Adversarial Examples on Image Recognition with DNNs

Yong Wu
Johns Hopkins University
Baltimore, MD 21218 USA
ywu118@jhu.edu

Zheng Qu
Johns Hopkins University
Baltimore, MD 21218 USA
zqu2@jhu.edu

Ran Liu
Johns Hopkins University
Baltimore, MD 21218 USA
rliu28@jhu.edu

Abstract

After AlexNet[29] was propesed in 2012, Deep Neural Networks (DNNs) classifiers have been widely implemented in many industries, such as autonomous driving, mobile check deposit, etc. However, almost all well-trained machine learning classifiers have been recently found to be vulnerable to adversarial examples, which may cause classifiers to misclassify with high confidence, especially the models with linearity. The adversarial examples, which looks almost the same as the originial ones, are indistinguishable to the human eyes. In this project, we explain how the general adversarial attack works, especially on the state-of-the-art classifier with convolutional neural networks like ResNet, as well as what damages it can bring to the classifiers. We proposed a "gradient ascent with noise" approach to craft adversarial sample, experimented the approach with ImageNet sub dataset, achieving decent accuracy with small perturbation rate.

1. Introduction

Since 2012, the Deep Neural Networks (DNNs) models have been one of the most popular and cutting-edged technology. The performance of the machine learning models can even catch up with human capacities, such as making strategies, identifying, classifying, etc. The product like Face++ has reached to the accuracy of 99.2% in face recognition [27]. However, after one article named "Intriguing Properties of Neural Networks" has been published in 2014, a severe potential vulnerability of the Neural Networks has been pointed out [11]. The property of not being linear of the models trained by neural networks is taken advantages by attackers to generate adversarial examples to cause classifiers to misclassify with high confidence [11]. To make things worse, the attackers can also lead the classifiers to classify the subjects as the results they want them to be [11].

Adversarial examples are generated based on the efforts of exploiting the properties of neural network. Ian J. Goodfellow et al.(2015) demonstrated explanations of adversarial examples [4]. The reason why adversarial examples exist is because there exists a large number of points from high dimensional space mapping to the same point in low dimensional space. Besides, basic machine learning classifiers have linearity more or less, even the super complex DNNs do (linear mapping from each layer to the next layer).

There are many methods to generate adversarial examples, such as Fast Gradient Sign Method (FGSM) and Broyden-Fletcher-Goldfarb-Shanno (BFGS) [4]. The Fast Gradient Sign Method (FGSM) was first proposed by Ian J. Goodfellow to generate adversarial examples [4]. This method optimizes the fast gradient method. In this paper, we focus on studying the existing method to generate adversarial examples and discuss the current work to optimize them. We discuss the main factors that affect the performance of adversarial examples and implement it into other field. In this project, We propose a gradient ascent with noise approach to generate adversarial examples, the approach is able to achieve 92% of accuracy with 0.6% perturbation rate.

2. Related Works

In this section, we explain every aspect of adversarial examples: basic principles of generating advesarial examples; how to improve the efficiency of algorithm and current evalution of others' work. Based on these methods, we propose our own algorithm named Gradient Ascent with Noise to get better performance.

2.1. Neural Networks

Neural Networks (NN) was proposed with the idea of using algorithms to mimic human brain, even it is still quite different from how human brain works. Multilayer Perception (MLP) Neural Networks is widely used in real problems, such as what the Fig 1 illustrates, usually the layer 1 is known as input layer, layer 4 is output layer, layers between the input layer and output layer are hidden layers, the number of hidden layers could be changed from 1 to hundreds depending on the real problems. The weights are adjusted during each layer, in practice, the weights can be learned by utilizing back propagation learning algorithm, random weights are assigned at the beginning of training, then use gradient descent algorithm to tune the weights to minimize the error of misclassification. [1]

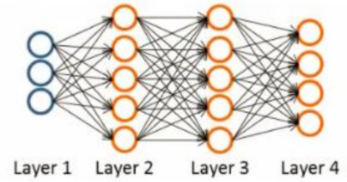


Figure 1: Neural Network[2]

2.2. Convolution Neural Networks

Large-scale image recognition suffered from the low accuracy. Alex Krizhevsky et Al.[29] proposed a new architecture using CNN (Convolutional Neural Network) to significantly improve the accuracy of estimating the content of photographs. The architecture was used to win the 2012 ILSVRC (ImageNet Large-Scale Visual Recognition Challenge). Yann LeCun et Al.'s $paper$[3] in 1998 proposed a method to use gradient-based learning algorithms to classify high-dimensional patterns like handwritten characters, the paper is widely cited. However, the accuracy of image recognition suffered many years, the Deep Convolutional Neural Networks significantly improved the accuracy, was considered as the pioneering publication.

The data used in Alex et al.'s paper[29] was provided by ILSVRC2012, the validation and test data consisted of over 15 million annotated images from a total of over 22,000 categories. The images were collected from flickr and other public search engines. This paper proposed the DCNN (Deep convolutional neural networks), also known AlexNet, it has been a popular model for image classification. To train AlexNet, which has 60 million parameters, as Figure 1 illustrates, the network was made up of 5 convolutional layers, max-pooling layers, dropout layers, and 3 fully connected layers. This paper used Rectified Linear Units (ReLU) and multiple GPU to accelerate training, used local response normalization and overlapping pooling to reduce error rate, applied data augmentation and dropout in the fully connected layer to prevent overfitting.

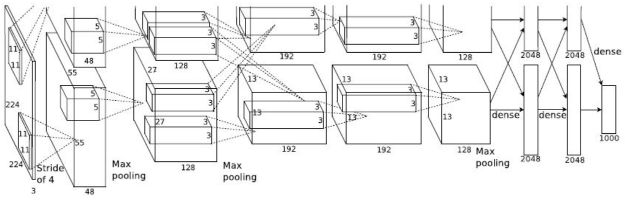


Figure 2: AlexNet Architecture[29]

AlexNet used RELU $f(x) = max(0,x)$ to train, instead of using the standard method $f(x) = tanh(x)$ or $f(x) = (I + e^{-x})^{-I}$, it can accelerate the training process. A 4-layer CNN with RELU converges six times faster than an equivalent network with tanh neurons. The paper also used the following local normalization scheme to help generalization. The normalization respectively reduced top-1 and top-5 error rates by 1.4% and 1.2%.

$$b^i_{x,y} = a^i_{x,y}/(k + \alpha \sum_{j=max(0,i-n/2)}^{min(N-1,i+n/2)} (a^j_{x,y})^2)^\beta$$ [29]

Alex et al.[29] won the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, the second-best entry achieved an error of 26.2%. The neural network developed by authors illustrated the benefits of CNNs, it was the first time a model performed so well on difficult ImageNet dataset. Before AlexNet was devised, support vector machine and other machine learning classifiers were widely used in ImageNet competition.

Nowadays, the exponentially increasing dataset, and the decreasing computation cost push the DNN to achieve surprising accuracy. Based on the creative work of AlexNet, a lot of CNNs are proposed, like VGG, GoolgeNet, ResNet, the depth of the DNN becomes deeper and deeper, achieving better and better accuracy.

2.3. Reinforcement Learning

Reinforcement learning is one of the most popular methods in machine learning. Reinforcement learning can learn how to do the work on their own, such as learning how to play "Flappy Bird", how to do calculation, etc. Instead of designing specific policies, we only need to give appropriate rewards to the algorithm based on how well it performs. Positive reward will be assigned to the algorithm when it does a good job, and vice versa.

Different from supervised learning methods, in which there has a specific set of training dataset, reinforcement learning method is trained during the training process. In reinforcement learning, interactions happen between agent and environment. Agent will observe the state of the environment, then the agent will take one action. After the environment receives the action, it will generate new state. This process continues. Generally speaking, we need to find out the value of the current state, and we can then choose the action with the largest value.

The mission of this reinforcement learning model is to achieve the reward as much as possible. All the following actions of the agent depend on the current observation, which is each state maps to one action. The process of finding this mapping relation is called policy. Therefore, the mission of the reinforcement learning is to find the best policy to achieve more reward. We can use neural network to calculate the value of the state.

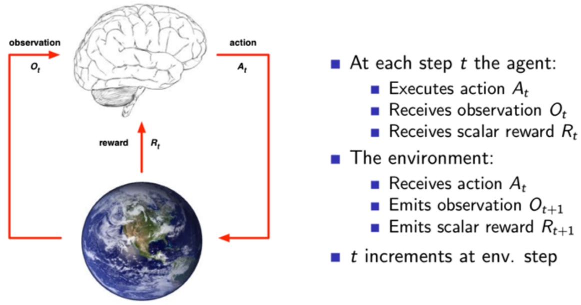

observation
O_t
action
A_t
reward
R_t
At each step t the agent:
Executes action A_t
Receives observation O_t
Receives scalar reward R_t
The environment:
Receives action A_t
Emits observation O_{t+1}
Emits scalar reward R_{t+1}
t increments at env. step

Figure 3: The process of interactions between Agents and Environment. [19]

2.4. The Explanations Of Adversarial Examples

Ian J. Goodfellow et al.(2015) demonstrated explanations about adversarial examples. In linear models, the most reason of adversarial example is the precision of an input feature. As expected, the classifier should be able to respond differently to a different input. However, if the difference of input a and input b is smaller than the precision of the features, the classifier is unable to distinguish the difference between a and b. That is, the classifier cannot response differently to input x' and x if perturbation a is smaller enough $x' = x + a$. This explanation shows in liner models one can create adversarial examples by taking advantages of this principle. [4]

The nonlinear models, such as sigmoid networks, also have linear behaviors, which results that the methods to create adversarial examples in linear models can work in the nonlinear models. E.g, LSTMs (Hochreiter et al., 1997), maxout networks (Goodfellow et al., 2013c), ReLUs (Jarrett et al.,2009) all behave linearly in some ways which makes them vulnerable to adversarial examples.

Ian J. Goodfellow introduced a method called "fast gradient sign method to generate adversarial examples: $a = \epsilon sign(i_x J(k, x, y))$.[4] In this method, x means the input feature, y is the target feature, k the parameters of the model we use and J function the cost function. One can use this method to get the perturbation a and this method also shows adversarial examples are resulted from the linear behavior as we discussed above. The linear behaviors reason can also explain the phenomenon that the example generated by one model would be misclassified by others. [4]

2.5. Fast Gradient Method

Ian J. Goodfellow et al.(2015) proposed a "fast gradient sign method[4] to generate adversarial examples and tested it in the laboratory environment. The main idea about fast gradient sign method is the perturbation formulation "$\eta = \epsilon sign(\nabla_x J(\theta, x, y))$"[4], here $J(\theta, x, y)$ means the cost function which is used to train the target model with parameter θ. The computational graph was exploited to help calculate related derivatives with back propagation, then added the sign of the derivative to the input rather than the model weights. Actually Papernot et. al[18] implemented several variations of fast gradient sign method using gradient ascent with different norms like norm 2, called fast gradient method in general. The Fast Gradient Method can successfully generate adversarial samples to deceive the state-of-the-art algorithms.

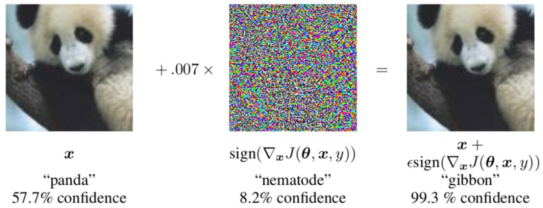

$+ .007 \times$
$=$
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

Figure 4: A demonstration of fast adversarial example generation applied to GoogLeNet. [4]

Fast Gradient Sign Method is able to generate adversarial examples efficiently and accurately, it has been widely used to generate adversarial samples, sometimes is combined with other approaches. In the real-world situation, the adversarial effectiveness could be affected by many factors, such as camera noise, transformations ability and others. For example, Luo et al., 2016; Lu et al.(2017) pointed out that when implement "fast gradient sign method to real world, the method would lose adversarial properties if transformation is smaller enough. So the basic method may not be very useful when to implementing in many real world environment.[4]

2.6. Optimized Method to Generate Adversarial Examples

Anish Athalye, Logan Engstrom and Andrew Ilyas et al.(2018) proposed a optimized method to generate adversarial examples by building 3D objects to simulate the real world and implementing their algorithm in it from different viewpoints.[5] In their work, instead of optimize a single example, they chose amount of distribution of transformation inputs generated by the adversary. Besides, they improve the basic method by redefine the distance between the outputs and inputs $\hat{x} = arg_{\hat{x}}max\mathbb{E}_{t\sim T}[logP(y|t(x))]$ $s.t.\mathbb{E}_{t\sim T}[d(t(x), t(x))] < \epsilon$.[5] In this algorithm, T means the distribution of transformation, which can be used to model by adding noise or random rotation, x' the input and t(x') the true input. The $\mathbb{E}_{t\sim T}[d(t(x), t(x))] < \epsilon$ [5]is the definition distance between inputs and outputs. Instead of simply using $x' - x$ as distance, Anish Athalye et al. redefine the distance which promised the new distance should be more effective but still within the accepted range. This algorithm is flexible that can be used as framework. When to implement in different objects, one can chose to adjust the distance metric, the inputs and the cost function to optimize it. In the paper of Anish Athalye et al., they defined $t(x) = Ax + \epsilon$ when doing the 2D cases. While in the 3D cases, they redefined the t(x) from a texture of 3D objects. Their experiments results shows the high effectiveness of their algorithm, which in the 2D cases, the algorithms can produce as high as 94.6% of adversarial examples and in the 3D cases the algorithms can produce average of 84.0% of adversarial examples.[5]

2.7. Evolutionary Algorithm to generate Adversarial Examples

Adversarial examples are mainly generated by exploiting the properties of neural network or related algorithms like back propagation. It is quite intriguing that evolutionary algorithm is able to produce adversarial examples, which can be imperceptible to human eyes. Anh Nguyen et al.[6] used evolutionary algorithm to produce "fooling examples" to cause DNNs to misclassify both on ImageNet and MNIST datasets. Evolutionary Algorithms, inspired by Darwinian evolution, are mainly used as optimization algorithms in machine learning. As Figure 4 illustrates, the algorithm introduces the same terminologies as human evolution, in each generation, the algorithm generates a lot of different individuals by adding mutations, feeding every individual to the target DNNs to get the corresponding reward, the best performing individual will be selected as the parent of the whole next generation, repeat the process until convergence.[6]

The paper[6] shows evolutionary algorithm is able to mislead the classification of deep neural network classifiers, the classifiers provide wrong labels with more than 99% confidence. In comparison with other approaches like Fast Gradient Sign Descent to generate adversarial examples, evolutionary approach could be regarded as an optimization method by means of some probabilistic mutation and evolution between generations, it works well even without knowing the architecture or structure of the target classifiers. The drawback of this approach is that it introduces several super parameters like mutation rate, and it is possible all individuals of a generation perform worse than their parent, in this case, more iterations are required to help algorithm to get converged. [6]

2.8. Black-box Attacks Against Machine Learning

Typically, to generate Adversarial Examples, one should either have a training data or have a basic knowledge of models. However, Nicolas Papernot, Patrick McDaniel and Ian Goodfellow et al. proposed a new black-box attack method against DNN classifier that do not require such knowledges, which only need to tag the inputs given by DNN and choose the inputs with tags.[8] To understand their method, first instead of attacking one DNN classifier, they assume attackers attack multiple classifier at the same time. Secondly, they assume each target classifier as an oracle, by observing the label of inputs and outputs, they get their targeted model without knowing model or training data.

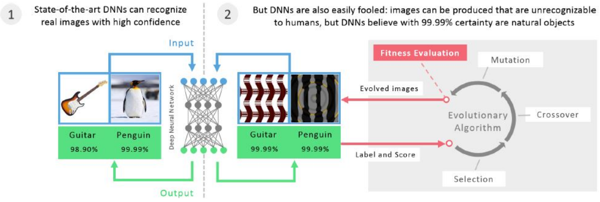

1
State-of-the-art DNNs can recognize real images with high confidence
2
But DNNs are also easily fooled: images can be produced that are unrecognizable to humans, but DNNs believe with 99.99% certainty are natural objects
Input
Deep Neural Network
Guitar
98.90%
Penguin
99.99%
Guitar
99.99%
Penguin
99.99%
Output
Fitness Evaluation
Mutation
Evolved images
Evolutionary Algorithm
Crossover
Label and Score
Selection

Figure 5: Evolutionary Algorithm architecture[6]

The threat model they used can be described as following: In order to generate as many as adversarial examples, they assume that the attacker do the black-box attack which means he do not know the architecture of models. The only things he can get is the input and output. Then the author assume that the attacker can attack multiple of targets at the same time and the outputs can be made as matrix. The results can be seem as below:[8]

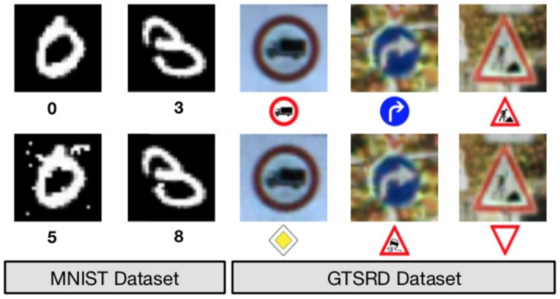
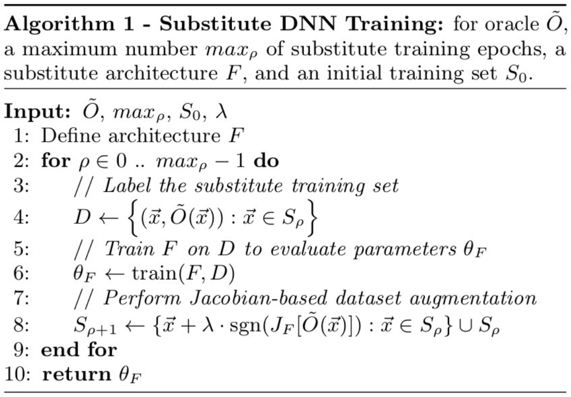


Figure 6: Adversarial samples[8]

To study Adversarial Capabilities, the author made following assumption. The $\widehat{O}(\vec{x})$ is the oracle function which can take the input x and place label on it. Also the $\widehat{O}(\vec{x})$indicates the most possible output. The equation can be concluded as below: $\widehat{O}(\vec{x}) = arg_{x \in 1..N-1} max O_j(\vec{x})$.[8] The O() function means the oracle of target classifier, $\vec{x}$ the input, $\widehat{O}(\vec{x})$ the label of input $\vec{x}$ after oracle. Their adversarial goal is to generate the adversarial examples that have minimal altered version of inputs. $\vec{x^*} = \vec{x} + argmin\{\vec{z} : \widehat{O}(\vec{x} + \vec{z} \neq \widehat{O}(\vec{x}))\} = \vec{x} + \delta_{\vec{x}}$.[8] $\vec{x^*}$ means the ideal outputs of inputs $\vec{x}$. To get $\vec{x^*}$, one should be able to find the perturbation that $\widehat{O}(\vec{x}+\vec{z}) \neq \widehat{O}(\vec{x})$.[8] To successfully implementing black-box attack on DNN classifier, the author use the method introduced above and combine with the substitute of target model. By learning the the architecture of substitute, the attacker can learn something about the relationship of inputs and outputs. Then by generating a Synthetic Dataset, one can get some information about the oracle. The Substitute DNN Training Algorithm can be concluded as following:



Algorithm 1 - Substitute DNN Training: for oracle $\tilde{O}$, a maximum number max_ρ of substitute training epochs, a substitute architecture F, and an initial training set S_0.

```
Input: $\tilde{O}$, $max_\rho$, $S_0$, $\lambda$
Define architecture $F$
for $\rho \in 0 \; .. \; max_\rho - 1$ do
    // Label the substitute training set
    $D \leftarrow \left\{ (\vec{x}, \tilde{O}(\vec{x})) : \vec{x} \in S_\rho \right\}$
    // Train $F$ on $D$ to evaluate parameters $\theta_F$
    $\theta_F \leftarrow \text{train}(F, D)$
    // Perform Jacobian-based dataset augmentation
    $S_{\rho+1} \leftarrow \{\vec{x} + \lambda \cdot \text{sgn}(J_F[\tilde{O}(\vec{x})]) : \vec{x} \in S_\rho\} \cup S_\rho$
end for
return $\theta_F$
```

Figure 7: Substitute DNN Training[8]

To illustrate this algorithm, first the attack collects a amount of the inputs and trains them through suitable architecture to get substitute F. Then repeat above steps until get better substitute DNNs F_ρ. After getting the trained substitute DNN, one can use this to get adversarial examples. They use fast gradient sign method to generate adversarial examples.They tested this algorithm in the laboratory environment and the results is promised. It shows this algorithm can generate 84.24% adversarial examples. They also tested their algorithm in some popular machine learning classifier hosted by Amazon and Google. The results show that at rates of 96.19% and 88.94 % adversarial examples successfully misclassified such models. [8]

2.9. Universal Adversarial Perturbations

Seyed-Mohsen Moosavi-Dezfooli et al.[7](2017) proposed a systematic algorithm for calculating the universal perturbation to produce adversarial examples. Given an image, it can be deemed as a vector, every element of the vector is a pixel of the image, adding a very small perturbation vector can cause the original natural image to be misclassified by state-of-the-art deep neural network classifiers with high confidence. The universal perturbation somehow explains the geometric correlation of dimension mapping, the original natural image can be deemed as a high dimensional vector, the classification can be regarded as a mapping operation from high dimensional space to low dimensional space. From the perspective of Geometry, there exist multiple different vectors in high dimensional space map to the same vector in low dimensional space.[7]

Let μ denotes distribution of image in space $\mathbb{R}^d$, $\hat{k}$ denotes the classification algorithm which classifies $x \in \mathbb{R}^d$ to label $\hat{k}(x)$. The objective of universal adversarial perturbation is to seek the universal vector $\nu \in \mathbb{R}^d$ to fool the classifier $\hat{k}$ for almost all data samples from distribution μ, that is $\hat{k}(x+\nu) \neq \hat{k}(x)$ for most $x \sim \mu$. [7]Here the perturbation magnitude is controlled by ℓ_p norm with $p \in [1, \infty)$, thus, the perturbation object is to satisfy the following two constraints.

- $\parallel \nu \parallel_p \leq \xi$, here ξ controls the magnitude of perturbation to make sure the perturbation is imperceptible to human eyes.

- $\mathbb{P}_{x \sim \mu}(\hat{k}(x+\nu) \neq \hat{k}(x)) \leq 1 - \delta$, here δ denotes the fooling rate.

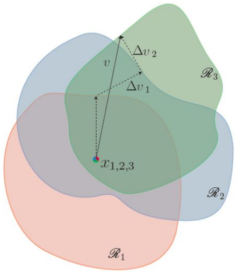


Figure 8: Universal Adversarial Perturbation[7]

Universal adversarial perturbation works by figuring out a small perturbation for a data point, aggregating perturbation vectors of a set of data points. In practice, the paper shows the data points for aggregating is much smaller than all input data set, and can be generalized well across other data points, even unseen data points, this property makes universal perturbation method is a viable approach. Universal adversarial perturbation brings very good transferability, both with respect to data set and classifiers. Another interesting thing is by using different norm, the similar process can get quite different working universal perturbation vectors. In comparison with Fast Gradient Sign Method, universal adversarial perturbation approach provides a more precise and transferable way to craft adversarial examples, a drawback is it require more computation, since universal perturbation works iteratively with a number of data points, another drawback is it needs to find out how many data points for the calculation, it requires huge computation to compute perturbation vector with all input data points. [7]

2.10. DeepFool approach to generate adversarial examples

Seyed-Mohsen Moosavi-Dezfooli et. al[9](2016) proposed an approach, named DeepFool, to generate minimal adversarial perturbation accurately and efficiently. γ denotes the minimal adversarial perturbation, $\hat{k}(x)$ denotes estimated label, robustness of $\hat{k}$ at point x is formulated as $\Delta(x,\hat{k} := min_r \parallel r \parallel_2$ subject to $\hat{k}(x+r) \neq \hat{k}(x))$. [9]Robustness of classifier is defined as $\rho_{adv}\hat(k) = \mathbb{E}_x \frac{\Delta(x,\hat{k})}{\|x\|_2}$,[9] here $\mathbb{E}_x$ means the expectation over the data distribution, obviously the work to generate minimal adversarial examples can somehow help use know which features work for the classification. For a specific multi-class classification algorithm $f : \mathbb{R}^n \to \mathbb{R}^c$,[9] n is the number of input vector, c denotes how many labels in total, $f_k(x)$ is the $f(x)$ with respect to k^{th} label or class. Suppose $f(x) = W^T x + b$, here vector is column vector. The minimal perturbation can be defined as the following. [9]

$$arg_r min \parallel r \parallel_2 \text{ s.t. } \exists k : w^T_{k(x_0)}(x_0 + r) + b_{\hat{k}(x_0)}$$ [9]

The above formulation corresponds to the computation of the distance $dist(x_0, P^c)$, here $P = \bigcap_{k=1}^{c}\{x : f_{\hat{k}(x_0)}(x) \geqslant f_k(x)\}$ denotes convex polyhedron, $\hat{l}(x_0)$ denotes the closest hyperplane of P. Minimal perturbation $r_*(x_0)$ is formulated as the following.

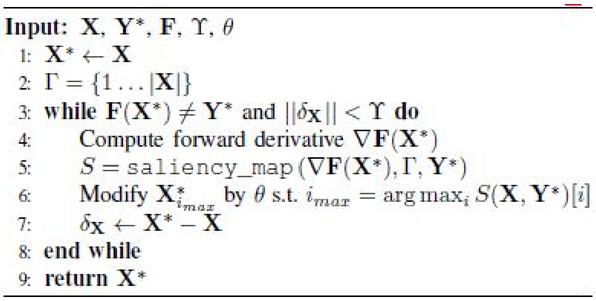
$$\hat{l}(x_0) = arg_{k \neq \hat{k}(x_0)} min \frac{|f_k(x_0) - f_{\hat{k}(x_0)}(x_0)|}{\|w_k - w_{\hat{k}(x_0)}\|2}$$

$$r_*(x_0) = \frac{|f_{\hat{l}(x_0)}(x_0) - f_{\hat{k}(x_0)}(x_0)|}{\|w_{\hat{l}(x_0)} - w_{\hat{k}(x_0)}\|_2^2}$$ [9]

DeepFool was experimented to be proven a reliable and efficient approach to generate adversarial examples, the perturbation is smaller than that generated by other methods, even Fast Gradient Sign Method, so the adversarial examples generated by DeepFool method is more imperceptible to human eyes than other approaches. Feeding more data could improve the performance of DeepFool.[9]

2.11. Saliency Maps approach

Nicolas Papernot et. al[10](2016) described a new algorithm to generate adversarial samples against feedforward deep neural networks, unlike other approaches which calculate the perturbation based on the target classifier's feedback, here the algorithm gets the adversarial samples by directly computing the mapping from input to output, and the approach can be used to get the targeted adversarial examples. The paper formalized the adversarial space against classification, and built the adversarial saliency maps. More formally, the classification $F : X \to Y$, X denotes the input vector, Y is corresponding label vector, getting the perturbation δ could be regarded as an optimization problem: $arg_{\delta_x} min \parallel \delta_x \parallel$ s.t. $F(X+\delta) = Y^*$,[10] Y^* is the desired wrong label.



```
Input: $\mathbf{X}$, $\mathbf{Y}^*$, $\mathbf{F}$, $\Upsilon$, $\theta$
X* ← X: $\mathbf{X}^* \leftarrow \mathbf{X}$
$\Gamma = \{1 \ldots |\mathbf{X}|\}$
while $\mathbf{F}(\mathbf{X}^*) \neq \mathbf{Y}^*$ and $||\delta_{\mathbf{X}}|| < \Upsilon$ do
    Compute forward derivative $\nabla\mathbf{F}(\mathbf{X}^*)$
    $S = \texttt{saliency\_map}(\nabla\mathbf{F}(\mathbf{X}^*), \Gamma, \mathbf{Y}^*)$
    Modify $\mathbf{X}^*_{i_{max}}$ by $\theta$ s.t. $i_{max} = \arg\max_i S(\mathbf{X}, \mathbf{Y}^*)[i]$
    $\delta_{\mathbf{X}} \leftarrow \mathbf{X}^* - \mathbf{X}$
end while
return $\mathbf{X}^*$
```

Figure 9: Saliency Maps Algorithm[10]

The saliency maps algorithm for generating adversarial examples is illustrated in Figure 6, Υ denotes distortion parameter, θ denotes feature variation parameter, F is feedforward deep neural networks, which takes input X, and outputs Y^*. The saliency maps algorithms mainly compute the forward derivative $\nabla F(x)$, then saliency maps algorithm constructs a saliency map, and modify input vector, repeat the process until getting the desired Y^*. The derivative is computed by DNN itself rather than related cost functions, and directly applies to the input feature vector rather than the DNN model weights. The paper presented that the saliency maps algorithms can be utilized to reliably construct targeted adversarial examples by a DNN with 97% adversarial success rate with only 4.02% modification. One drawback of the saliency maps algorithm is it requires knowledge architecture or structure of the target DNN model.[10]

2.12. Adversarial attacks on deep policies

We have shown several existing approaches to generate adversarial examples to attack DNNs, most of the approaches like Fast Gradient Sign Method have been experimented on image recognition. In fact, speech recognition, NLP applications like ChatBot, and Reinforcement Learning algorithms like deep policies are also equipping with deep neural networks. For example, ChatBot will use RNN help generate sequence, Reinforcement Learning applications like autonomous driving highly rely on DNNs for image recognition.

Jernej Kos et. al[13](2017) applied adversarial attack on deep reinforcement learning policies, the deep policies can be fooled by adversarial examples, here Fast Gradient Sign Method was used to generate the related adversarial perturbation to successfully attack the agents working on Atari Pong tasks using A3C algorithm.

2.13. Adversarial examples in the physical world

Kurakin et. al [12](2017) explored several potential security concerns caused by adversarial examples in physical world, feeding the DNNs classifier with adversarial images gained from mobile phone also can cause the classifier misclassify, it demonstrates that the adversarial attack is a big security concern in real physical world due to its attack effectiveness on machine learning classifiers. In this paper, several algorithms, such as Fast Gradient Method, have been experimented to be effectively applied to real world scenarios. Undoubtedly, there must be more physical scenarios that are vulnerable to adversarial examples. To better understand their method, they first introduce FAST METHOD, which is one of the straightest method to generate adversarial examples. This method was first introduced by Ian Goodfellow, 2014, $a = \epsilon sign(i_x J(k, x, y))$.[4]. The author then introduced a optimize of fast method. They use iterative procedure to generate adversarial examples: each time the pixel move minimal distance to its neighbourhood and clip the intermediate of each step to make sure it still looks similar as the original images. Such process can be concluded as below:$X_0^{adv} = X$, $X_{N+1}^{adv} = Clip_{X,\epsilon}\{X_N^{adv} + \alpha sign(\nabla_X J(X_N^{adv}, y_{true}))\}$. Then the author improve above methods to proposed ITERATIVE LEAST-LIKELY CLASS METHOD which do not need to increase the cost of correct class. Instead of just simply choosing the correct class, they specific the incorrect class of model should be chose, which benchmark improve the space complexity of algorithm. This process can be seen as below: $Y_{LL} = arg_\ell min\{p(y|X)\}$. So to make Y_{LL}, one should iteratively repeat $sign(\nabla_X J(X_N^{adv}, y_{true}))\}$, so the process can be seen as following:$X_0^{adv} = X$, $X_{N+1}^{adv} = Clip_{X,\epsilon}\{X_N^{adv} - \alpha sign(\nabla_X J(X_N^{adv}, y_{true}))\}$. After implementing such methods into ImageNet dataset, the results can be seen as following: [12]

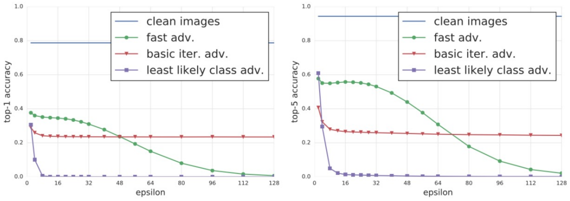


Figure 10: Top-1 and top-5 accuracy of Inception v3[12]

2.14. Learning to Protect Communications with Adversarial Neural Cryptography

Adversarial attacks, undoubtedly, cause more and more attention both in research community and information technology industry, especially considering a increasing number of deep learning applications are being launching . No one is able to pay the price if the adversarial attacks are ignored. How to enhance the security of communication of multiple neural network systems.

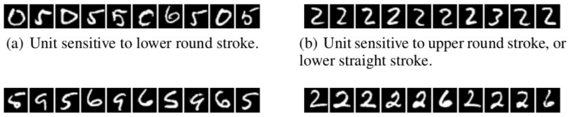
Abadi et. al[14](2016) applied adversarial training to different neural network models with a system. For simplicity, says neural network Alice and neural network Bob, in order not to be eavesdropped by Attacker between Alice and Bob, the neural networks can be trained to perform some types of encryption and decryption. The authors neither specified any encryption algorithms, nor taught the neural network how to perform encryption or decryption, just defined the confidentiality purpose in the objective function, the neural networks have been experimented to be able to project the security of communication.

2.15. Intriguing properties of neural networks

Christian Szegedy, Wojciech Zaremba and Ilya Sutskever et al. deep-explore the properties of deep neural networks.[11] One can conclude their findings as following:(1). About unit analysis, there is no distinguish difference to analyze high level units or liner combinations of high level units, which suggests that the space of high level units rather than high level units themselves contains semantic information in deep neural networks. (2). The inputs and output are not continues, so one can take advantages of this property to generate adversarial examples by create an minimal perceptible perturbation. To test their method, they use different framework on the experiment, including MNIST dataset, ImageNet dataset and 10M image samples from Youtube. For the first framework, they use hidden layers to connect the networks and use Softmax classifier to train the top of autoencoder. For the second framework, they use architecture that be described in (Krizhevsky et. al architecture). For the last one, they use unsupervised model to train the dataset.[11]

The first property is focus on the individual inputs. Traditionally, people are intend to research single feature to find the maximize the action value, which can be showed as: $x' = arg_{x \in \imath} max\langle \varnothing(x), e_\imath \rangle$. [11] $\imath$ is the set of inputs data from untrained network, $e_\imath$the basis vector. Based on the first property, one get reliable results by using combination of high level units instead using single high level inputs, which can be showed as following:$x' = arg_{x \in \imath} max\langle \varnothing(x), \upsilon \rangle$. υ is random direction which $\upsilon \in \mathbb{R}^n$.[11] To evaluate the above equation, the author test it on MNIST in natural basis and random directions. Then the author test it on AlexNet in random basis and natural basis. Both results shows that this method can successfully generate invariance of adversarial examples of the input distribution. The experiment results can be seen as below[11]:

To better understand its behavior, the author then



(a) Unit sensitive to lower round stroke.

(b) Unit sensitive to upper round stroke, or lower straight stroke.

(c) Unit senstive to left, upper round stroke.

(d) Unit senstive to diagonal straight stroke.

Figure 11: An MNIST experiment[11]

continue to do the Blind Spots in Neural Networks. To illustrate the implementation of image classifier, one have a formal description. Let classifier map each pixel of a image, one can get: $f : \mathbb{R}^m \longrightarrow \{1...k\}$. Then we define loss function associated continuous with f: $\mathbb{R}^m \times \{1..k\} \longrightarrow \mathbb{R}^+$. [11]The $1..K$ means sets of target label image. For each given input x, which $x \in \mathbb{R}^m$, we want to get the minimize $\|r_2\|$ which can fulfill:$f(x+r) \in \mathbb{R}^m$, $x + r \in [0,1]^m$.Then by line-search we can find $D(x,l)$, which can be used to find minimum $c > 0$ and minimum r to satisfy $f(x+r) \in \mathbb{R}^m$. The processing can be concluded as following: $Minimize\{c|r|\} + loss_f(x+r,l)$, $x+r \in [0,1]^m$. [11] They got an very promised experiment results which the test error below 1.2 %. For all the framework they tested, including MNIST, QuocNet and AlexNet, they can generate adversarial examples with minimal distance. Also the experiments results show a high transferability. Adversarial examples generated by one model can be applied to other models.[11]A part of experiment results can be seen as below:

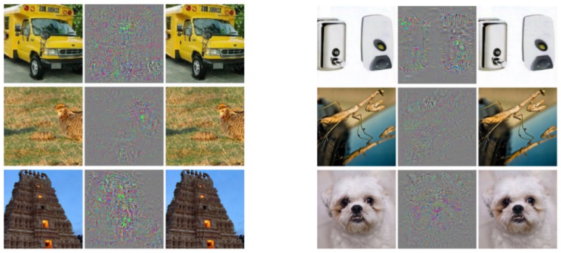


Figure 12: Adversarial examples generated for AlexNet[11]

2.16. Transferable Adversarial Examples

Yanpei Liu and Xinyun Chen et al. conclude the exiting method to generate ADVERSARIAL EXAMPLES, which is benefit for building konwledge foundation for our further research.[15] In their work, first, they introduced the ADVERSARIAL DEEP LEARNING PROBLEM. The ADVERSARIAL DEEP LEARNING PROBLEM can be concluded as following: given a deep learning classifier and inputs. We label some inputs and get some outputs. The ADVERSARIAL DEEP LEARNING PROBLEM is to find the ADVERSARIAL EXAMPLES of the outputs. Then they introduced three method to generate ADVERSARIAL EXAMPLES, which can be concluded as: optimization-based method, fast gradient method, and fast gradient sign method. The optimization-based method can be concluded as: $argmin_{x^*} \lambda d(x,x^*) - \ell d(1_y, j_\theta(x^*))$, ℓ [15]indicates a loss function, 1_y the label. This method was first introduced in Carlini & Wagner (2016). The Fast gradient sign (FGS) was first introduced by Ian J. Goodfellow. He introduced a method called fast gradient sign method to generate adversarial examples: $a = sign(i_x J(k,x,y))$.[4] In this method, x means the input feature, y the attack target, k the parameters of the model we use and J function the cost function. One can use this method to get the pertubation a and this method also shows adversarial examples are resulted from the linear behavior as we discussed above.[4] The fast gradient sign method was to optimize the fast gradient method, which means they are similar. In this method, they specific improve the sign method by redefining the distance, which redefine as $\frac{\nabla_x \ell}{\|\nabla_x \ell\|}$.[15] To examine the effectiveness of three methods, they made a different approach. To examine the optimization-based method, they adopted Adam Optimizer (Kingma & Ba (2014)) to search adversarial examples around input x. Besides, in order to search adversarial examples with large distortions, they increase the learning rate. However, this method shows poor transferability. The adversarial examples generated by one model cannot transfer to another models.For the fast gradient method, they used to approach described as (Ian J. Goodfellow 2016), which intend to get the minimal distortion and well transferability. What's more, they proposed a method that can generate adversarial examples for multiple models with high transferability, which is called as ensemble-based approach. $argmin_x^* - log((\sum_{i=1}^k \quad \alpha_i J_i(x^*))1_{y^*}) + \lambda d(x,x^*)$, $\sum_{i=1}^k \quad$ is the ensemble model.[15]

To better test the transferability of above methods, they test the decision boundary of different models,such as VGG-16,ResNet-50,ResNet-101,ResNet-152 and GoogLeNet. The VGG-16 is gradient direction, while the others are random direction. The results shows that the central area of the image can be classify correctly.[15]

2.17. Adversarial Examples For Semantic Image Segmentation

Many research about ADVERSARIAL EXAMPLES focus on the image classifier and speech classifier, but Volker Fischer et al. proposed a study to research adversarial examples on the task of semantic segmentation field.[16] In their work, they first they define adversarial examples that can be used for semantic segmentation. Then they explore the approach to generate adversarial examples such as

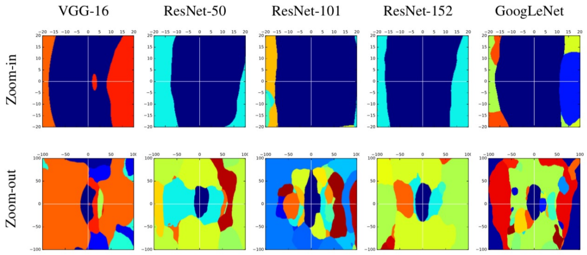


Figure 13: Decision regions of different models. [15]

optimization-based method, fast gradient method, and fast gradient sign method to adopt for semantic segmentation.

To understand their method, first, they define a semantic segmentation function f_θ and x with a true label of image inputs y^{true}. Besides, they define loss function of classifier $J_{cls}(f_\theta(x), y)$. To get the perturbation $\Im$, they chose a basic method to generate adversarial examples and optimize it, which is method presented by Kurakin et al. (2016). they iteratively compute as following: $\Im^0 = 0$, $\Im^{(n+1)} = Clip_\varepsilon\{\Im^n - \alpha sgn(\nabla_x J_{cls}(f_\theta(x + \Im^n), y^{target}))\}$. [16]

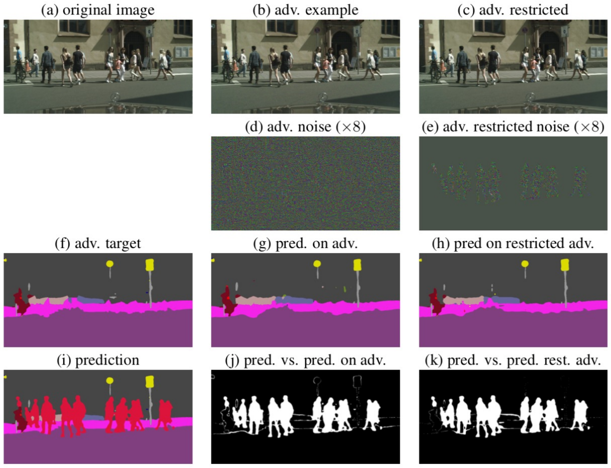


Figure 14: Adversarial example for the task of semantic segmentation. [16]

Then they defined a adversarial target for semantic segmentation. Their method can be concluded as following: first they define a adversarial target which covers all pixels of the image. Then each target of a class was replaced by the nearest different pixels.So they can measure the effectiveness of their method by comparing the percentage of pixel of the chosen class to be changed and the percentage of pixel that were preserved on the background. The test results can be seen as below: the light grey means the percentage of pixel of the chosen class to be changed and the red means the percentage of pixel that were preserved on the background. The results are promised that nearly 55.5% of trained samples intersected. adversarial perturbations were successfully applied to the all pixels of image. Also, they found it is difficult to detect the adversarial perturbation. From the experiment result, one can also see that nearly 85% of target pixel can be successfully hidden while nearly 97% of pixels were preserved on the background. Other conclusion can be seen that with smaller ε, more pixels were preserved on the background, but less target pixels be clocked, and more target pixels can be recovered. [16]

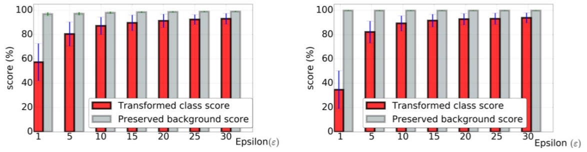


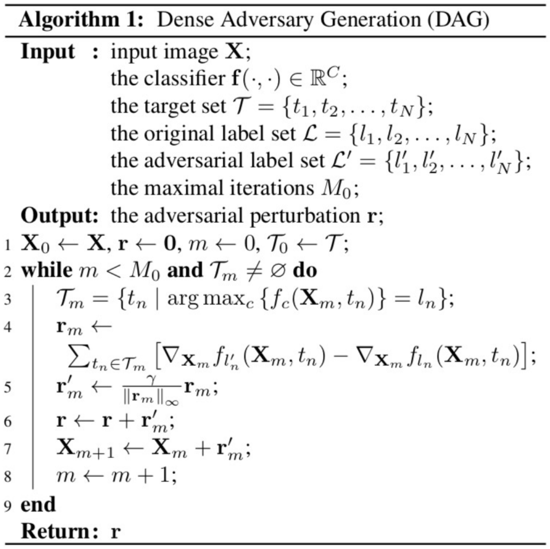
Figure 15: Mean and standard deviation for percentages of preserved. [16]

2.18. Adversarial Examples for Semantic Segmentation and Object Detection

Cihang Xie1, Jianyu Wang2 and Zhishuai Zhang et al. proposed a new algorithm which can generate a large amount of adversarial examples for segmentation and detection. Also, they tested the transferability of their algorithm, which shows that the Adversarial Examples generated by algorithm can transfer across different network without require of same training data.[17]

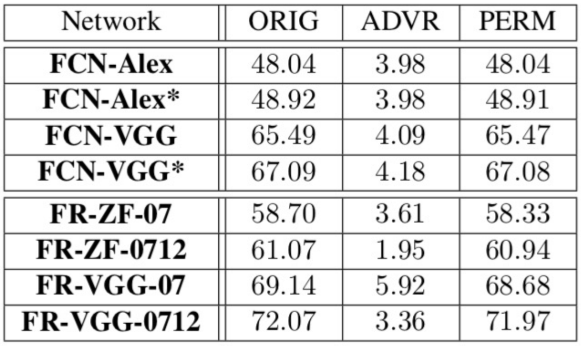
To understand Dense Adversary Generation(DAG) algorithm, first, one should choose a mount of targets, which indicates as $\tau = \{t_1, t_2, ..., t_N\}$. Each target t_N should be labeled as $\ell = \{l_1, l_2, ..., l_N\}$, l_N means the number of class. To get adversarial examples, one should make classifier misclassified as many as targets as possible. So let $f(x, t_n) \in \mathbb{R}^c$, there should be $\forall n$, $argmax_c\{f_c(X + r, t_n)\} \neq l_n$, r the adversarial perturbation of X. $\ell' = \{l'_1, l'_2, ...l'_n\}$ indicates the adversarial examples outputs label. So the loss function can be concluded as:$L(X, T, \ell, \ell') = \sum_{\iota=1}^{N} \quad [f_{\ell n}(X, t_n) - f_{\ell n'}(X, t_n)]$. Besides, they also optimize fast gradient sign method. [17]In their method, they redefine the fast gradient sign method as $r_m = \sum_{t_n \in \tau} \quad [\nabla_{x_m} f_{\ell n}(X, t_n) - \nabla_{x_m} f_{\ell n'}(X, t_n)]$.[17] Their experiments shows as below:

One can conclude that this algorithm is effective because the classifier misclassified more target after adapted DAG algorithm.



```
Algorithm 1: Dense Adversary Generation (DAG)
Input  : input image $\mathbf{X}$;
         the classifier $\mathbf{f}(\cdot,\cdot) \in \mathbb{R}^C$;
         the target set $\mathcal{T} = \{t_1, t_2, \ldots, t_N\}$;
         the original label set $\mathcal{L} = \{l_1, l_2, \ldots, l_N\}$;
         the adversarial label set $\mathcal{L}' = \{l'_1, l'_2, \ldots, l'_N\}$;
         the maximal iterations $M_0$;
Output: the adversarial perturbation $\mathbf{r}$;
$\mathbf{X}_0 \leftarrow \mathbf{X}, \mathbf{r} \leftarrow \mathbf{0}, m \leftarrow 0, \mathcal{T}_0 \leftarrow \mathcal{T}$;
while $m < M_0$ and $\mathcal{T}_m \neq \varnothing$ do
    $\mathcal{T}_m = \{t_n \mid \arg\max_c \{f_c(\mathbf{X}_m, t_n)\} = l_n\}$;
    $\mathbf{r}_m \leftarrow \sum_{t_n \in \mathcal{T}_m} \left[\nabla_{\mathbf{X}_m} f_{l'_n}(\mathbf{X}_m, t_n) - \nabla_{\mathbf{X}_m} f_{l_n}(\mathbf{X}_m, t_n)\right]$;
    $\mathbf{r}'_m \leftarrow \frac{\gamma}{\|\mathbf{r}_m\|_\infty} \mathbf{r}_m$;
    $\mathbf{r} \leftarrow \mathbf{r} + \mathbf{r}'_m$;
    $\mathbf{X}_{m+1} \leftarrow \mathbf{X}_m + \mathbf{r}'_m$;
    $m \leftarrow m + 1$;
end
Return: $\mathbf{r}$
```

Figure 16: Dense Adversary Generation(DAG)[17]



Network	ORIG	ADVR	PERM
FCN-Alex	48.04	3.98	48.04
FCN-Alex*	48.92	3.98	48.91
FCN-VGG	65.49	4.09	65.47
FCN-VGG*	67.09	4.18	67.08
FR-ZF-07	58.70	3.61	58.33
FR-ZF-0712	61.07	1.95	60.94
FR-VGG-07	69.14	5.92	68.68
FR-VGG-0712	72.07	3.36	71.97

Figure 17: Semantic segmentation (measured by mIOU, %) and object detection (measured by mAP, %) results of different networks.[17]

2.19. Adversarial Attacks on Neural Network Policies

Adversarial attacks are not only effective in computer vision applications, but also will do great harm when targeting neural network policies, which are trained with deep reinforcement learning method. The basic idea of launching adversarial attacks toward neural network policies is to add small perturbations to the original input to degrade the performance of the test time[20].

The main contribution is to find out whether the reinforcement learning algorithms used to train the policy and whether the adversary launches white-box or black-box attacks will impact the effectiveness of the adversarial examples. There are three main algorithms used in this experiment, which are DQN (Deep Q-Networks), TRPO (Trust Region Policy optimization), and A3C (Asynchronous Advantage Actor-Critic). The targets are four Atari games. The experiment results show that the policies trained by these three algorithms are vulnerable to the adversarial examples[20]. Besides, the policies trained by TRPO and A3C are more resistant to the adversarial examples than the policy trained by DQN. Secondly, the transferability property is also effective in reinforcement learning applications.

In this paper, the authors used Fast Gradient Sign Method (FGSM) to generate adversarial examples. Due to the feature that FGSM makes a linear approximation of a deep model, it can efficiently generate adversarial examples.

When applying FGSM, it is very importatnt to choose a norm constraint. The optimal perturbation used in this method for each norm constraint is[20]:

$\eta = \epsilon sign(\nabla_x J(\theta, x, y)) \quad for\ constraint \parallel \eta \parallel_\infty \leqslant \epsilon$

$\eta = \epsilon\sqrt{d} * \frac{\nabla_x J(\theta,x,y)}{\|\nabla_x J(\theta,x,y)\|_2} \quad for\ constraint \parallel \eta \parallel_2 \leqslant \parallel \epsilon 1_d \parallel_2$

maximally perturb highest-impact dimensions with budget

$\epsilon d \quad for\ constraint \parallel \eta \parallel_1 \leqslant \parallel \epsilon 1_d \parallel_1$

When it comes to the evaluation stage, the environment used in this paper is on four Atari 2600 games. The neural network policies trained by three algorithms are all vulnerable to white-box attacks, which means not matter how the policy is trained or what norm constraints are chosen for FGSM, the performance of the policy will be greatly decreased when under white-box adversarial example attacks[20].

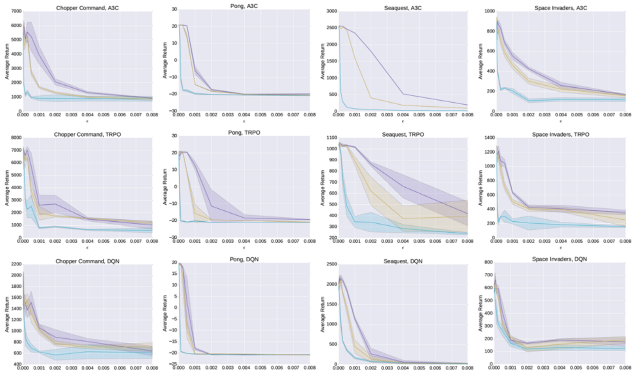


Figure 18: Comparison of different norm constraints chosen when applying FGSM on four Atari games trained with DQN, TRPO, and A3C. [20]

Besides, the policy trained through three algorithms are also vulnerable to the black-box attacks. In this scenario, the authors give the situations that refer to both the transferability across policies and algorithms.

2.20. Detect Adversarial Samples from Artifacts

Deep neural networks (DNNs) are one of the machine learning techniques of hierarchical architecture with multiple layers of nonlinear processing units. This paper is to investigate whether Deep Neural Networks can distinguish the normal and noisy counterparts of the adversarial examples[21].

Although DNNs is resistant to noisy inputs, but it is still vulnerable to some specific adversarial examples, which are constructed by taking a normal sample and then perturbing it. Very small perturbation in the inputs can lead to high efficiency of the misclassify in DNNs. Whats more, some adversarial examples can also make DNNs misclassify to a specific class.

There are two features that can be used in detecting adversarial examples, which are kernel density estimates in the subspace of the last hidden layer and Bayesian uncertainty estimates[21]. Some of the most recently popular attacks are also introduced in this paper, such as Fast Gradient Sign Method (FGSM), Basic Iterative Method (BIM), Jacobian-based Saliency Map Attack (JSMA) and Carlini & Wagner (C & W).

The best detecting performance can be achieved by combining both these methods at the same time because they are dealing with complementary conditions.

2.21. Tactics of Adversarial Attack on Deep Reinforcement Learning Agents

To degrade the performance of the reinforcement learning agents which are trained by the reinforcement learning algorithms including DQN and A3C[22]. This paper introduces two tactics, which are strategically-timed attack and the enchanting attack.

For the strategically-timed attack[22], the basic idea is to minimize the reward that the agents may get from the current state by attacking the small subset of time steps. It will be more difficult for the agent to detect the attack if the attack activity to this subset is limited. Besides, a new method has been approached to launch this adversarial example attack.

A relative action preference function is used to solve the when-to-attack problem. The preference of the agent in taking the actions from the most preferred to the least is computed by this function, and the degree of the preference depends on Deep Neural Network policy.

For policy gradient-based methods, a specific action has more possibilities to be performed means this action is very critical. The function can be defined as follows[22],

$$C(S_t) = max_{a_t}\ \pi(s_t, a_t) - min_{a_t}\ \pi(s_t, a_t)$$

For value-based methods, a softmax function is used to convert Q-values into a probability distribution over actions. The function can be defined as follows[22],

$$C(S_t) = max_{a_t}\ \frac{e^{\frac{Q(s_t,a_t)}{T}}}{\sum_{a_k} e^{\frac{Q(s_t,a_t)}{T}}} - min_{a_t}\ \frac{e^{\frac{Q(s_t,a_t)}{T}}}{\sum_{a_k} e^{\frac{Q(s_t,a_t)}{T}}}$$

To solve the how-to-attack problem, a perturbation is searched to be added to the observation, so the originally preferred action will be replaced by the originally least preferred one.

For the enchanting attack, the basic idea is to lure the agent to enter into the designated target state[22]. To achieve this goal, we need to combine a generative model and a planning algorithm. The function of the generative model is to predict the future states, and the planning algorithm is used to give out a preferred sequence of actions in which the agents can choose from.

Both the strategically-timed attack and enchanting attack are significantly effective in degrading the performance of the model trained by DQN and A3C in five Atari games.

2.22. SafetyNet: Detecting and Rejecting Adversarial Examples Robustly

A method is designed to generate a network to make the methods like DeepFool have great difficult in producing adversarial examples. SceneProof, which is a new application used to detect whether an image is being attacked or not, uses SafetyNet[23] to achieve this goal.

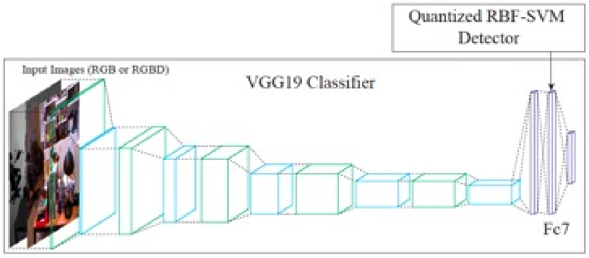


Figure 19: SafetyNet, which can detect out adversarial examples and make it difficult to produce adversarial examples, uses a conventional classifier with RBF-SVM. The adversarial example detector looks inside the later layers. [23]

The attacking methods used in this paper toward SafetyNet include Fast Sign method, Iterative method, DeepFool method, and Transfer method. The generalization across attacks is also tested by tried on other class of attack. SafetyNet can both spot and reject adversarial examples[23]. The attacks that are launched both to fool the classifier and hide from the detector fails in SafetyNet as well. Furthermore, SafetyThe transferred adversarial examples remain greatly effective on the models when doing black-box attacks even if these models are robust during white-box attacks. A new and simple but powerful white-box attack is proposed at the very beginning of this paper, which outperforms prior single-step attacks toward the models[23]. Secondly, Ensemble Adversarial Training is proposed, which greatly improve the robustness of the model against black-box attacks.

2.23. Ensemble Adversarial Training: Attacks and Defenses

The transferred adversarial examples remain greatly effective on the models when doing black-box attacks even if these models are robust during white-box attacks. A new simple but powerful white-box attack is proposed at the very beginning of this paper, which outperforms prior single-step attacks toward the models[24]. Secondly, Ensemble Adversarial Training is proposed, which greatly improve the robustness of the model against black-box attacks.

The adversarial training on MNIST (trained with two different convolutional networks) and ImageNet (Inception v4 model, Inception v3 model, adversarially trained Inception v3 model) shows greatly different robustness between black-box attacks and white-box attacks[24].

This new attack requires single gradient computation. RAND + FGSM usually have higher success rates than FGSM with given perturbation magnitude. The new attack is defined as follows[24],

$x' = x + \alpha * sign(\mathcal{N}(0^d, I^d))$

$x^{adv} = x' + (\epsilon - \alpha) * sign(\nabla_{x'} J(x', y_{true}))$

$for\, parameter \quad \epsilon\, and\, \alpha \qquad where(\alpha < \epsilon)$

After 6 epochs on MINST of training, ensemble adversarial training performs better on black-box attacks, but still encounters high error rate on white-box FGSM examples. After 12 epochs of training, robustness to white-box attacks is greatly improved. For ImageNet, both ImageNet and RAND + FGSM transfer at a lower rate[24].

2.24. Adversarial Example Defense: Ensembles of Weak Defenses are not Strong

There are many methods designed to defend adversarial examples toward neural networks, but most of them have been shown to be ineffective during practical applications. This paper mainly focuses on how to combine multiple defense methods together to make complementary.

Given the assumption that the adversary launches white-box attacks, which means that the adversary has full command of the algorithms used to trained the model, including the parameters, model structure, or even the defense strategies. There are two main types of adversaries, which are static adversary and adaptive adversary[25]. Static adversary refers to the attacks that do not take the possible defense methods into consideration when the attacks are launched. The adaptive adversary refers to the attackers that are aware of the possible defenses, and they are also able to adapt their attacks accordingly. Obviously, adaptive adversary is more powerful than the static one[25].

The first approach used is the feature squeezing defense. For the reason that the adaptive adversary can keep his adversarial examples adversarial even after the squeezing process finishes. The score will get smaller than the fixed threshold by constructing the adversarial examples both with and without squeezing[25].

Spatial smoothing alone is also ineffective defense against the adaptive attackers. The combination of two methods of squeezing defenses are better than the spatial smoothing alone, but still it is not that strong enough. The specialists+1 ensemble slightly increases the required distortion, but still not big enough. For the conclusion, the adversarial examples transfer across detectors and there is no strongly effective methods that can stop that.

2.25. Transferability of Adversariail Examples Across Algorithms

The transferability is one of the most significant features in adversarial attack. Transferability refers to the situation that an adversarial example specifically trained to misclassify one model has high possibilities of misclassifying other models when launching black-box attacks.

3. Gradient Ascent with Noise

We have discussed the existing working approaches for generating adversarial examples above. Some methods like saliency maps are used to generate targeted adversarial samples, the other approaches like fast gradient sign method are mainly utilized to produce non-targeted adversarial examples. In this project, we introduce uniform noise to the gradient, then use iterative gradient ascent way to generate samples. Based on the dimensional mapping property, there should be a large number of adversarial points in the input space, it is very likely gradient-based approach like fast gradient sign method fall within the local optimal dilemma.

3.1. Gradient Ascent with Noise Approach to Generate Adversarial Examples

In our Gradient Ascent with Noise approach, the noise is a vector with the same shape as the input vector, each element of the noise vector is from uniform distribution $U(0, 1)$. We introduce the uniform noise since it is able to narrow the perturbation, and noise could somehow reduce local optimal problem. The following formula is the perturbation computed in an iteration with our approach, α is the learning rate, ϵ denotes the noise, a vector with the same shape as $\nabla_x J(\theta, x, y)$, each element follows the uniform distribution $\epsilon_i \sim U(0, 1)$, $J(\theta, x, y)$ denotes the cost function, θ is the parameter of target model F.

$\eta = \alpha * \epsilon \odot \nabla_x J(\theta, x, y)$

Algorithm Gradient Ascent with Noise

```
Input: learning rate $\alpha$, image $I$, cost function $J(\theta, x, y)$,
target model $F$ with parameter $\theta$, ground truth label $L$,
iteration $T$
for t = 1, 2, ..., $T$ do
    Sample $\varepsilon$ for $\varepsilon_1, ...\varepsilon_n \sim \mathbb{U}(0,1)$
    Compute gradient $\nabla_x J(\theta, x, y)$
    perturbation $\eta = \alpha * \sqrt{t} * \varepsilon \odot \nabla_x J(\theta, x, y)$
    if($F(I + \eta)$ is expected) return $I + \eta$
    else $I = I + \eta$
end for
```

3.2. Dataset

For the framework we use, we use $x \in \mathbb{R}^m$ as input. There are many different dataset, such as ResNet-50, ResNet-101, ResNet-152, CIFAR-10, MNIST and ImageNet. Some datasets have limited number of classes and each class is distinct with others, such as MINST, CIFAR-10, this kind of dataset are easy to deal with and nearly most of the exiting methods can generate acceptable adversarial examples through these dataset. Whereas, other dataset, such as ImageNet, are more complicated, which has more classes and the distance of each class is limited. For this kind of dataset, most of methods are likely create error. To deal with this problem, we choose ImageNet as our dataset, ImageNet is widely exploited in DNNs, especially for benchmarking. We use Inception-V3 as the target classifier and generate adversarial examples against it. For better benchmarking, we directly use the dataset from Google team, issued in NIPS adversarial competition[28].

3.3. The result of Gradient Ascent with Noise Approach

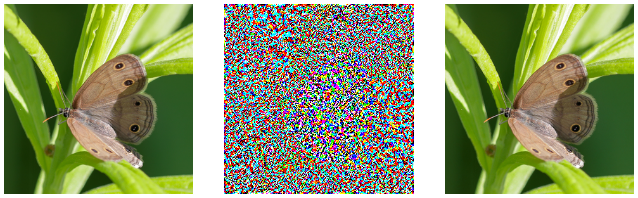
The Gradient Ascent with Noise approach is able to generate adversarial examples effectively, more details about the performance of gradient ascent with noise approach please refer to section "Evaluation". We also want to test the transferability of our algorithm, but due to our limited computational resources, we do not verify the transferability. Since our algorithm is extended from basic gradient method, we can hypothesize the effectiveness of the transferability of our algorithm as following. According to research of (Ian Goodfellow et.al, 2015), the adversarial examples generated by fast gradient method with on model can transfer to other models, which means it has high transferability. Since we introduce noise to generate perturbation, we assume gradient ascent with noise approach has high transferability. Because the limitation of computational resources, we cannot examine our thoughts in this project. We may leave this question latter to solve it.

3.4. Mining the Value of Adversarial Examples

Adversarial examples can make classifier misclassify any inputs to others, thus it can cause security vulnerabilities. Any operation system using machine learning classifier may have adversarial examples vulnerabilities, such as some IoT devices. Many IoT devices, like Amazon echo, using intelligent operation system. As time goes on, it will be equipped more intelligent operation system based on machine learning classifier. That is, Adversarial examples can cause such IoT devices face serious security problem. e.g. attackers may take advantage of adversarial examples to make smart-locker misclassify the face ID or finger ID, and enter into the house do some criminal things. Thus, taking deep exploration to adversarial examples and finding ways to prevent it have a great value in the security field. Such research has been made as we discussed in the related work section.

4. Evaluation

In this section, we present the performance of gradient ascent with noise. Due to the fact that we only have very limited computational resources, it takes us great amount of time to train our model and to generate adversarial examles. So we just randomly pick 100 images from the dataset for this evaluation. But we will continue our work to evalute our method with larger dataset in the future. The two performance evaluation metrics used in this work are accuracy and perturbation rate. Accuracy is the ratio of number of successful adversarial samples to the total number of original input images. Perturbation rate is the ratio of the sum of perturbation to sum of input vector. The following figure is an adversarial example generated by gradient ascent with noise approach.



(a) natural image (recognized as Butterfly)

(b) perturbation by gradient ascent with noise

(c) adversarial image (recognized as Starfish)

Figure 20: Example of Gradient Ascent with Noise on inception-v3

The overall accuracy of gradient ascent with noise approach is 92%, we get the accuracy with three iterations, it is supposed to be better with more iterations. The accuracy of 92% is competitive with the existing approaches like fast gradient sign approach. The relevant perturbation rate is 0.063%.

5. Future work

In the future, we would like to explore adversarial examples in more applications such as speech recognition, reinforcement learning, etc. To better support the gradient ascent with noise approach, a distributed and batched version should be supported. Due to our limited computational resources, a comprehensive evaluation on the property of transferability will be performed later. We are still significantly interested in figuring out more creative and transferable way to generate adversarial examples and more efficient method to defense such attack, especially in targeted adversarial samples. For example, the fast gradient method may be optimized under the method of attacking multiples targets aend get training simultaneously.

6. Summary

We specifically explain multiple existing methods of generating adversarial examples and perform a lot of research on adversarial attacks toward image recognition. In this project, we propose the gradient ascent with noise approach to generate adversarial samples, the noise is able to decrease the perturbation, and is supposed to introduce surprising transferability. We experiment the approach and get competitive accuracy with quite small perturbation rate.

7. Acknowledge

We would like to thank Dr. Timothy Leschke of Johns Hopkins University for contributing and continually guiding us in the project. We would also like to express our gratitude and appreciation to all the individual researchers who helped us in this work, and the Cleverhans community[18].

References

[1] Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, Wei-Yang Lin. Intrusion detection by machine learning: A review. Expert Systems with Applications 36 (2009) 1199412000.

[2] Andrew NG, Machine Learning course lecture note at Standford University.

[3] Y LeCun, L Bottou, Y Bengio and P Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, 1998.

[4] Ian J. Goodfellow, Jonathon Shlens and Christian Szegedy, ”EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES,” Published as a conference paper at ICLR 2015.

[5] Anish Athalye, Logan Engstrom, Andrew Ilyas, Kevin Kwok, “SYNTHESIZING ROBUST ADVERSARIAL EXAMPLES”, Under review as a conference paper at ICLR 2018.

[6] Anh Nguyen, Jason Yosinski, Jeff Clune. “Deep neural networks are easily fooled: High confidence predictions for unrecognizable images.” In Computer Vision and Pattern Recognition (CVPR 2015). IEEE, 2015.

[7] Seyed-Mohsen, Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawziz, Pascal Frossardy. “Universal adversarial perturbations.” IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[8] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, Ananthram Swami, “Practical Black-Box Attacks against Machine Learning.” ASIA CCS 17, April 02 - 06, 2017, Abu Dhabi, United Arab Emirates.

[9] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Pascal Frossard. “DeepFool: a simple and accurate method to fool deep neural networks.” IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[10] Nicolas Papernot, Patrick McDaniel, Somesh Jhay, Matt Fredriksonz, Z. Berkay Celik, Ananthram Swami. “The Limitations of Deep Learning in Adversarial Settings.” IEEE European Symposium on Security & Privacy, IEEE 2016. Saarbrucken, Germany.

[11] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow and Rob Fergus, “Intriguing properties of neural networks,” arXiv preprint arXiv:1312.6199v4, 2014.

[12] Alexey Kurakin, Ian J. Goodfellow . “Adversarial examples in the physical world.” arXiv preprint arXiv:1607.02533 [cs.CV], 2017

[13] Jernej Kos, Dawn Song. “Delving into adversarial attacks on deep policies.” arXiv preprint arXiv:1705.06452 [stat.ML], 2017

[14] Martn Abadi, David G. Andersen . “Learning to Protect Communications with Adversarial Neural Cryptography.” arXiv preprint arXiv:1610.06918v1 [cs.CR]

[15] anpei Liu, Xinyun Chen, Chang Liu, Dawn Song, ”DELVING INTO TRANSFERABLE ADVERSARIAL EX- AMPLES AND BLACK-BOX ATTACKS”, arXiv preprint arXiv:1611.02770v3 [cs.LG], 2017

[16] Volker Fischer, Mummadi Chaithanya Kumar, Jan Hendrik Metzen, Thomas Brox,”ADVERSARIAL EXAMPLES FOR SEMANTIC IMAGE SEGMENTATION”,arXiv preprint arXiv:1703.01101v1 [STAT.ML], 2017

[17] Cihang Xie1*, Jianyu Wang2*, Zhishuai Zhang1, Yuyin Zhou1, Lingxi Xie1, Alan Yuille1, ”Adversarial Examples for Semantic Segmentation and Object Detection”,arXiv preprint arXiv:1703.08603v3 [CS.CV], 2017

[18] Nicolas Papernot, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Fartash Faghri, Alexander Matyasko, Karen Hambardzumyan, Yi-Lin Juang, Alexey Kurakin, Ryan Sheatsley, Abhibhav Garg, Yen-Chen Lin. “cleverhans v2.0.0: an adversarial machine learning library”, arXiv preprint arXiv:1610.00768, 2007

[19] David Silver. “Introduction to Reinforcement Learning”

[20] Huang, Sandy, et al. ”Adversarial attacks on neural network policies.” arXiv preprint arXiv:1702.02284 (2017).

[21] Feinman, Reuben, et al. ”Detecting Adversarial Samples from Artifacts.” arXiv preprint arXiv:1703.00410 (2017).

[22] Lin, Yen-Chen, et al. ”Tactics of Adversarial Attack on Deep Reinforcement Learning Agents.” arXiv preprint arXiv:1703.06748 (2017).

[23] Lu, Jiajun, Theerasit Issaranon, and David Forsyth. ”Safetynet: Detecting and rejecting adversarial examples robustly.” arXiv preprint arXiv:1704.00103 (2017).

[24] Tramr, Florian, et al. ”Ensemble Adversarial Training: Attacks and Defenses.” arXiv preprint arXiv:1705.07204 (2017).

[25] He, Warren, et al. ”Adversarial Example Defenses: Ensembles of Weak Defenses are not Strong.” arXiv preprint arXiv:1706.04701 (2017).

[26] Tramr, Florian, et al. ”The Space of Transferable Adversarial Examples.” arXiv preprint arXiv:1704.03453 (2017).

[27] Face Recognition performance, “https://www.sighthound.com/technology/face-recognition/benchmarks/pubfig200”

[28] NIPS 2017: Non-targeted Adversarial Attack.“https://www.kaggle.com/c/nips-2017-non-targeted-adversarial-attack”

[29] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton.“ImageNet Classification with Deep Convolutional Neural Networks”. Advances in Neural Information Processing Systems 25 (NIPS 2012)