In this assignment you will train and test a convolutional neural network for image classification using the CIFAR-10 dataset. We will use a variation of the VGG-network proposed in Very Deep Convolutional Networks For Large-scale Image Recognition (Simonyan, Zisserman; 2015).

Instructions

- Download the CIFAR-10 dataset from `http://www.cs.toronto.edu/~kriz/cifar.html`
- Using tensorflow, train a convolutional neural network whose convolution filters are all of size 3×3. The overall structure of the network is as follows:
  (a) **CONV1**: convolutional layer with 3 inputs (RGB), 64 outputs (filter size is thus $64 \times 3 \times 3 \times 3$)
  (b) **POOL1**: 2×2 max-pooling layer
  (c) **CONV2**: convolutional layer with 64 outputs, 128 outputs (filter size $128 \times 64 \times 3 \times 3$)
  (d) **POOL2**: 2×2 max-pooling layer
  (e) **CONV3**: convolutional layer with 128 inputs, 256 outputs
  (f) **CONV4**: convolutional layer with 256 inputs, 256 outputs
  (g) **POOL3**: 2×2 max-pooling layer
  (h) **FC1**: fully connected layer with 256 inputs, 1024 outputs
  (i) **FC2**: fully connected layer with 1024 inputs, 1024 outputs
  (j) **SOFTMAX**: softmax layer for classification: 1024 inputs, 10 outputs
- For all convolution layers, stride S = 1, padding P = 1
- All layers, except for the pooling layers and for the last (softmax-)layer should use ReLU-nonlinearities.
- Train the network using Adam with momentum using 45000 randomly sampled examples from the training dataset. Use the remaining 5000 examples for validation.
- Use batch-normalization on the last layer activations (immediately before computing the softmax) when training the network.
- It will be necessary to experiment with learning rate and different parameter initializations (Xavier, He etc.) to find settings that are stable and yield good solutions.
- Use early stopping using the validation set with a patience of 5 epochs.
- Your code should support the following options:
  – --lr (initial learning rate η for gradient descent based algorithms)

- --batch_size (the batch size to be used - valid values are 1 and multiples of 5)
- --init (the initialization method to be used - 1 for Xavier, 2 for He)
- --save_dir (the directory in which the pickled model should be saved - by model we mean all the weights and biases of the network)

You should use the `argparse` module in python for parsing these parameters.

- The task is to get an accuracy of 90% on the test data.

Submission Instructions:

- You need to submit the source code for the assignment. Your code should include one file called train.py which should be runnable using the following command:

  python train.py --lr 0.01 --batch_size 20 --init 1 --save_dir <some_dir>

  All other supporting files used for generating plots, etc. should also be placed in the zip file.

- Prepare a report containing the following:
  - A plot of the learning curve showing iterations on the x-axis and negative log likelihood over labels on the y-axis. Make a single plot showing both the training loss and the validation loss.
  - The performance on the test data of the model that performs best on the validation data.
  - The parameter setting which gave you the best results.
  - Write down the dimensions of the input and output at each layer (for example, the input to CONV1 layer is $3 \times 32 \times 32$)
  - Exactly how many parameters does your network have? How many of these are in the fully connected layers and how many are in the convolutional layers?
  - Exactly how many "neurons" does your network have? How many of these are in the fully connected layers and how many are in the convolutional layers?
  - What was the effect of using batch normalization ?
  - Plot all the 64 layer 1 filters in an 8×8 grid. Do you observe any interesting patterns?
  - Discover some interesting neurons in CONV5. One way of doing this is to feed in a lot of images to the convnet and see if a particular neuron gets excited by similar kinds of images (say, by all car images). For such neurons trace back the patch in the original image which is responsible for exciting them and plot these patches. You can plot these patches for 10 interesting neurons. You also need to submit the code which you use for computing this trace back to the image patch.

– Apply guided back propagation on any 10 neurons in the CONV5 layer and plot the images which excite this neuron. The idea again is to discover interesting patterns which excite some neurons.

GPU Usage:

– If you write the code efficiently you should be able to run 1 epoch in 15 minutes on a CPU.

– If you want to use GPUs then you can register on AWS Educate [1] as a student using your iitm email id. Mention your University Name as "Indian Institute of Technology Madras" (no abbreviation). On registering you should get a credit of 100$. So each team will have 200 $. You can use these credits to rent EC2 GPU instances on the AWS cloud. Use these credits wisely as you will need them (even more) for the next assignment.

– Extra 5% marks for the first team who sets up an EC2 GPU instance on AWS and posts detailed instructions about how to do it. I recommend the P2.xlarge instance [2] which gives you access to 1 GPU at 0.90$ per hour. You should be able to run many experiments in parallel on one GPU so figure out how to use it wisely.

[1] `https://aws.amazon.com/education/awseducate/`

[2] `https://aws.amazon.com/ec2/instance-types/p2/`