Finding Dynamics Preserving Adversarial Winning Tickets

Xupeng Shi[*1] **Pengfei Zheng**[*2] **A. Adam Ding**[1] **Yuan Gao**[3] **Weizhong Zhang**[†4]

Abstract

Modern deep neural networks (DNNs) are vulnerable to adversarial attacks and adversarial training has been shown to be a promising method for improving the adversarial robustness of DNNs. Pruning methods have been considered in adversarial context to reduce model capacity and improve adversarial robustness simultaneously in training. Existing adversarial pruning methods generally mimic the classical pruning methods for natural training, which follow the three-stage 'training-pruning-fine-tuning' pipelines. We observe that such pruning methods do not necessarily preserve the dynamics of dense networks, making it potentially hard to be fine-tuned to compensate the accuracy degradation in pruning. Based on recent works of *Neural Tangent Kernel* (NTK), we systematically study the dynamics of adversarial training and prove the existence of trainable sparse sub-network at initialization which can be trained to be adversarial robust from scratch. This theoretically verifies the *lottery ticket hypothesis* in adversarial context and we refer such sub-network structure as *Adversarial Winning Ticket* (AWT). We also show empirical evidences that AWT preserves the dynamics of adversarial training and achieve equal performance as dense adversarial training.

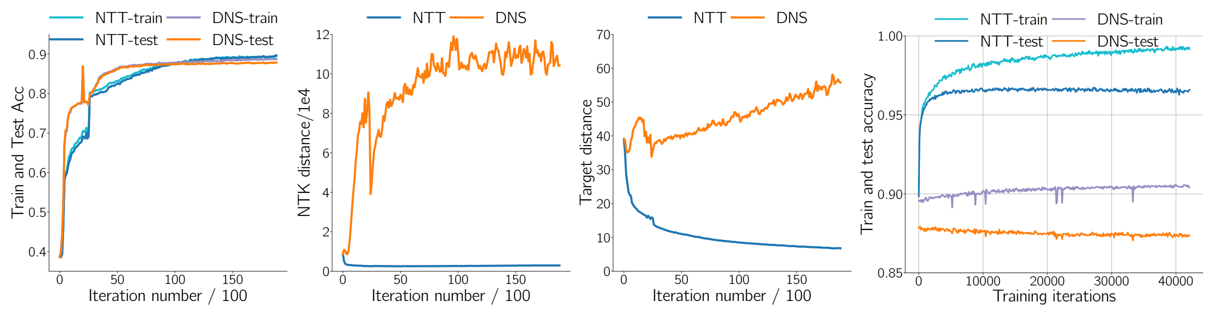
1 Introduction

Deep neural networks (DNN) are widely used as the state-of-art machine learning classification systems due to its great performance gains in recent years. Meanwhile, as pointed out in Szegedy et al. (2014), state-of-the-art DNN are usually vulnerable to attacks by *adversarial examples*, inputs that are distinguishable to human eyes but can fool classifiers to make arbitrary predictions. Such undesirable property may prohibit DNNs from being applied to security-sensitive applications. Various of adversarial defense methods (Goodfellow et al., 2015; Papernot et al., 2016; Samangouei et al., 2018; Schott et al., 2019; Sinha et al., 2018) were then proposed to prevent adversarial examples attack. However, most of the defense methods were quickly broken by new adversarial attack methods. *Adversarial training*, proposed in Madry et al. (2018), is one among the few that remains resistant to adversarial attacks.

On the other hand, DNNs are often found to be highly over-parameterized. Network pruning is shown to be an outstanding method which significantly reduces the model size. Typical pruning algorithms follow the three-stage 'training-pruning-fine-tuning' pipelines, where 'unimportant' weights are pruned according to certain pruning strategies, such as magnitudes of weights. However, as observed in Liu et al. (2019), fine-tuning a pruned model with inherited weights only gives comparable or worse performance than training that model with randomly initialized weights, which suggests that the inherited 'important' weights are not necessarily useful for fine-tuning. We argue below that the change of model outputs dynamics is a potential reason for this phenomenon.

As proposed in Lee et al. (2019), the dynamics of model outputs can be completely described by the *Neural Tangent Kernel* (NTK) and the initial predictions. Hence the difference of dynamics between two neural networks can be quantified by the difference of their NTKs and initial predictions. Based on this result, Liu and Zenke (2020) proposed *Neural Tangent Transfer* (NTT) to find trainable sparse sub-network structure which preserves the dynamics of model outputs by controlling the NTK distance and target distance between dense and sparse networks. In Figure 1, we empirically compare various statistics of NTT with the well-known *Dynamics Network Surgery* (DNS) proposed in Guo et al. (2016) during mask searching and retraining/fine-tuning procedures. In Figure 1 (a), train and test accuracy increase during mask search for both NTT and DNS. This indicates that both methods successfully find sparse network with good performance. However,

[*]Equal contribution, [†]Corresponding author. [1] Northeastern University, [2] Nanjing University, [3] Tencent AI Lab, [4] Hongkong University of Science and Technology. Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.



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

Figure 1: (a)-(c) Statistics of NTT and DNS during mask searching. (d) Train and test accuracy during fine-tuning.

as shown in Figure 1 (b) and (c), the NTK distances and target distances between dense and sparse networks obtained by NTT remain in a low scale, while for DNS these two quantities blow up. This indicates that DNS flows in a different way as NTT, which lead to a different dynamics as the original dense network. As a result, we can see in Figure 1 (d) that the sparse network found by DNS is harder to be fine-tuned, while we can train the sparse network obtained by NTT from scratch to get a better performance. This observation suggests that preserving the dynamics of outputs does help to find trainable sparse structures. Experimental details will be presented in the supplementary materials.

On the other hand, Frankle and Carbin (2018) conjectured the *Lottery Ticket Hypothesis* (LTH), which states that there exists sub-network structure which can reach comparable performance with the original network if trained in isolation. Such sub-network is called *winning ticket*. The existence of winning tickets allows us to train a sparse network from scratch with desirable performance. In particular, NTT as a foresight pruning method, provides a verification of LTH in natural training scenarios. Inspired by this observation, we consider the existence of winning ticket in adversarial context, which also preserves the dynamics of adversarial training. We call such a sparse structure an *Adversarial Winning Ticket* (AWT). The benefit of looking for AWT is that its robustness is guaranteed by robustness of dense adversarial training, which has been theoretically (Tu et al., 2019) and empirically (Madry et al., 2018) justified.

We briefly summarize the the contributions of this paper as follows:

- We systematically study the dynamics of adversarial training and propose a new kernel to quantify the dynamics. We refer this kernel as *Mixed Tangent Kernel* (MTK).
- We propose a method to find AWT, which can be used to verify the LTH in adversarial context. Unlike other pruning methods in adversarial setting, AWT is obtained at initialization.
- We conduct various experiments on real datasets which show that when fully trained, the AWT found by our method can achieves comparable performance when compared to dense adversarial training. These results verify the LTH empirically.

The rest of this paper is organized as follows: In Section 2, we discuss the related works. In Section 3, we develop the theory of adversarial training dynamics and state the existence theorem. In Section 4 we experiment on real datasets to test the performance of AWT. Finally we conclude and discuss some possible future works in Section 5. Proof details and additional experimental results are given in the Appendix.

2 Related Works

2.1 Adversarial Robust Learning

The study of adversarial examples naturally splits into two areas: attack and defense. Adversarial attack methods aim to fool state-of-the-art networks. In general, attack methods consist of white-box attack and black-box attack, depending on how much information about the model we can have. White-box attacks are widely used in generating adversarial examples for training or testing model robustness where we can have all information about the model. This includes *Fast Gradient Sign Method* (FGSM) (Goodfellow et al., 2015), *Deep Fool* (Moosavi-Dezfooli et al., 2016), *AutoAttack* (Croce and Hein, 2020) and so on. Black-box attacks (Chen et al., 2017; Maho et al., 2021) are usually developed to attack model in physical world, therefore we have very limited information about the model structure or parameters.

Meanwhile, defense methods have been studied to train an adversarial robust network which can prevent attacks from adversarial examples. Augmentation with adversarial examples generated by strong attack algorithms has been popular in the literature. Madry et al. (2018) motivates *Projected Gradient Descent* (PGD) as a universal 'first order adversary' and solve a min-max problem by iteratively generating adversarial examples and parameter updating on adversarial examples. Such method is referred as *adversarial training* (AT). The convergence and performance of AT have been theoretically justified by recent works (Tu et al., 2019; Zhang et al., 2020; Gao et al., 2019). Also, methods (Shafahi et al., 2019; Wong et al., 2020) have been developed to speed up the training of AT for large scale datasets such as ImageNet.

2.2 Sparse Learning

Pruning Methods Network pruning (Han et al., 2015; Guo et al., 2016; Zeng and Urtasun, 2018; Li et al., 2016; Luo et al., 2017; He et al., 2017; Zhu and Gupta, 2017; Zhou et al., 2021b,a) has been extensively studied in recent years for reducing model size and improve the inference efficiency of deep neural networks. Since it is a widely-recognized property that modern neural networks are always over-parameterized, pruning methods are developed to remove unimportant parameters in the fully trained dense networks to alleviate such redundancy. According to the granularity of pruning, existing pruning methods can be roughly divided into two categories, i.e., unstructured pruning and structured pruning. The former one is also called weight pruning, which removes the unimportant parameters in an unstructured way, that is, any element in the weight tensor could be removed. The latter one removes all the weights in a certain group together, such as kernel and filter. Since structure is taken into account in pruning, the pruned networks obtained by structured pruning are available for efficient inference on standard computation devices. In both structured and unstructured pruning methods, their key idea is to propose a proper criterion (e.g., magnitude of the weight) to evaluate the importance of the weight, kernel or filter and then remove the unimportant ones. he results in the literature (Guo et al., 2016; Liu et al., 2019; Zeng and Urtasun, 2018; Li et al., 2016) demonstrate that pruning methods can significantly improve the inference efficiency of DNNs with minimal performance degradation, making the deployment of modern neural networks on resource limited devices possible.

Along the research line of LTH, recent works, e.g., SNIP (Lee et al., 2018) and GraSP (Wang et al., 2019), empirically show that it is possible to find a winning ticket at intialization step, without iteratively training and pruning procedure as the classical pruning methods. The key idea is to find a sub-network, which preserves the gradient flow at initialization. NTT (Liu and Zenke, 2020) utilizes the NTK theory and prune the weights by preserving the training dynamics of model outputs, which is captured by a system of differential equations.

Adversarial Pruning Methods Recent works by Guo et al. (2018) have proven sparsity can improve adversarial robustness. A typical way of verifying the *Lottery Ticket Hypothesis* (LTH) is finding the winning ticket by iteratively training and pruning. Such strategy is also considered in adversarial context (Cosentino et al., 2019; Wang et al., 2020; Li et al., 2020; Gilles, 2020), with natural training replaced by adversarial training. Other score based pruning methods have also been considered (Sehwag et al., 2020). Recent work (Fu et al., 2021) also considered sub-network structure with inborn robustness without training.

Other works bring in the model compression methods into sparse adversarial training. Gui et al. (2019) integrates pruning, low-rank factorization and quantization into a unified flexible structural constraint. Ye et al. (2019) proposes concurrent weight pruning to reach robustness. Both works introduce certain sparse constraints and solve the optimization problem under *alternating direction method of multipliers* (ADMM) framework.

2.3 Neural Tangent Kernel

Recent works by Jacot et al. (2018) consider the training dynamics of deep neural network outputs and proposed the *Neural Tangent Kernel* (NTK). Jacot et al. (2018) shows under the infinite width assumption, NTK converges to a deterministic limiting kernel. Hence the training is stable under NTK. NTK theory has been widely used in analyzing the behavior of neural networks. Lee et al. (2019) proves infinitely wide *multilayer perceptrons* (MLP) evolve as linear model, which can be described as the solution of a different equation determined by the NTK at initialization. Arora et al. (2019) further shows that ultra-wide MLPs behave as kernel regression model under NTK. These results have also been applied to different areas in deep learning, such as foresight network pruning (Liu and Zenke, 2020), federated learning (Huang et al., 2021) and so on.

3 Dynamics Preserving Sub-Networks

In this section, we verify the *Lottery Ticket Hypothesis* (LTH) in adversarial context by showing the existence of *Adversarial Winning Ticket* (AWT). We first derive the equations describing the dynamics of adversarial training. Then we propose the optimization problem of finding the AWT by controlling the sparse adversarial training dynamics. Finally we prove an error bound between the dense model outputs and the sparse model outputs, which implies the AWT has the desired theoretical property.

3.1 Dynamics of Adversarial Training

Let $\mathcal{D} = X \times Y = \{(x_1, y_1), \cdots, (x_N, y_N)\}$ be the empirical data distribution, $f_\theta(x) \in \mathbb{R}^{k\times 1}$ the network function defined by a fully-connected network[1], and $f_\theta(X) = \mathrm{vec}\big([f_\theta(x)]_{x\in X}\big) \in \mathbb{R}^{k|\mathcal{D}|\times 1}$ be the model outputs on training data.

Recall that adversarial training solves the following optimization problem:

$$\begin{aligned}\min_\theta \mathcal{L} &= \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \max_{r\in S_\varepsilon(x)} \ell(f_\theta(x+r), y) \\ &= \frac{1}{N}\sum_{i=1}^{N} \max_{r_i\in S_\varepsilon(x_i)} \ell(f_\theta(x_i + r_i), y_i)\end{aligned} \tag{1}$$

The inner sub-problem of this min-max optimization problem is usually solved by an effective attack algorithm. If we use $\tilde{x}_j$ denote the adversarial example of x obtained at j-th step, then any k steps ℓ_p $(1 \le p \le \infty)$ iterative attack algorithm with allowed perturbation strength ε can be formulated as follows:

$$\begin{aligned}&\tilde{x}_0 = x, \quad \tilde{x}_t = \tilde{x}_{t-1} + r_t \quad \tilde{x} = \tilde{x}_k \\ &\text{s.t. } \|r_i\|_p \le \delta \quad \Big\|\sum r_i\Big\|_p \le \varepsilon \quad \forall 1 \le t \le k\end{aligned} \tag{2}$$

In practice, PGD attack as proposed in Madry et al. (2018) is a common choice. Also, for bounded domains, clip operation need to be considered so that each $\tilde{x}_t$ still belongs to the domain. However, such restriction is impossible to be analyzed in general. So we remove the restriction by assuming the sample space is unbounded. In this case, adversarial training algorithm updates the parameters by stochastic gradient descent on adversarial example batches. To be precise, we have the following discrete parameter updates:

$$\theta_{t+1} = \theta_t - \eta \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta}(\tilde{X}_t) \tag{3}$$

For an infinitesimal time $\mathrm{d}t$ with learning rate $\eta_t = \eta \mathrm{d}t$, one can obtain the continuous gradient descent by chain rules as follows:

$$\frac{\mathrm{d}\theta_t}{\mathrm{d}t} = \frac{\theta_{t+\mathrm{d}t} - \theta_t}{\mathrm{d}t} = -\eta \nabla_\theta^T f_t(\tilde{X}_t)\nabla_{f_t}\mathcal{L}(\tilde{X}_t) \tag{4}$$

where we use the short notation $f_t(x) = f_{\theta_t}(x)$ and the following notation for convenience[2]:

$$\nabla_f \mathcal{L}(X) = \begin{bmatrix} | \\ \nabla_f \ell(f(x_i), y_i) \\ | \end{bmatrix} \tag{5}$$

[1]We make this assumption because the NTK theory we are going to apply is valid for fully-connected networks only.

[2]We drop the labels Y here since in adversarial training, the labels assigned to adversarial examples are the same as the clean ones.

Accordingly, we can obtain the following theorem relating to dynamics of adversarial training.

Theorem 1. *Let $f_t(x)$ be the timely dependent network function describing adversarial training process and $\tilde{X}_t$ the adversarial examples generated at time t by any chosen attack algorithm. Then f_t satisfies the following differential equation:*

$$\begin{aligned}\frac{\mathrm{d}f_t}{\mathrm{d}t}(X) &= \nabla_\theta f_t(X)\frac{\mathrm{d}\theta_t}{\mathrm{d}t} \\ &= -\frac{\eta}{N}\nabla_\theta f_t(X)\nabla_\theta^T f_t(\tilde{X}_t)\nabla_{f_t}\mathcal{L}(\tilde{X}_t)\end{aligned} \tag{6}$$

Equation (6) is referred as the dynamics of adversarial training because it describes how the adversarially trained network function f_t evolves along time t. On the other hand, training a adversarial robust network f_θ is the same as solving Equation (6) for given certain initial conditions.

Let $\Theta_t(X, Y) = \nabla_\theta f_t(X)\nabla_\theta^T f_t(Y)$, then $\Theta(X, X)$ is the well-known empirical *Neural Tangent Kernel* (NTK) , which describes the dynamics of natural training as studied in Lee et al. (2019). To be precise, if f^{nat} is the model function of natural training, θ^{nat} the corresponding parameters and $\mathcal{L}_{nat}$ the corresponding loss, then the dynamics of natural training are given by

$$\frac{\mathrm{d}\theta_t^{nat}}{\mathrm{d}t} = -\eta\nabla_\theta^T f_t^{nat}(X)\nabla_{f_t^{nat}}\mathcal{L}_{nat}(X) \tag{7}$$

$$\frac{\mathrm{d}f_t^{nat}}{\mathrm{d}t}(X) = -\eta\Theta_t(X, X)\nabla_{f_t^{nat}}\mathcal{L}_{nat}(X) \tag{8}$$

Detailed calculations can be found in Lee et al. (2019). If we compare Equation (4) with Equation (7), we see that the gradient descent of adversarial training can be viewed as natural training with clean images X replaced by adversarial images $\tilde{X}_t$ at each step. This matches our intuition because in practice, the parameter update is based on the adversarial examples as we discussed above, so adversarial training is closely related to natural training on adversarial images.

However, if we compare Equation (6) and Equation (8), we can see from the evolution of the model outputs that the usual NTK is now replaced by $\Theta_t(X, \tilde{X}_t) = \nabla_\theta f_t(X)\nabla_\theta^T f_t(\tilde{X}_t)$, which we call *Mixed Tangent Kernel* (MTK). Unlike NTK, MTK is not symmetric in general. It involves both clean images X and adversarial images $\tilde{X}_t$. This indicates adversarial training is not simply a naturally model training on adversarial images, but some more complicated training method continuously couples clean images and adversarial images during training procedure. This coupling of clean images and adversarial images gives an intuition why adversarial training can achieve both good model accuracy and adversarial robustness.

Algorithm 1 Finding Adversarial Winning Ticket

```
1: Input: clean images $X$, labels $Y$, model structure $f$, dense initialization $\theta_0$, learning rate $\eta$, adversarial perturbation strength $\varepsilon$, sparsity level $k$, mask update frequency $T_m$.
2: Initialize: initial weight $w_0 = \theta_0$, initial binary mask $m$ based on $w_0$, adversarial images $R_0$, $t = 1$.
3: for $t = 1$ to $N$ do
4:     Sample a mini batch $S$ and calculate the gradient $\nabla_w \mathcal{L}_{awt}$ on $S$.
5:     $w \leftarrow w - \eta \nabla_w \mathcal{L}_{awt} - \beta m \odot w$
6:     if $t \ \% \ T_m = 0$ then
7:         update $m$ according to magnitudes of current $w$
8:     end if
9: end for
10: Return: $m$.
```

3.2 Finding Adversarial Winning Ticket

To verify the LTH in adversarial setting, we need to find out a trainable sparse sub-network which has similar training dynamics as the dense network. We are then aiming to find a mask m with given sparsity density p such that the sparse classifier $f^s(x) = f_{m\odot\theta}(x)$ can be trained to be adversarial robust from scratch[3]. For simplicity, we assume, as in Lee et al. (2019) and Liu and Zenke (2020), the cost function to be squared loss[4] $\ell(f_\theta(x), y) = \frac{1}{2}\|f_\theta(x) - y\|^2$. A discussion of other loss functions, such as cross-entropy, is given in Appendix B. Let $\tilde{X}$ be the collection of adversarial examples as above, then

$$\nabla_{f_t}\mathcal{L}(\tilde{X}_t) = f_t(\tilde{X}_t) - Y \tag{9}$$

And therefore, the dynamics of model outputs of dense network in Equation (6) becomes

$$\frac{\mathrm{d}f_t}{\mathrm{d}t}(X) = -\frac{\eta}{N}\Theta_t(X, \tilde{X}_t)(f_t(\tilde{X}_t) - Y) \tag{10}$$

To achieve our goal, note that the dense classifier $f_t(x)$ in Theorem 1 converges eventually to the solution of adversarial training, so it is supposed to be adversarial robust if fully-trained. On the other hand, the dynamics of the sparse adversarial training $f_t^s(x)$ can be described similarly as:

$$\frac{\mathrm{d}f_t^s}{\mathrm{d}t}(X) = -\frac{\eta}{N}\Theta_t^s(X, \tilde{X}_t^s)(f_t^s(\tilde{X}_t^s) - Y) \tag{11}$$

[3]Without loss of generality, we use superscript s to mean items correspond to sparse networks, while items without superscript correspond to dense networks.

[4]Norms without subscript will denote ℓ_2 norm.

where $\tilde{X}_t^s$ is the collection of adversarial images corresponding to sparse network and $\Theta_t^s(X, \tilde{X}_t^s)$ is the MTK of sparse classifier. Therefore, to get the desired mask m, it is sufficient to make $f_t^s(X) \approx f_t(X)$ for all t. According to Equation (10) and Equation (11), this can be achieved by making the MTK distance and adversarial target distance between dense and sparse networks close enough at any time t in the training. That is to say,

$$\Theta_t(X, \tilde{X}_t) \approx \Theta_t^s(X, \tilde{X}_t^s) \qquad f_t(\tilde{X}_t) \approx f_t^s(\tilde{X}_t^s) \tag{12}$$

for all t. Under mild assumptions, we may expect all these items are determined at $t = 0$ because of the continuous dependence of the solution of differential equations on the initial values. This then leads to the consideration of the following optimization problem:

$$\begin{aligned}\min_m \mathcal{L}_{awt} =& \frac{1}{N}\left\|f_{\theta_0}(\tilde{X}_0) - f^s_{m\odot\theta_0}(\tilde{X}_0^s)\right\|^2 \\ &+ \frac{\gamma^2}{N^2}\left\|\Theta_0(X, \tilde{X}_0) - \Theta_0^s(X, \tilde{X}_0^s)\right\|_F^2\end{aligned} \tag{13}$$

where $\|\cdot\|$ is the ℓ_2 norm of vectors and $\|\cdot\|_F$ is the Frobenius norm of matrices. In equation (13), the first and second items in the right hand side are referred as *target distance* and *kernel distance*, respectively. We call the resulting mask *Adversarial Winning Ticket* (AWT). Our method is summarized in algorithm 1. Since the binary mask m cannot be optimized directly, instead we train a student network $f_{m\odot w}(x)$. The mask m is then updated according to the magnitudes of current weights w after several steps, which is specified by the mask update frequency. To get sparse adversarial robust network, the obtained AWT $f_{m\odot\theta_0}(x)$ will be adversarially trained from scratch.

This intuition can be further illustrated by Figure 2. For each iteration of gradient descent ($t = 1$ to $t = 2$ in the figure), adversarial training contains two steps: adversarial attack (vertical dash line) and parameter update (horizontal dash line). Our goal is to make the blue curve (sparse) close to the green one (dense). This can be done by making the attack and parameter update curves of sparse and dense networks close enough for each time t. However, as we can see from the figure, f_t and f_t^s are determined by the initial condition, hence we get the above optimization problem.

Formally we have the following theorem to estimate the error bound between dense and sparse outputs.

Theorem 2. *Let $f_\theta(x)$ denote the dense network function. Suppose f_θ has identical number of neurons for each layer, i.e. $n_1 = n_2 = \cdots = n_L = n$ and assume n is large enough. Denote $f^s_{m\odot\theta}(x)$ the corresponding sparse network with $1 - p$ weights being pruned. Assume f and f^s have bounded first and second order*

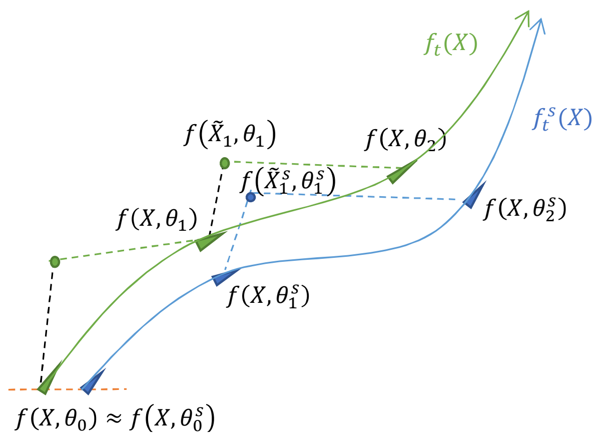

$f_t(X)$
$f_t^s(X)$
$f(\tilde{X}_1, \theta_1)$
$f(X, \theta_2)$
$f(\tilde{X}_1^s, \theta_1^s)$
$f(X, \theta_2^s)$
$f(X, \theta_1)$
$f(X, \theta_1^s)$
$f(X, \theta_0) \approx f(X, \theta_0^s)$

Figure 2: Schematic illustration of networks' outputs evolution under adversarial training. Solid lines represent continuous training dynamics of dense (green) and sparse (blue) networks. Triangular marks locate model outputs at each step under gradient descent. Vertical dash lines represent adversarial attacks and correspondingly, horizontal dash lines represent parameter updates with respect to given adversarial examples. The SGD process from $t = 1$ to $t = 2$ is marked.

derivatives with respect to x, i.e.

$$\max_{t,x} \left\{ \|\partial_x f_t\|_q, \|\partial_x f_t^s\|_q \right\} \leq C_{1,q}$$
$$\max_{t,x} \left\{ \left\|\partial_{xx}^2 f_t\right\|_{p,q}, \left\|\partial_{xx}^2 f_t^s\right\|_{p,q} \right\} \leq C_{2,q}$$

where we choose an ℓ_p attack to generate adversarial examples such that q is the conjugate of p in the sense of $1/p + 1/q = 1$.[5] *Denote the optimal loss value for AWT optimization problem (13) to be $\mathcal{L}_{awt}^* = \alpha^2$. Then for all $t \leq T$ with T the stop time, with learning rate $\eta = O(T^{-1})$, we have*

$$\mathop{\mathbb{E}}_{x \in \mathcal{D}} \|f_t(x) - f_t^s(x)\|^2 \leq 4(\alpha + 4C_q\varepsilon)^2 \qquad (14)$$

where $C_q = C_{1,q} + \varepsilon C_{2,q}$ is a constant.

Note that we put no restriction on any specific attack algorithm, hence we can choose any strong attack algorithm for generating adversarial examples. In practice, PGD attack is commonly chosen for adversarial training. Also, our theoretical results consider any ℓ_p attack with $1 \leq p \leq \infty$. The uniform bound assumption of derivatives are reasonable. If we take the Taylor expansion of f_t with respect to f_0, then the derivatives are functions of θ_t. Since we apply weight decay in our training, θ_t is uniformly bounded for all $t \leq T$, also we only have finite training data, so the derivatives can be assumed to be uniformly bounded. Moreover, C_q can be adjusted by carefully choosing the regularizing constant of weight decay. Proof details and a discussion of the constants are presented in Appendix A.

[5]If $p = \infty$, we take $q = 1$.

Equation (14) shows that the expected error between sparse and dense outputs are bounded by the optimal loss value and adversarial perturbation strength. In practice, the optimial loss value α^2 and adversarial perturbation strength ε are small, we may expect the output of AWT is close to dense output, hence is adversarial robust if fully-trained. Therfore Theorem 2 can be viewed as theoretical justification of the existence of LTH in adversarial setting, and we can find AWT by solving the optimization problem (13).

Theorem 2 reduces to natural training if we take $\varepsilon = 0$. In this case, the AWT found is winning ticket for natural training. Hence Theorem 2 also verifies the classical LTH as a special case. Meanwhile, our method reduces to *Neural Tangent Transfer* (NTT) in Liu and Zenke (2020) and Equation (14) gives an error bound of NTT. Furthermore, for ideal case when $\alpha = 0$, Equation (14) implies $f_t(x) = f_t^s(x)$ for all x, hence the dense and sparse networks have identical outputs for all time t, which extends Proposition 1 in Liu and Zenke (2020).

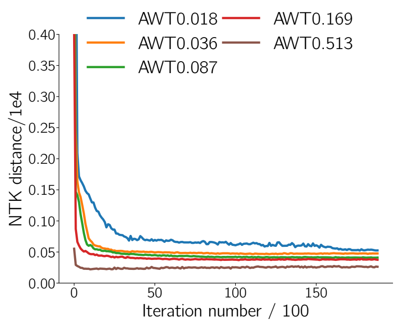
4 Experiments

We now empirically verify the performance of our method. To be precise, we first show the effectiveness of our method in preserving the dynamics of adversarial training, that is, our method can find a sparse sub-network, whose training dynamics are close to the dense network. Then we evaluate the robustness of the sparse neural networks obtained by our method. At last, we give a preliminary experimental result to show the possibility to extend our method to large-scaled problems.

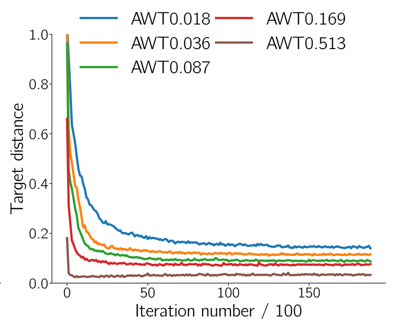
4.1 Implementation

We conduct experiments on standard datasets, including MNIST (LeCun et al., 1998) and CIFAR-10 (Krizhevsky et al., 2009). All experiments are performed in JAX (Bradbury et al., 2018), together with the neural-tangent library (Novak et al., 2020). Due to the high computational and storage costs of NTK, following the experimental setting in Liu and Zenke (2020), we mainly evaluate our proposed method on two networks: MLP and 6-layer CNN. The preliminary experiment of scalability in Section 4.4 is conducted on VGG-16 with CIFAR-10.

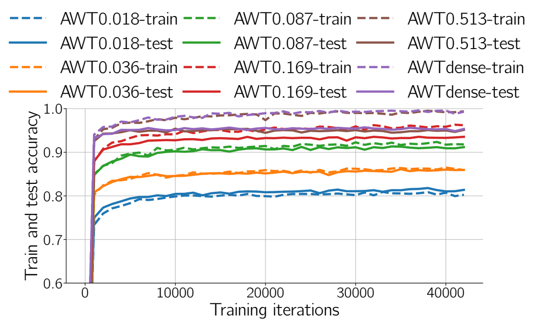
We use PGD attacks for adversarial training and robustness evaluation as suggested in Guo et al. (2020) and Wang et al. (2020). In practice, ℓ_∞ attacks are



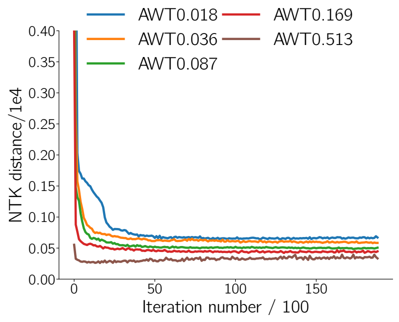
(a) $\ell_2, \varepsilon = 2$



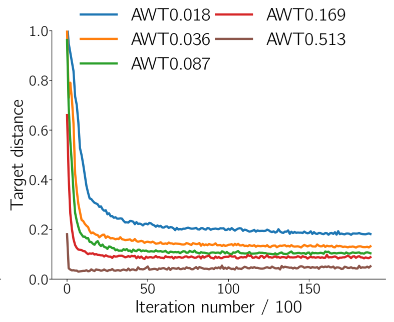
(b) $\ell_2, \varepsilon = 2$



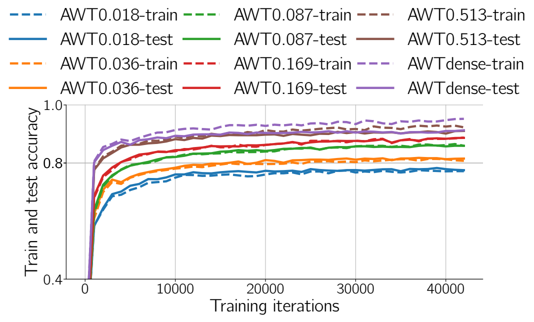
(c) $\ell_2, \varepsilon = 2$



(d) $\ell_\infty, \varepsilon = 0.3$



(e) $\ell_\infty, \varepsilon = 0.3$



(f) $\ell_\infty, \varepsilon = 0.3$

Figure 3: (a) and (b) are the statistics of AWT during mask searching on MNIST with MLP over different density levels under ℓ_2 attack. (c) presents the adversarial training and test accuracy curves in the training process under ℓ_2 attack. (d)-(e) are the results accordingly under ℓ_∞ attack.

commonly used and we use adversarial strength $\varepsilon = 0.3$ for MNIST and $\varepsilon = 8/255$ for CIFAR-10. We take 100 iterations for robustness evaluation, the step size is taken to be $2.5 \cdot \varepsilon/100$ as suggested by Madry et al. (2018). Other detailed experimental configurations such as the learning rate and batch size can be found in the supplementary materials.

4.2 Effectiveness in Preservation of Training Dynamics

In this part, we evaluate the ability of our method in preserving the training dynamics. To be precise, at each density level, we first present the evolution curves of kernel distance and target distance over the whole procedure of finding the adversarial winning ticket. Then we show the adversarial training/testing accuracy during the training process. Since Theorem 2 is valid for any ℓ_p attack algorithms, we also present experimental results under ℓ_2 attacks as well as ℓ_∞ attacks.

Figure 3 (a)/(d) and (b)/(e) show the kernel and target distance curves at different density levels under ℓ_2/ℓ_∞ attack. We can see that as the optimization goes on, both of the kernel and target distances decrease very quickly. As expected, the distance becomes smaller as the density level increases. Figures 3 (c)/(f) show the adversarial training/testing accuracy. We can see that when the density becomes larger, the accuracy curve gets closer to the dense one. This indicates that the training dynamics are well preserved.

density	Cosentino et al. 2019	AWT
full model	98.96/91.14	
51.3%	98.07/60.14	99.13/91.21
16.9%	97.73/59.91	96.58/89.30
8.7%	97.20/57.60	94.48/87.51
3.6%	95.58/48.81	91.74/83.60
1.8%	92.67/38.23	87.69/78.66

Table 1: Test accuracy on natural/adversarial examples over different density levels on MNIST with MLP.

To verify the quality of our winning ticket, we compare our method with the latest work by Cosentino et al. (2019), which finds the winning ticket by iteratively pruning and adversarial training. As indicated by the authors Cosentino et al. (2019), their method is computationally expensive so they only evaluated it on small MLPs. Therefore we only give the comparison

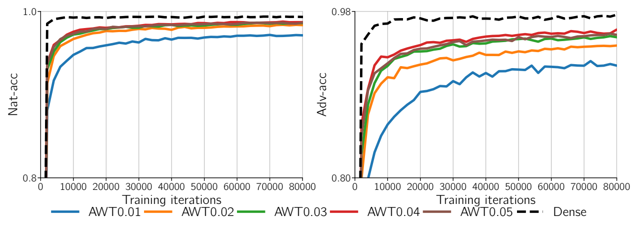


Figure 4: Test accuracy on natural and adversarial examples of CNN trained on MNIST. The density varies in $\{0.01, 0.02, 0.03, 0.04, 0.05\}$.

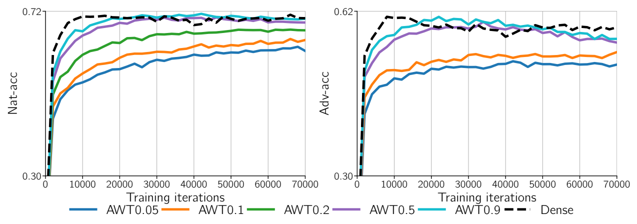


Figure 5: Test accuracy on natural and adversarial examples of CNN trained on CIFAR10.

result on MNIST with MLPs here. Specifically, we give the test accuracy on natural/adversarial examples in Table 1. It shows that our method can outperform the baseline with a large margin. For example, at the density of 1.8%, the adversarial test accuracy of our method is 40% higher than that of Cosentino et al. 2019. We can also see that the accuracy of our method can converge to the dense model much more quickly than the baseline as the density increases. This is benefited from the dynamics preserving property of our sparse sub-network structure.

4.3 Robustness of Trained Sparse Networks

In this section, we evaluate the robustness of fully adversarially trained AWT at different density levels.

We first present the test accuracy on natural and adversarial examples of the CNN models trained on MNIST and CIFAR10. For CIFAR10, the density varies in $\{0.05, 0.1, 0.2, 0.3, \ldots, 0.9\}$. For MNIST, since it can be classified with much sparser networks compared with CIFAR10, in this section, we only check densities between $\{0.01, \ldots, 0.05\}$ and the results under higher density levels can be found in the appendix. Figure 4 and 5 give the results on MNIST and CIFAR10, respectively. Both of these two Figures show that the models trained by our method have high natural and adversarial test accuracy even when the model is very sparse. For example, Figure 4 shows that at the density of 0.03, the model trained by our method can reach the test accuracy of 0.98 and 0.96 on natural and adversarial examples, which are quite close to the dense model. We can also see that the training dynamic, i.e., the test accuracy curves, can converge to that of the dense model as the density increases. And the sparse models obtained by our method can achieve comparable test accuracy with the dense model after trained with the same number of epochs.

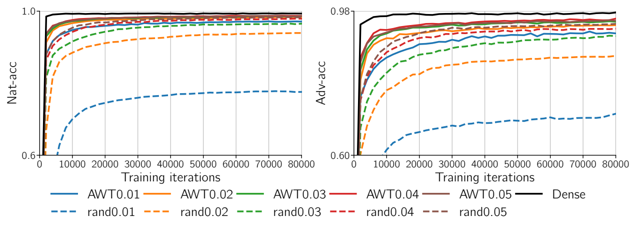


Figure 6: Natural and adversarial test accuracy of the models trained from AWT and random structure on MNIST with CNN.

We then compare the performance of models trained from our sparse structure and the random structure at the same density level. We give the results of CNN trained on MNIST with the density varies in $\{0.01, 0.02, \ldots, 0.05\}$ in Figure 6. More results can be found in the appendix. From Figure 6, our method can achieve much higher natural and adversarial test accuracy than the models trained from the random structure. For example, the network trained by our method at the density of 1% achieves higher adversarial test accuracy than the model trained from random structure at the density of 3%. It also shows that the learning curves of our method are much closer to the dense model than the random structure. This demonstrates that our sparse structures indeed have similar training dynamic with the dense model.

4.4 Discussion on Scalability

In the same situation as existing works on NTK, the expensive computational cost of NTK hinders us from conducting large-scale experiments. Fortunately, the following preliminary experiment shows that the methods such as sampling on the NTK matrix could be promising to improve the efficiency of our method.

To be specific, inspired from Jacobi preconditioner method (Concus et al., 1976) in optimization theory, we sample only the diagonal elements in the MTK matrices $\Theta_0(X, \tilde{X}_0)$ and $\Theta_0^s(X, \tilde{X}_0^s)$ in Equation (13) and keep all other settings the same as above. In this way, the computational complexity of training can be significantly reduced. We conduct a preliminary experiment on CIFAR-10 with large-sized model VGG-16. To the best of our knowledge, we are the first to apply NTK into models as large as VGG-16. The result is presented in Table 2. It shows that the performance degradation caused by the sampling is unnoticeable. We will investigate this kind of approaches to improve the efficiency of NTK based methods in the future.

density	WT	S-AWT	IWI
10%	45.15	46.68	47.53
5%	40.10	45.87	47.59

Table 2: Adversarial test accuracy of the VGG16 trained on CIFAR10 at different density levels. WT represents winning ticket, which applies iteratively pruning and adversarial training method. S-AWT is AWT with sampling. IWI represents Inverse Weight Inheritance. WT and IWI results are copied from Wang et al. (2020).

5 Conclusions

We study the evolution of adversarially trained networks and obtain a new type of kernel quantifying the dynamics of adversarial training. We verify the *Lottery Ticket Hypothesis* (LTH) (Frankle and Carbin, 2018) in adversarial setting by solving an optimization problem to find adversarial winning ticket (AWT), which can be adversarially trained to be robust from scratch. Our work includes the classical LTH in natural training as a special case and extends the bound of *Neural Tangent Transfer* (NTT) (Liu and Zenke, 2020). Unlike most of the adversarial pruning methods, which follow the classical pruning pipelines, i.e., prune the network during training, our method is a foresight pruning method. To the best of our knowledge, this is the first result showing that to identify winning tickets at initialization is possible in the adversarial training scenario.

We would like to point out our main contribution is the verification of LTH in adversarial setting, rather than proposing a practical pruning method. Similar to existing NTK based methods, We did not conduct experiment on large scale datasets, such as ImageNet, due to the well-known fact that the computation of NTK is expensive. Moreover, the preliminary experimental result in Section 4.4 indicates certain approximation methods such as sampling can be promising to reduce the computational cost of NTK without noticeable performance degradation. We will further explore this possibility in our future work.

References

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 8141–8150, 2019.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

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. AISec '17, page 15–26, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450352024. doi: 10.1145/3128572.3140448. URL `https://doi.org/10.1145/3128572.3140448`.

Paul Concus, Gene H Golub, and Dianne P O'Leary. A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations. In *Sparse matrix computations*, pages 309–332. Elsevier, 1976.

Justin Cosentino, Federico Zaiter, Dan Pei, and Jun Zhu. The search for sparse, robust neural networks. *arXiv preprint arXiv:1912.02386*, 2019.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.

Yonggan Fu, Qixuan Yu, Yang Zhang, Shang Wu, Xu Ouyang, David Cox, and Yingyan Lin. Drawing robust scratch tickets: Subnetworks with inborn robustness are found within randomly initialized networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Ruiqi Gao, Tianle Cai, Haochuan Li, Cho-Jui Hsieh, Liwei Wang, and Jason D Lee. Convergence of adversarial training in overparametrized neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

James Gilles. *The lottery ticket hypothesis in an adversarial setting*. PhD thesis, Massachusetts Institute of Technology, 2020.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL `http://arxiv.org/abs/1412.6572`.

Yihong Gu, Weizhong Zhang, Cong Fang, Jason D Lee, and Tong Zhang. How to characterize the landscape of overparameterized convolutional neural networks. *Advances in Neural Information Processing Systems*, 33:3797–3807, 2020.

Shupeng Gui, Haotao N Wang, Haichuan Yang, Chen Yu, Zhangyang Wang, and Ji Liu. Model compression with adversarial robustness: A unified optimization framework. In *Advances in Neural Information Processing Systems*, pages 1285–1296, 2019.

Minghao Guo, Yuzhe Yang, Rui Xu, Ziwei Liu, and Dahua Lin. When nas meets robustness: In search of robust architectures against adversarial attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 631–640, 2020.

Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. In *NIPS*, 2016.

Yiwen Guo, Chao Zhang, Changshui Zhang, and Yurong Chen. Sparse dnns with improved adversarial robustness. In *Advances in neural information processing systems*, pages 242–251, 2018.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397, 2017.

Baihe Huang, Xiaoxiao Li, Zhao Song, and Xin Yang. Fl-ntk: A neural tangent kernel-based framework for federated learning analysis. In *International Conference on Machine Learning*, pages 4423–4434. PMLR, 2021.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 8572–8583. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/0d1a9651497a38d8b1c3871c84528bd4-Paper.pdf`.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2018.

Bai Li, Shiqi Wang, Yunhan Jia, Yantao Lu, Zhenyu Zhong, Lawrence Carin, and Suman Jana. Towards practical lottery ticket hypothesis for adversarial training. *arXiv preprint arXiv:2003.05733*, 2020.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

Tianlin Liu and Friedemann Zenke. Finding trainable sparse networks through neural tangent transfer. *arXiv preprint arXiv:2006.08228*, 2020.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rJlnB3C5Ym`.

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

Thibault Maho, Teddy Furon, and Erwan Le Merrer. Surfree: A fast surrogate-free black-box attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10430–10439, June 2021.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. In *International Conference on Learning Representations*, 2020. URL `https://github.com/google/neural-tangents`.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE, 2016.

Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=BkJ3ibb0-`.

L Schott, J Rauber, M Bethge, and W Brendel. Towards the first adversarially robust neural network model on mnist. In *Seventh International Conference on Learning Representations (ICLR 2019)*, pages 1–16, 2019.

Vikash Sehwag, Shiqi Wang, Prateek Mittal, and Suman Jana. Hydra: Pruning adversarially robust neural networks. *Advances in Neural Information Processing Systems*, 33:19655–19666, 2020.

Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/7503cfacd12053d309b6bed5c89de212-Paper.pdf`.

Xupeng Shi and A Adam Ding. Understanding and quantifying adversarial examples existence in linear classification. *arXiv preprint arXiv:1910.12163*, 2019.

Aman Sinha, Hongseok Namkoong, and John Duchi. Certifiable distributional robustness with principled adversarial training. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=Hk6kPgZA-`.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2014.

Zhuozhuo Tu, Jingwei Zhang, and Dacheng Tao. Theoretical analysis of adversarial learning: A minimax approach. In H. Wallach, H. Larochelle, A. Beygelzimer,

F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/16bda725ae44af3bb9316f416bd13b1b-Paper.pdf`.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2019.

Shufan Wang, Ningyi Liao, Liyao Xiang, Nanyang Ye, and Quanshi Zhang. Achieving adversarial robustness via sparsity. *arXiv preprint arXiv:2009.05423*, 2020.

Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=BJx040EFvH`.

Shaokai Ye, Kaidi Xu, Sijia Liu, Hao Cheng, Jan-Henrik Lambrechts, Huan Zhang, Aojun Zhou, Kaisheng Ma, Yanzhi Wang, and Xue Lin. Adversarial robustness vs. model compression, or both? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 111–120, 2019.

Wenyuan Zeng and Raquel Urtasun. Mlprune: Multi-layer pruning for automated neural network compression. 2018.

Yi Zhang, Orestis Plevrakis, Simon S Du, Xingguo Li, Zhao Song, and Sanjeev Arora. Over-parameterized adversarial training: An analysis overcoming the curse of dimensionality. *Advances in Neural Information Processing Systems*, 33:679–688, 2020.

Xiao Zhou, Weizhong Zhang, Zonghao Chen, Shizhe Diao, and Tong Zhang. Efficient neural network training via forward and backward propagation sparsification. *Advances in Neural Information Processing Systems*, 34, 2021a.

Xiao Zhou, Weizhong Zhang, Hang Xu, and Tong Zhang. Effective sparsification of neural networks with global sparsity constraint. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3599–3608, 2021b.

Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

Supplementary Materials: Finding Dynamics Preserving Adversarial Winning Tickets

A Proof of Theorem

Let $\mathcal{D} = X \times Y = \{(x_1, y_1), \cdots, (x_N, y_N)\}$ be the empirical data distribution, $f_\theta(x) \in \mathbb{R}^k$ the network function, and $f_\theta(X) = \mathrm{vec}\big([f_\theta(x)]_{x\in X}\big) \in \mathbb{R}^{k|\mathcal{D}|\times 1}$ be the model outputs. Note that adversarial training optimizes the following objective function

$$\min_\theta \mathcal{L} = \mathbb{E}_{(x,y)\sim\mathcal{D}} \max_{r\in S_\varepsilon(x)} \ell(f_\theta(x+r), y) \tag{A.1}$$

We use the following notation for convenience:

$$\nabla_f \mathcal{L}(X) = \begin{bmatrix} | \\ \nabla_f \ell(f(x_i), y) \\ | \end{bmatrix}$$

For squared loss $\ell(f(x), y) = \frac{1}{2}\|f(x) - y\|_2^2$, this is just

$$\nabla_f \mathcal{L}(X) = \begin{bmatrix} | \\ f(x_i) - y_i \\ | \end{bmatrix} = f(X) - Y$$

Theorem 3. *Let $f_t(x) := f_{\theta_t}(x)$ be the timely dependent network function and $\tilde{X}_t$ the adversarial examples generated at time t. Then the continuous gradient descent of adversarial training is:*

$$\frac{\mathrm{d}\theta_t}{\mathrm{d}t} = -\frac{\eta}{N}\nabla_\theta^T f_t(\tilde{X}_t)\nabla_f \mathcal{L}(\tilde{X}_t) \tag{A.2}$$

As a result, f_t satisfies the following differential equation:

$$\begin{aligned}\frac{\mathrm{d}f_t}{\mathrm{d}t}(X) &= \nabla_\theta f_t(X)\frac{\mathrm{d}\theta_t}{\mathrm{d}t} \\ &= -\frac{\eta}{N}\nabla_\theta f_t(X)\nabla_\theta^T f_t(\tilde{X}_t)\nabla_f \mathcal{L}(\tilde{X}_t)\end{aligned} \tag{A.3}$$

Proof. Note that at time t adversarial training consists of an attack step and parameter update step. To be precise, for some chosen strong attack algorithm, we first generate the set of adversarial examples $\tilde{X}_t$, then update the parameter according to

$$\begin{aligned}\theta_{t+dt} &= \theta_t - \eta_t \frac{\partial \mathcal{L}}{\partial \theta_t}(\tilde{X}_t) \\ &= \theta_t - \frac{\eta_t}{N}\nabla_\theta^T f_t(\tilde{X}_t)\nabla_f \mathcal{L}(\tilde{X}_t)\end{aligned} \tag{A.4}$$

If we take the infinitesimal learning rate to be[6] $\eta_t = \eta dt$ and taking the limit $dt \to 0$, we obtain the continuous gradient descent as in Equation (A.2). The evolution of f_t in Equation (A.3) is a direct result by chain rule. □

[6]The discrete parameter update corresponds to the case when $dt = 1$.

We consider the following optimization problem to find AWT:

$$\min_m \mathcal{L}_{awt} = \frac{1}{N}\left\| f_0(\tilde{X}_0) - f_0^s(\tilde{X}_0^s) \right\|^2 + \frac{\gamma^2}{N^2}\left\| \Theta_0(X, \tilde{X}_0) - \Theta_0^s(X, \tilde{X}_0^s) \right\|_F^2 \tag{A.5}$$

where $\Theta(X, Y) = \nabla_\theta f(X) \nabla_\theta f^T(Y)$ is the empirical neural tangent kernel and sup-script s represents quantities involving sparse structure.

Theorem 4 (**Existence of adversarial winning ticket**). *Let $f_\theta(x)$ denote the dense network function. Suppose f_θ has identical number of neurons for each layer, i.e. $n_1 = n_2 = \cdots = n_L = n$ and assume n is large enough. Denote $f^s_{m\odot\theta}(x)$ the corresponding sparse network with $1-p$ weights being pruned. Assume f and f^s have bounded first and second order derivatives with respect to x, i.e.*

$$\max_{t,x} \left\{ \|\partial_x f_t\|_q, \|\partial_x f_t^s\|_q \right\} \le C_{1,q}$$
$$\max_{t,x} \left\{ \left\|\partial_{xx}^2 f_t\right\|_{p,q}, \left\|\partial_{xx}^2 f_t^s\right\|_{p,q} \right\} \le C_{2,q}$$

where we choose an ℓ_p attack to generate adversarial examples such that q is the conjugate of p in the sense of $1/p + 1/q = 1$.[7] *Denote the optimal loss value for AWT optimization problem (13) to be $\mathcal{L}^*_{awt} = \alpha^2$. Then for all $t \le T$ with T the stop time, with learning rate $\eta = O(T^{-1})$, we have*

$$\mathop{\mathbb{E}}_{x\in\mathcal{D}} \|f_t(x) - f_t^s(x)\|^2 \le 4(\alpha + 4C_q\varepsilon)^2 \tag{A.6}$$

where $C_q = C_{1,q} + \varepsilon C_{2,q}$ is a constant.

In order to prove the theorem, we need the following lemma of estimation of error bound.

Lemma 1. *For any $1 < p \le \infty$, assume an k iterative ℓ_p attack algorithm updates as $\tilde{x}_0 = x$, $\tilde{x}_t = \tilde{x}_{t-1} + r_t$ with $\|r_t\|_p \le \delta$ for any $1 \le t \le k$ and with total allowed perturbation strength $\|\sum r_j\|_p \le \varepsilon$. Assume $k\delta \le 2\varepsilon$. If the neural network function f has bounded first and second order derivative with respect to x, i.e. $\|\partial_x f\|_q \le C_{1,q}$, $\left\|\partial^2_{xx} f\right\|_{p,q} \le C_{2,q}$, where q be the conjugate of p such that $1/p + 1/q = 1$. Then for any adversarial example $\tilde{x}$ generated by the attack algorithm, we have*

$$|f(\tilde{x}) - f(x)| \le 2\varepsilon C_{1,q} + 2\varepsilon^2 C_{2,q} = 2\varepsilon C_q \tag{A.7}$$

Proof of Lemma: Consider the series of second order Taylor expansions for any $1 \le t \le k$

$$f(\tilde{x}_t) - f(\tilde{x}_{t-1}) = \partial_x f(\tilde{x}_{t-1}) r_t + \frac{1}{2} r_t^T \partial^2_{xx} f(\xi_{t-1}) r_t \tag{A.8}$$

We have

$$\begin{aligned} |f(\tilde{x}_t) - f(\tilde{x}_{t-1})| &\le \|\partial_x f(\tilde{x}_{t-1}) r_t\| + \left\| \frac{1}{2} r_t^T \partial^2_{xx} f(\xi_{t-1}) r_t \right\| \\ &\le \|\partial_x f(\tilde{x}_{t-1})\|_q \|r_t\|_p + \frac{1}{2}\|r_t\|_p \left\|\partial^2_{xx} f(\xi_{t-1}) r_t\right\|_q \\ &\le \|\partial_x f(\tilde{x}_{t-1})\|_q \delta + \frac{1}{2}\delta \left\|\partial^2_{xx} f(\xi_{t-1})\right\|_{p,q} \|r_t\|_p \\ &\le C_{1,q}\delta + \frac{1}{2} C_{2,q}\delta^2 \end{aligned} \tag{A.9}$$

where we use Hölder inequality in the second step and definition of (p, q) norm in the third step. On the other hand, by Mean-value theorem we have

$$\partial_x f(\tilde{x}_t) = \partial_x f(\tilde{x}_{t-1}) + \partial^2_{xx} f(\eta_{t-1}) r_t \tag{A.10}$$

Hence we have

$$\begin{aligned} \|\partial_x f(\tilde{x}_t)\|_q &\le \|\partial_x f(\tilde{x}_{t-1})\|_q + \left\|\partial^2_{xx} f(\eta_{t-1}) r_t\right\|_q \\ &\le \|\partial_x f(\tilde{x}_{t-1})\|_q + \left\|\partial^2_{xx} f(\eta_{t-1})\right\|_{p,q} \|r_t\|_p \\ &\le C_{1,q} + C_{2,q}\delta \end{aligned} \tag{A.11}$$

[7] If $p = \infty$, we take $q = 1$.

where we use Minkowski inequality in the first step. Together, we have the following estimation:

$$\begin{aligned}|f(\tilde{x}) - f(x)| =&|f(\tilde{x}_k) - f(\tilde{x}_0)|\\ \leq&\sum_{t=1}^{k}|f(\tilde{x}_t) - f(\tilde{x}_{t-1})|\\ \leq&\delta\sum_{t=1}^{k}\left\|\partial_x f(\tilde{x}_{t-1})\right\|_q + \frac{1}{2}\delta^2\sum_{t=1}^{k}\left\|\partial_{xx}^2 f(\xi_{t-1})\right\|_{p,q}\\ =&\delta\Bigg(\sum_{t=1}^{k}\left\|\partial_x f(\tilde{x}_0)\right\|_q + C_{2,q}\delta(t-1)\Bigg) + \frac{1}{2}\delta^2 k C_{2,q}\\ =&k\delta C_{1,q} + \frac{1}{2}k^2\delta^2 C_{2,q}\end{aligned} \tag{A.12}$$

Then Equation (A.7) is valid if we plug in the assumption $k\delta \leq 2\varepsilon$. □

Remark 1: We did not specify any particular algorithm in the presentation of our lemma. In practice, k steps PGD attacks is the common choice for inner subproblem of adversarial training. For k steps PGD attack, the number of attack iteration is usually taken to be 7 (ImageNet) or 20 (MNIST/CIFAR-10), so we may assume $k = \Omega(1)$ for future use. The assumption $k\delta \leq 2\varepsilon$ is also for practical consideration, where we usually choose the step size δ approximately to be $2\varepsilon/k$ as suggested in Madry et al. (2018).

Remark 2: We might get more accurate bound by looking deeper into the dynamics of continuous first-order attack:

$$\frac{\mathrm{d}x_t}{\mathrm{d}t} = \frac{\mathrm{d}\ell}{\mathrm{d}x}(f_t(x_t)), \qquad x_0 = x \tag{A.13}$$

The analysis of the above differential equations would possibly weaken the current assumption on derivatives. We would leave this as a future work.

Now we are ready to prove the main theorem.

Proof of Theorem. Suppose $\mathcal{L}^*_{awt} = \alpha^2$, then we have

$$\left\|f_0(\tilde{X}_0) - f_0^s(\tilde{X}_0^s)\right\| \leq \sqrt{N}\alpha, \qquad \left\|\Theta_0(X, \tilde{X}_0) - \Theta_0^s(X, \tilde{X}_0^s)\right\|_F \leq N\frac{\alpha}{\gamma}$$

To bound the distance between dense and sparse output, we do induction on time t and prove the following stronger estimation

$$\|f_t(X) - f_t^s(X)\| \leq \left(1 + \frac{t}{T}\right)\sqrt{N}\big(\alpha + 4C_q\varepsilon\big) \tag{A.14}$$

where $C_q = C_{1,q} + \varepsilon C_{2,q}$ and $C_{1,q}$ and $C_{2,q}$ are bounds of first and second order derivative of f. Note that at $t = 0$, we have

$$\begin{aligned}\|f_0(X) - f_0^s(X)\| \leq& \left\|f_0(\tilde{X}_0) - f_0^s(\tilde{X}_0^s)\right\| + \left\|f_0(\tilde{X}_0) - f_0(X)\right\| + \left\|f_0^s(\tilde{X}_0^s) - f_0^s(X)\right\|\\ \leq&\sqrt{N}\big(\alpha + 4C_q\varepsilon\big)\end{aligned} \tag{A.15}$$

At time t, assume we have

$$\|f_t(X) - f_t^s(X)\| \leq \left(1 + \frac{t}{T}\right)\sqrt{N}\big(\alpha + 4C_q\varepsilon\big) \tag{A.16}$$

Then according to the dynamical equation (A.3), we have

$$\begin{aligned}f_{t+1}(X) &= f_t(X) - \frac{\eta}{N}\Theta(X, \tilde{X}_t)\big(f_t(\tilde{X}_t) - Y\big)\\ f_{t+1}^s(X) &= f_t^s(X) - \frac{\eta}{N}\Theta^s(X, \tilde{X}_t^s)\big(f_t^s(\tilde{X}_t^s) - Y\big)\end{aligned} \tag{A.17}$$

Then

$$\left\|f_{t+1}(X)-f^s_{t+1}(X)\right\| \leq \|f_t(X)-f^s_t(X)\| + \frac{\eta}{N}\left\|(\Theta_t-\Theta^s_t)(f_t(\tilde{X}_t)-Y)\right\| + \frac{\eta}{N}\left\|\Theta^s_t(f_t(\tilde{X}_t)-f^s_t(\tilde{X}^s_t))\right\| \quad (A.18)$$

Let $K_t(X,X)=\nabla_\theta f_t(X)\nabla_\theta^T f_t(X)$ be the empirical neural tangent kernel at time t, then according to Theorem 2.1 in Lee et al. (2019), $\|K_t-K_0\|_F=\frac{C_K}{\sqrt{n}}$, where n is the number of neurons in each layer. Therefore we have

$$\begin{aligned}
&\left\|\Theta_t(X,\tilde{X}_t)-\Theta^s_t(X,\tilde{X}^s_t)\right\|_F \\
\leq&\left\|\Theta_t(X,\tilde{X}_t)-\Theta_0(X,\tilde{X}_0)\right\|_F + \left\|\Theta^s_t(X,\tilde{X}^s_t)-\Theta^s_0(X,\tilde{X}_0)\right\|_F + \left\|\Theta_0(X,\tilde{X}_0)-\Theta^s_0(X,\tilde{X}^s_0)\right\|_F \\
\leq&\|K_t(X,X)-K_0(X,X)\|_F + \|K^s_t(X,X)-K^s_0(X,X)\|_F + \left\|\nabla_\theta f_t(X)\nabla^2_{\theta,x}f(\xi)R_t\right\| \\
&+\left\|\nabla_\theta f^s_t(X)\nabla^2_{\theta,x}f(\tilde{\xi})R^s_t\right\|_F + N\frac{\alpha}{\gamma} \\
\leq&\frac{C_K}{\sqrt{n}}\left(1+\frac{1}{\sqrt{p}}\right)+2N\varepsilon C_{1,q}C_{2,q}+N\frac{\alpha}{\gamma}
\end{aligned} \quad (A.19)$$

And by Lemma (1), we have

$$\begin{aligned}
\left\|f(\tilde{X}_t)-f^s(\tilde{X}^s_t)\right\| &\leq \|f_t(X)-f^s_t(X)\| + \left\|f(\tilde{X}_t)-f_t(X)\right\| + \left\|f^s_t(\tilde{X}_t)-f^s_t(X)\right\| \\
&\leq \left(1+\frac{t}{T}\right)\sqrt{N}\big(\alpha+4C_q\varepsilon\big)+4\sqrt{N}C_q\varepsilon
\end{aligned} \quad (A.20)$$

Then Equation (A.18) reads

$$\begin{aligned}
&\left\|f_{t+1}(X)-f^s_{t+1}(X)\right\| \\
\leq&\left(1+\frac{t}{T}\right)\sqrt{N}\big(\alpha+4C_q\varepsilon\big)+\frac{\eta}{N}\left(\frac{C_K}{\sqrt{n}}\left(1+\frac{1}{\sqrt{p}}\right)+2N\varepsilon C_{1,q}C_{2,q}+N\frac{\alpha}{\gamma}\right)\sqrt{N}c \\
&+\frac{\eta}{N}NC_{2,q}\left[\left(1+\frac{t}{T}\right)\sqrt{N}\big(\alpha+4C_q\varepsilon\sqrt{N}\big)+4\sqrt{N}C_q\varepsilon\right]
\end{aligned} \quad (A.21)$$

where $c=\max\{|f(x)-y|:x\in X\}$ is bounded essentially. Take

$$\eta=\min\left\{\frac{1}{T(2+\frac{c}{\gamma})},\quad \frac{4C_q\varepsilon}{T\big(2cC_{1,q}C_{2,q}+8C_q)\big)}\right\} \quad (A.22)$$

One can check that if N is sufficiently large such that $\frac{C_K}{\sqrt{Nn}(1+\frac{1}{\sqrt{p}})}\to 0$, then Equation (A.21) reads

$$\left\|f_{t+1}(X)-f^s_{t+1}(X)\right\| \leq \left(1+\frac{t+1}{T}\right)\sqrt{N}\big(\alpha+4C_q\varepsilon\big) \quad (A.23)$$

which completes the proof. □

B Discussion on Possible Extensions to Other Loss Functions

We only consider the squared loss for simplicity in our main theorem. However, it is possible to extend our result to the cross-entropy loss case. Actually, we use cross-entropy loss in our experiment, so we have already checked our method empirically.

To see how our method works theoretically for cross-entropy loss, let

$$\ell_{ce}(f(x),y)=-\log\frac{e^{f_y}}{\sum_i e^{f_i}} \quad (B.24)$$

be the cross-entropy loss, where $f(x)=[\cdots,f_j(x),\cdots]^T$ is the model output. Note that we have

$$\frac{\partial\ell_{ce}}{\partial f_y}=\frac{e^{f_y}}{\sum_i e^{f_i}}-1,\qquad \frac{\partial\ell_{ce}}{\partial f_j}=\frac{e^{f_j}}{\sum_i e^{f_i}},j\neq y \quad (B.25)$$

Therefore,

$$\|\nabla_f \ell_{ce}(f(x), y) - \nabla_{f^s} \ell_{ce}(f(x), y)\|^2 = \sum_j \left(\frac{e^{f_j}}{\sum_i e^{f_i}} - \frac{e^{f_j^s}}{\sum_i e^{f_i^s}} \right)^2 \tag{B.26}$$

In practice, we may expect the change of model outputs is usually larger than the outputs after cross-entropy loss, i.e.

$$\forall j, \left(\frac{e^{f_j}}{\sum_i e^{f_i}} - \frac{e^{f_j^s}}{\sum_i e^{f_i^s}} \right)^2 \leq [f_i(x) - f_i^s(x)]^2 \tag{B.27}$$

Hence, this implies

$$\|\nabla_f \ell_{ce}(f(x), y) - \nabla_f \ell_{ce}(f^s(x), y)\| \leq \|f(x) - f^s(x)\| \tag{B.28}$$

Recall that the dynamics of adversarial training is

$$\frac{\mathrm{d} f_t}{\mathrm{d}t}(X) = -\eta \Theta_t(X, \tilde{X}_t) \nabla_{f_t} \mathcal{L}(\tilde{X}_t) \tag{B.29}$$

We may expect the optimization problem of finding adversarial winning ticket for general loss function is

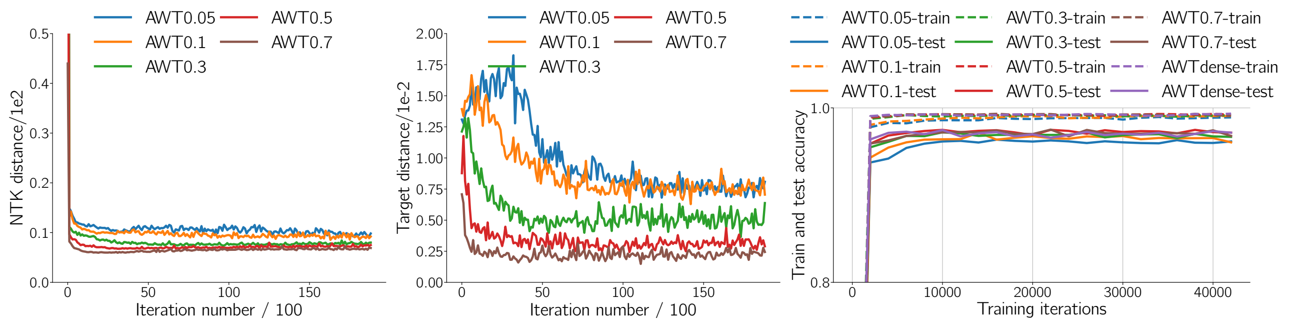
$$\min_m \mathcal{L}'_{awt} = \frac{1}{N} \left\| \nabla_f \mathcal{L}(\tilde{X}_0) - \nabla_{f^s} \tilde{\mathcal{L}}(\tilde{X}'_0) \right\| + \frac{1}{N^2} \left\| \Theta_0(X, \tilde{X}_0) - \Theta_0^s(X, \tilde{X}'_0) \right\|_F \tag{B.30}$$

Now we compare the optimization problem (B.30) with problem (A.5). The first term of problem (B.30), as discussed above, is bounded by $\|f_0(X + R_0) - f_0^s(X + R_0^s)\|$. This shows that, if we find adversarial winning ticket according to optimization problem (B.30), the resulting training dynamics is bounded by the one we obtained before, so is close to the dynamics of dense network also. This suggests that the optimization problem (A.5) is general for both squared loss and cross-entropy loss.

Remark: We may expect that for many other loss functions, the following condition is true

$$\|\nabla_f \ell(f(x), y) - \nabla_{f^s} \ell(f^s(x), y)\| \leq C \|f(x) - f^s(x)\| \tag{B.31}$$

Our method generalizes to the case using any loss function satisfying condition (B.31). Actually, for any convex loss function ℓ, if the second order derivative is bounded, then the above condition (B.31) is true by Taylor expansion.



(a) (b) (c)

Figure 7: (a) and (b) are the statistics of AWT in the procedure of searching masks on MNIST with CNN over different density levels. (c) presents the adversarial training and test accuracy curves in the training process

C More Experimental Results

C.1 Toy Example

We provide here a toy example to illustrate our method. Consider a binary classification problem of mixed Gaussian distribution $p(x) = 0.5\mathcal{N}(\mu_+, \Sigma_+) + 0.5\mathcal{N}(\mu_-, \sigma_-)$ and any linear classifier $C(x) = \text{sgn}(f_\theta(x)) = \text{sgn}(\theta \cdot x)$. Let

$\mu = 0.5(\mu_+ - \mu_-)$ be the mean difference and suppose $\Sigma_+ = \Sigma_- = \sigma^2 I_d$, then the adversarial accuracy with given adversarial perturbation strength ε can be calculated explicitly using the following result in Shi and Ding (2019):

$$\begin{aligned} p_{adv-acc} = 1 - p_{adv} &= p_m + \Phi\left(\frac{\theta \cdot \mu}{\|\theta\|\,\sigma} - \frac{\varepsilon}{\sigma}\right) \\ &= p_m + \Phi\left(\frac{\|\mu\|}{\sigma}\cos\gamma - \frac{\varepsilon}{\sigma}\right) \end{aligned} \tag{C.32}$$

where p_m is the misclassification rate, p_{adv} is the probability of the existence of adversarial examples, and Φ is the cumulative density function of standard normal distribution. This shows the adversarial robustness of $C(x)$ can be measured explicitly by the deflection angle γ between θ and μ. In particular, $p_{adv-acc}$ attains its maximum at $\theta = \mu$, i.e. Bayes classifier is the best classifier in the sense of adversarial robustness.

To illustrate our idea, we randomly generate 5000 data points from $p(x)$, where $\mu_+ = [3, 0, \cdots, 0] = -\mu_-$, $\Sigma_+ = \Sigma_- = I_d$ and the dimension is $d = 100$. We minimize the loss in equation (A.5) to obtain a sparse robust structure with a sparsity level at 10%, i.e. we only keep 10 nonzero coordinates of θ. Then we adversarially train the AWT to get the sparse robust network. We use SVM as baseline of model accuracy and standard adversarial training as baseline of adversarial accuracy. The result is summarized in table 3.

	Bayes	SVM	Adv.Tr	AWT
acc.	0.999	0.995	0.995	0.995
ang.	0.0	0.555	0.202	0.118
cos.	1.0	0.850	0.980	0.993
rob.	0.843	0.714	0.823	0.838

Table 3: 'cos.' represents cosine of angle. 'rob.' represents adversarial accuracy.

Table 3 shows that SVM reaches comparable model accuracy as Bayes model but with a large deflection angle ($0.555 \approx 31.8°$), this simulates the case that natural training can reach high accuracy but fails to be adversarial robust. Adversarial training reaches the same model accuracy but also improved adversarial accuracy (0.823), which is very close to the one of Bayes classifier. This is also reflected by the deflection angle ($0.202 \approx 11.6°$), which shows a significant decrease when compared with the deflected angle of SVM. Our method (AWT) gets slightly better performance than the dense adversarial training with only 10% of the weights left.

C.2 More Real Data Experiments

We first present the detailed experimental configurations and then provide the experimental results omited in the main text due to the space limitation.

Experimental Configuration We conduct experiments on standard datasets, including MNIST (LeCun et al., 1998), CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009). All experiments are performed in JAX (Bradbury et al., 2018), together with the neural-tangent library (Novak et al., 2020).

For the neural networks, in this paper, we evaluate our proposed method on two networks: MLP, 6-layer CNN and VGG16. The MLP is comprised of two hidden layers containing 300 and 100 units with rectified linear unit(ReLUs) followed with a linear output layer. The CNN has two convolutional layers with 32 and 64 channels, respectively. Each convoluational layer is followed by a max-pooling layer and the final two layers are fully connected with 1024 and 10/100 hidden nodes, respectively. For the experiments on VGG16, we explore the possibility of scaling up our method on large-size modern neural networks by using the technique such as sampling on the MTK matrix.

We evaluate the performance of our method on different density levels. To be precise: 1) For the experiments on MNIST with MLP, we vary the density level in $\{0.018, 0.036, 0.087, 0.169, 0.513\}$; For the experiments with CNN, the density level is set to be $[0.05, 0.1, 0.2, 0.3, \ldots, 0.9]$; 3) Since MNIST is much easier to be classified compared with CIFAR10 and CIFAR100, we also evaluate the performance of our method on MNIST with CNN at the density levels $\{0.01, 0.02, \ldots, 0.05\}$. Each experiment is comprised of two phases. For the experiments on MLP and CNN, in phase one, we run Algorithm 1 for 20 epochs to find the adversarial winning ticket. The

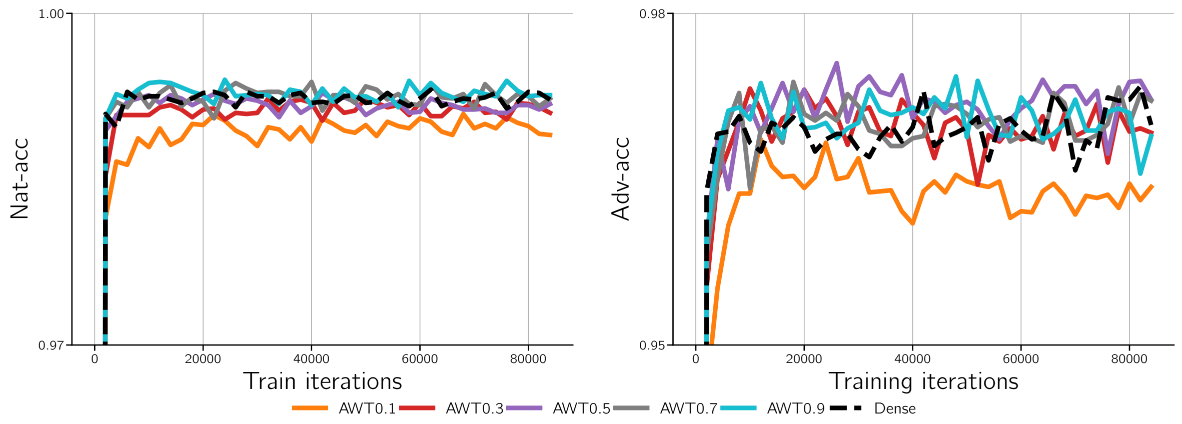


Figure 8: Test accuracy on natural and adversarial examples of CNN trained on MNIST. The density varies in $\{0.1, 0.3, 0.5, 0.7, 0.9\}$.

weight γ of the kernel distance in Eqn.(10) is choosen to be $1e-3$. In phase two, we adversarially train the winning ticket from the original initialization with the cross entropy loss for 100 epochs by using L_∞-PGD attack. The ϵ in PGD attack in both phase one and two is set to be 0.3 and 8/255 in the experiments on MNIST and CIFAR-10/100, respectively. In both of these two phases, we adopt adopt Adam (Kingma and Ba, 2014) to solve the corresponding optimization problem. The learning rate is set to be $5e-4$ and $1e-3$ in phase one and phase two, respectively. The batch size is 64. For the experiments on VGG16, we run phase one for 10 epochs and phase two for 20 epochs. Other settings such as ϵ are the same as the experiments on MLP and CNN.

Experiments in Introduction For the experimental results given in the introduction section, we adopt the above MLP network and do nature training. We compare the dynamic preserving abilities of NTT and DNS in pruning with the pruning rate being 0.02. NTT prunes the network at initialization, while DNS prunes the network during training. In both NTT and DNS, we prune the network in 20 epochs with batch size being 64. And then we fine tune the obtained sparse network for 50 epochs.

An Ablation Study To show whether the training dynamics is preserved, instead of only looking at the kernel distance and target distance, we adopt a technique named network grafting (Gu et al., 2020) to verify whether dynamics are preserved. The idea is if two networks have same dynamics, then one can be grafted/connected onto the other at each layer at any epoch of training without significant error increase. We give the result on MNIST with CNN in Figure 9, where the error increases are very small, especially when the densities are low. This verifies the claim.

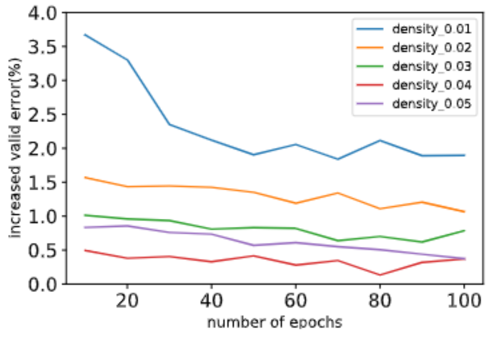


Figure 9: Graft results.

More Results Figure 7(a) and (b) are the statistics of AWT, i.e, the kernel and target distances, in the procedure of searching masks on MNIST with cnn over density levels of $\{0.05, 0.1, 0.2, 0.3, \ldots, 0.9\}$. Figure 7(c) is the adversarial training and test accuracy curves in the training process (phase two). To make the curves not too crowded, we omit the curves at the density levels of $\{0.2, 0.4, 0.6, 0.8, 0.9\}$. We can see that the target and kernel distances can decrease quickly in phase one and training dynamic of the winning ticket in phase two would become closer to the dense network when the density level increases. This is consistent with our theoretical analysis.

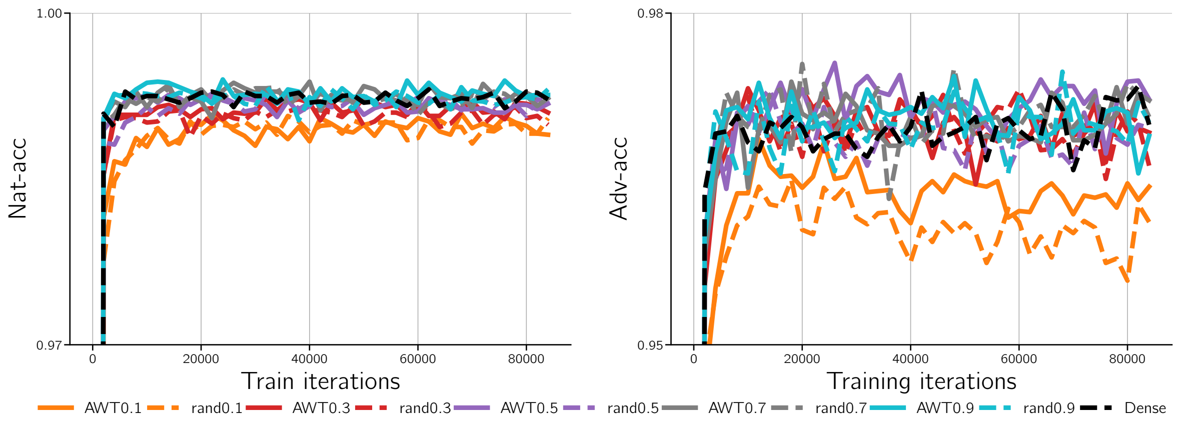


Figure 10: Natural and adversarial test accuracy of the models trained from AWT and random structure on MNIST with CNN. The density varies in $\{0.1, 0.3, 0.5, 0.7, 0.9\}$.

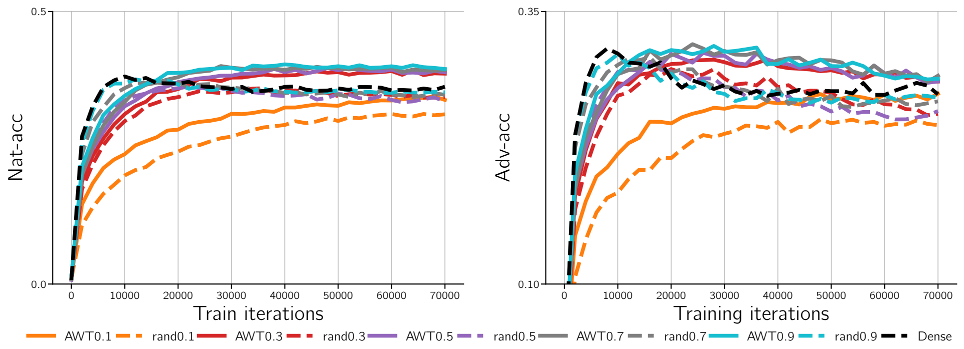


Figure 11: Natural and adversarial test accuracy of the models trained from AWT and random structure on CIFAR100 with CNN. The density varies in $\{0.1, 0.3, 0.5, 0.7, 0.9\}$.

Figure 8 presents the natural and adversarial testing accuracy of CNN trained on MNIST with the density level varies in $\{0.1, 0.2, \ldots, 0.9\}$. We can see that when the density level is larger than 0.2, the accuracy is very close to the dense mode. The reason could be that when the density level is larger than 0.2, the model begins to be overparameterized. This can also be seen in Figure 10. That is when the density level is larger than 0.2, there is even no significant difference between the winning ticket an the random structure. That's why we give the results with the density level varying in $\{0.01, 0.02, \ldots, 0.05\}$ in the main text.

Figure 11 shows the performance of the models trained on CIFAR100 from our winning ticket and the random structure. We can see that after training, our winning ticket has significantly higher natural and adversarial test accuracy than that of the random structure. In this experiment, all the models cannot achieve the comparable test accuracy on natural examples as ResNet18 reported in the existing studies. The reason is that our model is a 6-layer CNN, whose capacity is much smaller than ResNet18.