Debugging of Deep Neural Network

11485/685/785 Spring 2023: RECITATION 0G Part 1

Qin Wang

January 10, 2023

Table of Contents

Table of Contents

Common Scenarios

- "My code throws an error and stops running, and I don't understand the lengthy error message."
- "My code runs, but the accuracy is terribly low and not improving."
- "My model is taking forever to train."

Tip 1: Consolidate Hyperparameters

Putting everything in one place helps with model experimentation.
We can save everything in a `config dictionary`.

```
config = {
    'batch_size': 96,
    'epochs': 30,
    'lr': 1e-3,
    # Add more as you need them
    # e.g dropout rate, scheduler parameters
}
```

Tip 2: Write Sanity Checks

Write a **small** test case for each function you write.

- Print the type and shape of important variables
- Slice and print only a segment of high dimensional variables
- Visualize the data using `matplotlib`, we will have a short code demo at the end of this recitation
- Break and print after one iteration

Tip 3: Use a Debug Flag

Motivation

- A switch between:
  - Debugging mode: you want to print extra information
  - Training mode: you don't want to do anything redundant

Tip 3: Use a Debug Flag

Hence, to avoid the trouble, define a boolean `debug_flag`:

```
if debug_flag==True:
    print("debugging information")
```

By grouping all debugging print statements at one place, it also helps you think systematically about what to keep track of.

Tip 4: Try restarting the kernel

If you are running your code in Jupiter Notebook or Google Colab, sometimes you accidentally **ran a cell twice** or you **ran cells in the wrong order**.
Technically there is nothing wrong with your code – just restart kernel and rerun everything from the beginning!

Table of Contents

Syntax Errors

- Stack overflow is your best friend!

- Refer to Recitation 0A - Python & OOP Fundamentals for details

Logic/Math Errors

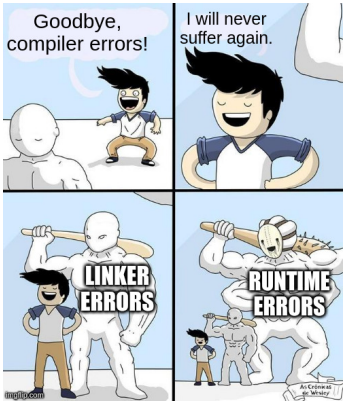
Common in Homework Part 1, in which you are trying to implement `MyTorch`. There will be a lot of matrix manipulation and math.

- Read the writeup carefully
- Check the shape of your variables
- Read numpy documentation of a function for its:
  - Purpose
  - Input type and shape
  - Output type and shape

Runtime Errors

Things to do when seeing Runtime errors

- Read Traceback to find the root of error
- Read library documentation for function specifics
- Set batch size to 1 and run the code on CPU – more readable error messages



Runtime Errors

Things to do when seeing Runtime errors

- Learn to use pdb, an interactive python debugger, we will have a code demo
- **Stack overflow is your best friend!**

stackoverflow

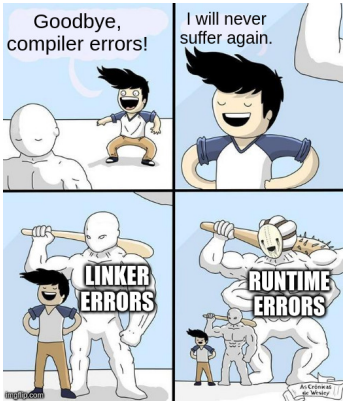


Table of Contents

Time issue

Now my model runs, but it's taking 40 min to train an epoch while only taking others 10 min...

Time issue

Things to check when your training time is unusually long

- Check if you are using GPU
- Check batch size (32-128, as large as your GPU doesn't complain)
- Check your **dataloader** and **training loop**: Most iterations happen there
- Use `mixed_precision` during training
- Use **time** module to check which part of the code is taking unusually long

Table of Contents

Memory issue

Now my model is training normally :D
After 30 epochs:

Memory issue

Common Errors:

If you put too many things on GPU, you will see this:

```
RuntimeError: CUDA out of memory.
```

Things to try:

- Reduce batch size
- Use cuda mixed precision [1]

[1] Read Tutorial before starting on HW P2s:
https://pytorch.org/tutorials/recipes/recipes/amp_recipe.html

Memory issue

Common Errors:

```
RuntimeError: CUDA out of memory.
```

Things to try:

- Check if you used *torch.inference_mode()* during validation and testing:
  - Disables gradient calculation, which is only needed for backward-prop during training
  - Reduces memory consumption
- Call *torch.cuda.empty_cache()* help reduce fragmentation of GPU memory in certain cases.

Memory issue

Common Errors:

Forgetting to move data to GPU for training, validation and testing of the model.

```
RuntimeError: Expected object of device type
cuda but got device type cpu
```

Memory issue

- In order to train a model on the GPU it is first necessary to send the model itself to the GPU:

```
device = "cuda" # GPU
model = model.to(device=device)
```

Memory issue

- In order to train a model on the GPU it is first necessary to send the model itself to the GPU:

```
device = "cuda" # GPU
model = model.to(device=device)
```

- The second requirement for running the training loop on the GPU is to move the training data:

```
x, label = x.to(device), label.to(device)
```

Memory issue

Common Errors:

If you are not careful, there might be a mismatch between the locations of different data being used in a function.

Things to try:

- You can find out which device your tensor data are on at different points in the code by using the device property:

```
print(x_train.device)
```

- To move the data to CPU or to GPU:

```
x = x.to(device="cpu")  # move to CPU
x = x.to(device="cuda") # move to GPU
```