# Automated machine learning for neural network verification

König, H.M.T.

# Chapter 6

# Adversarially Robust Model Selection via Racing

In the previous chapters, we have introduced several meta-algorithmic approaches to formally verify the robustness of neural network models against perturbations in their inputs, such as the ones that occur in adversarial attacks. Nonetheless, this particular verification task remains computationally challenging.

In addition to other performance metrics of a neural network model, such as accuracy, one can compute *robust accuracy* by counting the fraction of inputs that are provably robust with regard to the given property. However, this adds significant overhead to the evaluation procedure, due to the high computational demands incurred by most formal verification algorithms, as explained earlier (see, *e.g.*, Chapter 2. This overhead grows substantially if multiple models are considered and compared against each other in (robust) performance; this challenge is not only faced by practitioners (*e.g.*, during model evaluation) but also encountered during Neural Architecture Search (NAS), where the goal is to select a suitable model from a large search space (see, *e.g.*, Elsken et al. [28]). In this context, adding robust accuracy as a selection criterion would hardly be feasible due to the large computational costs.

In this chapter, we seek to improve the efficiency of local robustness verification from a previously unexplored, meta-algorithmic point of view. Specifically, we propose a method to efficiently evaluate and compare the robustness of different neural network models (or variations of the same model) against adversarial attacks. Moreover, we consider the problem of selecting the most robust model, *i.e.*, the model with the

highest certified robust accuracy, from a given set of trained neural networks, whilst making the most efficient use of the computational budget.

In a nutshell, our proposed method employs a racing algorithm, in which a given set of neural network models are subjected to local robustness verification with respect to adversarial attacks. After each input iteration, the performance of each network (in terms of robust accuracy) is measured, and the verification procedure stops for a given network as soon as its robust accuracy is lower than the robust accuracy obtained by its competitors. Racing approaches are well studied and have already been successfully employed in other, resource-intense domains, such as hyperparameter optimisation [45, 7].

Complementary to the racing approach, we propose a novel sampling strategy based on the likelihood of a given input instance being adversarially robust. Essentially, this strategy prioritises input instances during the verification procedure that are most likely to expose vulnerabilities of the neural network model and, therefore, provide valuable insights into its robustness after fewer input iterations of the verification procedure and, hence, at a lower computational cost. At the same time, it reduces the risk of selecting sub-optimal models, which might show higher robust accuracy than other candidate models after verifying some randomly sampled input instances but might perform worse overall. In fact, when using random sampling, the only way to mitigate this risk would be to increase the number of input iterations – with the associated costs involved.

To enable the proposed sampling strategy, we must define or estimate the likelihood of a network input being adversarially robust. In our case, this involves estimating the proximity of a network input to the decision boundaries of the model, captured by means of $\Delta$-values, which will be explained in the following. Using this strategy, we can bias the sampling towards inputs for which adversarial attacks are most likely to occur. Although the relation between adversarial examples and the decision boundary of a neural network has been extensively studied [23, 120, 20, 38, 74], we are not aware of any existing work leveraging these insights in the context of local robustness verification procedures. In summary, the main contributions of this chapter are as follows:

- We propose an efficient model selection method based on a novel heuristic that quantifies the likelihood of a network being adversarially robust with respect to a given input;

- we introduce $\Delta$-values, which serve as a proxy for the distance of an input instance to the decision boundaries of a neural network model;

- we provide statistical evidence demonstrating significant differences in the empirical cumulative distribution of $\Delta$-values between robust and non-robust instances;
- we evaluate our method on two diverse sets of neural networks trained on the MNIST and CIFAR-10 datasets, achieving a 108-fold reduction in cumulative running time for MNIST networks and a 42-fold reduction for CIFAR-10 networks.

## 6.1 Adversarially Robust Model Selection

Given a set of neural network models $\mathcal{N} = \{N_1, N_2, \ldots, N_m\}$ and a set of input instances $\mathcal{X}$, the objective is to identify the model $N^* \in \mathcal{N}$ that maximises robust accuracy with respect to adversarial attacks. Notice that robust accuracy is computed by verifying all instances $x \in \mathcal{X}$, measuring the fraction of cases in which the model correctly classifies an instance and adversarial input perturbations do not change the original output produced by the model.

To address this problem, we propose a method with two main components: a racing approach and a sampling strategy based on a sorting mechanism for the input instances on which the network is verified. We considered two variants of the racing approach. The first one is a naïve racing approach in which the best-performing candidate models are selected at every input iteration, whereas the second one represents an adaptation of the *F-Race* algorithm [7], which gathers statistical evidence against some candidate models before they are discarded. Both variants of the racing approach as well as our proposed sorting mechanism will be further explained in the following.

### 6.1.1 Naïve Racing Approach

Generally, the idea of a racing approach is to evaluate a finite set of candidate models while allocating the computational resources among them in a systematic way (see, *e.g.*, [79]). To do so, the racing approach verifies step-by-step each candidate model in the given set, where in this context, a step corresponds to an input instance on which the neural network models are verified. At each step, all the remaining candidate models are verified, possibly in parallel, and candidate models are discarded once they are outperformed by others, *i.e.*, once one or more networks have obtained a higher robust accuracy.

An overview of this approach, which we refer to as the naïve racing approach for the remainder of this chapter, can be found in Algorithm 2. After each iteration over the input instances, it identifies the model with the highest robust accuracy (determined

**Algorithm 2** Racing approach for robust model selection

**Require:** Trained neural network models $\mathcal{N} = \{N_1, N_2, \ldots, N_m\}$; Network input instances $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$; Verification algorithm $\textsc{Verify}(N_i, x_j)$ that returns *sat*, *unsat* or *unsolved*;

**Ensure:** Model with highest robust accuracy: $N_{\text{selected}}$

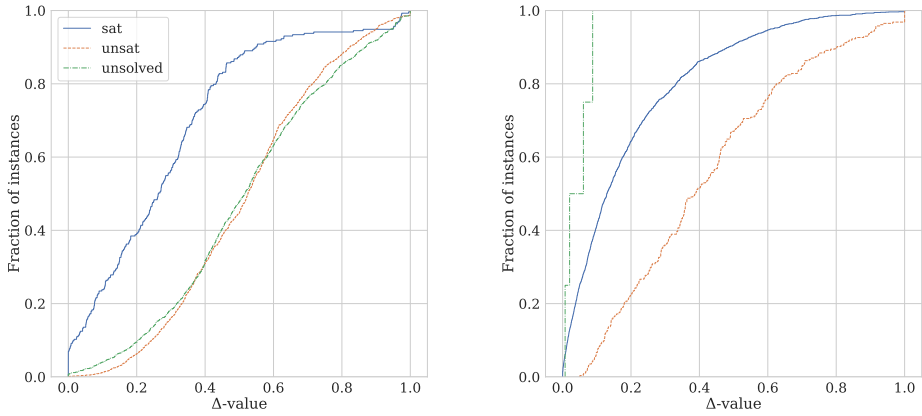
1: $\mathcal{C} \leftarrow \mathcal{N}$
2: $u_i \leftarrow 0$ with $i = 1, 2, \ldots, |N|$
3: **for all** $x_j \in \mathcal{X}$ **do**
4:  **for all** $N_i \in \mathcal{C}$ **do**
5:  **if** $\textsc{Verify}(N_i, x_j)$ is *unsat* **then**
6:  $u_i \leftarrow u_i + 1$
7:  **end if**
8:  **end for**
9:  $\mathcal{C} \leftarrow \{ N_i \mid i \in \arg\max_i \{u_i\} \}$
10: **end for**
11: Randomly select one element $N_{\text{selected}}$ from set $\mathcal{C}$
12: **Return** $N_{\text{selected}}$

based on $u_i$ which represents the number of unsat instances for network $N_i$) and updates the set of candidate models $\mathcal{C}$ accordingly. Notice that the selection criterion on line 9 can by virtue of the arg max operator return a set of multiple networks. Moreover, $u_i$ increases whenever a network $N_i$ is found to be robust, *i.e.*, *unsat*, w.r.t. to a given input. On the other hand, an instance that is misclassified by the model would be considered as *sat*. The algorithm stops once all input instances have been processed, and the final output is the model with the highest determined robust accuracy.

### 6.1.2 F-Race

An important aspect of the model selection problem outlined above is that it can be viewed as a stochastic problem. In fact, although the process of formally verifying the behaviour of a neural network model with respect to certain input instances is deterministic (*i.e.*, multiple runs on the same input instance will always lead to the same result), its outcome depends on the particular instance to which it is applied. Concurrently, the specific instance being verified can be regarded as having been sampled from an underlying probability distribution, which may be unknown. For the naïve racing approach, this could lead to models being prematurely discarded after a few input iterations, even if that model would achieve the highest robust accuracy overall, *i.e.*, if it was verified with respect to all available input instances.

To address this stochasticity, the authors of [7] proposed *F-Race*, a widely known,




**(a)** MNIST (VeriNet, $\epsilon = 0.04$) **(b)** CIFAR ($\alpha\beta$-CROWN, $\epsilon = 0.008$)

**Figure 6.1:** Empirical cumulative probability distribution of normalised $\Delta$-values for *sat*, *unsat* and unsolved instances for the considered MNIST and CIFAR networks, respectively. Notably, the plot shows a statistically significant difference between the empirical distribution functions of $\Delta$-values for *sat* and *unsat* instances. Specifically, for both MNIST and CIFAR networks, *sat* instances generally have smaller $\Delta$-values than *unsat* instances. Statistical significance is determined by means of a Kolmogorov–Smirnov test with a significance threshold of 0.05.

state-of-the-art racing algorithm. F-Race can be considered an extension of the naïve racing approach, where the naïve selection criterion (line 9 in Algorithm 2) is replaced with a statistical test. Concretely, after each iteration over the input instances, F-Race performs a statistical test, typically, the non-parametric Friedman test, to determine if there are significant differences in the number of *unsat* instances per neural network model. If the null hypothesis is rejected or, in other words, significant differences exist, F-Race applies post-tests to identify the models which are performing statistically significantly worse than the best, and updates $\mathcal{C}$ accordingly. The algorithm stops when all input instances have been processed, and the final output is the model with the highest robust accuracy.

Since we are interested in the fraction of instances that are *unsat*, we used Cochran's Q test to determine if there are significant differences among the *unsat* counts for each of the networks. Notice that the Cochran's Q test is identical to the Friedman test but applicable when the responses are binary. When only two candidate networks remain, we used the McNemar test (without continuity correction), which can be seen as a

special case of Cochran's Q test [102]. Any significant Cochran's Q (or McNemar) statistic is followed by Dunn's post-hoc test with a significance threshold of p= 0.05, and networks are selected if they have a significantly higher certified robust accuracy than their competitors.

### 6.1.3 Sorting Mechanism

In addition, we propose a sampling strategy based on a mechanism that sorts the considered input instances according to their likelihood of being adversarially robust. The key idea behind this mechanism is that by exposing a neural network model to inputs that are least likely to be adversarially robust, we can more quickly gather insights into its vulnerability or, similarly, its robustness. In other words, if we initially verify a neural network model on its most "challenging" input instances, *i.e.*, instances on which it is most likely not robust, but obtain robustness guarantees for these instances, we can at least heuristically assume the model to also be robust with respect to the remaining instances.

To enable this sorting mechanism, we must define or estimate the likelihood of a neural network model input being adversarially robust. In our case, this involves estimating their distance from the decision boundaries of the model, captured by means of network outputs. Intuitively, if an input lies very close to the boundary between two classes, it can be assumed that small perturbations, such as those applied to adversarial examples, have a higher chance to change the prediction made by the model.

In this study, we estimate the distance to an adjacent class boundary as the difference between the neural network output corresponding to the most likely class and that corresponding to the second-most likely class, and we refer to this difference as $\Delta$. Formally, we define $\Delta := \max(\{y_1, y_2, \ldots, y_n\}) - \max(\{y_1, y_2, \ldots, y_n\} \setminus \max(\{y_1, y_2, \ldots, y_n\}))$, where $y_n$ refers to the network output for a given class $n$. Based on the resulting $\Delta$-values, we can, for each neural network model individually, sort the input instances in an non-decreasing order, where the smaller the value of $\Delta$, the closer we assume an instance to lie to an adjacent class boundary.

## 6.2 Setup of Experiments

We compiled two sets of neural network models: one set consisting of 31 neural networks trained on the MNIST dataset and one set containing 27 neural networks trained on the CIFAR-10 dataset. All networks were taken from the repository of the ERAN

verification system [85, 96, 99, 97, 98] and greatly vary in terms of architecture, training method as well as robust accuracy. Details of the considered networks can be found in the supplementary material. We verified each network for local robustness with respect to the first 100 instances in the test set of the MNIST and CIFAR-10 datasets, respectively.

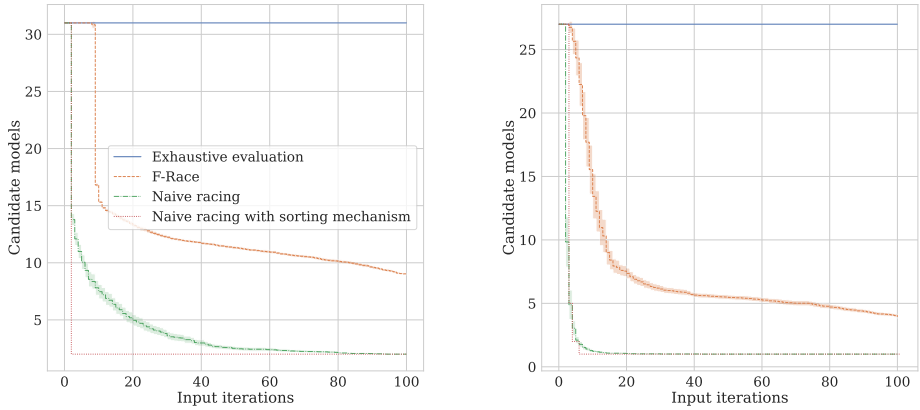
To verify the MNIST networks, we used the state-of-the-art complete CPU-based verification algorithm VeriNet [40] with a perturbation radius of $\epsilon = 0.04$, which lies well within the range of commonly chosen values for $\epsilon$ when verifying networks trained on MNIST [114, 8, 109]. Verification queries ran with a time budget of 3600 seconds on a cluster of machines equipped with Intel Xeon E5-2683 CPUs with 32 cores, 40 MB cache size and 94 GB RAM, running CentOS Linux 7.

To verify the more challenging CIFAR networks, we used $\alpha\beta$-CROWN, a state-of-the-art complete GPU-accelerated verification method [111]. For these networks, we verified local robustness with $\epsilon = 0.008$, a value in line with commonly chosen values of $\epsilon$ for networks trained on CIFAR (see, *e.g.*, [87]). Again, all verification queries ran with a time budget of 3600 seconds on machines equipped with NVIDIA GeForce GTX 1080 Ti GPUs with 11 GB video memory. Overall, the verification of the CIFAR networks used in our study consumed 558 hours in GPU time, whereas the verification of the MNIST networks demanded a total of 1380 CPU hours.

We note that, although the verification algorithms presented above are complete, they were sometimes unable to solve an instance due to time or memory limitations; we report such instances as unsolved.

## 6.3 Empirical Results

In the following, we will compare our proposed selection method against the F-Race approach, the naïve racing approach as well as selection based on exhaustive evaluation. The latter represents the conceptually simplest baseline for selecting the most robust model from a given set of neural network models. Using this approach, each model is verified with respect to all available input instances during the verification procedure. At each input iteration, the candidate model with the highest certified robust accuracy is selected as the incumbent; *i.e.*, the model that would be returned if the process were terminated at the given iteration. Differently from the racing approaches, the exhaustive evaluation approach does not eliminate any candidate models during the selection process; therefore, when run to completion, it will always achieve a regret of zero.

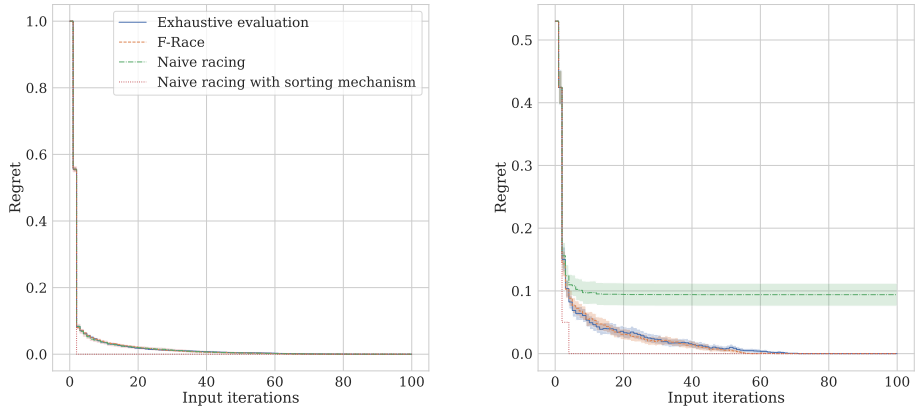



**(a)** MNIST (VeriNet, $\epsilon = 0.04$) **(b)** CIFAR ($\alpha\beta$-CROWN, $\epsilon = 0.008$)

**Figure 6.2:** Number of candidate models as determined by each method after every input iteration. For the three methods that do not use the sorting mechanism, the line represents the average number of candidate models over 200 random input orders, each with different random seeds, along with along with the respective 95% confidence intervals. Clearly, naïve racing, coupled with our proposed sorting mechanism, reduces the number of candidates after substantially fewer input iterations than other methods. Notice that selection based on exhaustive evaluation does not eliminate models from the set of candidates, which, therefore, does not decrease in size.

We evaluated each method in terms of cumulative running time and regret. The former describes the total running time consumed by the verification algorithm until all given input instances have been processed and the most robust model has been determined. Regret, in the context of model selection, describes the difference between the performance of the selected model and the performance of the best model that could have been chosen based on complete and perfect knowledge. In other words, it represents the loss incurred by selecting a sub-optimal model.

Formally, the regret $R$ is defined as follows. Suppose we have a set of candidate models $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$, and we want to select one model from this set based on certified robust accuracy. Let $C_{\text{best}}$ be the best model in the set, *i.e.*, the model with the highest certified robust accuracy $ra$. Then, $R := ra(C_{\text{best}}) - ra(C_{\text{selected}})$, where $ra(C_{\text{best}})$ represents the robust accuracy of the best model and $ra(C_{\text{selected}})$ the robust accuracy of the selected model.



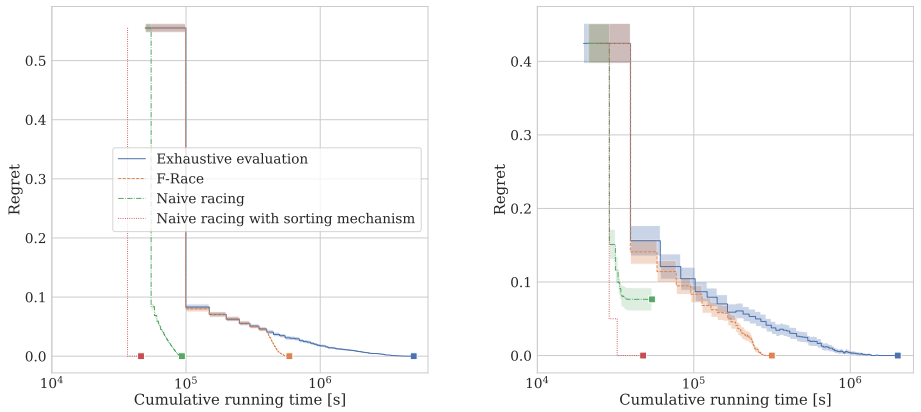

**(a)** MNIST (VeriNet, $\epsilon = 0.04$) **(b)** CIFAR ($\alpha\beta$-CROWN, $\epsilon = 0.008$)

**Figure 6.3:** Regret achieved by the considered methods, where regret describes the difference between the performance of the selected model and the performance of the best model that could have been chosen given all available information. For methods not using the sorting mechanism, the regret was averaged over 200 random input orders, each with different random seeds, and is shown with a 95% confidence interval. The plots show that naïve racing, coupled with our proposed sorting mechanism, achieves optimal regret with fewer input iterations than other methods.

### 6.3.1 Local Robustness at the Decision Boundary

First of all, we investigated the relationship between the local robustness of a neural network model and the estimated distance of an input instance from the decision boundary of the model. More specifically, we examined the empirical cumulative probability distribution of $\Delta$-values across all considered models, giving rise to 3100 individual verification problem instances for MNIST and 2800 for CIFAR. Remember that $\Delta$-values serve as a proxy for the distance of an instance from the closest adjacent class boundary. We normalised these values per network under consideration.

The empirical cumulative distribution of the $\Delta$-values is visualised in Figure 6.1. Notice that some instances could not be verified due to timeouts or memory limitations; we show these instances as unsolved. The plots clearly show that *sat* instances, *i.e.*, instances for which an adversarial example could be found, tend to have a smaller $\Delta$-value than those that are *unsat*, *i.e.*, robust. The difference in distributions is determined as statistically significant by means of a Kolmogorov–Smirnov test with a standard significance threshold of 0.05.




**(a)** MNIST (VeriNet, $\epsilon = 0.04$) **(b)** CIFAR ($\alpha\beta$-CROWN, $\epsilon = 0.008$)

**Figure 6.4:** Regret as a function of cumulative running time for each of the considered methods. Running time represents wall-clock time on the machine on which the experiments were carried out. For methods not using the sorting mechanism, the regret was averaged over 200 random input orders, each with different random seeds, and is shown with a 95% confidence interval. The plots show that naïve racing, coupled with our proposed sorting mechanism, achieves optimal regret while using substantially less compute time than other methods. Each line ends once a specific method has processed all given input instances.

At the same time, Figure 6.1 shows that there exist instances, which are found to be *sat* despite having a relatively large $\Delta$-value, *i.e.*, a $\Delta$-value close to the end of the (normalised) range of values. Upon further investigation, we found that for MNIST, such instances occurred for 12 out of the 31 neural network models we considered and for 15 out of the 27 CIFAR networks. Notice that for these models, no instance was found to be robust, which indicates that large $\Delta$-values can occur also for *sat* instances if a neural network model generally suffers from poor robustness. However, this observation does not affect the performance of our proposed selection method, as models which are non-robust with respect to any input instance would be discarded from the set of candidate models early in the selection process regardless of their $\Delta$-value and, hence, the sorting of input instances. Notice that when removing these neural network models from the set, the difference in $\Delta$-values between *sat* and *unsat* instances grows even larger; more details can be found in the supplementary material.

### 6.3.2 Evaluation of Our Proposed Selection Method

We evaluated our proposed selection method, naïve racing coupled with the sorting mechanism, in terms of cumulative running time and regret, and compared its performance against the following three baselines: (i) F-Race, (ii) naïve racing without a sorting mechanism and (iii) selection based on exhaustive evaluation. For methods that do not employ the sorting mechanism (*i.e.*, all baselines), we repeated the selection process 200 times, where each time the order of the input instances was based on a different random seed. We report the average running time over all runs, along with the respective 95% confidence intervals.

Figure 6.2a displays the size of the set of candidate networks trained on the MNIST dataset throughout the selection process. It shows that our proposed selection method reduces the number of candidate models after fewer iterations compared to each considered baseline. At the same time, for the exhaustive evaluation approach, the number of considered models remains constant, resulting in a larger number of queries that need to be performed at every input iteration.

As the number of candidate models reduces very quickly, it could be assumed that the aggressive nature of our selection method might lead to a sub-optimal outcome of the model selection process. We investigated this potential trade-off and show the results in Figure 6.3a. As can be seen, every method reached an optimal regret, indicating that the significant speed-up does not necessarily compromise on the quality of the selection process. However, we note that some of the MNIST networks were found to be fully robust. These are, consequently, always selected by any of the selection methods, even those that are more aggressive. Lastly, notice that F-Race eliminates candidate models based on statistical evidence, which can lead to models being selected that are less robust than others but where this difference is not found to be statistically significant at the given iteration.

We also tested our method on networks trained on the more challenging CIFAR dataset. Neural networks trained on this dataset are generally more difficult to verify than those trained on the MNIST dataset [71]. Figure 6.2b shows the size of the set of candidate CIFAR networks throughout the selection process. Again, we found that our proposed selection method eliminates candidate models after fewer iterations compared to other methods. Concurrently, the difference between the naïve racing approach with and without the sorting mechanism is much smaller than the difference observed on MNIST networks.

However, Figure 6.3b shows the advantage of the sorting mechanism: the naïve

racing approach using the sorting mechanism very quickly converges towards an optimal regret, while other methods either require substantially more iterations or do not reach the optimum at all. In fact, on this set of models, the naïve racing approach without the sorting mechanism always resulted in a sub-optimal model choice. Overall, these results clearly demonstrate that our new method can effectively select the most robust model, and does so in a more efficient way than F-Race, which discards models only after it obtained statistical significance between the robust accuracy of the candidate models.

Lastly, we studied in more detail the efficiency of our method compared to the baselines we considered, in terms of regret achieved for a specific time budget. This is visualised in Figure 6.4a for MNIST networks and Figure 6.4b for CIFAR networks. Notably, these plots reveal that for both sets of models, our method selects the best-performing, *i.e.*, most robust model while demanding less compute time than any of the considered baselines, especially selection based on exhaustive evaluation. In fact, for MNIST networks, the cumulative running required to complete the selection process is reduced by several orders of magnitude, *i.e.*, a 108-fold speedup factor, when compared to selecting based on exhaustive evaluation (1380.93 *vs* 12.83 hours). Furthermore, for CIFAR networks, our selection method achieved a 41-fold speedup compared to the exhaustive evaluation approach (558.44 *vs* 13.18 hours). Generally, this decrease in cumulative running time occurs because our selection method iteratively eliminates models from the set of candidates, subsequently reducing the number of verification queries in the following iterations, as previously explained. We note that the number of verification queries directly depends on the number of models, which decreases throughout the selection process.

These results highlight that our proposed selection method is well-suited for scenarios in which computing resources are limited, as it is likely to select, within any given amount of running time, models that are more robust than those determined by the baselines considered in our study.

## 6.4 Conclusions and Future Work

In this chapter, we have demonstrated the effectiveness of advanced model selection techniques in the context of neural network verification. Specifically, we studied the problem of selecting the most robust neural network model from a given set of models, whilst reducing the compute time needed to obtain robustness certificates for the given input instances.

To enable our proposed selection method, we introduced a novel sorting mechanism based on the likelihood of an input instance being robust with respect to adversarial input perturbations. This likelihood is captured by means of $\Delta$-values, which serve as a proxy for the distance of an input instance to the model decision boundaries, and we present statistical evidence indicating significant differences in the empirical cumulative distribution of these values for robust and non-robust instances. Overall, our method guides the allocation of computing resources required to perform local robustness verification towards adversarially robust models and can, in principle, be used in combination with any verification system.

We empirically evaluated our method on two diverse sets of 31 and 27 neural networks, trained on the MNIST and CIFAR-10 datasets, respectively. Our results clearly show that our proposed model selection method significantly reduces the cumulative running time required to select the most robust neural network model from these sets. Thereby, we provide an answer to the fourth research question (RQ4) of how to efficiently select the neural network model from a given set of models that achieves the highest certified robust accuracy. Specifically, compared to the exhaustive evaluation approach, our method achieved a speedup factor of 108 for the set of MNIST networks and a speedup factor of 42 for the set of CIFAR networks while still selecting the most robust model.

In future work, we plan to apply our method to other verification tasks (*e.g.*, robustness verification under bias field perturbations), network architectures and datasets, and to perform a systematic analysis of the relationship between $\Delta$-values and the robustness of neural network models. In addition, we are interested in the precise relationship between the $\Delta$-value and the distance to the nearest decision boundary.