Optimizing large deep learning models

How can we find suitable hyperparameters?

- Typically, we don’t have the computational resources to make large grid searches
  - We need a clever way of selecting suitable hyperparameters
- What are hyperparameters?
  - Hyperparameters define the settings or configurations that control and define the learning process of a model
  - They are not learned from the data but instead they are set prior to the training process
  - Examples:
    - learning rate
    - number/dimension of layers
    - batch size
    - dropout rate

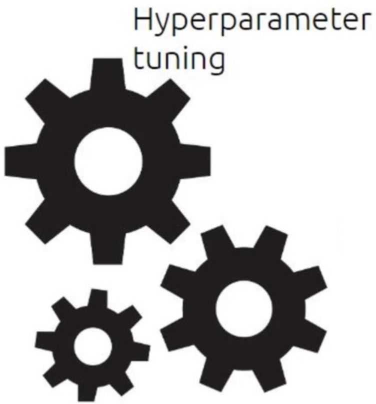


Image copied from: https://ubuntu.com/blog/hyperparameter-tuning-kubeflow

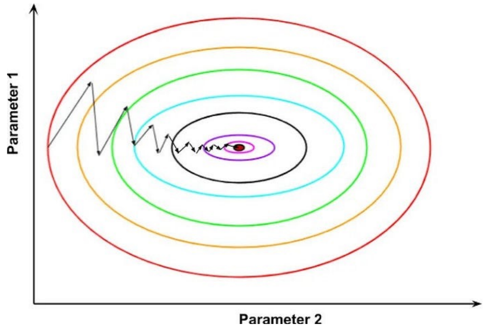
Selecting the batch size (to minimize training time)

- The same final performance should be attainable using any batch size
  - Set batch size once at the beginning
  - The best best size is typically the largest batch size that can be processed
    - Larger batch sizes typically reduce training time
  - Gradient accumulation simulates a larger batch size than the hardware can support -> does not provide any training speed advantages
- Many other hyperparameters are sensitive to the batch size
  - If you change batch size, you need to tune some hyperparameters again. For example:
    - Learning rate and momentum of optimizer
    - Regularization coefficients

https://arxiv.org/abs/1811.03600

Choosing the initial configuration

- It is often preferable to start with a simple optimizer. For example:
  - SGD with fixed momentum
  - Adam with fixed $\varepsilon, \beta_1, \beta_2$



- Architecture choices
  - Start with a relatively simply architecture and avoid any ”unnecessary” complexities
  - Complexity of the model can be improved later

Optimizer image copied from: https://medium.com/@francescofranco_39234/adam-optimization-in-machine-learning-cfeb10a27a86

How to tune hyperparameters and optimize model performance?

- We can follow the guidelines described in the *Deep Learning Tuning Playbook*
- Optimizing many hyperparameters at once is difficult because of limited computational resources
- We want to incrementally add/test features and hyperparameters
  - We test each feature/hyperparameter in a separate round of experiments
  - Adjusting multiple hyperparameters simultaneously might not allow to judge the effect of each feature/hyperparameter
- Division of hyperparameters in three different categories
  - Scientific: the hyperparameters whose effect we try to measure
  - Nuisance: need to be optimized over for a fair comparison
  - Fixed: have their values fixed in the current round of experiments

Instructions extracted from: https://github.com/google-research/tuning_playbook

How to test a feature or different values for a hyperparameter (1)

- 1. Each round of experiments should have a clear goal
  - Examples for goals:
    - Try a new pre-processing technique
    - Understand the effect of activation function choice
    - Understand the effect of dropout rate
- 2. Divide all hyperparameters into scientific, nuisance, and fixed parameters
  - Scientific hyperparameters are defined by the goal
  - Select nuisance hyperparameters
    - Optimizer and regularization parameters are typically selected for a fair comparison
  - The remaining hyperparameters are fixed hyperparameters

Instructions extracted from: https://github.com/google-research/tuning_playbook

How to test a feature or different values for a hyperparameter (2)

- Previous steps:
  - 1. Setting a goal
  - 2. Selecting scientific, nuisance, and scientific hyperparameters
- 3. Design and execute this round of experiments:
  - Define the search spaces for scientific and nuisance hyperparameters
    - Manually define all configurations
    - Use a search algorithm
  - Was the search space large enough?
    - If the optimal values are close to the search space borders, increase the search space
- 4. Evaluate results:
  - Is the improve better than random variation?
  - When deciding whether to adopt a change, you could re-run the best trial multiple times with different random seeds

Instructions extracted from: https://github.com/google-research/tuning_playbook

How to test a feature or different values for a hyperparameter (summary)

- Each round of experiments
  - 1. Setting a goal
  - 2. Selecting scientific, nuisance, and scientific hyperparameters
  - 3. Design and execute this round of experiments
  - 4. Evaluate the results
- Moving from broader goals (exploration phase) to more specific goals (exploitation phase)
  - At the beginning, asking more fundamental questions. For example:
    - Which hyperparameters should we include at all (dropout, L2 regularization, …)?
    - What activation function should we use?
  - Later, setting goals to find specific values of hyperparameters. For example:
    - What is the optimal dropout rate?
    - How many hidden layers should we use?

Instructions extracted from: https://github.com/google-research/tuning_playbook

Choice of number of epochs

- Number of epochs should not be a tunable hyperparameter
- Select max. number of epochs and use it in all trials
- Save a checkpoint after each training epoch:
  - After training, select the model and epoch that reached the lowest validation error
    - If best model is always during first 20% of training epochs, you might want to reduce the number of epochs
    - If best model is usually among last 20% of epochs, increase number of epochs (if computation resources allow this)
- If computational resources are very limited, we might want to select a much lower number of epochs during the exploration phase

Instructions extracted from: https://github.com/google-research/tuning_playbook