6th September

1) Dataset creation

Training Set: http://www.pascal-network.org/challenges/VOC/databases.html
Test set was downloaded from ImageNet database by storing the urls in a file then using the Following command

```
wget -i url_list.txt
```

I also created a text file storing the image names along with its class labels.
Then following command creates an lmdb file which will be fed to GoogLe Net for feature extraction.
#assuming the VOC database is extracted to /home/username/caffe/data/

```
./build/tools/convert_imageset --resize_height=256 --resize_width=256
./data/VOC2005_1/PNGImages/cars_and_bikes/
./data/VOC2005_1/PNGImages/cars_and_bikes.txt cars_and_bikes
```

The same is also applied to the test set.

2) Feature Extraction

The model I used is from the following link.
http://vision.princeton.edu/pvt/GoogLeNet/ImageNet/
I compared feature extraction with caffe built in GoogLe Net with this version, and it seems performances are near each other.
One important thing to note is that, above model has a version trained with places dataset which might increase our chances of detecting pools, assuming that the extracted features are more appropriate to outdoor scenery.

After copying the weights and model files into caffe directory, the following command can be run to extract features, **without labels**.

Note that the training dataset is called training_lmdb.

```
./build/tools/extract_features.bin imagenet_googlenet.caffemodel
train_val_googlenet.prototxt cls3_pool training_features X lmdb
```

Where X is the number of batches to be processed. As I set the batch size to 1 in train_val_googlenet.prototxt, X should be equal to the number of images in training set.
training_features is the extracted features from the training set.

The same command can be applied for training set as well, though one should remember to change the `train_val_googlenet.prototxt's` data source parameters to training set.

3) Merging Labels with Extracted Features

Attached is the python code for creating a modified lmdb file that contains labels.
While using the code, one must be careful to not the change the order of the training and the testing data, while feeding them to GoogLe Net, in which case the corresponding labels would not be updated synchronously.(They should be fed to the network as they are shown in nautilus.)

The input files are a lmdb file and a text file that contains only the class labels.
It then modifies the elements of the lmdb file to contain the corresponding class label.

Example usage:

```
python lmdb_creato r.py
```

4) Creating a Network and Training

The model file is attached as well as the solver files.
Example usage of command line interface for training as follows:

```
./build/tools/caffe train --solver=cab_solver.prototxt
```

5) Testing

Command line example for testing is as follows:
./build/tools/caffe test -model cab_train.prototxt -weights _iter_10000.caffemodel -gpu 0 -iterations X

Where X is the number of test images, as the batch size for test data is 1.

Out of 92 images, network classifies only 4 of the images incorrectly, thus having 95% accuracy.