# 16824 HW 1 Write-up

**Jagjeet Singh**

- **Task 0: MNIST 10-digit classification in TensorFlow**

1. What test accuracy does your model get?

   The final test accuracy was **96.94%**

2. What happens if you train for more iterations (30000)?

   When trained for 30000 iterations, the accuracy increased to **98.21%**

3. Make the training loss and validation accuracy curve. Show for at least 100 points between 0 to 20000 iterations.

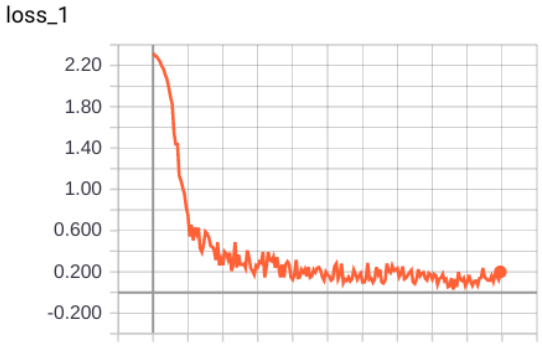



Figure 1: Training Loss for 20k iterations

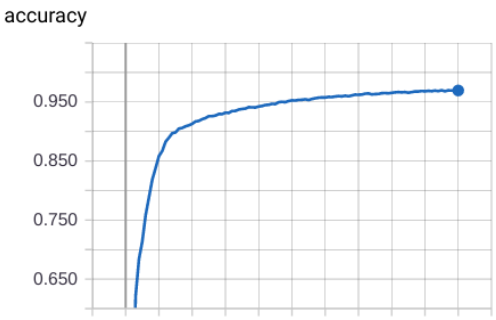



Figure 2: Validation accuracy for 20k iterations

Learnings: This part mainly helped me in getting acquainted with Tensoflow. In addition to the default Estimator API, I tried to pass several values from cnn_model_fn back to main and realized that train estimator cannot return anything.

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

> **NOTE:** The code expects Pascal data to be in the same folder as the code. Please keep the data accordingly and you can keep *anything* as the second argument for running the code. I didn't want to touch other parts of the code so I didn't change it.Eg.
> python 01_Pascalpy some_file_name

1. Write a data loader for PASCAL 2007
2. Modify the MNIST model function to be suitable for multi-label classification
3. show the training loss and test accuracy (mAP) curves. Train for only 1000 iterations.

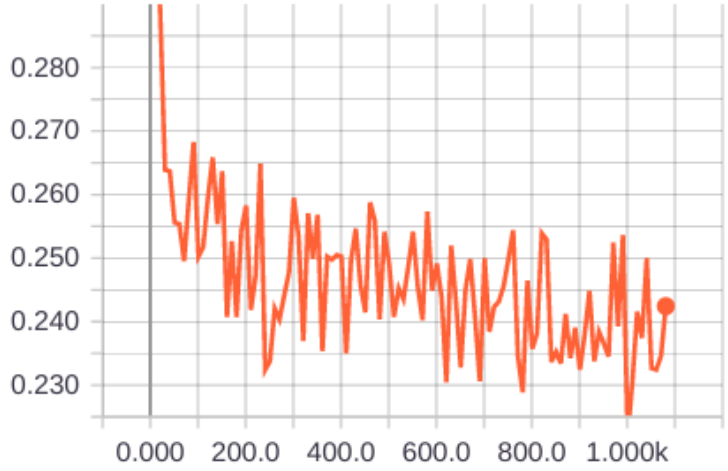



Figure 3: Training Loss for 1000 iterations

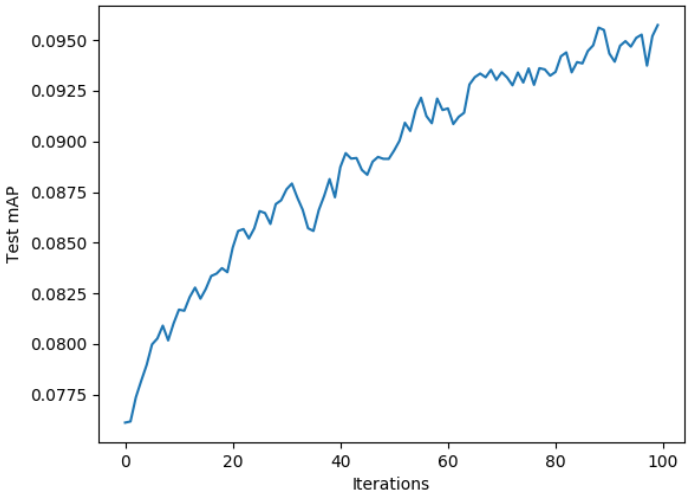



Figure 4: Test mAP for 1000 iterations

**Learnings:** The main learning in this part was to understand and implement multi-label classification. Modifying the loss metric to accommodate for multiple labels and understanding the suitability of mAP for this case is a key takeaway. I tried to plot test mAP and training loss on the same Tensorboard link but was not able to.

- **Task 2: Lets go deeper! AlexNet for PASCAL classification**
  1. Replace the MNIST model we were using before with this model.
  2. Implement the above solver parameters
  3. Implement the data augmentation and generate the loss and mAP curves as before

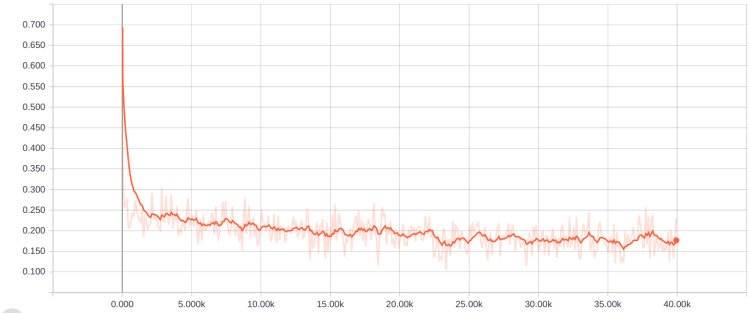



Figure 5: Training Loss for 40000 iterations

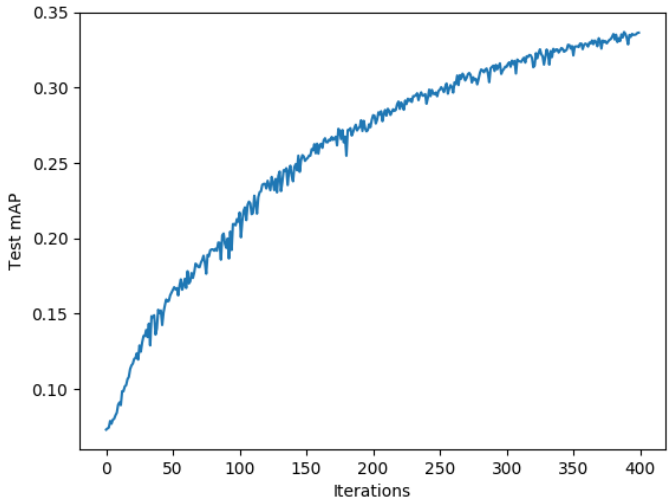



Figure 6: Test mAP for 40000 iterations

**Learnings:** The main takeaway from this part is implementing a CNN architecture from scratch. I could clearly see the effect of increasing depth and data augmentation on mAP. Although I implemented data augmentation to test set as well, I was not convinced. That would just give us a different estimate of performance and will have no role in achieving a better model.

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

  1. Modify the network architecture from Task 2 to implement the VGG-16 architecture (refer to the original paper). Use the same hyperparameter settings from Task 2, and try to train the model. Add the train/test loss/accuracy curves into the report.

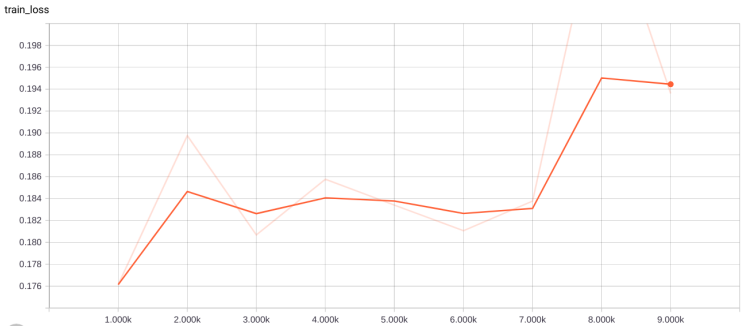



Figure 7: Training Loss for 10000 iterations

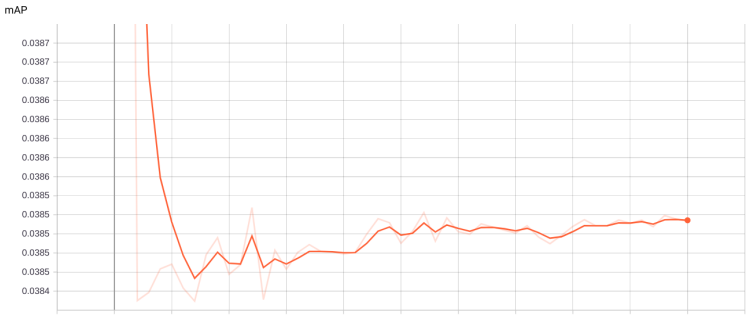



Figure 8: Test mAP for 40000 iterations

> **Learning:** he performance is clearly unsatisfactory. The reason for such low mAP is the Gaussian Initialization. I came to know later that removing Gaussian Initialization will improve the performance but I had already exhausted enough of my AWS credits. As such, I couldn't re-train the network to verify if change in initialization improves mAP. That said, using Tensorboard is the best takeaway from this part.

  2. The task in this section is to log the following entities: a) Training loss, b) Learning rate, c) Histograms of gradients, d) Training images and e) Network graph into tensorboard. Add screenshots from your tensorboard into the report

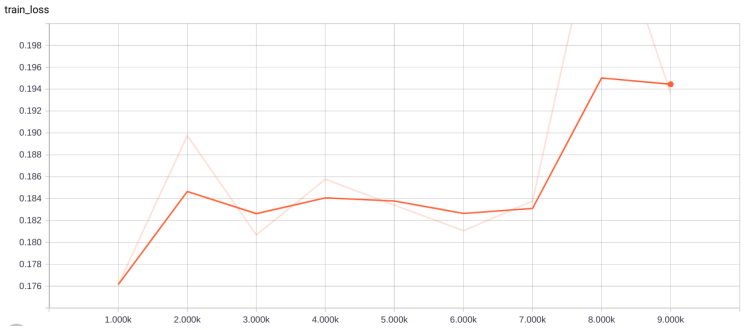



Figure 9: Training Loss from Tensorboard for 10000 iterations

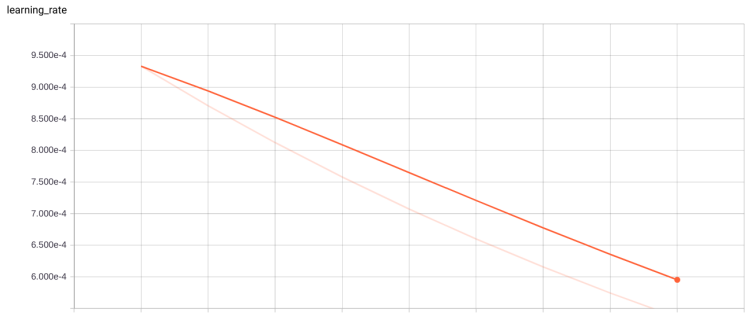



Figure 10: Learning Rate for 10000 iterations

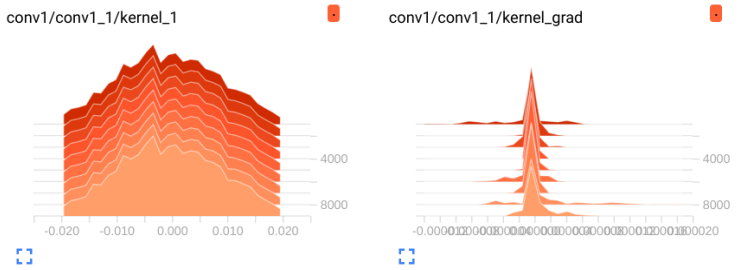



Figure 11: Histogram of conv1_1 weights and gradients

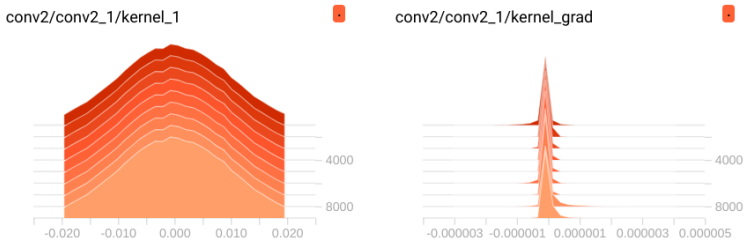



Figure 12: Histogram of conv2_1 weights and gradients

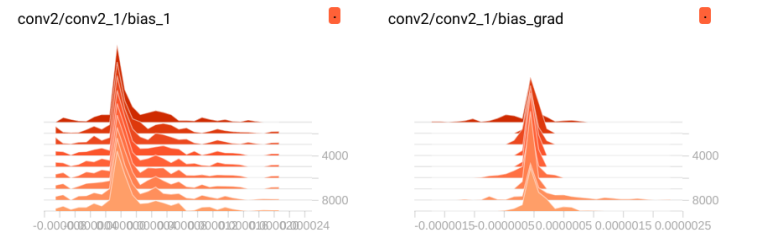



Figure 13: Histogram of conv2_1 bias and gradients

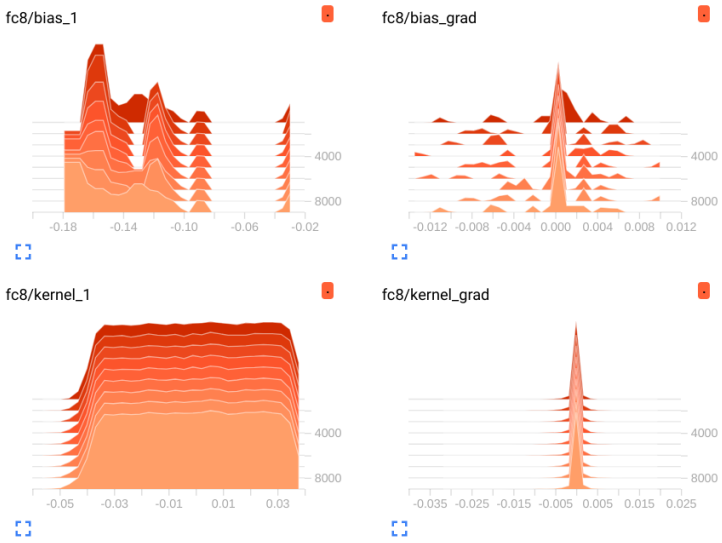



Figure 14: Histogram of fc8 weights, bias and their gradients

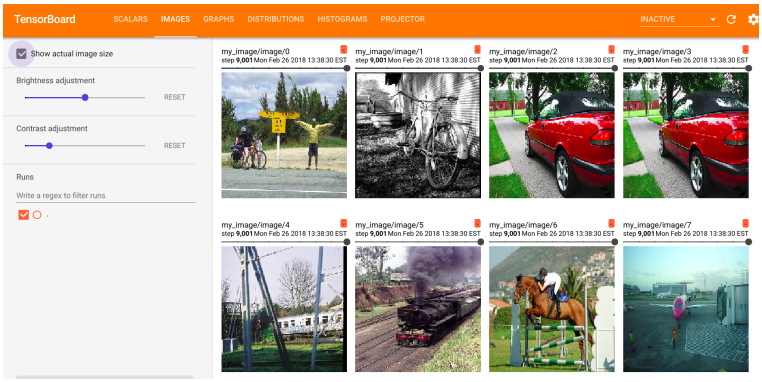



Figure 15: Sample images from Tensorboard

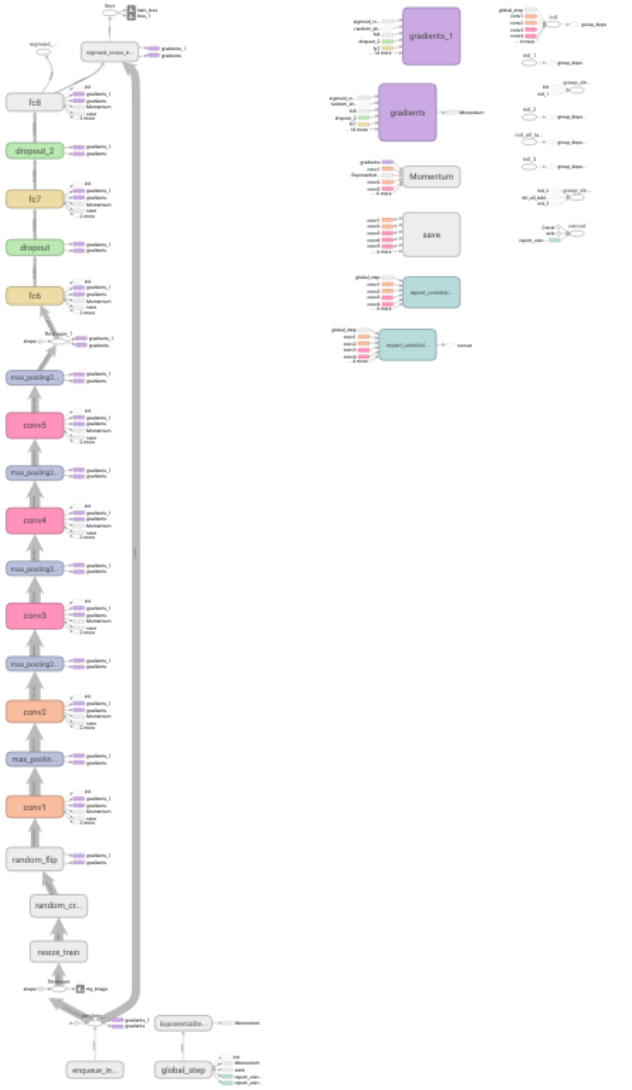



Figure 16: Graph for VGGNet

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

  1. Load the pre-trained weights upto fc7 layer, and initialize fc8 weights and biases from scratch. Then train the network as before and report the training/validation curves and final performance on PASCAL test set.

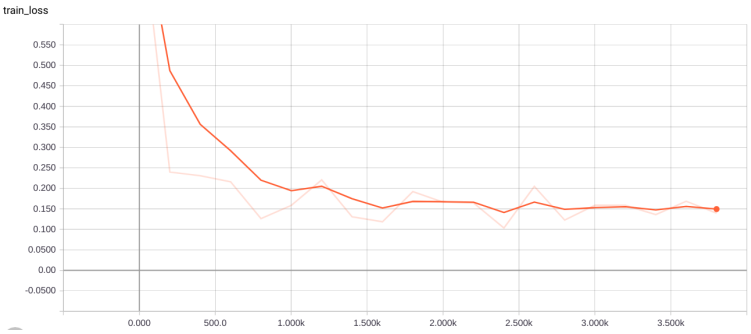



Figure 17: Training Loss for 4000 iterations

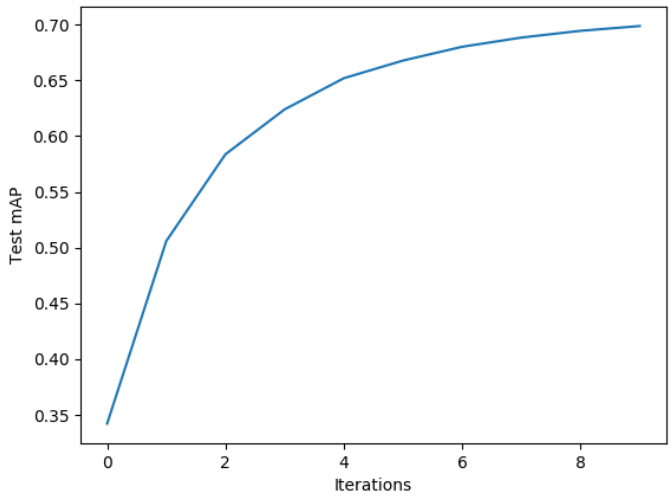



Figure 18: Test mAP for 4000 iterations

The final test mAP after 4000 iterations was **69.91%**. The mAP was still increasing but because of shortage of computing resources, I wasn't able to train it for more iterations

**Learnings:** This part enforced the importance of Transfer Learning in my mind. The training gave surprising good results within 1000 iterations. I wanted to play around more with different optimizers, initialization and batch normalization but I wasn't able to do that because of restriction in time and computing power.

- **Task 5: Analysis**
  1. Conv-1 filters

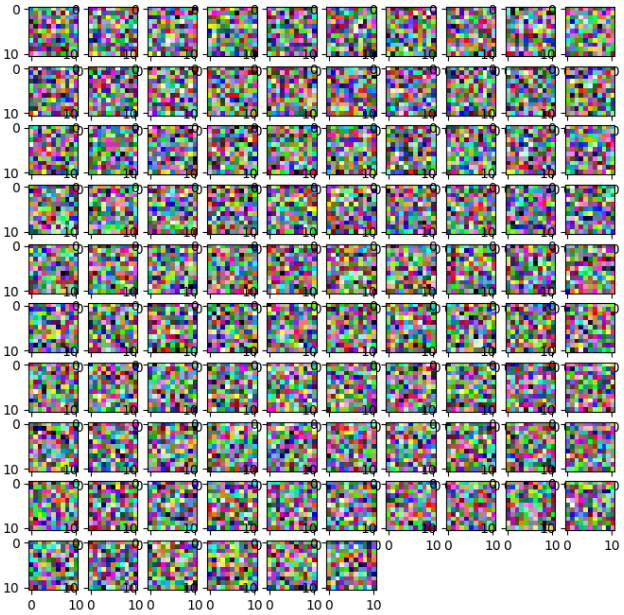


Figure 19: Conv1 filters after 20000 iterations

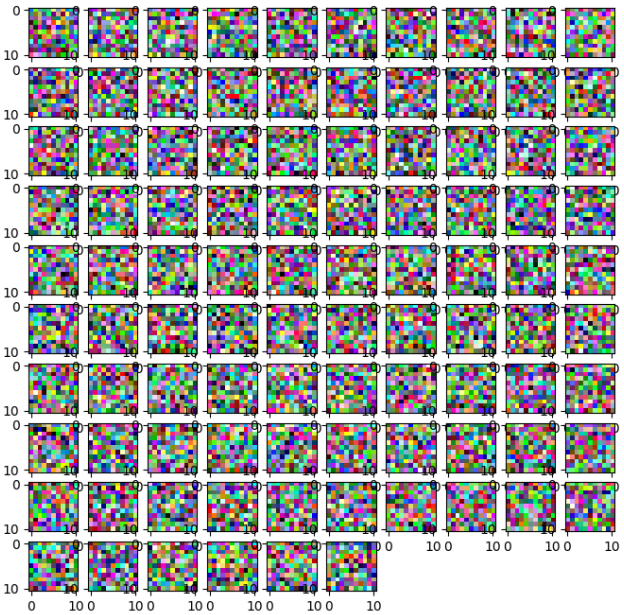


Figure 20: Conv1 filters after 30000 iterations

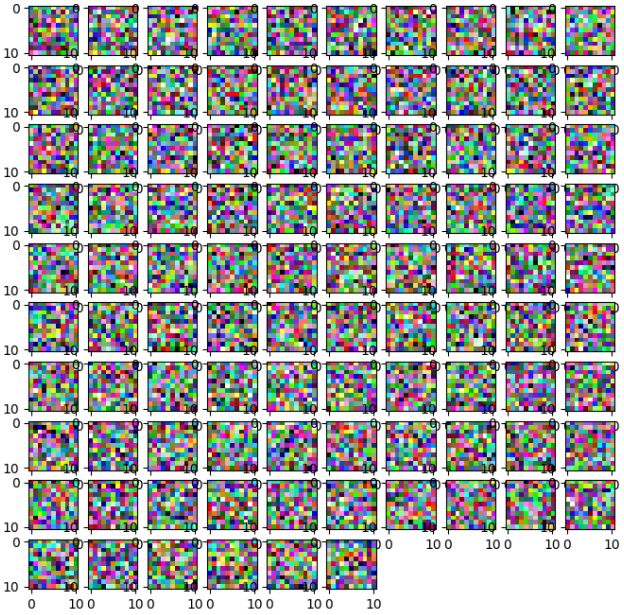


Figure 21: Conv1 filters after 40000 iterations

2. Nearest neighbors

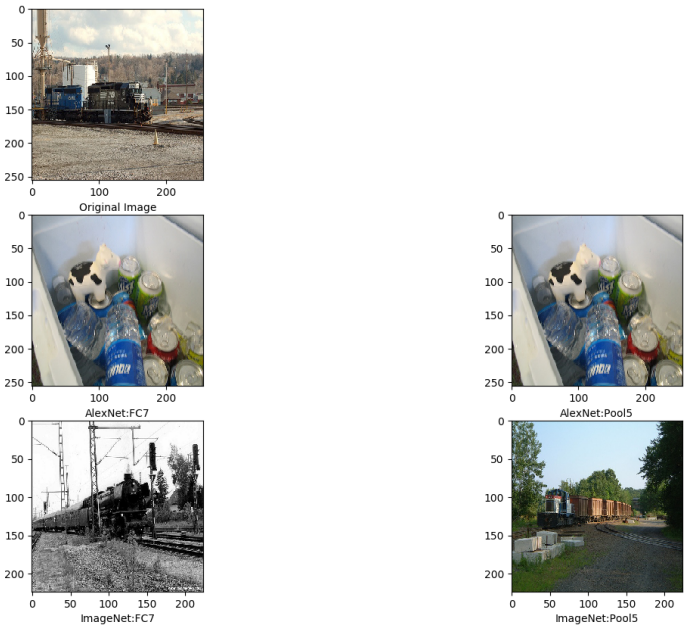



Figure 22: Sample Image 1

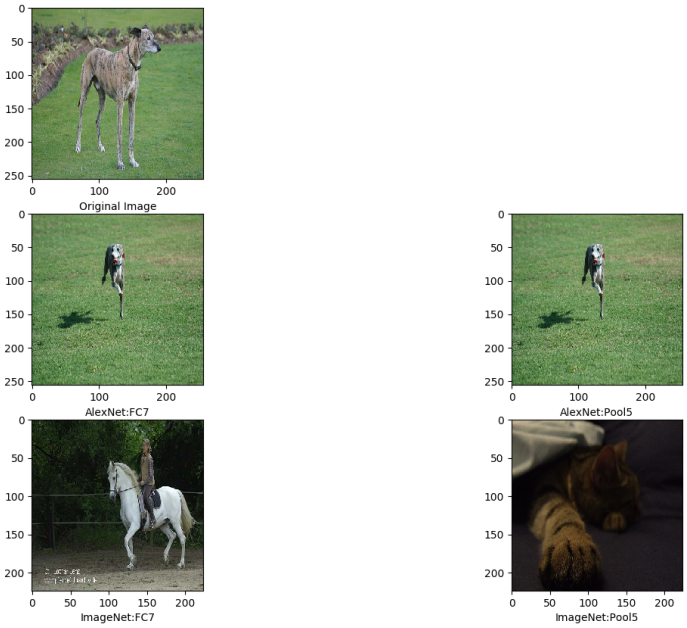



Figure 23: Sample Image 2


Figure 24: Sample Image 3

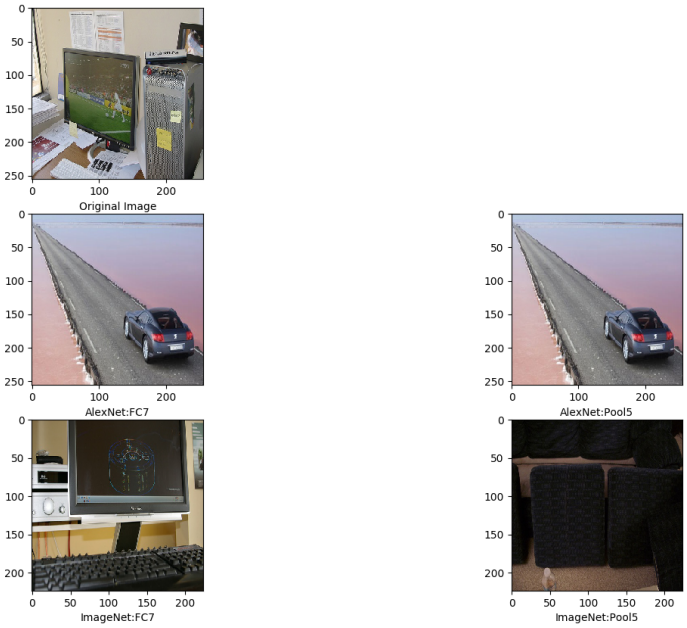



Figure 25: Sample Image 4


Figure 26: Sample Image 5

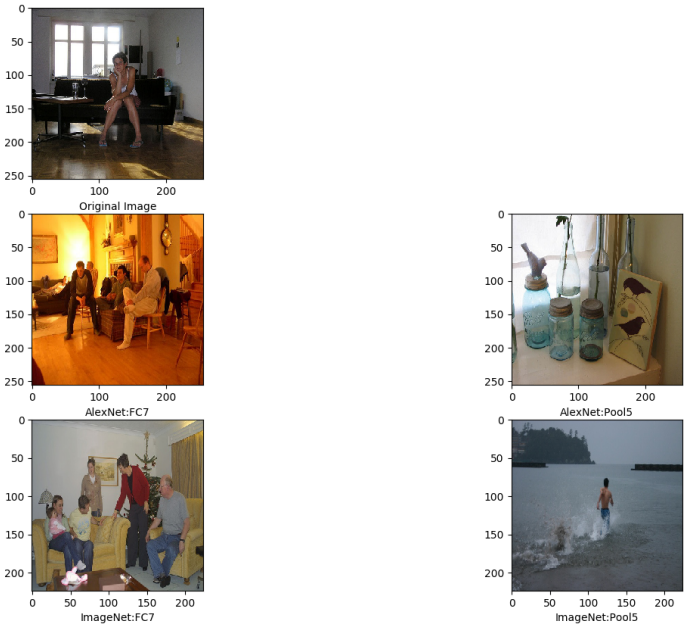



Figure 27: Sample Image 6


Figure 28: Sample Image 7

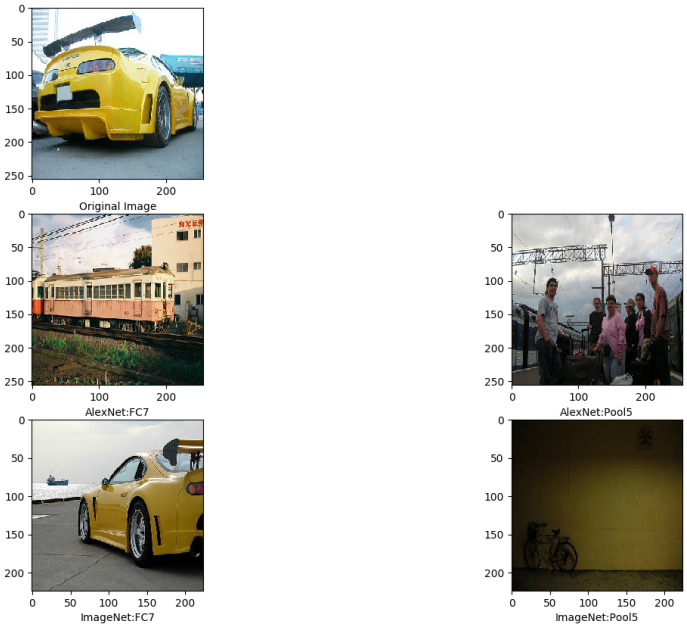



Figure 29: Sample Image 8

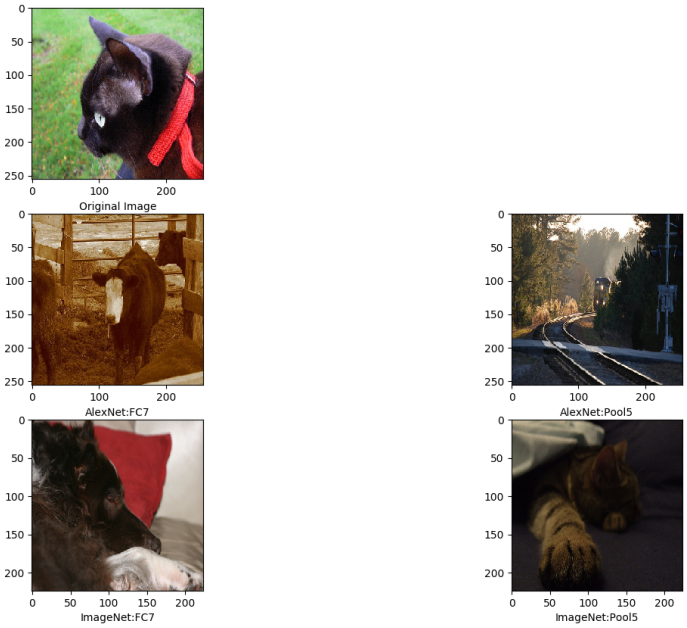



Figure 30: Sample Image 9


Figure 31: Sample Image 10

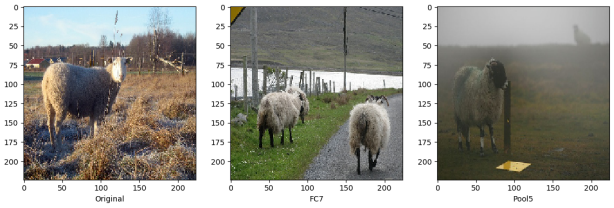



Figure 32: Sample Cases where IMAGENET perfectly predicted the class even with different size, orientation, view and background of the object

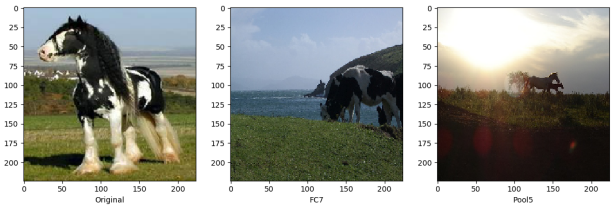



Figure 33: Sample Cases where IMAGENET perfectly predicted the class even with different size, orientation, view and background of the object

**Inferences:**
1. VGG Pre-trained on ImageNet comfortably defeated AlexNet in predicting classes. Example: Sample Image 1, 5, 7, 10
2. FC7 features were more accurate than pool5 features in predicting nearest neighbors. Example: Sample Images 3, 8, 10
3. The neighbors were at times driven by a single dominant feature. Example, a yellow wall was found as a nearest neighbor for many images because of its similarity with the image in terms of color.
4. AlexNet features were disturbed by dominant background colors. In many images, if there was grass in the original image, AlexNet showed random images which had grass. ImageNet was less prone to this.

3. tSNE visualization of intermediate features

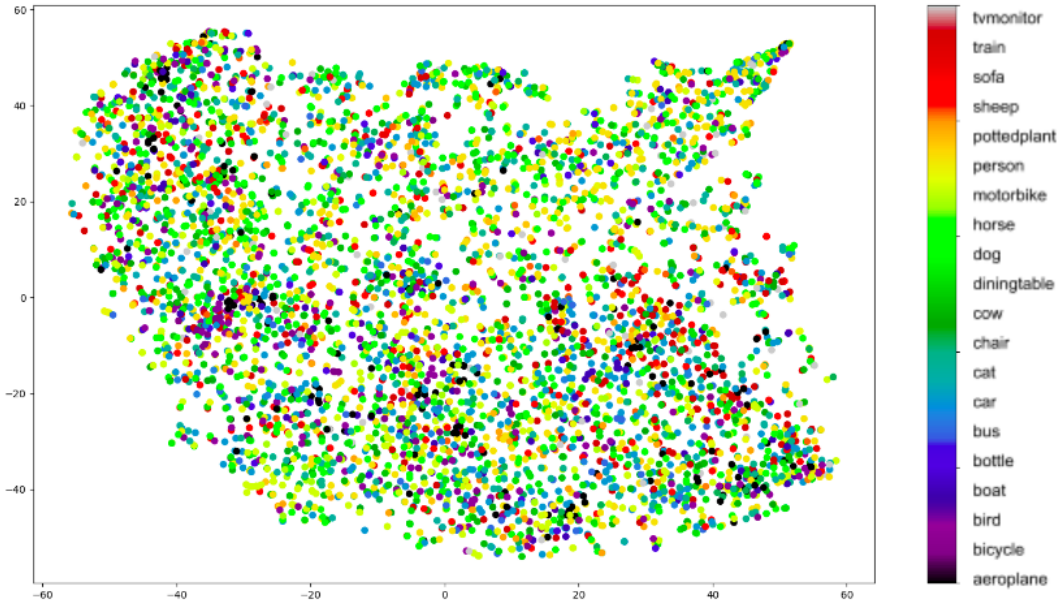



Figure 34: tSNE visualization for fc7 features of AlexNet

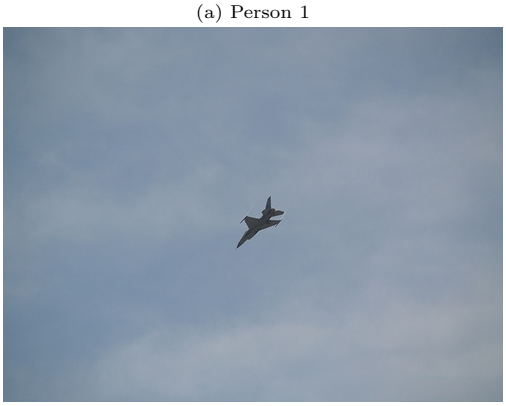
4. Are some classes harder?

**ALEXNET**
**Obtained:** 0.336400994986217 mAP
aeroplane: 0.565
bicycle: 0.308
bird: 0.203
boat: 0.282
bottle: 0.146
bus: 0.218
car: 0.577
cat: 0.292
chair: 0.374
cow: 0.163
diningtable: 0.275
dog: 0.255
horse: 0.586
motorbike: 0.427
person: 0.757
pottedplant: 0.159
sheep: 0.225
sofa: 0.280
train: 0.3871
tvmonitor: 0.239

**PRE-TRAINED VGG**
**Obtained:** 0.661 mAP
aeroplane: 0.860
bicycle: 0.760
bird: 0.785
boat: 0.750
bottle: 0.335
bus: 0.659
car: 0.853
cat: 0.783
chair: 0.543
cow: 0.498
diningtable: 0.516
dog: 0.785
horse: 0.745
motorbike: 0.685
person: 0.934
pottedplant: 0.279
sheep: 0.489
sofa: 0.597
train: 0.817
tvmonitor: 0.555

(a) Person 1

(b) Person 2

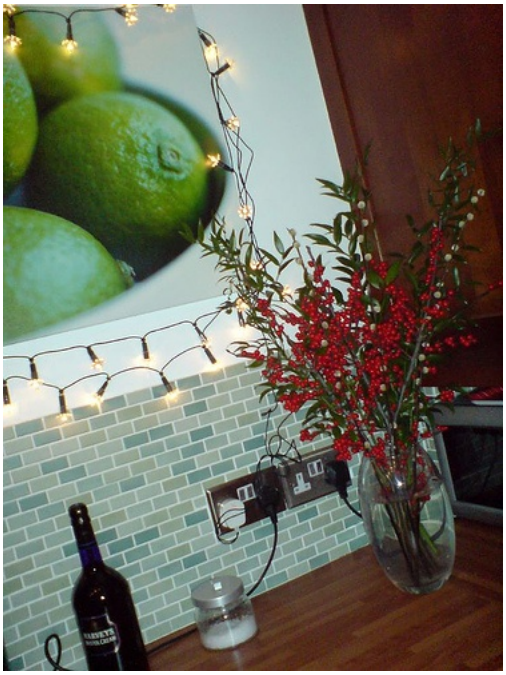


(c) Plane 1

(d) Plane 2

Figure 35: Sample images for **easy** classes in both AlexNet and Pre-trained VGG: Person and Aeroplane

Explanation for easy classes:
Persons: Generally, persons are difficult to predict because there is a lot of different postures that humans can take. But, when I looked at the PASCAL dataset, it mostly contained images of humans standing upright and most of their body visible. As such, it's easier for the network to predict because there is not much variation and person class can be identified by the context obtained from arms, legs, face etc.
Aeroplane: Most of the images of aeroplane are mainly occupied by blue sky. As such, it's easier for the model to classify the image as aeroplane given that there is a lot of blue color in the image.
Also, both the above classes have significant amount of training data.



(a) Bottle

(b) Plant

Figure 36: Sample images for **hard** classes in both AlexNet and Pre-trained VGG: Bottle and Potted Plant

Explanation for hard classes:
Both 'bottles' and 'potted plants' have significant amount of intra-class variation. There is no uniform feature representation that could possibly cover many instance of these classes.

Explanation for increase in mAP due to Pre-training:
Classes which higest gain in mAP due to Pre-training: Bird, Cat Dog, Bus, Train
Reason: When we look at ImageNet data, it has more than 2800k images for animals, 800k for birds and 374k for vehicles. As such, the model was already trained on these classes to learn a great deal of intra-class variation. This is not true for other classes like bottles and plants as they were not explicit classes in ImageNet.