# ***Assignment 2***

# Backprop, Hyperparameter searching, Setup training, Pytorch

Hui Xue

[hui.xue@nih.gov](mailto:hui.xue@nih.gov)

1. **Multi-choice and short-answer questions**
2. Which of the following activation functions can lead to vanishing gradient problem?
3. ReLU
4. Leaky ReLU
5. Tanh
6. Sigmoid
7. You design a loss function , where **.** , . Compute **:**
8. Backprop is a key step in deep learning training. Which of the following statements about backprop are correct?
9. Backprop starts from the input and computes derivatives through all layers until the loss function.
10. For a given layer with given downstream gradient, its backprop has nothing to do with loss function.
11. If two layers have their downstream gradients ready, their backprop can be computed in parallel.
12. It is almost impossible to manually perform backprop for a complex network.
13. Batch is a subset of training samples. Which of the following statements about batch and batch size are correct?
    * 1. We should always use mini-batch, instead of processing whole dataset.
      2. Batch\_size=1 is a bad idea. You should not use it.
      3. Larger the batch size, smaller the learning rate.
      4. When doing the training, batch\_size is a hyper-parameter.
14. After training the model, you got training accuracy of 97%, validation accuracy 87% and test accuracy 77%. What kind of problem are you facing?
    * 1. Bias problem.
      2. Variance problem.
      3. Both bias and variance problems
      4. No problem, I am happy.
15. Draw the computational graph to compute and perform forward and backward pass to compute function value and ,, **.** Suppose, , .
16. Draw the computational graph to compute , where and . Perform forward and backward pass to compute function value and ,,, **.**
17. It is import to understand what to do after diagnosing the bias or variance in the training. Here are a set of actions. Sort them into three bins: actions to reducing bias, actions to reduce variance, actions to reduce both.

Actions:

1. Increase number of layers
2. Increase number of neurons
3. Increase regularization strength
4. Add more data
5. Train longer
6. Add data augmentation
7. Add early stopping to the training process
8. Choose a different network architecture

Actions to reduce bias:

Actions to reduce variance:

Actions to reduce both:

1. In your language, explain what the learning rate scheduler is? Why do we want this component in the training loop? Compared to use a constant learning rate, what are the advantages to use a variable learning rate? Is there a reason to increase learning rate during training?
2. A local shop runs a website allowing customers to put in the review comments for the shop. The owner of this business learned you can do deep learning and invited you to build a model to classify the customer reviews to be positive, negative or neural. They collected a dataset including N=50,000 reviews with classification.
3. Before building the model, it may be a good idea to estimate the bayes error. How will you get a rough estimation of the best error rate?
4. The goal is to achieve 95% accuracy in this task. How will you split the data to train/dev/test sets?
5. After training the first model, you got 87% accuracy on training set and 82% on dev set. On the test set, you only got 67% accuracy. What will be the next steps to improve the model?
6. You found a big dataset (N=1,000,000) of customer review classification from an online dealer. This online dealer has a much bigger inventory than the local shop. Is it a good idea to add this dataset into your training process? If so, how will you do it?
7. After a few weeks, you finally got a model which achieves 96% accuracy in training and 95% in dev set. On the test set, the accuracy is now 92%. What can you do to close the final gap?
8. Your employer wants to learn how the model works before deploying it. You picked a test sample with “negative” class and ran it through the model. The model gave probabilities of (0.1, 0.85, 0.05). You are trying to make this case show the model is minimizing the loss to make prediction. What will be the CE-loss in this case? You then picked another case with “negative” class as the ground-truth. This time model gave (0.65, 0.25, 0.1). What will be the CE-loss for this case?
9. **Pytorch autograd**

It is absolutely essential to understand the dynamic computational graph and autograd in Pytorch. Please read the Pytorch tutorial pages:

[https://pytorch.org/tutorials/beginner/basics/autogradqs\_tutorial.html#](https://pytorch.org/tutorials/beginner/basics/autogradqs_tutorial.html)

<https://pytorch.org/docs/stable/notes/autograd.html>

<https://expoundai.wordpress.com/tag/static-vs-dynamic-graphs/>

You can also search and find more tutorials online about autograd and computational graph online.

1. Please explain the dynamic computational graph and static computational graph. While Tensorflow 1.x uses the static graph as the default, in new version, it has changed to use dynamic graph. What are the advantages of dynamic graphs vs. the static graph?
2. Training DL models can be limited by the available amount of system RAM. This limits the batch\_size one can use during training. One useful method to mitigate the problem is the Gradient Accumulation. Read about this idea (e.g. <https://www.quora.com/What-is-gradient-Accumulation-in-deep-learning-1>). Here are two implementation of this idea. Which one is correct? Explain the reason.

Method 1:

dataset = MyDataSet(batch\_size=10, ...)

optimizer.zero\_grad()

for i, (input, target) in enumerate(dataset):

    pred = model(input)

    L = loss(pred, target)

    L.backward()

    if((i+1)%10==0):

        optimizer.step()

        optimizer.zero\_grad()

Method 2:

dataset = MyDataSet(batch\_size=10, ...)

L = 0

for i, (input, target) in enumerate(dataset):

    pred = model(input)

    L += loss(pred, target)

    if((i+1)%10==0):

        L /= 10

        optimizer.zero\_grad()

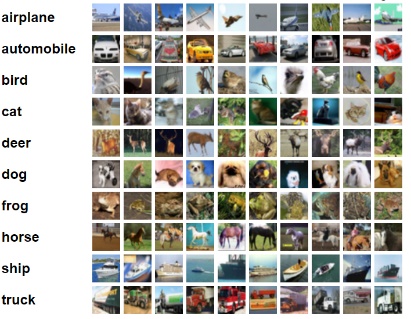
        L.backward()

        optimizer.step()

1. Now, with the knowledge of how the auto-differential mechanism works in Pytorch, please implement the autograd functions in a2\_autograd.py. You can test the implementation by running “pytest -v src/a2\_autograd.py”. **Submit the output of this testing run.**

NOTE: Pytorch and Tensorflow have their own autograd engines. They are designed for deep learning purpose, favoring the derivatives to a scalar function. For more general purpose autograd which has much wider utilities in many fields (far beyond DL), you can check JAX (<https://jax.readthedocs.io/en/latest/index.html>).

1. Now it is time to get into the Pytorch. In the first task, you will implement the same 3-layer MLP network to perform the classification. Instead of working on MNIST, we will work on the CIFAR-10:

 The CIFAR-10 dataset consists of 60000 32x32x3 color images in 10 classes, with 6000 images per class. Typically, there are 50000 training images and 10000 test images.

1. To develop a model with Pytorch, it is useful to utilize the facilities provided in the software package. In particular, the dataset class and data loader. Please review the Pytorch dataset and dataloader : <https://pytorch.org/tutorials/beginner/basics/data_tutorial.html>

Some helping functions have been implemented in the src/util.py, to read and split CIFAR-10 dataset to train, validation and test. Reviewing these functions and using these functions to finish the dataset class in the src/dataset.py for CIFAR10Dataset. Load and plot a mini-batch and submit your plotting, by running “python3 src/dataset.py”.

1. Finish the pytorch MLP model in the *a2\_pytorch\_mlp.py*. You need to finish the model code to set up a N-layer MLP, with numbers of hidden neurons as input parameter, with the ReLU as non-linear activation functions. As learned from the autograd in Pytorch, you need to implement the forward pass for this model. The autograd will compute gradients for all learnable model parameters.

The training loop has been implemented for you. Read and understand the code and run the training with “python3 src/a2\_pytorch\_mlp.py” with default parameters and submit the plotting for training and validation loss and accuracy and report the testing accuracy you get.

If the computer where you train the model on has GPU, you can use GPU by doing

“python3 src/a2\_pytorch\_mlp.py --use\_cuda”. You can also change the parameters and try to get higher accuracy.

1. **Experiment management in DL training**. We are now introducing the concept and tooling for deep learning experiment management.

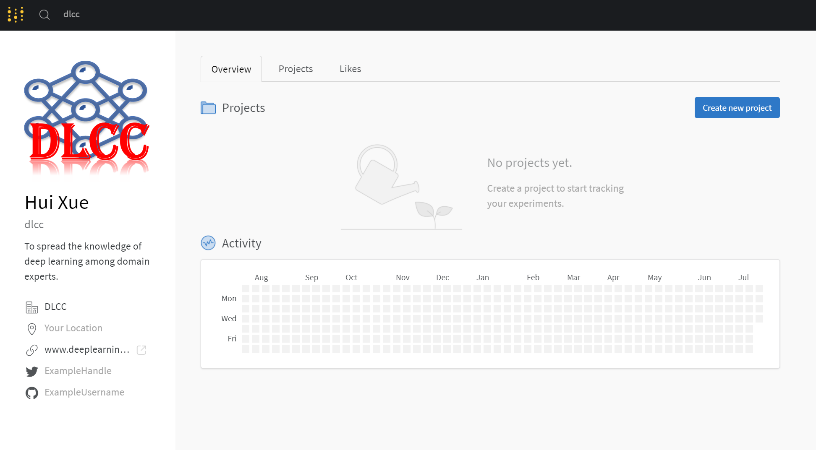
**What is deep learning experiment management?** Suppose you are developing a DL model. With more efforts putting into the R&D, your model grows bigger, better, but more complicated with more hyper-parameters (e.g. number of layers, type of layers, type of optimization etc.). It will quickly become hard to track the parameter settings between different “runs” or “experiments”. Very often, one may make changes to the experiment setup and get a good accuracy, but cannot recover the changes leading to this performance. In other words, without experiment manage, the DL training becomes irreproducible, which is bad engineering practice. Therefore, the experiment management tool is required to help you track, manage, compare and report different deep learning experiments.

Read this post about experiment management: <https://deeplearningcrashcourse.org/A%20quick%20guide%20to%20managing%20machine%20learning%20experiments%20_%20by%20Shashank%20Prasanna%20_%20Towards%20Data%20Science.pdf>

We will use a platform called “Weights and Biases” (https://wandb.ai/site) for DL experiment management. Here is a post to introduce this tool: <https://towardsdatascience.com/introduction-to-weight-biases-track-and-visualize-your-machine-learning-experiments-in-3-lines-9c9553b0f99d>.

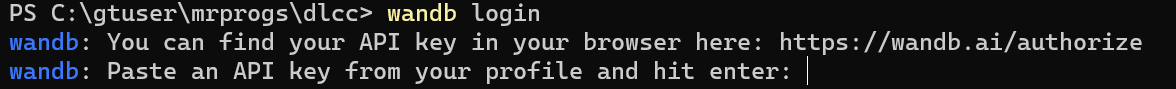
You can also review the tutorial of wandb: <https://docs.wandb.ai/guides/integrations/pytorch>

1. Register an account for yourself at <https://wandb.ai/site> and install wandb in your computer. For example, for ubuntu system, “pip3 install wandb” and “wandb login”. For windows, “pip install wandb”.

As an example, this is the account registered for the course.

You can find the API key in the setting section of your account.

The API key is used for login:

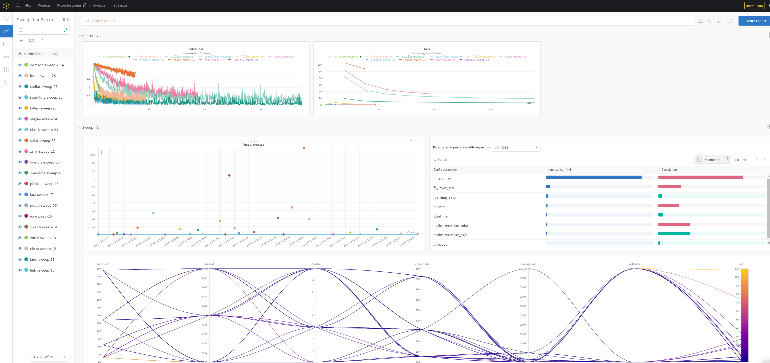


1. Finish the pytorch MLP model in the *a2\_pytorch\_mlp\_wandb.py*. An option is added in the model to use either ReLU or Sigmoid activation function. For the training, an option is added to use SGD or Adam optimization. You can pick two learning rate schedulers: torch.optim.lr\_scheduler.StepLR and torch.optim.lr\_scheduler.OneCycleLR.

Add the support for these new options and add calling to wandb to log **your training parameters, example batch images, and training and validation loss/accuracy in the code**. Train the model with experiment management and submit the screen snap of your training session.

1. One nice feature offered by the wandb platform is the hyperparameter sweeping. It is a function to search through the hyperparameter spaces and find the model with best loss/accuracy, with full automation. Read more about wandb hyperparameter sweep here: <https://wandb.ai/site/articles/introduction-hyperparameter-sweeps>

To define which metric to watch for “the best model” and the hyperparameter space, a dictionary can be supplied. Please check the *sweep\_config* structure in src/*a2\_pytorch\_mlp\_wandb.py.* After logging into your wandb account, you can run the hyperparameter sweep by *“python3 src/a2\_pytorch\_mlp\_wandb.py –sweep”.* After running this for a while, log into your account and check the sweeping results. Submit the screen snap of your sweep results. What is the best accuracy you can get?

An example of hyperparameter sweeping results.