Decreasing Model Stealing Querying for Black Box Adversarial Attacks

Steffano Psathas
Supervisors: Chi Hong, Jiyue Huang, Stefanie Roos
EEMCS, Delft University of Technology, The Netherlands
22-6-2022

A Dissertation Submitted to EEMCS faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

Decreasing Model Stealing Querying for Black Box Adversarial Attacks

Steffano Psathas
Supervisors: Chi Hong, Jiyue Huang, Stefanie Roos
EEMCS, Delft University of Technology, The Netherlands

Abstract

A machine learning classifier can be tricked using adversarial attacks, attacks that alter images slightly to make the target model misclassify the image. To create adversarial attacks on black-box classifiers, a substitute model can be created using model stealing. The research question this report address is the topic of using model stealing while minimizing the amount of querying the substitute model needs to train. The solution used in this report is a variant of the ActiveThief algorithm that makes use of active learning to determine which data is being queried. The paper experiments with different subset selection strategies to find the most informative data points. Also, a seeding algorithm based on clustering is explored and finally, a stopping criterion for the ActiveThief algorithm is proposed. These variations are evaluated on their accuracy and the number of queries they take to achieve that accuracy. This paper shows cluster seeding is an alternative to random seeding in ActiveThief. This paper also presents different subset selection strategies that outperform the random sampling strategy. Finally, a stopping criterion based on entropy is introduced that halts the algorithm when an uncertainty threshold is reached.

1 Introduction

In machine learning, some classifiers can receive a certain input and predict to which class it belongs. This is an important aspect for a lot of self-functioning devices, for example, self-driving cars. These cars rely on the input images they receive and have to recognize certain aspects of traffic such as stop signs and crossroads. If the self-driving car has trouble recognizing these traffic signs or mistaking them for something else, it could lead to disastrous consequences.

When an input image is specifically altered to trick the machine learning classifier it is called an adversarial attack [1]. These adversarial attacks work by adding small perturbations to the image that should be almost unnoticeable to the human eye but causes the classifier algorithm to incorrectly classify it. These perturbations are based on the specific target model the attacker tries to trick. Since most target models are black-box models, the attacker has to estimate the target model's workings.

This can be done using a substitute model that mimics the workings of the specific target model [2]. The target model is the model that is being attacked, while the substitute model is a model that tries to copy the target model. Because it is often unknown on which dataset the target model is trained on, it is difficult to find the right datasets to train the substitute model on. To combat this, model stealing can be used. Model stealing is the querying of the target model and feeding the results to the substitute model as a training set. This way the substitute model is trained on the target model's output, which enables the substitute model to copy the workings of the target model. A problem with model stealing is that it requires a lot of querying the target model for each data point in the dataset. All this querying can be time-consuming and quite expensive if the model is monetized. Another problem with model stealing is that it requires real-life example images which can be difficult to find in large quantities. So it is important to find a method to reduce the number of queries needed to train this target model while still having an accurate substitute model.

The purpose of this paper is to present a variation of a technique that addresses the question *assuming that there is an unlabeled dataset from real-world examples, how can you use a subset of that dataset to allow for less target model querying during model stealing while maintaining the accuracy of the substitution model?*. The solution this paper offers is a variation on the ActiveThief algorithm [3]. The ActiveThief algorithm is a model stealing method that makes use of active learning to find the most informative set of data points to query to the target model. This paper experiments with different extensions and improvements on the ActiveThief algorithm inspired by other research.

First, in Section 2 the related work to this problem is expanded on and explained. Following that in Section 3 the methodology and contributions are explained. After that the evaluation and results are examined in Section 4. Next the results are discussed in Section 5. Subsequent, in Section 6 some awareness on responsible research is given. Finally a conclusion and suggestions for future work are given in Section 7.

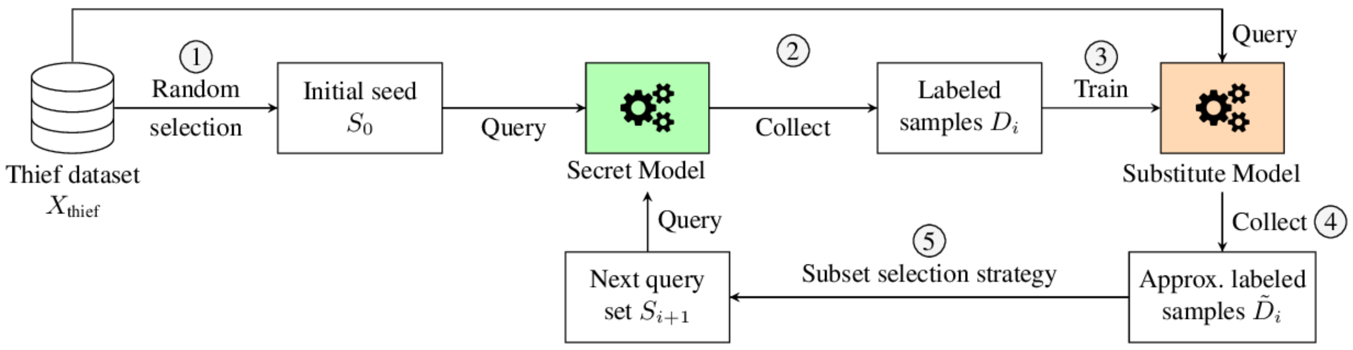


Figure 1: The ActiveThief algorithm.
Source: Adapted from [3]

2 Related Work

It is important to examine how to successfully use model stealing and how to minimize the query costs. This section examines pieces of literature that relate to model stealing. A lot of the literature related to model stealing discusses defending from model stealing. However, the literature studied in this section achieved successful model stealing. First model stealing is examined and after that active learning is explored.

Model Stealing

Model stealing is a method to copy the workings of the target model. The target model is the model that is under attack by an adversarial attack [1] [4]. With model stealing a substitute model is created to imitate the target model. To get the substitute model to behave like the target model it should copy the outcomes of the target model. This is what the substitute model is trained on. The target model is queried with data and returns the predicted labels. Subsequently, the substitute model is trained on the results of querying the target model. This way the substitute model with enough training data can copy the target model.

In [2], Tramèr et al. show that model stealing is possible on black-box targets. These targets include logistic regression, neural networks, and decision trees. For neural networks, they achieved a classifying accuracy of 99.16% while needing 108,200 queries to train the substitute model. This paper uses the idea of model stealing described in by Tramèr et al. but tries to decrease the number of queries needed to successfully use model stealing.

Da Silva et al. in [5] display that a substitute model can be trained using data from the problem domain and non-problem domain. The problem domain is the domain for which the classifier is supposed to work, so a classifier that is trained on numbers would have less trouble classifying numbers than letters. Their experiment showed that it is possible to train a substitute model with high accuracy without using data from the problem domain. While using non-problem domain examples could be considered in this research, the focus is on using real-world examples from the problem domain.

In the article of [6], Ilyas et al. did experiments in different scenarios. The first experiment uses a query limit to control the maximum amount of queries the substitute model is allowed to use. The second experiment examines the substitute model when it only receives one label instead of all probabilities. The final experiment uses partial data, for instance when a query only returns a part of the probabilities. In all scenarios, model stealing was successfully done.

The biggest problem with model stealing is that it requires a lot of data [7]. Especially when using real-world examples as data. In [7] and [8] Kariyappa et al. and Truong et al. propose a method using Data Free Model Extraction to generate their own dataset to use for model stealing.

Active Learning

A solution to the problem of needing a lot of data could be active learning. Active learning is a machine learning technique to select the most informative data points to train the model on. Active learning starts with seeding of some sort [9]. Seeding is the selection of the first batch to initialize the model on. After the seeding, a subset selection strategy is used to find the most informative data points in the remaining dataset [10]. This subset is used to train the classifier. This process is done iteratively until an accuracy or training threshold is reached.

Active learning is normally performed on one classifier, so to use it for model stealing offers to be a challenge. However, in the article of [11], Yu et al. discuss the use of active learning for model stealing and decreasing the query costs. They offer multiple algorithms to select the subset that gets queried to the target model. It is shown that active learning is effective and can be used to minimize query costs. This research makes use of a labeled dataset to choose which data points to use for model stealing, while this research focuses on an unlabeled dataset.

3 Methodology

To solve the problem of decreasing the amount of querying while maintaining the accuracy of the machine learning classifier, active learning can be used. For the solution proposed in this paper, the ActiveThief algorithm is used. To better get an understanding of the ActiveThief algorithm, it is going to be described in the next section. Followed by that is explained what enhancements are added to the ActiveThief algorithm to attempt better performance.

ActiveThief

The ActiveThief algorithm is an active learning model that is used for model stealing. The algorithm uses active learning to determine which data points are sent to the target model for querying [3]. The workings of ActiveThief are shown in Figure 1.

The algorithm starts with an unlabeled dataset. A part of this dataset is used as an initial seed to query the target model. The results of this query are used to train the substitute model. From this initial training of the substitute model, the model is now capable of giving label approximations of the dataset. These approximations can be used in a subset selection strategy to find a subset of the most informative data points. This subset gets queried to the target model to retrieve its labels. This again is used for training the substitute model. Then the cycle continues with finding a subset to query the target model with and train the substitute model with the results. This process is done for a certain amount of iterations.

There seem to be three aspects of the ActiveThief algorithm that could have an alteration that could improve the algorithm. These three aspects are the seeding of the algorithm, the subset selection strategy, and the stopping criteria. In the next sections, these aspects get covered more in-depth.

Seeding Algorithm

The seeding algorithm in the ActiveThief algorithm is based on randomly selecting a sample. With bad luck, this can all be from the same class, which would not be beneficial for the training of the substitute model. The proposed seeding algorithm makes use of clustering and follows Algorithm 1. It is inspired by the solutions proposed in [12]. Clustering finds data points closely related to each other. Using this concept can be beneficial to create a seed that is not composed of data points from the same class, which should be more informative than random seeding. The unlabeled dataset gets queried to the substitute model which gives a prediction of the estimated class. These predictions are used by a K-Means cluster algorithm [13] to find N clusters. The distance between the cluster centers and the predictions by the substitute model is calculated by the L2 norm [14]. The distances furthest away from the cluster centers are used as the initial batch to query the target model. The batch has a batch size of B.

Algorithm 1 Cluster Seeding

```
    Input dataset, N, B
▷ dataset: The dataset used for model stealing
▷ N: The amount of clusters
▷ B: The batch size
    Output a list of data points with size B
  $predictions \leftarrow substitute\_model(dataset)$
  $clusters \leftarrow kmeans(predictions, N)$
  for $data$ in $predictions$ do
      $distances \leftarrow distance(data, clusters)$
  end for
  $distances \leftarrow sort(distances)$        ▷ Sorts the distances in descending order
  return $take(distances, B)$
```

Subset Selection Strategy

The subset selection strategy used in ActiveThief works on uncertainty sampling. A new subset selection strategy or a combination of different strategies can be used to improve the effectiveness of the ActiveThief algorithm. Each time the algorithm iterates a new subset is selected to query the target model.

Random Strategy: The random strategy is randomly sampling from the dataset. This strategy is used as a baseline.

Uniform strategy: The uniform strategy takes into account how many labels there are available [15]. It uses the predicted labels from the substitute model to evenly fill the batch with an equal amount of data points from each label. This strategy is beneficial to even out the batch when one label is predominantly present in the batch and helps diversify it.

Uncertainty strategy: The uncertainty strategies use uncertainty sampling to find the most informative subset. Uncertainty strategies are based on the probabilities the substitute models return for each class. The more uncertain the substitute model is on a data point, the higher the chance that it is used for the next batch of training. To find the most uncertain data points, the strategy can use one of these three formulas. The first one is the least confidence formula from [10] where $\hat{y}$ is the predicted class for data point x. This formula returns the data points where the probability for the class is the lowest. The least confidence strategy has the following formula:

$$x^*_{LC} = \arg\max 1 - P(\hat{y}|x) \tag{1}$$

The following uncertainty formula is margin sampling [10]. Margin sampling takes the difference between the highest and the second-highest prediction for a data point. The lowest difference means the most uncertain and is therefore used for the next batch. Margin sampling uses the following formula:

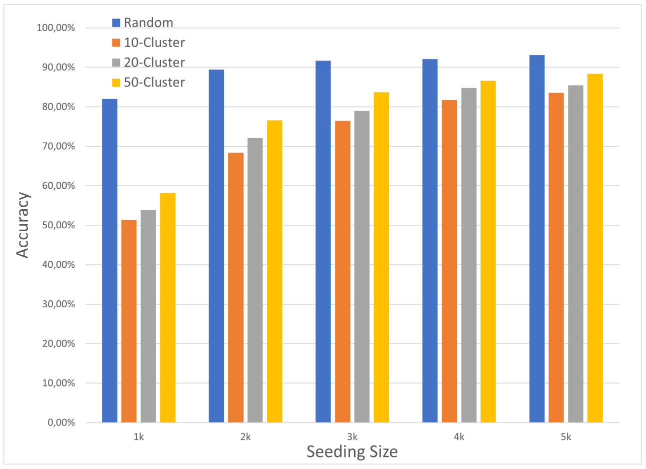
$$x^*_{MS} = \arg\min P(\hat{y_1}|x) - P(\hat{y_2}|x) \tag{2}$$

The final uncertainty strategy is known as entropy [16]. Entropy is an often-used measure in machine learning because it works well in multi-label classifiers since it looks at all the received probabilities. Entropy uses the next formula:

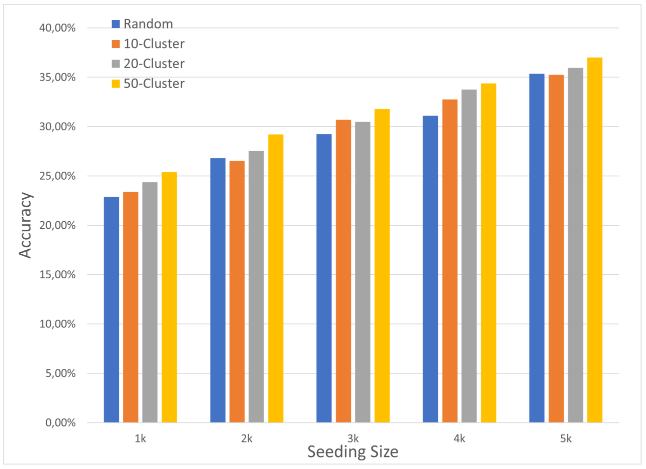
$$x^*_H = \sum_i P(y_i|x) \log P(y_i|x) \tag{3}$$

Clustering Strategy: The clustering strategy is based on the one from [17] and looks similar to the one used in the cluster seeding mentioned earlier. Instead of finding clusters and using them to calculate the distance from the data points, the earlier queried data points are used to find the centers based on their probabilistic results. The distance between the center points and the remaining data points in the data set is calculated with the L2 norm [14]. The furthest data point from each center is used in the new batch. Using clustering is beneficial because it diversifies the data points used in the new batch. This strategy, however, has higher calculation costs since for every iteration it needs to calculate the distance between all the data points again.

Clustering + Entropy Strategy: While the clustering strategy helps with diversifying the new batch, it does not give the most informative data points. Together with an uncertainty



(a) Results on the MNIST dataset.



(b) Results on the FashionMNIST dataset.

Figure 2: Random seeding versus cluster seeding using different number of clusters with different seeding size.

strategy, the combination can give the most informative data points, while maximizing the diversity. Entropy is chosen for the uncertainty strategy because it is one of the most used uncertainty measures and has already proven to work in [3]. In this combination, the entropy strategy is used to create a subset of the most informative data points. After that, the clustering strategy is used on the subset to find the most distant data points to use for the next batch.

Uniform + Entropy Strategy: Like the clustering strategy, the uniform strategy also diversifies the batch. Just like the clustering + entropy strategy, this strategy uses entropy to find the most informative data points and the uniform strategy to diversify the new batch by equally distributing the predicted classes.

Clustering + Entropy + Uniform Strategy: In this strategy, entropy is used to find a subset of the most informative data points. The clustering strategy is used for diversity and to find the most distant data points. The uniform strategy is used to uniformly divide the data points equally on their predicted class. This strategy maximizes the information gain of entropy and the diversifying of the clustering and uniform strategies.

Stopping Criteria

The ActiveThief algorithm currently has no stopping criteria. It iterates over the algorithm a certain amount of times. This means that the algorithm already converged to a certain point or has not converged yet at all. When using stopping criteria the algorithm can use fewer or more queries to reach convergence. The stopping criterion proposed in this article is based on [18] [19], where they use confidence measures to calculate when the algorithm has to stop. The stopping criterion this paper proposes uses entropy to determine if the ActiveThief algorithm should halt or not. Since entropy expresses the amount of information gain a certain data point gives, it can be used to compare the previous iteration with the current one. If the current iteration's entropy does not decrease sufficiently, the algorithm stops. The entropy difference should drop under a value α to stop the algorithm. The stopping criterion follows the next equation:

$$|x^*_{i-1} - x^*_i| \leq \alpha \tag{4}$$

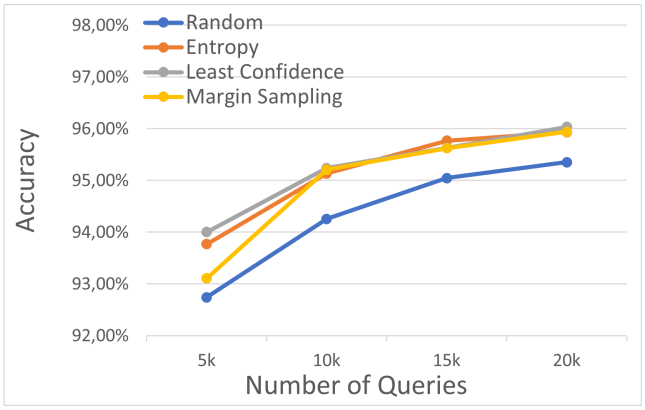
The parameter α is a hyper parameter that can be tuned to achieve the best stopping strategy.

4 Experimental Setup and Results

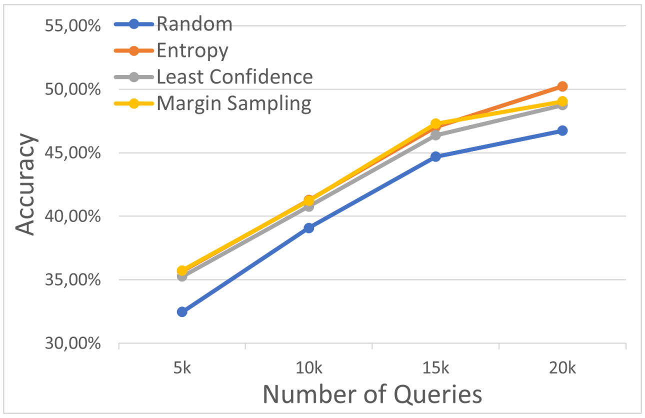
The experiments that are conducted all make use of the Modified National Institute of Standards and Technology (MNIST) dataset [20] and the FashionMNIST dataset [21]. The MNIST dataset consists of handwritten numbers from zero to nine and the FashionMNIST dataset contains pieces of clothing. They both contain 60,000 training data points and 10,000 testing data points. These datasets are used because they are one of the most accessible and widely used datasets. The training sets are used for the ActiveThief algorithm, while the test sets are to verify the accuracy of the substitute model. The substitute model and the target model are both two-layer Convolutional Neural Networks [22]. The target model is initially trained with the training sets. For the training of the models, the RMSProp optimizer [23] is used. The loss function used to train the models is the cross-entropy loss function [24]. The accuracy mentioned in the experiments is the percentage of images that the substitute model classifies correctly. The accuracy is calculated using the test set and calculated by dividing the number of images correctly classified by the substitute model by the size of the test set. The number of queries mentioned in the experiments is the number of data points used to query the target model and use the result for training. When using the whole dataset to train the substitute model, the accuracy for reaches 98.19% and 62.52% for the MNIST and FashionMNIST datasets respectively.

Experimenting with Seeding

The experiment with seeding compares the cluster seeding with the random seeding, where the random seeding is used

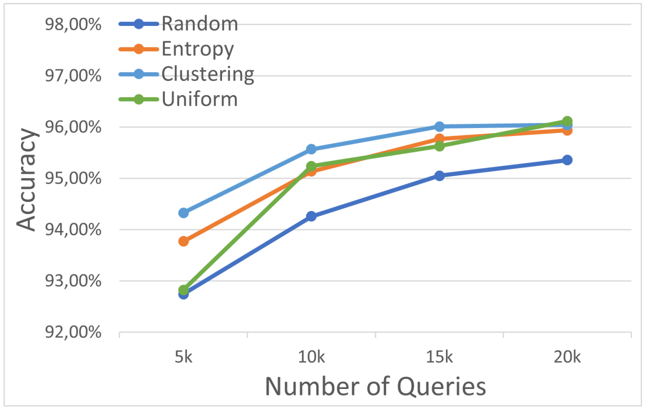


(a) Results on the MNIST dataset.

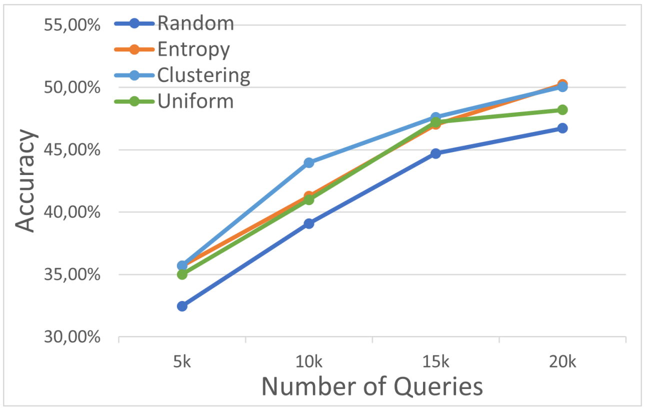


(b) Results on the FashionMNIST dataset.

Figure 3: Uncertainty subset selection strategies versus the random sampling strategy.



(a) Results on the MNIST dataset.



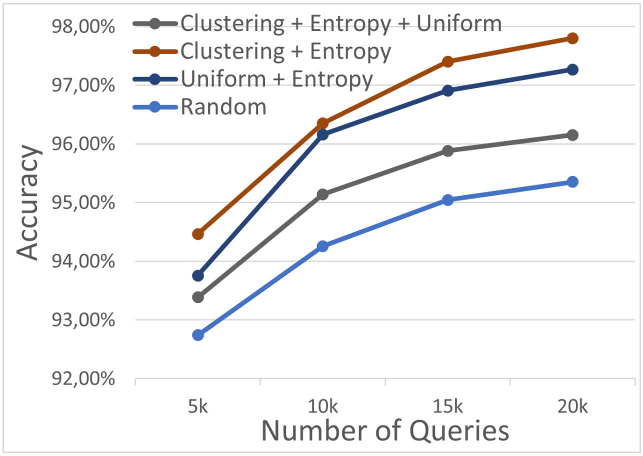
(b) Results on the FashionMNIST dataset.

Figure 4: Entropy, uniform and clustering strategies versus the random sampling strategy.

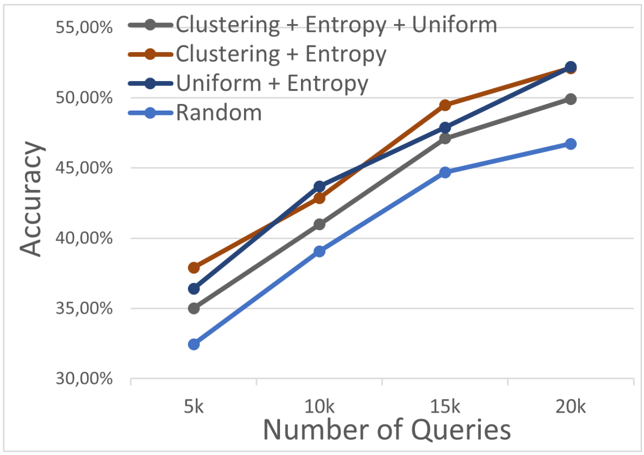
as the baseline. It does that by running one iteration. This experiment is done with different batch sizes ranging from 1k to 5k queries. For cluster seeding, there is also the possibility to change the initial cluster size. This also uses different cluster sizes with 10, 20, and 50 clusters. These experiments are done five times for each possible combination on both datasets. The experiment marks down the accuracy after the seeding process. The experiment gave the results depicted in the graphs in Figure 2. Both graphs show the results of using cluster seeding in comparison with random seeding. Figure 2a shows the results on the MNIST dataset and Figure 2b show the result on the FashionMNIST dataset. Both graphs show that using more cluster centers performs better than when using fewer clusters. On the MNIST dataset, random seeding performs better than cluster seeding, while on the FashionMNIST dataset the result is the other way around.

Experimenting with Subset Selection Strategy

For the subset selection strategy, there are a lot of strategies and combinations of strategies to experiment with. For each strategy the accuracy after a certain amount of queries is recorded. These query sizes consist of 5k, 10k 15k, and 20k, which are respectively 8%, 16%, 25%, and 33% of the whole dataset. The strategies evaluated consist of all mentioned in Section 3. Each strategy is evaluated five times with a batch size of 1k which is repeated for each combination. Random sampling is used as the baseline of the experiment. The accuracy is noted down after the training using the subset selection strategies. The two graphs that are shown in Figure 3 display the results of the uncertainty measures versus the random sampling strategy. All the uncertainty measures perform similarly on both datasets. They all outperform the random sampling strategy in terms of accuracy. The graphs displayed in Figure 4 show the comparisons between the entropy, uniform, clustering, and random sampling strategies. All the strategies outperform the random sampling strategy in terms of accuracy. The cluster strategy slightly outperforms the entropy and the uniform strategies. In Figure 5 the results of the combination of different strategies are graphed. It shows that combining strategies can be effective to improve performance. The cluster strategy together with entropy performs the best, after that closely behind is the uniform strategy with entropy. The strategy that combines clustering, uniform, and entropy gets outperformed by the other two combinations.

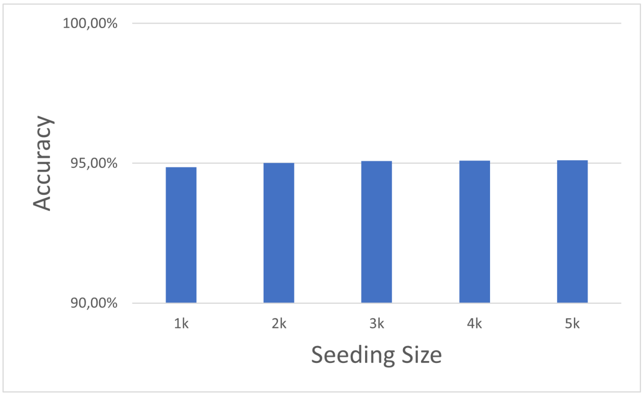


(a) Results on the MNIST dataset.

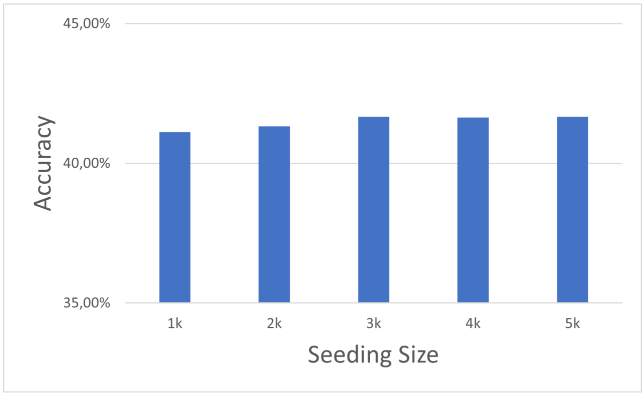


(b) Results on the FashionMNIST dataset.

Figure 5: Combination of several strategies versus the random sampling strategy.

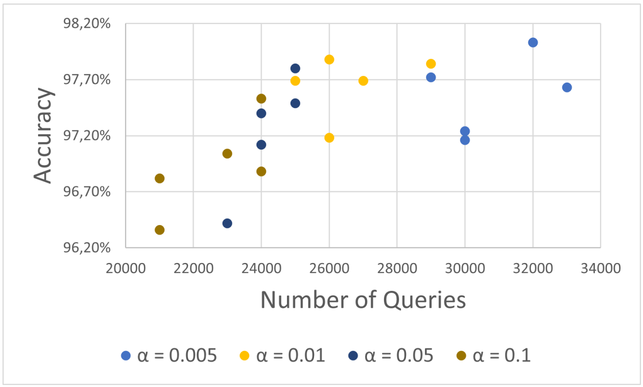


(a) Results on the MNIST dataset.

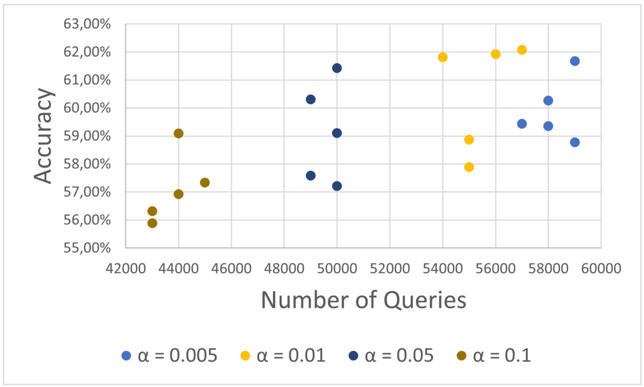


(b) Results on the FashionMNIST dataset.

Figure 6: The effects of seeding size towards the accuracy.



(a) Results on the MNIST dataset.



(b) Results on the FashionMNIST dataset.

Figure 7: The effects of α on the stopping criteria.

Experimenting with Seeding Size Influence

Next to that, an experiment on the influence of the seeding batch size on the efficacy of the algorithm is conducted. This experiment only uses one strategy, this is the entropy strategy. This is done because entropy is one of the most used measures and has already been shown to work in [3]. This experiment uses the seeding batch sizes ranging from 1k to 5k queries. Onwards from that, the entropy strategy continues until the threshold of 10k queries is met. This is also repeated five times for each combination of parameters. In Figure 6 the results of the experiment are depicted. The seeding size increases with every experiment but does not have a significant effect on the resulting final accuracy. All the seeding sizes have similar accuracy.

Experimenting with Stopping Criteria

The stopping criterion based on entropy is evaluated on the effect of the α parameter. In this experiment the entropy-based strategy is tested using different α and parameters, with values of 0.005, 0.01, 0.05, and 0.1 for this evaluation. The result of the experiment is the number of queries until the stopping criteria 'deems' the algorithm finished. Each experiment is run five times. The graphs that are shown in Figure 7 display the results of the experiment. All the results from the same α seem to cluster together. The graphs also suggest that increasing the α value satisfies the stopping criteria at an earlier number of queries. This also results in a slightly lower accuracy compared to using a higher number of queries.

5 Discussion

In this section, the results of the experiments are more closely inspected. Explanations of the specific results are given and compared to other papers. Also, a reflection on the experiments is given to provide improvements for further studies. The different sections of the ActiveThief algorithm are discussed in the order of seeding strategy, subset selection strategy, and stopping criteria.

Seeding Strategy

In this subsection, the seeding strategy and its results are discussed. It starts with the seeding strategy experiment and followed by that the seeding size influence gets discussed. The results in Section 4 seem to suggest that using more clusters in cluster seeding results in a higher accuracy after seeding. This seems logical since there are more centers to compare the data points on, which results in a more representative subset. This is also something that corresponds with [17], where more clusters used correspond to more accurate clustering. Next to that, the random strategy outperforms the cluster strategy on the MNIST dataset. On the FashionMNIST dataset, it is the other way around. Since MNIST achieves a higher accuracy with less querying, it could be considered an easier dataset. This means that the data points are not close to the decision boundaries of the machine learning classifier. This could mean that K-Means clustering needs more clusters to effectively outperform the random strategy. The effects of seeding size on the final accuracy are minimal. This corresponds with the findings in [12] where they experimented with seeding algorithms. They explain that the initial seeding does not have a significant impact on the final accuracy. This happens because the subset selection strategies find the most informative data points to train the machine learning classifier, which gives more information than any seeding strategy. This is also shown when comparing the seeding strategy with 5k queries in Figure 2 to any of the subset selection strategies with 5k queries in Figures 3, 4, and 5. The subset selection strategies all outperform the seeding algorithm. However since some strategies rely on previous queried data points, the cluster seeding could have a performance impact on those strategies.

Subset Selection Strategy

In this subsection, the subset selection strategies and their evaluations are discussed. In Section 4 multiple different subset selection strategies are evaluated. The Figure 3 the uncertainty strategies are evaluated. They all achieve similar accuracy, which could be because all the uncertainty measures try to calculate the same thing, but with a slightly different method. The uncertainty results are similar to the results posted in [3]. The new strategies of uniform and clustering outperform the random sampling strategy. Uniform compared to entropy have similar results where entropy is slightly better. This is because uniform does not look at how informative a data point is but tries to have a uniform distribution of expected labels. This gives the classifier enough examples from each class to train on, but they might not be the most informative examples. The clustering outperforms both the entropy and the uniform strategy. The clustering strategy finds the most distant data points from the previous queried data points, which helps in finding diverse data points. The combining of strategies works effectively. The combination of clustering and entropy and the combination of uniform and entropy outperform all the other strategies. This comes because the entropy strategy helps with finding the most informative data points, while the clustering and uniform strategies help with the diversifying of the data points. This also explains why clustering, entropy, and uniform underperforms compared to the other two combinations of strategies. The clustering and uniform strategies increase the diversity of data points, but since they both do that in different ways they cancel each other out, which results in a worse strategy than the other combination strategies.

Stopping Criteria

This subsection discusses the effectiveness of the proposed stopping criterion. The graphs in Figure 7 show a clear clustering of results with the same α value. It also shows that a higher α value results in earlier convergence and requires less number of queries. However, this earlier convergence leads to overall lower accuracy. This trend is also found in [18] where they use different variations of the uncertainty-based stopping criterion. The use of the whole dataset results in an accuracy of 98.19% and 62.52% for the MNIST and FashionMNIST datasets respectively, which all the α values get rather close to. That suggests that the uncertainty-based stopping criterion stops the algorithm at the moment when the highest achievable accuracy is almost met. Since the accuracy of using the whole dataset is 98.19% and 62.52% for the MNIST and FashionMNIST datasets respectively, the optimal α value is suggested to be between 0.01 and 0.05. This seems to be the case because for those two values the number of queries is minimized while keeping the accuracy around the same as using the whole dataset. This stopping criterion works because the experiments make use of a complete dataset of real-world examples, however since model stealing is often used when there are not a lot of real-world examples the stopping criterion could in those cases not even be met.

6 Responsible Research

The use of adversarial attacks sounds in essence unethical since the goal is to trick a classifier into misclassifying images. However, the findings in this research about model stealing could help future studies in preventing these attacks on machine learning classifiers. This research could also be used to have advances in the defense against model stealing.

Since the method we created is based on the ActiveThief algorithm, it is presumed to be easily reproducible. The experiment follows the parameters as closely as possible as mentioned in [3] to obtain similar, matching results. Although there might be some minor differences in results because training the classifier is based on some randomness for example when using random seeding.

The datasets used for the experiment are the MNIST and FashionMNIST datasets. These datasets are experimented on since they are easily accessible and widely used datasets in the field of machine learning. Therefore this should not result in ethical implications. The datasets have 70,000 entries, for which 60,000 entries are split into a training set and the remaining 10,000 entries are split into a testing set.

7 Conclusions and Future Work

In this paper, extensions to the ActiveThief algorithm are presented. The ActiveThief algorithm shows how to reduce the number of queries required to train a substitute model during model stealing using unlabeled real-world examples. The extensions proposed in this paper include a seeding algorithm using clustering, different types of subset selection strategies, and a stopping criterion. The cluster seeding algorithm performs better than random seeding on the FashionMNIST dataset. The subset selection strategies that combine entropy with either a uniform or clustering strategy perform the best on both datasets. The stopping criterion halts the algorithm when the uncertainty inequality is reached. This paper shows that the ActiveThief algorithm and its extensions on it can decrease the number of queries needed while remaining the accuracy of using a whole dataset.

In future work on this topic, the effectiveness of cluster seeding can be researched on different datasets. Since in this research only two datasets are examined that both reacted differently to cluster seeding, more datasets can give more insights into the effectiveness of cluster seeding on different numbers of clusters. Next to that the influence of cluster seeding on subset selection strategies that require previously selected data points can be part of future research. In this research, the effectiveness of seeding size is only evaluated with the entropy strategy and random seeding. This effectiveness can be different when using cluster seeding together with a clustering strategy and could be explored in future research. In future work new or a new combination of subset selection strategies can also be introduced and evaluated. Finally, the stopping criterion using entropy can be further researched by using different uncertainty measures or even changing the method of using the uncertainty measure.

References

[1] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against deep learning systems using adversarial examples," *CoRR*, vol. abs/1602.02697, 2016.

[2] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," *CoRR*, vol. abs/1609.02943, 2016.

[3] S. Pal, Y. Gupta, A. Shukla, A. Kanade, S. Shevade, and V. Ganapathy, "Activethief: Model extraction using active learning and unannotated public data," vol. 34, pp. 865–872, Apr. 2020.

[4] X. Gong, Q. Wang, Y. Chen, W. Yang, and X. Jiang, "Model extraction attacks and defenses on cloud-based machine learning models," *IEEE Communications Magazine*, vol. 58, no. 12, pp. 83–89, 2020.

[5] J. R. C. da Silva, R. F. Berriel, C. Badue, A. F. de Souza, and T. Oliveira-Santos, "Copycat CNN: stealing knowledge by persuading confession with random non-labeled data," *CoRR*, vol. abs/1806.05476, 2018.

[6] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," *CoRR*, vol. abs/1804.08598, 2018.

[7] S. Kariyappa, A. Prakash, and M. Qureshi, "Maze: Data-free model stealing attack using zeroth-order gradient estimation," 2020.

[8] J. Truong, P. Maini, R. J. Walls, and N. Papernot, "Data-free model extraction," *CoRR*, vol. abs/2011.14779, 2020.

[9] D. Dligach and M. Palmer, "Good seed makes a good crop: Accelerating active learning using language modeling," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, (Portland, Oregon, USA), pp. 6–10, Association for Computational Linguistics, June 2011.

[10] B. Settles, "Active learning literature survey," Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[11] H. Yu, K. Yang, T. Zhang, Y.-Y. Tsai, T.-Y. Ho, and Y. Jin, "Cloudleak: Large-scale deep learning models stealing through adversarial examples.," 2020.

[12] C. J. Mahoney, N. Huber-Fliflet, H. Zhao, J. Zhang, P. Gronvall, and S. Ye, "Evaluation of seed set selection approaches and active learning strategies in predictive coding," *CoRR*, vol. abs/1906.04367, 2019.

[13] Y. Li and H. Wu, "A clustering method based on k-means algorithm," *Physics Procedia*, vol. 25, pp. 1104–1109, 12 2012.

[14] L. Chanzi, Q. Chen, B. Zhou, and H. Li, “l 1 - and l 2 -norm joint regularization based sparse signal reconstruction scheme,” *Mathematical Problems in Engineering*, vol. 2016, pp. 1–11, 01 2016.

[15] R. Willett, R. Nowak, and R. Castro, “Faster rates in regression via active learning,” vol. 18, 2005.

[16] A. Holub, P. Perona, and M. C. Burl, “Entropy-based active learning for object recognition,” pp. 1–8, 2008.

[17] Z. Bodó, Z. Minier, and L. Csató, “Active learning with clustering,” vol. 16, pp. 127–139, 16 May 2011.

[18] J. Zhu, H. Wang, E. Hovy, and M. Ma, “Confidence-based stopping criteria for active learning for data annotation,” vol. 6, apr 2010.

[19] Z. Pullar-Strecker, K. Dost, E. Frank, and J. Wicker, “Hitting the target: Stopping active learning at the cost-based optimum,” *CoRR*, vol. abs/2110.03802, 2021.

[20] L. Deng, “The mnist database of handwritten digit images for machine learning research [best of the web],” *Signal Processing Magazine, IEEE*, vol. 29, pp. 141–142, 11 2012.

[21] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms,” *CoRR*, vol. abs/1708.07747, 2017.

[22] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” in *2017 International Conference on Engineering and Technology (ICET)*, pp. 1–6, 2017.

[23] F. Zou, L. Shen, Z. Jie, W. Zhang, and W. Liu, “A sufficient condition for convergences of adam and rmsprop,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[24] M. Martinez and R. Stiefelhagen, “Taming the cross entropy loss,” in *Pattern Recognition* (T. Brox, A. Bruhn, and M. Fritz, eds.), (Cham), pp. 628–637, Springer International Publishing, 2019.