Visual Learning and Recognition

Assignment 1

Luka Eerens
lte@andrew.cmu.edu

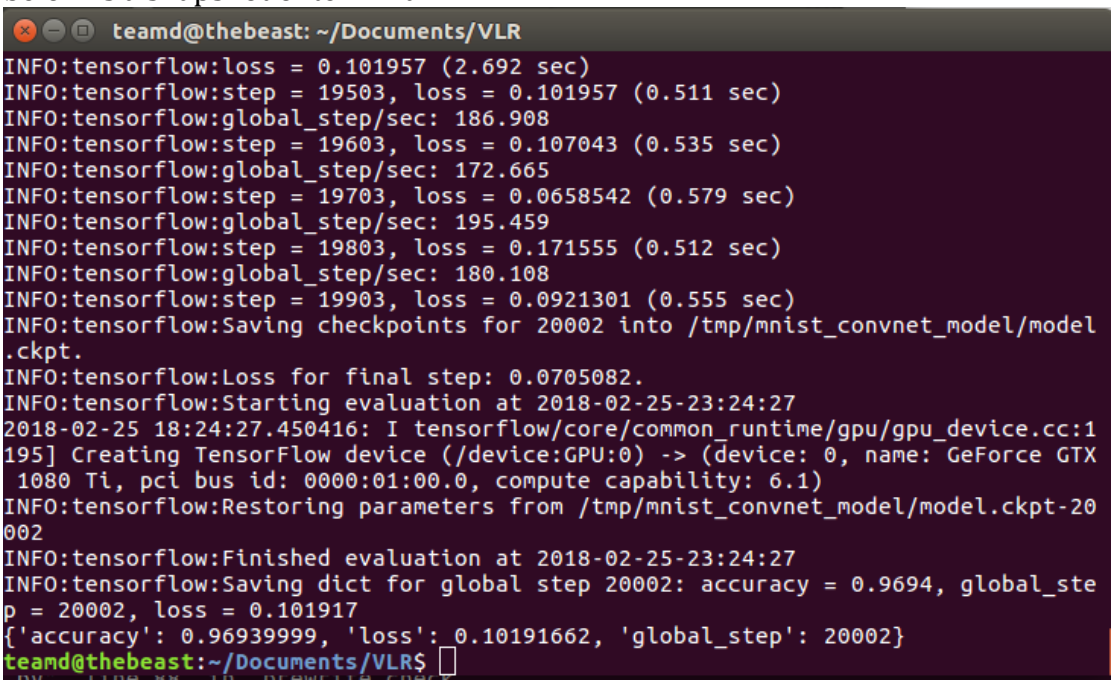
Masters in Robotics System Development (MRSD)

Friday 2nd of March

Note:
Every screenshot from TensorBoard will be also provided as a jpg image with this submission in case you would like to see it in greater detail.

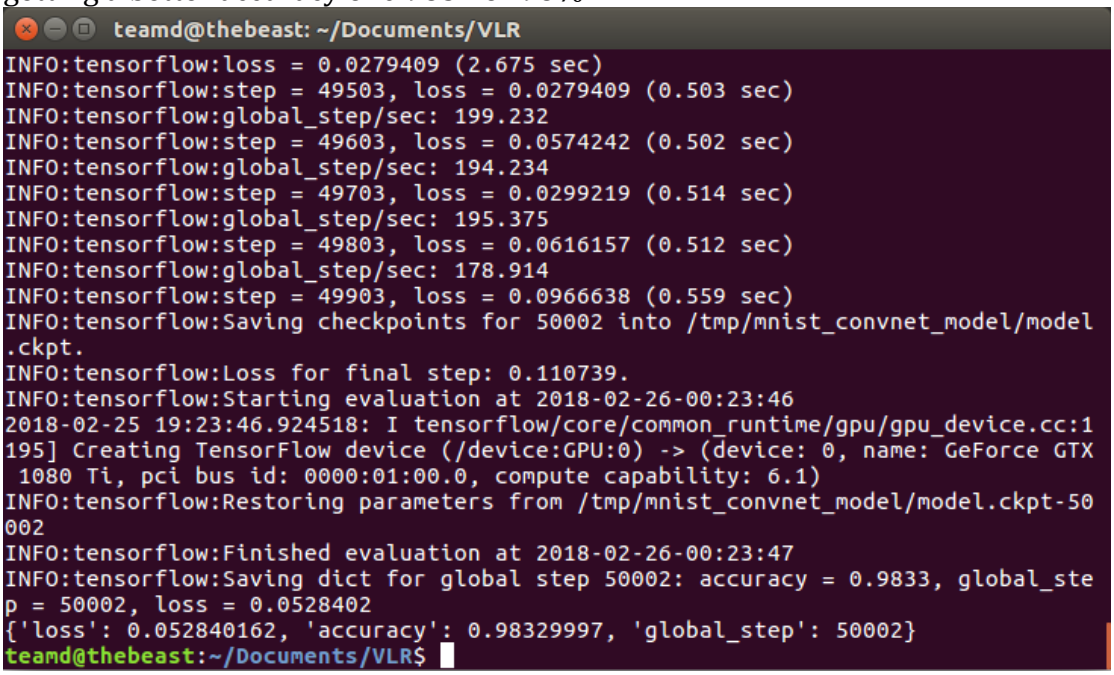
Task 0: MNIST 10-digit classification in TensorFlow (5 points)

Q 0. 1: What test accuracy does your model get:
With 20,000 iterations, I end up getting an accuracy of 0.9694 or around 97%, below is a snapshot of terminal.



There is no secret sauce to this one, just run the code from terminal making sure that it shares a directory with the folder that contains the MNIST dataset (how it was given for the assignment).

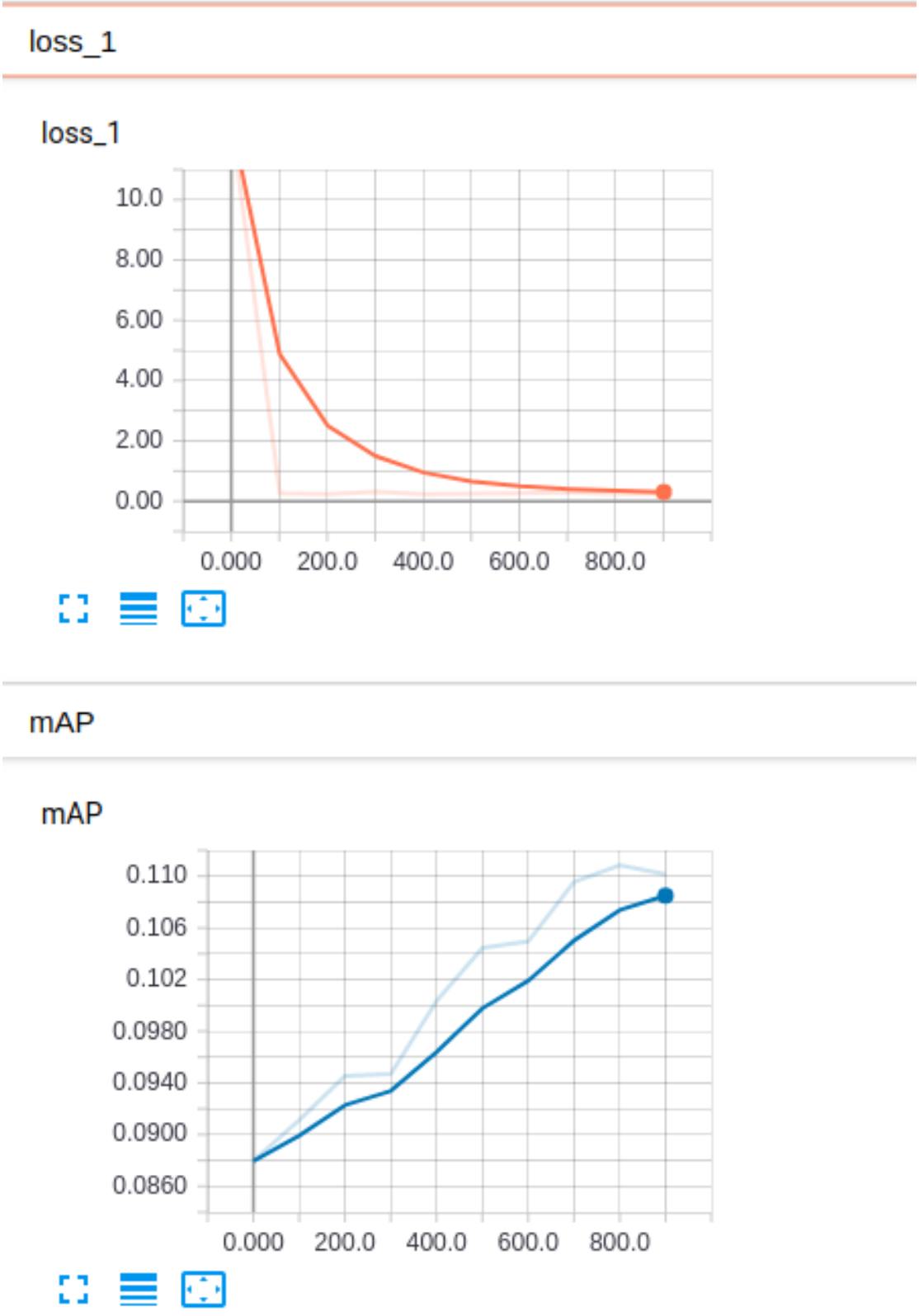
Q 0.2: What happens if you train for more iterations (30000)?
With 30,000 iterations, and everything else kept the same as before, I end up getting a better accuracy of 0.9832 or 98%.



The same steps were followed as Q0.1 (python, terminal, directory, MNIST etc.)

Q 0.3: Make the training loss and validation accuracy curve. Show for at least 100 points between 0 to 20000 iterations.
Below is a chart that shows how the training loss

Q 1: Training Loss and Accuracy over 100 iterations:

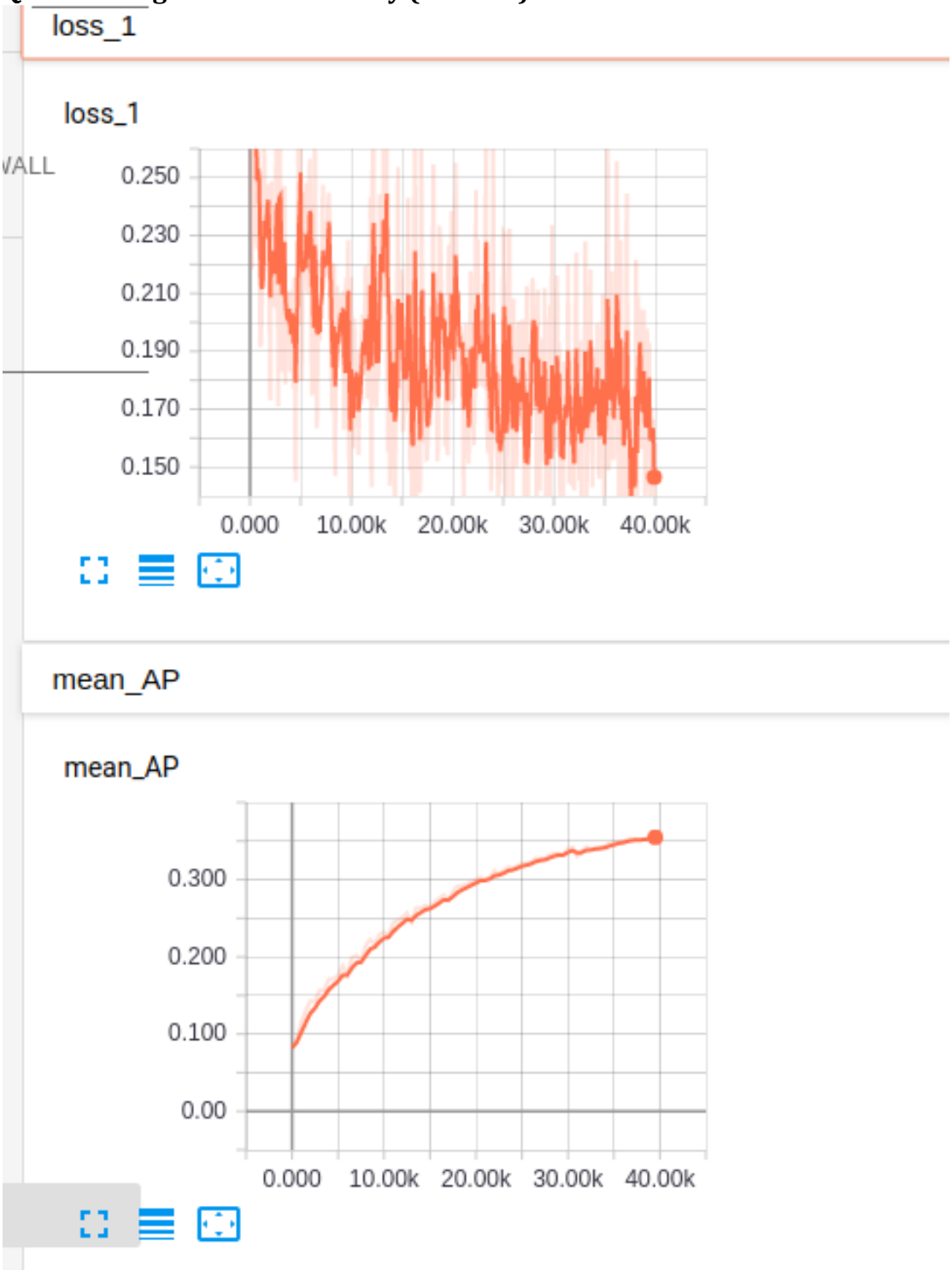


Below is the prediction accuracy for each class

```
Random AP: 0.0718976459016 mAP
GT AP: 1.0 mAP
Obtained 0.110157647033 mAP
per class:
aeroplane: 0.279898737696
bicycle: 0.0460165150591
bird: 0.073987870123
boat: 0.111638291394
bottle: 0.056779580701
bus: 0.0406457763007
car: 0.198953565432
cat: 0.104226071948
chair: 0.1389099761
cow: 0.0261122794875
diningtable: 0.0536206795886
dog: 0.146529656015
horse: 0.0649900030703
motorbike: 0.0401938143428
person: 0.52062205087
pottedplant: 0.048746246607
sheep: 0.0323976599533
sofa: 0.080271153297
train: 0.0692462837183
tvmonitor: 0.0693667289645
teamd@thebeast:~/hw1$
```

It is interesting that the there is such a diversity in the predictive power of the network but this is obviously because the dataset is heavily biased by having amount of images for cat, car, dog, and person labels and less for the other classifications.

Q 2: Training Loss and Accuracy (AlexNet):

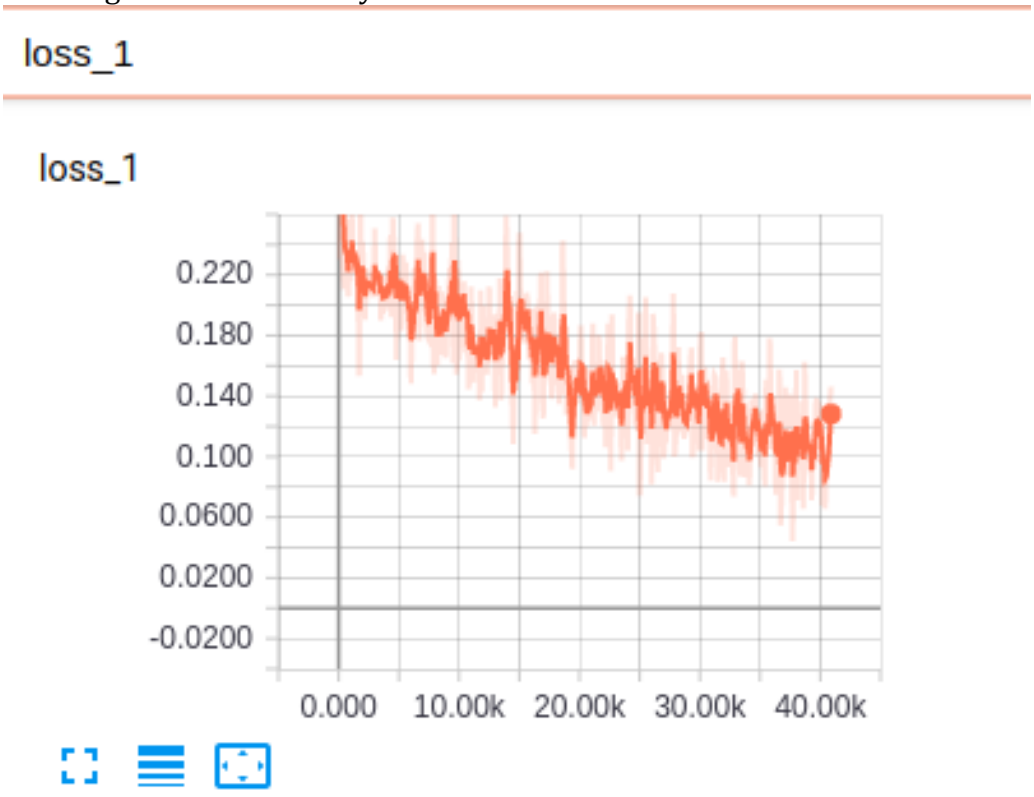


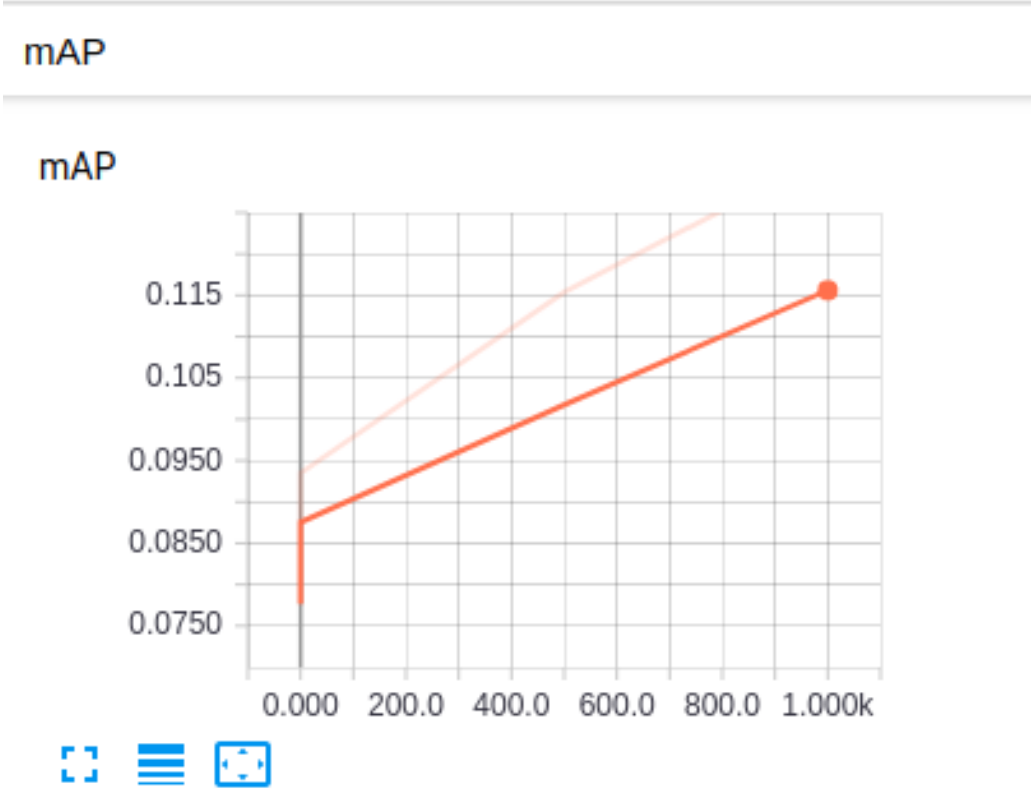
Interesting things to observe is how the loss is more zig-zaggy than that of the the previous problem. It shows that a bigger, deeper network means that the hypothesis function is higher dimensional which means that during back propagation, it will have to navigate through orders of magnitude more cliffs and valleys in higher dimensions.

The exponential learning rate decay also allows it to do more gradual gradient descent, and thus reduce the propensity to clear hypothesis function canyons or jumping out of these.

Q 3: VGG

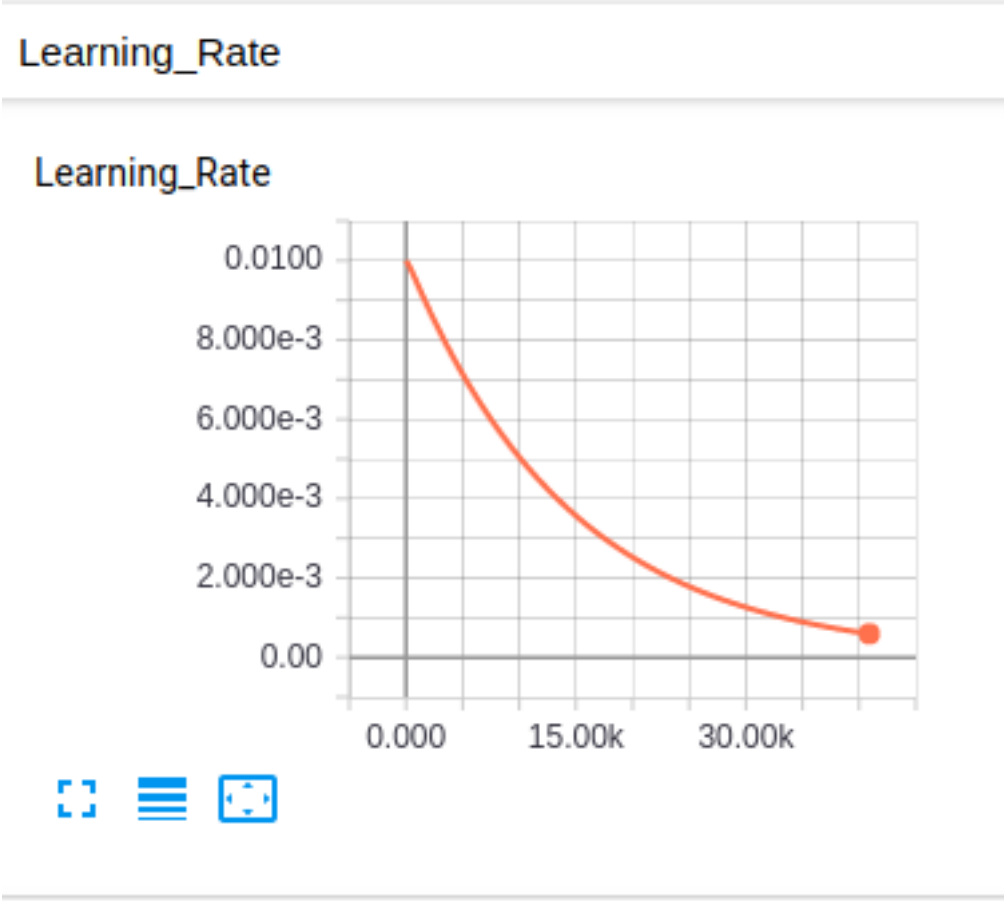
Training Loss and accuracy:





Here we are able to see, that a bigger deeper network is able to converge to a lower loss than a smaller one. This while achieve a high accuracy than the small MNIST neural net based image classifier from question 2.

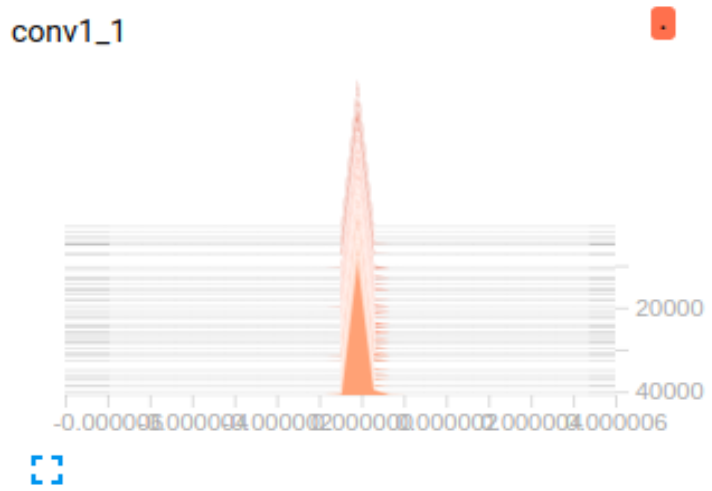
Learning Rate:



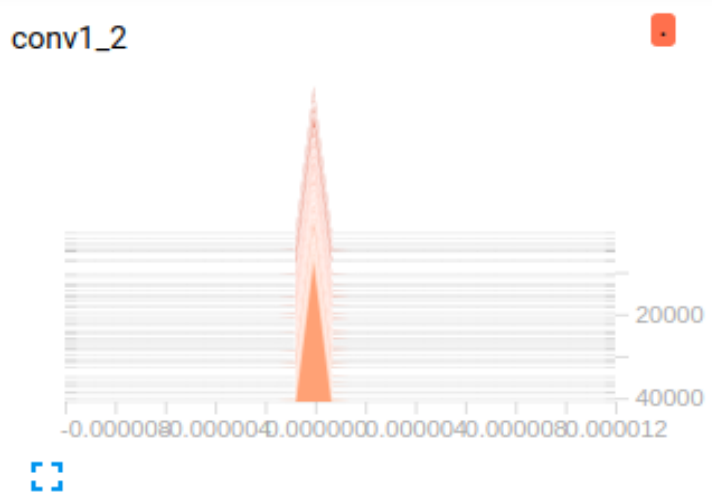
Histograms and Gradients:

(Over the next couple of pages)

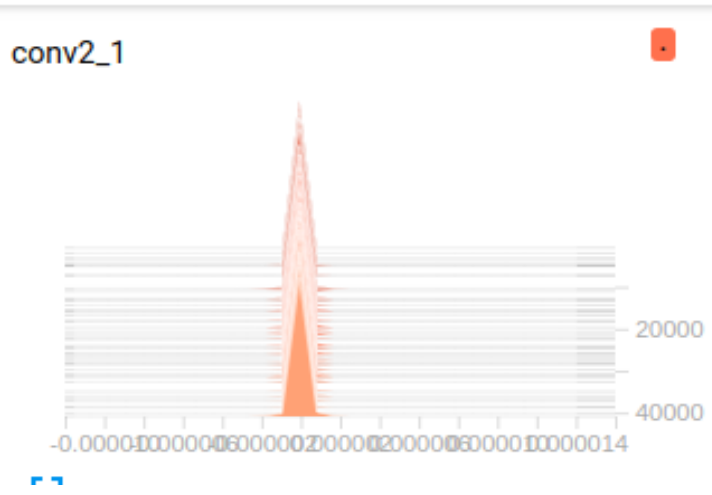
conv1_1

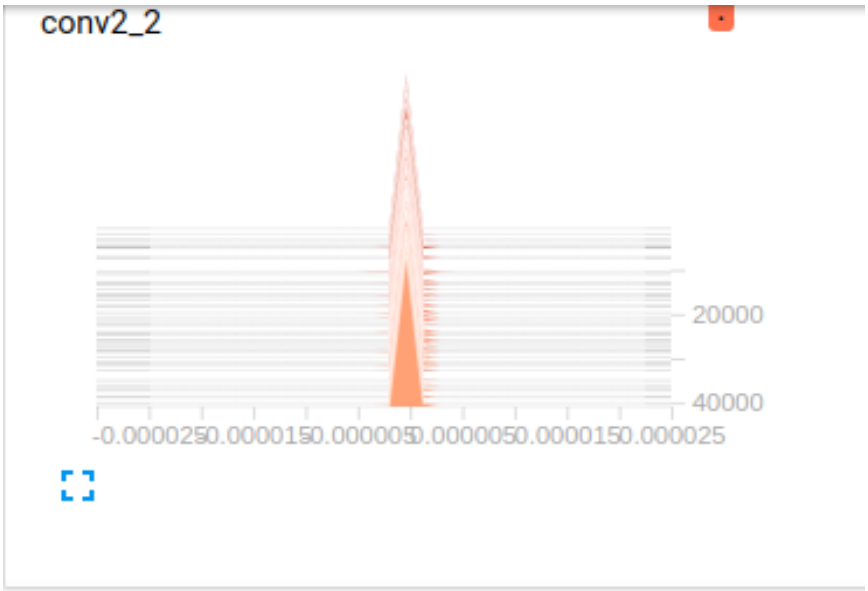


conv1_2

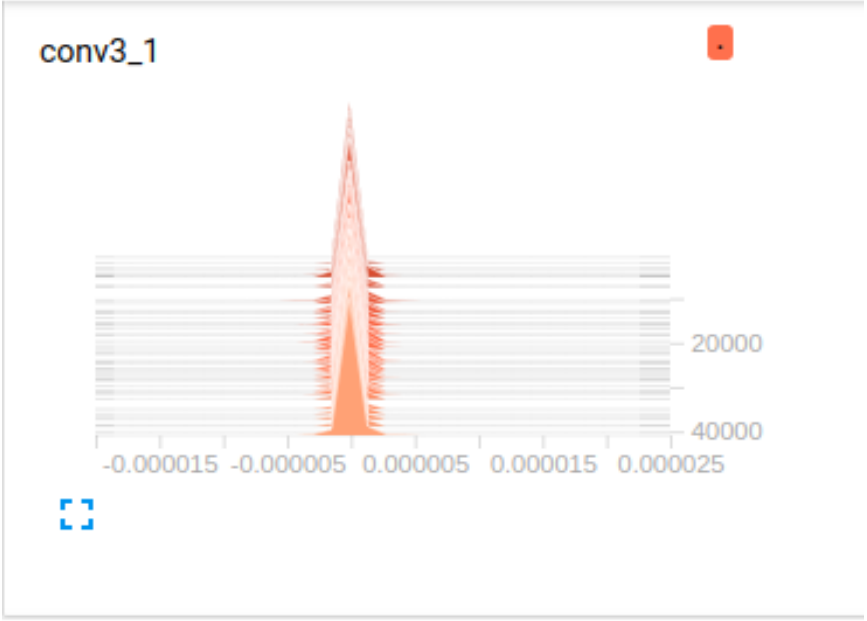


conv2_1

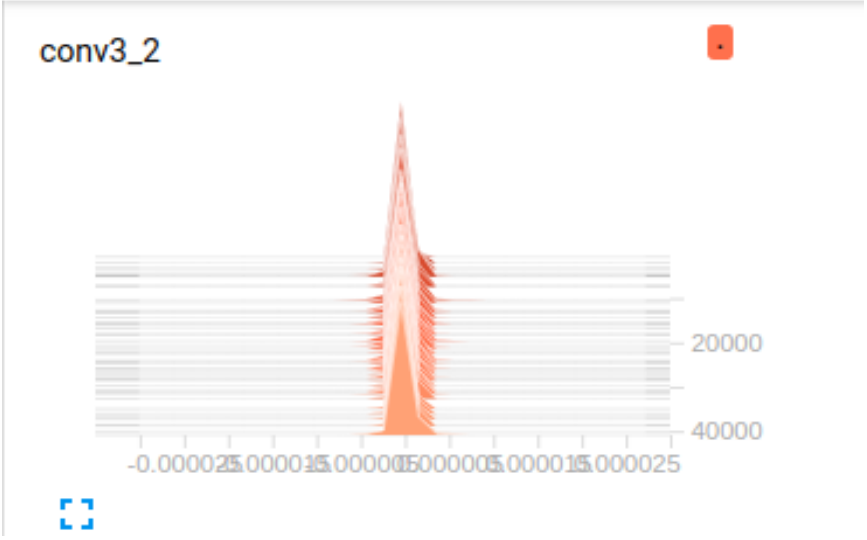




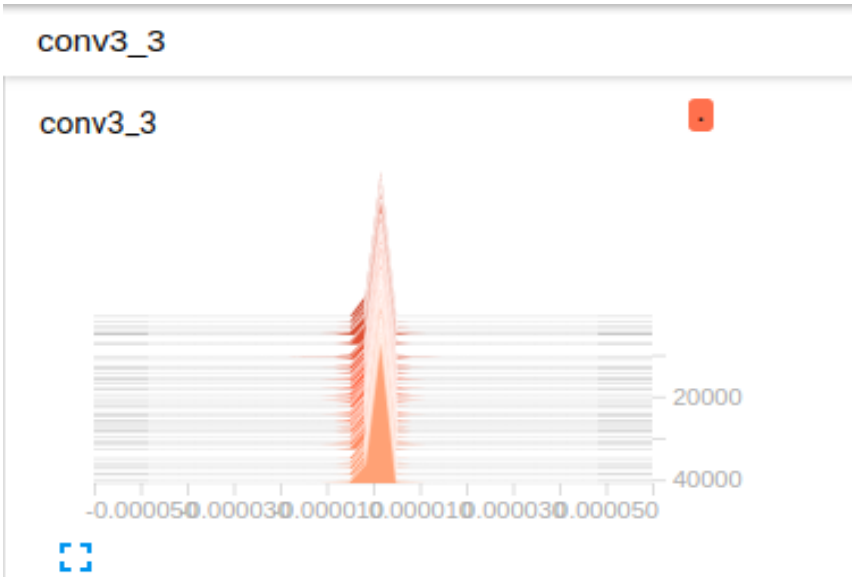
conv3_1



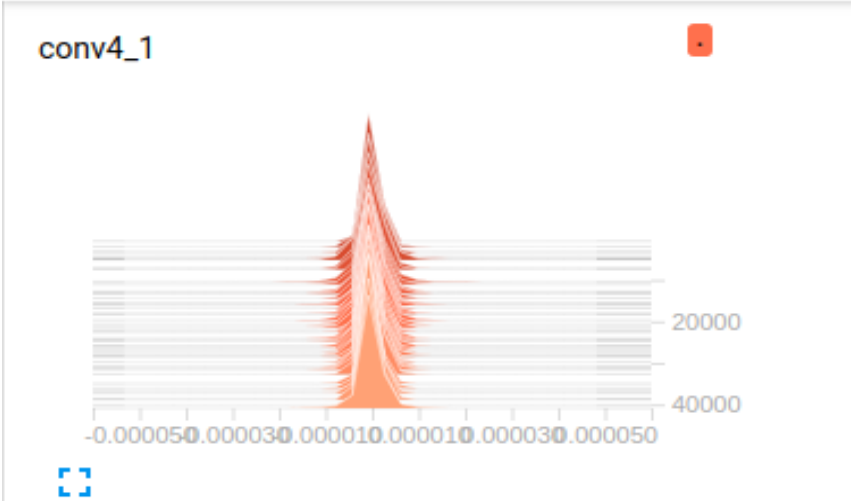
conv3_2



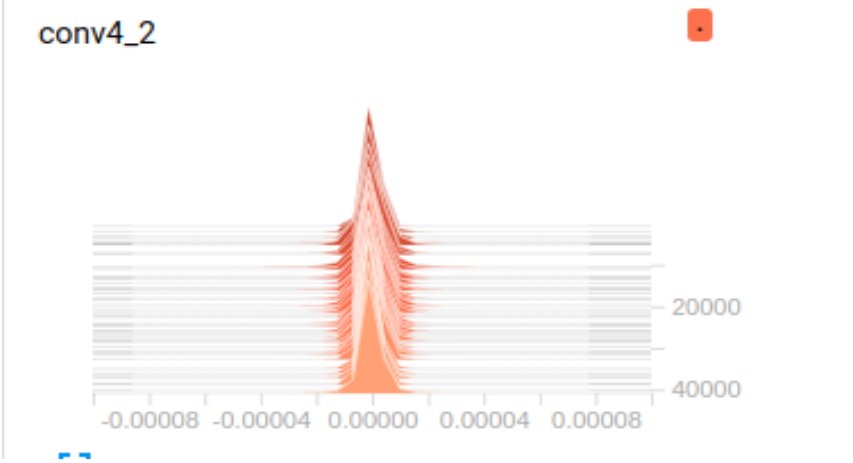
conv3_3

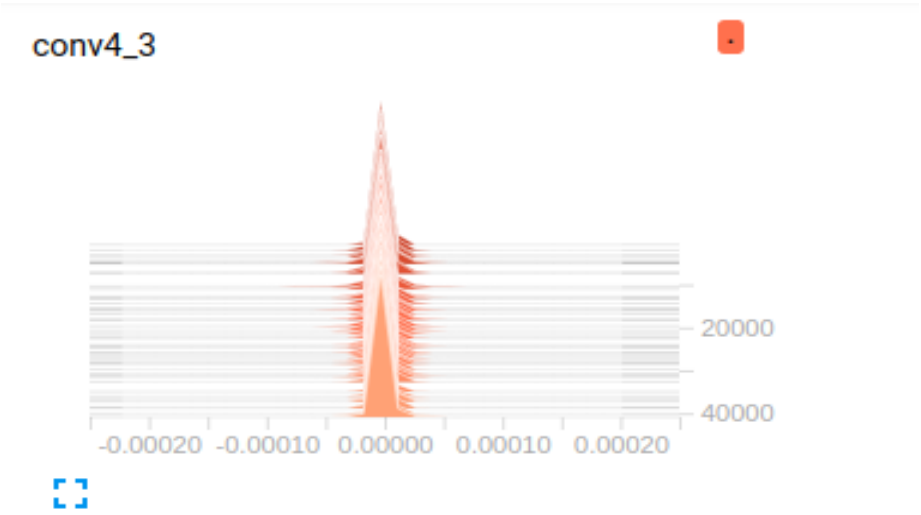


conv4_1

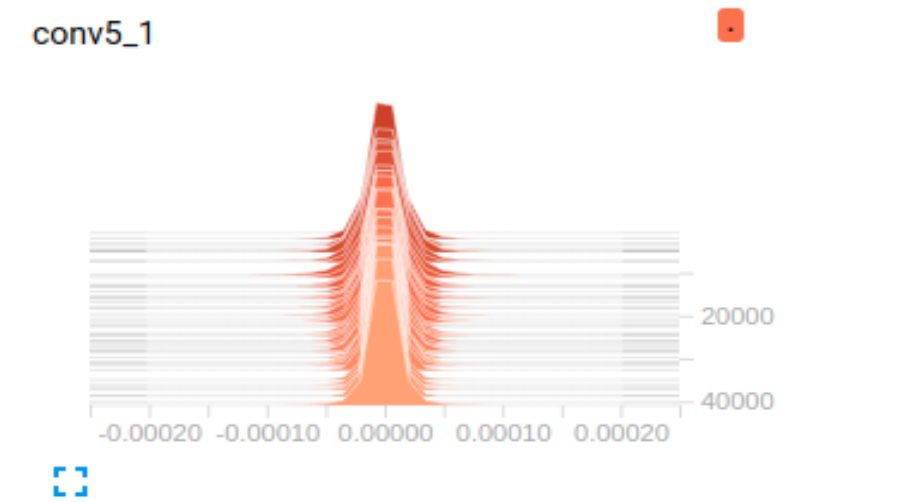


conv4_2

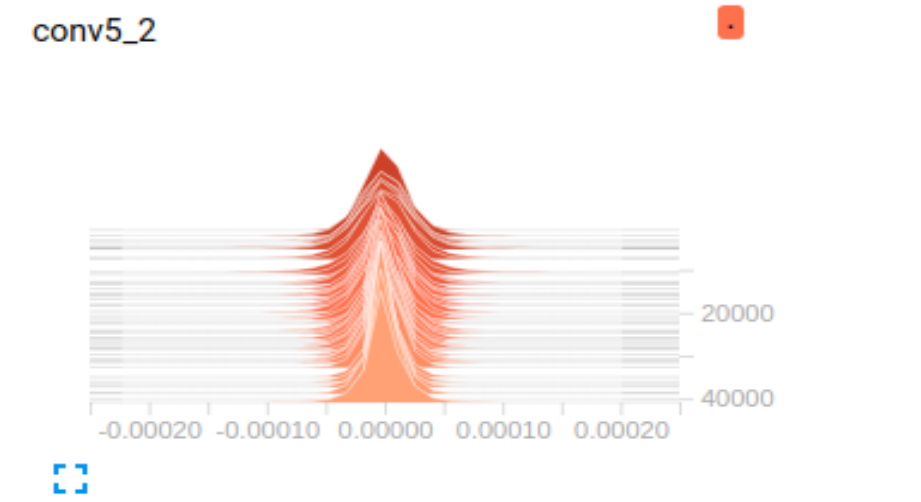


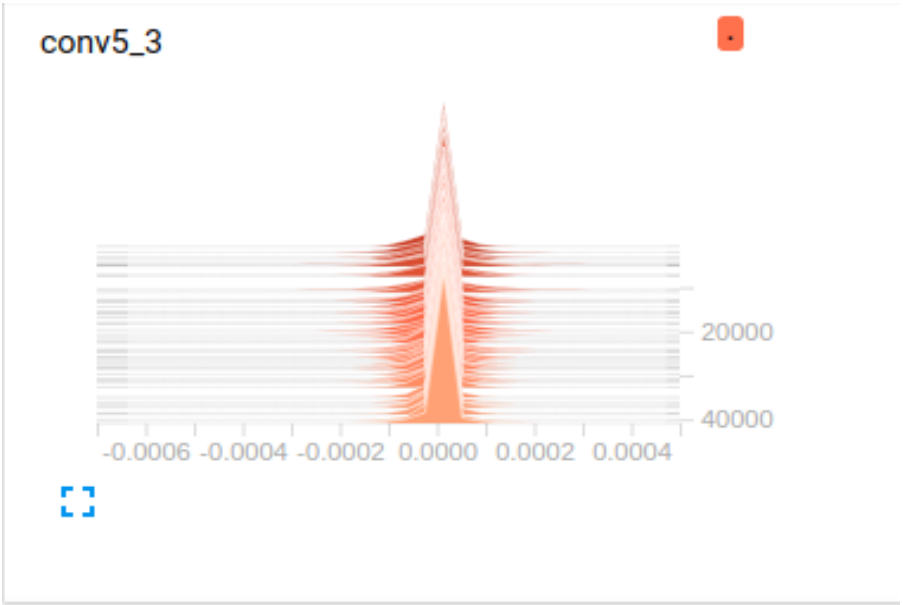


conv5_1

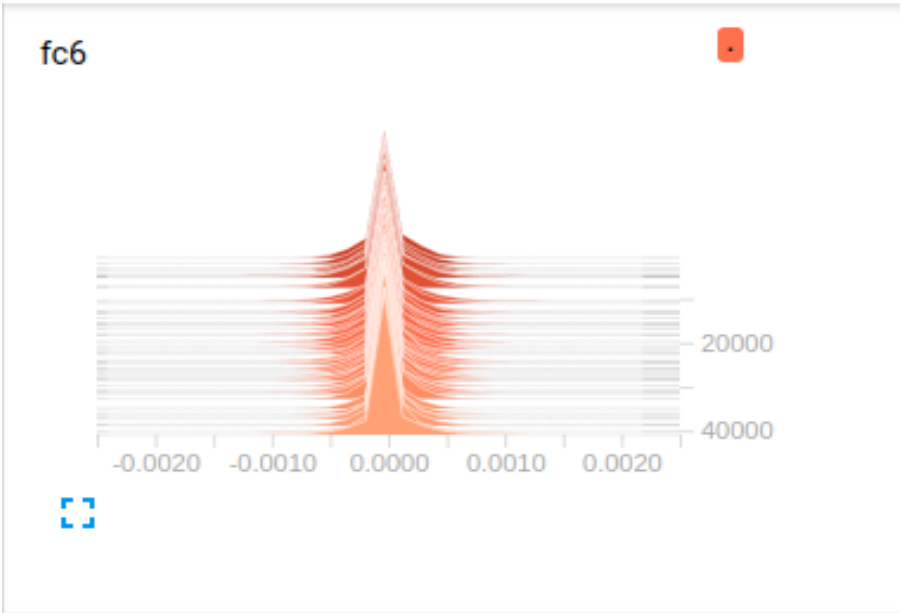


conv5_2

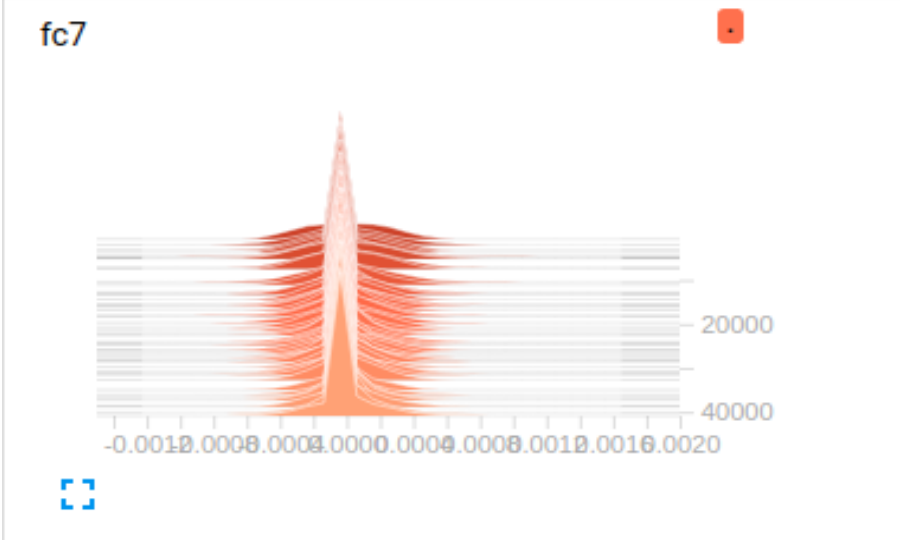



conv5_3
20000
40000
-0.0006 -0.0004 -0.0002 0.0000 0.0002 0.0004

fc6

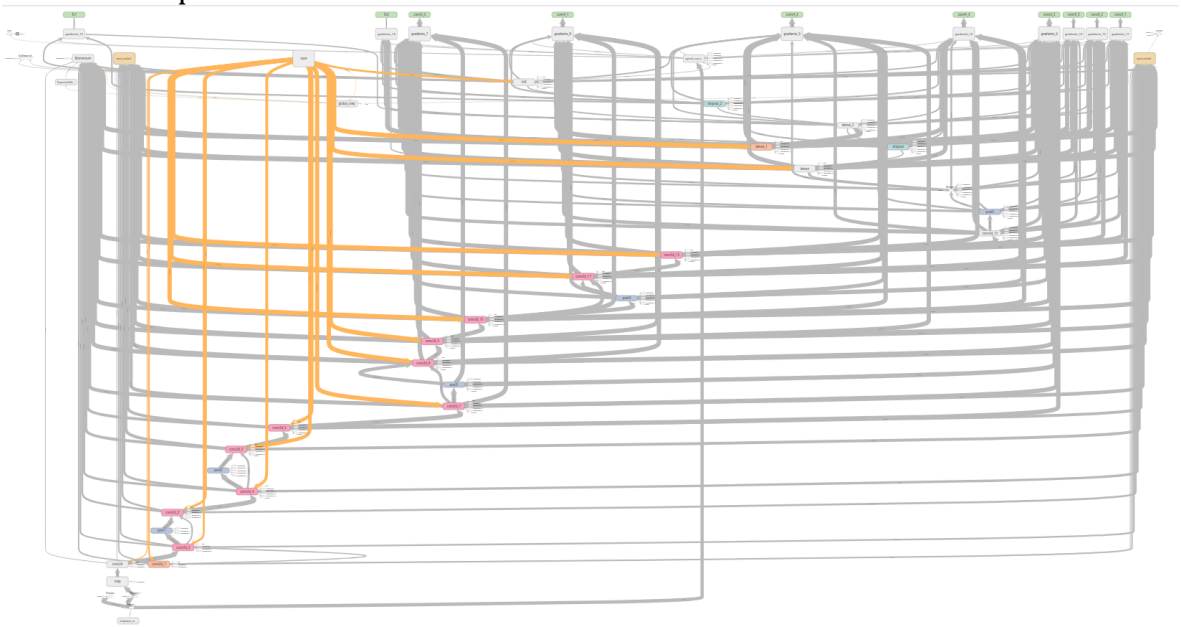

fc6
20000
40000
-0.0020 -0.0010 0.0000 0.0010 0.0020

fc7

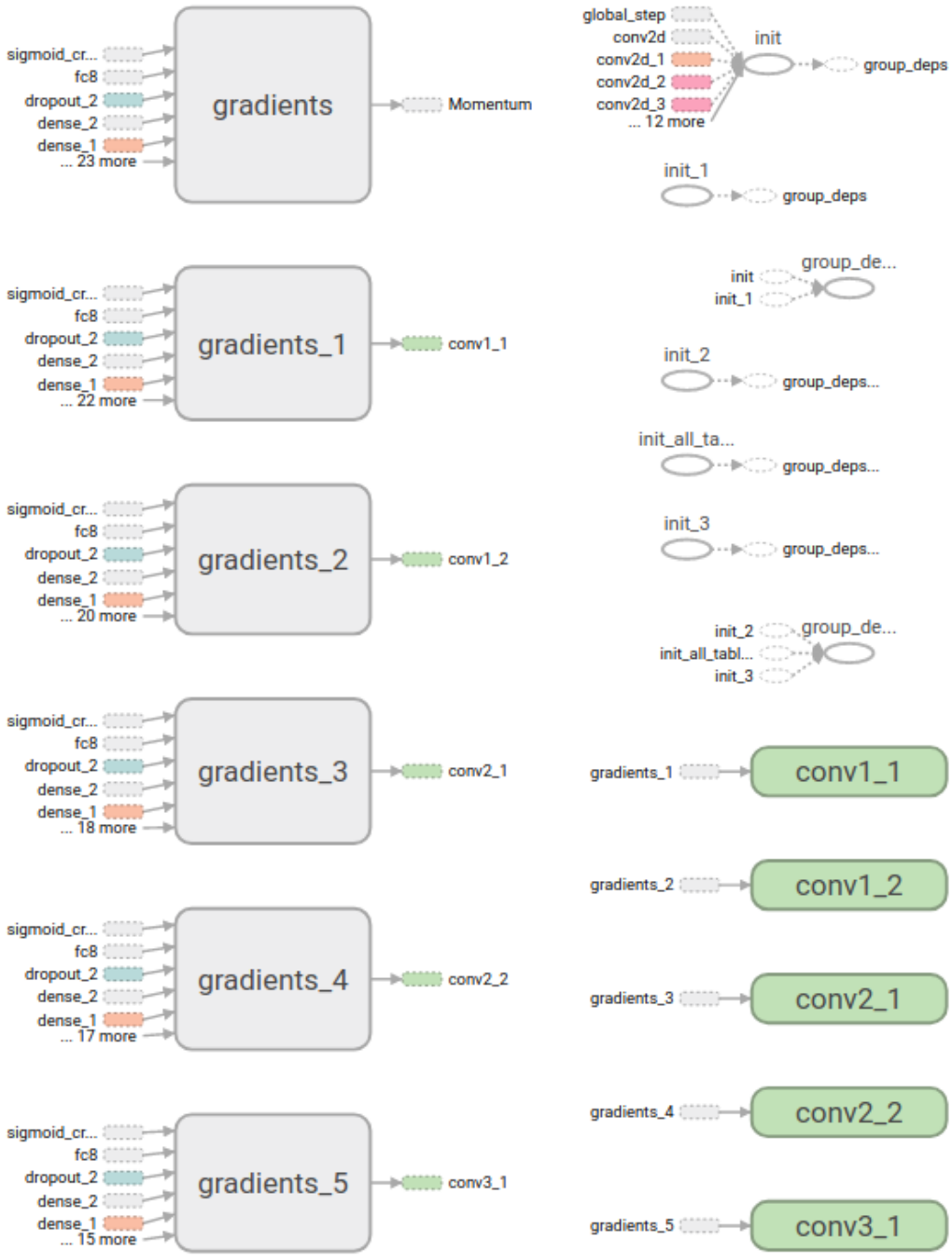

fc7
20000
40000

Training Images:

Network Graph:

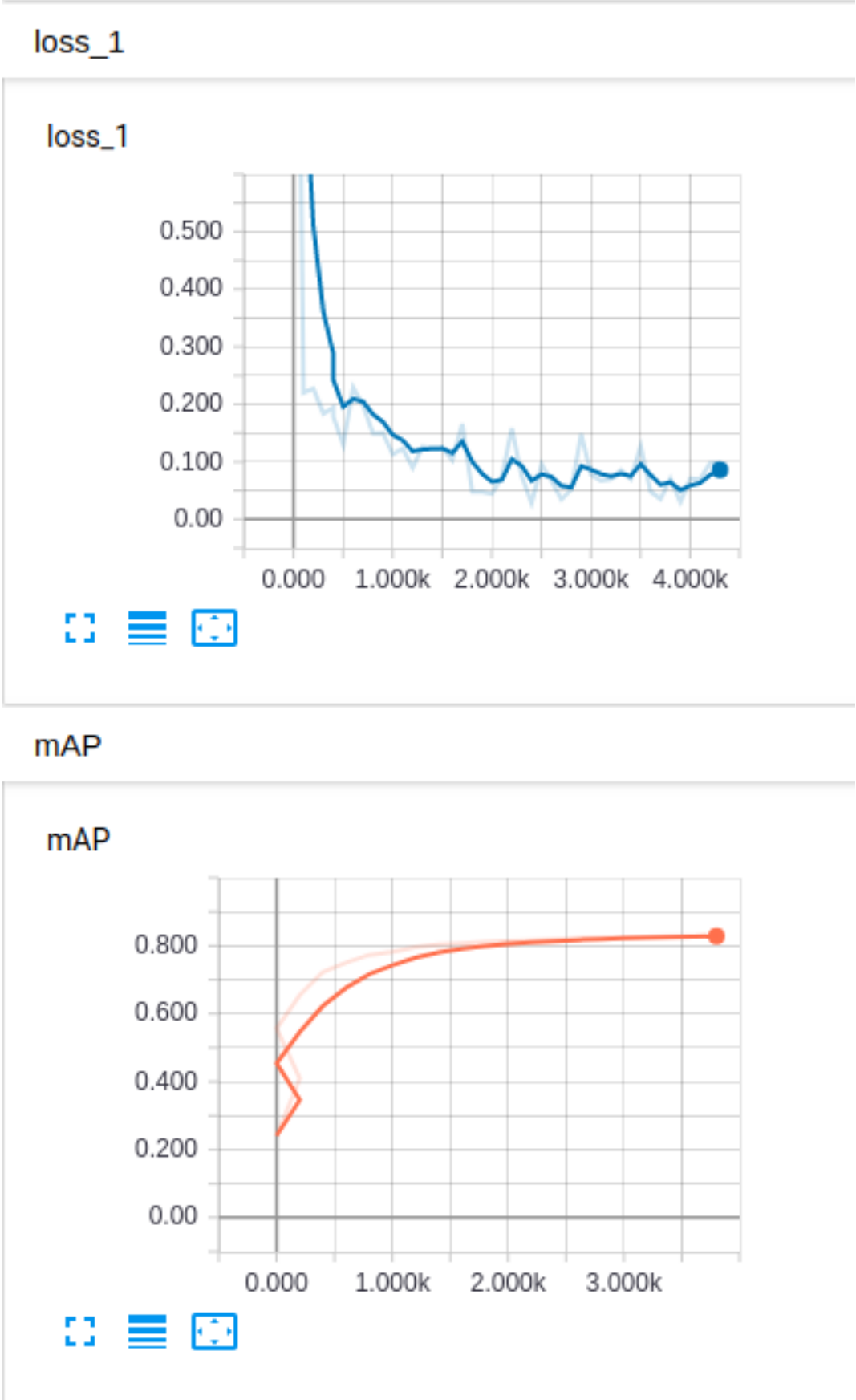


Auxiliary Subgraph:



This VGG network section was developed and influenced by tbe code of 2 other people:
Vivek Gr and Ritwik Das, both of whom are in the MRSD program.

Q4: Pre-trained VGG
Loss and Accuracy:



Obviously a pre-trained network is able to perform very well because the low level filters for detecting lines, edges and contours (among other features) are already learnt. The back propagation weight update signal, thus forces the majority of the weight update on the neurons in the layers closer to the output as those are the layers that extract more abstract features that ultimately allow the network to classify abstract objects.

To compare the accuracy of this pre-trained VGG network to some of the previous networks in this homework assignment, here is a screenshot of the prediction accuracies for each classification.

```
Random AP: 0.0746812127445 mAP
GT AP: 1.0 mAP
Obtained 0.829850535234 mAP
per class:
aeroplane: 0.939463248645
bicycle: 0.921055739532
bird: 0.927184715416
boat: 0.898924919065
bottle: 0.465870071525
bus: 0.834142355335
car: 0.928901503717
cat: 0.915895336065
chair: 0.663454629339
cow: 0.760095566826
diningtable: 0.763876924734
dog: 0.899072550047
horse: 0.906324820214
motorbike: 0.905084201023
person: 0.961733445764
pottedplant: 0.641860491135
sheep: 0.790245692398
sofa: 0.737194800078
train: 0.942464562477
tvmonitor: 0.794165131348
teamd@thebeast:~/hw1$
```

It is interesting how certain classes that would otherwise, obtain relatively average prediction accuracies, now have almost leading prediction accuracies. For example the train class has the second highest accuracy here, compared to a very average one for the MNIST network trained on Pascal. This could be because learning the low level features for detecting trains, is very time consuming and tricky from scratch. The loss function could cause the weights to update in a fashion that works against the effectiveness of the feature detectors for trains and what have you.

Q5
Hereafter is a print of the nearest neighbors for:
1. pool5 features from the AlexNet (trained from scratch)
2. fc7 features from the AlexNet (trained from scratch)
3. pool5 features from the VGG (finetuned from ImageNet)
4. fc7 features from VGG (finetuned from ImageNet)

```
layer name: Pool5_AlexNet
{2146: 836, 1998: 3852, 3249: 3559, 2034: 4122, 2771: 4459, 3413: 2293, 4470: 4577, 4023: 841, 3165: 4128, 4510: 4169}

layer name: FC7_AlexNet
{2146: 4213, 1998: 2731, 3249: 1928, 2034: 1126, 2771: 4459, 3413: 4927, 4470: 4044, 4023: 3559, 3165: 2770, 4510: 703}

layer name: Pool5_VGGNet
{2146: 1746, 1998: 4837, 3249: 4573, 2034: 2517, 2771: 4151, 3413: 1570, 4470: 1767, 4023: 2561, 3165: 1927, 4510: 1668}

layer name: FC7_VGGNet
{2146: 550, 1998: 1694, 3249: 4868, 2034: 2401, 2771: 838, 3413: 777, 4470: 2237, 4023: 3744, 3165: 4942, 4510: 4295}
```

Examples of some classes that are harder

Mobile objects are more likely to be occluded, oriented in different directions, and so allow the network to learn the concept with rotational quasi invariance, thus building up the predictive power. Static objects tend to be photographed from similar perspectives, thus making the networks less robust and flexible in detecting them from new perspectives.

There is a large quantity of data for cats, people, dogs and this makes the networks better able to detect them as the weights are updated by a larger amount of related photos.

Q6

Below is the prediction accuracy for each class of image with mixup (left) and baseline vgg (right) from question 4.

```
Random AP: 0.073758247139828886 mAP
GT AP: 1.0 mAP
Obtained 0.7667962848203731 mAP
per class:
aeroplane: 0.9079090540606433
bicycle: 0.8841020665043746
bird: 0.8961885287997863
boat: 0.8530733643509872
bottle: 0.333282876696633353
bus: 0.8121404692519958
car: 0.9248236308605768
cat: 0.8880507289029206
chair: 0.6267110480481914
cow: 0.5676953051331011
diningtable: 0.7228883285624668
dog: 0.8542688025037998
horse: 0.8629562547578661
motorbike: 0.8826162221918782
person: 0.9543754909638446
pottedplant: 0.5303857131382876
sheep: 0.5706398588794859
sofa: 0.631549173623427
train: 0.9248890218241582
tvmonitor: 0.7073797573530359
```

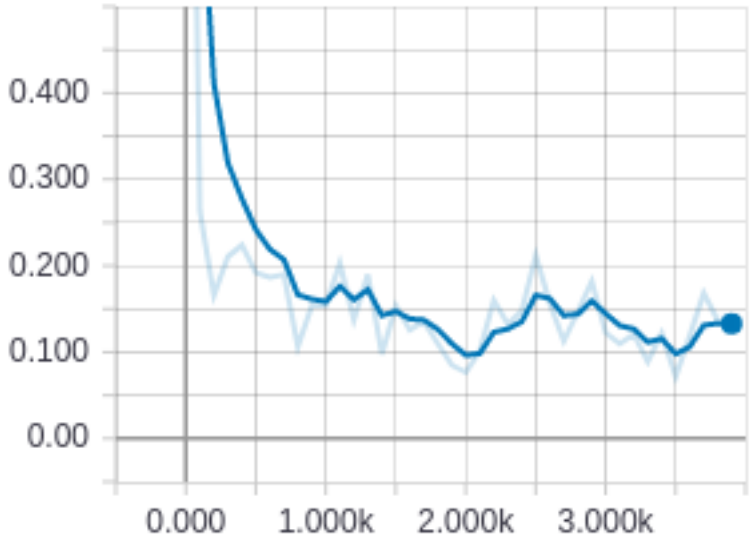
```
Random AP: 0.0746812127445 mAP
GT AP: 1.0 mAP
Obtained 0.829850535234 mAP
per class:
aeroplane: 0.939463248645
bicycle: 0.921055739532
bird: 0.927184715416
boat: 0.898924919065
bottle: 0.465870071525
bus: 0.834142355335
car: 0.928901503717
cat: 0.915895336065
chair: 0.663454629339
cow: 0.760095566826
diningtable: 0.763876924734
dog: 0.899072550047
horse: 0.906324820214
motorbike: 0.905084201023
person: 0.961733445764
pottedplant: 0.641860491135
sheep: 0.790245692398
sofa: 0.737194800078
train: 0.942464562477
tvmonitor: 0.794165131348
teamd@thebeast:~/hw1$
```

Looks like mixup performs better across the board, except for cats, which is probably because there are so many cat images on Pascal that linear mixup of images may smooth out abstract high level features that would otherwise be clearly exposed from the shear amount of cat images.

Here are the training and testing loss and accuracy charts:

loss_1

loss_1



mAP

mAP