Assignment – 1

16824 – Visual Learning & Recognition

Balarama Raju Buddharaju

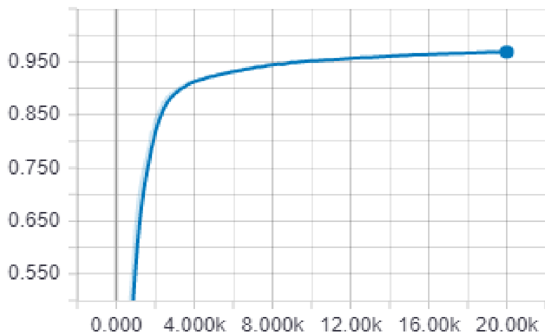
Task 0: MNIST 10-digit classification in TensorFlow:

Q 0.1) The model achieves a test accuracy of **96.9%** for 20,000 iterations (*00_mnist.py*).

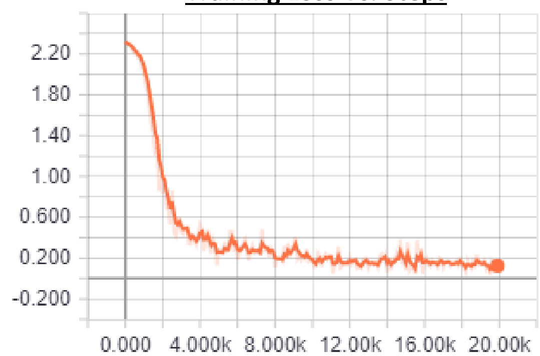
Q 0.2) The model achieves a test accuracy of **98.3%** for 30,000 iterations (*00_mnist_30k.py*).

Q 0.3) For the case of 20,000 iterations, the following plots were obtained:

Validation accuracy Vs. Steps



Training Loss Vs. Steps



Observations:

Neural Networks can do a brilliant job!

Classifying numbers is quite easy a task to achieve with not so deep a network and we can be assured of a good validation accuracy as there isn't much variance in how numbers could be written in non-cursive writing.

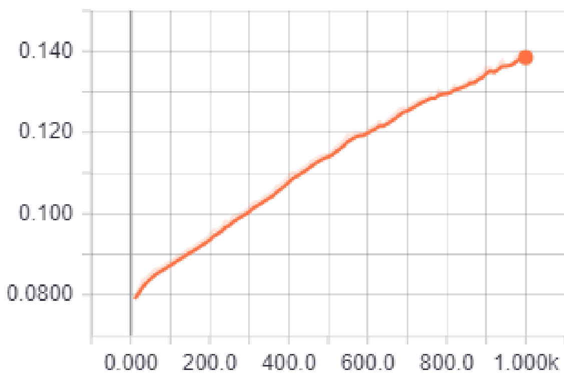
Task 1: 2-layer network for PASCAL multi-label classification

Q 1.1) Filled in the function definition for load_pascal(). Made effective use of python dictionaries and list comprehensions. Also checked the data to understand why some of the images were given not-so-confident label. For e.g., in one of the images, there was a dog in the background of a woman's picture and it was only partly visible. Though we as humans quickly notice the dog in the background, it gets tricky for the network as it will be primarily trained on data with complete images of the dogs. So, unless there are sizable number of images where dogs are partly visible, it is too much to expect the network to distinguish a partly visible dog (the importance of an i.i.d dataset!).

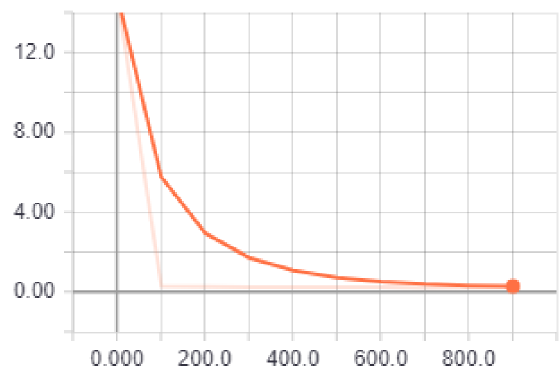
Q 1.2) Made modifications to the MNIST model function as directed. (*01_pascal.py*)

Q 1.3) I obtained the following curves for training loss and test accuracy for 1000 iterations (validation accuracy **0.14 mAP**):

<u>**Validation accuracy Vs. Steps**</u>



<u>**Training loss Vs. Steps**</u>



Observations:

The need for deeper networks

The mAP achieved is very low showing that the simple, shallower model we were using for MNIST is not good enough for a dataset as complicated as Pascal VOC. This calls for a deeper network.

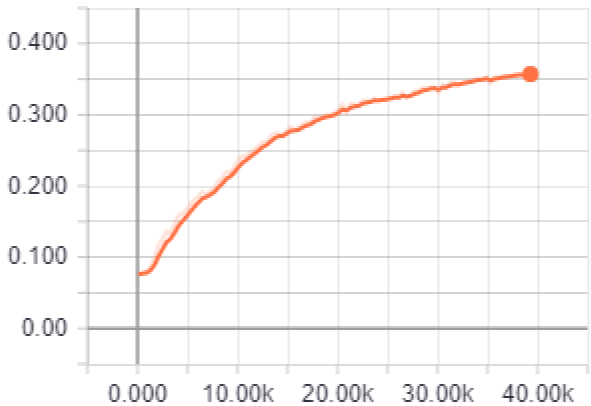
Task 2: Lets go deeper! AlexNet for PASCAL classification

Q 2.1) Replaced the MNIST model we were using before with this model. (*01_pascal_alexnet.py*)

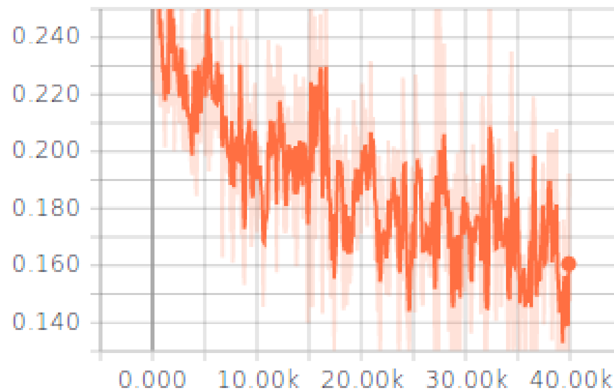
Q 2.2) Implemented the given solver parameters. (*01_pascal_alexnet.py*)

Q2.3) Performed data augmentation using "tf.image.random_flip_left_right" and "tf.random_crop". *(01_pascal_alexnet.py).* Achieved **0.38mAP**.

Validation accuracy Vs. Steps



Training Loss Vs. Steps



Observations:

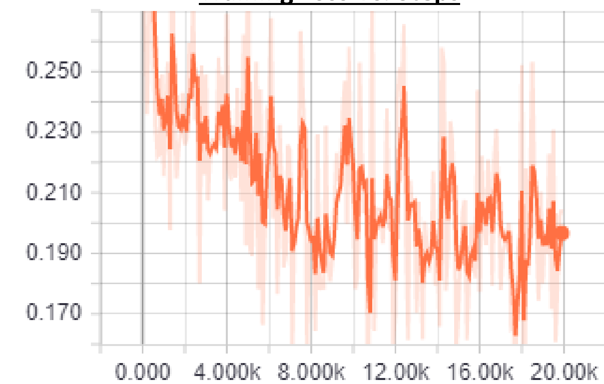
The role of data augmentation

Without data augmentation, I saw that my training loss was continuously dropping while my validation mAP was not showing corresponding improvement. It was clear that the network started to overfit to the training data. I understood the tendency of deep networks to overfit to training data. I saw how data augmentation helped with this problem by directing the network to learn the features as patterns rather than just overfit to data it has seen so far.

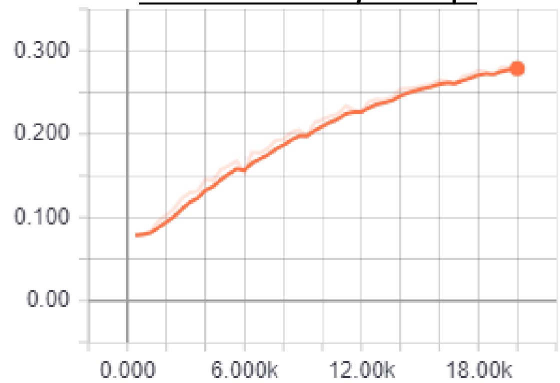
Task 3: Even deeper! VGG-16 for PASCAL classification:

Q 3.1) Modified the network architecture to “very deep” VGG-16 architecture (accuracy achieved: **0.28 mAP**)

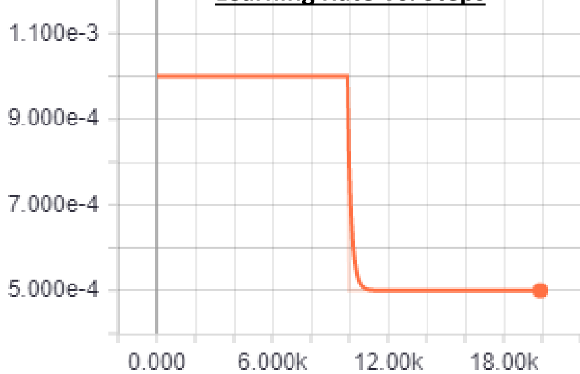
Training Loss Vs. Steps



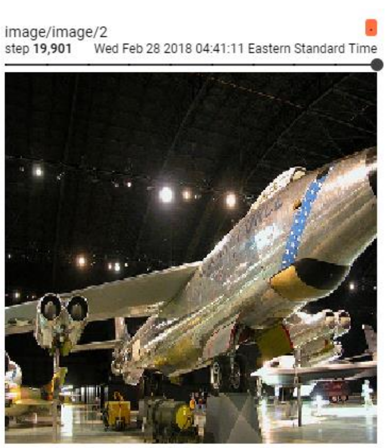
Validation accuracy Vs. Steps

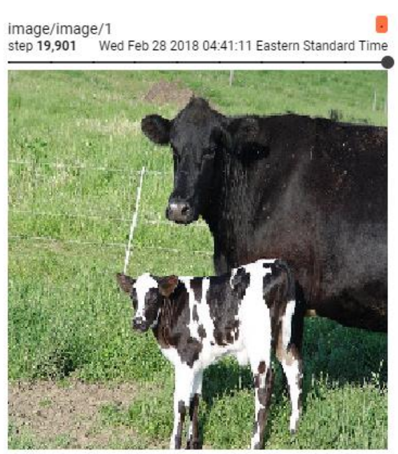


Learning Rate Vs. Steps

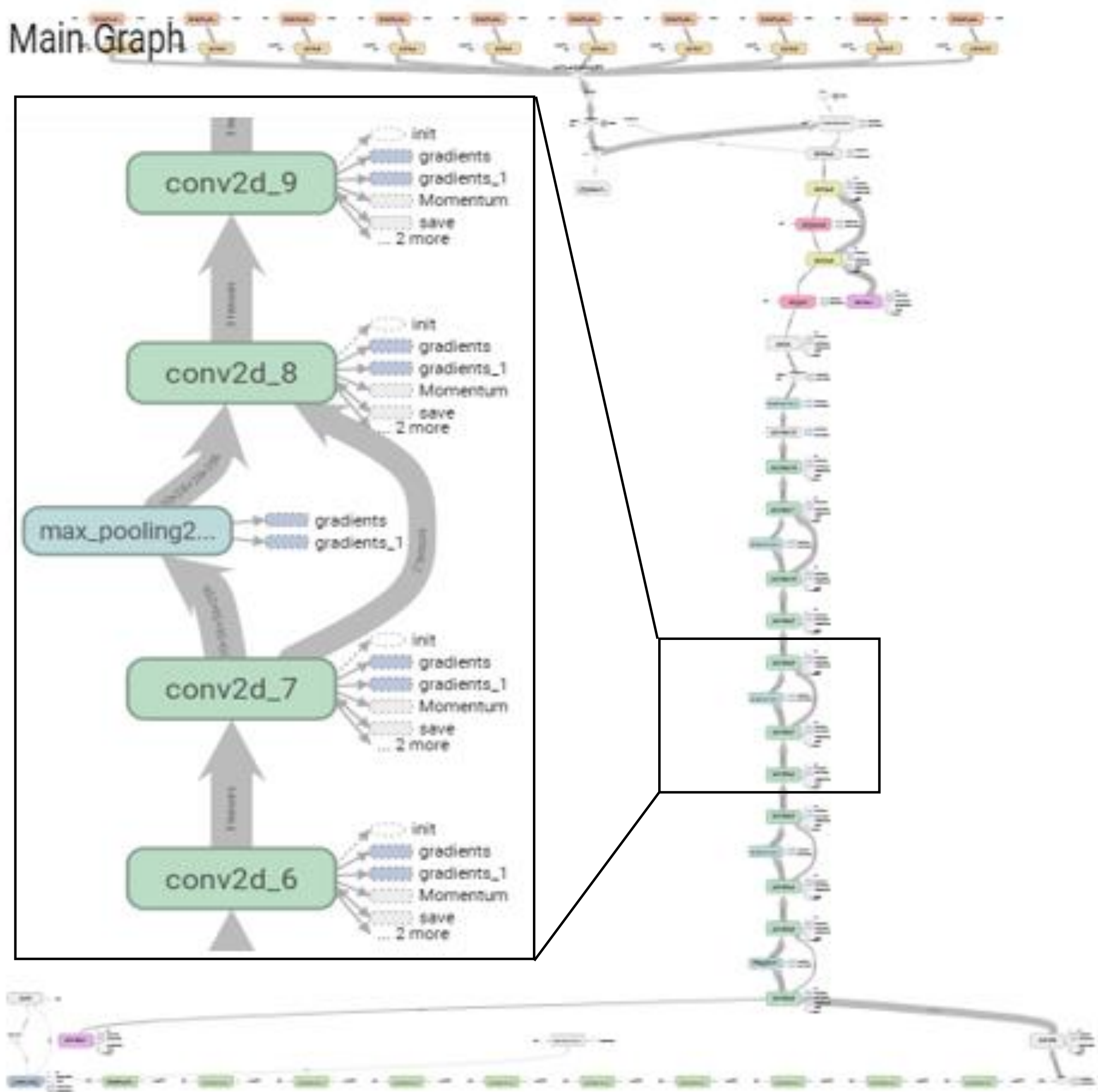


Training images

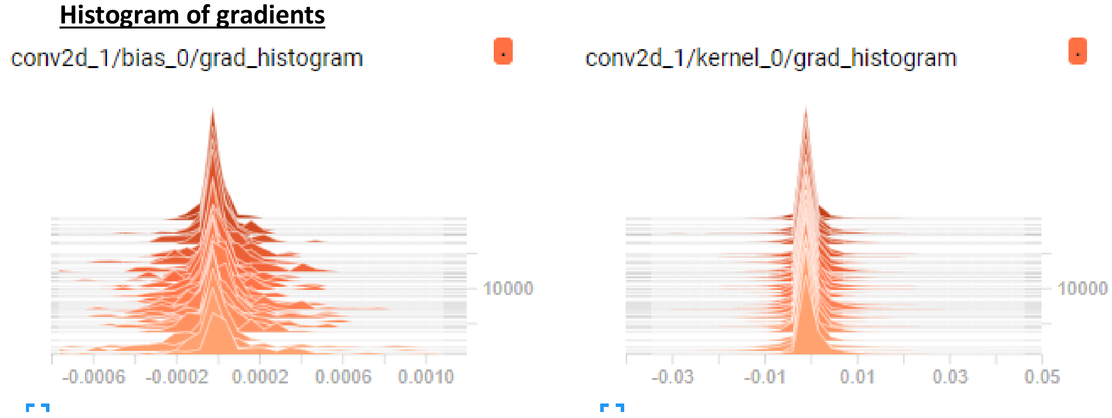


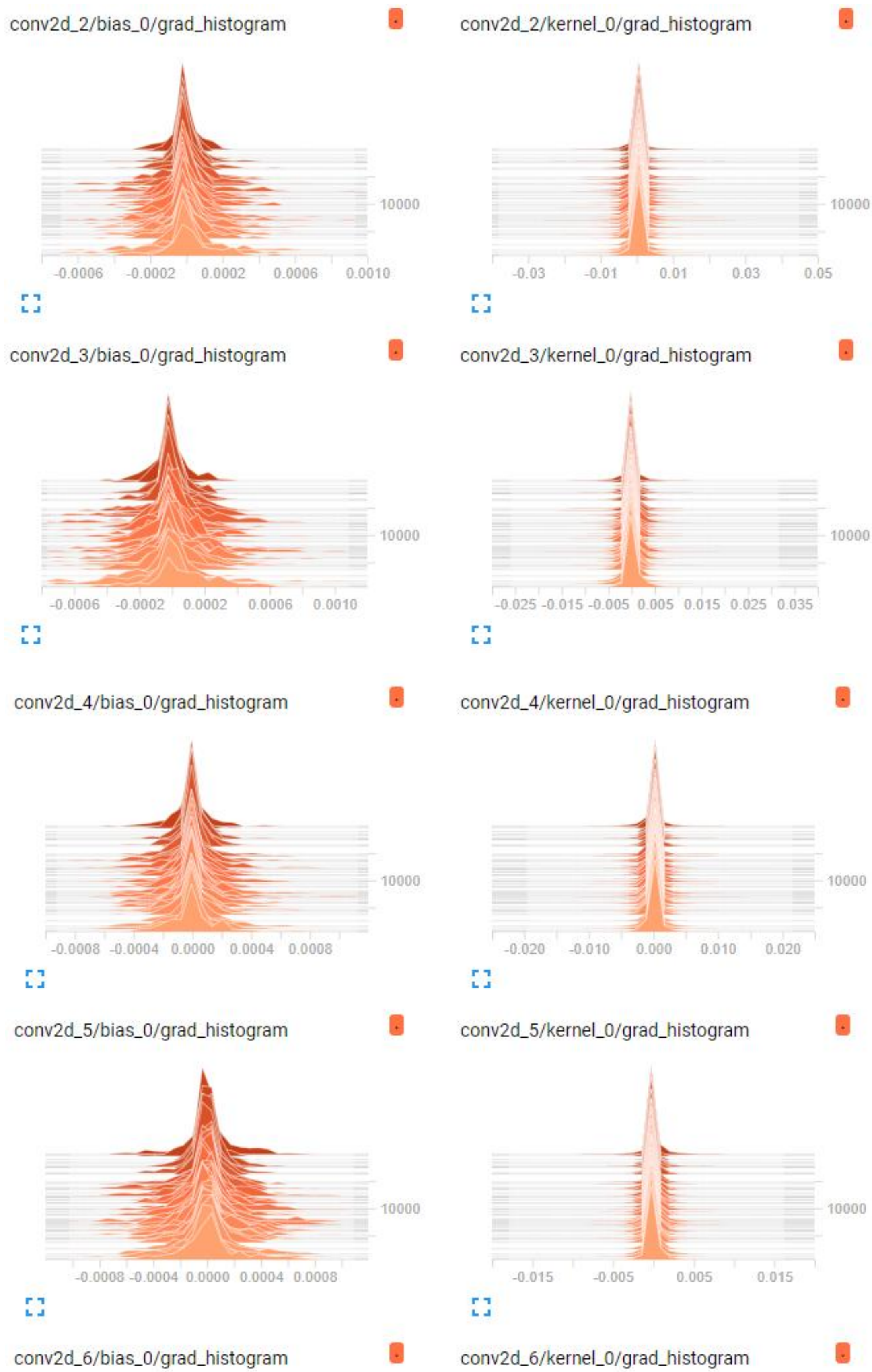


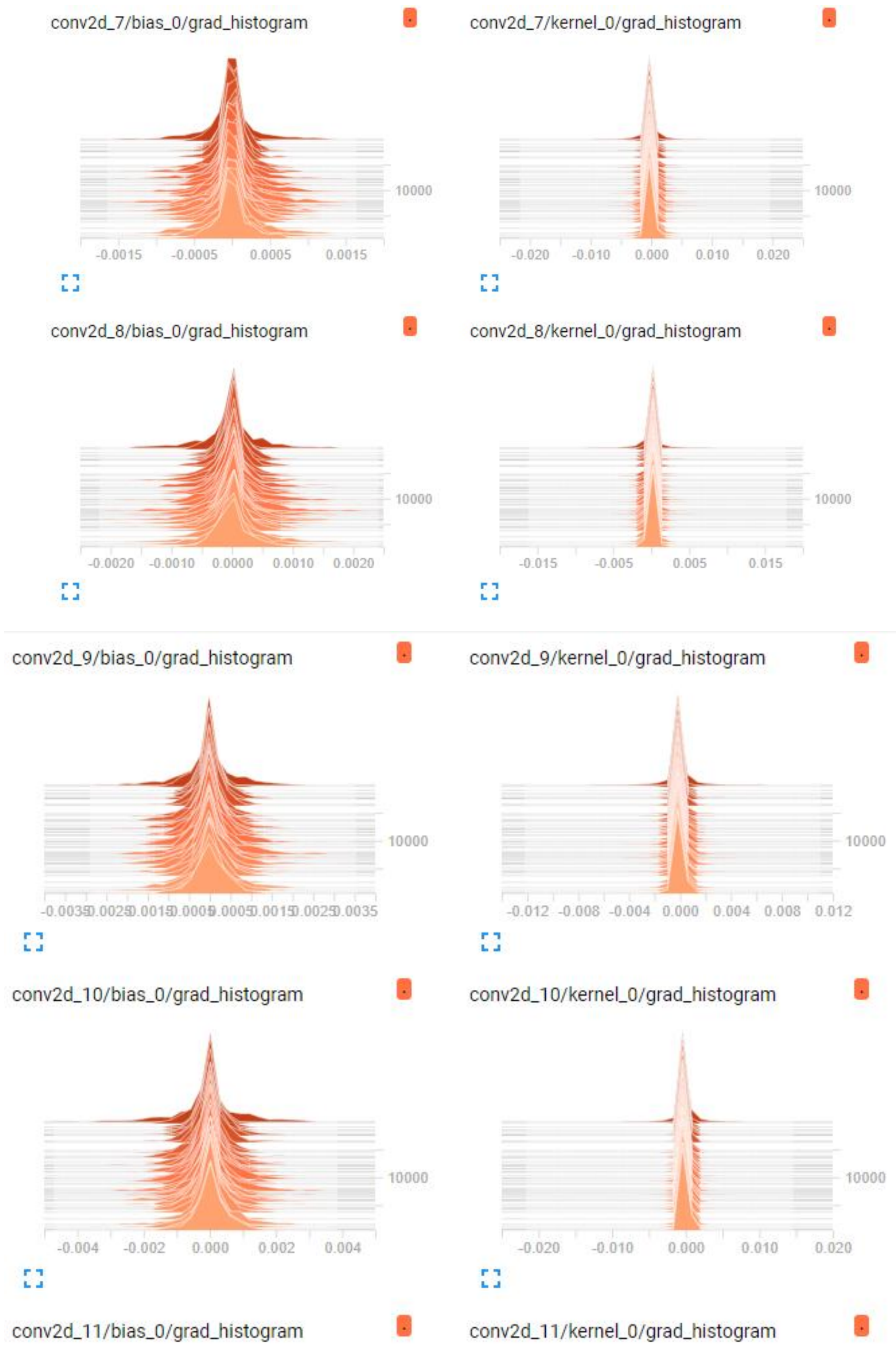
Network graph



Histogram of gradients




conv2d_2/bias_0/grad_histogram
10000
-0.0006 -0.0002 0.0002 0.0006 0.0010
conv2d_2/kernel_0/grad_histogram
10000
-0.03 -0.01 0.01 0.03 0.05
conv2d_3/bias_0/grad_histogram
10000
-0.0006 -0.0002 0.0002 0.0006 0.0010
conv2d_3/kernel_0/grad_histogram
10000
-0.025 -0.015 -0.005 0.005 0.015 0.025 0.035
conv2d_4/bias_0/grad_histogram
10000
-0.0008 -0.0004 0.0000 0.0004 0.0008
conv2d_4/kernel_0/grad_histogram
10000
-0.020 -0.010 0.000 0.010 0.020
conv2d_5/bias_0/grad_histogram
10000
-0.0008 -0.0004 0.0000 0.0004 0.0008
conv2d_5/kernel_0/grad_histogram
10000
-0.015 -0.005 0.005 0.015
conv2d_6/bias_0/grad_histogram
conv2d_6/kernel_0/grad_histogram

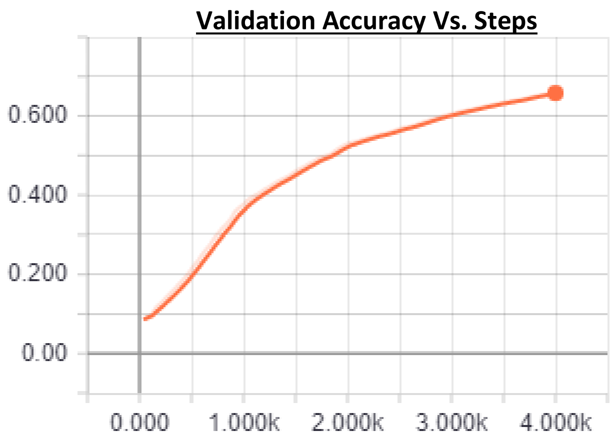


Observations:

Tensorboard is such a beauty only when it renders things. Getting things up and running is quite a challenge. I understand that Tensorflow is the industry standard for DNN frameworks but it becomes too tricky for people who are looking to try different hypotheses in a short timeframe. This makes dynamics frameworks such as PyTorch more popular among researchers.

Task 4: Standing on the shoulder of the giants: finetuning from ImageNet

Q 4.1 Trained the VGG16 network with pretrained weights upto fc7 layer. Obtained accuracy of **0.64 mAP**.



Observations:

The utility of pre-trained networks:

Clearly, in a very few iterations, we get the pre-trained network learn weights for our dataset. This clearly shows how transfer learning is possible and makes lives easy for a lot of people who are not interested in training networks from scratch.

Task 5: Analysis

Q 5.1 Visualization of conv1 filters of Caffenet at 3 distinct stages of training

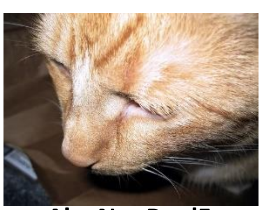
Weight visualization of Caffenet at checkpoint 32601

Weight visualization of Caffenet at checkpoint 39601

Weight visualization of Caffenet at checkpoint 40000

Q 5.2 Nearest neighbours based on fc7, pool5 layers of Alexnet and fc7, pool5 layers of VGG Net for 10 categories.

Person



AlexNet Pool5

AlexNet fc7

VGG pool5

VGG fc7

Car


AlexNet Pool5


AlexNet fc7

VGG pool5


VGG fc7

Train


AlexNet Pool5


AlexNet fc7


VGG pool5


VGG fc7

Aeroplane


AlexNet Pool5


AlexNet fc7


VGG pool5


VGG fc7

Horse

AlexNet Pool5

AlexNet fc7

VGG pool5

VGG fc7

Bus

AlexNet Pool5

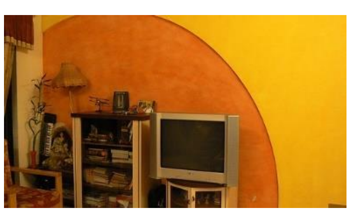
AlexNet fc7

VGG pool5

VGG fc7

Bicycle

AlexNet Pool5



AlexNet fc7

VGG pool5

TV Monitor

AlexNet Pool5

AlexNet fc7

VGG pool5

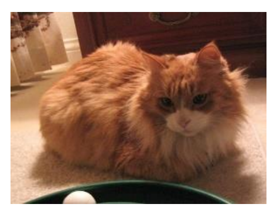
VGG fc7

Cat

AlexNet Pool5

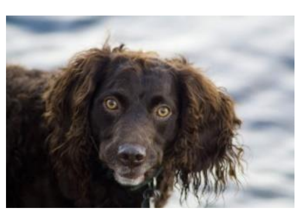
AlexNet fc7

VGG pool5



VGG fc7

Dog



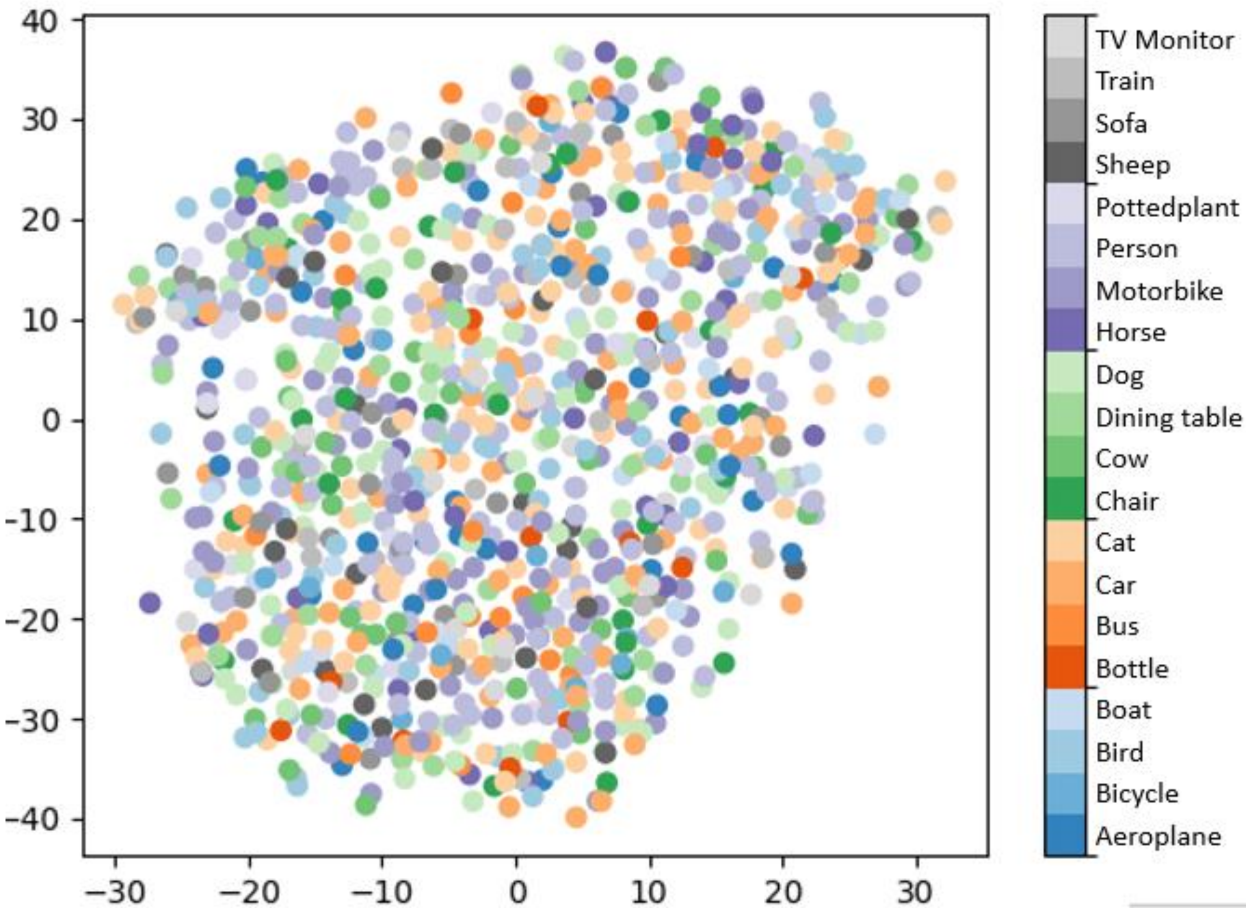
AlexNet Pool5

AlexNet fc7

VGG pool5

VGG fc7

Q 5.3) Obtained the following tSNE projection of 1000 randomly chosen images using fc7 features



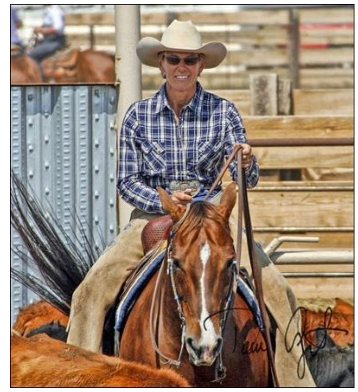
Q 5.4) Obtained the following performance metrics from VGG pre-trained and Alexnet

<u>Performance for VGG Pretrained</u>
Obtained 0.6415005856045797 mAP
per class:
aeroplane: 0.8327171510873679
bicycle: 0.6885207649291529
bird: 0.59345305343339
boat: 0.6097828610097589

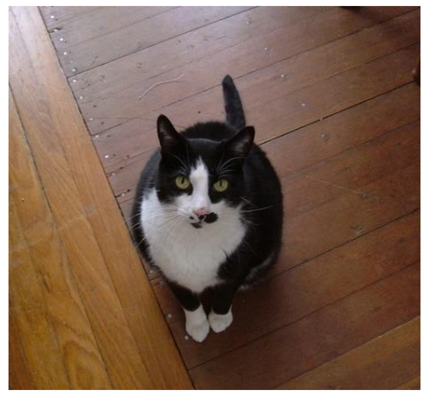
bottle: 0.31762247408387813
bus: 0.5242178733868459
car: 0.8378877594941565
cat: 0.41546577091297547
chair: 0.61732088423014403
cow: 0.3121152898534332
diningtable: 0.2865101445596775
dog: 0.5750898367552809
horse: 0.6776008139664495
motorbike: 0.6987990710256297
person: 0.9194599653389182
pottedplant: 0.35740746408262033
sheep: 0.446644131959046
sofa: 0.4066759725228843
train: 0.8182191240412602
tvmonitor: 0.83450130541872395

Performance for Alexnet:
Obtained 0.359823878478 mAP
per class:
aeroplane: 0.612454354906
bicycle: 0.31946478575
bird: 0.23601705859
boat: 0.36319083382
bottle: 0.140594071937
bus: 0.26307407375
car: 0.611779293415
cat: 0.29532757398
chair: 0.358246152662
cow: 0.178192962151
diningtable: 0.303387698253
dog: 0.246110284785
horse: 0.600332104846
motorbike: 0.471373815932
person: 0.754381273811
pottedplant: 0.174043909891
sheep: 0.245437332182
sofa: 0.272134563979
train: 0.480139424747
tvmonitor: 0.270796000162

The above metrics show that persons and cars achieve highest recognition metric while the potted plant, cow, cat and bottle have poor recognition results. This is probably because many-a-times humans occur in standard poses and cars have rigid well-defined structures.



On the other hand, the classes with poorer performance tend to occur in variety of shapes and sizes. This makes it difficult for the network to infer patterns that could be used to detect these classes.



The classes of birds, cows, cats and dogs see a good improvement in mAP when using transfer learning with ImageNet. This must be because these were the classes VGG16 was pre-trained on even before seeing our dataset. So it comes with an experience to identify/detect these classes with considerable ease when compared to a network trained from scratch.