Growing Random Forest with Diverse Trees through Evolution for Semantic Segmentation

Vorgelegt von
I-Feng Lin
aus Taiwan.

Von der Fakultät VI - Geodäsie und Geoinformatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades
Master of Science
- M.Sc. -
genehmigte Abschlussarbeit.

Gutachter :	Prof. Dr.-Ing. Olaf *Hellwich*
	Prof. Dr.-Ing. Frank *Neitzel*
Betreuer :	Dr.-Ing. Ronny *Hänsch*

Eidesstattliche Versicherung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Berlin, den 18. Dezember 2017 .....

I-Feng Lin

Growing Random Forest with Diverse Trees through Evolution for Semantic Segmentation

Abstract: Random Forest is a machine learning model that has been well researched and has gained established mathematical foundation. An understanding of the underlying mechanisms suggests approaches to improve the model further. By extending the flexibility of the model's technical architecture and designing appropriate criteria for evaluation of its performance, this study progresses from the adjustment of the parameters to a multi-objective optimization of an ensemble classifier that minimizes the correlations between its individual classifiers and maximizes their correctness and precision. Genetic algorithms are used to solve this optimization problem, and through the evolutionary progress, the random forest decision trees achieve higher correctness, greater precision and lower correlation with other trees in the population. Various evaluation techniques and evolutionary strategies are tested to develop the most effective learning model, to which I give the name Evolutionary Random Forest. Finally, the new model is used to solve the semantic segmentation problem with the data in The Cityscapes Dataset. On the contrary to expectations, no significant improvement has been found in the solutions. The new model is helpful on finding local optimums in the solution space, but the conventional model is a fairly effective tool to the problem in this study.

Keywords: Random Forest, supervised learning, genetic algorithms, evolutionary strategies, semantic segmentation, urban scene understanding

Wachsender Random Forest mit verschiedenen Bäumen durch Evolution für semantische Segmentierung

Zusammenfassung: Random Forest ist ein maschinelles Lernmodell, das inzwichen gut erforscht worden ist und auf eine etablierten mathematischen Grundlage aufbaut. Ein Verständnis der zugrundeliegenden Mechanismen weist auf Ansätze hin, das Modell weiter zu verbessern. Durch die Vergrößung der Flexibilität der technischen Architektur des Modells und die Entwicklung geeigneter Bewertungskriterien für seine Leistung wird die Anpassung der Parameter zu einer multiobjektiven Optimierung eines Ensemble-Klassifikators, die die Korrelationen zwischen einzelner Lerner minimiert und die individuelle Korrektheit und Genauigkeit maximieren. Genetische Algorithmen werden benutzt, das Optimierungsproblem zu lösen, und durch die evolutionäre Prozess erreichen die Random Forest Entscheidungsbäume höhere Korrektheit, großere Genauigkeit und geringere Korrelation mit anderen Bäumen in der Population. Verschiedene Evaluationstechniken und Evolutionsstrategien werden getestet, um das effektivste Lernmodell zu entwickeln, das ich Evolutionärer Random Forest nenne. Schließlich wird das Modell verwendet, um das Problem der semantischen Segmentierung mit den Daten in The Cityscapes Dataset zu lösen. Entgegen der Erwartungen wurde keine signifikante Verbesserungen gegenüber dem Standardemodell gefunden. Das neue Modell ist hilfreich, um lokale Optimale im Lösungsraum zu finden, aber das konventionelle Modell ist ein ziemlich effektives Werkzeug für das Problem in dieser Studie.

Keywords: Random Forest, überwachtes Lernen, Genetische Algorithmen, Evolutionärer Algorithmus, semantische Segmentierung

Acknowledgments

Thank Tsung Cho Chang Foundation in Taiwan for the two-year scholarship. Money always helps.

The time frame of the study covers my family's Europe trip and my wedding in Denmark. I send my respects to those who can manage research, family, and marriage, and thank all the distractions that keep me joyful to bring the study to this status.

Contents

Chapter 1

Introduction

Contents

This chapter gives an overview of the problem discussed in this study and of algorithms for solving it found in previous research. First, Section 1.1 summarizes the background needed to understand the problem in this study and introduces a number of algorithms to solve it. Next, Section 1.2 introduces the main concept of a new approach and develops the idea into an empirically-tested implementation, which are the main contributions of this study. Section 1.3 briefly summarizes the studies and contributions of other researchers, which the work presented here builds on.

1.1 Background

Semantic segmentation breaks down an image into parts and classifies each of them into a pre-defined class. Automatic semantic segmentation has a wide range applications from urban scene understanding [Gislason 2006] [Cordts 2017] to remote sensing image classification [Belgiu 2016]. Figure 1.1 illustrates an example of an image and its semantic segmented results, in which roads, vehicle, and humans are labeled with different colors. It is owing to the nature of some applications that predictive models require have high segmentation speed as well as high accuracy.

Random Forest [Breiman 2001] is one of the most accurate and efficient learning algorithms well-researched [Raczko 2017] and it has been successfully being applied to image classification [Bosch 2007] and pixel labeling [Gislason 2006] [Stückler 2012]. Random forest is an ensemble learning method, by which multiple decision trees are generated, and their predictions are combined to make a collective prediction for classification or regression [Liaw 2002]. Ensemble learning for classification can be used to resolve the high variance caused by a single classifier, which is usually poor in generalizing training samples [Opitz 1999]. Random Forest has been used to solve pixel-level semantic segmentation for RGB-depth images [Stückler 2012] and RGB-depth videos [Stückler 2015].

An ensemble composed of many weak classifiers, which have simple and erroneous assumptions of data, lead to a classifier that has a higher bias [Opitz 1999]. The root

Figure 1.1: On the left is a image taken from the view of a streetcar. On the right is its result of semantic segmentation: driveways in purple, pedestrian areas in pink, other vehicles in blue, humans in red, and buildings in gray. The images are captured from The Cityscapes Dataset [Cordts 2016].

of this issue is the trade-off between bias and variance [Breiman 1996b], which can be adjusted by manipulating hyperparameters (e.g., number of trees and their maximal size) and tree growing strategies (e.g., node splitting schemes) in a random forest model, but there are challenges and limitations. First, the optimal architecture and hyperparameters are often problem-dependent [Louppe 2014]. Simple forests may have too few parameters to optimize the solutions, while complex ones take much time to tune. Second, though the trees in a forest have different parameters, they all share the same set of hyperparameters that enable only one growth strategy. Third, the trees grow with a random factor without communication. Some of them may have similar 'weaknesses,' which decrease the predictive performance of a random forest classifier by making identical wrong predictions.

There are a variety of attempts in improving the predictive performance of Random Forest[Fawagreh 2014]. Many of them aimed at decreasing the correlation of the model without lowing its strength by manipulating the parameters and algorithms of inducing decision trees [Robnik-Šikonja 2004] [Gashler 2008]. In other words, adjusting a random forest model is a multi-objective optimization problem because both correlation and strength influence the performance [Breiman 2001].

Genetic algorithms [Goldberg 2006] are often used to solve optimization problems [Gen 2000]. They are a family of models that encode solutions into a chromosome-like data structure and create an optimal one through evolution. Evolution consists of four stages: initialization, selection, reproduction, and termination. First, the initial pool of solutions is often generated randomly. Solutions are evaluated for their performance in solving the problem. Those with better evaluation scores have a higher chance to be cloned into the next generation. Next, solutions are recombined with another to form new solutions. Selection and reproduction repeat until a terminal condition is met, e.g., a performance threshold or a maximal number of repetitions. Ideally, the terminal generation has one or more optimal solutions.

Genetic algorithms can also be applied to Random Forest induction. One study used genetic algorithms to enhance the accuracy of a random forest model, which

generates small forests by cloning trees randomly from a vast forest. The created forest classifiers are evaluated for their correctness rate in a classification problem, and they undergo the selection and reproduction processes to find an optimal forest through evolution. This optimized random forest model has a better predictive performance than the conventional random forest model[Bader-El-Den 2012]. Recently, genetic algorithms were used to encode random forest decision trees and compose a lowly-correlated random forest. Solutions in the algorithm are the encoded random forests decision trees of different parameters that induce trees differently, so as to reduce their correlation. However, the study showed no significant predictive improvement by lowing the correlation [Spitta 2015]. Following this inception, this study reexamines the role of correlation in predictive performance. The models implemented in this study use genetic algorithms with appropriate operations to mixes different tree induction parameters and strategies as a means to obtain accurate and lowly-correlated random forests classifiers.

1.2 Contributions

This study introduces a genetic algorithm enhanced random forest model, which is named *evolutionary random forests,* and examined it for a pixel-level semantic segmentation problem. Unlike a conventional random forest model, the evolutionary model induces diverse trees with evolutionary strategies. Every tree grows based on different hyperparameters, e.g., maximal height, splitting rules, etc., to decrease the similarities in structure. The design results in a much larger number of parameters for the entire forest, proportional to the number of trees. The model optimizes the trees with genetic algorithms by evaluating for their individual performance in the forest. While the performance of a tree can be intuitively defined by its predictive accuracy, such a design can result in the high similarity between trees when there is an optimum that attracts all the trees, and the resulting forest is virtually a conventional random forest. To decrease the similarity of trees, a correlation cost is added to the evaluation besides the predictive accuracy. Any two trees that tend to make the same errors are considered highly correlated, whose score in evaluation is accordingly decreased. The resulting forest is expected to have an accurate ensemble classifier of low variance and low correlation.

This study clarifies the mathematical foundation from previous research that motivates the design of the new model. A series of experiments are conducted to design an powerful learning model. The design choices are discussed in detail based on the empirical results. Finally, the study compares the results with those of a conventional random forest model and tries to explain them by illustrating the parameters of the trained models.

1.3 Related work

This study is motivated by and based on the following studies. [Breiman 2001] introduced the random forest learning model and determined the upper bound error of a resemble classifier from its strength and correlation. [Bernard 2010] showed that this upper bound error had statistical relation with the actual predictive error, which can be minimized by maximizing the strength and minimizing the correlation. These two studies are the basis of this study that a strong and de-correlated random forest can have a better predictive performance. [Robnik-Šikonja 2004] investigated a few approaches to increase the strength and to decrease correlation of random forest classifiers. One of the techniques is using a mix of algorithms to induce decision trees to decrease their correlation. Evolutionary Random Forest realizes the mechanism by introducing tree-level parameters for deciding tree induction schemes. Moreover, Evolutionary Random Forest allows trees that are even more variant to coexist in a forest in different numbers. [Ciss 2015] showed that the test error was bounded by the intrinsic out-of-bag error in Random Forest, which is essential for this study to optimize random forests based on the out-of-bag predictions. Also, the study well examined the linkages between correlation, variance and prediction error, and showed that a low error rate requires a low correlation and a low variance. The finding inspired this study to integrate the measurements into fitness functions. [Goldberg 2006] introduced genetic algorithms for optimization problems [Gen 2000] showed that they were effective in many real-world optimization problems. [Srinivas 1994] introduced the adaptive genetic algorithm model that was applied to a variety of problems successfully with better convergence. The studies motivated this study to solve the random forest optimization problem with genetic algorithms and to adopt the adaptive model.

Chapter 2

Preliminaries

Contents

This chapter explains and illustrates the semantic segmentation problem, introduces state-of-the-art models to solve it, and describes some remaining challenges. Section 2.1 discusses the problem in greater detail, while the other sections are about generic machine learning models and techniques. Section 2.2 explains classifiers and fundamental learning mechanisms used in these models. The main components of Evolutionary Random Forest, namely random forests and genetic algorithms, are explained in Sections 2.3 and 2.4, respectively.

2.1 Semantic segmentation

Semantic segmentation has been well researched and applied to many applications, but algorithms to improve the performance on this problem are still under rapid development [Thoma 2016]. Image segmentation is the process of partitioning images into regions or segments, which are more meaningful to humans or easier for analysis. While segmentation groups pixels to a segment, semantic segmentation provides more information by labeling pixels with a defined class, which separates itself from others with its high-level meaning (usually to humans,) such as different objects in images. Another closely-related field is object detection, which locates objects in images, often with bounding-boxes, and classifies each of them. In other words, the former is a pixel-level segmentation, and the latter is an instance-level segmentation. Figure 2.1 illustrates their differences. This study focuses only on

the pixel-level semantic segmentation problem, particularly in the application of streetcar vision.

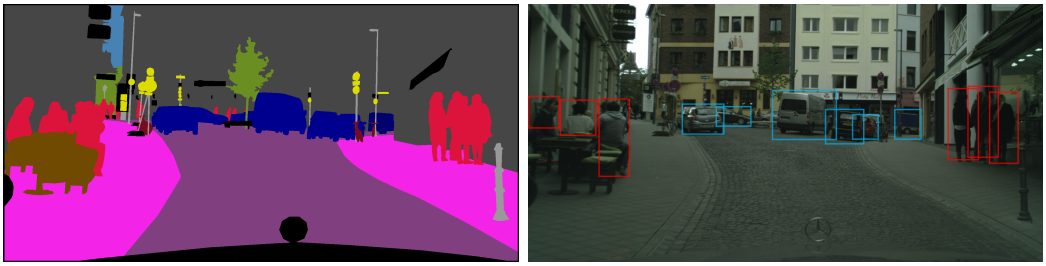

Figure 2.1: Comparison of semantic segmentation and object detection. On the left is a semantic segmented image and on the right is the same image, in which objects are located by bounding boxes and recognized as either cars (blue) or humans (red.)

2.2 Decision trees

Decision tree learning is a learning predictive model, in which pre-defined variables of an observation in question (for example, the RGB values of a pixel to be labeled) are used to predict its target value. When the target values are discrete, such as classes or categories, the trees are called *classification trees*.

The learning algorithm is rather simple. Start at the *root* with all labeled samples. These samples are divided into two groups based on one of their variables. This is a *split,* which creates two *nodes.* Repeat this for the nodes until a node has only one (or no more than n_{node}) sample(s). These nodes are terminal nodes or called *leaves.* In every leaf, a distribution of target values can be calculated, and the majority is its prediction. In the case $n_{node} = 1$, a leaf is simply labeled with the true value of the only sample. In addition, a minimal size of leaves is sometimes used to stop splitting when neither of the new nodes has at least n_{leaf} samples. Decision trees with larger n_{node} and n_{leaf} make fewer variant predictions because more samples are generalized. To predict an unlabeled observation, put it at the root and let it go through the splits to see at which leaf it ends.

Figure 2.2 illustrates a decision tree to classify samples of different shapes with $n_{node} = 3$ and $n_{leaf} = 1$. In this example, the node circled in purple would split if a smaller n_{node} is set because it still has two samples for splitting, but the split would only decrease the generalization without any predictive improvement. On the other hand, the node circled in red would not split if $n_{leaf} = 2$ because one of its subnodes has only one sample. The node would become a leaf labeled with the class triangle because triangles are the majority of classes among its samples.

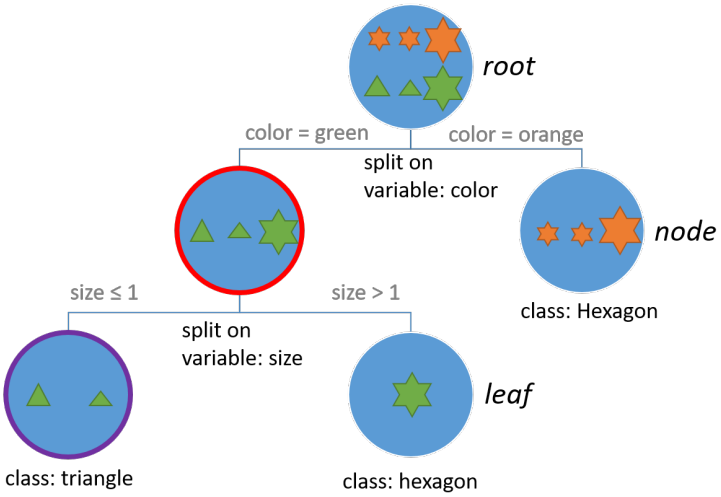

Figure 2.2: A decision tree to classify different shapes of items. In this example, each node must have at least three samples ($n_{node} = 3$) and every leaf must have at least one sample ($n_{leaf} = 1$.)

The two parameters, n_{node} and n_{leaf}, control the *variance* as well as the depth of decision trees. The variance measures the precision of predictions, i.e., how far they are away from the average value, which is a component of predictive performance. More related issues are discussed in Section 2.2.3.

The selection method of variables and their values for splitting is also essential to the predictive performance. Commonly, the selection tries all the available variables with the median values among all samples and chooses the set of variable and value that yields the largest statistical dispersion. The criteria to select a split are discussed in detail in Section 3.1.

2.2.1 Feature extraction

Features are extracted from the training images to train the segmentation models. They create *variables* of observations on which predictions are based. Selection of features is crucial to classifiers. Good features are sets of variables that distinguish the true class from other classes. While more features often provide more information about an observation in question, some variables may have a dependency on others and become redundant, and some are irrelevant to the true class of the observation. In decision trees, redundant and irrelevant variables increase the searching time for the best split, but they do not affect predictive performance because the most discriminative variables are always chosen. However, in other models, including Random Forest, redundant and irrelevant variables are potentially harmful to predictions. Section 4.5.6 discuss the issues with features in depth.

2.2.2 Supervised learning

In supervised learning, models are trained with a large amount of data with ground truth labels. Based on this large set of input and expected output, the training models adjust their parameters for future predictions. Many successful methods in semantic segmentation use supervised learning [Stückler 2012] [Stückler 2015] [Uhrig 2016], whose validity depends on the size and composition of training data and the variability of data to be predicted. An appropriate dataset is essential to any supervised learning models. In image analysis, this involves scenes, scales, lighting, angles of view, etc.

2.2.3 Overfitting

All machine learning methods can result in overfitting due to their intrinsic mechanisms [Louppe 2014]. A model overfits the training data by learning noises in the data instead of the relevant relations between the variables and expected outputs. A fully-grown decision tree ($n_{node} = n_{leaf} = 1$) is bound to overfit the training data in practice because every observation creates a leaf without any generalization. In other words, they have high-variant predictions when the observation of interest is unseen during training. A few tree induction strategies address this issue by making trees smaller (having fewer nodes or leaves) than fully-grown trees. A strategy is to set a minimal number of samples in a node (n_{node}) or that in a terminal node (n_{leaf}) so that a split occurs only when a node reaches the certain size n_{node} and new nodes reach the size n_{leaf}. They allow nodes and leaves to contain more samples and stop the tree from growing further. In many implementations, a maximal height of trees is often used to limit its growth. While these strategies are applied to single decision tree classifiers, other methods use multiple classifiers to avoid overfitting and improve predictive performance. They are collectively known as *ensemble learning* and explained in the following section.

2.3 Ensemble learning

Ensemble methods use a set of classifiers to make predictions by combining their individual predictions or their *votes* in a sense. Ensemble methods outperform single classifiers in predictive performance. [Opitz 1999] [Dietterich 2000] Most ensemble methods require *resampling* the original training set to create new subsets randomly and to train multiple classifiers with different training sets. Resampling is effective in increasing the predictive performance by reducing a resemble learner's *variance* [Opitz 1999]. Do not confuse the variance here with the variance of decision trees mentioned in Section 2.2. The variance of an ensemble measures the disagreement of votes on an observation in question. A clearer definition is given in Section 2.3.2.

One of the resampling algorithms is bootstrap aggregating or called *bagging*, which is a common approach that has been shown to improve the accuracy of classification trees [Breiman 1996a]. Bagging generates multiple training subsets by

randomly sampling from the original training set with replacement. The individual learners are trained with different subsets so that they do not overfit the entire training set. Bagging improves the accuracy of learning methods that are sensitive to the perturbation of training data [Breiman 1996a].

The training samples that are not in the training subset of an individual classifier are called its *out-of-bag samples*. Since the individual has not 'seen' them during training, they can be used to test the accuracy of the individual classifier. Out-of-bag predictions can be obtained by making collectively predictions, in which the individuals only vote on their out-of-bag samples. The out-of-bag error is often used to estimate the strength and correlation of Random Forest [Breiman 2001], which is discussed in Section 2.3.2.

Resampling can also be used to learn imbalanced data, so as to improve classification of rare classes. There are two common ways. One is over-sampling the minority classes so that every class has the same number of samples in the new training set. The other is downsampling the majority classes to achieve the same goal. Downsampling has been shown to be effective in improving prediction of minority classes. [Chen 2004]. Another study shows that downsampling outperforms oversampling in decision tree classifiers. [Drummond 2003]

2.3.1 Random forests

Random Forest is an approach that uses multiple decision trees grown using independently resampled training sets [Breiman 2001]. Because of bagging, trees in Random Forest are not identically trained and thus predict observations differently. Additionally, Random Forest applied a hyperparameter to try m_{try} variables for split selection, instead of every variable, when inducing trees. The mechanism introduces randomness into split selections so that trees classify samples based on variables differently. Conventionally, Random Forest induces all trees using the same algorithm and hyperparameters.

While Random Forest is a state-of-the-art predictive model today, some research further improves its predictive performance. One study mixes multiple tree growing strategies in a forest and shows that this so-called heterogeneous forest is more robust than conventional Random Forest when the number of irreverent features increases. [Gashler 2008]. Another study shows that a mixed set of variable selection functions in a forest yields a significant improvement in accuracy. [Robnik-Šikonja 2004] Similarly, this work aims at improving predictive performance by modifying the composition of a random forest model.

2.3.2 Strength and correlation

In an ensemble classifier naturally desires a high accuracy of its individual classifiers, but the predictive performance decreases when most classifiers vote for the same wrong class. When there are more than two classes, a correct collective prediction can be made even when the absolute majority is wrong if the votes for the correct

class take the relative majority. An optimal ensemble always has more votes for the correct class than for other classes. In other words, an ideal ensemble should have classifiers that agree with each other when their votes are correct but disagree as much as possible when their votes are incorrect. Hence two criteria can be defined for a good ensemble: *strength* and *correlation*, which were defined in a previous study, and their influence on the generalization error was determined in [Breiman 2001]. First, a margin function is defined for Random Forest as follows:

$$mr(x, y) = P\Theta(h(x, \Theta) = y) - \max_{j \neq y} P_\Theta(h(x, \Theta) = j) \tag{2.1}$$

where x is input, y is the true class, $h(x, \Theta)$ is the voted class from tree Θ, and P_Θ is the proportion of votes for a certain class over all trees. Namely, the margin is the proportion of correct votes minus the largest proportion of wrong votes. The strength s of an ensemble is defined as the expectation value of the margin function:

$$s = E_{x,y}[mr(x, y)] \tag{2.2}$$

Moreover, a raw margin function is defined for every tree Θ:

$$rm(\Theta, x, y) = I(h(x, \Theta) = y) - I(h(x, \Theta) = \hat{j}(x, y)) \tag{2.3}$$

where $\hat{j}(x, y)$ is the class that has the largest proportion of votes among false classes.

$$\hat{j}(, x, y) = arg \max_{j \neq y} P_\Theta(h(x, \Theta) = j) \tag{2.4}$$

The correlation of two trees (Θ, Θ') are defined as the correlation of $(rm(\Theta, x, y), rm(\Theta', x, y))$. The variance of a decision tree in the ensemble is the variance of the $rm(\Theta, x, y)$.

The upper bound of the generalization error of a random forest is noted as PE and calculated by its strength and correlation:

$$PE \leq \frac{\bar{\rho}(1 - s^2)}{s^2} \tag{2.5}$$

Following the equation, the error upper bound can be minimized by minimizing the ratio of correlation over strength:

$$\frac{\bar{\rho}}{s^2} \tag{2.6}$$

Furthermore, a study showed that the testing error rate could be minimized by selecting the trees that maximize the strength and minimize the correlation of Random Forest [Bernard 2010]. Moreover, the test error is bounded by the intrinsic out-of-bag error in Random Forest [Ciss 2015]. Based on the above, it is feasible to optimize a random forests model by evaluating forests based on the correlation-over-strength ratio calculated from the out-of-bag error.

2.4 Genetic algorithms

Genetic algorithms are methods to optimize solutions by exploring encoded solution spaces with recombination and variation operators. Candidates of solutions are encoded in a data structure, called a *genotype* or alternatively a *chromosome*, and evaluated for the *fitness* of their output (phenotype) among the whole *population*. Through a selection mechanism, those chromosomes with a higher degree of fitness have a better chance to be cloned into the next *generation*. Some chromosomes can be chosen multiple times, while those not chosen are removed. Conventionally, the size of the population stays the same through generations. The remaining individuals are allowed to exchange part of the information on their chromosomes, i.e., to exchange *genes*, and this recombination is called a *crossover*. Also, genes may change randomly into another value, which is called a *mutation*. The members of the new generation are evaluated for their fitness, and the whole process repeats until a specified number of generations has been reached.

A genetic algorithm is determined by the mechanisms of crossovers and their probability, and by the probability of random mutations. The outcome of the algorithm is much determined by the fitness criteria used to evaluate the individuals in each generation.

Genetic algorithms have been used to grow decision trees [Fu 2001] [Fu 2003] and to build random forests [Bader-El-Den 2012]. However, to the author's best knowledge, there has been no successful attempt to optimize random forests with genetic algorithms with trees using different values of parameters.

Genetic algorithms present proposed solutions as chromosomes and encode their parameters as genes. Conventionally, parameters are expressed as binary digits, and a chromosome is simply a concatenation of them. The advantage of the representation is making genetic changes naive, i.e., to exchange strings of bits in crossovers or flip binary bits in mutations. However, in simple concatenation, genes are not equally subject to genetic operations, as some genes take more space on a chromosome than others. Besides, it is difficult to control the effects of mutations, as a change of upper bits has a more significant impact on an individual's performance, and this is often unwanted in a late generation when a certain degree of progress has been made.

Another encoding scheme is to encode each parameter as one gene with a data structure suitable for its value range. Because genetic operations are a gene-level recombination or variation, they need specialized implementations for different data structures.

2.4.1 Selection

When creating a new generation, members are selected with replacement from the previous generation. The selection scheme should not only let fitter members survive but also allow diversity in the new population. Tournament selections have been shown to be robust and useful compared to many other schemes [Blickle 1995]

[Miller 1995]. The algorithm is simple: k members are firstly selected without replacement from the last generation, and the one that has the highest fitness is cloned into the new generation. Repeat it until the new population is full. k is a hyperparameter to adjust the degree of randomness. The selection is completely random when $k = 1$ and only in favor of the fittest when k equals to the size of the population.

2.4.2 Genetic operators

The main operators to recombine encoded parameters are crossover and mutation. The two operations occur with independent probabilities. Crossovers are defined as exchanging genes locating at the same positions in chromosome between two individuals. A common scheme is a uniform crossover, which uses a ratio to decide how many genes the two corresponding chromosomes exchange, regardless of their locations. In other words, every gene has an equivalent and independent probability of crossover. The mutation changes the value of a gene into a random value within its predefined range. 2.3 illustrates how the genetic operations work.

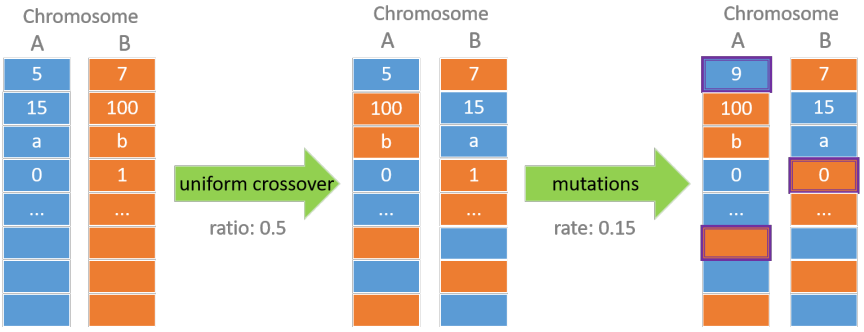

Figure 2.3: Genetic operations: two chromosomes exchange genes uniformly with a exchange ratio 0.5 and genes mutates with a mutation rate 0.15. Mutated genes are circled in purple.

In practice, adjusting the probabilities of crossover and mutation is problem-dependent. One common strategy is starting with high probabilities and decreasing them over generations so that the algorithms explore the solution space widely in early generations and converge in later ones. Another strategy, called adaptive genetic algorithms, adjusts the probabilities based on individual fitness. Individuals that are less fit have higher probabilities to undergo crossovers and mutations, so as to disrupt their offspring in the next generation. This algorithm was shown to be faster in finding the optimums than conventional genetic algorithms [Srinivas 1994].

Chapter 3

Implementation

Contents

This chapter looks into the design of Evolutionary Random Forest. First, Section 3.1 specifies different tree growing strategies used in this model. Section 3.2 explains how the parameters in growth strategies are encoded into chromosomes for genetic algorithms. Next, Section 3.3 introduces the recombination operators of chromosomes and illustrates their way of working. Finally, Sections 3.4 and 3.5 explain how the random forest decision trees, arriving from these chromosomes, are evaluated and how the chromosomes are selected for the next generation in evolution.

3.1 Growth strategies

In Evolutionary Random Forest, every tree has different parameters for their growth strategies. The parameters are the minimal size of nodes and that of leaves (n_{node} and n_{tree} explained in Section 2.2,) the number of trials for variable selection for splitting (previously mentioned m_{try} in Section 2.3.1,) the variable and value selection schemes for splitting, and accessibility to variables for selections. This study uses two different variable selection schemes. The first one is the Gini index, used by Classification and Regression Trees [Loh 2011], which is the calculation of impurity of a subset of data. The higher the impurity is, the more likely a sample mismatches the label of the subset according to the distribution of samples. For a node, its Gini is the sum of the square of the probability of each class. Let p_i be a fraction of samples of class i, where $i \in \{1, 2, ...C\}$. Its Gini is defined as follows:

$$Gini(p_i) = 1 - \sum_{i=1}^{C} p_i^2 \tag{3.1}$$

The Gini index after splitting is the sum of the subnodes weighted by their size. Let $p_{i,k}$ be a fraction of samples of class i in a subnode k, where $k \in \{1, 2\}$ in a

binary decision tree. The decrease of impurity is the subtraction of the Gini index of the parent node from that of the subnodes:

$$\begin{aligned} ImpurityDecrease_i = Gini(p_i) - \sum_{k=1}^{2} Gini(p_{i,k}) = \\ - \sum_{i=1}^{C} p(y_i)^2 + \sum_{k=1}^{2} \sum_{i=1}^{C} p(v_{i,k}) p(y_i|v_{i,k})^2 \end{aligned} \tag{3.2}$$

The selection scheme picks the variable that maximizes the decrease of impurity among all trials.

The second variable selection scheme is the information gain, used by C4.5 [Quinlan 2014], namely difference in entropy. Entropy is defined as the average amount of information produced by a source of data. The higher entropy is, the less predictable, or the more diverse, a subset is. Entropy is defined as:

$$Entropy(p_i) = - \sum_{i=1}^{C} p_i log_2 p_i \tag{3.3}$$

The information gain can be calculated by subtracting the entropy of the subnodes from that of the parent node:

$$\begin{aligned} InformationGain_i = \sum_{k=1}^{2} Entropy(p_{i,k}) - Entropy(p_i) \\ = \sum_{i=1}^{C} p_i log_2 p_i - \sum_{k=1}^{2} \sum_{i=1}^{C} p(y_i|v_{i,k}) log_2 p(y_i|v_{i,k}) \end{aligned} \tag{3.4}$$

The selection scheme picks the split that maximizes the gain in entropy among all trials.

A threshold value is required to divide the samples into two subsets based on the selected variable. Three value selection schemes are used in this study: linear searches, medium values, and random values. A linear search tries all the values appearing in the samples on the node to split, calculates either the Gini index or the information gain, and selects the value that produces the largest impurity decrease after splitting. The second selection scheme uses the medium value of the values appearing in the samples on the node. The last scheme chooses a random value within the range of the variable.

3.2 Parameter encoding

Every parameter is encoded in one gene of a corresponding data structure as listed in Table 3.1. Variable selection schemes for splitting are either the Gini index (0) or information gain (1). Value selection schemes are linear searches (0), medium values (1) or random values (2). Note that there are a few constraints in the chromosomes

parameter	data structure	range
Number of trials for variable selection	integer	[1 N_{var}]
Minimal size of nodes	integer	[1 100]
Minimal size of leaves	integer	[1 100]
Split variable selection function	boolean	[0 1]
Split value selection function	char	[0 2]
Accessibility to variables for splitting	boolean array	$[0\ 1]^{N_{var}}$
bagging size (in %)	int	[10 100]

Table 3.1: Chromosome: encoded parameters for solutions in genetic algorithm. N_{var} is the number of variables in features.

caused by the nature of decision trees. First, the minimal size of a leaf cannot be larger than that of nodes. In the implementation, the gene is presented by a percentage of minimal node size between 10% and 100%. Second, the number of trials for variable selection cannot exceed the number of available variables for splitting. This constraint is more complicated to hold on chromosomes. Instead, it is addressed in tree growing scheme. The actual number of trials is the minimum of the number of trials and the number of available variables.

3.3 Genetic operations

The genetic operations used in this model are uniform crossovers and mutations. Three different evolutionary strategies are implemented to manipulate the genetic operations differently for convergences. The conventional model starts with a commonly-used crossover probability 0.8 and mutation rate 0.15, which decrease by 3% after every generation.

The adaptive genetic algorithm model, introduced by [Srinivas 1994], uses crossover and mutation probabilities associated with individuals' fitness, the average fitness f_{avg} and the best fitness f_{best} in the population. Let f_i be the fitness of tree i and the operation rate is:

$$operationRate_i = min(k, \frac{k(f_{best} - f_i)}{f_{best} - f_{avg}}) \tag{3.5}$$

where k is a hyperparameter. As suggested by [Srinivas 1994], the implemented model uses $k = 1.0$ for crossovers so that every individual weaker than average recombine their genes through the operation, and uses $k = 0.5$ for mutations. The fitter an individual is, the fewer genetic changes occur on that individual.

The third evolutionary model uses an adaptive genetic algorithm, in which the degree of genetic changes decays over generations. The exchange ratio of uniform crossovers starts from 0.5 and decreases by 3% every generation. Mutations change a gene to another value within a range from its original value. For integers, the mutation range is calculated by the parameter value range multiplying a mutation

range factor, which starts from 1.0 and decreases by 3% every generation. For categorical values (boolean and char values), the mutation range factor indicates the probability that a random jump occurs.

Decreasing probabilities or degrees of genetic operations allow the population explores the solution space globally in early generations and converges at optimal solutions in later ones.

3.4 Evaluation

Members of a population are evaluated by a fitness function for their performance in the population. In Evolutionary Random Forest, the fitness function evaluates a random forest decision tree classifier. Desired solutions in this genetic algorithm model are collections of members (a population of trees) that minimizes the overall correlation-over-strength ratio (Equation 2.6,) which is an evaluation of a forest. The fitness function is designed to favor trees that decrease the ratio when they are added into the forest in progress.

While the accuracy of individual classifiers, which can be calculated with out-of-bag predictions, is supposed to be maximized, the accuracy of an ensemble classifier is optimal only when the individuals are accurate **and** de-correlated. However, the variance of an ensemble increases when the correlation decreases because individuals are encouraged to make different predictions, so the correlation must decrease **faster** than the increasing variance to achieve low prediction error [Ciss 2015].

The trees are evaluated for their correctness, precision, and correlation for the multiple objectives. The correctness of a tree is 1.0 minus its out-of-bag error rate. The precision is the inverse of the variance of the raw margins (Equation 2.4) given all out-of-bag samples. The two measurements define the *healthiness* of a tree i in this study as follow:

$$healthiness_i = correctness_i \times \sqrt{precision_i} = \frac{1 - error_{oob,i}}{\sqrt{variance_i}} \tag{3.6}$$

where $error_{oob,i}$ is the out-of-bag error rate of the tree i. Now consider the correlation of a tree. The correlation here is differently defined from what is in the correlation-over-strength ratio (Equation 2.5) calculated from raw margins. Since only the correlation that involves a wrong prediction should be a cost of fitness, the predictions made correctly by both trees are taken out from the calculation. This correlation is named a *correlation cost* in this study to differentiate it from the correlation of raw margins. The fitness function of the tree i is defined as follow:

$$fitness_i = healthiness_i \times decorrelation_i = \frac{1 - error_{oob,i}}{\sqrt{variance_i}}(1 - \bar{\rho}_i)^\lambda \tag{3.7}$$

where $\bar{\rho}$ is the average correlation cost between the tree and other members of the population, and λ is a hyperparameter to manipulate the influence of correlation costs. The fitness function with $\lambda = 0$ ignore correlation costs and that with a larger λ causes a larger correlation cost.

3.5 Selection

The implemented selection scheme is tournament selections of size $\lfloor\sqrt{n_{tree}}\rfloor$, where n_{tree} is the number of trees. In the selection of the first member, trees' accuracy and variance are the only elements for evaluating fitness. After a member is cloned into the population of the next generation, a correlation cost with this tree will be added to the trees that are less healthy than it, and their fitness is recalculated for the next selection. The correlation cost is added only to the less healthy trees to ensure a healthier tree's chance to be selected not affected by a prior selection of highly-correlated trees.

Chapter 4

Experiments

Contents

This chapter covers the experiments conducted in this study. Section 4.1 introduces the dataset used for training the model and testing it. Section 4.2 explains how features are extracted from the images in this dataset. Section 4.3 describes the measurements applied to stages of the learning process to obtain learning curves, and Section 4.4 explains a calculation applied to smooth the curves for analysis. Section 4.5 describes a series of experiments, comparing models with different architectures and hyperparameters, and arrives at the best models used to solve the chosen problem in this study. The solutions of the problems and the results are presented in Section 4.6.

4.1 Dataset

The dataset used for training in the experiments is from The Cityscapes Dataset [Cordts 2016]. This dataset consists of stereo video sequences of street scenes in different cities and the annotated frames as images, which provide this study the

raw images, their pixel-level labeling, and their depth maps for supervised learning. The training set consists of 2 975 labeled images which are taken in 18 different cities. Every pixel is labeled with one of the 19 classes, in addition to background and license plates of no interest. These classes can also be categorized into seven categories: ground, construction, object, sky, human, and vehicle. In this study, pixels are classified only by their category.

Because of the computing capacity, the experiments in this study are conducted with a smaller but sufficiently effective training subset, which consists of 5 pixels randomly from every image in the training data. The prior distribution of classes is calculated and incorporated for tree predictions. Bagging is weighted by the inverse distribution of classes so that trees draw roughly the same number of samples from each class.

4.2 Features

In the experiments, features are the subtracted RGB values of two offset regions of a pixel in question, following the approach in [Stückler 2012]. First, the extraction generates two regions of random dimensions and at a random relative position to the pixel in question. Next, the offsets and the dimensions of regions are normalized with the depth of that pixel. Pixels in a larger depth (farther away from the viewpoint) create smaller and closer regions for scale invariance and vice versa. The RGB values within the two regions are summed by channel, and the difference of the summed values in three channels between the two regions is one feature, which is composed of three variables. If any part of the two regions is out of the image, fixed (very large) values are assigned to the variables. More features can be generated by generating more pairs of offset regions.

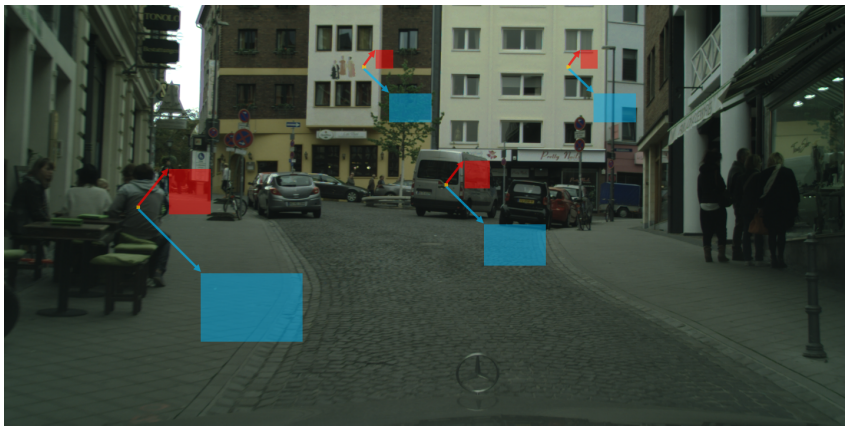

Figure 4.1: Illustration of the feature extraction: the orange pixels are the pixels in question. Two offset regions (blue and red) of random dimensions and at offset positions are generated and normalized with the depth of the pixels in question.

4.3 Learning measurements

Different measurements are made on the models in every generation to show the progress of evolution. The experiments measure the correlation-over-strength ratio (Equation 2.6,) which should be minimized, and the average fitness of the population, which should be maximized. Besides these two measurements, the average variance and correlation of all members in the population (calculated from the margin function Equation 2.4) and the strength of the ensemble classifier (Equation 2.2) are also measured. According to the fitness function (Equation 3.7,) low variance, high strength, and low average correlation maximize the fitness of the individuals, which aims at minimizing the correlation-over-strength ratio of the ensemble classifier.

4.4 Moving average

Neither Random Forest nor genetic algorithms are deterministic. Bagging samples in Random Forest can cause that trees with the same parameters has different predictive performance. Genetic algorithms explore the solution space with randomness introduced from the selection and reproduction process. The randomness brings on noises in the observations of learning measurements that may veil the trend of the learning progress. An idea to discover the underlying trend is averaging some sequent observations to produce smoother curves. Moving average is a calculation for analysis by creating average values of subsets of data. Given a data series p_i, where $i \in \{1, 2, ...N\}$ and a moving-average window size k, the calculated series is as follow:

$$p'_i = \frac{p_{i-\frac{k-1}{2}} + p_{i-\frac{k-1}{2}+1} + p_{i-\frac{k-1}{2}+2} + ... + p_{i+\frac{k-1}{2}}}{k} \tag{4.1}$$

Note that $p'_1, p'_2, ...p'_{k-1}$ and $p'_{N-k+1}, p'_{N-k+2}, ...p'_N$ do not exist because they are near the boundaries of the original series. Figure 4.2 illustrates how the moving average helps to identify the trend of measurements. The plots of the curves calculated with the moving average show a much clearer upward trend of strength and fitness and a downward trend of variance and correlation-over-strength ratio.

In order to illustrate the trends clearly, all plotted curves in the following sections are calculated by a moving average with $k = 5$ to clarify the trends in measurements.

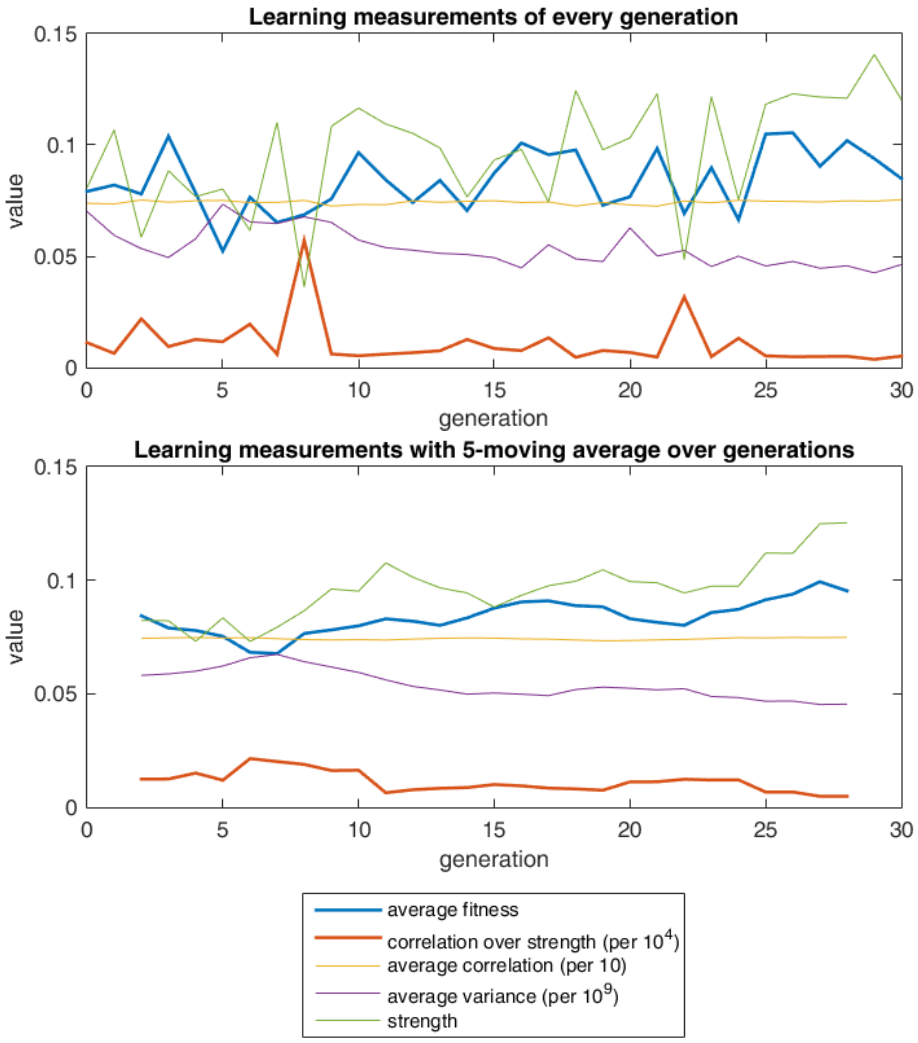

Figure 4.2: Plots of learning measurements without the moving average (top) and with the moving average of $k = 5$ (bottom.) The lines are the average fitness (blue), correlation over strength (orange), average correlation (yellow), variance (purple) and strength (green).

4.5 Experimental models

The architecture and hyperparameters of the evolutionary random forest model are decided through a series of experiments. The following subsections train models with different settings and compare their results. Table 4.1 lists the setup of the prototype of training models.

hyperparameters	
number of features	240
number of trees	25
maximal number of nodes in a tree	2^{14}
number of generations	30
crossover and mutation	adaptive
starting crossover ratio	0.5
crossover ratio decaying rate	3% per generation
starting mutation range factor	1.0
mutation mutation range decaying rate	3% per generation
λ in fitness function	3

Table 4.1: Prototype: the basis of experimental models

4.5.1 Downsized correlation calculation

Conventional Random Forest has a time complexity of $O(n_{tree} \times m_{try} \times n_{samples} log(n_{samples}))$, where n_{tree} is the number of trees, m_{try} is the number of trials of variables at splits, and $n_{samples}$ is the number of samples. In Evolutionary Random Forest, it is essential to calculate correlations between trees for calculating their fitness. The calculation has the time complexity of $O(n_{tree}^2 \times n_{mutualOOB})$, where $n_{mutualOOB}$ is the number of out-of-bag samples shared by two trees, which takes significantly more time when n_{tree} increases. In order to decrease this extra time cost, the technique here uses only a subset of out-of-bag samples for correlation calculation. Note that:

$$\sqrt{n_{samples}} << n_{mutualOOB} \tag{4.2}$$

when $n_{samples}$ is large. The fraction of non-repeated in-bag samples out of all samples for a tree is roughly 0.632 when it draws $n_{samples}$ samples with replacement [Efron 1997], and the equation:

$$n_{mutualOOB} = (1 - 0.632)(1 - 0.632)n_{samples} = 0.135 n_{samples} \tag{4.3}$$

holds because $n_{samples}$ is typically much larger than 1 000.

The following experiment examines if the reduced subset can still achieve a successful evolution. The model using the downsized correlation calculation should have a convergence of fitness, average correlation and correlation-over-strength ratio similar to those in the standard model using all out-of-bag samples for the correlation calculation. For every pair of trees, the downsized model uses only $\sqrt{n_{samples}}$ out of their mutual out-of-bag samples are used for the calculation. For example, let there be 10 000 samples in the training set. The expected value of $n_{mutualOOB}$ is 1 350, but instead this model uses only $\sqrt{10000} = 100$ samples to calculate the correlation between any two trees. Table 4.2 lists the two models' training time and the minimum of the correlation-over-strength ratio among generations. The model using the downsized correlation calculation is more than three times faster than the

standard model and achieves an even lower correlation-over-strength ratio. Figure 4.3 illustrates their correlation over strength, average fitness and correlation of every generation calculated with a moving average. The fitness has an upward trend in both models, and the average correlations of both models converge at close values. The results show that the introduced model can have a convergence at least as good as the standard model. The downsized correlation calculation is used in every other model in this study.

models	standard	downsized
training time	12 696 seconds	4 028 seconds
lowest correlation over strength	21.11	20.67

Table 4.2: Training time of 30 generations used by the standard model and the downsized models and their lowest correlation-over-strength ratio among all generations.

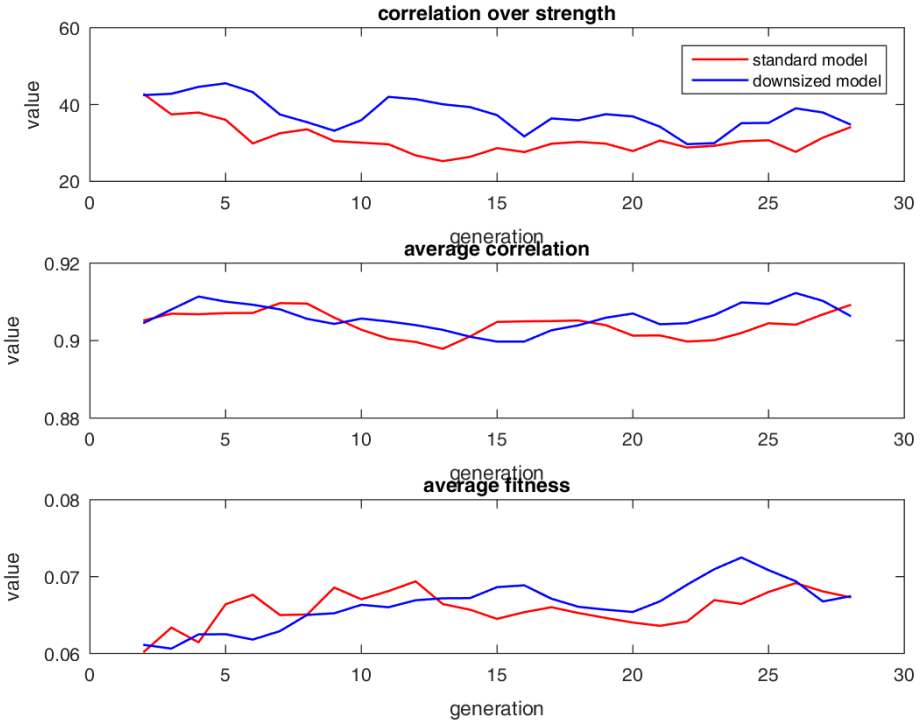

Figure 4.3: Plots of the correlation over strength (top,) the average correlation(middle,) and the average fitness (bottom) of the standard model (red lines) and the model using downsized correlation calculation (blue lines.)

4.5.2 Correlation weight λ

The correctness and precision of trees and their correlations are the factors of the fitness function (Equation 3.7.) The hyperparameter λ allows adjusting the weight on the correlation. The following experiment compares the learning performance of models using different λ values 0.5, 1, 2, 3 and 4. A larger λ the increases fitness faster as the correlation decreases. The ideal output of the fitness function is negatively correlated to the correlation-over-strength ratio of the population so that the ratio is minimized when fitness is maximized. Table 4.3 lists the correlation coefficients between fitness and strength, that between fitness and average correlation, and that between fitness and the correlation-over-strength ratio. The fitness is positively correlated with the strength for all $\lambda > 0$, and it is negatively correlated with the average correlation when $\lambda \geq 2$. The fitness function using $\lambda = 3.$ has the maximal negative correlation between fitness and the correlation-over-strength ratio.

All fitness functions that have a negative correlation with the correlation-over-strength ratio are candidates of a suitable function. Figure 4.4 illustrates the learning measurements of the models using different λ in the fitness function. Except for models using $\lambda = 0$ and $\lambda = 1$, all models decrease the ratio through evolution.

models	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 2$	$\lambda = 3$	$\lambda = 4$
correlation coefficients of average fitness and strength	-0.04	0.07	0.32	0.49	0.56	0.34
correlation coefficients of average fitness and average correlation	0.62	0.37	0.14	-0.04	-0.23	-0.01
correlation coefficients of average fitness and correlation over strength	0.44	-0.17	-0.16	-0.29	-0.42	-0.22

Table 4.3: Correlation coefficients between the fitness, average correlation, and correlation-over-strength ratio in models in different fitness functions.

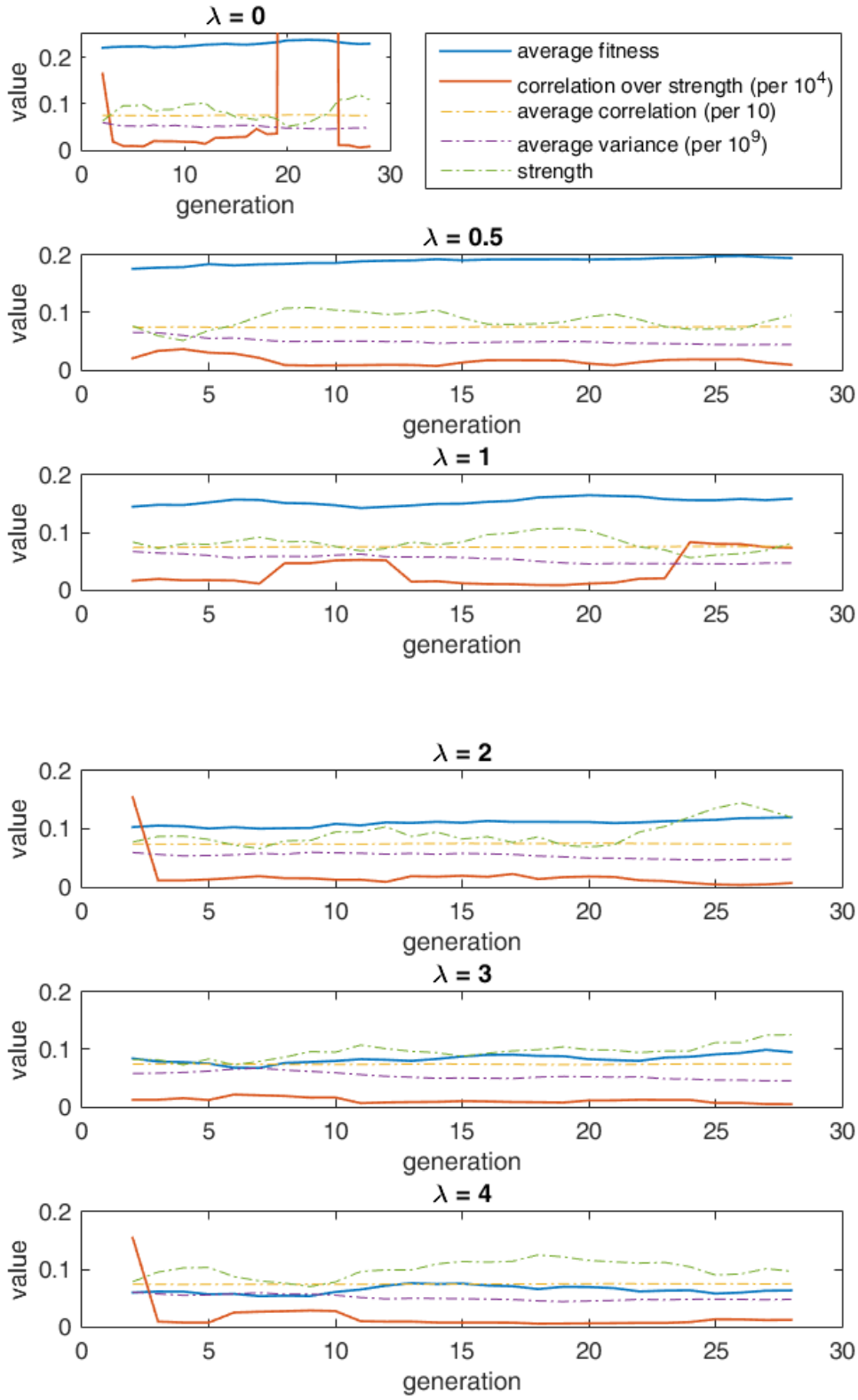

Figure 4.4: Plots of learning measurements of models using different lambda values. The lines are the average fitness (blue,) correlation over strength (orange,) average correlation (yellow,) variance (purple,) and strength (green.)

4.5.3 Evolutionary strategies

Genetic algorithms can have different strategies to make populations converge to one or multiple optimal solutions. The conventional approach is to use crossover and mutation rates that reduce over generations. With fewer genetic operation changes in later generations, the members of the final population will be much similar to each other and ideally optimal. Another approach is using adaptive genetic algorithms, in which individuals with lower fitness have greater probabilities of genetic operations, as explained in Section 2.4.2. Three different models are designed to investigate the outcome of these approaches to solve the problem in the study.

The first model uses a conventional algorithm, in which genetic changes occur more in early generations than later ones. The second is an adaptive genetic algorithm model, in which the genetic operation rates are associated with the members' fitness. The third one also uses an adaptive genetic algorithm, but the uniform crossover ratio and mutation range factor decay over generations to improve the convergence.

Figure 4.5 plots the learning measurements of the three models. The goal of convergences is to maximize the average fitness and thereby minimizes the correlation-over-strength ratio stably in later generations. In the illustration, the curves of the conventional algorithm show no clear increasing or decreasing trends. The adaptive genetic algorithm model has a slightly decreasing trend of the ratio, but it is unstable even in the last generations. The third model, using an adaptive genetic algorithm with a decaying crossover ratio and mutation range factor, shows a clear downward trend of the ratio.

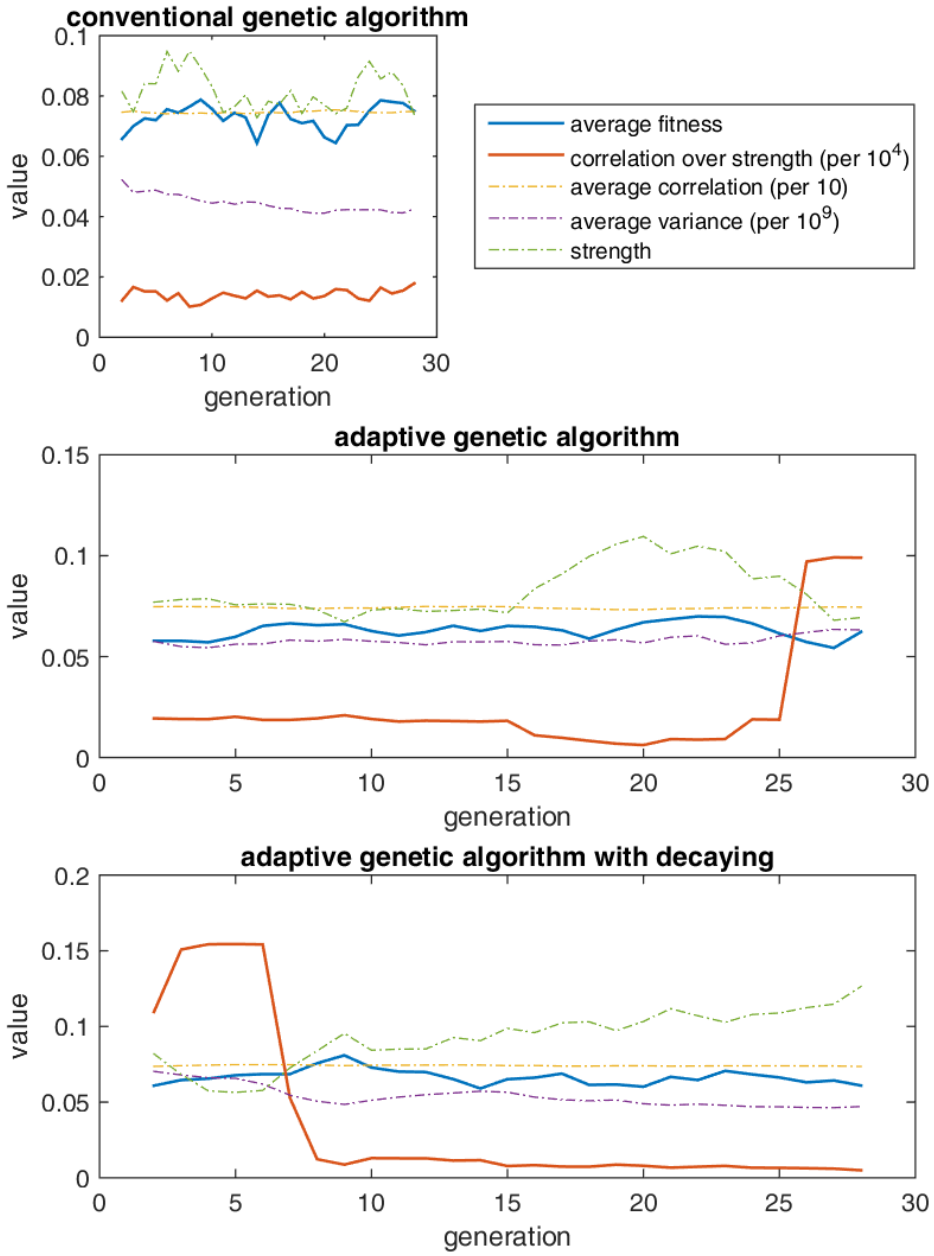

Figure 4.5: Plots of learning measurements of models using different evolutionary strategies: the conventional model (top,) the adaptive genetic algorithm model (middle,) and the adaptive genetic algorithm model with decaying crossover ratio and mutation range factor (bottom.) The lines are the average fitness (blue,) correlation over strength (orange,) average correlation (yellow,) variance (purple,) and strength (green.)

4.5.4 Number of trees

In Random Forest, more trees are helpful to decrease the variance, but it is problem-dependent to determine a sufficient number of trees for satisfactory performance. Unnecessarily many trees result in a waste of time on growing and testing them, especially on the correlation calculation explained in Section 4.5.1. The following experiment trains five different models using 5, 10, 15, 20, 25, and 30 trees. They extract 60 features from each training sample. Table 4.4 lists their training time and lowest correlation-over-strength ratio among all generations, and Figure 4.6 plots the learning measurements of different models for comparison. The average fitness decreases when there are more trees due to the increasing correlation in the population. The strength increases with the number of trees, with exceptions of models using 15 and 30 trees. The model using 25 trees has the lowest correlation-over-strength ratio.

models	5 trees	10 trees	15 tress	20 trees	25 trees	30 trees
training time (seconds)	484	737	1 164	1 734	2 310	2 980
lowest correlation over strength	38.40	30.95	39.54	30.50	24.43	31.07

Table 4.4: Training time and lowest correlation-over-strength ratio among generations of the prototype using different numbers of trees

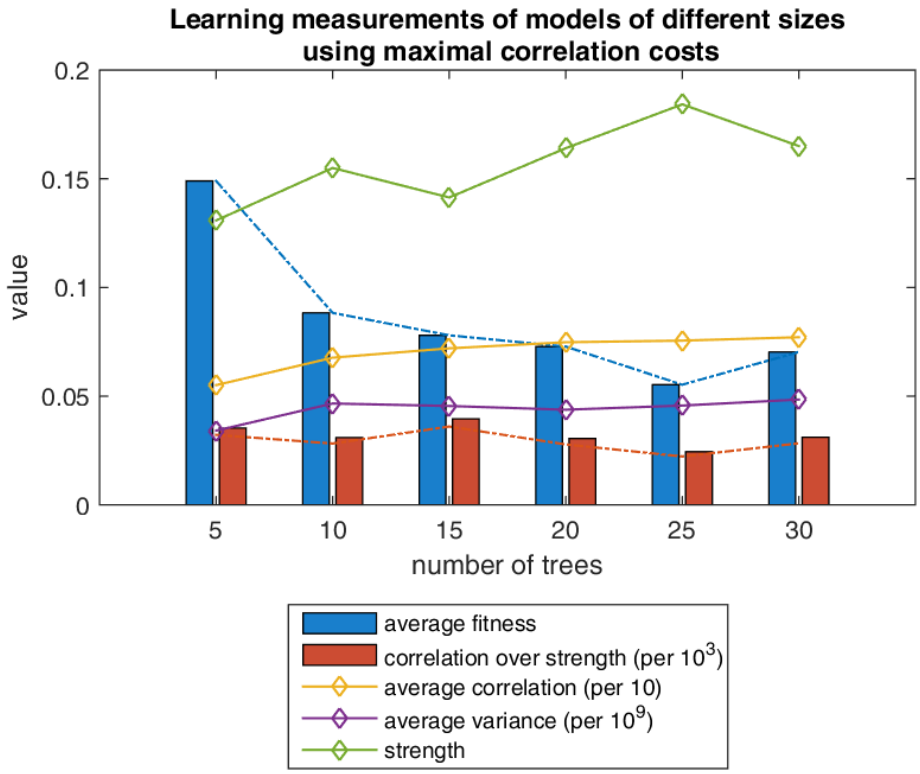

Figure 4.6: Plots of learning measurements of models using 5, 10, 15, 20, 25, and 30 tress. The bars indicate the lowest correlation-over-strength ratio among all generations (orange) and the average fitness of the same generation (blue). Lines indicate the average correlation (yellow), variance (purple) and strength (green) of the corresponding generation.

4.5.5 Average v.s. maximal correlation costs

The correlation cost in the fitness function presented in Equation 3.7 is defined as the average correlation cost between a tree and others. In the speculation, a maximal correlation cost may also effectively decrease the correlation of an ensemble. The following experiments compare the prototype with a model using maximal correlation costs (by replacing $\bar{\rho}$ with $\max(\rho)$ in Equation 3.7.) In the first experiment, the two models use 10 and 25 trees to compare their average fitness and correlation-over-strength ratio as illustrated in Figure 4.7. Note that the here introduced model has a lower fitness because a maximal cost is naturally larger than an average cost. The results show that the model using maximal correlation costs has a significantly smoother convergence when using 10 trees, and it also has comparable results with the prototype when using 25 trees.

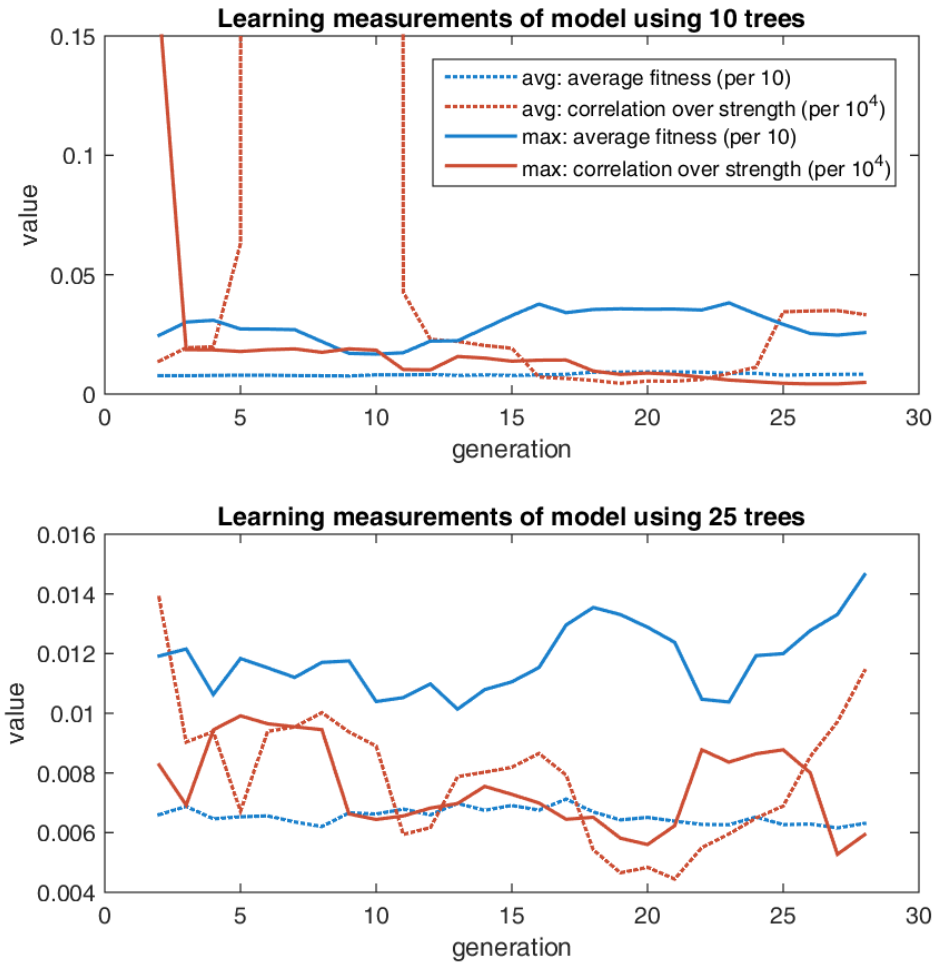

Figure 4.7: Plots of learning measurements of models using 10 and 25 tress: the average fitness (blue lines) and the ratio of correlation over strength (orange lines). The dotted lines illustrate the model using average correlation costs and the solid lines illustrate that using maximal correlation costs function.

The good performance of the models using maximal correlation costs intrigues a second experiment to find out how well this model works using more trees. Figure 4.8 illustrates the results of using 5, 10, 15, 20, 25, and 30 trees, and shows the model using 10 trees has the lowest correlation-over-strength ratio. The strength and the average fitness decrease when more trees are added.

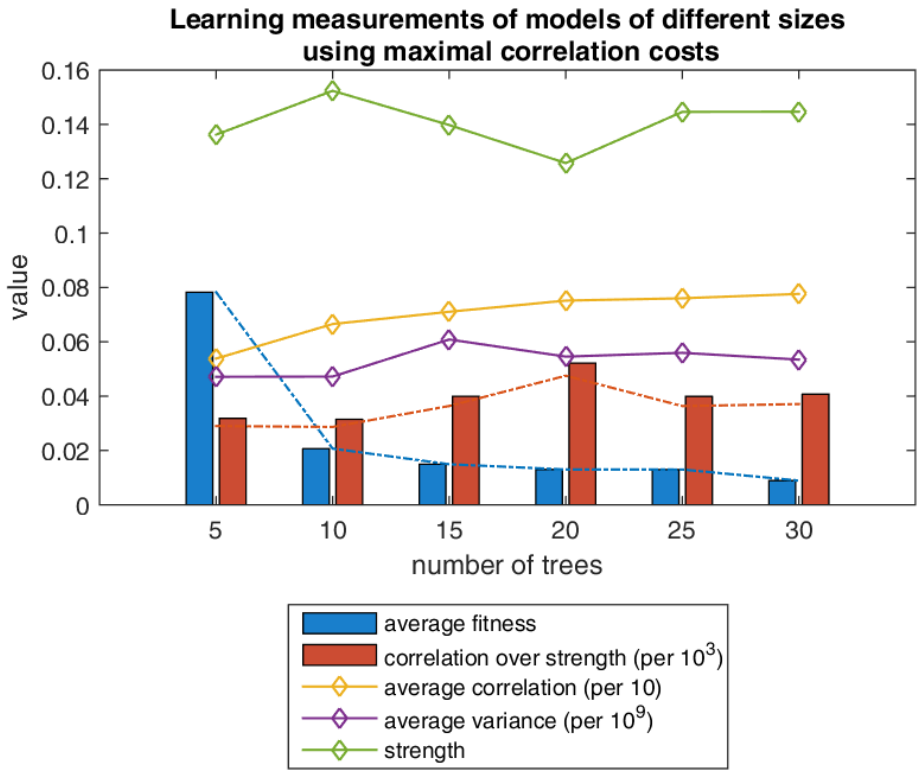

Figure 4.8: Plots of learning measurements of models using maximal correlation costs and 5, 10, 15, 20, 25, and 30 trees. The bars indicate the lowest correlation-over-strength ratio among all generations (orange) and the average fitness of the same generation (blue). Lines indicate the average correlation (yellow), variance (purple) and strength (green) of the corresponding generation.

4.5.6 Number of features

As explained in Section 4.2, more features can be generated simply by generating more pairs of random regions for comparison. However, irrelevant features can be harmful to the predictive performance of the random forest algorithm, because they decrease the probabilities of selecting good variables in a limited number of trials. Besides, as the number of variables increases, it allows a larger number of trials for variable selection (m_{try} mentioned in Section 2.3.1.) With more trials for splitting a node, the selection takes more time accordingly. The following experiment tries to find the number of features that optimizes the predictive performance.

All five models in the experiment are trained with the same set of pixels, but they generate different numbers of features from each pixel: 10, 20, 40, 80, and 160 features. Since every feature creates three variables, the upper bound of m_{try} in the models is set with 30, 60, 120, 240, and 480, respectively. All models have 25

trees. Table 4.5 lists their training time and lowest correlation-over-strength ratio among generations. The increment of time is small compared to the increment of the amounts of data. The lowest correlation-over-strength ratio decreases as the number of features increases when it is not larger than 80. Figure 4.9 plots the learning measurements and shows that the model using 80 features achieves the best performance with a significantly lower average correlation and a relatively high strength.

	models				
number of features	10	20	40	80	160
Number of trials for variable selection	[1, 30]	[1, 60]	[1, 120]	[1, 240]	[1, 480]
training time (seconds)	2 578	2 919	3 869	4 026	5 951
lowest correlation over strength	69.44	57.49	47.35	30.09	43.16

Table 4.5: Training time and the lowest correlation-over-strength ratio of models containing different numbers of features

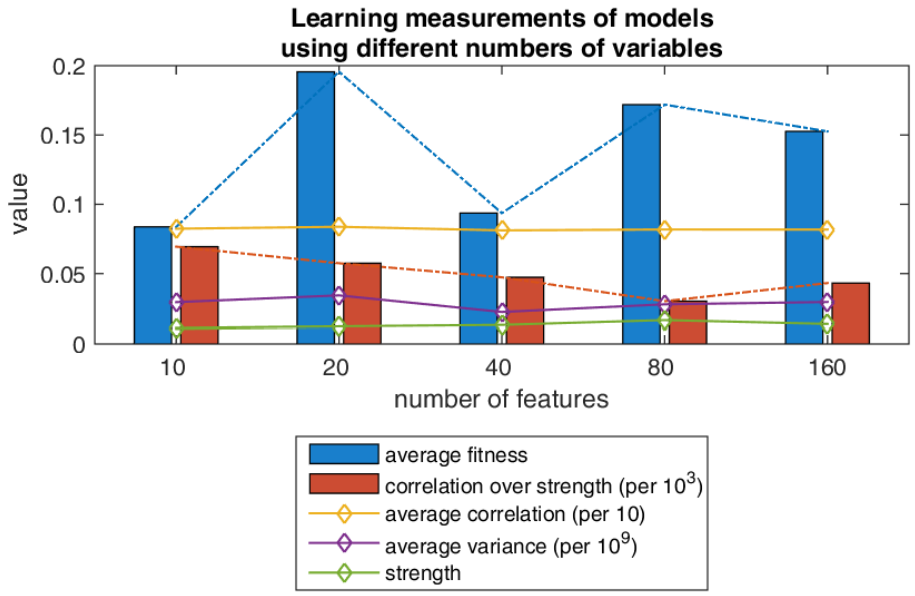

Figure 4.9: Plots of learning measurements of models using 10, 20, 40, 80, and 160 features per pixel. The bars indicate the lowest correlation-over-strength ratio among all generations (orange) and the average fitness of the same generation (blue). Lines indicate the average correlation (yellow,) variance (purple,) and strength (green) of the corresponding generation.

4.6 Final models

Based on the experiments in Section 4.5, two final models are set up for further experiments to compare their performance with conventional models. Table 4.6 lists the hyperparameters of the first model, named Model A, which uses the setup that has the best predictive performance in the experimental results. Note that the bagging size is removed from chromosomes and fixed as a hyperparameter because trees that have a larger bagging size always have higher fitness in the experimental models.

The second model, named Model M, is largely the same as Model A, except for a few differences listed in Table 4.7. The key contrast is that Model A uses an average correlation cost and Model M uses a maximal cost. Based on the results in Section 4.5.4, models using an average correlation cost need more trees to perform well, and those using a maximal cost perform worse with too many trees. Hence, the two models also use different numbers of trees: 25 for Model A and 10 for Model M.

hyperparameters	
number of features	80
number of trees	25
maximal number of nodes in a tree	2^{14}
number of generations	50
bagging size	100% with replacement
crossover and mutation	adaptive
starting crossover ratio	0.5
crossover ratio decaying rate	3%
starting mutation range factor	full range
mutation range factor decaying rate	3%
λ in fitness function	3
correlation cost	average

Table 4.6: The architecture and hyperparameters of Model A

differences	Model A	Model M
number of trees	25	10
correlation costs	average	maximal

Table 4.7: The differences between Model A and Model M

4.6.1 Model A v.s. Model M

This experiment compares the learning performance of two models. The training is accelerated by downsized correlation calculation discussed in Section 4.5.1. Figure 4.10 plots the learning curves of the two models. Model A, which has a steady improvement through the whole evolution process, shows greater success in a convergence of learning curves than Model M. On the other hand, Model M has more

jumps in performance through the evolution before it arrives at an optimum and then jumps away. Table 4.8 lists their training time and lowest correlation-over-strength ratio. It is worth noted that Model A requires roughly two and half times the training time of Model M to achieve the slightly better performance.

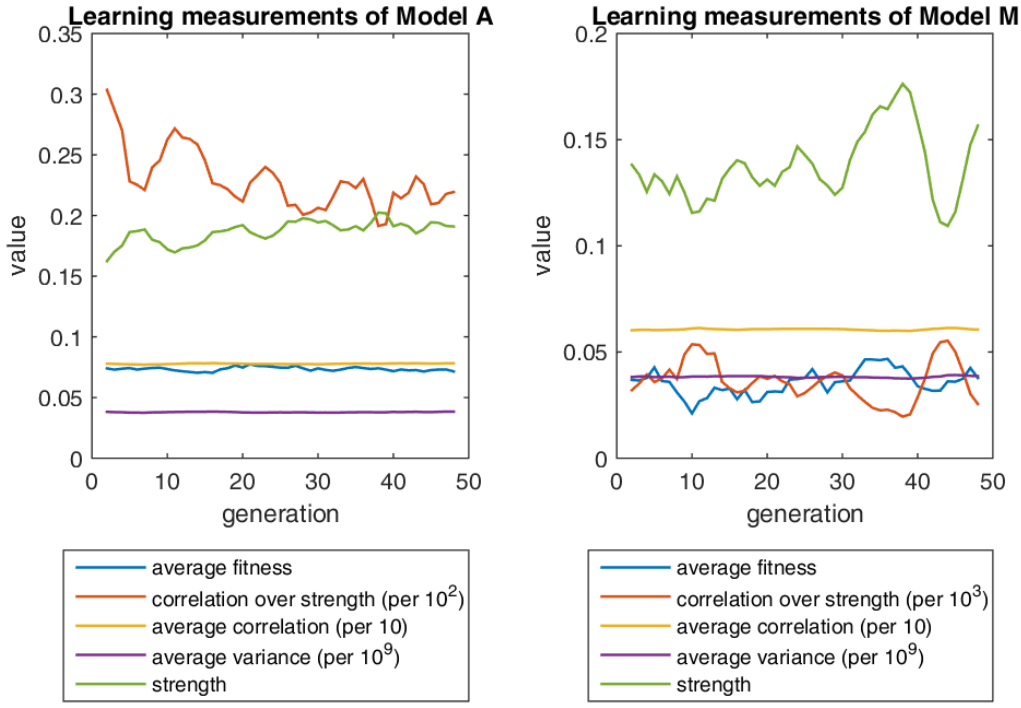

Figure 4.10: Plots of learning measurements of Model A (left) and Model M (right): the average fitness (blue,) correlation over strength (orange,) average correlation (yellow,) variance (purple,) and strength (green.) Note that the order of magnitude of the ratio changes between two plots for better illustration.

models	Model A	Model M
training time (seconds)	5 289	2 205
lowest correlation over strength	17.65	18.14

Table 4.8: Training time and lowest correlation-over-strength ratio of Model M and Model A

To understand the changing of the trees caused of the learning process, Figure 4.11 and 4.12 illustrate some of the tree parameters (minimal node and leaf sizes, and split selection schemes) of Model A and Model M in different generations through evolution. Both models show clear convergences in the final generation. Model A has a big cluster of trees close to the parameter set $m_{try} = 45, n_{node} = 80$, and various n_{leaf}. Around a half of the trees use the Gini index and random values for the split

selection, and the other half use the information gain with a mix of value selection schemes. Model M has two clusters of trees. One is close to the set of parameters $(m_{try} = 50, n_{node} = 50, n_{leaf} = 10)$, using the Gini index with either linear searches or medium values for split selection. The other is close to $(m_{try} = 160, n_{node} = 80)$ and various n_{leaf}, mostly using the information gain, either with medium values or random values.

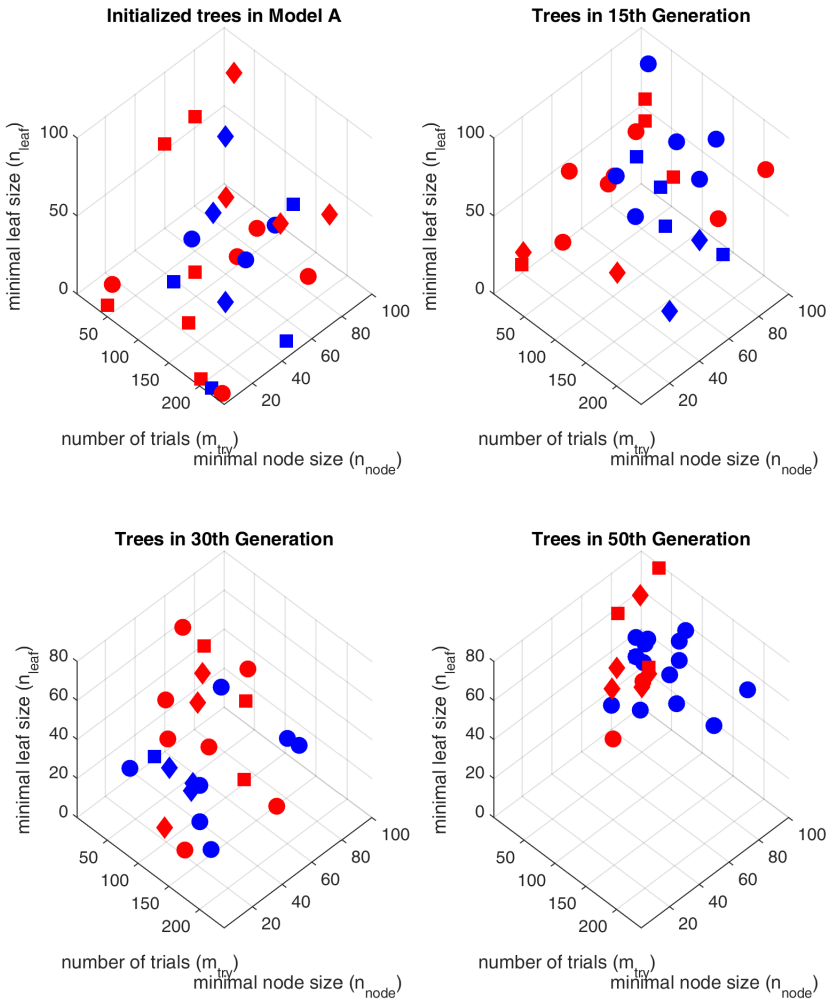

Figure 4.11: Parameters of trees in different generations grown by Model A. Every icon represents a tree. The x-value is its number of trials for variance selection. The y-value is its minimal number of samples in a node. The z-value is its minimal number of samples in a leaf. The colors indicate its variable selection scheme: red is the Gini index and blue is the information gain. The shapes indicate its value selection scheme: a diamond is linear searches, a square is medium values, and a circle is random values.

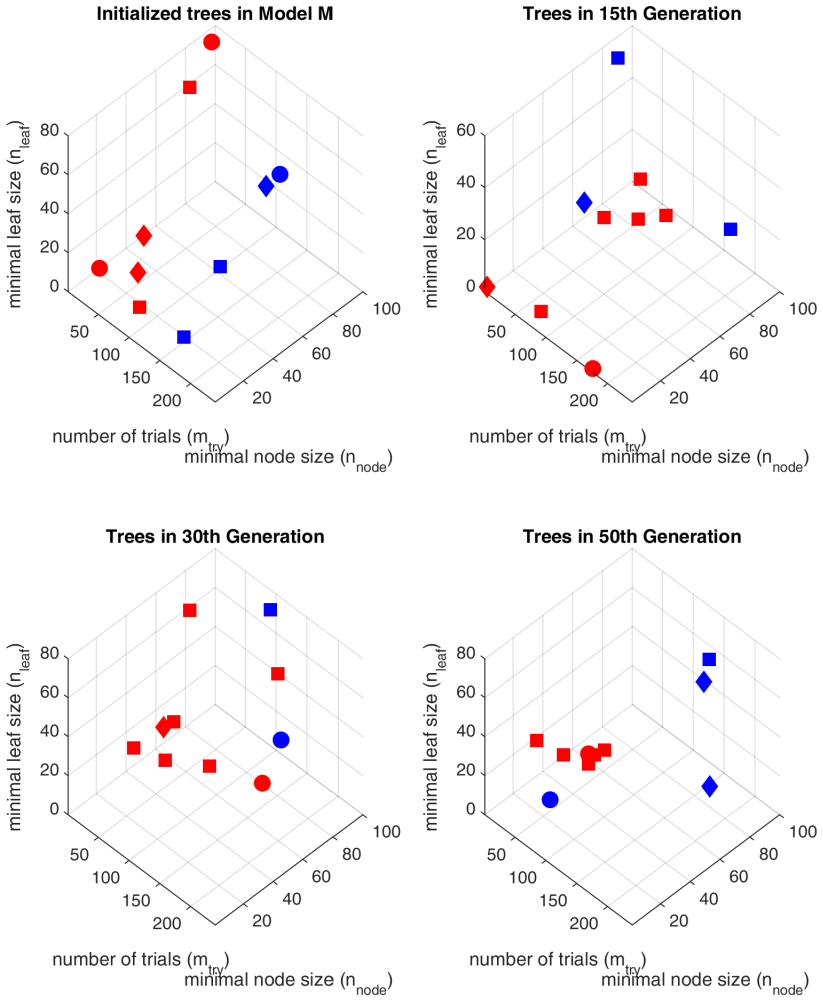

Figure 4.12: Parameters of trees in different generations grown by Model M. Every icon represents a tree. The x-value is its number of trials for variance selection. The y-value is its minimal number of samples in a node. The z-value is its minimal number of samples in a leaf. The colors indicate its variable selection scheme: red is the Gini index and blue is the information gain. The shapes indicate its value selection scheme: a diamond is linear searches, a square is medium values, and a circle is random values.

4.6.2 Convention v.s. Evolution

Conventional Random Forest has bagging and splitting selection mechanisms to reduce the correlation between trees. Evolutionary Random Forest is designed with a more complex architecture to reduce the correlation even lower and faster than the increment of the variance. The following comparison verifies whether Evolutionary Random Forest has lower correlation and lower variance than conventional Random Forest, which has only one set of parameters to grow all the trees.

In this experiment, the conventional model is set with naive parameters: 80 trials of the Gini index for variable section ($m_{try} = 80$), linear researches for value selection, and no restrictions of the number of samples in nodes and leaves ($n_{node} = n_{leave} = 1$). Figure 4.13 plots the results of the conventional model using 10 and 25 trees with dotted lines and the learning curves of Model A and Model M with solid lines. The conventional models have higher strength and lower correlation than the evolutionary models in most of the generations.

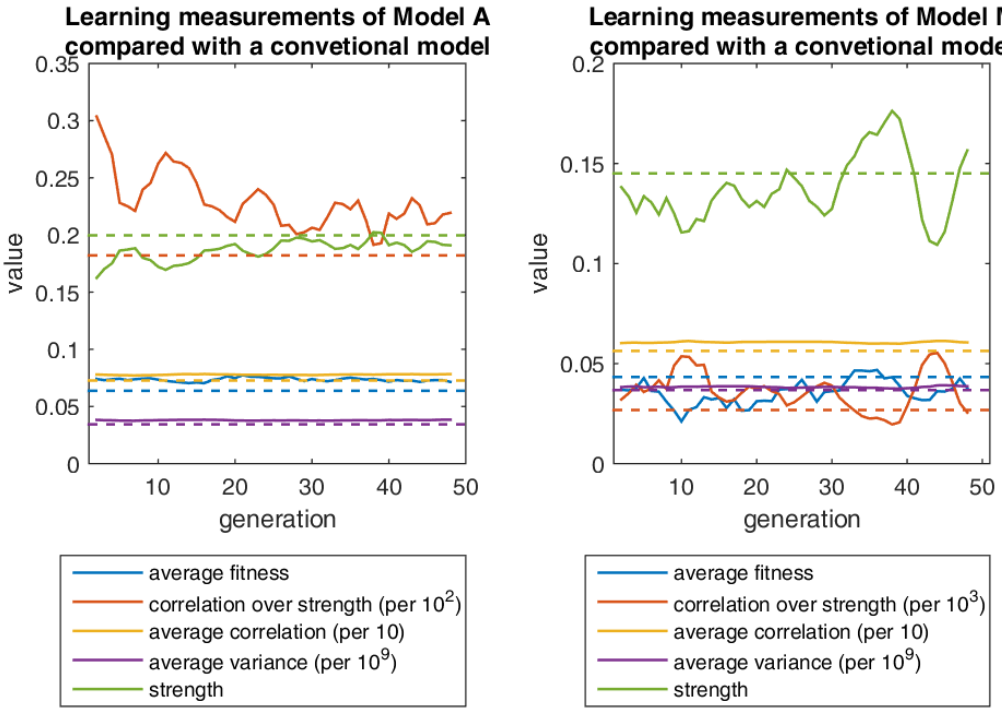

Figure 4.13: Plots of learning measurements of Model A (left) and Model M (right) in comparison with conventional models of the same sizes (25 and 10 trees, respectively.) The lines are the average fitness (blue,) correlation over strength (orange,) average correlation (yellow,) variance (purple,) and strength (green.) The dotted lines illustrate the outcome of the conventional models and the solid lines illustrate the evolutionary models. Note that the order of magnitude of the ratio changes between two plots for better illustration.

4.6.3 Alternative models

In Section 4.6.2, the evolutionary models show unsteady improvements over generations, which result in the performance worse than that of the conventional models. The following experiment tries to address the unsuccessful learning progress by decreasing the crossover ratio and mutation range factor even faster over generations. In addition, in the doubt that the restricted accessibility to variables stops the accuracy of individual trees from growing, the modified models allow all variables to be available for split selections in every tree. Table 4.9 lists the changes of the models, named Model M- and Model A- because they are simplified versions of Model M and Model A, respectively.

Figure 4.14 illustrates their learning curves compared with the results of the conventional models of the same sizes, which use 25 and 10 trees, respectively. Model M- makes significant progress in strength and fitness in the later generations with some fluctuations, and the results are better than those of the conventional model in some generations. Model A- shows no improvement from Model A, which is not better than the conventional model of the same size.

Figure 4.15 and 4.16 illustrate the parameters of trees in different generations, showing clear and different convergences of two models. The trees in Model A- converges towards the set of parameters ($m_{try} = 180, n_{node} = 80$) and various n_{leaf}, using either the Gini index with linear searches or random values, or the information gain with linear searches. Model M- has a convergence close to ($m_{try} = 10, n_{node} = 70, n_{leaf} = 7$), mostly using linear searches or random values for split selections.

models	Model M- and Model A-
crossover ratio decaying rate	5%
mutation range factor decaying rate	5%
Availability of variables for splitting	all

Table 4.9: Differences of Model M- and Model A- from Model M and Model A

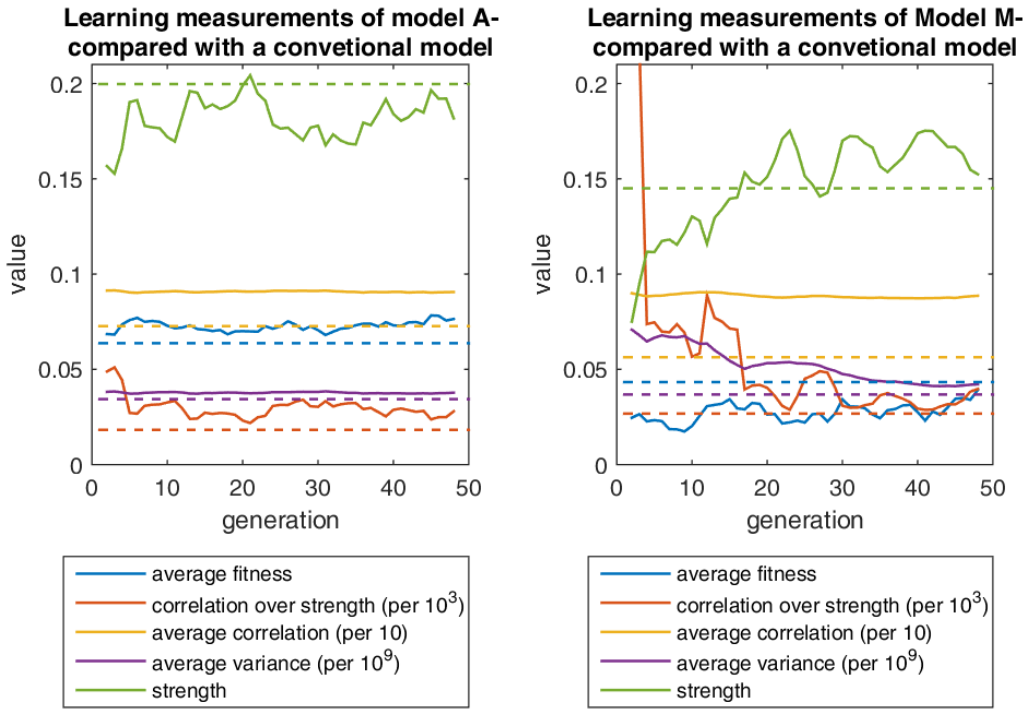

Figure 4.14: Plots of learning measurements of Model A- (left) and Model M- (right) in comparison with conventional models of the same size (25 and 10 trees, respectively.) The lines are the average fitness (blue,) correlation over strength (orange,) average correlation (yellow,) variance (purple,) and strength (green.) The dotted lines illustrate the outcome of the conventional models and the solid lines illustrate the evolutionary models.

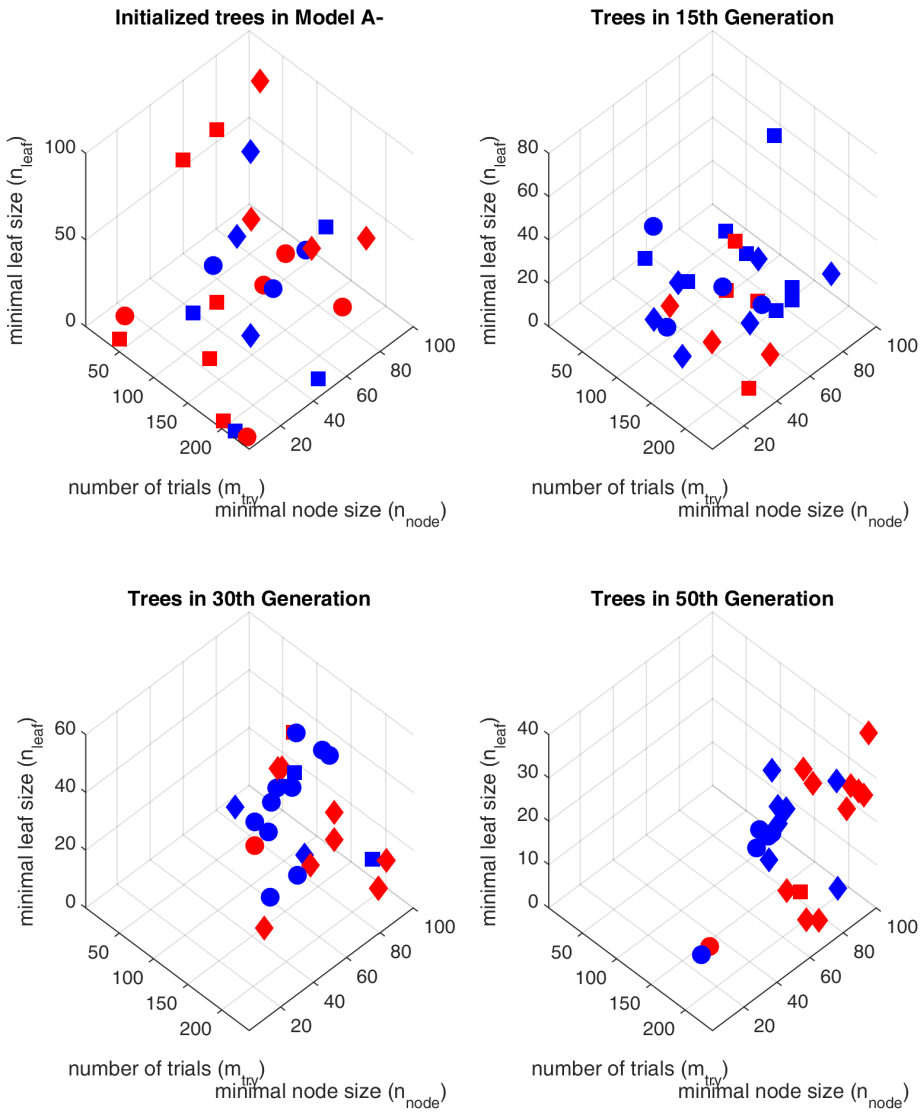

Figure 4.15: Parameters of trees in different generations grown by Model A-. Every icon represents a tree. The x-value is its number of trials for variance selection. The y-value is its minimal number of samples in a node. The z-value is its minimal number of samples in a leaf. The colors indicate its variable selection scheme: red is the Gini index and blue is the information gain. The shapes indicate its value selection scheme: a diamond is linear searches, a square is medium values, and a circle is random values.

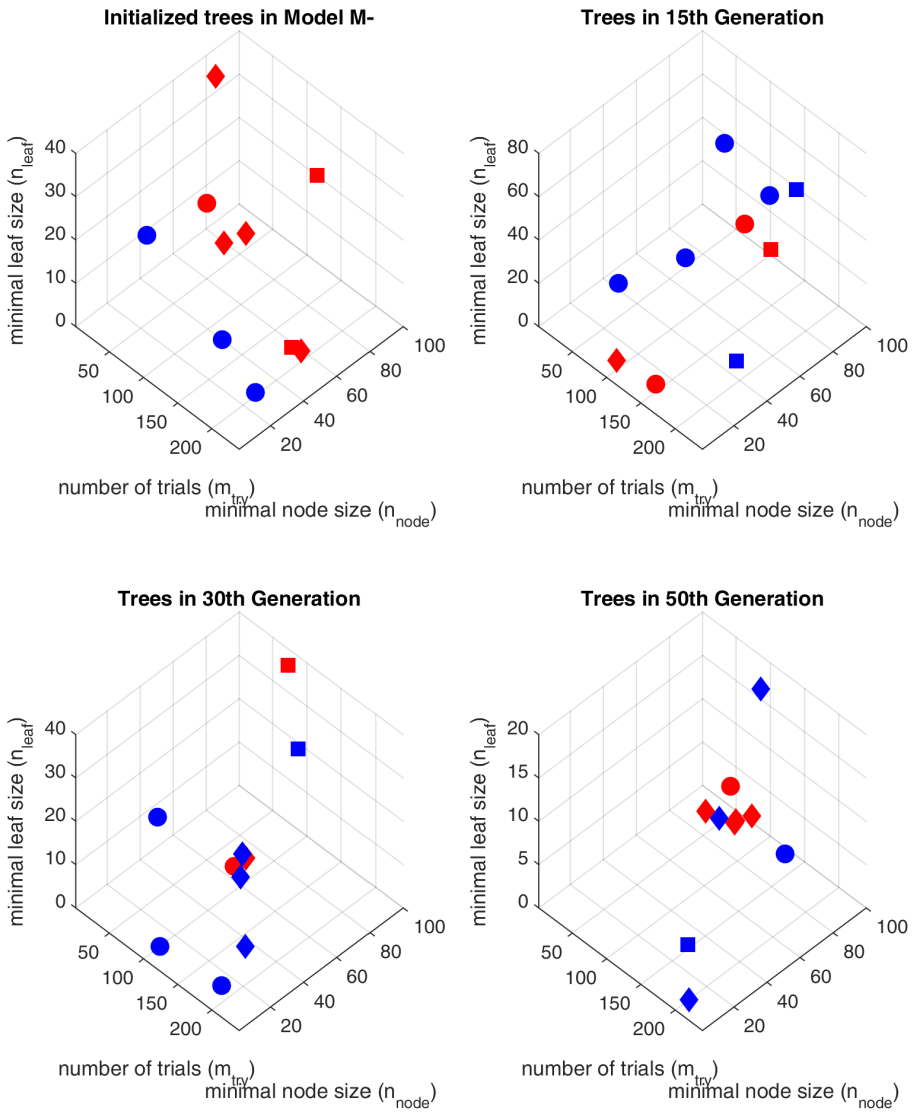

Figure 4.16: Parameters of trees in different generations grown by Model M-. Every icon represents a tree. The x-value is its number of trials for variance selection. The y-value is its minimal number of samples in a node. The z-value is its minimal number of samples in a leaf. The colors indicate its variable selection scheme: red is the Gini index and blue is the information gain. The shapes indicate its value selection scheme: a diamond is linear searches, a square is medium values, and a circle is random values.

4.6.4 Fine-tuned conventional models

Some populations in the evolutionary models shows better performance than a conventional model with naive parameters, which can be adjusted even better in practice. The convergences in the evolutionary models indicate the optimums in the solution space. The following experiment verifies if the found optimums can be used to tune a conventional model. The convergences of trees in the four final models indicate sets of parameters that produce the highest fitness, which are used as the candidates of optimal parameters of conventional models. Each evolutionary model provides a set of parameters, to which the majority of trees converges. The experiments builds conventional models based on the parameters and examines their performances, as listed in Table 4.10. The set of parameters derived from the convergence in Model A, which produces a significantly better result than all other candidates, is used to build a fine-tuned conventional model. Note that increasing the numbers of trees used by these models show now significant improvements in the experiment.

	conventional models			
derived from	Model A	Model M	Model A-	Model M-
number of trees	10	10	10	10
minimal node size	80	50	80	70
minimal leaf size	40	30	30	7
number of trials for variable selections	50	45	180	120
lowest correlation over strength	15.46	66.38	962 161	958 075

Table 4.10: Parameters used by the candidates of fine-tuned conventional models and their correlation-over-strength ratio in results

Figure 4.17 illustrates the learning curves of all final models in comparison with the fine-tuned conventional model. Model A and Model A- have better fitness but still a lower correlation-over-strength ratio than the fine-tuned conventional model. Model M, which is competitive with a naive conventional model, shows no better performance except for a lower average correlation. Also note that both Model A and Model M have a significantly lower average correlation than all other models. Model M- has a very high degree of variance that decreases over generations and increasing the strength in a good convergence, but the results are still not worse than those of the fine-tuned model, which achieve better strength and a lower strength-over-correlation ratio than all evolutionary models.

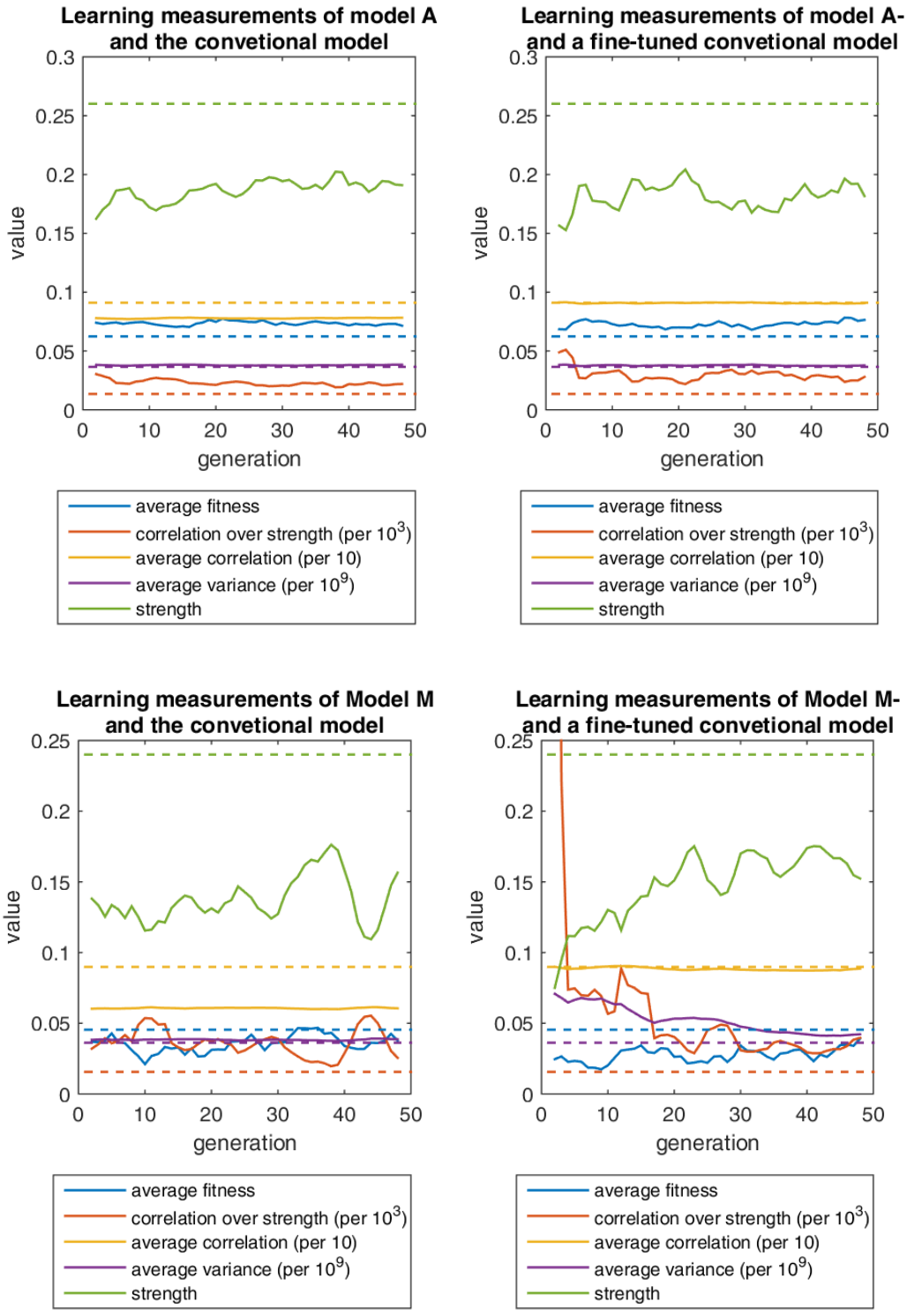

Figure 4.17: Plots of learning measurements of Model A (top left), Model A- (top right), Model M (bottom left,) and Model M- (bottom right) in comparison with a fine-tuned conventional model. The lines are the average fitness (blue,) correlation over strength (orange,) average correlation (yellow,) variance (purple,) and strength (green.) The dotted lines illustrate the outcome of the conventional models and the solid lines illustrate the evolutionary models.

Chapter 5

Discussion

Contents

This chapter discusses the choices made to arrive at the final models as used in Section 4.6 and compare their success with the conventional models. Section 5.1 presents arguments, under different headings, for making the choices that lead to the final models. In Section 5.2, reasons are considered why the evolutionary models do not improve the performance of the conventional models.

5.1 Design choices

5.1.1 Training set size and resampling

In this study, the full dataset has more than 6 trillion pixels. While a large amount of data is desired, the size of the training set is limited by the computing capacity. To avoid exceeding the memory limit, it is common to set a maximal height of trees. However, in this study, the data structure of nodes is a continuous vector. It is practical to limit the maximal number of nodes in a tree to limit the memory usage.

Neither oversampling nor undersampling is used in this study because the prior probabilities are incorporated in decision trees, and the bagging mechanism used in this study samples the classes roughly equally. Compared to downsampling, the advantage is that the bagging has access to all the samples in the training set. Also, the number of out-of-bag samples grows proportionally to the size of the training set, so there are also more out-of-bag samples for error, variance, and correlation calculation. The memory and time used by the calculation increase accordingly. A simple trick is to limit the number of out-of-bag samples used for the calculation, like the downsized correlation calculation discussed in Section 4.5.1. However, the technique should be used carefully, because an overly-downsized out-of-bag subset may not reflect well the distribution of samples, depending on the complexity of data.

5.1.2 Fitness functions

The design of the fitness function aims at minimizing the correlation-over-strength ratio through evolution. Since the evolution favor higher fitness, the ratio should be negatively correlated with the fitness, so as to be minimized. While Evolutionary Random Forest aims at reducing the ensemble variance, a low variance naturally causes a high correlation among trees. The goal of an ideal model is to decrease variance faster than the increment of the correlation. Raising the weight on the correlation cost in the fitness function (i.e., increasing λ) is tuning the growth of the correlation to overcome the unwanted aftereffect. Among the experimental models, the fitness function with $\lambda = 3$ produces learning curves that are most close to the desire.

Both the calculations of correlation costs, an average function and a maximum function, are effective in reducing average correlations of a population, but they bring on different performance when using different numbers of trees. The maximum function works well when there are not many trees. This is likely because there are not many optimums in the solution space in the targeted problem. When multiple trees move towards the same optimum, they obtain low fitness caused by a large correlation cost. Although the models give healthier trees no correlation costs, the intrinsic randomness of the model may alternatively favor different trees of similar parameters. Because of the high correlation costs on unfavored trees, the evolution keep most trees distant from the optimums so that no tree to minimize the correlation costs. These consequences are observed in the final populations as illustrated in Figure 5.1. The trees distribute more widely in the results of models using maximal correlation costs. The phenomenon is helpful when trees are fewer than optimums in the solution space, but harmful the other way round.

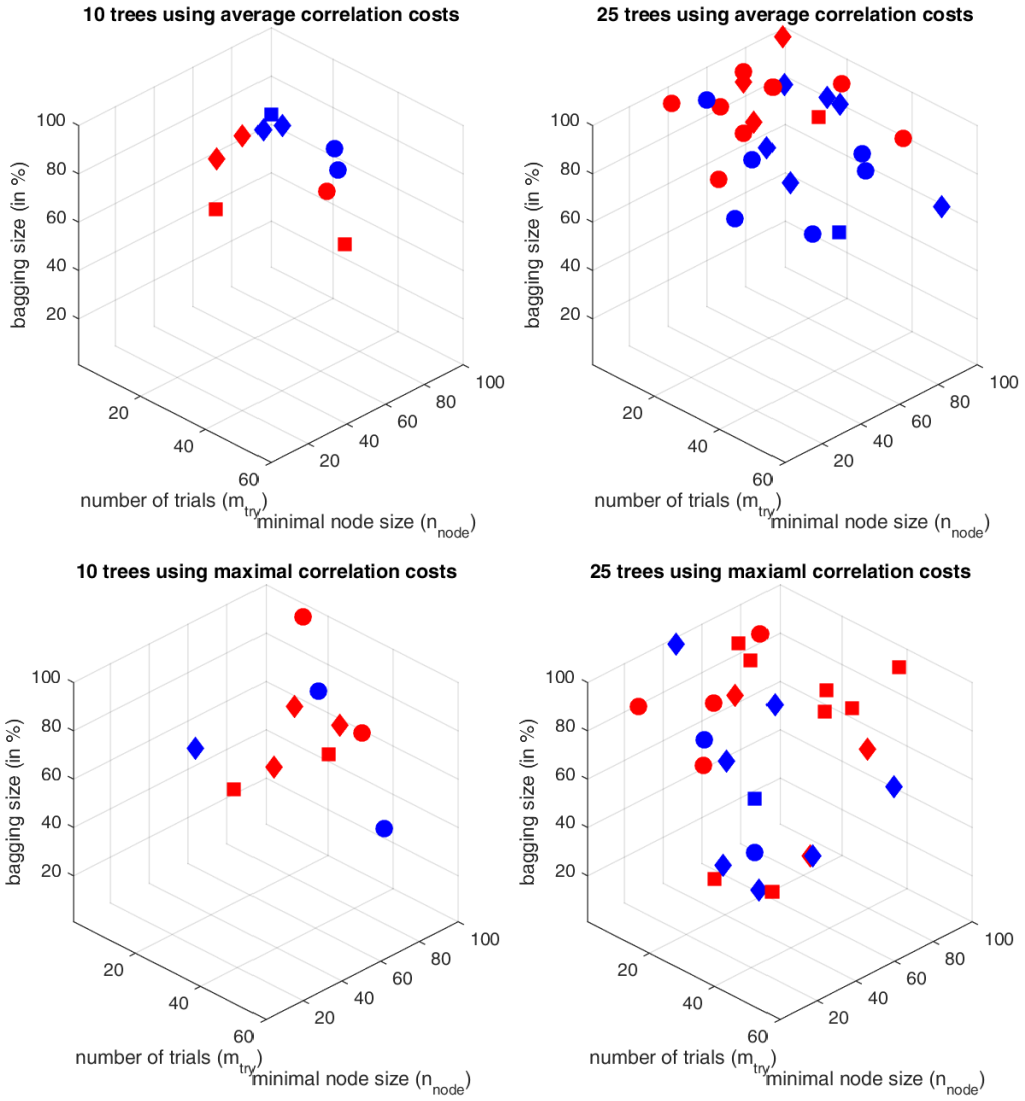

Figure 5.1: Parameters of trees with different correlation costs: average (top) and maximal (bottom). The left scatter plots are models of 10 trees and the right are of 25 trees. Every icon represents a tree. The x-value is its number of trials for variance selection. The y-value is its minimal number of samples in a node. The z-value its bagging size in percentage of all training samples. The colors indicate its variable selection scheme: red is the Gini index and blue is the information gain. The shapes indicate its value selection scheme: a diamond is a linear search, a square is medium values, and a circle is random values.

5.1.3 Convergence strategies

The conventional genetic algorithm, in which the probabilities of genetic changes decrease, does not work well with this model. As shown in section 4.5.3, the fitness values keep showing great variations through generations, and the correlation-over-strength ratio does not tend to decrease. This strategy does not work well because it encourages a high similarity of solutions that lead to a high correlation, which is harmful to the predictive performance.

In the adaptive genetic algorithms, the probabilities of changes being made on the chromosomes depend on the fitness, and the fitter half of the population are usually allowed to remain unchanged. This mechanism allows sufficient differences among trees to produce a low correlation-over-strength ratio, but the other half is unstable in reproduction, and their 'wrong votes' can cause a reduction of strength of the forest.

Finally, the adaptive genetic algorithm with a decaying crossover ratio and mutation range removes the disadvantages of the two former strategies and has a significantly better performance. Figure 5.2 presents the parameters of trees in the final generation. The z-axis is the bagging size of trees, which should be maximized regardless of other parameters. The conventional genetic algorithm shows a high convergence of the parameters. The adaptive algorithm has a wide distribution, in which even outlying values (low z-values) are possible. The adaptive algorithm with decaying also has a wide distribution, but far-away values are much less likely to occur.

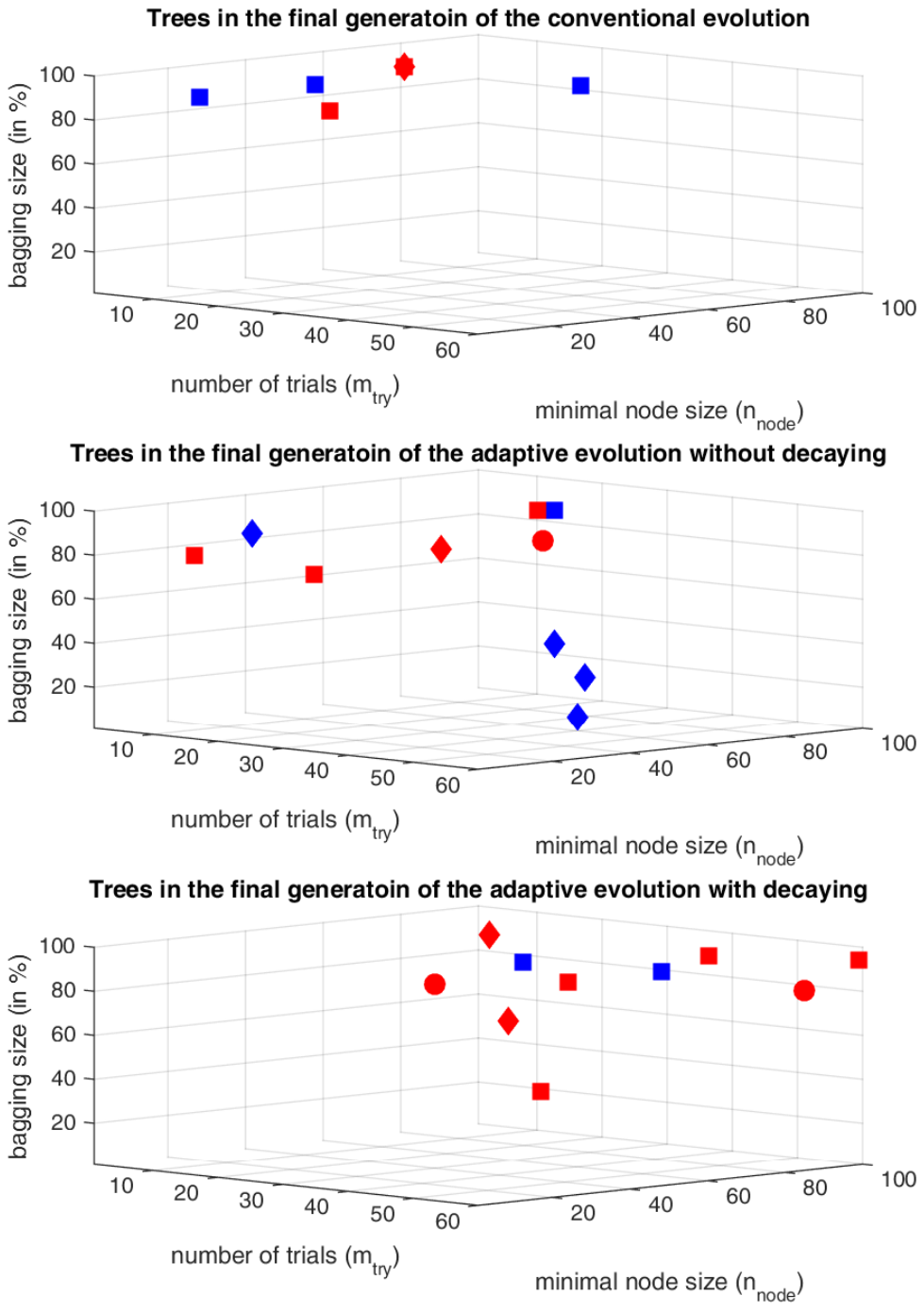

Figure 5.2: Parameters of trees grown by different evolutionary strategies: conventional (top,) adaptive (middle,) and adaptive with decaying (bottom). Every icon represents a tree. The x-value is its number of trials for variance selection. The y-value is its minimal number of samples in a node. The z-value is its bagging size in percentage of all training samples. The colors indicate its variable selection scheme: red is the Gini index and blue is the information gain. The shapes indicate its value selection scheme: a diamond is linear searches, a square is medium values, and a circle is random values.

5.2 Performance

Every evolutionary model fails to compete with the fine-tuned conventional model. Since the four final models, Model M , Model A, Model M-, and Model A-, show different learning progresses, separate discussions are made to find out the root cause of the failure. Figure 5.3 illustrates the trees in the final generation of the four models.

Model A produces a clear cluster of trees converging to a set of parameters in the final generation. When applying the parameters to a conventional model, they build a very powerful random forest model. Though Model A has most trees close to the optimum, the trees keep a distance in the solution space away from it because of the correlation costs. The learning progress of Model A shows steady improvements over generations, but most trees cannot get close to the optimums to produce the best ensemble predictive performance. The reduce of average correlation does not compensate the large decrease of strength.

Model M brings on two clusters of trees in the final generation. One set has small node sizes, and the other has a large number of trials for variable selections. Both sets of parameters lead to the same property: higher variance. Since Model M punishes the correlated trees by giving them a maximal correlation cost of all associated trees, the high costs encourage trees to make variant predictions, which make them less likely to vote the same as other trees. The trees become virtually fully-grown decision trees, which are known to have high variance, when using large numbers of trials or small node sizes. The produced population of Model M has no trees in the optimum that Model A finds, as illustrated in Figure 5.3. The phenomenon gives Model M a significantly lower average correlation, but no improvement in strength.

Without the tree-level restricted accessibility to variables to decorrelate trees, Model A- has the same trend that Model M has, that trees tend to use parameters that result in a high variance. The difference is that the average correlation costs used by Model A- allow trees to be close in the solution space without a large cost, which separate trees into two clusters in Model M. Model A- has a large cluster at the sets of parameters that have very large numbers of trials for variable selections, which are nearly invalid for conventional Random Forest.

Finally, Model M-has a good convergence of learning measurements in the experiments but cannot improve further after a number of generations. The final generation of Model M- in Figure 5.3 shows a cluster of sets of parameters of large node sizes, small leaf sizes, and a rather small numbers of trials for variable selections. While the parameters are close to those used by the fine-tuned model, the variable selection schemes in Model M- are mostly the information gain with linear searches, in contrast to the Gini index with random values used by the fine-tuned model. The variable selection schemes play a significant role here in the performance differences, but it is not clear to the author how Model M- arrived at the inferior configuration.

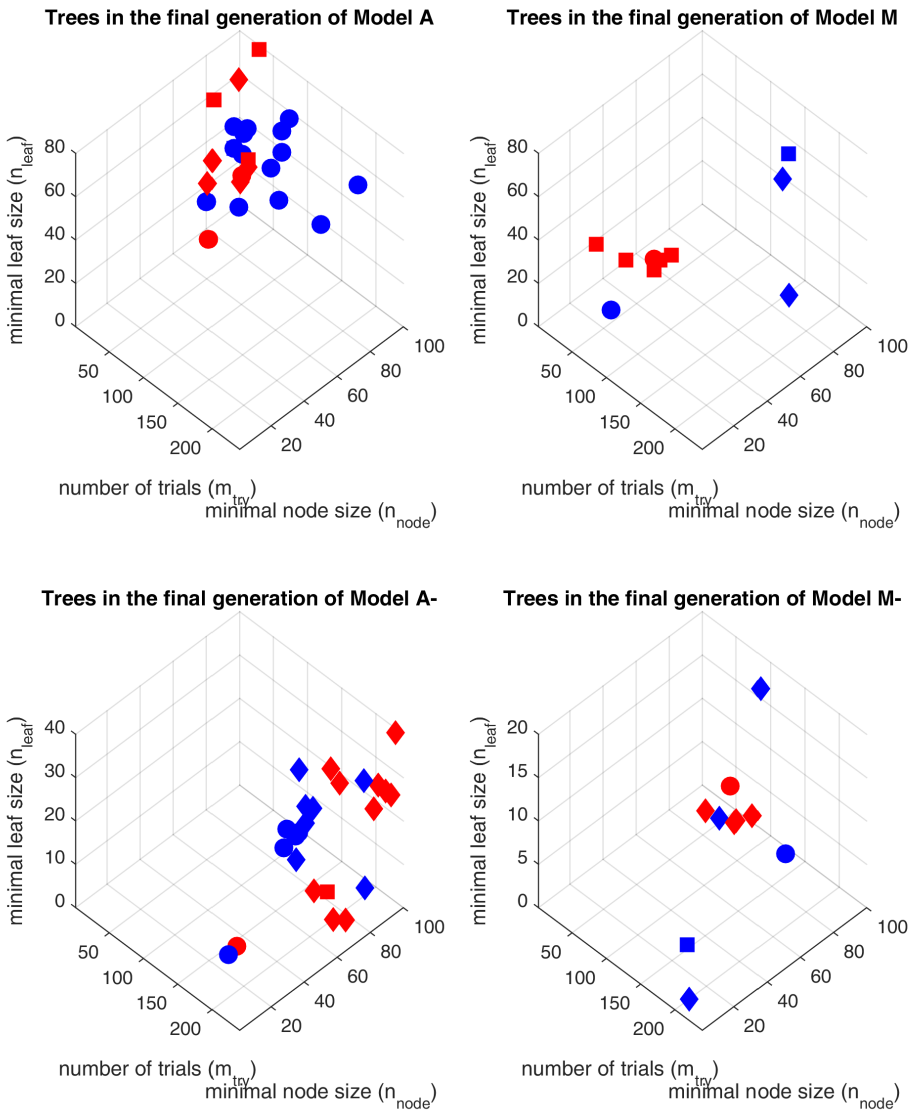

Figure 5.3: Parameters of trees with different models: Model A (top left), Model M (top right), Model A- (bottom left), and Model M- (bottom right.) The x-value is trees' number of trials for variance selection. The y-value is their minimal number of samples in a node. The z-value their minimal number of samples in a leaf. The colors indicate its variable selection scheme: red is the Gini index and blue is the information gain. The shapes indicate its value selection scheme: a diamond is a linear search, a square is medium values, and a circle is random values.

To summarize the convergences and their consequences in the models, two pitfalls hinder the performance of evolutionary models from surpassing that of fine-tuned conventional models, as illustrated in Figure 5.4. The accuracy of trees is optimal when they have reasonable variance, and the predictive performance is optimal when trees locate at different optimums in the solution space, as the illustrated ideal model.

The first pitfall occurs to forest models of high variability (Model A and Modal M,) which provide rich options for parameters and allow trees to decrease the correlations by increasing only some variance. Because of correlation costs, the majority of trees keep a distance from the optimums, at which healthier tree locate. The wide distribution of trees decreases much the strength of a forest, even though it contributes to the decrease of average correlation. Trees could be forced to find other optimums by increasing the correlation costs, but there would still be many trees locate in the solution space near the optimums because the trees can fit in the space by using varieties of parameters that produce low correlation costs (and low accuracy.)

The second pitfall happens to forest models of low variability (Model A- and Model M-,) which have limited options for their parameters. Trees in these models incline to gain high variance to decorrelate each other. The part of the solution space that produces higher variance can have more trees without causing high correlation costs because high variance naturally decreases the correlations. The attraction towards high variance hinders trees from going to the space that produces low variance but high accuracy. The trees in these models scatter in the part of the solution space in which correlation costs are minimized and low predictive accuracy is produced.

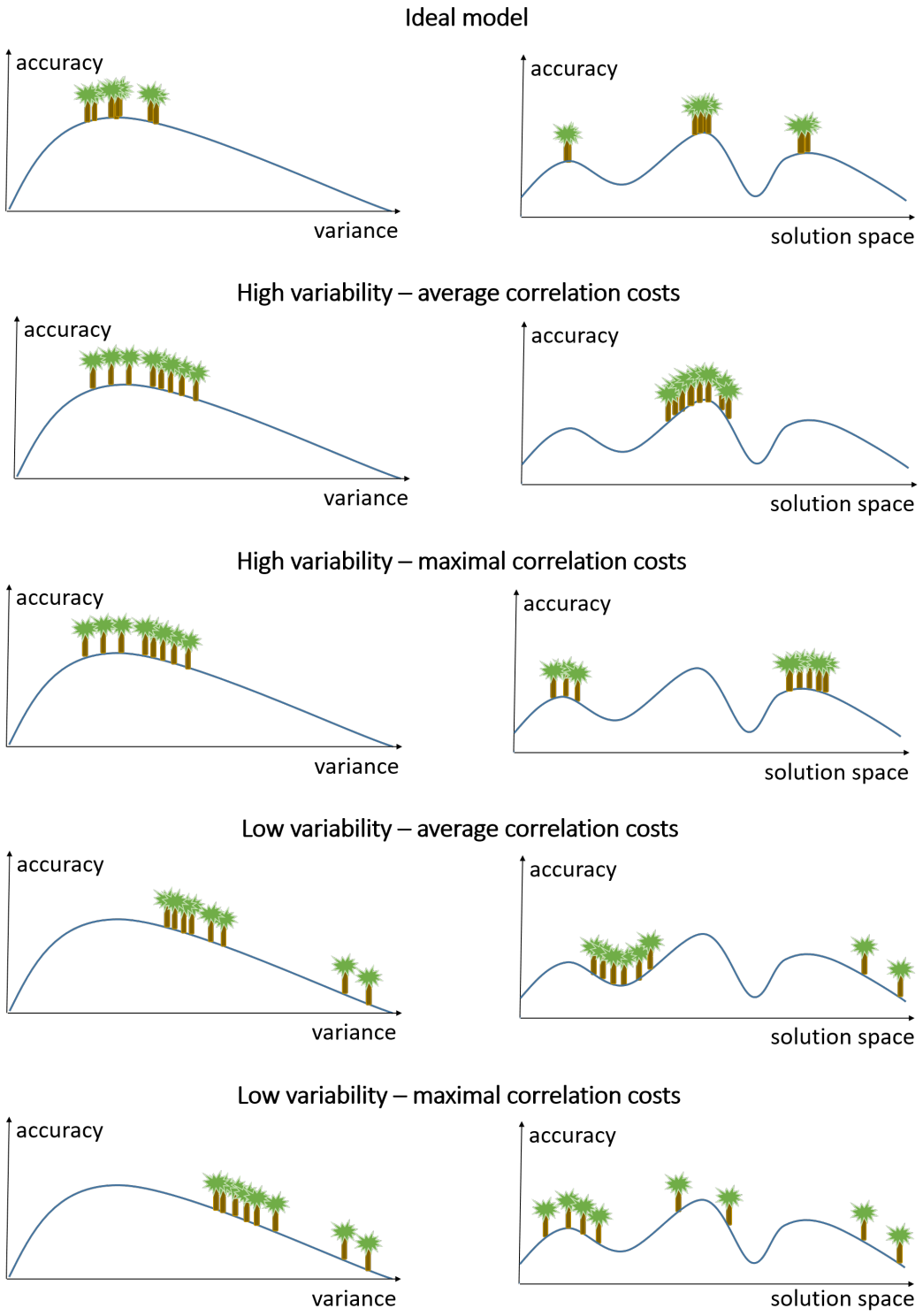

Figure 5.4: Illustrations of an ideal random forest model and models that fail to evolve towards it. The left side illustrate how trees distribute in terms of variance and their corresponding accuracy. The right side illustrate how trees distribute in the solution space, in which trees have higher variance and locate farther away result in lower correlation costs, and their corresponding accuracy. From top to bottom are different evolutionary random forest models.

Chapter 6
Conclusion

According to previous research, the generalization error of Random Forest is bounded by the strength and correlation of an ensemble classifier. This study shows further that a careful design of the evaluation function enables predicting the upper bound by evaluating the performance of individual classifiers. With the calculation of a tree's performance based on its correctness, variance, and correlation cost, an ensemble classifier can be improved by selecting and creating fitter trees for the population. This finding opens the opportunities for other approaches to improve Random Forest by manipulating the individual learners and comparing their performance.

Unlike the usual goal of genetic algorithms, which is to find an optimal individual among the populations, the optimization solved in this study is to compose an optimal population. Neither the conventional genetic algorithm nor the adaptive genetic algorithm performs the task well in this study. The introduced model, an adaptive algorithm with decaying genetic changes, achieves the best results in this particular optimization problem. The finding suggests that this decaying adaptive model can be used for optimization problems that require maintaining a diversity of the population.

The biggest drawback of Evolutionary Random Forest is tremendous amounts of training time. The evaluation of correlation and the reproduction of generations cause significantly more time than conventional Random Forest. To overcome the disadvantage, this study shows that the downsized correlation calculation can decrease the evaluation time without reducing the model's accuracy. Also, the experimental models show that there are optimal numbers of trees and numbers of features to achieve good performance with Evolutionary Random Forest. This finding encourages tuning these hyperparameters thoughtfully without wasting training time and computing resources. Finally, the models using maximal correlation costs require much fewer trees to obtain their optimal performance. The decrement of trees speeds up both the evaluation and evolution.

On the contrary to expectations, Evolutionary Random Forest cannot outperform fined-tuned conventional Random Forest in the experiments. The production of Evolutionary Random Forest fails in a number of scenarios. Highly-variable forest models enable trees to find optimums in the solution space, but the trees distribute widely around the optimums to decrease correlations and thus have poorer performance. Lowly-variable forest models push trees towards solutions of high variance and low accuracy, and their predictive performance cannot surpass conventional

models despite the positive progress of evolution.

The limited computing power used in this study confines the experiments to a tiny portion of training samples, which limit the predictive performance of the final models. A larger training set may allow Evolutionary Random Forest to find more optimums in the solution space and improve its predictive performance more than conventional Random Forest. The idea requires further studies for verification.

In the author's expectation, the hyperparameters in Evolutionary Random Forest could also be adjusted through evolution according to the performance of populations. By comparing populations in different generations and tuning the hyperparameters towards those having better performance, the model would arrive at one population that uses an optimal set of hyperparameters. The strategy could empower the model to be automatically optimized without tedious experiments. This study leaves this function unrealized and suggests further research towards this direction.

Bibliography

[Bader-El-Den 2012] Mohamed Bader-El-Den and Mohamed Gaber. *Garf: towards self-optimised random forests.* In Neural information processing, pages 506–515. Springer, 2012. 3, 11

[Belgiu 2016] Mariana Belgiu and Lucian Drăguţ. *Random forest in remote sensing: A review of applications and future directions.* ISPRS Journal of Photogrammetry and Remote Sensing, vol. 114, pages 24–31, 2016. 1

[Bernard 2010] Simon Bernard, Laurent Heutte and Sébastien Adam. *A study of strength and correlation in random forests.* In International Conference on Intelligent Computing, pages 186–191. Springer, 2010. 4, 10

[Blickle 1995] Tobias Blickle and Lothar Thiele. *A comparison of selection schemes used in genetic algorithms*, 1995. 11

[Bosch 2007] Anna Bosch, Andrew Zisserman and Xavier Munoz. *Image classification using random forests and ferns.* In Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, pages 1–8. IEEE, 2007. 1

[Breiman 1996a] Leo Breiman. *Bagging predictors.* Machine learning, vol. 24, no. 2, pages 123–140, 1996. 8, 9

[Breiman 1996b] Leo Breiman. *Bias, variance, and arcing classifiers.* 1996. 2

[Breiman 2001] Leo Breiman. *Random forests.* Machine learning, vol. 45, no. 1, pages 5–32, 2001. 1, 2, 4, 9, 10

[Chen 2004] Chao Chen, Andy Liaw and Leo Breiman. *Using random forest to learn imbalanced data.* University of California, Berkeley, vol. 110, 2004. 9

[Ciss 2015] Saïp Ciss. *Generalization Error and Out-of-bag Bounds in Random (Uniform) Forests.* 2015. 4, 10, 16

[Cordts 2016] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth and Bernt Schiele. *The cityscapes dataset for semantic urban scene understanding.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3213–3223, 2016. 2, 18

[Cordts 2017] Marius Cordts. *Understanding Cityscapes: Efficient Urban Semantic Scene Understanding.* PhD thesis, Technische Universität, 2017. 1

[Dietterich 2000] Thomas G Dietterich*et al.* *Ensemble methods in machine learning.* Multiple classifier systems, vol. 1857, pages 1–15, 2000. 8

[Drummond 2003] Chris Drummond, Robert C Holte*et al.* *C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling.* In Workshop on learning from imbalanced datasets II, volume 11. Citeseer Washington DC, 2003. 9

[Efron 1997] Bradley Efron and Robert Tibshirani. *Improvements on cross-validation: the 632+ bootstrap method.* Journal of the American Statistical Association, vol. 92, no. 438, pages 548–560, 1997. 22

[Fawagreh 2014] Khaled Fawagreh, Mohamed Medhat Gaber and Eyad Elyan. *Random forests: from early developments to recent advancements.* Systems Science & Control Engineering: An Open Access Journal, vol. 2, no. 1, pages 602–609, 2014. 2

[Fu 2001] Zhiwei Fu and Fannie Mae. *A computational study of using genetic algorithms to develop intelligent decision trees.* In Evolutionary Computation, 2001. Proceedings of the 2001 Congress on, volume 2, pages 1382–1387. IEEE, 2001. 11

[Fu 2003] Zhiwei Fu, Bruce L Golden, Shreevardhan Lele, S Raghavan and Edward A Wasil. *A genetic algorithm-based approach for building accurate decision trees.* INFORMS Journal on Computing, vol. 15, no. 1, pages 3–22, 2003. 11

[Gashler 2008] Mike Gashler, Christophe Giraud-Carrier and Tony Martinez. *Decision tree ensemble: Small heterogeneous is better than large homogeneous.* In Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on, pages 900–905. IEEE, 2008. 2, 9

[Gen 2000] Mitsuo Gen and Runwei Cheng. Genetic algorithms and engineering optimization, volume 7. John Wiley & Sons, 2000. 2, 4

[Gislason 2006] Pall Oskar Gislason, Jon Atli Benediktsson and Johannes R Sveinsson. *Random forests for land cover classification.* Pattern Recognition Letters, vol. 27, no. 4, pages 294–300, 2006. 1

[Goldberg 2006] David E Goldberg. Genetic algorithms. Pearson Education India, 2006. 2, 4

[Liaw 2002] Andy Liaw, Matthew Wiener*et al.* *Classification and regression by randomforest.* R news, vol. 2, no. 3, pages 18–22, 2002. 1

[Loh 2011] Wei-Yin Loh. *Classification and regression trees.* Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 1, no. 1, pages 14–23, 2011. 13

[Louppe 2014] Gilles Louppe. *Understanding random forests: From theory to practice.* arXiv preprint arXiv:1407.7502, 2014. 2, 8

[Miller 1995] Brad L Miller, David E Goldberg*et al. Genetic algorithms, tournament selection, and the effects of noise.* Complex systems, vol. 9, no. 3, pages 193–212, 1995. 12

[Opitz 1999] David W Opitz and Richard Maclin. *Popular ensemble methods: An empirical study.* J. Artif. Intell. Res.(JAIR), vol. 11, pages 169–198, 1999. 1, 8

[Quinlan 2014] J Ross Quinlan. C4. 5: programs for machine learning. Elsevier, 2014. 14

[Raczko 2017] Edwin Raczko and Bogdan Zagajewski. *Comparison of support vector machine, random forest and neural network classifiers for tree species classification on airborne hyperspectral APEX images.* European Journal of Remote Sensing, vol. 50, no. 1, pages 144–154, 2017. 1

[Robnik-Šikonja 2004] Marko Robnik-Šikonja. *Improving random forests.* In European conference on machine learning, pages 359–370. Springer, 2004. 2, 4, 9

[Spitta 2015] D. Spitta. *Genetic Randomized Trees for Object Classification.* Berlin Institute of Technology, 2015. 3

[Srinivas 1994] Mandavilli Srinivas and Lalit M Patnaik. *Adaptive probabilities of crossover and mutation in genetic algorithms.* IEEE Transactions on Systems, Man, and Cybernetics, vol. 24, no. 4, pages 656–667, 1994. 4, 12, 15

[Stückler 2012] Jörg Stückler, Nenad Biresev and Sven Behnke. *Semantic mapping using object-class segmentation of RGB-D images.* In Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, pages 3005–3010. IEEE, 2012. 1, 8, 19

[Stückler 2015] Jörg Stückler, Benedikt Waldvogel, Hannes Schulz and Sven Behnke. *Dense real-time mapping of object-class semantics from RGB-D video.* Journal of Real-Time Image Processing, vol. 10, no. 4, pages 599–609, 2015. 1, 8

[Thoma 2016] Martin Thoma. *A survey of semantic segmentation.* arXiv preprint arXiv:1602.06541, 2016. 5

[Uhrig 2016] Jonas Uhrig, Marius Cordts, Uwe Franke and Thomas Brox. *Pixel-level encoding and depth layering for instance-level semantic labeling.* In German Conference on Pattern Recognition, pages 14–25. Springer, 2016. 8