Stochastic Frank Wolfe for Constrained Nonconvex Optimization

Anonymous authors
Paper under double-blind review

Abstract

We provide a practical convergence analyses of Stochastic Frank Wolfe (SFW) and SFW with momentum with constant and decaying learning rates for constrained nonconvex optimization problems. We show that a convergence measure called the Frank Wolfe gap converges to zero only when we decrease the learning rate and increase the batch size. We apply SFW algorithms to adversarial attacks and propose a new adversarial attack method, Auto-SFW. Finally, we compare existing methods with the SFW algorithms in attacks against the latest robust models.

1 Introduction

1.1 Background

Nonconvex optimization is necessary for training deep neural networks. First-order methods, such as stochastic gradient descent (SGD) (Robbins & Monro, 1951), adaptive moment estimation (Adam) (Kingma & Ba, 2015) and their variants (Polyak, 1964; Nesterov, 1983; Duchi et al., 2011; Tieleman & Hinton, 2012; Reddi et al., 2018), are still very powerful methods, and their convergence analysis for nonconvex optimization has been widely studied (Fehrman et al., 2020; Bottou et al., 2018a; Scaman & Malherbe, 2020; Loizou et al., 2021; Zaheer et al., 2018; Zou et al., 2019; Chen et al., 2019; Zhou et al., 2020; Chen et al., 2021; Iiduka, 2022).

When solving constrained optimization problems with these methods, it is necessary to compute the projection onto the constraint set at each iteration. In many problem settings, the computational cost of projections to the constraint set such as the Euclidean norm ball can be very high, and in extreme cases the projections can even be computationally infeasible (Collins et al., 2008). Here, we focus on the Frank Wolfe algorithm (Frank & Wolfe, 1956), also called the conditional gradient algorithm (Levitin & Polyak, 1966), a projection-free first-order method for constrained optimization.

The Frank Wolfe algorithm is a classical first-order method for solving convex optimization problems with closed convex constraint sets. In recent years, it has received renewed attention thanks to its ability to efficiently handle structured constraints that appear in machine learning. The algorithm and its many variants, such as stochastic Frank Wolfe (SFW), have been well studied in the convex or strongly convex setting (Jaggi, 2013; Freund & Grigas, 2014; Lacoste-Julien & Jaggi, 2015; Goldfarb et al., 2017; Locatello et al., 2017; Zhang et al., 2019c; Tang et al., 2022), and they have been applied to matrix completion (Freund et al., 2017; Locatello et al., 2019), regression (Négiar et al., 2020; Dvurechensky et al., 2023; Wirth et al., 2023), and support vector machine (SVM)(Hazan & Luo, 2016; Lu & Freund, 2021). Even in the nonconvex setting, convergence analyses have been provided for many variants (Reddi et al., 2016; Gu et al., 2019; Grigas et al., 2019; Chen et al., 2020a; Pokutta et al., 2020; Sahu & Kar, 2020; Combettes et al., 2021; Nazykov et al., 2024), and some have been successful in experiments on deep neural networks (DNNs) (Berrada et al., 2019; Miao et al., 2022). In particular, Frank Wolfe-type algorithms have been shown to be effective in making adversarial attacks (see Section 4).

1.2 Motivation

1. Impractical analyses. Several previous studies provide convergence analyses of SFW methods for nonconvex optimization and many of the analyses share common problems of practicality and correctness. Some of the previous studies present guarantees of convergence by deriving the inequality, $\frac{1}{T}\sum_{t=0}^{T-1}\mathbf{gap} \leq \frac{A}{T\gamma} + B\gamma$, and use $\gamma = \frac{1}{\sqrt{T}}$ to derive $\frac{1}{T}\sum_{t=0}^{T-1}\mathbf{gap} \leq \frac{A}{\sqrt{T}} + \frac{B}{\sqrt{T}} = \mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$, where **gap** is the convergence criterion, T the number of steps, $\gamma > 0$ the learning rate, and $A, B > 0$ constants for simplicity. Recall that convergence of a sequence (a_n) to 0 is a necessary and sufficient condition of the following: $\forall\epsilon > 0, \exists n_0 \in \mathbb{N}\colon \forall n \geq n_0 \Rightarrow |a_n| < \epsilon$. Therefore, if the learning rate is set as $\gamma = \frac{1}{\sqrt{T}}$, the total number of iterations T is predetermined and fixed, so $\frac{A}{\sqrt{T}} + \frac{B}{\sqrt{T}}$ is a constant, T cannot approach infinity, and **gap** is not guaranteed to converge to 0. This is evident from the fact that $\frac{1}{T}\sum_{t=0}^{T-1}\mathbf{gap} \leq \frac{A}{T\gamma} + B\gamma$ for any T, and even if $T \to \infty$, convergence of **gap** to 0 is still not guaranteed due to the extra term $B\gamma$. Thus, these analyses do not exactly show $\frac{1}{T}\sum_{t=0}^{T-1}\mathbf{gap} = \mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$ and $\frac{1}{T}\sum_{t=0}^{T-1}\mathbf{gap} \to 0$ $(T \to \infty)$.

Table 1: Comparison of our analysis with previous studies of SFW methods in nonconvex optimization. T means the total number of iterations and $t \in [T]$ denotes an iteration or time. L means Lipschitz constant. $\gamma_t = \mathcal{O}\left(\sqrt{\frac{1}{TL}}\right)$ in Learning Rate column indicates that the learning rate γ_t is set at $\gamma_t = \frac{K}{TL}$ using some positive constant K, emphasizing in particular that it is based on T and L. "Noise" in the Gradient column means that algorithm uses noisy observation, i.e., $g(\boldsymbol{\theta}) = \nabla f(\boldsymbol{\theta}) + (\text{Noise})$, of the full gradient $\nabla f(\boldsymbol{\theta})$, while "mini-batch" in the Gradient column means that algorithm uses a mini-batch gradient $\nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}) = \frac{1}{b}\sum_{i\in[b]}\mathsf{G}_{\xi_{t,i}}$ with a batch size b $(\leq n)$, where $\mathsf{G}_{\xi_{t,i}}$ is stochastic gradient and n is the number of training data. The Momentum column states whether the SFW algorithm includes a momentum factor.

Algorithm	Learning Rate	Gradient (batch size)	Momentum
SFW (Reddi et al., 2016)	$\gamma_t = \mathcal{O}\left(\sqrt{\frac{1}{TL}}\right)$	mini-batch $(b = T)$	No
AsySFW (Gu et al., 2019)	$\gamma_t = \mathcal{O}\left(\sqrt{\frac{1}{TL}}\right)$	mini-batch $(b = T)$	No
FW-SD (Grigas et al., 2019)	$\gamma_t = \mathcal{O}\left(\sqrt{\frac{1}{L}}\right)$	mini-batch $(b_t \leq n)$	No
SFW (Négiar et al., 2020)	$\gamma_t = \frac{2}{t+2}$	mini-batch $(b \leq n)$	No
SCG (Mokhtari et al., 2020)	$\gamma_t = \frac{1}{T}$	Noise	Yes (Increasing)
AdaSFW (Combettes et al., 2021)	$\gamma_t = \frac{2}{t+2}$	mini-batch $\left(b_t = \mathcal{O}\left(\frac{t^2}{L^2}\right)\right)$	No
AdaCSFW (Combettes et al., 2021)	$\gamma_t = \frac{2}{t+2}$	mini-batch $(b \leq n)$	No
any SFW methods (Nazykov et al., 2024)	$\gamma_t = \sqrt{\frac{1}{T}}$	Noise	No
SFW and SFWM (ours)	$\gamma_t = \gamma$	mini-batch $(b \leq n)$	No/Yes
SFW and SFWM (ours)	γ_t defined as (1)	mini-batch $(b \leq n)$	No/Yes
SFW and SFWM (ours)	$\gamma_t = \gamma$	b_t defined as (2)	No/Yes
SFW and SFWM (ours)	γ_t defined as (1)	b_t defined as (2)	No/Yes

Table 1 compares our analysis with previous studies on SFW methods for nonconvex optimization. Some of the previous studies on Frank Wolfe methods (Reddi et al., 2016; Gu et al., 2019; Chen et al., 2020a; Mokhtari et al., 2020; Nazykov et al., 2024) used a learning rate γ that includes the total number of iterations T such that $\gamma = \frac{1}{\sqrt{T}}$. Even if T is predetermined, this setup may be experimentally unrealistic, since T can be very large depending on the training dataset and the number of epochs. Reddi et al. (2016); Gu et al. (2019) used a batch size determined by $b = T$, but this setting is not realistic from the standpoint of computational complexity because b becomes too large for medium-sized or larger experiments. Some studies (Reddi et al., 2016; Gu et al., 2019; Grigas et al., 2019) also include a Lipschitz constant in the learning rate. Since that information is not available in advance, this setup would also be impractical. A few studies (Négiar

et al., 2020; Combettes et al., 2021) used a traditional learning rate that decreases as $\gamma_t := \frac{2}{t+2}$ depending on time t. This setting may not be suitable for practical use, especially for large-scale optimizations such as DNN training, because the learning rate quickly becomes too small. These analyses do not explain the effectiveness of SFW methods in large-scale optimization of DNNs as is evident in (Miao et al., 2022). We therefore aim to provide an analysis of the convergence of SFW when using constant and decaying learning rates, which would be experimentally realistic. We also aim to provide a similar analysis for SFW with momentum (SFWM), a natural extension of SFW.

2. Are SFW attacks effective against robust models? Chen et al. (2020a) showed that the Frank Wolfe algorithm is effective in adversarial attacks against non-robust models. Therefore, we would like to clarify whether it is effective against robust models. Furthermore, we propose Auto-SFW (ASFW), inspired by Auto-Projected Gradient Descent (APGD) (Croce & Hein, 2020b) and Auto-Conjugate Gradient Descent (ACG) (Yamamura et al., 2022), and clarify its performance for robust models (see Section 4 for details).

1.3 Contribution

1. Practical convergence analysis of SFW and SFW with momentum (Section 3). We provide convergence analyses of SFW (Algorithm 2) and SFW with momentum (Algorithms 3) using a user-defined learning rate that is independent of unknowns that cannot be known a priori. To evaluate convergence, we use the Frank Wolfe gap $\mathcal{G}(\boldsymbol{\theta})$, which is a commonly used measure in convergence analyses of Frank Wolfe algorithms (see Section 2.1 for details).

Let $(\boldsymbol{\theta}_t) \in \mathbb{R}^d$ be the sequence generated by each of SFW (Algorithm 2) and SFWM (Algorithm 3). In Section 3, we will show that, under certain assumptions, the average of $\mathbb{E}[\mathcal{G}(\boldsymbol{\theta}_t)]$ has an upper bound as shown in Table 2, where the constant learning rate is $\gamma_t := \gamma$ and the constant batch size is $b_t = b$. In addition, the decaying learning rate and increasing batch size are defined as follows:

$$\text{Decaying Learning Rate (I): } \gamma_t := \frac{1}{t+1},$$

$$\text{Decaying Learning Rate (II): } \gamma_t := \frac{1}{(t+1)^a} \ (a \in [0.5, 1)),$$

$$\text{Decaying Learning Rate (III): } \gamma_t := (\underbrace{\gamma, \gamma, \cdots, \gamma}_{K}, \underbrace{\eta\gamma, \eta\gamma, \cdots, \eta\gamma}_{K}, \cdots, \underbrace{\eta^{P-1}\gamma, \eta^{P-1}\gamma, \cdots, \eta^{P-1}\gamma}_{K}), \tag{1}$$

$$\text{Increasing Batch Size: } b_t := (\underbrace{b, b, \cdots, b}_{E}, \underbrace{\lambda b, \lambda b, \cdots, \lambda b}_{E}, \cdots, \underbrace{\lambda^{Q-1} b, \lambda^{Q-1} b, \cdots, \lambda^{Q-1} b}_{E}), \tag{2}$$

where $\eta \in (0, 1)$, $\lambda > 1$, $PK = T$, and $QE = T$. $\gamma > 0$ is the initial learning rate and b ($\leq n$) is the initial batch size, where n is the number of training data.

Table 2: Convergence rate of our analysis. (LR: learning rate).

	Constant Batch Size	Increasing Batch Size
Constant LR	$\mathcal{O}\left(\frac{1}{T} + \frac{1}{\sqrt{b}} + \gamma\right)$ (Theorem 3.1)	$\mathcal{O}\left(\frac{1}{T} + \gamma\right)$ (Theorem 3.2)
Decaying LR (I)	$\mathcal{O}\left(\frac{1}{\sqrt{b}}\right)$ (Theorem 3.3)	-
Decaying LR (II)	$\mathcal{O}\left(\frac{1}{T^{\min\{1-a,a\}}} + \frac{1}{\sqrt{b}}\right)$ (Theorerm 3.4(i))	$\mathcal{O}\left(\frac{1}{T^{\min\{1-a,a\}}}\right)$ (Theorem 3.5(i))
Decaying LR (III)	$\mathcal{O}\left(\frac{1}{T} + \frac{1}{\sqrt{b}}\right)$ (Theorem 3.4(ii))	$\mathcal{O}\left(\frac{1}{T}\right)$ (Theorem 3.5(ii))

Note that all of our theorems are common to SFW and SFWM and that the momentum factor does not appear. This is due to our key lemma (Lemma A.3) and is one of our technical contributions. Table 2

shows that the extra term independent of T disappears from the upper bound of gap only when using both decreasing learning rates (II) and (III) and an increasing batch size, resulting in $\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] \to 0$ ($T \to \infty$). In particular, SFW and SFWM have a convergence rate of $\mathcal{O}(1/T)$ when using a decaying learning rate (III) and an increasing batch size. We applied these algorithms to deep-learning training to verify their performance (Section 3.3). Note that SFW with increasing batch sizes has been studied by (Goldfarb et al., 2017; Hazan & Luo, 2016; Reddi et al., 2016) and SGD with increasing batch size also has been well studied by (Byrd et al., 2012; Friedlander & Schmidt, 2012; Balles et al., 2017; De et al., 2017; Bottou et al., 2018b; Smith et al., 2018).

2. Application to adversarial attack (Section 4). Our convergence analysis can be applied to any constrained nonconvex optimization problem. In this paper, the SFW algorithms are used to generate adversarial examples. In Section 4.3, we propose a new adversarial attack method, Auto-SFW (ASFW), an adaptation of the APGD approach to SFW. Furthermore, we show that ASFW has comparable performance to APGD, which itself has state-of-the-art performance, and discuss its limitation in Section 4.5.

2 Preliminaries

Let $\mathbb{N}$ be the set of non-negative integers. For $m \in \mathbb{N} \setminus \{0\}$, define $[m] := \{1, 2, \ldots, m\}$. Let $\mathbb{R}^d$ be a d-dimensional Euclidean space with inner product $\langle \cdot, \cdot \rangle$, which induces the norm $\|\cdot\|$. The DNNs is parameterized by a vector $\boldsymbol{x} \in \mathbb{R}^d$, which is optimized by minimizing the empirical loss function $f(\boldsymbol{\theta}) := \frac{1}{n}\sum_{i\in[n]} f_i(\boldsymbol{\theta})$, where $f_i(\boldsymbol{\theta})$ is the loss function for $\boldsymbol{\theta} \in \mathbb{R}^d$ and the i-th training data $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ ($i \in [n]$). Let ξ be a random variable that does not depend on $\boldsymbol{\theta} \in \mathbb{R}^d$, and let $\mathbb{E}_\xi[X]$ denote the expectation with respect to ξ of a random variable X. $\xi_{t,i}$ is a random variable generated from the i-th sampling at time t, and $\boldsymbol{\xi}_t := (\xi_{t,1}, \xi_{t,2}, \ldots, \xi_{t,b})$ is independent of $(\boldsymbol{\theta}_k)_{k=0}^t \subset \mathbb{R}$, where b ($\leq n$) is the batch size. The independence of $\boldsymbol{\xi}_0, \boldsymbol{\xi}_1, \ldots$ allows us to define the total expectation $\mathbb{E}$ as $\mathbb{E} = \mathbb{E}_{\boldsymbol{\xi}_0}\mathbb{E}_{\boldsymbol{\xi}_1}\cdots\mathbb{E}_{\boldsymbol{\xi}_t}$. Let $\mathsf{G}_{\boldsymbol{\xi}_t}(\boldsymbol{\theta})$ be the stochastic gradient of $f(\cdot)$ at $\boldsymbol{\theta} \in \mathbb{R}^d$. The mini-batch $\mathcal{S}_t$ consists of b_t samples at time t, and the mini-batch stochastic gradient of $f(\boldsymbol{\theta}_t)$ for $\mathcal{S}_t$ is defined as $\nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t) := \frac{1}{b_t}\sum_{i\in[b_t]} \mathsf{G}_{\xi_{t,i}}(\boldsymbol{\theta}_t)$.

We will impose the following conditions, which are standard ones for nonconvex optimization in deep neural networks (see, e.g., (Chen et al., 2019)).

Assumption 2.1.

(A1) *The domain $\Omega \subset \mathbb{R}^d$ is convex and compact with diameter D such that $\forall \boldsymbol{x}, \boldsymbol{y} \in \Omega\colon \|\boldsymbol{x} - \boldsymbol{y}\| \leq D$.*

(A2) *$f_i\colon \mathbb{R}^d \to \mathbb{R}$ ($i \in [n]$) are continuously differentiable and L-smooth on Ω, i.e.,*

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \Omega : \|\nabla f_i(\boldsymbol{x}) - \nabla f_i(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|.$$

(A3) *Let $(\boldsymbol{\theta}_t)_{t\in\mathbb{N}} \subset \mathbb{R}^d$ be the sequence generated by an optimizer.*
(i) *For each iteration t, $\mathbb{E}_{\xi_t}\left[\mathsf{G}_{\xi_t}(\boldsymbol{\theta}_t)\right] = \nabla f(\boldsymbol{\theta}_t)$.*

(ii) *There exists a nonnegative constant σ^2 such that $\mathbb{E}_{\xi_t}\left[\|\mathsf{G}_{\xi_t}(\boldsymbol{\theta}_t) - \nabla f(\boldsymbol{\theta}_t)\|^2\right] \leq \sigma^2$.*

(A4) *For each iteration t, the optimizer samples the mini-batch $\mathcal{S}_t \subset \mathcal{S}$ and estimates the full gradient ∇f as*

$$\nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t) := \frac{1}{b_t}\sum_{i\in[b]} \mathsf{G}_{\xi_{t,i}}(\boldsymbol{\theta}_t) = \frac{1}{b_t}\sum_{\{i\colon (\boldsymbol{x}_i, \boldsymbol{y}_i)\in\mathcal{S}_t\}} \nabla f_i(\boldsymbol{\theta}_t).$$

Problem 2.1. *Under Assumption 2.1, we would like to find a local minimizer of f over Ω, i.e.,*

$$\min_{\boldsymbol{\theta}\in\Omega}\left\{f(\boldsymbol{\theta}) := \frac{1}{n}\sum_{i=1}^n f_i(\boldsymbol{\theta})\right\}.$$

2.1 Stochastic Frank Wolfe algorithm

Algorithm 1 Frank Wolfe

Require: $(\gamma_t)_{t\in\mathbb{N}} \subset \mathbb{R}_{++}$
  $t \leftarrow 0, \boldsymbol{\theta}_0 \in \Omega$
  loop
    $\boldsymbol{v}_t = \underset{\boldsymbol{v}\in\Omega}{\operatorname{argmax}} \langle \boldsymbol{v}, -\nabla f(\boldsymbol{\theta}_t)\rangle$
    $\boldsymbol{d}_t = \boldsymbol{v}_t - \boldsymbol{\theta}_t$
    $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \gamma_t \boldsymbol{d}_t$
    $t \leftarrow t+1$
  end loop

Algorithm 2 Stochastic Frank Wolfe (SFW)

Require: $(\gamma_t)_{t\in\mathbb{N}} \subset \mathbb{R}_{++}$
  $t \leftarrow 0, \boldsymbol{\theta}_0 \in \Omega$
  loop
    $\boldsymbol{v}_t = \underset{\boldsymbol{v}\in\Omega}{\operatorname{argmax}} \langle \boldsymbol{v}, -\nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t)\rangle$
    $\boldsymbol{d}_t = \boldsymbol{v}_t - \boldsymbol{\theta}_t$
    $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \gamma_t \boldsymbol{d}_t$
    $t \leftarrow t+1$
  end loop

The Frank Wolfe algorithm (Frank & Wolfe, 1956) (Algorithm 1) is a classical first-order method. In general, $\|\nabla f(\boldsymbol{\theta})\|$ cannot be used as a measure of convergence in constrained optimization. So instead, the Frank Wolfe gap $\mathcal{G}(\boldsymbol{\theta})$ such that

$$\mathcal{G}(\boldsymbol{\theta}) := \max_{\boldsymbol{v}\in\Omega} \langle \boldsymbol{v} - \boldsymbol{\theta}, -\nabla f(\boldsymbol{\theta})\rangle,$$

is introduced as a measure of convergence. Suppose that $\boldsymbol{\theta}^\star$ is the local minimizer of f over Ω; from the optimality conditions,

$$\forall \boldsymbol{v} \in \Omega : \langle \boldsymbol{v} - \boldsymbol{\theta}^\star, \nabla f(\boldsymbol{\theta}^\star)\rangle \geq 0 \;\text{ i.e. }\; \mathcal{G}(\boldsymbol{\theta}^\star) \leq 0.$$

Therefore, our goal is to make $\mathcal{G}(\boldsymbol{\theta})$ small. We will consider the following optimizer that uses the mini-batch stochastic gradient $\nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t)$ instead of the full gradient $\nabla f(\boldsymbol{\theta}_t)$, described by Algorithm 2, for solving Problem 2.1. In addition, attempts have been made to add a momentum term in several different variants and under many different names (Mokhtari et al., 2018; Chen et al., 2018; Mokhtari et al., 2020; Xie et al., 2020; Chen et al., 2020a; Pokutta et al., 2020). In accordance with Pokutta et al. (2020), we will refer to Algorithm 3 as SFW with momentum (SFWM).

Algorithm 3 SFW with momentum (SFWM)

Require: $(\gamma_t)_{t\in\mathbb{N}} \subset \mathbb{R}_{++}$, $\beta \in [0,1)$
  $t \leftarrow 0, \boldsymbol{\theta}_0 \in \Omega, \boldsymbol{m}_{-1} \leftarrow \mathbf{0}$
  loop
    $\boldsymbol{m}_t = \beta \boldsymbol{m}_{t-1} + (1-\beta)\nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t)$
    $\boldsymbol{v}_t = \underset{\boldsymbol{v}\in\Omega}{\operatorname{argmax}} \langle \boldsymbol{v}, -\boldsymbol{m}_t\rangle$
    $\boldsymbol{d}_t = \boldsymbol{v}_t - \boldsymbol{\theta}_t$
    $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \gamma_t \boldsymbol{d}_t$
    $t \leftarrow t+1$
  end loop

3 Theoretical Main Results

3.1 Convergence analyses of SFW algorithms using a constant learning rate

Assumption 3.1. (C1) $\gamma_t := \gamma$ *for all* $t \in \mathbb{N}$.

The following are convergence analyses of Algorithm 3 with a constant learning rate. The proofs of Theorems 3.1 and 3.2 are given in Appendix B.

Theorem 3.1 (**Constant Learning Rate and Constant Batch Size**). *Suppose that Assumptions 2.1 and 3.1 hold and consider the sequence $(\boldsymbol{\theta}_t)_{t\in\mathbb{N}}$ generated by each of Algorithms 2 and 3 with a constant batch size $b_t := b$. Then, the following holds:*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] \leq \frac{\mathbb{E}\left[f(\boldsymbol{\theta}_0)\right] - \mathbb{E}\left[f(\boldsymbol{\theta}^\star)\right]}{\gamma T} + \frac{D\sigma}{\sqrt{b}} + \frac{LD^2\gamma}{2} \ = \mathcal{O}\left(\frac{1}{T} + \frac{1}{\sqrt{b}} + \gamma\right),$$

where $\boldsymbol{\theta}^\star$ is a local minimizer of f over Ω.

Theorem 3.2 (**Constant Learning Rate and Increasing Batch Size**). *Suppose that Assumptions 2.1 and 3.1 hold and consider the sequence $(\boldsymbol{\theta}_t)_{t\in\mathbb{N}}$ generated by each of Algorithms 2 and 3 with an increasing batch size $b_t := (\underbrace{b, b, \cdots, b}_{E}, \underbrace{\lambda b, \lambda b, \cdots, \lambda b}_{E}, \cdots, \underbrace{\lambda^{Q-1}b, \lambda^{Q-1}b, \cdots, \lambda^{Q-1}b}_{E})$, where $\lambda > 1$ and $QE = T$. Then, the following holds:*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] \leq \frac{\mathbb{E}\left[f(\boldsymbol{\theta}_0)\right] - \mathbb{E}\left[f(\boldsymbol{\theta}^\star)\right]}{\gamma T} + \frac{D\sigma}{T}\frac{E\sqrt{\lambda}}{\sqrt{b}(\sqrt{\lambda}-1)} + \frac{LD^2\gamma}{2} \ = \mathcal{O}\left(\frac{1}{T} + \gamma\right),$$

where $\boldsymbol{\theta}^\star$ is a local minimizer of f over Ω.

3.2 Convergence analyses of SFW algorithms using a decaying learning rate

The following is a convergence analysis of Algorithm 3 with a decaying learning rate (I). The proofs of Theorems 3.3 is given in Appendix C.

Theorem 3.3 (**Decaying Learning Rate (I) and Constant Batch Size**). *Suppose that Assumption 2.1 holds and consider the sequence $(\boldsymbol{\theta}_t)_{t\in\mathbb{N}}$ generated by each of Algorithms 2 and 3 with a constant batch size $b_t := b$. Then, the following is true:*

$$\frac{\sum_{t=0}^{T-1}\gamma_t\mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right]}{\sum_{t=0}^{T-1}\gamma_t} \leq \frac{\mathbb{E}\left[f(\boldsymbol{\theta}_0)\right] - \mathbb{E}\left[f(\boldsymbol{\theta}^\star)\right]}{\sum_{t=0}^{T-1}\gamma_t} + \frac{D\sigma}{\sqrt{b}} + \frac{LD^2}{2}\frac{\sum_{t=0}^{T-1}\gamma_t^2}{\sum_{t=0}^{T-1}\gamma_t},$$

where $\boldsymbol{\theta}^\star$ is a local minimizer of f over Ω. If we use $\gamma_t = \frac{1}{(t+1)}$ such that $\sum_{t=0}^{+\infty}\gamma_t = +\infty$ and $\sum_{t=0}^{+\infty}\gamma_t^2 < +\infty$, then

$$\frac{\sum_{t=0}^{+\infty}\gamma_t\mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right]}{\sum_{t=0}^{+\infty}\gamma_t} = \mathcal{O}\left(\frac{1}{\sqrt{b}}\right).$$

The following are convergence analyses of Algorithm 3 with a decaying learning rate (II) and (III). The proofs of Theorems 3.4 and 3.5 are given in Appendices D and E.

Theorem 3.4 (**Decaying Learning Rate (II, III) and Constant Batch Size**). *Suppose that Assumption 2.1 and the monotone decreasing property of γ_t hold and consider the sequence $(\boldsymbol{\theta}_t)_{t\in\mathbb{N}}$ generated by each of Algorithms 2 and 3 with a constant batch size $b_t := b$. Then, the following is true:*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] \leq \frac{2\max\{\bar{f}, |f(\boldsymbol{\theta}^\star)|\}}{T\gamma_{T-1}} + \frac{D\sigma}{\sqrt{b}} + \frac{LD^2}{2T}\sum_{t=0}^{T-1}\gamma_t,$$

where $\boldsymbol{\theta}^\star$ is a local minimizer of f over Ω, and $\bar{f}$ is an upper bound of f.
(i) *If we use $\gamma_t = \frac{1}{(t+1)^a}(a \in [\frac{1}{2}, 1))$, then*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] \leq \frac{2\max\{\bar{f}, |f(\boldsymbol{\theta}^\star)|\}}{T^{1-a}} + \frac{D\sigma}{\sqrt{b}} + \frac{LD^2}{2(1-a)T^a} = \mathcal{O}\left(\frac{1}{T^{\min\{1-a,a\}}} + \frac{1}{\sqrt{b}}\right).$$

(ii) *If we use* $\gamma_t = (\underbrace{\gamma, \gamma, \cdots, \gamma}_{K}, \underbrace{\eta\gamma, \eta\gamma, \cdots, \eta\gamma}_{K}, \cdots, \underbrace{\eta^{P-1}\gamma, \eta^{P-1}\gamma, \cdots, \eta^{P-1}\gamma}_{K})$, *then*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] \leq \frac{2\max\{\bar{f}, |f(\boldsymbol{\theta}^\star)|\}}{T\underline{\gamma}} + \frac{D\sigma}{\sqrt{b}} + \frac{\gamma_\infty L D^2}{2T} = \mathcal{O}\left(\frac{1}{T} + \frac{1}{\sqrt{b}}\right),$$

where $\eta \in (0,1)$, $PK = T$, $\gamma_{T-1} = \eta^{P-1}\gamma \geq \underline{\gamma}$, *and* $\gamma_\infty := \sum_{t=0}^{T-1}\gamma_t < +\infty$.

Theorem 3.5 (**Decaying Learning Rate (II, III) and Increasing Batch Size**). *Suppose that Assumption 2.1 and the monotone decreasing property of* γ_t *hold and consider the sequence* $(\boldsymbol{\theta}_t)_{t\in\mathbb{N}}$ *generated by each of Algorithms 2 and 3 with an increasing batch size* $b_t := (\underbrace{b, b, \cdots, b}_{E}, \underbrace{\lambda b, \lambda b, \cdots, \lambda b}_{E}, \cdots, \underbrace{\lambda^{Q-1}b, \lambda^{Q-1}b, \cdots, \lambda^{Q-1}b}_{E})$, *where* $\lambda > 1$ *and* $QE = T$. *Then, the following is true:*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] \leq \frac{2\max\{\bar{f}, |f(\boldsymbol{\theta}^\star)|\}}{T\gamma_{T-1}} + \frac{D\sigma}{T}\frac{E\sqrt{\lambda}}{\sqrt{b}(\sqrt{\lambda}-1)} + \frac{LD^2}{2T}\sum_{t=0}^{T-1}\gamma_t,$$

where $\boldsymbol{\theta}^\star$ *is a local minimizer of* f *over* Ω, *and* $\bar{f}$ *is an upper bound of* f.

(i) *If we use* $\gamma_t = \frac{1}{(t+1)^a}(a \in [\frac{1}{2}, 1))$, *then*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] \leq \frac{2\max\{\bar{f}, |f(\boldsymbol{\theta}^\star)|\}}{T^{1-a}} + \frac{D\sigma}{T}\frac{E\sqrt{\lambda}}{\sqrt{b}(\sqrt{\lambda}-1)} + \frac{LD^2}{2(1-a)T^a} = \mathcal{O}\left(\frac{1}{T^{\min\{1-a,a\}}}\right).$$

(ii) *If we use* $\gamma_t = (\underbrace{\gamma, \gamma, \cdots, \gamma}_{K}, \underbrace{\eta\gamma, \eta\gamma, \cdots, \eta\gamma}_{K}, \cdots, \underbrace{\eta^{P-1}\gamma, \eta^{P-1}\gamma, \cdots, \eta^{P-1}\gamma}_{K})$, *then*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] \leq \frac{2\max\{\bar{f}, |f(\boldsymbol{\theta}^\star)|\}}{T\underline{\gamma}} + \frac{D\sigma}{T}\frac{E\sqrt{\lambda}}{\sqrt{b}(\sqrt{\lambda}-1)} + \frac{\gamma_\infty L D^2}{2T} = \mathcal{O}\left(\frac{1}{T}\right),$$

where $\eta \in (0,1)$, $PK = T$, $\gamma_{T-1} = \eta^{P-1}\gamma \geq \underline{\gamma}$, *and* $\gamma_\infty := \sum_{t=0}^{T-1}\gamma_t < +\infty$.

Our theorems were analyzed for the sequence $(\boldsymbol{\theta}_t)$ generated by SFWM (Algorithm 3), but the momentum factor β does not appear in the upper bound for the Frank Wolfe gap, thanks to the key lemma (Lemma A.3). Therefore, Theorems 3.1-3.5 also hold for the sequence $(\boldsymbol{\theta}_t)$ generated by SFW (Algorithm 2).

3.3 Numerical Results

To verify the performance of Algorithms 2 and 3, we conducted numerical experiments on training ResNet18 (He et al., 2016) with the CIFAR100 (Krizhevsky, 2009) dataset under the L_2 constraint. The training took 200 epochs. The initial learning rate γ_0 was 0.1 for all algorithms and the momentum factor β was 0.9 for SFWM. The radius of the L_2 constraint was set to 300. Figure 1 plots the test accuracy and loss function values with the SFW methods. SFW (constant bs), SFWM (constant bs), SVFW (Reddi et al., 2016), SAGAFW (Reddi et al., 2016), and Mokhtari's SFW (Mokhtari et al., 2020) used a constant batch size of 2^{10} and a constant learning rate $\gamma_t = \gamma_0 = 0.1$, i.e., this setting is based on Theorem 3.1. SFW (increasing bs) and SFWM (increasing bs) used batch sizes that quadruple every 40 epochs; i.e., this setting is based on Theorem 3.2. The rising curve is test accuracy and the falling curve is the training loss function.

Figure 2 similarly plots test accuracy and the loss function values for a batch size of 2^{10} and a decaying learning rate that halves every 40 epochs; i.e., this setting is based on Theorem 3.4(ii). SFW (increasing bs) and SFWM (increasing bs) used batch sizes that quadruple every 40 epochs; i.e., this setting is based on Theorem 3.5(ii).

Both figures show that SFWM (constant bs) achieve higher test accuracy and lower loss function values than those of SFW (constant bs). Note, however, that this is due to the large batch size. We have performed

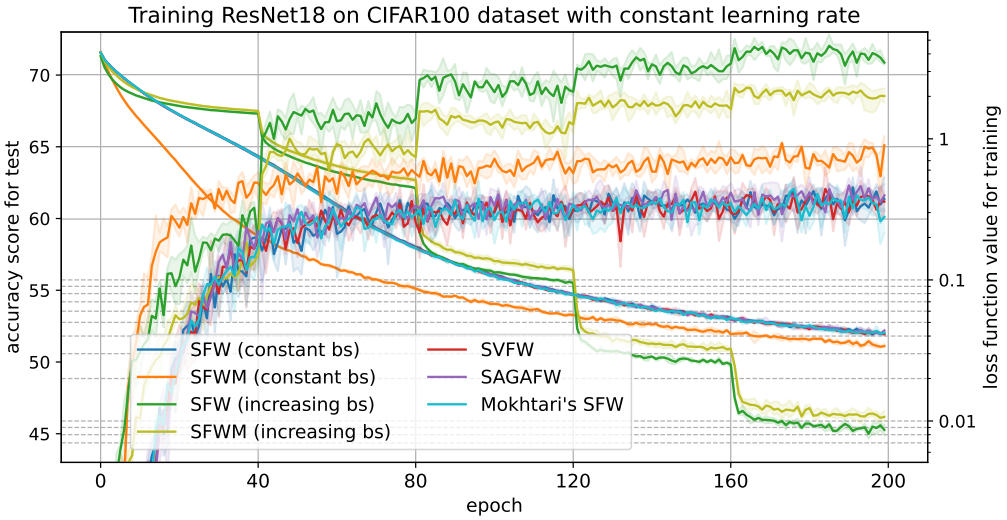

Figure 1: Accuracy score for the testing and loss function value for training versus the number of epochs in training ResNet18 on the CIFAR100 dataset with the L_2 constraint and a constant learning rate. The solid line represents the mean value, and the shaded area represents the maximum and minimum over three runs. The batch size was increased every 40 epochs as $[8, 32, 128, 512, 2048]$ for SFW (increasing bs) and SFWM (increasing bs). (bs: batch size).

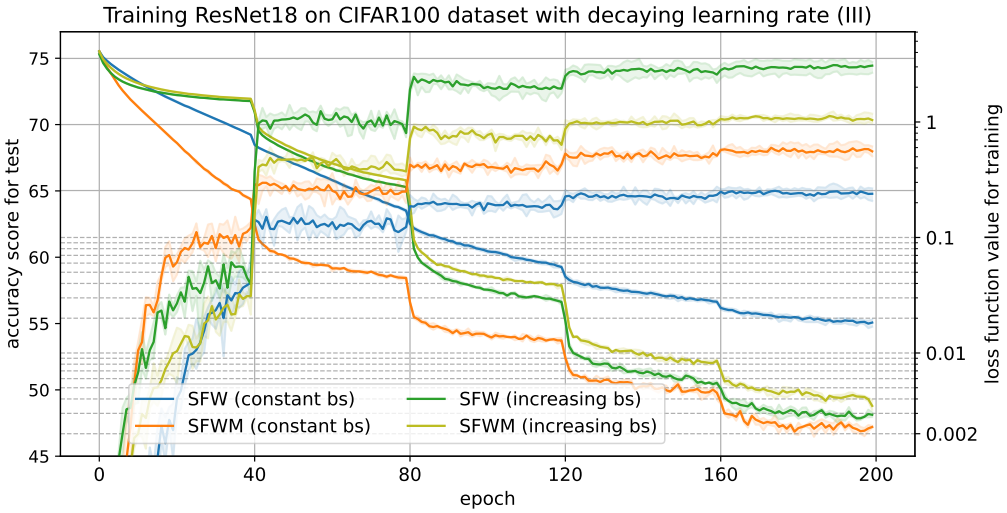

Figure 2: Accuracy score for the testing and loss function value for training versus the number of epochs in training ResNet18 on the CIFAR100 dataset with the L_2 constraint and a decaying learning rate. The solid line represents the mean value, and the shaded area represents the maximum and minimum over three runs. For SFW and SFWM, the learning rate was decreased every 40 epochs as $[0.1, 0.05, 0.025, 0.0125, 0.00625]$ and batch size was fixed at 2^{10}. The batch size was increased every 40 epochs as $[8, 32, 128, 512, 2048]$ and the learning rate was decreased using the same rule for SFW (increasing bs) and SFWM (increasing bs). (bs: batch size).

similar experiments with different batch sizes and observed that the exact opposite results are obtained when the batch size is small (see Figures 3 and 4 in Appendix F.1). Thus, for SFW, adding momentum helps when the batch size is large, but has the opposite effect when the batch size is small. This is the same phenomenon observed in SGD and SGD with momentum that some previous studies have observed experimentally (Shallue et al., 2019; Jelassi & Li, 2022; Kunstner et al., 2023). In addition, Figures 1 and 2 show that, for all methods, the decaying learning rate achieves higher test accuracy and lower loss function values than the constant learning rate does. This finding is theoretically supported by Theorems 3.1 and

3.4(ii) (see also Table 2). Furthermore, Figures 1 and 2 show that, for SFW, the increasing batch size achieves higher test accuracy and lower loss function values than the constant batch size does. This is also theoretically supported by Theorems 3.1,3.2,3.4(ii), and 3.5(ii). For SFWM, however, using a decaying learning rate and increasing batch size improves the test accuracy, but worsens the loss function value. This cannot be explained by our theorem and is a limitation of our theory.

We also performed similar experiments for decaying learning rates (I) $\gamma_t := \frac{1}{t+1}$ based on Theorem 3.3 and (II) $\gamma_t := \frac{1}{\sqrt{t+1}}$ based on Theorem 3.4(i) (see Figures 5 and 6 in Appendix F.1) and observed their limitations.

There are not many constrained nonconvex optimization problems in the machine-learning field, and applications of SFW methods to them may be limited. In fact, the ResNet18 training we have performed in this section does not need to be constrained, and unconstrained optimization algorithms such as SGD are sufficient.

4 Application to Adversarial Attacks

In this section, we explain that the optimization problem for the adversarial attack is an instance of a constrained non-convex optimization problem and experimentally verify whether this attack succeeds with SFW methods.

Deep neural networks are vulnerable to adversarial examples (Szegedy et al., 2014; Goodfellow et al., 2015), which are generated by adding perturbations to real images that are too small for the human eye to perceive. Improving the robustness of classifiers to adversarial examples has become one of the most studied topics in the machine learning community. Adversarial training (Goodfellow et al., 2015), which includes adversarial examples in the training data, is effective in improving robustness. Generating adversarial examples is also important for robust model development because adversarial examples can be used to evaluate the robustness of a model. An adversarial attack is a method to create adversarial examples, and various algorithms have been proposed. Depending on the amount of information an attacker has access to, adversarial attacks can be divided into white-box attacks (Szegedy et al., 2014; Goodfellow et al., 2015) and black-box attacks (Papernot et al., 2016; Chen et al., 2017). In a white-box attack, the attacker has access to the complete information including the weights of the target model, while in a black-box attack, the adversary has access only to the inputs and outputs of the target model. In this paper, we focus on white-box attacks.

4.1 White-box attack

For white-box attacks, Szegedy et al. (2014) proposed to use the box-constrained limited-memory Broyden-Fletcher-Goldferb-Shanno (L-BFGS) algorithm. Goodfellow et al. (2015) proposed the fast gradient sign method (FGSM) to overcome the speed limitation of L-BFGS. Kurakin et al. (2016) proposed an iterative-FGSM (I-FGSM) algorithm that iterates over the one-iteration FGSM algorithm. Madry et al. (2018) showed that I-FGSM with the L_∞ norm is approximately equivalent to projected gradient descent (PGD). PGD is the most popular attack because it is computationally inexpensive and has been successful in many cases. Auto-PGD (APGD) (Croce & Hein, 2020a) is an attack that searches for perturbations while dynamically varying the learning rate of PGD. In this paper, SFW is used to attack. Note that Chen et al. (2020a); Sahu & Kar (2020); Huang et al. (2020a); Kazemi et al. (2021); Imtiaz et al. (2022) tackled adversarial attacks using the Frank Wolfe algorithm or its variants.

4.2 Problem setting

Given a training dataset $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) | \boldsymbol{x}_i \in \mathbb{R}^d, \boldsymbol{y}_i \in \mathbb{R}^c\}_{i=0}^{n-1}$ drawn i.i.d. from the distribution, where $c \in \mathbb{N}$ is the number of classes in the training dataset, consider a classification with a neural network $g(\boldsymbol{x}, \boldsymbol{\theta})$. Here, $\boldsymbol{x}_i \in \mathbb{R}^d$ is the real image, $\boldsymbol{y}_i$ is the label for $\boldsymbol{x}_i$, and $\boldsymbol{\theta}$ is the parameter of the model g. Let $g(\boldsymbol{x})$ be the predicted labels of the classifier and $f_i(g(\boldsymbol{x}_i), \boldsymbol{y}_i)$ be the loss function. The parameters of the neural network are updated to minimize $f(\boldsymbol{\theta}) := \frac{1}{n} \sum_{i=0}^{n-1} f_i(g(\boldsymbol{x}_i), \boldsymbol{y}_i)$. When training is completed, the classifier g will be able to classify the input image $\boldsymbol{x}$ with high accuracy. We aim to skew the predictions of the trained classifier by adding a miniscule amount of noise to the input image $\boldsymbol{x}$. Given a distance

function $d(\cdot,\cdot)$ and a positive number $\epsilon > 0$, the generated adversarial example $\boldsymbol{x}_{\mathrm{adv}}$ can be expressed as $\boldsymbol{x}_{\mathrm{adv}} \in \{\hat{\boldsymbol{x}}_i \mid g(\hat{\boldsymbol{x}}_i) \neq y_i, d(\hat{\boldsymbol{x}}_i, \boldsymbol{x}_i) \leq \epsilon\}$. The adversarial attack can be formulated as follows:

$$\max_{\hat{\boldsymbol{x}} \in \mathbb{R}^d} f(g(\hat{\boldsymbol{x}}), \boldsymbol{y}) \text{ s.t. } d(\hat{\boldsymbol{x}}, \boldsymbol{x}) \leq \epsilon.$$

This is an example of Problem 2.1, and the constraint set Ω in Problem 2.1 can be expressed as $\Omega = \{\hat{\boldsymbol{x}} \in \mathbb{R}^d \mid d(\hat{\boldsymbol{x}}, \boldsymbol{x}) \leq \epsilon\}$, which satisfies Assumption (A1). In experiments, the Euclidean norm or maximum norm is often chosen as the distance function $d(\cdot,\cdot)$. ϵ is important because it controls the amount of noise that can be added to the real images. The larger ϵ is, the more it can distort the real image and increase the success rate of the attack. However, since we want to add noise that is imperceptible to humans, we should use a small value of ϵ. In experiments, ϵ is often set to a small value such as 8/255 or 4/255. See Section 4.5 for the detailed experimental setup.

4.3 Auto-Stochastic Frank Wolfe Attack

We propose the Auto-Stochastic Frank Wolfe (ASFW) attack as a new adversarial attack derived from the SFW approach. The proposed scheme is summarized in Algorithm 4. In the part of Algorithm 4 that

Algorithm 4 Auto-Stochastic Frank Wolfe

Require: $f, \Omega, \boldsymbol{x}_0, \gamma_0, N_{iter}, W = \{w_0, \cdots, w_n\}$
  $\boldsymbol{x}_{\mathrm{adv}} \leftarrow \boldsymbol{x}_0, \boldsymbol{m}_{-1} \leftarrow \boldsymbol{0}$
  for $t = 0$ to $N_{iter} - 1$ **do**
    $\boldsymbol{v}_t = \underset{\boldsymbol{v} \in \Omega}{\operatorname{argmax}} \langle \boldsymbol{v}, \nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t) \rangle$
    $\boldsymbol{d}_t = \boldsymbol{v}_t - \boldsymbol{\theta}_t$
    $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \gamma_t \boldsymbol{d}_t$
    if $f(\boldsymbol{x}_{t+1}) > f(\boldsymbol{x}_{\mathrm{adv}})$ **then**
      $\boldsymbol{x}_{\mathrm{adv}} \leftarrow \boldsymbol{x}_{t+1}$
    end if
    if $t \in W$ **then**
      if Condition (i) or (ii) is satisfied **then**
        $\gamma_{t+1} \leftarrow \gamma_t / 2$
        $\boldsymbol{x}_{t+1} \leftarrow \boldsymbol{x}_{\mathrm{adv}}$
      end if
    end if
  end for

calculates $\boldsymbol{v}_t$, $\langle \boldsymbol{v}, \nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t) \rangle$ is used instead of $\langle \boldsymbol{v}, -\nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t) \rangle$. This is because we are trying to increase the function value. In Algorithm 4, the method proposed in APGD is used for step size selection (Croce & Hein, 2020b). The initial step size γ_0 is set to 2ϵ. When the number of iterations reaches the pre-calculated checkpoint $w_j \in W$, the step size γ_t is halved if either of the following two conditions is satisfied. These conditions indicate that the attack is not going well.

(i) $N_c < \rho \cdot (w_j - w_{j-1})$,
(ii) $\gamma_{w_{j-1}} = \gamma_{w_j}$ and $f_{\max}^{(w_{j-1})} = f_{\max}^{(w_j)}$,
where $N_c := \#\{i = w_{j-1}, \cdots, w_j - 1 \mid f(\boldsymbol{x}_{t+1}) > f(\boldsymbol{x}_t)\}$, $\rho > 0$, and $f_{\max}^{(t)} := \max\{f(\boldsymbol{x}_k) | k = 0, \cdots, t\}$.

4.4 Adversarial robustness

Many methods have been proposed to improve the robustness of models, such as gradient regularization (Ross & Doshi-Velez, 2018), curvature regularization (Moosavi-Dezfooli et al., 2018; Huang et al., 2020c), randomized smoothing (Cohen et al., 2019), local linearization (Qin et al., 2019), and adversarial training (Goodfellow et al., 2015; Madry et al., 2018; Zhang et al., 2019b; Wu et al., 2020; Wang et al., 2020; Zhang et al., 2020; Huang et al., 2021b; Balaji et al., 2019; Zhang et al., 2021; Pang et al., 2022; Wang et al., 2023). Among these methods, adversarial training is the most standard one to improve adversarial robustness. It

improves robustness by learning adversarial examples in addition to the usual training data. For non-robust models, almost all methods can achieve a classification accuracy of 0% (for example, see (Chen et al., 2020a, Table 1 and 2)). So in this paper, we evaluate the attack performance of SFW algorithms by attacking robust models trained with adversarial training and compare it with PGD.

4.5 Numerical Results

To evaluate the attack performance of ASFW, we conducted an experiment comparing the performance of APGD and ASFW against state-of-the-art robust models listed in RobustBench (Croce et al., 2021). Table 3 shows the results of APGD and ASFW intercepting the classification task of the CIFAR100 dataset with several robust models. The loss function was the cross entropy loss. The constraint set was L_∞ with diameter $\epsilon = 8/255$ and the attack was executed over the course of 100 steps. Clean accuracy refers to the classification accuracy achieved by the model before the attack, while adversarial accuracy refers to the classification accuracy achieved by the model after the attack. Therefore, a lower adversarial accuracy implies a more successful attack. Table 3 shows that ASFW has an attack performance almost equal to that of APGD, a state-of-the-art attack method. We also performed PGD, SFW, APGD, and ASFW attacks on the image classification task of the CIFAR10, CIFAR100, and ImageNet dataset and obtained similar results (see Tables 4-7 in Appendix F.2). Throughout all of the experimental results, the difference in performance between ASFW and APGD is less than 1 percent. This is also the case in previous study that proposed ACG (Yamamura et al., 2022) and may be due to the very small size of the constraint set in an adversarial attack. Although we have indeed shown that SFW algorithms are effective for adversarial attacks, the choice of optimization method may not be so important for adversarial attacks.

Table 3: Adversarial accuracy achieved by APGD and ASFW. CIFAR100 dataset/L_∞ with $\epsilon = 8/255$

paper	Architecture	clean accuracy	APGD	ASFW
(Wang et al., 2023)	WideResNet-70-16	75.22	48.11	48.16
(Wang et al., 2023)	WideResNet-28-10	72.58	44.05	44.12
(Debenedetti et al., 2023)	XCiT-L12	70.76	38.97	39.02
(Rebuffi et al., 2021)	WideResNet-70-16	63.56	38.28	38.31
(Debenedetti et al., 2023)	XCiT-M12	69.21	38.69	38.77
(Pang et al., 2022)	WideResNet-70-16	65.56	36.66	36.71
(Debenedetti et al., 2023)	XCiT-S12	67.34	37.08	37.06
(Rebuffi et al., 2021)	WideResNet-28-16	62.41	35.73	35.8
(Jia et al., 2022)	WideResNet-34-20	67.31	36.46	36.62
(Addepalli et al., 2022a)	WideResNet-34-10	68.75	36.84	36.92
(Cui et al., 2021)	WideResNet-34-10	62.97	37.05	37.17
(Sehwag et al., 2022)	WideResNet-34-10	65.93	35.77	35.89
(Pang et al., 2022)	WideResNet-28-10	63.66	35.25	35.26
(Jia et al., 2022)	WideResNet-34-10	64.89	35.28	35.53
(Addepalli et al., 2022b)	WideResNet-34-10	65.73	35.64	36.07
(Cui et al., 2021)	WideResNet-34-20	62.55	34.08	34.27
(Cui et al., 2021)	WideResNet-34-10	60.64	33.99	34.10
(Rade & Moosavi-Dezfooli, 2021)	PreActResNet-18	61.50	32.58	32.61
(Wu et al., 2020)	WideResNet-34-10	60.38	33.25	33.34
(Rebuffi et al., 2021)	PreActResNet-18	56.87	31.77	31.8
(Hendrycks et al., 2019)	WideResNet-28-10	59.23	32.88	33.01
(Addepalli et al., 2022a)	ResNet-18	65.45	33.55	33.68
(Cui et al., 2021)	WideResNet-34-10	70.25	29.98	30.02
(Addepalli et al., 2022b)	PreActResNet-18	62.02	32.88	32.94
(Chen et al., 2022)	WideResNet-34-10	62.15	30.98	31.13
(Rice et al., 2020)	PreActResNet-18	53.83	20.63	20.57

5 Conclusions

We provided a practical convergence analysis of SFW and SFW with momentum with a constant or decaying learning rate for solving constrained nonconvex optimization problems. In our analysis, the learning rate and the batch size are independent of unknown parameters and are experimentally realistic. We showed that the Frank Wolfe gap has a convergence rate of $\mathcal{O}(1/T)$ only when we decrease the learning rate and increase the batch size. Our numerical experiments show that SFW with momentum outperforms SFW in both test accuracy and loss function value when the batch size is large and that using a decaying learning rate and increasing batch size achieves higher test accuracy and lower loss function values than does using a constant learning rate in image classification tasks with ResNet18. We also showed experimentally that SFW algorithms perform as well as PGD in adversarial attacks and that our proposed ASFW performs as well as APGD.

References

Sravanti Addepalli, Samyak Jain, and Venkatesh Babu R. Efficient and effective augmentation strategy for adversarial training. In *Advances in Neural Information Processing Systems 36*, 2022a.

Sravanti Addepalli, Samyak Jain, Gaurang Sriramanan, and R. Venkatesh Babu. Scaling adversarial training to large perturbation bounds. In *European Conferene on Computer Vision*, volume 13665, pp. 301–316, 2022b.

Maksym Andriushchenko and Nicolas Flammarion. Understanding and improving fast adversarial training. In *Advances in Neural Information Processing Systems 34*, 2020.

Yogesh Balaji, Tom Goldstein, and Judy Hoffman. Instance adaptive adversarial training: Improved accuracy tradeoffs in neural nets. `https://arxiv.org/abs/1910.08051`, 2019.

Lukas Balles, Javier Romero, and Philipp Hennig. Coupling adaptive batch sizes with learning rates. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence*, 2017.

Leonard Berrada, Andrew Zisserman, and M. Pawan Kumar. Deep Frank-Wolfe For Neural Network Optimization. *International Conference on Learning Representations*, 2019.

Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018a.

Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018b.

Richard H. Byrd, Gillian M. Chin, Jorge Nocedal, and Yuchen Wu. Sample size selection in optimization methods for machine learning. *Mathematical Programming*, 134(1):127–155, 2012.

Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C. Duchi, and Percy Liang. Unlabeled data improves adversarial robustness. In *Advances in Neural Information Processing Systems 32*, pp. 11190–11201, 2019.

Jinghui Chen, Dongruo Zhou, Jinfeng Yi, and Quanquan Gu. A frank-wolfe framework for efficient and effective adversarial attacks. In *the 34th AAAI Conference on Artificial Intelligence*, pp. 3486–3494, 2020a.

Jinghui Chen, Dongruo Zhou, Yiqi Tang, Ziyan Yang, Yuan Cao, and Quanquan Gu. Closing the generalization gap of adaptive gradient methods in training deep neural networks. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, volume 452, pp. 3267–3275, 2021.

Jinghui Chen, Yu Cheng, Zhe Gan, Quanquan Gu, and Jingfeng Zhang. Efficient robust training via backward smoothing. In *Thirty-Sixth AAAI Conference on Artificial Intelligence*, pp. 6222–6230, 2022.

Lin Chen, Christopher Harshaw, Hamed Hassani, and Amin Karbasi. Projection-free online optimization with stochastic gradient: From convexity to submodularity. In *Proceedings of 35th International Conference on Machine Learning*, volume 80, pp. 813–822, 2018.

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 15–26, 2017.

Tianlong Chen, Sijia Liu, Shiyu Chang, Yu Cheng, Lisa Amini, and Zhangyang Wang. Adversarial robustness: From self-supervised pre-training to fine-tuning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 696–705, 2020b.

Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of Adam-type algorithms for non-convex optimization. In *Proceedings of the International Conference on Learning Representations*, 2019.

Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 1310–1320, 2019.

Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter L. Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *Journal of Machine Learning Research*, 9:1775–1822, 2008.

Cyrille W. Combettes, Christoph Spiegel, and Sebastian Pokutta. Projection-Free Adaptive Gradients for Large-Scale Optimization. `https://arxiv.org/pdf/2009.14114.pdf`, 2021.

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*, volume 119, pp. 2206–2216, 2020a.

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*, volume 119, pp. 2206–2216, 2020b.

Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. RobustBench: a standardized adversarial robustness benchmark. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1*, 2021.

Jiequan Cui, Shu Liu, Liwei Wang, and Jiaya Jia. Learnable boundary guided adversarial training. In *2021 IEEE/CVF International Conference on Computer Vision*, pp. 15701–15710, 2021.

Sihui Dai, Saeed Mahloujifar, and Prateek Mittal. Parameterizing activation functions for adversarial robustness. In *2022 IEEE Security and Privacy Workshops (SPW)*, pp. 80–87, 2022.

Soham De, Abhay Kumar Yadav, David W. Jacobs, and Tom Goldstein. Automated inference with adaptive batches. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54, pp. 1504–1513, 2017.

Edoardo Debenedetti, Vikash Sehwag, and Prateek Mittal. A light recipe to train robust vision transformers. In *First IEEE Conference on Secure and Trustworthy Machine Learning*, 2023.

Gavin Weiguang Ding, Yash Sharma, Kry Yik Chau Lui, and Ruitong Huang. Max-margin adversarial (MMA) training: Direct input space margin maximization through adversarial training. In *Proceedings of the 6th International Conference on Learning Represantations*, 2018.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

Pavel Dvurechensky, Kamil Safin, Shimrit Shtern, and Mathias Staudigl. Generalized self-concordant analysis of Frank–Wolfe algorithms. *Mathematical Programming*, 198(1):255–323, 2023.

Benjamin Fehrman, Benjamin Gess, and Arnulf Jentzen. Convergence rates for the stochastic gradient descent method for non-convex objective functions. *Journal of Machine Learning Research*, 21:1–48, 2020.

Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 1956.

Robert M. Freund and Paul Grigas. New analysis and results for the Frank-Wolfe method. *Mathematical Programming*, 155(1):199–230, 2014.

Robert M. Freund, Paul Grigas, and Rahul Mazumder. An extended frank–wolfe method with "in-face" directions, and its application to low-rank matrix completion. *SIAM Journal on Optimization*, 27(1): 319–346, 2017.

Michael P. Friedlander and Mark Schmidt. Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal on Scientific Computing*, 34(3), 2012.

Donald Goldfarb, Garud Iyengar, and Chaoxu Zhou. Linear Convergence of Stochastic Frank Wolfe Variants. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54, pp. 1066–1074, 2017.

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

Sven Gowal, Sylvestre-Alvise Rebuffi, Olivia Wiles, Florian Stimberg, Dan Andrei Calian, and Timothy Mann. Improving robustness using generated data. In *Advances in Neural Information Processing Systems*, volume 34, pp. 4218–4233, 2021.

Paul Grigas, Alfonso Lobos, and Nathan Vermeersch. Stochastic In-Face Frank-Wolfe Methods for Non-Convex Optimization and Sparse Neural Network Training. `https://arxiv.org/pdf/1906.03580.pdf`, 2019.

Bin Gu, Wenhan Xian, and Heng Huang. Asynchronous Stochastic Frank-Wolfe Algorithms for Non-Convex Optimization. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 737–743, 2019.

Elad Hazan and Haipeng Luo. Variance-reduced and projection-free stochastic optimization. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pp. 1263–1271, 2016.

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

Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 2712–2721, 2019.

Feihu Huang, Lue Tao, and Songcan Chen. Accelerated stochastic gradient-free and projection-free methods. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 4519–4530, 2020a.

Hanxun Huang, Yisen Wang, Sarah M. Erfani, Quanquan Gu, James Bailey, and Xingjun Ma. Exploring architectural ingredients of adversarially robust deep neural networks. In *Advances in Neural Information Processing Systems 34*, pp. 5545–5559, 2021a.

Lang Huang, Chao Zhang, and Hongyang Zhang. Self-adaptive training: beyond empirical risk minimization. In *Advances in Neural Information Processing Systems 33*, 2020b.

Tianjin Huang, Vlado Menkovski, Yulong Pei, and Mykola Pechenizky. Bridging the performance gap between FGSM and PGD adversarial training. https://arxiv.org/abs/2011.05157, 2020c.

Tianjin Huang, Vlado Menkovski, Yulong Pei, and Mykola Pechenizky. Calibrated adversarial training. In *Proceedings of the 13th Asian Conference on Machine Learning*, volume 157, pp. 626–641, 2021b.

Hideaki Iiduka. Appropriate learning rates of adaptive learning rate optimization algorithms for training deep neural networks. *IEEE Transactions on Cybernetics*, 52(12):13250–13261, 2022.

Andrew Ilyas, Logan Engstrom, Shibani Santurkar, Dimitris Tsipras, and Hadi Salman. Robustness library. https://github.com/MadryLab/robustness, 2019.

Tooba Imtiaz, Morgan Kohler, Jared Miller, Zifeng Wang, Mario Sznaier, Octavia Camps, and Jennifer Dy. Saif: Sparse adversarial and interpretable attack framework. , 2022.

Martin Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pp. 427–435, 2013.

Samy Jelassi and Yuanzhi Li. Towards understanding how momentum improves generalization in deep learning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 9965–10040, 2022.

Xiaojun Jia, Yong Zhang, Baoyuan Wu, Ke Ma, Jue Wang, and Xiaochun Cao. LAS-AT: adversarial training with learnable attack strategy. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13388–13398, 2022.

Qiyu Kang, Yang Song, Qinxu Ding, and Wee Peng Tay. Stable neural ode with lyapunov-stable equilibrium points for defending against adversarial attacks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 14925–14937, 2021.

Ehsan Kazemi, Thomas Kerdreux, and Liquang Wang. Generating structured adversarial attacks using frank-wolfe method. https://arxiv.org/abs/2102.07360, 2021.

Diederik P Kingma and Jimmy Lei Ba. A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, pp. 1–15, 2015.

Alex Krizhevsky. Learning multiple layers of features from tiny images. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf, 2009.

Frederik Kunstner, Jacques Chen, Jonathan Wilder Lavington, and Mark Schmidt. Noise is not the main factor behind the gap between SGD and adam on transformers, but sign descent might be. In *Proceedings of the 8th International Conference on Learning Representations*, 2023.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. https://arxiv.org/abs/1607.02533, 2016.

Simon Lacoste-Julien and Martin Jaggi. On the global linear convergence of Frank-Wolfe optimization variants. In *Advances in Neural Information Processing Systems*, pp. 496–504, 2015.

E.S. Levitin and Boris T. Polyak. Constrained minimization methods. *USSR Computational Mathematics and Mathematical Physics*, 6(5):1–50, 1966.

Chang Liu, Yinpeng Dong, Wenzhao Xiang, Xiao Yang, Hang Su, Jun Zhu, Yuefeng Chen, Yuan He, Hui Xue, and Shibao Zheng. A comprehensive study on robustness of image classification models: Benchmarking and rethinking. https://arxiv.org/abs/2302.14301, 2023.

Francesco Locatello, Rajiv Khanna, Michael Tschannen, and Martin Jaggi. A Unified Optimization View on Generalized Matching Pursuit and Frank-Wolfe. In Aarti Singh and Xiaojin (Jerry) Zhu (eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54, pp. 860–868, 2017.

Francesco Locatello, Alp Yurtsever, Olivier Fercoq, and Volkan Cevher. Stochastic Frank-Wolfe for Composite Convex Minimization. In *Advances in Neural Information Processing Systems 32*, 2019.

Nicolas Loizou, Sharan Vaswani, Issam Laradji, and Simon Lacoste-Julien. Stochastic polyak step-size for SGD: An adaptive learning rate for fast convergence: An adaptive learning rate for fast convergence. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 130, 2021.

Haihao Lu and Robert M. Freund. Generalized stochastic Frank–Wolfe algorithm with stochastic "substitute" gradient for structured convex optimization. *Mathematical Programming*, 187(1):317–349, 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 Represantations*, 2018.

Lu Miao, Xiaolong Luo, Tianlong Chen, Wuyang Chen, Dong Liu, and Zhangyang Wang. Learning pruning-friendly networks via Frank-Wolfe: One-shot, any-sparsity, and no retraining. In *Proceedings of the 10th International Conference on Learning Representations*, 2022.

Aryan Mokhtari, S. Hamed Hassani, and Amin Karbasi. Conditional gradient method for stochastic submodular maximization: Closing the gap. In *International Conference on Artificial Intelligence and Statistics*, volume 84, pp. 1886–1895, 2018.

Aryan Mokhtari, Hamed Hassani, and Amin Karbasi. Stochastic conditional gradient methods: From convex minimization to submodular maximization. *Journal of Machine Learning Research*, 21(105):1–49, 2020.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard. Robustness via curvature regularization, and vice versa. `http://arxiv.org/abs/1811.09716`, 2018.

Ruslan Nazykov, Aleksandr Shestakov, Vladimir Solodkin, Aleksandr Beznosikov, Gauthier Gidel, and Alexander V. Gasnikov. Stochastic Frank-Wolfe: Unified analysis and zoo of special cases. In *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics*, volume 238, pp. 4870–4878, 2024.

Geoffrey Négiar, Gideon Dresdner, Alicia Tsai, Laurent El Ghaoui, Francesco Locatello, and Fabian Pedregosa. Stochastic Frank-Wolfe for Constrained Finite-Sum Minimization. *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Yurii Evgen'evich Nesterov. A method for unconstrained convex minimization problem with the rate of convergence $\mathcal{O}(1/k^2)$. *Doklady AN USSR*, 269:543–547, 1983.

Tianyu Pang, Xiao Yang, Yinpeng Dong, Kun Xu, Hang Su, and Jun Zhu. Boosting adversarial training with hypersphere embedding. In *Advances in Neural Information Processing Systems 33*, 2020.

Tianyu Pang, Min Lin, Xiao Yang, Jun Zhu, and Shuicheng Yan. Robustness and accuracy could be reconcilable by (Proper) definition. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 17258–17277, 2022.

Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. `https://arxiv.org/abs/1605.07277`, 2016.

Sebastian Pokutta, Christoph Spiegel, and Max Zimmer. Deep Neural Network Training with Frank-Wolfe. `https://arxiv.org/pdf/2010.07243.pdf`, 2020.

Boris T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4:1–17, 1964.

Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli. Adversarial robustness through local linearization. `http://arxiv.org/abs/1907.02610`, 2019.

Rahul Rade and Seyed-Mohsen Moosavi-Dezfooli. Helper-based adversarial training: Reducing excessive margin to achieve a better accuracy vs. robustness trade-off. In *International Conference on Machine Learning 2021 Workshop on A Blessing in Disguise: The Prospects and Perlis of Adversarial Machine Learning*, 2021.

Sylvestre-Alvise Rebuffi, Sven Gowal, Dan A. Calian, Florian Stimberg, Olivia Wiles, and Timothy Mann. Fixing data augmentation to improve adversarial robustness. `https://arxiv.org/abs/2103.01946`, 2021.

Sashank J. Reddi, Suvrit Sra, Barnabás Póczos, and Alex Smola. Stochastic Frank-Wolfe Methods for Nonconvex Optimization. In *54th Annual Allerton Conference on Communication, Control, and Computing*, pp. 1214–1251, 2016.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of Adam and beyond. *Proceedings of The International Conference on Learning Representations*, 2018.

Leslie Rice, Eric Wong, and Zico Kolter. Overfitting in adversarially robust deep learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 8093–8104, 2020.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22:400–407, 1951.

Andrew Slavin Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pp. 1660–1669, 2018.

Anit Kumar Sahu and Soummya Kar. Decentralized zeroth-order constrained stochastic optimization algorithms: Frank–wolfe and variants with applications to black-box adversarial attacks. *Proceedings of the IEEE*, 108(11):1890–1905, 2020.

Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? In *Advances in Neural Information Processing Systems 33*, 2020.

Kevin Scaman and Cedric Malherbe. Robustness analysis of non-convex stochastic gradient descent using biased expectations. In *Advances in Neural Information Processing Systems*, volume 33, pp. 16377–16387, 2020.

Vikash Sehwag, Shiqi Wang, Prateek Mittal, and Suman Jana. HYDRA: pruning adversarially robust neural networks. In *Advances in Neural Information Processing Systems 33*, 2020.

Vikash Sehwag, Saeed Mahloujifar, Tinashe Handina, Sihui Dai, Chong Xiang, Mung Chiang, and Prateek Mittal. Robust learning meets generative models: Can proxy distributions improve adversarial robustness? In *Proceedings of the International Conference on Learning Representations*, 2022.

Christopher J. Shallue, Jaehoon Lee, Joseph M. Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E. Dahl. Measuring the effects of data parallelism on neural network training. *Journal of Machine Learning Research*, 20(112):1–49, 2019.

Naman D Singh, Francesco Croce, and Matthias Hein. Revisiting adversarial training for imagenet: Architectures, training and generalization across threat models. `https://arxiv.org/abs/2303.01870`, 2023.

Samuel L. Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V. Le. Don't decay the learning rate, increase the batch size. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

Kaustubh Sridhar, Oleg Sokolsky, Insup Lee, and James Weimer. Improving neural network robustness via persistency of excitation. In *American Control Conference*, pp. 1521–1526, 2022.

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

Tongyi Tang, Krishna Balasubramanian, and Thomas Chun Man Lee. High-probability bounds for robust stochastic Frank-Wolfe algorithm. In *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180, pp. 1917–1927, 2022.

Tijmen Tieleman and Geoffrey Hinton. Rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4:26–31, 2012.

Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *Proceedings of the 8th International Conference on Learning Represantations*, 2020.

Zekai Wang, Tianyu Pang, Chao Du, Min Lin, Weiwei Liu, and Shuicheng Yan. Better diffusion models further improve adversarial training. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 36246–36263, 2023.

Elias Wirth, Thomas Kerdreux, and Sebastian Pokutta. Acceleration of Frank-Wolfe Algorithms with Open Loop Step-Sizes. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, 2023.

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 Represantations*, 2020.

Dongxian Wu, Shu-Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. In *Advances in Neural Information Processing Systems 33*, 2020.

Jiahao Xie, Zebang Shen, Boyu Wang, and Hui Qian. Efficient projection-free online methods with stochastic recursive gradient. In *the 34th AAAI Conference on Artificial Intelligence*, pp. 6446–6453, 2020.

Yuancheng Xu, Yanchao Sun, Micah Goldblum, Tom Goldstein, and Furong Huang. Exploring and exploiting decision boundary dynamics for adversarial robustness. In *Proceedings of the International Conference on Learning Representations*, 2023.

Keiichiro Yamamura, Haruki Sato, Nariaki Tateiwa, Nozomi Hata, Toru Mitsutake, Issa Oe, Hiroki Ishikura, and Katsuki Fujisawa. Diversified adversarial attacks based on conjugate gradient method. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 24872–24894, 2022.

Manzil Zaheer, Sashank J. Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Dinghuai Zhang, Tianyuan Zhang, Yiping Lu, Zhanxing Zhu, and Bin Dong. You only propagate once: Accelerating adversarial training via maximal principle. In *Advances in Neural Information Processing Systems 32*, pp. 227–238, 2019a.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 7472–7482, 2019b.

Jingfeng Zhang, Xilie Xu, Bo Han, Gang Niu, Lizhen Cui, Masashi Sugiyama, and Mohan Kankanhalli. Attacks which do not kill training make adversarial learning stronger. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 11278–11287, 2020.

Jingfeng Zhang, Jianing Zhu, Gang Niu, Bo Han, Masashi Sugiyama, and Mohan S. Kankanhalli. Geometry-aware instance-reweighted adversarial training. In *Proceedings of the 9th International Conference on Learning Represantations*, 2021.

Mingrui Zhang, Zebang Shen, Aryan Mokhtari, Hamed Hassani, and Amin Karbasi. One Sample Stochastic Frank-Wolfe. In *International Conference on Artificial Intelligence and Statistics*, 2019c.

Dongruo Zhou, Jinghui Chen, Yuan Cao, Yiqi Tang, Ziyan Yang, and Quanquan Gu. On the convergence of adaptive gradient methods for nonconvex optimization. *12th Annual Workshop on Optimization for Machine Learning*, 2020.

Fangyu Zou, Li Shen, Zequn Jie, Weizhong Zhang, and Wei Liu. A sufficient condition for convergences of adam and rmsprop. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11119–11127, 2019.

A Proposition and Lemmas

Proposition A.1. *For all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ and all $\alpha \in \mathbb{R}$, the following holds:*

$$\|\alpha \boldsymbol{x} + (1-\alpha)\boldsymbol{y}\|^2 = \alpha\|\boldsymbol{x}\|^2 + (1-\alpha)\|\boldsymbol{y}\|^2 - \alpha(1-\alpha)\|\boldsymbol{x}-\boldsymbol{y}\|^2.$$

Proof. Since $2\langle \boldsymbol{x}, \boldsymbol{y}\rangle = \|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2 - \|\boldsymbol{x}-\boldsymbol{y}\|^2$ holds, for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ and all $\alpha \in \mathbb{R}$,

$$\begin{aligned}
\|\alpha \boldsymbol{x} + (1-\alpha)\boldsymbol{y}\|^2 &= \alpha\|\boldsymbol{x}\|^2 + 2\alpha(1-\alpha)\langle \boldsymbol{x}, \boldsymbol{y}\rangle + (1-\alpha)^2\|\boldsymbol{y}\|^2 \\
&= \alpha\|\boldsymbol{x}\|^2 + \alpha(1-\alpha)(\|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2 - \|\boldsymbol{x}-\boldsymbol{y}\|^2) + (1-\alpha)^2\|\boldsymbol{y}\|^2 \\
&= \alpha\|\boldsymbol{x}\|^2 + (1-\alpha)\|\boldsymbol{y}\|^2 - \alpha(1-\alpha)\|\boldsymbol{x}-\boldsymbol{y}\|^2.
\end{aligned}$$

This completes the proof. □

Lemma A.1. *Suppose that (A1)-(A4) hold and consider Algorithm 3. Then, the following holds:*

$$\gamma_t \mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] \leq \mathbb{E}\left[f(\boldsymbol{\theta}_t)\right] - \mathbb{E}\left[f(\boldsymbol{\theta}_{t+1})\right] + \frac{D\sigma\gamma_t}{\sqrt{b_t}} + \frac{LD^2\gamma_t^2}{2}.$$

Proof. Let $t \in \mathbb{N}$. (A2) and the definition of $\boldsymbol{\theta}_{t+1}$ guarantee that

$$\begin{aligned}
f(\boldsymbol{\theta}_{t+1}) &\leq f(\boldsymbol{\theta}_t) + \langle \nabla f(\boldsymbol{\theta}_t), \boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t\rangle + \frac{L}{2}\|\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t\|^2 \\
&= f(\boldsymbol{\theta}_t) + \langle \nabla f(\boldsymbol{\theta}_t), (\boldsymbol{\theta}_t + \gamma_t(\boldsymbol{v}_t - \boldsymbol{\theta}_t)) - \boldsymbol{\theta}_t\rangle + \frac{L}{2}\|\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t\|^2 \\
&= f(\boldsymbol{\theta}_t) + \gamma_t\langle \nabla f(\boldsymbol{\theta}_t), \boldsymbol{v}_t - \boldsymbol{\theta}_t\rangle + \frac{L\gamma_t^2}{2}\|\boldsymbol{v}_t - \boldsymbol{\theta}_t\|^2 \\
&\leq f(\boldsymbol{\theta}_t) + \gamma_t\langle \nabla f(\boldsymbol{\theta}_t), \boldsymbol{v}_t - \boldsymbol{\theta}_t\rangle + \frac{LD^2\gamma_t^2}{2}.
\end{aligned}$$

Let $\hat{\boldsymbol{v}}_t := \operatorname*{argmax}_{\boldsymbol{v}\in\Omega}\langle \boldsymbol{v}, -\nabla f(\boldsymbol{\theta}_t)\rangle$ for all $t \in \mathbb{N}$. From the definition of $\boldsymbol{v}_t$, $\forall \boldsymbol{v} \in \Omega : \langle \nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t), \boldsymbol{v}_t\rangle \leq \langle \nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t), \boldsymbol{v}\rangle$. Then,

$$\begin{aligned}
f(\boldsymbol{\theta}_{t+1}) &\leq f(\boldsymbol{\theta}_t) + \gamma_t\langle \boldsymbol{m}_t, \boldsymbol{v}_t - \boldsymbol{\theta}_t\rangle + \gamma_t\langle \nabla f(\boldsymbol{\theta}_t) - \boldsymbol{m}_t, \boldsymbol{v}_t - \boldsymbol{\theta}_t\rangle + \frac{LD^2\gamma_t^2}{2} \\
&\leq f(\boldsymbol{\theta}_t) + \gamma_t\langle \boldsymbol{m}_t, \hat{\boldsymbol{v}}_t - \boldsymbol{\theta}_t\rangle + \gamma_t\langle \nabla f(\boldsymbol{\theta}_t) - \boldsymbol{m}_t, \boldsymbol{v}_t - \boldsymbol{\theta}_t\rangle + \frac{LD^2\gamma_t^2}{2} \\
&= f(\boldsymbol{\theta}_t) + \gamma_t\langle \nabla f(\boldsymbol{\theta}_t), \hat{\boldsymbol{v}}_t - \boldsymbol{\theta}_t\rangle + \gamma_t\langle \boldsymbol{m}_t - \nabla f(\boldsymbol{\theta}_t), \hat{\boldsymbol{v}}_t - \boldsymbol{\theta}_t\rangle + \gamma_t\langle \nabla f(\boldsymbol{\theta}_t) - \boldsymbol{m}_t, \boldsymbol{v}_t - \boldsymbol{\theta}_t\rangle + \frac{LD^2\gamma_t^2}{2} \\
&= f(\boldsymbol{\theta}_t) + \gamma_t\langle \nabla f(\boldsymbol{\theta}_t), \hat{\boldsymbol{v}}_t - \boldsymbol{\theta}_t\rangle + \gamma_t\langle \boldsymbol{m}_t - \nabla f(\boldsymbol{\theta}_t), \hat{\boldsymbol{v}}_t - \boldsymbol{v}_t\rangle + \frac{LD^2\gamma_t^2}{2} \\
&= f(\boldsymbol{\theta}_t) - \gamma_t\mathcal{G}(\boldsymbol{\theta}_t) + \gamma_t\langle \boldsymbol{m}_t - \nabla f(\boldsymbol{\theta}_t), \hat{\boldsymbol{v}}_t - \boldsymbol{v}_t\rangle + \frac{LD^2\gamma_t^2}{2} \\
&\leq f(\boldsymbol{\theta}_t) - \gamma_t\mathcal{G}(\boldsymbol{\theta}_t) + \gamma_t\|\boldsymbol{m}_t - \nabla f(\boldsymbol{\theta}_t)\|D + \frac{LD^2\gamma_t^2}{2}.
\end{aligned}$$

The last inequality follows from the Cauchy-Schwarz inequality and (A1). Taking the expectation with respect to ξ_t on both sides and using Lemma A.3, we have

$$\begin{aligned}
\mathbb{E}\left[f(\boldsymbol{\theta}_{t+1})\right] &\leq \mathbb{E}\left[f(\boldsymbol{\theta}_t)\right] - \gamma_t\mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] + D\gamma_t\mathbb{E}\left[\|\boldsymbol{m}_t - \nabla f(\boldsymbol{\theta}_t)\|\right] + \frac{LD^2\gamma_t^2}{2} \qquad (3) \\
&\leq \mathbb{E}\left[f(\boldsymbol{\theta}_t)\right] - \gamma_t\mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] + D\gamma_t\sqrt{\frac{\sigma^2}{b_t}} + \frac{LD^2\gamma_t^2}{2}.
\end{aligned}$$

This completes the proof. □

Lemma A.2. *Algorithm 3 has the property that, under (A3)(ii) and (A4), for all $t \in \mathbb{N}$,*

$$\mathbb{E}\left[\|\nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t) - \nabla f(\boldsymbol{\theta}_t)\|^2\right] \leq \frac{\sigma^2}{b_t}.$$

Proof. (A3) and the definition of $\nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t)$ guarantee that

$$\begin{aligned}
\mathbb{E}\left[\|\nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t) - \nabla f(\boldsymbol{\theta}_t)\|^2\right] &= \mathbb{E}\left[\left\|\frac{1}{b_t}\sum_{i=1}^{b_t} \mathsf{G}_{\xi_{t,i}}(\boldsymbol{\theta}_t) - \nabla f(\boldsymbol{\theta}_t)\right\|^2\right] \\
&= \mathbb{E}\left[\left\|\frac{1}{b_t}\sum_{i=1}^{b_t} \mathsf{G}_{\xi_{t,i}}(\boldsymbol{\theta}_t) - \frac{1}{b_t}\sum_{i=1}^{b_t} \nabla f(\boldsymbol{\theta}_t)\right\|^2\right] \\
&= \mathbb{E}\left[\left\|\frac{1}{b_t}\sum_{i=1}^{b_t} (\mathsf{G}_{\xi_{t,i}}(\boldsymbol{\theta}_t) - \nabla f(\boldsymbol{\theta}_t))\right\|^2\right] \\
&= \frac{1}{b_t^2}\mathbb{E}\left[\left\|\sum_{i=1}^{b_t} (\mathsf{G}_{\xi_{t,i}}(\boldsymbol{\theta}_t) - \nabla f(\boldsymbol{\theta}_t))\right\|^2\right] \\
&= \frac{1}{b_t^2}\mathbb{E}\left[\sum_{i=1}^{b_t} \left\|\mathsf{G}_{\xi_{t,i}}(\boldsymbol{\theta}_t) - \nabla f(\boldsymbol{\theta}_t)\right\|^2\right] \\
&\leq \frac{\sigma^2}{b_t}.
\end{aligned}$$

This completes the proof. □

Lemma A.3. *Algorithm 3 has the property that, under (A3)(ii) and (A4), for all $t \in \mathbb{N}$,*

$$\mathbb{E}\left[\|\boldsymbol{m}_t - \nabla f(\boldsymbol{\theta}_t)\|\right] \leq \sqrt{\frac{\sigma^2}{b_t}}.$$

Proof. The definition of $\boldsymbol{m}_t$ implies that

$$\begin{aligned}
\|\boldsymbol{m}_t - \nabla f(\boldsymbol{\theta}_t)\|^2 &= \|\beta \boldsymbol{m}_{t-1} + (1-\beta)\nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t) - \nabla f(\boldsymbol{\theta}_t)\|^2 \\
&= \|\beta(\boldsymbol{m}_{t-1} - \nabla f(\boldsymbol{\theta}_t)) + (1-\beta)(\nabla f_{\mathcal{S}_t} - \nabla f(\boldsymbol{\theta}_t))\|^2 \\
&= \beta^2\|\boldsymbol{m}_{t-1} - \nabla f(\boldsymbol{\theta}_t)\|^2 + (1-\beta)^2\|\nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t) - \nabla f(\boldsymbol{\theta}_t)\|^2 \\
&\quad + 2\beta(1-\beta)\langle \nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t) - \nabla f(\boldsymbol{\theta}_t), \boldsymbol{m}_{t-1} - \nabla f(\boldsymbol{\theta}_t)\rangle.
\end{aligned}$$

Therefore, from Assumption (A3)(i) and $\beta < 1$, we obtain

$$\begin{aligned}
\mathbb{E}\left[\|\boldsymbol{m}_t - \nabla f(\boldsymbol{\theta}_t)\|^2\right] &= (1-\beta)^2\mathbb{E}\left[\|\nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t) - \nabla f(\boldsymbol{\theta}_t)\|^2\right] + \beta^2\mathbb{E}\left[\|\boldsymbol{m}_{t-1} - \nabla f(\boldsymbol{\theta}_t)\|^2\right] && (4) \\
&< (1-\beta)^2\mathbb{E}\left[\|\nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t) - \nabla f(\boldsymbol{\theta}_t)\|^2\right] + \mathbb{E}\left[\|\boldsymbol{m}_{t-1} - \nabla f(\boldsymbol{\theta}_t)\|^2\right]. && (5)
\end{aligned}$$

On the other hand, Proposition A.1 guarantees that

$$\begin{aligned}
\mathbb{E}\left[\|\boldsymbol{m}_t - \nabla f(\boldsymbol{\theta}_t)\|^2\right] &= (1-\beta)\mathbb{E}\left[\|\nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t) - \nabla f(\boldsymbol{\theta}_t)\|^2\right] + \beta\mathbb{E}\left[\|\boldsymbol{m}_{t-1} - \nabla f(\boldsymbol{\theta}_t)\|^2\right] && (6) \\
&\quad - \beta(1-\beta)\mathbb{E}\left[\|\boldsymbol{m}_{t-1} - \nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t)\|^2\right]. && (7)
\end{aligned}$$

From (4) and (7), we have

$$\begin{aligned}
\mathbb{E}\left[\|\boldsymbol{m}_{t-1} - \nabla f(\boldsymbol{\theta}_t)\|^2\right] &= \mathbb{E}\left[\|\boldsymbol{m}_{t-1} - \nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t)\|^2\right] - \mathbb{E}\left[\|\nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t) - \nabla f(\boldsymbol{\theta}_t)\|^2\right] && (8) \\
&\leq \mathbb{E}\left[\|\boldsymbol{m}_{t-1} - \nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t)\|^2\right]. && (9)
\end{aligned}$$

Therefore, from (5) and (8), we obtain

$$\mathbb{E}\left[\|\boldsymbol{m}_t - \nabla f(\boldsymbol{\theta}_t)\|^2\right] \leq \beta(-2+\beta)\mathbb{E}\left[\|\nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t) - \nabla f(\boldsymbol{\theta}_t)\|^2\right] + \mathbb{E}\left[\|\boldsymbol{m}_{t-1} - \nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t)\|^2\right]. \tag{10}$$

Now, let us show that, for all $t \in \mathbb{N}$,

$$\mathbb{E}\left[\|\boldsymbol{m}_{t-1} - \nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t)\|^2\right] \leq \beta(2-\beta)\mathbb{E}\left[\|\nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t) - \nabla f(\boldsymbol{\theta}_t)\|^2\right]. \tag{11}$$

If (11) does not hold, there exists $t_0 \in \mathbb{N}$ such that

$$\mathbb{E}\left[\left\|\boldsymbol{m}_{t_0-1} - \nabla f_{\mathcal{S}_{t_0}}(\boldsymbol{\theta}_{t_0})\right\|^2\right] > \beta(2-\beta)\mathbb{E}\left[\left\|\nabla f_{\mathcal{S}_{t_0}}(\boldsymbol{\theta}_{t_0}) - \nabla f(\boldsymbol{\theta}_{t_0})\right\|^2\right],$$

which implies

$$\mathbb{E}\left[\left\|\nabla f_{\mathcal{S}_{t_0}}(\boldsymbol{\theta}_{t_0}) - \nabla f(\boldsymbol{\theta}_{t_0})\right\|^2\right] < \frac{1}{\beta(2-\beta)}\mathbb{E}\left[\left\|\boldsymbol{m}_{t_0-1} - \nabla f_{\mathcal{S}_{t_0}}(\boldsymbol{\theta}_{t_0})\right\|^2\right]. \tag{12}$$

Hence, from (10) and (12),

$$\begin{aligned}\mathbb{E}\left[\|\boldsymbol{m}_{t_0} - \nabla f(\boldsymbol{\theta}_{t_0})\|^2\right] &< \beta(-2+\beta)\left\{\frac{1}{\beta(2-\beta)}\mathbb{E}\left[\left\|\boldsymbol{m}_{t_0-1} - \nabla f_{\mathcal{S}_{t_0}}(\boldsymbol{\theta}_{t_0})\right\|^2\right]\right\} \\ &\quad + \mathbb{E}\left[\left\|\boldsymbol{m}_{t_0-1} - \nabla f_{\mathcal{S}_{t_0}}(\boldsymbol{\theta}_{t_0})\right\|^2\right] \\ &= 0.\end{aligned}$$

Since $\mathbb{E}\left[\|\boldsymbol{m}_{t_0} - \nabla f(\boldsymbol{\theta}_{t_0})\|^2\right] \geq 0$, there is a contradiction. Therefore, (11) holds for all $t \in \mathbb{N}$. Thus, Lemmas A.2, (4), (9), and (11) ensure that

$$\begin{aligned}\mathbb{E}\left[\|\boldsymbol{m}_t - \nabla f(\boldsymbol{\theta}_t)\|^2\right] &\leq (1-\beta)^2\mathbb{E}\left[\|\nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t) - \nabla f(\boldsymbol{\theta}_t)\|^2\right] + \beta^3(2-\beta)\mathbb{E}\left[\|\nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t) - \nabla f(\boldsymbol{\theta}_t)\|^2\right] \\ &= \left\{(1-\beta)^2 + \beta^3(2-\beta)\right\}\mathbb{E}\left[\|\nabla f_{\mathcal{S}_t}(\boldsymbol{\theta}_t) - \nabla f(\boldsymbol{\theta}_t)\|^2\right] \\ &\leq \frac{\sigma^2}{b_t}.\end{aligned}$$

This completes the proof. □

Lemma A.4. *Let the batch size increase as* $b_t := (\underbrace{b, b, \cdots, b}_{E}, \underbrace{\lambda b, \lambda b, \cdots, \lambda b}_{E}, \cdots, \underbrace{\lambda^{Q-1}b, \lambda^{Q-1}b, \cdots, \lambda^{Q-1}b}_{E})$. *Then, for all* $t \in \mathbb{N}$,

$$\sum_{t=0}^{T-1} \frac{1}{\sqrt{b_t}} \leq \frac{E\sqrt{\lambda}}{\sqrt{b}(\sqrt{\lambda}-1)},$$

where $\lambda > 1$ *and* $QE = T$.

Proof. From the definition of b_t,

$$\begin{aligned}\sum_{t=0}^{T-1} \frac{1}{\sqrt{b_t}} &= \frac{E}{\sqrt{b}} + \frac{E}{\sqrt{\lambda b}} + \cdots + \frac{E}{\sqrt{\lambda^{Q-1}b}} \\ &= \frac{E}{\sqrt{b}}\left(1 + \frac{1}{\sqrt{\lambda}} + \cdots + \frac{1}{\sqrt{\lambda^{Q-1}}}\right) \\ &\leq \frac{E}{\sqrt{b}} \cdot \frac{\sqrt{\lambda}}{\sqrt{\lambda}-1}.\end{aligned}$$

This completes the proof. □

B Proofs of Theorems 3.1 and 3.2

The following is a convergence analysis of Algorithm 3 using a constant learning rate.

Proof. From Lemma A.1, we have

$$\gamma_t \mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] \leq \mathbb{E}\left[f(\boldsymbol{\theta}_t)\right] - \mathbb{E}\left[f(\boldsymbol{\theta}_{t+1})\right] + \frac{D\sigma\gamma_t}{\sqrt{b_t}} + \frac{LD^2\gamma_t^2}{2}.$$

Summing over t, we get

$$\begin{aligned}\sum_{t=0}^{T-1} \gamma_t \mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] &\leq \mathbb{E}\left[f(\boldsymbol{\theta}_0)\right] - \mathbb{E}\left[f(\boldsymbol{\theta}_T)\right] + D\sigma \sum_{t=0}^{T-1} \frac{\gamma_t}{\sqrt{b_t}} + \frac{LD^2}{2} \sum_{t=0}^{T-1} \gamma_t^2 \\ &\leq \mathbb{E}\left[f(\boldsymbol{\theta}_0)\right] - \mathbb{E}\left[f(\boldsymbol{\theta}^\star)\right] + D\sigma \sum_{t=0}^{T-1} \frac{\gamma_t}{\sqrt{b_t}} + \frac{LD^2}{2} \sum_{t=0}^{T-1} \gamma_t^2.\end{aligned}$$

Assumption (C1) guarantees that

$$\gamma \sum_{t=0}^{T-1} \mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] \leq \mathbb{E}\left[f(\boldsymbol{\theta}_0)\right] - \mathbb{E}\left[f(\boldsymbol{\theta}^\star)\right] + D\sigma\gamma \sum_{t=0}^{T-1} \frac{1}{\sqrt{b_t}} + \frac{LD^2}{2} T\gamma^2.$$

Hence,

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] \leq \frac{\mathbb{E}\left[f(\boldsymbol{\theta}_0)\right] - \mathbb{E}\left[f(\boldsymbol{\theta}^\star)\right]}{\gamma T} + \frac{D\sigma}{T} \sum_{t=0}^{T-1} \frac{1}{\sqrt{b_t}} + \frac{LD^2\gamma}{2}.$$

Then, for a constant batch size $b_t := b$, we have

$$\begin{aligned}\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] &\leq \frac{\mathbb{E}\left[f(\boldsymbol{\theta}_0)\right] - \mathbb{E}\left[f(\boldsymbol{\theta}^\star)\right]}{\gamma T} + \frac{D\sigma}{\sqrt{b}} + \frac{LD^2\gamma}{2} \\ &= \mathcal{O}\left(\frac{1}{T} + \frac{1}{\sqrt{b}} + \gamma\right)\end{aligned}$$

This completes the proof for Theorem 3.1.
Next, for an increasing batch size $b_t := (\underbrace{b, b, \cdots, b}_{E}, \underbrace{\lambda b, \lambda b, \cdots, \lambda b}_{E}, \cdots, \underbrace{\lambda^{Q-1} b, \lambda^{Q-1} b, \cdots, \lambda^{Q-1} b}_{E})$, from Lemma A.4, we have

$$\begin{aligned}\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] &\leq \frac{\mathbb{E}\left[f(\boldsymbol{\theta}_0)\right] - \mathbb{E}\left[f(\boldsymbol{\theta}^\star)\right]}{\gamma T} + \frac{D\sigma}{T} \frac{E\sqrt{\lambda}}{\sqrt{b}(\sqrt{\lambda} - 1)} + \frac{LD^2\gamma}{2} \\ &= \mathcal{O}\left(\frac{1}{T} + \gamma\right).\end{aligned}$$

This completes the proof for Theorem 3.2. □

C Proof of Theorem 3.3

Proof. From Lemma A.1 and $b_t := b$, we have

$$\gamma_t \mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] \leq \mathbb{E}\left[f(\boldsymbol{\theta}_t)\right] - \mathbb{E}\left[f(\boldsymbol{\theta}_{t+1})\right] + \frac{D\sigma\gamma_t}{\sqrt{b}} + \frac{LD^2\gamma_t^2}{2}.$$

Summing over t, we get

$$\frac{\sum_{t=0}^{T-1}\gamma_t\mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right]}{\sum_{t=0}^{T-1}\gamma_t} \leq \frac{\mathbb{E}\left[f(\boldsymbol{\theta}_0)\right]-\mathbb{E}\left[f(\boldsymbol{\theta}^\star)\right]}{\sum_{t=0}^{T-1}\gamma_t} + \frac{D\sigma}{\sqrt{b}} + \frac{LD^2}{2}\frac{\sum_{t=0}^{T-1}\gamma_t^2}{\sum_{t=0}^{T-1}\gamma_t}.$$

If we use $\gamma_t = \frac{1}{(t+1)}$ such that $\displaystyle\sum_{t=0}^{+\infty}\gamma_t = +\infty, \sum_{t=0}^{+\infty}\gamma_t^2 < +\infty$, then it is guaranteed that $\sum_{t=0}^{+\infty}\gamma_t^2/\sum_{t=0}^{+\infty}\gamma_t = 0$. Hence,

$$\frac{\sum_{t=0}^{+\infty}\gamma_t\mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right]}{\sum_{t=0}^{+\infty}\gamma_t} \leq \frac{D\sigma}{\sqrt{b}}.$$

This completes the proof. □

D Proof of Theorem 3.4

Proof. From Lemma A.1 and $b_t := b$, we have

$$\mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] \leq \frac{1}{\gamma_t}\left(\mathbb{E}\left[f(\boldsymbol{\theta}_t)\right]-\mathbb{E}\left[f(\boldsymbol{\theta}_{t+1})\right]\right) + \frac{D\sigma}{\sqrt{b}} + \frac{LD^2\gamma_t}{2}.$$

Summing over t, we get

$$\sum_{t=0}^{T-1}\mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] \leq \underbrace{\sum_{t=0}^{T-1}\frac{1}{\gamma_t}\left(\mathbb{E}\left[f(\boldsymbol{\theta}_t)\right]-\mathbb{E}\left[f(\boldsymbol{\theta}_{t+1})\right]\right)}_{\Gamma_T} + \frac{D\sigma}{\sqrt{b}}T + \frac{LD^2}{2}\sum_{t=0}^{T-1}\gamma_t.$$

From (A4), there exists a real number $\bar{f}$ such that $\forall\boldsymbol{\theta}\in\Omega \Rightarrow f(\boldsymbol{\theta}) \leq \bar{f}$. Accordingly, we have

$$\begin{aligned}
\Gamma_T &= \frac{\mathbb{E}\left[f(\boldsymbol{\theta}_0)\right]}{\gamma_0} + \sum_{t=0}^{T-1}\left(\frac{\mathbb{E}\left[f(\boldsymbol{\theta}_t)\right]}{\gamma_t} - \frac{\mathbb{E}\left[f(\boldsymbol{\theta}_t)\right]}{\gamma_{t-1}}\right) - \frac{\mathbb{E}\left[f(\boldsymbol{\theta}_T)\right]}{\gamma_{T-1}} \\
&= \frac{\mathbb{E}\left[f(\boldsymbol{\theta}_0)\right]}{\gamma_0} + \sum_{t=0}^{T-1}\left(\frac{1}{\gamma_t} - \frac{1}{\gamma_{t-1}}\right)\mathbb{E}\left[f(\boldsymbol{\theta}_t)\right] - \frac{\mathbb{E}\left[f(\boldsymbol{\theta}_T)\right]}{\gamma_{T-1}} \\
&\leq \frac{\bar{f}}{\gamma_0} + \bar{f}\sum_{t=0}^{T-1}\left(\frac{1}{\gamma_t} - \frac{1}{\gamma_{t-1}}\right) - \frac{f(\boldsymbol{\theta}^\star)}{\gamma_{T-1}} \\
&\leq \frac{\bar{f}}{\gamma_0} + \bar{f}\left(\frac{1}{\gamma_{T-1}} - \frac{1}{\gamma_0}\right) - \frac{f(\boldsymbol{\theta}^\star)}{\gamma_{T-1}} \\
&= \frac{1}{\gamma_{T-1}}\left(\bar{f} - f(\boldsymbol{\theta}^\star)\right) \\
&\leq \frac{1}{\gamma_{T-1}}\left(\bar{f} + |f(\boldsymbol{\theta}^\star)|\right) \\
&\leq \frac{2}{\gamma_{T-1}}\max\{\bar{f}, |f(\boldsymbol{\theta}^\star)|\}
\end{aligned} \tag{13}$$

The first inequality follows from $\mathbb{E}\left[f(\boldsymbol{\theta}_T)\right] \geq f(\boldsymbol{\theta}^\star)$ and $\frac{1}{\gamma_t} - \frac{1}{\gamma_{t-1}} \geq 0$ since γ_t is monotone decreasing. Hence,

$$\begin{aligned}
\sum_{t=0}^{T-1}\mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] &\leq \frac{2}{\gamma_{T-1}}\max\{\bar{f}, |f(\boldsymbol{\theta}^\star)|\} + \frac{D\sigma}{\sqrt{b}}T + \frac{LD^2}{2}\sum_{t=0}^{T-1}\gamma_t, \\
\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] &\leq \frac{2}{T\gamma_{T-1}}\max\{\bar{f}, |f(\boldsymbol{\theta}^\star)|\} + \frac{D\sigma}{\sqrt{b}} + \frac{LD^2}{2T}\sum_{t=0}^{T-1}\gamma_t.
\end{aligned}$$

(i) If we use $\gamma_t = \frac{1}{(t+1)^a} (a \in [\frac{1}{2}, 1))$, then

$$\gamma_{T-1} = \frac{1}{T^a} \text{ and } \frac{1}{T}\sum_{t=0}^{T-1} \gamma_t \leq 1 + \int_0^{T-1} \frac{dt}{(t+1)^a} \leq \frac{1}{1-a} \cdot \frac{1}{T^a}.$$

Therefore,

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] &\leq \frac{2\max\{\bar{f}, |f(\boldsymbol{\theta}^\star)|\}}{T^{1-a}} + \frac{D\sigma}{\sqrt{b}} + \frac{LD^2}{2(1-a)T^a} \\ &= \mathcal{O}\left(\frac{1}{T^{\min\{1-a,a\}}} + \frac{\sigma}{\sqrt{b}}\right).\end{aligned}$$

(ii) If we use $\gamma_t = (\underbrace{\gamma, \gamma, \cdots, \gamma}_{K}, \underbrace{\eta\gamma, \eta\gamma, \cdots, \eta\gamma}_{K}, \cdots, \underbrace{\eta^{P-1}\gamma, \eta^{P-1}\gamma, \cdots, \eta^{P-1}\gamma}_{K})$, then there exist real numbers $\underline{\gamma}$ and γ_∞ such that $\gamma_{T-1} = \eta^{P-1}\gamma \geq \underline{\gamma}$ and $\sum_{t=0}^{T-1} \gamma_t = \gamma_\infty < +\infty$. Hence we have that

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] &\leq \frac{2\max\{\bar{f}, |f(\boldsymbol{\theta}^\star)|\}}{T\underline{\gamma}} + \frac{D\sigma}{\sqrt{b}} + \frac{\gamma_\infty LD^2}{2T} \\ &= \mathcal{O}\left(\frac{1}{T} + \frac{\sigma}{\sqrt{b}}\right).\end{aligned}$$

This completes the proof. □

E Proof of Theorem 3.5

Proof. From Lemma A.1, we have

$$\mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] \leq \frac{1}{\gamma_t}\left(\mathbb{E}\left[f(\boldsymbol{\theta}_t)\right] - \mathbb{E}\left[f(\boldsymbol{\theta}_{t+1})\right]\right) + \frac{D\sigma}{\sqrt{b_t}} + \frac{LD^2\gamma_t}{2}.$$

Summing over t, we get

$$\sum_{t=0}^{T-1} \mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] \leq \underbrace{\sum_{t=0}^{T-1} \frac{1}{\gamma_t}\left(\mathbb{E}\left[f(\boldsymbol{\theta}_t)\right] - \mathbb{E}\left[f(\boldsymbol{\theta}_{t+1})\right]\right)}_{\Gamma_T} + D\sigma \sum_{t=0}^{T-1} \frac{1}{\sqrt{b_t}} + \frac{LD^2}{2}\sum_{t=0}^{T-1} \gamma_t.$$

From Equation (13) and Lemma A.4, we have

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] \leq \frac{2}{T\gamma_{T-1}} \max\{\bar{f}, |f(\boldsymbol{\theta}^\star)|\} + \frac{D\sigma}{T}\frac{E\sqrt{\lambda}}{\sqrt{b}(\sqrt{\lambda}-1)} + \frac{LD^2}{2T}\sum_{t=0}^{T-1} \gamma_t.$$

(i) If we use $\gamma_t = \frac{1}{(t+1)^a} (a \in [\frac{1}{2}, 1))$, then

$$\gamma_{T-1} = \frac{1}{T^a} \text{ and } \frac{1}{T}\sum_{t=0}^{T-1} \gamma_t \leq 1 + \int_0^{T-1} \frac{dt}{(t+1)^a} \leq \frac{1}{1-a} \cdot \frac{1}{T^a}.$$

Therefore,

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] &\leq \frac{2\max\{\bar{f}, |f(\boldsymbol{\theta}^\star)|\}}{T^{1-a}} + \frac{D\sigma}{T}\frac{E\sqrt{\lambda}}{\sqrt{b}(\sqrt{\lambda}-1)} + \frac{LD^2}{2(1-a)T^a} \\ &= \mathcal{O}\left(\frac{1}{T^{\min\{1-a,a\}}}\right).\end{aligned}$$

(ii) If we use $\gamma_t = (\underbrace{\gamma, \gamma, \cdots, \gamma}_{K}, \underbrace{\eta\gamma, \eta\gamma, \cdots, \eta\gamma}_{K}, \cdots, \underbrace{\eta^{P-1}\gamma, \eta^{P-1}\gamma, \cdots, \eta^{P-1}\gamma}_{K})$, then there exist real numbers $\underline{\gamma}$ and γ_∞ such that $\gamma_{T-1} = \eta^{P-1}\gamma \geq \underline{\gamma}$ and $\sum_{t=0}^{T-1} \gamma_t = \gamma_\infty < +\infty$. Hence we have that

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left[\mathcal{G}(\boldsymbol{\theta}_t)\right] \leq \frac{2\max\{\bar{f}, |f(\boldsymbol{\theta}^\star)|\}}{T\underline{\gamma}} + \frac{D\sigma}{T}\frac{E\sqrt{\lambda}}{\sqrt{b}(\sqrt{\lambda}-1)} + \frac{\gamma_\infty L D^2}{2T}$$
$$= \mathcal{O}\left(\frac{1}{T}\right).$$

This completes the proof. □

F Full Experimental Results

The code used is available at our GitHub repository (`https://anonymous.4open.science/r/sfw25`). The experimental environment consisted of NVIDIA DGX A100×8GPU and Dual AMD Rome7742 2.25-GHz, 128 Cores×2CPU. The software environment was Python 3.8.2, Pytorch 1.6.0, and CUDA 11.6. All experiments were performed using a single GPU.

F.1 Supplemental Results for Section 3.3

Figure 3 plots the test accuracy and loss function values for a batch size of 2^5 and a constant learning rate $\gamma_t = \gamma_0 = 0.1$. Figure 4 similarly plots the test accuracy and loss function values for a batch size of 2^5 and a decaying learning rate that halves every 40 epochs; i.e., this setting is based on Theorems 3.4(ii). Both figures show that SFW achieves higher test accuracy and lower loss function values than those of SFWM, in contrast to Figures 1 and 2.

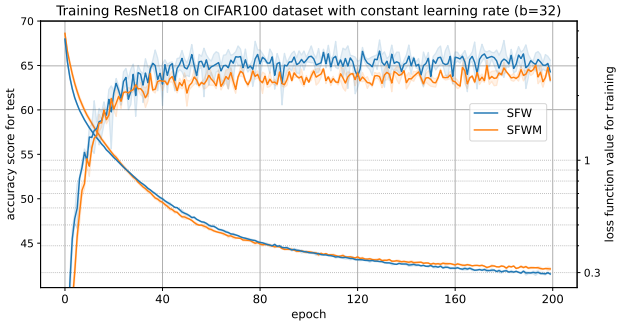

Figure 3: Accuracy score for the testing and loss function value for training versus the number of epochs in training ResNet18 on the CIFAR100 dataset with the L_2 constraint and a constant learning rate. The solid line represents the mean value, and the shaded area represents the maximum and minimum over three runs.

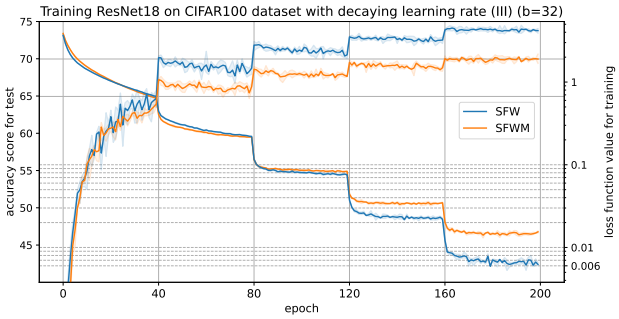

Figure 4: Accuracy score for the testing and loss function value for training versus the number of epochs in training ResNet18 on the CIFAR100 dataset with the L_2 constraint and a decaying learning rate. The solid line represents the mean value, and the shaded area represents the maximum and minimum over three runs. The learning rate was decreased every 40 epochs as $[0.1, 0.05, 0.025, 0.0125, 0.00625]$ and batch size was fixed at 2^5.

We also performed similar experiments for decaying learning rates (I) $\gamma_t := \frac{1}{t+1}$ based on Theorems 3.3 and (II) $\gamma_t := \frac{1}{\sqrt{t+1}}$ based on Theorems 3.4(i). Theoretically, this is an excellent learning rate setting that can remove the extra term in the upper bound of the Frank Wolfe gap, but experimentally, it is found to be unusable because the learning rate becomes too small from the early stages of learning (see Figures 5 and 6).

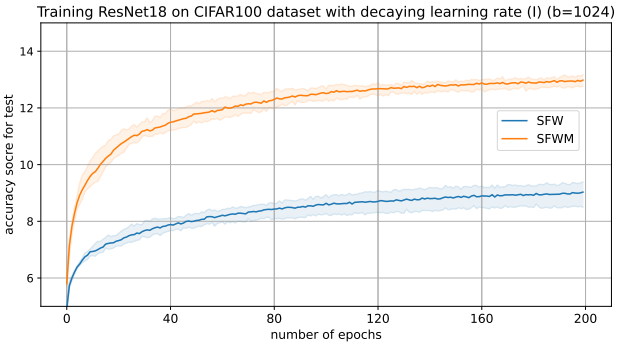

Figure 5: Accuracy score for the testing versus the number of epochs in training ResNet18 on the CIFAR100 dataset with the L_2 constraint and a decaying learning rate (I) $\gamma_t := \frac{1}{t+1}$. The batch size was fixed at 2^{10}. The solid line represents the mean value, and the shaded area represents the maximum and minimum over three runs.

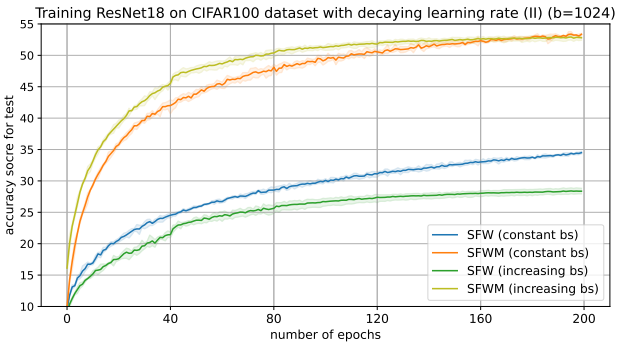

Figure 6: Accuracy score for the testing versus the number of epochs in training ResNet18 on the CIFAR100 dataset with the L_2 constraint and a decaying learning rate (II) $\gamma_t := \frac{1}{\sqrt{t+1}}$. The batch size was fixed at 2^{10}.The solid line represents the mean value, and the shaded area represents the maximum and minimum over three runs.

F.2 Supplemental Results for Section 4.5

We attacked the robust models listed in RobustBench (Croce et al., 2021) with PGD, SFW, APGD, and ASFW to verify their performance.

Table 4: Adversarial accuracy achieved by PGD and SFW. CIFAR100 dataset/L_∞ with $\epsilon = 8/255$

paper	Architecture	clean accuracy	PGD	SFW
(Wang et al., 2023)	WideResNet-70-16	75.22	48.41	48.44
(Wang et al., 2023)	WideResNet-28-10	72.58	44.21	44.26
(Debenedetti et al., 2023)	XCiT-L12	70.76	39.28	39.39
(Rebuffi et al., 2021)	WideResNet-70-16	63.56	38.58	38.69
(Debenedetti et al., 2023)	XCiT-M12	69.21	39.22	39.25
(Pang et al., 2022)	WideResNet-70-16	65.56	36.87	36.94
(Debenedetti et al., 2023)	XCiT-S12	67.34	37.47	37.42
(Rebuffi et al., 2021)	WideResNet-28-16	62.41	36.09	36.13
(Jia et al., 2022)	WideResNet-34-20	67.31	37.20	37.66
(Addepalli et al., 2022a)	WideResNet-34-10	68.75	37.03	37.21
(Cui et al., 2021)	WideResNet-34-10	62.97	37.48	37.8
(Sehwag et al., 2022)	WideResNet-34-10	65.93	36.04	36.15
(Pang et al., 2022)	WideResNet-28-10	63.66	35.39	35.43
(Jia et al., 2022)	WideResNet-34-10	64.89	36.19	36.57
(Addepalli et al., 2022b)	WideResNet-34-10	65.73	36.90	36.64
(Cui et al., 2021)	WideResNet-34-20	62.55	34.62	34.63
(Cui et al., 2021)	WideResNet-34-10	60.64	34.55	34.71
(Rade & Moosavi-Dezfooli, 2021)	PreActResNet-18	61.50	32.69	32.75
(Wu et al., 2020)	WideResNet-34-10	60.38	33.65	33.73
(Rebuffi et al., 2021)	PreActResNet-18	56.87	31.95	32.0
(Hendrycks et al., 2019)	WideResNet-28-10	59.23	33.75	33.79
(Addepalli et al., 2022a)	ResNet-18	65.45	33.79	33.97
(Cui et al., 2021)	WideResNet-34-10	70.25	30.41	30.51
(Addepalli et al., 2022b)	PreActResNet-18	62.02	33.14	33.29
(Chen et al., 2022)	WideResNet-34-10	62.15	31.53	31.87
(Rice et al., 2020)	PreActResNet-18	53.83	20.95	21.01

Table 5: Adversarial accuracy achieved by PGD and SFW. ImageNet dataset/L_∞ with $\epsilon = 4/255$

paper	Architecture	clean accuracy	PGD	SFW
(Liu et al., 2023)	ConvNeXt-L	78.02	60.66	60.68
(Singh et al., 2023)	ConvNeXt-L + ConvStem	77.00	59.26	59.26
(Singh et al., 2023)	ConvNeXt-B + ConvStem	75.88	58.56	58.60
(Liu et al., 2023)	ConvNeXt-B	76.7	58.30	58.32
(Singh et al., 2023)	ViT-B + ConvStem	76.30	57.12	57.18
(Singh et al., 2023)	ConvNeXt-S + ConvStem	74.08	55.22	55.24
(Singh et al., 2023)	ConvNeXtt-T + ConvStem	72.70	53.32	53.36
(Singh et al., 2023)	ViT-S + ConvStem	72.58	51.34	51.36
(Debenedetti et al., 2023)	XCiT-L12	73.76	49.88	49.9
(Debenedetti et al., 2023)	XCiT-M12	74.04	48.14	48.1
(Debenedetti et al., 2023)	XCiT-S12	72.34	45.28	45.3
(Salman et al., 2020)	WideResNet-50-2	68.64	41.42	41.60
(Salman et al., 2020)	ResNet-50	64.06	39.18	39.28
(Ilyas et al., 2019)	ResNet-50	62.52	33.24	33.40
(Wong et al., 2020)	ResNet-50	55.64	30.50	30.12
(Salman et al., 2020)	ResNet-18	52.92	29.98	30.14

Table 6: Adversarial accuracy achieved by PGD and SFW. CIFAR10 dataset/L_∞ with $\epsilon = 8/255$

paper	Architecture	clean accuracy	PGD	SFW
(Wang et al., 2023)	WideResNet-70-16	93.25	73.62	73.62
(Wang et al., 2023)	WideResNet-28-10	92.44	70.39	70.38
(Rebuffi et al., 2021)	WideResNet-70-16	92.23	69.86	69.99
(Gowal et al., 2021)	WideResNet-70-16	88.74	68.97	69.05
(Rebuffi et al., 2021)	WideResNet-106-16	88.50	68.07	68.26
(Rebuffi et al., 2021)	WideResNet-70-16	88.54	67.65	67.84
(Kang et al., 2021)	WideResNet-70-16	93.73	90.86	90.72
(Xu et al., 2023)	WideResNet-28-10	93.69	67.21	67.13
(Gowal et al., 2021)	WideResNet-28-10	87.50	65.94	66.09
(Pang et al., 2022)	WideResNet-70-16	89.01	66.81	66.83
(Rade & Moosavi-Dezfooli, 2021)	WideResNet-34-10	91.47	65.89	65.89
(Sehwag et al., 2022)	ResNest152	87.30	65.25	65.28
(Huang et al., 2021a)	WideResNet-34-R	91.23	65.06	65.23
(Huang et al., 2021a)	WideResNet-34-R	90.56	64.12	64.42
(Dai et al., 2022)	WideResNet-28-10-PSSiLU	87.02	64.14	64.18
(Pang et al., 2022)	WideResNet-28-10	88.61	64.87	64.86
(Rade & Moosavi-Dezfooli, 2021)	WideResNet-28-10	88.16	64.87	63.89
(Rebuffi et al., 2021)	WideResNet-28-10	87.33	64.22	64.39
(Sridhar et al., 2022)	WideResNet-34-15	86.53	63.20	63.43
(Wu et al., 2020)	WideResNet-28-10	88.25	63.58	63.7
(Sridhar et al., 2022)	WideResNet-28-10	89.46	62.56	62.65
(Zhang et al., 2021)	WideResNet-28-10	89.36	67.64	67.81
(Carmon et al., 2019)	WideResNet-28-10	89.69	62.31	62.39
(Gowal et al., 2021)	PreActResNet-18	87.35	61.13	61.14
(Addepalli et al., 2022b)	WideResNet-34-10	85.32	64.86	65.06
(Addepalli et al., 2022a)	WideResNet-34-10	88.71	61.10	61.16
(Rade & Moosavi-Dezfooli, 2021)	PreActResNet-18	89.02	61.58	61.52
(Jia et al., 2022)	WideResNet-70-16	85.66	60.96	61.18
(Debenedetti et al., 2023)	XCiT-L12	91.73	59.18	59.31
(Debenedetti et al., 2023)	XCiT-M12	91.30	59.09	59.26
(Sehwag et al., 2020)	WideResNet-28-10	88.98	59.94	60.08
(Rebuffi et al., 2021)	PreActResNet-18	83.53	59.66	59.82
(Wang et al., 2020)	WideResNet-28-10	87.50	62.65	62.69
(Jia et al., 2022)	WideResNet-34-10	84.98	59.79	60.04
(Wu et al., 2020)	WideResNet-34-10	85.36	59.17	59.24
(Debenedetti et al., 2023)	XCiT-S12	90.06	58.98	58.99
(Sehwag et al., 2022)	ResNet-18	84.59	58.81	58.79
(Hendrycks et al., 2019)	WideResNet-28-10	87.11	57.58	57.68
(Pang et al., 2020)	WideResNet-34-20	85.14	62.19	62.37
(Cui et al., 2021)	WideResNet-34-20	88.70	55.44	55.53
(Zhang et al., 2020)	WideResNet-34-10	84.52	57.09	57.13
(Rice et al., 2020)	WideResNet-34-20	85.34	57.32	57.33
(Huang et al., 2020b)	WideResNet-34-10	83.48	56.15	56.20
(Zhang et al., 2019b)	WideResNet-34-10	84.92	55.09	55.13
(Cui et al., 2021)	WideResNet-34-10	88.22	54.24	54.35
(Addepalli et al., 2022a)	ResNet-18	85.71	56.56	56.58
(Chen et al., 2020b)	ResNet-50	86.04	54.34	54.37
(Chen et al., 2022)	WideResNet-34-10	85.32	54.60	55.12
(Addepalli et al., 2022b)	ResNet-18	80.24	56.19	56.29
(Zhang et al., 2019a)	WideResNet-34-10	87.20	46.37	46.38
(Andriushchenko & Flammarion, 2020)	PreActResNet-18	79.84	47.42	47.49
(Wong et al., 2020)	PreActResNet-18	83.34	46.55	46.74
(Ding et al., 2018)	WideResNet-28-4	84.36	51.18	51.43

Table 7: Adversarial accuracy achieved by APGD and ASFW. CIFAR10 dataset/L_∞ with $\epsilon = 8/255$

paper	Architecture	clean accuracy	APGD	ASFW
(Wang et al., 2023)	WideResNet-70-16	93.25	73.51	73.5
(Wang et al., 2023)	WideResNet-28-10	92.44	70.23	70.32
(Rebuffi et al., 2021)	WideResNet-70-16	92.23	69.4	69.54
(Gowal et al., 2021)	WideResNet-70-16	88.74	68.52	68.56
(Rebuffi et al., 2021)	WideResNet-106-16	88.50	67.64	67.82
(Rebuffi et al., 2021)	WideResNet-70-16	88.54	67.27	67.37
(Kang et al., 2021)	WideResNet-70-16	93.73	84.98	86.98
(Xu et al., 2023)	WideResNet-28-10	93.69	67.05	67.16
(Gowal et al., 2021)	WideResNet-28-10	87.50	65.56	65.77
(Pang et al., 2022)	WideResNet-70-16	89.01	66.69	66.74
(Rade & Moosavi-Dezfooli, 2021)	WideResNet-34-10	91.47	65.66	65.73
(Sehwag et al., 2022)	ResNest152	87.30	65.08	65.06
(Huang et al., 2021a)	WideResNet-34-R	91.23	64.52	64.77
(Huang et al., 2021a)	WideResNet-34-R	90.56	63.55	63.8
(Dai et al., 2022)	WideResNet-28-10-PSSiLU	87.02	63.99	64.02
(Pang et al., 2022)	WideResNet-28-10	88.61	64.73	64.73
(Rade & Moosavi-Dezfooli, 2021)	WideResNet-28-10	88.16	63.78	63.81
(Rebuffi et al., 2021)	WideResNet-28-10	87.33	63.99	64.08
(Sridhar et al., 2022)	WideResNet-34-15	86.53	62.96	63.03
(Wu et al., 2020)	WideResNet-28-10	88.25	63.35	63.49
(Sridhar et al., 2022)	WideResNet-28-10	89.46	62.06	62.27
(Zhang et al., 2021)	WideResNet-28-10	89.36	66.38	66.58
(Carmon et al., 2019)	WideResNet-28-10	89.69	61.71	62.06
(Gowal et al., 2021)	PreActResNet-18	87.35	60.84	60.94
(Addepalli et al., 2022b)	WideResNet-34-10	85.32	64.41	64.52
(Addepalli et al., 2022a)	WideResNet-34-10	88.71	60.79	60.96
(Rade & Moosavi-Dezfooli, 2021)	PreActResNet-18	89.02	61.39	61.46
(Jia et al., 2022)	WideResNet-70-16	85.66	60.63	60.61
(Debenedetti et al., 2023)	XCiT-L12	91.73	58.98	59.18
(Debenedetti et al., 2023)	XCiT-M12	91.30	59.01	59.03
(Sehwag et al., 2020)	WideResNet-28-10	88.98	59.62	59.79
(Rebuffi et al., 2021)	PreActResNet-18	83.53	59.58	59.64
(Wang et al., 2020)	WideResNet-28-10	87.50	61.72	61.95
(Jia et al., 2022)	WideResNet-34-10	84.98	59.51	59.58
(Wu et al., 2020)	WideResNet-34-10	85.36	58.79	58.91
(Debenedetti et al., 2023)	XCiT-S12	90.06	58.78	58.84
(Sehwag et al., 2022)	ResNet-18	84.59	58.4	58.54
(Hendrycks et al., 2019)	WideResNet-28-10	87.11	57.2	57.38
(Pang et al., 2020)	WideResNet-34-20	85.14	61.59	61.78
(Cui et al., 2021)	WideResNet-34-20	88.70	55.03	55.29
(Zhang et al., 2020)	WideResNet-34-10	84.52	56.73	56.95
(Rice et al., 2020)	WideResNet-34-20	85.34	56.89	57.04
(Huang et al., 2020b)	WideResNet-34-10	83.48	55.77	55.99
(Zhang et al., 2019b)	WideResNet-34-10	84.92	54.79	55.00
(Cui et al., 2021)	WideResNet-34-10	88.22	53.84	54.11
(Addepalli et al., 2022a)	ResNet-18	85.71	56.25	56.36
(Chen et al., 2020b)	ResNet-50	86.04	54.22	54.29
(Chen et al., 2022)	WideResNet-34-10	85.32	53.72	54.11
(Addepalli et al., 2022b)	ResNet-18	80.24	55.84	55.97
(Zhang et al., 2019a)	WideResNet-34-10	87.20	46.1	46.23
(Andriushchenko & Flammarion, 2020)	PreActResNet-18	79.84	46.95	47.1
(Wong et al., 2020)	PreActResNet-18	83.34	45.89	46.16
(Ding et al., 2018)	WideResNet-28-4	84.36	50.14	50.33

Table 8: Adversarial accuracy achieved by APGD and ASFW. ImageNet dataset/L_∞ with $\epsilon = 4/255$

paper	Architecture	clean accuracy	APGD	ASFW
(Liu et al., 2023)	ConvNeXt-L	78.02	60.28	60.34
(Singh et al., 2023)	ConvNeXt-L + ConvStem	77.00	58.98	59.02
(Singh et al., 2023)	ConvNeXt-B + ConvStem	75.88	58.36	58.4
(Liu et al., 2023)	ConvNeXt-B	76.7	58.02	58.06
(Singh et al., 2023)	ViT-B + ConvStem	76.30	56.92	57.0
(Singh et al., 2023)	ConvNeXt-S + ConvStem	74.08	55.00	55.06
(Singh et al., 2023)	ConvNeXtt-T + ConvStem	72.70	52.94	53.08
(Singh et al., 2023)	ViT-S + ConvStem	72.58	50.98	51.04
(Debenedetti et al., 2023)	XCiT-L12	73.76	49.6	49.66
(Debenedetti et al., 2023)	XCiT-M12	74.04	47.6	47.7
(Debenedetti et al., 2023)	XCiT-S12	72.34	44.76	44.94
(Salman et al., 2020)	WideResNet-50-2	68.64	40.86	40.9
(Salman et al., 2020)	ResNet-50	64.06	38.50	38.76
(Ilyas et al., 2019)	ResNet-50	62.52	32.18	32.42
(Wong et al., 2020)	ResNet-50	55.64	29.38	28.54
(Salman et al., 2020)	ResNet-18	52.92	29.26	29.48