Transparent Interpretation with Knockout

Xing Han [1]

Abstract

How can we find a subset of training samples that are most responsible for a complicated black-box machine learning model prediction? More generally, how can we explain the model decision to end-users in a transparent way? We propose a new model-agnostic algorithm to identify a minimum number of training samples that are indispensable for a given model decision at a particular test point, as the model decision would otherwise change upon the removal of these training samples. In line with the counterfactual explanation, our algorithm identifies such a set of indispensable samples iteratively by solving a constrained optimization problem. Further, we efficiently speed up the algorithm through approximation. To demonstrate the effectiveness of the algorithm, we apply it to a variety of tasks including data poisoning detection, training set debugging, and understanding loan decisions. Results show that our algorithm is an effective and easy to comprehend tool to help better understand local model behaviors and therefore facilitate the application of machine learning in domains where such understanding is a requisite and where end-users do not have a machine learning background.

1. Introduction

The lack of transparency of the best-performing black-box models, represented by Deep Neural Networks, significantly limits their applications in domains where related decision-making is at high-stakes and therefore a good understanding of model behaviors is a requisite. Motivated by the growing interests and demand in model interpretation, we see a rapidly growing literature on this topic. A variety of techniques, such as influence function (Koh & Liang, 2017), prototypes (Bien et al., 2011), representer points (Yeh et al., 2018), have been proposed to reveal insights of black-box machine learning models. However, most of the existing explanation techniques themselves are quite complicated. Even though it is not a problem for researchers and developers in the machine learning community, there is a great gap in developing end-user (who typically do not have access to sophisticated machine learning algorithms) friendly model explanations methods, which is able to communicate the model decision clearly and intuitively. Such transparent user-oriented model interpretations are crucial to build trust of the machine learning system and facilitate their applications in more real-world scenarios. This paper aims to fill this gap by introducing an effective and easy to comprehend algorithm to cast light on local model predictions and to ease the communication with end-users. For example, when fatal accident happens in autonomous driving (e.g., AI fails to detect human when it should), legal investigation groups may want to know "why the algorithm system come up with the current result"; in normal prediction tasks, engineers want to understand why their algorithm make mistakes on important instances, and identify if there are mislabels or poisoned data in the training dataset.

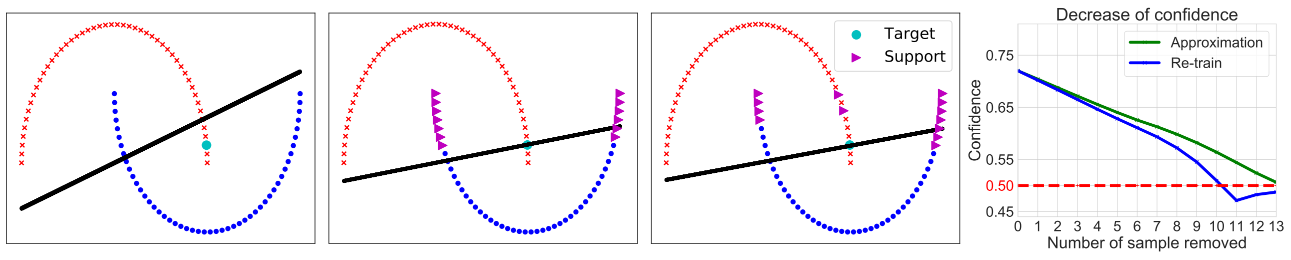
We frame our problem setting as the following: assume $a = \mathcal{A}(D, x^*)$ is a decision made on data instance x^* by algorithm $\mathcal{A}$ based on training data D. We want to understand which part of the training data D that causes algorithm $\mathcal{A}$ to make the current decision. The key idea of our solution is that training sample(s), denoted by c, can be defined as (one of) the explanations of the decision $a = \mathcal{A}(D, x^*)$ if the decision would be different when property c is not presented in the training data. In other words, the condition of being an explanation is $a \neq a' = \mathcal{A}(D \setminus c, x^*)$, where $D \setminus c$ is the modified data-set by removing training sample(s) c. Furthermore, we aims to identify (one of) the most efficient set c as the explanation of model prediction. To this end, our approach constructs the property c iteratively and the process is elaborate in Section 2. In contrast to existing explanation methods, our algorithm is based on an intuitive framework and end-user friendly.

2. Transparent Interpretation Framework

Problem Setup Consider a classification problem where the decision is binary $a \in \{0, 1\}$, and we can obtain an optimal classifier $\hat{\pi}$ using some machine learning algorithm

[1]The University of Texas at Austin, USA. Correspondence to: Xing Han <aaronhan223@utexas.edu>.

2020 ICML Workshop on Human Interpretability in Machine Learning (WHI 2020). Copyright 2020 by the author(s).



(a) (b) (c) (d)

Figure 1. Behavior of classifier on half-moon dataset. (a) Initial decision boundary. (b) Decision boundary after removing supports by approximation. (c) Decision boundary after removing supports by re-training. (d) Decrease of confidence as supports removed[2].

$\mathcal{A}$ applied on a training dataset $\mathcal{D} = \{z_i := (x_i, y_i)\}_{i=1}^n$ based on some objective loss:

$$\hat{\pi} = \arg\min_{\pi} L(\pi) = \arg\min_{\pi} \frac{1}{n}\sum_{i=1}^{n} \delta(z_i, \pi), \qquad (1)$$

where δ is some loss (e.g., cross entropy) of data instance z_i. The goal of our method is to explain why the classifier $\hat{\pi}$ gives a prediction a instead of $\bar{a} = 1 - a$ at a data point x_k. We formulate the problem as finding a (set of) data instance which contributes the most to the claim C:

$$\hat{\pi}(a|x_k) > \hat{\pi}(\bar{a}|x_k),$$

where $\pi(a|x_k)$ is the prediction probability of decision a given by the classifier at x_k.

Counterfactual Explanation To find the set of data points that can explain the claim C efficiently, we first consider the following "counterfactual" explanation, which enforces the claim C to be explained as the opposite:

$$\hat{\pi}_c = \arg\min_{\pi} L(\pi), \quad s.t. \quad \hat{\pi}_c(a|x_k) < \hat{\pi}_c(\bar{a}|x_k) - \epsilon, \qquad (2)$$

where ϵ is a slack variable. Solving (2) gives another solution that attempts to minimize the loss function $L(\pi)$ but with the opposite decision at the data point x_k. Due to the additional constraint, $L(\hat{\pi}_c)$ must be larger than (or equal to) $L(\hat{\pi})$. Recognizing that the data instance $z_{\hat{i}}$ contributing the most to the claim C tends to be the one which voids the most in the "counterfactual" optimization, we can select the most responsible instance for the claim C as

$$\hat{i} = \arg\min_{i}(\delta(z_i, \hat{\pi}_c) - \delta(z_i, \hat{\pi})).$$

Then, we remove $z_{\hat{i}}$ from the training dataset $\mathcal{D}$ and add $z_{\hat{i}}$ to a new dataset $\mathcal{B}$. After this, we solve (1) and (2) again but this time on the updated dataset $\mathcal{D}/z_i$, and select the second most responsible instance in the same way as in the previous round. We repeat the process until the difference

Algorithm 1 Proposed method for finding a set of supports

```
Given dataset $\mathcal{D} = \{z_i := (x_i, y_i)\}_{i=1}^n$, the test point $x_k$ and an empty set $\mathcal{B}$.
Initialize $\mathcal{D}' = \mathcal{D}$.
repeat
  Optimize unconstrained classifier $\hat{\pi}$ using equation (1) on $\mathcal{D}'$;
  Optimize constrained classifier $\hat{\pi}_c$ using equation (2) on $\mathcal{D}'$;
  Select data instance by $\hat{i} = \arg\min_i(\delta(z_i, \hat{\pi}_c) - \delta(z_i, \hat{\pi}))$;
  Add data instance $z_{\hat{i}}$ to set $\mathcal{B}$.
  Remove data instance $z_{\hat{i}}$ from $\mathcal{D}'$: $\mathcal{D}' := \mathcal{D}'/z_{\hat{i}}$ .
until the difference between $L(\hat{\pi}_c)$ and $L(\hat{\pi})$ is statistically insignificant.
```

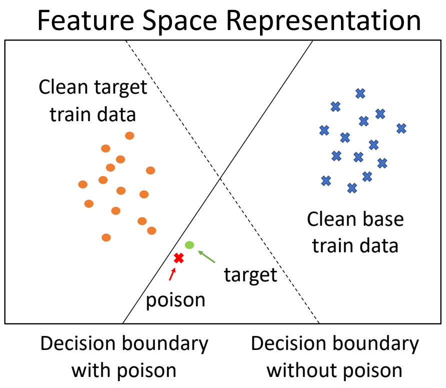
between $L(\hat{\pi}_c)$ and $L(\hat{\pi})$ is statistically insignificant. This yields a set of data instances $\mathcal{B}$ that are most responsible for the given statement. The full algorithm is described in Algorithm 1.

It is worthy to highlight two features of our approach. First, the principle guiding the selection of the most responsible instance is intuitive and easy for users to grasp. Second, the exiting condition of the iteration process ensures the efficiency of the resulted $\mathcal{B}$. By efficiency, we mean the elements in c is quasi-minimized. Specifically, if c satisfies the explanation condition we define above, then it is obvious that c plus some extra samples e together also qualifies as an explanation. But the joint set of c and e is less efficient than c itself, as the former contains redundant elements. In this sense, our algorithm aims to identify (one of) the most efficient set c that is able to explain a local decision made by any black-box model.

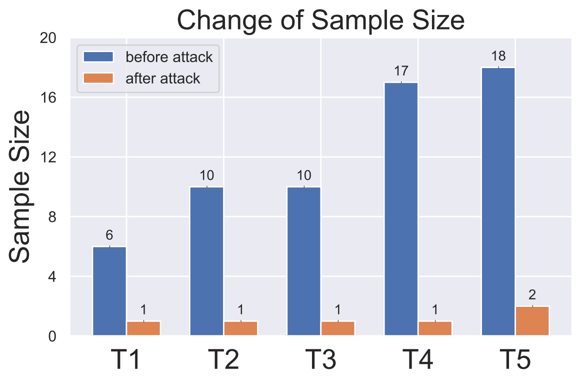
2.1. Efficiently Find Decision Supports

One-step Newton Approximations Algorithm 1 can be implemented iteratively to find a subset of training data that is responsible for a local prediction. However, it is computationally expensive to solve the constraint optimization problem explicitly and to re-train the model on the revised dataset. To efficiently construct the re-

[2]Gif can be found at tiny.cc/67j8bz.


Feature Space Representation
Clean target train data
Clean base train data
target
poison
Decision boundary with poison
Decision boundary without poison

(a) Feature Space Demonstration



(b) Data Poisoning Detection

Figure 2. (a) An illustration of how a successful attack might work by shifting the decision boundary. (b) The comparison of the number of samples selected by our algorithm, before and after attack.

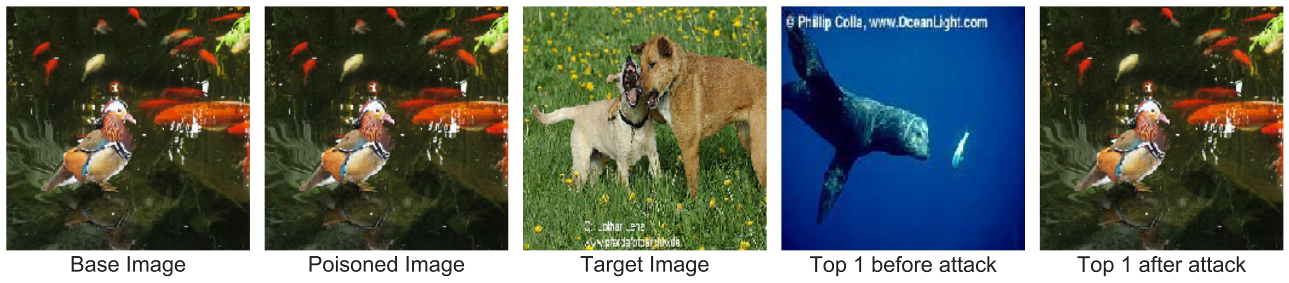


Figure 3. An example of data poisoning detection. From left to right: an image in the training set; its corresponding poisoned image crafted by Shafahi et al. (2018); a test set image we want to attack; the top influential image before and after attack.

sponsible set, we first use a second-order Taylor expansion of the loss function $L(z,\theta)$ parameterized by θ at $\hat{\theta}$: $f(\theta) = L(z,\hat{\theta}) + (\theta-\hat{\theta})^\top \nabla L(z,\hat{\theta}) + \frac{1}{2}(\theta-\hat{\theta})^\top \mathcal{H}_{\hat{\theta}}(\theta-\hat{\theta})$, where $\mathcal{H}_{\hat{\theta}}$ denotes the Hessian matrix of the loss function at $\hat{\theta}$. We then formulate a quadratic programming problem subject to linear inequality constraint:

$$\hat{\theta}' = \arg\min_{\theta}\{f(\theta) \mid \log \pi(\bar{a}|x_k,\theta) - \log \pi(a|x_k,\theta) - \epsilon \geq 0\}. \tag{3}$$

Parameter $\hat{\theta}'$ under the constraint can be obtained by solving (3) via KKT conditions. We use $\hat{\theta}'$ to update the original parameter $\hat{\theta}$ until a sufficient number of iterations is reached. After a data instance has been selected and removed from the training set, we approximate its effect on the model parameters using one-step newton update instead of re-training the model: $\hat{\theta} \leftarrow \hat{\theta} + \frac{1}{n}\mathcal{H}_{\hat{\theta}}^{-1}\nabla_\theta L(z_{\hat{\imath}},\hat{\theta})$. We can further speed up by avoiding the calculation of the Hessian-vector products (HVP), because it is equivalent to $(\theta - \hat{\theta})$ as it is an optimal solution of a quadratic formula.

Exiting Conditions We evaluate the effect of removing the identified set of responsible samples by comparing $L(\hat{\pi}_c)$ and $L(\hat{\pi})$. The loss under the counterfactual optimization is inherently greater than the original loss due to the imposed constraint, but their difference may not be statistically distinguishable when sufficient points are removed. We use a stopping criteria of $L(\hat{\pi}_c) + \epsilon < L(\hat{\pi})$, where ϵ is arbitrary small constant. With a small threshold ϵ, this stopping condition guarantees that the claim is falsified when the selected points are removed.

Illustration Using Simulated Data We demonstrate the working mechanism of our algorithm in Figure 1 by applying it to the half-moon dataset, which contains 100 samples divided into two classes. We first train a linear classifier and select a target data point that is wrongly classified with confidence 74% (Figure 1(a)). We then iteratively find a subset of samples that support this decision following Algorithm 1. As a comparison, we use both one-step approximation and re-training (Figure 1(b) and Figure 1(c)). We find that approximation method identifies 13 samples to support its decision while re-training method selects 11 samples (after removing these samples, the target will become an

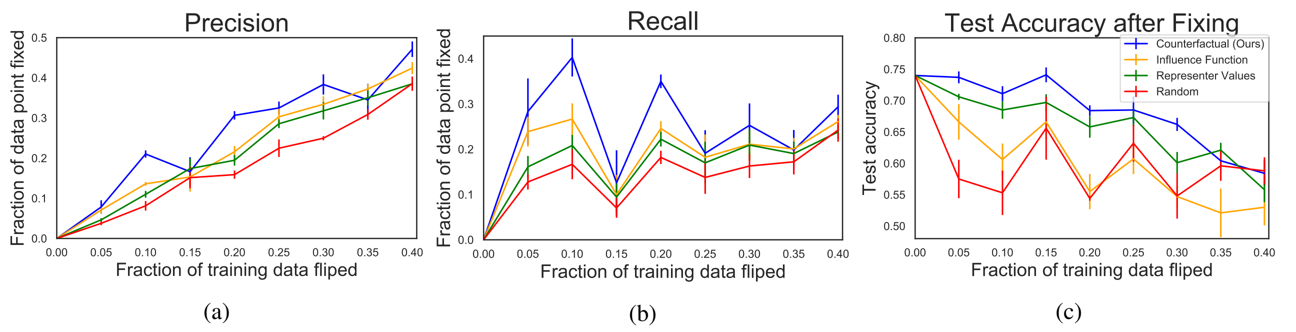


Figure 4. Training set debugging on Enron1 spam dataset. Precision is defined as the number of correctly fixed training sample divided by the number of selected training sample. Recall is defined as the number of correctly fixed training sample divided by the total number of training sample flipped. All algorithms have no access to the test data.

ambiguous point that lies on the decision boundary). Figure 1(d) compares the change of confidence as each support is removed. The re-training approach reaches the decision boundary faster and the 12^{th} and 13^{th} samples are selected to balance the decision boundary. Both methods identify a set of support with a significant overlap between them.

3. Experiments

We perform a variety of experiments to demonstrate the effectiveness of our algorithm in different application domains, including detection of data poisoning attack, training set debugging and understanding loan decisions. Code implementation for reproducing our results can be found at bit.ly/2Yhysvh.

3.1. Data Poisoning Detection

Data poisoning is a type of attack on machine learning models in which an attacker manipulates the behaviour of the model by adding examples to the training set. Shafahi et al. (2018) has shown recently that complex neural network models, especially in transfer learning settings, are highly vulnerable to data poisoning attack: by crafting a poisoning sample that is close to the target testing data in the feature space but with different label, the attacker can achieve a 100% success rate of causing a misclassification. We can visually demonstrate the data poisoning process in Figure 2(a), where a poison instance with opposite class is placed close to the target data point and flips the decision boundary to cause misclassification. Although this “one shot kill” (Shafahi et al., 2018) can make a successful data poisoning attack in almost all the cases, our method can efficiently detect the poisoning sample inside the training set.

We validate our algorithm by attacking a pretrained Inception V3 (Szegedy et al., 2016) network on ImageNet (Russakovsky et al., 2015) dog-vs-fish dataset. 500 instances were selected from each class as the training data, and we randomly select 5 targets (T1 to T5) from the test set to create poisoning samples. Figure 2(b) shows the number of supporting points selected by our method, before and after attacking. It requires a greater amount of supporting points to flip the decision boundary in the clean dataset than the attacked dataset. This is expected, since the decision of the attacked data would be flipped once the poisoning sample is found and removed. Figure 3 demonstrates an example of the data poisoning detection. We can see our algorithm is able to correctly identify the harmful image as the top 1 supporting point after the attack.

3.2. Training Set Debugging

A good training set is essential for learning models that generalize well. Bugs and flaws can adversely affect the performance of learning. Because the whole training set is too large for human inspection, it is highly desirable to have algorithms to automatically identify data points with bugs and prioritize the most problematic points for checking. Previous works (Koh & Liang, 2017; Yeh et al., 2018; Khanna et al., 2018; Zhang et al., 2018) have demonstrated different approaches for solving this problem. However, these methods do not provide indication of how much training data need to be checked in order to improve the test accuracy to a desired level. Our method defines this threshold and achieves a better quantitative performance.

We use the Enron1 spam dataset (Metsis et al., 2006) with 3317 training, 830 validation, and 1035 test examples, whose labels in the training set are randomly flipped in certain percentage. After training a logistic regression classifier on the noisy training set, we find samples that support incorrect predictions with high-confidence in the validation set, which are likely to be bugs in the training set. We compare our method with a number of baseline algorithms, including influence function (Koh & Liang, 2017), representer point (Yeh et al., 2018), and random selection. For each method, we focus on the top-N influential samples where

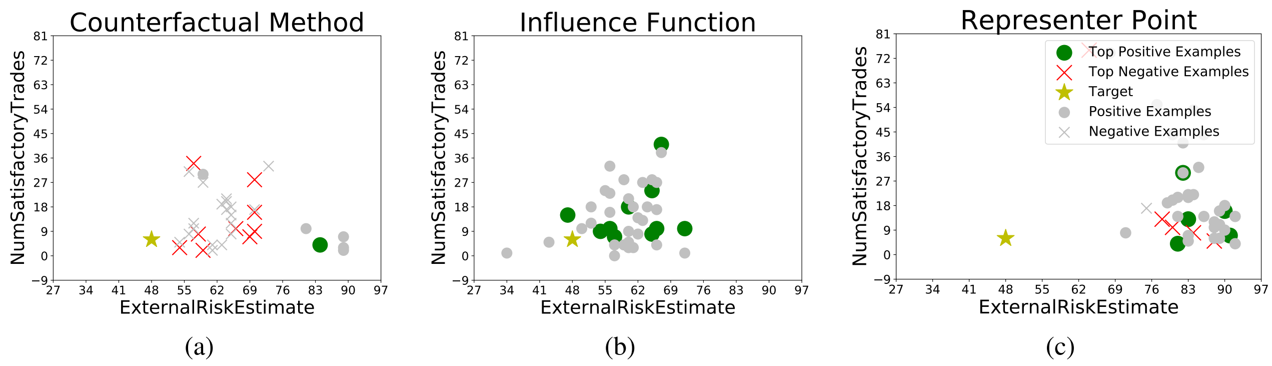


Figure 5. Distribution of training samples that support the wrong prediction of target with 99% confidence. The top 10 positive / negative supports are labeled as green dot / red cross, the following 30 positive / negative supports are labeled as grey dot / grey cross.

N is the number of supporting samples from our algorithm.

Figure 4 (a), (b) demonstrate the precision and recall of the top-N selection. We find that our method tends to give the highest precision and recall in detecting training set bugs, and also the highest test accuracy after fixing these bugs. Note that the recall does not monotonically increase with the fraction of training data fixed; this is expected, because when the number of flips grows, the number of supporting samples is not necessarily larger.

3.3. Understanding Loan Decisions

FICO Score is used in more than 90% of lending decisions, which can significant impact on both customers and financial institutions. However, when machine learning methods are applied in loan decisions, it inevitably causes errors and mistakes. Our method can serve as a reliable debugging tools when mistakes are discovered, and allows people to quickly trace the source of the error to develop strategies for improving the system.

We validate our algorithm on the HELOC (Home Equity Line of Credit) credit application dataset[3], which is used in FICO 2018 xML challenge. Figure 5 shows the explanations provided by our method and baselines for a wrongly predicted target, where the data points are visualized on two important features (`ExternalRiskEstimate` and `NumSatisfactoryTrades`). In this experiment, we choose a target in test set with positive label (good credit performance), but is predicted as negative (bad credit performance) with high confidence by the classifier. Figure 5(a) shows the supporting points selected by our method. We can see the target is surrounded by samples with negative labels, with only a few positive examples in a further distance. This provides an reasonable explanation of the mistake, and allows people to further inspect the negative supporting points in order to fix the problem. On the other hand, we find that the influence function method (Figure 5(b)) can only select positive supporting points with positive labels; although the selected points are close to the target, they do not provide an explanation of how mistake is made. In comparison, we find that the representer point method (shown in Figure 5(c)) provides supporting points from both classes, but all with large distance from the target and hence provide no convincing explanation to the mistake.

[3]https://community.fico.com/s/explainable-machine-learning-challenge

4. Conclusion

We developed a model-agnostic and intuitive algorithm to identify a set of responsible samples that a given model decision relies on. We build the supporting set iteratively by solving a quadratic programming problem with linear constraint, and speed up the re-training process through updating the model parameter with one-step newton. To demonstrate the effectiveness of our method empirically, we apply it to a variety of real-world tasks and multiple benchmark datasets, and compare the performance of our algorithm with other prevailing explanation approaches. Results show that our algorithm produces better quantitative results as well as more intuitive and reliable explanations for local model predictions.

In future work, we would like to explore the application of our method in other domains where interpretability is desired and required, such as medical decisions, time-series forecasting or language/speech models. We also plan to investigate the robustness of our method and provide a certified confidence lower bound on the interpretation.

References

Bien, J., Tibshirani, R., et al. Prototype selection for interpretable classification. *The Annals of Applied Statistics*, 5(4):2403–2424, 2011.

Khanna, R., Kim, B., Ghosh, J., and Koyejo, O. Interpreting black box predictions using fisher kernels. *arXiv preprint arXiv:1810.10118*, 2018.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1885–1894. JMLR. org, 2017.

Metsis, V., Androutsopoulos, I., and Paliouras, G. Spam filtering with naive bayes-which naive bayes? In *CEAS*, volume 17, pp. 28–69. Mountain View, CA, 2006.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3): 211–252, 2015.

Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*, pp. 6103–6113, 2018.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Yeh, C.-K., Kim, J., Yen, I. E.-H., and Ravikumar, P. K. Representer point selection for explaining deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 9291–9301, 2018.

Zhang, X., Zhu, X., and Wright, S. Training set debugging using trusted items. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.