COLLIDER: A Robust Training Framework for Backdoor Data

Hadi M. Dolatabadi, Sarah Erfani, and Christopher Leckie

School of Computing and Information Systems
The University of Melbourne
Parkville, Victoria, Australia
hadi.mohagheghdolatabadi@student.unimelb.edu.au

Abstract. Deep neural network (DNN) classifiers are vulnerable to backdoor attacks. An adversary poisons some of the training data in such attacks by installing a trigger. The goal is to make the trained DNN output the attacker's desired class whenever the trigger is activated while performing as usual for clean data. Various approaches have recently been proposed to detect malicious backdoored DNNs. However, a robust, end-to-end training approach, like adversarial training, is yet to be discovered for backdoor poisoned data. In this paper, we take the first step toward such methods by developing a robust training framework, COLLIDER, that selects the most prominent samples by exploiting the underlying geometric structures of the data. Specifically, we effectively filter out candidate poisoned data at each training epoch by solving a geometrical coreset selection objective. We first argue how clean data samples exhibit (1) gradients similar to the clean majority of data and (2) low local intrinsic dimensionality (LID). Based on these criteria, we define a novel coreset selection objective to find such samples, which are used for training a DNN. We show the effectiveness of the proposed method for robust training of DNNs on various poisoned datasets, reducing the backdoor success rate significantly.

Keywords: backdoor attacks, data poisoning, coreset selection, local intrinsic dimensionality, efficient training.

1 Introduction

Deep neural networks (DNN) have gained unprecedented attention recently and achieved human-level performance in various tasks such as object detection [1]. Due to their widespread success, neural networks have become a promising candidate for use in safety-critical applications, including autonomous driving [2,3] and face recognition [4,5]. Unfortunately, it has been shown that neural networks may exhibit unexpected behavior when facing an adversary.

It has been shown that neural networks suffer from *backdoor attacks* [6]. In such attacks, the attacker has control over the training process. Usually, the adversary poisons a portion of the training data by installing a trigger on natural inputs. Then, the neural network is trained either by the adversary or the user

on this poisoned training data. The attacker may add its trigger to the inputs at inference time to achieve its desired output. However, such poisoned neural networks behave ordinarily on clean data. As such, defending neural networks against backdoor attacks can empirically be an arduous task.

In the most common setting for backdoor defense, motivated by the rise of Machine Learning as a Service (MLaaS) [7], it is assumed that the user outsources training of its desired model to a third party. The adversary then can exploit this freedom and provide a malicious, backdoored neural network to the user [8,9,10,11,12]. From this perspective, the status-quo defense strategies against backdoor attacks can be divided into two categories [13]. In *detection-based* methods the goal is to identify maliciously trained neural networks [14,11]. *Erasing-based* approaches, in contrast, try to effectively eliminate the backdoor data ramifications in the trained model, and hence, give a pure, backdoor-free network [8,9,15,16,17].

A less-explored yet realistic scenario in defending neural networks against backdoor poisonings is when the user obtains its training data from untrustworthy sources. Here, the attacker can introduce backdoors into the user's model by solely poisoning the training data [6,18,13,19]. In this setting, existing approaches have several disadvantages. First, these methods may require having access to a *clean held-out validation dataset* [18]. This assumption may not be valid in real-world applications where collecting new, reliable data is costly. Moreover, such approaches may need a *two-step training procedure*: a neural network is first trained on the poisoned data. Then, the backdoor data is removed from the training set using the previously trained network. After purification of the training set, the neural network needs to be re-trained [18,19]. Finally, some methods achieve robustness by *training multiple neural networks* on subsets of training data to enable a "majority-vote mechanism" [13,20,21]. These last two assumptions may also prove expensive in real-world applications where it is more efficient to train a *single* neural network only *once*. As a result, one can see that a standard, robust, and end-to-end training approach, like adversarial training, is still lacking for training on backdoor poisoned data.

To address these pitfalls, in this paper we leverage the theory of coreset selection [22,23,24,25,26] for end-to-end training of neural networks. In particular, we aim to sanitize the possibly malicious training data by training the neural network on a subset of the training data. To find this subset in an online fashion, we exploit coreset selection by identifying the properties of the poisoned data. To formulate our coreset selection objective, we argue that the *gradient space characteristics* and *local intrinsic dimensionality* (LID) of poisoned and clean data samples are different from one another. We empirically validate these properties using various case studies. Then, based on these two properties, we define an appropriate coreset selection objective and effectively filter out poisoned data samples from the training set. As we shall see, this process is done online as the neural network is being trained. As such, we can effectively eliminate the previous methods' re-training requirement. We empirically show the successful performance of our method, named COLLIDER, in training robust neural net-

works under various backdoor data poisonings resulting in about 25% faster training.

Our contributions can be summarized as follows:

- To the best of our knowledge, we are the first to introduce a practical algorithm for single-run training of neural networks on backdoor data by using the idea of coreset selection.
- We characterize clean data samples based on their gradient space and local intrinsic dimensionality and define a novel coreset selection objective that effectively selects them.
- We perform extensive experiments under different settings to show the excellent performance of the proposed approach in reducing the effect of backdoor data poisonings on neural networks in an online fashion.

2 Related Work

This section reviews some of the most related work to our proposed approach. For a more thorough overview of backdoor attacks and defense, please see [27,13]

2.1 Backdoor Attacks

In BadNets, Gu *et al.* [6] showed that neural network image classifiers suffer from backdoor attacks for the first time. Specifically, the training data is poisoned by installing small triggers in the shape of single pixels or checkerboard patterns on a few images. This poisoned data may come from any class. Therefore, their label needs to be modified to the target class by the adversary. As the labels are manipulated in addition to the training data, this type of backdoor data poisoning is known as *dirty-label* attacks. Similar findings have also been demonstrated on face-recognition networks using dirty-label data poisoning [28,29].

Different from dirty-label attacks, one can effectively poison the training data without changing the labels. As the adversary does not alter the labels even for the poisoned data, such attacks are called *clean-label* [30,31]. To construct such backdoor data, Turner *et al.* [31] argue that the underlying image, before attaching the trigger, needs to become "hard-to-classify." Intuitively, this choice would force the neural network to rely on the added trigger rather than the image semantics and hence, learn to associate the trigger with the target class more easily [31]. To this end, Turner *et al.* [31] first render hard-to-classify samples using adversarial perturbation or generative adversarial network interpolation and then add the trigger. To further strengthen this attack, they reduce the trigger intensity (to go incognito) and install the trigger to all four corners of the poisoned image (to evade data augmentation). Various stealthy trigger patterns can also help in constructing powerful clean-label attacks. As such, different patterns like sinusoidal strips [32], invisible noise [33,34], natural reflections [35], and imperceptible warping (WANet) [36] have been proposed as triggers.

In a different approach, Shafahi *et al.* [30] use the idea of "feature-collision" to create clean-label poisoned data. In particular, they try to find samples that are

(1) similar to a base image and (2) close to the target class in the feature space of a pre-trained DNN. This way, if the network is re-trained on the poisoned data, it will likely associate target class features to the poisoned data and hence, can fool the classifier. Saha *et al.* [37] further extend "feature-collision" to data samples that are poisoned with patch triggers. These triggers are installed at random locations in a given image.

2.2 Backdoor Defense

Existing backdoor defense techniques can be divided into several categories [13]. Some methods aim at *detecting backdoor poisoned data* [18,38,39,40]. Closely related to our work, Jin *et al.* [41] try to detect backdoor samples using the local intrinsic dimensionality of the data. To this end, they extract features of each image using a pre-trained neural network on clean data. In contrast, as we shall see in Sec. 4, we use the feature space of the same neural network we are training, which may have been fed with poisoned data. *Identification of poisoned models* is another popular defense mechanism against backdoor attacks. In this approach, given a DNN model, the aim is to detect if it has backdoors. Neural cleanse [9], DeepInspect [10], TABOR [42], and Universal Litmus Patterns [11] are some of the methods that fall into this category. Furthermore, some techniques aim at *removing the backdoor data effects in a trained neural network* [8,9,15,16,17].

Most related to this work are approaches that try to *avoid learning the triggers during training* [18,19]. To this end, they first train a neural network on poisoned data. Then, using the backdoored DNN and a clean validation set, they extract robust statistical features associated with clean samples. Next, the training set is automatically inspected, and samples that do not meet the cleanness criteria are thrown away. Finally, the neural network is re-trained on this new training dataset. In contrast, our approach does not require additional certified clean data. Moreover, our proposed method trains the neural network only once, taking less training time compared to existing methods [18,19]. Other approaches in this category, such as deep partition aggregation [20] and bagging [21], require training multiple networks to enable a "majority-vote mechanism" [13]. However, our approach focuses on the robust training of a single neural network.

3 Background

As mentioned in Sec. 1, our approach consists of a coreset selection algorithm based on the gradient space attributes and the local intrinsic dimensionality of the data. This section reviews the background related to coreset selection and local intrinsic dimensionality.

3.1 Coreset Selection

Coreset selection refers to algorithms that create weighted subsets of the original data. For deep learning, these subsets are selected so that training a model over

them is approximately equivalent to fitting a model on the original data [25]. Closely related to this work, Mirzasoleiman *et al.* [26] exploit the idea of coreset selection for training with noisy labels. It is argued that data with label noise would result in neural network gradients that differ from clean data. As such, a coreset selection objective is defined to select the data with "the most centrally located gradients" [26]. The network is then trained on this selected data.

3.2 Local Intrinsic Dimensionality

Traditionally, classical expansion models such as generalized expansion dimension (GED) [43] were used to measure the intrinsic dimensionality of the data. As a motivating example, consider two equicenter balls of radii r_1 and r_2 in a d-dimensional Euclidean space. Assume the volumes of these balls are given as V_1 and V_2, respectively. Then, the space dimension can be deduced using

$$\frac{V_2}{V_1} = \left(\frac{r_2}{r_1}\right)^d \Rightarrow d = \frac{\ln\left(V_2/V_1\right)}{\ln\left(r_2/r_1\right)}.$$

To estimate the data dimension d, GED formulations approximate each ball's volume by the number of data samples they capture [44,43].

By extending the aforementioned setting into a statistical one, classical expansion models can provide a local view of intrinsic dimensionality [45,46]. To this end, the natural analogy of volumes and probability measures is exploited. In particular, instead of a Euclidean space, a statistical setting powered with continuous distance distributions is considered.

Definition 1 (Local Intrinsic Dimensionality (LID) [45,47]). *Let $\boldsymbol{x} \in \mathbb{X}$ be a data sample. Also, let $r > 0$ denote a non-negative random variable that measures the distance of $\boldsymbol{x}$ to other data samples, and assume its cumulative distribution function is denoted by $F(r)$. If $F(r)$ is positive and continuously differentiable for every $r > 0$, then the LID of $\boldsymbol{x}$ at distance r is given by*

$$\mathrm{LID}_F(r) \triangleq \lim_{\epsilon \to 0^+} \frac{\ln(F((1+\epsilon)r)/F(r))}{\ln(1+\epsilon)} = \frac{rF'(r)}{F(r)},$$

whenever the limit exists. The LID at $\boldsymbol{x}$ is defined by taking the limit of $\mathrm{LID}_F(r)$ *as $r \to 0^+$*

$$\mathrm{LID}_F = \lim_{r \to 0^+} \mathrm{LID}_F(r). \tag{1}$$

Calculating Eq. (1) limit is not straightforward as it requires knowing the exact distance distribution $F(r)$. Instead, several estimators using extreme value theory (EVT) have been proposed [48,49]. Given its efficacy, here we use the following maximum likelihood estimator for LID [49,47]

$$\widehat{\mathrm{LID}}(\boldsymbol{x}) = -\left(\frac{1}{k}\sum_{i=1}^{k} \log \frac{r_i(\boldsymbol{x})}{r_k(\boldsymbol{x})}\right)^{-1}, \tag{2}$$

where $r_i(\boldsymbol{x})$ is the distance between the data $\boldsymbol{x}$ and its i-th nearest neighbor from the dataset. As seen, only data samples in a local neighborhood would be considered in such an estimate. Thus, under this regime, a *local view* of the intrinsic dimensionality is obtained.

Finally, note that even computing the estimate given in Eq. (2) might become computationally prohibitive, especially for high-dimensional datasets with thousands of samples. Thus, to further make this computation straightforward, Ma *et al.* [50,47] propose an alternative approach. First, randomly sampled mini-batches of data are used to determine the set of nearest neighbors. Second, instead of working with high-dimensional data samples directly, their feature space representation given by an underlying DNN is used. We will also be using this approximator in our approach.

4 Proposed Method

In this section, we formally define the problem of training neural networks with backdoor poisoned data. Next, we argue that two critical features can characterize clean samples in a backdoor poisoned dataset. First, gradient updates for clean data differ in their magnitudes and directions from those of poisoned data [51]. Second, clean data usually has lower LID than its tampered counterparts [47,41,46]. We define a COreset selection algorithm with LocaL Intrinsic DimEnisonality Regularization based on these two criteria, called COLLIDER. Using COLLIDER, we effectively select data samples that satisfy the properties mentioned above and are likely to be clean. As such, the neural network trained on this subset of data would not be affected by the installed triggers. As we shall see, COLLIDER has several advantages over existing methods. Namely, it does not require (1) having access to clean validation data, (2) re-training the whole model, or (3) training multiple neural networks.

4.1 Problem Statement

Let $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n} \subset \mathbb{X} \times \mathbb{C}$ denote a training dataset. Each sample consists of an image $\boldsymbol{x}_i$ from domain $\mathbb{X}$ and a label y_i which takes one of the k possible values from the set $\mathbb{C} = [k] = \{1, 2, \ldots, k\}$. Suppose that $f_{\boldsymbol{\theta}} : \mathbb{X} \to \mathbb{R}^k$ denotes a neural network image classifier with parameters $\boldsymbol{\theta}$. It takes an image $\boldsymbol{x}_i \in \mathbb{X}$ and outputs a k-dimensional real-valued vector, also known as the logit. This vector gives the predicted label of the input $\boldsymbol{x}_i$ by $\hat{y}_i = \arg\max f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$. The neural network is trained to minimize an appropriately defined objective function

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i \in V} \ell\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i\right) = \sum_{i \in V} \ell_i(\boldsymbol{\theta}), \tag{3}$$

over the training set. Here, $\ell(\cdot)$ stands for a standard cost function such as cross-entropy, and $V = [n] = \{1, 2, \ldots, n\}$ denotes the set of all training data. Optimizing Eq. (3) using gradient descent requires finding the gradient of the

loss for all the data points in V, which is costly. In practice, one usually takes random mini-batches of training data and performs stochastic gradient descent.

Here, we assume that the training dataset consists of backdoor poisoned data. In particular, let $\mathcal{B} : \mathbb{X} \to \mathbb{X}$ denote a backdoor data poisoning rule by which the data is tampered.[1] In targeted backdoor attacks, which we consider here, the adversary first replaces some training data samples $\boldsymbol{x}_i$ with their poisoned counterparts $\mathcal{B}(\boldsymbol{x}_i)$. The goal is to force the model to unintentionally learn the injection rule, so that during inference for any test sample $(\boldsymbol{x}_{\text{test}}, y_{\text{test}})$ the neural network outputs

$$y_{\text{test}} = \arg\max f_{\boldsymbol{\theta}}(\boldsymbol{x}_{\text{test}}) \quad \& \quad t = \arg\max f_{\boldsymbol{\theta}}(\mathcal{B}(\boldsymbol{x}_{\text{test}})), \tag{4}$$

where $t \in \mathbb{C}$ denotes the attacker's intended target class. In this paper, we want to meticulously select a subset S of the entire training data V such that it only contains clean data samples. To this end, we carefully identify properties of the clean data that should be in the set S. We define a coreset selection objective based on these criteria to form the subset S at each epoch. The model is then trained on the data inside the coreset S.

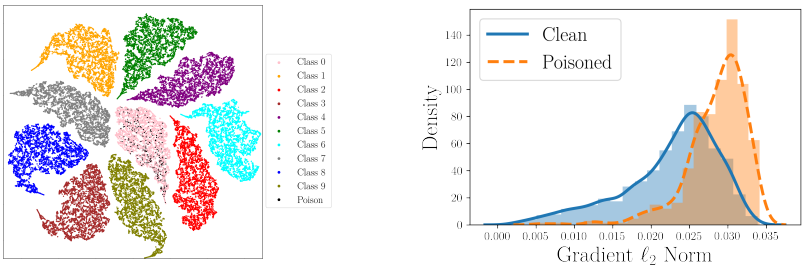
4.2 Clean vs. Poisoned Data Properties

Next, we argue how the gradient space and LID differences between clean and poisoned samples can help us identify poisoned data. Based on these arguments, we will then define a coreset selection objective to effectively filter out such samples from the training set in an online fashion.

Gradient Space Properties. Recently, Hong *et al.* [51] have empirically shown that in the gradient space, poisoned data exhibit different behavior compared to clean samples. Specifically, it is shown that the gradient updates computed on poisoned data have comparably (1) larger ℓ_2 norm and (2) different orientation in contrast to clean data. Given the empirical observations of Hong *et al.* [51], we conclude that the backdoor poisoned data have different gradients compared to the clean data. Moreover, Mirzasoleiman *et al.* [26] studied neural network training with noisy labeled data and argued that in the gradient space, clean data samples tend to be tied with each other. Putting these two observations together, we deduce that the clean data samples are usually tied together in the gradient space while poisoned data is scattered in the gradient space. Thus, we can use this property to find the set of "most centrally located samples" in the gradient space and filter out poisoned data.

We empirically investigate our assumptions around gradient space properties in Fig. 1. Specifically, we visualize the gradients of a randomly initialized neural network for the CIFAR-10 dataset poisoned with the backdoor triggers of Gu *et al.* [6]. To this end, we first show the t-SNE [52] plot of the gradients in Fig. 1a.

[1] The adversary may also change the injected data labels. For notation brevity, we assume that the injection rule only changes the image itself, not the label.



(a) t-SNE [52] plot of a randomly initialized neural network gradient.

(b) Distribution of the neural network gradient norm after 3 epochs of training.

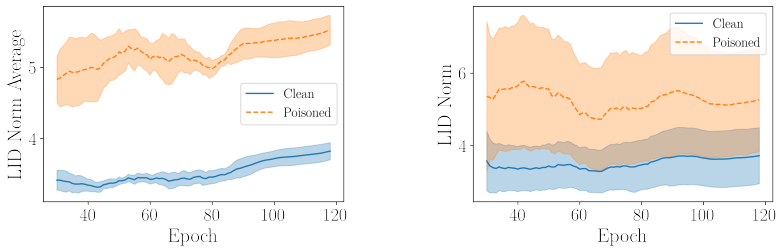
Fig. 1: Visualization of the loss gradients for a neural network on poisoned CIFAR-10 dataset using the last layer approximation of Sec. 4.3. As seen, the gradients of the poisoned data are scattered in the gradient space and tend to exhibit a larger norm.

As can be seen, the poisoned data is scattered in the gradient space. Moreover, as seen in Fig. 1b, the ℓ_2 norm of the gradients tends to be larger for the poisoned data. This case study demonstrates how our assumptions around gradient space properties hold for poisoned data.

Unfortunately, as we will empirically see, after the first few epochs, the gradient dissimilarities between clean and poisoned samples gradually vanish. This is in line with empirical observations of Hong *et al.* [51]. As such, we need an additional regularizer to help us maintain the performance of data filtering throughout training. To this end, we make use of LID.

Local Intrinsic Dimensionality Properties. As discussed above, gradient space properties may not be sufficient to design a robust training algorithm against backdoor data. Thus, we look into the geometrical structure of the underlying data to further enhance the performance of our method. To this end, we utilize the LID of the data [45,53].

Empirically, Ma *et al.* [50] show that neural networks tend to progressively transform the data into subspaces of low local intrinsic dimensionality, such that each clean data point is surrounded by other clean samples. The LID has also been successfully applied to characterize adversarial subspaces [47]. In particular, Ma *et al.* [47] argue how the LID values of adversarially perturbed examples are probably higher than their underlying clean data. This is motivated by the fact that legitimate perturbed examples are likely to be found in a neighborhood where not many clean examples reside, and hence, have high intrinsic dimensionality. For k-NN classifiers, Amsaleg *et al.* [46] provide a rigorous theoretical analysis. It has been proved that if data samples have large LID values, the level of perturbation required to turn them into adversarial examples diminishes. This theoretical study supports the empirical observations of Ma *et al.* [47] that tampered examples exhibit high LID values. These results collectively indicate that manipulating clean data samples will likely increase their LID.



(a) (b)

Fig. 2: The LID of clean and BadNet poisoned data samples for CIFAR-10 dataset. (a) average LID norm across 5 different seeds (b) LID distribution for a single run. As expected, benign data samples have a lower LID compared to poisoned data.

Similarly, one would expect that successful backdoor attacks effectively move a sample away from all clean data. This way, they try to eventually form a subspace secluded from the clean samples that can subvert the neural network's decision into the attacker's target class. Thus, we contemplate that a neighborhood with higher dimensionality is needed to shelter poisoned samples compared to clean data. In other words, clean data samples are expected to have low LID values. In Fig. 2, we empirically show the distribution of LID values for clean and poisoned data recorded during training a neural network on the CIFAR-10 dataset. As expected, poisoned data generally exhibit a higher LID.

4.3 COLLIDER: COreset selection with LocaL Intrinsic DimEnisonality Regularization

In this section, we exploit the properties of clean data discussed in Sec. 4.2 and define our coreset selection objective. As stated in Sec. 4.2, we need to first identify the set of most centrally located samples in the gradient space as the clean data are likely to be tied together. Thus, as our base component, we use a coreset selection objective to find clusters of data that are tied together in the gradient space [26]

$$S^*(\boldsymbol{\theta}) \in \arg\min_{S \subseteq V} \sum_{i \in V} \min_{j \in S} d_{ij}(\boldsymbol{\theta}) \quad \text{s.t.} \quad |S| \leq k. \tag{5}$$

Here, $d_{ij}(\boldsymbol{\theta}) = \|\nabla \ell_i(\boldsymbol{\theta}) - \nabla \ell_j(\boldsymbol{\theta})\|_2$ shows the ℓ_2 distance of loss gradients between samples i and j, and k denotes the number of samples in the coreset.

To add the LID to our approach, we need to ensure that the underlying neural network has reached a point where it can extract meaningful features from the data. Remember from Sec. 3 that this is important as these features are used to approximate the LID values, and thus, they need to have a meaningful structure across data. Therefore, we run our basic coreset selection objective of Eq. (5) during the first epochs until we obtain a steady validation accuracy. Afterward,

we start to consider the LID values. Remember that we want to encourage the selection of samples that are likely to be clean, i.e., have low LID values. Thus, we add a regularizer to our coreset selection objective that promotes this. In particular, Eq. (5) is re-written as

$$S^*(\boldsymbol{\theta}) \in \underset{S\subseteq V,|S|\leq k}{\arg\min} \sum_{i\in V} \min_{j\in S} d_{ij}(\boldsymbol{\theta}) + \lambda \mathrm{LID}\left(\boldsymbol{x}_j\right), \tag{6}$$

where λ is a hyperparameter that determines the relative importance of LID against the gradient term. Here, $\mathrm{LID}\left(\cdot\right)$ is computed as discussed in Sec. 3 using mini-batches of data. As seen in Eq. (6), we encourage the algorithm to select data points with low LID values.

4.4 Practical Considerations

Gradient Approximation and Greedy Solvers. As pointed out in [25,26], finding the optimal solution of Eq. (5) and as a result Eq. (6) is intractable. This is since computing $d_{ij}(\boldsymbol{\theta})$ for all $i \in V$ requires backpropagation over the entire training set, which is computationally prohibitive. More importantly, finding the optimal coresets as in Eq. (5) is shown to be NP-hard [25].

To address the first issue, the following upper-bound is usually used instead of the exact $d_{ij}(\boldsymbol{\theta})$ values [54,25,26]

$$\begin{aligned} d_{ij}(\boldsymbol{\theta}) &= \left\|\nabla \ell_i\left(\boldsymbol{\theta}\right) - \nabla \ell_j\left(\boldsymbol{\theta}\right)\right\|_2 \\ &\leq c_1 \left\| \Sigma_L'\left(z_i^{(L)}\right) \nabla \ell_i^{(L)}(\boldsymbol{\theta}) - \Sigma_L'\left(z_j^{(L)}\right) \nabla \ell_j^{(L)}(\boldsymbol{\theta}) \right\| + c_2, \end{aligned} \tag{7}$$

where $\Sigma_L'\left(z_i^{(L)}\right) \nabla \ell_i^{(L)}(\boldsymbol{\theta})$ denotes the gradient of the loss with respect to the neural network's penultimate layer, shown by $z_i^{(L)}$. Also, c_1 and c_2 are constants, and L denotes the number of DNN layers. This upper-bound can be computed efficiently, with a cost approximately equal to forward-propagation [26]. As for the NP-hardness issue, there exist efficient greedy algorithms that can solve Eq. (5) sub-optimally [55,56,57]. Specifically, Eq. (5) is first turned into its *submodular optimization* equivalent, and then solved. Details of this re-formulation can be found in [25,26] and Appendix A.

Repetitive Coreset Selection. As our coresets are a function of the neural network parameters $\boldsymbol{\theta}$, we need to update them as the training goes on. Thus, at the beginning of every epoch, we first select data samples S by solving Eq. (6). We then train the neural network using mini-batch gradient descent on the data that falls inside the coreset S.

LID Computation. For LID estimation, we select neighbors of each sample from the same class. This way, we ensure that clean samples have similar feature representations among themselves. As a result, our LID estimates can better capture the dimensionality differences between the clean and poisoned samples. To further stabilize LID estimates, we use a moving average of LID values taken

over a few successive epochs, not a single one, as the LID estimate of a training sample. Moreover, as the training evolves, we permanently eliminate samples with the highest LID values. This is motivated by the fact that our LID estimates are the average of LID values for the past epochs. Thus, samples that exhibit a high LID value over successive training epochs are likely to be poisoned data. However, we must be careful not to remove too many samples so that the solution to Eq. (6) becomes trivial. As a result, we select the number of removals such that the coreset selection always has a non-trivial problem to solve. According to Eq. (6), this means that at the last epoch, we need to have at least k samples left in our training set. Thus, we gradually eliminate $n - k$ samples throughout the training. Also, as in Mirzasoleiman *et al.* [26], we use mix-up to enhance the quality of our coresets further. Finally, note that the computational complexity of the LID regularization is minimal. This is because for computing the $d_{ij}(\boldsymbol{\theta})$ terms in Eq. (5) we have already computed the logits. The same values can be re-used for LID calculation. The final algorithm can be found in Appendix A.

5 Experimental Results

This section presents our experimental results. We demonstrate how by using COLLIDER one can train robust neural networks against backdoor data poisonings. Furthermore, we show the role of each component in COLLIDER in our extensive ablation studies.

Settings. We use CIFAR-10 [58], SVHN [59], and ImageNet-12 [1,35] datasets in our experiments. To test our approach against various backdoor data poisonings, we use BadNets [6], label-consistent attacks [31], sinusoidal strips [32], and triggers used in HTBA [37] to poison the aforementioned datasets. We randomly select a target class for each dataset and poison it with backdoor-generated data. The ratio of poisoned samples in the target class is referred to as the *injection rate*. Samples from each backdoor poisoned dataset can be found in Appendix B. Furthermore, we use ResNet [60] as our DNN architecture. For training, we use a stochastic gradient descent (SGD) optimizer with momentum of 0.9 and weight decay 5e-4. The initial learning rate is set to 0.1, which is divided by 10 at epochs 80 and 100 for CIFAR-10 and SVHN (epochs 72 and 144 for ImageNet-12). In what follows, the *coreset size* refers to the ratio of the original data in each class that we intend to keep for training. For more details on the experimental settings and also extra experiments see Appendices B and C.

Baselines. Our approach differs significantly in its assumptions from the current backdoor defenses. Thus, for a fair comparison, we compare COLLIDER to two other baselines: (1) the usual training, denoted as "vanilla", and (2) basic coreset selection (Eq. (5)), denoted as "coresets". Furthermore, to provide a complete picture of the performance of our proposed approach, we also include Spectral Signatures (SS) [18], Activation Clustering (AC) [39], SPECTRE [19],

Table 1: Clean test accuracy (ACC) and attack success rate (ASR) in % for backdoor data poisonings on CIFAR-10 (BadNets and label-consistent) and SVHN (sinusoidal strips) datasets. The results show the mean and standard deviation for 5 different seeds. The poisoned data injection rate is 10%. For BadNets and label-consistent attacks, the coreset size is 0.3. It is 0.4 for sinusoidal strips.

Training	BadNets [6]		Label-consistent [31]		Sinusoidal Strips [32]	
	ACC	ASR	ACC	ASR	ACC	ASR
Vanilla	92.19 ± 0.20	99.98 ± 0.02	92.46 ± 0.16	100	95.79 ± 0.20	77.35 ± 3.68
SS [18]	92.05 ± 0.43	1.18 ± 0.36	92.24 ± 0.35	0.53 ± 0.09	95.38 ± 0.28	77.30 ± 2.50
AC [39]	91.78 ± 0.21	99.87 ± 0.08	91.59 ± 0.31	75.44 ± 42.53	95.45 ± 0.20	77.43 ± 4.59
Spectre [19]	91.28 ± 0.22	98.17 ± 1.97	91.78 ± 0.37	0.51 ± 0.15	95.41 ± 0.12	8.51 ± 7.03
NAD [17]	72.19 ± 1.73	3.55 ± 1.25	70.18 ± 1.70	3.44 ± 1.50	92.41 ± 0.34	6.99 ± 3.02
Coresets	84.86 ± 0.47	74.93 ± 34.6	83.87 ± 0.36	7.78 ± 9.64	92.30 ± 0.19	24.30 ± 8.15
Collider (Ours)	80.66 ± 0.95	4.80 ± 1.49	82.11 ± 0.62	5.19 ± 1.08	89.74 ± 0.31	6.20 ± 3.69

and Neural Attention Distillation (NAD) [17] to our comparisons. It should be noted that these approaches have a different set of assumptions compared to Collider, and the comparisons are not fair. This is because SS [18], AC [39], and Spectre [19] train the neural network *twice*, and NAD [17] is trying to erase the backdoor after the training is complete.

5.1 Clean Test Accuracy and Attack Success Rate

In Tab. 1 we show our experimental results. We measure the performance of each training algorithm by two quantities: (1) accuracy on the clean test set (ACC) and (2) the attack success rate (ASR), which indicates the accuracy of the installed triggers on non-target class images. As seen, using Collider we can reduce the attack success rate significantly in nearly all the cases. Moreover, although the coreset selection based on the gradient space properties can improve robustness (see the "coresets" row), its performance is insufficient to provide a robust model. This, as we see, can be improved using the intrinsic dimensionality of the data that captures the geometrical structure of the training samples.

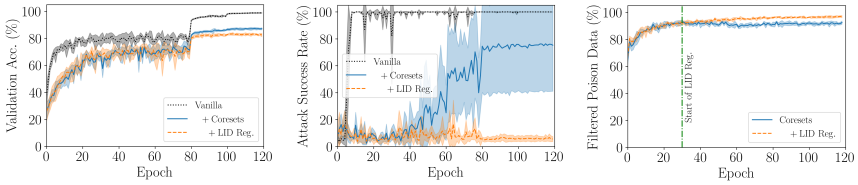
Interestingly, observe how our approach is a middle-ground between SS [18] and NAD [17] settings. Specifically, in SS [18] the entire network is re-trained to get a better performance, but this is inefficient. To show this, in Tab. 2 we report the total training time for our approach against SS [18] that trains the neural network twice. As seen, our approach results in an average of 22% reduction in training time compared to existing methods that need re-training.[2]

At the other end of the spectrum, NAD [17] tries to erase the backdoors from the vanilla network without re-training. In Collider we are trying to prevent the network from learning the triggers in an online fashion, such that there would be no need to re-train the network. More importantly, NAD [17] shows the difficulty of this task because once the model learns the triggers, one cannot erase them without sacrificing too much of the clean accuracy.

[2] Note that here we measure the training time for SS [18] as a representative of techniques that need re-training, e.g., AC [39] and Spectre [19].

Table 2: Total training time (in minutes) for experiments of Tab. 1. The results show the mean and standard deviation for 5 different seeds.

Method	BadNets [6]	Label-consistent [31]	Sinusoidal Strips [32]
Coresets	61.35 ± 0.31	63.09 ± 0.36	66.04 ± 1.11
COLLIDER	62.56 ± 0.13	67.10 ± 0.95	64.53 ± 0.38
SS [18] (∼AC [39]/SPECTRE [19])	85.48 ± 0.28	85.26 ± 0.26	79.46 ± 0.86



(a) (b) (c)

Fig. 3: Ablation study on the effect of coreset selection and LID regularization terms in COLLIDER for training robust models on BadNet poisonings. (a) validation accuracy (b) attack success rate (c) percentage of poisoned data filtered. Epoch 30 (annotated in (c)) is the start of our LID regularization. As seen, while the difference between the percentage of the filtered poison data is very close, this translates to a significant difference in ASR.

As seen in Tab. 1, while COLLIDER can significantly reduce the ASR, it also exhibits a gap in terms of ACC with other methods. As a remedy, we investigate using non-coreset data in training to increase the clean accuracy gap while maintaining backdoor robustness. To this end, we use semi-supervised learning and treat the non-coreset data as unlabeled samples. As shown in Appendix C, one can enhance the ACC while decreasing ASR. For a detailed description of this version of our approach and other extensive experiments, please see Appendix C.

5.2 Ablation Studies

To clarify the role of each component in COLLIDER, we record some performance measures during training. For each algorithm, we record the accuracy on the validation set and the attack success rate on the test set. Also, to better monitor the coresets, we record the percentage of the poisoned data filtered during training. This measure is defined as the complement of the number of poisoned data in the coreset divided by the total number of poisoned data. This ratio is always between zero and one: one means that the selected coreset does not contain any poisoned data. As seen in Fig. 3c, after the first few epochs where the performance of the basic coreset selection improves, it suddenly degrades. At this point, we are ready to enable the LID regularization in COLLIDER since we have reached a point with steady validation accuracy. This addition helps the coreset selection to maintain its performance throughout the training. This

slight difference is decisive when it comes to the attack success rate, which, as seen in Fig. 3b, can determine the robustness of the final model.[3]

5.3 Trade-offs

Finally, there are some trade-offs in COLLIDER that need to be discussed. First, in Fig. 7 in Appendix C one can see that as we increase the coreset size, the clean test accuracy and the attack success rate both increase. This is inevitable as by increasing the coreset size, we are likely to let more poisoned data into the selected subset used for training. As a result, the attack success rate starts to increase. However, we can see that this trade-off is less severe in COLLIDER due to its use of LID that can capture the poisoned samples more accurately. Second, in Fig. 8 of Appendix C we show the performance of our proposed method as the injection rate is varied. For the experiments in this figure, the coreset size is fixed to 0.3 or 30%. Ideally, we should be able to train a robust model with a 70% injection rate successfully. However, as discussed in Sec. 4.2, our solutions to the coreset selection objective of Eq. (6) are always sub-optimal as finding the exact solution is NP-hard. Thus, as the injection rate increases, so does the attack success rate. Again, the rate of this trade-off is less dramatic for COLLIDER compared to gradient-based coreset selection.

6 Conclusion

We proposed a robust, end-to-end training algorithm for backdoor datasets in this paper. Our method, named COLLIDER, exploits the idea of coreset selection to purify the given data before using them for training a neural network. Specifically, we saw how clean and poisoned data samples differ in their gradient space and local intrinsic dimensionality attributes. Based on these differences, we defined a coreset selection algorithm that tries to filter out malicious data effectively. The neural network can then be trained on this carefully selected data. We showed the performance of the proposed approach in improving the robustness of neural networks against backdoor data poisonings and identified the role of each component through extensive ablation studies. While successful in decreasing the attack success rate significantly, we saw that COLLIDER also reduces the clean test accuracy slightly.

Acknowledgments. We thank Michael Houle for thoughtful discussions and comments on LID evaluation. This research was undertaken using the LIEF HPC-GPGPU Facility hosted at the University of Melbourne. This Facility was established with the assistance of LIEF Grant LE170100200. Sarah Erfani is in part supported by Australian Research Council (ARC) Discovery Early Career Researcher Award (DECRA) DE220100680.

[3] Note that the sudden jump observed in the validation accuracy during epochs 80 and 100 is due to the use of a multi-step learning rate scheduler discussed at the beginning of Sec. 5.

References

1. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M.S., Berg, A.C., Li, F.: ImageNet large scale visual recognition challenge. International Journal of Computer Vision **115** (2015) 211–252
2. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the KITTI vision benchmark suite. In: Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2012) 3354–3361
3. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. In: Proceedings of the 4th International Conference on Learning Representations (ICLR). (2016)
4. Deng, J., Guo, J., Xue, N., Zafeiriou, S.: ArcFace: Additive angular margin loss for deep face recognition. In: Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2019) 4690–4699
5. Schroff, F., Kalenichenko, D., Philbin, J.: FaceNet: A unified embedding for face recognition and clustering. In: Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2015) 815–823
6. Gu, T., Dolan-Gavitt, B., Garg, S.: BadNets: Identifying vulnerabilities in the machine learning model supply chain. CoRR **abs/1708.06733** (2017)
7. Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP). (2017) 3–18
8. Liu, K., Dolan-Gavitt, B., Garg, S.: Fine-Pruning: Defending against backdooring attacks on deep neural networks. In: Proceedings of the 21st International Symposium Research in Attacks, Intrusions, and Defenses (RAID). (2018) 273–294
9. Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., Zhao, B.Y.: Neural Cleanse: Identifying and mitigating backdoor attacks in neural networks. In: Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP). (2019) 707–723
10. Chen, H., Fu, C., Zhao, J., Koushanfar, F.: DeepInspect: A black-box trojan detection and mitigation framework for deep neural networks. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI). (2019) 4658–4664
11. Kolouri, S., Saha, A., Pirsiavash, H., Hoffmann, H.: Universal litmus patterns: Revealing backdoor attacks in CNNs. In: Proceedings of the 2020 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2020) 298–307
12. Sikka, K., Sur, I., Jha, S., Roy, A., Divakaran, A.: Detecting trojaned DNNs using counterfactual attributions. CoRR **abs/2012.02275** (2020)
13. Goldblum, M., Tsipras, D., Xie, C., Chen, X., Schwarzschild, A., Song, D., Madry, A., Li, B., Goldstein, T.: Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. CoRR **abs/2012.10544** (2020)
14. Wang, R., Zhang, G., Liu, S., Chen, P., Xiong, J., Wang, M.: Practical detection of trojan neural networks: Data-limited and data-free cases. In: Proceedings of the 16th European Conference on Computer Vision (ECCV). (2020) 222–238
15. Liu, X., Li, F., Wen, B., Li, Q.: Removing backdoor-based watermarks in neural networks with limited data. In: Proceedings of the 25th International Conference on Pattern Recognition (ICPR). (2020) 10149–10156
16. Zhao, P., Chen, P., Das, P., Ramamurthy, K.N., Lin, X.: Bridging mode connectivity in loss landscapes and adversarial robustness. In: Proceedings of the 8th International Conference on Learning Representations (ICLR). (2020)

17. Li, Y., Lyu, X., Koren, N., Lyu, L., Li, B., Ma, X.: Neural attention distillation: Erasing backdoor triggers from deep neural networks. In: Proceedings of the 9th International Conference on Learning Representations (ICLR). (2021)
18. Tran, B., Li, J., Madry, A.: Spectral signatures in backdoor attacks. In: Proceedings of the Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems (NeurIPS). (2018) 8011–8021
19. Hayase, J., Kong, W., Somani, R., Oh, S.: SPECTRE: Defense against backdoor attacks via robust covariance estimation. In: Proceedings of the 38th International Conference on Machine Learning (ICML). (2021) 4129–4139
20. Levine, A., Feizi, S.: Deep partition aggregation: Provable defenses against general poisoning attacks. In: Proceedings of the 9th International Conference on Learning Representations (ICLR). (2021)
21. Jia, J., Cao, X., Gong, N.Z.: Intrinsic certified robustness of bagging against data poisoning attacks. CoRR **abs/2008.04495** (2020)
22. Har-Peled, S., Mazumdar, S.: On coresets for k-means and k-median clustering. In: Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC). (2004) 291–300
23. Agarwal, P.K., Har-Peled, S., Varadarajan, K.R.: Approximating extent measures of points. Journal of the ACM **51** (2004) 606–635
24. Campbell, T., Broderick, T.: Bayesian coreset construction via greedy iterative geodesic ascent. In: Proceedings of the 35th International Conference on Machine Learning (ICML). (2018) 697–705
25. Mirzasoleiman, B., Bilmes, J.A., Leskovec, J.: Coresets for data-efficient training of machine learning models. In: Proceedings of the 37th International Conference on Machine Learning (ICML). (2020) 6950–6960
26. Mirzasoleiman, B., Cao, K., Leskovec, J.: Coresets for robust training of deep neural networks against noisy labels. In: Proceedings of the Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS). (2020)
27. Li, Y., Wu, B., Jiang, Y., Li, Z., Xia, S.: Backdoor learning: A survey. CoRR **abs/2007.08745** (2020)
28. Chen, X., Liu, C., Li, B., Lu, K., Song, D.: Targeted backdoor attacks on deep learning systems using data poisoning. CoRR **abs/1712.05526** (2017)
29. Liu, Y., Ma, S., Aafer, Y., Lee, W., Zhai, J., Wang, W., Zhang, X.: Trojaning attack on neural networks. In: Proceedings of the 25th Annual Network and Distributed System Security Symposium (NDSS). (2018)
30. Shafahi, A., Huang, W.R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., Goldstein, T.: Poison frogs! targeted clean-label poisoning attacks on neural networks. In: Proceedings of the Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems (NeurIPS). (2018) 6106–6116
31. Turner, A., Tsipras, D., Madry, A.: Label-consistent backdoor attacks. CoRR **abs/1912.02771** (2019)
32. Barni, M., Kallas, K., Tondi, B.: A new backdoor attack in CNNS by training set corruption without label poisoning. In: Proceedings of the 2019 IEEE International Conference on Image Processing (ICIP). (2019) 101–105
33. Zhong, H., Liao, C., Squicciarini, A.C., Zhu, S., Miller, D.J.: Backdoor embedding in convolutional neural network models via invisible perturbation. In: Proceedings of the 10th ACM Conference on Data and Application Security and Privacy (CODASPY). (2020) 97–108

34. Li, S., Xue, M., Zhao, B., Zhu, H., Zhang, X.: Invisible backdoor attacks on deep neural networks via steganography and regularization. IEEE Transactions on Dependable and Secure Computing (2020)
35. Liu, Y., Ma, X., Bailey, J., Lu, F.: Reflection backdoor: A natural backdoor attack on deep neural networks. In: Proceedings of the 16th European Conference on Computer Vision (ECCV). (2020) 182–199
36. Nguyen, T.A., Tran, A.T.: WaNet - imperceptible warping-based backdoor attack. In: Proceedings of the 9th International Conference on Learning Representations (ICLR). (2021)
37. Saha, A., Subramanya, A., Pirsiavash, H.: Hidden trigger backdoor attacks. In: Proceedings of the 34th AAAI Conference on Artificial Intelligence. (2020) 11957–11965
38. Gao, Y., Xu, C., Wang, D., Chen, S., Ranasinghe, D.C., Nepal, S.: STRIP: a defence against trojan attacks on deep neural networks. In: Proceedings of the 35th Annual Computer Security Applications Conference (ACSAC). (2019) 113–125
39. Chen, B., Carvalho, W., Baracaldo, N., Ludwig, H., Edwards, B., Lee, T., Molloy, I., Srivastava, B.: Detecting backdoor attacks on deep neural networks by activation clustering. In: Workshop on Artificial Intelligence Safety, 33rd AAAI Conference on Artificial Intelligence. (2019)
40. Peri, N., Gupta, N., Huang, W.R., Fowl, L., Zhu, C., Feizi, S., Goldstein, T., Dickerson, J.P.: Deep k-nn defense against clean-label data poisoning attacks. In: Proceedings of the 16th European Conference on Computer Vision (ECCV) Workshops. (2020) 55–70
41. Jin, K., Zhang, T., Shen, C., Chen, Y., Fan, M., Lin, C., Liu, T.: A unified framework for analyzing and detecting malicious examples of DNN models. CoRR **abs/2006.14871** (2020)
42. Guo, W., Wang, L., Xing, X., Du, M., Song, D.: TABOR: A highly accurate approach to inspecting and restoring trojan backdoors in AI systems. CoRR **abs/1908.01763** (2019)
43. Houle, M.E., Kashima, H., Nett, M.: Generalized expansion dimension. In: 12th IEEE International Conference on Data Mining (ICDM) Workshops. (2012) 587–594
44. Karger, D.R., Ruhl, M.: Finding nearest neighbors in growth-restricted metrics. In: Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC). (2002) 741–750
45. Houle, M.E.: Local intrinsic dimensionality I: an extreme-value-theoretic foundation for similarity applications. In: Proceedings of the 10th International Conference on Similarity Search and Applications SISAP. (2017) 64–79
46. Amsaleg, L., Bailey, J., Barbe, A., Erfani, S.M., Furon, T., Houle, M.E., Radovanovic, M., Nguyen, X.V.: High intrinsic dimensionality facilitates adversarial attack: Theoretical evidence. IEEE Transactions on Information Forensics Security (2021) 854–865
47. Ma, X., Li, B., Wang, Y., Erfani, S.M., Wijewickrema, S.N.R., Schoenebeck, G., Song, D., Houle, M.E., Bailey, J.: Characterizing adversarial subspaces using local intrinsic dimensionality. In: Proceedings of the 6th International Conference on Learning Representations (ICLR). (2018)
48. Levina, E., Bickel, P.J.: Maximum likelihood estimation of intrinsic dimension. In: Proceedings of the Advances in Neural Information Processing Systems 17: Annual Conference on Neural Information Processing Systems (NeurIPS). (2004) 777–784

49. Amsaleg, L., Chelly, O., Furon, T., Girard, S., Houle, M.E., Kawarabayashi, K., Nett, M.: Estimating local intrinsic dimensionality. In: Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (2015) 29–38
50. Ma, X., Wang, Y., Houle, M.E., Zhou, S., Erfani, S.M., Xia, S., Wijewickrema, S.N.R., Bailey, J.: Dimensionality-driven learning with noisy labels. In: Proceedings of the 35th International Conference on Machine Learning (ICML). (2018) 3361–3370
51. Hong, S., Chandrasekaran, V., Kaya, Y., Dumitras, T., Papernot, N.: On the effectiveness of mitigating data poisoning attacks with gradient shaping. CoRR **abs/2002.11497** (2020)
52. Van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. Journal of Machine Learning Research (JMLR) **9** (2008)
53. Houle, M.E.: Local intrinsic dimensionality II: multivariate analysis and distributional support. In: Proceedings of the 10th International Conference on Similarity Search and Applications SISAP. (2017) 80–95
54. Katharopoulos, A., Fleuret, F.: Not all samples are created equal: Deep learning with importance sampling. In: Proceedings of the 35th International Conference on Machine Learning (ICML). (2018) 2530–2539
55. Minoux, M.: Accelerated greedy algorithms for maximizing submodular set functions. In: Optimization Techniques. Springer (1978) 234–243
56. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions - I. Mathematical Programming **14** (1978) 265–294
57. Wolsey, L.A.: An analysis of the greedy algorithm for the submodular set covering problem. Combinatorica **2** (1982) 385–393
58. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto (2009)
59. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning. (2011)
60. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2016) 770–778
61. Berthelot, D., Carlini, N., Goodfellow, I.J., Papernot, N., Oliver, A., Raffel, C.: MixMatch: A holistic approach to semi-supervised learning. In: Proceedings of the Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems (NeurIPS). (2019) 5050–5060
62. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Proceedings of the 3rd International Conference on Learning Representations (ICLR). (2015)