Learning Transferable Adversarial Examples via Ghost Networks

Yingwei Li\textsuperscript{1}, Song Bai\textsuperscript{2}, Yuyin Zhou\textsuperscript{1}, Cihang Xie\textsuperscript{1}, Zhishuai Zhang\textsuperscript{1} and Alan Yuille\textsuperscript{1}
\textsuperscript{1}Johns Hopkins University \textsuperscript{2}University of Oxford
yingwei.li@jhu.edu, \{songbai.site, zhuyuyiner, cihangxie306, zhshuai.zhang, alan.l.yuille\}@gmail.com

\section*{Abstract}
Recent development of adversarial attacks has proven that ensemble-based methods outperform traditional, non-ensemble ones in black-box attack. However, these methods generally require a family of diverse models, and ensembling them together afterward, both of which are computationally expensive. In this paper, we propose Ghost Networks to generate transferable adversarial examples efficiently. The critical principle of ghost networks is to apply feature-level perturbations to an existing model to potentially create a huge set of diverse models. After that, models are subsequently fused by longitudinal ensemble. Compared to traditional ensemble methods, both steps require almost no extra time and space consumption. Extensive experimental results suggest that the number of networks is essential for improving the transferability of adversarial examples, but it is less necessary to independently train different networks and ensemble them in an intensive aggregation way. Instead, our work can be used as a computationally cheap and easily applied plug-in to improve adversarial approaches both in single-model and multi-model attack, compatible with residual and non-residual networks. By reproducing the NeurIPS 2017 adversarial competition, our method outperforms the No.1 attack submission by a large margin, demonstrating its effectiveness and efficiency.

\section{Introduction}
In recent years, Convolutional Neural Networks (CNNs) have advanced performance in various vision tasks. However, it has been observed that attacking CNNs by adding human imperceptible perturbations to input images can cause networks to make incorrect predictions [Szegedy et al., 2014]. These perturbed images are termed as adversarial examples.

Two attack settings are later developed, i.e., white-box attack and black-box attack. In white-box attack, attackers can access the model [Carlini and Wagner, 2017b; Kurakin et al., 2017]. By contrast, in black-box attack, attackers cannot access the target model. A typical solution is to generate adversarial examples with strong transferability (the same in-
transferability. Chen et al. [2017]; Bhagoji et al. [2018]; Guo et al. [2018a] propose query-based methods to attack black-box model without substitute models, which require extensive information from the target model. In conclusion, acquiring and integrating information from various models to approximate the target model is the key to achieving better transferability. However, most works are inefficient and inadequate to learn adversarial examples with strong transferability.

In this paper, we propose a highly efficient alternative called Ghost Networks to address this issue. As shown in Fig. 1, the basic principle is to generate a vast number of virtual models built on a base network (a network trained from scratch). The word “virtual” means that these ghost networks are not stored or trained. Instead, they are generated by imposing erosion on certain intermediate structures of the base network on-the-fly. However, with an increasing number of models we have, a standard ensemble [Liu et al., 2017] would be problematic owing to its complexity. Accordingly, we propose Longitudinal Ensemble, a specific fusion method for ghost networks, which conducts an implicit ensemble during attack iterations. Consequently, adversarial examples can be easily generated without sacrificing computational efficiency.

To summarize, the contributions of our work are divided into three folds: 1) Our work is the first one to explore network erosion to learn transferable adversarial examples, not solely relying on multi-network ensemble. 2) We observe that the number of different networks used for ensemble (intrinsic networks) is essential for transferability. However, it is less necessary to train different models independently. Ghost networks can be a competitive alternative with extremely low complexity. 3) Ghost network is generic. Seemingly an ensemble method for multi-model attacks, it can also be applied to single-model attacks where only one trained model is accessible. Furthermore, it is also compatible with various network structures, attack methods, and adversarial settings.

Extensive experimental results demonstrate our method improves the transferability of adversarial examples, acting as a computationally cheap plug-in. In particular, by reproducing NeurIPS 2017 adversarial competition [Kurakin et al., 2018], our work outperforms the No.1 attack submission by a large margin, demonstrating its effectiveness and efficiency.

2 Backgrounds

This section introduces two iterative-based methods, Iterative Fast Gradient Sign Method (I-FGSM) [Kurakin et al., 2017] and Momentum I-FGSM (MI-FGSM) [Dong et al., 2018].

I-FGSM initializes an adversarial example $I_{adv}^0 = I$ and then iteratively updates it by

$$I_{adv}^{n+1} = \text{Clip}_f \{I_{adv}^n + \alpha \text{sign}(\nabla L(I_{adv}^n, \theta))\},$$  

where $L$ is the loss function of a network with parameter $\theta$. The clip function $\text{Clip}_f$ ensures the generated adversarial example within the $\epsilon$-ball of the original image $I$ with ground-truth $y_{true}$, $n$ is the iteration number, and $\alpha$ is the step size.

MI-FGSM integrates the momentum term into the attack process to stabilize update directions and escape from poor local maxima. At the $n$th iteration, the accumulated gradient is

$$g_{n+1} = \mu \cdot g_n + \frac{\nabla L(I_{adv}^n, y_{true}; \theta)}{||\nabla L(I_{adv}^n, y_{true}; \theta)||_1},$$  

where $\mu$ is the momentum decay factor. The sign of $g_{n+1}$ is then used to generate the adversarial example, by

$$I_{adv}^{n+1} = \text{Clip}_f \{I_{adv}^n + \alpha \text{sign}(g_{n+1})\}. \quad (3)$$

3 Ghost Networks

The goal of this work is to learn transferable adversarial examples. Given a clean image $I$, we want to find an adversarial example $I_{adv} = I + r$, which is still visually similar to $I$ after adding adversarial noise $|r|_\infty < \epsilon$ but fools the classifier.

Without additional cost, we generate a huge number of ghost networks from a single trained model for later attack by applying feature-level perturbations to non-residual and residual based networks in Sec. 3.1 and Sec. 3.2, respectively. These ghost networks are efficiently ensembled by our customized fusion method, longitudinal ensemble, see Sec. 3.3.

3.1 Dropout Erosion

Revisit Dropout. Dropout [Srivastava et al., 2014] is one of the most popular techniques in deep learning. By randomly dropping out units from the model during training phase, dropout can prevent deep neural networks from overfitting. Let $x_l$ be the activation in the $l$th layer, at the training time, the output $y_l$ after a dropout layer can be defined as

$$y_l = r_l \ast x_l, \quad r_l \sim \text{Bernoulli}(p), \quad (4)$$

where $\ast$ denotes an element-wise product and $\text{Bernoulli}(p)$ denotes the Bernoulli distribution with the probability $p$ of elements in $r_l$ being 1. At the test time, units in $x_l$ are always present, thus to keep the output $y_l$ the same as the expected output at the training time, $y_l$ is set to be $px_l$.

Perturb Dropout. Dropout provides an efficient way of approximately combining different neural network architectures and thereby prevents overfitting. Inspired by this, we propose to generate ghost networks by inserting the dropout layer. To make ghost networks as diverse as possible, we densely apply dropout to every block throughout the base network, rather than simply enable default dropout layers [Carlini and Wagner, 2017a]. Form our preliminary experiments, the latter cannot provide transferability. Therefore, diversity is not limited to high-level features but applied to all feature levels.

Let $f_l$ be the function between the $l$th and $(i + 1)$th layer, i.e., $x_{l+1} = f_l(x_l)$, then the output of $f_l$ after applying dropout erosion, i.e., $g_l(x_l)$, is

$$g_l(x_l) = f_l\left(\frac{r_l \ast x_l}{1 - \Lambda}\right), \quad r_l \sim \text{Bernoulli}(1 - \Lambda), \quad (5)$$

where $\Lambda = 1 - p$, and $p$ has the same meaning as in Eq. (4), indicating the probability that $x_l$ is preserved. To keep the expected input of $f_l(\cdot)$ consistent after erosion, the activation of $x_l$ should be divided by $1 - \Lambda$.

During the inference, the output feature after $(L-1)$th dropout layer ($L > l$) is

$$x_L = g_{L-1} \circ g_{L-2} \circ g_{L-3} \circ \cdots \circ g_1(x_1), \quad (6)$$

where $\circ$ denotes composite function, i.e., $g \circ f(x) = g(f(x))$.

By combining Eq. (5) and Eq. (6), we observe that when $\Lambda = 0$ (means $p = 1$), all elements in $r_l$ equal to 1. In this
Figure 2: An illustration of skip connection (a, Eq. (8)) and skip connection erosion (b, Eq. (9)).

In this case, we do not impose any perturbations to the base network. When $\Lambda$ gradually increases to 1 ($p$ decreases to 0), the ratio of elements dropped out is $\Lambda$. In other words, $(1 - \Lambda)$ of elements can be back-propagated. Hence, larger $\Lambda$ implies a heavier erosion on the base network. Therefore, we define $\Lambda$ to be the magnitude of erosion.

When perturbing dropout layers, the gradient in back-propagation can be written as

$$\frac{\partial x_L}{\partial x_l} = \prod_{i \leq l < L} \left( \frac{r_i}{1 - \Lambda} \right) \frac{\partial}{\partial x_l} f_l\left( \frac{r_i x_i}{1 - \Lambda} \right).$$

(7)

As shown in Eq. (7), deeper networks with larger $L$ are influenced more easily according to the product rule. Sec. 4.2 will experimentally analyze the impact of $\Lambda$.

**Generate Ghost Network.** The generation of ghost networks via perturbing dropout layer proceeds in three steps: 1) randomly sample a parameter set from the Bernoulli distribution $r = \{r_1, r_2, ..., r_l, ..., r_L\}$; 2) apply Eq. (5) to the base network with the parameter set $r$ and get the perturbed network; 3) repeat step 1) and step 2) to independently sample $r$ for $N$ times and obtain a pool of ghost networks $M = \{M_1, M_2, ..., M_N\}$ which can be used for adversarial attacks.

### 3.2 Skip Connection Erosion

**Revisit Skip Connection.** He et al. [2016a] propose skip connections in CNNs, which makes it feasible to train very deep neural networks. The residual block is defined by

$$x_{l+1} = h(x_l) + F(x_l, W_l),$$

(8)

where $x_l$ and $x_{l+1}$ are the input and output to the $l$-th residual block with the weights $W_l$. $F(\cdot)$ denotes the residual function. As suggested in He et al. [2016b], it is crucial to uses the identity skip connection, i.e., $h(x_l) = x_l$, to facilitate the residual learning process, otherwise the network may not converge to a good local minima.

**Perturb Skip Connection.** Following the principle of skip connection, we propose to perturb skip connections to generate ghost networks.

Specifically, the network weights are first learned using identity skip connections, then switched to the randomized skip connection (see Fig. 2). To this end, we apply randomized modulating scalar $\lambda_l$ to the $l$-th residual block by

$$x_{l+1} = \lambda_l x_l + F(x_l, W_l),$$

(9)

where $\lambda_l$ is drawn from the uniform distribution $U[1 - \Lambda, 1 + \Lambda]$. One may have observed several similar formulations on skip connection to improve the classification performance, e.g., the gated inference in Veit and Belongie [2018] and lesion study in Veit et al. [2016]. However, our work focuses on attacking the model with a randomized perturbation on skip connection, i.e., the model is not trained via Eq. (9).

During inference, the output after $(L - 1)$th layer is

$$x_L = \left( \prod_{i=1}^{L-1} \lambda_i x_i \right) + \sum_{i=1}^{L-1} \left( \prod_{j=i+1}^{L-1} \lambda_j \right) F(x_i, W_i).$$

(10)

The gradient in back-propagation is then written as

$$\frac{\partial x_L}{\partial x_l} = \left( \prod_{i=1}^{L-1} \lambda_i \right) + \sum_{i=1}^{L-1} \left( \prod_{j=i+1}^{L-1} \lambda_j \right) \frac{\partial F(x_i, W_i)}{\partial x_l}.$$  

(11)

Similar to the analysis in Sec. 3.1, we conclude from Eq. (10) and Eq. (11) that a larger $\Lambda$ will have a greater influence on the base network and deeper networks are easily influenced.

**Generate Ghost Network.** The generation of ghost networks via perturbing skip connections is similar to that via perturbing the dropout layer. The only difference is we need to sample a set of modulating scalars $\lambda = \{\lambda_1, \lambda_2, ..., \lambda_L\}$ from the uniform distribution for each skip connection.

### 3.3 Longitudinal Ensemble

The existing iteration-based ensemble-attack approach [Liu et al., 2017] require averaging the outputs (e.g., logits, classification probabilities, losses) of different networks. However, such a standard ensemble is too costly and inefficient for us because we can readily obtain a huge candidate pool of qualified neural models by using Ghost Networks.

To remedy this, we propose longitudinal ensemble, a specific fusion method for Ghost Networks, which constructs an implicit ensemble of the ghost networks by randomizing the perturbations during iterations of adversarial attack (e.g., I-FGSM [Kurakin et al., 2017] and MI-FGSM [Dong et al., 2018]). Suppose we have a base model $B$, which can generate a pool of networks $M = \{M_1, M_2, ..., M_N\}$, where $N$ is the model number. The critical step of longitudinal ensemble is that at the $j$-th iteration, we attack the ghost network $M_j$ only. In comparison, for each iteration, standard ensemble methods require fusing gradients of all the models in the model pool $M$, leading to high computational cost. We illustrate the difference between the standard ensemble and the longitudinal ensemble method in Fig. 3.

The longitudinal ensemble shares the same prior as Liu et al. [2017] that if an adversarial example is generated by attacking multiple networks, it is more likely to transfer to other networks. However, longitudinal ensemble method removes
4 Experiments

In this section, we give a comprehensive experimental evaluation of the proposed Ghost Networks. In order to distinguish models trained from scratch and the ghost networks we generate, we call the former one the base network or base model in this paper, it is extremely cheap to generate a huge number of ghost networks that still preserve relative low error rates. We apply dropout erosion in Sec. 3.1 to non-residual networks (Inc-v3 and Inc-v4) and skip connection erosion in Sec. 3.2 to residual networks (Res-50, Res-101, Res-152 and IncRes-v2). Fig. 4 (a) and (b) present the accuracy curves of the dropout erosion and skip connection erosion, respectively.

The classification accuracies of different models are negatively correlated to the magnitude of erosion $\Lambda$ as expected. By choosing the performance drop approximately equal to 10% as a threshold, we can determine the value of $\Lambda$ individually for each network. Specifically, in our following experiments, $\alpha$ are 0.006, 0.012, 0.22, 0.16, 0.12 and 0.08 for Inc-v3, Inc-v4, Res-50, Res-101, Res-152, and IncRes-v2 respectively unless otherwise specified. As emphasized throughout this paper, it is extremely cheap to generate a huge number of ghost networks that still preserve relative low error rates.

Model Diversity. To measure diversity, we use Res-50 as the backbone model. We denote the base Res-50 described in Sec. 4.1 as Res-50-A, and independently train two additional models with the same architecture, denoted by Res-50-B and Res-50-C. Meanwhile, we apply skip connection erosion to Res-50-A, then obtain three ghost networks denoted as Res-50S-A, Res-50S-B, and Res-50S-C, respectively.

We employ the Jensen-Shannon Divergence (JSD) as the evaluation metric for model diversity. Concretely, we compute the pairwise similarity of the output probability distribution (i.e., the predictions after softmax layer) for each pair of networks as in Huang et al. [2017]. Given any image, let $X$ and $Y$ denote the softmax outputs of two networks, then

$$JSD(X||Y) = \frac{1}{2}D(X||Z) + \frac{1}{2}D(Y||Z),$$  \hspace{1cm} (12)$$

where $Z$ is the average of $X$ and $Y$, i.e., $Z = (X + Y)/2$. $D(\cdot)$ is the Kullback-Leibler divergence.

In Fig. 5, we report the averaged JSD for all pairs of networks over the ILSVRC 2012 validation set. As can be drawn, the diversity among ghost networks is comparable or even more significant than independently trained networks.
average attack rate for black-box models. All the individual attack rate is shown in Table 1. To save space, we report the intrinsic number of models is 100. Because the descriptive power of ghost networks is inferior to base networks. However, by leveraging the longitudinal ensemble, our work achieves a much higher attack rate in most settings (Exp. S3 vs. Exp. S1). This observation firmly demonstrates the effectiveness of ghost networks in learning transferable adversarial examples. It should be mentioned that the computational cost of Exp. S3 almost remains the same as Exp. S1 for two reasons. First, the 10 ghost networks used in Exp. S3 are not trained but eroded from the base model and used on-the-fly. Second, multiple ghost networks are fused via the longitudinal ensemble, instead of the standard ensemble method in Liu et al. [2017].

The proposed ghost networks can also be fused via the standard ensemble method, as shown in Exp. S4. In this case, we achieve a higher attack rate at the sacrifice of computational efficiency. This observation inspires us to combine the standard ensemble and the longitudinal ensemble as shown in Exp. S5. As we can see, Exp. S5 consistently beats all the compared methods in all the black-box settings. Of course, Exp. S5 is as computational expensive as Exp. S4. However, the additional computational overhead stems from the standard ensemble rather than longitudinal ensemble.

Note that in all the experiments presented in Table 1, we use only one individual base model. Even in the case of Exp. S3, all the to-be-fused models are ghost networks. However, the generated ghost networks are never stored or trained, meaning no extra space complexity. Therefore, the benefit of ghost networks is obvious. Especially when comparing Exp. S5 and Exp. S1, ghost networks can achieve a substantial improvement in black-box attack.

Based on the experimental results above, we arrive at a similar conclusion as Liu et al. [2017]: the number of intrinsic models is essential to improve the transferability of adversarial examples. However, a different conclusion is that it
is less necessary to train different models independently. Instead, ghost networks is a computationally cheap alternative enabling good performance. When the number of intrinsic models increases, the attack rate will increase. We will further exploit this phenomenon in multi-model attack.

In Fig. 6, we select two base models, i.e., Res-50, and Inc-v3, to attack and present their performances when testing on all the 9 base models. It is easy to observe the improvement of transferability by adopting ghost networks.

### 4.4 Multi-model Attack

In this subsection, we evaluate ghost networks in multi-model setting, where attackers have access to multiple base models.

**Same Architecture and Different Parameters**

We firstly evaluate a simple setting of multi-model attack, where base models share the same network architecture but have different weights. The same three Res-50 models as in Sec. 4.2 are used. The settings of 6 experiments are shown in Table 2. Besides a new parameter #B (the number of trained-from-scratch models), others are the same as the single model attack setting. When #B is 1, we will use Res-50-A as the only one base model, and settings are the same as Sec. 4.3. When #B is 3, #S is always 3, and each branch of the standard ensemble will be assigned to a different base model. In Exp. M4 and Exp. M6, the ghost network(s) in each standard ensemble branch will be generated by the base model assigned to that branch.

The adversarial examples generated by each method are used to test on all the 9 models. We report the average attack rates in Table 2. It is easy to understand that Exp. M2 performs better than Exp. M1, Exp. M3, and Exp. M4 as it has three independently trained models. However, by comparing Exp. M5 with Exp. M2, we observe a significant improvement of attack rate. Although Exp. M5 only has 1 base model and Exp. M2 has 3, Exp. M5 actually fuses 30 intrinsic models. Such a result further supports our previous claim that the number of intrinsic models is essential, but it is less necessary to obtain them by training from scratch independently. Similarly, Exp. M6 yields the best performance as it has 3 independently trained models and 30 intrinsic models.

**Different Architectures**

Besides the baseline comparison above, we then evaluate ghost networks in the multi-model setting following Liu et al. [2017]. We attack an ensemble of 5 out of 6 normally-trained models in this experiment, then test the hold-out network (black-box setting). We also attack an ensemble of 6 normally-trained models and test on the 3 adversarially-trained networks to evaluate the transferability of the generated adversarial examples in black-box attack.

The results are summarized in Table 3, the performances in black-box attack are significantly improved. When testing on the three adversarially-trained networks, the improvement is more notable. These results further testify the ability of ghost networks to learn transferable adversarial examples.

### 4.5 NeurIPS 2017 Adversarial Challenge

Finally, we evaluate our method in a benchmark test of the NeurIPS 2017 Adversarial Challenge [Kurakin et al., 2018]. For performance evaluation, we use the top-3 defense submissions (black-box models), i.e., TsAIL, iyswin and Anil Thomas, and three official baselines (white-box models), i.e., Inc-v3, IncRes-v2_ens, and Inc-v3.

Our settings are exactly the same with the No.1 attack submission [Dong et al., 2018], but apply our method to each clean trained model. The results are summarized in Table 4. Consistent with previous experiments, we observe that by applying ghost networks, the performance of the No. 1 submission can be significantly improved, especially with black-box attack. This suggests that our proposed method can generalize well to other defense mechanisms.

### 5 Conclusion

This paper focuses on learning transferable adversarial examples for adversarial attacks. We propose, for the first time, to exploit network erosion to generate a kind of virtual models called ghost networks. Ghost networks, together with the coupled longitudinal ensemble strategy, requiring almost no additional time and space consumption, is an effective tool to improve existing methods in learning transferable adversarial examples. Extensive experiments have firmly demonstrated the efficacy of ghost networks. Meanwhile, one can potentially apply erosion to residual unit by other methods or densely erode other typical layers (e.g., batch norm [Ioffe and Szegedy, 2015] and relu [Nair and Hinton, 2010]) through a neural network. We suppose these methods could improve the transferability as well, and leave these issues as future work.
References
Shumeet Baluja and Ian Fischer. Learning to attack: Adversarial transformation networks. In AAAI, 2018.
Argun Nitin Bhagoji, Warren He, Bo Li, and Dawn Song. Practical black-box attacks on deep neural networks using efficient query mechanisms. In ECCV, 2018.
Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In AISec, 2017.
Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In IEEE S&P, 2017.
Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, 2017.
Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Xiaolin Hu, Jianguo Li, and Jun Zhu. Boosting adversarial attacks with momentum. In CVPR, 2018.
Chuan Guo, Jared S Frank, and Kilian Q Weinberger. Low frequency adversarial perturbation. arXiv preprint arXiv:1809.08758, 2018.
Chuan Guo, Mayank Rana, Moustapha Cissé, and Laurens van der Maaten. Countering adversarial images using input transformations. In ICLR, 2018.
Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016.
Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In ECCV, 2016.
Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. In ICLR, 2017.
Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In ICML, 2015.
Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In ICLR Workshop, 2017.
Alexey Kurakin, Ian Goodfellow, Samy Bengio, Yinpeng Dong, Fangzhou Liao, Ming Liang, Tianyu Pang, Jun Zhu, Xiaolin Hu, Cihang Xie, et al. Adversarial attacks and defences competition. arXiv preprint arXiv:1804.00097, 2018.
Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In ICLR, 2017.
Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. Towards robust neural networks via random self-ensemble. In ECCV, 2018.
Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In ICLR, 2018.
Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In ICML, 2010.
Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. arXiv preprint arXiv:1605.07277, 2016.
Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Belongie. Generative adversarial perturbations. In CVPR, 2017.
Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. JMLR, 15(1):1929–1958, 2014.
Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In ICLR, 2014.
Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Intriguing properties of neural networks. In ICLR, 2018.
Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In ECCV, 2018.
Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In NIPS, 2016.
Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. In IJCAI, 2018.
Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. In ICLR, 2018.
Cihang Xie, Zhishuai Zhang, Jianyu Wang, Yuyin Zhou, Zhou Ren, and Alan Yuille. Improving transferability of adversarial examples with input diversity. arXiv preprint arXiv:1803.06978, 2018.
Wen Zhou, Xin Hou, Yongjun Chen, Mengyun Tang, Xiangqi Huang, Xiang Gan, and Yong Yang. Transferable adversarial perturbations. In ECCV, 2018.