Hyperparameter Tuning in Pytorch

Presented by Taehee Kim
21.02.18

KAIST AI
Graduate School of AI

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 (Warmup 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....
  - training time for a model

Hyperparameters

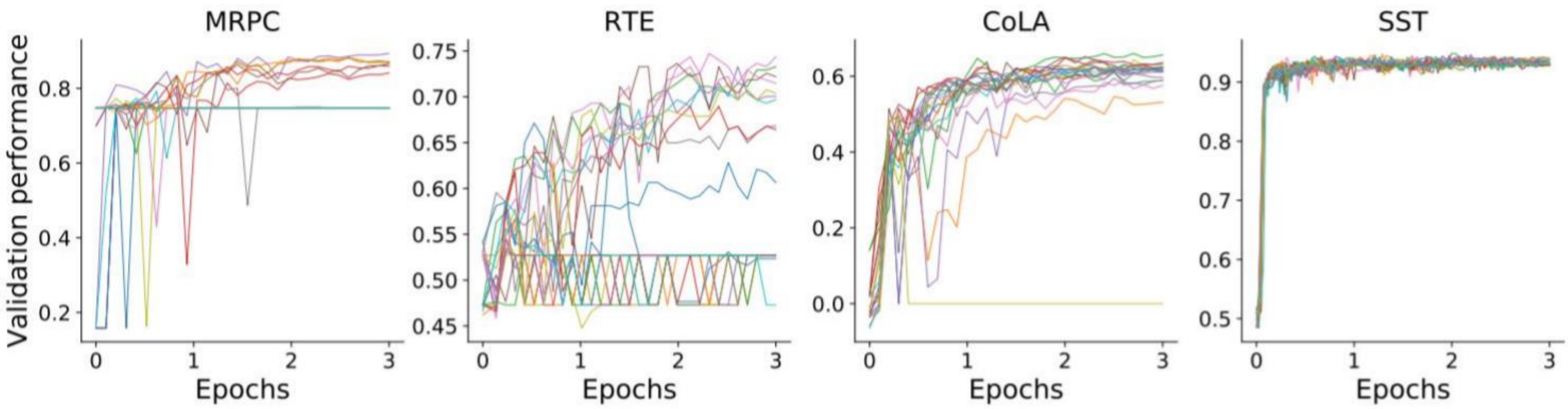
- ???
  - Accumulation steps
  - Random seed
  - Number of evaluation

Hyperparameters

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

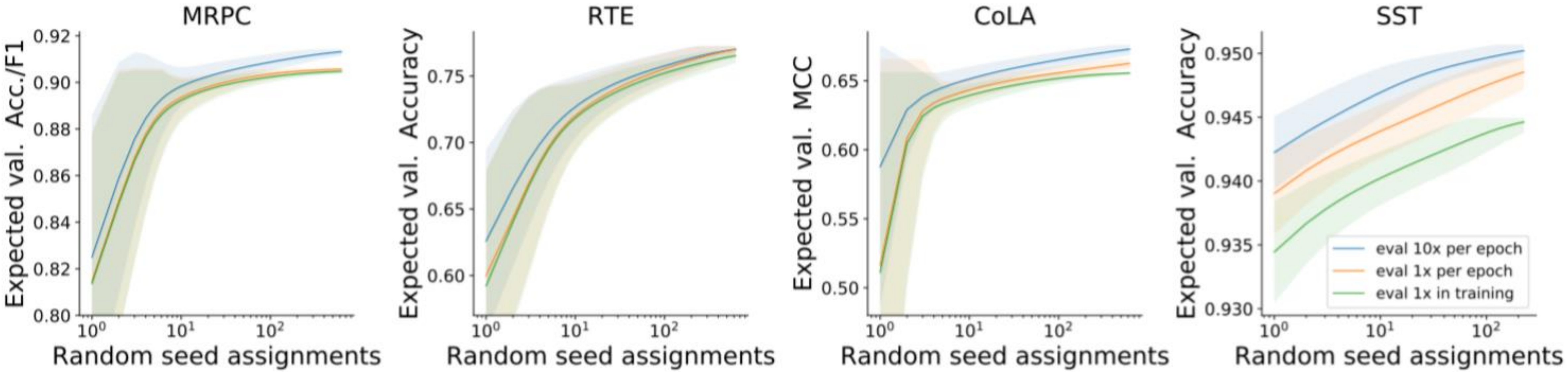
Hyperparameters

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



Hyperparameters

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



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).

Hyperparameter Tuning

- 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, mini-batch size, momentum beta, number of hidden units, number of layers, learning rate decay, regularization lambda, activation functions, beta1 & beta2 in Adam

Hyperparameter Tuning

- 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, mini-batch size, momentum beta, number of hidden units, number of layers, learning rate decay, regularization lambda, activation functions, beta1 & beta2 in Adam

Hyperparameter Tuning

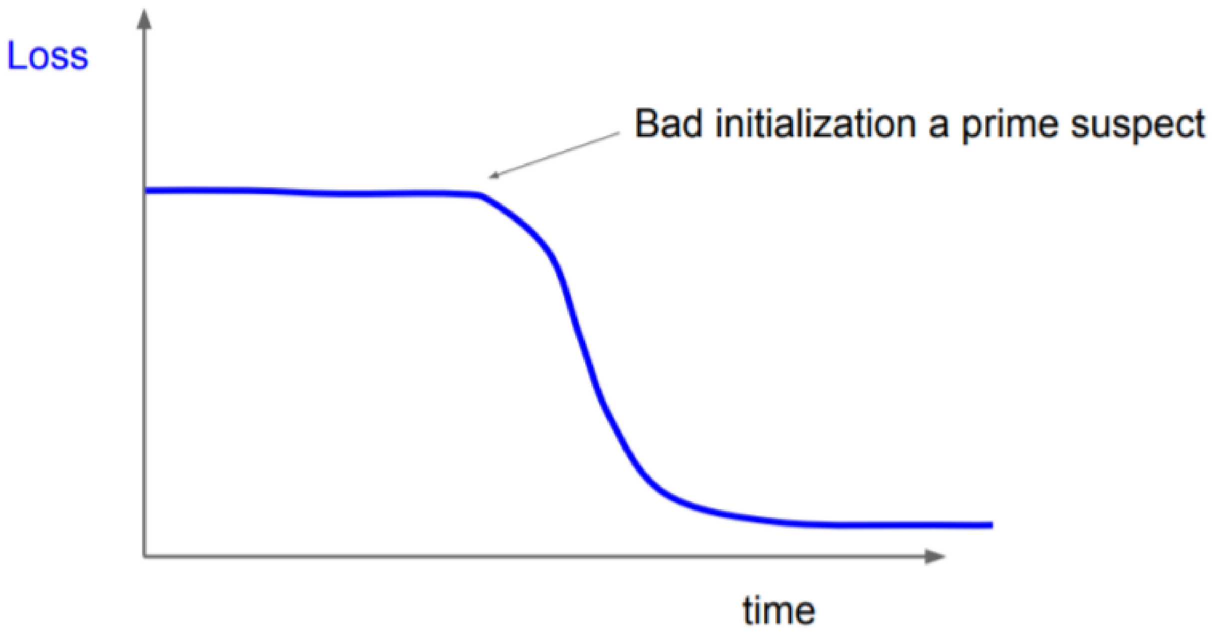
- (Maybe for researchers) Best practice for hyperparameters optimization
  - Learning rate, Learning rate, Learning rate
  - Step 1 (coarse-grained)
    - Turn off learning rate scheduler and train a model
    - Find a learning rate that makes loss low at the early stage of training compared to other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

Hyperparameter Tuning

- (Maybe for researchers) Best practice for hyperparameters optimization
  - Step 2 (fine-grained)
    - 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

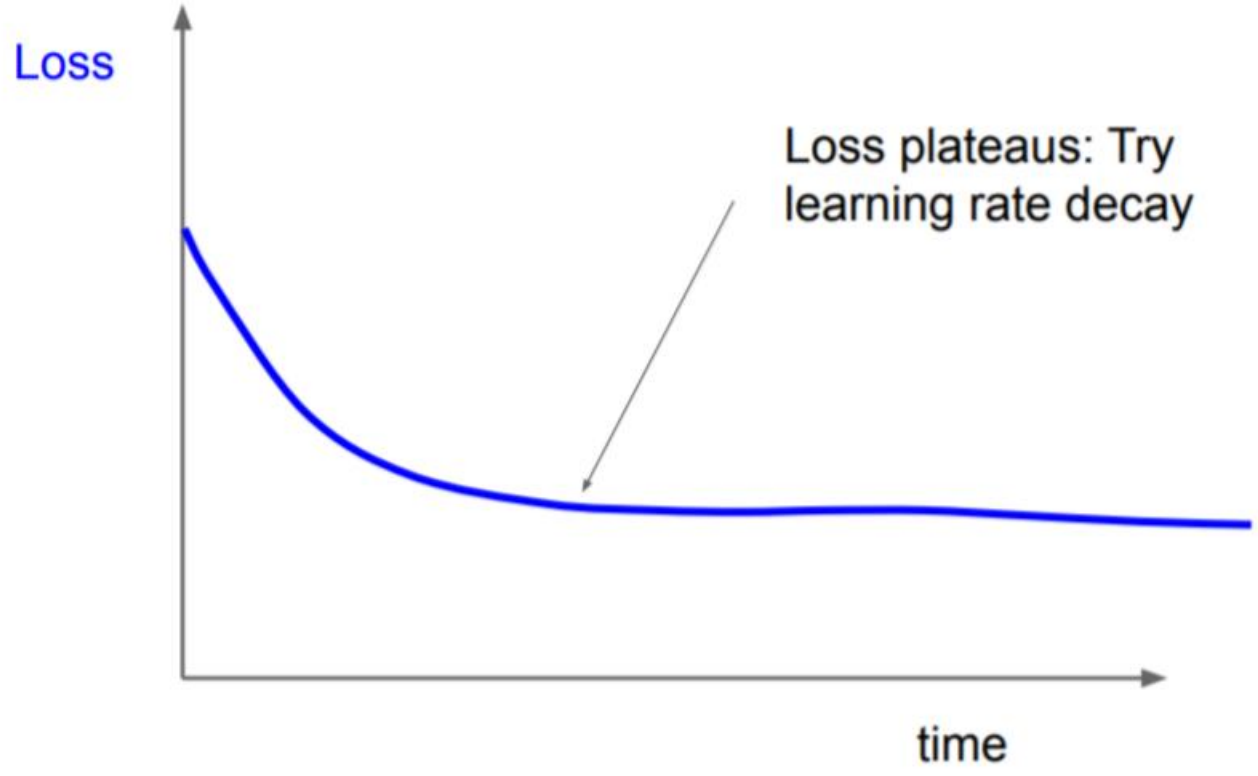
Hyperparameter Tuning

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



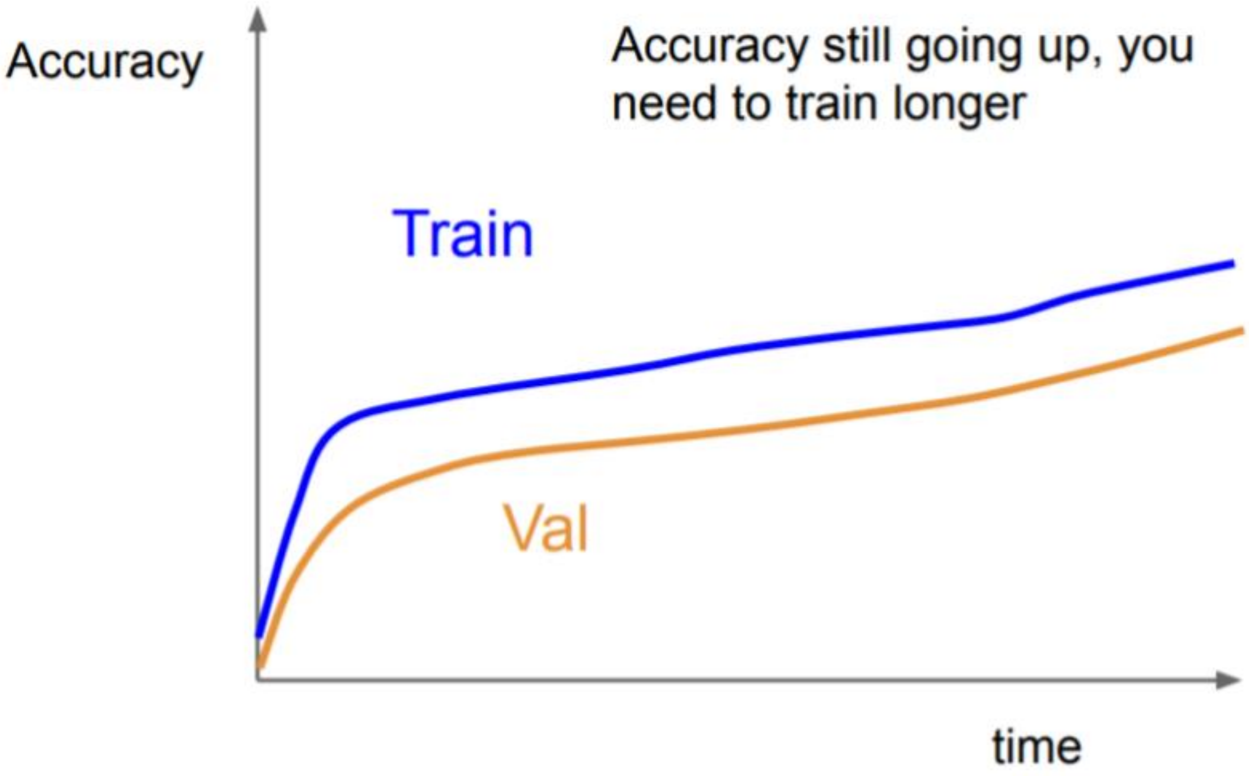
Hyperparameter Tuning

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



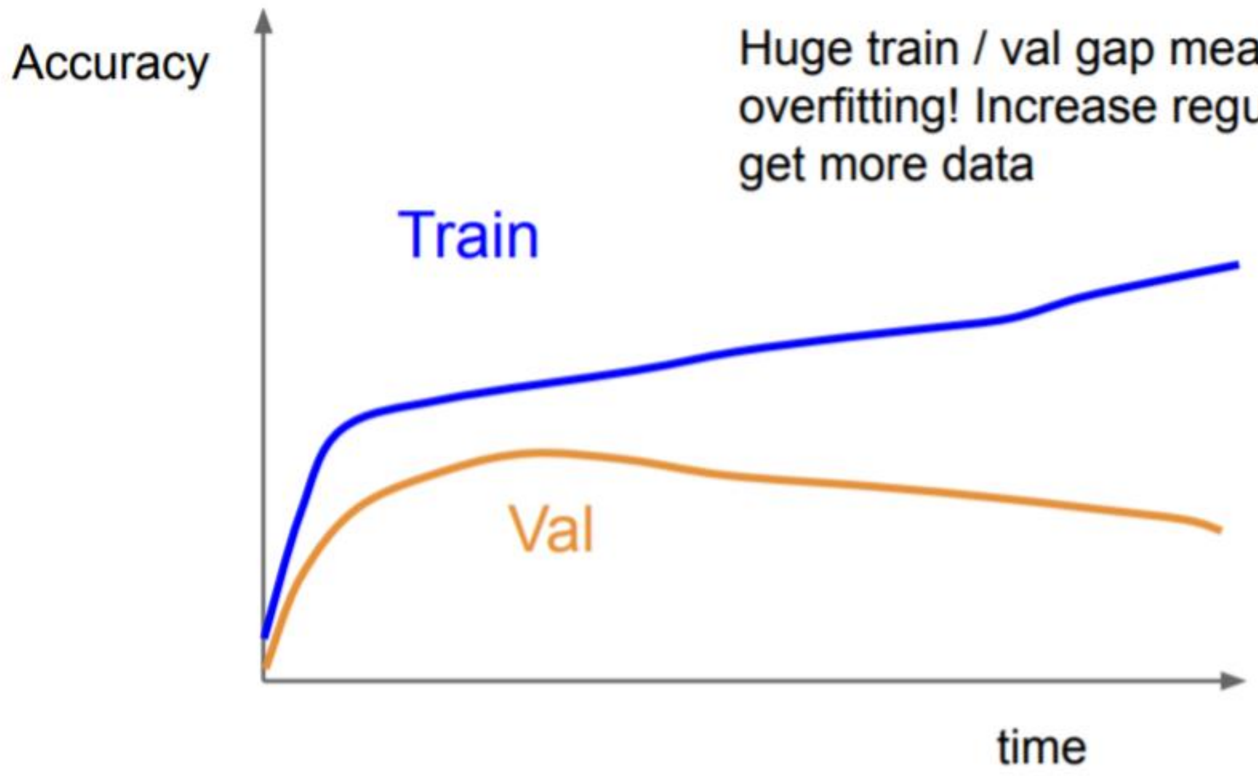
Hyperparameter Tuning

- (Maybe for researchers) Best practice for hyperparameters optimization
  - Step 4 (fine-grained)
    - Look at the training & validation learning curves



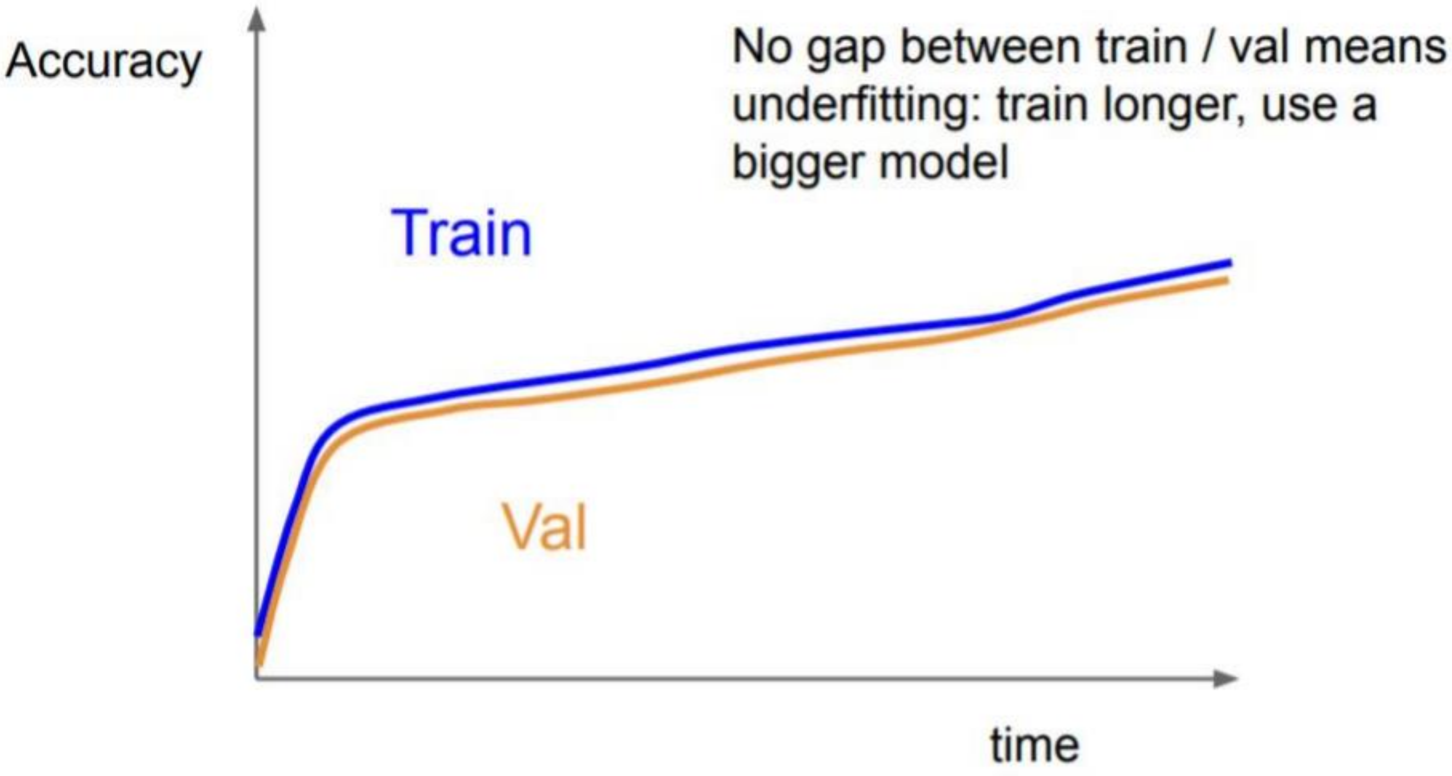
Hyperparameter Tuning

- (Maybe for researchers) Best practice for hyperparameters optimization
  - Step 4 (fine-grained)
    - Look at the training & validation learning curves



Hyperparameter Tuning

- (Maybe for researchers) Best practice for hyperparameters optimization
  - Step 4 (fine-grained)
    - Look at the training & validation learning curves



Conclusion

- 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 who use similar model is a good starting point
  - Use MLOps tool (e.g., wandb)