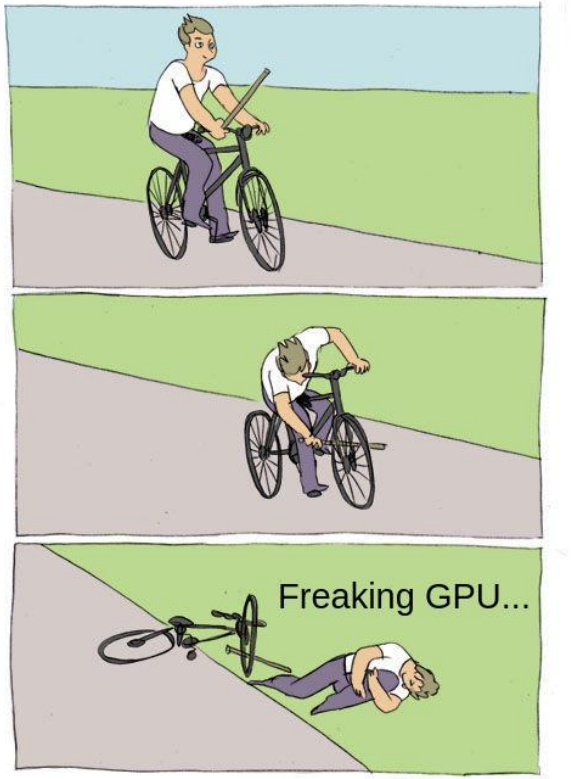
Parallel training of deep ensembles for uncertainty estimation

Gleb Bazhenov, Saydash Miftakhov

2021

Executive Summary



Motivation

Want to estimate an uncertainty of a model in the classification task

- train many models
- average them and measure the entropy of predictive distribution
- the key word is **many**

Models are completely independent!!!

Resources

Used Google Colab

- Widely used
- Pretty slow
- No solutions found

Problem

- Deep Ensemble training:
  - Slow
  - Uses <<100% of GPU
- **Idea**: Start many processes in parallel
- Then we'll use ~100% of resources, we thought

First Attempt

- Doesn’t work… WHY???
- “GPU should work well, CPU is a bottleneck!!!”
- **Idea**: Store data in the main process and broadcast only batches

New Approach

- Uses much less memory
- Still does not speed up the training
- Maybe something’s wrong with GPU?..

Results?

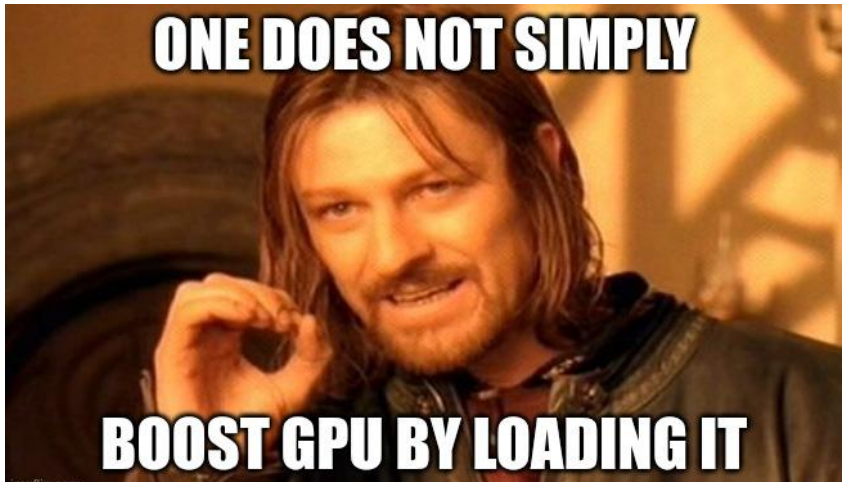
- No speedup in any setting
- No idea where to go
- 13 hours till presentations

The Best Solution Award goes to…

- … the idea “try asking TA”

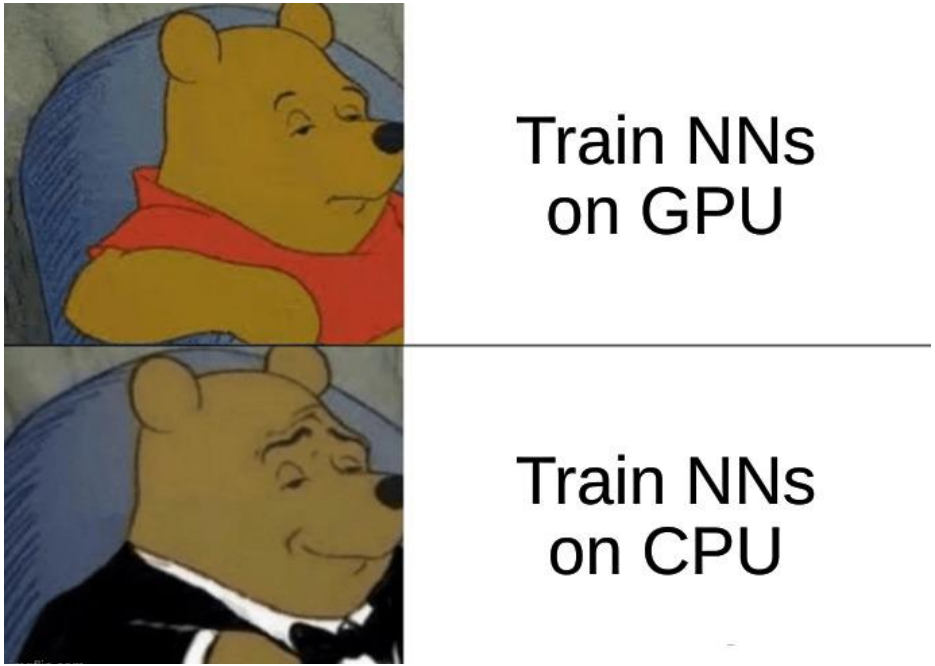
What happened in fact:

- 1 GPU is still 1 core: we just created many tasks and placed them in a queue
- Only way for speeding up: get rid of “friction” along the way



Positive News by CNN

- We have 30% speedup while training on CPU with 2 MPI processes!!! (on 2 CPU cores)
- OK, let’s make a presentation
  CPUs are good
  GPUs are not
- Wait…

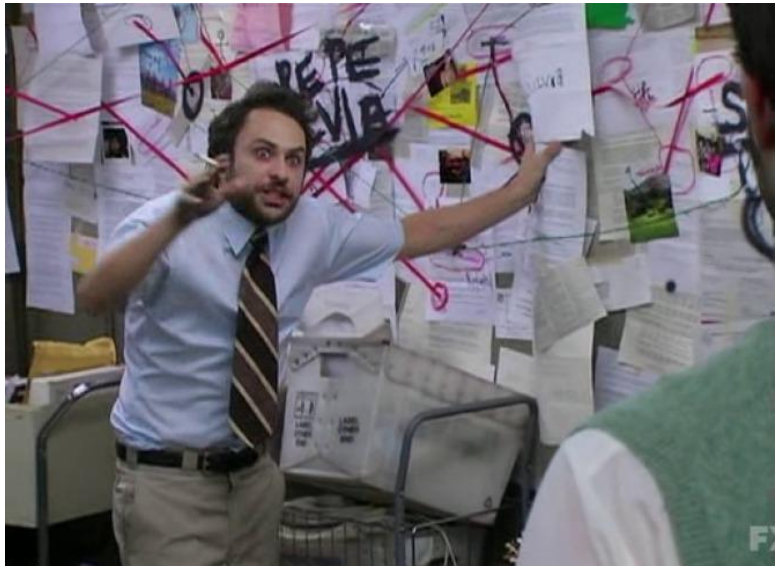


Best Bug-Fix Ever

- Deleted redundant data transfer between processes
- Achieved significant boost in training speed
- Let’s dive into details

Experiments Setup

- PyTorch
- Simple NN for MNIST classification:
  - 2 Convolution layers (conv + maxpool)
  - 2 Fully Connected layers
- Training for 2 full epochs on 60k samples
- MPI used for running parallel processes
- 1 process = 1 model



Results Overview

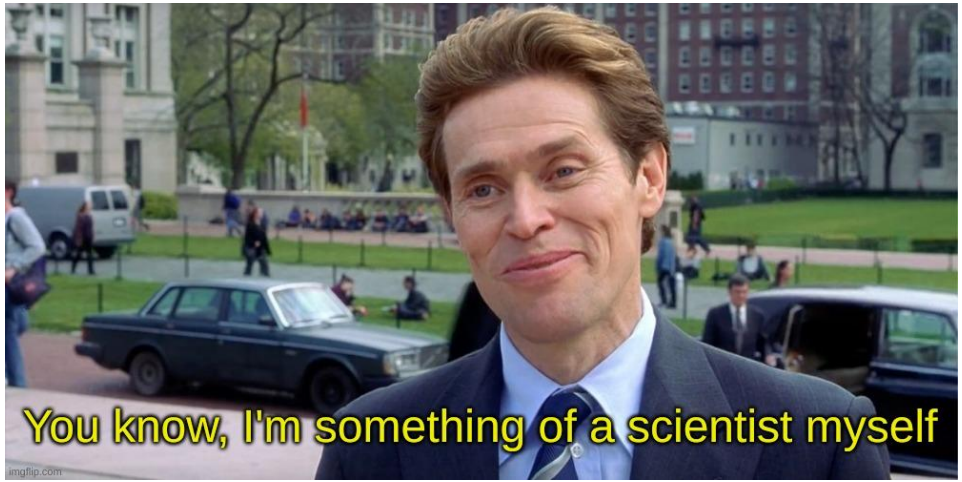
Execution time for different setups:

	Parallel, no communication	Batch broadcasting	No parallelism
2 models on CPU	167.1 seconds	**157.7 seconds**	206.4 seconds
2 models on GPU	25.5 seconds	**24.0 seconds**	37.6 seconds
2 CPU + 1 GPU	**181.4 seconds**	241.2 seconds	224.9 seconds

*Includes time for loading local data, initializing a model, and running a train loop

Summary

- Paralleling speeds up training NNs even in Google Colab
- If you feel lost, contact TAs
- Link to the Colab
- Aaaaand (-->)



Thanks for your attention and patience!