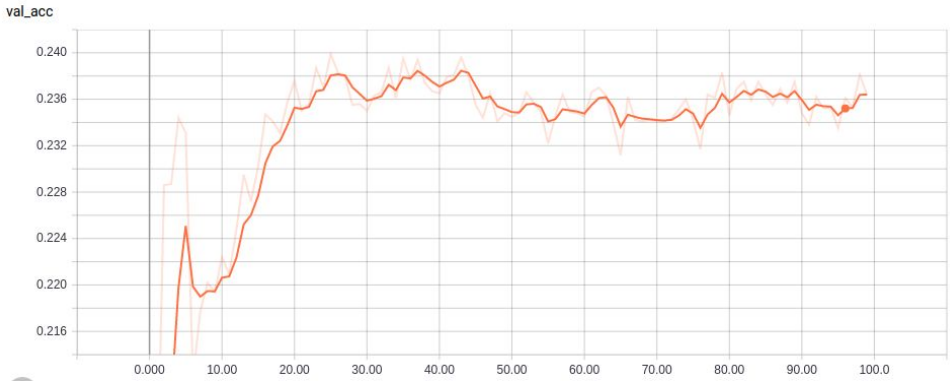
Training accuracy:



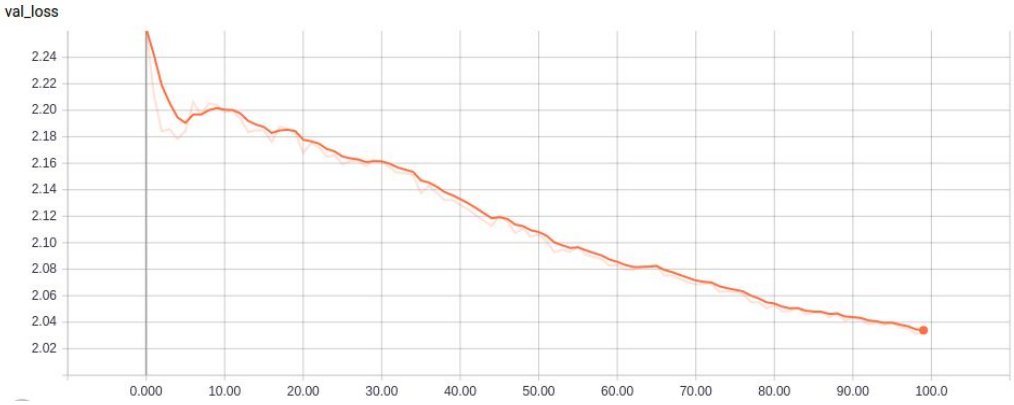
Training loss:



Validation accuracy:



Validation loss:



Gradient for kernel in first layer: