Using FGSM Targeted Attack to Improve the Transferability of Adversarial Example

Jin Xu, Zhendong Cai, Wei Shen
School of Software Engineering, Tongji University
Shanghai, China
e-mail: slhjin@tongji.edu.cn, 1650527@tongji.edu.cn, weishen@tongji.edu.cn

***Abstract*—At present, many people pay attention to the safety problems of artificial intelligence, and the emergence of adversarial examples is one of these problems. The adversarial examples can be used to attack a neural network classification model to make its classification wrong. It is an important method to improve the attack effect of adversarial examples by improving the transferability of adversarial examples and enabling them to attack multiple different neural network classification models at the same time. When we use FGSM algorithm to attack a model, first, we set ∈ a medium magnitude value, and then use targeted attack, which can improve the transferability of the adversarial examples generated by this algorithm. We can use FGSM algorithm to carry out white-box attack on the neural network model. The first step is to set a fixed value. Then, because FGSM algorithm is non-targeted attack, when adding a termination condition of iterative attack, we can use FGSM algorithm to carry out targeted attack, so that the generated adversarial examples can be identified as specific tags by the neural network model. We found that the transferability of the adversarial examples generated in this way is improved and these adversarial examples can attack many different neural network models.**

Keywords-adversarial examples; FGSM algorithm; transferability

I. INTRODUCTION

It is important to improve the transferability [1], [2] of adversarial examples [3]. If we are able to generate adversarial examples with good transportability through a network that we know its internal structure and parameters, then we can use these adversarial examples to attack a neural network model [4] that we does not know its internal structure and parameters, which is called black-box attack [5]. This is very important for adversarial examples, after all, we do not know its internal structure for most of the neural network models to be attacked. Therefore, it is of great significance to improve the transferability of adversarial examples.

FGSM algorithm [6] is introduced here. FGSM algorithm is a method proposed by Goodfellow [7] to quickly generate adversarial examples. In Goodfellow's version, FGSM algorithm is used for non-targeted attacks [8], which means that it is sufficient to generate a sample that makes neural network classification wrong, and there is no specific target value set for this sample.

However, the adversarial example generated by FGSM non- targeted attack has poor transferability and usually only has attack effect on the attacked neural network model. Once we use this generated sample to attack other neural network models, it can not work. This is the defect of FGSM non-targeted attack, and the generated adversarial example is too poor in transportability.

A big problem in FGSM algorithm is that the value of the ∈ [6] is manually set. Therefore, how to pinpoint a suitable value is a problem worth solving.

We propose our method: we set ∈ a medium magnitude value, cannot be too large or too small, and use FGSM algorithm to carry out targeted attack. In addition to FGSM algorithm, we set a target value to the sample generated, which means if we input this sample to the model, the model will output a certain wrong classification that equals our target value. To get this, we iteratively add an imperceptibly small noise to the sample until the target value reaches. And the transferability of the adversarial examples generated by this method is greatly improved.

Our experiments prove that the transferability of the adversarial examples is greatly improved by setting a medium value and using FGSM algorithm to carry out targeted attacks. The rest of this article is organized as follows. In section 2, we introduce the concept of adversarial example, the main idea of FGSM algorithm, the definitions of non-targeted attack and targeted attack, and the concept of white-box attack and black-box attack. In section 3, we extended FGSM algorithm to make it available for targeted attack and introduced the whole process of targeted attack. In section 4, we carried out experiments and analyzed the experimental results. Finally, in the fifth part, we draw a conclusion and analyze the future research prospects.

II. PRELIMINARIES

A. Adversarial Examples

Christian Szegedy et al [3] in a paper published by ICLR2014, they put forward the concept of adversarial examples which means that inputing samples formed by intentionally adding slight noise to it may cause a well-trained model to output a wrong classification with high confidence. That is to say, the adversarial examples make the depth model classification wrong by superimposing carefully constructed disturbances that are imperceptible to human beings on the original image.

978-1-7281-4784-0/19/$31.00 ©2019 IEEE

We describe the adversarial examples from a mathematical point of view. The input data is x, the classifier is f, and the corresponding classification result is expressed as $f(x)$. Suppose there is a very small disturbance , making $f(x+\epsilon) \neq f(x)$, that is, the classification result has changed, then $x+\epsilon$ is an adversarial example.

Adversarial example transferability is the property that some adversarial examples produced to mislead a specific model f can mislead other models f' – even if their architectures greatly differ [1][2]. This is distinct from knowledge transfer [9], which refers to techniques designed to transfer the generalization knowledge learned by a model f during training, and encoded in its parameters, to another model f'.

B. FGSM Algorithm

FGSM (Fast gradient sign method): This is an algorithm based on gradient to generate adversarial examples proposed by Goodfellow [6]. Assuming that the original data is x, the adversarial example x^*, the result of image recognition is y, and slight change η is superimposed on the original image, the mathematical formula is as follows:

$$x^* = x + \eta$$

The modified image is inputted into the classification model, and x is multiplied by the parameter matrix $w^\top$.

$$w^\top x^* = w^\top x + w^\top \eta$$

Its training goal is to maximize the loss function $J(x, y)$ to obtain the adversarial example x^*, where J is the loss function to measure the classification error in the classification algorithm, usually taking the cross entropy loss function. Maximizing J means that the noise-added samples no longer belong to class y, thus achieving the goal. In the whole optimization process, L_∞ constraint $||x^* - x||_\infty \leq \epsilon$ must be satisfied, that is, the error between the original sample and the adversarial example must be within a certain range.

$$x^* = x + \epsilon \cdot \text{sign}(\nabla_x J(x, y))$$

where sign () is a sign function, and the partial derivative of the loss function to x is in parentheses.

Activation function [10] also has influence on classification results. The generation process of adversarial example is to seek to make minor modifications and maximize changes to classification results through the activation function.

Goodfellow pointed out that if our change is completely consistent with the direction of change of the gradient, then the classification result will be greatly changed, $\eta = \text{sign}(w)$, and sign function can ensure the same direction as the gradient function.

The original FGSM algorithm only modifies the image once after calculating the gradient. The improved FGSM will iterate over the image many times. We use the iteration-based version.

C. Non-targeted Attack and Targeted Attack

Non-targeted attack: for a picture, an adversarial example is generated so that the label on the classifier is independent of the original label, that is, as long as the attack is successful, there is no restriction on which class the adversarial example ultimately belongs to [8].

Targeted attack: for a picture, generate an adversarial example, so that the label on the classifier is exactly the same as the label on the target, that is, not only the attack is required to be successful, but also the generated adversarial example belongs to a specific class.

D. Attack Methods

Attacks on neural network models can be divided into white-box attack and black-box attack [8]. White-box attack requires complete acquisition of the model, understanding of the structure of the model and the specific parameters of each layer, complete control of the input of the model, and fine modification of the input data. Compared with white-box attack, black-box attack treats the model as a black box, does not know the internal structure details of the model and only can control the input of the model, and the attack difficulty obviously increases. The input can be constructed based on a certain algorithm, and then the input is modified iteratively according to the feedback of the model. It can also be based on the idea of transfer learning [9], using an open source model similar to white-box attack and using the adversarial examples to carry out black-box attack.

III. Our Work

In the training process of deep learning, the loss function between the predicted value and the real value of the sample data is calculated, and then the parameters of the model are adjusted through the chain rule in the process of back propagation. The value of the loss function is continuously reduced, and the parameters of each layer of the model are iteratively calculated.

As shown in Fig.1, the basic process of generating adversarial examples can also refer to this process. The difference is that in the iterative training process, we fix the parameters of the network, regard the adversarial examples as the only parameters that need training, and adjust the adversarial examples through the back propagation process.

The adversarial examples are initialized with the original image values, the gradient is calculated by the loss function, and then the adversarial examples are updated iteratively by FGSM algorithm until the predicted value of the adversarial examples reaches the expectation. We set different values, and we set a fixed classification label for the attack target. We do not stop the iteration until the classification label is reached. In this way, we generate the adversarial example. Then we compare the transferability of the adversarial example generated by two different attack methods of non-targeted attack and targeted attack under the premise of different values.

The process for generating an adversarial example is as follows:

- Load the library files that need to be used.
- Load the pre-training model and set the tag value of the target attack.

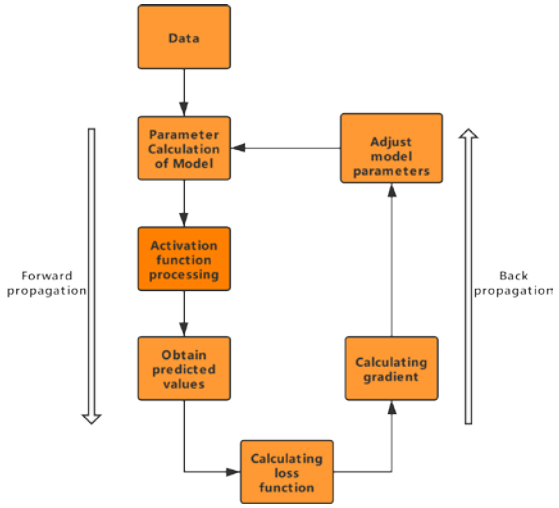


Figure 1. The training process of neural network.

The process for generating an adversarial example is as follows:

- Load the library files that need to be used.
- Load the pre-training model and set the tag value of the target attack.
- Load model inputs and outputs.
- The original image is predicted.
- The maximum number of iterations is defined, value is defined, loss function is cross entropy, and gradient is defined according to loss function and input.
- Start iteration, input target label and adversarial example, and return corresponding predicted value, gradient and loss function. Exit when the tag value of the predicted value is satisfied.

This process can be described with Fig.2.

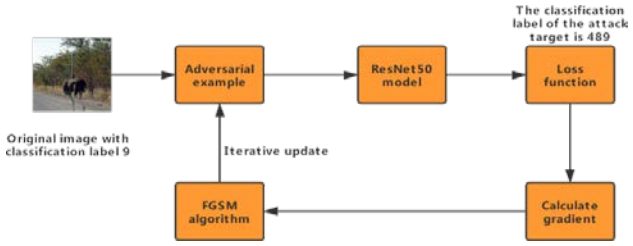


Figure 2. The process of targeted attack using FGSM algorithm.

IV. Experimental

A. Experimental Configuration

Our experiments are completed on a computer with Inter(R) i5-8250U CPU@1.60GHz with 8GB RAM.

Data set: the data set of nips2017 adversarial example attack and defense match [11], with 1,000 original images and a size of 299*299, which are all aimed at ImageNet [12] and can be classified. ImageNet has 1,000 categories, and when the model predicts a picture, it will give 1,000 categories ranking from high to low in probability. Top-1 Accuracy [13] is the accuracy of the first category matching the actual results. In the comparison of the experimental results, we only select the pictures with the same top-1 classification as the original pictures when the six classifier models are classified, and use this standard to remove the inconsistent pictures when the pictures top-1 are classified in order to eliminate the error factors generated by this.

Figure 3. The original image.

B. Comparison

We select keras [14] model with pre-training weight, the model weight is imagenet, and the attacked model is Resnet50 [15]. We assign different values to ϵ and set it to two attack modes of non-targeted attack and targeted attack respectively to generate adversarial examples.Then we use models with different structures such as Resnet 50, VGG 16, VGG 19 [16],

Inception V3 [17], Xception [18], Inception Resnetv2 [19], etc. to load imagenet weights that have been pre-trained, test the classification of the adversarial example, and then compare the transferability of this sample.

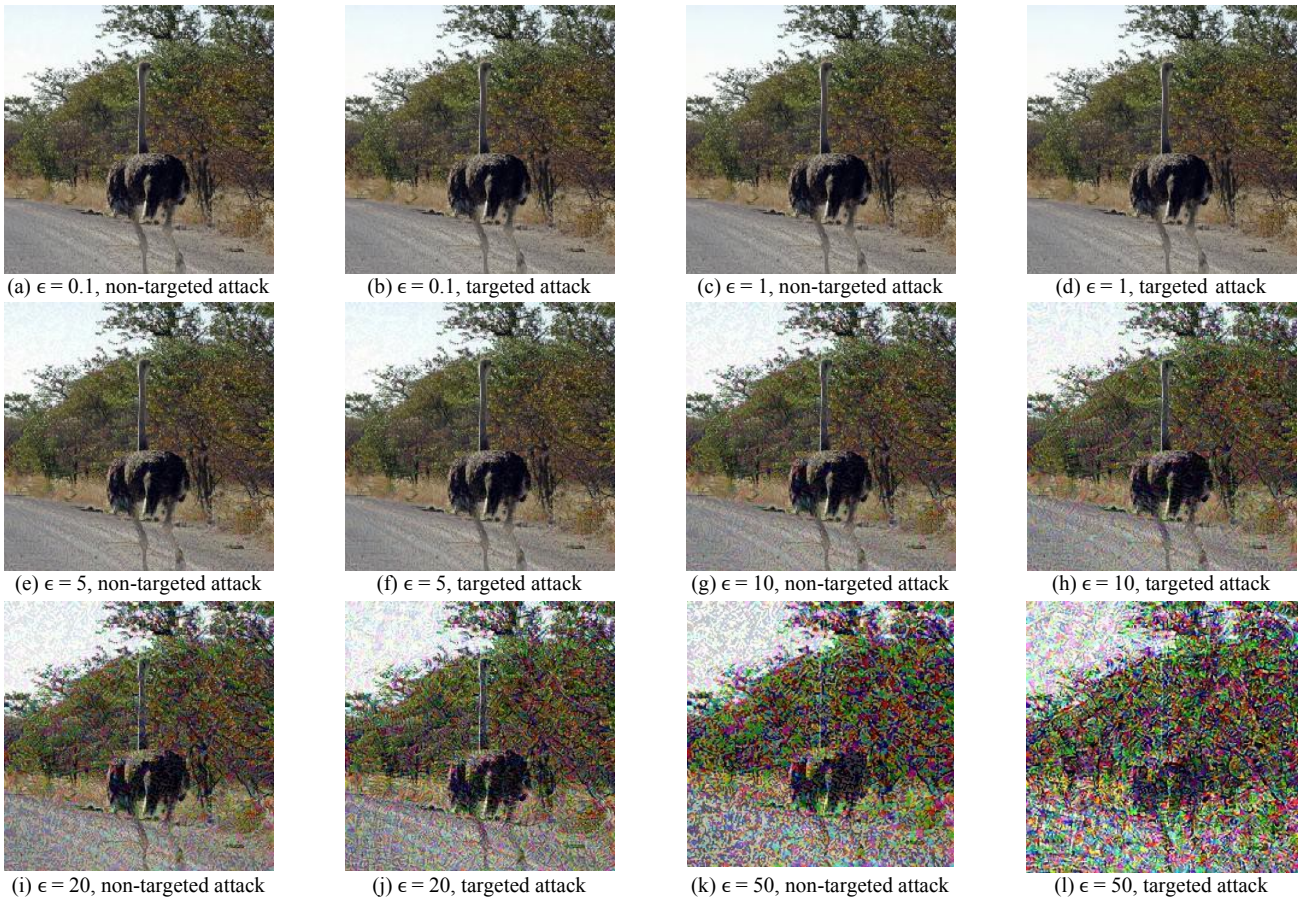
The following is the experimental result. The value in the table I is the percentage of classification errors of each classifier caused by the adversarial examples generated under this condition. The larger the value, the better the sample's transferability under this condition.

We can see that with the increasing value, whether there is non-targeted attack or there is a targeted attack, the noise of the samples generated is increasing. At the same time, the probability of misclassification for each classifier is increasing, which indicates that the sample's transportability is improving. Moreover, with the increasing ϵ value, the transportability of the adversarial examples generated by the targeted attack is higher than that of the adversarial examples generated by the non-targeted attack.

Table I shows the error rates of samples generated under non-targeted attack and targeted attack identified by different classifiers under different ϵ values. The higher the value, the greater the probability of classification errors of this classifier, that is, the higher the success rate of this sample attack. For different classifiers, if the attack success rate of the sample is high, it means that the sample is good in transportable.

TABLE I. SHOWS THE ERROR RATES OF SAMPLES, GENERATED BY NON-TARGETED ATTACK AND TARGETED ATTACK, IDENTIFIED BY DIFFERENT CLASSIFIERS UNDER DIFFERENT VALUES

	ϵ = 0.1		ϵ = 1		ϵ = 5		ϵ = 10		ϵ = 20	
	non-targeted attack	targeted attack	non-targeted attack	targeted attack	non-targeted attack	targeted attack	non-targeted attack	targeted attack	non-targeted attack	targeted attack
ResNet50	9.72 %	4.17 %	18.31 %	4.23 %	73.61 %	51.39 %	78.87 %	78.87 %	84.72 %	100 %
VGG16	6.94 %	6.94 %	9.86 %	7.04 %	23.61 %	22.22 %	50.7 %	59.15 %	75 %	84.72 %
VGG19	5.56 %	5.56 %	8.45 %	7.04 %	22.22 %	25 %	54.93 %	60.56 %	75 %	90.28 %
InceptionV3	5.56 %	5.56 %	12.68 %	11.27 %	23.61 %	19.44 %	39.44 %	39.44 %	56.94 %	66.67 %
Xception	6.94 %	6.94 %	8.45 %	5.63 %	20.83 %	22.22 %	38.03 %	33.8 %	55.56 %	61.11 %
InceptionResNetV2	5.56 %	4.17 %	5.63 %	4.23 %	13.89 %	19.44 %	29.58 %	32.39 %	50 %	52.78 %



(a) ϵ = 0.1, non-targeted attack (b) ϵ = 0.1, targeted attack (c) ϵ = 1, non-targeted attack (d) ϵ = 1, targeted attack
(e) ϵ = 5, non-targeted attack (f) ϵ = 5, targeted attack (g) ϵ = 10, non-targeted attack (h) ϵ = 10, targeted attack
(i) ϵ = 20, non-targeted attack (j) ϵ = 20, targeted attack (k) ϵ = 50, non-targeted attack (l) ϵ = 50, targeted attack

Figure 4. Samples generated by non-targeted attack and targeted attack for the same picture under different ϵ values.

It can be seen that the noise increases with the increase of value. When ϵ = 0.1, no matter there is non-targeted attack or there is a targeted attack, it fails. At this time, ResNet50 network can accurately check the label of the sample as ostrich. But the attack reduces the confidence of ResNet50 network in identifying the sample as ostrich. We find that the confidence is reduced from 99% to 80%, and the sample generated at this time also does not have transportability. Several other networks can accurately identify it as ostrich with high confidence.

When ϵ = 1, the situation changes and the non-targeted attack fails. At this time, ResNet50 network can accurately identify the label of the sample as ostrich. However the targeted attack succeeds. ResNet50 network can classify the label of the sample as chainlink fence with 15% confidence, which is obviously a wrong classification. Similarly, the samples generated at this time are not transportable.

When ϵ = 5, both non-targeted attack and targeted attack can make ResNet50 network classify the sample as bison, but the sample generated at this time is not transportable, and several other networks can identify it as ostrich. It is worth noting that the adversarial example generated by target attack reduces the confidence of other classifiers to identify it as ostrich, such as VGG19, InceptionV3, and Xception networks.

TABLE II. THE RECOGNITION RESULTS OF SAMPLES GENERATED FROM THE SAME PICTURE UNDER DIFFERENT ϵ VALUES, NON-TARGETED ATTACK AND TARGETED ATTACK ON DIFFERENT CLASSIFIERS

		ResNet50	VGG16	VGG19	InceptionV3	Xception	InceptionResNetV2
original image		ostrich 0.99971753	ostrich 0.99991906	ostrich 0.99990273	ostrich 0.987463	ostrich 0.97217089	ostrich 0.96222496
ϵ = 0.1	non-targeted	ostrich 0.80473757	ostrich 0.9992761	ostrich 0.99980241	ostrich 0.94291228	ostrich 0.992374	ostrich 0.95119387
	targeted	ostrich 0.83085716	ostrich 0.99963105	ostrich 0.9998253	ostrich 0.94221818	ostrich 0.99458343	ostrich 0.95068085
ϵ = 1	non-targeted	ostrich 0.51373941	ostrich 0.99906534	ostrich 0.99962616	ostrich 0.95655906	ostrich 0.99121279	ostrich 0.94655895
	targeted	chainlink fence 0.15653476	ostrich 0.99966478	ostrich 0.9992879	ostrich 0.94793266	ostrich 0.98965561	ostrich 0.95248675
ϵ = 5	non-targeted	bison 0.27141547	ostrich 0.99781078	ostrich 0.96695352	ostrich 0.94727683	ostrich 0.73872596	ostrich 0.93547541
	targeted	bison 0.18710431	ostrich 0.97474378	ostrich 0.78089303	ostrich 0.89195031	ostrich 0.63508177	ostrich 0.92537707
ϵ = 10	non-targeted	bison 0.14348988	ostrich 0.94952458	ostrich 0.31492546	ostrich 0.78850192	ox 0.43075097	ostrich 0.67484534
	targeted	chainlink fence 0.99363786	giant schnauzer 0.31914723	giant schnauzer 0.13973011	Bouvier des Flandres 0.66497898	ostrich 0.3187179	ostrich 0.83296829
ϵ = 20	non-targeted	American black bear 0.20400785	ostrich 0.49310943	giant schnauzer 0.0471575	bison 0.45673236	ox 0.69823444	ox 0.74456733
	targeted	jigsaw puzzle 0.34723106	jigsaw puzzle 0.66301399	jigsaw puzzle 0.13508141	ox 0.15790665	ox 0.18005101	wombat 0.10734188

When ϵ = 10, there is non-targeted attack. ResNet50 and Xception classify the image incorrectly, and although other classifiers can identify them as ostrich, their confidence levels are greatly reduced. There is a targeted attack, which makes ResNet50, VGG16, VGG19, and InceptionV3 all ouput wrong classification. At this time, the adversarial example has certain transportability. Although Xception and InceptionResNetV2 can classify it correctly, their confidence levels are also reduced.

When ϵ = 20, there is non-targeted attack. ResNet50, VGG19, InceptionV3, Xception, InceptionResNetV2 all classify the image incorrectly. Only VGG16 correctly identifies the sample as ostrich with 49% confidence, which indicates that the adversarial example at this time has certain transportability. Targeted attacks make all networks misclassified. At this time, the adversarial example has strong transferability and can be used for black box attacks.

When ϵ = 50, the sample generated by non-targeted attack and targeted attack is shown as sub-picture k and sub-picture l in Fig 4:

It can be found that at this time, the noise of the generated samples is extremely large, and it is difficult for human eyes to accurately distinguish the samples. Such samples should be abandoned samples, and cannot be called adversarial examples.

We found that the transferability of adversarial examples changes with ϵ value. When value is relatively small, e.g., ϵ value is equal to 0.1, the noise generated is relatively small. At this time, the attack effect is not good whether there is no target attack or there is a target attack. The success rate of the attack is not high, and the generated adversarial example is also poor in transferability. When ϵ value is relatively large, for example ϵ value is equal to 20, the success rate of attack is improved. At the same time, targeted attack is better than non-targeted attack in transportability. Using this, black box attack can be carried out. When the ϵ value is particularly large, too much noise is generated that completely covers the original picture. So a particularly large ϵ value cannot be used. For example, when ϵ value is equal to 50.

V. CONCLUSION

In this paper, we have found a method to improve the transferability of the adversarial examples, that is, when we use FGSM algorithm to attack a model, first, we set ϵ a medium magnitude value, and then use targeted attack, which can improve the transferability of the adversarial examples generated by this algorithm.

Firstly, we set ϵ a medium value, such as about 20. The value cannot be too small, such as 0.1. Because at this time, the success rate of attack is very low, only less than 10%. And the value cannot be too large, such as 50. Because the sample noise generated is too large that the human eye can no longer accurately distinguish the original picture. Secondly, when ϵ value is set to a medium size, the transferability of the adversarial examples generated by targeted attack is higher than that of the adversarial examples generated by non-targeted attack. Therefore, through the above two steps, we can carry out migration attack and

black-box attack on unknown models by utilizing the generated adversarial examples with good transportability.

We know that implementing targeted attack after setting ϵ a medium value can improve the transferability of the adver- sarial example. We can not only get the perfect transferability, but also ensure that the noise of the adversarial example does not exceed the limit of accurate identification by human eyes. Therefore, we need to continuously do experimental verification to achieve the best attack effect. Further, we need to explore what factors affect the transferability of the adversarial examples and whether they can be quantitatively verified and analyzed.

REFERENCES

[1] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in ma- chine learning: from phenomena to black-box attacks using adversarial samples," arXiv preprint arXiv:1605.07277, 2016.

[2] M. Naseer, S. H. Khan, S. Rahman, and F. Porikli, "Distorting neural representations to generate highly transferable adversarial examples," arXiv preprint arXiv:1811.09020, 2018.

[3] Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," arXiv preprint arXiv:1312.6199, 2013.

[4] E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[5] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adver- sarial examples and black-box attacks," in 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, 2017.

[6] J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," arXiv preprint arXiv:1412.6572, 2014.

[7] https://scholar.google.ca/citations?user=iYN86KEAAAAJ&hl=en.

[8] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," IEEE Access, vol. 6, pp. 14 410–14 430, 2018.

[9] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira, "Analysis of rep- resentations for domain adaptation," in Advances in neural information processing systems, 2007, pp. 137–144.

[10] N. Chapados, Y. Bengio, P. Vincent, J. Ghosn, C. Dugas, I. Takeuchi, and L. Meng, "Estimating car insurance premia: a case study in high-dimensional data inference," in Advances in Neural Information Processing Systems 14, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. MIT Press, 2002, pp. 1369–1376.

[11] https://nips.cc/Conferences/2017/CompetitionTrack.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in 2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009, pp. 248–255.

[13] Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural infor- mation processing systems, 2012, pp. 1097–1105.

[14] https://keras.io/.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[17] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2818–2826.

[18] Chollet, "Xception: Deep learning with depthwise separable convolu- tions," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1251–1258.

[19] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in Thirty-First AAAI Conference on Artificial Intelligence, 2017.