Meta-Hyperband: Hyperparameter optimization with meta-learning and Coarse-to-Fine

Samin Payrosangari[1], Afshin Sadeghi[1,2](✉), Damien Graux[3], and Jens Lehmann[1,2]

[1] Department of Computer Science, University of Bonn, Germany
[2] Fraunhofer IAIS, Sankt Augustin, Germany
[3] ADAPT SFI Research Centre, Trinity College Dublin, Ireland
`saminpayro@gmail.com, sadeghi@cs.uni-bonn.de, damien.graux@adaptcentre.ie, jens.lehmann@cs.uni-bonn.de`

Abstract. Hyperparameter optimization is one of the main pillars of machine learning algorithms. In this paper, we introduce Meta-Hyperband: a Hyperband based algorithm that improves the hyperparameter optimization by adding levels of exploitation. Unlike Hyperband method, which is a pure exploration bandit-based approach for hyperparameter optimization, our meta approach generates a trade-off between exploration and exploitation by combining the Hyperband method with meta-learning and Coarse-to-Fine modules. We analyze the performance of Meta-Hyperband on various datasets to tune the hyperparameters of CNN and SVM. The experiments indicate that in many cases Meta-Hyperband can discover hyperparameter configurations with higher quality than Hyperband, using similar amounts of resources. In particular, we discovered a CNN configuration for classifying CIFAR10 dataset which has a 3% higher performance than the configuration founded by Hyperband, and is also 0.3% more accurate than the best-reported configuration of the Bayesian optimization approach. Additionally, we release a publicly available pool of historically well-performed configurations on several datasets for CNN and SVM to ease the adoption of Meta-Hyperband.

Keywords: Hyperparameter Optim. · Meta-learning · Coarse-to-Fine

1 Introduction

In recent years, machine learning has opened up its way promoting automation in various domains such as image and face recognition [6], cancer detection [14] or speech recognition [4]. Each machine learning model has particular parameters, such as learning rate, learning rate decay or regularization coefficient which specify its architecture. These parameters are called the "hyperparameters" of the model which should be initialized by data scientists [1]. Specifying the right values for hyperparameters is critical and may lead to a model with high accuracy [17]. Contrariwise, choosing wrong values reduces the performance of the model significantly. Therefore, several approaches for hyperparameter optimization have been proposed. Nevertheless, some of these methods are still immature and need improvement or modification to enhance the probability of finding a near optimum model architecture. During hyperparameter optimization, one needs to evaluate the performance of various sets of values, training the model in individual rounds. To do so, a naive idea is sampling random hyperparameter configurations from space [1] and the best discovered values of hyperparameters, leading to lower generalization error, are chosen. To increase the speed of hyperparameter optimization, a novel approach called Hyperband has been recently proposed [11] based on dynamic resource allocation. In a nutshell, many random hyperparameter configurations are sampled from the initial space, and instead of allocating the maximum amount of training resources to each of them, they get lower amount of resources at the beginning and compete with each other for higher amount of resources. Subsequently, only a portion of configurations with lower validation loss will proceed to the next training round with more resources to demonstrate their real performance. Therefore, one could evaluate several random configurations faster than simple random search by early elimination of unintelligent configurations. The Hyperband algorithm is practical as it assesses numerous configurations, efficient in finding a high performance hyperparameter configuration for a dataset, and 5 to 30 times faster than its competitors such as different variations of Sequential model-based optimization like Spearmint, TPE and SMAC [11]. However, as Hyperband only targets random search and pure exploration, it has a deficiency in leveraging achieved information from previous evaluations and other machine learning experiments.

To tackle this limitation, we propose **Meta-Hyperband** to increase the quality of the discovered hyperparameter configurations. Meta-Hyperband adds extents of exploitation to Hyperband. Alongside random configurations sampled from the predefined space in Hyperband, Meta-Hyperband embeds a meta-learning module to benefit from historical machine learning experiments which have been performed on different datasets. In addition, a Coarse-to-Fine module is included in Meta-Hyperband to search the areas around the best configurations discovered during the ongoing hyperparameter optimization process, as the current best configuration might be close to the optimum hyperparameter configuration.

The organization of this paper is as follows. Section 2 reminds the background, Section 3 makes an overview on the related works in hyperparameter optimization. Section 4 describes Meta-Hyperband in details and Section 5 presents the results of experiments on CNN and SVM. Finally, in Section 6 concludes the study.

2 Background

Hyperparameter configuration: A “configuration” of a machine learning model is a unique set of values for different hyperparameters of the model. For instance, {learning rate = 0.5, regularization rate = 0.5, number of nodes in second layer = 10} is one configuration and {learning rate = 0.005, regularization rate = 0.5, number of nodes in second layer = 11} is another. Furthermore, If the model has t hyperparameters $h_1, \ldots, h_t$ and they vary in ranges $r_1, \ldots, r_t$

respectively, the “space” is defined as the t dimensional space $r_1 \times \cdots \times r_t$, and each configuration is allowed to be sampled from this space [5].

Meta-learning: Leveraging experiments and meta-data achieved, from previous learning procedures, in order to improve the quality of learning is known as meta-learning [18]. In this article, meta-learning corresponds to benefiting from historically well-performed hyperparameter configurations of a model on various datasets, to discover an optimum configuration for a new dataset.

Coarse-to-Fine: is a variation of the random search for hyperparameter optimization, which limits the space of possible hyperparameters configurations in an organized manner [12]. In pure random search, one picks many random configurations until reaching a high-quality model. In Coarse-to-Fine after randomly picking several initial configurations, the searching space is shrunk to the surrounding of the current best configuration [3]. The idea behind Coarse-to-Fine is to eliminate the impractical parts of the initial space and exploit the fruitful parts as the optimum hyperparameter configuration might be close to the best configuration which is discovered after an initial search in the main space.

3 Related Work

Random search is the basic idea for hyperparameter tuning. After random search, grid search was proposed which suffers from the curse of dimensionality and it limits the choices for more critical hyperparameters such as learning rate [1]. Although, hyperparameter optimization is time consuming, it influences the performance significantly. Thus, several other approaches have been designed. These approaches fall into two main categories: 1) Sequential model-based optimization and 2) Bandit-based optimization.

Sequential model-based optimization. SMBO approaches are methods for optimizing a black-box function, where calculation of the function for each individual point is costly, analogically to hyperparameter optimization problem.

From Different variations of SMBO, Bayesian optimization(BO) method uses “expected improvement (EI)” as the acquisition function to address the hyperparameter optimization task, since EI has shown exceptional performance in optimizing several other multidimensional black-box functions [17]. Another method is Tree-structured parzen estimation (TPE) which uses Tree-structured parzen estimator to define the surrogate, and an EI acquisition function that is based on this surrogate function [2]. Sequential model-based algorithm configuration (SMAC) is another SMBO that derives the surrogate of the black-box function using random forests. To optimize hyperparameters, a random forest is trained on a pool of historically evaluated configurations. Therefore, the forest predicts the performance of the particular model for new hyperparameter configuration [7]. Subsequently, it leverages the EI acquisition function to determine next point to be evaluated at each iteration of optimization procedure [7]. The methods in SMBO category struggle with the dimensionality issue [15], and in this condition, their performance is equivalent to random search. Moreover, these methods can't deal with non-convex functions [11]. Bandit-based methods, which are introduced in the next section, target these issues.

Bandit-based optimization In bandit-based hyperparameter optimization environment, more resources are allocated to promising hyperparameter configurations, known as adaptive resource allocation, and the unpropitious ones are stopped early resulting in an order of magnitude faster optimization compared to SMBO methods [8]. From the perspective of this category of methods, the problem of hyperparameter optimization is considered as an infinite-armed bandit problem in a non-stochastic environment. In this scenario, each arm is a hyperparameter configuration and pulling an arm corresponds to evaluating the performance of the configuration by training for a specific amount of epochs and returning the loss. One early bandit-based method is Successive Halving: given a total amount of budget B, the budget is uniformly divided among a set of n randomly sampled hyperparameter configurations. Its algorithm for hyperparameter optimization is:

1. Sample n hyperparameter configurations from the space
2. Given the total amount of budget B, Uniformly allocate a small portion of the budget B to each configuration.
3. Train the model with each of these configurations individually with the allocated budget
4. Eliminate half of the configurations which have shown lower performance,
5. Keep the other half and allocate a bigger portion of the B to each of them.
6. Repeat from step 3 until one config. remains and report it as the best one.

Using this algorithm more hyperparameter configurations can be assessed using total amount of budget B, because early termination of unpromising configurations releases resources for evaluating more Hyperparameter configurations, and only those configurations which have shown promising performance will be eligible to get more resources out of the total budget. Hyperband algorithm is a pure exploration infinite-arm bandit method that extends Successive Halving by repeating it several times, but with a different number of initial configurations n [11]. This extension of successive halving performs the hyperparameter optimization task about 5 to 30 times faster than well-known methods such as SMBO variations and random search [11].

In Hyperband, each run of Successive Halving is called a "bracket" [11]. This method reports the best configuration after execution of all individual Brackets. The reason for implementing multiple brackets with different values of n is that, it's indeterminable whether it's better to keep n small or large. When n is small, each configuration gets more resources and when a big n is chosen each configuration gets a lower number of resources but more configurations could be evaluated. In some problems, individual configurations might have a slow convergence rate, so they need more resources to reveal their performance and to compete with other configurations. Keeping the trade-off between the number of configurations n and the amount of resources which is allocated to each configuration B/n, is called the n versus B/n trade-off. The inputs of Hyperband include the maximum number of resources R which could be allocated to each individual configuration (i.e. determines the budget limit) and η which determines the portion of configurations that should be discarded at each round inside the brackets. Respectively, R and η determine the number of brackets and the initial number of configurations n_i in individual brackets. We introduce an algorithm in Section 4 that applies an improved sampling method for the initial configurations in each bracket, without touching the other variables such as R and η.

4 Meta-Hyperband

We propose **Meta-Hyperband**, a meta-learning hyperparameter optimization algorithm that benefits from a pool of historically well-performed hyperparameter configurations on various datasets in previous experiments. The idea is that one of these configurations might perform well on the new dataset directly or by little manipulation obtained by the Coarse-to-Fine module. Therefore, the Meta-Hyperband algorithm adds meta-learning as well as Coarse-to-Fine modules to the Hyperband algorithm. An advantage of Meta-Hyperband (over Hyperband) is its trade-off between exploration and exploitation as part of the brackets run a random search: a part of them run a meta-learning and the rest run a Coarse-to-Fine sampling on the best configurations discovered in the previous steps. The inputs to our algorithm are:

- **R:** The maximum amount of resources which should be allocated to each configuration. Basically, this value changes for different problems. For example, in Deep Learning, the number of training iterations might be selected as the resource and the maximum iterations which is required for a successful training depends on the problem.
- **Meta-learning pool:** A pool of best hyperparameter configurations discovered in historical experiments of the corresponding machine learning model. We show (see Section 5) that having a meta-learning pool improves the results significantly and leads to a faster approach. Therefore, contributing in building such a pool for different models helps to improve the performance of the models on new datasets.
- **Coarse-to-Fine coefficient:** The area around a proper configuration to be explored. If the initially specified range for a hyperparameter is (a, b) and the current best value for it is h_1, then h_1 is considered as the coarse and we try to fine tune h_1 in its surrounding [3]. In this article, considering that the Coarse-to-Fine coefficient is equal to r, the shrunk range for this hyperparameter is $(h_1 - rh_1, h_1 + rh_1)$. Here, the amount of moving around h_1 depends on itself by multiplying the radius by h_1, in order to avoid outlying values which are either far away from h_1 or too close to it. By limiting the range of all the hyperparameters in one configuration using the same coefficient, the space is restricted using the Coarse-to-Fine module. The value of r defines the extent of space shrinkage, as smaller values apply more aggressive space restriction than larger values. The default value we consider for r is 0.2. According to our

experiments, lower values of r restrict the space such that the performance improvement by fine-tuned configurations is not outstanding.

In Meta-Hyperband, 5 brackets (from 0 to 4) are considered. In the initialization phase, the maximum bracket number called s_{max} equals 4, and η defines the portion of resources which should be eliminated in each round in brackets. In Hyperband, η is an input and the s_{max} which is the index of last bracket, is defined by $\lfloor \log_\eta R \rfloor$. Differently, in Meta-Hyperband, the value of s_{max} is predefined, and using the same rule the equation of $\eta = \lfloor \sqrt[4]{R} \rfloor$ holds. The reason behind this setup is that, as a proper arrangements for Hyperband it was proposed to determine η in a way that according to R, the algorithm runs in 5 brackets, so that with different values of sampled configurations in different brackets the n versus B/n trade-off is holds [11]. The Meta-Hyperband algorithm is defined as:

1. Bracket$_1$, meta-learning: Sample n_1 hyperparameter configurations from the pool of meta-learning, and apply the Successive Halving steps 2-6 (see Section 3) on them. If there is no pool, the Bracket performs a random search.
2. Bracket$_4$, random search: Randomly sample n_4 hyperparameter configurations from space, and apply the Successive Halving steps 2-6 on them.
3. Bracket$_2$, random search: Randomly sample n_2 hyperparameter configurations from space, and apply the Successive Halving steps 2-6 on them.
4. Bracket$_3$, Coarse-to-Fine:
   (a) Sample $n_3/2$ fine-grained configurations where coarse is the best of Bracket$_1$, and apply the Successive Halving steps 2-6 on them.
   (b) Sample $n_3/2$ fine-grained configurations where coarse is the best of brackets 2 and 4, and apply the Successive Halving steps 2-6 on them.
5. Bracket$_0$, Coarse-to-Fine: Sample n_0 fine-grained configurations where coarse is the best of all previous steps, and apply the steps 2-6 on them.

In this algorithm, the number of configurations n_i in Bracket i, and the resource allocation are similar to Hyperband so the inequality $n_4 > n_3 > n_2 > n_1 > n_0$ holds. Therefore, in brackets with higher index, more configurations are sampled and each of them receives a smaller portion of the total budget B of the bracket in comparison to the brackets with lower index. Therefore, the Bracket with highest index (Bracket$_4$) samples R configurations initially and Bracket$_0$ samples $s_{max}+1$ (5 in case of Meta-Hyperband) but allocates maximum amount of resources R to each of them. Algorithm 1 shows the pseudocode of Meta-Hyperband. Inside each bracket of Meta-Hyperband, the algorithm performs the sampling task according to the role of each bracket as discussed. In each bracket, the n_i configurations that are present are trained using the specified amount of resources r_i, $1/\eta$ of them with lower validation loss win the competition and proceed to next round, and the rest of the configurations are discarded. At the end of each bracket, one configuration is reported as the best configuration of that bracket. After locating Bracket$_1$, which is the meta-learning sampling bracket, Brackets 4 and 2 are located, where the random search in the main space of configurations takes place. The Bracket$_4$ samples too many configurations and increases the chance

Algorithm 1 Meta-Hyperband algorithm

```
Inputs:      $R$, Coarse-to-Fine coefficient, meta-learning pool
Initialize:     $s_{max} = 4, \eta = \lfloor \sqrt[4]{R} \rfloor, B = (s_{max} + 1)R$
for $s = \{1, 4, 2, 3, 0\}$ do
    $n = \lceil \frac{B}{R} \frac{n^S}{s+1} \rceil$, $r = R\eta^{-s}$, $T = []$
    $path$ = path-to-save-configurations
    if $s == 1$ then
        $T$ = get-meta-hyperparameter-configs($n$)
    else if $s == 4$ $or$ $2$ then
        $space$ = get-main-space()
        $T$ = get-random-hyperparameter-configs($n$, $space$)
    else if $s == 3$ then
        $a$) $space$ = get-Coarse2Fine-space(best config. from bracket 1)
        $T$ = get-fine-parameters($space$, $\frac{n}{2}$)
        $b$) $space$ = get-Coarse2Fine-space(best config. from brackets 2 & 4)
        $T.append$(get-fine-parameters($space$, $\frac{n}{2}$))
    else if $s == 0$ then
        $space$ = get-Coarse2Fine-space(best config. from at all other brackets)
        $T$ = get-fine-parameters($space, n$)
    for $i \in \{0, ..., s\}$ do
        $n_i = \lfloor n \times \eta{-}i \rfloor$, $r_i = \lfloor r \times \eta i \rfloor$
        $L$ = evaluate-the-validation-loss($t$) : $t \in T$
        $T = top(T, L, \lfloor \frac{n_i}{\eta} \rfloor)$
Result: The best discovered configuration with lowest validation loss.
```

of exploring the space thoroughly. Moreover, Bracket$_2$ samples lower number of random configurations but allocates more resources to each of them. After completion of the 3 previous brackets, the best configuration discovered in the Bracket$_1$, as well as the best configuration of Brackets 4 and 2, are taken into Bracket$_3$ which performs Coarse-to-Fine on these two current best configurations. This avoids trapping in the area around a local optimum because it considers two of the best configurations instead of considering one global best. Finally, Bracket$_0$ takes the global best of all previous brackets and applies Coarse-to-Fine on it. If the meta-learning pool is missing, the Coarse-to-Fine module can still be leveraged. And if this pool contains many configurations, sampling from it can be moved to other brackets of the Meta-Hyperband than the proposed bracket in this paper. Our analysis of Hyperband shows that in most of the hyperparameter optimization experiments all the Brackets are not used optimally. Experiments in Section 5 prove the dominance of Meta-Hyperband in discovering high performance configurations and leveraging all the brackets.

5 Experiments

To compare the performance of Meta-Hyperband and Hyperband, several experiments are performed on CNN and SVM using 5 benchmark datasets: Cifar10 [9], Cifar100 [9], SVHN [13], MNIST [10] and Fashion-MNIST [19]. To perform the experiments, firstly Hyperband is used several times to discover the best hyperparameter configuration of each model for each of the datasets. Subsequently, all the discovered configurations are gathered to generate the pool of configurations for each model to feed Meta-Hyperband. These pools of configurations are enriched every time a new proper configuration for a dataset is discovered[1].

[1] Our sources → https://github.com/saminpayro/Meta_Hyperband_implementation

Hyperparameter	Range	Scale
Learning rate reduction	[0,3]	integer
Initial learning rate	$[5 \times 10^{-5}, 5]$	log
Conv1 L2 penalty	$[5 \times 10^{-5}, 5]$	log
Conv2 L2 penalty	$[5 \times 10^{-5}, 5]$	log
Conv3 L2 penalty	$[5 \times 10^{-5}, 5]$	log
Fc10 L2 penalty	$[5 \times 10^{-3}, 500]$	log
Local response normalization scale	$[5 \times 10^{-6}, 5]$	log
Local response normalization power	$[0.01, 3]$	linear

Table 1. Hyperparameter configuration space for CNN [11].

Method/Results	Discovery bracket	Top-one test loss 75 epochs	Top-one test loss 300 epochs
Hyperband	Bracket$_4$, random	0.214500	0.195400
Hyperband	Bracket$_4$, random	0.219100	0.186600
Hyperband	Bracket$_4$, random	0.213800	0.202100
Meta-Hyperband	Bracket$_3$a, Coarse-to-Fine	0.173000	0.146800
Meta-Hyperband	Bracket$_4$, random	0.177000	0.147600
Meta-Hyperband	Bracket$_3$a, Coarse-to-Fine	0.191900	0.150800

Table 2. Hyperband & Meta-Hyperband on CIFAR10 using CNN: R=300 and η=4.

Method/Results	Best config discovery bracket	Top-one test loss 6 epochs
Hyperband	Bracket$_4$, random	0.080042
Hyperband	Bracket$_2$, random	0.058208
Hyperband	Bracket$_3$, random	0.093667
Meta-Hyperband	Bracket$_0$, Coarse-to-Fine	0.066625
Meta-Hyperband	Bracket$_4$, random	0.064833
Meta-Hyperband	Bracket$_3$a, Coarse-to-Fine	0.062667

Table 3. Hyperband & Meta-Hyperband on SVHN using CNN. R=600 and η=4.

Method/Results	Best config discovery bracket	Top-one test loss after 300 epochs
Hyperband	Bracket$_4$, random	0.8300
Hyperband	Bracket$_3$, random	0.7986
Meta-Hyperband	Bracket$_4$, random	0.5893
Meta-Hyperband	Bracket$_3$b, Coarse-to-Fine	0.5764

Table 4. Hyperband & Meta-Hyperband on CIFAR100 using CNN. R=300 and η=4.

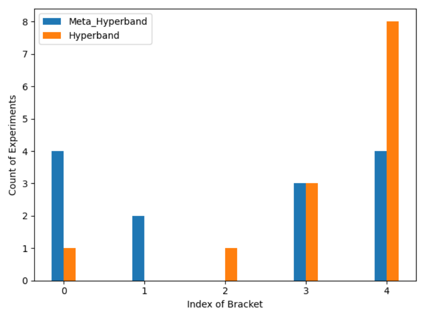

Fig. 1. Bar chart displaying the role of different brackets in 13 Hyperband and 13 Meta-Hyperband CNN experiments.

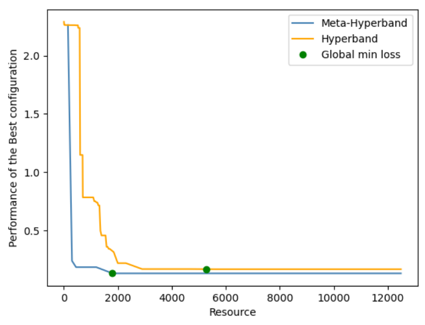

Fig. 2. Performance of the best discovered configuration in function of resource usage on two CNN with SVHN.

While applying Meta-Hyperband on a dataset, only the configurations corresponding to the rest of the datasets in the pool are involved in the meta-learning bracket. The value of η is initialized to 4 according to the amount of resources for our datasets, so Meta-Hyperband consists of 5 brackets. Moreover, the Coarse-to-Fine radius is initialized to $0.2 \times h$ for a hyperparameter h (this radius can also be treated as a hyperparameter). The resource in CNN is defined as the number of training iterations, and in SVM as the amount of training data points.

5.1 Experiments on CNN

In this study, the 18% error architecture proposed by Alex Krizhevsky is used for CNN model[2] and implemented by Cuda-convnet2 framework[3]. This architecture of CNN is also used in Hyperband [11] and ensures a fair comparison between the two approaches. Furthermore, the 8 hyperparameters of CNN are tuned (see Table 1). Table 1 also shows the ranges for them, which are similar to what was considered in [11] for fair comparison.

Each resource unit in our study is the training for 100 iterations which takes place on one batch of data of size 10k, and mini-batch size of 100 datapoints. Each method is repeated 13 times to gain an average performance. The bar chart (Fig.1) displays the performance of brackets in both approaches, by counting the number of experiments that their best configuration was discovered in Bracket$_i$. The diversity of the bars in case of Meta-Hyperband shows that there is a higher balance in the performance of brackets, and despite Hyperband brackets 0, 1, 3, 4 all have an active contribution in discovering a proper configuration for the CNN. The plots (Fig.2) display the progress of the two approaches with respect to the resource usage for 2 experiments on SVHN dataset, in which the Meta-Hyperband has discovered the best configuration with consumption of less than 3000 units of resources compared to Hyperband. Table 2 displays the results of experiments on Cifar10. Among the 6 experiments, Meta-Hyperband has discovered configurations with around more than 3% lower test error than Hyperband after 75 and 300 epochs. The lowest error discovered by Meta-Hyperband on test set is 14.68% after 300 epochs training, which is even better than the configuration reported by Bayesian Optimization method (14.98%) [17].

In another experiment, the SVHN dataset is used as the basis. Table 3 compares the quality of the best configurations according to the test error and after 6 epochs training. Among the 6 experiments, Meta-Hyperband has discovered configurations with around 3% lower test error than 2 runs of Hyperband. In addition, configurations from the meta-learning pool of configurations, which were corresponding to Fashion-MNIST and CIFAR10, perform well on SVHN as well. For instance, one historical configuration of Fashion-MNIST produces 8% test loss on SVHN as well, and the best configuration (reported on Table 3 last row) is the result of Coarse-to-Fine on a historical configuration of CIFAR10 from the meta-learning pool. In addition, Table 4 shows that during experiments

[2] See "example layers" directory in http://code.google.com/p/cuda-convnet/
[3] https://code.google.com/archive/p/cuda-convnet2/

Hyperparameter	**Type**	**Values**
preprocessor	Categorical	min/max, standardize, normalize
Kernel	Categorical	rbf, polynomial, sigmoid
C	Continuous	$log[10^{-3}, 10^{5}]$
γ	Continuous	$log[10^{-5}, 10]$
degree	if kernel=poly	integer [2, 5]
coef0	if kernel=poly, sigmoid	uniform [-1.0, 1.0]

Table 5. List of 6 hyperparameters for SVM with their sampling ranges here.

Method/Results	**Discovery bracket**	**Top-one error**
Hyperband	Bracket$_4$, random	0.4467
Meta-Hyperband	Bracket$_0$, Coarse-to-Fine	0.4402
Hyperband	Bracket$_2$, random	0.4455
Meta-Hyperband	Bracket$_1$, meta-learning	0.4429
Hyperband	Bracket$_4$, random	0.4477
Meta-Hyperband	Bracket$_0$, Coarse-to-Fine	0.4452

Table 6. Hyperband & Meta-Hyperband on CIFAR10 using SVM: R=400, $\eta = 4$.

on CIFAR100 dataset, Meta-hyperband outperformed Hyperband (more than 2% lower test error). Rows 3 and 4 show that, in Meta-Hyperband both random and non-random brackets are leveraged for discovering a proper configuration.

5.2 Experiments on SVM

The list of 6 hyperparameters tuned for SVM is displayed in Table 5. For SVM, one unit of resource equals 100 data points, and the maximum number of resources should be equal to the size of the training set for each dataset. In this project, we used 60k training set for SVHN, and 40k training sets for CIFAR10 and CIFAR100 respectively. Fig.3 displays the performance of different brackets in both approaches, by counting the number of experiments that their best configuration was discovered in Bracket$_i$. According to this figure, in the SVM case, Bracket$_4$ of Hyperband is also the most effective bracket.

For SVM experiments, instead of repeating the execution of Meta-Hyperband for all the brackets, some configurations which have been discovered previously during the Hyperband test, are considered as the best of random brackets in Meta-Hyperband as well to ensure a fair comparison between the two methods.

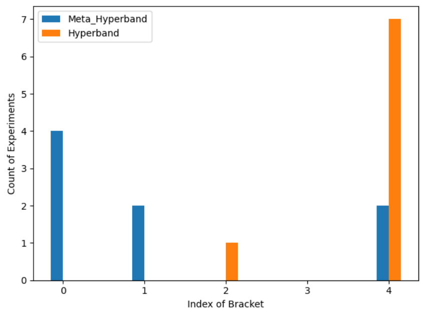

Fig. 3. Role of different brackets in Hyperband vs Meta-Hyperband in SVM.

Tables 6, 7 and 8 display the results of experiments on CIFAR10, SVHN, and CIFAR100 for SVM model. In these tables, each Meta-Hyperband row is paired to its above Hyperband row, which means that the best configurations discovered by Hyperband in steps 4 or 2 are considered as the best configuration of random brackets for the paired Meta-Hyperband experiment.

Analysing the diversity of best configuration discovery among the brackets of Meta-Hyperband on SVM in the bar chart of Fig.3 shows that Bracket$_0$

Method/Results	**Discovery bracket**	**Top-one error**
Hyperband	Bracket$_4$, random	0.232406269207
Meta-Hyperband	Bracket$_4$, random	0.232406269207
Hyperband	Bracket$_4$, random	0.245789072890
Meta-Hyperband	Bracket$_0$, Coarse-to-Fine	0.245678902347
Hyperband	Bracket$_4$, random	0.22303318992
Meta-Hyperband	Bracket$_4$,random	0.22303318992

Table 7. Hyperband & Meta-Hyperband on SVHN using SVM: R=600, $\eta = 4$.

Method/Results	**Discovery bracket**	**Top-one error**
Hyperband	Bracket$_4$, random	0.7486
Meta-Hyperband	Bracket$_1$, meta-learning of SVHN historical configuration	0.7166
Hyperband	Bracket$_4$, random	0.7486
Meta-Hyperband	Bracket$_0$, Coarse-to-Fine of SVHN historical configuration	0.7156

Table 8. Hyperband & Meta-Hyperband on CIFAR100 using SVM: R=400, $\eta = 4$.

was the most effective bracket in discovering high performance configurations, also Bracket$_4$ still plays an important role by exploring the space. Moreover, 2 configurations were discovered in Bracket$_1$ which speed up the hyperparameter tuning of Meta-Hyperband.

6 Conclusion and Future Work

In this study, we proposed Meta-Hyperband, our method to discover hyperparameter configurations which outperforms the literature algorithms, by introducing levels of exploitation using meta-learning as well as Coarse-to-Fine modules. Our approach provides a trade-off between exploration and exploitation, and benefits from the information gained from previous experiments. By considering more than one best configuration during the exploitation, it reduces the probability of trapping into a local optimum and the application of the historical or meta-information alongside the random search leads to a proper hyperparameter configuration faster than traditional methods. Meta-Hyperband has been validated through several experiments on CNN and SVM to tune hyperparameters for learning five different datasets. The best configurations discovered for SVM and CNN generated a pool that is shared and can be used in other experiments.

A recent approach to optimize hyperparameters, proposed by [16] for the MDE$_{nn}$ version of MDE, integrates the hyperparameters into the primary model's optimization. There, after initiating hyperparameters, the model optimizes them by back-propagation. It is interesting for future work to extend Meta-Hyperband in this direction and combine the Bandit-based algorithm with the models optimization, especially for NN-based models.

Acknowledgements

This study is supported by MLwin (*Maschinelles Lernen mit Wissensgraphen*, GA 01IS18050F of the German Federal Ministry of Education and Research), by the EU project Cleopatra (GA 812997) and by the Marie Skłodowska-Curie GA 801522 at the ADAPT SFI Research Centre (grant 13/RC/2106).

References

1. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research 13 pp. 281–305 (2012)
2. Bergstra, J., Rémi Bardenet, Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. Neural Information Processing Systems pp. 2546–2554 (2011)
3. Charniak, E., Johnson, M.: Coarse-to-fine n-best parsing and maxent discriminative reranking. In: Annual Meeting on Association for Computational Linguistics. pp. 173–180. ACL'05 (2005)
4. Deng, L., Li, X.: Machine learning paradigms for speech recognition: An overview. Transactions on Audio, Speech, and Language Processing **21**(5), 1060–1089 (2013)
5. Feurer, M., Springenberg, J.T., Hutter, F.: Initializing bayesian hyperparameter optimization via meta-learning. AAAI Conference on Artificial Intelligence (2015)
6. Guodong Guo, Li, S.Z., Kapluk Chan: Face recognition by support vector machines. In: Int. Conf. on Automatic Face and Gesture Recognition (2000)
7. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. Lecture Notes in Computer Science (LNCS), Vol. 6683 p. 507–523 (2011)
8. Jamieson, P., Talwalkar, A.: Non-stochastic best arm identification and hyperparameter optimization. AISTATS (2015)
9. Krizhevsky, A.: Learning multiple layers of features from tiny images. Technical report, Department of Computer Science, University of Toronto (2009)
10. LeCun, Y., Bottou, L., Bengio, Y., , Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324 (1998)
11. Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., Talwalkar, A.: Hyperband: A novel bandit-based approach to hyperparameter optimization. ICLR (2017)
12. Moshkelgosha, V., Behzadi-Khormouji, H., Yazdian-Dehkordi, M.: Coarse-to-fine parameter tuning for content-based object categorization. In: Int. Conf. on Pattern Recognition and Image Analysis (IPRIA). pp. 160–165. IEEE (2017)
13. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Y Ng, A.: Reading digits in natural images with unsupervised feature learning. NIPS (2011)
14. Rejani, Y.I.A., Selvi, S.T.: Early detection of breast cancer using SVM classifier technique. CoRR **abs/0912.2314** (2009)
15. Rolland, P., Scarlett, J., Bogunovic, I., Cevher, V.: High-dimensional bayesian optimization via additive models with overlapping groups. AISTATS (2018)
16. Sadeghi, A., Graux, D., Yazdi, H.S., Lehmann, J.: MDE: multi distance embeddings for link prediction in knowledge graphs. In: 24th European Conf. on Artificial Intelligence (ECAI) (2020)
17. Snoek, J., Larochelle, H., Adams, R.: Practical bayesian optimization of machine learning algorithms. In Neural Information Processing Systems (NIPS) (2012)
18. Vilalta, R., Drissi, Y.: A perspective view and survey of meta-learning. Artificial Intelligence Review **18**(2), 77–95 (2002)
19. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)