Exercise 2

Akarsh Gajbhiye, akarshgajbhiye@gmail.com,

Introduction

The given folder contains implementation of Transfer Learning Networks and Model like VGG16, ResNet50, ResNet101 in Keras/Tensorflow for image recognition tasks . These models were originally developed for ImageNet Image Recognition challenges, they specialise in image recognition tasks ,and are known called **Convolutional Neural Networks** ,
Convolutional Neural Networks can learn extremely complex mapping functions when trained on enough data., these models are can be understood as deep neural networks that recognises finer details at initial layers and deeper details and relations at later layers . They are based on Convolutional Neural Networks .Basically the training of a CNN involves, finding of the right values on each of the filters so that an input image when passed through the multiple layers, activates certain neurons of the last layer so as to predict the correct class.
Though training a CNN from scratch is possible for small projects, most applications require the training of very deep CNN's takes huge amount of data processing and computational power. That's where **transfer learning** comes into play. In transfer learning, we take the pre-trained weights of an already trained model(one that has been trained on millions of images belonging to 1000's of classes, on several high power GPU's for several days) and use these already learned features to predict new classes.

The advantages of transfer learning are that:

1: There is no need of an extremely large training dataset.

2: Not much computational power is required.As we are using pre-trained weights and only have to learn the weights of the last few layers.

There are several models that have been trained on the image net dataset and have been open sourced.For example LeNet, VGG-16, VGG-19, Inception-V3, ResNet etc.

Implementation

For scraping images from www.ikea.com , I used bs4 library in python . These scripts can be found in /dataset_generator .
I have implemented VGG16 , Resnet50, ResNet101 in Keras , these can be found in Models folders . The weights for VGG16 and ResNet50 were downloaded directly using keras . However there were no official weights available for ResNet101 on Keras website so I took the libery of using weights from
https://gist.github.com/flyyufelix/65018873f8cb2bbe95f429c474aa1294 .
The implementation of ResNet101 and ResNet101v2 were inspired from @flyyufelix ‘s repository .
The VGG16 model is Keras implementation of https://arxiv.org/abs/1409.1556 , available on Keras itself .
The ResNet Models are Keras implementation of https://arxiv.org/abs/1512.03385 , whereas ResNet101_v2 is based a modified version of ResNet101 , where fine tuning on ResNet101 is done from layer 5 of original ResNet ,, the Layers 1-4 were frozen with theirs weights set as weights imported from link given above and further training was done on the remaining last layers . The networks used codes in
Validation scores of different models:

Model		**Validation Accuracy Score**
● **VGG16:**		**0.0225**
● **ResNet50:**		**0.25**
● **ResNet101:**		**0.2**
● **ResNet101_v2 :**	**.299 =**	**0.3**

These scores are not good for image recognition tasks , but we see a gradual progression in Validation scores as we move towards more complex models .
All the computations were done using Tensorflow-gpu/Keras-gpu on Nvidia 940M 4GB graphic driver . Further improvement in scores can be seen if trained using larger dataset and better GPU .