Assignment_2:170668_170511

Abstract:

We were assigned to train a neural net which could classify an image into coarse grained classes and further into fine grained classes condition being that the total number of parameters being learned by the model is less than twice the total parameters of Resnet-18. So, we trained models considering several design principles to scale up convolutional networks and studied them in the context of the Inception V3 and Resnet-34 architecture.
Total number of parameters for
Inception V3: 23.2 M
Resnet - 34: 22 M
Our basic idea was to use pre-trained model trained on ImageNet dataset. It's relevance to our current dataset made it most relevant.

Inception V3

The Inception architecture is designed to perform well even under strict constraints on memory and computational budget. The computational cost of Inception is much lower than VGGNet or its higher performing successors. This only reaffirms the networks' use in big-data scenarios,where huge amount of data needed to be processed at reasonable cost or scenarios where memory or computational capacity is inherently limited, for example in mobile vision settings.

Resnet-34

ResNets are being implemented in almost all of AI's new tech to create state-of-the-art systems.The principle on which ResNets work is to build a deeper networks compared to other plain networks and simultaneously find a optimised number of layers to negate the vanishing gradient problem.

Main Idea for Architecture :

Our basic idea was to use a neural network model, pretrained on a large and relevant dataset (ImageNet) as a feature extractor, and then training layers over it. In our final model, we have fine tuned resnet over 7 convolutional hidden layer, and used it as a feature extractor. The final classification is then performed by using a single linear classifier for coarse grain classification and a 3 layered linear classifier for fine grain classification.

For Coarse Grained Classification

Inceptionv3:

We trained single linear fc-layer with five output classes for classifying into coarse grained classes.
Here, all the layers but one were freezed and then the deep neural net was trained.
On Inception V3 neural net our best accuracy was 99.99%

- We started of by using only the fc layer, only changing the no. of output classes. The accuracy was 99%
- We then tried adding a 2 layers and the accuracy soared to 99.97% on test data and 100% on validation!

ResNet 34 :

On Resnet 34, the best case accuracy was 99.23%

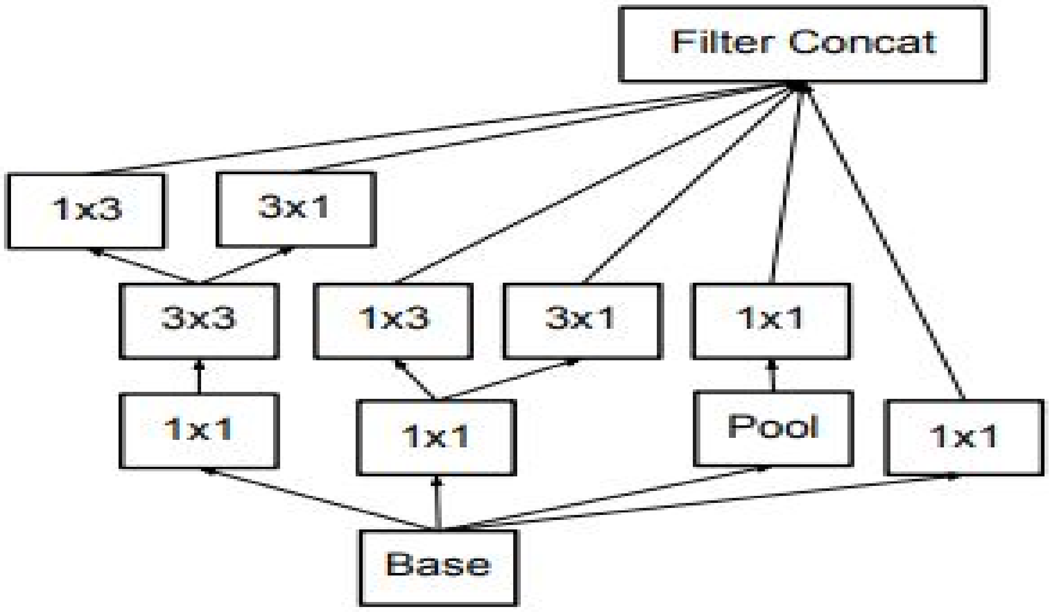
- We started off in a similar fashion as inception v3, getting a maximum accuracy of 94.83% over the best start learning rate (We used a lr scheduler for this and started with 15 different values ranging from 0.002 to 0.005, getting the best acc. At lr=0.002)
- We then tried adding a few layers on top(1 to 4) The results show marginal increase in output
- We then unfreez a few layers from the network itself and tried fine tuning it over various learning rates and epochs
- We got the best accuracy of 99.23% on unfreezing 2 end layers (7 convolutional layers) and then training the model with lr scheduler starting @0.01 with a gamma of 0.4 over 200 epochs on a batch size of 32.

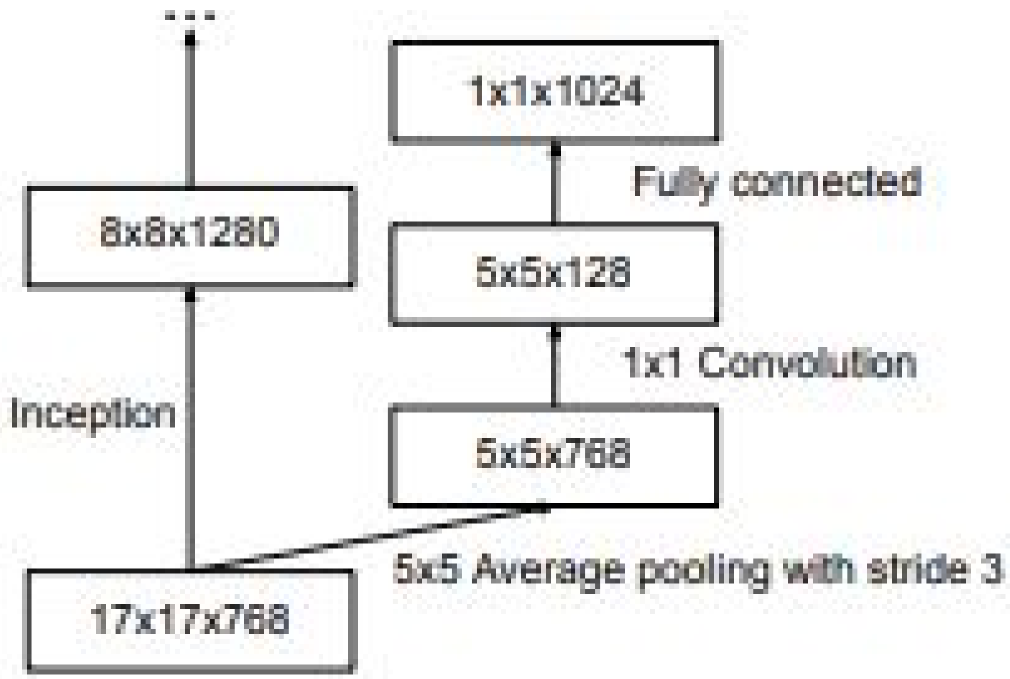
For Fine Grained Classification

We trained two linear fc-layer with thirty six output classes for classifying into fine grained classes.
Here again, initially all the layers but one were freezed and then the deep neural net was trained.
On Inception V3 neural net our best validation set accuracy was 79.78%
And on ResNet 34 neural net our best training set accuracy was 90.24%

As the accuracy were not satisfying so we experimented with various techniques on top of the things used for coarse grain classification.
Primarily, we tried tuning the final classifier layers for the classification.

- After getting good and consistent coarse grain classification for the last resnet model, we tried to classify the fine grained model with a single classifier. This gave an accuracy of 73%.
- We then tried increasing the layers for the classifier. With a linear classifier, the accuracy turned out to be the best for this configuration, coming at 86.78%. (We tried with 3 hidden layers, but the improvement was very marginal)
- We also tried fiddling with various weight functions like ReLU, Leaky ReLu, Sigmoid and ReLU6 for the classifier. It turned out that the linear classifier worked the best for our purpose, at least with the no. of hidden layers being 1, 2 and 3. The different configurations showcased the best training/validation accuracy for various weight functions as 83.88/89.42, 81.65/87.32, 71.22/73.44 and 83.12/84.54 respectively
- We also tried to add a single differently weighed layer(ReLU, Leaky ReLU, Sigmoid and ReLu6) as the first and second hidden layers too. For strange reasons, the accuracy in that case fell from both of the earlier cases. We still don’t understand why!





.
INCEPTION V3 MODEL