Auto$k$RF: Efficient and Scalable Local Ensemble Learning with Automated Hyper-parameter Tuning for Large-Scale Classification

Minh-Trung Nguyen[1], Thanh Ma[1], Thanh-Nghi Do[1,2]

[1] College of Information and Communication Technology
Can Tho University, 94000-Cantho, Vietnam
[2] UMI UMMISCO 209 (IRD/UPMC)
Sorbonne University, Pierre and Marie Curie University-Paris 6, France
{nmtrung,mtthanh,dtnghi}@ctu.edu.vn

Abstract. This paper introduces auto$k$RF, a family of automated learning algorithms designed to optimize hyper-parameters for k local Random Forest models in large-scale classification tasks. The auto$k$RF algorithms automatically determine an appropriate number of clusters k to partition the training dataset, then train separate Random Forest models within each cluster. This localized approach enables efficient parallel classification on multi-core CPUs. To improve model performance, each cluster applies the .632 bootstrap estimator, while hyper-parameter tuning is carried out using grid search (GS), reinforcement learning (RL), and Bayesian optimization (BO). Experiments on the ImageNet dataset (1,281,167 images, 768 features, 1,000 classes) demonstrate that our proposal (kRF, auto$k$RF-GS, auto$k$RF-RL, and auto$k$RF-BO) consistently outperform the standard Random Forest algorithm, achieving significant improvements in both classification accuracy and training efficiency. On a Linux Ubuntu 24.04 system with an Intel Core i5-12400 CPU (4.4 GHz, 6 cores, 12 threads) and 32 GB of RAM, these algorithms complete the ImageNet classification task in 9.2, 10.83, 11.04, and 10.58 minutes respectively, with classification accuracy over 88%.

Keywords: Random forest · Local learning · Automatic hyper-parameters tuning · Large-scale dataset

1 Introduction

The rapid expansion of data volume and the increasing complexity of feature spaces in modern computing applications pose significant challenges to conventional machine learning algorithms. As datasets grow in size and dimensionality, these algorithms often encounter difficulties related to computational scalability, sensitivity to noise, and the ability to extract relevant patterns from high-dimensional data. Addressing these issues requires more robust, adaptive approaches capable of handling diverse and complex information landscapes without compromising accuracy or efficiency.

In traditional machine learning, Random Forest (RF [5]) is a robust and scalable approach. By using ensemble learning and randomization, it minimizes variance and enhances generalization across diverse data distributions. Its ability to support parallelization also makes it ideal for large-scale learning where computational efficiency is important [21]. Despite their strengths, RFs encounter scalability issues with very large datasets, as constructing each decision tree requires a full scan of the training data—resulting in training times that increase linearly with both dataset size and the number of trees [17]. Additionally, in high-dimensional spaces, RFs may be less effective at capturing intricate feature interactions compared to more sophisticated models [26].

To overcome the challenges mentioned, we introduce a parallel local learning algorithm called kRF, inspired by clustering-based approaches, for efficient classification of large-scale datasets. Unlike traditional global RF models, which often encounter scalability limitations, kRF builds multiple local RFs concurrently by dividing the data using the k-means algorithm [22] and training independent models within each cluster. This approach reduces computational load and supports parallel processing on multi-core systems. To eliminate the need for manual hyper-parameter tuning, we introduce the auto$k$RF algorithms, which automates the optimization process by integrating multiple strategies: grid search [2,33], the .632 bootstrap estimator [15], hill climbing heuristics [28], Q-Learning [36], and Bayesian Optimization [30]. The auto$k$RF-GS variant combines grid search, bootstrap sampling, and hill climbing, while auto$k$RF-RL and auto$k$RF-BO use Q-Learning and Bayesian Optimization, respectively, to adapt hyper-parameters dynamically based on validation performance. It is important to note that this study primarily focuses on enhancing the performance of the RF algorithm, rather than conducting a comparative analysis across multiple classifiers. While improvements are achieved in both training time and classification accuracy, our central objective lies in optimizing computational efficiency—specifically, reducing training time when working with large-scale datasets. This focus addresses a critical challenge in the era of big data, where the scalability and responsiveness of machine learning models are key to their practical applicability.

The numerical test results on the ImageNet dataset [7] show that kRF, auto$k$RF-GS, auto$k$RF-RL, and auto$k$RF-BO consistently surpass the standard RF algorithm, delivering notable gains in classification accuracy and training efficiency. On a Linux Ubuntu 24.04 system equipped with an Intel Core i5-12400 CPU (4.4 GHz, 6 cores, 12 threads) and 32 GB of RAM, these algorithms completed the ImageNet classification task in 9.2, 10.83, 11.04, and 10.58 minutes, respectively, achieving classification accuracy above 88% (a specific comparison presented in Section 3.3).

The rest of this paper is structured as follows. Section 2 describes the kRF algorithm and the proposed methods, including auto$k$RF-GS, auto$k$RF-RL, and auto$k$RF-BO, for automated hyper-parameter optimization in large-scale dataset classification tasks. Section 3 details the empirical results. Section 4 discusses related work and provides a brief discussion, followed by the conclusion and future directions in Section 5.

2 Methods

As previously outlined, the core emphasis of this study is on developing an enhanced variant of the Random Forest (RF) algorithm. Consequently, a concise overview of the standard RF methodology will be provided in the following section to establish the foundational context. In the following sections, our proposed improvements and the novel algorithmic framework will be presented in detail.

2.1 Random forest

Random Forests (RF), developed by Breiman [5], are a robust ensemble technique that integrates bootstrap aggregating (bagging) with decision trees to improve accuracy and generalization in supervised learning tasks. The learning algorithm aims to construct multiple decision trees during training and combining their outputs to improve accuracy and stability. Each tree is built on a random subset of the data and features, and the prediction are made by majority voting (for classification) or averaging (for regression). This approach reduces overfitting and enhances robustness. Additionally, *it supports parallel processing*, making it efficient for deal with large datasets. Nevertheless, as the number of trees in the ensemble increases, the computational cost also grows significantly. Although parallelization can mitigate this to some extent, large-scale datasets such as ImageNet [7] still pose considerable scalability challenges. In addition, the reliance on manual hyper-parameter tuning hinders the widespread use of RF in large-scale or automated machine learning workflows [26]. In this study, we introduce novel algorithms designed to enhance the performance and scalability of the RF. The technical details and methodological advancements of these proposed algorithms will be thoroughly discussed in the following sections.

2.2 Learning algorithm for the k-local random forest model (kRF)

Scaling the RF algorithm has become increasingly important due to the surge in data volume and complexity in modern large-scale classification tasks. Although RF is widely recognized for its robustness and strong predictive performance, it suffers from several limitations when applied to massive datasets. Notably, training a single RF model on a large dataset can be computationally expensive, memory-intensive, and difficult to parallelize efficiently, leading to substantial delays and system bottlenecks.

Hence, to scale the RF algorithm for large-scale classification tasks, we propose the k-local RF algorithm (kRF), which introduces a scalable and computationally efficient alternative. The motivation behind this approach lies in the principle of *localized learning*. Instead of constructing a global model on the entire dataset, we partition the data into k disjoint clusters using the k-means algorithm. Each subset captures a regionally homogeneous portion of the input space, allowing independent RF models to be trained in parallel. This design naturally supports multi-core computation and significantly reduces training time, as each local model handles a smaller and more coherent data distribution. As

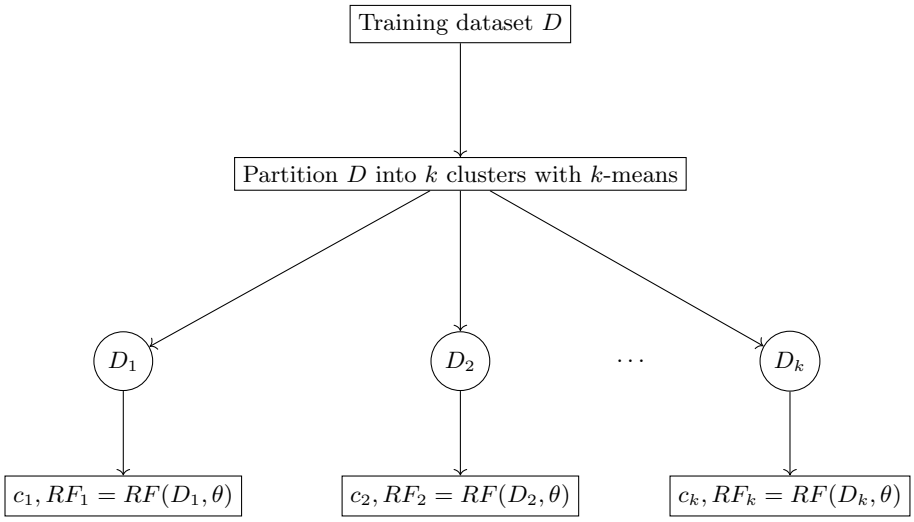

Fig. 1: Flowchart illustrating the process of parallel training for a k-local random forest model. The chart begins with a box labeled "Training dataset D", which is partitioned into k clusters using k-means. Arrows lead to circles labeled D_1, D_2, ..., D_k. Each circle connects to a box showing the formula c_i, $RF_i = RF(D_i, theta)$, where i ranges from 1 to k, representing the training of random forest models on each cluster.

illustrated in Figure 1, the training process of kRF consists of two main stages: (1) partitioning the training dataset D into k disjoint subsets $\{D_1, D_2, \ldots, D_k\}$ using k-means clustering algorithm [22], and (2) training independent RF models on each cluster. This localized approach enables efficient parallel training across multiple CPU cores, as described in Algorithm 1. The input to the algorithm includes a training dataset D containing m datapoints, a specified number of clusters k, and a set of hyper-parameters θ that configure the Random Forest training process. The output is a set of k local Random Forest models, each trained on a distinct cluster and associated with its corresponding centroid, enabling efficient and scalable classification through localized inference.

Moreover, during inference, the system intelligently selects the most appropriate model by identifying the nearest cluster centroid to the input data point. This targeted prediction strategy not only improves computational efficiency but also enhances model accuracy by reducing the influence of irrelevant patterns found in distant regions of the feature space. In particular, a formal definition is introduced as follows:

Definition 1 (Nearest-Centroid Local Model Selection). *Given a set of k cluster centroids $\{c_1, c_2, \ldots, c_k\} \in \mathbb{R}^n$, each associated with a local Random Forest model RF_j, the Nearest-Centroid Local Model Selection assigns a test*

datapoint $x \in \mathbb{R}^n$ to the local model RF_{NN} based on the closest centroid in Euclidean space:

$$RF_{NN}(x) \leftarrow RF_{j^*}, \quad \textit{where} \quad j^* = \arg\min_{j \in \{1,\ldots,k\}} \|x - c_j\|_2.$$

Algorithm 1 Parallel training algorithm for k-local random forests (kRF)

input:
training dataset D with m datapoints;
number of clusters k;
hyper parameters $\theta = \{\text{n\_estimators}, \text{max\_depth}, \text{max\_features}\}$
output:
k local RF models trained on k clusters

begin
  /* *k-means performs data clustering on D in parallel;* */
  Create k clusters denoted by $D_1, D_2, \ldots, D_k$
  Compute their corresponding centers $c_1, c_2, \ldots, c_k$
  /* *Parallel training of the k-local RF model;* */
  for $i \leftarrow 1$ to k
    /* Learning a local RF from D_i; */
    $RF_i \leftarrow \text{RF}(D_i, \theta)$
  end
  return kRF-model $= \{(c_1, RF_1), (c_2, RF_2), \ldots, (c_k, RF_k)\}$
end

According to the computational framework introduced by Hastie et al. [18], the training complexity of a standard RF algorithm is given by $O(T \cdot m \cdot n' \cdot c \cdot \log m)$, where T denotes the number of trees, m is the number of training datapoints, n represents the number of features (the maximum number of random features $n' = \sqrt{n}$), and c classes. When training is parallelized across a processor with P cores—assuming that each decision tree is constructed independently—the complexity can be reduced to:

$$O\left(\frac{T}{P} \cdot m \cdot n' \cdot c \cdot \log m\right) \tag{1}$$

The proposed kRF algorithm enhances computational efficiency by dividing the dataset into k disjoint clusters and training k independent local RF models in parallel. The computational cost of executing k-means clustering with t iterations on a processor with P cores is given by:

$$O\left(\frac{m}{P} \cdot n \cdot k \cdot t\right) \tag{2}$$

Remark 1 (Computational Complexity and Speedup of kRF with P processors). Assuming that each cluster is balanced, the size of each cluster is $\frac{m}{k}$, and the

number of classes in each cluster is $\frac{c \cdot r}{k}$, where $1 < r < k$, and the computational workload is evenly distributed across P processors, the total training complexity of the kRF algorithm can be expressed as:

$$\mathcal{O}\left(\frac{T}{P} \cdot m \cdot n' \cdot \frac{c \cdot r}{k} \cdot \log \frac{m}{k}\right), \tag{3}$$

where T is the number of trees, m the number of training samples, n the number of features, and c the number of classes.

By omitting the clustering overhead, the theoretical speedup ratio of kRF over standard RF becomes:

$$\text{Speedup} = \frac{k \cdot \log m}{r \cdot (\log m - \log k)} \tag{4}$$

This result demonstrates that kRF offers significant computational gains in large-scale settings, especially with increasing k, though it may involve trade-offs in accuracy and model generalization.

The speedup ratio in formula 4 demonstrates that the kRF algorithm can substantially reduce training time compared to the standard RF. As emphasized in prior works [4,9,11], the parameter k gives a trade-off between computational efficiency and generalization performance. Specifically, increasing k reduces the computational load by enabling finer-grained parallelism and smaller local models, but it also increases model locality, which may reduce classification accuracy. In contrast, using a smaller value of k tends to preserve global data structure and improve accuracy, despite providing less significant savings in training time. Therefore, choosing an appropriate k is important to balance efficiency and predictive performance in the kRF algorithm.

2.3 Automatic hyper-parameters tuning for k local random forests

The training procedure of the kRF algorithm, as outlined in Algorithm 1, is governed by four key parameters: the number of local models k, and the hyper-parameter set $\theta = \{\texttt{n\_estimators}, \texttt{max\_depth}, \texttt{max\_features}\}$. As emphasized in prior studies [5,26], these hyper-parameters play a pivotal role in shaping the performance, generalization ability, and computational efficiency of Random Forest models. However, determining optimal values often proves difficult, particularly for non-specialist users, due to the complex interdependencies and dataset-specific behaviors involved in the tuning process. Specially, the manual tuning can be challenging for non-expert users.

To address the limitations of manual hyper-parameter tuning, we propose auto$k$RF algorithms, which automate the selection of hyper-parameters, such as `k`, `n_estimators`, `max_depth`, and `max_features`, for each local RF model within the kRF framework. Unlike conventional global tuning approaches, Auto$k$RF algorithms assign a distinct hyper-parameter set θ_i to each local model trained on its corresponding data partition D_i, optimizing for classification accuracy.

Algorithm 2 Automated hyper-parameter tuning for k-local random forests (auto$k$RF)

input:
training dataset D with m datapoints
output:
k local RF model
begin
  Set $k \leftarrow \lfloor \frac{m}{500} \rfloor$
  / k-means performs data clustering on D in parallel; */*
  Create k clusters denoted by $D_1, D_2, \ldots, D_k$
  Compute their corresponding centers $c_1, c_2, \ldots, c_k$
  / Parallel training of the k-local RF model; */*
  for $i \leftarrow 1$ to k
    /* Searching the optimal hyper-parameters for a local RF from D_i; */
    $\theta^* = \text{RF-}\{\text{GS}|\text{RL}|\text{BO}\}(D_i)$
    /* Learning a local RF from D_i with θ^*; */
    $RF_i \leftarrow \text{RF}(D_i, \theta^*)$
  end
  return kRF-model $= \{(c_1, RF_1), (c_2, RF_2), \ldots, (c_k, RF_k)\}$
end

To evaluate model performance during tuning, we propose to use the .632 bootstrap estimator [14], due to its simplicity and efficiency in terms of accuracy and time. For each partition D_i, a bootstrap sample B_i is generated, typically containing about 63.2% unique datapoints from D_i; the remaining 36.8% constitute the out-of-bag (OOB) set OOB_i. The model M_i is trained on B_i and evaluated on both B_i and OOB_i, with its prediction error estimated as:

$$\widehat{\text{Err}}^{.632} = 0.632 \cdot \widehat{\text{Err}}(OOB_i, M_i) + 0.368 \cdot \widehat{\text{Err}}(B_i, M_i). \tag{5}$$

This estimator reduces overfitting bias while maintaining low variance, making it well-suited for localized model evaluation.

According to the performance analysis of local learning algorithms [4,11], each local data partition should contain approximately 500 datapoints to achieve a balance between model accuracy and computational complexity. In our proposed auto$k$RF algorithms presented in Algorithm 2, we determine the parameter k such that each resulting cluster contains roughly 500 datapoints for training the k-local RF model. Subsequently, we automate the tuning of key hyper-parameters ($\theta^* = n\_estimators^*$, max_depth^*, $max\_features^*$) during the training process of the k-local RF models.

In Algorithm 2, we introduce *three distinct strategies* for hyper-parameter optimization in the context of k-local Random Forests.

(1) The first approach, referred to as auto$k$RF-GS, leverages exhaustive *Grid Search* combined with bootstrap sampling and hill climbing to systematically explore the parameter space. Moreover, auto$k$RF-GS, automates hyper-parameter tuning during the training of k-local RF models. It performs an exhaustive grid

search over predefined hyper-parameter ranges: `n_estimators` $\in [25, 50, 75, 100]$, `max_depth` $\in [20, 30, 45, None]$, and `max_features` $\in$ ["*sqrt*", "*log*2"]. To improve efficiency, auto$k$RF-GS incorporates a hill-climbing strategy [28] that terminates the search early if no performance gain is observed over two consecutive steps.

(2) Our second automated learning algorithm, auto$k$RF-RL, leverages reinforcement learning [36] to optimize hyper-parameters during the training of k-local RF models. Q-learning is a model-free reinforcement learning technique that learns optimal actions through experience, without requiring a model of the environment. It iteratively updates a Q-table based on the Bellman equation, using observed rewards to refine estimates of action values (`n_estimators`, `max_depth`, and `max_features`). We adopt Q-learning to optimize hyper-parameters for RF within a multi-armed bandit (MAB) framework. Each action in the MAB corresponds to a specific combination of hyper-parameters, and rewards are defined as the negative validation error. At each iteration, the algorithm chooses between exploring new hyper-parameter configurations or exploiting the best-known ones, using an ϵ-greedy policy. The Q-values are updated after each episode based on the observed reward, allowing the model to gradually converge to an optimal configuration.

(3) The third automated learning algorithm, called auto$k$RF-BO, applies Bayesian Optimization (BO [30]) for performing hyper-parameter tuning during the training of k-local RF models. BO approximates the objective function using a surrogate probabilistic model—typically a Gaussian Process (GP) which not only predicts performance but also quantifies uncertainty. At each iteration, an acquisition function, such as Expected Improvement (EI) or Upper Confidence Bound (UCB), is maximized to select the next promising hyper-parameter configuration for evaluation. This technique enables BO to balance exploration of unknown regions with exploitation of high-performing areas, making it ideal for high-dimensional and costly search spaces. In our implementation, BO is integrated into the auto$k$RF-BO algorithm to guide hyper-parameter selection for each local model. Given a local partition (D_i), BO constructs a surrogate model over the hyper-parameter space {`n_estimators`, `max_depth`, `max_features`}, aiming to approximate the validation accuracy. The acquisition function is repeatedly optimized to propose new configurations, which are evaluated to refine the surrogate model.

Remark 2 (Computational Complexity of autokRF). The overall training complexity of the *autok*RF algorithm can be formally estimated as:

$$O(m \cdot n \cdot k \cdot t) + O(k \cdot H) + O\left(T \cdot m \cdot n \cdot \log \frac{m}{k}\right),$$

where m is the number of training instances, n the number of features, $k = \left\lfloor \frac{m}{500} \right\rfloor$ the number of clusters, t the number of k-means iterations, T the number of trees in each forest, and H *the computational cost of hyper-parameter tuning per cluster (via grid search, reinforcement learning, or Bayesian optimization)*. This formulation emphasizes the efficiency gains from localized training and parallel execution, while acknowledging that the tuning phase may incur significant overhead.

3 Numerical Test Results

We aim to evaluate the performance of our proposed algorithms, including kRF, auto$k$RF-GS, auto$k$RF-RL, and auto$k$RF-BO, for large-scale data classification. These algorithms were implemented in Python. For developing kRF, auto$k$RF-GS, auto$k$RF-RL, and auto$k$RF-BO, we used the Scikit-learn library [25] and Keras [6] with TensorFlow [1] as the back-end, and the Bayesian Optimization library [24]. The parallelization was achieved on multi-core CPUs via the joblib library for shared-memory parallelism.

The experiments were performed on a Linux Ubuntu 24.04 system equipped with an Intel Core i5-12400 CPU (4.4 GHz, 6 cores, 12 threads) and 32 GB of RAM. Now, we provide the dataset *(ImageNet utilized in this study)*, parameters and experimental results of our proposal in the following sections.

3.1 ImageNet Dataset

We perform experimental evaluation on the ImageNet ILSVRC2012 dataset [7], a well-known and challenging benchmark consisting of 1,281,167 images across 1,000 classes, commonly used for image classification research.

For feature extraction, we used the pre-trained Vision Transformer (ViT) model [13], specifically the Base variant with a 16×16 patch size. The MLP classification head is removed, and the model is used to extract 768-dimensional feature vectors from the images. The dataset is split into a training set of 1,024,933 images and a test set of 256,234 images.

We assess performance using Top-1 accuracy, which measures the percentage of test images whose top predicted label matches the ground truth. In this 1,000-class classification setting, the expected accuracy from random guessing is just 0.1%.

3.2 Tuning hyper-parameters

Due to the large dataset size and hardware limitations, we only tuned the baseline Random Forest's `n_estimators` hyper-parameter (tested values: 50, 75, 100), while keeping other hyper-parameters (`max_depth`, `max_features`) at their default settings. The results indicate that `n_estimators` = 100 yields the highest classification accuracy. Therefore, we adopt `n_estimators` = 100 as the standard configuration across all methods, capping the maximum number of trees at 100 to ensure both fairness and computational feasibility in the evaluation.

For our new local algorithm kRF, there is a balance between a local learning system's capacity and the number of training examples. A large value of k reduces the training time of kRF but increases the locality of the model, which may lead to a decrease in classification accuracy. Conversely, a small value of k does not significantly reduce training time but can improve the accuracy of the kRF model. And then, the parameter k is typically chosen to create cluster sizes of 200–500, as illustrated in [4,11]. For a training dataset with m datapoints, we tested various values of k: $m/500$, $m/750$, $m/1000$, $m/1250$, and $m/1500$. We found that $k = m/1000$ yielded the best results.

3.3 Classification Results

Classification results are presented in Table 1, with visual comparisons illustrated in Figures 3 and 2, highlighting both accuracy and training time. The baseline Random Forest (RF) model achieves an accuracy of 88.13%, but with a significantly long training time of 187.04 minutes, underscoring its computational inefficiency on large-scale datasets.

Table 1: Classification results

No	Algorithms	Accuracy (%)	Time (min)
1	RF	88.13	187.04
2	kRF	88.32	**9.2**
3	auto$k$RF-GS	88.35	10.83
4	auto$k$RF-RL	88.32	9.93
5	auto$k$RF-BO	**88.41**	10.58

All proposed methods outperform the baseline RF in terms of accuracy while drastically reducing training time. The kRF algorithm achieves an accuracy of 88.32% and completes training in only 9.2 minutes—the fastest among all methods—demonstrating the effectiveness of local learning with parallel execution.

Among the autotuned variants, auto$k$RF-BO delivers the highest classification accuracy of 88.41%, representing a 0.28% improvement over RF. The auto$k$RF-GS and auto$k$RF-RL variants also show consistent gains, with accuracies of 88.35% and 88.32%, respectively, while maintaining training times under 11 minutes.

These results illustrate that the auto$k$RF algorithms consistently enhance both predictive performance and training efficiency, with auto$k$RF-BO offering the best balance between accuracy and computational cost. The improvements, though modest in percentage terms, are meaningful given the scale and complexity of the dataset, and the dramatic reduction in training time makes these approaches well-suited for real-world applications.

4 Related Work

Random Forests, developed by Breiman [5], are robust tools in the classification and regression of high-dimensional and noisy data effectively [8,21,27,31,34]. However, the complexity of the training algorithm for the RF model is very high when handling very large datasets like the ImageNet challenge [7]. Studies addressing the processing of large datasets have been proposed in the works of [4,9,11], reducing algorithm complexity through parallel training of SVM classifiers and local neural networks on multi-core processors.

Furthermore, when training any machine learning model, tuning appropriate hyper-parameters to achieve a high-quality model is a time-consuming task.

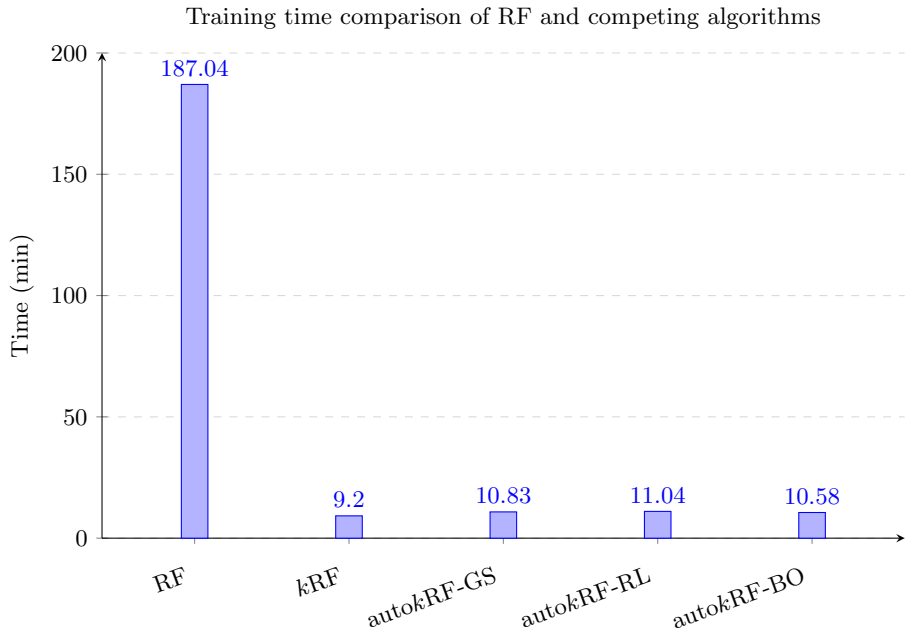

Fig. 2: Bar chart comparing training times of different algorithms. The vertical axis represents time in minutes, ranging from 0 to 200. The horizontal axis lists five algorithms: RF, kRF, auto$k$RF-GS, auto$k$RF-RL, and auto$k$RF-BO. RF has the highest training time at 187.04 minutes, while the other algorithms range from 9.2 to 11.04 minutes, indicating significantly shorter training times.

When users must persistently apply a trial-and-error strategy to find an optimal set of hyper-parameters, it is a very hard and monotonous process. Prior work by [10] developed automated hyper-parameter tuning for local SVM model training, using grid search and the .632 bootstrap method [14]. Auto-WEKA [33] is an automated machine learning tool built on the WEKA platform, designed to simplify the process of selecting and tuning machine learning models. Using Bayesian optimization, it efficiently explores the hyper-parameter space, making it particularly helpful for non-expert users who want to develop effective predictive models without manual trial-and-error. Hyperopt [2] uses Bayesian optimization techniques, specifically Tree-structured Parzen Estimator (TPE) and Random Search, to efficiently explore the hyper-parameter space, used for optimizing complex models like neural networks and ensemble methods, offering a flexible and scalable approach compared to grid search or manual tuning. HPO [3] aims to find the optimal combination of these settings to maximize model accuracy, minimize error, or achieve other performance metrics, using grid search, random search, Bayesian optimization, gradient-based methods, evolutionary algorithms and reinforcement learning.

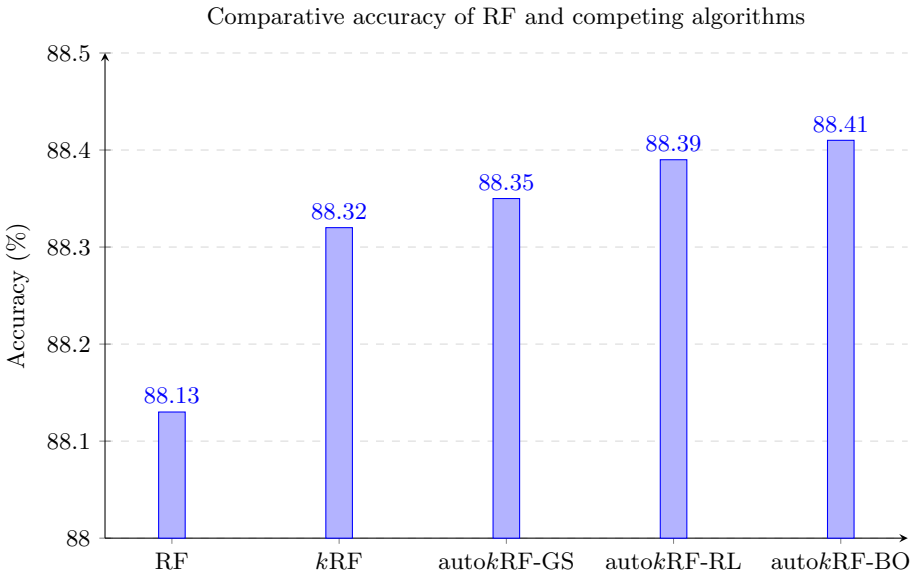

Fig. 3: Bar chart comparing the accuracy percentages of different algorithms: RF, kRF, auto$k$RF-GS, auto$k$RF-RL, and auto$k$RF-BO. The y-axis represents accuracy in percentage, ranging from 88 to 88.5%. RF shows an accuracy of 88.13%, kRF at 88.32%, auto$k$RF-GS at 88.35, auto$k$RF-RL at 88.39%, and auto$k$RF-BO at 88.41%. The chart highlights the comparative performance of these algorithms.

Our research proposes kRF and auto$k$RF algorithms for automatically tuning hyper-parameters [19,26,35] during the parallel training of local RF models for large-scale data classification [12,32]. By employing a local learning paradigm in conjunction with parallelized training across multiple clusters [29], the kRF framework significantly curtails computational overhead while preserving competitive classification performance. Noteworthy performance gains are further realized through the integration of *automated hyper-parameter optimization* [16,38]. A major advantage of the proposed methodology is its architectural adaptability. First, the clustering-based decomposition [20,39] enables efficient model parallelism while retaining performance stability. Second, the inclusion of automated tuning modules effectively navigates the high-dimensional hyper-parameter space [23,37], reducing reliance on expert intervention. Third, the simplicity and interpretability of the k-means partitioning scheme facilitate straightforward integration into existing machine learning workflows.

In general, the kRF and auto$k$RF frameworks provide a principled, efficient, and scalable approach to classification under high-dimensional conditions. Their ability to harmonize computational speed and predictive robustness renders them highly applicable to real-world machine learning systems, particularly in domains where both performance and efficiency are paramount.

5 Conclusion and Future Work

Our paper addresses the growing challenge of large-scale, high-dimensional data in machine learning by proposing a novel parallel local learning algorithm, kRF. By partitioning the data set using clustering of k mean and training independent RF models within each cluster, kRF significantly improves scalability and computational efficiency. To further enhance usability and model performance, we introduce the auto$k$RF suite of algorithms, which automate hyper-parameter tuning through a combination of search strategies, heuristics, reinforcement learning, and Bayesian optimization. Experimental results in the ImageNet dataset demonstrate that kRF, auto$k$RF-GS, auto$k$RF-RL, and auto$k$RF-BO outperform standard RFs, achieving greater 88% classification accuracy and completing training in less than 11 minutes on a modern multicore system. These advancements make kRF and auto$k$RF highly effective for large-scale classification tasks, offering both improved accuracy and computational efficiency.

Future research will pay attention at extending auto$k$RF to support streaming data and online learning scenarios. We also plan to explore alternative clustering techniques to improve the diversity of the local model and the alignment of the boundaries. Finally, our objective is to apply auto$k$RF in other domains such as genomics and cybersecurity to evaluate its generalization across various types of data.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), `https://www.tensorflow.org/`, software available from tensorflow.org
2. Bergstra, J., Komer, B., Eliasmith, C., Yamins, D., Cox, D.D.: Hyperopt: A python library for model selection and hyperparameter optimization. Comput. Sci. Discov. **8**(1), 014008 (2015)
3. Bischl, B., Binder, M., Deng, D., et al.: Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. WIREs Data Mining and Knowledge Discovery **13**, e1484 (2023). `https://doi.org/10.1002/widm.1484`
4. Bottou, L., Vapnik, V.: Local learning algorithms. Neural Computation **4**(6), 888–900 (1992)
5. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001)
6. Chollet, F., et al.: Keras. `https://keras.io` (2015)
7. Deng, J., Berg, A.C., Li, K., Fei-Fei, L.: What does classifying more than 10,000 image categories tell us? In: Computer Vision – ECCV 2010, Part V. Lecture Notes in Computer Science, vol. 6315, pp. 71–84. Springer, Heraklion, Greece (2010)
8. Diaz-Uriarte, R., Alvarez de Andrés, S.: Gene selection and classification of microarray data using random forest. BMC Bioinformatics **7**, 3 (2006)

9. Do, T.N.: Non-linear classification of massive datasets with a parallel algorithm of local support vector machines. In: Advanced Computational Methods for Knowledge Engineering, pp. 231–241. Springer (2015)
10. Do, T.N.: Automatic learning algorithms for local support vector machines. SN Computer Science **1**(2), 1–11 (2020)
11. Do, T.N., Poulet, F.: Parallel learning of local svm algorithms for classifying large datasets. Trans. Large-Scale Data Knowl. Centered Syst. **31**, 67–93 (2016)
12. Do, T.N., Poulet, F.: Classifying very high-dimensional and large-scale multi-class image datasets with latent-lsvm. In: 2016 intl IEEE conferences on ubiquitous intelligence & computing, advanced and trusted computing, scalable computing and communications, cloud and big data computing, internet of people, and smart world congress (uic/atc/scalcom/cbdcom/iop/smartworld). pp. 714–721. IEEE (2016)
13. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net (2021)
14. Efron, B., Tibshirani, R.J.: An Introduction to the Bootstrap. Chapman and Hall/CRC (1993)
15. Efron, B., Tibshirani, R.J.: Improvements on cross-validation: The .632+ bootstrap method. Journal of the American Statistical Association **92**(438), 548–560 (1997)
16. Geitle, M., Olsson, R.: A new baseline for automated hyper-parameter optimization. In: International Conference on Machine Learning, Optimization, and Data Science. pp. 521–530. Springer (2019)
17. Ham, J., Chen, Y., Crawford, M.M., Ghosh, J.: Investigation of the random forest framework for classification of hyperspectral data. IEEE Transactions on Geoscience and Remote Sensing **43**(3), 492–501 (2005)
18. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference and Prediction. Springer, Berlin, 2nd edn. (2009)
19. Hutter, F., Lücke, J., Schmidt-Thieme, L.: Beyond manual tuning of hyperparameters. KI-Künstliche Intelligenz **29**, 329–337 (2015)
20. Jiang, M., Wang, J., Hu, L., He, Z.: Random forest clustering for discrete sequences. Pattern Recognition Letters **174**, 145–151 (2023)
21. Liaw, A., Wiener, M.: Classification and regression by randomforest. R News **2**(3), 18–22 (2002)
22. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proc. of the 5th Berkeley Symposium on Mathematical Statistics and Probability. vol. 1, pp. 281–297. University of California Press (1967)
23. Malu, M., Dasarathy, G., Spanias, A.: Bayesian optimization in high-dimensional spaces: A brief survey. In: 2021 12th International Conference on Information, Intelligence, Systems & Applications (IISA). pp. 1–8. IEEE (2021)
24. Nogueira, F.: Bayesian Optimization: Open source constrained global optimization tool for Python (2014–), `https://github.com/bayesian-optimization/BayesianOptimization`
25. 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., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)

26. Probst, P., Wright, M., Boulesteix, A.L.: Hyperparameters and tuning strategies for random forest. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **9**(3), e1301 (2019). https://doi.org/10.1002/widm.1301
27. Qi, Z.: Text classification algorithm based on random forest. In: Physics Procedia. vol. 25, pp. 1960–1966 (2012)
28. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach (4th Edition). Pearson (2020), http://aima.cs.berkeley.edu/
29. Senagi, K., Jouandeau, N.: Parallel construction of random forest on gpu. The Journal of Supercomputing **78**(8), 10480–10500 (2022)
30. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 25, pp. 2951–2959 (2012)
31. Su, Y., Jelinek, F., Khudanpur, S.: Large-scale random forest language models for speech recognition. In: Proceedings of Interspeech 2007. pp. 598–601 (2007). https://doi.org/10.21437/Interspeech.2007-259
32. Tang, T., Chen, S., Zhao, M., Huang, W., Luo, J.: Very large-scale data classification based on k-means clustering and multi-kernel svm. Soft Computing **23**, 3793–3801 (2019)
33. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of the 19th ACM SIGKDD Conference. pp. 847–855 (2013)
34. Verikas, A., Gelzinis, A., Bacauskiene, M.: Mining data with random forests: A survey and results of new tests. Pattern Recognition **44**(2), 330–349 (2011)
35. Victoria, A.H., Maragatham, G.: Automatic tuning of hyperparameters using bayesian optimization. Evolving Systems **12**(1), 217–223 (2021)
36. Watkins, C.J.: Learning from Delayed Rewards. Phd thesis, King's College, University of Cambridge, Cambridge, England (1989)
37. Wu, J., Chen, S., Liu, X.: Efficient hyperparameter optimization through model-based reinforcement learning. Neurocomputing **409**, 381–393 (2020)
38. Yu, T., Zhu, H.: Hyper-parameter optimization: A review of algorithms and applications. arXiv preprint arXiv:2003.05689 (2020)
39. Zheng, W., Tan, Y., Yan, Z., Yang, M.: A novel clustering-based evolutionary algorithm with objective space decomposition for multi/many-objective optimization. Information Sciences **677**, 120940 (2024)