Hyperparameters are configuration settings that are external to the machine learning model itself and need to be manually set before the training process begins. Setting/tuning these hyperparameters is extremely difficult for several reasons. There are many hyperparameters, and each hyperparameter can take on multiple values. One hyperparameter can affect another, thus understanding the complex dependencies requires a lot of experimentation which many can't afford as tuning them is time-consuming and expensive. Worst of all there is no universal solution, as one set of hyperparameters won't work for a different problem/model. So we provide a comprehensive approach to this problem.

Start with a problem statement and query the necessary data and make it ready (cleaning) for the model training.

Principles: Strategy to tune hyperparameters:

1. Identify an appropriately-scoped goal for the next round of experiments.
2. Design and run a set of experiments that makes progress towards this goal.
3. Learn what we can from the results.
4. Consider whether to launch the new best configuration.

- *Most of the time, our primary goal is to gain insight into the problem.*

Try to explore the problem instead of blindly focusing on reducing validation errors.

- *Each round of experiments should have a clear goal and be sufficiently narrow in scope so that the experiments can make progress toward the goal.*

If we try to add multiple features or answer multiple questions at once, we may not be able to disentangle the separate effects on the results.

1. Start with a model configuration.

Model architectures typically have various hyperparameters that determine the model's size and other details (e.g. number of layers, layer width, type of activation function). Start with small models and grow eventually. Try to choose standard model configuration, so one doesn't have to spend a lot of time tuning model hyper-parameters.

Choose a starting optimizer, epoch, and performance metrics (train and validation).

2. Choosing Batch Size

Changing the batch size *without changing any other details of the training pipeline* will often affect the validation set performance but the difference can be obtained by tuning hyperparameters like optimizer and regularizer. So batch size shouldn't be tuned for validation performance.

Also, batch size decreases the training steps never increases it.

Thumb Rule: Choose the max batch size hardware supports.

Batch size is chosen first as it affects optimizers and regularizers.

- The optimal values of most hyperparameters are sensitive to batch size. Therefore, changing the batch size typically requires starting the tuning process all over again.
- The hyperparameters that interact most strongly with the batch size, and therefore are most important to tune separately for each batch size, are the optimizer hyperparameters (e.g. learning rate, momentum) and the regularization hyperparameters.
- Keep this in mind when choosing the batch size at the start of a project. If you need to switch to a different batch size later on, it might be difficult, time-consuming, and expensive to re-tune everything for the new batch size.

## Identify scientific, nuisance, and fixed hyperparameters

All hyperparameters will come under any one of the above-mentioned categories. Scientific hyperparameters are those whose effect on the model's performance we're trying to measure. These hyperparameters are often the ones that we want to tune to achieve better performance. For instance, in a deep learning model, the number of layers, the learning rate, and the batch size can all be scientific hyperparameters.

Nuisance hyperparameters are those that are not of direct interest to us but need to be optimized to fairly compare different values of the scientific hyperparameters.

Nuisance hyperparameters are often chosen to make sure that the comparison of different scientific hyperparameters is as fair as possible. For example, if our goal is to "determine whether a model with more hidden layers will reduce validation error", then the number of hidden layers is a scientific hyperparameter. The learning rate is a nuisance hyperparameter because we can only fairly compare models with different numbers of hidden layers if the learning rate is tuned separately for each number of layers. So for each hidden layer, we find the best learning rate and then compare different hidden layers.

Fixed hyperparameters are those whose values are set and do not change during the experiments. These hyperparameters are often chosen because they do not significantly affect the model's performance or because we want to fix their values to ensure consistency across different experiments. An example of a fixed hyperparameter is the random seed used to initialize the model's weights.

This is the biggest challenge. For example, we start with fixing epoch size. Then we experiment to find the best optimizer and find Adam is best. But SGD might have been best with a different epoch. So often we are constrained by time and computing. So fix the hyperparameters to compare others. But we should always ask the question ‘Is the best hyperparameter in the previous experiment sensitive to current scientific hyperparameters?’

In the simplest case, we would make a separate study for each configuration of the scientific parameters, where each study tunes over the nuisance hyperparameters.

For example, if our goal is to select the best optimizer out of Nesterov momentum and Adam, we could create one study in which optimizer="Nesterov_momentum" and the nuisance hyperparameters are {learning_rate, momentum}, and another study in which optimizer="Adam" and the nuisance hyperparameters are {learning_rate, beta1, beta2, epsilon}. We would compare the two optimizers by selecting the best-performing trial from each study.

3. Choose Optimizer

No optimizer is best across all learning problems. So do trails with popular optimizers like Adam, SGD, and RMSProp. Each of these optimizers has furthermore tunable parameters.

Goal: Which optimizer produces the best accuracy?

Scientific Hyperparameters: Adam, SGD, RMSProp

Nuisance Hyperparameters: learning rate, beta1, beta2, epsilon for Adam, ….

Fixed: Epochs, BatchSize,...

We should not assume two conditional hyperparameters are the same just because they have the same name! In the above example, the conditional hyperparameter called learning_rate is a *different* hyperparameter for optimizer="Nesterov_momentum" versus optimizer="Adam". Its role is similar (although not identical) in the two algorithms, but the range of values that work well in each of the optimizers is typically different by several orders of magnitude.

Adam:

making general statements about search spaces and how many points one should sample from the search space is very difficult. Note that not all the hyperparameters in Adam are equally important. The following rules of thumb correspond to different "budgets" for the number of trials in a study.

- If < 10 trials in a study, only tune the (base) learning rate.
- If 10-25 trials, tune the learning rate and beta1.
- If 25+ trials, tune the learning rate, beta1, and epsilon.
- If one can run substantially more than 25 trials, additionally tune beta2.

4. Epochs

We can tune epochs at the end. If the accuracy of trails improving at the end, then increase epochs.