The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)

Exploring the Vulnerability of Deep Neural Networks: A Study of Parameter Corruption

Xu Sun,[1,2]* Zhiyuan Zhang,[1]* Xuancheng Ren,[1] Ruixuan Luo,[2] Liangyou Li[3]
[1]MOE Key Laboratory of Computational Linguistic, School of EECS, Peking University
[2]Center for Data Science, Peking University
[3]Huawei Noah's Ark Lab
{xusun, zzy1210,renxc,luoruixuan97}@pku.edu.cn, liliangyou@huawei.com

Abstract

We argue that the vulnerability of model parameters is of crucial value to the study of model robustness and generalization but little research has been devoted to understanding this matter. In this work, we propose an indicator to measure the robustness of neural network parameters by exploiting their vulnerability via parameter corruption. The proposed indicator describes the maximum loss variation in the non-trivial worst-case scenario under parameter corruption. For practical purposes, we give a gradient-based estimation, which is far more effective than random corruption trials that can hardly induce the worst accuracy degradation. Equipped with theoretical support and empirical validation, we are able to systematically investigate the robustness of different model parameters and reveal vulnerability of deep neural networks that has been rarely paid attention to before. Moreover, we can enhance the models accordingly with the proposed adversarial corruption-resistant training, which not only improves the parameter robustness but also translates into accuracy elevation.

Introduction

Despite the promising performance of Deep neural networks (DNNs), research has discovered that DNNs are vulnerable to adversarial examples, i.e., simple perturbations to input data can mislead models (Goodfellow, Shlens, and Szegedy 2015; Kurakin, Goodfellow, and Bengio 2017; Madry et al. 2018). These findings concern the vulnerability of DNNs against input data. However, the vulnerability of DNNs does not only exhibit in input data. As functions of both input data and model parameters, the parameters of neural networks are a source of vulnerability of equal importance. For neural networks deployed on electronic computers, parameter attacks can be conducted in the form of training data poisoning (Dai, Chen, and Li 2019; Chen et al. 2017; Gu et al. 2019), bit flipping (Rakin, He, and Fan 2020), compression (Arora et al. 2018) or quantization (Nagel et al. 2019; Weng et al. 2020). For neural networks deployed in physical devices, advances in hardware neural networks (Feldmann et al. 2019; Misra and Saha 2010; Abdelsalam et al. 2018; Salimi-Nezhad et al. 2019; Weber, da Silva Labres, and Cabrera 2019; Bui and Phillips 2019) also call for study

*Equal Contribution.

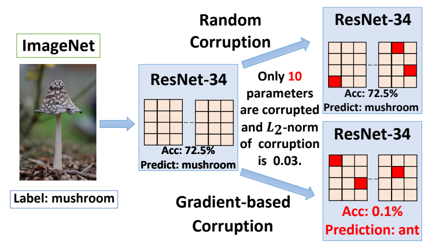


Figure 1: Parameter corruptions with ResNet-34 on ImageNet. It shows that deep neural networks are robust to random corruptions, but the accuracy can drop significantly in the worst case suggested by the gradient-based method. The accuracy is measured on the development set.

in parameter vulnerability because of hardware deterioration and background noise, which can be seen as parameter corruption. More importantly, study on parameter vulnerability can deepen our understanding of various mechanisms in neural networks, inspiring innovation in architecture design and training paradigm.

To probe the vulnerability of neural network parameters and evaluate the *parameter robustness*, we propose an indicator that measures the maximum loss change caused by small perturbations on model parameters in the non-trivial worst-cased scenario. The perturbations can be seen as artificial parameter corruptions. We give an infinitesimal gradient-based estimation of the indicator that is efficient for practical purposes compared with random corruption trials, which can hardly induce optimal loss degradation. Our theoretical and empirical results both validate the effectiveness of the proposed gradient-based method. As shown in Figure 1, model parameters are generally resistant to random corruptions but the worst outlook can be quite bleak suggested by the gradient-based corruption result.

Intuitively, the indicator shows the maximum altitude ascent within a certain distance of the current parameter on the loss surface, as illustrated conceptually in Figure 2. The traditional learning algorithms focus on obtaining lower loss, which means generally the parameters at point B are preferred. However, the local geometry of the landscape also indicates the generalization performance of the learning algorithm (Keskar et al. 2017; Chaudhari et al. 2017). The pa-

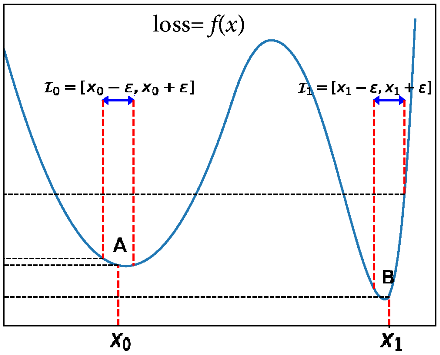


Figure 2: In this illustration of the loss function, traditional optimizer prefers B with the lower loss rather than A, because B has the lower loss. However, parameters at point B are more vulnerable to parameter corruption, as $\max_{x\in\mathcal{I}_0}(f(x)-f(x_0)) < \max_{x\in\mathcal{I}_1}(f(x)-f(x_1))$. Based on our experiments, we argue that parameters that are resistant to corruption, e.g., at point A, can embody potentially better robustness and generalization.

rameters at point B demonstrate critical vulnerability to parameter corruptions, while the parameters at point A are a better choice since larger perturbations are required to render significant loss change. It is also observed in our experiments that the parameters at point A have better generalization performance as a result of corruption-resistance.

Equipped with the proposed indicator, we are able to systematically analyze the parameter robustness and probe the vulnerability of different components in a deep neural network via observing the accuracy degradation after applying corruptions to its parameters. Furthermore, the comparisons between the gradient-based and the random corruption for estimating the indicator suggest that the neighborhood of the learned parameters on the loss surface is generally flattish except for certain steep directions. If we can push the parameters away from the steep directions, the robustness of the parameters can be improved significantly. Therefore, we propose to conduct adversarial corruption-resistant training that incorporates virtual parameter corruptions to find parameters without steep directions in the neighborhood. Experimental results show that the proposed method not only improves the parameter robustness of deep neural networks but also elevates their accuracy in application tasks.

Our main contributions are as follows:

- To understand the parameter vulnerability of deep neural networks, which is fundamentally related to model robustness and generalization, we propose an indicator that measures the maximum loss change if small perturbations are applied on parameters, i.e., parameter corruptions. The proposed gradient-based estimation is far more effective in exposing the vulnerability than random corruption trials, validated by both theoretical and empirical results.
- The indicator is used to probe the vulnerability of different kinds of parameters with diverse structural characteristics in a trained neural network. Through systematic analyses of representative architectures, we summarize divergent vulnerability of neural network parameters, especially bringing attention to normalization layers.
- To improve the robustness of the models with respect to parameters, we propose to enhance the training of deep neural networks by taking the parameter vulnerability into account and introduce the adversarial corruption-resistant training that can improve the accuracy and the generalization performance of deep neural networks.

Parameter Corruption

In this section, we introduce the problem of parameter corruption and the proposed indicator. Then, we describe the Monte-Carlo estimation and the gradient-based estimation of the indicator backed with theoretical support.

Before delving into the specifics, we first introduce our notations. Let $\mathcal{N}$ denote a neural network, $\mathbf{w} \in \mathbb{R}^k$ denote a k-dimensional subspace of its parameter space, and $\mathcal{L}(\mathbf{w};\mathcal{D})$ denote the loss function of $\mathcal{N}$ on the dataset $\mathcal{D}$, regarding to the specific parameter subspace $\mathbf{w}$. Taking a k-dimensional subspace allows a more general analysis on a specific group of parameters.

To expose the vulnerability of model parameters, we propose to adopt the approach of parameter corruption. To formally analyze its effect on neural networks and eliminate trivial corruption, we formulate the parameter corruption as a small perturbation $\mathbf{a} \in \mathbb{R}^k$ to the parameter vector $\mathbf{w}$. The corrupted parameter vector becomes $\mathbf{w}+\mathbf{a}$. The small perturbation requirement is realized as a constraint set of the parameter corruptions.

Definition 1 (Corruption Constraint)**.** *The corruption constraint is specified by the set*

$$S = \{\mathbf{a} : \|\mathbf{a}\|_p = \epsilon \textit{ and } \|\mathbf{a}\|_0 \leq n\}, \tag{1}$$

where $\|\cdot\|_0$ denotes the number of non-zero elements in a vector and $1 \leq n \leq k$ denotes the maximum number of corrupted parameters. ϵ is a small positive real number and $\|\cdot\|_p$ denotes the L_p-norm where $p \geq 1$ such that $\|\cdot\|_p$ is a valid distance in Euclidean geometry.

For example, the set $S = \{\mathbf{a} : \|\mathbf{a}\|_2 = \epsilon\}$ specifies that the corruption should be on a hypersphere with a radius of ϵ and no limit on the number of corrupted parameters.

Suppose $\Delta\mathcal{L}(\mathbf{w},\mathbf{a};\mathcal{D}) = \mathcal{L}(\mathbf{w}+\mathbf{a};\mathcal{D}) - \mathcal{L}(\mathbf{w};\mathcal{D})$ denotes the loss change. To evaluate the effect of parameter corruption, it is most reasonable to consider the worst-case scenario and thus, we propose the indicator as the maximum loss change under the corruption constraints. The optimal parameter corruption is defined accordingly.

Definition 2 (Indicator and Optimal Parameter Corruption)**.** *The indicator $\Delta_{max}\mathcal{L}(\mathbf{w},S,\mathcal{D})$ and the optimal parameter corruption $\mathbf{a}^*$ are defined as:*

$$\Delta_{max}\mathcal{L}(\mathbf{w},S,\mathcal{D}) = \max_{\mathbf{a}\in S} \Delta\mathcal{L}(\mathbf{w},\mathbf{a},\mathcal{D}), \tag{2}$$

$$\mathbf{a}^* = \arg\max_{\mathbf{a}\in S} \Delta\mathcal{L}(\mathbf{w},\mathbf{a},\mathcal{D}). \tag{3}$$

Let $\mathbf{g}$ denote ${\partial\mathcal{L}(\mathbf{w};\mathcal{D})}/{\partial\mathbf{w}}$ and $\mathbf{H}$ denote the Hessian matrix; suppose $\|\mathbf{g}\|_2 = G > 0$. Using the second-order Taylor

Algorithm 1 Random Corruption

Require: Parameter vector $\mathbf{w} \in \mathbb{R}^k$, set of corruption constraints S
1: Sample $\mathbf{r} \sim N(0, 1)$
2: Solve the random corruption $\tilde{\mathbf{a}}$ according to Eq.(5)
3: Update the parameter vector $\mathbf{w} \leftarrow \mathbf{w} + \tilde{\mathbf{a}}$

Algorithm 2 Gradient-Based Corruption

Require: Parameter vector $\mathbf{w} \in \mathbb{R}^k$, set of corruption constraints S, loss function $\mathcal{L}$ and dataset $\mathcal{D}$
1: Obtain the gradient $\mathbf{g} \leftarrow {}^{\partial \mathcal{L}(\mathbf{w};\mathcal{D})}/_{\partial \mathbf{w}}$
2: Solve the corruption $\hat{\mathbf{a}}$ in Eq.(10) with S and $\mathbf{g}$
3: Update the parameter vector $\mathbf{w} \leftarrow \mathbf{w} + \hat{\mathbf{a}}$

expansion, we estimate the loss change and the indicator:

$$\Delta\mathcal{L}(\mathbf{w}, \mathbf{a}; \mathcal{D}) = \mathbf{a}^{\mathrm{T}}\mathbf{g} + \frac{1}{2}\mathbf{a}^{\mathrm{T}}\mathbf{H}\mathbf{a} + o(\epsilon^2) = f(\mathbf{a}) + o(\epsilon). \quad (4)$$

Here, $f(\mathbf{a}) = \mathbf{a}^{\mathrm{T}}\mathbf{g}$ is a first-order estimation of $\Delta\mathcal{L}(\mathbf{w}, \mathbf{a}; \mathcal{D})$ and meanwhile the inner product of the parameter corruption $\mathbf{a}$ and the gradient $\mathbf{g}$, based on which we maximize the alternative inner product instead of initial loss function to estimate the indicator.

We provide and analyze two methods to understand the effect of parameter corruption, which estimate the value of the indicator based on constructive, artificial, theoretical parameter corruptions. Comparing the two methods, the random parameter corruption gives a Monte-Carlo estimation of the indicator and the gradient-based parameter corruption gives an infinitesimal estimation that can effectively capture the worst case. Please refer to Appendix for detailed proofs of propositions and theorems.

Random Corruption

We first analyze the random case. As we know, randomly sampling a perturbation vector $\mathbf{a}$ does not necessarily conform to the constraint set S and it is complex to generate corruption uniformly distributed in S as the generation is determined by the shape of S and is not universal enough. To eliminate the problem, we define the random parameter corruptions used in this estimation as maximizing an alternative inner product $\mathbf{a}^{\mathrm{T}}\mathbf{r}$ under the constraint, based on a random vector $\mathbf{r}$ instead of the gradient $\mathbf{g}$ to ensure the randomness.

Definition 3 (Random Parameter Corruption and Monte–Carlo Estimation)**.** *Given a randomly sampled vector $\mathbf{r} \sim N(0, 1)$, a valid random corruption $\tilde{\mathbf{a}}$ for a Monte-Carlo estimation of the indicator in the constraint set S, which has a closed-form solution, is*

$$\tilde{\mathbf{a}} = \arg\max_{\mathbf{a}\in S} \mathbf{a}^T\mathbf{r} = \epsilon\left(sgn(\mathbf{h}) \odot \frac{|\mathbf{h}|^{\frac{1}{p-1}}}{\||\mathbf{h}|^{\frac{1}{p-1}}\|_p}\right) \quad (5)$$

where $\mathbf{h} = top_n(\mathbf{r})$. The $top_n(\mathbf{v})$ function retains top-n magnitude of all $|\mathbf{v}|$ dimensions and set other dimensions to 0, $sgn(\cdot)$ denotes the signum function, $|\cdot|$ denotes the point-wise absolute function, and $(\cdot)^\alpha$ denotes the point-wise α-power function. The loss change with the random corruption is a Monte-Carlo estimation of the indicator.

The procedure to derive the random corruption vector under the Monte-Carlo estimation of the indicator is shown in Algorithm 1. The correctness and randomness of the resulting corruption vector are assured and the theoretical results are given in Appendix. Without losing generality, we discuss the characteristics of the loss change caused by random corruption under a representative corruption constraint in Theorem 1. The proof and further analysis are in Appendix.

Theorem 1 (Distribution of Random Corruption)**.** *Given the constraint set $S = \{\mathbf{a} : \|\mathbf{a}\|_2 = \epsilon\}$ and a generated random corruption $\tilde{\mathbf{a}}$ by Eq.(5), which in turn obeys a uniform distribution on $\|\tilde{\mathbf{a}}\|_2 = \epsilon$. The first-order estimation of $\Delta_{max}\mathcal{L}(\mathbf{w}, S, \mathcal{D})$ and the expectation of the loss change caused by random corruption is*

$$\Delta_{max}\mathcal{L}(\mathbf{w}, S, \mathcal{D}) = \epsilon G + o(\epsilon); \quad (6)$$

$$\mathbb{E}_{\|\tilde{\mathbf{a}}\|_2=\epsilon}[\Delta\mathcal{L}(\mathbf{w}, \tilde{\mathbf{a}}; \mathcal{D})] = O\left(\frac{tr(\boldsymbol{H})}{k}\epsilon^2\right). \quad (7)$$

Define $\eta = {}^{|\tilde{\mathbf{a}}^T\mathbf{g}|}/_{\epsilon G}$ and $\eta \in [0, 1]$, which is an estimation of ${}^{|\Delta\mathcal{L}(\mathbf{w},\tilde{\mathbf{a}},\mathcal{D})|}/_{\Delta_{max}\mathcal{L}(\mathbf{w},S,\mathcal{D})}$, then the probability density function $p_\eta(x)$ of η and the cumulative density $P(\eta \leq x)$ function of η are

$$p_\eta(x) = \frac{2\Gamma(\frac{k}{2})}{\sqrt{\pi}\Gamma(\frac{k-1}{2})}(1 - x^2)^{\frac{k-3}{2}}; \quad (8)$$

$$P(\eta \leq x) = \frac{2xF_1(\frac{1}{2}, \frac{3-k}{2}; \frac{3}{2}; x^2)}{B(\frac{k-1}{2}, \frac{1}{2})}; \quad (9)$$

where k denotes the number of corrupted parameters and $\Gamma(\cdot)$, $B(\cdot, \cdot)$ and $F_1(\cdot, \cdot; \cdot; \cdot)$ denote the gamma function, beta function and hyper-geometric function, respectively.

Theorem 1 states that the expectation of loss change of random corruption is an infinitesimal of higher order compared to $\Delta_{\max}\mathcal{L}(\mathbf{w}, S, \mathcal{D})$ when $\epsilon \to 0$. In addition, it is unlikely for multiple random trials to induce the optimal loss change corresponding to the indicator. For a deep neural network, the number of corrupted parameters can be considerably large. According to Eq.(8), η will be concentrated near 0. Thus, theoretically, it is not generally possible for the random corruption to cause substantial loss changes in this circumstance, making it ineffective in finding vulnerability.

Gradient-Based Corruption

To arrive at the optimal parameter corruption that renders a more accurate estimation of the proposed indicator, we further propose a gradient-based method based on maximizing the first-order estimation $f(\mathbf{a}) = \mathbf{a}^{\mathrm{T}}\mathbf{g}$ of the indicator.

Definition 4 (Gradient-Based Corruption and Estimation)**.** *Maximizing the first-order estimation $f(\mathbf{a}) = \mathbf{a}^T\mathbf{g}$ of the*

Dataset	ImageNet (Acc ↑)		CIFAR-10 (Acc ↑)		PTB-LM (Log PPL ↓)		PTB-Parsing (UAS ↑)		De-En (BLEU ↑)	
Base model	ResNet-34				LSTM		MLP		Transformer	
w/o corruption	72.5 $\star$		94.3 $\star$		4.25 $\star$		87.3 $\star$		35.33 $\star$	
Approach	Random	Proposed	Random	Proposed	Random	Proposed	Random	Proposed	Random	Proposed
n=k, ϵ=10^{-4}	$\star$	62.2 (-10.3)	$\star$	93.3 (-1.0)	$\star$	$\star$	$\star$	$\star$	$\star$	35.21 (-0.12)
n=k, ϵ=10^{-3}	$\star$	22.2 (-50.3)	$\star$	36.1 (-58.2)	$\star$	$\star$	$\star$	80.6 (-6.7)	$\star$	33.62 (-1.71)
n=k, ϵ=10^{-2}	30.3 (-42.2)	0.1 (-72.4)	75.1 (-19.2)	10.0 (-84.3)	$\star$	4.52 (+0.27)	79.8 (-7.5)	6.1 (-81.2)	34.82 (-0.51)	0.17 (-35.16)
n=k, ϵ=10^{-1}	0.1 (-72.4)	0.1 (-72.4)	10.0 (-84.3)	10.0 (-84.3)	4.43 (+0.18)	13.25 (+9.00)	0.0 (-87.3)	0.0 (-87.3)	0.00 (-35.33)	0.00 (-35.33)
n=k, ϵ=1	0.1 (-72.4)	0.1 (-72.4)	10.0 (-84.3)	10.0 (-84.3)	32.21 (+27.96)	48.92 (+44.67)	0.0 (-87.3)	0.0 (-87.3)	0.00 (-35.33)	0.00 (-35.33)
n=100, ϵ=10^{-2}	$\star$	$\star$	$\star$	$\star$	$\star$	$\star$	$\star$	64.6 (-22.7)	$\star$	$\star$
n=100, ϵ=10^{-1}	$\star$	67.5 (-5.0)	$\star$	$\star$	$\star$	$\star$	$\star$	11.0 (-76.3)	$\star$	31.68 (-3.65)
n=100, ϵ=1	$\star$	0.1 (-72.4)	$\star$	$\star$	$\star$	$\star$	87.1 (-0.2)	0.0 (-87.3)	35.25 (-0.08)	0.00 (-35.33)
n=100, ϵ=10^{1}	0.2 (-72.3)	0.1 (-72.4)	77.1 (-17.2)	44.8 (-49.5)	$\star$	$\star$	31.9 (-55.4)	0.0 (-87.3)	11.58 (-23.75)	0.00 (-35.33)
n=100, ϵ=10^{2}	0.1 (-72.4)	0.1 (-72.4)	10.1 (-84.2)	9.6 (-84.7)	16.90 (+12.65)	23.55 (+19.30)	0.0 (-87.3)	0.0 (-87.3)	0.00 (-35.33)	0.00 (-35.33)

Table 1: Comparisons of gradient-based corruption and random corruption under the corruption constraint ($L_{+\infty}$), with further study on the number n of parameters to be corrupted. Here, all parameters can be corrupted, that is, k stands for the total number of model parameters and $n = k$ means the number of changed parameters is not limited. ↑ means the higher value the better accuracy and ↓ means the opposite. $\star$ denotes scores close to the original score without corruption (difference less than 0.1).

indicator, the gradient-based parameter corruption $\hat{\mathbf{a}}$ *in* S *is*

$$\hat{\mathbf{a}} = \arg\max_{\mathbf{a}\in S} \mathbf{a}^T\mathbf{g} = \epsilon\left(sgn(\mathbf{h}) \odot \frac{|\mathbf{h}|^{\frac{1}{p-1}}}{\||\mathbf{h}|^{\frac{1}{p-1}}\|_p}\right); \quad (10)$$

$$f(\hat{\mathbf{a}}) = \hat{\mathbf{a}}^T\mathbf{g} = \epsilon\|\mathbf{h}\|_{\frac{p}{p-1}}; \quad (11)$$

where $\mathbf{h} = top_n(\mathbf{g})$*, other notations are used similarly to Definition 3. The resultant corruption vector leads to a gradient-based estimation of the indicator.*

The procedure of the gradient-based method is summarized in Algorithm 2. The error bound of the gradient-based estimation is described in Theorem 2. The proof and further analysis of computational complexity are in Appendix.

Theorem 2 (Error Bound of the Gradient-Based Estimation)**.** *Suppose* $\mathcal{L}(\mathbf{w};\mathcal{D})$ *is convex and* L*-smooth with respect to* $\mathbf{w}$ *in the subspace* $\{\mathbf{w}+\mathbf{a} : \mathbf{a} \in S\}$*, where* $S = \{\mathbf{a} : \|\mathbf{a}\|_p = \epsilon$ *and* $\|\mathbf{a}\|_0 \le n\}$.[1] *Suppose* $\mathbf{a}^*$ *and* $\hat{\mathbf{a}}$ *are the optimal corruption and the gradient-based corruption in* S *respectively.* $\|\mathbf{g}\|_2 = G > 0$. *It is easy to verify that* $\mathcal{L}(\mathbf{w}+\mathbf{a}^*;\mathcal{D}) \ge \mathcal{L}(\mathbf{w}+\hat{\mathbf{a}};\mathcal{D}) > \mathcal{L}(\mathbf{w};\mathcal{D})$ *. It can be proved that the loss change of the gradient-based corruption is the same order infinitesimal of that of the optimal parameter corruption:*

$$\frac{\Delta_{max}\mathcal{L}(\mathbf{w},S;\mathcal{D})}{\Delta\mathcal{L}(\mathbf{w},\hat{\mathbf{a}};\mathcal{D})} = 1 + O\left(\frac{Ln^{g(p)}\sqrt{k}\epsilon}{G}\right); \quad (12)$$

where $g(p)$ *is formulated as* $g(p) = \max\{\frac{p-4}{2p}, \frac{1-p}{p}\}$.

Theorem 2 guarantees when perturbations to model parameters are small enough, the gradient-based corruption can accurately estimate the indicator with small errors. In Eq.(12), the numerator is the proposed indicator, which is the maximum loss change caused by parameter corruption, and the denominator is the loss change with the parameter corruption generated by the gradient-based method. As we can see, when ϵ, the p-norm of the corruption vector, tends to zero, the term $O(\cdot)$ will also tend to zero such that the ratio becomes one, meaning the gradient-based method is an infinitesimal estimation of the indicator.

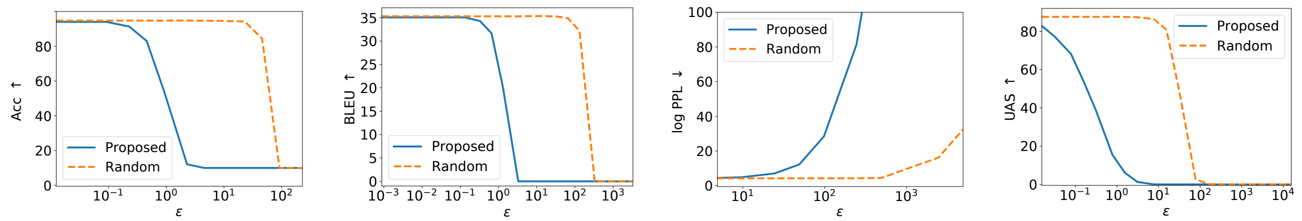
[1]Note that $\mathcal{L}$ is only required to be convex and L-smooth in a neighbourhood of $\mathbf{w}$, instead of the entire $\mathbb{R}^k$.

Experiments

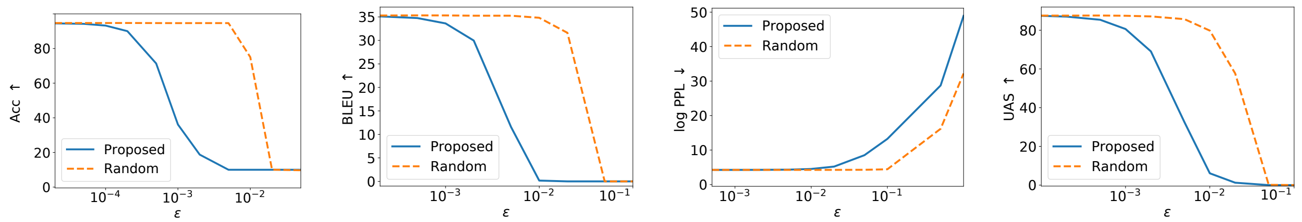
We first empirically validate the effectiveness of the proposed gradient-based corruption compared to random corruption. Then, it is applied to evaluate the robustness of neural network parameters by scanning for vulnerability and counteract parameter corruption via adversarial training.

Experimental Settings

We use four widely-used tasks including benchmark datasets in CV and NLP and use diverse neural network architecture. On the image classification task, the base model is ResNet-34 (He et al. 2016), the datasets are CIFAR-10 (Torralba, Fergus, and Freeman 2008) and ImageNet, and the evaluation metric is accuracy. On the machine translation task, the base model is Transformer provided by fairseq (Ott et al. 2019), the dataset is German-English translation dataset (De-En) Ott et al. (2019); Ranzato et al. (2016); Wiseman and Rush (2016), and the evaluation metric is BLEU score. On the language modeling task, the base model is LSTM following Merity, Keskar, and Socher (2017, 2018), the dataset is the English Penn TreeBank (PTB-LM) (Marcus, Santorini, and Marcinkiewicz 1993), and the evaluation metric is Log Perplexity (Log PPL). On the dependency parsing task, the base model is MLP following Chen and Manning (2014), the dataset is the English Penn TreeBank dependency parsing (PTB-Parsing) (Marcus, Santorini, and Marcinkiewicz 1993), and the evaluation metric is Unlabeled Attachment Score (UAS). For the detailed experimental setup, please refer to Appendix.



(a) ResNet-34 (L_2) (b) Transformer (L_2) (c) LSTM (L_2) (d) MLP (L_2)



(e) ResNet-34 ($L_{+\infty}$) (f) Transformer ($L_{+\infty}$) (g) LSTM ($L_{+\infty}$) (h) MLP ($L_{+\infty}$)

Figure 3: Results of gradient-based corruption and random corruption under the corruption constraints ($n = k$). Results of ResNet-34 are from CIFAR-10. Gradient-based corruption performs more effectively than the random corruption.

Validation of Gradient-Based Corruption

The comparative results between the gradient-based corruption and the random corruption are shown in Figure 3 and Table 1. Figure 3 shows that parameter corruption under the corruption constraint can result in substantial accuracy degradation for different sorts of neural networks and the gradient-based parameter corruption requires smaller perturbation than the random parameter corruption. The gradient-based corruption works for smaller corruption length and causes more damage at the same corruption length. To conclude, the gradient-based corruption effectively defects model parameters with minimal corruption length compared to the random corruption, thus being a viable and efficient approach to find the parameter vulnerability.

Probing the Vulnerability of DNN Parameters

Here we use the indicator to probe the Vulnerability of DNN Parameters. We use the gradient-based corruption on parameters from separated components and set n as the maximum number of the corrupted parameters. We probe the vulnerability of network parameters in terms of two natural structural characteristics of deep neural networks: the type, e.g., whether they belong to embeddings or convolutions, and the position, e.g., whether they belong to lower layers or higher layers. Due to limited space, the results of different layers in neural networks and detailed visualization of the vulnerability of different components are shown in Appendix.

Vulnerability in Terms of Parameter Types Figure 4 (a-b) show the distinguished vulnerability of different selected components in ResNet-34 and Transformer. Several observations can be drawn from the results: *(1) Normalization layers are prone to parameter corruption.* The batch normalization in ResNet-34 and the layer normalization in Transformer are most sensitive in comparison to other components in each network. It is possible that since these components adjust the data distribution, a slight change in scaling or biasing could lead to systematic disorder in the whole network. *(2) Convolution layers are more sensitive to corruption than fully-connected layers.* Since parameters in convolution, i.e., the filters, are repeatedly applied to the input feature grids, they might exert more influence than parameters in fully-connected layers that are only applied to the inputs once. *(3) Embedding and attention layers are relatively robust against parameter corruption.* It is obvious that embeddings consist of word vectors and fewer word vectors are corrupted if the corrupted number of parameters is limited, thus scarcely affecting the model. The robustness of attention is intriguing and further experimentation is required to understand its characteristics.

Vulnerability in Terms of Parameter Positions The illustration of division of different layers and results of parameter corruption on different layers are shown in Figure 4 (c-f). We can draw the following observations: *(1) Lower layers in ResNet-34 are less robust to parameter corruption.* It is generally believed that lower layers in convolutional neural networks extract basic visual patterns and are very fundamental in classification tasks (Yosinski et al. 2014), which indicates that perturbations to lower layers can fundamentally hurt the whole model. *(2) Upper layers in Transformer Decoder are less robust to parameter corruption.* From the sequence-to-sequence perspective, the encoder layers encode the sequence from shallow semantics to deep semantics and the decoder layers decode the sequence in a reversed order. It means that the higher layers are responsible for the choice of specific words and have a direct impact on the generated sequence. For Transformer Encoder, the parameter corruption exhibits inconspicuous trends.

As we can see, the proposed indicator reveals several problems that are rarely paid attention to before. Especially, the results on normalization layers should provide verifica-

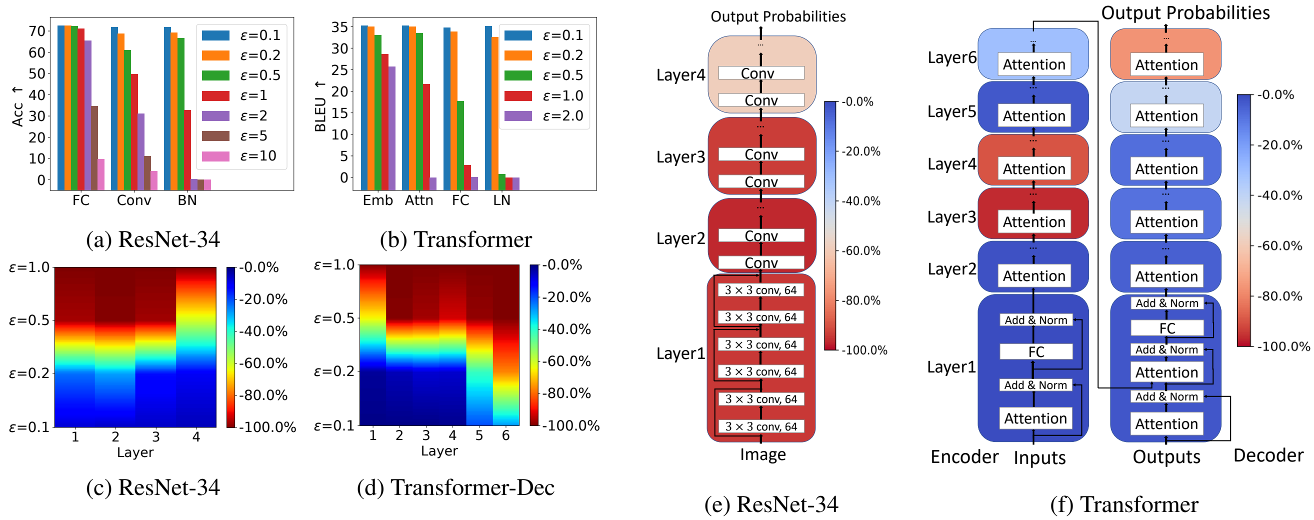


Figure 4: Results of gradient-based corruption on (a-b) different components of ResNet-34 (ImageNet) and Transformer under the corruption constraint ($n = 10$, $L_{+\infty}$-norm) and (c-f) different layers in ResNet-34 and Transformer ($n = 100$, $L_{+\infty}$-norm). Conv: convolution; Emb: embedding; FC: fully-connected; Attn: attention; BN: batch normalization; LN: layer normalization. ϵ is set to be 10 and 0.2 for ResNet-34 and Transformer, respectively. Warmer colors indicate significant accuracy degradation.

tion for the heuristic designs of future architecture.

Adversarial Corruption-Resistant Training

As shown by the probing results, the indicator can reveal interesting vulnerability of neural networks, which leads to poor robustness against parameter corruption. An important question is what we could do about the discovered vulnerability in practice, since it could be the innate characteristic of the neural network components and cannot be eliminated in design. However, if we can automatically drive the parameters from the area with steep surroundings measured by the indicator, we can obtain models that achieve natural balance on accuracy and parameter robustness.

Adversarial Corruption-Resistant Loss

To this end, we propose the adversarial corruption-resistant loss $\mathcal{L}_a$ to counteract parameter corruptions in an adversarial way. The key idea is to routinely corrupt the parameters and minimize both the induced loss change and the original loss. Intuitively, the proposed method tries to keep the parameters away from the neighborhood where there are sheer directions around, which means the parameters should be situated at the center of a flattish basin in the loss landscape.

Concretely, given batched data $\mathcal{B}$, *virtual* gradient-based corruption $\hat{\mathbf{a}}$ on parameter $\mathbf{w}$, we propose to minimize both the loss with corrupted parameter $\mathbf{w}+\hat{\mathbf{a}}$ and the original loss by minimizing a new loss $\mathcal{L}^*(\mathbf{w};\mathcal{B})$:

$$\begin{aligned}\mathcal{L}^*(\mathbf{w};\mathcal{B}) &= (1-\alpha)\mathcal{L}(\mathbf{w};\mathcal{B}) + \alpha\mathcal{L}(\mathbf{w}+\hat{\mathbf{a}};\mathcal{B}) && (13)\\ &\approx (1-\alpha)\mathcal{L}(\mathbf{w};\mathcal{B}) + \alpha[\mathcal{L}(\mathbf{w};\mathcal{B}) + f(\hat{\mathbf{a}})] && (14)\\ &= \mathcal{L}(\mathbf{w};\mathcal{B}) + \alpha f(\hat{\mathbf{a}}). && (15)\end{aligned}$$

According to Eq.(10), when $S = \{\|\mathbf{a}\|_p = \epsilon\}$, $f(\hat{\mathbf{a}})$ can be written as $f(\hat{\mathbf{a}}) = \epsilon\|\mathbf{g}\|_{p/(p-1)}$, where $\mathbf{g} = \nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w};\mathcal{B})$, which can be seen as a regularization term in the proposed adversarial corruption-resistant training. We can see that it actually serves as *gradient regularization* by simple derivation. Therefore, we define the adversarial corruption-resistant loss $\mathcal{L}_a(\mathbf{w};\mathcal{B})$ as

$$\mathcal{L}_a(\mathbf{w};\mathcal{B}) = \mathcal{L}(\mathbf{w};\mathcal{B}) + \lambda\|\mathbf{g}\|_{p/(p-1)} \qquad (16)$$

where $\mathcal{L}_a$ is equivalent to Eq.(15) when $\lambda = \alpha\epsilon$. Alizadeh et al. (2020) adopts the L_1-norm of gradients as regularization term to improve the robustness of model against quantization, which can be treated as the $L_{+\infty}$ bounded parameter corruption. In our proposed universal framework, we adopt the $L_{p/(p-1)}$-norm of gradients as regularization term to resist the L_p-norm bounded parameter corruption.

Relations to Resistance against Data Perturbations

In the common L_2 or $L_{+\infty}$ cases, our *gradient regularization* term can be written as $\|\mathbf{g}\|_{p/(p-1)} = \|\mathbf{g}\|_2$ when $p = 2$, and $\|\mathbf{g}\|_{p/(p-1)} = \|\mathbf{g}\|_1$ when $p = +\infty$.

The formulation of the gradient regularization $\mathcal{L}(\mathbf{w};\mathcal{B}) + \|\mathbf{g}\|_1$ (or $\|\mathbf{g}\|_2$) is similar to the weight regularization $\mathcal{L}(\mathbf{w};\mathcal{B}) + \|\mathbf{w}\|_1$ (or $\|\mathbf{w}\|_2$). Shaham, Yamada, and Negahban (2015) indicates that L_1 or L_2 *weight regularization* is equivalent to resist $L_{+\infty}$ and L_2 *data perturbations* respectively under some circumstances. Complementarily, we show that L_1 and L_2 *gradient regularization* is equivalent to resist $L_{+\infty}$ and L_2 *parameter corruptions*, respectively.

Experiments

We conduct experiments on the above benchmark datasets to validate that the proposed corruption-resistant training functions as designed. For ImageNet, due to its huge size, we test our corruption resistant training method on a subset of ImageNet, the Tiny-ImageNet dataset. We find that optimizing

Dataset	Tiny-ImageNet (Acc ↑)			CIFAR-10 (Acc ↑)			PTB-LM (PPL ↓)			PTB-Parsing (UAS ↑)			De-En (BLEU ↑)		
Base model	ResNet-34						LSTM			MLP			Transformer		
Approach	ϵ	Baseline	Proposed	ϵ	Baseline	Proposed	ϵ	Baseline	Proposed	ϵ	Baseline	Proposed	ϵ	Baseline	Proposed
w/o corrupt.	-	66.56	**67.06 (+0.50)**	-	94.26	**96.12 (+1.86)**	-	70.09	**68.28 (-1.81)**	-	87.26	**87.89 (+0.63)**	-	35.33	**35.69 (+0.36)**
Corrupt.	0.05	64.88	65.79	0.05	93.20	95.65	2	4.28	4.24	0.005	86.74	87.67	0.5	31.68	34.84
	0.1	60.70	63.47	0.1	89.84	94.59	5	4.43	4.24	0.01	85.32	87.26	0.75	22.13	33.58
	0.2	41.79	52.51	0.2	71.44	87.92	10	4.94	4.24	0.05	72.32	82.06	1	11.17	31.25
	0.5	1.07	7.92	0.5	13.77	21.42	20	6.45	4.27	0.1	61.80	73.19	1.25	3.47	24.47
	1	0.55	1.26	1	10.00	10.94	50	12.11	4.62	0.2	49.46	57.73	1.5	1.56	14.91

Table 2: Results of the proposed corruption-resistant training, which not only improves the accuracy without corruption but also enhances the robustness against corruption. In parameter corruptions ($n = k$, L_2-norm), all parameters can be corrupted.

$\mathcal{L}^*$ in Eq.(13) directly instead of adopting the gradient regularization term can further improve the accuracy on some tasks. Therefore, we sometimes adopt a variant of $\mathcal{L}_a$ by directly optimizing $\mathcal{L}^*$ in Eq.(13). Detailed experimental settings and supplemental results are reported in Appendix.

In Table 2, we can see that incorporating virtual gradient-based corruptions into adversarial training can help improve both the test accuracy and the robustness of neural networks against parameter corruption. In particular, we can see that parameters that are resistant to corruption, may entail better generalization, reflected as higher accuracy on the test set.

We also find that the accuracy of the uncorrupted neural network can often be improved substantially with small magnitude of virtual parameter corruptions. However, when the magnitude of virtual parameter corruptions grows too large, virtual parameter corruptions will harm the learning process and the accuracy the uncorrupted neural network will drop. In particular, the accuracy can be treated as a unimodal function of the magnitude of virtual parameter corruptions approximately, whose best configuration can be determined easily.

Related Work

Vulnerability of Deep Neural Networks Existing studies concerning vulnerability or robustness of neural networks mostly focus on generating adversarial examples (Goodfellow, Shlens, and Szegedy 2015) and adversarial training algorithms given adversarial examples in the input data (Zhu et al. 2019). Szegedy et al. (2014) first proposed the concept of adversarial examples and found that neural network classifiers are vulnerable to adversarial attacks on input data. Following that study, different adversarial attack algorithms (Moosavi-Dezfooli, Fawzi, and Frossard 2016; Kurakin, Goodfellow, and Bengio 2017) were developed. Another class of studies (Dai, Chen, and Li 2019; Chen et al. 2017; Gu et al. 2019; Kurita, Michel, and Neubig 2020) known as backdoor attacks injected vulnerabilities to neural networks by data poisoning, which requires access to the training process of the neural network models.

Adversarial Training Other related work on adversarial examples aimed to design adversarial defense algorithms to evaluate and improve the robustness of neural networks over adversarial examples (Carlini and Wagner 2017; Madry et al. 2018; Zhu et al. 2019). As another application of adversarial training, GAN (Goodfellow et al. 2014) has been widely used in multiple machine learning tasks, such as computer vision (Ma et al. 2017; Vondrick, Pirsiavash, and Torralba 2016), natural language processing (Yang et al. 2017; Dai et al. 2017) and time series synthesis (Donahue, McAuley, and Puckette 2018; Esteban, Hyland, and Rätsch 2017).

Changes in Neural Network Parameters Existing studies also concern the influence of noises or changes in neural network parameters by training data poisoning (Dai, Chen, and Li 2019; Chen et al. 2017; Gu et al. 2019), bit flipping (Rakin, He, and Fan 2020), compression (Arora et al. 2018) or quantization (Nagel et al. 2019; Weng et al. 2020). Lan et al. (2019) proposes the Loss Change Allocation indicator (LCA) to analyze the allocation of loss change partitioned to different parameters.

To summarize, existing related work mostly focuses on adversarial examples and its adversarial training. However, we focus on parameter corruptions of neural networks so as to find vulnerable components of models and design an adversarial corruption-resistant training algorithm to improve the parameter robustness.

Conclusion

To better understand the vulnerability of deep neural network parameters, which is not well studied before, we propose an indicator measuring the maximum loss change when a small perturbation is applied to model parameters to evaluate the robustness against parameter corruption. Intuitively, the indicator describes the steepness of the loss surface around the parameters. We show that the indicator can be efficiently estimated by a gradient-based method and random parameter corruptions can hardly induce the maximum degradation, which is validated both theoretically and empirically. In addition, we apply the proposed indicator to systematically analyze the vulnerability of different parameters in different neural networks and reveal that the normalization layers, which are important in stabilizing the data distribution in deep neural networks, are prone to parameter corruption. Furthermore, we propose an adversarial learning approach to improve the parameter robustness and show that parameters that are resistant to parameter corruption embody better robustness and accuracy.

Acknowledgements

We thank the anonymous reviewers for their constructive comments. This work is partly supported by Beijing Academy of Artificial Intelligence (BAAI).

Ethics Statement

This paper presents a study on parameter corruptions in deep neural networks. Despite the promising performance in benchmark datasets, the existing deep neural network models are not robust in real-life scenarios and run the risks of adversarial examples, backdoor attacks (Dai, Chen, and Li 2019; Chen et al. 2017; Gu et al. 2019; Kurita, Michel, and Neubig 2020), and the parameter corruption issues.

Unlike adversarial examples and backdoor attacks, parameter corruptions have drawn limited attention in the community despite its urgent need in areas such as hardware neural networks and software neural networks applied in a difficult hardware environment. Our work takes a first step towards the parameter corruptions and we are able to investigate the robustness of different model parameters and reveal vulnerability of deep neural networks. It provides fundamental guidance for applying deep neural networks in the aforementioned scenarios. Moreover, we also propose an adversarial corruption-resistant training to improve the robustness of neural networks, making such models available to many more critical applications.

On the other hand, the method used in this work to estimate the loss change could also be taken maliciously to tamper with the neural network applied in business. However, such kind of "attack" requires access to the storage of parameters, meaning that the system security would have been already breached. Still, it should be recommended that certain measures are taken to verify the parameters are not changed or check the parameters are corrupted in actual applications.

References

Abdelsalam, A. M.; Boulet, F.; Demers, G.; Langlois, J. M. P.; and Cheriet, F. 2018. An Efficient FPGA-based Overlay Inference Architecture for Fully Connected DNNs. In *2018 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, 1–6.

Alizadeh, M.; Behboodi, A.; van Baalen, M.; Louizos, C.; Blankevoort, T.; and Welling, M. 2020. Gradient ℓ_1 Regularization for Quantization Robustness. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. URL https://openreview.net/forum?id=ryxK0JBtPr.

Arora, S.; Ge, R.; Neyshabur, B.; and Zhang, Y. 2018. Stronger Generalization Bounds for Deep Nets via a Compression Approach. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, 254–263.

Bui, T. T. T.; and Phillips, B. 2019. A Scalable Network-on-Chip Based Neural Network Implementation on FPGAs. In *2019 IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF)*, 1–6.

Carlini, N.; and Wagner, D. A. 2017. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, 39–57. doi:10.1109/SP.2017.49.

Chaudhari, P.; Choromanska, A.; Soatto, S.; LeCun, Y.; Baldassi, C.; Borgs, C.; Chayes, J. T.; Sagun, L.; and Zecchina, R. 2017. Entropy-SGD: Biasing Gradient Descent Into Wide Valleys. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. URL https://openreview.net/forum?id=B1YfAfcgl.

Chen, D.; and Manning, C. D. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, 740–750.

Chen, X.; Liu, C.; Li, B.; Lu, K.; and Song, D. 2017. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. *CoRR* abs/1712.05526. URL http://arxiv.org/abs/1712.05526.

Dai, J.; Chen, C.; and Li, Y. 2019. A Backdoor Attack Against LSTM-Based Text Classification Systems. *IEEE Access* 7: 138872–138878.

Dai, Z.; Yang, Z.; Yang, F.; Cohen, W. W.; and Salakhutdinov, R. 2017. Good Semi-supervised Learning That Requires a Bad GAN. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, 6510–6520.

Donahue, C.; McAuley, J. J.; and Puckette, M. S. 2018. Synthesizing Audio with Generative Adversarial Networks. *CoRR* abs/1802.04208.

Esteban, C.; Hyland, S. L.; and Rätsch, G. 2017. Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs. *CoRR* abs/1706.02633.

Feldmann, J.; Youngblood, N.; Wright, C. D.; Bhaskaran, H.; and Pernice, W. 2019. All-optical spiking neurosynaptic networks with self-learning capabilities. *Nature* 569(7755): 208–214.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A. C.; and Bengio, Y. 2014. Generative Adversarial Networks. *CoRR* abs/1406.2661.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and Harnessing Adversarial Examples. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Gu, T.; Liu, K.; Dolan-Gavitt, B.; and Garg, S. 2019. BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. *IEEE Access* 7: 47230–47244. doi:10.1109/ACCESS.2019.2909068.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Keskar, N. S.; Mudigere, D.; Nocedal, J.; Smelyanskiy, M.; and Tang, P. T. P. 2017. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. URL https://openreview.net/forum?id=H1oyRlYgg.

Kurakin, A.; Goodfellow, I. J.; and Bengio, S. 2017. Adversarial examples in the physical world. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*.

Kurita, K.; Michel, P.; and Neubig, G. 2020. Weight Poisoning Attacks on Pre-trained Models. *CoRR* abs/2004.06660. URL https://arxiv.org/abs/2004.06660.

Lan, J.; Liu, R.; Zhou, H.; and Yosinski, J. 2019. LCA: Loss Change Allocation for Neural Network Training. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, 3614–3624.

Ma, L.; Jia, X.; Sun, Q.; Schiele, B.; Tuytelaars, T.; and Gool, L. V. 2017. Pose Guided Person Image Generation. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, 406–416.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Marcus, M. P.; Santorini, B.; and Marcinkiewicz, M. A. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics* 19(2): 313–330.

Merity, S.; Keskar, N. S.; and Socher, R. 2017. Regularizing and Optimizing LSTM Language Models. *arXiv preprint arXiv:1708.02182* .

Merity, S.; Keskar, N. S.; and Socher, R. 2018. An Analysis of Neural Language Modeling at Multiple Scales. *arXiv preprint arXiv:1803.08240* .

Misra, J.; and Saha, I. 2010. Artificial neural networks in hardware: A survey of two decades of progress. *Neurocomputing* 74(1-3): 239–255. doi:10.1016/j.neucom.2010.03.021. URL https://doi.org/10.1016/j.neucom.2010.03.021.

Moosavi-Dezfooli, S.; Fawzi, A.; and Frossard, P. 2016. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2574–2582. doi:10.1109/CVPR.2016.282.

Nagel, M.; van Baalen, M.; Blankevoort, T.; and Welling, M. 2019. Data-Free Quantization Through Weight Equalization and Bias Correction. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, 1325–1334. IEEE. doi:10.1109/ICCV.2019.00141. URL https://doi.org/10.1109/ICCV.2019.00141.

Ott, M.; Edunov, S.; Baevski, A.; Fan, A.; Gross, S.; Ng, N.; Grangier, D.; and Auli, M. 2019. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038* .

Rakin, A. S.; He, Z.; and Fan, D. 2020. TBT: Targeted Neural Network Attack With Bit Trojan. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, 13195–13204. IEEE. doi:10.1109/CVPR42600.2020.01321. URL https://doi.org/10.1109/CVPR42600.2020.01321.

Ranzato, M.; Chopra, S.; Auli, M.; and Zaremba, W. 2016. Sequence Level Training with Recurrent Neural Networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Salimi-Nezhad, N.; Ilbeigi, E.; Amiri, M.; Falotico, E.; and Laschi, C. 2019. A Digital Hardware System for Spiking Network of Tactile Afferents. *Frontiers in Neuroscience* 13.

Shaham, U.; Yamada, Y.; and Negahban, S. 2015. Understanding Adversarial Training: Increasing Local Stability of Neural Nets through Robust Optimization. *CoRR* abs/1511.05432. URL http://arxiv.org/abs/1511.05432.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I. J.; and Fergus, R. 2014. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Torralba, A.; Fergus, R.; and Freeman, W. T. 2008. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence* 30(11): 1958–1970.

Vondrick, C.; Pirsiavash, H.; and Torralba, A. 2016. Generating Videos with Scene Dynamics. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 613–621.

Weber, T. O.; da Silva Labres, D.; and Cabrera, F. L. 2019. Amplifier-based MOS Analog Neural Network Implementation and Weights Optimization. In *2019 32nd Symposium on Integrated Circuits and Systems Design (SBCCI)*, 1–6.

Weng, T.; Zhao, P.; Liu, S.; Chen, P.; Lin, X.; and Daniel, L. 2020. Towards Certificated Model Robustness Against Weight Perturbations. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, 6356–6363. AAAI Press. URL https://aaai.org/ojs/index.php/AAAI/article/view/6105.

Wiseman, S.; and Rush, A. M. 2016. Sequence-to-Sequence Learning as Beam-Search Optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, 1296–1306.

Yang, Z.; Hu, J.; Salakhutdinov, R.; and Cohen, W. W. 2017. Semi-Supervised QA with Generative Domain-Adaptive Nets. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, 1040–1050. doi:10.18653/v1/P17-1096.

Yosinski, J.; Clune, J.; Bengio, Y.; and Lipson, H. 2014. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, 3320–3328.

Zhu, C.; Cheng, Y.; Gan, Z.; Sun, S.; Goldstein, T.; and Liu, J. 2019. FreeLB: Enhanced Adversarial Training for Language Understanding. *CoRR* abs/1909.11764.