Hyperparameter Tuning in Python: a Complete Guide

* 8 mins read
* Authors Shahul ES, Aayush Bajaj
* Updated March 4th, 2022

Choosing the correct hyperparameters for machine learning or deep learning models is one of the best ways to extract the last juice out of your models. In this article, I will show you some of the best ways to do hyperparameter tuning that are available today.

What is the difference between parameter and hyperparameter?

* **Model parameters**: These are the parameters that are estimated by the model from the given data. For example the weights of a deep neural network.
* **Model hyperparameters**: These are the parameters that cannot be estimated by the model from the given data. These parameters are used to estimate the model parameters. For example, the learning rate in deep neural networks.

**PARAMETERS**

**HYPERPARAMETERS**

They are required for making predictions

They are required for estimating the model parameters

They are estimated by optimization algorithms(Gradient Descent, Adam, Adagrad)

They are estimated by hyperparameter tuning

They are not set manually

They are set manually

The final parameters found after training will decide how the model will perform on unseen data

The choice of hyperparameters decide how efficient the training is. In gradient descent the learning rate decide how efficient and accurate the optimization process is in estimating the parameters

*Model parameters vs model hyperparameters | Source: [GeeksforGeeks](https://www.geeksforgeeks.org/difference-between-model-parameters-vs-hyperparameters/" \t "_blank)*

What is hyperparameter tuning and why it is important?

[Hyperparameter tuning](https://towardsdatascience.com/hyperparameter-tuning-c5619e7e6624) (or hyperparameter optimization) is the process of determining the right combination of hyperparameters that maximizes the model performance. It works by running multiple trials in a single training process. Each trial is a complete execution of your training application with values for your chosen hyperparameters, set within the limits you specify. This process once finished will give you the set of hyperparameter values that are best suited for the model to give optimal results.

Needless to say, It is an important step in any Machine Learning project since it leads to optimal results for a model. If you wish to see it in action, [here’s a research paper](https://arxiv.org/pdf/2007.07588.pdf" \t "_blank) that talks about the importance of hyperparameter optimization by experimenting on datasets.

How to do hyperparameter tuning? How to find the best hyperparameters?

Choosing the right combination of hyperparameters requires an understanding of the hyperparameters and the business use-case. However, technically, there are two ways to set them.

Manual hyperparameter tuning

Manual hyperparameter tuning involves experimenting with different sets of hyperparameters manually i.e. each trial with a set of hyperparameters will be performed by you. This technique will require a robust experiment tracker which could track a variety of variables from images, logs to system metrics.

There are a few experiment trackers that tick all the boxes. Neptune is one of them. It offers an intuitive UI and an open-source package neptune-client to facilitate logging in your code. You can easily log hyperparameters and see all types of data results like images, metrics, etc. Head over to the docs to see how you can [log different metadata to Neptune](https://docs.neptune.ai/you-should-know/what-can-you-log-and-display).

Alternative solutions include W&B, Comet, or MLflow. Check more [tools for experiment tracking & management here](https://neptune.ai/blog/best-ml-experiment-tracking-tools" \t "_blank).

**Advantages of manual hyperparameter optimization**:

* Tuning hyperparameters manually means more control over the process.
* If you’re researching or studying tuning and how it affects the network weights then doing it manually would make sense.

**Disadvantages of manual hyperparameter optimization**:

* Manual tuning is a tedious process since there can be many trials and keeping track can prove costly and time-consuming.
* This isn’t a very practical approach when there are a lot of hyperparameters to consider.

Read about [how to manually optimize Machine Learning model hyperparameters here](https://machinelearningmastery.com/manually-optimize-hyperparameters/" \t "_blank).

Automated hyperparameter tuning

Automated hyperparameter tuning utilizes already existing algorithms to automate the process. The steps you follow are:

* First, specify a set of hyperparameters and limits to those hyperparameters’ values (note: every algorithm requires this set to be a specific data structure, e.g. dictionaries are common while working with algorithms).
* Then the algorithm does the heavy lifting for you. It runs those trials and fetches you the best set of hyperparameters that will give optimal results.

In the blog, we will talk about some of the algorithms and tools you could use to achieve automated tuning. Let’s get to it.

Hyperparameter tuning methods

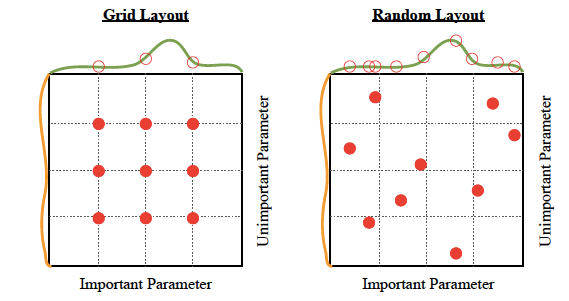
In this section, I will introduce all of the hyperparameter optimization methods that are popular today.

Random Search

In the [random search method](https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf" \t "_blank), we create a grid of possible values for hyperparameters. Each iteration tries a random combination of hyperparameters from this grid, records the performance, and lastly returns the combination of hyperparameters that provided the best performance.

Grid Search

In the [grid search method](https://towardsdatascience.com/grid-search-for-model-tuning-3319b259367e" \t "_blank), we create a grid of possible values for hyperparameters. Each iteration tries a combination of hyperparameters in a specific order. It fits the model on each and every combination of hyperparameters possible and records the model performance. Finally, it returns the best model with the best hyperparameters.

[*Source*](https://nanonets.com/blog/hyperparameter-optimization/)

Bayesian Optimization

Tuning and finding the right hyperparameters for your model is an optimization problem. We want to minimize the loss function of our model by changing model parameters. Bayesian optimization helps us find the minimal point in the minimum number of steps. [Bayesian optimization](https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f" \t "_blank) also uses an acquisition function that directs sampling to areas where an improvement over the current best observation is likely.

Tree-structured Parzen estimators (TPE)

The idea of [Tree-based Parzen optimization](https://optunity.readthedocs.io/en/latest/user/solvers/TPE.html" \t "_blank) is similar to Bayesian optimization. Instead of finding the values of p(y|x) where y is the function to be minimized (e.g., validation loss) and x is the value of hyperparameter the TPE models P(x|y) and P(y). One of the great drawbacks of tree-structured Parzen estimators is that they do not model interactions between the hyper-parameters. That said TPE works extremely well in practice and was battle-tested across most domains.

Hyperparameter tuning algorithms

These are the algorithms developed specifically for doing hyperparameter tuning.

Hyperband

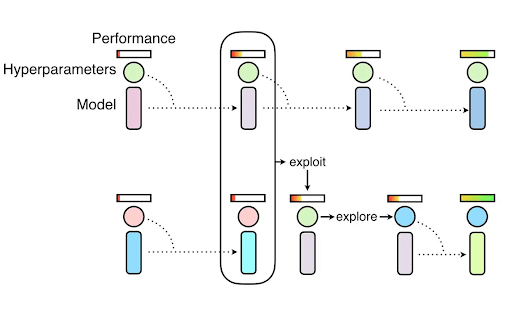
Hyperband is a variation of random search, but with some[explore-exploit](https://en.wikipedia.org/wiki/Multi-armed_bandit" \l "Empirical_motivation" \t "_blank) theory to find the best time allocation for each of the configurations. You can check this [research paper](https://arxiv.org/abs/1603.06560" \t "_blank)for further references.

Population-based training (PBT)

This technique is a hybrid of the two most commonly used search techniques: Random Search and manual tuning applied to Neural Network models.

PBT starts by training many neural networks in parallel with random hyperparameters. But these networks aren’t fully independent of each other.

It uses information from the rest of the population to refine the hyperparameters and determine the value of hyperparameter to try. You can check this [article](https://deepmind.com/blog/article/population-based-training-neural-networks" \t "_blank) for more information on PBT.

[*Source*](https://deepmind.com/blog/article/population-based-training-neural-networks)

BOHB

BOHB (Bayesian Optimization and HyperBand) mixes the Hyperband algorithm and Bayesian optimization. You can check this [article](https://www.automl.org/blog_bohb/" \t "_blank) for further reference.

**LEARN MORE**

[HyperBand and BOHB: Understanding State of the Art Hyperparameter Optimization Algorithms](https://neptune.ai/blog/hyperband-and-bohb-understanding-state-of-the-art-hyperparameter-optimization-algorithms" \t "_blank)

Tools for hyperparameter optimization

Now that you know what are the methods and algorithms let’s talk about tools, and there are a lot of those out there.

**Some of the best hyperparameter optimization libraries are:**

1. [Scikit-learn](https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide#scikit-learn)
2. [Scikit-Optimize](https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide#scikit-optimize)
3. [Optuna](https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide#optuna)
4. [Hyperopt](https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide#hyperopt)
5. [Ray.tune](https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide#ray)
6. [Talos](https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide#talos)
7. [BayesianOptimization](https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide#bayesianoptimization)
8. [Metric Optimization Engine (MOE)](https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide#moe)
9. [Spearmint](https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide#spearmint)
10. [GPyOpt](https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide#gpyopt)
11. [SigOpt](https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide#sigopt)
12. [Fabolas](https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide#fabolas)

1. Scikit-learn

[Scikit-learn](https://scikit-learn.org/) has implementations for grid search and random search and is a good place to start if you are building models with sklearn.

For both of those methods, scikit-learn trains and evaluates a model in a k fold cross-validation setting over various parameter choices and returns the best model.

Specifically:

* **Random search:** with [randomsearchcv](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html" \t "_blank)runs the search over some number of random parameter combinations
* **Grid search:**[gridsearchcv](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html" \t "_blank)runs the search over all parameter sets in the grid

Tuning models with scikit-learn is a good start but there are better options out there and they often have random search strategies anyway.

**MAY BE USEFUL**

 Check how you can keep track of your hyperparameters search when working with [Scikit-learn](https://docs.neptune.ai/integrations-and-supported-tools/model-training/sklearn" \t "_blank).

2. Scikit-optimize

[Scikit-optimize](https://scikit-optimize.github.io/stable/index.html) uses a [Sequential model-based optimization](https://ml.informatik.uni-freiburg.de/papers/11-LION5-SMAC.pdf) algorithm to find optimal solutions for hyperparameter search problems in less time.

Scikit-optimize provides many features other than hyperparameter optimization such as:

* store and load optimization results,
* convergence plots,
* comparing surrogate models

**MAY BE USEFUL**

 Check how you can keep track of your hyperparameters search when working with [Scikit-optimize](https://docs.neptune.ai/integrations-and-supported-tools/hyperparameter-optimization/scikit-optimize" \t "_blank).

3. Optuna

[Optuna](https://optuna.org/)uses a historical record of trails details to determine the promising area to search for optimizing the hyperparameter and hence finds the optimal hyperparameter in a minimum amount of time.

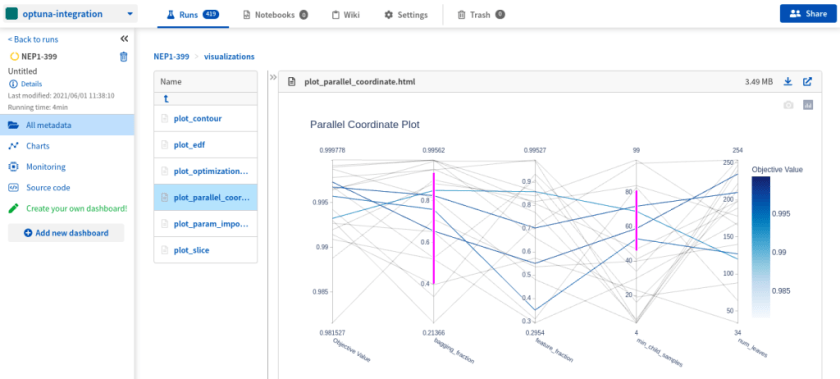
It has the pruning feature which automatically **stops the unpromising trails in the early stages of training**. Some of the key features provided by optuna are:

* Lightweight, versatile, and platform-agnostic architecture
* Pythonic search spaces
* Efficient optimization algorithms
* Easy parallelization
* Quick visualization

 You can refer to the official [documentation](https://optuna.readthedocs.io/en/stable/tutorial/first.html" \t "_blank) for tutorials on how to start using optuna.

**MAY BE USEFUL**

 Check how you can keep track of your hyperparameters search when working with [Optuna](https://docs.neptune.ai/integrations-and-supported-tools/hyperparameter-optimization/optuna" \t "_blank).

[](https://i0.wp.com/neptune.ai/wp-content/uploads/Neptune-Optuna-example.png?ssl=1)*Optuna hyperparameter optimization metadata logged to Neptune | [Documentation](https://docs.neptune.ai/integrations-and-supported-tools/hyperparameter-optimization/optuna" \t "_blank)*

4. Hyperopt

[Hyperopt](http://hyperopt.github.io/hyperopt/)is one of the most popular hyperparameter tuning packages available. Hyperopt allows the user to describe a search space in which the user expects the best results allowing the algorithms in hyperopt to search more efficiently.

Currently, three algorithms are implemented in hyperopt.

* [Random Search](http://www.jmlr.org/papers/v13/bergstra12a.html?source=post_page---------------------------)
* [Tree of Parzen Estimators (TPE)](https://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf)
* [Adaptive TPE](https://journals.sagepub.com/doi/pdf/10.1177/0020294020932347)

To use hyperopt, you should first describe:

* the objective function to minimize
* space over which to search
* the database in which to store all the point evaluations of the search
* the search algorithm to use

This[tutorial](https://github.com/hyperopt/hyperopt/wiki/FMin" \t "_blank) will walk you through how to structure the code and use the hyperopt package to get the best hyperparameters.

You can also read [this](https://mlwhiz.com/blog/2019/10/10/hyperopt2/" \t "_blank) article to learn more about how to use Hyperopt.

**RELATED ARTICLES**

[Optuna vs Hyperopt: Which Hyperparameter Optimization Library Should You Choose?](https://neptune.ai/blog/optuna-vs-hyperopt" \t "_blank)

5. Ray Tune

[Ray Tune](https://docs.ray.io/en/latest/tune/index.html) is a popular choice of experimentation and hyperparameter tuning at any scale. Ray uses the power of distributed computing to speed up hyperparameter optimization and has an implementation for several states of the art optimization algorithms at scale.

Some of the core features provided by ray tune are:

* distributed asynchronous optimization out of the box by[leveraging Ray](https://github.com/ray-project/ray" \t "_blank).
* Easily scalable.
* Provided SOTA algorithms such as [ASHA](https://ray.readthedocs.io/en/latest/tune-schedulers.html#asynchronous-hyperband),[BOHB](https://ray.readthedocs.io/en/latest/tune-searchalg.html#bohb), and[Population-Based Training](https://ray.readthedocs.io/en/latest/tune-schedulers.html#population-based-training-pbt).
* Supports Tensorboard and MLflow.
* Supports a variety of frameworks such Sklearn, XGBoost, TensorFlow, PyTorch, etc.

You can refer to this [tutorial](https://docs.ray.io/en/latest/tune/tutorials/overview.html) to learn how to implement ray tune for your problem.

6. Keras Tuner

[Keras Tuner](https://keras.io/keras_tuner/) is a library that helps you pick the optimal set of hyperparameters for your TensorFlow program. When you build a model for hyperparameter tuning, you also define the hyperparameter search space in addition to the model architecture. The model you set up for hyperparameter tuning is called a *hypermodel*.

You can define a hypermodel through two approaches:

* By using a model builder function
* By subclassing the HyperModel class of the Keras Tuner API

You can also use two pre-defined HyperModel classes –[HyperXception](https://keras-team.github.io/keras-tuner/documentation/hypermodels/#hyperxception-class) and[HyperResNet](https://keras-team.github.io/keras-tuner/documentation/hypermodels/#hyperresnet-class) for computer vision applications.

You can refer to this [official tutorial](https://www.tensorflow.org/tutorials/keras/keras_tuner" \t "_blank) for further implementation details.

**MAY BE USEFUL**

 Check how you can keep track of your hyperparameters search when working with [Keras Tuner.](https://docs.neptune.ai/integrations-and-supported-tools/hyperparameter-optimization/keras-tuner" \t "_blank)

7. BayesianOptimization

[BayesianOptimization](https://github.com/fmfn/BayesianOptimization)is a package designed to minimize the number of steps required to find a combination of parameters that are close to the optimal combination.

This method uses a proxy optimization problem (finding the maximum of the acquisition function) which, although it’s still a hard problem, it’s cheaper in the computational sense, and common tools can be employed. Therefore Bayesian Optimization is most adequate for situations where sampling the function to be optimized is a very expensive endeavor.

Visit the GitHub repo [here](https://github.com/fmfn/BayesianOptimization" \t "_blank) to see it in action.

8. Metric Optimization Engine

[MOE (Metric Optimization Engine)](https://github.com/Yelp/MOE) is an efficient way to optimize a system’s parameters when evaluating parameters is time-consuming or expensive.

It is ideal for problems in which

* the optimization problem’s objective function is a black box, not necessarily convex or concave,
* derivatives are unavailable,
* and we seek a global optimum, rather than just a local one.

This ability to handle black-box objective functions allows us to use MOE to optimize nearly any system, without requiring any internal knowledge or access.

Visit the GitHub [repo](https://github.com/Yelp/MOE) to read more about it.

9. Spearmint

[Spearmint](https://github.com/HIPS/Spearmint)is a software package that also performs Bayesian optimization. The software is designed to automatically run experiments (thus the code name spearmint) in a manner that iteratively adjusts a number of parameters so as to minimize some objectives in as few runs as possible.

Read and experiment about Spearmint in this GitHub [repo](https://github.com/HIPS/Spearmint).

10. GPyOpt

[GPyOpt](https://github.com/SheffieldML/GPyOpt)is Gaussian process optimization using [GPy](http://sheffieldml.github.io/GPy/). It performs global optimization with different acquisition functions.

Among other functionalities, it is possible to use GPyOpt to optimize physical experiments (sequentially or in batches) and tune the parameters of Machine Learning algorithms. It is able to handle large data sets via sparse Gaussian process models.

Unfortunately, GPyOpt maintenance has been shut by the authors of the repo but you can still use the package for your experiments.

Head over to its GitHub repo [here](https://github.com/SheffieldML/GPyOpt" \t "_blank).

11. SigOpt

[SigOpt](https://sigopt.com/)fully integrates automated hyperparameter tuning with training runs tracking to give you a sense of the bigger picture and the path to reach your best model.

With features like highly customizable search spaces and multimetric optimization, SigOpt can advance your model with a simple API for sophisticated hyperparameter tuning before taking it into production.

Visit the documentation [here](https://app.sigopt.com/docs/intro/intelligent_optimization" \t "_blank) to learn more about SigOpt’s hyperparameter tuning.

12. Fabolas

While traditional Bayesian hyperparameter optimizers model the loss of machine learning algorithms on a given dataset as a black box function to be minimized, FAst Bayesian Optimization on LArge data Sets (FABOLAS) models loss and computational cost across dataset size and uses these models to carry out Bayesian optimization with an extra degree of freedom.

You can check out the function implementing fabolas [here](https://github.com/automl/RoBO/blob/master/robo/fmin/fabolas.py" \l "L31" \t "_blank) and the research paper [here](https://arxiv.org/pdf/1605.07079.pdf" \t "_blank).

**READ ALSO**

[Best Tools for Model Tuning and Hyperparameter Optimization](https://neptune.ai/blog/best-tools-for-model-tuning-and-hyperparameter-optimization" \t "_blank)

Hyperparameter tuning resources and examples

In this section, I will share some hyperparameter tuning examples implemented for different ML and DL frameworks.

Random forest hyperparameter tuning

* [Understanding Random forest hyperparameters](https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74)
* [Bayesian hyperparameter tuning for random forest](https://blog.dominodatalab.com/hyperopt-bayesian-hyperparameter-optimization/)
* [Random forest tuning using grid search](https://www.kaggle.com/sociopath00/random-forest-using-gridsearchcv)

XGBoost hyperparameter tuning

* [XGBoost hyperparameters tuning python](https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/)
* [XGBoost hyperparameters tuning in R](https://www.analyticsvidhya.com/blog/2016/01/xgboost-algorithm-easy-steps/)
* [XGBoost hyperparameter using hyperopt](https://www.kaggle.com/alexandrnikitin/xgboost-hyperparameter-optimization)
* [Optuna hyperparameter tuning example](https://www.kaggle.com/corochann/optuna-tutorial-for-hyperparameter-optimization)

LightGBM hyperparameter tuning

* [Understanding LightGBM parameters](https://neptune.ai/blog/lightgbm-parameters-guide)
* [LightGBM hyperparameter tuning example](https://www.kaggle.com/mlisovyi/lightgbm-hyperparameter-optimisation-lb-0-761)
* [Optuna for LIghtGBM hyperparameter tuning](https://tech.preferred.jp/en/blog/lightgbm-tuner-new-optuna-integration-for-hyperparameter-optimization/)

CatBoost hyperparameter tuning

* [Catboost hyperparamters overview](https://catboost.ai/docs/concepts/parameter-tuning.html)
* [Grid search for catboost hyperparameter tuning](https://effectiveml.com/using-grid-search-to-optimise-catboost-parameters.html)

Keras hyperparameter tuning

* [Hyperparameter tuning using Keras- tuner example](https://blog.tensorflow.org/2020/01/hyperparameter-tuning-with-keras-tuner.html)
* [Keras CNN hyperparameter tuning](https://www.sicara.ai/blog/hyperparameter-tuning-keras-tuner)
* [How to use Keras models in scikit-learn grid search](https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/)
* [Keras Tuner: Lessons Learned From Tuning Hyperparameters of a Real-Life Deep Learning Model](https://neptune.ai/blog/keras-tuner-tuning-hyperparameters-deep-learning-model)

PyTorch hyperparameter tuning

* [Hyperparameter tuning for Pytorch](https://towardsdatascience.com/rocking-hyperparameter-tuning-with-pytorchs-ax-package-1c2dd79f2948)
* [Using optuna for hyperparameter tuning](https://medium.com/pytorch/using-optuna-to-optimize-pytorch-hyperparameters-990607385e36)

Final thoughts

Congratulations, you’ve made it to the end! Hyperparameter tuning represents an integral part of any Machine Learning project, so it’s always worth digging into this topic. In this blog, we talked about different hyperparameter tuning algorithms and tools which are widely used and studied. But even though, we covered a good chunk of techniques and tools, as a wise man once said, there’s no end to knowledge.

Here are some of the latest research happening in the area that might interest you:

* [Improving Hyperparameter Optimization By Planning Ahead](https://arxiv.org/pdf/2110.08028.pdf)
* [Hyperparameter Optimization: Foundations, Algorithms, Best Practices and Open Challenges](https://arxiv.org/pdf/2107.05847.pdf)
* [Experimental Investigation And Evaluation Of Model-Based Hyperparameter Optimization](https://arxiv.org/pdf/2107.08761.pdf)

That’s it for now, stay tuned for more, adios!