ℓ_∞-ROBUSTNESS AND BEYOND: UNLEASHING EFFICIENT ADVERSARIAL TRAINING

Anonymous authors
Paper under double-blind review

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, which has hampered its effectiveness. Recently, *Fast Adversarial Training* (Wong et al., 2020) was proposed that can obtain robust models within minutes. 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 more principled approach towards reducing the time complexity of robust training. Unlike Fast Adversarial Training, our approach can be adapted to a wide variety of training objectives, including TRADES, ℓ_p-PGD, and Perceptual Adversarial Training. Our experimental results indicate that using coreset selection, one can train robust models 2-3 times faster while maintaining the clean and robust accuracy almost intact.

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 (He et al., 2016; Wu et al., 2019) to photo-realistic image generation (Karras et al., 2020; Vahdat & Kautz, 2020) and beyond. Despite their impressive performance, neural networks are vulnerable to adversarial attacks (Biggio et al., 2013; Szegedy et al., 2014): 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 (Eykholt et al., 2018) and medical diagnosis (Ma et al., 2021). 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] In this approach, a perturbed version of the training data is constructed first. Then, the neural network is optimized using these perturbed inputs instead of the clean samples. This procedure needs to be done iteratively as the perturbations depend on the neural network weights. Since the weights are optimized during training, the perturbations also need to be adjusted for each data sample in every iteration.

Various adversarial training methods primarily differ in the ways that they define and find the perturbed version of the input (Madry et al., 2018; Zhang et al., 2019; Laidlaw et al., 2021). However, they all require repetitive constriction of these perturbations during training which is often cast as another non-linear optimization problem. As such, the time/computational complexity of adversarial training in neural networks is massively higher than their vanilla training. In practice, neural

[1]Note that adversarial training in the literature generally refers to a particular approach proposed by Madry et al. (2018). 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(Zhang et al., 2019), ℓ_p-PGD (Madry et al., 2018) or Perceptual Adversarial Training (PAT) (Laidlaw et al., 2021) are all considered as different versions of adversarial training.

networks require massive amounts of training data (Adadi, 2021) and need to be trained multiple times with various hyper-parameters to get their best performance (Killamsetty et al., 2021a). Thus, reducing the time/computational complexity of adversarial training is critical to enable the environmentally efficient application of robust neural networks in real-world scenarios (Schwartz et al., 2020; Strubell et al., 2019).

Fast Adversarial Training (Wong et al., 2020) 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) (Goodfellow et al., 2015) does not help training arbitrary robust models (Tramèr et al., 2018; Madry et al., 2018), Wong et al. (2020) 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, Fast Adversarial Training can greatly reduce the time/computational complexity of training robust neural networks.

Despite its success, Fast Adversarial Training may exhibit unexpected behavior under different settings. For instance, it was shown that Fast Adversarial Training may suffer from *catastrophic overfitting* where the robust accuracy during training may suddenly drop to 0% (Wong et al., 2020; Andriushchenko & Flammarion, 2020). A more fundamental issue with Fast Adversarial Training and its variations such as `GradAlign` (Andriushchenko & Flammarion, 2020) is that they are specifically designed for ℓ_∞ adversarial training. This is because FGSM, which is particularly an ℓ_∞ perturbation generator, is at the heart of these methods. As a result, the quest for finding a unified approach that can reduce the time complexity of adversarial training is not over.

Motivated by the limited scope of Fast Adversarial Training, 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 needs to be done iteratively, we speculate that perhaps we 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. In particular, we draw an elegant connection between adversarial training and adaptive coreset selection algorithms. To this end, we use Danskin's Theorem and show how we can effectively approximate the entire training data with an informative 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. We show that our adaptive coreset selection process is only required every few epochs, hence, effectively reducing the time complexity of robust training algorithms. We show how our proposed approach 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 time complexity of various robust training objectives by a factor of 2-3 times without sacrificing too much clean and 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 for robust training of neural networks at scale.
- We show that our approach can be applied to a variety of robust learning objectives, including TRADES (Zhang et al., 2019), ℓ_p-PGD (Madry et al., 2018) and Perceptual (Laidlaw et al., 2021) Adversarial Training. As such, our approach encompasses a broader range of robust models, unlike Fast Adversarial Training which is specifically designed for ℓ_∞ robustness.
- Through our extensive experiments, we show that the proposed approach can result in a 2-3 fold reduction of the time complexity in adversarial training, while preserving the clean and robust accuracy.

2 BACKGROUND AND RELATED WORK

In this section, we review the related background to our 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 softmax 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

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \boldsymbol{\Phi}\left(\boldsymbol{x}_i, y_i; f_{\boldsymbol{\theta}}\right). \tag{1}$$

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. By writing the training objective in this format, we can denote both vanilla and robust adversarial training using the same notation. Below we show how various choices of the function $\boldsymbol{\Phi}$ amounts to different training objectives.

Vanilla Training. In case of vanilla training, the function 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. In other words, for vanilla training we 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, ℓ_p-PGD, and Perceptual Adversarial Training. In these cases, the training objective is itself an optimization problem

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

where $\mathrm{d}(\cdot, \cdot)$ is an appropriate distance measure over image domain $\mathbb{X}$, and ε denotes a scalar. The constraint over $\mathrm{d}\left(\tilde{\boldsymbol{x}}, \boldsymbol{x}\right)$ is used to ensure visual similarity between $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}$. It can be shown that solving Eq. (3) amounts to finding an adversarial example $\tilde{\boldsymbol{x}}$ for the clean sample $\boldsymbol{x}$ (Madry et al., 2018). Different choices of the visual similarity measure $\mathrm{d}(\cdot, \cdot)$ and solvers for Eq. (3) results in different adversarial training objectives.

- FGSM (Goodfellow et al., 2015) assumes that $\mathrm{d}\left(\tilde{\boldsymbol{x}}, \boldsymbol{x}\right) = \left\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\right\|_{\infty}$. Using this ℓ_∞ assumption, the solution to Eq. (3) is computed using one iteration of gradient ascent.
- In ℓ_p-PGD, Madry et al. (2018) utilize 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).
- Finally, Perceptual Adversarial Training (Laidlaw et al., 2021) replaces $\mathrm{d}(\cdot, \cdot)$ with *Learned Perceptual Image Patch Similarity* (LPIPS) distance (Zhang et al., 2018). Then, Laidlaw et al. (2021) 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 this method is creating a trade-off between clean and robust accuracy. In particular, the TRADES (Zhang et al., 2019) objective can be written as

$$\boldsymbol{\Phi}\left(\boldsymbol{x}, y; f_{\boldsymbol{\theta}}\right) = \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) + \max_{\tilde{\boldsymbol{x}}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}), f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)/\lambda \quad \text{s.t.} \quad \mathrm{d}\left(\tilde{\boldsymbol{x}}, \boldsymbol{x}\right) \leq \varepsilon, \tag{4}$$

where λ is a regularization parameter 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 certain attributes of the entire population (Feldman, 2020). Traditionally, coreset selection has been used for different machine learning tasks such as k-means and k-medians (Har-Peled & Mazumdar, 2004), Naïve Bayes and nearest neighbor classifiers (Wei et al., 2015), and Bayesian inference (Campbell & Broderick, 2018).

Recently, coreset selection algorithms are developed for neural network training (Mirzasoleiman et al., 2020a;b; Killamsetty et al., 2021b;a). The main idea behind such methods is to give an approximation of the full gradient using a 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, an objective function is formed. The goal of this optimization problem is to find a *weighted subset* of the training data that can approximate the full gradient.

It can be shown that this problem is NP-hard (Mirzasoleiman et al., 2020a;b). Roughly speaking, various coreset selection methods differ from each other in how they approximate the solution of the aforementioned objective. For instance, CRAIG (Mirzasoleiman et al., 2020a) casts this objective as a *submodular set cover problem* and uses existing greedy solvers to get an approximate solution. As another example, GRAD-MATCH (Killamsetty et al., 2021a) analyzes the convergence of stochastic gradient descent using adaptive data subset selection. Based on this study, Killamsetty et al. (2021a) propose to use Orthogonal Matching Pursuit (OMP) (Pati et al., 1993) as a greedy solver of the data selection objective. More information about these methods is provided in Appendix A.

The aforementioned coreset selection algorithms can only be used for vanilla training of neural networks. As such, they still suffer from adversarial vulnerability. Different from these methods, in this paper we extend coreset selection algorithms for robust training of neural networks, and show how they can be adapted to various robust training objectives including TRADES (Zhang et al., 2019), ℓ_p-PGD (Madry et al., 2018) and Perceptual (Laidlaw et al., 2021) Adversarial Training.

3 PROPOSED METHOD

As discussed in Section 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. Fast Adversarial Training (Wong et al., 2020) tries to eliminate this issue by using FGSM as its adversarial example generator. However, this simplification 1) may lead to catastrophic overfitting (Wong et al., 2020; Andriushchenko & Flammarion, 2020), and 2) is not easy to generalize to all types of adversarial training as FGSM is specifically designed for ℓ_∞ attacks.

Instead of using a faster adversarial example generator, here we take a different, *orthogonal* path and try to effectively reduce the training set size. This way, the original adversarial training algorithm can still be used on this smaller subset of training data. This approach can reduce the time/computational complexity while optimizing the same 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 are going to 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. However, the issue with existing coreset selection methods such as CRAIG (Mirzasoleiman et al., 2020a) or GRAD-MATCH (Killamsetty et al., 2021a) is that they are specifically designed for vanilla training of neural networks, 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 later can be used alongside newer versions of greedy coreset selection algorithms.

To find this general modification, recall from Section 2.2 that 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. This subset is obtained such that the weighted gradients of the samples inside the coreset can provide a good approximation of the full gradient.

We notice that various coreset selection methods proposed for vanilla neural network training only differ in the second step of the aforementioned process. As such, we can narrow down the changes that 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.

3.1 Coreset Selection for Efficient Adversarial Training

As discussed above, the first step in finding a weighted subset of training data is the computation of the loss gradient with respect to the neural network weights. This computation needs to be done for the entire training set. In particular, using our notation from Section 2.1, this step can be written as

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

where $V = [n] = \{1, 2, \ldots, n\}$ denotes the training set indices.

For vanilla neural network training (see Section 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 Danskin's (1969) Theorem as stated below.

Theorem 3.1 (Danskin (1967) (Theorem A.1 in Madry et al. (2018))) *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*

$$\phi(\boldsymbol{\theta}) = \max_{\delta \in \mathcal{S}} \ell(\boldsymbol{\theta}, \boldsymbol{\delta})$$

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 3.1 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).

3.1.1 ℓ_p-PGD and Perceptual Adversarial Training

Going back to Eq. (5), 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

$$\nabla_{\boldsymbol{\theta}} \boldsymbol{\Phi}\left(\boldsymbol{x}, y; f_{\boldsymbol{\theta}}\right) = \nabla_{\boldsymbol{\theta}} \max_{\tilde{\boldsymbol{x}}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}), y\right) \quad \text{s.t.} \quad \mathrm{d}\left(\tilde{\boldsymbol{x}}, \boldsymbol{x}\right) \leq \varepsilon \tag{6}$$

for every training sample. Based on Danskin's Theorem, we can deduce

$$\nabla_{\boldsymbol{\theta}} \boldsymbol{\Phi}\left(\boldsymbol{x}, y; f_{\boldsymbol{\theta}}\right) = \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}^*), y\right), \tag{7}$$

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

$$\max_{\tilde{\boldsymbol{x}}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}), y\right) \quad \text{s.t.} \quad \mathrm{d}\left(\tilde{\boldsymbol{x}}, \boldsymbol{x}\right) \leq \varepsilon. \tag{8}$$

Note that the conditions under which Danskin's Theorem hold might not be satisfied for neural networks in general. This is due to the existence of discontinuous functions such as ReLU activation in neural networks. More importantly, finding the exact solution of Eq. (8) 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. (8) objective (Kolter & Madry, 2018). Based on this approximation, we can re-write Eq. (7) as

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

In other words, to perform coreset selection for ℓ_p-PGD (Madry et al., 2018) and Perceptual (Laidlaw et al., 2021) Adversarial Training, one needs to add a pre-processing step to the gradient computation. During this step, adversarial examples for the entire training set need to be constructed. Afterwards, the coresets can be built as in vanilla neural networks.

3.1.2 TRADES ADVERSARIAL TRAINING

For TRADES (Zhang et al., 2019), 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

$$\nabla_{\boldsymbol{\theta}} \boldsymbol{\Phi}\left(\boldsymbol{x}, y; f_{\boldsymbol{\theta}}\right) = \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) + \nabla_{\boldsymbol{\theta}} \max_{\tilde{\boldsymbol{x}}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}), f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)/\lambda \quad \text{s.t.} \ \mathrm{d}\left(\tilde{\boldsymbol{x}}, \boldsymbol{x}\right) \leq \varepsilon. \tag{10}$$

The first term is the normal gradient of the neural network. For the second term, we apply Danskin's Theorem to get

$$\nabla_{\boldsymbol{\theta}} \boldsymbol{\Phi}\left(\boldsymbol{x}, y; f_{\boldsymbol{\theta}}\right) \approx \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) + \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_{\mathrm{adv}}), f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)/\lambda, \tag{11}$$

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

$$\max_{\tilde{\boldsymbol{x}}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}), f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)/\lambda \quad \text{s.t.} \quad \mathrm{d}\left(\tilde{\boldsymbol{x}}, \boldsymbol{x}\right) \leq \varepsilon. \tag{12}$$

Then, we compute the gradient term $\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_{\mathrm{adv}}), f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)$ in Eq. (11) using the multi-variable chain rule. To show this, let us assume that $\boldsymbol{w}(\boldsymbol{\theta}) = f_{\boldsymbol{\theta}}(\boldsymbol{x}_{\mathrm{adv}})$ and $\boldsymbol{z}(\boldsymbol{\theta}) = f_{\boldsymbol{\theta}}(\boldsymbol{x})$. We can write the aforementioned gradient as

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_{\mathrm{adv}}), f_{\boldsymbol{\theta}}(\boldsymbol{x})\right) &= \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{CE}}\left(\boldsymbol{w}(\boldsymbol{\theta}), \boldsymbol{z}(\boldsymbol{\theta})\right) \\
&\overset{(1)}{=} \tfrac{\partial \mathcal{L}_{\mathrm{CE}}}{\partial \boldsymbol{w}} \nabla_{\boldsymbol{\theta}} \boldsymbol{w}(\boldsymbol{\theta}) + \tfrac{\partial \mathcal{L}_{\mathrm{CE}}}{\partial \boldsymbol{z}} \nabla_{\boldsymbol{\theta}} \boldsymbol{z}(\boldsymbol{\theta}) \\
&\overset{(2)}{=} \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_{\mathrm{adv}}), \texttt{freeze}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)\right) \\
&\qquad + \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{CE}}\left(\texttt{freeze}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_{\mathrm{adv}})\right), f_{\boldsymbol{\theta}}(\boldsymbol{x})\right).
\end{aligned} \tag{13}$$

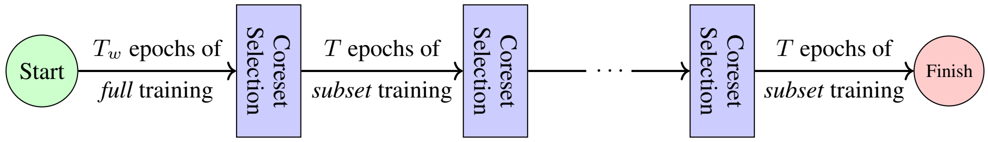
Here, step (1) is derived using the multi-variable chain rule. Also, step (2) is the re-writing of step (1) by using the $\texttt{freeze}(\cdot)$ kernel that stops the gradients from backpropagating through its argument function. Putting Eqs. (11) and (13) together, we can write the TRADES gradient as

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \boldsymbol{\Phi}\left(\boldsymbol{x}, y; f_{\boldsymbol{\theta}}\right) = \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) &+ \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{CE}}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_{\mathrm{adv}}), \texttt{freeze}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)\right)/\lambda \\
&+ \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{CE}}\left(\texttt{freeze}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_{\mathrm{adv}})\right), f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)/\lambda.
\end{aligned} \tag{14}$$

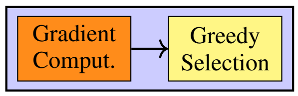
After the gradient is computed for the entire training data using the above formulation, one can use coreset selection algorithms to find the weighted subset of the training data.

3.1.3 PRACTICAL CONSIDERATIONS AND THE FINAL ALGORITHM

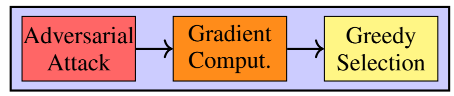
Following the above discussion, to effectively reduce the training set size using coreset selection we need to compute the loss gradient for the entire training data. For adversarial training this amounts to adding a pre-processing step where we need to build perturbed versions of the training data using their respective objectives in Eqs. (8) and (12). Then, the gradients can be computed using Eqs. (9) and (14). Afterwards, greedy subset selection algorithms can be used to construct the coresets based on the value of the gradients. Finally, having selected the coreset data, one can run 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 training size to enhance the computational efficiency of our proposed solution and as such, we are able use it along with any adversarial training objectives. We stress once again that our approach is *orthogonal* to the



(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.

Figure 1: Overview of neural network training using coreset selection.

existing work around efficient adversarial training in the sense that it can also be combined with those approaches to make them even faster. We leave this direction for future work.

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 (Killamsetty et al., 2021b;a) has shown that this selection needs to be done every T epochs, where T is usually greater than 15. Furthermore, there are small, yet important practical changes that we employ while using coreset selection to increase efficiency. We summarize these practical tweaks below. Further detail can be found in Killamsetty et al. (2021a); Mirzasoleiman et al. (2020a).

- **Gradient Approximation**: As we saw, both Eqs. (9) 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. (9) and (14) with their last-layer approximation (Katharopoulos & Fleuret, 2018; Mirzasoleiman et al., 2020a; Killamsetty et al., 2021a). In other words, instead of backpropagating through the entire network, one can backpropagate up until the last layer. This estimate has an approximate complexity equal to forwardpropagation, and it has been shown to work well in practice (Mirzasoleiman et al., 2020a;b; Killamsetty et al., 2021b;a).
- **Batch-wise Coreset Selection**: As discussed in Section 3.1, data selection is usually done in a *sample-wise* fashion where each data sample is separately considered to be selected. This way, one needs to find the data candidates out of the entire training set. To increase efficiency, Killamsetty et al. (2021a) 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 can increase efficiency as the sample size reduces from the number of data points to the number of batches.
- **Warm-start with the Entire Data**: Finally, as we shall see in the experiments, it is important to warm-start the training using the entire dataset. Afterwards, the coreset selection is activated and training is only performed using the data that remains in the coreset.

Figure 1 summarizes the coreset selection for vanilla and adversarial training.

4 Experimental Results

In this section, we present our experimental results. We show how our proposed approach can efficiently reduce the time complexity of various robust training objectives in different settings. To this end, we train our approach using TRADES (Zhang et al., 2019), ℓ_p-PGD (Madry et al., 2018) and Perceptual (Laidlaw et al., 2021) Adversarial Training on CIFAR-10 (Krizhevsky & Hinton, 2009), SVHN (Netzer et al., 2011), and a subset of ImageNet (Russakovsky et al., 2015) with 12 classes. For TRADES and ℓ_p-PGD training, we use ResNet-18 (He et al., 2016) classifiers, while for Perceptual Adversarial Training we use ResNet-50 architectures. Further implementation details can be found in Appendix B.

TRADES and ℓ_p-PGD Robust Training. In our first set of experiments, we train ResNet-18 classifiers on CIFAR-10 and SVHN datasets using TRADES, ℓ_∞ and ℓ_2-PGD adversarial training

Table 1: Clean and robust accuracy (%), and total training time (mins) of different adversarial training methods. For each method, all the hyper-parameters were kept the same as full training. For our proposed approach, the difference with full training is shown in parentheses. Note that the robust accuracy for each objective was computed accordingly, the detail of which can be found in the Appendix.

Object.	Dataset	Training	Clean Acc. (%)	Robust Acc. (%)	Train. Time (mins)
TRADES	CIFAR-10	Adversarial CRAIG (Ours)	79.31 (−1.07)	30.20 (−0.56)	486.18 (−615.66)
		Adversarial GRADMATCH (Ours)	79.22 (−1.16)	30.66 (−0.10)	475.68 (−626.16)
		Full Adversarial Training	80.38	30.76	1101.84
ℓ_∞-PGD	CIFAR-10	Adversarial CRAIG (Ours)	84.00 (−0.84)	34.56 (−0.72)	352.98 (−480.36)
		Adversarial GRADMATCH (Ours)	84.30 (−0.54)	35.01 (−0.27)	342.54 (−490.80)
		Full Adversarial Training	84.84	35.28	833.34
ℓ_2-PGD	SVHN	Adversarial CRAIG (Ours)	94.65 (−0.54)	93.51 (−0.66)	211.08 (−709.50)
		Adversarial GRADMATCH (Ours)	93.61 (−1.58)	92.30 (−1.87)	206.22 (−714.36)
		Full Adversarial Training	95.19	94.17	920.58

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 as well as using our adversarial coreset selection approach. For our approach, we use batch-wise versions of CRAIG (Mirzasoleiman et al., 2020a) and GRADMATCH (Killamsetty et al., 2021a) with warm-start. We set the coreset size (the percentage of training data to be selected) to 40% for CIFAR-10 and 20% 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 Table 1. For our approach, we also report the difference with respect to full training in parenthesis. As can be seen, in all of the cases we can reduce the training time by more than a factor of two while keeping both the clean and robust accuracy almost intact.

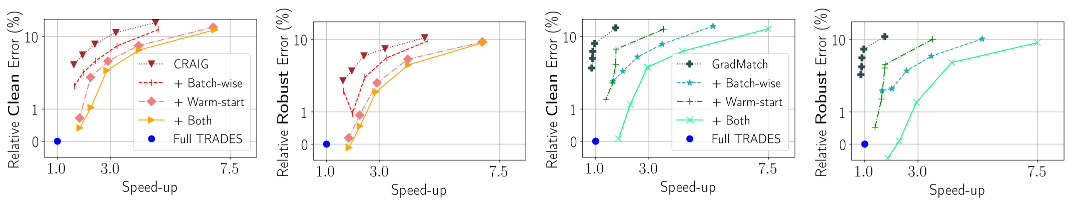
Perceptual Adversarial Training and Robustness vs. Unseen Attacks. As discussed in Section 2, Perceptual Adversarial Training (PAT) (Laidlaw et al., 2021) replaces the visual similarity measure $\mathrm{d}(\cdot,\cdot)$ in Eq. (3) with LPIPS (Zhang et al., 2018) distance. The logic behind this choice is that ℓ_p norms can only capture a small portion of images that are similar to the clean one, limiting the search space of adversarial attacks. Motivated by this reason, Laidlaw et al. (2021) proposes two different ways of finding the solution to Eq. (3) when $\mathrm{d}(\cdot,\cdot)$ is the LPIPS distance. The first version uses PGD, and the second one 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. (2021) proposed to utilize a fast version of LPA to enable its usage in adversarial training. More information about this approach can be found in Laidlaw et al. (2021).

For our next set of experiments, we train ResNet-50 classifiers using Fast-LPA. In this case, 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 (Mirzasoleiman et al., 2020a) and GRADMATCH (Killamsetty et al., 2021a) with warm-start. The coreset size for CIFAR-10 and ImageNet-12 were set to 40% and 50%, respectively. As in Laidlaw et al. (2021), we measure the performance of the trained models against unseen attacks during training, as well as the two variants of perceptual attacks. The unseen attacks for each dataset were selected in a similar manner to Laidlaw et al. (2021), and the attack parameters can be found in Appendix B. We also record the total training time taken by each method.

Table 2 summarizes our results on PAT using Fast-LPA. As can be seen, our adversarial coreset selection approach can deliver almost the same performance while reducing the training time by a factor of two. These results indicate the flexibility of our adversarial coreset selection that can be 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 using our approach we can make Fast-LPA even faster.

Table 2: Clean and robust accuracy (%), and total training time (mins) of Perceptual Adversarial Training for CIFAR-10 and ImageNet-12 datasets. The training objective uses Fast Lagrangian Perceptual Attack (LPA) (Laidlaw et al., 2021) to train the network. At test time the networks are evaluated against attacks that were not seen during training, as well as different versions of Perceptual Adversarial Attack (PPGD and LPA). In each dataset, the unseen attacks were selected similar to Laidlaw et al. (2021). For more information about the settings, please see the Appendix.

Dataset	Training	Clean	Unseen Attacks						Seen Attacks		Train. Time (mins)
			Auto-ℓ_2	Auto-ℓ_∞	JPEG	StAdv	ReColor	Mean	PPGD	LPA	
CIFAR-10	Adv. CRAIG (Ours)	83.21	39.98	33.94	-	49.60	62.69	46.55	19.56	7.42	767.34
	Adv. GRADMATCH (Ours)	83.14	39.20	34.11	-	48.86	62.26	46.11	19.94	7.54	787.26
	Full PAT (Fast-LPA)	86.02	43.27	37.96	-	48.68	62.23	48.04	22.62	8.01	1682.94
ImageNet-12	Adv. CRAIG (Ours)	86.99	51.54	60.42	71.79	37.47	44.04	53.05	29.04	14.07	2817.06
	Adv. GRADMATCH (Ours)	87.08	51.38	60.64	72.15	35.83	45.83	53.17	28.36	13.11	2865.72
	Full PAT (Fast-LPA)	91.22	57.37	66.89	76.25	19.29	46.35	53.23	33.17	13.49	5613.12



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

Figure 2: Relative error vs. speed up curves for different versions of adversarial coreset selection in training CIFAR-10 models using the TRADES objective. In each curve, the coreset size is changed from 50% to 10% (left to right). (a, b) Clean and robust error vs. speed up compared to full TRADES for different version of Adversarial CRAIG. (c, d) Clean and robust error vs. speed up compared to full TRADES for different version of Adversarial GRADMATCH.

Trade-offs. Finally, we study the accuracy vs. speed-up trade-off in adversarial coreset selection. For this study, we train our adversarial coreset selection method using different versions of CRAIG (Mirzasoleiman et al., 2020a) and GRADMATCH (Killamsetty et al., 2021a) on CIFAR-10 using TRADES. In particular, for each method, we start with the base algorithm and add the batch-wise selection and warm-start one by one. Also, to capture the effect of the coreset size, we vary this number from 50% to 10% in each case. Figure 2 shows the clean and robust error vs. speed-up compared to full adversarial training. As can be seen, in each case the combination of warm-start and batch-wise versions of the adversarial coreset selection gives the best performance. This observation is in line with that of Killamsetty et al. (2021a) around vanilla coreset selection. Moreover, as we gradually decrease the coreset size, we see that the training speed goes up. However, this gain in training speed is achieved at the cost of increasing the clean and robust error.

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 can impede the training speed. Unlike previous works that try to solve this issue by making the adversarial attack simpler, here, we took an orthogonal path to effectively reduce the training set size without modifying the attacker. To this end, we discussed how coreset selection can be viewed as a two-step process, where first the gradients for the entire training data are computed, and then greedy solvers choose a weighted subset of data that can approximate the full gradient. Using Danskin's Theorem, we drew a connection between greedy coreset selection algorithms and adversarial training. 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 factors of more than 2-3 while keeping the clean and robust accuracy almost intact.

REFERENCES

Amina Adadi. A survey on data-efficient algorithms in big data era. *Journal of Big Data*, 8(1):1–54, 2021.

Maksym Andriushchenko and Nicolas Flammarion. Understanding and improving fast adversarial training. In *Proceedings of the Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems*, 2020.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Zelezný (eds.), *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*, volume 8190 of *Lecture Notes in Computer Science*, pp. 387–402. Springer, 2013.

Trevor Campbell and Tamara Broderick. Bayesian coreset construction via greedy iterative geodesic ascent. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pp. 697–705, 2018.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *Proceedings of the 37th International Conference on Machine Learning (ICML) 2020*, pp. 2206–2216, 2020.

John M Danskin. *The theory of max-min and its application to weapons allocation problems*, volume 5. Springer Science & Business Media, 1967.

Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceeding of the 2018 IEEE Conference on Computer Vision and Pattern Recognition CVPR*, pp. 1625–1634, 2018.

Dan Feldman. Introduction to core-sets: an updated survey. *CoRR*, abs/2011.09384, 2020.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

Sariel Har-Peled and Soham Mazumdar. 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.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Daniel Kang, Yi Sun, Dan Hendrycks, Tom Brown, and Jacob Steinhardt. Testing robustness against unforeseen adversaries. *CoRR*, abs/1908.08016, 2019.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the 2020 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8107–8116, 2020.

Angelos Katharopoulos and François Fleuret. 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.

KrishnaTeja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, Abir De, and Rishabh K. Iyer. GRAD-MATCH: gradient matching based data subset selection for efficient deep model training. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5464–5474, 2021a.

KrishnaTeja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh K. Iyer. GLISTER: generalization based data subset selection for efficient and robust learning. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, pp. 8110–8118, 2021b.

Zico Kolter and Aleksander Madry. 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).

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 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.

Cassidy Laidlaw and Soheil Feizi. 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.

Cassidy Laidlaw, Sahil Singla, and Soheil Feizi. Perceptual adversarial robustness: Defense against unseen threat models. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, 2021.

Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. 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.

Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

Xingjun Ma, Yuhao Niu, Lin Gu, Yisen Wang, Yitian Zhao, James Bailey, and Feng Lu. Understanding adversarial attacks on deep learning based medical image analysis systems. *Pattern Recognition*, 110:107332, 2021.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*, pp. 234–243. Springer, 1978.

Baharan Mirzasoleiman, Jeff A. Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pp. 6950–6960, 2020a.

Baharan Mirzasoleiman, Kaidi Cao, and Jure Leskovec. 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)*, 2020b.

George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Mathematical Programming*, 14(1):265–294, 1978.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*, volume 1, pp. 40–44, 1993.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *Communication of the ACM*, 63(12):54–63, 2020.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics ACL*, pp. 3645–3650, 2019.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian J. Goodfellow, Dan Boneh, and Patrick D. McDaniel. Ensemble adversarial training: Attacks and defenses. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

Arash Vahdat and Jan Kautz. 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.

Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pp. 1954–1963, 2015.

Laurence A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.

Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, 2020.

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pp. 7472–7482, 2019.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 586–595, 2018.

SUPPLEMENTARY MATERIALS

A GREEDY SUBSET SELECTION ALGORITHMS

In this section, we briefly review the technical details of greedy subset selection algorithms used in our experiments. Further details can be found in Mirzasoleiman et al. (2020a); Killamsetty et al. (2021a).

A.1 CRAIG

As discussed previously, the goal of coreset selection is to find a subset of the training data such that the weighted gradient computed over this subset can give a good approximation to the full gradient. Thus, CRAIG (Mirzasoleiman et al., 2020a) starts with explicitly writing down this objective as

$$\left\| \sum_{i \in V} \nabla_{\boldsymbol{\theta}} \boldsymbol{\Phi}\left(\boldsymbol{x}_i, y_i; f_{\boldsymbol{\theta}}\right) - \sum_{j \in S} \gamma_j \nabla_{\boldsymbol{\theta}} \boldsymbol{\Phi}\left(\boldsymbol{x}_j, y_j; f_{\boldsymbol{\theta}}\right) \right\|. \tag{15}$$

Here, $V = [n] = \{1, 2, \ldots, n\}$ denotes the training set. The goal is to find a coreset $S \subset V$ and its associated weights γ_j such that the objective of Eq. (15) is minimized. To this end, Mirzasoleiman et al. (2020a) find an upper-bound on the estimation error of Eq. (15). This way, it is shown that the coreset selection objective can be approximated by

$$S^* = \arg\min_{S \subseteq V} |S|, \quad \text{s.t.} \quad L(S) \triangleq \sum_{i \in V} \min_{j \in S} d_{ij} \leq \epsilon, \tag{16}$$

where

$$d_{ij} \triangleq \max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \left\| \nabla_{\boldsymbol{\theta}} \boldsymbol{\Phi}\left(\boldsymbol{x}_i, y_i; f_{\boldsymbol{\theta}}\right) - \nabla_{\boldsymbol{\theta}} \boldsymbol{\Phi}\left(\boldsymbol{x}_j, y_j; f_{\boldsymbol{\theta}}\right) \right\| \tag{17}$$

denotes the maximum pairwise gradient distances computed for all $i \in V$ and $j \in S$. Then, Mirzasoleiman et al. (2020a) cast Eq. (16) as the well-known *submodular set cover problem* for which greedy solvers exist (Minoux, 1978; Nemhauser et al., 1978; Wolsey, 1982).

A.2 GRAD-MATCH

Killamsetty et al. (2021a) studies the convergence of adaptive data subset selection algorithms using *stochastic gradient descent* (SGD). It is shown that the convergence bound consists of two terms: an irreducible noise-related term, and an additional gradient error term just like Eq. (15). Based on this analysis, Killamsetty et al. (2021a) then proposes to minimize this objective directly. To this end, they use the famous Orthogonal Matching Pursuit (OMP) (Pati et al., 1993) as their greedy solver, resulting in an algorithm called GRAD-MATCH. It is then proved that since GRAD-MATCH minimizes the Eq. (15) objective directly, it achieves a lower error compared to CRAIG that only minimizes an upper-bound to Eq. (15).

B IMPLEMENTATION DETAILS

In this section, we provide the details of our experiments in Section 4. We used a single NVIDIA Tesla V100-SXM2-16GB GPU for CIFAR-10 (Krizhevsky & Hinton, 2009) and SVHN (Netzer et al., 2011), and a single NVIDIA Tesla V100-SXM2-32GB GPU for ImageNet-12 (Russakovsky et al., 2015; Liu et al., 2020). The code will be released upon publication.

B.1 TRAINING SETTINGS.

Table 3 shows the entire set of hyper-parameters and settings used for training the models of Section 4.

Table 3: Training details for experimental results of Section 4.

Hyperparameter	Experiment				
	TRADES	ℓ_∞-PGD	ℓ_2-PGD	Fast-LPA	Fast-LPA
Dataset	CIFAR-10	CIFAR-10	SVHN	CIFAR-10	ImageNet-12
Model Arch.	ResNet-18	ResNet-18	ResNet-18	ResNet-50	ResNet50
Optimizer	SGD	SGD	SGD	SGD	SGD
Scheduler	Cosine Annealing	Cosine Annealing	Multi-step	Multi-step	Multi-step
Initial lr.	0.01	0.01	0.1	0.1	0.1
lr. Decay (epochs)	-	-	0.1 (75, 90, 100)	0.1 (75, 90, 100)	0.1 (45, 60, 80))
Weight Decay	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$2 \cdot 10^{-4}$	$2 \cdot 10^{-4}$
Batch Size (full)	128	128	128	50	50
Total Epochs	200	200	120	120	90
Coreset Size	40%	40%	20%	40%	50%
Coreset Batch Size	20	20	20	20	20
Warm-start Epochs	48	48	15	29	27
Coreset Selection Period (epochs)	20	20	20	10	15
Visual Similarity Measure	ℓ_∞	ℓ_∞	ℓ_2	LPIPS (AlexNet)	LPIPS (AlexNet)
ε **(Bound on Visual Sim.)**	$8/255$	$8/255$	$80/255$	0.5	0.25
Attack Iterations (Training)	10	7	10	10	10
Attack Iterations (Coreset Selection)	10	1	10	10	10
Attack Step-size	$1/255$	$1/255$	$8/255$	-	-

B.2 EVALUATION SETTINGS

For the evaluation of TRADES and ℓ_p-PGD adversarial training, we use PGD attacks. In particular, for TRADES and ℓ_∞-PGD adversarial training we use ℓ_∞-PGD attacks with $\varepsilon = 8/255$, step-size $\alpha = 1/255$, 50 iterations, and 10 random restarts. Also, for ℓ_2-PGD adversarial training we use ℓ_2-PGD attacks with $\varepsilon = 80/255$, step-size $\alpha = 8/255$, 50 iterations and 10 random restarts.

For Perceptual Adversarial Training (PAT), we report the attacks' settings in Table 4. Note that for each case, we chose the same set of unseen/seen attacks for evaluation. However, since we trained our model with slightly different ε bounds, we changed the attacks' settings accordingly.

Table 4: Hyper-parameters of attacks used for the evaluation of PAT models in Section 4.

Dataset	Attack	Bound	Iterations
CIFAR-10	AutoAttack-ℓ_2 (Croce & Hein, 2020)	1	20
	AutoAttack-ℓ_∞ (Croce & Hein, 2020)	$8/255$	20
	StAdv (Xiao et al., 2018)	0.02	50
	ReColor (Laidlaw & Feizi, 2019)	0.06	100
	PPGD (Laidlaw et al., 2021)	0.40	40
	LPA (Laidlaw et al., 2021)	0.40	40
ImageNet-12	AutoAttack-ℓ_2 (Croce & Hein, 2020)	$1200/255$	20
	AutoAttack-ℓ_∞ (Croce & Hein, 2020)	$4/255$	20
	JPEG (Kang et al., 2019)	0.125	200
	StAdv (Xiao et al., 2018)	0.02	50
	ReColor (Laidlaw & Feizi, 2019)	0.06	200
	PPGD (Laidlaw et al., 2021)	0.35	40
	LPA (Laidlaw et al., 2021)	0.35	40