Hyperparameter Tuning & Reproducibility in Pytorch

Presented by Taehee Kim
21.01.14

DAVIAN
Data and Visual Analytics Lab

Hyperparameters

- Model-free hyperparameters
    - Learning rate
    - Batch size per gpu
    - Training epoch
    - Learning rate scheduler(Warm up steps, lambda, step size ...)
    - Optimizer (beta1, beta2 in Adam)
    - Weight initialization
    - Early stop strategy
    - Regularization
    - Dropout
    - Perturbation or noise for an input
    - ...

Hyperparameters

- Model hyperparameters
  - Kernel size
  - number of layer
  - number of hidden units
  - number of embedding units
  - pooling
  - activation function

Hyperparameters

- Model-free & Model hyperparameters
  - Learning rate x Batch size per gpu x Training epoch x Learning rate scheduler(Warm up steps, lambda, step size ...) x Optimizer (beta1, beta2 in Adam) x Weight initialization x Early stop strategy x Regularization x Dropout x Kernel size x number of layer x number of hidden units x number of embedding units x pooling layer x activation function x ...
  - 10 x 3 x 3 x 3 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x ... = 552,960
  - 552,960 x training time for a model = 921 hours

Hyperparameters

- ???
    - Accumulation steps
    - Random seed
    - Number of gpu
    - Number of evaluation

Hyperparameters

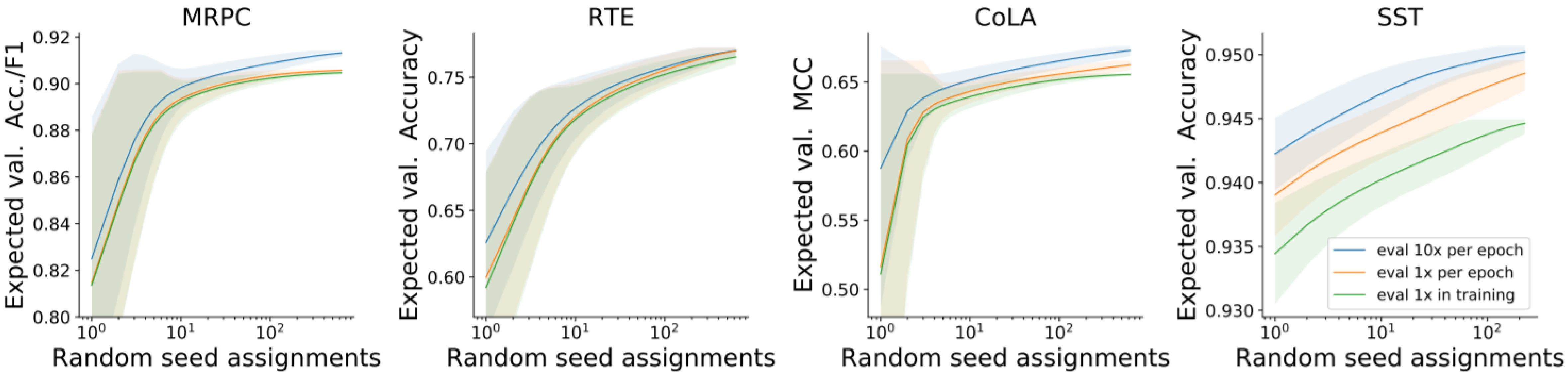
- Accumulation Steps
    - 16 batch with no accumulation vs 4 batch with 4 accumulation step, which can result in better performance ?

Hyperparameters

- Accumulation Steps
  - 16 batch with no accumulation vs 4 batch with 4 accumulation step, which can result in better performance ?
    - Sampled gradient’s standard deviation
    - Gradients based on 4 distributions with specific mean and variance (the number of sample is 4)
    - Gradients based on 1 distribution with specific mean and variance (the number of sample is 16)

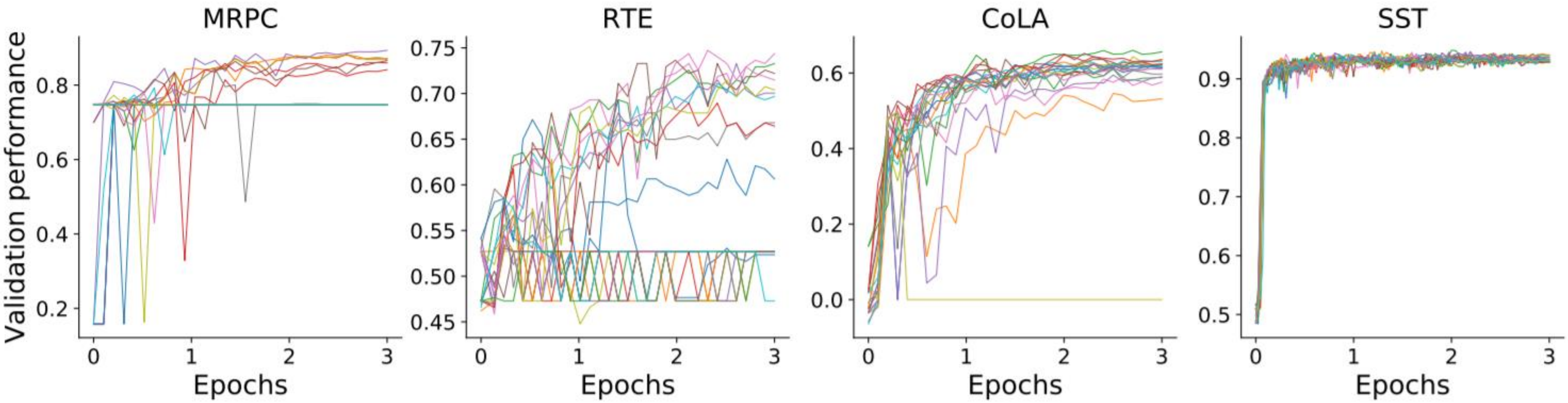
Hyperparameters

- Number of evaluation
  - Expected validation performance as the number of evaluation increases



Hyperparameters

- Random Seed
    - There is a promising seeds
    - These seeds can be distinguished early in training



Hyperparameters

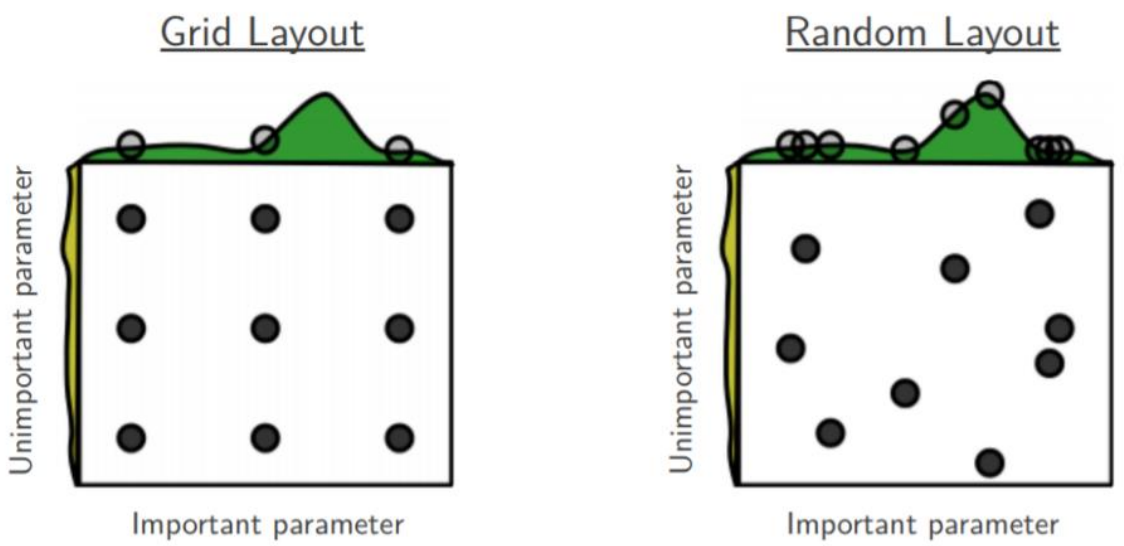
- How to control randomness?
  - random.seed()
  - np.random.seed()
  - torch.manual_seed()
  - torch.cuda.manual_seed() / torch.cuda.manual_seed_all()
  - torch.backends.cudnn.deterministic = True
  - torch.backends.cudnn.benchmark = False
  - [torch.set_deterministic()]
  - if you use CUDA tensors, we need to set the environment variable CUBLAS_WORKSPACE_CONFIG according to [CUDA documentation]

Note: The non-deterministic behavior of multi-stream execution is due to library optimizations in selecting internal workspace for the routines running in parallel streams. To avoid this effect user can either:

- provide a separate workspace for each used stream using the `cublasSetWorkspace()` function, or
- have one cuBLAS handle per stream, or
- use cublasLtMatmul() instead of *gemm*() family of functions and provide user owned workspace, or
- set a debug environment variable CUBLAS_WORKSPACE_CONFIG to ":16:8" (may limit overall performance) or ":4096:8" (will increase library footprint in GPU memory by approximately 24MiB).

Hyperparameters

- Hyperparameters Optimization
    - Grid Search
    - Random Search
    - Bayesian



- Priority
    - For me, 1 tier = Learning rate / 1.5 tier: Batch size
    - Hyperparameters importance are (as for Andrew Ng): Learning rate, momentum beta, mini-batch size, number of hidden units, number of layers, learning rate decay, regularization lambda, activation functions, beta1 & beta2 in Adam

Hyperparameters

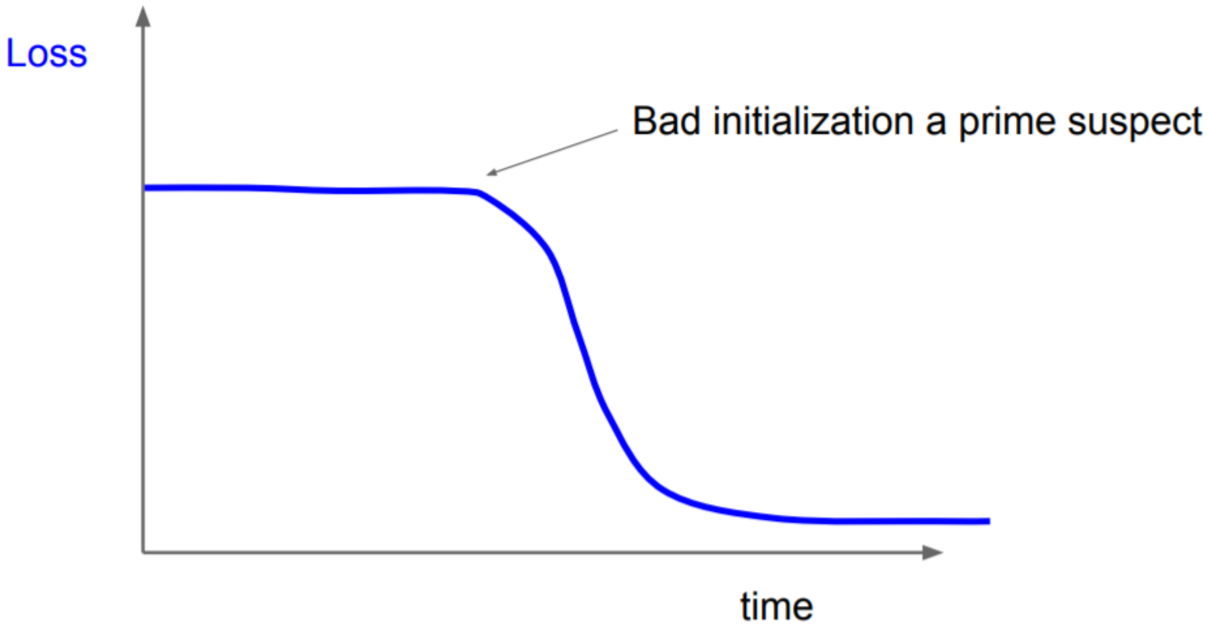
- (Maybe for researchers) Best practice for hyperparameters optimization
  - Learning rate, Learning rate, Learning rate
  - Step 1
    - Turn off learning rate scheduler and train a model
    - Find a learning rate that makes loss low at the early stage of training compared to other learning rates

Hyperparameters

- (Maybe for researchers) Best practice for hyperparameters optimization
    - Step 2
        - Turn on learning rate scheduler and train a model
        - Find a learning rate that makes loss low at the early stage of training compared to other learning rates
        - Random search around the learning rate found in step 1
        - Make possible batch sizes larger

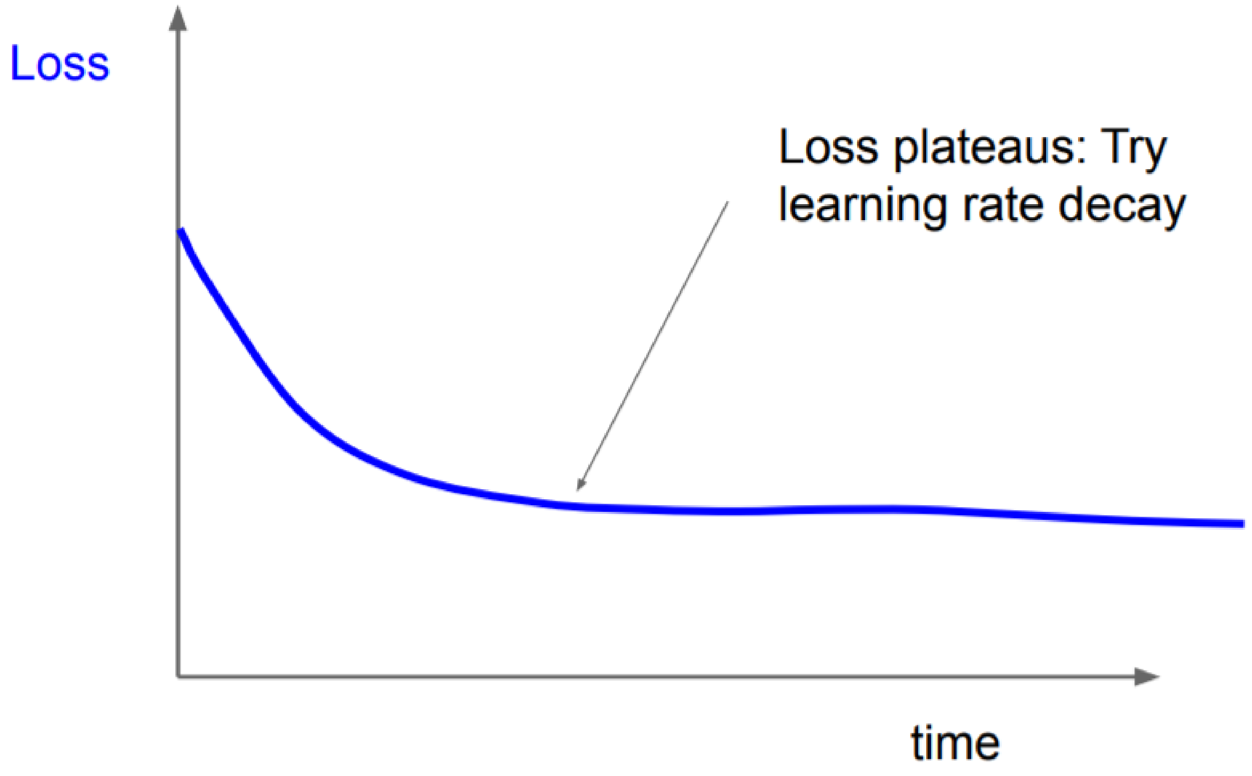
Hyperparameters

- (Maybe for researchers) Best practice for hyperparameters optimization
  - Step 3
    - Look at learning curves



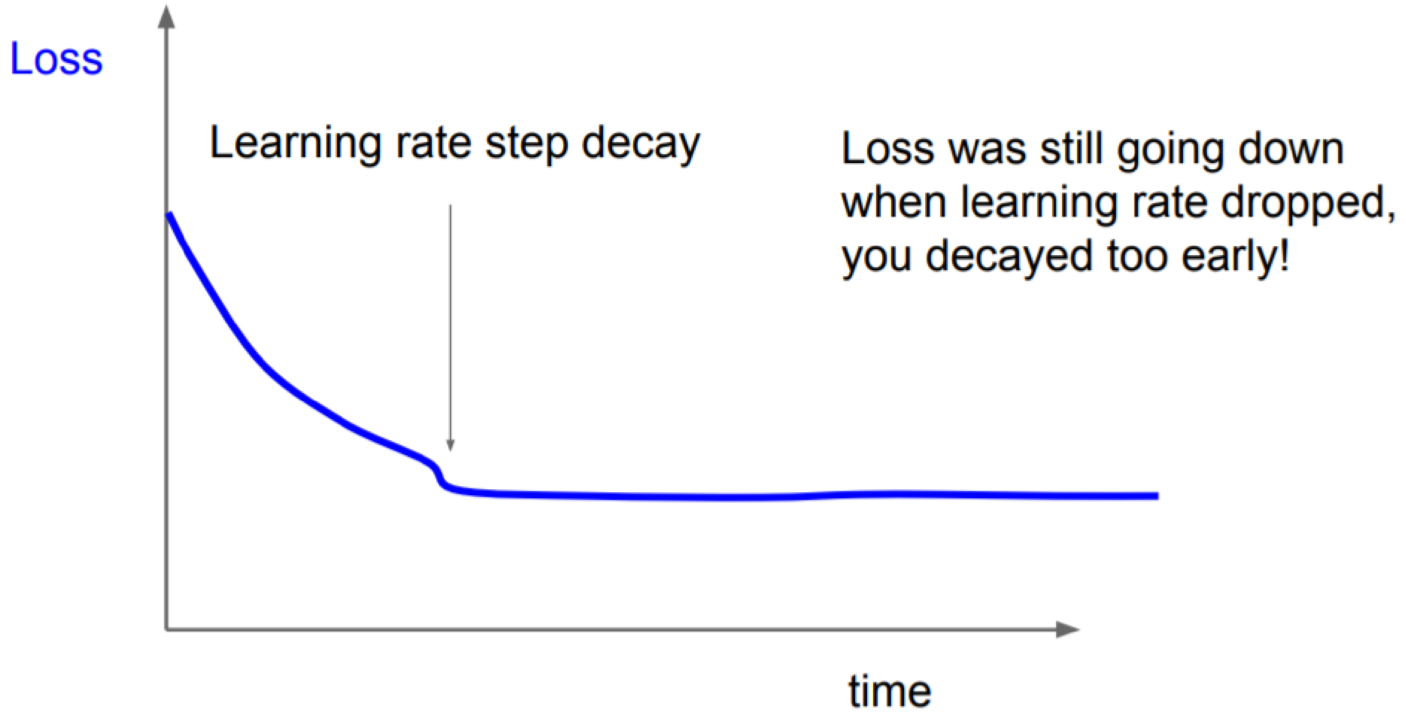
Hyperparameters

- (Maybe for researchers) Best practice for hyperparameters optimization
    - Step 3
        - Look at learning curves



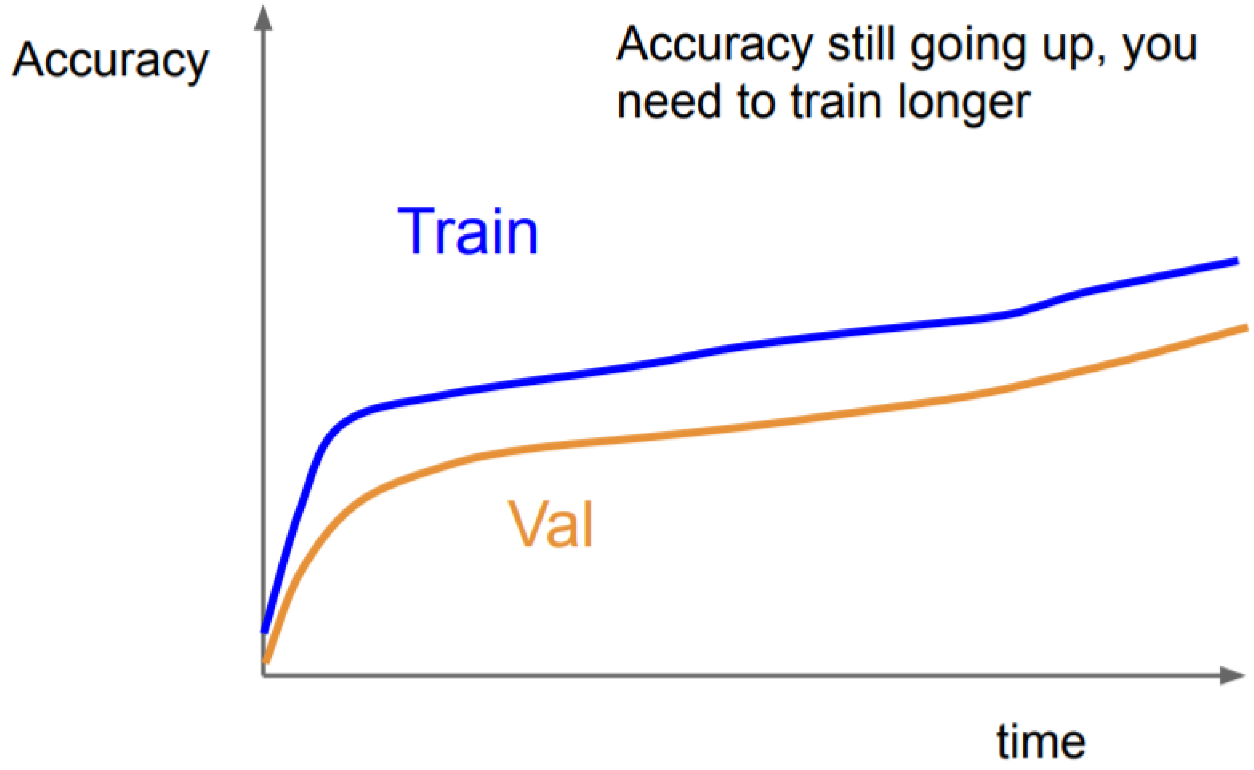
Hyperparameters

- (Maybe for researchers) Best practice for hyperparameters optimization
  - Step 3
    - Look at learning curves



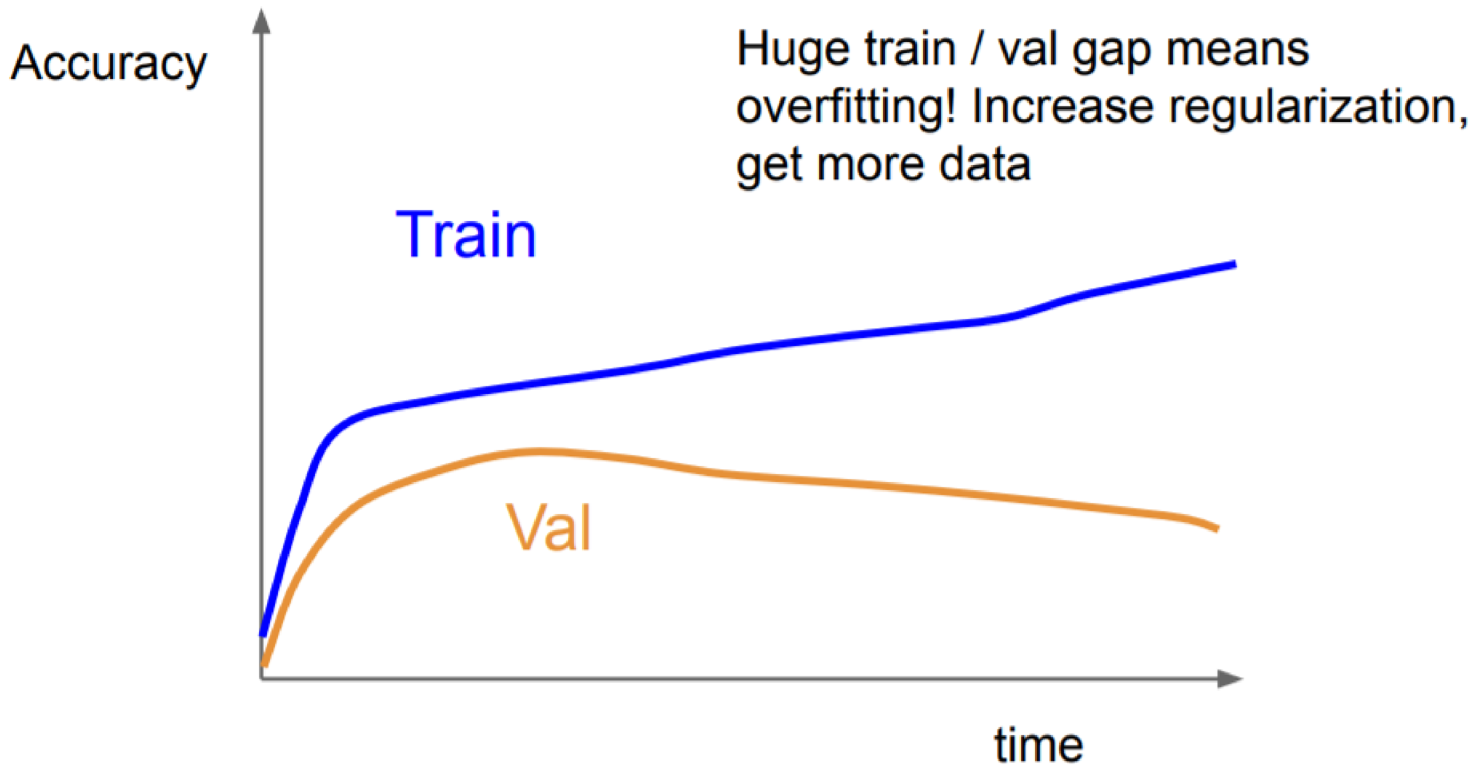
Hyperparameters

- (Maybe for researchers) Best practice for hyperparameters optimization
    - Step 3
        - Look at learning curves



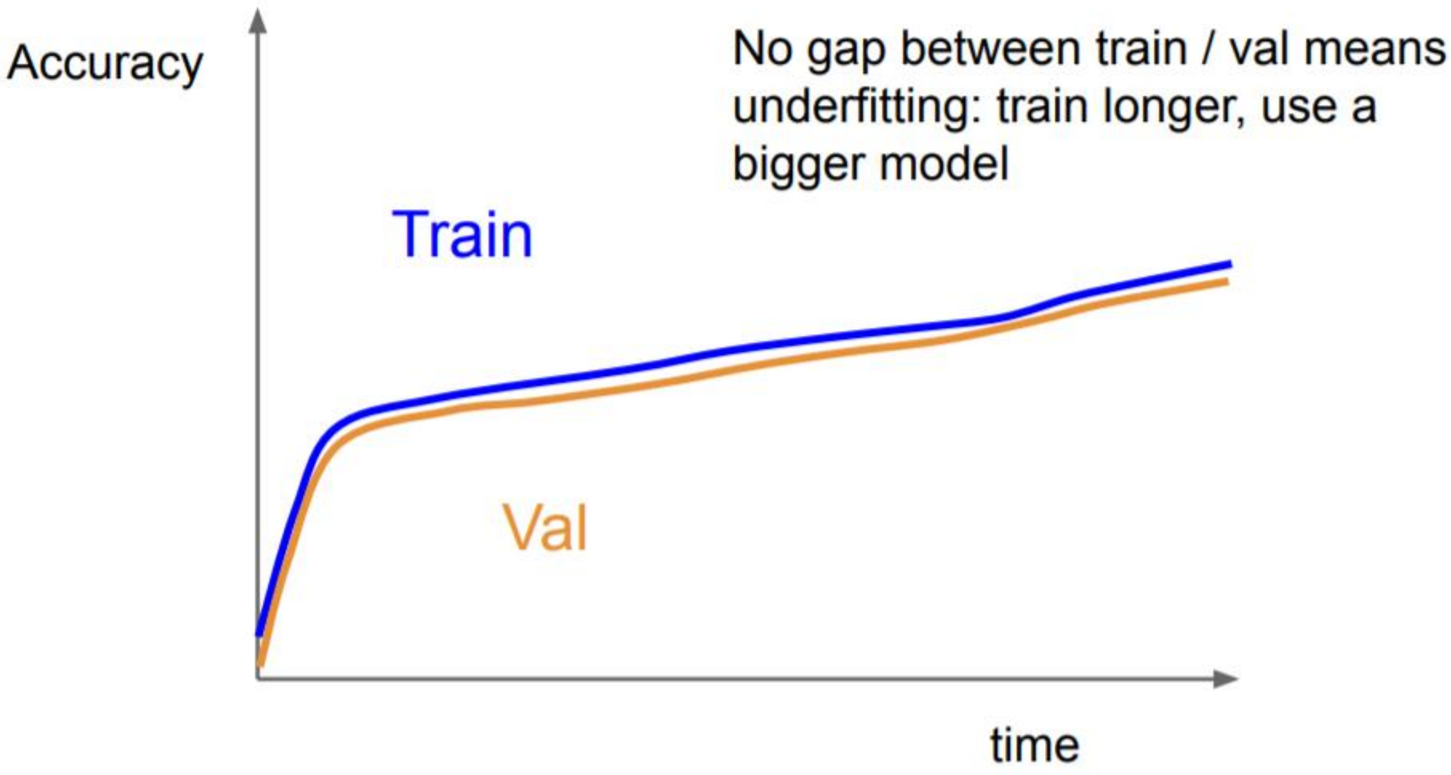
Hyperparameters

- (Maybe for researchers) Best practice for hyperparameters optimization
  - Step 3
    - Look at learning curves



Hyperparameters

- (Maybe for researchers) Best practice for hyperparameters optimization
    - Step 3
        - Look at learning curves



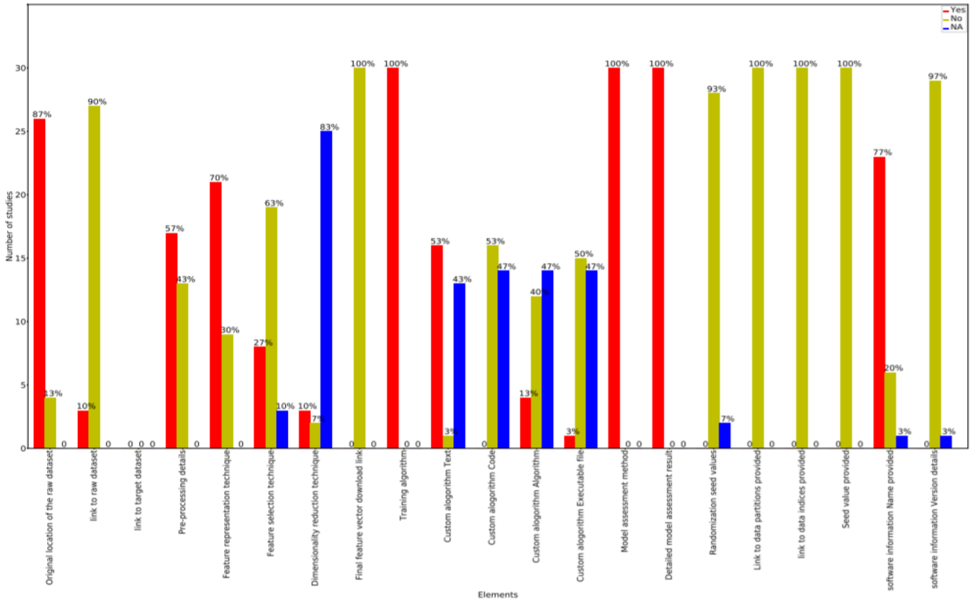
Reproducibility

- Reproducibility in machine learning

COMPUTER SCIENCE

Artificial intelligence faces reproducibility crisis

Unpublished code and sensitivity to training conditions make many claims hard to verify



Reproducibility

- Reproducibility over different machines
  - GPU card
  - CPU

Reproducibility over Different Machines

albanD Dec '19

Hi,

The problem is that across different machines, the hardware (for example different GPU cards) or different library version (cudnn adding/deleting algorithms for example) can give different results. So there is little we can do to ensure the same results 😕

ngimel commented on May 12, 2020 Contributor

This is expected behavior, we don't guarantee bitwise accuracy for different hardware.

ngimel closed this on May 12, 2020

Reproducibility

- Multi-GPU
  - 1 gpu, 16 batch vs 4 gpu, 4 batch per gpu, which can result in better performance ?

Conclusion for Reproducibility

- Reproducibility over different machines (X)
- Multiple executions with controlled random seeds, given the same inputs, will produce the same result (O)
- Tips for Geeks who struggle for state-of-the-art performance or want to beat competitors
  - Train your model with the largest batch size that memory allows in single gpu
  - Evaluate as many checkpoints as possible
  - Before the test, combine train set and validation set and train the model with the combined dataset
  - Do ensemble
  - The hyperparameters configuration of competitors using similar models is a good starting point