ENEE731/CMSC828G Project 1
Chethan Mysore Parameshwara

Question 1 : Deep Learning on CIFAR-10

For training the cifar-10 dataset, I have used the caffe framework with Alex Krizhevsky's cuda-convnet network. Cuda-convnet has 3 conv layers and 3 pooling layers with Relu.

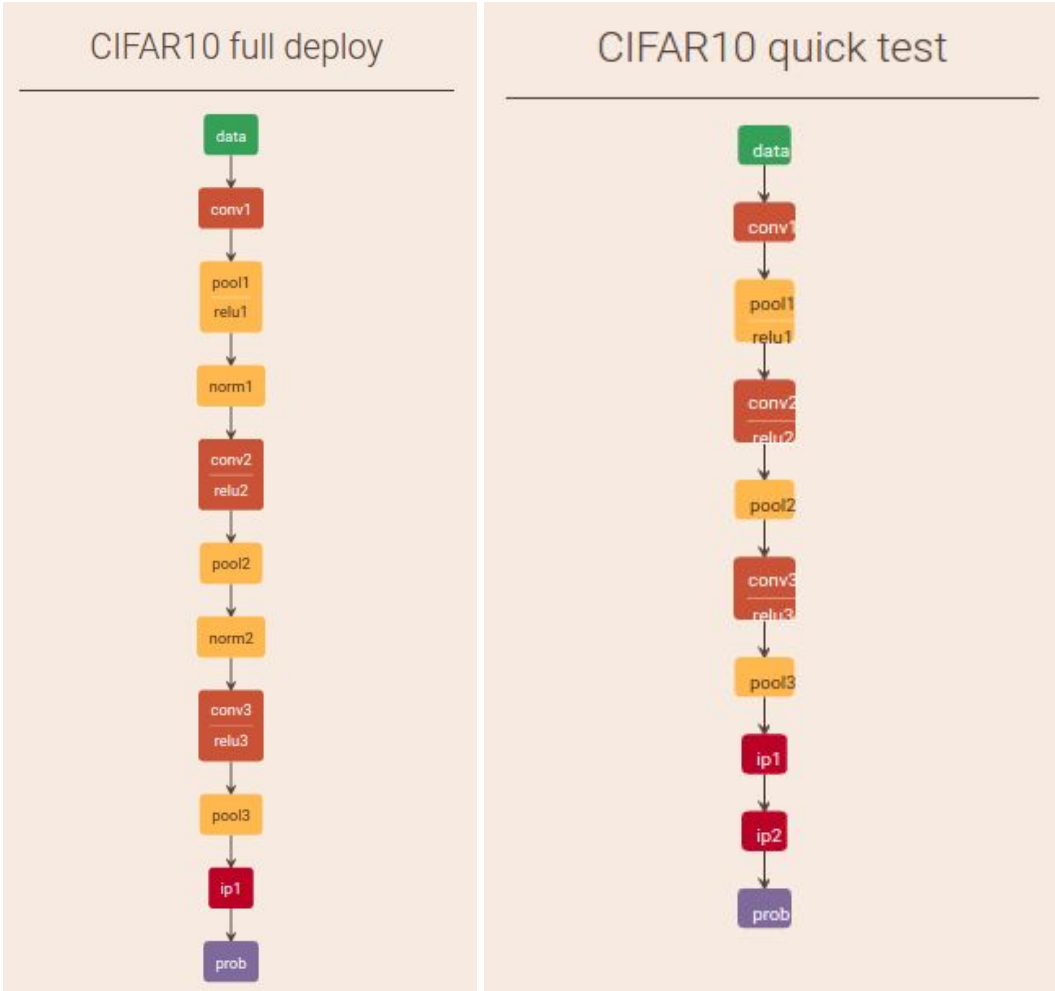

Fig 1 . Network diagram

I ran the training on two variants of cuda-convnet which differs by normalization unit after pooling. 76.95% accuracy was achieved with CIFAR10-quick network and 81.68% was achieved with CIFAR10-full network. Apart from this, training has been performed by varying Solver Prototxt parameters and corresponding accuracies are recorded in the form of table

Exp No.	base_lr	momentum	weight_decay	lr_policy	solver_type	Epoch	Accuracy
1	0.001	0.9	0.004	step	SGD	8	63.63
2	0.001	-	0.004	step	AdaGrad	8	52.21
3	0.01	-	0.04	step	AdaGrad	8	10

4	0.01	0.9	0.04	step	SGD	8	45.83
5	0.001	0.9	0.004	fixed	SGD	8	70.77
6	0.0001	0.9	0.004	fixed	SGD	16	63.46
7	0.001	0.9	0.004	fixed	SGD	40	76.96
8	0.001	0.9	0.004	fixed	SGD	120	81.68
9	0.001	0.9	0.004	fixed	SGD	8	75.38
10	0.001	0.1	0.004	fixed	SGD	8	67.8
11	0.001	-	0.004	fixed	AdaGrad	8	10
12	0.001	0.9	0.4	fixed	SGD	8	42.88
13	0.001	0.9	0.00004	fixed	SGD	8	74.84
14	0.1	0.9	0.004	fixed	SGD	8	10
15	0.00001	0.9	0.004	fixed	SGD	8	53.49

Table 1 : Experiments with varying parameters

Experiments Design :

In solver prototxt, there are different parameters which defines the optimization solver type and its parameters. Among that, I have modified the following parameters and recorded the accuracy.

Base_lr : This parameter indicates the base (beginning) learning rate of the network.

From experiments 4, 5 , 13 and, 14, it can seen clearly that by increasing and decreasing the learning rate from the optimal value accuracy suffers. And, by increasing the learning rate accuracy dropped to lowest value. Time taken by the low learning rate experiment is much higher than the time taken by high learning rate experiment.

Lr_policy : This parameter indicates how the learning rate should change over time.

Options include: "step" - drop the learning rate in step sizes indicated by the gamma parameter.
"multistep" - drop the learning rate in step size indicated by the gamma at each specified stepvalue.
"fixed" - the learning rate does not change.
"exp" - gamma^iteration

From experiments 1 and 5, we can see that fixed learning policy gives the better result than step.

Momentum : This parameter indicates how much of the previous weight will be retained in the new calculation.

From experiments 9 and 10, we can see that by decreasing the momentum accuracy also decreases significantly. And also, momentum is tuned only for SGD not for AdaGrad.

Weight_decay : This parameter indicates the factor of (regularization) penalization of large weights.

From experiments 5, 12 and 13, we can see that by decreasing the weight decay, accuracy also increases significantly from 45% to 70%.

Solver type : This parameter indicates the back propagation algorithm used to train the network.
Options include:
Stochastic Gradient Descent "SGD", AdaDelta "AdaDelta", Adaptive Gradient "AdaGrad" , Adam "Adam", Nesterov's Accelerated Gradient "Nesterov", RMSprop "RMSProp"

From the many experiments on cifar-10, SGD works better than AdaGrad.

Epoch : From experiments 7 and 8, we can conclude that accuracy will improve upon increasing number of epochs. But certainly, time taken by the high epochs is significantly higher than low epoch experiments. This would not always help because it may overfit the model.

Question 2 : LBP with SVM classifier (Source code is included with the report)

In this section, I used python scripts and MATLAB toolbox to extract LBP features and LIBSVM python library to classify using SVM.

Initially, I wrote python script to extract lbp feature where I considered lbp scores for each pixel without rotation invariance and formed one histogram for whole image without considering cell. I got the accuracy of 26%. The feature dimension was [1 256].

Further, with aforementioned feature extractor python script I included rotation invariance option. The accuracy improved to 36%. The feature dimension was [1 59].

At last I used MATLAB lbp feature extraction from CV toolbox where I set the parameter radius to 1 and neighbourhood to 8 pixels with [16 16] cell size and also considered the rotation invariance. The accuracy rose to 43.08%. The feature dimension was [1 236].

It is evident from experiments that performance of deep features are much better than hand engineered features. Also, much more controllable through tuning parameters.