ENSEMBLE SQUARED: A META AUTOML SYSTEM

Jason Yoo [1] **Tony Joseph** [1] **Dylan Yung** [2] **S. Ali Nasseri** [1] **Frank Wood** [1]

ABSTRACT

The continuing rise in the number of problems amenable to machine learning solutions, coupled with simultaneous growth in both computing power and variety of machine learning techniques has led to an explosion of interest in automated machine learning (AutoML). This paper presents Ensemble Squared (Ensemble^2), a "meta" AutoML system that ensembles at the level of AutoML systems. Ensemble^2 exploits the diversity of existing, competing AutoML systems by ensembling the top-performing models simultaneously generated by a set of them. Our work shows that diversity in AutoML systems is sufficient to justify ensembling at the AutoML system level. In demonstrating this, we also establish a new state of the art AutoML result on the OpenML classification challenge.

1 INTRODUCTION

Advances in computer hardware and the ever-expanding abundance of data are enabling the construction of machine learning models that are yielding progressively more value in a growing set of application domains. Unfortunately, as is well understood, there is no single best machine learning model. This means that for every new application or problem, a laborious, largely manual process must be following that includes data cleaning, feature engineering, and the design, testing, and selection of a machine learning model and pipeline. This is what data scientists do, often in collaboration with domain experts. The demand for data science talent on the job market exceeds supply (Manyika et al., 2011; Pompa & Burke, 2017; McKinsey Analytics, 2016) and for this reason the societal "value add" from machine learning is bottlenecked by the availability of data scientists and the difficulty of applied machine learning.

One solution to this is to automate much of the machine learning model and pipeline selection process using automated machine learning (AutoML). AutoML systems allow data scientists to focus directly on value creation, i.e. solving the underlying problem, while other tasks such as data cleaning, model selection, model fitting, and hyperparameter tuning are handled automatically. The full promise of AutoML is that, eventually, it will enable even non-data-scientists to extract value from their own data.

A variety of AutoML systems exist today (Erickson et al., 2020; Feurer et al., 2015; Heffetz et al., 2020). DARPA even funded a program, in which we participated, called the Data-Driven Discovery of Models (or "D3M") program[1], whose aim was to accelerate the development of systems and expand the variety of approaches to AutoML. The AutoML community has matured sufficiently to have started to coalesce around reporting results on the OpenML classification benchmark (Gijsbers et al., 2019). Our experience in the D3M program combined with reviewing these systems and their comparable reported results led us to ask the following question: Ensembling individual models has been shown to be an extremely powerful idea (Hansen & Salamon, 1990); could it be that there is sufficient diversity in AutoML systems so that ensembling AutoML systems (really ensembling models and pipelines derived from a collection of systems) would lead to quantitative gains?

This is the question, or hypothesis, that this paper addresses. Ensemble^2, the system we build, ensembles machine learning pipelines searched for in parallel by a set of AutoML systems, and achieves a new state of the art baseline on an OpenML classification benchmark. To our knowledge no other existing work ensembles results among different AutoML systems (our base learners), exploiting the variability between them in terms of pipeline search, model selection, and hyperparameter optimization strategies.

To be clear we are aware that numerous AutoML solutions already rely on *internally* ensembling machine learning model pipelines discovered during their internal search phase (Feurer et al., 2015; 2020; Erickson et al., 2020; Chen et al., 2018; Zaidi et al., 2020), and some even go so far as to explicitly search for best base learners to ensemble. They do so for the same reason that we explore ensembling AutoML systems *externally*: ensembling can reduce both bias and variance of a machine learning models' predictions.

[1]University of British Columbia, Vancouver, Canada [2]Georgia Institute of Technology, Atlanta, GA, USA. Correspondence to: Jason Yoo <jasony97@cs.ubc.ca>.

[1]https://www.darpa.mil/program/data-driven-discovery-of-models

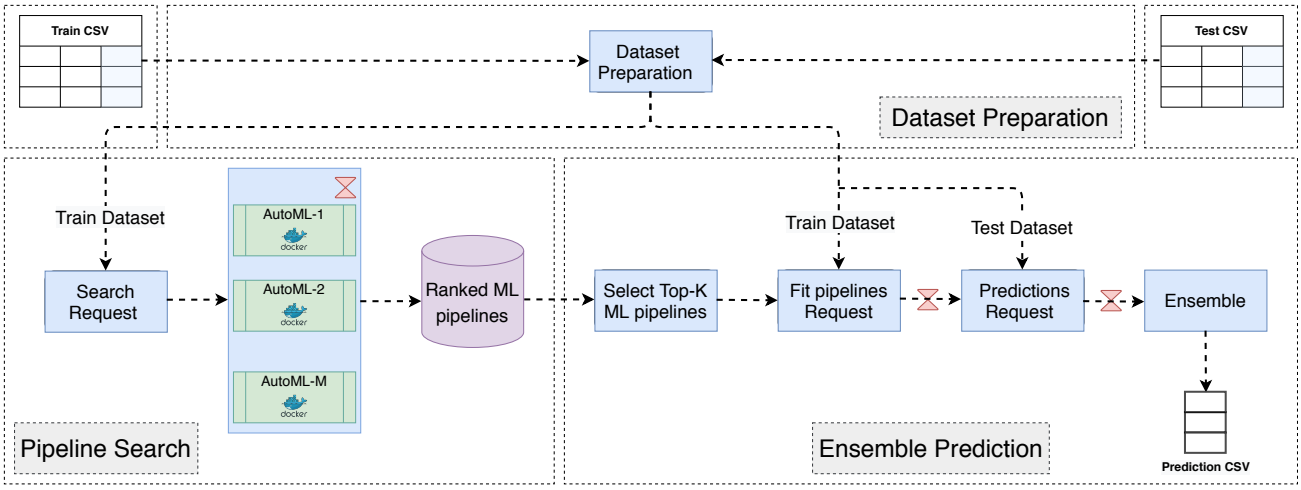

Figure 1. Overview of Ensemble^2 workflow. Ensemble^2 is made up of three underlying subsystems. First, the *Dataset Preparation subsystem* takes the training dataset comma separated value (CSV) file along with a target column, performs dataset curation to identify the problem and column types (if not provided), and converts it to a format acceptable to the AutoML systems. The *Pipeline Search subsystem* spins up M different AutoML systems (our base learners) as docker containers and performs the pipeline search procedure in parallel for a set time duration. Once the time limit has been hit and all discovered pipelines (P) have been collected, it ranks the pipelines based on their validation scores. The ranked pipelines, and the prepared training and test datasets are then passed to the third subsystem, the *Ensemble Prediction subsystem*. Here the top K pipelines (where K is a hyperparameter and $K \leq P$), are selected and fitted to the given training dataset. After the fitting process, predictions are produced for the test dataset for all K pipelines. The predictions are then passed onto the ensembling module, which generates the final Ensemble^2 predictions using majority voting.

Ensemble^2 has an additional benefit of being more robust than its base learners. Existing AutoML systems are surprisingly brittle. Because different AutoML systems apply different pre-processing steps, they can fail to return a solution on some datasets (such as ones with a lot of missing values) (Zöller & Huber, 2019). Ensemble^2, by its construction, is trivially more robust than all of its underlying AutoML systems and only fails when all base systems fail.

Figure 1 presents an overview of Ensemble^2. Our system consists of two subsystems, one that performs pipeline search using the base learners in parallel and ranks all returned pipelines, and another that ensembles results from a specific number of top pipelines using majority voting. Table 2 shows Ensemble^2's performance on the OpenML benchmark test datasets relative to the AutoML systems it ensembles. This table shows that it achieves the highest average rank overall and that it outperforms notable current AutoML systems such as AutoGluon (Erickson et al., 2020) and Auto-Sklearn-2 (Feurer et al., 2020).

2 Background

Ensemble^2 relies on ensembling the best pipelines from 6 base AutoML systems: AlphaD3M (Drori et al., 2019), Axolotl[2], CMU AutoML[3], Auto-Sklearn (Feurer et al., 2015), Auto-Sklearn 2.0 (Feurer et al., 2020), and AutoGluon (Erickson et al., 2020). These systems were selected for their differing search strategies and search spaces. Table 1 contains a brief overview of these systems and the different approaches they take to the automated pipeline search and optimization problem.

2.1 Problem Formulation

While there are several definitions of the AutoML problem, our work follows the definition in (Zöller & Huber, 2019). Let a machine learning pipeline $P : \mathcal{X} \rightarrow \mathcal{Y}$ be a sequential combination of algorithms that transforms a feature vector $x \in \mathcal{X}$ to a target value $y \in \mathcal{Y}$. For example, y is an one-hot vector of class labels for a classification problem and a real number for a one-dimensional regression problem. Let $\mathcal{A} = \{A^{(1)}, A^{(2)}, ..., A^{(n)}\}$ be a fixed set of data-cleaning, feature pre-processing, and estimator algorithms where each algorithm $A^{(i)}$ is configured by a set of hyperparameters $\lambda^{(i)}$ from the domain $\Lambda^{(i)}$. Then, P's structure can be described as a Directed Acyclic Graph (DAG) where each node is an algorithm $A^{(i)}$ and each edge represents the data flow between algorithms.

The objective of an AutoML system is to find the configura-

[2] https://gitlab.com/axolotl1/axolotl
[3] https://github.com/autonlab/cmu-ta2

AutoML System	Primitive Library	Model Discovery and Hyperparameter Tuning	Internal Ensembling
AlphaD3M	D3M	Deep RL	-
Axolotl	D3M	Templates + BayesOpt	-
CMU AutoML	D3M	Templates + Grid Search	-
Auto-SkLearn	SKlearn	BayesOpt + Meta-Learn	Forward Search
Auto-SkLearn 2.0	SKlearn	Portfolio Learning	Forward Search
AutoGluon	Gluon	Fixed Defaults (Set Adaptively)	Multi-Layer Stacking and Bagging

Table 1. Base AutoML Systems used in Ensemble^2. This table highlights some of the differences between these AutoML systems and the diversity of methods that Ensemble^2 benefits from.

tion of algorithms and hyperparameters that minimizes the loss on an unseen test dataset as following

$$P^* = \text{argmin}_{P \in \mathcal{P}} \mathcal{L}(\mathcal{D}_{train}, \mathcal{D}_{test}, P) \quad (1)$$

where P is a valid pipeline from the space of all valid DAG-structured pipelines $\mathcal{P}$, $\mathcal{L}$ is the task loss, and $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$ are the training and test datasets.

Ensemble^2 estimates P^* above by ensembling the top-k best pipelines discovered by its base AutoML systems. The top-k is evaluated by choosing the k models with the lowest loss on the validation dataset. Once the top-k is chosen any ensembling strategy can be used on the set of top-k models. The base AutoML systems partition $\mathcal{D}_{train}$ into training and validation sets respectively based on their own system.

2.2 Ensemble Learning

Ensembles are commonly used to boost performance by reducing model bias and variance (Rahman & Tasnim, 2014). Popular ensemble architectures often utilize one of voting, bagging, boosting, and stacking techniques. Voting involves having the base learners vote on the correct class with equal weight and taking the class with the most votes as the final prediction. Bagging involves independently training the base learners using a randomly drawn subset of the training set and having the base learners vote with equal weight. Boosting involves incrementally constructing an ensemble by training base learners to better classify training data points that the previous base learners misclassified. Lastly, stacking involves training a classifier that learns to combine the predictions of many base learners and make a final prediction.

2.3 Base AutoML Systems

As previously mentioned, six base AutoML systems were used in Ensemble^2. These systems were selected as base AutoML systems because of their strong individual performance as well as their different search strategies and search spaces. The different search strategies are elaborated below, so we only detail their different search spaces in this paragraph. Auto-Sklearn 1.0 & Auto-Sklearn 2.0 restrict their ML algorithms (ex. data-cleaning, feature pre-processing, and estimator) to scikit-learn library (Pedregosa et al., 2011). AlphaD3M, Axolotl, and CMU AutoML restrict their ML algorithms in the D3M library. Lastly, AutoGluon uses a mix of scikit-learn library, neural networks, and custom-made tree-based ML algorithms.

AlphaD3M AlphaD3M (Drori et al., 2019) uses reinforcement learning to tackle the pipeline search problem. It first formulates automated machine learning as a combinatorial optimization problem to find a sequence of machine learning algorithms that performs the best on the test dataset according to a user-defined metric. It then casts the combinatorial optimization problem as a sequential decision-making problem where a machine learning pipeline is iteratively built one component at a time.

AlphaD3M uses AlphaZero's Monte Carlo Tree Search (MCTS) and neural network setup (Silver et al., 2017). MCTS asymmetrically explores promising trajectories by nature and balances between exploration and exploitation. Neural networks enhance the MCTS runtime by guiding the search process with a learned heuristic and providing state value estimates. These techniques enable AlphaD3M to run a diverse set of pipelines while biasing its search around pipelines that performed well on validation dataset.

Reinforcement learning solutions are quite sample-inefficient. To mitigate these problems, AlphaD3M trains its neural networks in an offline manner on a set of meta-training datasets. At test time, AlphaD3M runs MCTS with the trained neural network on an unseen dataset while continuing to update the neural network weights occasionally.

Axolotl Axolotl takes the middle ground between a template-based approach and Bayesian optimization. To find promising pipelines in a short amount of time, Axolotl fixes its pre-processing steps and cycles through its estimator search space. Hyperparameters are set to default values as the goal isn't to tune every pipeline but to identify k most promising estimators that, combined with the fixed

pre-processing steps, score the highest on the validation dataset. Once the k best pipelines have been found, Axolotl initiates a Gaussian Process Bayesian optimization on each of those pipelines. Axolotl aims to achieve strong initial performance through quick template-based evaluations and strong asymptotic performance by applying Bayesian optimization on promising pipelines over time. However, the use of templates constrains the search space and means that not all possible pipeline configurations are explored.

CMU AutoML CMU AutoML takes a template approach to the pipeline synthesis problem. More specifically, it uses multiple hand-crafted pre-processing steps and cycles through all possible pre-processing and estimator combinations to quickly assess the optimality of many pipelines in a short amount of time. This follows from the authors' finding that foregoing hyperparameter search and instead using that time to evaluate as many model as possible was able to achieve superior performance when dealing with short search times. However, some key pipeline component hyperparameters are grid-searched.

AutoGluon AutoGluon (Erickson et al., 2020) employs a unique stacking technique called Multi-Layer Stack Ensembling alongside k-fold ensemble bagging. Multi-Layer Stack Ensembling involves stacking models in multple layes and training in a layer-wise manner which would guarantee high-quality predictions within a given time frame. To prevent overfitting, Autogluon relies on K-fold ensemble bagging at all layers.

Auto-Sklearn Auto-Sklearn 1.0 (Feurer et al., 2015) uses Bayesian optimization to search through the model and hyperparameter space of selected scikit-learn (Pedregosa et al., 2011) modules. Meta-learning is used to warm start the search procedure. When the model search is over, Auto-Sklearn 1.0 constructs an ensemble from discovered pipelines by employing forward search (Caruana et al., 2004).

Auto-Sklearn 2.0 (Feurer et al., 2020) improves upon Auto-Sklearn 1.0 by employing portfolio learning. It first runs Bayesian optimization on all meta-train datasets and constructs a portfolio of pipelines to run on all future tasks. In addition, it learns what model selection strategy to use on a new dataset based on simple meta-features involving some combination of k-fold validation, validation holdout set, successive halving (Jamieson & Talwalkar, 2016), and more. It also uses various other improvements such as intermittent result storage to be able to return results quicker for larger datasets in a short amount of time.

3 METHODOLOGY

3.1 Ensemble Construction

While there are many possible ensemble architectures for a multi-class classification problem with $|\mathcal{C}|$ different classes, Ensemble^2 simply employs the majority voting scheme for now since the primary goal of this paper was to see whether there is any merit in combining multiple AutoML systems. Other architectures based on bagging and stacking are also possible. Here, we briefly describe the majority voting scheme.

Let us define the feature vector space as $\mathcal{X}$, and the $|C|$-dimensional one-hot target vector space as $\mathcal{Y}$. We denote $P : \mathcal{X} \rightarrow \mathcal{Y}$ to be a machine learning pipeline with fully defined algorithm and hyperparameter values. Let $\mathbf{P} = \{P_1, P_2, ..., P_N\}$ be a set of N machine learning pipelines from the base AutoML systems that achieves the lowest loss on the validation dataset when the search procedure is over. Lastly, we denote the indicator function as I. A majority voting ensemble model $M : \mathcal{X} \rightarrow \mathcal{Y}$ assigns the label $\hat{y}$ to a data point x as follows:

$$\hat{y} = \underset{y}{\operatorname{argmax}} \left[\sum_{i=1}^{N} \mathrm{I}(\mathbf{P}_i(x) = y) \right] \tag{2}$$

where $\mathbf{P}_i$ are trained on $\mathcal{D}_{train}$ with the base classifier's loss.

The advantage of the majority voting approach is the fact that no additional training is required aside from training the pipelines in $\mathbf{P}$. It is conceptually the simplest to implement and often yields good results.

4 EXPERIMENTAL SETUP

Ensemble^2 uses standard APIs (including the Data-Driven Discovery of Models (D3M) API[4], and the Auto-SKlearn API[5]) to interface with the base AutoML systems. These standard APIs allow us to seamlessly integrate with any future AutoML systems that follow these standardized APIs. API usage allows for a client-server setup, where the AutoML systems runs on the server side and the user schedules various commands to the AutoML systems from the client side. The AutoML systems are all containerized using Docker, thereby bypassing the concern of installing required packages and managing conflicting dependencies. This also allows us easily scale the system as required and run multiple AutoML systems in parallel. This enables *Ensemble^2* to act as a client and schedule various pipeline search tasks to the base AutoML system containers.

[4] https://gitlab.com/datadrivendiscovery/ta3ta2-api

[5] https://automl.github.io/auto-sklearn/master/index.html

As depicted in 1, there are three main stages in Ensemble^2's workflow: dataset preparation, pipeline search, and ensembling.

4.1 Dataset Preparation

Given a train/test dataset pair, Ensemble^2 converts the datasets into a type that can be used by the relevant system. This process involves assigning correct column types (including inferring column types automatically if that information is not provided), and assigning the relevant semantic type to each feature and target column. The base AutoML systems can have it's own profiler/heuristics to then infer the column type. In addition, it infers what kind of task to perform as well. For example, it may label the task as a binary classification, multiclass classification, or regression. This stage removes any ambiguity regarding the dataset and the task for the base AutoML systems.

4.2 Pipeline Search

Ensemble^2 spins up a Docker container for each of its base AutoML system in parallel. It sends search requests to each base AutoML system to perform the pipeline search procedure for a given duration of time with a specific seed. Discovered ML pipelines along with its validation scores are continuously streamed from the container to Ensemble^2, which in turn saves the information. Once the time limit is over and all discovered pipelines have been collected, Ensemble^2 fits the pipelines with the complete training set and generates test dataset predictions. It saves the predictions for the ensembling procedure.

Ensemble^2 is quite robust; if any AutoML system happens to fail, all of its previously discovered pipelines are saved and it does not influence the performance of other AutoML systems running concurrently. Ensemble^2 only truly fails when all of its base AutoML systems fail without discovering any pipelines. This scenario is not likely in most cases, since different AutoML systems take different precautions to avoid this scenario. Just like how ensembles work better than their individual learners when the learners make independent errors, Ensemble^2 is more robust than any of the base AutoML systems if the AutoML systems have independent modes of failure. This is another reason why we chose AutoML systems that use different approaches to perform pipeline search, as these systems are likely to have different failure modes.

4.3 Ensemble Prediction

While many ensembling schemes are possible at this point, Ensemble^2 simply chooses K pipelines with the highest validation dataset performance and generates the final test dataset predictions using majority voting. In the future, we plan to experiment with many different ensembling schemes such as stacking, bootstrapped majority voting, forward search majority voting, and meta-feature-aware stacking. K is set to 3 for the experiments.

5 Empirical Evaluation

5.1 Datasets

All experiments were performed on the 41 OpenML classification benchmark datasets (Gijsbers et al., 2019). The benchmark was curated such that its member datasets vary in the number of data points and features by orders of magnitudes. The member datasets vary in the number of categorical features, numerical features, and missing values. The benchmark also does not contain classification problems that are too easy to solve (e.g. most artificially generated datasets) and is gradually updated overtime to prevent AutoML tools from overfitting to the member datasets. The evaluation metric is accuracy for all datasets in the benchmark.

5.2 Experimental Setup

For all experiments, each AutoML system was run in parallel to other AutoML systems. Each AutoML system was given access to 4 CPUs and 8GB of RAM. Experiments were performed on four machines with a Intel Core i7-5820K, 3.3 GHz processor, 48 GB DDR4 RAM. All AutoML systems were run for one hour. Unless otherwise mentioned, Ensemble^2 ensembles the top 3 pipelines with the best validation scores from its base AutoML systems.

Ensemble^2 was compared against six state-of-the-art (SOTA) AutoML systems: AutoGluon, Auto-Sklearn, Auto-Sklearn 2.0, AlphaD3M, Axolotl, and CMU AutoML. AutoGluon's score is measured by the performance of its best pipeline, which is its default configuration. Auto-Sklearn and Auto-Sklearn 2.0's scores are measured by the performance of the size 3 ensembles generated by forward search (Caruana et al., 2004) from its search procedure. AlphaD3M, Axolotl, and CMU AutoML's scores are measured by taking the top 3 pipelines discovered during search and ensembling them using majority voting. Lastly, Ensemble^2's score is measured by taking the top 3 pipelines discovered by all six SOTA systems and ensembling them using majority voting. All systems were run for 1 hour, and had 20 minutes to generate test-set predictions. Failed AutoML runs were run one more time with identical configurations.

5.3 Result

Table 2 summarizes the results of this experiment. The average rank of each AutoML system relative to one another is written in the bottom row of the table, with ties allowed.

OpenML Dataset ID	AlphaD3M	Axolotl	CMU AutoML	AutoGluon	Auto-Sklearn	Auto-Sklearn-2	Ensemble^2
2	0.600 ± 0.000	0.981 ± 0.000	0.822 ± 0.000	**0.990 ± 0.001**	0.985 ± 0.003	0.976 ± 0.007	0.982 ± 0.003
3	0.515 ± 0.002	0.990 ± 0.001	**0.996 ± 0.000**	0.993 ± 0.002	0.654 ± 0.463	0.995 ± 0.002	0.994 ± 0.002
5	0.357 ± 0.011	**0.744 ± 0.005**	0.475 ± 0.000	0.683 ± 0.005	0.724 ± 0.016	0.479 ± 0.339	0.705 ± 0.011
12	0.092 ± 0.003	0.937 ± 0.005	0.207 ± 0.000	0.973 ± 0.001	0.651 ± 0.460	0.974 ± 0.001	**0.974 ± 0.001**
31	0.670 ± 0.025	0.726 ± 0.002	0.745 ± 0.000	**0.783 ± 0.001**	0.756 ± 0.017	**0.783 ± 0.002**	0.775 ± 0.002
54	0.293 ± 0.011	0.743 ± 0.013	0.768 ± 0.001	**0.841 ± 0.010**	0.797 ± 0.016	0.754 ± 0.018	0.834 ± 0.012
1067	0.767 ± 0.036	0.813 ± 0.032	0.845 ± 0.000	0.857 ± 0.002	0.857 ± 0.004	0.848 ± 0.006	**0.860 ± 0.001**
1111	0.655 ± 0.463	0.964 ± 0.024	**0.983 ± 0.000**	**0.983 ± 0.000**	**0.983 ± 0.000**	**0.983 ± 0.000**	**0.983 ± 0.000**
1169	0.339 ± 0.240	-	0.650 ± 0.000	0.661 ± 0.005	0.438 ± 0.309	**0.666 ± 0.001**	0.662 ± 0.005
1461	0.843 ± 0.005	0.893 ± 0.007	0.903 ± 0.000	**0.909 ± 0.000**	0.904 ± 0.001	0.906 ± 0.001	0.908 ± 0.000
1464	0.723 ± 0.009	0.747 ± 0.000	0.743 ± 0.000	0.750 ± 0.007	0.753 ± 0.018	0.760 ± 0.024	**0.768 ± 0.022**
1468	-	0.927 ± 0.009	0.937 ± 0.002	0.914 ± 0.006	0.924 ± 0.004	**0.941 ± 0.007**	0.938 ± 0.004
1486	0.395 ± 0.279	0.966 ± 0.003	0.970 ± 0.001	**0.973 ± 0.001**	0.966 ± 0.002	0.648 ± 0.458	**0.973 ± 0.001**
1489	0.584 ± 0.001	0.890 ± 0.000	0.901 ± 0.000	**0.902 ± 0.002**	0.886 ± 0.005	0.890 ± 0.001	**0.902 ± 0.002**
1590	0.434 ± 0.307	0.856 ± 0.002	0.860 ± 0.000	**0.876 ± 0.000**	0.873 ± 0.001	0.864 ± 0.012	**0.876 ± 0.000**
1596	0.255 ± 0.180	-	0.650 ± 0.035	0.880 ± 0.006	0.317 ± 0.448	**0.955 ± 0.000**	0.925 ± 0.021
4135	0.607 ± 0.429	0.946 ± 0.001	0.947 ± 0.000	**0.949 ± 0.000**	0.629 ± 0.445	0.947 ± 0.001	**0.949 ± 0.000**
23512	0.337 ± 0.238	0.447 ± 0.317	0.719 ± 0.000	0.484 ± 0.342	0.483 ± 0.342	0.727 ± 0.002	**0.725 ± 0.003**
23517	0.503 ± 0.000	0.518 ± 0.001	0.519 ± 0.000	0.511 ± 0.005	0.519 ± 0.001	**0.520 ± 0.001**	0.517 ± 0.002
40668	0.439 ± 0.310	0.704 ± 0.071	0.836 ± 0.003	0.837 ± 0.005	0.840 ± 0.002	**0.850 ± 0.002**	0.846 ± 0.002
40685	0.431 ± 0.305	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**	0.789 ± 0.000	0.997 ± 0.000	**1.000 ± 0.000**
40975	0.614 ± 0.057	0.973 ± 0.000	0.969 ± 0.003	**0.979 ± 0.001**	0.643 ± 0.455	0.977 ± 0.006	**0.979 ± 0.001**
40981	0.523 ± 0.007	0.862 ± 0.000	0.841 ± 0.000	0.864 ± 0.007	0.586 ± 0.414	**0.874 ± 0.013**	0.867 ± 0.010
40984	0.149 ± 0.003	0.924 ± 0.001	0.939 ± 0.000	**0.940 ± 0.001**	0.918 ± 0.001	0.929 ± 0.002	0.939 ± 0.001
40996	0.066 ± 0.047	-	0.501 ± 0.142	**0.896 ± 0.001**	-	-	**0.896 ± 0.001**
41027	0.343 ± 0.242	0.867 ± 0.005	0.874 ± 0.000	**0.962 ± 0.002**	0.883 ± 0.005	0.866 ± 0.002	**0.962 ± 0.002**
41138	0.643 ± 0.455	0.662 ± 0.455	0.993 ± 0.000	**0.994 ± 0.000**	0.992 ± 0.000	0.993 ± 0.000	0.993 ± 0.000
41142	-	-	0.708 ± 0.009	**0.735 ± 0.003**	0.726 ± 0.004	0.725 ± 0.002	**0.735 ± 0.003**
41143	0.477 ± 0.001	0.800 ± 0.000	0.800 ± 0.002	0.805 ± 0.003	0.801 ± 0.002	0.267 ± 0.378	**0.806 ± 0.003**
41146	0.351 ± 0.248	0.934 ± 0.000	0.955 ± 0.000	0.948 ± 0.001	0.939 ± 0.005	0.934 ± 0.003	**0.949 ± 0.001**
41147	0.333 ± 0.236	-	0.645 ± 0.007	-	0.682 ± 0.001	**0.684 ± 0.001**	0.683 ± 0.001
41150	0.605 ± 0.000	0.542 ± 0.385	0.940 ± 0.000	0.946 ± 0.001	0.939 ± 0.000	0.945 ± 0.002	**0.947 ± 0.001**
41159	-	-	-	**0.821 ± 0.003**	-	-	0.819 ± 0.001
41161	-	-	-	**0.997 ± 0.001**	-	-	**0.997 ± 0.001**
41163	-	-	0.488 ± 0.122	**0.989 ± 0.000**	0.975 ± 0.002	0.932 ± 0.003	0.985 ± 0.003
41164	0.186 ± 0.001	0.287 ± 0.035	0.188 ± 0.000	**0.721 ± 0.003**	0.679 ± 0.006	0.656 ± 0.012	**0.721 ± 0.003**
41165	-	-	-	**0.493 ± 0.005**	-	-	0.486 ± 0.010
41166	0.109 ± 0.077	0.373 ± 0.150	0.611 ± 0.060	**0.713 ± 0.003**	0.682 ± 0.005	0.654 ± 0.001	0.707 ± 0.006
41167	0.003 ± 0.000	-	0.583 ± 0.027	**0.909 ± 0.001**	0.684 ± 0.001	0.503 ± 0.356	**0.909 ± 0.001**
41168	0.248 ± 0.175	0.576 ± 0.149	0.679 ± 0.027	**0.719 ± 0.002**	0.704 ± 0.002	0.713 ± 0.001	0.718 ± 0.003
41169	0.026 ± 0.000	0.310 ± 0.012	0.350 ± 0.014	0.394 ± 0.000	0.296 ± 0.001	0.332 ± 0.003	**0.395 ± 0.001**
Average Accuracy (across datasets)	0.354	0.583	0.696	0.819	0.679	0.718	**0.845**
Average Rank (out of 7)	5.683	4.415	3.488	2.024	3.439	2.756	**1.659**

Table 2. Comparison between Ensemble^2 and SOTA AutoML systems. All systems were run for one hour each. Ensemble^2 uses an ensemble of size three. The highest accuracy achieved by a method on a dataset is shown in boldface and dash (−) shows a method failed to produce outputs for a dataset.

Overall, Ensemble^2 achieved the highest average rank of 1.659, which is noticeably higher than the second highest average rank of 2.024 achieved by AutoGluon.

To ensure that the difference in performance distribution over the datasets in the benchmark between pairs of AutoML systems was statistically significant, we ran Wilcoxon signed-rank test with $\alpha = 0.05$. Since Ensemble^2 achieved first place, we computed the test statistic between Ensemble^2 with every other AutoML system. The resulting p-values for rejecting the null hypothesis that the pair produced the same results all indicated statistical significance. Specifically, all p-values were below 10^{-8} except for the AutoGluon paired test which had a still-significant p-value of 0.018. These findings show that there is room for improvement by ensembling pipelines discovered by different AutoML systems.

Another strength of Ensemble^2 is the fact that Ensemble^2 produces a pipeline when any of its base learners produces a pipeline. In the OpenML benchmark, this enabled Ensemble^2 to succeed at every single task in the benchmark - something no other AutoML systems could do.

We noticed that AutoGluon achieves first place on more datasets than Ensemble^2 on Table 2 ($24/41$ vs $19/41$ datasets). This was surprising to us since Ensemble^2 had both higher average accuracy and rank compared to AutoGluon. When we computed the number of times an AutoML system achieves first place only between AutoGluon and Ensemble^2, we discovered that Ensemble^2 had first place on 30 of the 41 datasets and Autogluon had first place on 24 out of the 41 datasets. This entails that Ensemble^2 does have more wins over AutoGluon across benchmark datasets, but the datasets on which Ensemble^2 performs better on can have other AutoML systems that perform better than both Ensemble^2 and AutoGluon. In addition, as the AutoGluon paper reported, out-of-memory error was a common failure mode across base AutoML systems aside from AutoGluon.

5.4 AutoML System Performance Correlation

We investigate whether Ensemble^2's base AutoML systems perform similarly across benchmark tasks by looking at the correlation between the base system accuracy. This correlation measures whether the base systems generally perform better on different kinds of datasets. The less correlated the base systems are, the more suited they are for different kinds of datasets. Ensemble^2 obviously benefits from having a suite of lowly correlated base systems. Figure 2 shows that some diversity exists in the performance of the best pipelines generated by Ensemble^2's base systems. For example, the correlation between AutoGluon and Auto-Sklearn is $\sim$0.33 and the correlation between AlphaD3M and Auto-Sklearn is $\sim$0.52. Hence, we argue that Ensemble^2 is overall more well-rounded than its base systems.

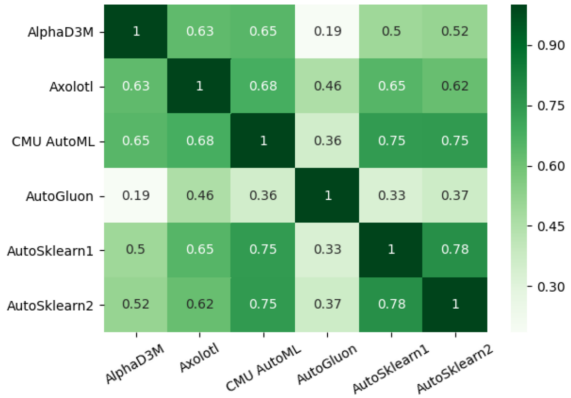

Figure 2. Correlation in test set performance between base AutoML systems and Ensemble^2.

6 Conclusion

In this paper we have established that ensembling AutoML systems can lead to quantitive gains in accuracy. This, and our explicit examination of the correlation between the accuracy of various AutoML systems across a variety of problems, suggests that there is, currently, exploitable diversity between AutoML systems. The computational complexity of combinatorial search in high dimensions, one of the abstract computational problems underlying machine learning pipeline search, suggests that any practical AutoML system will always have to rely on some kind of heuristic or greedy algorithm. For this reason, it is likely that diversity amongst AutoML systems will remain and that the kind of ensembling we explore here will remain valuable into the future.

There are some caveats to the results that we have presented that warrant both disclosure and consideration. We developed our ensembling hypothesis and particular choice of ensembling technique strictly before examining performance on the test data, here the OpenML AutoML benchmark set of test datasets. On this point you have to trust us that we did *not* extensively evaluate a large variety of ensembling approaches and only then choose a good one to report. If we would have done this, we would have leaked information about the test set into our choice of ensembling approach and then we would have had to report results on a different held out test set.

Additionally we deviate from the AutoML literature somewhat in reporting only wall-clock time-equivalent results. It could be argued that a more fair comparison would be to give each baseline AutoML system an equal amount of compute time to the total compute time (not wallclock time) used by the ensemble, particularly as some baseline AutoML systems, themselves, are able to parallelize internally. While this is true, it only impacts comparisons between base-

line accuracies for each of the baseline AutoML systems and the final ensemble. The fact that ensembling AutoML systems produces a quantitative accuracy gain is orthogonal to however long the ensembled baseline AutoML systems are run. That being said, significant computational resource limitations aside, it would be a valuable, if expensive, experiment to test how much of the AutoML system ensembling result we reported, if any, goes away as the base learner AutoML systems are themselves allowed to run longer.

ACKNOWLEDGEMENTS

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), the Canada CIFAR AI Chairs Program, and the Intel Parallel Computing Centers program. This material is based upon work supported by the United States Air Force Research Laboratory (AFRL) under the Defense Advanced Research Projects Agency (DARPA) Data Driven Discovery Models (D3M) program (Contract No. FA8750-19-2-0222) and Learning with Less Labels (LwLL) program (Contract No.FA8750-19-C-0515). Additional support was provided by UBC's Composites Research Network (CRN), Data Science Institute (DSI) and Support for Teams to Advance Interdisciplinary Research (STAIR) Grants. This research was enabled in part by technical support and computational resources provided by WestGrid (https://www.westgrid.ca/) and Compute Canada (www.computecanada.ca).

REFERENCES

Caruana, R., Niculescu-Mizil, A., Crew, G., and Ksikes, A. Ensemble selection from libraries of models. In Brodley, C. E. (ed.), *Machine Learning, Proceedings of the Twenty-first International Conference(ICML)*, volume 69 of *ACM International Conference Proceeding Series*. ACM, 2004.

Chen, B., Wu, H., Mo, W., Chattopadhyay, I., and Lipson, H. Autostacker: a compositional evolutionary learning system. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 402–409. ACM, 2018.

Drori, I., Krishnamurthy, Y., Lourenço, R., Rampin, R., Cho, K., Silva, C. T., and Freire, J. Automatic machine learning by pipeline synthesis using model-based reinforcement learning and a grammar. *Computing Research Repository*, abs/1905.10345, 2019.

Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., and Smola, A. J. Autogluon-tabular: Robust and accurate AutoML for structured data. *Computing Research Repository*, 2020.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, pp. 2962–2970, 2015.

Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M., and Hutter, F. Auto-Sklearn 2.0: The next generation. *Computing Research Repository*, abs/2007.04074, 2020.

Gijsbers, P., LeDell, E., Thomas, J., Poirier, S., Bischl, B., and Vanschoren, J. An open source automl benchmark, 2019.

Hansen, L. K. and Salamon, P. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

Heffetz, Y., Vainshtein, R., Katz, G., and Rokach, L. Deepline: Automl tool for pipelines generation using deep reinforcement learning and hierarchical actions filtering. In *Conference on Knowledge Discovery and Data Mining*, pp. 2103–2113. ACM, 2020.

Jamieson, K. G. and Talwalkar, A. Non-stochastic best arm identification and hyperparameter optimization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 51, pp. 240–248, 2016.

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., and Byers, A. H. Big data: The next frontier for innovation, competition and productivity. Technical report, McKinsey Global Institute, 2011.

McKinsey Analytics. The age of analytics: competing in a data-driven world. Technical report, San Francisco: McKinsey & Company, 2016.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., VanderPlas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research (JMLR)*, 12:2825–2830, 2011.

Pompa, C. and Burke, T. Data science and analytics skills shortage: equipping the apec workforce with the competencies demanded by employers. *APEC Human Resource Development Working Group*, 2017.

Rahman, A. and Tasnim, S. Ensemble classifiers and their applications: A review. *Computing Research Repository*, abs/1404.4088, 2014.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

Zaidi, S., Zela, A., Elsken, T., Holmes, C., Hutter, F., and Teh, Y. W. Neural ensemble search for performant and calibrated predictions. *Computing Research Repository*, abs/2006.08573, 2020.

Zöller, M.A. and Huber, M. F. Benchmark and survey of automated machine learning frameworks. *Computing Research Repository*, 2019.