AUTOML WITH ENSEMBLE OF DEEP NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Automated Machine Learning with ensembling seeks to automatically build ensembles of Deep Neural Networks (DNNs) to achieve qualitative predictions. AutoML and Ensemble of Deep Neural Network produce qualitative results but they are computing intensive methods in both building and inference run time. Therefore, an ideal method would produce at one AutoML run time different ensembles regarding the desired trade-off between accuracy and inference speed. Despite multiple initiatives for non-deep machine learning have been proposed there still no consensus on how to construct efficiently ensemble of deep neural networks. First, we evaluate the performance and robustness of multiple procedures. Then, an ensemble selection method SMBOF is proposed to generate efficient ensembles by controlling their computing cost. Finally, we propose an AutoML workflow using SMBOF that we compare to several baselines. It shows robust results leveraging multiple modern GPUs on two datasets but is not limited to.

1 INTRODUCTION

Deep Neural networks (DNNs) are notoriously difficult to tune, train, and ensemble to achieve qualitative results. Automatic machine learning with ensembling or "AutoML+ensembling" provides a simple interface to train and evaluate many ensembles of DNNs to achieve high accuracy by reducing overfitting.

Nowadays, multiple researchers and practitioners have well understood the benefit of homogeneous ensembles of DNNs to reduce overfitting Sollich & Krogh (1995). Then, authors Liao & Moody (1999) Gashler et al. (2008) propose heterogeneous ensembles to not only reduce variance but attempt to cancel out their different biases too at the cost of a more complex procedure. For example, in cyber-attack detection Haider et al. (2020), time series classification Pathak et al. (2018), medical image analysis Cea et al. (2021), semi-supervision Sun & Ge (2021) and unbalanced text classification Sun et al. (2020). Further, several winners and top performers on challenges routinely use ensembles to improve accuracy Guyon et al. (2019). However, ensembles of DNNs suffer from three main limitations to be widely deployed in research and industrial applications.

The first limitation is a lack of understanding about the best way to build base DNNs to construct an ensemble. Multiple AutoML workflows Feurer et al. (2015) Thornton et al. (2012) LeDell & Poirier (2020) Chen et al. (2018) have been invented aiming at vector classification or vector regression to build base models and ensemble them to reduce overfitting. They never have been generalized at scale on deep learning and yet they are models the most affected by overfitting due to a large number of parameters. It is reasonable to think HPO algorithms can generate DNNs but the choice of which algorithm to create the best library to then combined needs to be studied experimentally. We distinguish also multiple ensembling procedures. Post-hoc ensembling where ensembles are built from a library of previously trained DNNs by HPO with a combinatorial optimizer, and ad-hoc ensembling where hyperparameters encode directly the ensemble's hyperparameters. The ensembling procedure may either build homogeneous ensembles (same hyperparameters but different run times) or heterogeneous heterogeneous ensembles (each DNN has different hyperparameters). Finally, different combination rules exist when working with DNN, majority voting, averaging, and weighted averaging Hashem (1997). AutoML+ensembling is still unexplored in this context of deep learning.

The second limitation is the lack of control of the computing cost of the produced ensemble. In the previous works, authors apply ensembling Caruana et al. (2004) Feurer et al. (2015) of non-deep machine learning and propose that the number of models to put in the ensemble as a threshold between accuracy and computing cost. In DNNs such as applied on image recognition, the combined models are heterogeneous with orders of magnitude of difference in inference time. That is why the number of models as the constraint is not relevant to control the ensemble computational cost. Moreover, when generating automatically multiple models it is known that Johnston et al. (2017) there is no clear correlation between accuracy and DNNs cost, meaning that sometimes fast DNNs can be prioritized without significantly lowering the accuracy. Because today AutoML+ensembles are not computing cost-aware they generally fail to found efficient ensembles. An ideal AutoML+ensembling should found a good trade-off between multiple small DNNs or a few large DNNs.

It is time to address this missing piece in deep learning pipelines between AutoML of Deep Neural Network and ensembles. To summarize our contributions, we run extended benchmarks with hyperparameter optimization, ensemble construction, and combination rule. We propose a new simple algorithm named SMOBF (scalarized-multi objective with budget greedy forward) to post-hoc build an efficient ensemble based on their accuracy and a desired maximum computing cost measured as the sum of their time to predict 2000 images.

This paper is as follows. In section II we go into further detail in the related fields: AutoML and AutoML+ensembling. In section (III) the different steps of the workflow AutoML+ensembling are analyzed and introduced. In section (IV) we analyze our AutoML procedure compared to some baselines.

2 Analysis of the AutoML field

2.1 AutoML of one model

The empirical nature of research in Deep Learning leads scientists to try many model architecture settings, optimization settings, and pre-processing settings to find the best-suited one for data. AutoML is made of 3 modules, the DNNs search space (the "hyperparameter space"), the DNNs sampling strategy (or "hyperparameter optimization"), and the evaluation phase consisting in returning the score associated with one hyperparameter value.

Any model previously trained with hyperparameter λ sampled from hyperparameter space Λ is written M_λ. The hyperparameter optimization goal defined in equation 1 consists in finding the best hyperparamer $\lambda* \in \Lambda$ building a model to reduce the error measured by E. The error is measured on the validation data x_{valid} matching labels y_{valid}.

$$\lambda* = \arg\min_{\lambda \in \Lambda} E(M_\lambda(x_{valid}), y_{valid}) \tag{1}$$

In the literature, AutoML algorithms are typically compared based on their results in the evaluation phase. While this may seem intuitive, this field is now facing multiple methodological questions Li & Talwalkar (2020) on the relevance of comparing multiple workflows made of different stages and different initial conditions. And more, to fairly compare their robustness, multiple datasets, and multiple random seeds must be reported which is a computing-intensive research area limiting initiatives.

Therefore, our paper does not claim the superiority of our AutoML+ensembling method in all cases but it is rather the first step toward AutoML+ensembling of DNNs aiming at the desired trade-off between ensembles accuracy and their computing cost for practical usage. Our work is applied on the computed vision task but can be generalized on other tasks where qualitative prediction is required leveraging one or more clusters of GPUs.

No Free Lunch theorem Wolpert & Macready (1997) proves that no black-box hyperparameter optimization (HPO) can show superior performance to random search in all cases. Nevertheless, early stopping Prechelt (1998) Li et al. (2017) has been also proposed to stop the less promising trials to focus hardware on the most promising trials accelerating the overall optimization process. Recently, FLAML Wang et al. (2021) has been proposed to search not only hyperparameters based on their score but faster training too.

2.2 AUTOML WITH ENSEMBLING

AutoML which not only selects the best model but combines them is valuable for domains that require the best possible accuracy. Several benchmarks were performed and AutoML+ensembling is today considered as the big AutoML challenge series winner on data points like AutoML challenges Guyon et al. (2019). Previous researches on non-deep machine learning ensembles show that over-fitted machine learning algorithms predictions can be averaged out to get more accurate results Sollich & Krogh (1995). This phenomenon is mainly explained by the Law of Large Numbers which claims that the average of the results obtained from many non-biased trials should be close to the expected value. These results are especially interesting for deep learning models, they are models especially affected by random effects (over-fitting) due to their huge number of parameters.

We observe two main trends in AutoML with ensembling. First, *AutoML+Ad-hoc ensembling* Thornton et al. (2012) LeDell & Poirier (2020) Guan et al. (2020) Chen et al. (2018) Olson et al. (2016) which consists in searching directly ensembles. The hyperparameter space describes an ensemble generally of fixed size. In the end, the HPO algorithm keeps the best ensemble and wastes all the others. It suffers from a lack of flexibility in the case of deep neural networks, because the number of models in an ensemble is generally fixed, and it is generally unknown much in advance how much computing power is required to get reasonable accuracy. In other words, changing the number of DNNs in the ensembles requires a new AutoML runtime with the new ensemble size.

Second, *AutoML+Post-hoc ensembling* Caruana et al. (2004) Tsoumakas et al. (2008) Feurer et al. (2015) runs a standard HPO algorithm providing a library of trained models, then constructs ensembles from the library based on a greedy algorithm often named "forward greedy". It starts with the empty ensemble and successively adds the best available model to improve the ensemble target metric. The algorithm stops when no available model can improve accuracy.

This approach is flexible in the context of non-deep machine learning because we can produce several ensembles with different numbers of models with the same library of DNNs allowing a trade-off between computing cost and accuracy. However, there is a lack of control in DNN such as applied to image recognition, the combined models are heterogeneous (e.g., different number of layers, different number of filters per convolution, ...) with sometimes an order of magnitude of difference in inference time. Thus, the computing cost is poor controlled. We propose instead a new multi-objective ensemble selection algorithm named SMBOF to efficiently control the produced ensembles' trade-offs between accuracy and computing cost.

Another line of research consists in improving the training speed of ensembles for a given hyperparameter set. Snapshot Learning Huang et al. (2017a) consists to approximate usual homogeneous ensembles with only one training time. Some methods, speed up the workload of one given ensemble with efficient communication scheme Pittman et al. (2018) Guan et al. (2020) and efficiently use large number of GPUs Guan et al. (2020). Those works are complementary to ours. It is useful when the desired hyperparameter set is already known, for instance, it allows to efficiently re-train the ensemble when the training dataset is updated. In post-hoc ensembling, the hyperparameter search workload may be handled like independent DNNs workload, so those efficient ensemble training pipelines are not applicable during the hyperparameter tuning phase.

Efficient hyperparameter optimizers Wang et al. (2021) Li et al. (2020) keeping hardware for promising DNNs are useful in the context of AutoML and in the context of AutoML+Ensembling. We may expect they produce on average more DNNs with the same amount of computation time.

3 PROPOSED WORKFLOW

In this section, we give an overview of the proposed workflow 1 before understanding the nature of our work. Then, we will go into further detail in each step of the proposed AutoML workflow: HPO to construct a library of models, Ensemble Selection to construct ensembles based on their validation accuracy and computing cost, and finally the combination rule.

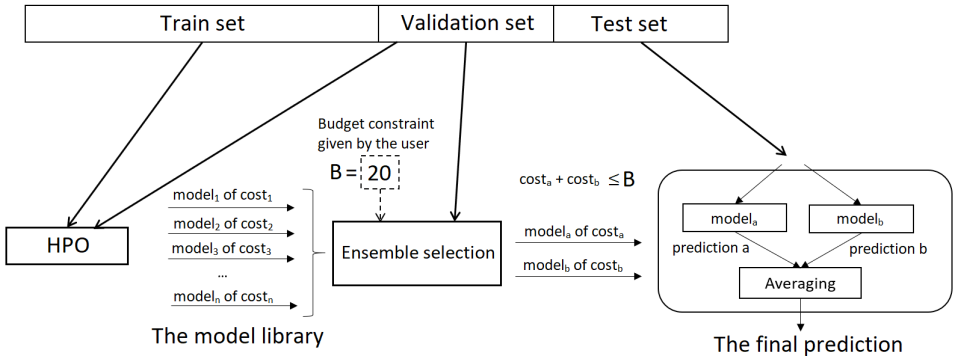

Figure 1: The proposed workflow runs 3 steps 1) The HPO algorithm generates a library of models. The trials are distributed on several GPUs synchronized by a master process. We recommend asynchronous Hyperband based on experimental results. 2) We propose a new multi-objective Ensemble Selection algorithm to search the most efficient ensemble honoring a budget given by the user (here B=20). 3) The returned ensemble predicts by combining (averaging) DNNs predictions on new data.

3.1 Detail of the workflow

Hyperparameter optimization (HPO). The choice of the HPO algorithm has several impacts in terms of the number of produced models at a given time horizon, the accuracy of models, and their diversity. After several experiments, we recommend Hyperband based on our experiments.

In this regard, asynchronous Hyperband Li et al. (2020) is a robust hyper-parameter optimizer to then combine DNNs based on our experiments. It produces the biggest library of models among tested algorithms because most of the sub-optimal models do not reach the maximum number of epochs. It is also a lock-free distributed algorithm until the termination of the algorithm, spending most of its runtime to train models occupying all GPUs and storing them on the library. More models intuitively increase the probability to found good combinations between them during the ensemble selection phase. Then, Hyperband produced also diverse models hyperparameters due to the initial random sampling of hyper-parameters. Finally, Hyperband performs explore-exploit in the time (not in the hyperparameter space). A larger number of models, hardware occupancy, and hyperparametric diversity are three reasons explaining why Hyberband is robust to build a library of models but we still ignore in which proportion they are important. Detailed analysis of the produced ensembles in terms of bias-variance-covariance decomposition Brown et al. (2005) and distribution of the accuracy at multiple runtime and multiple datasets would allow providing more insight.

Ensemble selection. An ensemble selection algorithm finds the best combination possible honoring the budget given by the user. We propose SMOBF greedy standing for "Scalarized Multi-Objective with Budget Forward greedy".

Forward greedy Caruana et al. (2004) starts with the empty ensemble and successively adds the best available model to improve the ensemble target metric. The algorithm stops when no available model can improve accuracy or respect the budget.

We propose new equation 2 to inform the greedy algorithm to favor accurate and cheap DNNs before consuming the overall budget. The current ensemble is a with its computing cost C_a and its predictions y_a. Penalty P_a returns 0 when the budget B is honored ($C_a \leq B$) or an arbitrary large number when it is not to exclude a from the possible ensembles.

The weight w allows controlling the nature of the solution found by the greedy algorithm by placing greater or lesser emphasis on the objectives. The greedy algorithm is run multiple times with different values $w = 0.1$, $w = 0.01$, and $w = 0.001$. Scalarization is a convenient way to handle multi-objective problems by reducing them to a single objective problem, so a simple mono-objective optimization can be performed.

$$score_a = (1 - w) * E(y_a, y) + w * C_a + P_a \tag{2}$$

Then, the best ensemble is picked according to its validation cross-entropy loss and respecting the given budget. To improve the robustness of the method on new applications C_a is standardized and it is the rate between the sum of computing cost of base models (time to predict 2000 images) and the budget B.

Ensemble selection algorithm computes the validation loss of candidate ensembles to evaluate how well a solution will generalize on the test database. Since the data used for validating is taken away from training the individual models, keeping the validating set small is important. Smaller validating sets are however easier to overfit. Contrary to common AutoML on data points datasets, due to the cost of training and evaluating one model on images datasets, we do not repeat the experience with K-fold cross-validation.

In the library of models, some models diverge or have such poor performance compared to other models that they are unlikely to be useful to improve any ensemble building Caruana et al. (2006). Eliminating these models from the library should not reduce the performance and facilitates the ensemble selection task by decreasing the number of non-promising combinations. Pruning works as follows: models are sorted by their performance on the validation metric and only the top X% of them are used for ensemble selection. After pruning, the predictions on the validation set are cached before running the ensemble selection algorithm. This allows handling only predictions vector and not models during the ensemble selection process.

Ensemble combiner rule. We use the simple average as a combiner rule. More advanced methods exist such as "ensemble selection with replacement" Caruana et al. (2004), weighted averaging, and stacking. Those methods are calibrated on a validation set and thus prone to over-fitting.

Now that the workflow is developed, the final accuracy depends on two settings. The tuning effort of the library of models and the budget given by the user to generate a new ensemble.

3.2 Distributed Ensemble Selection

Ensemble selection is a greedy algorithm evaluating all neighbors of a current ensemble at each iteration. For each ensemble a, the equation 2 is performed. To be efficient, this algorithm does not handle directly the models but their predictions which are cached. To assess one ensemble it generally takes 0.5 seconds on one CPU with 100 elements per prediction (100 classes) and 2000 data samples.

In the case of more complex tasks, this procedure can be much more computing-intensive. For instance, in the case of the semantic segmentation images, the class predictions are at pixel scale. A typical dataset for autonomous vehicle applications Cordts et al. (2016) contains 256x256 input images, 30 classes per pixel and 500 validation samples. Linear scaling indicates that the computing cost of this procedure would take 40 minutes (256x256x30x500 predictions elements compared to 100x2000 predictions elements in CIFAR100 taking 0.5 seconds). Because SMOBF is an optimization algorithm we use a similar distributed framework (previous section) to distribute neighbors evaluation on CPUs or GPUs to alleviate this computing cost.

4 Experiments and results

We experiment and discuss our workflow by varying the three steps of the AutoML workflow of deep neural networks on CIFAR100 and the microfossils datasets. More details is given in appendix A.

We are aware the accuracy may be limited by the hyperparameter space and the ResNet framework. Our first goal is to compare different procedures relatively against each other. A promising line of research is searching for more efficient deep neural networks such as EfficientNet Tan & Le (2019). Additionally, combine models with different convolutional block types and not only residual blocks should again increase the diversity of ensembles and reduce the variance and bias errors of the produced ensemble.

4.1 STEP 1 - HPO

4.1.1 HPO ALGORITHM COMPARISON

We compare in table 1 our presented workflow by varying the HPO algorithm to generate a different library of models. Random Search (RS) Bergstra & Bengio (2012), asynchronous Hyperband (HB) Li et al. (2020), Bayesian Optimization with Gaussian Processes (BOGP) Hoffman et al. (2011), BOHB Falkner et al. (2018), Sequential Model-based Algorithm Configuration (SMAC) Hutter et al. (2011), Tree Parzen Estimator (TPE) Bergstra et al. (2011), Genetic Algorithm (GA). To combine generated models, we benchmark them with the SMOBF greedy ensemble selection algorithm.

Not surprisingly, when the budget increases the ensemble selection can find better ensembles from any library. This is explained by the fact that the number of available good combinations between them increases. In the table 1 some algorithms converge and do not found better ensembles after high budgets such as the Bayesian method and genetics ones. More budget reduces also the volatility of this accuracy at different run times. For CIFAR100 the standard deviation for all HPO algorithms ranges from 1 (B=20) to 0.2 (B=320). With micro-fossils it goes from 1.6 (B=125) to 0.4 (B=1000).

A priori, accurate base DNNs, diverse DNNs, and bigger library size are expected to improve the accuracy but we ignore in which proportion, and if this proportion change from one application to another one.

Accurate base DNNs. When we are looking at the distribution of DNNs accuracy we observe that different HPO algorithm choices produce different distributions. Random Search and Hyperband due to the exploration they produce similar distributions after different run times. Bayesian optimization can produce a very large number of similar performing models and different run times produce different local minimum exploitation and thus different distributions. However, in AutoML we are generally interested in the best-produced models and not only the distribution of models, in this case, we observe no method systematically producing the best DNNs in all run time.

And more, because we ensemble models we are not only searching for the most accurate models but the best models to combine. This makes our context original compared to previous works and it justifies our need to compare multiple HPO algorithms.

Diversity of base DNNs. Hyperband and Random Search, due to the exploration of hyper-parameters, are expected to generate more diverse models. However, the hyper-parameter diversity is maybe not always mandatory to generate good ensembles. For instance, in later experiments, we observe ad-hoc homogeneous generally outperforms ad-hoc heterogeneous ensembles.

Bigger library size. Hyperband generally creates a bigger quantity of DNNs compared to HPO without early stopping. The early stopping allows reducing the time before the final epoch is reached. And more, its asynchronous versions allow keeping GPU occupied, in comparison, population-based methods (parallel SMBO or genetics) generally generate a sequence of populations to assess and wait until they all finish to compute the next population letting some GPUs vacant.

Early stopping and asynchronous technics together create generally a bigger quantity of models when the time horizon is fixed. However, this number is unstable at different runtimes due to different DNNs explored and their very different computing requirement.

4.1.2 ENSEMBLE PROCEDURES

We compare in figure 2 different ensemble construction procedures: Post-hoc and ad-hoc generated with Random Search. The ensemble methods may not be worth the inference computing cost, so we compare also the "no ensemble" procedure which is a simple HPO of DNNs. All those procedures were performed by using the random search HPO algorithm to generate hyperparameters during 6 days on 6 GPUs. All of them were run 3 times with different random seeds.

In "ad-hoc" all ensembles have a fixed size of 4 and the produced ensembles. "ad-hoc" ensembles and "no ensemble" DNNs are associated with their validation score and their computing cost. Then, we select the best one for each given budget and it is evaluated on the test score. It is possible to found no model when the budget is too low. On the opposite, when the budget is already high enough, increasing it may produce multiple times the same ensemble. Both cases are mentioned in table 2 with the symbol '-'.

Data	Budget	RS	HB	BOGP	BOHB	SMAC	TPE	GA
C100	20	31.26	27.8	31.58	29.24	**24.44**	29.97	24.61
	40	24.24	**22.87**	28.2	25.6	22.97	26.01	22.91
	60	22.27	21.85	26.91	23.53	22.65	25.44	**21.51**
	80	22.69	21.73	26.2	22.58	22.17	24.25	**21.07**
	120	21.63	**21.55**	24.78	21.86	22.17	23.03	21.95
	160	**20.11**	**21.11**	24.24	21.64	21.87	22.47	21.91
	240	20.7	**20.46**	23.76	21.2	21.87	22.55	21.91
	320	20.64	**20.42**	23.76	21.1	21.87	22.46	21.91
Micro	125	13.45	11.93	14.44	12.18	**10.27**	13.91	10.33
	250	10.98	9.82	10.93	11.71	**9.63**	10.66	9.71
	375	10.84	**9.32**	9.93	10.5	9.45	10.49	9.5
	500	10.23	**8.91**	9.93	9.84	9.27	9.06	9.28
	750	9.7	**8.87**	9.21	9.28	9.18	9.3	9.18
	1000	9.69	**8.46**	8.77	8.8	9.09	9.21	9.07

Table 1: Our workflow error (%) by comparing seven HPO algorithms was run on both datasets for 6 days on 6 GPUs. Each HPO generates a library of models for a given dataset. The budget is expressed as "sum of DNNS times (seconds) to predict 2K images on 1 GPU"

Data	Budget	Posthoc Avg.	ad-hoc Homo. Weighted avg.	ad-hoc Hetero. Weighted avg.	No Ensembles
C100	20	31.26 ± 0.99	30.0 ± 2.86	-	32.89 ± 2.85
	40	24.24 ± 0.96	26.25 ± 0.93	26.31 ± 0.35	30.51 ± 0.95
	60	22.27 ± 0.83	25.49 ± 0.66	25.46 ± 0.57	30.36 ± 0.64
	80	22.69 ± 0.71	-	$23.93*$	
	120	21.63 ± 0.58	-	-	
	160	20.11 ± 0.31	-	-	
	240	20.7 ± 0.18	-	-	
	320	20.64 ± 0.15	-	-	
Micro	125	13.45 ± 1.56	14.70 ± 3.97	14.78 ± 1.22	13.48 ± 3.97
	250	10.98 ± 0.83	12.57 ± 0.74	12.73 ± 0.31	12.82 ± 0.83
	375	10.84 ± 0.7	11.67 ± 0.68	11.78 ± 0.28	$12.55*$
	500	10.23 ± 0.68	-	-	-
	750	9.7 ± 0.61	-	-	-
	1000	9.69 ± 0.42	-	-	-

Table 2: Comparison of four procedures: posthoc ensembles, ad-hoc homogeneous, ad-hoc heterogeneous and no ensembles. We run 3 random seeds. For each run time we generate DNNs during 6 days on 6 GPUs.

The proposed workflow allows finer control of the ensemble built and generally outperforms other methods.

4.1.3 EFFECT OF TUNING TIME

In AutoML we often want to see the evolution of performance over the tuning time, not only the final performance after a time horizon. Therefore, we assess the workflow varying the budgets until it does not affect the ensemble construction anymore and at different tuning time snapshots. Those figures are presented in figure 2 3.

First, regarding the benefit of the tuning time, we observe two main trends. When the budget is very small (B=20) models accuracy converges early to 18, 24, or 48 hours reaching the limit of the hyperparameter optimization with a little (or without) ensembling. When the budget is bigger the exploding number of available combinations of ensembles leads to the discovery of better ensembles. The posthoc ensembling is a promising line of research that deserves more attention and more understanding of how DNNs interact with each other.

Then, we observe that increasing the budget systematically leads to an increasing accuracy (colored lines are rarely crossed) but this trend decline. For example, in the figure 2 we show that when Hyperband is finished, the benefit is obvious from 1 to 2 models: +5 point of accuracy percentage, but the improvement is small from B=120 to B=320: +1 point of accuracy percentage.

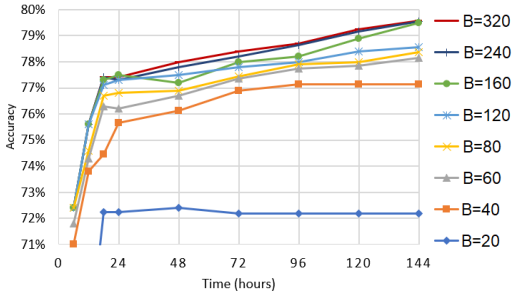

Figure 2: Ensemble from Hyperband algorithm on the CIFAR100 dataset

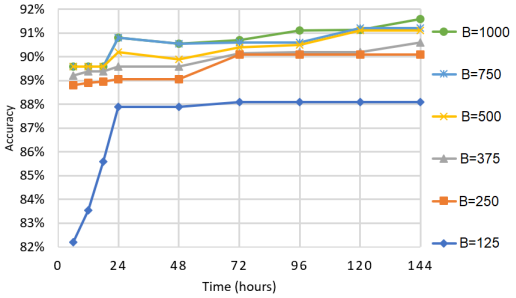

Figure 3: Ensemble from Hyperband on the microfossils dataset

4.2 STEP 2 - ENSEMBLE SELECTION

4.2.1 ENSEMBLE SELECTION PRUNING

We try multiple pruning factors X such X% only the top X% are kept. It reduces the size of the library of models and thus helps the ensemble selection algorithm. When the pruning factor is above 20% it does not reduce accuracy and reduce the ensemble selection time, while the threshold under 15% reduces the target metric in some experiments. Therefore, for all experiments of this paper, the pruning factor is set to 20%.

4.2.2 ENSEMBLE COMBINERS

We compare averaging rule and weighted averaging rule Hashem (1997) in this context of AutoML+ensembling for ad-hoc homogeneous and ad-hoc heterogeneous. To compute weights, we grid search multiple values in $\{1, 2, 4, 8\}$ then weights are normalized. The ad-hoc ensembles have a fixed size 4 and 4 possibles weight values making 256 combinations, 175 after redundancy elimination taking about 80 seconds to compute.

The first observation is that weighted averaging significantly improves heterogeneous ensembles +1.50 point of accuracy compared to simply averaging. The gain is only +0.26 point of percent with ad-hoc homogeneous AutoML+ensemble. This performance gap may be explained by the fact heterogeneous hyperparameters produce diverse accuracy between base DNNs and a calibration step is needed to give more weights to the best performing models, this is less important with homogeneous DNNs producing about the same accuracy. However, this grid search computing quickly becomes intractable when the number of models is larger.

In posthoc ensembling, two procedures can be applied to give more weights to some DNNs with reasonable computing costs. First ensemble selection with replacement Caruana et al. (2004) consists in making possible the selection of multiple times the same DNN of the library. The other method consists in using simply the averaging combination to run posthoc ensembling and return one ensemble, and then calibrate the weights of this ensemble to perform weighted averaging. Both procedures fail to improve accuracy compared to the averaging rule. The ensemble selection method evaluates thousands of ensembles on the validation dataset to return the best one. Therefore, using again validation set to tune weights seems to overfit the validation dataset too much.

Finally, we evaluate majority voting, it is a method ignoring uncertainty estimate so it can lose useful information during the combination. On the other hand, it is less sensitive to DNNs which are over-confident or under-confident relative to the others. Our experiments show this rule underperforms averaging rule to -0.87 point of percentage on different budgets and both datasets.

It appears that posthoc ensembling with SMBOF efficiently controls the computing cost and accuracy by controlling DNNs to gather. A more complex combination rule seems useless to calibrate ensembles because the ensemble construction is already calibrated by evaluating a large number of combinations.

4.2.3 SMOBF GREEDY COMPARED TO FORWARD GREEDY (BASELINE)

The ensemble selection algorithm often used by most advanced AutoML software Caruana et al. (2004) Feurer et al. (2015) is *forward greedy with fixed number of models* that we pick as baseline. We perform three runtimes of the overall AutoML workflow and observe that SMOBF is Pareto dominant or equivalent to the baseline each time.

The figures 4, 5 compares those two algorithms in function of the cost (vertically) and the error rate (horizontally) of produced ensembles with SMOBF greedy (blue) and the baseline (orange) on one runtime. In those figures "BX" mention means a budget of X and "#Y" means an ensemble of size Y.

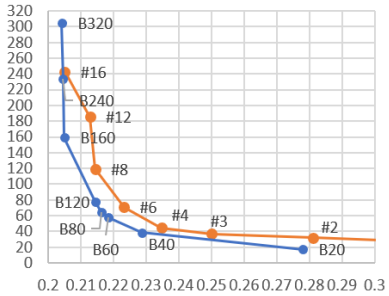

Figure 4: Ensembles generated from Hyperband algorithm on the CIFAR100 dataset

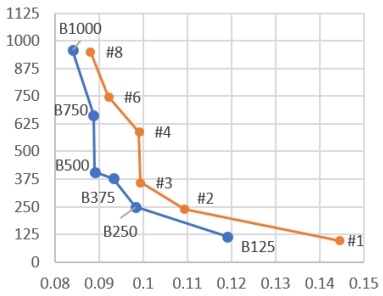

Figure 5: Ensembles generated from Hyperband algorithm on the microfossils dataset

The gain of SMOBF is particularly obvious when the budget is small on both criteria, but it is reduced when the budget is relaxed. Indeed, when we target the best ensemble at any cost or any size, the objective of the two algorithms converges. On the opposite when the budget is small, SMOBF informs the greedy algorithm to go toward efficient models before the budget is consumed.

CONCLUSION

Due to the increasing number of new Deep Learning applications and datasets, Auto Machine Learning (AutoML) methods are an important line of research. We propose an AutoML workflow capable to tune, train, ensembling, and deploy DNNs automatically but that runs a heavy workload at each stage. We aim to fill the gap between Machine Learning researches, the new GPU clusters, and the end-user application quality of service. To go toward this direction, we formulate the problems by aiming at the accuracy, the inference speed, and the flexibility of the underlying heterogeneous infrastructure. First, we presented the experimental results demonstrating that asynchronous Hyperband is suitable for parallelism and generates the best library of models to ensemble them. We then propose a new Ensemble Selection strategy that allows controlling the final ensemble computing cost of heterogenous DNNs.

In future work, we intend to propose an open-source framework based on Ray framework to leverage modern GPU clusters. The design of this workflow allows the application to a large number of applications by adapting the combination rule, the hyper-parameter space, and the deep neural network framework.

REFERENCES

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305, 2012. URL `http://dblp.uni-trier.de/db/journals/jmlr/jmlr13.htmlBergstraB12`.

James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 24*, pp. 2546–2554. Curran Associates, Inc., 2011. URL `http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf`.

Gavin Brown, Jeremy Wyatt, Rachel Harris, and Xin Yao. Diversity creation methods: A survey and categorisation. *Information Fusion*, 6:5–20, 03 2005. doi: 10.1016/j.inffus.2004.04.004.

Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, pp. 18, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138385. doi: 10.1145/1015330.1015432. URL https://doi.org/10.1145/1015330.1015432.

Rich Caruana, Art Munson, and Alexandru Niculescu-Mizil. Getting the most out of ensemble selection. pp. 828–833, 12 2006. doi: 10.1109/ICDM.2006.76.

María V. Sainz de Cea, David Gruen, and David Richmond. Pneumoperitoneum detection in chest x-ray by a deep learning ensemble with model explainability. In *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, pp. 1637–1641, 2021. doi: 10.1109/ISBI48211.2021.9434122.

Boyuan Chen, Harvey Wu, Warren Mo, Ishanu Chattopadhyay, and Hod Lipson. Autostacker: A compositional evolutionary learning system. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, pp. 402–409, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356183. doi: 10.1145/3205455.3205586. URL https://doi.org/10.1145/3205455.3205586.

François Chollet. Xception: Deep learning with depthwise separable convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1800–1807, 2017.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1436–1445, 2018.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 28*, pp. 2962–2970. Curran Associates, Inc., 2015. URL http://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning.pd

Michael Gashler, Christophe Giraud-Carrier, and Tony Martinez. Decision tree ensemble: Small heterogeneous is better than large homogeneous. *2008 Seventh International Conference on Machine Learning and Applications*, pp. 900–905, 01 2008. doi: 10.1109/ICMLA.2008.154.

Hui Guan, Laxmikant Kishor Mokadam, Xipeng Shen, Seung-Hwan Lim, and Robert Patton. Fleet: Flexible efficient ensemble training for heterogeneous deep neural networks. In I. Dhillon, D. Papailiopoulos, and V. Sze (eds.), *Proceedings of Machine Learning and Systems*, volume 2, pp. 247–261, 2020. URL https://proceedings.mlsys.org/paper/2020/file/ed3d2c21991e3bef5e069713af9fa6ca-Pape

Isabelle Guyon, Lisheng Sun-Hosoya, Marc Boullé, Hugo Jair Escalante, Sergio Escalera, Zhengying Liu, Damir Jajetic, Bisakha Ray, Mehreen Saeed, Michèle Sebag, Alexander Statnikov, Wei-Wei Tu, and Evelyne Viegas. *Analysis of the AutoML Challenge Series 2015–2018*, pp. 177–219. Springer International Publishing, Cham, 2019. ISBN 978-3-030-05318-5. doi: 10.1007/978-3-030-05318-$5_1 0.URL$

A EXPERIMENTAL DATA SET AND HYPERPARAMETER SETTINGS

This section describes machine learning experiments for reproducibility purposes.

A.1 THE TWO DATASETS USED

The CIFAR100 dataset. CIFAR100 Krizhevsky (2009) consists of 60,000 32x32 RGB images in 100 classes. For each class, there are 580 training images, 20 validation images and 100 testing images.

The Microfossils dataset. Microfossils are extremely useful in age dating, correlation, and paleo-environmental reconstruction to refine our knowledge of geology. Microfossil species are identified and counted on large microscope images and thanks to their frequencies we can compute the date of sedimentary rocks.

To do reliable statistics, a big number of objects needs to be identified. That is why we need deep learning to automate this work. Today, thousands of fields of view (microscopy imagery) need to be shot for 1 rock sample. In each field of view, there are hundreds of objects to identify. Among these objects, there are non-fossils (crystals, rock grains, etc...) and others are fossils that we are looking for to study rocks.

Our dataset contains 91 classes of 224x224 RGB images (after homemade preprocessing). Microfossils are calcareous objects took with polarized light microscopy. The classes are unbalanced, we have from 50 images to 2500 images by class, with a total of 32K images in all the datasets. The train/validation/test split is as following: 72% 8% 20%. The F1 score was used and labeled as 'accuracy' on all benchmarks.

A.2 HYPERPARAMETER CONFIGURATION SPACE

Table 3 shows all hyperparameters properties in this workflow. We use ResNet-based architectures due to their simplicity to yield promising and robust models on many datasets. We explore different residual block versions: "V1", "V2" He et al. (2016) and "next" Xie et al. (2017). Regarding the optimization method, we use Adam optimizer Kingma & Ba (2014) due to its well-known performance and its lower learning rate tuning requirement.

The simplicity of the ResNet architecture makes this work easy to test by a scientist on its image dataset Zagoruyko & Komodakis (2016). Exploring other convolutional block type like VGG Zhang et al. (2016), Xception Chollet (2017), DenseNet Huang et al. (2017b) and EfficientNet Tan & Le (2019) are potential improvement which could increase the degree of liberty in DNNs construction, the diversity of DNNs and the accuracy of ensembles.

The CIFAR100 dataset contains 32x32 images while usually ResNet is adapted to be used on ImageNet (224x244 images). Those different resolutions need some adaptation. Therefore, on the CIFAR100 case, the first convolutional layer is replaced from the 7x7 kernel size with a stride of 2, to a 3x3 kernel size with a stride of 1. With equivalent settings, our CNN framework produces nearly the same number of weights between the CIFAR100 and microfossils dataset, but the time complexity is factor 11 different because of different signal resolutions flowing through the layers.

Category	Name	Type	Range
Optim.	Learning rate	Continuous	[0.001; 0.01]
	Batch size	Discrete	[8; 48]
	L2 regularization factor	Continuous	[0; 0.1]
NN archi.	Convolution type	Categorical	{v1,v2,next}
	Activation function	Categorical	{tanh,relu,elu}
	Number of filters in the first convolutional layer	Discrete	[32; 128]
	Multiplier of filters in the 4 blocks	Discrete	[32; 128]
	Number of convolutional block in the first stage	Discrete	[1;11]
	Number of convolutional block in the second stage	Discrete	[1;11]
	Number of convolutional block in the third stage	Discrete	[1;11]
	Number of convolutional block in the fourth stage	Discrete	[1;11]
Data augment.	Max zoom	Continuous	[0; 0.6]
	Max translation	Continuous	[0; 0.6]
	Max shearing	Continuous	[0; 0.3]
	Max channel shifting	Continuous	[0; 0.3]
	Max rotation measured in degrees	Discrete	[0; 90]

Table 3: The hyperparameter space experimented based on the ResNet neural architecture framework

POST-HOC HOMOGENOUS OF DNNS WITH POPULAR RESNET ARCHITECTURES

Tables 4 and 5 show the workflow error by replacing the library from any HPO algorithm with a library from a fixed ResNet architecture. Then SBMOF is applied to build ensembles. Libraries have been generated at different run times for 6 days and 6 GPUs.

Results show it underperforms the post-hoc ensembles of heterogeneous DNNs generated with an HPO. The second observation is that a homogeneous ensemble of DNNs benefits also from the ensemble selection algorithm.

arch.	resnet18	resnet34	resnet50	resneXt50	resnet101	resnet152
#weights	31.6M	53.1M	53.1M	13.1M	95.8M	128.4M
cost	10.83	14.35	14.05	11.87	21.14	28.56
#1	**35.30**	35.77	37.63	42.28	35.85	37.82
#2	32.98	**21.70**	33.60	38.67	32.77	34.86
#3	29.88	**28.58**	31.29	36.16	30.30	32.16
#4	28.66	**27.63**	29.76	35.09	29.74	31.65
#6	27.70	**26.88**	28.16	32.15	28.93	30.26
#8	26.98	**26.21**	27.33	31.86	28.26	29.33
#12	26.59	**25.88**	26.82	31.44	71.85	29.46
#16	26.66	**25.74**	26.80	31.44	28.15	29.45

Table 4: Comparison different ResNet architectures and ensemble size on the CIFAR100 dataset. The cost is expressed in second at inference time.

arch.	resnet18	resnet34	resnet50	resneXt50	resnet101	resnet152
#weights	31.6M	53.1M	53.1M	13.1M	95.8M	128.4M
cost	120.53	169.71	164.96	118.64	254.45	354.29
#1	**13.17**	14.76	14.35	15.09	15.78	14.91
#2	12.26	**12.23**	12.46	12.74	13.96	13.39
#3	11.78	**11.45**	11.52	12.07	12.46	12.34
#4	11.66	**11.29**	11.37	11.98	12.26	12.09
#6	10.90	**10.50**	10.73	11.73	11.76	11.32
#8	10.87	**10.41**	10.61	11.27	11.60	11.20

Table 5: Comparison different ResNet architectures and ensemble size on the microfossils dataset