ℓ_∞-Robustness and Beyond: Unleashing Efficient Adversarial Training

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. Neural networks are vulnerable to adversarial attacks: adding well-crafted, imperceptible perturbations to their input can modify their output. Adversarial training is one of the most effective approaches in training robust models against such attacks. However, it is much slower than vanilla training of neural networks since it needs to construct adversarial examples for the entire training data at every iteration, hampering its effectiveness. Recently, *Fast Adversarial Training* (FAT) was proposed that can obtain robust models efficiently. However, the reasons behind its success are not fully understood, and more importantly, it can only train robust models for ℓ_∞-bounded attacks as it uses FGSM during training. In this paper, by leveraging the theory of coreset selection, we show how selecting a small subset of training data provides a *general*, more principled approach toward reducing the time complexity of robust training. Unlike existing methods, our approach can be adapted to a wide variety of training objectives, including TRADES, ℓ_p-PGD, and Perceptual Adversarial Training (PAT). Our experimental results indicate that our approach speeds up adversarial training by 2-3 times while experiencing a slight reduction in the clean and robust accuracy.

Keywords: adversarial training, coreset selection, efficient training.

1 Introduction

Neural networks have achieved great success in the past decade. Today, they are one of the primary candidates in solving a wide variety of machine learning tasks, from object detection and classification [12,42] to photo-realistic image generation [14,38] and beyond. Despite their impressive performance, neural networks are vulnerable to adversarial attacks [3,35]: adding well-crafted, imperceptible perturbations to their input can change their output. This unexpected behavior of neural networks prevents their widespread deployment in safety-critical applications, including autonomous driving [8] and medical diagnosis [24]. As such, training robust neural networks against adversarial attacks is of paramount importance and has gained lots of attention.

Adversarial training is one of the most successful approaches in defending neural networks against adversarial attacks.[1] This approach first constructs a perturbed version of the training data. Then, the neural network is optimized on these perturbed inputs instead of the clean samples. This procedure must be done iteratively as the perturbations depend on the neural network weights. Since the weights are optimized during training, the perturbations must also be adjusted for each data sample in every iteration.

Various adversarial training methods primarily differ in how they define and find the perturbed version of the input [25,44,22]. However, they all require repetitive construction of these perturbations during training which is often cast as another non-linear optimization problem. As such, the time and computational complexity of adversarial training is massively higher than vanilla training. In practice, neural networks require massive amounts of training data [1] and need to be trained multiple times with various hyper-parameters to get their best performance [16]. Thus, reducing the time/computational complexity of adversarial training is critical in enabling the environmentally efficient application of robust neural networks in real-world scenarios [33,34].

Fast Adversarial Training (FAT) [41] is a successful approach proposed for efficient training of robust neural networks. Contrary to the common belief that building the perturbed versions of the inputs using *Fast Gradient Sign Method* (FGSM) [10] does not help in training arbitrary robust models [36,25], Wong *et al.* [41] show that by carefully applying uniformly random initialization before the FGSM step one can make this training approach work. Using FGSM to generate the perturbed input in a single step combined with implementation tricks such as mixed precision and cyclic learning rate, FAT can significantly reduce the training time of robust neural networks.

Despite its success, FAT may exhibit unexpected behavior in different settings. For instance, it was shown that FAT suffers from *catastrophic overfitting* where the robust accuracy during training suddenly drops to 0% [41,2]. A more fundamental issue with FAT and its variations such as `GradAlign` [2] is that they are specifically designed and implemented for ℓ_∞ adversarial training. This is because FGSM, particularly an ℓ_∞ perturbation generator, is at the heart of these methods. As a result, the quest for a unified, systematic approach that can reduce the time complexity of all types of adversarial training is not over.

Motivated by the limited scope of FAT, in this paper we take an important step towards finding a general yet principled approach for reducing the time complexity of adversarial training. We notice that repetitive construction of adversarial examples for each data point is the main bottleneck of robust training. While this process needs to be done iteratively, we speculate that perhaps we

[1] Note that adversarial training in the literature generally refers to a particular approach proposed by Madry *et al.* [25]. For the purposes of this paper, we refer to any method that builds adversarial attacks around the training data and incorporates them into the training of the neural network as adversarial training. Using this taxonomy, methods such as TRADES [44], ℓ_p-PGD [25] or Perceptual Adversarial Training (PAT) [22] are all considered different versions of adversarial training.

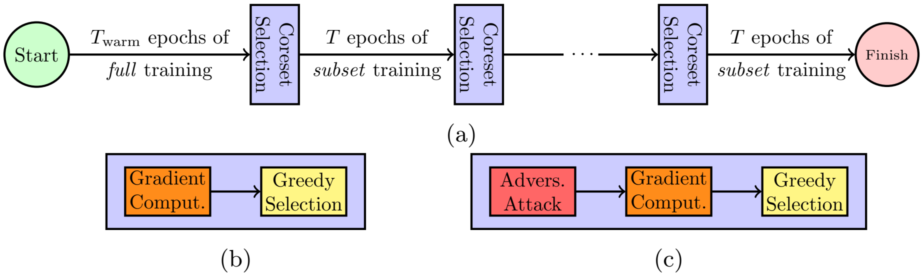

Fig. 1: Overview of neural network training using coreset selection. (a) Selection is done every T epochs. During the next episodes, the network is only trained on this subset. (b) Coreset selection module for vanilla training. (c) Coreset selection module for adversarial training.

can find a subset of the training data that is more important to robust network optimization than the rest. Specifically, we ask the following research question: *Can we train an adversarially robust neural network using a subset of the entire training data without sacrificing clean or robust accuracy?*

In this paper, we show that the answer to this question is affirmative: by selecting a *weighted* subset of the data based on the neural network state, we run *weighted* adversarial training only on this selected subset. We draw an elegant connection between adversarial training and adaptive coreset selection algorithms to achieve this goal. In particular, we use Danskin's theorem and demonstrate how the entire training data can effectively be approximated with an informative weighted subset. To conduct this selection, our study shows that one needs to build adversarial examples for the entire training data and solve a respective subset selection objective. Afterward, training can be performed on this selected subset of the training data. In our approach, shown in Fig. 1, adversarial coreset selection is only required every few epochs, effectively reducing the training time of robust learning algorithms. We demonstrate how our proposed method can be used as a general framework in conjunction with different adversarial training objectives, opening the door to a more principled approach for efficient training of robust neural networks in a general setting. Our experimental results show that one can reduce the training time of various robust training objectives by 2-3 times without sacrificing too much clean or robust accuracy. In summary, we make the following contributions:

– We propose a practical yet principled algorithm for efficient training of robust neural networks based on adaptive coreset selection. To the best of our knowledge, we are the first to use coreset selection in adversarial training.
– We show that our approach can be applied to a variety of robust learning objectives, including TRADES [44], ℓ_p-PGD [25] and Perceptual [22] Adversarial Training. Our approach encompasses a broader range of robust models compared to the limited scope of the existing methods.

- Through extensive experiments, we show that the proposed approach can result in a 2-3 fold reduction of the training time, with only a slight reduction in the clean and robust accuracy.

2 Background and Related Work

2.1 Adversarial Training

Let $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n} \subset \mathbb{X} \times \mathbb{C}$ denote a training dataset consisting of n i.i.d. samples. Each data point contains an input data $\boldsymbol{x}_i$ from domain $\mathbb{X}$ and an associated label y_i taking one of k possible values $\mathbb{C} = [k] = \{1, 2, \dots, k\}$. Without loss of generality, in this paper we focus on the image domain $\mathbb{X}$. Furthermore, assume that $f_{\boldsymbol{\theta}} : \mathbb{X} \to \mathbb{R}^k$ denotes a neural network classifier with parameters $\boldsymbol{\theta}$ that takes $\boldsymbol{x} \in \mathbb{X}$ as input and maps it to a logit value $f_{\boldsymbol{\theta}}(\boldsymbol{x}) \in \mathbb{R}^k$. Then, training a neural network in its most general format can be written as the following minimization problem:

$$\min_{\boldsymbol{\theta}} \sum_{i \in V} \boldsymbol{\Phi}\left(\boldsymbol{x}_i, y_i; f_{\boldsymbol{\theta}}\right), \tag{1}$$

Here, $\boldsymbol{\Phi}\left(\boldsymbol{x}, y; f_{\boldsymbol{\theta}}\right)$ is a function that takes a data point $(\boldsymbol{x}, y)$ and a function $f_{\boldsymbol{\theta}}$ as its inputs, and its output is a measure of discrepancy between the input $\boldsymbol{x}$ and its ground-truth label y. Also, $V = [n] = \{1, 2, \dots, n\}$ denotes the entire training data. By writing the training objective in this format, we can denote both vanilla and adversarial training using the same notation. Below we show how various choices of the function $\boldsymbol{\Phi}$ amount to different training objectives.

Vanilla Training. In case of vanilla training, the function $\boldsymbol{\Phi}$ is a simple evaluation of an appropriate loss function over the neural network output $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ and the ground-truth label y. For instance, for vanilla training we can have:

$$\boldsymbol{\Phi}\left(\boldsymbol{x}, y; f_{\boldsymbol{\theta}}\right) = \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right), \tag{2}$$

where $\mathcal{L}_{\mathrm{CE}}(\cdot, \cdot)$ is the cross-entropy loss.

FGSM, $\boldsymbol{\ell_p}$-PGD, and Perceptual Adversarial Training. In these cases, the training objective is itself an optimization problem:

$$\boldsymbol{\Phi}\left(\boldsymbol{x}, y; f_{\boldsymbol{\theta}}\right) = \max_{\tilde{\boldsymbol{x}}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}), y\right) \text{ s.t. } \mathrm{d}\left(\tilde{\boldsymbol{x}}, \boldsymbol{x}\right) \leq \varepsilon \tag{3}$$

where $\mathrm{d}(\cdot, \cdot)$ is an appropriate distance measure over image domain $\mathbb{X}$, and ε denotes a scalar. The constraint over $\mathrm{d}(\tilde{\boldsymbol{x}}, \boldsymbol{x})$ is used to ensure visual similarity between $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}$. Solving Eq. (3) amounts to finding an adversarial example $\tilde{\boldsymbol{x}}$ for the clean sample $\boldsymbol{x}$ [25]. Different choices of the visual similarity measure $\mathrm{d}(\cdot, \cdot)$ and solvers for Eq. (3) result in different adversarial training objectives.

- FGSM [10] assumes that $\mathrm{d}(\tilde{\boldsymbol{x}}, \boldsymbol{x}) = \left\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\right\|_{\infty}$. Using this ℓ_∞ assumption, the solution to Eq. (3) is computed using one iteration of gradient ascent.

- ℓ_p-PGD [25] utilizes ℓ_p norms as a proxy for visual similarity $\mathrm{d}(\cdot,\cdot)$. Then, several steps of projected gradient ascent is taken to solve Eq. (3).
- Perceptual Adversarial Training (PAT) [22] replaces $\mathrm{d}(\cdot,\cdot)$ with *Learned Perceptual Image Patch Similarity* (LPIPS) distance [45]. Then, Laidlaw *et al.* [22] propose to solve this maximization objective using either projected gradient ascent or Lagrangian relaxation.

TRADES Adversarial Training. This approach uses a combination of Eqs. (2) and (3). The intuition behind TRADES [44] is to create a trade-off between clean and robust accuracy. In particular, the objective is written as:

$$\boldsymbol{\Phi}\left(\boldsymbol{x}, y; f_{\boldsymbol{\theta}}\right) = \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) + \max_{\tilde{\boldsymbol{x}}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}), f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)/\lambda, \tag{4}$$

such that $\mathrm{d}(\tilde{\boldsymbol{x}}, \boldsymbol{x}) \leq \varepsilon$. Here, λ is a coefficient that controls the trade-off.

2.2 Coreset Selection

Adaptive data subset selection, and *coreset selection* in general, is concerned with finding a weighted subset of the data that can approximate specific attributes of the entire population [9]. Traditionally, coreset selection has been used for different machine learning tasks such as k-means and k-medians [11], Naïve Bayes and nearest neighbor classifiers [39], and Bayesian inference [4].

Recently, coreset selection algorithms are being developed for neural network training [27,28,17,16]. The main idea behind such methods is to approximate the full gradient using a weighted subset of the training data. These algorithms start with computing the gradient of the loss function with respect to the neural network weights. This gradient is computed for *every* data sample in the training set. Then, a selection criterion is formed. This criterion aims to find a *weighted subset* of the training data that can approximate the full gradient. In Sec. 3 we provide a detailed account of these methods.

Existing coreset selection algorithms can only be used for the vanilla training of neural networks. As such, they still suffer from adversarial vulnerability. This paper extends coreset selection algorithms to robust neural network training and shows how they can be adopted to various robust training objectives.

3 Proposed Method

As discussed in Sec. 1, the main bottleneck in the time/computational complexity of adversarial training stems from constructing adversarial examples for the entire training set at each epoch. FAT [41] tries to eliminate this issue by using FGSM as its adversarial example generator. However, this simplification 1) may lead to catastrophic overfitting [41,2], and 2) is not easy to generalize to all types of adversarial training as FGSM is designed explicitly for ℓ_∞ attacks.

Instead of using a faster adversarial example generator, here we take a different, *orthogonal* path and try to reduce the training set size effectively. This way, the original adversarial training algorithm can still be used on this smaller subset of training data. This approach can reduce the training time while optimizing a similar objective as the original training. In this sense, it leads to a more *unified* method that can be used along with various types of adversarial training objectives, including the ones that already exist and the ones that will be proposed in the future.

The main hurdle in materializing this idea is the following question: *How should we select this subset of the training data without hurting either the clean or robust accuracy?* To answer this question, we propose to use coreset selection on the training data to reduce the sample size and improve training efficiency.

3.1 Problem Statement

Using our general notation from Sec. 2.1, we write both vanilla and adversarial training using the same objective:

$$\min_{\boldsymbol{\theta}} \sum_{i \in V} \boldsymbol{\Phi}\left(\boldsymbol{x}_i, y_i; f_{\boldsymbol{\theta}}\right), \tag{5}$$

where V denotes the entire training data, and depending on the training task, $\boldsymbol{\Phi}\left(\boldsymbol{x}_i, y_i; f_{\boldsymbol{\theta}}\right)$ takes any of the Eqs. (2) to (4) forms. We adopt this notation to make our analysis more accessible.

As discussed in Sec. 2.2, coreset selection can be seen as a two-step process. First, the gradient of the loss function with respect to the neural network weights is computed for each training sample. Then, based on the gradients obtained in step one, a weighted subset (a.k.a. the coreset) of the training data is formed (see Fig. 1b). This subset is obtained such that the weighted gradients of the samples inside the coreset can provide a good approximation of the full gradient.

Specifically, using our universal notation in Eq. (5), we write coreset selection for both vanilla and adversarial training as:

$$\min_{S \subseteq V, \gamma} \left\| \sum_{i \in V} \nabla_{\boldsymbol{\theta}} \boldsymbol{\Phi}\left(\boldsymbol{x}_i, y_i; f_{\boldsymbol{\theta}}\right) - \sum_{j \in S} \gamma_j \nabla_{\boldsymbol{\theta}} \boldsymbol{\Phi}\left(\boldsymbol{x}_j, y_j; f_{\boldsymbol{\theta}}\right) \right\|, \tag{6}$$

where $S \subseteq V$ is the coreset, and γ_j's are the weights of each sample in the coreset. Once the coreset S is found, instead of training the neural network using Eq. (5), we can optimize its parameters using a weighted objective over the coreset:

$$\min_{\boldsymbol{\theta}} \sum_{j \in S} \gamma_j \boldsymbol{\Phi}\left(\boldsymbol{x}_j, y_j; f_{\boldsymbol{\theta}}\right). \tag{7}$$

It can be shown that solving Eq. (6) is NP-hard [27,28]. Roughly, various coreset selection methods differ in how they approximate the solution of the aforementioned objective. For instance, Craig [27] casts this objective as a *submodular set cover problem* and uses existing greedy solvers to get an approximate solution. As another example, GradMatch [16] analyzes the convergence of stochastic gradient descent using adaptive data subset selection.

Based on this study, Killamsetty *et al.* [16] propose to use Orthogonal Matching Pursuit (OMP) [31,7] as a greedy solver of the data selection objective. More information about these methods is provided in Appendix A.

The issue with the aforementioned coreset selection methods is that they are designed explicitly for vanilla training of neural networks (see Fig. 1b), and they do not reflect the requirements of adversarial training. As such, we should modify these methods to make them suitable for our purpose of robust neural network training. Meanwhile, we should also consider the fact that the field of coreset selection is still evolving. Thus, we aim to find a general modification that can later be used alongside newer versions of greedy coreset selection algorithms.

We notice that various coreset selection methods proposed for vanilla neural network training only differ in their choice of greedy solvers. Therefore, we narrow down the changes we want to make to the first step of coreset selection: gradient computation. Then, existing greedy solvers can be used to find the subset of training data that we are looking for. To this end, we draw a connection between coreset selection methods and adversarial training using Danskin's theorem, as outlined next. Our analysis shows that for adversarial coreset selection, one needs to add a pre-processing step where adversarial attacks for the raw training data need to be computed (see Fig. 1c).

3.2 Coreset Selection for Efficient Adversarial Training

As discussed above, to construct the Eq. (6) objective, we need to compute the loss gradient with respect to the neural network weights. Once done, we can use existing greedy solvers to find the solution. The gradient computation needs to be performed for the entire training set. In particular, using our notation from Sec. 2.1, this step can be written as:

$$\nabla_{\boldsymbol{\theta}} \boldsymbol{\Phi}\left(\boldsymbol{x}_i, y_i; f_{\boldsymbol{\theta}}\right) \quad \forall \quad i \in V, \tag{8}$$

where V denotes the training set.

For vanilla neural network training (see Sec. 2.1) the above gradient is simply equal to $\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i\right)$ which can be computed using standard backpropagation. In contrast, for the adversarial training objectives in Eqs. (3) and (4), this gradient requires taking partial derivative of a maximization objective. To this end, we use the famous Dasnkin's theorem [6] as stated below.

Theorem 1 (Theorem A.1 [25]). *Let $\mathcal{S}$ be a nonempty compact topological space, $\ell : \mathbb{R}^m \times \mathcal{S} \to \mathbb{R}$ be such that $\ell(\cdot, \boldsymbol{\delta})$ is differentiable for every $\boldsymbol{\delta} \in \mathcal{S}$, and $\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}, \boldsymbol{\delta})$ is continuous on $\mathbb{R}^m \times \mathcal{S}$. Also, let $\boldsymbol{\delta}^*(\boldsymbol{\theta}) = \{\boldsymbol{\delta} \in \arg\max_{\boldsymbol{\delta} \in \mathcal{S}} \ell(\boldsymbol{\theta}, \boldsymbol{\delta})\}$. Then, the corresponding max-function $\phi(\boldsymbol{\theta}) = \max_{\boldsymbol{\delta} \in \mathcal{S}} \ell(\boldsymbol{\theta}, \boldsymbol{\delta})$ is locally Lipschitz continuous, directionally differentiable, and its directional derivatives along vector $\boldsymbol{h}$ satisfy:*

$$\phi'(\boldsymbol{\theta}, \boldsymbol{h}) = \sup_{\boldsymbol{\delta} \in \boldsymbol{\delta}^*(\boldsymbol{\theta})} \boldsymbol{h}^\top \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}, \boldsymbol{\delta}).$$

In particular, if for some $\boldsymbol{\theta} \in \mathbb{R}^m$ *the set* $\boldsymbol{\delta}^*(\boldsymbol{\theta}) = \{\boldsymbol{\delta}^*_{\boldsymbol{\theta}}\}$ *is a singleton, then the max-function is differentiable at* $\boldsymbol{\theta}$ *and*

$$\nabla \phi(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \ell \left(\boldsymbol{\theta}, \boldsymbol{\delta}^*_{\boldsymbol{\theta}}\right).$$

In summary, Theorem 1 indicates how to take the gradient of a max-function. To this end, it suffices to 1) find the maximizer, and 2) evaluate the normal gradient at this point.

Now that we have stated Danskin's theorem, we are ready to show how it can provide the connection between coreset selection and the adversarial training objectives of Eqs. (3) and (4). We do this for the two cases of adversarial training and TRADES as outlined next.

Case 1. (ℓ_p-PGD and Perceptual Adversarial Training) Going back to Eq. (8), we know that to perform coreset selection, we need to compute this gradient term for our objective in Eq. (3). In other words, we need to compute:

$$\nabla_{\boldsymbol{\theta}} \boldsymbol{\Phi}\left(\boldsymbol{x}, y; f_{\boldsymbol{\theta}}\right) = \nabla_{\boldsymbol{\theta}} \max_{\tilde{\boldsymbol{x}}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}), y\right) \tag{9}$$

under the constraint $\mathrm{d}\left(\tilde{\boldsymbol{x}}, \boldsymbol{x}\right) \leq \varepsilon$ for every training sample. Based on Danskin's theorem, we can deduce:

$$\nabla_{\boldsymbol{\theta}} \boldsymbol{\Phi}\left(\boldsymbol{x}, y; f_{\boldsymbol{\theta}}\right) = \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}^*), y\right), \tag{10}$$

where $\boldsymbol{x}^*$ is the solution to:

$$\arg\max_{\tilde{\boldsymbol{x}}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}), y\right) \quad \text{s.t.} \quad \mathrm{d}\left(\tilde{\boldsymbol{x}}, \boldsymbol{x}\right) \leq \varepsilon. \tag{11}$$

The conditions under which Danskin's theorem hold might not be satisfied for neural networks in general. This is due to the presence of functions with discontinuous gradients, such as ReLU activation, in neural networks. More importantly, finding the exact solution of Eq. (11) is not straightforward as neural networks are highly non-convex. Usually, the exact solution $\boldsymbol{x}^*$ is replaced with its approximation, which is an adversarial example generated under the Eq. (11) objective [18]. Based on this approximation, we can re-write Eq. (10) as:

$$\nabla_{\boldsymbol{\theta}} \boldsymbol{\Phi}\left(\boldsymbol{x}, y; f_{\boldsymbol{\theta}}\right) \approx \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_{\mathrm{adv}}), y\right). \tag{12}$$

In other words, to perform coreset selection for ℓ_p-PGD [25] and Perceptual [22] Adversarial Training, one needs to add a pre-processing step to the gradient computation. At this step, adversarial examples for the entire training set must be constructed. Then, the coresets can be built as in vanilla neural networks.

Case 2. (TRADES Adversarial Training) For TRADES [44], the gradient computation is slightly different as the objective in Eq. (4) consists of two terms. In this case, the gradient can be written as:

$$\nabla_{\boldsymbol{\theta}} \boldsymbol{\Phi}\left(\boldsymbol{x}, y; f_{\boldsymbol{\theta}}\right) = \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) + \nabla_{\boldsymbol{\theta}} \max_{\tilde{\boldsymbol{x}}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}), f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)/\lambda, \tag{13}$$

where $\mathrm{d}(\tilde{\boldsymbol{x}}, \boldsymbol{x}) \leq \varepsilon$. The first term is the normal gradient of the neural network. For the second term, we apply Danskin's theorem to obtain:

$$\nabla_{\boldsymbol{\theta}} \boldsymbol{\Phi}\left(\boldsymbol{x}, y ; f_{\boldsymbol{\theta}}\right) \approx \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right)+\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{\mathrm{adv}}\right), f_{\boldsymbol{\theta}}(\boldsymbol{x})\right) / \lambda, \tag{14}$$

where $\boldsymbol{x}_{\mathrm{adv}}$ is an approximate solution to:

$$\underset{\tilde{\boldsymbol{x}}}{\arg \max } \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}), f_{\boldsymbol{\theta}}(\boldsymbol{x})\right) / \lambda \quad \text{s.t.} \quad \mathrm{d}(\tilde{\boldsymbol{x}}, \boldsymbol{x}) \leq \varepsilon. \tag{15}$$

Having found the loss gradients $\nabla_{\boldsymbol{\theta}} \boldsymbol{\Phi}\left(\boldsymbol{x}_i, y_i ; f_{\boldsymbol{\theta}}\right)$ for ℓ_p-PGD, PAT (Case 1), and TRADES (Case 2), we can construct Eq. (6) and use existing greedy solvers like Craig [27] or GradMatch [16] to find the coreset. As we saw, adversarial coreset selection requires adding a pre-processing step where we need to build perturbed versions of the training data using their respective objectives in Eqs. (11) and (15). Then, the gradients are computed using Eqs. (12) and (14). Afterward, greedy subset selection algorithms are used to construct the coresets based on the value of the gradients. Finally, having selected the coreset data, one can run *weighted* adversarial training only on the data that remains in the coreset. As can be seen, we are not changing the essence of the training objective in this process. We are just reducing the dataset size to enhance our proposed solution's computational efficiency; as such, we can use it along with any adversarial training objective.

3.3 Practical Considerations

Since coreset selection depends on the current values of the neural network weights, it is important to update the coresets as the training evolves. Prior work [17,16] has shown that this selection needs to be done every T epochs, where T is usually greater than 15. Also, we employ small yet crucial practical changes while using coreset selection to increase efficiency. We summarize these practical tweaks below. Further detail can be found in [16,27].

Gradient Approximation. As we saw, both Eqs. (12) and (14) require computation of the loss gradient with respect to the neural network weights. This is equal to backpropagation through the entire neural network, which is not very efficient. Instead, it is common to replace the exact gradients in Eqs. (12) and (14) with their last-layer approximation [15,27,16]. In other words, instead of backpropagating through the entire network, one can backpropagate up until the penultimate layer. This estimate has an approximate complexity equal to forwardpropagation, and it has been shown to work well in practice [27,28,17,16].

Batch-wise Coreset Selection. As discussed in Sec. 3.2, data selection is usually done in a *sample-wise* fashion where each data sample is separately considered to be selected. This way, one must find the data candidates from the entire training set. To increase efficiency, Killamsetty *et al.* [16] proposed the *batch-wise* variant. In this type of coreset selection, the data is first split into several batches. Then, the algorithm makes a selection out of these batches. Intuitively, this change increases efficiency as the sample size is reduced from the number of data points to the number of batches.

Warm-start with the Entire Data. Finally, we warm-start the training using the entire dataset. Afterward, coreset selection is activated, and training is only performed using the data in the coreset.

Final Algorithm Fig. 1 and Alg. 1 in Appendix B.1 summarize our coreset selection approach for adversarial training. As can be seen, our proposed method is a generic and principled approach in contrast to existing methods such as FAT [41]. In particular, our approach provides the following advantages compared to existing methods:

1. The proposed approach does not involve algorithmic level manipulations and dependency on specific training attributes such as ℓ_∞ bound or cyclic learning rate. Also, it controls the training speed through coreset size, which can be specified solely based on available computational resources.
2. The simplicity of our method makes it compatible with any existing/future adversarial training objectives. Furthermore, as we will see in Sec. 4, our approach can be combined with any greedy coreset selection algorithms to deliver robust neural networks.

These characteristics increase the likelihood of applying our proposed method for robust neural network training no matter the training objective. This contrasts with existing methods that solely focus on a particular training objective.

4 Experimental Results

In this section, we present our experimental results.[2] We show how our proposed approach can efficiently reduce the training time of various robust objectives in different settings. To this end, we train neural networks using TRADES [44], ℓ_p-PGD [25] and PAT [22] on CIFAR-10 [19], SVHN [30], and a subset of ImageNet [32] with 12 classes. For TRADES and ℓ_p-PGD training, we use ResNet-18 [12] classifiers, while for PAT we use ResNet-50 architectures.

4.1 TRADES and ℓ_p-PGD Robust Training

In our first experiments, we train ResNet-18 classifiers on CIFAR-10 and SVHN datasets using TRADES, ℓ_∞ and ℓ_2-PGD adversarial training objectives. In each case, we set the training hyper-parameters such as the learning rate, the number of epochs, and attack parameters. Then, we train the network using the entire training data and our adversarial coreset selection approach. For our approach, we use batch-wise versions of Craig [27] and GradMatch [16] with warm-start.

[2] Our implementation can be found in this repository.

Table 1: Clean (ACC) and robust (RACC) accuracy, and total training time (T) of different adversarial training methods. For each objective, all the hyper-parameters were kept the same as full training. For our proposed approach, the difference with full training is shown in parentheses. The results are averaged over 5 runs. More detail can be found in Appendix C.

Objec.	**Data**	**Training Method**	**Performance Measures**		
			↑ **ACC** (%)	↑ **RACC** (%)	↓ **T** (mins)
TRADES	**CIFAR-10**	Adv. Craig (Ours)	83.03 (−2.38)	41.45 (−2.74)	179.20 (−165.09)
		Adv. GradMatch (Ours)	83.07 (−2.34)	41.52 (−2.67)	178.73 (−165.56)
		Full Adv. Training	85.41	44.19	344.29
ℓ_∞-PGD	**CIFAR-10**	Adv. Craig (Ours)	80.37 (−2.77)	45.07 (+3.68)	148.01 (−144.86)
		Adv. GradMatch (Ours)	80.67 (−2.47)	45.23 (+3.84)	148.03 (−144.84)
		Full Adv. Training	83.14	41.39	292.87
ℓ_2-PGD	**SVHN**	Adv. Craig (Ours)	95.42 (+0.10)	49.68 (−3.34)	130.04 (−259.42)
		Adv. GradMatch (Ours)	95.57 (+0.25)	50.41 (−2.61)	125.53 (−263.93)
		Full Adv. Training	95.32	53.02	389.46

We set the *coreset size* (the percentage of training data to be selected) to *50%* for CIFAR-10 and *30%* for SVHN to get a reasonable balance between accuracy and training time. We report the clean and robust accuracy (in %) as well as the total training time (in minutes) in Tab. 1. For our approach, we also report the difference with full training in parentheses. In each case, we evaluate the robust accuracy using an attack with similar attributes as the training objective (for more information, see Appendix C).

As seen, in all cases, we reduce the training time by more than a factor of two while keeping the clean and robust accuracy almost intact. Note that in these experiments, all the training attributes such as the hyper-parameters, learning rate scheduler, etc. are the same among different training schemes. This is important since we want to clearly show the relative boost in performance that one can achieve just by using coreset selection. Nonetheless, it is likely that by tweaking the hyper-parameters of our approach, one can obtain even better results in terms of clean and robust accuracy.

4.2 Perceptual Adversarial Training vs. Unseen Attacks

As discussed in Sec. 2, PAT [22] replaces the visual similarity measure $\mathrm{d}(\cdot, \cdot)$ in Eq. (3) with LPIPS [45] distance. The logic behind this choice is that ℓ_p norms can only capture a small portion of images similar to the clean one, limiting the search space of adversarial attacks. Motivated by this reason, Laidlaw *et al.* [22] propose two different ways of finding the solution to Eq. (3) when $\mathrm{d}(\cdot, \cdot)$ is the

Table 2: Clean (ACC) and robust (RACC) accuracy and total training time (T) of Perceptual Adversarial Training for CIFAR-10 and ImageNet-12 datasets. At inference, the networks are evaluated against five attacks that were not seen during training (Unseen RACC) and different versions of Perceptual Adversarial Attack (Seen RACC). In each case, the average is reported. For more information and details about the experiment, please see the Appendices C and D.

Data	**Training Method**	↑ **ACC** (%)	↑ **RACC** (%)		↓ **T** (mins)
			Unseen	Seen	
CIFAR-10	Adv. CRAIG (Ours)	83.21 (−2.81)	46.55 (−1.49)	13.49 (−1.83)	767.34 (−915.60)
	Adv. GRADMATCH (Ours)	83.14 (−2.88)	46.11 (−1.93)	13.74 (−1.54)	787.26 (−895.68)
	Full PAT (Fast-LPA)	86.02	48.04	15.32	1682.94
ImageNet	Adv. CRAIG (Ours)	86.99 (−4.23)	53.05 (−0.18)	22.56 (−0.77)	2817.06 (−2796.06)
	Adv. GRADMATCH (Ours)	87.08 (−4.14)	53.17 (−0.06)	20.74 (−2.59)	2865.72 (−2747.40)
	Full PAT (Fast-LPA)	91.22	53.23	23.33	5613.12

LPIPS distance. The first version uses PGD, and the second is a relaxation of the original problem using the Lagrangian form. We refer to these two versions as PPGD (Perceptual PGD) and LPA (Lagrangian Perceptual Attack), respectively. Then, Laidlaw *et al.* [22] proposed to utilize a fast version of LPA to enable its efficient usage in adversarial training.

For our next set of experiments, we show how our approach can be adapted to this unusual training objective. This is done to showcase the compatibility of our proposed method with different training objectives as opposed to existing methods that are carefully tuned for a particular training objective. To this end, we train ResNet-50 classifiers using Fast-LPA. We train the classifiers on CIFAR-10 and ImageNet-12 datasets. Like our previous experiments, we set the hyper-parameters of the training to be fixed and then train the models using the entire training data and our adversarial coreset selection method. For our method, we use batch-wise versions of CRAIG [27] and GRADMATCH [16] with warm-start. The *coreset size* for CIFAR-10 and ImageNet-12 were set to *40%* and *50%*, respectively. We measure the performance of the trained models against unseen attacks during training and the two variants of perceptual attacks as in [22]. The unseen attacks for each dataset were selected similarly to [22]. We also record the total training time taken by each method.

Tab. 2 summarizes our results on PAT using Fast-LPA (full results can be found in Appendix D). As seen, our adversarial coreset selection approach can deliver a competitive performance in terms of clean and average unseen attack accuracy while reducing the training time by at least a factor of two. These results indicate the flexibility of our adversarial coreset selection that can be

Table 3: Clean (ACC) and robust (RACC) accuracy, and average training speed ($\mathrm{S}_{\mathrm{avg}}$) of Fast Adversarial Training [41] without and with our adversarial coreset selection on CIFAR-10. The difference with full training is shown in parentheses for our proposed approach.

Training Method	**Performance Measures**			
	↑ **ACC** (%)	↑ **RACC** (%)	↓ $\mathbf{S}_{\mathrm{avg}}$ (min/epoch)	↓ **T** (min)
Fast Adv. Training	86.20	47.54	0.5178	31.068
+ Adv. CRAIG (Ours)	82.56 (−3.64)	47.77 (+0.23)	0.2783	16.695 (−14.373)
+ Adv. GRADMATCH (Ours)	82.53 (−3.67)	47.88 (+0.34)	0.2737	16.419 (−14.649)

combined with various objectives. This is due to the orthogonality of the proposed approach with the existing efficient adversarial training methods. In this case, we see that we can make Fast-LPA even faster using our approach.

4.3 Compatibility with Existing Methods

To showcase that our adversarial coreset selection approach is complementary to existing methods, we integrate it with a stable version of Fast Adversarial Training (FAT) [41] that does not use a cyclic learning rate. Specifically, we train a neural network using FAT [41], and then add adversarial coreset selection to this approach and record the training time and clean/robust accuracy. We run the experiments on the CIFAR-10 dataset and train a ResNet-18 for each case. We set the *coreset size* to *50%* for our methods. The results are shown in Tab. 3. As can be seen, our approach can be easily combined with existing methods to deliver faster training. This is due to the orthogonality of our approach that we discussed previously.

Moreover, we show that adversarial coreset selection gives a better approximation to ℓ_∞-PGD adversarial training compared to using FGSM [10] as done in FAT [41]. To this end, we use our adversarial GRADMATCH to train neural networks with the original ℓ_∞-PGD objective. We also train these networks using FAT [41] that uses FGSM. We train neural networks with a perturbation norm of $\|\varepsilon\|_\infty \leq 8$. Then, we evaluate the trained networks against PGD-50 adversarial attacks with different attack strengths to see how each network generalizes to unseen perturbations. As seen in Fig. 2, adversarial coreset selection is a closer approximation to ℓ_∞-PGD compared to FAT [41]. This indicates the success of the proposed approach in retaining the characteristics of the original objective as opposed to existing methods.

4.4 Ablation Studies

In this section, we perform a few ablation studies to examine the effectiveness of our adversarial coreset selection method. First, we compare a random data

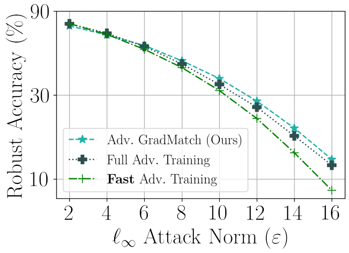

Fig. 2: Robust accuracy as a function of ℓ_∞ attack norm. We train neural networks with a perturbation norm of $\|\varepsilon\|_\infty \leq 8$ on CIFAR-10. At inference, we evaluate the robust accuracy against PGD-50 with various attack strengths.

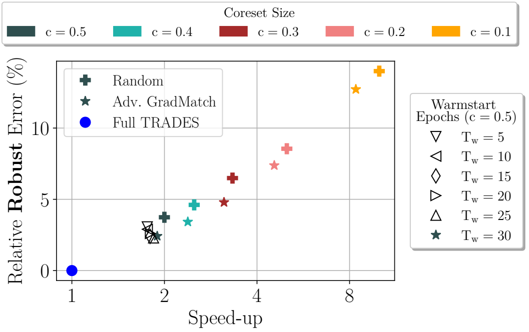

Fig. 3: Relative robust error vs. speed up for TRADES. We compare our adversarial coreset selection (GradMatch) for a given subset size against random data selection. Furthermore, we show our results for a selection of different warm-start settings.

selection with adversarial GradMatch. Fig. 3 shows that for any given coreset size, our adversarial coreset selection method results in a lower robust error. Furthermore, we modify the warm-start epochs for a fixed coreset size of 50%. As seen, the proposed method is not very sensitive to the number of warm-start epochs, although a longer warm-start is generally beneficial. More experiments on the accuracy vs. speed-up trade-off and the importance of warm-start and batch-wise adversarial coreset selection can be found in Appendix D.

5 Conclusion

In this paper, we proposed a general yet principled approach for efficient adversarial training based on the theory of coreset selection. We discussed how repetitive computation of adversarial attacks for the entire training data could impede the training speed. Unlike previous methods that try to solve this issue by making the adversarial attack more straightforward, here, we took an orthogonal path to reduce the training set size without modifying the attacker. We drew a connection between greedy coreset selection algorithms and adversarial training using Danskin's theorem. We then showed the flexibility of our adversarial coreset selection method by utilizing it for TRADES, ℓ_p-PGD, and Perceptual Adversarial Training. Our experimental results indicate that adversarial coreset selection can reduce the training time by more than 2-3 times with only a slight reduction in the clean and robust accuracy.

Acknowledgements. 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.

References

1. Adadi, A.: A survey on data-efficient algorithms in big data era. Journal of Big Data **8**(1), 1–54 (2021)
2. Andriushchenko, M., Flammarion, N.: Understanding and improving fast adversarial training. In: Proceedings of the Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS) (2020)
3. Biggio, B., Corona, I., Maiorca, D., Nelson, B., Srndic, N., Laskov, P., Giacinto, G., Roli, F.: Evasion attacks against machine learning at test time. In: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD). pp. 387–402 (2013)
4. Campbell, T., Broderick, T.: Bayesian coreset construction via greedy iterative geodesic ascent. In: Proceedings of the 35th International Conference on Machine Learning (ICML). pp. 697–705 (2018)
5. Croce, F., Hein, M.: Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In: Proceedings of the 37th International Conference on Machine Learning (ICML). pp. 2206–2216 (2020)
6. Danskin, J.M.: The theory of max-min and its application to weapons allocation problems, vol. 5. Springer Science & Business Media (1967)
7. Elenberg, E.R., Khanna, R., Dimakis, A.G., Negahban, S.N.: Restricted strong convexity implies weak submodularity. CoRR **abs/1612.00804** (2016)
8. Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., Song, D.: Robust physical-world attacks on deep learning visual classification. In: Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1625–1634 (2018)
9. Feldman, D.: Introduction to core-sets: an updated survey. CoRR **abs/2011.09384** (2020)
10. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: Proceedings of the 3rd International Conference on Learning Representations (ICLR) (2015)
11. 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). pp. 291–300 (2004)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016)
13. Kang, D., Sun, Y., Hendrycks, D., Brown, T., Steinhardt, J.: Testing robustness against unforeseen adversaries. CoRR **abs/1908.08016** (2019)
14. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8107–8116 (2020)
15. 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). pp. 2530–2539 (2018)
16. Killamsetty, K., Sivasubramanian, D., Ramakrishnan, G., De, A., Iyer, R.K.: GRAD-MATCH: gradient matching based data subset selection for efficient deep model training. In: Proceedings of the 38th International Conference on Machine Learning (ICML). pp. 5464–5474 (2021)

17. Killamsetty, K., Sivasubramanian, D., Ramakrishnan, G., Iyer, R.K.: GLISTER: generalization based data subset selection for efficient and robust learning. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence. pp. 8110–8118 (2021)
18. Kolter, Z., Madry, A.: Adversarial robustness: Theory and practice. `https://adversarial-ml-tutorial.org/` (2018), tutorial in the Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems (NeurIPS)
19. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto (2009)
20. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Proceedings of the Advances in Neural Information Processing Systems 25: Annual Conference on Neural Information Processing Systems (NeurIPS). pp. 1106–1114 (2012)
21. Laidlaw, C., Feizi, S.: Functional adversarial attacks. In: Proceedings of the Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems (NeurIPS). pp. 10408–10418 (2019)
22. Laidlaw, C., Singla, S., Feizi, S.: Perceptual adversarial robustness: Defense against unseen threat models. In: Proceedings of the 9th International Conference on Learning Representations (ICLR) (2021)
23. 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). pp. 182–199 (2020)
24. Ma, X., Niu, Y., Gu, L., Wang, Y., Zhao, Y., Bailey, J., Lu, F.: Understanding adversarial attacks on deep learning based medical image analysis systems. Pattern Recognition **110**, 107332 (2021)
25. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. In: Proceedings of the 6th International Conference on Learning Representations (ICLR) (2018)
26. Minoux, M.: Accelerated greedy algorithms for maximizing submodular set functions. In: Optimization Techniques, pp. 234–243. Springer (1978)
27. 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). pp. 6950–6960 (2020)
28. 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)
29. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions - I. Mathematical Programming **14**(1), 265–294 (1978)
30. 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)
31. Pati, Y.C., Rezaiifar, R., Krishnaprasad, P.S.: Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In: Proceedings of 27th Asilomar Conference on Signals, Systems and Computers. vol. 1, pp. 40–44 (1993)
32. 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 (IJCV) **115**(3), 211–252 (2015)

33. Schwartz, R., Dodge, J., Smith, N.A., Etzioni, O.: Green AI. Communication of the ACM **63**(12), 54–63 (2020)
34. Strubell, E., Ganesh, A., McCallum, A.: Energy and policy considerations for deep learning in NLP. In: Korhonen, A., Traum, D.R., Màrquez, L. (eds.) Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL). pp. 3645–3650 (2019)
35. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., Fergus, R.: Intriguing properties of neural networks. In: Proceedings of the 2nd International Conference on Learning Representations (ICLR) (2014)
36. Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I.J., Boneh, D., McDaniel, P.D.: Ensemble adversarial training: Attacks and defenses. In: Proceedings of the 6th International Conference on Learning Representations (ICLR) (2018)
37. Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., Madry, A.: Robustness may be at odds with accuracy. In: Proceedings of the 7th International Conference on Learning Representations (ICLR) (2019)
38. Vahdat, A., Kautz, J.: NVAE: A deep hierarchical variational autoencoder. In: Proceedings of the Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS) (2020)
39. Wei, K., Iyer, R., Bilmes, J.: Submodularity in data subset selection and active learning. In: Proceedings of the 32nd International Conference on Machine Learning (ICML). pp. 1954–1963 (2015)
40. Wolsey, L.A.: An analysis of the greedy algorithm for the submodular set covering problem. Combinatorica **2**(4), 385–393 (1982)
41. Wong, E., Rice, L., Kolter, J.Z.: Fast is better than free: Revisiting adversarial training. In: Proceedings of the 8th International Conference on Learning Representations (ICLR) (2020)
42. Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. `https://github.com/facebookresearch/detectron2` (2019)
43. Xiao, C., Zhu, J., Li, B., He, W., Liu, M., Song, D.: Spatially transformed adversarial examples. In: Proceedings of the 6th International Conference on Learning Representations (ICLR) (2018)
44. Zhang, H., Yu, Y., Jiao, J., Xing, E.P., Ghaoui, L.E., Jordan, M.I.: Theoretically principled trade-off between robustness and accuracy. In: Proceedings of the 36th International Conference on Machine Learning (ICML). pp. 7472–7482 (2019)
45. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 586–595 (2018)