Homework 6.

Due on Gradescope by 5 PM, Saturday, May 22

Submission format. Please submit the programming problem through **google colab notebook** as you have done in the previous homeworks. Please submit your notebook in **.ipynb** format instead of PDF on Gradescope. Students are recommended to organize their code into modular cells with proper comments for ease of readability and follow the same steps for submitting the colab notebook as done in previous homeworks. You will see a separate assignment on gradescope for Programming of this homework – be sure to submit each problem to the right assignment on gradescope.

Theoretical Part

Deep neural network design choices:

1. You are asked to design a deep learning model for a binary classification problem to discriminate cat and dog images. Which activation function would you choose for the last (output) layer? Explain your reasoning clearly for full credit. Activation function choices:

   (a) Softmax

   (b) Sigmoid

   (c) ReLU

   (d) Linear

2. Let us consider a deep learning model for a estimating airfare from Santa Barbara to San Francisco. Which one of the following loss functions is most suitable for this task? Choose one and explain your reasoning.

   (a) Cross Entropy

   (b) Hinge Loss

   (c) Mean Absolute Error (L1 loss)

   (d) Mean Squared Error (L2 Loss)

3. How can you find out that your model is overfitting? Suggest a way to solve this issue.

Programming Part

1. **A deeper CNN architecture:** You have trained a two-layer CNN on the CIFAR10 dataset for different configurations in the previous homework. In this homework,

you will implement a deeper CNN architecture and train it on CIFAR10 dataset using PyTorch. The model architecture is shown in Fig. 1. To begin with, implement the network by choosing *convolutional block without batch normalization* (as shown in Fig. 2) for the parts of network labeled *block* 1 through **N** in Fig. 1. Start off with $\mathbf{N} = \mathbf{10}$. Next, do another training where you replace *block* with *convolutional block with batch normalization* (as shown in Fig. 3). For this experiment as well, use 10 as the number of blocks ($\mathbf{N} = 10$). Therefore, you will train two deep CNN architectures, both containing a total of 23 convolutional layers with the specified parameters - one with and another without batch normalization. Aim for a target test accuracy of around 55%.

(a) Plot Train and Test accuracy curve versus the number of iterations for both the training runs: one with and another without batch normalization.

(b) **Optimal value of N when training without batch normalization:** What trend do you see for test accuracy when training the model *without* batch normalization for N = 10? What happens if N is reduced? Find the optimal value of **N** in Fig. 1 that shows an improvement in test accuracy for this experiment (you may have to train the model a few times with different values of N). Is there an improvement in test accuracy with reduced value of **N** compared to the originally-recommended $\mathbf{N} = 10$? You may realize that deeper networks *without* proper normalization layers won't always provide an improvement in accuracy. How many blocks (Fig. 2) did you finally end up using for the experiment *without* batch normalization to see an improvement in the test accuracy?

(c) Plot 2 misclassified and 2 correctly classified samples for each class for both experiments. In the case of the experiment without batch normalization, use the network architecture containing the optimal number of blocks (based on your answer to the point (b)).

(d) Compare this result with the previous homework, as well as the changes in performance with and without batch normalization, and write your observations.

Note for the programming part:

1. Normalize your input data to be 0-mean, unit variance (hint: you can use **torchvision.transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))** in the list of transforms to be applied during data loading). Remember, you input data should be in range of 0 to 1 to apply this transform.

2. Unless otherwise specified, the bias parameter is absent from a given layer.

3. Other training details: loss = Cross Entropy loss, Optimizer = SGD, learning rate = 0.01, number of iterations = 20000, batch size = 128.

4. Use **torch.manual_seed(4)** and **numpy.random.seed(4)** for both models to set the same seed for random generators.

5. Use **torch.nn.BatchNorm2d(channel_dim)** for batch normalization. A good tutorial on batch norm is given here: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-batch-normalization/.

6. Be sure to set hardware accelerator to GPU so your training completes in reasonable amount of time.

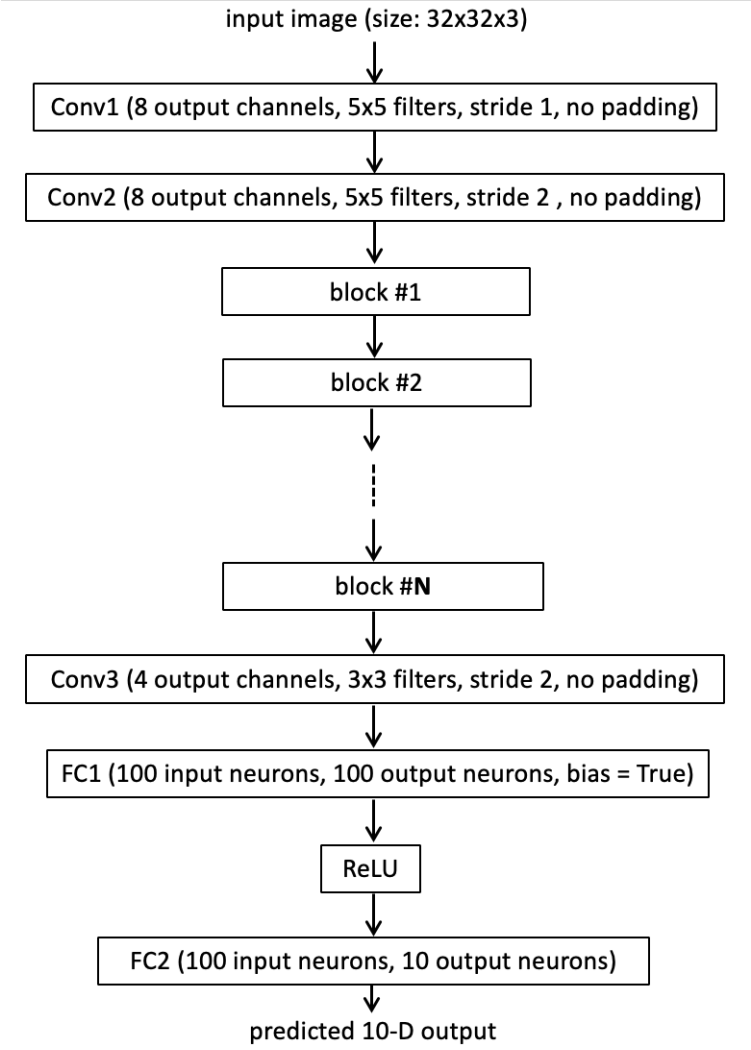

Figure 1: Overall network architecture: the *block* is either a pair of convolutional layers with (Fig. 3) or without batch normalization (Fig. 2). There are a total of **N** such blocks. FC = fully connected layer (linear layer)

.

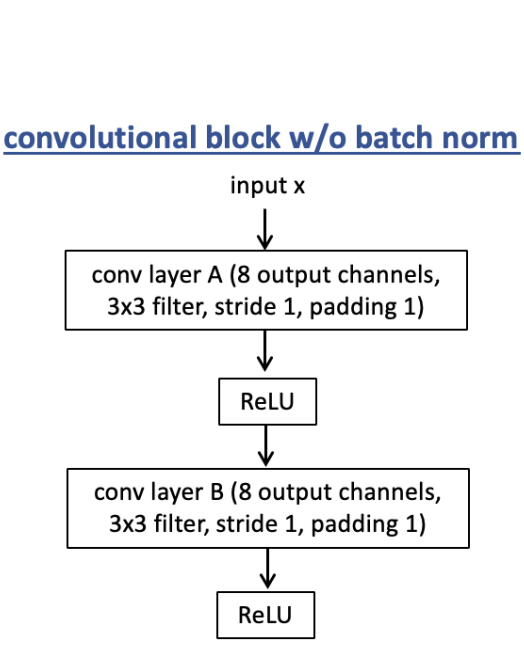

Figure 2: A block of conv layers without batch normalization.

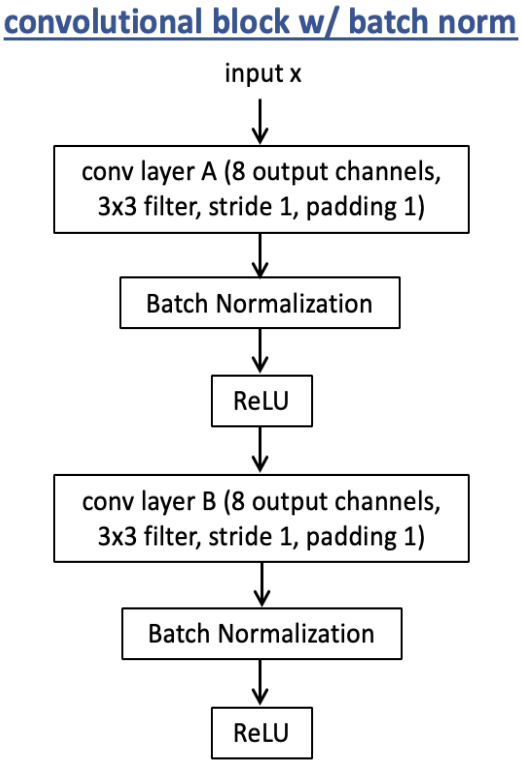

Figure 3: A block of conv layers with batch normalization.