AUTOMATA : Gradient Based Data Subset Selection for Compute-Efficient Hyper-parameter Tuning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Deep neural networks have seen great success in recent years; however, training a deep model is often challenging as its performance heavily depends on the hyper-parameters used. In addition, finding the optimal hyper-parameter configuration, even with state-of-the-art (SOTA) hyper-parameter optimization (HPO) algorithms, can be time-consuming, requiring multiple training runs over the entire dataset for different possible sets of hyper-parameters. Our central insight is that using an informative subset of the dataset for model training runs involved in hyper-parameter optimization, allows us to find the optimal hyper-parameter configuration significantly faster. In this work, we propose AUTOMATA, a gradient-based subset selection framework for hyper-parameter tuning. We empirically evaluate the effectiveness of AUTOMATA in hyper-parameter tuning through several experiments on real-world datasets in the text, vision, and tabular domains. Our experiments show that using gradient-based data subsets for hyper-parameter tuning achieves significantly faster turnaround times and speedups of **3**$\times$**-30**$\times$ while achieving comparable performance to the hyper-parameters found using the entire dataset.

1 Introduction

In recent years, deep learning systems have found great success in a wide range of tasks, such as object recognition [15], speech recognition [17], and machine translation [1], making people's lives easier on a daily basis. However, in the quest for near-human performance, more complex and deeper machine learning models trained on increasingly large datasets are being used at the expense of substantial computational costs. Furthermore, deep learning is associated with a significantly large number of hyper-parameters such as the learning algorithm, batch size, learning rate, and model configuration parameters (*e.g.*, depth, number of hidden layers, *etc.*) that need to be tuned. Hence, running extensive hyper-parameter tuning and auto-ml pipelines is becoming increasingly necessary to achieve state-of-the-art models. However, tuning the hyper-parameters requires multiple training runs over the entire datasets (which are significantly large nowadays), resulting in staggering compute costs, running times, and, more importantly, CO2 emissions.

To give an idea of staggering compute costs, we consider an image classification task on a relatively simple CIFAR-10 dataset where a single training run using a relatively simple model class of Residual Networks [16] for 300 epochs on a V100 GPU takes around 4 hours. If we perform 1000 training runs (which is not uncommon today) naively using grid search for hyper-parameter tuning, it will take 4000 GPU hours. The resulting CO2 emissions would be between 440 to 1000 kg of CO2 emitted[1], which is equivalent to 1100 to 2500 miles of car travel in the US. Similarly, the costs of

[1] `https://mlco2.github.io/impact/#compute`

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

training state-of-the-art NLP models and vision models on larger datasets like ImageNet are even more staggering [50][2].

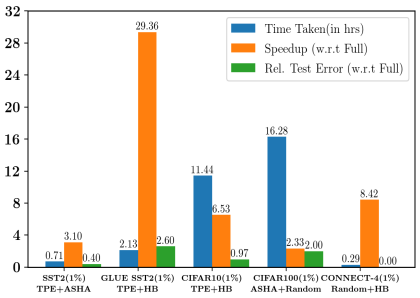

Figure 1: AUTOMATA's performance summary showing speedups, relative test errors, and tuning times on SST2, glue-SST2, CIFAR10, CIFAR100, and CONNECT-4 datasets. We observe that AUTOMATA achieves speedups(and similar energy savings) of 10x - 30x with around 2% performance loss using Hyperband as a scheduler. Similarly, even when using a more efficient ASHA scheduler, AUTOMATA achieves a speedup of around 2x-3x with a performance loss of 0%-2%.

Naive hyper-parameter tuning methods like grid search [2] often fail to scale up with the dimensionality of the search space and are computationally expensive. Hence, more efficient and sophisticated Bayesian optimization methods [4, 19, 5, 48, 27] have dominated the field of hyper-parameter optimization in recent years. Bayesian optimization methods aim to identify good hyper-parameter configurations quickly by building a posterior distribution over the search space and by adaptively selecting configurations based on the probability distribution. More recent methods [51, 52, 10, 27] try to speed up configuration evaluations for efficient hyper-parameter search; these approaches speed up the configuration evaluation by adaptively allocating more resources to promising hyper-parameter configurations while eliminating poor ones quickly.

Recent works like SHA [20], Hyperband [34], ASHA [36] use aggressive early-stopping strategies to stop not-so-promising configurations quickly while allocating more resources to the promising ones. Generally, these resources can be the size of the training set, number of gradient descent iterations, training time, etc. Nickson et al. [42], Krueger et al. [29] try to quickly evaluate a configuration's performance on a large dataset by evaluating the training runs on small, random subsets and they empirically show that small data subsets could suffice to estimate a configuration's quality. Similarly, past works [42, 29] show that very small data subsets can be effectively used to find the best hyper-parameters quickly. However, all these approaches have naively used random training data subsets and did not place much focus on selecting informative subsets instead. Our central insight is that using small informative data subsets allows us to find good hyper-parameter configurations more effectively than random data subsets.

An earlier work called GRADMATCH [23] demonstrated the effectiveness of gradient-based data subset selection for selecting informative data subsets by using it for efficient model training in a supervised learning setting. In this work, we empirically investigate the advantage of using gradient-based subset selection algorithms for hyper-parameter tuning compared with using random subsets and the full dataset for hyper-parameter tuning. Fundamentally, we use gradient-based subset selection algorithms to select data subsets that can be used for efficiently tuning the hyper-parameters. To this end, we propose AUTOMATA, an efficient hyper-parameter tuning framework that combines existing hyper-parameter search and scheduling algorithms with intelligent subset selection. Further, our experimental results illustrate that very small subsets can be used for effective hyper-parameter tuning because we are primarily concerned with retaining the ordering of hyper-parameters rather than the final accuracy, thereby enabling us to achieve better speedups and energy efficiency (and more significant CO2 emissions reduction).

1.1 Related Work

Hyper-parameter tuning approaches: A number of algorithms have been proposed for hyper-parameter tuning including grid search[3], Bayesian algorithms [3], random search [45], etc. Furthermore, a number of scalable toolkits and platforms for hyper-parameter tuning exist like Raytune [38][4], H2O automl [33], etc. See [47, 56] for a survey of current approaches and also tricks for hyper-parameter tuning for deep models. The biggest challenges of existing hyper-parameter tuning approaches are a) the large hyper-parameter search space and b) the increased training times of training models. Recently, Li *et. al.* [35] have proposed an efficient approach for parallelizing hyper-parameter tuning using Asynchronous Successive Halving Algorithm (ASHA). AUTOMATA is

[2] `https://tinyurl.com/a66fexc7`
[3] `https://tinyurl.com/3hb2hans`
[4] `https://docs.ray.io/en/master/tune/index.html`

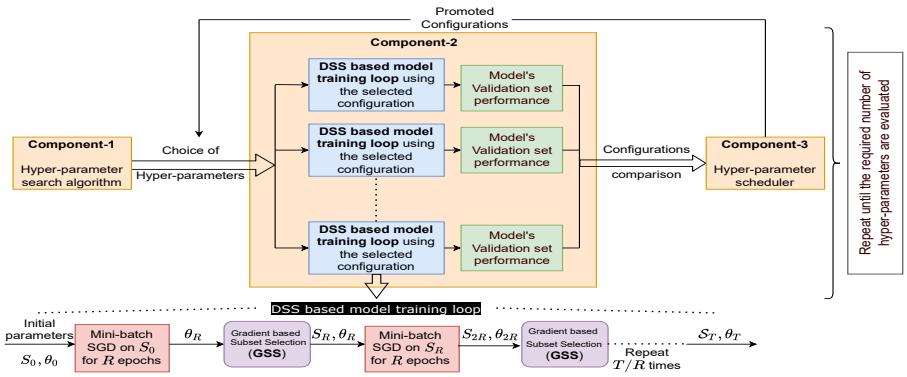

Figure 2: Diagram of AUTOMATA, including hyper-parameter search, subset based configuration evaluation (where models are trained on subsets of data), and hyper-parameter scheduler.

complementary to such approaches and we show that our work can be be combined effectively with them.

Data Subset Selection: Several recent papers have used submodular functions[5] for data subset selection towards various applications like speech recognition [55, 54], machine translation [26] and computer vision [21]. Other common approaches for subset selection include the usage of coresets. Coresets are weighted subsets of the data, which approximate certain desirable characteristics of the full data (*e.g.*, the loss function) [12]. Coreset algorithms have been used for several problems including k-means and k-median clustering [14], SVMs [8] and Bayesian inference [6]. Recent coreset selection-based methods [40, 24, 23, 25] have shown great promise for efficient and robust training of deep models. CRAIG [40] tries to select a coreset summary of the training data that estimate the full training gradient closely. Whereas GLISTER [24] poses the coreset selection problem as a discrete-continuous bilevel optimization problem that minimizes the validation set loss. Similarly, RETRIEVE [25] also uses a discrete bilevel coreset selection problem to select unlabeled data subsets for efficient semi-supervised learning. Another approach GRAD-MATCH [23] selects coreset summary that approximately matches the full training loss gradient using orthogonal matching pursuit.

The contributions of our work can be summarized as follows: **AUTOMATA Framework:** We propose AUTOMATA a framework that combines intelligent subset selection with hyper-parameter search and scheduling algorithms to enable faster hyper-parameter tuning. Our work studies the role of intelligent data subset selection for hyper-parameter tuning. In particular, we seek to answer the following question: *Is it possible to use small informative data subsets between 1% to 30% for faster configuration evaluations in hyper-parameter tuning, thereby enabling faster tuning times while maintaining comparable accuracy to tuning hyper-parameters on the full dataset?* **Effectiveness of AUTOMATA :** We empirically demonstrate the effectiveness of AUTOMATA framework used in conjunction with existing hyper-parameter search algorithms like TPE, Random Search, and hyper-parameter scheduling algorithms like Hyperband, and ASHA through a set of extensive experiments on multiple real-world datasets. Further, we empirically show that the gradient-based subset selection of AUTOMATA retains the hyper-parameter ordering better compared to other baseline subset selection strategies including RANDOM. We give a summary of the speedup vs. relative performance achieved by AUTOMATA compared to full data training in Figure 1. More specifically, AUTOMATA achieves a speedup of 3x - 30x with minimal performance loss for hyper-parameter tuning. Further, in Section 3, we show that the gradient-based subset selection approach of AUTOMATA outperforms the previously considered random subset selection for hyper-parameter tuning.

[5]Let $V = \{1, 2, \cdots, n\}$ denote a ground set of items. A set function $f : 2^V \rightarrow \mathbf{R}$ is a submodular [13] if it satisfies the diminishing returns property: for subsets $S \subseteq T \subseteq V$ and $j \in V \backslash T$, $f(j|S) \triangleq f(S \cup j) - f(S) \geq f(j|T)$.

2 AUTOMATA Framework

In this section, we present AUTOMATA, a hyper-parameter tuning framework, and discuss its different components shown in Figure 2. The AUTOMATA framework consists of three components: a hyper-parameter search algorithm that identifies which configuration sets need to be evaluated, a gradient-based subset selection algorithm for training and evaluating each configuration efficiently, and a hyper-parameter scheduling algorithm that provides early stopping by eliminating the poor configurations quickly. With AUTOMATA framework, one can use any of the existing hyper-parameter search and hyper-parameter scheduling algorithms and still achieve significant speedups with minimal performance degradation due to faster configuration evaluation using gradient-based subset training.

2.1 Notation

Denote by $\mathbb{H}$, the set of configurations selected by the hyper-parameter search algorithm. Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, denote the set of training examples, and $\mathcal{V} = \{(x_j, y_j)\}_{j=1}^{M}$ the validation set. Let θ_i denote the classifier model parameters trained using the configuration $i \in \mathbb{H}$. Let $\mathcal{S}_i$ be the subset used for training the ith configuration model θ_i and $\boldsymbol{w}_i$ be its associated weight vector i.e., each data sample in the subset has an associated weight that is used for computing the weighted loss. We superscript the changing variables like model parameters θ, subset $\mathcal{S}$ with the timestep t to denote their specific values at that timestep. Next, denote by $L_T^j(\theta_i) = L_T(x_j, y_j, \theta_i)$, the training loss of the jth data sample in the dataset for i^{th} classifier model, and let $L_T(\theta_i) = \sum_{k \in \mathcal{D}} L_T^k(\theta_i)$ be the loss over the entire training set for i^{th} configuration model. Let, $L_T^j(\mathcal{S}, \theta_i) = \sum_{k \in \mathcal{X}} L_T(x_k, y_k, \theta_i)$ be the loss on a subset $\mathcal{S} \subseteq \mathcal{V}$ of the training examples at timestep j. Let the validation loss be denoted by L_V.

2.2 Component-1: Hyper-parameter Search Algorithm

Given a hyper-parameter search space, hyper-parameter search algorithms provide a set of configurations that need to be evaluated. A naive way of performing the hyper-parameter search is Grid-Search, which defines the search space as a grid and exhaustively evaluates each grid configuration. However, Grid-Search is a time-consuming process, meaning that thousands to millions of configurations would need to be evaluated if the hyper-parameter space is large. In order to find optimal hyper-parameter settings quickly, Bayesian optimization-based hyper-parameter search algorithms have been developed. To investigate the effectiveness of AUTOMATA across the spectrum of search algorithms, we used the Random Search method and the Bayesian optimization-based TPE method as representative hyper-parameter search algorithms. We provide more details on Random Search and TPE in Appendix D.

2.3 Component-2: Subset based Configuration Evaluation

Earlier, we discussed how a hyper-parameter search algorithm presents a set of potential hyper-parameter configurations that need to be evaluated when tuning hyper-parameters. Every time a configuration needs to be evaluated, prior work trained the model on the entire dataset until the resource allocated by the hyper-parameter scheduler is exhausted. Rather than using the entire dataset for training, we propose using subsets of informative data selected based on gradients instead. As a result, given any hyper-parameter search algorithm, we can use the data subset selection to speed up each training epoch by a significant factor *(say 10x)*, thus improving the overall turnaround time of the hyper-parameter tuning. However, the critical advantage of AUTOMATA is that we can achieve speedups while still retaining the hyper-parameter tuning algorithm's performance in finding the best hyper-parameters. The fundamental feature of AUTOMATA is that the subset selected by AUTOMATA changes adaptively over time, based on the classifier model training. Thus, instead of selecting a common subset among all configurations, AUTOMATA selects the subset that best suits each configuration. We give a detailed overview of the gradient-based subset selection process of AUTOMATA below.

2.3.1 Gradient Based Subset Selection (GSS)

The key idea of gradient-based subset selection of AUTOMATA is to select a subset $\mathcal{S}$ and its associated weight vector $\boldsymbol{w}$ such that the weighted subset loss gradient best approximates the entire training loss gradient. The subset selection of AUTOMATA for i^{th} configuration at time step t is as follows:

$$\boldsymbol{w}_i^t, \mathcal{S}_i^t = \underset{\boldsymbol{w}_i^t, \mathcal{S}_i^t : |\mathcal{S}_i^t| \leq k, \boldsymbol{w}_i^t \geq 0}{\operatorname{argmin}} \| \sum_{l \in \mathcal{S}_i^t} \boldsymbol{w}_{il}^t \nabla_\theta L_T^l(\theta_i^t) - \nabla_\theta L_T(\theta_i^t) \| + \lambda \left\| \boldsymbol{w}_i^t \right\|^2 \tag{1}$$

The additional regularization term prevents assignment of very large weight values to data samples, thereby reducing the possibility of overfitting on a few data samples. A similar formulation for subset selection in the context of efficient supervised learning was employed in a recent work called GRAD-MATCH [23]. The authors of the work [23] proved that the optimization problem given in Equation (1) is approximately submodular. Therefore, the above optimization problem can be solved using greedy algorithms with approximation guarantees [9, 39]. Similar to GRAD-MATCH, we use a greedy algorithm called orthogonal matching pursuit (OMP) to solve the above optimization problem. The goal of AUTOMATA is to accelerate the hyper-parameter tuning algorithm while preserving its original performance. Efficiency is, therefore, an essential factor that AUTOMATA considers even when selecting subsets. Due to this, we employ a faster per-batch subset selection introduced in the work [23] in our experiments, which is described in the following section.

Per-Batch Subset Selection: Instead of selecting a subset of data points, one selects a subset of mini-batches by matching the weighted sum of mini-batch training gradients to the full training loss gradients. Therefore, one will have a subset of slected mini-batches and the associated mini-batch weights. One trains the model on the selected mini-batches by performing mini-batch gradient descent using the weighted mini-batch loss. Let us denote the batch size as B, and the total number of mini-batches as $b_N = \frac{N}{B}$, and the training set of mini-batches as $\mathcal{D}_\mathcal{B}$. Let us denote the number of mini-batches that needs to be selected as $b_k = \frac{k}{B}$. Let us denote the subset of mini-batches that needs to be selected as $\mathcal{S}_{\mathcal{B}i}$ and denote the weights associated with mini-batches as $\boldsymbol{w}_{\mathcal{B}i} = \{\boldsymbol{w}_{\mathcal{B}i1}, \boldsymbol{w}_{\mathcal{B}i2} \cdots \boldsymbol{w}_{\mathcal{B}ik}\}$ for the ith model configuration. Let us denote the mini-batch gradients as $\nabla_\theta L_T^{\mathcal{B}_i}(\theta_i), \cdots, \nabla_\theta L_T^{\mathcal{B}_{b_N}}(\theta_i)$ be the mini-batch gradients for the ith model configuration. Let us denote $L_T^\mathcal{B}(\theta_i) = \sum_{i \in [1, b_N]} L_T^{\mathcal{B}_k}(\theta_i)$ be the loss over the entire training set. The subset selection problem of mini-batches at time step t can be written as follows:

$$\boldsymbol{w}_{\mathcal{B}i}^t, \mathcal{S}_{\mathcal{B}i}^t = \underset{\boldsymbol{w}_{\mathcal{B}i}^t, \mathcal{S}_{\mathcal{B}i}^t : |\mathcal{S}_{\mathcal{B}i}^t| \leq b_k, \boldsymbol{w}_{\mathcal{B}i}^t \geq 0}{\operatorname{argmin}} \| \sum_{l \in \mathcal{S}_{\mathcal{B}i}^t} \boldsymbol{w}_{\mathcal{B}il}^t \nabla_\theta L_T^{\mathcal{B}_l}(\theta_i^t) - \nabla_\theta L_T^\mathcal{B}(\theta_i^t) \| + \lambda \left\| \boldsymbol{w}_{\mathcal{B}i}^t \right\|^2 \tag{2}$$

In the per-batch version, because the number of samples required for selection is b_k is less than k, the number of greedy iterations required for data subset selection in OMP is reduced, resulting in a speedup of $B\times$. A critical trade-off in using larger batch sizes is that in order to get better speedups, we must also sacrifice data subset selection performance. Therefore, it is recommended to use smaller batch sizes for subset selection to get a optimal trade-off between speedups and performance. A solution to the batch size dependency is to use a smaller batch size only for subset selection and combine the smaller batches to produce larger batches for training. In order to demonstrate that the effectiveness of AUTOMATA is not dependent on the size of the training batch, in our experiments on Images, we use a fixed batch size (instead of as a hyper-parameter) of $B = 20$, and on text datasets, we use the batch size as a hyper-parameter with $B \in [16, 32, 64]$. Apart from per-batch selection, we use model warm-starting to get more informative data subsets. Further, in our experiments, we use a regularization coefficient of $\lambda = 0$ after analyzing the performance of AUTOMATA with different values of λ. We present the ablation study on λ in Appendix F.2.1. We give more details on warm-starting below.

Warm-starting data selection: We warm-start each configuration model by training on the entire training dataset for a few epochs similar to [23]. The warm-starting process enables the model to have informative loss gradients used for subset selection. To be more specific, the classifier model is trained on the entire training data for $T_w = \frac{\kappa T k}{N}$ epochs, where k is the coreset size, T is the total number of epochs, κ is the fraction of warm start, and N is the size of the training dataset. We use a κ value of 0 (*i.e.*, no warm start) for experiments using Hyperband as scheduling algorithm, and a κ value of 0.35 for experiments using ASHA. We present the ablation studies for κ in Appendix F.2.2.

2.4 Component-3: Hyper-parameter Scheduling Algorithm

Hyper-parameter scheduling algorithms improve the overall efficiency of the hyper-parameter tuning by terminating some of the poor configurations runs early. In our experiments, we consider Hyperband [34], and ASHA [36], which are extensions of the Sequential Halving algorithm (SHA) [20] that uses aggressive early stopping to terminate poor configuration runs and allocates an increasingly exponential amount of resources to the better performing configurations. SHA starts with n number of initial configurations, each assigned with a minimum resource amount r. The SHA algorithm uses a reduction factor η to reduce the number of configurations in each round by selecting the top $\frac{1}{\eta}^{th}$ fraction of configurations while also increasing the resources allocated to these configurations by η times each round. We discuss Hyperband and ASHA and the issues within SHA that each of them addresses in more detail in Appendix E.

Detailed pseudocode of the AUTOMATA algorithm is provided in Appendix C due to space constraints in the main paper. We use the popular deep learning framework [43] for implementation of AUTOMATA framework, Ray-tune[38] for hyper-parameter search and scheduling algorithms, and CORDS [22] for subset selection strategies.

3 Experiments

In this section, we present the effectiveness and the efficiency of AUTOMATA framework for hyper-parameter tuning by evaluating AUTOMATA on datasets spanning text, image, and tabular domains. Further, to assess AUTOMATA's effectiveness across the spectrum of existing hyper-parameter search and scheduling algorithms, we conduct experiments using combinations of different search and scheduling algorithms. As discussed earlier, we employ Random Search [45], TPE [4] as representative hyper-parameter search algorithms, and Hyperband [34], ASHA [36] as representative hyper-parameter scheduling algorithms. It should be noted that the combination of TPE and HyperBand is not the same as the BOHB algorithm [11]. In BOHB [11], a multivariate kernel density estimator (KDE) was used, whereas the TPE [4] used a hierarchy of one-dimensional KDEs. However, we believe the takeaways would remain the same even with other approaches. We repeat each experiment five times on the text and tabular datasets, thrice on the image datasets, and report the mean accuracy and speedups in the plots. Below, we provide further details on datasets, baselines, models, and the hyper-parameter search space used for experiments.

3.1 Baselines

Our experiments aim to demonstrate the consistency and efficiency of AUTOMATA more specifically, the effectiveness of AUTOMATA's gradient-based subset selection (GSS) for hyper-parameter tuning. As baselines, we replace the GSS subset selection strategy in AUTOMATA with different subset selection strategies, namely RANDOM (randomly sample a same sized subset as AUTOMATA from the training data), CRAIG [40] (a gradient-based subset selection proposed for efficient supervised learning), and FULL (using the entire training data for model training during configuration evaluation). For ease of notation, we refer to baselines by the names of corresponding subset selection strategies. Note that by CRAIG baseline, we mean the faster per-batch version of CRAIG [40] for subset selection shown [23] to be more efficient than the original. In addition, for all methods, we do not use any warm-start for experiments with Hyperband and use a warm start of $\kappa = 0.35$ for experiments with ASHA. We give more details on the reason for using warm-start with ASHA and no warm-start with Hyperband in Appendix F.2.2. We perform experiments with different subset size fractions of 1%, 5%, 10%, and 30%. In our experiments, we compare our approach's accuracy and efficiency (time/energy) with Full training, Per Batch CRAIG selection, and Random selection.

3.2 Datasets, Model Architecture, and Experimental Setup

To demonstrate the effectiveness of AUTOMATA for hyper-parameter tuning, we performed experiments on datasets spanning text, image and tabular domains. Text datasets include SST2 [49], SST5 [49], glue-SST2 [53], and TREC6 [37, 18]. Image datasets include CIFAR10 [28], CIFAR100 [28], and Street View House Numbers (SVHN) [41]. Tabular datasets include DNA, SATIMAGE, LETTER, and CONNECT-4 from **LIBSVM** (a library for Support Vector Machines (SVMs)) [7]. We give more details on dataset sizes and splits in Appendix F.1. For the Text datasets,

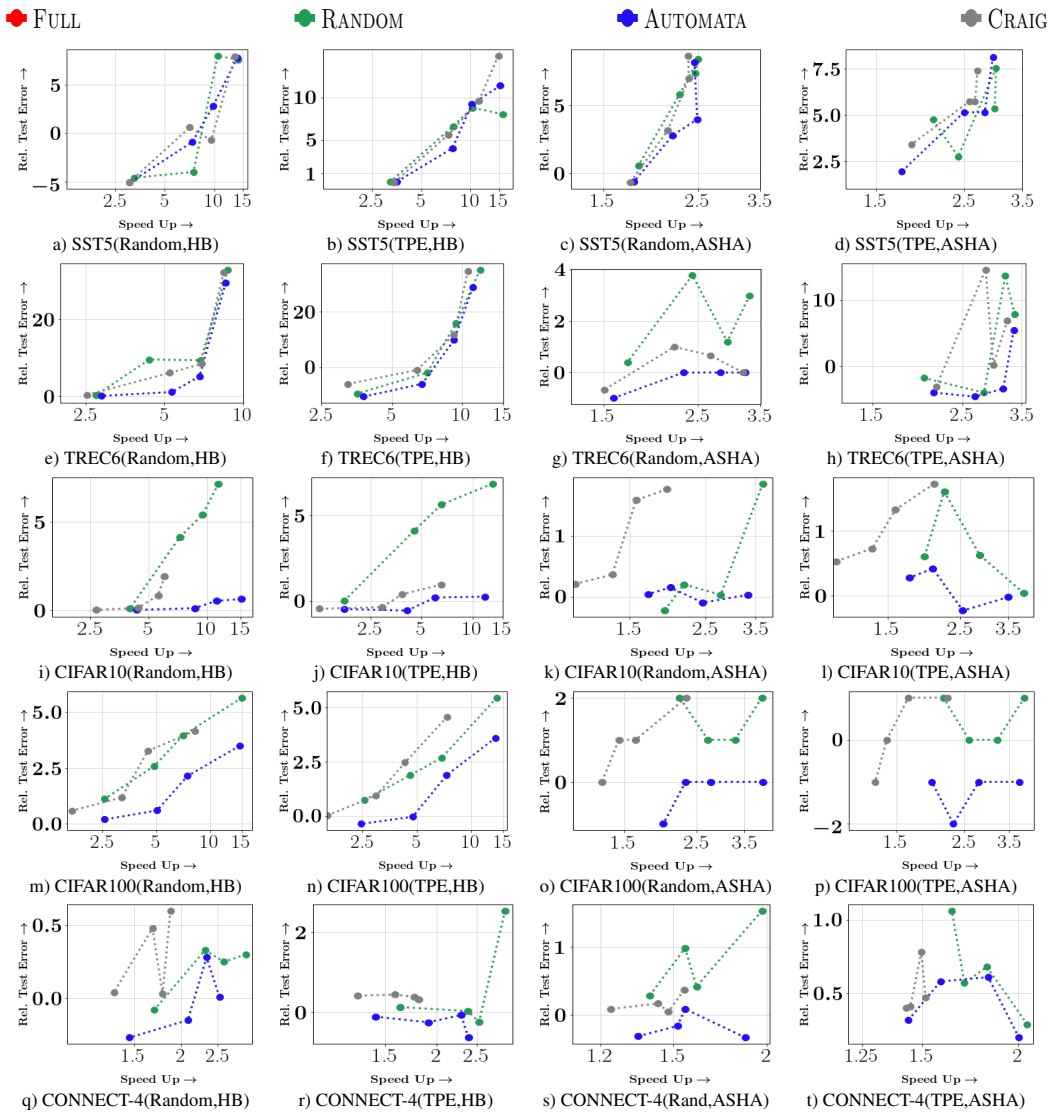

Figure 3: Comparison of performance of AUTOMATA with baselines(RANDOM, CRAIG, FULL) for Hyper-parameter tuning. In sub-figures (a-t), we present speedup *vs.* relative test error (in %), compared to Full data tuning for different methods. On each scatter plot, smaller subsets appear on the right, and larger ones appear on the left. Results are shown for (a-d) SST5, (e-h) TREC6, (i-l) CIFAR10, (m-p) CIFAR100, and (q-t) CONNECT-4 datasets with different combinations of hyper-parameter search and scheduling algorithms. *The scatter plots show that* AUTOMATA *achieves the best speedup-accuracy tradeoff in almost every case* ***(bottom-right corner of each plot indicates the best speedup-accuracy tradeoff region)***.

we use the LSTM model (from PyTorch) with trainable GloVe [44] embeddings of 300 dimension as input. For Image datasets, we use the ResNet18 [16] and ResNet50 [16] models. For Tabular datasets, we use a multi-layer perceptron with 2 hidden layers. Once the best hyper-parameter configuration is found, we perform one more *final training* of the model using the best configuration on the entire dataset and report the achieved test accuracy. We use *final training* for all methods except FULL since the models trained on small data subsets (especially with small subset fractions of 1%, 5%) during tuning do not achieve high test accuracies. We also include the final training times while calculating the tuning times for a more fair comparison[6] For text datasets, we train the LSTM model for 20 epochs while choosing subsets (except for FULL) every 5 epochs. The hyper-parameter space includes learning rate, hidden size & number of layers of LSTM, batch size of training. Some experiments (with TPE as the search algorithm) use 27 configurations in the hyper-parameter space, while others use 54. More details on hyper-parameter search space for text datasets are given in Appendix F.3.1. For image datasets, we train the ResNet [16] model for 300 epochs while choosing subsets (except for FULL) every 20 epochs. We use a Stochastic Gradient Descent (SGD) optimizer with momentum set to 0.9 and weight decay factor set to 0.0005. The hyper-parameter search space consists of a choice between the Momentum method and Nesterov Accelerated Gradient method, choice of learning rate scheduler and their corresponding parameters, and four different group-wise learning rates. We use 27 configurations in the hyper-parameter space for Image datasets. More details on hyper-parameter search space for image datasets are given in Appendix F.3.2. For tabular datasets, we train a multi-layer perceptron with 2 hidden layers for 200 epochs while choosing subsets every 10 epochs. The hyper-parameter search space consists of a choice between the SGD optimizer or Adam optimizer, choice of learning rate, choice of learning rate scheduler, the sizes of the two hidden layers and batch size for training. We use 27 configurations in the hyper-parameter space for Tabular datasets. More details on hyper-parameter search space for tabular datasets are provided in Appendix F.3.3.

3.3 Retention of Hyper-parameter Ordering

Subset Selection Method	**Spearman Rank Correlation**
RANDOM	0.359
CRAIG	0.575
GSS (Ours)	0.653

Table 1: Spearman's rank correlation values of hyper-parameter ordering.

We evaluate the effectiveness of different subset selection strategies in retaining hyper-parameter ordering by comparing their ordering obtained using AUTOMATA's gradient-based subset selection (GSS), RANDOM, CRAIG strategies with that obtained using full data. To examine this, we experiment on the MNIST [32] dataset using a LeNet [31] model with hyper-parameter search space, including learning rate(over convolution layers), learning rate2(over fully connected layers), and a scheduler. We use 27 search configurations, with Random search as the search algorithm and no scheduler. Table 1 shows the Spearman rank correlation values between the hyper-parameter ordering obtained using 10% subsets selected by RANDOM, CRAIG, and AUTOMATA's gradient-based subset selection (GSS) with Full data hyper-parameter ordering using only Random search with no scheduler on the MNIST [32] dataset. In terms of maintaining the hyper-parameter ordering, GSS is more effective than RANDOM and CRAIG, while RANDOM is the least effective strategy. Spearman ranking values showcase the importance of using informative data subset selection over random subset selection for hyper-parameter tuning.

3.4 Hyper-parameter Tuning Results

Results comparing the accuracy vs. efficiency tradeoff of different subset selection strategies for hyper-parameter tuning are shown in Figure 3. Performance is compared for different sizes of subsets of training data: 1%, 5%, 10%, and 30% along with four possible combinations of search algorithm (Random or TPE) and scheduling algorithm (ASHA or Hyperband).

Text datasets results: Sub-figures(3a, 3b, 3c, 3d) show the plots of relative test error *vs.* speed ups, both *w.r.t* full data tuning for SST5 dataset with different combinations of search and scheduling methods. Similarly, in sub-figures(3e, 3f, 3g, 3h) we present the plots of relative test error *vs.* speed

[6]Note that with a 30% subset, final training is not required as the models trained with 30% subsets achieve similar accuracy to full data training. However, for the sake of consistency, we use *final training* with 30% subsets as well.

ups for TREC6 dataset. From the results, we observe that AUTOMATA achieves best speed up *vs.* accuracy tradeoff and consistently gives better performance even with small subset sizes unlike other baselines like RANDOM, CRAIG. In particular, AUTOMATA achieves a speedup of 9.8× and 7.35× with a performance loss of 2.8% and a performance gain of 0.9% respectively on the SST5 dataset with TPE and Hyperband. Additionally, AUTOMATA achieves a speedup of around 3.15×, 2.68× with a performance gain of 3.4%, 4.6% respectively for the TREC6 dataset with TPE and ASHA.

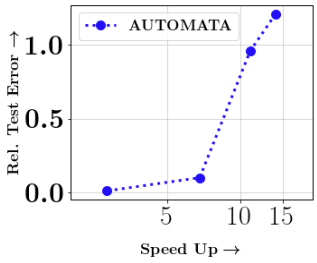

Figure 4: Performance of AUTOMATA on CIFAR10 dataset with 110 search configurations

Image datasets results: Sub-figures(3i, 3j, 3k, 3l) show the plots of relative test error *vs.* speed ups, both *w.r.t* full data tuning for CIFAR10 dataset with different combinations of search and scheduling methods. Similarly, sub-figures (3m, 3n, 3o, 3p) show the plots of relative test error *vs.* speed ups on CIFAR100. The results show that AUTOMATA achieves the best speed up *vs.* accuracy tradeoff consistently compared to other baselines. More specifically, AUTOMATA achieves a speedup of around 15×, 8.7× with a performance loss of 0.65%, 0.14% respectively on the CIFAR10 dataset with Random and Hyperband. Further, AUTOMATA achieves a speedup of around 3.7×, 2.3× with a performance gain of 1%, 2% for CIFAR100 dataset with TPE and ASHA. In Figure 4, we show the effectiveness of the AUTOMATAwith large configuration space by repeating the experiment on the CIFAR10 dataset with 110 configurations using Random as the search algorithm and Hyperband as the scheduler.

Tabular datasets results: Sub-figures(3q, 3r, 3s, 3t) show the plots of relative test error *vs.* speed ups for the CONNECT-4 dataset. In practice, larger subset sizes should lead to better performance. However, the plots(3q, 3r, 3s, 3t) do not depict the same picture. This is due to the inherent class imbalance within the CONNECT-4 dataset. In this case, smaller subset sizes with gradient based subset selection mitigated the effects of class imbalance and enabled better model performance by selecting a similar proportion of samples from each class. Furthermore, using larger subset sizes caused gradient-based subset selection to select more samples from overrepresentative classes, thus reducing the final model's accuracy. Finally, AUTOMATA consistently achieved better speedup *vs.* accuracy tradeoff compared to other baselines on CONNECT-4 as well.

Despite the fact that CRAIG and GSS have similar optimization problems, the performance of CRAIG is poor compared to AUTOMATA, since CRAIG suboptimally optimizes an upper bound of the gradient error term. In addition, CRAIG is slower from a computational complexity standpoint since it requires the construction of a similarity kernel. Owing to space constraints, we provide additional results showing the accuracy *vs.* efficiency tradeoff on additional text, image, and tabular datasets in the Appendix F.4. It is important to note that AUTOMATA obtains better speedups when used for hyper-parameter tuning on larger datasets and larger models (in terms of parameters). Apart from the speedups achieved by AUTOMATA, we show in Appendix F.5 that it also achieves similar reductions of energy consumption and CO2 emissions, thereby making it more environmentally friendly.

4 Conclusion, Limitations, and Broader Impact

We introduce AUTOMATA, an efficient hyper-parameter tuning framework that uses intelligent subset selection for model training for faster configuration evaluations. Further, we perform extensive experiments showing the effectiveness of AUTOMATA for Hyper-parameter tuning. In particular, it achieves speedups of around 10× - 15× using Hyperband as scheduler and speedups of around 3× even with a more efficient ASHA scheduler. AUTOMATA significantly decreases CO2 emissions and energy-efficient, reducing the environmental impact of such hyper-parameter tuning on society at large. We hope that the AUTOMATA framework will encourage further research on efficient subset selection approaches for faster hyper-parameter tuning, helping us move closer to the goal of Green AI [46]. Research like this will help substantially reduce the cost of training large AI models and performing AutoML, thereby help democratizing machine learning and help usage by smaller companies, individuals and academic groups. One of the limitations of AUTOMATA is that in scenarios in which no performance loss is desired, we do not know the minimum subset size that gives the best speed up and, therefore, need to use larger subset sizes such as 10%, 30%. In the future, we consider adaptively changing subset sizes based on model performance for each configuration to remove the dependency on subset size.

References

[1] L. Barrault, O. Bojar, M. R. Costa-jussà, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, P. Koehn, S. Malmasi, C. Monz, M. Müller, S. Pal, M. Post, and M. Zampieri. Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy, Aug. 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5301. URL `https://aclanthology.org/W19-5301`.

[2] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10):281–305, 2012. URL `http://jmlr.org/papers/v13/bergstra12a.html`.

[3] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *25th annual conference on neural information processing systems (NIPS 2011)*, volume 24. Neural Information Processing Systems Foundation, 2011.

[4] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL `https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf`.

[5] J. Bergstra, D. Yamins, and D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 115–123, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL `https://proceedings.mlr.press/v28/bergstra13.html`.

[6] T. Campbell and T. Broderick. Bayesian coreset construction via greedy iterative geodesic ascent. In *International Conference on Machine Learning*, pages 698–706, 2018.

[7] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[8] K. L. Clarkson. Coresets, sparse greedy approximation, and the frank-wolfe algorithm. *ACM Transactions on Algorithms (TALG)*, 6(4):1–30, 2010.

[9] A. Das and D. Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. *arXiv preprint arXiv:1102.3975*, 2011.

[10] T. Domhan, J. T. Springenberg, and F. Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 3460–3468. AAAI Press, 2015. ISBN 9781577357384.

[11] S. Falkner, A. Klein, and F. Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1437–1446. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/falkner18a.html`.

[12] D. Feldman. Core-sets: Updated survey. In *Sampling Techniques for Supervised or Unsupervised Tasks*, pages 23–44. Springer, 2020.

[13] S. Fujishige. *Submodular functions and optimization*. Elsevier, 2005.

[14] S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 291–300, 2004.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[17] J. R. Hershey, S. J. Rennie, P. A. Olsen, and T. T. Kristjansson. Super-human multi-talker speech recognition: A graphical modeling approach. *Comput. Speech Lang.*, 24(1):45–66, Jan. 2010. ISSN 0885-2308. doi: 10.1016/j.csl.2008.11.001. URL `https://doi.org/10.1016/j.csl.2008.11.001`.

[18] E. Hovy, L. Gerber, U. Hermjakob, C.-Y. Lin, and D. Ravichandran. Toward semantics-based answer pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research*, 2001. URL `https://www.aclweb.org/anthology/H01-1069`.

[19] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In C. A. C. Coello, editor, *Learning and Intelligent Optimization*, pages 507–523, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-25566-3.

[20] K. Jamieson and A. Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In A. Gretton and C. C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 240–248, Cadiz, Spain, 09–11 May 2016. PMLR. URL `https://proceedings.mlr.press/v51/jamieson16.html`.

[21] V. Kaushal, R. Iyer, S. Kothawade, R. Mahadev, K. Doctor, and G. Ramakrishnan. Learning from less data: A unified data subset selection and active learning framework for computer vision. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1289–1299. IEEE, 2019.

[22] K. Killamsetty, D. Bhat, G. Ramakrishnan, and R. Iyer. CORDS: COResets and Data Subset selection for Efficient Learning, March 2021. URL `https://github.com/decile-team/cords`.

[23] K. Killamsetty, D. Sivasubramanian, G. Ramakrishnan, A. De, and R. Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5464–5474. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/killamsetty21a.html`.

[24] K. Killamsetty, D. Sivasubramanian, G. Ramakrishnan, and R. Iyer. Glister: Generalization based data subset selection for efficient and robust learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(9):8110–8118, May 2021. URL `https://ojs.aaai.org/index.php/AAAI/article/view/16988`.

[25] K. Killamsetty, X. Zhao, F. Chen, and R. K. Iyer. RETRIEVE: Coreset selection for efficient and robust semi-supervised learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=jSz59N8NvUP`.

[26] K. Kirchhoff and J. Bilmes. Submodularity for data selection in machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 131–141, 2014.

[27] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter. Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets. In A. Singh and J. Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 528–536. PMLR, 20–22 Apr 2017. URL `https://proceedings.mlr.press/v54/klein17a.html`.

[28] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[29] T. Krueger, D. Panknin, and M. Braun. Fast cross-validation via sequential testing. *J. Mach. Learn. Res.*, 16(1):1103–1155, jan 2015. ISSN 1532-4435.

[30] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019.

[31] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4): 541–551, 1989.

[32] Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[33] E. LeDell and S. Poirier. H2o automl: Scalable automatic machine learning. In *Proceedings of the AutoML Workshop at ICML*, volume 2020, 2020.

[34] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *J. Mach. Learn. Res.*, 18(1):6765–6816, jan 2017. ISSN 1532-4435.

[35] L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, M. Hardt, B. Recht, and A. Talwalkar. A system for massively parallel hyperparameter tuning. *arXiv preprint arXiv:1810.05934*, 2018.

[36] L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, J. Ben-tzur, M. Hardt, B. Recht, and A. Talwalkar. A system for massively parallel hyperparameter tuning. In I. Dhillon, D. Papailiopoulos, and V. Sze, editors, *Proceedings of Machine Learning and Systems*, volume 2, pages 230–246, 2020. URL `https://proceedings.mlsys.org/paper/2020/file/f4b9ec30ad9f68f89b29639786cb62ef-Paper.pdf`.

[37] X. Li and D. Roth. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002. URL `https://www.aclweb.org/anthology/C02-1150`.

[38] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

[39] M. Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization techniques*, pages 234–243. Springer, 1978.

[40] B. Mirzasoleiman, J. Bilmes, and J. Leskovec. Coresets for data-efficient training of machine learning models, 2020.

[41] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[42] T. Nickson, M. A. Osborne, S. Reece, and S. J. Roberts. Automated machine learning on big data using stochastic algorithm tuning, 2014.

[43] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[44] J. Pennington, R. Socher, and C. Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, Oct. 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL `https://aclanthology.org/D14-1162`.

[45] N. Pinto, D. Doukhan, J. J. DiCarlo, and D. D. Cox. A high-throughput screening approach to discovering good forms of biologically inspired visual representation. *PLoS Computational Biology*, 5, 2009.

[46] R. Schwartz, J. Dodge, N. Smith, and O. Etzioni. Green ai. *Communications of the ACM*, 63:54 – 63, 2020.

[47] L. N. Smith. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018.

[48] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. M. A. Patwary, P. Prabhat, and R. P. Adams. Scalable bayesian optimization using deep neural networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 2171–2180. JMLR.org, 2015.

[49] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, Oct. 2013. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/D13-1170`.

[50] E. Strubell, A. Ganesh, and A. McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.

[51] K. Swersky, J. Snoek, and R. P. Adams. Multi-task bayesian optimization. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL `https://proceedings.neurips.cc/paper/2013/file/f33ba15effa5c10e873bf3842afb46a6-Paper.pdf`.

[52] K. Swersky, J. Snoek, and R. P. Adams. Freeze-thaw bayesian optimization. *CoRR*, abs/1406.3896, 2014. URL `http://arxiv.org/abs/1406.3896`.

[53] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. 2019. In the Proceedings of ICLR.

[54] K. Wei, Y. Liu, K. Kirchhoff, C. Bartels, and J. Bilmes. Submodular subset selection for large-scale speech training data. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3311–3315. IEEE, 2014.

[55] K. Wei, Y. Liu, K. Kirchhoff, and J. Bilmes. Unsupervised submodular subset selection for speech data. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4107–4111. IEEE, 2014.

[56] T. Yu and H. Zhu. Hyper-parameter optimization: A review of algorithms and applications. *arXiv preprint arXiv:2003.05689*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] . Please refer to Section 4 of the main paper.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] . We discuss broader impacts and societal impacts in Section 4 of the main paper.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Please refer to Appendix A
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Please refer to Appendix F.4
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Please refer to Appendix F.4
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Please refer to Appendix F.4
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] Please refer to Appendix B
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We are providing the source code in the supplementary material
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] We are using publicly available datasets, the authors pointed us to the links.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] Our data does not use personally identifiable information or offensive content. Check Appendix F.3
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]