Fault detection method based on high-level feedback and reconstruction for image recognition

Xu, Jin

2021

Xu, J. (2021). Fault detection method based on high-level feedback and reconstruction for image recognition. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/146731>

<https://doi.org/10.32657/10356/146731>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 19:52:19 SGT

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**FAULT DETECTION METHOD BASED ON HIGH-
LEVEL FEEDBACK AND RECONSTRUCTION FOR
IMAGE RECOGNITION**

FAULT DETECTION METHOD BASED ON HIGH-LEVEL FEEDBACK
AND RECONSTRUCTION FOR IMAGE RECOGNITION

XU JIN

XU JIN

SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

2021

2021

**FAULT DETECTION METHOD BASED ON HIGH-LEVEL FEEDBACK AND
RECONSTRUCTION FOR IMAGE RECOGNITION**

XU JIN

School of Mechanical & Aerospace Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

2021

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

20/12/2019

.....
Date

.....
Xu Jin

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

20/12/2019

.....
Date

.....
ASSOC PROF Xie Ming

Authorship Attribution Statement

This thesis **does not** contain any materials from papers published in peer-reviewed journals or from papers accepted at conferences in which I am listed as an author.

20/12/2019

.....

Date

.....

Xu Jin

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere appreciation to my supervisor, Associate Professor Xie Ming, for his support and guidance all along these years. From each and every discussion with him, I got sparkles of insight and new ideas to move forward.

I am also indebted to my family for giving me the courage to finish this difficult journey. Especially, I would like to thank my wife, Sylvia, for all the understanding and support from her during this journey.

Finally, my gratitude is extended to all the friends who have discussed with me on this research topic or related topics. All of you helped in different ways for me to achieve the final result.

TABLE OF CONTENTS

LIST OF TABLES	vii
LIST OF FIGURES	viii
NONMENCLATURE	x
ABSTRACT	xii
CHAPTER 1 PROBLEM STATEMENT	1
1.1 Task of Image Recognition.....	1
1.2 Introduction of Deep Learning.....	5
1.3 Limitation of Deep Learning Methods.....	10
1.4 Adversarial Examples.....	14
1.4.1 Discovery of Adversarial Examples.....	14
1.4.2 For Critical Applications such as Autonomous Driving.....	15
1.5 Problems of Existing Defense Methods and Possible Directions.....	16
1.6 Contributions of This Study.....	18
1.7 Chapter Description.....	21
CHAPTER 2 LITERATURE REVIEW	23
2.1 Adversarial Examples in Conventional Machine Learning	24
2.2 Adversarial Examples in Deep Learning	25
2.2.1 Adversaries for Different Applications	26
2.2.2 Characteristics of Adversarial Examples	40
2.2.3 Analysis on the Cause for Adversarial Examples	42
2.3 Method of Categorizing the Literature.....	45
2.3.1 Categorization of Attacks.....	47
2.3.2 Categorization for Defenses.....	50
2.4 Review of Adversarial Attack Methods	55
2.5 Review of Defense Methods	66
2.5.1 Modifying Data.....	67
2.5.2 Modifying Model.....	72
2.5.3 Network Add-ons.....	81
2.6 Summary	90
CHAPTER 3 METHODOLOY OF DBFR FRAMEWORK	92
3.1 Formulation of the Problem.....	93

3.2 Introduction of the Proposed Method.....	96
3.3 Details of the Proposed Method.....	104
3.3.1 Architecture.....	104
3.3.2 Training.....	106
3.3.3 Quantitative Comparison of Reconstructed and Input Images.....	107
CHAPTER 4 EXPERIMENT.....	109
4.1 Network Implementation Details.....	110
4.2 Input.....	112
4.2.1 Dataset of Traffic Signs GTSRB.....	112
4.2.2 Dataset of Dogs vs. Cats.....	113
4.3 Results.....	114
4.3.1 Dataset of Traffic Signs GTSRB.....	114
4.3.2 Dataset of Dogs vs. Cats	120
4.4 State-of-the-Art Comparison	127
4.5 Discussion on the Experiment Results.....	128
4.6 Summary.....	132
CHAPTER 5 CONCLUSIONS AND DISCUSSIONS	133
5.1 Discussions.....	133
5.2 Conclusions.....	135
5.3 Future Works	136
REFERENCES	139

LIST OF TABLES

Table 1	Example of adversarial examples for text classification.....	33
Table 2	Example of adversarial examples for machine translation.....	34
Table 3	List of reviewed adversarial attack methods.....	56
Table 4	List of reviewed defense methods.....	66
Table 5	Mean SSIM for images reconstructed from Left and Right decoders	116
Table 6	STD of SSIM for images reconstructed from Left and Right decoders ..	116
Table 7	Statistics of SSIM values for images reconstructed from two decoders .	119
Table 8	Mean SSIM for images reconstructed from Dog and Cat decoders	122
Table 9	STD of SSIM for images reconstructed from Dog and Cat decoders	122
Table 10	Comparison with SOTA methods.....	127

LIST OF FIGURES

Figure 1	Relationship of a few related concepts.....	2
Figure 2	Training and inference stages of machine learning algorithms.....	3
Figure 3	Difference between traditional machine learning and deep learning..	4
Figure 4	A convolutional layer in CNN.....	8
Figure 5	A typical structure for deep learning based classification model.....	9
Figure 6	Example of an unusual image.....	11
Figure 7	Unrecognizable objects to human being.....	11
Figure 8	Physical attack by slightly changing the normal input.....	12
Figure 9	Fast adversarial examples applied to GoogLeNet	14
Figure 10	Fake lane model in physical world	15
Figure 11	DBFR framework in comparison with existing methods	19
Figure 12	AEs for semantic segmentation and object detection.....	28
Figure 13	A dodging attack by perturbing an entire face.....	29
Figure 14	Adversarial attack on edge detection model.....	30
Figure 15	An example from the SQuAD dataset.....	31
Figure 16	Training Architecture with a GAN and an Inverter.....	33
Figure 17	Example of adversarial attack on mobile phone cameras.....	36
Figure 18	Physical adversarial perturbations in the real world.....	37
Figure 19	Illustration of the transfer learning attack setup.....	38
Figure 20	An impersonation using frames.....	39
Figure 21	A cap with some infrared LED.....	39
Figure 22	LID scores of 100 normal, noisy, and Opt attack examples.....	42
Figure 23	Adversarial crafting framework.....	47
Figure 24	Categorization of adversarial defense approaches.....	50
Figure 25	Adversarial examples for a linear binary classifier.....	62
Figure 26	Illustration of the original and null labeling methods.....	69
Figure 27	An overview of defense mechanism through distillation.....	73
Figure 28	Intuition behind the Deep k-Nearest Neighbors.....	80
Figure 29	The idea of high-level representation guided denoiser.....	82
Figure 30	Three different training methods for HGD.....	82
Figure 31	Overview of the Defense-GAN algorithm.....	83

Figure 32	An illustration of the reformer’s effect on adversarial perturbations..	84
Figure 33	A demonstration of APE-GAN.....	84
Figure 34	Illustration of defense against universal perturbations.....	85
Figure 35	Illustration of defense procedure.....	86
Figure 36	MagNet workflow.....	88
Figure 37	What is encoded by a CNN?.....	97
Figure 38	Illustration of HybridNet behavior.....	99
Figure 39	CIFAR-10 loopy model with loop layer.....	100
Figure 40	Illustration of the time-unfolded overall pipeline.....	101
Figure 41	The new framework with fault detection based on reconstruction.....	102
Figure 42	The proposed add-on network DBFR for detection.....	103
Figure 43	Architecture of the reconstruction networks.....	105
Figure 44	Analysis based on reconstructed images from all encoders.....	105
Figure 45	Fine-tuning of the classification network based on ResNet-50.....	110
Figure 46	Detailed structure of the decoder networks	111
Figure 47	Reconstructed images using Left and Right decoders respectively ...	115
Figure 48	Original image and adversarial image with C&W L-2 method	117
Figure 49	Adversarial image and the reconstructed images.....	117
Figure 50	Reconstructed images with unseen input.....	119
Figure 51	The kernel density of difference between SSIM of two decoders.....	120
Figure 52	Reconstructed images using Dog and Cat decoders respectively	121
Figure 53	Original image and adversarial image with C&W L-2 method	123
Figure 54	Adversarial image and the reconstructed images.....	124
Figure 55	Reconstructed images with unseen input.....	125
Figure 56	The histogram of difference between SSIM of two decoders.....	126
Figure 57	Distribution of the logits for clean and adversarial images.....	128
Figure 58	Linear relationship for the logits of clean and adversarial images.....	129
Figure 59	Examples of reconstructed images with low qualities.....	130
Figure 60	Input image and reconstructed images using U-net decoders.....	131
Figure 61	Architecture of the closed-loop image recognition network.....	137

Nomenclature

Roman Symbols

A	matrix
$\mathbf{a}$	vector
a	scalar
W	Weight matrix
$\mathcal{D}$	Dataset
$\mathbf{X}$	Dataset inputs (matrix with N rows, one for each data point)
$\mathbf{Y}$	Dataset outputs (matrix with N rows, one for each data point)
x_n	Input data point for model
y_n	Output data point for model
$\hat{y}_n$	Model output on input x_n
Q	Input dimensionality
D	Output dimensionality
L	Number of network layers
K_i	Number of network units in layer i

Greek Symbols

ϵ	a random variable
$\hat{\epsilon}$	a random variable realization
$\boldsymbol{\epsilon}$	a vector of random variables
$\boldsymbol{\omega}$	a set of random variables (for example a random weight matrix)

Superscripts

$\boldsymbol{\omega}$	parameters of a certain function (e.g. f^ω)
-----------------------	---

Subscripts

1	variable (e.g. $W_1: Q \times K, W_2: K \times D$)
$1, ij$	specific element of variable W_1 (e.g. $w_{1,ij}$ denotes the element at row i column j of the variable W_1)
i	row or column of a matrix (with bold letter, e.g. $\mathbf{x}_1$)
ij	specific element of a matrix (e.g. x_{ij})
q	with reference to a variable $W: Q \times K$, $\mathbf{w}_q$ denotes a row, and $\mathbf{w}_k$ denotes a column of W

Other Symbols

$\mathcal{N}$	The Gaussian distribution
$\mathbb{R}$	The real numbers

Acronyms / Abbreviations

AE	Adversarial Examples
BIM	Basic Iterative Method
CNN	Convolutional Neural Network
DL	Deep Learning
FGSM	Fast Gradient Sign Method
GAN	Generative Adversarial Network
GTSRB	German Traffic Sign Recognition Benchmark
HOG	Histogram of Oriented Gradients
KNN	K Nearest Neighbour
LSTM	Long Short-Term Memory
MNIST	Modified National Institute of Standards and Technology
MSE	Mean Square Error
NN	Neural Network
pdf	Probability Density Function
PSNR	Peak Signal-to-Noise Ratio
RL	Reinforcement Learning
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
SIFT	Scale-Invariant Feature Transformation
SSIM	Structural Similarity
SVM	Support Vector Machine
VQA	Visual Question Answer
WGAN	Wasserstein Generative Adversarial Network
a.e.	Almost Everywhere
e.g.	Exempli Gratia (“for the sake of an example”)
i.e.	Id Est (“it is”)
i.i.d.	Independent and Identically Distributed
s.t.	Such That
w.r.t.	With Respect To

ABSTRACT

Image recognition problem is a typical challenge for artificial intelligence. Deep learning methods have shown great progress in this area in recent years. However, this approach has the limitation of outputting wrong result with high confidence when the input is not well represented in the training data, such as adversarial examples (natural images with intentionally designed small perturbations that are imperceptible by human beings) and unusual images that is not covered in the training. This limitation causes great safety concerns for the deployment of existing deep learning systems for critical applications such as autonomous driving and medical diagnosis. However, it is not practical to include all adversarial or unusual images in training because images are highly variable.

It is therefore desirable to design an image recognition system with the ability of fault detection. When an “unseen” image, either adversarial or unusual, is input to the system, the fault detection mechanism recognizes the situation and raise a red flag. With the fault detected, further steps such as intervention like double checking by a person can be taken to avoid damages.

Inspired by the human perception system, the research reported in this thesis focusses on a novel fault detection mechanism and a reconstruction process. This mechanism, named Detection Based on Feedback and Reconstruction (DBFR), requires no change of the existing recognition system and generates a reconstructed image as an intuitive “internal model” to indicate whether the interpretation is supported by what is learned in the training while performing the inference. By comparing the reconstructed image with the input image, the incorrect interpretation can be detected with high probability. Experiments validate the superior effectiveness of this method over existing defense methods and the reconstructed images help to interpret the internal decision process of the deep learning system, which is currently treated as a black box in most applications.

CHAPTER 1 PROBLEM STATEMENT

THE IMPORTANCE OF FAULT DETECTION FOR IMAGE RECOGNITION SYSTEM BASED ON DEEP LEARNING

1.1 Task of Image Recognition

According to the Oxford Dictionary, Artificial Intelligence is defined as “the theory and development of computer systems able to perform tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages”. To achieve the first sub-task, visual perception, Computer Vision technology is needed. According to Prof. Fei-Fei Li in a Ted Talk, Computer Vision is defined as “a subset of mainstream Artificial Intelligence that deals with the science of making computers or machines visually enabled, i.e., they can analyze and understand an image.”

One important problem in the field of Computer Vision is Image Recognition, whose task is to find and identify objects in an image. Image Classification is a sub-problem of Image Recognition, which embodies the ability to analyze the patterns contained in an image (e.g. brightness, color and texture) and apply semantic attributes to it (e.g. identifying the image as that of a cat). However, Image Classification and Image Recognition are used interchangeably in a lot of scenarios. Figure 1 illustrates the relationship among these concepts.

In this thesis, the study is focused on assigning a single semantic attribute to an input image containing a single object in the foreground as the main content. The deep-learning based recognition system is expected to learn the features of the main object, which is present and occupies most pixels in each image of the training set, and ignore the features of the background or other minor objects that are not consistent and occupies much less pixels in the training images. With this restriction, the image recognition problem is simplified to a single label classification problem.

Figure 1. Relationship of a few related concepts

In practice, image classifiers are often used as an implementation of the Image Recognition task. For example, ImageNet [109] is an image database organized according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is associated with hundreds and thousands of images (currently, there is an average of over five hundred images per node). Classifying any input image into one of these nodes works as an elementary implementation of image recognition. There are other datasets with multi-labels, such as MS-COCO and Open Images, which gives multiple labels to a cluttered scene. For the simplicity of discussion, only single object images are discussed in this study.

Various algorithms for Image Classification have been developed in the past decades and the standard pipeline is: first a set of well-defined features are extracted; then these features are fed into sophisticated feature encoding or transformation (e.g. dimension reduction or feature pooling) procedures; finally those high-level features are sent to trained classifiers. Although many powerful features (e.g. SIFT – scale-invariant feature transform, LBP – local binary patterns, HOG – histogram of oriented gradients, Gabor – a linear filter named after Denis Gabor) have been proposed and tested, the selection of features is still undoubtedly one of the major bottlenecks for improving the performance of object recognition.

Because of huge variations and the high dimensionality of image data, it is difficult to design a high-level feature extraction algorithm and classifier for them. So these

processes become highly dependent on a technique called Machine Learning. Machine Learning is the scientific study of algorithms and statistical models that computer systems use to effectively perform a specific task without using explicit instructions, relying on patterns and inferences instead. Machine Learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions on new inputs ("test data") without being explicitly programmed to perform the task.

Figure 2 illustrates the working of the most common type of Machine Learning algorithms for classification, Supervised Learning, in the training and inference stages respectively. Each training example has one or more inputs and a desired output, also known as a supervisory signal. Through iterative optimization of an objective function, supervised learning algorithms learn a function that can be used to predict the output associated with new inputs. An optimal function will allow the algorithm to correctly determine the outputs for inputs that were not a part of the training data.

Training/Labelled Images

a) Training process of machine learning algorithms

Test Data

b) Inference process of machine learning algorithms

Figure 2. The training and inference stages of machine learning algorithms

Deep learning (also known as deep structured learning or hierarchical learning) is a new kind of Machine Learning methods based on the many layers used in artificial neural networks. It has enabled the great progress of Image Classification in recent years. The convolution neural network (CNN), a common example of deep learning algorithms, adopts a deep architecture that are made of many stacked convolutional and fully-connected layers. The designed architecture of CNN is end-to-end trainable and able to automatically learn features performing well for specific targets at different abstraction levels.

Figure 3 illustrates the differences between deep learning based and the traditional machine learning methods. The main difference lies in the way of feature extraction. The traditional machine learning methods are more focused on the classifier, with the feature extraction still depending on deliberately designed feature extractors. However, in deep learning, the features are also learned, often together with the classifier. The higher level features are extracted progressively from the input image. Another major difference between is the problem-solving approach. Deep learning techniques tend to solve the problem end to end, whereas traditional machine learning techniques need the problem statements to break down to different parts to be solved first and then their results to be combine at the final stage.

a) Traditional Image Classification: Fixed/Handcrafted Feature Extractor

b) Deep Learning: Hierarchical Features are learned and End-to-End Trainable

Figure 3. Difference between traditional machine learning and feedforward deep learning

1.2 Introduction of Deep Learning

As analyzed in Section 1.1, the deep learning method is a continuation from the traditional learning method. It shares with traditional methods in many basic concepts and processes. For example, both the traditional machine learning and deep learning methods are based on a simplest statistical tools of linear regression.

In linear regression, a set of N input- output pairs $\{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_N, \mathbf{y}_N)\}$ is given and their relation is to be modeled. A typical example is that they are the observation of some variables and their resulting output of a complex system respectively. With the assumption that there exists a linear function mapping between each $\mathbf{x}_i \in \mathbb{R}^Q$ to $\mathbf{y}_i \in \mathbb{R}^D$ (with $\mathbf{y}_i$ potentially mixed with observation noise), the model is then a linear transformation of the inputs: $\mathbf{f}(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}$, with $\mathbf{W}$ being some Q by D matrix over the real numbers and $\mathbf{b}$ a real vector with D elements. Different sets of parameters $\mathbf{W}$, $\mathbf{b}$ define different linear transformations. The objective is to find the set of parameters that would optimize an objective function, for example, minimizing the average squared error over the observed data: $\frac{1}{N} \sum_i \|\mathbf{y}_i - (\mathbf{W}\mathbf{x}_i + \mathbf{b})\|^2$.

This basic linear regression model can then be extended to model non-linear relation using linear regression on some non-linear basis function $\mathbf{f}(\mathbf{x})$. With the use of a set of fixed and mutually orthogonal basis functions, parameterized basis functions can be used. The most basic model in both traditional neural networks and deep learning models can be described as a hierarchy of these parametrized basis functions. The output of the parameterized basis functions composes a feature vector and they are then fed into a linear transformation.

With a serial connection of several different parametrized basis function and linear transformation, a feedforward neural network is formed. It is called a neural network as similar structure is observed in the neural system of mammals including human beings. This hierarchy can be directly used for regression while for classification a logistic function (a common function of S-shaped curve - sigmoid curve with equation $f(x) = \frac{L}{1+e^{-k(x-x_0)}}$) is composed at the end, which “flattens” the linear model’s output to obtain a probability vector.

To explain the feedforward neural network, a simple model with only one single hidden layer is analyzed. The generalization to multiple layers is straightforward. The input layer is denoted as $\mathbf{x}$ - a column vector with Q elements. It is transformed with an affine transformation to a column vector with K elements, which forms the feature vector of the hidden layer. The linear mapping is denoted by $\mathbf{W}_1$ (called “weight matrix”) and by $\mathbf{b}$ the translation (called “bias”) used to transform the input $\mathbf{x}$ to obtain $\mathbf{W}_1 \mathbf{x} + \mathbf{b}$. An element-wise non-linear function $\sigma(\cdot)$ (such as the rectified linear function - *ReLU* or hyperbolic tangent function - *TanH*) is then applied to the transformation output, resulting in a hidden layer with each element called a “network unit”.

A second linear transformation with weight matrix $\mathbf{W}_2$ then follows to map the hidden layer to the output layer of the neural network, a row vector with D elements. $\mathbf{W}_1$ is a $Q \times K$ matrix, $\mathbf{W}_2$ is a $K \times D$ matrix, and $\mathbf{b}$ is a K dimensional vector. This standard network output

$$\hat{\mathbf{y}} = \mathbf{W}_2 \sigma(\mathbf{W}_1 \mathbf{x} + \mathbf{b})$$

given some input $\mathbf{x}$. This equation describes the transformation by the single hidden layer neural network.

To use this network for regression, the Euclidean loss (or L_2 loss) is used

$$E^{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}}(\mathbf{X}, \mathbf{Y}) = \frac{1}{2N} \sum_{i=1}^N \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2 \quad (1.1)$$

where $\{\mathbf{y}_1, \dots, \mathbf{y}_N\}$ are N observed outputs – possibly with observation noise, and $\{\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_N\}$ are the outputs of the model with corresponding observed inputs $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$. By minimizing this loss w.r.t. $\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}$, a model can be achieved to work well on training data and also generalize well to unseen test data $\mathbf{X}_{test}, \mathbf{Y}_{test}$.

To use the single-layer model for classification, to predict the probability of $\mathbf{x}$ being classified with a label in the set $\{1, \dots, D\}$, the output of the model needs to pass through an element-wise softmax function to obtain normalized scores: $\hat{p}_d = \exp(\hat{y}_d) / (\sum_{d'} \exp(\hat{y}_{d'}))$. Taking the log of $\hat{p}_d$ (with d being the observed label) results in a softmax loss,

$$E^{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}}(\mathbf{X}, \mathbf{Y}) = -\frac{1}{N} \sum_{i=1}^N \log(\hat{p}_{i, d_i}) \quad (1.2)$$

where $d_i \in \{1, 2, \dots, D\}$ is the observed class for input i . The use of softmax function and loss is by design to train the neural network based on Maximum Likelihood Estimation.

A big challenge with the vanilla model above is overfitting – working well on the training set $\mathbf{X}, \mathbf{Y}$ while underperform on the test set $\mathbf{X}_{test}, \mathbf{Y}_{test}$. To solve this problem, it is a common practice to add in a regularization term during optimization. L_2 regularization is often used for each parameter weighted by some weight decays λ_i , resulting in a new loss function, which proves to result in models with better generalization ability. The regularization term penalizes large parameters. Each time some parameter is updated to become significantly large, it will increase the value of the cost function by the regularization term, and as a result, it will be penalized and updated to a small value.

$$\mathcal{L}(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) := E^{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}}(\mathbf{X}, \mathbf{Y}) + \lambda_1 \|\mathbf{W}_1\|^2 + \lambda_2 \|\mathbf{W}_2\|^2 + \lambda_3 \|\mathbf{b}\|^2 \quad (1.3)$$

The above single hidden layer neural network with the Euclidean loss is identical to a basis function regression model. Extending this simple neural network model to multiple layers results in a more expressive model. It is well known that “flat” basis function regression, for example the single-hidden-layer neural network, can model any function up to any given precision with a large enough number of basis functions. However, with a multi-layer hierarchy we can use much smaller models to achieve better results. In this sense hierarchical basis function models are more expressive than “flat” structure. This phenomenon is also summarized as “deeper is better than wider”. The success of deep learning networks is generally attributed to their expressiveness with manageable amount of parameters following this direction.

The simple network can also be extended to more specialized models, such as Convolutional neural networks (CNNs). Convolutional neural networks are popular deep learning tools for image processing, which can solve tasks that until recently were considered to lie beyond the reach. The model is made of a recursive application of convolution and pooling layers, followed by inner product layers at the end of the

network (simple neural network as described above). The current popular structure of CNNs are a progressive result based upon earlier works of Fukushima [183] with hand-picked features, Cresceptron of Weng et. al. [184] with learned features and Hebbina learning and the error backpropagation by Werbos [185].

A convolution layer is a linear transformation that preserves spatial information in the input image as illustrated in Figure 4. The input image (input layer) has a given height, width, and channels (three channels for RGB for example). Each kernel is convolved with each image patch (a single image patch is depicted under the left-most circle highlight) continuously or with a striding length to generate the output feature maps. In image processing, different kind of well-designed kernels has been long used for blurring, sharpening, embossing, edge detection, and more. It has been proven to be an effective way of extracting meaningful features from the input image. However, in deep learning, such kernels are not designed but learned through an end-to-end training process.

The convolution process (no striding) is defined as:

$$g_k(u, v) = \sum_{l=1}^c \omega_{k,l} * f_l(u, v) = \sum_{l=1}^c \sum_{i=-a}^a \sum_{j=-b}^b \omega_{k,l}(i, j) f_l(u - i, v - j) \quad (1.4)$$

where $g_k(u, v)$ is the k -th layer of filtered image (feature map), $f_l(u, v)$ is the l -th channel of original image, $\omega_{k,l}$ is the l -th channel of the k -th filter kernel. The kernel height and width are normally odd numbers $2a + 1$ and $2b + 1$ respectively. The parameters of the kernels $\omega_{k,l}$ are to be learned through the training process.

Figure 4. A convolution layer in a CNN (adapted from [179])

The convolution is normally combined with a ReLU (rectified linear unit) process to bring in non-linearity. The use of ReLU is a trade-off between the ease of training and the need of non-linearity for generalization of deep neural networks. Then it is followed by a pooling layer, which simply takes the output of a convolution layer and reduces its dimensionality (by taking the maximum of each (2, 2) block of pixels for example). Pooling is important to enable the level-up of features by focusing on the most important output from lower-level while increase the field of reception (in the input image) of the kernel in the next level. Based on these two layers, the deep learning neural networks are generally described by various depth of concatenating convolution with ReLU layers and pooling layers alternatively.

For classification, the common practice is to add in a couple of FC (fully connected) layers to work on the result feature maps to learn a complex “mapping” function before the softmax layer. Thus a typical deep learning network for classification can be illustrated as Figure 5. The layers before the FC layers can be viewed as feature extraction layers, whose purpose is to learn the deep feature, which is within a subspace with much a lower dimension than the original image data space. The FC layers and the softmax layer is commonly viewed as a classification part which learns the mapping from the deep feature space to the final class labels. The full model is then trained in an end-to-to end manner.

Figure 5. A typical structure for deep learning based classification model

The CNN networks perform well in the classification task exploiting the feature locality. This is important because images are the 2D projection of a real project in 3D space. With pooling, the translation-invariant locality is exploited at different granularities, therefore being able to model hierarchically higher-level features. By examining the activations of different layers, it is found that in deeper layers, the features learned in previous layers come together and make shapes which often have semantic meaning. Aided by the increase of amount of data available and the computing power of hardware such as GPU (Graphic Processing Unit), the performance of deep learning method progress rapidly in recent years.

1.3 Limitation of Deep Learning Methods

The idea of learning concepts from input images with multi-layer (more than three layers) neural networks and back propagation method have been studied [183][184][185] during the eighties and nineties. However, the concept of deep learning only becomes popular with the introduction of trainable deep learning models about 10 years ago [1][2]. Ever since then, deep learning has pushed the performance of image classification in many challenging tasks such as MNIST (Modified National Institute of Standards and Technology) and ImageNet to a high level well beyond traditional methods and even comparable or superior to human beings (This performance comparison is done with special task setting of ImageNet. A more detailed comparison can be found in [186] and the comparison with a different task setting – with image distortions - can be found in [187]).

However, since the deep learning method is data-driven, it has problem on “unseen” or “unusual data” though it is reported to generalize well for data lying on the manifolds formed by the training data. The first kind of these examples are unusual data not covered by the training dataset, e.g. ImageNet. When the image below is input into one of the state-of-the-art deep learning neural networks (<https://imaggga.com/auto-tagging-demo>), the recognition result gives “rattan” with 37.71% confidence, “wicker” with 30.53% confidence and “switch” with 29.9% confidence respectively.

Figure 6. Example of an unusual image (image credit: Ian Goodfellow)

Besides unusual images, artificial images that are totally different from natural images are also generated by various methods to fool the deep learning models. Figure 7 shows two unrecognizable (to human beings) images [140] that are recognized by DNN as “cheetah” and “baseball” respectively with over 99.6% confidence.

Figure 7. Unrecognizable objects to human being (images from [140])

Another type of images are based on tampering of the natural images. The result is recognizable by human being but leads to wrong recognition result by DNN. These attacks can be effective even in physical world. Figure 8 shows a slightly tampered stop sign that is recognized by a DNN network as “speed limit 45” in the physical world with over 66.67% success rate [76].

Figure 8. Physical attack by slightly changing the normal input (image from [76])

In addition to these attacks that are noticeable by human beings, most existing methods are focused on attacks that are named as “adversarial examples” [3], which are natural images added with small intended variations imperceptible to human beings. These adversarial examples cause the neural networks to completely change its prediction and reports high confidence on the wrong prediction. These attacks attract more attention as they can escape from human inspection and only mislead the network decision, which may lead to serious damages. More details of adversarial examples are given in the section 1.4.

Moreover, the same adversarial example is found to be able to fool multiple networks with different structures. Many examples have shown that this robustness vulnerability can be a big problem for the real applications of deep learning networks, especially for critical areas such as autonomous driving and disease diagnosis.

This vulnerability illustrates a deeper problem of the existing deep learning method: the lack of uncertainty measurement. Actually, deep learning models are often viewed as deterministic functions and as a result viewed as operating in a very different settings to the probabilistic models which possesses uncertainty information. The “confidence” value attached to the softmax layer output is not actually a reliable estimator of confidence [179]. Adversarial examples work as good counter cases to illustrate the weakness.

A lot of research has been done with the aim to provide a solution to guarantee the robustness of the decisions made by deep learning networks, with the possible presence

of adversarial examples. However, it is revealed that the difficulty in finding such a solution lies in the following aspects: 1) the high dimensionality of the input space; 2) the huge amount of possible adversarial examples; 3) the linearity in working models which makes the deep network to be easily trained; and 4) the compromise between inference accuracy and robustness to adversarial examples. All existing defense methods are of limited effectiveness and can be defeated with a new adversarial attack method.

Thus it is valuable to solve the problem in another direction, i.e. to provide a fault detection mechanism for the existing deep learning networks. The inhibition of possible wrong answers is much better than directly output it with high confidence, especially for critical applications such as autonomous driving or medical diagnosis.

The challenge lies in the fact that there is no explicit “model” in the common deep learning framework. So it is difficult for the network to know what it does not know besides the current measure of confidence – the softmax output, which has been proven to be ineffective for adversarial examples. “NULL” label has been studied in [86] as a method to block the transferability of adversarial examples, but it is still within the framework of adversarial training, which still depends on the training dataset of adversarial examples and is impractical.

Inspired by the recognition process of human beings, which is not affected by the adversarial examples, this study is the first to propose a high-level feedback mechanism in the inference stage to detect the system’s own fault in its recognition proposal. The DBFR (Detection Based on Feedback and Reconstruction) method makes no change to the existing network models and circumvents the limitations of these models with the computed result as a high-level feedback to a reconstruction process. The reconstruction process works as an “verification model” of the deep learning network to validate whether the proposal is supported by this “verification model”, which is built upon the learning on the training data. Experiment results show that with the feedback mechanism added onto existing methods, the adversarial attacks can be defended much more efficiently with minimum computation cost. This study paves a new way for future research towards a final solution for the adversarial attack problem.

1.4 Adversarial Examples

1.4.1 Discovery of Adversarial Examples

Deep learning is a very useful tool that drives the recent progress in a lot of perception tasks, especially in the areas of image recognition and classification. With the introduction of deep Convolutional Neural Network (CNN) models and applicable training methods, it seems that any task of recognition and classification can be easily achieved with a deep neural network and a large enough set of training data. The general sentiment is that although there is still a gap in understanding the deep learning method, image recognition and classification is practically a solved problem, with the deep CNN model being able to learn the hierarchical information of any “label” and generalize well on unseen images, with enough training data.

However, when Szegedy [3] discovered the existence of “adversarial examples”, one of the “intriguing properties” of neural networks, the awareness of possible setbacks of deep learning model in real applications is suddenly awakened. As revealed in [3], with human-crafted variations small and imperceptible to human beings added onto the natural images, the well-trained deep neural networks give wrong predictions with high confidence. Figure 9 illustrates adversarial examples generated by methods in [1] and applied to GoogLeNet on the ImageNet dataset. This finding illustrates that although the existing deep learning models work well on natural images, its robustness is very weak and they are exposed to possible adversarial attacks without the notice of human beings in real applications. Further research have shown that this vulnerability is not only for deep learning networks, but also for most machine learning methods [3], even to time-limited humans [6].

Figure 9. Fast adversarial examples applied to GoogLeNet (images from [1])

After [3], many works have been done in the areas of both “adversarial attack” (how to generate adversarial examples) and “adversarial defense” (how to detect the adversarial examples or improve the model’s robustness so that it will not be fooled by such examples). Overall, the ease of optimization for deep learning networks has come at the cost of models that are easily misled. For current common networks that have been carefully designed to be sufficiently linear so that gradient-based optimization can work well, adversarial examples are of huge number, transferrable across many models, difficult to be detected and can be effective even in real world applications; the existing defense is of limited effectiveness, computationally impractical and at the cost of reduced accuracy.

1.4.2 For Critical Applications such as Autonomous Driving

Without an effective solution, adversarial attacks can lead to disastrous results in critical real-world applications such as Autonomous Driving and Disease Diagnosis. For example, it has been shown that a slighted modified stop sign can be recognized as speed limit sign in the physical world [76]. The automatic lane changing functions of a commercial Tesla Model-S car can be triggered by a few small patches pasted on the road [4] as illustrated in Figure 10. In medical applications, [5] has shown that adversarial examples can fool the deep networks for the diagnosis of diseases with the input images of Fundoscopy, Chest X-Ray and Dermoscopy.

Figure 10. Fake lane model in physical world (image from [4])

These examples illustrate the necessity to solve this vulnerability of deep neural networks against possible adversarial attacks before they can be applied in real applications. One component of the full solution is to make the neural network able to report its own fault when the decision is not supported by the training data.

1.5 Problems of Existing Defense Methods and Possible Directions

Detailed review and analysis of the existing defenses is covered in Chapter 2 and only a brief summary of their problems is given here.

The first approach towards the defense is based on detection. The reasoning is that the adversarial examples are lying on manifolds different from the natural images and if they can be detected, then special treatment can be done to solve the problem. However, It is shown that the detection of adversarial examples is difficult as the distribution of such examples are dense and there seems to be no intrinsic properties for adversarial examples that is different from natural examples.

For example, ten detection methods based on various approaches including secondary classification based detection, PCA (Principal Component Analysis) based detection, distributional detection and normalization detection were all defeated by constructing new loss functions [150].

The second approach is based on the pre-processing of the adversarial examples so that the noise-like perturbations can be removed and the processed images will come back to the manifold of the natural images. Methods based on filtering, auto-encoder noise removal or GAN (Generative Adversarial Network) have been proposed with certain level of success. However, to solve the problem completely requires the coverage of the sub-space formed by all adversarial examples distributions, which is impractical.

Most working defense methods are based on adversarial training to improve the robustness of the network model near manifold formed by the natural images. There are mainly three approaches: 1) to include adversarial examples into the training

process; 2) to take robustness into account for the loss design and 3) to make changes to the network model with the consideration of robustness. However, [7][8] showed that adversarial examples in machine learning are likely not due to information-theoretic limitations, but rather it could be due to computational complexity. It has been shown from statistical aspects [9] that the sample complexity of robust training can be significantly larger than that of standard training.

Other methods of modifying the network structure such as changing the network structure, using network add-ons or GAN based methods have been proposed. However, limited success is seen and all the methods are soon defeated by improved attacks with new loss function designs [168].

Even with the computing difficulty and limited success of these approaches, recent research demonstrates the existence of trade-off between accuracy and robustness, both experimentally [10] and theoretically [11], under current deep learning framework.

However, this is in great contrast to the perception ability of human beings, which is not fooled by the adversarial examples and there seems to be no compromise in the accuracy for robustness. One hypothesis is that though the adversarial attack is a difficult problem for early/low-level vision, it can be easily solved by the higher-level vision of human beings. This illustrates that the solution to the adversarial example problem may not be improving the robustness of the deep-learning based models but building different mechanisms.

As pointed out by [6], two major high level computational differences between CNNs and the human brains are: 1) the brain does not treat a scene as a single static image, but actively explore it with saccades; and 2) while most CNNs are of feedforward architecture, the visual cortex has many times more feedback than feedforward connections, as well as extensive recurrently dynamics.

Foveation based method [12], applying the CNN in different image regions, is an attempt to solve the problem simulating the saccades in human recognition process and its result shows some effectiveness in alleviating the adversarial effects. However, it is shown to be with limited effectiveness and can be defeated with stronger attacks.

It is worthy to note that Cresceptron [184] uses feedback to reconstruct the edges of input images for segmentation after recognition and a Developmental Network (DN) [186] uses feedback mechanisms almost everywhere as top-down contexts/states for every neuron except the last muscle neurons.

This study is the first to focus on the feedback mechanism to focused on solving the problem of adversarial attacks for existing CNNs. The proposed method is called Detection based on high-level Feedback and Reconstruction (DBFR). This solution is based on the observation that human beings do not make recognition passively, but actively make hypotheses, verifying a hypothesis and changing the attention if needed. This type of high-level feedback is critical for a person to focus on high-level knowledge of the scene and ignore perturbations of less significance, for example adversarial examples. The verification process also makes the person to possibly realize that a reliable recognition is not possible as the input is new or difficult. This feedback is exactly what is missing in most existing solutions.

1.6 Contributions of This Study

In this study, we would like to first analyze the limitations of existing defense methods against the adversarial attacks. In a nutshell, no published methods have been shown to be robust to all adversarial attacks because the adversarial examples are abundant, difficult to be detected effectively and the adversarial training based on existing framework is impractical.

Inspired by the human recognition process, a novel method - DBFR is proposed to include a high-level feedback loop into the existing inference pipeline for self-fault detection. With this mechanism in place, the deep-learning based recognition system will inhibit its output when it is not supported by the training data, which includes the case for adversarial examples and “unseen” data. Notes shall be taken this study is limited to the input images with isolated single object. For images of cluttered scene, attention mechanism is needed and are not covered in this study.

Figure 11 shows the DBFR method in comparison to existing ones. Figure 11(a) illustrates the traditional deep learning framework which shows robustness to clean images but is highly vulnerable to adversarial examples. Figure 11(b) illustrates the existing adversarial training framework which can increase the robustness of the target model with enhanced training data which is of impractical amount. Figure 11(c) shows the DBFR feedback-based fault detection method to “recognize” the mistake based on a high-level feedback guided reconstruction process.

Figure 11. DBFR framework in comparison with existing methods

The DBFR framework has been implemented with the ResNet-50 and tested with several attack methods on realistic datasets. By comparing the original image and the reconstructed images for both clean data and that of a typical adversarial example, it shows that the reconstruction process is a good reflection of whether the classifying result is supported by the training data. With a natural test image, the reconstruction result is very similar to the input and with the adversarial example, the reconstructed image based on the wrong “class ID” is quite different from the input image.

“Unseen” or unusual images are also input into the DBFR model. The reconstructed images are again significantly different from the input image, though the behavior is slightly different from the results on the adversarial images.

Based on this reconstruction difference, the fault in the classification result can be detected, whether for adversarial examples or unseen/unusual input. The experiment against adversarial attack shows that both adversarial examples generated by various methods and the unseen/unusual images can be detected effectively. Details of the results are covered in Chapter 4.

The main contribution of this thesis is the proposal of DBFR (detection based on feedback and reconstruction) framework to overcome the problem of blindly outputting high confidence with the input of adversarial examples or unseen images, as exhibited by most existing deep learning based recognition systems. This framework is implemented as an auxiliary part of existing system without any changes of the recognition system itself. However, with this framework implemented, the system is able to detect the situation when it is applied to adversarial examples or unseen/usual inputs.

This thesis also reveals the importance of feedback loop as one of the main differences in the perception process between a deep learning network and a human being. Instead of increasing the training data size by brute force, this thesis proposes a novel solution based on feedback in the inference stage, which is much more effective and computing efficient.

With a reconstruction module, the proposed framework also provides an intuitive representation for the decision-making mechanism of a deep learning network, which could provide insights to researchers for both understanding the existing models and searching for a robust image recognition solution.

This study provides a novel direction towards the final solution for a robust image recognition system based on deep learning. Only with such a solution in place can these systems be reliably applied in the real world.

1.7 Chapter Description

This thesis contains five chapters.

Chapter 1 describes the importance of fault detection for image recognition systems based on deep learning. With a brief introduction of deep learning and adversarial examples, the limitations of deep-learning based image recognition systems are illustrated. Based on an analysis of the failure of existing methods in contrast to the robustness of human recognition system, a new method based on high level feedback and a reconstruction process is proposed.

Chapter 2 contains a literature survey on existing adversarial attack and defense methods. With a systematic categorization for attacks and defenses respectively, existing methods for both sides are analyzed thoroughly. The limitations of existing defense methods are pointed out and the analysis leads to the proposal of the new Detection based on Feedback and Reconstruction method.

Chapter 3 describes the methodology of the proposed method. With a proper formulation of the fault detection problem, the network architecture, training process and the quantitative measurement of similarity between the reconstructed and input images is introduced in detail. Without any modification of the existing neural network, a reconstruction network is built and trained with the same set of training data. Using the proper training loss, the reconstruction network efficiently learns features for both intra-class similarities and the inter-class differences for the reconstruction. The reconstructed image guided by the recognition feedback serves as an “internal model”

for the decision process. By comparing the reconstructed image and the original input, whether the recognition result is supported by the training data is detected. This serves as an effective fault detection mechanism for the recognition system.

Chapter 4 describes the experimental results of the proposed method. Using two examples of publicly available datasets and a commonly used classification network structure, the reconstruction network is trained following the proposed methodology in the previous chapter. Several typical attack methods including the reportedly strongest C&W attack are used to generate adversarial examples for the test. With a proper thresholding of Structural Similarity Index (SSIM) values comparing the reconstructed and input images, both adversarial examples and unusual images are detected effectively. A thorough comparison between this method and other existing defense methods is done and the result shows that the proposed method is significantly more effective.

Chapter 5 contains the discussions, conclusions and future work. Several interesting findings in the experiments are discussed and the future work towards a full solution for robust image recognition is highlighted. This study paves the way to the final solution for a robust image recognition system that uses deep learning methods. Only with such a solution in place can these systems be reliably applied in the real world.

The problem of Adversarial Examples is not unique to deep learning but also to conventional machine learning methods. This problem becomes a hot topic recently only because of deep learning's excellent performance (much better than conventional machine learning and up to human level) on natural images, in contrast to its vulnerability of being fooled by adding in minute noises. This vulnerability is of great concern because it exists in every kind of models for all applications and has been shown to be effective in the physical world. The adversarial examples are also transferrable across different models and their distribution in image space is dense.

The question of whether adversarial examples are inevitable is ill-posed. Clearly, any classification problem has a fundamental limit on robustness to adversarial attacks that cannot be escaped by any classifier. However, in most literature with deep learning methods, the problem is confined to the robustness with a limitation on the adversarial examples to be "imperceptible by human being", either explicitly or implicitly. Various studies have shown that this limit can be quantized as l_1 -norm or l_∞ -norm less than 15 [12] for 8-bit digital images. Although without proof, the common belief is that there must be a solution existing in this case, as human vision - which is also based on a neural network - is not fooled by these adversarial examples.

There are some theoretical studies [37] on the robustness for general classification problems. Analysis shows that the limits depend not only on fundamental properties of the dataset, but also on the strength of the adversary and the metric used to measure perturbations. Unfortunately, it is impossible to know the exact properties of real-world image distributions or the resulting fundamental limits of adversarial training for specific datasets.

Regardless of difficulties in theoretical study, a lot of application research have been done in both the areas of generating adversarial examples and defense against the adversarial attacks. The attacks and defense is inherently a game played by both sides. This chapter provides a detailed latest survey of the major methods in both camps based on different categorization methods respectively. The current observation is that once

a defense method is proposed, there will be a new attacking method to defeat it soon later.

There have been some excellent review works for this area [164][165][166][167]. They provided both reference to the related studies and insights into the development of this field. However, there are a lot of new theoretical studies and practical methods not covered in these works and in this review, efforts have been put to include all the latest methods with a focus on those that lead to the new solution DBFR method.

This chapter is organized as follows. The review of adversarial problem in conventional machine learning is first provided. Then this problem is more closely examined in the context of deep learning. The influence and characteristics of adversarial examples and the cause for its existence are both reviewed in detail. Section 2.3 provides the categorization methods for both attack and defense literature, which serves as an index organizing all the different methods proposed. Section 2.4 reviews the various methods for adversarial attacks, which are continuously proposed to challenge the existing deep learning methods and proposed defense. Section 2.5 reviews the defense methods proposed so far and their effects on the attacks. It is a common phenomenon that these proposed methods are defeated by another attack method designed against them. Some promising methods, especially those related to the DBFR method is highlighted at the end of this section. The last section provides a summary of the literature survey and confirms the methodology of DBFR framework.

2.1 Adversarial Examples in Conventional Machine Learning

Until recently, machine learning was difficult and the performance is far from that of human beings. Adversarial examples were not interesting to most researchers because mistakes were the rule, not the exception.

Nevertheless, adversarial examples have been discussed in the context of security problem for machine learning since decades ago. Barreno et al. presented an initial investigation on the security problems of machine learning [13], [19]. At that time, the main targets are conventional machine learning-based systems with handcrafted features, such as spam filters, intrusion detection, biometric authentication, fraud

detection, etc. [13]. A typical example at that time is that spam emails are often modified by adding characters to avoid detection by anti-spam algorithms [14][15][16]. They categorized attacking against traditional machine learning system in three aspects: 1) influence: whether attacks can access and poison the training data; 2) security violation: whether an adversarial example belongs to false positive or false negative; 3) specificity: whether an attack is targeted at a particular instance or a class of systems.

The adversarial example problem is formulated as a game between adversary and classifier by Dalvi et al. [14]. Both the adversary and the classifier are sensitive to cost and the attack and defense forms an iterative game. This formulation is very powerful and can be also used to describe the current research in this area for deep learning. A gradient-based approach is used to generate adversarial examples for linear classifier, support vector machines (SVM) and a conventional neural network by Biggio et al. [17]. MNIST dataset is used to evaluate the effectiveness of the adversarial examples. However, there is less constraint on the adversary strength thus a human could easily distinguish the adversarial digit images. Several proactive defenses and reactive approaches are also reviewed and discussed by Biggio et al. in the context to improve the security of machine learning models [18].

2.2 Adversarial Examples in Deep Learning

In principle, there is no fundamental difference between the mechanism of adversarial examples for traditional machine learning and those for deep learning. [20] provided a comprehensive overview of security issues in machine learning and those recent findings in deep learning and established a unifying threat model. A “no free lunch” theorem was introduced: there is a tradeoff between accuracy and robustness.

Traditionally, most machine learning work has taken place in the context of the I.I.D (independent identically distributed) assumptions. It means that all of the examples in the training and test set are generated independently from each other and are all drawn from the same data-generating distribution. However, in the context of adversarial examples in deep learning, there are two main relevant violations of the I.I.D assumptions:

1. The test data is not drawn from the same distribution as the training data. The attacker intentionally shifts the distribution at test time toward unusual inputs such as the adversarial stop sign that will be processed incorrectly;
2. The test examples are not necessarily drawn independently from each other. A real attacker can search for a single input that causes a mistake, and then send that input repeatedly, every time they interact with the system.

In conventional ML (machine learning), both attacking and defending methods paid great attention to features, giving less attention to the impact on humans. This leads to two major differences in adversarial generation: 1) adversarial examples in conventional machine learning require knowledge of feature extraction, while deep learning usually needs only raw data input; 2) in conventional machine learning the adversarial examples are discussed without constraint on their strengths, while in the context of deep learning, it is commonly assumed that the difference between adversarial examples and natural images are imperceptible.

With all these differences, the methods of generating adversarial examples and corresponding defense in the context of deep learning rarely resembles those discussed in traditional machine learning, although in principle there is no difference between their working mechanism.

2.2.1. Adversarial Examples for Different Applications

Adversarial examples have been discovered in all the application areas of deep learning methods. These include but not confined to:

- Classification, segmentation and detection, face recognition and others in Computer Vision;
- Reading comprehension, text classification and machine translation in Natural Language Processing;
- Malware detection and intrusion detection in Cyber Security;
- Road sign recognition, face recognition and machine vision in Physical World.

They are also found for all neural network models including: Feedforward CNN Models, Autoencoders, Generative Models, Recurrent Neural Networks and Deep Reinforcement Learning.

In other words, there is no model in any application that can be free from adversarial attacks.

However, most of the current research work on both attack and defense focus on one typical problem of image classification in Computer Vision. This study will focus on the same, though the proposed idea and defense method can be extended to all the other applications. Only in this section, the adversarial examples for other application areas will be briefly introduced to give an overview of the problem.

Applications other than Classification in Computer Vision

Detection and Segmentation

Xie et al. [21] extended the adversarial attacks to more difficult problems of semantic image segmentation and object detection. The authors observe that both segmentation and detection are based on the classification of multiple objects in an image. Based on this idea, a new algorithm called Dense Adversary Generation (DAG) is proposed. This algorithm produces large categories of adversarial samples suitable for segmentation and detection in the most advanced deep models. The authors also pointed out that adversarial disturbances can be transmitted through networks based on different architectures with different training data, even for different kinds of identification tasks. Figure 12 shows a typical example of a network spoofing that uses the methods in [21] for image segmentation and detection.

Figure 12. Adversarial example for semantic segmentation and object detection [21]

Face Recognition

Deep-neural-network based Face Recognition System (FRS) and Face Detection System (FDS) have already been widely deployed in commercial products due to their high performance. [41] first provided a design of eyeglass frames to attack a deep-neural-network based FRS [47], which composes 11 blocks with 38 layers and one triplet loss function for feature embedding. Based on the triplet loss function, [41] designed a *softmaxloss* function:

$$J(x) = -\log \left(\frac{e^{\langle h_c, f(x) \rangle}}{\sum_{c=1}^m e^{\langle h_c, f(x) \rangle}} \right) \quad (2.1)$$

where h_c is a one-hot vector of class c , $\langle \cdot, \cdot \rangle$ denotes inner product. Then they used L-BFGS Attack to generate adversarial examples.

In the attack, the mean perturbation of a pixel that overlaid the face was 1.7, with standard deviation 0.93. An example is shown in Figure 13 and test shows a 100% success rate on three typical deep neural networks.

Figure 13. A dodging attack by perturbing an entire face. Left: an original image of actress Eva Longoria (by Richard Sandoval / CC BY-SA / cropped from <https://goo.gl/7QUvRq>). Middle: A perturbed image for dodging. Right: The applied perturbation, after multiplying the absolute value of pixels' channels $\times 20$ [41]

In a further step, [41] implemented adversarial eyeglass frames to achieve attack in the physical world, which will be reviewed in more details in the upcoming section “Physical World Attacks”.

Other Tasks in Computer Vision

Actually, adversarial attacks have been implemented in all high-level computer vision tasks such as image caption [153] and visual question answer (VQA) [154].

Hongge Chen and et al [153] propose Show-and-Fool, a novel algorithm for crafting adversarial examples in neural image captioning, to study the robustness of language grounding to adversarial perturbations in machine vision and perception. The proposed algorithm provides two evaluation approaches, which check whether neural image captioning systems can be misled to output some randomly chosen captions or keywords. The experiments show that the algorithm can successfully craft visually-similar adversarial examples with randomly targeted captions or keywords, and the adversarial examples can be made highly transferable to other image captioning systems.

In [154], the first extensive study on adversarial examples for Visual Question Answer (VQA) systems was conducted. In particular, focus was put on generating targeted adversarial examples for a VQA system while the target is considered to be a question-answer pair. The evaluation shows that the success rate of whether a targeted

adversarial example can be generated is mostly dependent on the choice of the target question-answer pair, and less on the choice of images to which the question refers. The language prior phenomenon of a VQA model (when given a particular question, i.e., the target question, the model is likely to take the question into consideration, and so it always responds with an answer that makes sense to the question) is also reported and it can explain why targeted adversarial examples are hard to generate for some question-answer targets.

Recent studies have found that the adversarial attacks can also be applied to more basic and low-level tasks such as edge detection [155]. In [155], it's shown that adding small perturbations to an image causes HED, a CNN-based edge detection model, to fail to locate edges, to detect nonexistent edges, and even to hallucinate arbitrary configurations of edges. More surprisingly, these adversarial examples transfer to other CNN-based vision models for image classification and segmentation.

Figure 14. Adding a small perturbation (right) to an image causes a state-of-the-art edge detection model to produce a contrived pattern. The same perturbation causes a VGG16 model to misclassify the image [155]

Natural Language Processing (NLP)

Reading Comprehension

Many tasks in natural language processing can be attacked by adversarial examples. People usually generate adversarial examples by adding/deleting words in sentences. The task of reading comprehension (a.k.a. question answering) is to read paragraphs and answer questions about the paragraphs. To generate adversarial examples that are consistent with the correct answer and do not confuse human, Jia and Liang added distracting (adversarial) sentences to the end of a paragraph [48]. They found that models for the reading comprehension task are over-stable instead of oversensitivity, which means deep learning models cannot tell the subtle but critical difference in the paragraphs.

Article: Super Bowl 50

Paragraph: *“Peyton Manning became the first quarterback ever to lead two different teams to multiple Super Bowls. He is also the oldest quarterback ever to play in a Super Bowl at age 39. The past record was held by John Elway, who led the Broncos to victory in Super Bowl XXXIII at age 38 and is currently Denver’s Executive Vice President of Football Operations and General Manager. Quarterback Jeff Dean had jersey number 37 in Champ Bowl XXXIV.”*

Question: *“What is the name of the quarterback who was 38 in Super Bowl XXXIII?”*

Original Prediction: John Elway

Prediction under adversary: Jeff Dean

Figure 15. An example from the SQuAD dataset. The BiDAF Ensemble model originally gets the answer correct, but is fooled by the addition of an adversarial distracting sentence (in blue) [48]

Two sorts of techniques are proposed to generate adversarial examples: 1) adding grammatical sentences similar to the inquiry but not contradictory to the correct answer (AddSent); 2) including a sentence with arbitrary English words (AddAny). [48] successfully fooled all the models (sixteen models) they tested on Stanford Question

Answering Dataset (SQuAD) [49]. The adversarial examples also have the capability of transferability and cannot be improved by adversarial training. However, the adversarial sentences require human input to fix the errors in the sentences.

Text Classification

J. Li and et al [50] aimed to fool a deep learning-based sentiment classifier by removing the minimum subset of words in the given text. Reinforcement learning (RL) was used to find an approximate subset, where the reward function was proposed as $\frac{1}{\|D\|}$ when the sentiment label changes, and 0 otherwise. $\|D\|$ denotes the number of words in set D and the removal of all words in D changes the label. The reward function also included a regularizer to make a sentence contiguous. Sample results are presented in Table 1. The reinforcement learning model identifies aspect-specific sentiment phrases, providing a rationale for why the sentiment model makes a certain decision. By comparing Table 1a with Table 1b, it can be seen that the reinforcement model trained based on the memory based model offers better interpretability than the one trained based on Long Short-Term Memory networks (LSTM). The latter model not only requires erasing more words to flip the model’s decision, but also sometimes deletes passages describing different aspects or overall sentiment.

(1) clean updated room. friendly efficient staff. rate was too high 199 plus they charged 10 day for internet access in the room.
(2) the location is fantastic. the staff are helpful and service oriented. sleeping rooms meeting rooms and public lavatories not cleaned on a daily basis. the hotel seems a bit old and a bit tired overall. trolley noise outside can go into the wee hours. if you get a great price for a few nights this hotel may be a good choice. breakfast is very nice remember if you just stick to the cold buffet it is cheaper.
(3) location is nice. but goes from bad to worse once you walk through the door. staff very surly and unhelpful. room and hallway had a very strange smell. rooms very run down. so bad that i checked out immediately and went to another hotel. intercontinental chain should be ashamed.
(4) i took my daughter and her step sister to see a show at webster hall. it is so overpriced i 'm in awe. i felt safe. the rooms were tiny. lots of street noise all night from the partiers at the ale house below.

a) Examples of minimum set of erased words based on Bi-LSTM model

(1) clean updated room. friendly efficient staff. rate was too high 199 plus they charged 10 day for internet access in the room. the location is fantastic. the staff are helpful and service oriented.
(2) sleeping rooms meeting rooms and public lavatories not cleaned on a daily basis. the hotel seems a bit old and a bit tired overall. trolley noise outside can go into the wee hours. if you get a great price for a few nights this hotel may be a good choice . breakfast is very nice remember if you just stick to the cold buffet it is cheaper.
(3) location is nice. but goes from bad to worse once you walk through the door. staff very surly and unhelpful. room and hallway had a very strange smell. rooms very run down. so bad that i checked out immediately and went to another hotel. intercontinental chain should be ashamed.
(4) i took my daughter and her step sister to see a show at webster hall. it is so overpriced i 'm in awe. i felt safe. the rooms were tiny. lots of street noise all night from the partiers at the ale house below.

b) Examples of minimum set of erased words based on memory-network model

Table 1: Examples of minimal set of erased words to change the model decision for different aspects based on different models. Each of the colors represents a specific aspect, i.e., rooms, service, value and location [50]

The changes in [48], [50] can easily be recognized by humans. More natural adversarial examples for texture data was proposed by Natural GAN [51]. A framework as illustrated in Figure 16 was proposed to generate natural and legible adversarial examples that lie on the data manifold, by searching in the semantic space of dense and continuous data representation, utilizing the recent advances in generative adversarial networks. In order to represent natural instances of the domain, a Wasserstein GAN (WGAN) is first trained on corpus X to provide a generator $\mathcal{G}_\theta$ that maps random dense vectors z to samples x from the domain of X . A matching inverter I is then separately trained to map data instances to corresponding dense representations. As shown in Figure 16, the reconstruction error of x and the divergence between sampled z and $I_\gamma(\mathcal{G}_\theta(z))$ is to be minimized together to encourage the latent space to be normally distributed.

Figure 16. Training Architecture with a GAN and an Inverter. Loss of the inverter combines reconstruction error of x with divergence between Gaussian distribution z and $I_\gamma(\mathcal{G}_\theta(z))$ [51]

The generated adversaries were presented to demonstrate the potential of the proposed approach for black-box classifiers for a wide range of applications such as image classification, textual entailment (a task designed to evaluate common-sense reasoning for language, requiring both natural language understanding and logical inferences for text snippets), and machine translation. Table 2 shows the adversaries generated to mislead the machine translation.

Source Sentence (English)	Generated Translation (German)
s : A man and woman sitting on the sidewalk.	Ein Mann und eine Frau, die auf dem Bürgersteig sitzen .
s' : A man and woman stand on the bench.	Ein Mann und eine Frau stehen auf der Bank.

Table 2. Machine Translation. “Adversary” that introduces the word “stehen” into the translation [51]

Cyber Security

Malware Detection

Deep learning has been used in static and behavioral-based malware detection due to its capability of detecting zeroday malware [52][53][54][55]. Recent studies generated adversarial malware samples to evade deep learning-based malware detection [56][57][58][59].

K. Grosse and et al [56] adapted Jacobian-based Saliency Map Attack (JSMA) method to attack Android malware detection model. W. Xu and et al [59] evaded two Portable Document Format (PDF) malware classifier, PDFrate and Hidost, by modifying PDF. [59] parsed the PDF file and changed its object structure using genetic programming. The adversarial PDF file was then packed with new objects. The method is able to automatically find evasive variants for both classifiers for all of the 500 malicious seeds in the study. These results suggest a general method for evaluating classifiers used in security applications, and raise serious doubts about the effectiveness of classifiers based on superficial features in the presence of adversaries.

[60] used GAN to generate adversarial domain names to evade detection of domain generation algorithms. [58] proposed a GAN based algorithm, MalGan, to generate malware examples and evade black-box detection. [58] used a substitute detector to simulate the real detector and leveraged the transferability of adversarial examples to attack the real detector. MalGan was evaluated by 180K programs with API features. However, [58] required the knowledge of features used in the model. [57] used a large number of features (2,350) to cover the required feature space of portable executable (PE) files. The features included PE header metadata, section metadata, import & export table metadata. [56] also defined several modifications to generate malware evading deep learning detection. The solution was trained by reinforcement learning, where the evasion rate is considered as a reward.

Physical World Attacks

Although many adversarial examples are generated in “laboratory settings”, it should be noted that they form the basis of the real-world attacks, and almost each one of them has the potential to significantly affect deep learning in practice. Some researchers focus on the threats in real physical worlds and they illustrate the potential damage that the adversarial examples can do.

Kurakin et al. [61] first demonstrated that the threat of adversarial attack also exists in the physical world. To prove it, they printed out adversarial images and took snapshots with the phone’s camera. These images were input into the TensorFlow Camera Demo app using the Inception model of Google for object classification. The results showed that these images were largely misclassified even when displayed on screen and captured by the camera. As is shown in Figure 17, they took a clean image from the dataset (a) and used it to generate adversarial images with various sizes of adversarial perturbation. Then, they printed clean and adversarial images and used the TensorFlow Camera Demo app to classify them. A clean image (b) is recognized correctly as a “washer” when perceived through the camera, while adversarial images (c) and (d) are misclassified. The images were printed and the TensorFlow Camera Demo app was used to classify them. The images also show network confidence in the range [0,1] for each image.

Figure 17. Example of adversarial attack on mobile phone cameras [61]

Eykholt et al. [76] proposed a white-box adversarial sample generation method to attack their own trained road sign recognition models, including LISA-CNN models used LISA [77], a U.S. traffic sign dataset containing 47 different road signs, and GTSRB-CNN models, which trained on the German Traffic Sign Recognition Benchmark (GTSRB) [78]. They proposed two effective kinds of disturbance installation methods for road sign recognition scenarios, i.e., posters and stickers, as shown in Figure 18. They followed Sharif et al. [41] in constructing the loss function and took into account the printability and location limitations. Their assessment showed that they had achieved a 100% success rate in the poster installation driving test.

Distance/Angle	Subtle Poster	Subtle Poster Right Turn	Camouflage Graffiti	Camouflage Art (LISA-CNN)	Camouflage Art (GTSRB-CNN)
5' 0°
5' 15°
10' 0°
10' 30°
40' 0°
Targeted-Attack Success	100%	73.33%	66.67%	100%	80%

Figure 18. The effect of generating and evaluating physical adversarial perturbations in the real world [76]

Gu et al. [79] demonstrated an attack framework called BadNets in a real-world scenario, where they created a street sign classifier that recognizes a stop sign as a speed limit when a special sticker was added to the stop sign. Their most challenging attack was in a transfer learning setting; the attack setup is shown in Figure 19. In this setting, a BadNet trained on U.S. traffic signs was downloaded by a user who unwittingly uses the BadNet to train a new model to detect Swedish traffic signs using transfer learning. In addition, the study found that, even after the network was fine-tuned with additional training data, this method still performs well.

Figure 19. Illustration of the transfer learning attack setup [79]

Melis et al. [80] used the technology in [3] to prove the vulnerability of the robots to the input image under the proposed adversarial attack, which enables creating adversarial examples misclassified with higher confidence, under a maximum input perturbation. Xu et al. [81] launched an adversarial attack for Visual Turing test, known as Visual Question Answer (VQA). The result shows that the combined and non-combined VQA architectures using deep neural networks are vulnerable to adversarial attacks.

Sharif et al. [41] focused on facial biometric systems, which are widely used in surveillance and access control. They develop a systematic method to automatically carry out attacks, which are realized through printing a pair of eyeglass frames. When the eyeglass frame is worn by an adversary whose image is supplied to a state-of-the-art face-recognition algorithm, the eyeglasses allow him to evade being recognized or to impersonate another individual. Figure 20 shows impersonation using the frame. Their investigation focuses on white-box face-recognition systems, but they also demonstrate how similar techniques can be used in black-box scenarios, as well as to avoid face detection.

Figure 20. An impersonation using frames. Left: Actress Reese Witherspoon. Image classified correctly with probability 1. Middle: Perturbing frames to impersonate (actor) Russel Crowe as seen by the face recognition system. Right: The target [41]

Zhou et al. [82] studied an interesting example of an actual adversarial attack and found that infrared light completely invisible to humans can also be used to generate disturbances. As is shown in Figure 21, the adversary installed some infrared Light Emitting Diodes (LEDs) on the cap peak, which are facing the attacker's face. These infrared LEDs could illuminate the attacker's face with some spots in the camera's perspective, but people nearby would not notice. With this technique, they successfully attacked the facial authentication system in white-box settings.

Figure 21. A cap with some infrared LED and an example of adversarial image seen by the camera [82]

This method differs from other attack methods in two aspects. Firstly, this method does not directly optimize perturbation. They established a model describing the infrared points generated by LED lamps with position, radius, and brightness of the points as parameters. Therefore, the optimizer optimized the model parameters to accurately search for disturbances close to the real infrared points. Secondly, they developed a real-time feedback system for adversaries to adjust the LED position to help adversaries better achieve disturbance. Therefore, adversaries do not need to print disturbance for each target; instead, they only need to adjust the device to attack different targets.

2.2.2 Characteristics of Adversarial Examples

With the adversarial examples studied in many models for various applications, their characteristics have shown some common properties, regardless of the detailed method in their generation. Adversarial samples have three basic characteristics [21], i.e., transferability, regularization effect, and adversarial instability.

Transferability

Adversarial examples are not limited to attacking a specific neural network. It is unnecessary to obtain architecture and parameters of the model when constructing adversarial examples, as long as the model is trained to perform the same task. Adversarial examples generated from one model M1 can fool a different model M2 with a similar probability. Therefore, an attacker can use adversarial examples to attack the models that perform the same task, which means that an attacker can construct adversarial examples in the known machine learning model and then attack related unknown models [28].

Regularization effect

Regularization is a form of regression, that constrains/regularizes the coefficient estimates towards zero. In other words, this technique discourages learning a more complex or flexible model, so as to avoid the risk of overfitting.

Adversarial training [23] is a regularization method that can reveal the defects of models and improve the robustness of samples. However, compared to other regularization methods, the cost of constructing a large number of AEs for adversarial training is expensive. Unless researchers can find shortcuts for constructing AEs in the future, they are more likely to use dropout [145] or weight decay (L2 regularization).

Adversarial instability

After physical transformation such as cropping, translation and rotation, or changing the image quality such as denoising, lower resolution or super-resolution, it is easy to lose its own adversarial for adversarial samples. In this case, adversarial samples will be correctly classified by target models. This instability characteristic challenges attackers to construct robust AEs and creates the difficulty of deploying AEs in the real world.

The adversarial subspace is characterized using local intrinsic dimensionality (LID) in [169]. LID assesses the space-filling capability of the region surrounding a reference example, based on the distance distribution of the example to its neighbors. Explanations about how adversarial perturbation can affect the LID characteristic of adversarial regions is provided, and then it is empirically shown that LID characteristics can facilitate the distinction of adversarial examples generated using state-of-the-art attacks. The preliminary results show that the detection by LID can outperform several state-of-the-art detection measures by large margins for five attack strategies across three benchmark datasets.

Figure 22. LID scores (at the softmax layer) of 100 normal (blue), noisy (green), and Opt attack (red x-cross) examples from the CIFAR-10 dataset [169]

These characteristics are important for both attacks and defenses. For example, many black box attacks make use of the property of transferability while some defense methods propose ways of cutting off the transferability as an approach to defend against the attacks. Regularization effect has been used mainly for defense where models with adversarial training show improved robustness. Adversarial instability has also been used for defense where various mechanisms such as cropping, transformation, change of resolutions and etc have been proposed as defense methods.

2.2.3 Analysis on the Cause for Adversarial Examples

In theory, the existence of adversarial examples for any classifier is doomed. Implied in the theoretical study of [161], in tasks involving small distinguishability, no classifier in the considered set will be robust to adversarial perturbations, even if a good accuracy is achieved. Moreover, it shows a clear distinction between the robustness of a classifier to random noise and its robustness to adversarial perturbations. Specifically, in high dimensions, the former is shown to be much larger in strength than the latter for linear classifiers. This result gives a theoretical explanation for the discrepancy between the two robustness properties, which was empirically observed in [3] in the context of neural networks.

However, with the constraint of the adversarial strength to be imperceptible, why the deep learning models are so vulnerable, compared to the natural neural networks such as the human brain, is still an intriguing question. Analyzing the cause of AEs can help researchers to fix the vulnerability effectively. However, this is still an open problem and the difficulties come from the high dimensionality of the inputs and the unknown distribution of the natural images.

In the literature related to adversarial attacks on deep learning in Computer Vision, there are varied views on the cause for existence of adversarial examples. These views generally align well with the local empirical observations made by researchers while attacking or defending deep neural networks. However, they often fall short in terms of generalization.

At first researchers [29] suspected that adversarial examples arose from extreme nonlinearity of deep neural networks. It is commonly believed that this nonlinearity enables the deep learning models to approximate the complex functions involved in the tasks of mapping a raw image onto high level concept such as a class label. However, if the inputs of a linear model had enough dimensions, adversarial examples can also be constructed successfully with high confidence by adding small perturbations to the inputs [23].

Later the linearity hypothesis of Goodfellow et al. [23] explains the Fast Gradient Sign Method (FGSM) and related attacks very well and became popular. In the high dimensional linear classifier, each individual input feature is normalized. For one dimension of each input, small perturbations will not change the overall prediction of the classifier. However, small perturbations to all dimensions of the inputs will lead to an effective change of the output. However, Tanay and Griffin [24] demonstrated image classes that do not suffer from adversarial examples for linear classifier, which is not in-line with the linearity hypothesis. Not to mention, the linearity hypothesis itself deviates strongly from the previously prevailing opinion that the adversarial examples stem from highly non-linear decision boundaries induced by deep neural networks. There are also other examples in the literature where the linearity hypothesis is not directly supported [25].

Some researchers [28] suspected that the reason is over-fitting or insufficient regularization of the model which leads to insufficient generalization ability that learning models predict unknown data. However, by adding perturbations to a regularized model, Goodfellow et al. [23] found that the effectiveness against adversarial examples was not improved significantly.

With more detailed studies into the decision boundaries, flatness of decision boundaries [33] and large local curvature of the decision boundaries [34] and low flexibility of the networks [35] have been studied as the cause of existence of adversarial examples. However, they do not perfectly align with each other and none of these explanations applies to all of the adversarial examples.

Anyway, it is clear that the inner representations learned by current deep learning neural network is very different from that of a human being. As discovered in [38], firstly the neurons in DNNs do not truly detect objects/parts in input images, but they respond to objects/parts as recurrent discriminative patches. Secondly, the visual representations are not robust distributed codes of visual concepts, which limits their effectiveness when transferred to other tasks. Similarly, it is suggested in [30] that the learned decision boundaries of deep neural networks are inherently incorrect, which do not detect semantic objects. Y. Dong and et al [31] showed that if a dataset is generated by a smooth generative model with large latent space, there is no robust classifier to adversarial examples. Similarly, A. Fawzi and et al [32] proves that if a model is trained on a sphere dataset and misclassifies a small part of the dataset, then there exist adversarial examples with a small perturbation.

A model called BagNet is proposed in [171]. This is a simple variant of the ResNet-50 architecture, classifies an image based on the occurrences of small local image features without taking into account their spatial ordering. This strategy is closely related to the bag-of-feature (BoF) models popular before the onset of deep learning and reaches a surprisingly high accuracy on ImageNet (87.6% top-5 for 33×33 px features and Alexnet performance for 17×17 px features). The comparison between the popular DDNs and the BagNet suggests that the decision-making of many DNNs trained on

ImageNet follows at least in part a similar bag-of-feature strategy, that is, they depend mostly on local features.

It has been further shown that ImageNet (natural images) trained CNNs are strongly biased towards recognizing textures rather than shapes, which is in stark contrast to human behavioral evidence and reveals fundamentally different classification strategies [39]. For the CNNs to learn shape-based representations, the images need to be stylized, with the original shapes of the images but with textures from different paintings, before input to the deep neural networks.

A large scale, high-resolution comparison of the core visual object recognition behavior of humans, monkeys, and state-of-the-art deep artificial neural networks [157] supports the difference and directly shows that current deep CNN neural network models cannot account for the image-level behavioral patterns of primates, even when images are not optimized to be adversarial.

Without a theoretical framework for the cause of adversarial examples, it is useful to compare the behaviors of deep learning neural networks and that of the human being. It is also logical that the most promising defense in the current situation should come from the learning of human brains.

2.3 Method of Categorizing the Literature

With the increasing attention put into this research area, there are lot of research works on both attack and defense appearing every day. It is important to categorize the approaches for a systematic analysis.

The attack and the defense is an on-going game, thus they have to be reviewed concurrently to form a complete view of both sides. However, there is a great difference in both the objective and the implementation for adversarial attack and defense, so the methods of attack and defense also need to be categorized in separate ways.

As defined by Goodfellow, “adversarial examples are inputs to machine learning models that an attacker has intentionally designed to cause the model to make a

mistake”. According to the different influence of the perturbation on the classifier, the adversarial goals can be further classified into four types:

- (1) Confidence reduction: reduce the confidence of output classification.
- (2) Non-targeted misclassification: alter the output classification to any class which is different from the original class.
- (3) Targeted misclassification: force the output classification to be the attacker-specific target class.
- (4) Source/Target misclassification: select a specific input to generate an attacker-specific target class.

On the contrary, the defense goal in general is to increase the robustness of the neural networks and not to be fooled by the adversarial examples. However, as the researchers pointed out, there is a compromise in performance and cost. According to different benefit and cost compromises, the goals can also be classified into four types:

- 1) Low impact on the model architecture: when constructing any defense against AEs, the primary consideration is the minimal modification to model architectures.
- 2) Maintain model speed: running time is very important for the availability of DNNs. It should not be affected during testing. With the deployment of defenses, DNNs should still maintain high performance on large datasets.
- 3) Maintain accuracy: defenses should have little impact on the classification accuracy of models.
- 4) Defenses should be targeted: defenses should be effective on the examples which are relatively close to the training set. Since the examples that are far from the dataset are relatively secure, the perturbations to these examples are easily detected by the classifier.

In the following two sections, the categorization methods for both attack and defense will be introduced.

2.3.1 Categorization of Attacks

According to the different stages of the target model, the adversarial attacks can be divided into three categories: attacks in the training stage, attacks in the testing stage and attacks in the model deployment stage. Since the attack methods in the model deployment stage are very similar to the methods in the testing stage, for the sake of simplicity, this study only discusses the attacks in the testing stage. This is also where most of the attacking methods are focused on.

This study categorizes the attacks based on three dimensions: the knowledge available to the attackers, the adversary specificity and the perturbation norm used as the constraint.

Adversary's Knowledge

Based on an adversary’s knowledge, adversarial attacks can be divided into white-box attacks and black-box attacks.

In white-box scenarios, the adversaries have access to the parameter, algorithms, and structure of the target model. They can construct adversarial samples to carry out attacks by utilizing this knowledge. Papernot et al. [40] introduced an adversarial crafting framework, which can be divided into the direction sensitivity estimation step and perturbation selection step as illustrated in Figure 23. It can be generalized for any supervised learning algorithm (Image Credit: Papernot et al. [40]). On this basis, the researchers have proposed a variety of different attack methods.

Figure 23. Adversarial crafting framework [40]

Instead, in the black-box scenarios, the adversaries cannot obtain information about the target model, but they can train a local substitute model by querying the target model, utilizing the transferability of adversarial samples or using a model inversion method.

The black-box attacks can be divided into non-adaptive, adaptive and strict black-box attacks.

In the non-adaptive and strict black-box scenario, the adversaries cannot access the internal structure and parameters of the target model but can train a local substitute model which approximates the decision boundary of the target model by accessing the target model through the API interfaces. In particular, the machine learning models in the cloud environment are more accessible by API interfaces. Once the substitute model is trained with high confidence, any white box attack strategies can be applied to the substitute model to generate adversarial samples and then deceive the target model by using the transferability of adversarial samples.

In the adaptive black-box scenario, adversaries cannot access the model but can observe the output corresponding to specific input and establish a substitute model similar to the target model by querying the target model as an oracle. The effectiveness of oracle attack is closely related to the input and query times of asking the target model.

Adversarial Specificity

Based on the adversarial specificity, the attacks can be also classified into targeted attacks and non-targeted attacks.

Targeted attacks misguide deep neural networks to a specific class. Targeted attacks usually occur in the multiclass classification problem. For example, an adversary fools an image classifier to predict all adversarial examples as one class. In a face recognition/biometric system, an adversary tries to disguise a face as an authorized user (Impersonation) [41]. Targeted attacks usually maximize the probability of targeted adversarial class.

Non-targeted attacks do not assign a specific class to the neural network output. The adversarial class of output can be arbitrary except the original one. For example, an adversary makes his/her face misidentified as an arbitrary face in face recognition system to evade detection (dodging) [41]. Non-targeted attacks are easier to implement compared to targeted attacks since it has more options and space to redirect the output. Non-targeted adversarial examples are usually generated in two ways: 1) running several targeted attacks and taking the one with the smallest perturbation from the results; 2) minimizing the probability of the correct class.

Some generation approaches (e.g., extended BIM - Basic Iterative Method) can be applied to both targeted and non-targeted attacks. For binary classification, targeted attacks are equivalent to non-targeted attacks.

Perturbation Norm

It is also important to take note of the perturbation norm (sometimes referred to as “perturbation measure”) used as the constraint when designing the adversarial attacks. This is an indication of the restricted l_p -norm (for a real number $p \geq 1$, it is defined by $\|x\|_p = (|x_1|^p + |x_2|^p + \dots + |x_n|^p)^{1/p}$) of the perturbations to make them imperceptible. As theoretical study [37] has shown that “the choice of norm can have a dramatic effect on the strength of theoretical guarantees for the existence of adversarial examples”. The commonly used norms are l_∞ , l_0 , l_2 and l_1 and these are used as one of the metrics to categorize the attack methods.

Using this categorization method, all the attack methods can be put into Table 3 at the beginning of Section 2.4. Three other attributes are also included in the table: whether the attack is image specific or universal, whether it is one shot or iterative learning and a measure of the attack strength based on impression from the reviewed literature.

2.3.2 Categorization for Defenses

In order to improve the robustness of neural networks against adversarial attack, researchers have proposed a mass of adversarial defense methods, which can be divided into three main categories:

- 1) Using modified training during learning or modified input during testing;
- 2) Modifying networks, e.g. by adding more layers/subnetworks, changing loss/activation functions etc;
- 3) Using external models as network add-on when classifying unseen examples.

The approaches along the first direction do not directly deal with the learning models. On the other hand, the other two categories are more concerned with the neural networks themselves. The techniques under these categories can be further divided into two types; namely (a) complete defense and (b) detection only. The ‘complete defense’ approaches aim at enabling the targeted network to achieve its original goal on the adversarial examples, e.g. a classifier predicting labels of adversarial examples with acceptable accuracy. On the other hand, ‘detection only’ approaches are meant to raise the red flag on potentially adversarial examples to reject them in any further processing. The taxonomy of the described categories is also shown in Figure 24. The remaining section is organized according to this taxonomy. In the used taxonomy, the difference between ‘modifying’ a network and employing an ‘add-on’ is that the former makes changes to the original deep neural network architecture/parameters during training. On the other hand, the latter keeps the original model intact and appends external model(s) to it during testing.

Figure 24. Categorization of adversarial defense approaches

Modifying Data

The methods of modifying data refer to modifying the training dataset in the training stage or changing the input data in the testing stage. There are six methods to modify data.

Adversarial training is a widely used method of modifying data where the adversarial examples are treated as hard negative cases and additional trainings can be done especially for them. Szegedy et al. [3] injected adversarial samples and modified their labels to improve the robustness of the target model. By using adversarial training, Goodfellow et al. [24] reduced the misidentification rate on the Mixed National Institute of Standards and Technology (MNIST) [83] dataset from 89.4% to 17.9%, Huang et al. [84] increased the robustness of target model by punishing misclassified adversarial samples. However, it is unrealistic to introduce all unknown attack samples into the adversarial training.

The second method is called gradient hiding [85]; it hides gradient information of the target model from the adversaries. However, by learning the proxy black-box model with gradient and using the adversarial samples generated by this model [23], this method can easily be fooled.

Since the transferability attribute holds even if the neural networks have different architectures or trained on the disjoint dataset, the third method is blocking the transferability to prevent the black-box attack. Hosseini et al. [86] proposed a Three-step Null Labeling method to prevent the transferability of adversarial samples. The advantage of this method is marking the perturbation input as an empty label rather than classifying it as the original label.

The fourth method is data compression, which improves the robustness by compressing data. Dziugaite et al. [87] and Das et al. [88] used JPG compression and a JPEG compression method to prevent FGSM (Fast Gradient Sign Method) attacks, respectively. The limitation of this method is that a large amount of compression will lead to an accuracy decrease of original image classification, while a small compression is often not enough to remove the impact of disturbance.

The fifth approach is foveation based defense. Luo et al. [12] demonstrated that significant robustness against the adversarial attacks using L-BFGS [3] and FGSM [24] is possible with the “foveation” mechanism – applying neural network in different regions of images.

The last one of modifying data is data randomization [21]. Wang et al. [89] used a data conversion module separated from the network model to eliminate the possible adversarial disturbance in the image, and conducted data expansion operations in the training process, which could slightly improve the robustness of the target model.

Modifying Model

The “complete defense” approaches can be classified into eight categories.

The first method is regularization, which aims to improve the generalization ability of the target model by adding regular terms which are known as penalty terms to the cost function and make the model have good adaptability to resist attacks on an unknown dataset in prediction. Biggio et al. [90] used a regularization method to limit the vulnerability of data when training an SVM model. The works [91][92][93] used the regularization method to improve the robustness of the algorithm and achieved good results.

The second method is defensive distillation [40], which produced a model with a smoother output surface and less sensitivity to disturbance, so as to improve the robustness of the model, and can reduce the success rate of adversarial attack by 90%. Moreover, Papernot et al. [94] proposed extensible defense distillation technology to guarantee the effectiveness in black-box attacks.

The third method is defensive quantization [159], which aims to design a novel quantization methodology to jointly optimize the efficiency and robustness of deep learning models. An empirical study shows that vanilla quantization makes the model more vulnerable to adversarial attack and it’s observed that the inferior robustness comes from the error amplification effect, where the quantization operation further

enlarges the distance caused by amplified noise. Based on this observation, a novel Defensive Quantization (DQ) method is proposed to maintain the magnitude of the adversarial noise non-expansive during inference by controlling the Lipschitz constant (“intuitively, a Lipschitz continuous function is limited in how fast it can change: there exists a real number such that, for every pair of points on the graph of this function, the absolute value of the slope of the line connecting them is not greater than this real number; the smallest such bound is called the Lipschitz constant of the function” – abstract from Wikipedia) of the network during quantization.

The fourth method is using a deep contractive network (DCN) [44], which uses a noise reduction automatic encoder to reduce the adversarial noise.

The fifth method is inserting a mask layer before processing the classified network model [45]. The mask layer is used to encode the differences between original images and the output features of the previous network model layers.

The sixth method of modifying model is using Parseval networks [46]. This network adopts hierarchical regularization by controlling the global Lipschitz constant of the network.

The seventh method makes use of biologically inspired protection. Based on the observation that human beings are not fooled by the adversarial examples, researchers are trying to identify the mechanisms that human beings possess that make them resistant to adversarial examples. Non-linear activations [98], Dense Associative Memory models [99] are some of the proposed mechanisms for defense.

The last category includes other miscellaneous approaches. They will be discussed in more details in Section 2.5.2.8.

Detection Only Approaches

Different from the “complete defense” approaches, which aim at enabling the targeted network to achieve its original goal on the adversarial examples, e.g. a classifier predicting labels of adversarial examples with acceptable accuracy, “detection only”

approaches are meant to raise the red flag on potentially adversarial examples to reject them in any further processing.

There are also detection only approaches within the “modifying network” category. However, the difference between “modifying a network” and employing an “add-on” is that the former makes changes to the original deep neural network architecture/parameters during training. On the other hand, the latter keeps the original model intact and appends external model(s) to it during testing.

The methods belongs to this category include SafetyNet, Detector subnetwork and others. Detailed survey is covered in the Section 2.5.2.9.

Network Add-ons

Network add-ons refers to using additional tools as an auxiliary tool for the neural network model when classifying unseen examples.

The complete defense methods can be further grouped into Input Reconstruction, Network Verification and Ensembled Defense.

Input Reconstruction aims to transform adversarial examples to the clean data via reconstruction. After transformation, the adversarial examples will be “transferred” onto the manifolds of the natural images and will not affect the prediction of deep learning models. Deep Contractive Autoencoder [44], PixelDefend [28] are typical methods of this type.

Network Verification is a promising solution to defend adversarial examples, because it may detect the new unseen attacks. Network verification checks the properties of a neural network: whether an input violates or satisfies the property. Katz et al. proposed a verification method for neural networks with ReLU activation function, called Reluplex [100]. They used Satisfiability Modulo Theory (SMT) solver to verify the neural networks. As a solution to the heavy computation problem of Reluplex, DeepSafe [101] provides safe regions of deep neural network using Reluplex instead of checking each point individually.

Due to the multi-facet of adversarial examples, multiple defense strategies can be performed together (parallel or sequential) to defend adversarial examples. As found in [127], an attack leading one model to misclassify does not imply the same for other networks performing the same task. This makes ensemble methods an attractive defense strategy against adversarial attacks. It is empirically shown for the MNIST and the CIFAR-10 data sets that ensemble methods not only improve the accuracy of neural networks on test data but also increase their robustness against adversarial perturbations.

Detection Only Approaches

Adversarial Detection is to use additional tool to detect whether an input is adversarial example before applying the recognition or classification. Based on the reasonable assumption that the difference between an adversarial example and a natural image is not a white noise, it is possible to train another neural network model to detect it. Feature Squeezing [43], MagNet [96] are typical methods of this type.

Based on the structure of the network model used for the detection, it can be categorized into feedforward and feedback structures. Most of existing methods belong to the feedforward structure. However, there is a group of researchers who noticed the importance of feedback structure and proposed methods to solve the problem using a feedback structure. Detailed review of these methods is covered in Section 2.5.3.

2.4 Review of Adversarial Attacks

In this study, we focus on the attack methods for classification networks. The attack methods will be reviewed mainly in chronological order, but they will be all put into a bigger view based on the categorization methods proposed in the previous section.

Method	Black or White Box	Targeted or Non-targeted	Perturbation Norm	Specific or Universal	Learning	Strength
L-BFGS	White	Targeted	l_∞	Image Specific	One Shot	***
FGSM	White	Targeted	l_∞	Image Specific	One Shot	***
One Step Target Method	White	Targeted	l_2, l_∞	Image Specific	One Shot	***
IGSM	White	Targeted	l_∞	Image Specific	Iterative	****
BIM & ILCM	White	Non-targeted	l_∞	Image Specific	Iterative	****
JSMA	White	Targeted	l_0	Image Specific	Iterative	***
One-pixel	Black	Non-targeted	l_0	Image Specific	Iterative	**
C&W attacks	White	Targeted	l_0, l_2, l_∞	Image Specific	Iterative	*****
DeepFool	White	Non-targeted	l_2, l_∞	Image Specific	Iterative	****
Universal Perturbations	White	Non-targeted	l_2, l_∞	Universal	Iterative	*****
UPSET	Black	Targeted	l_∞	Univeral	Iterative	****
ANGRI	Black	Targeted	l_∞	Image Specific	Iterative	****
Houdini	Black	Targeted	l_2, l_∞	Image Specific	Iterative	****
ATNs	White	Targeted	l_∞	Image Specific	Iterative	****
Ensemble Attack	White	Targeted	l_∞	Image Specific	Iterative	****

Table 3. The list of reviewed adversarial attacks in the proposed categorization

Box-constrained L-BFGS

Szegedy et al. [3] firstly proposed L-BFGS (Limited-memory; Broy-den, Fletcher, Goldforb, Shanno) to construct adversarial examples for deep learning networks in 2014. Given an image X , attackers construct an image X' similar to X with l_2 norm and X' can be labelled as a different class. The optimization problem is:

$$\text{minimize } \|X - X'\|_2^2 \quad (2.2)$$

where $\|X - X'\|_2^2$ is L_2 norm. The attack goal is to make $f(X') = l, X' \in [0,1]^n$, where l is the target class. $f(X') = l$ is the nonlinear and non-convex function which is difficult to be solved directly. Therefore, the problem is solved approximately by the box-constrained L-BFGS.

$$\text{minimize } c \cdot \|X - X'\|_2^2 + \text{loss}_{F,l}(X') \quad (2.3)$$

where c is a randomly initialized hyper-parameter, which is determined by linear search; $\text{loss}_{F,l}(X')$ is the loss function. $f(X') = l$ is approximated by minimizing the loss function. Although this method has high stability and effectiveness, the calculation is complicated.

Fast Gradient Sign Method (FGSM)

Goodfellow et al. [24] proposed a simple and fast method to construct adversarial examples, named fast gradient sign method (FGSM). The generated images are misclassified by adding perturbations and linearizing the cost function in the gradient direction. Given an original image X , the problem can be solved with:

$$X^{adv} = X + \epsilon \cdot \text{sign}(\nabla_X J(X, y_{true})) \quad (2.4)$$

where X^{adv} represents an adversarial example from X , ϵ is a randomly initialized hyper-parameter, $\text{sign}(\cdot)$ is a sign function, y_{true} is the true label corresponding to X , and $J(\cdot)$ is the cost function used to train the neural network, $\nabla X(\cdot)$ represents the gradient of X and it can be efficiently calculated using back propagation.

There are two main differences between FGSM and LBFGS. First, FGSM is optimized with the L_∞ norm while L-BFGS is optimized with L_2 norm. Second, FGSM is a fast adversary construction method because it does not require an iterative procedure to

compute the adversarial examples. Hence it has a lower computation cost than other methods.

One-Step Target Class Method

This method [62], a variant of FGSM, maximizes the probability $P(y_{target}|X)$ of some specific target class y_{target} , which is unlikely to be a true class for a given sample. For a model with cross-entropy loss, the adversarial samples could be made in the one-step target class method:

$$X' = X - \epsilon * \text{sign}(\nabla_x J(X, y_{target})) \quad (2.5)$$

The gradients with l_2 and l_∞ generates two different results respectively.

IGSM

It is difficult for FGSM to control the perturbation in constructing adversarial examples. Kurakin et al. [61] proposed an optimized FGSM, named iterative gradient sign method (IGSM), which applies perturbations to multiple smaller steps and clips the results after each iteration to ensure that the perturbations are within the neighborhood of the original image. For the N -th iteration, the update process is:

$$X_0^{adv} = X, X_{N+1}^{adv} = \text{Clip}_{X,\epsilon}\{X_N^{adv} + \epsilon \cdot \text{sign}(\nabla_x J(X_N^{adv}, y_{true}))\} \quad (2.6)$$

where $\text{Clip}_{X,\epsilon}(\cdot)$ denotes $[X - \epsilon, X + \epsilon]$.

IGSM is non-linear in the gradient direction and requires multiple iterations, which is simpler than L-BFGS method in calculation, and the success rate of adversarial example construction is higher than FGSM. IGSM can be further divided into two types:

- 1) reducing the confidence of the original prediction as the original class;
- 2) increasing the confidence of the prediction that originally belongs to the class with the smallest probability.

Basic Iterative Method (BIM)

Kurakin and et al [62] proposes a straightforward extension of FGSM to apply FGSM multiple times with a small step size:

$$X'_0 = X, X'_{n+1} = \text{Clip}_{X,\epsilon}\{X'_n + \alpha * \text{sign}(\nabla_x J(X'_n, y_{true}))\} \quad (2.7)$$

Here, α denotes step length, and $\text{Clip}_{X,\epsilon}\{A\}$ is the element-wise clipping of X . This method generally does not rely on the approximation of the model and produces additional harmful adversarial samples when this algorithm runs more iterations. The articles [63][64] have conducted research on this basis and produced good results.

Least-Likely-Class Iterative Methods (ILCM)

FGSM and L-BFGS try to increase the probability of predicting wrong results, but do not specify which wrong class should be selected by the model. These methods are sufficient for small datasets such as MNIST and CIFAR-10. On ImageNet, with a larger number of classes and varying degrees of significance between classes, FGSM and L-BFGS may construct uninteresting misclassifications, such as misclassifying one type of cat into another cat. To generate more meaningful AEs, a novel AE generation method is proposed by perturbing the target class with the lowest probability so that this least-likely class turns to become the correct class after the perturbation, which is called iterative least-likely class method (ILCM) [61]. To make the adversarial image X^{adv} be classified as y_{LL} , we have the following procedure:

$$X_0^{adv} = X, X_{N+1}^{adv} = \text{Clip}_{X,\epsilon}\{X_N^{adv} - \epsilon * \text{sign}(\nabla_x J(X_N^{adv}, y_{LL}))\} \quad (2.8)$$

where y_{LL} represents the least likely (the lowest probability) target class. For a classifier with good performance, the least likely class is usually quite different from the correct class. Therefore, this attack method can lead to some interesting errors, such as misclassifying a cat as an aircraft. It is also possible to use a random class as the target class, which is called iteration random class method.

Jacobian-based Saliency Map Attack (JSMA)

Papernot et al. [42] proposed Jacobian-based Saliency Map Attack (JSMA), which is based on the L_0 distance norm. The basic idea is to construct a saliency map with the gradients and model the gradients based on the impact of each pixel. The gradients are directly proportional to the probability that the image is classified as the target class, i.e., changing a large value will significantly increase the likelihood that the model classifies the image as the target class. JSMA allows us to select the most important pixel (the maximum gradient) based on the saliency map and then perturb the pixel to increase the likelihood of labeling the image as the target class. More specifically, JSMA includes the following steps:

- (1) Compute forward derivative $\nabla F(X)$.

$$\nabla F(X) = \frac{dF(X)}{dX} = \left[\frac{dF_j(X)}{dX_i} \right]_{i \in 1 \dots M, j \in 1 \dots N} \quad (2.9)$$

- (2) Construct a saliency map S based on the forward derivative.
- (3) Modify the most important pixel based on the saliency map, repeat this process until the output is the target class or the maximum perturbation is obtained.

When the model is sensitive to changes in the inputs, it is easier for JSMA to calculate the minimum perturbation to generate the AEs. JSMA has high computational complexity while the generated AEs have high success rate and transfer rate.

One Pixel Attack

Su et al. [65] proposed an extreme attack method, which can be achieved by changing only one-pixel value in the image. They used the differential evolution algorithm to iteratively modify each pixel to generate a sub-image and compared it with the parent image to retain the sub-image with the best attack effect according to the selection criteria to realize the adversarial attacks.

Carlini and Wagner Attacks (C&W)

Carlini and Wagner [66] proposed a powerful attack method based on L-BFGS. The attack with L_0, L_2, L_∞ distance norm can be targeted or non-targeted, and we take the non-targeted L_2 norm as an example here. The corresponding optimization problem is:

$$\text{minimize } \|\delta\|_2 + c * f(X + \delta) \quad (2.10)$$

where $X + \delta \in [0,1]^n$, c is a hyper-parameter that can balance these two terms, and δ is a small perturbation. The definition of the objective function $f(X')$ is as follows:

$$f(X') = \max (\max\{Z(X')_i: i \neq t\} - Z(X')_t, -l) \quad (2.11)$$

where $Z(X')$ is the last hidden layer, t is the true label, and l is a hyper-parameter, which is used to control the confidence level of the model misclassification, and the AE X' can be classified as t with high confidence by adjusting the value of l . In general, high confidence attacks have large perturbations and high transfer rates, and CW Attack based on the l_0, l_2, l_∞ distance metric can defeat the defensive distillation (distillation is a training procedure initially designed to train a DNN using knowledge transferred from a different DNN and it is proposed as a defense method) [40]. There are three improvements to this attack based on L-BFGS:

- Use the gradient of the actual output in the model instead of the gradient of softmax.
- Apply different distance metrics (L_0, L_2, L_∞).
- Apply different objective functions $f(X')$.

DeepFool

Mohsen et al. [67] proposed a non-targeted attack method based on L_2 norm, called DeepFool. Assuming that the neural network is completely linear, there must be a hyperplane separating one class from another. Based on this assumption, the optimal solution to this problem is analyzed and used to construct AEs. The corresponding optimization problem is as follows:

$$r_*(X_0) = \text{argmin} \|r\|_2 \quad (2.12)$$

subject to $\text{sign}(f(X_0 + r)) \neq \text{sign}(f(X_0))$, where r indicates the perturbation.

Figure 25. Adversarial examples for a linear binary classifier [67]

As shown in Figure 25, X_0 is the original example, $f(X)$ is a linear binary classifier, the straight line $WX + b = 0$ is the decision boundary, and $r_*(X)$ is the distance from the original example to the decision boundary, i.e., the distance from X_0 to the straight line $WX + b = 0$. The distance is equivalent to the perturbation $\Delta(X_0; f)$. Therefore, when $\Delta(X_0; f) > r_*(X)$, the AE can be generated.

Compared with L-BFGS, DeepFool is more efficient and powerful. The basic idea is to find the decision boundary that is the closest to X in the image space, and then use the boundary to fool the classifier. It is difficult to solve this problem directly in neural networks with high dimension and nonlinear space. Therefore, a linearized approximation is used to iteratively solve this problem. The approximation is to linearize the intermediate X_0 classifier in each iteration, and obtain an optimal update direction on the linearized model. Then X_0 is iteratively updated in this direction by a small step α , repeating the linear update process until X_0 crosses the decision boundary. Finally, AEs can be constructed with subtle perturbations.

Universal Adversarial Perturbations

FGSM, JSMA and DeepFool can only generate adversarial perturbations to fool a network on a single image. Moosavi-Dezfooli et al. [68] proposed a universal image-

agnostic perturbation attack method which fools classifiers by a single adversarial perturbation to all images. The specific problem can be defined as finding a universal perturbation $\mathbf{v}$, such that $\hat{k}(\mathbf{X} + \mathbf{v}) \neq \hat{k}(\mathbf{X})$ for most examples in the dataset subject to the distribution μ .

$$\hat{k}(\mathbf{X} + \mathbf{v}) \neq \hat{k}(\mathbf{X}) \quad \text{for most } \mathbf{X} \sim \mu \quad (2.13)$$

where $\hat{k}$ denotes a classification function that labels each image. Such perturbation $\mathbf{v}$ is called the universal perturbation. The attackers' goal is to find the $\mathbf{v}$ that satisfies the following two constraints

$$\begin{cases} 1. \|\mathbf{v}\|_p \leq \epsilon \\ 2. \int_{\mathbf{X} \sim \mu}^P (\hat{k}(\mathbf{X} + \mathbf{v}) \neq \hat{k}(\mathbf{X})) \gg 1 - \delta \end{cases} \quad (2.14)$$

where $\|\cdot\|_p$ denotes the p -norm (or l_p), the parameter ϵ controls the magnitude of the perturbation $\mathbf{v}$, and quantifies the desired fooling rate for all images. The attack method has two characteristics: 1) the perturbation is related to the target model rather than the image; 2) the small perturbation will not change the structure of the image itself.

UPSET and ANGRI

Sarkar et al. [69] proposed two black-box attack algorithms, namely UPSET: Universal Perturbations for Steering to Exact Targets, and ANGRI: Antagonistic Network for Generating Rogue Images for targeted fooling of deep neural networks. For 'n' classes, UPSET seeks to produce 'n' image agnostic perturbations such that when the perturbation is added to an image that does not belong to a targeted class, the classifier will classify the perturbed image as being from that class. The power of UPSET comes from a residual generating network $R(\cdot)$, that takes the target class 't' as input and produces a perturbation $R(t)$ for fooling. The overall method solves the following optimization problem using the so-called UPSET network:

$$I_p = \max(\min(sR(t) + I_c; 1), -1) \quad (2.15)$$

where the pixel values in I_c are normalized to lie in $[-1, 1]$, and 's' is a scalar. To ensure I_p to be a valid image, all values outside the interval $[-1, 1]$ are clipped. As compared to the image-agnostic perturbations of UPSET, ANGRI computes image-specific

perturbations in a closely related manner, for which we refer to the original work. The perturbations resulting from ANGRI are also used for targeted fooling. Both algorithms have been reported to achieve high fooling ratios on MNIST [70] and CIFAR-10 [71] datasets.

Houdini

Cisse et al. [72] proposed a method called HOUDINI, which deceives gradient-based machine learning algorithms. It realized the adversarial attack by generating adversarial sample specific to the task loss function, i.e., using the gradient information of the differentiable loss function of the network to generate disturbance. They demonstrated that HOUDINI can be used to deceive not only an image classification network but also a speech recognition network.

Adversarial Transformation Networks (ATNs)

Baluja and Fischer [73] trained feed-forward neural networks to generate adversarial examples against other targeted networks or set of networks. The trained models were termed Adversarial Transformation Networks (ATNs). The adversarial examples generated by these networks are computed by minimizing a joint loss function comprising two parts. The first part restricts the adversarial example to having perceptual similarity with the original image, whereas the second part aims at altering the prediction of the targeted network on the resulting image.

Along the same direction, Hayex and Danezis [74] also used an attacker neural network to learn adversarial examples for black-box attacks. In the presented results, the examples computed by the attacker network remain perceptually indistinguishable from the clean images but they are misclassified by the targeted networks with overwhelming probabilities - reducing classification accuracy from 99.4% to 0.77% on MNIST data [70], and from 91.4% to 6.8% on the CIFAR-10 dataset [71].

Ensemble Attack

Liu et al. [75] proposed an ensemble attack method combining multiple models to construct AEs. If an adversarial image remains adversarial for multiple models, it is likely to be transferred to other models. Formally, given k white-box models with softmax outputs being $J_1, \dots, J_k$, an original image $\mathbf{X}$ and its true label y , the ensemble-based approach solves the following optimization problem (for targeted attack):

$$\operatorname{argmin}_{\mathbf{X}'} -\log ((\sum_{i=1}^k \alpha_i J_i(\mathbf{X}')) \cdot \mathbf{1}_{y'}) + \lambda d(\mathbf{X}, \mathbf{X}') \quad (2.16)$$

where y' is the target label specified by the attacker, $\sum_{i=1}^k \alpha_i J_i(\mathbf{X}')$ is the ensemble model, and α_i is weight the of the i -th model, $\sum_{i=1}^k \alpha_i = 1$, λ is a randomly initialized parameter that is used to control the weight of the two terms.

The goal is to ensure the generated AEs are still adversarial for the other black-box model J_{k+1} . Since the decision boundaries for different models are almost the same, the transferability of targeted AEs is improved significantly.

2.5 Review of the Adversarial Defense

Defense Strategies		Methods	Representative Papers
Modifying Data		Adversarial Training	[24][62][84][85][102][103]
		Gradient Hiding	[156]
		Blocking Transferability	[86]
		Data Compression	[87][88][104][105][160]
		Foveation Based	[12]
		Data Randomization	[21][89]
Modifying Model	Complete	Regularization	[90][91][92][93][111-114]
		Defensive Distillation	[40][94][115]
		Defensive Quantization	[159]
		Deep Contractive Network	[117][118]
		Mask Layer Insertion	[45]
		Parseval Network	[46]
		Biologically Inspired Methods	[98][99]
		Miscellaneous Approaches	[110][122-136]
	Detection Only	SafetyNet, Feature Squeezing, Detector Subnetworks, Deep k-NN and etc	[43][68][137-139][141-142] [144][145][173]
Network Add-ons	Complete	Input Reconstruction	[28][72][97][104]
		Network Verification	[100][101][125]
		Ensembled Defense	[28][96][127]
	Detection Only	Feedforward Networks	[43][96][148][149]
		Feedback Networks	NIL

Table 4. List of all reviewed defense methods following the categorization method

2.5.1 Modifying Data

2.5.1.1 Adversarial Training

Since the discovery of adversarial examples for the deep neural networks [3], there has been a general consensus in the related literature that robustness of neural networks against these examples improves with adversarial training. Adversarial examples have been used to improve anti-interference ability for AI models.

In 2015, Goodfellow et al. [24] proposed adversarial training to improve the robustness of the model. The basic idea is to add AEs into the training data to continuously generate new AEs at each step of the training. The number and relative weight of AEs in each batch are controlled by the loss function independently. The corresponding loss function is as follows:

$$LOSS = \frac{1}{(m-k)+\lambda k} (\sum_{i \in CLEAN} L(X_i|y_i) + \lambda \sum_{i \in ADV} L(X_i^{ADV}|y_i)) \quad (2.17)$$

where $L(X|y)$ is a loss function of the example X with a true label y , m is the total number of training examples, k is the number of AEs, and λ is a hyper-parameter used to control the relative weight of the AEs in the loss function. When $k = 0.5m$, i.e., when the number of AEs is the same as the number of original examples, the model has the best effect in the adversarial training.

Adversarial training is not the same as data augmentation. The augmented data may appear in the test set, while AEs are usually not shown in the test set but can reveal the defects of the model. Adversarial training can be viewed as the process of minimizing classification error rates when the data is maliciously perturbed.

[24] and [84] were evaluated only on the MNIST dataset. [24] reduced the misidentification rate on the MNIST dataset from 89.4% to 17.9% by using adversarial training. A comprehensive analysis of adversarial training methods on the ImageNet dataset was presented in [62]. They used half adversarial examples and half origin examples in each step of training. From the results, adversarial training increased the

robustness of neural networks for one-step attacks (e.g., FGSM) but would not help under iterative attacks (e.g., BIM and ILLC methods). [62] suggested that adversarial training is used for regularization only to avoid overfitting (e.g., the case in [24] with the small MNIST dataset).

Although adversarial training improves robustness of a network, to be really effective, it requires that training is performed using strong attacks and the architecture of the network is sufficiently expressive. Since adversarial training necessitates increased training/data size, we refer to it as a ‘brute-force’ strategy. It is also commonly observed in the literature that brute-force adversarial training results in regularizing the network (e.g. [24]) to reduce over-fitting, which in turn improves robustness of the networks against the adversarial attacks. Inspired by this observation, Miyato et al. [102] proposed a ‘Virtual Adversarial Training’ approach to smooth the output distributions of the neural networks. A related ‘stability training’ method is also proposed by Zheng et al. [103] to improve the robustness of neural networks against small distortions to input images. It is noteworthy that whereas adversarial training is known to improve robustness of neural networks, Moosavi-Dezfooli [68] showed that effective adversarial examples can again be computed for already adversarially trained networks.

Although a model is robust to white-box attacks after adversarial training, it is still vulnerable to the AEs generated from other models, i.e., the model is not robust to black-box attacks. Based on this attribute, Tramèr et al. [85] proposed the concept of ensemble adversarial training. The main idea is to augment the training data which constructed not only from the model being trained but also from the other pre-trained models, which increases the diversity of AEs and improves the generalization ability.

2.5.1.2 Gradient Hiding

Gradient hiding [156] is proposed as a natural defense against gradient based attacks presented in [85] and attacks using adversarial crafting method such as FGSM. It is found that neural networks trained with this input gradient regularization exhibit robustness to transferred adversarial examples generated to fool all of the other models. This method hides information about model gradient from the adversaries, i.e., if a model is non-differentiable (e.g., a decision tree, a nearest neighbor classifier, or a

random forest), the gradient-based attack is invalid. However, by learning the proxy black-box model with gradient and using the adversarial samples generated by this model [23][157], the method can easily be fooled in this case.

2.5.1.3 Blocking the Transferability

Since the transferability attribute holds even if the classifiers have a different architecture or are trained on a disjoint dataset, the key to preventing a black-box attack is to prevent the transferability of adversarial samples. Hosseini et al. [86] proposed a three-step NULL Labeling method, in order to prevent the adversarial samples from one network to another network. Its main idea is adding a new NULL label to the dataset and classify them to NULL label by training the classifier to resist adversarial attacks. This method generally includes three main steps, i.e., initial training target classifier, computing NULL probabilities, and adversarial training, as is shown in Figure 26. The classifier assigns a probability vector to each image. With more perturbation, the NULL labeling method assigns a higher probability to the NULL label, while the original labeling increases the probabilities of other labels.

Figure 26. Illustration of the original and null labeling methods on an image from the MNIST dataset, and three adversarial examples with different perturbations [86]

The advantage of this method is marking the perturbation input as an empty label rather than classifying it as the original label. At present, this method is one of the most effective defense methods against adversarial attacks, which accurately resists adversarial attacks, as well as does not affect the classification accuracy of the original data.

2.5.1.4 Data Compression

Dziugaite et al. [87] noted that most of the popular image classification datasets comprise JPG images and found that JPG compression method can improve the recognition accuracy decline caused by FGSM attack disturbance on a large number of network models. Nevertheless, it is concluded that compression alone is far from an effective defense.

JPEG compression was also studied by Guo et al. [105] for mitigating the effectiveness of adversarial images. Das et al. [88] used a similar JPEG compression method to study a defense method against FGSM and DeepFool attacks. However, these image compression technologies still cannot serve as an effective defense against more powerful attacks, such as Carlini & Wagner attacks [66]. Moreover, Shin and Song [106] have demonstrated the existence of adversarial examples that can survive JPEG compression.

Similarly, the Display Compression Technology (DCT) compression method [88] used in the fight against universal disturbance attacks [104] has also been proved to be insufficient. In another related approach, Bhagoji et al. [107] proposed to compress input data using Principal Component Analysis for adversarial robustness. However, Xu et al. [108] noted that this compression also results in corrupting the spatial structure of the image, hence often adversely affecting the classification performance.

The biggest limitation of these defense methods based on data compression is that a large amount of compression will lead to a decrease in the accuracy of original image classification, while a small amount of compression is often not enough to remove the impact of disturbance.

A recent method [160] based on image super-resolution proposes the use of deep image restoration networks to learn mapping functions that can bring off-the-manifold adversarial samples onto the natural image manifold, thus restoring classification towards correct classes. A distinguishing feature of this approach is that, in addition to providing robustness against attacks, it simultaneously enhances image quality and retains models performance on clean images.

2.5.1.5 Foveation Based Defense

Luo et al. [12] demonstrated that significant robustness against the adversarial attacks using L-BFGS [3] and FGSM [24] is possible with ‘foveation’ mechanism - applying neural network in different regions of images. It is hypothesized that CNN-based classifiers trained on large datasets, such as ImageNet [109] are generally robust to scale and translation variations of objects in the images. However, this invariance does not extend to adversarial patterns in the images. This makes foveation as a viable option for reducing the effectiveness of adversarial attacks proposed in [3], [24]. However, foveation is yet to demonstrate its effectiveness against more powerful attacks.

2.5.1.6 Data Randomization

Xie et al. [21] demonstrated that the operation of random resizing adversarial samples can reduce the effectiveness of adversarial samples. Similarly, adding some random textures to the adversarial samples can also reduce their deception to the network model. Wang et al. [89] used a data conversion module separated from the network model to eliminate the possible adversarial disturbance in the image, and conducted data expansion operations in the training process, such as adding some Gaussian randomization processing, which could slightly improve the robustness of the network model. In the literature, evidence is also found that data augmentation during training (e.g. Gaussian data augmentation [46]) also helps in improving robustness of neural networks to adversarial attacks, albeit only slightly.

2.5.2 Modifying Model

2.5.2.1 Regularization

This method aims to improve the generalization ability of the target model by adding regular terms which are known as penalty terms to the cost function and make the model have good adaptability to resist attacks on an unknown dataset in prediction. Biggio et al. [90] used a regularization method to limit the vulnerability of data when training the SVM model. The articles [92][93] used regularization method to improve the robustness of the algorithm and achieved good results in resisting adversarial attacks.

Ross and Doshi-Velez [111] studied input gradient regularization [112] as a method for adversarial robustness. Their method trains differentiable models (e.g. deep neural networks) while penalizing the degree of variation resulting in the output with respect to change in the input. Implying, a small adversarial perturbation becomes unlikely to change the output of the trained model drastically. It is shown that this method, when combined with brute-force adversarial training, can result in very good robustness against attacks like FGSM [3] and JSMA [42]. However, each of these methods almost double the training complexity of a network, which is already prohibitive in many cases.

Previously, Lyu et al. [91] also used the notion of penalizing the gradient of loss function of network models with respect to the inputs to incorporate robustness in the networks against L-BFGS [3] and FGSM [24] based attacks. Similarly, Shaham et al. [113] attempted to improve the local stability of neural networks by minimizing the loss of a model over adversarial examples at each parameter update. They minimized the loss of their model over worst case adversarial examples instead of the original data. In a related work, Nguyen and Sinha [114] introduced a masking based defense against C&W attack [66] by adding noise to the logit outputs of networks.

2.5.2.2 Defensive Distillation

Papernot et al. [40] proposed a defensive distillation method to resist attacks on the basis of distillation technology [115]. Distillation was introduced by Hinton et al. [115] as a training procedure to transfer knowledge of a more complex network to a smaller

network. The original distillation technology aims to compress the large-scale model into small-scale and retain the original accuracy, while the defensive distillation does not change the scale of the model, and produces a model with a smoother output surface and less sensitivity to disturbance to improve the robustness of the model. As is shown in Figure 27, they firstly trained an initial network F on data X with a softmax temperature of T , and then used the probability vector $F(X)$, which includes additional knowledge about classes compared to a class label, predicted by network F to train a distilled network F^d at temperature T on the same data X . Softmax is usually used to normalize the last layer of DNN and produce the probability of classes. The softmax output of the first DNN, also the input of the next DNN, can be described as:

$$q_i = \frac{\exp(\frac{z_i}{T})}{\sum_j \exp(\frac{z_j}{T})} \quad (2.18)$$

where T is a *temperature* parameter to control the level of knowledge distillation. In deep neural networks, *temperature* T is set to 1. When T is large, the output of softmax will be vague (when $T \rightarrow \infty$, the probability of all classes $\rightarrow \frac{1}{m}$). When T is small, only one class is close to 1 while the rest goes to 0. This schema of network distillation can be duplicated several times and connects several deep neural networks.

Network Distillation defense was tested on the MNIST and CIFAR-10 datasets and reduced the success rate of JSMA attack by 0.5% and 5% respectively. However, the effectiveness of defense distillation cannot be guaranteed in black-box attacks. Therefore, Papernot et al. [94] proposed extensible defense distillation technology.

Figure 27. An overview of defense mechanism based on a transfer of knowledge contained in probability vectors through distillation [94]

It is worth noting that the ‘Carlini and Wagner’ (C&W) attacks [66] are claimed to be successful against the defensive distillation technique.

2.5.2.3 Defensive Quantization

Another method called defensive quantization (DQ) is similar to this idea in that it discretizes the feature calculation. A novel quantization methodology is designed in [159] to jointly optimize the efficiency and robustness of deep learning models. By controlling the Lipschitz constant of the network during quantization, the magnitude of the adversarial noise remains non-expansive during inference. Extensive experiments on CIFAR-10 and SVHN datasets demonstrate that this new quantization method can defend neural networks against adversarial examples, and even achieves superior robustness than their full-precision counterparts while maintaining the same hardware efficiency as vanilla quantization approaches. As a by-product, DQ can also improve the accuracy of quantized models without adversarial attack.

2.5.2.4 Deep Contractive Network (DCN)

In the early attempts of making deep learning robust to adversarial attacks, Gu et al. [44] introduced a kind of deep compression network, which uses noise reduction automatic encoder to reduce the adversarial noise. It was shown that Denoising Auto Encoders [116] can reduce adversarial noise, however stacking them with the original networks can make the resulting network even more vulnerable to perturbations. Based on this phenomenon, DCN adopted a smoothing penalty similar to a Convolutional Autoencoder (CAE) [117] in the training process, and was proved to have a certain defensive effect against attacks such as L-BGFS [3]. However, many stronger attacks have been introduced since DCNs were initially proposed. A related concept of using auto encoders for adversarial robustness of the neural networks can be also found in [118].

2.5.2.5 Mask Layer Insertion

Gao et al. [45] proposed to insert a mask layer before processing the classified network model. The added layer is explicitly trained by forward-passing clean and adversarial pair of images, and it encodes the differences between the output features of the previous layers for those image pairs. It is generally believed that the most important weight in the additional layer corresponds to the most sensitive feature in the network. Therefore, in the final classification, these features are masked by forcing the additional layers with a primary weight of zero. In this way, the deviation of classification results caused by adversarial samples can be shielded.

2.5.2.6 Parseval Networks

Cisse et al. [46] proposed a network called Parseval as a defensive method against adversarial attacks. This network adopts hierarchical regularization by controlling the global Lipschitz constant of the network. Considering the network can be viewed as a combination of functions at each layer, it is possible to have robust positive ions for small input perturbations by maintaining a small Lipschitz constant for these functions, they proposed to control the spectral norm of the network weight matrix by parameterizing the spectral norm of the network weight matrix through Parseval tight frames [119], so it was called “Parseval” network.

2.5.2.7 Biologically Inspired Protection

Nayebi and Ganguli [98] demonstrated natural robustness of neural networks against adversarial attacks with highly non-linear activations (similar to nonlinear dendritic computations and unlike more linear activation functions such as RELU, which is commonly used in deep learning models). It is noteworthy that the Dense Associative Memory models of Krotov and Hopfield [99] also work on a similar principle for robustness against the adversarial examples. Considering the linearity hypothesis of Goodfellow et al. [3], [98] and [99] seem to further the notion of susceptibility of modern neural networks to adversarial examples being the effect of linearity of activations. It is noted that Brendel and Bethge [120] claim that the attacks fail on the biologically inspired protection [98] only due to numerical limitations of computations. Stabilizing the computations again allow successful attacks on the protected networks.

2.5.2.8 Miscellaneous Approaches

Among other noteworthy efforts in making neural networks robust to adversarial attacks, Zantedeschi et al. [110] proposed to use bounded ReLU [121] to reduce the effectiveness of adversarial patterns in the images. Jin et al. [122] introduced a feedforward CNN that made use of additive noise to mitigate the effects of adversarial examples. Sun et al. [123] proposed ‘HyperNetworks’ that use statistical filtering as a method to make the model robust. Madry et al. [124] studied adversarial defense from the perspective of robust optimization. It is showed that adversarial training with a PGD (projected gradient descent) adversary can successfully defend against a range of other adversaries. Later, Carlini et al. [125] also verified this observation.

Na et al. [126] proposed a network that is regularized with a unified embedding for classification and low-level similarity learning. The network is trained using the distance between clean and the corresponding adversarial embeddings. Strauss et al. [127] studied ensemble of methods to defend a network against the perturbations. Kadran et al. [128] modified the output layer of a neural network to induce robustness against the adversarial attacks. Wang et al. [129], [130] developed adversary resistant neural networks by leveraging non-invertible data transformation in the network. Lee et al. [131] developed manifold regularized networks that use a training objective to minimizes the difference between multi-layer embedding results of clean and adversarial images.

Another group of works that can be seen as adversarial attacks/defenses with guarantees is related to verification of deep neural networks, e.g. [134], [135]. Kolter and Wong [132] and Raghunathan et al. [133] are among very few provable methods in defense against the adversarial attacks. Kotler and Wong [132] proposed to learn ReLU-based classifier that show robustness against small adversarial perturbations. A neural network is trained and it provably achieves high accuracy (about 90%) against any adversary in a canonical setting ($\epsilon = 0.1$ for l_∞ -norm perturbation on MNIST).

Raghunathan et al. [133] studied the problem of defense for neural networks with one hidden layer. Their approach produces a network and a certificate on MNIST dataset

such that no attack perturbing image pixels by at most $\varepsilon = 0.1$ could results in more than 35% test error.

Given that these methods are computationally infeasible to apply on larger networks, the only defenses that have been extensively evaluated are those of Madry et al. [124] giving 89% accuracy against large epsilon (0.3/1) on MNIST and 45% for moderate epsilon (8/255) on CIFAR.

Ororbia et al. [136] show that many different proposals of adversarial training are instances of more general regularized objective, they termed DataGrad. The proposed DataGrad framework can be seen as an extension of layerwise contractive autoencoder penalty.

2.5.2.9 Detection Only Networks

SafetyNet

Lu et al. [137] hypothesized that adversarial examples produce different patterns of ReLU activations in (the late stages of) networks than what is produced by clean images. Based on this hypothesis, they proposed to append a Radial Basis Function SVM classifier to the targeted models such that the SVM uses discrete codes computed by the late stage ReLUs of the network. To detect perturbation in a test image, its code is compared against those of training samples using the SVM. Effective detection of adversarial examples generated by [24], [61], [67] is demonstrated by their framework, named SafetyNet.

Feature Squeezing

Feature squeezing is a model enhancement technique [43], whose main idea is to detect adversarial examples with high accuracy and few false positives by comparing a DNN model's prediction on the original input with that on squeezed inputs. There are two heuristic methods, one is to reduce the color depth at the pixel level, i.e., to encode the color with fewer values; the other is using a smooth filter on the image, i.e., multiple inputs are mapped to a single value, thus making the model safer under noise and

confrontational attack. Although this technique can effectively prevent adversarial attacks, it also reduces the accuracy of the classification of real samples.

Detector Subnetwork

Metzen et al. [138] proposed to augment a targeted network with a subnetwork that is trained for a binary classification task of detecting adversarial perturbations in inputs. It is shown that appending such a network to the internal layers of a model and using adversarial training can help in detecting perturbations generated using FGSM [24], BIM [61] and DeepFool [67] methods. However, Lu et al. [137] later showed that this approach is again vulnerable to counter counter measures.

Exploiting Convolution Filter Statistics

Li and Li [139] used statistics of the convolution filters in CNN-based neural networks to classify the input images as clean or adversarial. A cascaded classifier is designed that uses these statistics, and it is shown to detect more than 85% adversarial images generated by the methods in [3], [140]. In addition, another method called Mean Blur [139] uses the filter to perform mean blurring on the input image and can effectively improve the robustness of models.

Additional Class Augmentation

Generally, there are two kinds of secondary classification detection methods. The first one is adversarial training detector [141], [142], which is similar to adversarial training. The main idea is to add a new classification label to AEs during training. If an AE is detected, the model will classify it into a new class. Hosseini et al. [86] also employed a similar strategy to detect black-box attacks. The second one is to take the characteristics extracted from AEs and original examples during the convolution layer as input. Then the labelled input data is used to train neural network detectors. This method performed well on detecting over 85% of AEs.

Detection Based on Principal Component Analysis (PCA)

The essence of PCA is to transform the original features linearly and map them to a low-dimensional space with the best possible representation of the original features. This method uses PCA in the input layer due to the greater weight of AEs processed by PCA than original examples [68].

Detection Based on Distribution

There are two main distribution-based detection methods. The first one uses the maximum mean discrepancy [141] which measures the distance between two different but related distributions. Assuming that there are two sets of images S_1 and S_2 , S_1 contains all the original examples, S_2 contains either all AEs or all original examples. If S_1 and S_2 have the same distribution, then S_2 has original examples; otherwise, S_2 is full of AEs. The second one uses kernel density estimation [144]. Since AEs have a different density distribution from the original examples, they can be detected with high confidence by the estimation of the density ratio. If the density ratio of one example is close to 1, it belongs to original example. If the density ratio is much larger than 1, it belongs to AEs.

Detection Based on Deep k -Nearest Neighbors

The Deep k -Nearest Neighbors (DkNN) is proposed in [174] to exploit the structure of deep learning to enable new learning-based inference and decision strategies that achieve desirable properties such as robustness and interpretability. This hybrid classifier combines the k -nearest neighbors algorithm with representations of the data learned by each layer of the DNN: a test input is compared to its neighboring training points according to the distance that separates them in the representations. The idea of this method is illustrated in Figure 28. Consider a deep neural network (left), representations output by each layer (middle) and the nearest neighbors found at each layer in the training data (right). Drawings of pandas and school buses indicate training points. Confidence is high when there is homogeneity among the nearest neighbor labels (e.g., here for the unmodified panda image). Interpretability of the outcome of each layer is provided by the nearest neighbors. Robustness stems from detecting nonconformal predictions from nearest neighbor labels found for out-of-distribution

inputs (e.g., an adversarial panda) across different layers. Representation spaces are high-dimensional but depicted in 2D for clarity.

Figure 28. Intuition behind the Deep k-Nearest Neighbors (DkNN) [174]

It is shown that the labels of these neighboring points afford confidence estimates for inputs outside the model’s training manifold, including on malicious inputs like adversarial examples – and therein provides protections against inputs that are outside the models understanding. This is because the nearest neighbors can be used to estimate the nonconformity of, i.e., the lack of support for, a prediction in the training data. The neighbors also constitute human-interpretable explanations of predictions. The DkNN algorithm is evaluated on several datasets, and it is shown that the confidence estimates accurately identify inputs outside the model, and that the explanations provided by nearest neighbors are intuitive and useful in understanding model failures.

Other Detection Methods

Dropout randomization [145] is a method to use dropout randomly during AE detection. Original examples always generate correct labels, but AEs are of high possibility to be different from the label corresponding to original examples.

2.5.3 Network Add-ons

2.5.3.1 Input Reconstruction

Adversarial examples can be transformed to clean data via reconstruction. After transformation, the adversarial examples will not affect the prediction of deep learning models.

PixelDefend

PixelDefend reconstructed the adversarial images back to the training distribution [28] using PixelCNN. PixelDefend changed all pixels along each channel to maximize the probability distribution:

$$\begin{aligned} \max_{x'} P_t(x') \\ \text{s. t. } \|x' - x\|_\infty \leq \epsilon_{defend} \end{aligned} \quad (2.19)$$

where P_t denotes the training distribution, defend controls the new changes on the adversarial examples. PixelDefend also leveraged adversarial detecting, so that if an adversarial example is not detected as malicious, no change will be made to the adversarial examples (defend = 0).

High-Level Representation Guided Denoiser (HGD)

High-level representation guided denoiser (HGD) is proposed as a defense for image classification [97]. Standard denoiser suffers from the error amplification effect, in which small residual adversarial noise is progressively amplified and leads to wrong classifications. HGD overcomes this problem by using a loss function defined as the difference between the target model's outputs activated by the clean image and denoised image. Compared with ensemble adversarial training which is the state-of-the-art defending method on large images, HGD has three advantages. First, with HGD as a defense, the target model is more robust to either white-box or black-box adversarial attacks. Second, HGD can be trained on a small subset of the images and generalizes well to other images and unseen classes. Third, HGD can be transferred to

defend models other than the one guiding it. In 2017 NIPS competition on defense against adversarial attacks, the HGD solution won the first place and outperformed other models by a large margin.

Figure 29. The idea of high-level representation guided denoiser [97]

Three different HGD training methods are proposed as shown in Figure 30. The square boxes stand for data blobs, the circles and ovals stand for networks. D stands for denoiser. The Convolutional Neural Network (CNN) is the model to be defended. The parameters of CNN are shared and fixed.

Figure 30. Three different training methods for HGD [97]

Defense-GAN

Samangouei et al. [72] proposed a mechanism applicable to both white-box and black-box attacks to reduce the efficiency of adversarial perturbation. This method utilizes the

power of generative adversarial network [95], and the main idea is to “project” input images onto the range of the generator G by minimizing the reconstruction error $\|G(z) - x\|_2^2$, prior to feeding the image x to the classifier. Therefore, compared with the adversarial samples, the legitimate samples are closer to the range of G , thus greatly reducing the potential adversarial perturbation. The process of the defense-GAN mechanism is shown in Figure 31. Although defensive-GAN has been proved to be quite effective in defense against attacks, its success depends on GAN’s expressiveness and generative ability. In addition, the training of GAN is challenging, i.e., without proper training, the performance of defense-GAN will obviously decline.

Figure 31. Overview of the Defense-GAN algorithm [72]

MagNet and APE-GAN

MagNet is proposed as a framework for defending neural network classifiers against adversarial examples [96]. MagNet neither modifies the protected classifier nor requires knowledge of the process for generating adversarial examples. MagNet includes one or more separate detector networks and a reformer network. The detector networks learn to differentiate between normal and adversarial examples by approximating the manifold of normal examples. Since they assume no specific process for generating adversarial examples, they generalize well. The reformer network moves adversarial examples towards the manifold of normal examples, which is effective for correctly classifying adversarial examples with small perturbation. The intrinsic difficulties in defending against whitebox attack is discussed and a mechanism to defend against graybox attack is proposed. Inspired by the use of randomness in cryptography, diversity is used to strengthen MagNet. It is shown empirically that MagNet is effective against the most advanced state-of-the-art attacks in blackbox and graybox scenarios without sacrificing false positive rate on normal examples.

Figure 32. An illustration of the reformer’s effect on adversarial perturbations [96]

An effective framework based Generative Adversarial Nets named APE-GAN is proposed and implemented to defend against the adversarial examples in [175]. The experimental results on three benchmark datasets, including MNIST, CIFAR10 and ImageNet indicate that APE-GAN is effective to resist adversarial examples generated from five attacks.

Figure 33. A demonstration of APE-GAN applied to MNIST and CIFAR10 classification networks [175]

Perturbation Rectifying Network

Akhtar et al. [104] proposed a defense framework against adversarial attacks generated using universal perturbations [68]. The framework appends extra ‘pre-input’ layers to the targeted network and trains them to rectify a perturbed image so that the classifier’s prediction becomes the same as its prediction on the clean version of the same image.

The pre-input layers are termed Perturbation Rectifying Network (PRN), and they are trained without updating the parameters of the targeted network. A separate detector is trained by extracting features from the input-output differences of PRN for the training images. A test image is first passed through the PRN and then its features are used to detect perturbations. If adversarial perturbations are detected, the output of PRN is used to classify the test image. Fig. 34 illustrates the rectification performed by PRN. The removed patterns are separately analyzed for detection.

Figure 34. Illustration of defense against universal perturbations: The approach rectifies an image to restore the network prediction. The pattern removed by rectification is separately analyzed to detect perturbation [104]

Defense Against Adversarial Images using Web-Scale Nearest-Neighbor Search

In [173], it is hypothesized that adversarial perturbations move the image away from the image manifold in the sense that there exists no physical process that could have produced the adversarial image. This hypothesis suggests that a successful defense mechanism against adversarial images should aim to project the images back onto the image manifold. A defense framework is proposed to approximate this projection onto the unknown image manifold by a nearest-neighbor search against a web-scale image

database containing tens of billions of images. Empirical evaluations of this defense strategy on ImageNet suggest that it is very effective in attack settings in which the adversary does not have access to the image database.

Figure 35. Illustration of defense procedure for improving adversarial robustness in image classification [173]

At the same time, two novel attack methods are also proposed to break nearest neighbor defenses and demonstrate conditions under which nearest-neighbor defense fails. A series of ablation experiments are performed, which suggest that there is a trade-off between robustness and accuracy in the defenses, that a large image database (with hundreds of millions of images) is crucial to getting a good performance, and that careful construction of the image database to avoid the overlap between the attack and defense is important to be robust against attacks tailored to circumvent the defenses.

As illustrated in [168], the proposed defenses tend to be defeated with some simple improvements in the attack. It is demonstrated in [168] that MagNet and APE-GAN are not effective on the MNIST and CIFAR-10 datasets. It is shown that Mag-Net can be bypassed with greater than 99% success, and the APE-GAN can be bypassed with 100%, with only a slight increase in distortion with the C&W attack. MagNet is defeated by making use of the transferability property of adversarial examples: the adversary trains their own copy of the defense, constructs adversarial examples on their model, and supplies these adversarial examples to the defender. It turns out that these examples will also fool the defender's model. In both cases, adversarial examples are not more visually detectable than an undefended network.

2.5.3.2 Network Verification

Verifying properties of deep neural networks is a promising solution to defend adversarial examples, because it may detect new unseen attacks. Network verification checks the properties of a neural network: whether an input violates or satisfies the property.

Katz et al. proposed a verification method for neural networks with the ReLU activation function, called Reluplex [100]. They used the Satisfiability Modulo Theory (SMT) solver to verify the neural networks. The authors showed that within a small perturbation, there was no existing adversarial example to misclassify the neural networks. They also proved that the problem of network verification is NP-complete. Carlini et al. extended their assumption of ReLU function by presenting $\max(x, y) = \text{ReLU}(x - y) + y$ and $\|x\| = \text{ReLU}(2x) - x$ [125]. However, Reluplex runs very slowly due to the large computation in verifying the networks and only works for DNNs with several hundred nodes [152]. [152] proposed two potential solutions: 1) prioritizing the order of checking nodes 2) sharing information of verification.

Instead of checking each point individually, Gopinath et al. proposed DeepSafe to provide safe regions of a deep neural network [101] using Reluplex. They also introduced the notion of targeted robustness which, for a given target label and region, ensures that a NN does not map any input in the region to the target label.

2.5.3.3 Ensembled Defense

The ensemble of several defense methods can also be viewed as a kind of network addition and it is included in this section. Besides the explicit ensemble method [127], aforementioned PixelDefend [28] is also composed of an adversarial detector and an “input reconstructor” to establish a defense strategy.

MagNet also included one or more detectors and a reconstructor (“reformer” in the paper) as Plan A and Plan B [96]. The detectors are used to find the adversarial examples which are far from the boundary of the manifold. The reformer is bring examples close to the manifold back onto the manifold. Figure 36 illustrates the workflow of the defense of two phases.

Figure 36. MagNet workflow: one or more detectors first detects if input x is adversarial; If not, reconstruct x to x^* before feeding it to the classifier [96]

However, after investigating several defensive approaches, [146] showed that the ensemble of those defensive approaches does not make the neural networks strong.

2.5.3.4 Adversarial Detection based on Feed-forward Networks

MagNet

Meng et al. [96] proposed a framework named MagNet, which treats the output of the last layer of the classifier as a black-box without reading any data of the inner layer or modifying the classifier. MagNet employs a detector to identify legitimate and adversarial samples. The detector measures the distance from a given sample to the manifold and rejects the sample if the distance exceeds a certain threshold. It also uses a reformer to transform the adversarial sample with an automatic encoder into a similar legitimate sample. In white-box attacks, the performance of MagNet decreases significantly because the adversaries is provided with the parameters of MagNet. Therefore, the author proposed the idea of using multiple automatic encoders and randomly selecting one at a time to make it difficult for an adversary to predict which

automatic encoder is used. It is noteworthy that Carlini and Wagner [188] very recently demonstrated that this defense technique can also be defeated with slightly larger perturbations.

Feature Squeezing

Xu et al. [43] uses feature squeezing to detect adversarial perturbation to an image. Two external models are added to the classifier network, such that these models reduce the color bit depth of each pixel in the image and perform spatial smoothing over the image. The predictions of the targeted network with the original and the squeezed images are compared. If a large difference is found between the predictions, the image is determined to be an adversarial example. Whereas [43] demonstrated the effectiveness of this approach against more classical attacks [24], a follow-up work [108] also claims that the method works reasonably well against the more powerful C&W attacks [66]. He et al. [146] also combined feature squeezing with the ensemble method proposed in [147] to show that strength of defenses does not always increase by combining them.

Other Detection Methods

Liang et al. [148] treated perturbations to images as noise and used scalar quantization and spatial smoothing filter to separately detect such perturbations. Gebhart and Schrater [149] viewed neural network computation as information flow in graphs and proposed a method to detect adversarial perturbations by applying persistent homology to the induced graphs.

However, Carlini and Wagner summarized most of these adversarial detecting methods and showed that these methods could not defend against their previous attack C&W's Attack with slight changes of loss function [150], [151].

2.5.3.5 Adversarial Detection based on Feedback Networks

It has been noticed that most of the existing CNN models only learn features through a feedforward structure and no feedback information from top to bottom layers is exploited to enable the networks to refine themselves. With this observation, feedback

has been studied in the context of recurrently boosting the performance based on previous prediction [162][163]. However, the feedback mechanism has not been studied in the context of robustness towards adversarial examples.

The DBFR framework is the first to make use of the high-level feedback combined with the learned features to detect the possible fault of the system with the input of adversarial examples or unusual data.

2.6 Summary

From this intensive literature survey, it can be concluded that the problem of adversarial examples originates from the inherent limitation of the current deep network models. The feedforward only structure of most existing models makes them easy to optimize while at the same time also makes them easy to be attacked.

Although a lot of defense methods have been proposed, the problem cannot be solved by simple improvements either in the data (by adversarial training), the training process (by adding regulation terms) or small changes to the network structure (Perseval network or the propose add-ons).

It is also observed that a complete solution is difficult for the supervised-learning based systems. In a practical sense, for any methods aiming at a complete solution, a new attack method is very likely to be designed to defeat it based on counter-counter measures. The detection only approach, however, is based on the assumption that there is a distinguishable difference between the distribution of adversarial examples and natural images. There is no theoretical proof of this assumption but it is supported by the fact that human beings are not easily fooled by the adversarial examples. Thus the detection only approach is more promising and may pave the way towards a full solution.

It is observed that deep learning systems make decision mostly based on combinations of local features, instead of overall structure [171]. Although feedback is introduced in some attention-based methods in the aim to solve this problem [162][163], the results

are not effective against the adversarial examples. It is valuable to study the feedback system in a higher level, the full image semantic information.

As shown in this study, different from all the existing methods, adding a high-level feedback (the full image semantic label) to the existing neural network, together with a reconstruction module, to detect the “unseen input” solves the problem in another dimension. The results show that the detection is very effective regardless of the adversarial attack methods.

CHAPTER 3 METHODOLOGY OF DBFR FRAMEWORK

This thesis implements DBFR framework - an image recognition system with the ability of detecting input data that does not follow the distribution of the training data, such as adversarial examples and “unseen” data, and preventing the blindly output of wrong recognition results. The detection is based on a high-level feedback mechanism using a class-specific reconstruction process and a quantitative comparison between the reconstructed and the input images. Before the classifier outputs the inference result – class ID, a reconstruction based on this hypothesis will be done as a confirmation process. The reconstructed image is compared to the input image to double check whether the recognition result agrees with what is learned from images for the same class ID. If a natural image that falls onto the manifolds of the training data is input, the classifier outputs the correct class ID. Consequently, the reconstruction based on features learned from the training data for this class ID will generate an image very close to the input image. The comparison of the reconstructed and input images gives a high score measuring the similarity. However, in the case of an adversarial example or unseen image as input, the classifier outputs a wrong class ID, then the reconstructed image based on mapping learned from the training data for this specific class ID will be much different from the input image. The comparison gives a low score measuring the similarity between the reconstructed and input images. Thus using the DBFR method, the system can detect a fault in its own classification output.

The reconstructed image serves as an intuitive feedback from the network while it recognizes the input image. It not only enables a fault detection mechanism but also reveals what has been learned by the original recognition network. The reconstruction networks for all the existing classes form an internal model of the image recognition system. With this framework, not only the fault in reported recognition results can be detected, unseen data that is not covered in the training dataset can also be detected.

This chapter gives a formal formulation of the problem, details of the DBFR network structure, the training process of the reconstruction networks and the quantitative measurement of similarity.

3.1 Formulation of the Problem

Formulation of Classification Problem

$[0,1]^n$ is used to denote the unit hypercube in n dimensional space. The pixel value of a digital image can be normalized to $[0,1]$ and an image can be then represented by a point in this hypercube. For an image with pixel number of $W \times H$ and channel number of L ($L = 1$ for grayscale images), the number $n = W \times H \times L$. The problem of classification is to map data points that lie in a space Ω , the hypercube for image space, into m different object classes. The m object classes are defined by probability density functions $\{\rho_c\}_{c=1}^m$, where $\rho_c: \Omega \rightarrow \mathbb{R}$. A random point from class c is a random variable with density ρ_c .

The “classifier” function f is defined as $f: \Omega \rightarrow \{1, 2, \dots, m\}$ that partitions Ω into disjoint measurable subsets, one for each class label. The classifier we considered is discrete valued – it provides a label for each data point but not a confidence level.

Formulation of Adversarial Attack Problem

In theory, adversarial examples are any confusing image that “tricks” the recognition system and can be constructed from anywhere in the full input image space. For example, some random and meaningless images could be recognized by the system with high confidence. However, in practice, almost all the attacks are based on modification of natural images, that is, adding small perturbation to meaningful images to trick the recognition system.

Given a trained deep learning model f and an original input data sample $\mathbf{x} \in \Omega$, generating an adversarial example $\mathbf{x}' \in \Omega$ can generally be described as a box-constrained optimization problem. Depending on the knowledge of f , the attacks can be grouped as white or black. Depending on whether the target class label l' is specified, the attack can be categorized as targeted and non-targeted.

$$\min_{\mathbf{x}'} \|\mathbf{x}' - \mathbf{x}\|_p \quad (3.1)$$

$$\begin{aligned}
& s. t. f(\mathbf{x}') = l' \\
& f(\mathbf{x}) = l \\
& l \neq l' \\
& \mathbf{x} \in [0,1]^n \text{ and } \mathbf{x}' \in [0,1]^n
\end{aligned}$$

where $l \in \{1, 2, \dots, m\}$ and $l' \in \{1, 2, \dots, m\}$ denote the output label of $\mathbf{x}$ and $\mathbf{x}'$, $\|\cdot\|_p$ denotes the distance between two data sample using p -norm. For the perturbation, it is defined as $\boldsymbol{\eta} = \mathbf{x}' - \mathbf{x}$ to be added on $\mathbf{x}$.

It is difficult to find explicit solutions of Equation 3.1 as DNN model f is nonlinear and very complex. Instead, most of the attack methods are iteration based.

As theoretical analysis such as [37] reveals, the defense problem is an ill-posed problem because the existence of adversarial examples is inevitable without constraint on the adversarial strength. Almost all existing discussion on adversarial examples are in the context with a limitation on the adversarial strength: the difference between an adversarial example and the original natural image is imperceptible.

Since it is hard to model human perception, researchers proposed three popular metrics to approximate human's perception of visual difference, namely l_0 , l_2 , and l_∞ . These metrics are special cases of the l_p norm:

$$\|\boldsymbol{\eta}\|_p = (\sum_{i=1}^n |\eta_i|^p)^{\frac{1}{p}} \quad (3.2)$$

These three metrics, with $p = 0, 2$ and ∞ respectively, focus on different aspects of visual significance. l_0 counts the number of pixels with different values at corresponding positions in the two images. It answers the question of how many pixels are changed. l_2 measures the Euclidean distance between the two images. l_∞ measures the maximum difference for all pixels at corresponding positions in the two images.

Based on psychological studies, the difference measured in l_0 , l_2 , and l_∞ should be no more than 15 for the 8-bit image input to be unnoticeable by a human being.

Formulation of Defense Problem

The ultimate target is to find a robust classifying neural network that satisfies:

$$\begin{aligned} f(\mathbf{x}) &= l \\ f(\mathbf{x}') &= l, \text{ for any } \mathbf{x}' \text{ satisfying } \|\mathbf{x}' - \mathbf{x}\| < \delta_0 \end{aligned} \quad (3.3)$$

The norm may be measured in l_0 , l_2 , and l_∞ and a rough estimation is $\delta_0 = 15$ for a 8-bit pixel value or in normalized format $\epsilon_0 = \frac{15}{255}$.

Formulation of Different Approaches of Defense

For methods based on modifying data, they can be formulated as a transfer g so that for any $\mathbf{x}'$ satisfying the perturbation strength constraint $\|\mathbf{x}' - \mathbf{x}\| < \epsilon_0$

$$f(g(\mathbf{x}')) = l \quad (3.4)$$

For methods based on modifying network, it can be formulated as changing f to $f' = f + \delta(f)$, so that for any $\mathbf{x}'$ satisfying the perturbation strength constraint $\|\mathbf{x}' - \mathbf{x}\| < \epsilon_0$, $f'(\mathbf{x}') = l$.

The most common changes in f is through adversarial training, without changing the network structures. In other words, the adversarial training changes f^ω to $f^{\omega'}$, where ω and ω' are the parameters for the network with and without adversarial training.

For the general detection methods, they can be formulated as finding a function D for:

$$D(\mathbf{x}') = 1$$

for any adversarial examples satisfying the conditions, while

$$D(\mathbf{x}) = 0$$

for any natural images. The function D can be dependent or independent on f based on whether the defense is working on the data part, or it is modifying the network or using add-ons.

The DBFR method is based on an estimated inversion of feature extraction f_E , which is a part of the classification function f . The classification function is formulated as:

$$f = f_C(f_E(\mathbf{x})) \quad (3.5)$$

where f_E is the feature extraction part and f_C is the classification part.

The DBFR method is based on a group of class-wise reconstruction function $\widetilde{f_{E,l}^{-1}}$ as an approximation to the inversion of f_E . For each possible class label l , a separate reconstruction function is obtained based on the training data that belongs to this class. These reconstruction functions have the property of:

$$|\widetilde{f_{E,l'}^{-1}}(f_E(\mathbf{x}')) - \mathbf{x}'| > d_0 \quad (3.6)$$

For any adversarial example $\mathbf{x}'$ which makes $f_C(\mathbf{x}') = l'$, while for any non-adversarial input $\mathbf{x}$ which belongs to the class l :

$$|\widetilde{f_{E,l}^{-1}}(f_E(\mathbf{x})) - \mathbf{x}| < d_0 \quad (3.7)$$

where d_0 is a threshold for the detection and $|\cdot|$ is a measurement between the reconstructed and input images. In the implementation, the feature extraction f_E is fixed for the original recognition system and the set of $\widetilde{f_{E,l}^{-1}}$ is realized as decoder networks trained on the training data with the same class label i respectively for $i \in \{1, 2, \dots, m\}$. The most common choices for comparison $|\cdot|$ are l_1 difference or Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity (SSIM) measures.

3.2 Introduction of the DBFR Method

By the categorization method in Section 2, the DBFR method is a detection only approach based on Network add-ons.

This method is inspired by the idea of ‘‘Understanding Deep Image Representations by Inverting Them’’ [172]. In [172], a direct analysis of the visual information contained in representations is conducted by asking the following question: given an encoding of an image, to which extent is it possible to reconstruct the image itself? To answer this question, a general framework to invert representations is proposed. It is shown that this method can invert representations such as HOG and SIFT more accurately than recent alternatives while being applicable to CNNs too. Then this technique is used to study the inverse of recent state-of-the-art CNN image representations for the first time.

Figure 37 shows five possible reconstructions of the reference image obtained from the 1,000-dimensional code extracted at the penultimate layer (the layer before the softmax is applied) of a reference CNN trained on the ImageNet data. From the viewpoint of the model, all these images are practically equivalent. By examining the different copies of reconstructed images, which are nearly indistinguishable from the viewpoint of the CNN model, the network is shown to capture just a sketch of the objects, which evidently suffices for classification.

Figure 37. What is encoded by a CNN? [172]

However, as information is abandoned through the classifying network, the reconstruction is neither unique nor pixel-level authentic thus a quantitative comparison with the original input is impossible.

The DBFR method is also related to the concept of Deep k-Nearest Neighbors [174], which combines the k-nearest neighbors algorithm with representations of the data learned by each layer of the DNN: a test input is compared to its neighboring training points according to the distance that separates them in the representations. The labels of these neighboring points afford confidence estimates for inputs outside the model's training manifold, including malicious inputs like adversarial examples. This is because the nearest neighbors can be used to estimate the nonconformity of, i.e., the lack of support for, a prediction in the training data. The neighbors also constitute human-

interpretable explanations of predictions. The DBFR method can be viewed as further development of this idea to provide a reconstructed image as the confidence estimates for the model's training manifold, which also serves the purpose of human-interpretable explanations.

The DBFR method aims to use an additional network to learn the abandoned information in the classifier and reconstruct the exact input based on both the high-level feedback (the class ID) from the existing neural network, the high level feature learned in the classifier and complement features learned by an additional reconstruction branch.

The idea of image reconstruction from the semantic feature vectors have been tested in [180]. The objective is to verify whether it is possible to reconstruct the original image from the semantic feature vector. A convolutional neural network architecture is designed based on Deep Convolutional Adversarial Generative Network (DCGAN) [182] to generate the images. The conclusion is that the reconstruction of training data is possible, however, the reconstruction is not able to reconstruct personal information because it is only able to reconstruct main concepts of the image and previously seen details. The DBFR method is also based on image reconstruction from the semantic feature vectors. However, with different decoders for specific semantic concepts and use of SSIM to measure the similarity with the input, the DBFR method is to enable the image recognition system to detect its own fault.

The collaboration of image classification and reconstruction has been studied in [170]. HybridNet is proposed for leveraging unlabeled data to improve generalization performances of image classifiers. HybridNet is a two-branch encoder-decoder architecture. The first branch receives supervision signal and is dedicated to the extraction of invariant class-related representations. The second branch is fully unsupervised and dedicated to model information discarded by the first branch to reconstruct input data. To further support the expected behavior of this model, an original training objective is proposed. It favors stability in the discriminative branch and complementarity between the learned representations in the two branches. HybridNet is able to outperform state-of-the-art results on CIFAR-10, SVHN and STL-10 in various semi-supervised settings. In addition, visualizations and ablation studies

validate our contributions and the behavior of the model on both CIFAR-10 and STL-10 datasets.

Figure 38 illustrates the behavior of HybridNet. the input image is processed by two network paths of weights W_c and W_u ; each path produces a partial reconstruction, and both are summed to produce the final reconstruction, while only one path is used to produce a classification prediction. Thanks to a joint training of both tasks, the weights W_c and W_u influence each other to cooperate.

Figure 38. Illustration of HybridNet behavior [170]

However, the HybridNet uses two networks independently with the aim to improve the classification network's performance. It has not been studied in the presence of adversarial examples and there is no concept of feedback mechanism.

It has been noticed that most of the existing CNN models only learn features through a feedforward structure and no feedback information from top to bottom layers is exploited to enable the networks to refine themselves. With this observation, feedback has been studied in the context of recurrently boosting the performance based on previous prediction.

[162] proposes a way to simulate feedback loops in the brain by unrolling loopy neural networks several timesteps, and the properties of these networks are investigated. Different variants of loops, including multiplicative composition of inputs and additive

composition of inputs are compared. It is demonstrated that loopy networks outperform deep feedforward networks with the same number of parameters on the CIFAR-10 dataset, as well as non-loopy versions of the same network, and perform equally well on the MNIST dataset. Figure 39 illustrates the loopy network structure and its unrolled version.

Figure 39. CIFAR-10 loopy model with loop layer. Unrolled network on the right.

Image from [162]

A "Learning with Rethinking" algorithm is proposed in [163]. By adding a feedback layer and producing the emphasis vector, the model is able to recurrently boost the performance based on previous prediction. Particularly, it can be employed to boost any pre-trained models. This algorithm is tested on four object classification benchmark datasets: CIFAR-100, CIFAR-10, MNIST-background-image and ILSVRC-2012 dataset. These results have demonstrated the advantage of training CNN models with the proposed feedback mechanism. Figure 40 illustrates the pipeline of this method. With a pre-trained model, several emphasis layers are inserted. Initial emphasis vectors in $T=1$ are fixed for all 1, and other emphasis vectors are calculated from feedback layers in the following iterations. The emphasis layers then alter the response of corresponding feature maps through the inserted emphasis layers. The total loss during training is the sum of all l_t with equal weights. The blue solid and red dash arrows refer to the forward pass and the backward pass respectively.

However, the feedback mechanism has not been studied in the context of robustness towards adversarial examples. They are different from the DBFR method which uses a high-level feedback mechanism as a counter measure for the adversarial examples.

Figure 40. Illustration of the time-unfolded overall pipeline [163]

Different from all the above methods, the DBFR method employs a reconstruction process based on high-level feedback to verify whether the classification result agrees with the class-specific features learned on the training data of the particular class. This is a new framework with a verification mechanism based on reconstruction. Compared to existing feedforward framework that is illustrated in black color, this new framework has additional blocks and data flow as illustrated in blue color in Figure 41. This new framework endows the recognition system the ability of learning its own boundary and raise a red flag when the reconstruction result does not agree with the recognition.

Figure 41. The new framework with fault detection based on reconstruction

Figure 42 illustrates the structure of the DBFR method. First of all, the proposal of the original image recognition system serves as a high-level feedback to choose the reconstruction network to be used for the verification. Then, by comparing the reconstructed image, generated by the specific reconstruction network, with the original image, whether the classification result is supported by the training data can be detected.

With an adversarial or unseen input that does not fall on the manifold formed by the training data, the reconstructed image using the selected reconstruction network is far different from the original input. Using a proper parameter to quantify the similarity between the reconstructed and input images, the adversarial or unseen input is detected and the system will inhibit output of the “misleading” result and trigger higher level of intervention.

Figure 42. The add-on network DBFR for detection

3.3 Details of the DBFR Method

Without modification on the original classification network, the DBFR method trains a set of reconstruction networks as add-on, whose weights are dependent on the class ID output from the classification network. During the inference, the proposal of the class ID output from the original image recognition system works as a high-level feedback to determine which reconstruction network to be used for the reconstruction. The selected reconstruction network then takes the feature extracted from the input image by the feature extractor in the original image recognition system as input and reconstructs an image based on the class-specific mapping from feature space to image space learned during the training stage. By comparing the reconstructed image with the original input, the similarity in their contents reflects whether the image recognition result is supported by the information learned during the training.

3.3.1 Architecture of the Network for Reconstruction

The reconstruction networks use an architecture following the concept of Autoencoder [176]. Note shall be taken that the contracting branch of an Autoencoder is different from that of the common image recognition system as it is designed to extract the features “losslessly”. However, the contracting branch of DBFR system shares the same structure as the original classification network, thus the process of feature extraction is lossy. A expanding path following the structure of “decoder” part of an Autoencoder is added on the contracting branch to enable the image reconstruction from the lossy features extracted. Each contracting branch and an expanding path form a reconstruction network. Each reconstruction network uses different set of weights for both feature extractor and decoder that are learned by a separate training process on each specific class of training data.

Figure 43 shows the architecture of the DBFR network. The top network is the existing classifier that is composed of the feature extraction part E_c and fully connected part C . The feature extraction part is also known as “encoder”, which is composed of several concatenated blocks of a convolutional/ReLU layer followed by a pooling layer. The extracted feature h_c is then input into a fully connected part followed by a softmax to generate the classification result $\hat{y}$. This part is fixed in the full process. The lower part is

a set of expanding branches D_i , also known as “decoder”, that takes h_c obtained in the classifier network as the input and learns the missing features to reconstruct the image based on the same semantic meaning, i.e. the same class ID. That is, one decoder for each class is trained for the reconstruction of the image.

Figure 43. Architecture of the reconstruction networks

In the inference stage, the reconstruction network will take the decision of the original classification network as input to select the corresponding reconstruction network. It is the combination of the encoder with the decoder that determines how the reconstructed image looks like. The reconstructed image is then compared to the input image to verify whether the decision of class ID agrees with the learning of the corresponding decoder.

In practice, it is more reliable to compare the reconstructed image from each decoder to the input image and analyze the vector formed by the SSIM measurement, as illustrated in Figure 44. The combination of all reconstruction networks forms the total internal model of the image recognition system and the analysis on V_{SSIM} gives a better judgement for both adversarial and unseen images.

Figure 44. Analysis based on reconstructed images from all encoders

3.3.2 Training of the Reconstruction Network

By design, the contracting (feature extractor) branch is not trainable and the weights are the same as those of the corresponding part in the original classification network.

Only the expanding branches (decoders) are trainable and reconstruction loss shall be used. The reconstruction loss can be a simple L-1 ($p = 1$) or L-2 ($p = 2$) distance between the input and the reconstructed images.

$$l_{rec} = \|\hat{x} - x\|^p \quad (3.8)$$

Note should be taken that a different set of weights for every decoder in each reconstruction network will be learned for each class. This is to learn the different mapping from the lossy feature space to reconstruct images for each class.

This is different from an Auto-encoder which uses the same encoder and decoder for any input. Auto-encoders are trained unsupervisedly to learn compact feature representation while in the DBFR method, the reconstruction network is trained supervisedly to learn the mapping from the lossy feature space to the image space for each specific class (high-level concept).

Though it may seem troublesome to have many decoder networks, it is necessary because the feature extracted from the original image recognition system is lossy and different decoder networks learn the class-specific mapping from the lossy feature space to the original image space.

3.3.3 Quantitative Comparison of Reconstructed and Input Images

The DBFR method detects anomaly by comparing the reconstructed image with the original input image. Two commonly used image comparing methods are PSNR and SSIM.

PSNR is most easily defined via the mean square error (MSE). Given a noise-free $m \times n$ monochrome image I and its noisy approximation K , MSE is defined as:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \quad (3.9)$$

The PSNR (in dB) is defined as:

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right) = 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE) \quad (3.10)$$

Here MAX_I is the maximum possible pixel value of the image.

The SSIM formula is based on three comparison measurements between the samples of x and y : luminance (l), contrast (c) and structure (s). The individual comparison functions are:

$$\begin{aligned} l(x, y) &= \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \\ c(x, y) &= \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \\ s(x, y) &= \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3} \end{aligned} \quad (3.11)$$

where μ_x is the average of x , μ_y is the average of y , σ_x^2 is the variance of x , σ_y^2 is the variance of y , σ_{xy} is the covariance of x and y . $c_1 = (k_1L)^2$ and $c_2 = (k_2L)^2$ are two variables to stabilize the division with weak denominator. L is the maximum pixel value of the image, $k_1 = 0.01$ and $k_2 = 0.03$ by default. $c_3 = c_2/2$.

SSIM is then a weighted combination of these comparative measures:

$$SSIM(x, y) = [l(x, y)^\alpha c(x, y)^\beta s(x, y)^\gamma] \quad (3.12)$$

When setting α, β, γ to 1, the SSIM index is a special case of SSIM measurement as:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)} \quad (3.13)$$

As there is information loss in the encoding process, the reconstructed images can never be the same as the original images. In fact, the resulting images are blurry even with clean images as input. For this application scenario, SSIM metric is a good choice to capture the essence in the reconstruction, while ignoring the details.

When the adversarial or unusual input image causes incorrect classification result for the original network, the decoder selected according to this decision for reconstruction generates an image significantly different from the input. A proper threshold value on SSIM measurement is used to detect this fault effectively.

After the detection, a more thorough analysis on V_{SSIM} which reflected the similarity between reconstructed image from each class with that of the input image can be done to further differentiate whether it is an adversarial attack or an unseen image.

CHAPTER 4 EXPERIMENT RESULTS

This chapter summarizes experimental results for the DBFR method.

The original classification network is based on the Resnet-50 backbone and pretrained on ImageNet dataset. This is a widely used network model for many classification tasks. To simplify the problem in discussion, its average pooling layer and linear dense layer are fine-tuned for a binary classification. The feature extractor (“encoder”) part is kept intact. With this simplification, only two decoders need to be trained. The structure of the decoders follows that of autoencoders which reconstruct the image representations from the extracted features.

The first experiment is on the reconstruction results with different reconstruction models. This is to verify that the two decoders have learned different reconstruction mapping from the feature space during training based on different semantic settings. It is interesting to note that the same feature vector can be mapped to different images using the two encoders. This forms the basis for using semantic feedback in reconstruction to detect the fault in the classification results.

The second experiment is on the detection of fault in the recognition process. Two kinds of data are tested. The first one is the adversarial attacks generated by various attacking methods and the second one is the unseen data, for which human images are used as a good example. The results show that the proposed method works for both kinds of data and detects the fault in recognition for both cases effectively. For the adversarial attacks, the detection rate is higher than the state-of-the-art algorithms.

Two datasets, namely GTSRB (German Traffic Sign Recognition Benchmark) traffic sign dataset and Kaggle Dogs vs. Cats dataset, were used in the experiment to verify the proposed methods. The use of these two datasets is to cover different application areas and complexity of problems. For the GTSRB dataset, class 33 ("Turning right ahead", referred to as "Right" in this chapter) and 34 ("Turning left ahead", referred to as "Left" in this chapter) are used for binary classification. This classification is referred to as Left vs. Right in the experiment.

4.1 Network Implementation Details

For the classification network, the backbone of ResNet-50 is used. The weights of the feature extractor are fixed as those pre-trained on ImageNet dataset and only the fully connected network is fine-tuned on the test datasets.

For the Left vs Right dataset, transfer learning technique is applied by fine-tuning the pretrained ResNet50 on ImageNet and reached 98.2% Top-1 accuracy on the test dataset.

Similar fine-tuning is applied on Dogs vs Cats dataset as well and Figure 45 shows the training results output from TensorBoard on Dogs vs Cats dataset along the training epochs. It shows that after fine-tuning for about 15 epochs, the accuracy on dog-cat dataset is 99.4% on clean images.

Figure 45. Fine-tuning of the classification network based on ResNet-50

For the reconstruction networks, the encoder weights are the same of the classification network's feature extractor and are kept fixed. The decoder part is similar to the common design of image decoder from autoencoder and the weights are trainable.

Detailed decoder structures for the two problems (Left vs Right and Dogs vs Cats) are shown below in Figure 46. It shows the detailed definition of each layer in the decoders, from the block directly connected to the feature extractor (UnetGeneratorBlock: 1-2 in a) and ConvTranspose2d-2 in b)) to the final output layer (AvgPool2d: 1-8 in a) and UnetGeneratorBlock-20 in b)) with the same size of the input image. The decoder structures for two problems are the same and only the sizes of each layer are different to match with the different input image sizes (33 x 33 and 256 x 256 respectively).

Layer (type:depth-idx)	Output Shape	Param #
Resnet: 1-1	[1, 2]	--
ResNet: 2	[1]	--
Conv2d: 3-1	[1, 64, 16, 16]	(9,408)
BatchNorm2d: 3-2	[1, 64, 16, 16]	(128)
ReLU: 3-3	[1, 64, 16, 16]	--
MaxPool2d: 3-4	[1, 64, 8, 8]	--
Sequential: 3-5	[1, 256, 8, 8]	(215,808)
Sequential: 3-6	[1, 512, 4, 4]	(1,219,584)
Sequential: 3-7	[1, 1024, 2, 2]	(7,098,368)
Sequential: 3-8	[1, 2048, 1, 1]	(14,964,736)
Linear: 3-9	[1, 2]	(4,096)
UnetGeneratorBlock: 1-2	[1, 2048, 2, 2]	--
Sequential: 2-1	[1, 2048, 2, 2]	--
ConvTranspose2d: 3-10	[1, 2048, 2, 2]	67,108,864
ReLU: 3-11	[1, 2048, 2, 2]	--
BatchNorm2d: 3-12	[1, 2048, 2, 2]	4,096
UnetGeneratorBlock: 1-3	[1, 1024, 4, 4]	--
Sequential: 2-2	[1, 1024, 4, 4]	--
ConvTranspose2d: 3-13	[1, 1024, 4, 4]	33,554,432
ReLU: 3-14	[1, 1024, 4, 4]	--
BatchNorm2d: 3-15	[1, 1024, 4, 4]	2,048
UnetGeneratorBlock: 1-4	[1, 512, 8, 8]	--
Sequential: 2-3	[1, 512, 8, 8]	--
ConvTranspose2d: 3-16	[1, 512, 8, 8]	8,388,608
ReLU: 3-17	[1, 512, 8, 8]	--
BatchNorm2d: 3-18	[1, 512, 8, 8]	1,024
UnetGeneratorBlock: 1-5	[1, 256, 16, 16]	--
Sequential: 2-4	[1, 256, 16, 16]	--
ConvTranspose2d: 3-19	[1, 256, 16, 16]	2,097,152
ReLU: 3-20	[1, 256, 16, 16]	--
BatchNorm2d: 3-21	[1, 256, 16, 16]	512
UnetGeneratorBlock: 1-6	[1, 3, 32, 32]	--
Sequential: 2-5	[1, 3, 32, 32]	--
ConvTranspose2d: 3-22	[1, 3, 32, 32]	12,291
Tanh: 3-23	[1, 3, 32, 32]	--
ReplicationPad2d: 1-7	[1, 3, 33, 33]	--
AvgPool2d: 1-8	[1, 3, 32, 32]	--

a) Decoder structure for Left vs Right problem

Layer (type)	Output Shape	Param #
ReLU-1	[-1, 2048, 8, 8]	0
ConvTranspose2d-2	[-1, 2048, 16, 16]	67,108,864
BatchNorm2d-3	[-1, 2048, 16, 16]	4,096
UnetGeneratorBlock-4	[-1, 2048, 16, 16]	0
ReLU-5	[-1, 2048, 16, 16]	0
ConvTranspose2d-6	[-1, 1024, 32, 32]	33,554,432
BatchNorm2d-7	[-1, 1024, 32, 32]	2,048
UnetGeneratorBlock-8	[-1, 1024, 32, 32]	0
ReLU-9	[-1, 1024, 32, 32]	0
ConvTranspose2d-10	[-1, 512, 64, 64]	8,388,608
BatchNorm2d-11	[-1, 512, 64, 64]	1,024
UnetGeneratorBlock-12	[-1, 512, 64, 64]	0
ReLU-13	[-1, 512, 64, 64]	0
ConvTranspose2d-14	[-1, 256, 128, 128]	2,097,152
BatchNorm2d-15	[-1, 256, 128, 128]	512
UnetGeneratorBlock-16	[-1, 256, 128, 128]	0
ReLU-17	[-1, 256, 128, 128]	0
ConvTranspose2d-18	[-1, 3, 256, 256]	12,291
Tanh-19	[-1, 3, 256, 256]	0
UnetGeneratorBlock-20	[-1, 3, 256, 256]	0

b) Decoder structure for Dogs vs Cats problem

Figure 46. Detailed structure of the decoder networks

The overall network structure is similar to what is described in [180], in which the same decoder is used for all classes of images, taking the feature extracted from the feature extractor in the original classification network. The result in [180] shows that detailed image cannot be recovered from the feature vector extracted in ResNet 50. However, in the DBFR method, different reconstruction networks are trained for different classes and the result shows that meaningful images can be reconstructed when the correct reconstruction network is used and nonsense images will be constructed when a wrong reconstruction network is used. Note should be taken that the reconstructed image is with a lower resolution compared to the input image, even when the correct decoder is used, because the feature extractor (“encoder”) is lossy.

For the experiment on Left vs Right dataset, only two reconstruction networks need to be trained respectively. During the training, l_1 loss (MAE loss) is used to generate the best result. Only images of the same class are used for the training of the reconstruction network for the particular class. For the experiment on Dogs vs Cats dataset, again two reconstruction networks need to be trained respectively. However, during the training, l_2 loss (MSE loss) is used to achieve the best results and again only images of the same class is used for the training of the reconstruction network for the particular class.

4.2 Input

A lot of existing defense methods are using MNIST or CFAR as the testing data set. However, these two sets are quite different from real applications in both the image size and content complexity. In this study, more realistic images are used. Two sets of data are used for the experiment: 1) GTSRB Traffic Sign dataset and 2) Dogs vs. Cats dataset from Kaggle.

4.2.1 Dataset of Traffic Signs GTSRB

For the first dataset, to simply the problem without losing generality, class 33 ("Turning right ahead", referred to as "Right" below) and class 34 ("Turning left ahead", referred to as "Left" below) from GTSRB (<https://benchmark.ini.rub.de/>) are used for binary

classification. Class 17 ("No entry", referred to as "Unseen" below) to simulate unseen data. The training dataset contains 481 Right images and 298 Left images. The validation dataset contains 137 Right images and 83 Left images. And the test dataset contains 71 Right images and 39 Left images.

One of the purposes of the DBFR method is to build an “internal model” for the decision process so that the image recognition system is able to detect its own fault. The fault normally happens for two scenarios: adversarial input and unseen data.

The adversarial images are generated from clean images of the GTSRB dataset with various attacking methods implemented in the Cleverhans [177], a Python library to benchmark machine learning systems’ vulnerability to adversarial examples. The experiment is focused on CW- L_2 attack, which is the most difficult among all existing methods. 63 adversarial Right images are generated respectively using the Cleverhans library in the experiment.

For unseen data, 1109 Stop sign images are used as unseen data to test the image recognition system for the GTSRB dataset. For a traditional classification network, either “Left” or “Right” label will be output as result, however, by comparing the reconstructed image to the original image following the proposed method, it is possible to reason that the input image is neither of them with a high probability.

4.2.2 Dataset of Dogs vs. Cats

For the second experiment, Dogs vs. Cats dataset on Kaggle is used for the experiments (<https://www.kaggle.com/c/dogs-vs-cats/data>). From the total of 25,000 images of cats and dog in the dataset, 2,500 cat images and 2,500 dog images are used as test images while the others are using for training.

The adversarial images are generated from clean images of the Dogs vs. Cats dataset. The experiment is focused on CW- L_2 attack, which is the most difficult among all existing methods. 500 adversarial cat images and 500 adversarial dog images are generated respectively using the Cleverhans library for the experiment.

For unseen data, 2,500 randomly selected Align & Cropped human face images from CelebA dataset are used as unseen images for the Dogs vs Cats dataset. For a traditional classification network, either “cat” or “dog” label will be output as result, however, by comparing the reconstructed image to the original image following the proposed method, it is possible to reason that the input image is neither of them with a high probability.

4.3 Results

4.3.1 Dataset of Traffic Signs GTSRB

Reconstruction

The first experiment is on the reconstruction of images using different reconstruction networks. Figure 47 illustrates the images reconstructed with two different reconstruction networks using the same input image to the recognition system. In each row, the left one is the input image, the center one is the reconstructed image with “Left” reconstruction network and the right one is the reconstructed image with “Right” reconstruction network. It is clearly seen that with the correctly matched reconstruction network, the reconstructed image is much closer to the one reconstructed with the wrong reconstruction network.

Figure 47. Reconstructed images using the two reconstruction networks respectively

As there is information loss in the “encoder”, the reconstructed image will have less details when compared to the original image. In this case, SSIM is found to be effective in quantifying this similarity.

The statistics of this comparison on test data are shown in Table 5 and Table 6 respectively. Table 5 shows the mean SSIM values when comparing the reconstructed image using different decoders to the input images. For example, the value 0.4328 is the average of the SSIM values of all the 39 test cat images compared to the respective reconstructed image with the “Left decoder”. Table 6 shows the standard deviation of the SSIM values correspondingly. Note that both the example images and the SSIM measurements are done on test images instead of training images, in order to illustrate the real ability learned during the training process. PSNR has also been tested to measure the similarity, but the results show that PSNR cannot give meaningful results as SSIM.

The reconstruction of different reconstruction networks is highly effective. With 39 test Left images, only 15.4% of “Left reconstruction network” images have lower SSIM than “Right reconstruction network” images. With 71 test Right images, only 15.5% of the “Right reconstruction network” images have lower SSIM than “Left reconstruction network” images. These SSIM values are significant, however not impressive when compared to the results of the Dogs vs Cats dataset which is going to be discussed in

the next section. The possible reason could be the limited image size of the dataset that affects the effectiveness in the SSIM measurement.

		Input	
		Left	Right
reconstruction network	to Left	0.4328	0.1955
	to Right	0.2162	0.3641

Table 5. Mean SSIM values for images reconstructed from Left and Right decoders

		Input	
		Left	Right
reconstruction network	to Left	0.1826	0.1114
	to Right	0.1392	0.1482

Table 6. STD of SSIM values for images reconstructed from Left and Right decoders

Adversarial Input

Various kinds of adversarial attack methods are used to generate the adversarial inputs. Figure 48 shows a clean and an adversarial image using C&W L-2 method [66]. The source codes of <https://github.com/rwightman/pytorch-nips2017-attack-example> are used to generate the adversarial image. As shown, the adversarial image and clean image looks nearly the same for human eyes.

For the Left vs Right dataset, adversarial are only successfully generated on right images. The reason why the adversarial cannot be generated for the Left images is out of the scope of this study, however the possibility of failure was mentioned in the original method [66]. It is valuable to observe that for binary classification of this

dataset, it is relatively hard to fool the original classifier. The final adversarial images are with noticeable differences from the input images, with an average l_2 distance of 12.3.

Figure 48. Original image (left) and adversarial image with C&W L-2 method (right)

C&W attack on the original classification network is very effective. The error rate reaches 100% with an average l_2 distance of 21.4 within 1000 iterations for all the test images for adversarial Right images.

Using the proposed reconstruction method, the fault caused by the adversarial attack can be detected by comparing the reconstructed image using the wrong reconstruction network and the input image. Figure 49 shows the results on one example. The leftmost image is the original image and the second image from left is the adversarial input. This adversarial image is mis-classified as “Left”, however, the reconstructed image by “Left” reconstruction network (third image from left) is less similar to the original image than that reconstructed by “Right” reconstruction network (the rightmost image).

Figure 49. Adversarial image and the reconstructed images

First of all, the SSIM measurement on all adversarial results are lower than that on the clean images for the Right decoder. This shows that the adversarial images have net effects on the extracted features and such effects can be detected by the reconstruction networks.

Within the 71 adversarial Right images, 15.5% of “Right reconstruction network” images have higher SSIM than “Left reconstruction network” images, i.e. the adversarial attack is detected. Compared to the reconstruction result of clean images, this rate is quite low. It shows that the C&W attack is really strong in shifting the feature vector from one class to the target class with minimum changes in the pixels. However, with a well-trained reconstruction network, a significant fraction of the adversarial images can be detected.

As a conclusion, by using a comparison of SSIM between these two decoders, the adversarial input can be detected with a detection rate of 15.5% (number of detected adversarial images divided by total number of input adversarial images).

Unseen Input

Figure 50 shows an example of the reconstructed images with the input of a Stop sign image. From left to right are the original image, image reconstructed with “Left” reconstruction network and image reconstructed with “Right” reconstruction network. It is clearly seen that neither of the reconstructed image resembles the original image. This fact can be used to determine that the decision of the recognition system is not supported by what is learned during training.

Figure 50. Reconstructed images with unseen input

With a total of 1110 test images of Stop sign in GTSRB, Table 7 shows the statistics of the SSIM values of the reconstruction results from the Left and Right decoders. The values are much lower than the SSIM of Left images reconstructed by “Left” reconstruction network or Right images reconstructed by “Right” reconstruction network. At the same time, the difference between the SSIM values of the two images reconstructed by two decoders are very small. This shows that it is possible to detect the unseen input by examining the SSIM values of two decoders.

SSIM		Unseen
to Left	mean	0.1131
	std dev	0.1111
to Right	mean	0.1095
	std dev	0.1066
to_Left v.s. to_Right	mean	0.00035
	std dev	0.1007

Table 7. Statistics of SSIM values for images reconstructed from two decoders

This difference operation is possible because it has clear physical meaning and the threshold is actually determined by the ability of the decoders for each class. For the general case of more than two classes, the vector formed by the SSIM value from all decoders needs to be analyzed to detect the “unseen” data.

Actually, if the difference between the SSIM by “Left” and “Right” reconstruction networks is calculated (by subtracting “Left reconstruction network” SSIM by “Right reconstruction network” SSIM), there is a clear statistic difference among the unseen (left), clean (center) and the adversarial (right) inputs as shown in Figure 51. The unseen images lead to a value in-between those of clean and adversarial images. This fact indicates that unseen data demonstrates different behavior from both clean and adversarial images of Right images. With a proper setting of the threshold, the unseen images can be detected (from clean and adversarial images) with a detection rate of 56.8%, much higher than the detection rate of adversarial images. Though not very high, this is a solid proof of the ability of detection.

Figure 51. The kernel density of difference between SSIM of two decoders

In summary, the proposed method has the capability to provide a good performance to various input, whether adversarial or unseen data, on the Left vs Right dataset.

4.3.2 Dataset of Dogs vs. Cats

Reconstruction

Figure 52 illustrates the images reconstructed with two different reconstruction networks using the same input image to the recognition system. In each row, the left one is the input image, the center one is the reconstructed image with “dog” reconstruction network and the right one is the reconstructed image with “cat” reconstruction network. It is clearly seen that with the correctly matched reconstruction network, the reconstructed image is much closer to the one reconstructed with the wrong reconstruction network.

Figure 52. Reconstructed images using the two reconstruction networks respectively

The statistics of this comparison on test data are shown in Table 8 and Table 9 respectively. Table 7 shows the mean SSIM values when comparing the reconstructed image using different decoders to the input images. For example, the value 0.4067 is the average of the SSIM values of all the 2,500 test cat images compared to the respective reconstructed image with the “cat decoder”. Table 8 shows the standard deviation of the SSIM values correspondingly. Note that both the example images and the SSIM measurements are done on test images instead of training images, in order to illustrate the real ability learned during the training process.

The reconstruction of different reconstruction networks is actually very effective. With 2,500 test cat images, only 0.64% of “cat reconstruction network” images have lower SSIM than “dog reconstruction network” images. With 2,500 test dog images, only 19.6 % of the “dog reconstruction network” images have lower SSIM than “cat reconstruction network” images. The possible reason for this difference could be that the “dog” reconstruction network is not well trained. Actually, the concept of “dog” is much more complex than that of “cat” as there are over 190 (or 340) different breeds of dogs while there are only about 40 (42 to be exact) breeds of cats according to <https://www.wonderopolis.org/>. However, the efforts and domain knowledge required to get enough training data is beyond this study.

		Input	
		Cat	Dog
reconstruction network	to cat	0.4067	0.3237
	to dog	0.3465	0.3323

Table 8. Mean SSIM values for images reconstructed from Dog and Cat decoders

		Input	
		Cat	Dog
reconstruction network	to cat	0.1027	0.1090
	to dog	0.0902	0.1081

Table 9. STD of SSIM values for images reconstructed from Dog and Cat decoders

Adversarial Input

For the Dogs vs Cats dataset, C&W L2 adversaries are successfully generated for both classes with almost unnoticeable differences. Figure 53 shows a clean and an adversarial image using C&W L-2 method [66]. As shown, the adversarial image and clean image looks nearly the same for human eyes.

Figure 53. Original image and adversarial image with C&W L-2 method

C&W attack on the original classification network is very effective. The error rate reaches 100% with an average l_2 distance of 1.4 within 1000 iterations for all the test images for both datasets.

Using the proposed reconstruction method, the fault caused by the adversarial attack can be detected by comparing the reconstructed image using the wrong reconstruction network and the input image. Figure 54 shows the results on two examples. The top input adversarial image is mis-classified as “dog”, however, the reconstructed image by “dog” reconstruction network is less similar to the original image than that reconstructed by “cat” reconstruction network.

Again, the SSIM measurement on all adversarial results are lower than that on the clean images, for both decoders. This shows that the adversarial images have net effects on the extracted features and such effects can be detected by the reconstruction networks.

Figure 54. Adversarial image and the reconstructed images

Within the 500 adversarial cat images, 32.4% of “cat reconstruction network” images have higher SSIM than “dog reconstruction network” images, i.e. the adversarial attack is detected. Compared to the reconstruction result of clean images, this rate is quite low. It shows that the C&W attack is really strong in shifting the feature vector from one class to the target class with minimum changes in the pixels. However, with a well-trained reconstruction network, a significant fraction of the adversarial images can be detected.

However, within the 500 adversarial dog images, only 3.2% of “dog reconstruction network” images have higher SSIM than “cat reconstruction network” images, i.e. the adversarial attack is detected. This ratio is even lower than the performance on adversarial cat images and the difference may again be caused by the under-training of “dog reconstruction network”.

As a conclusion, by using a comparison of SSIM between these two decoders, the adversarial input can be detected with a detection rate of 17.8% (number of detected adversarial images divided by total number of input adversarial images) with false alarm rate of 3.2% (number of clean images detected as adversarial images/number of total input clean images).

Unseen Input

Figure 55 shows an example of the reconstructed images with the input of a human face image. From left to right are the original image, image reconstructed with “cat” reconstruction network and image reconstructed with “dog” reconstruction network. It is clearly seen that neither of the reconstructed image resembles the original image. This fact can be used to determine that the decision of the recognition system is not supported by what is learned during training.

Figure 55. Reconstructed images with unseen input

With a total of 1000 test images of Celeb A, the proposed method records SSIM value of average value of 0.3986 with a standard deviation of 0.0726. This is comparable to the SSIM of cat images reconstructed by “cat” reconstruction network or dog images reconstructed by “dog” reconstruction network. One of the possible reasons is due to the poor reconstruction quality of the decoder.

The other reason is that the detection of “unseen” data is a more difficult problem than the simple “fault” detection. It requires the comparison of the reconstructed images from the decoders of all known classes.

Actually, if the difference between the SSIM by “cat” and “dog” reconstruction networks is calculated (by subtracting “cat reconstruction network” SSIM by “dog reconstruction network” SSIM), there is a clear statistic difference among the unseen, clean and the adversarial inputs as shown in Figure 56. The unseen images lead to a value in-between those of clean and adversarial images. This fact indicates that unseen data demonstrates different behavior from both clean and adversarial images of dog and cat. With a proper setting of the threshold, the unseen images can be detected (from clean images) with a detection rate of 64.4%, much higher than the detection rate of adversarial images. Though not very high, this is a solid proof of the ability of detection.

This difference operation is possible because it has clear physical meaning and the threshold is actually determined by the ability of the decoders for each class. For the general case of more than two classes, the vector formed by the SSIM value from all decoders needs to be analyzed to detect the “unseen” data.

Figure 56. The histogram of difference between SSIM of two decoders

In summary, the proposed method has the capability to provide a good performance to various input, whether adversarial or unseen data, on the dog-cat dataset.

4.4 State-of-the-Art Comparison

Table 10 summarizes the performance of proposed method and several state-of-the-art defense methods against the C&W attack [168]. The computation complexity is also included. It is difficult to do a fair comparison as the dataset and the applied attack strength are different. However, it is clear that the DBFR method is much more effective on a more realistic dataset with minimum computing complexity.

Defense Method	Detection Rate	Dataset	Attack Strength (L2)	Computing complexity
MagNet	1%	MNIST	2.25	Moderate
APE-GAN	0%	MNIST	2.17	High
DBFR (ours)	15.5%	GTSRB (Left/Right)	12.3	Low
DBFR (ours)	17.8%	Cat/Dog	1.4	Low

Table 10. Comparison of the performance of proposed method with SOTA methods

When encountering with the problem of unseen data, this method is the first one to work as an effective fault detection method with a detection rate of 56.8% for Left vs. Right dataset and 64.4% for Dogs vs. Cats dataset. Compared to the existing results that output a recognition result without knowing what is happening, the proposed method provides both an intuitive image and a quantified method to show that the decision is not supported by the training data.

4.5 Discussion on the Experiment Results

Linearity of the Deep Learning Networks

During the study, comparison analysis has been done on the logits (values before the softmax layer) of the original classification network with the input of clean and adversarial images. Figure 57 shows the distribution of the values of two logits with the input of clean and adversarial images (1000 each for dogs and cats) respectively. It clearly how the distribution of the logits becomes inseparable with the adversarial images.

Figure 57. Distribution of the logits for clean and adversarial images

What is more interesting is to compare the logits of the adversarial images with those of the corresponding clean images. Figure 58 clearly shows a linear relationship between them. The figure shows the distribution of 500 samples. It shows that logits of cats and dogs are separately clustered: for the first logits Z1: cat (red) vs dog (blue) and for the second logits Z2: cat (orange) vs dog (green). The R-Squared values for the four linear regression are 0.76, 0.81, 0.82 and 0.83 respectively and the values indicate strong linear relationship.

This supports the view that the vulnerability of deep learning network stems from its strong linearity. In the DBFR method, the reconstruction network and the SSIM comparison brings in non-linearity into the system and makes it more robust to the adversarial attacks.

Figure 58. Linear relationship for the logits of clean and adversarial images

Quality of the Reconstructed Images

During the training, with more epochs, the SSIM value will generally increase and the reconstructed images gradually turned from black-and-white to colorful and with more acutance. However, in general, the quality of the final reconstructed images by the DBFR method is quite low. This is generally caused by the loss of high frequency information in the compression process (similar effect is also observed for auto-encoders which are lossless) and lossy nature of the feature extractor of the classifier.

However, by a close examination of the data, it is revealed that the reconstruction quality is also related to the contents of the input images. The reconstructed image quality is relatively higher for those with clear body, simple background and high contrast. For those input images with only part of the body, complex background or low contrast, the reconstruction has relatively lower quality and SSIM scores. Figure 59 shows some results to illustrate this observation.

a) Original input images

b) Reconstructed images with corresponding decoders

Figure 59. Examples of reconstructed images with low qualities

The above observation on the quality of the reconstructed images is reasonable and proves that the reconstruction is based on “high-level” concept. The problem is there are also such images in the training dataset. It is hypothesized that with a stricter cleaning on the training images, the quality of the reconstructed images will be improved and so will the performance of the DBFR method.

The U-net structure [181] has been tested in the hope of generating images with better quality for the decoder. This type of network structure takes features of different levels from the feature extractor and concatenate with the features of corresponding levels in the decoder. In the experiment, ResNet50 network uses 4 layers for feature extraction and the DBFR method takes the output of layer4 as the “seed” to reconstruct the image. In the test, the output of layer3 is also provided into the corresponding layer in the decoder branch to provide “additional details”. In the test, decoders with U-net structure generate better reconstruction results as shown in Figure 60. In the figure, left is the original image, center is the reconstructed image using “dog” decoder with U-net structure and right is the reconstructed image using “cat” decoder with U-net structure.

Figure 60. Input image and reconstructed image using decoders with U-net structure

However, the decoders lose the ability to differentiate the reconstruction for images of different classes. That is, the reconstructed results of “dog” and “cat” decoders are no more differentiable both visually and quantitatively measured by SSIM. This is most likely caused by the skip connections inhibiting the decoders from learning new features for the inter-class difference. For this reason, the U-net structure is abandoned. More powerful generators, e.g. GAN, that are solely based on the final results of the feature extractor still needs to be explored to improve the quality of the reconstructed images.

4.6 Summary

Experimental results of the proposed DBFR framework are given in this chapter. They prove the effectiveness of the method on two example datasets for different applications and give indications of the direction of future works.

First of all, the reconstruction results of different decoders are significantly different. This shows that the inter-class difference are well learned using the DBFR framework. Although the reconstructed images are blurry, SSIM is still able to differentiate their different similarity to the original image with various reconstruction networks. As the experiment results show, the DBFR method is much more robust in defending adversarial attacks, especially for the strong ones such as C&W method. At the same time, it provides a novel detection mechanism for unseen data for the first time.

This result, in contrast to that of existing methods, reveals the problem of current deep learning, or in general, machine learning methods. Being data driven, the features these methods learn are tightly coupled with the specific task they are carrying out, without a real understanding of the physical meaning behind the forming of data. The supervised learning learns the mapping between the data and a specific label, which may generalize well for common scenes but cannot generalize to unusual or unseen scenarios. This makes the learning methods fragile to malicious attacks making use of this vulnerability.

But with knowledge of its own boundary, following the proposed feedback-based reconstruction method - DBFR, the application of deep learning models is more trustworthy as it is able to detect its own fault instead of blindly output garbage decisions.

CHAPTER 5 CONCLUSIONS AND DISCUSSIONS

The Detection-Based on-Feedback and Reconstruction (DBFR) method is proven to work as an effective fault detecting mechanism for image recognition systems using deep learning methods. The experiment results show superior robustness when compared to state-of-the-art methods. More importantly, this method provides both a new approach towards the final solution for adversarial examples and a novel representation of “internal model” for the deep learning system that is intuitive and useful.

5.1 Discussions

Possible Combination with Other Methods

The DBFR method works only as an augmentation on any existing classification networks, so it can be applied to all existing classification methods. Also, there is no conflict between the proposed framework and other existing defenses. Thus the proposed method can be combined with any existing method as a further enhancement.

Are the Results on Binary Classification Convincing?

Experiments are only carried out on the Left-Right and dog-cat dataset, which are both binary classification problems, for the ease of discussion. However, there is no inherent limitation in the method that prevent it being applied to multi-class classification problem. For example, the comparison of SSIM for two reconstruction networks can be extended to the quantitative analysis of the vector formed by SSIM values for all reconstruction networks for each class, using methods such as KNN (k nearest neighbor) or SVM (support vector machine). The effectiveness is based on clear physical meaning of the measurement SSIM on decoders for different classes (high-level concepts).

Reconstruction as a Representative of “Internal Model”

The combination of different decoders and the corresponding feature extractor are used as internal models for each class as learned by the image recognition system. With the same feature extraction and decoding process, different images can be reconstructed based on the choice of the combinations. This provides a feedback mechanism similar to what is observed in the recognition process of a human being. This feedback mechanism works effectively as a fault detection method to check whether the recognition decision agrees with what is learned in the training. This works even for unseen data, for which no existing method is reported to take care of.

The DBFR framework requires as many reconstruction networks as the number of classes for the image recognition task. It means the complexity scales linearly. This is a necessary cost to pay as the current deep learning is totally data driven and there lacks an abstract internal model to represent the high-level knowledge. The possible direction of reducing the number of reconstruction network is discussed in the section of Future Work.

The DBFR method is not a cure for the existing deep-learning based methods. Learning based methods have its inherent boundary drawn by the training dataset and the expressiveness linked with the network structure and training process. The generalization ability is limited and not fully understood. The DBFR method just provides an indicator of the boundary of a recognition system but it is not able to extend the boundary.

Quality of the Reconstruction

Although the detection ability of adversarial examples is improved with DBFR method, the success rate of adversarial attack is still significant. One possible reason is the difficulty in the training of decoders to learn mapping from a lossy feature space back to the image space formed by the training data of each class. This is indicated by the

quality of the reconstructed image with the original input. A lot of details are lost and the resolution is much lower.

One idea is to make use of a more powerful generator, for example GAN to fill-in the details that are lost when going through the feature extractor of the original classification network. This is similar to the “verifying” process of the human being when faced with difficult recognition problem. With a possible recognition proposal, the human being checks some details of the input to see whether it matches with the details that the proposal should possess.

5.2 Conclusions

This thesis introduced the Detection-Based on-Feedback and Reconstruction (DBFR) framework, which detects the adversarial inputs and unseen data by reconstruction at inference time to provide fault detection. The DBFR algorithm compares a reconstruction based on the classifying result with the original input to decide whether the classifying result agrees with what is learned in the training process. When the reconstructed and the input images are very similar, the prediction is likely to be accurate. If the reconstructed and input images are not in agreement, then the prediction does not have the training data support to be credible. This is the case with inputs that are maliciously perturbed by an adversary to produce an adversarial example. With analysis of the reconstructed images from decoders of all classes, unseen data not covered in the training dataset is also reliably detected. The reconstruction also enables interpretability of model predictions because they are points in the input domain that serve as support for the prediction and are easily understood and interpreted by human observers.

As the experiment results show, we conclude:

- The DBFR framework enables an image recognition system to detect its own fault, based on high-level feedback and reconstruction process. It works for both adversarial or unseen inputs;

- The DBFR method provides an effective detection mechanism for unseen data based on a reconstruction process;
- The DBFR method provides much more robust recognition results with adversarial attacks of various kinds and in theory can work for any attack methods;
- The reconstructed images with the DBFR method work as an intuitive representation of the internal model of the image recognition system and helps to interpret the deep learning models.

5.3 Future Works

An immediate work to improve the current results is to make use of more powerful class specific generators to reconstruct high quality images based on the lossy features extracted. These generators provide class-dependent details in order to generate realistic high-resolution images from the result of the feature extractor of the original classifier. It is expected that the detection of both adversarial examples and unseen data will be improved with reconstruction of better quality.

As discussed in Chapter 4 for the detection of “unseen” data, the DBFR framework can lead to an image recognition system that learns to expand its own recognition ability. Whenever a fault is detected and a higher level intervention is brought in, it will lead to two possible actions: to include the incorrectly labelled image into the training dataset, or to add in a new class of the “unseen” data that belongs to a concept different from its training data. With this kind of iteration, the boundary of the recognition ability of the system keeps expanding while the boundary remains monitored following the DBFR framework.

It is also straight forward to feedback the reconstructed image to the image recognition system again to form a closed-loop system. Even though the reconstructed images are blurry, they do represent structural information, either agrees or disagrees with that of the input image. Thus, it may be useful to feed this information (with/without future

processing) back into the original image recognition system for better training of the full image recognition system. If high quality images can be obtained by more powerful generator such as GAN as discussed above, the feedback is more straightforward. With a proper design of the loss function, the image recognition system is not only reinforced to separate the images of different classes successfully, but also to make sure the classification agrees with the reconstruction, which is based on the internal model of the concepts of each class.

Figure 61 illustrates the structure of the closed-loop image recognition network that checks the recognition result with its own internal model. Compared to the DBFR framework, the differences are: 1) generators, instead of decoders, are used to provide reconstructed images with high quality; 2) the reconstructed image is directly fed back into the classification network; 3) the closed loop will be applied for both training and inference stages. This closed-loop structure is similar to the working mechanism of the recognition process of human beings, with feedback in the loop.

How to design more powerful generators, how to properly feedback the reconstructed image back to the system and how to train the closed loop network effectively are still open questions.

Figure 61. Architecture of the closed loop image recognition network

Besides examining the reconstructed images and recognition results, it is also interesting to see whether models, e.g. abstract concepts such as body parts, can be learned and used for recognition in the above structure, as human beings do. This would

require both techniques to examine the high-level features learned by the model, as well as design of new network structure that enables the emergence of such concepts in the training.

A further extension of this work is to fit into a broader framework of image understanding based on context awareness [189]. A good example is the scenario of system initialization for autonomous driving. Even before the autonomous vehicle drives out of the carpark, it has to understand where it is and what kind of setting it is in before any recognition. The images taken in a carpark are totally different from the images to be expected on the road. Without knowing the context, the image recognition system tends to make mistakes because the image recognition system is an ill-posed problem as the images are 2D projection of the 3D world objects, whose solution is not unique in general. With the context awareness and the detection of objects of interest, the image recognition system with fault detection provides reliable results for the image understanding task.

REFERENCES

- [1] Hinton, G. E., Osindero, S. and Teh, Y. A fast learning algorithm for deep belief nets. *Neural Computation* 2006
- [2] Bengio, Y., Lamblin, P., Popovici, P., Larochelle, H. Greedy Layer-Wise Training of Deep Networks. *NIPS* 2006
- [3] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever and et. Al. Intriguing properties of neural networks. In *Proceedings of ICLR 2014, 2nd International Conference on Learning Representations*. Banff, AB, Canada.
- [4] Tencent Keen Security Lab. Experimental Security Research of Tesla Autopilot. <https://keenlab.tencent.com/en/2019/03/29/Tencent-Keen-Security-Lab-Experimental-Security-Research-of-Tesla-Autopilot/>
- [5] Finlayson, Samuel G., Isaac S. Kohane, and Andrew L. Beam. "Adversarial Attacks Against Medical Deep Learning Systems." *arXiv preprint arXiv:1804.05296*(2018).
- [6] F. Elsayed, Gamaleldin & Shankar, Shreya & Cheung, Brian & Papernot, Nicolas & Kurakin, Alex & Goodfellow, Ian & Sohl-Dickstein, Jascha. Adversarial Examples that Fool both Human and Computer Vision. *arXiv preprint arXiv:1802.08195*. 2018
- [7] Sébastien Bubeck, Eric Price, and Ilya Razenshteyn. Adversarial examples from computational constraints. *arXiv preprint arXiv:1805.10204*, 2018.
- [8] Sébastien Bubeck, Yin Tat Lee, Eric Price, and Ilya Razenshteyn. Adversarial examples from cryptographic pseudo-random generators. *arXiv preprint arXiv:1811.06418*, 2018.
- [9] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Małdry. Adversarially robust generalization requires more data. In *Advances in Neural Information Processing Systems* 31, pages 5019–5031, 2018.

- [10] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In International Conference on Learning Representations, 2019.
- [11] Zhang, Hongyang & Yu, Yaodong & Jiao, Jiantao & P. Xing, Eric & El Ghaoui, Laurent & I. Jordan, Michael. Theoretically Principled Trade-off between Robustness and Accuracy. arXiv preprint arXiv:1901.08573. 2019.
- [12] Luo, Yan, et al. Foveation-based mechanisms alleviate adversarial examples. arXiv preprint arXiv:1511.06292. 2019.
- [13] M. Barreno, B. Nelson, A. D. Joseph, and J. Tygar, “The security of machine learning,” *Machine Learning*, vol. 81, no. 2, pp. 121–148, 2010.
- [14] N. Dalvi, P. Domingos, S. Sanghai, and D. Verma, “Adversarial classification,” in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004, pp. 99–108.
- [15] D. Lowd and C. Meek, “Adversarial learning,” in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 2005, pp. 641–647.
- [16] B. Biggio, G. Fumera, and F. Roli, “Multiple classifier systems for robust classifier design in adversarial environments,” *International Journal of Machine Learning and Cybernetics*, vol. 1, no. 1-4, pp. 27–41, 2010.
- [17] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrđić, P. Laskov, G. Giacinto, and F. Roli, “Evasion attacks against machine learning at test time,” in *Joint*

- European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, 2013, pp. 387–402.
- [18] F. Roli, B. Biggio, and G. Fumera, “Pattern recognition systems under attack,” in Iberoamerican Congress on Pattern Recognition. Springer, 2013, pp. 1–8.
- [19] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, “Can machine learning be secure?” in Proceedings of the 2006 ACM Symposium on Information, computer and communications security. ACM, 2006, pp. 16–25.
- [20] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, “Sok: Towards the science of security and privacy in machine learning,” arXiv preprint arXiv:1611.03814, 2016.
- [21] Xie, C.; Wang, J.; Zhang, Z.; Zhou, Y.; Xie, L.; Yuille, A. Adversarial examples for semantic segmentation and object detection. arXiv 2017, arXiv:1703.08603.
- [22] Zhang, J.; Jiang, X. Adversarial Examples: Opportunities and Challenges. arXiv 2018, arXiv:1809.04790.
- [23] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, “Practical Black-Box Attacks against Machine Learning,” in Proceedings of ACM Asia Conference on Computer and Communications Security, 2017, pp. 506–519.
- [24] Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and harnessing adversarial examples. arXiv 2014, arXiv:1412.6572.
- [25] T. Tanay, L. Griffin, A Boundary Tilting Perspective on the Phenomenon of Adversarial Examples, arXiv preprint arXiv:1608.07690, 2016.

- [26] Y. Luo, Xavier Boix, Gemma Roig, Tomaso Poggio, and Qi Zhao, Foveation-based mechanisms alleviate adversarial examples, arXiv preprint arXiv:1511.06292, 2015.
- [27] K. Pei, Y. Cao, J. Yang, and S. Jana, “Deepxplore: Automated whitebox testing of deep learning systems,” Proceedings of the ACM SIGOPS 26th symposium on Operating systems principles, 2017.
- [28] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, “Pixeldefend: Leveraging generative models to understand and defend against adversarial examples,” arXiv preprint arXiv:1710.10766, 2017.
- [29] Q. Zheng, M. Yang, J. Yang, Q. Zhang, and X. Zhang, “Improvement of Generalization Ability of Deep CNN via Implicit Regularization in Two-Stage Training Process,” IEEE Access, vol. 6, pp. 15844-15869, 2018.
- [30] G. Philipp and J. G. Carbonell, “The Nonlinearity Coefficient – Predicting Overfitting in Deep Neural Networks,” 2018, arXiv:1806.00179
- [31] Y. Dong, H. Su, J. Zhu, and F. Bao, “Towards interpretable deep neural networks by leveraging adversarial examples,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
- [32] A. Fawzi, H. Fawzi, and O. Fawzi, “Adversarial vulnerability for any classifier,” arXiv preprint arXiv:1802.08686, 2018.
- [33] J. Gilmer, L. Metz, F. Faghri, S. S. Schoenholz, M. Raghu, M. Wattenberg, and I. Goodfellow, “Adversarial spheres,” arXiv preprint arXiv:1801.02774, 2018.

- [34] A. Fawzi, S. Moosavi-Dezfooli, P. Frossard, Robustness of classifiers: from adversarial to random noise, In *Advances in Neural Information Processing Systems*, pp. 1632-1640, 2016.
- [35] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, P. Frossard, S. Soatto, Analysis of universal adversarial perturbations, arXiv preprint arXiv:1705.09554, 2017
- [36] A. Fawzi, O. Fawzi, P. Frossard, Analysis of classifiers' robustness to adversarial perturbations, arXiv preprint arXiv:1502.02590, 2015.
- [37] Ali Shafahi, W. Ronny Huang, Christoph Studer, Soheil Feizi, Tom Goldstein. Are adversarial examples inevitable? arXiv preprint arXiv: 1809.02104, 2019.
- [38] Yinpeng Dong, Hang Su, Jun Zhu, Fan Bao. Towards Interpretable Deep Neural Networks by Leveraging Adversarial Examples. arXiv preprint arXiv: 1708.05493, 2017.
- [39] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, Wieland Brendel. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. Accepted at ICLR 2019 (oral).
- [40] Papernot, N.; McDaniel, P.; Wu, X.; Jha, S.; Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In *Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, USA, 22–26 May 2016; pp. 582–597.

- [41] M., S. Bhagavatula, L. Bauer, and M. K. Reiter, “Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition,” in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016, pp. 1528–1540.
- [42] N. Papernot, P. Mcdaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, “The limitations of deep learning in adversarial settings,” in Proceedings of IEEE European Symposium on Security and Privacy (EuroS&P), 2016, pp. 372-387.
- [43] Xu, W.; Evans, D.; Qi, Y. Feature squeezing: Detecting adversarial examples in deep neural networks. arXiv 2017, arXiv:1704.0115.
- [44] Gu, S.; Rigazio, L. Towards deep neural network architectures robust to adversarial examples. arXiv 2014, arXiv:1412.5068.
- [45] Gao, J.; Wang, B.; Lin, Z.; Xu, W.; Qi, Y. Deepcloak: Masking deep neural network models for robustness against adversarial samples. arXiv 2017, arXiv:1702.06763.
- [46] Cisse, M.; Adi, Y.; Neverova, N.; Keshet, J. Houdini: Fooling deep structured prediction models. arXiv 2017, arXiv:1707.05373.
- [47] O. M. Parkhi, A. Vedaldi, A. Zisserman et al., “Deep face recognition,” in BMVC, vol. 1, no. 3, 2015, p. 6.
- [48] R. Jia and P. Liang, “Adversarial examples for evaluating reading comprehension systems,” in Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2017.

- [49] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, “Squad: 100,000+ questions for machine comprehension of text,” arXiv preprint arXiv:1606.05250, 2016.
- [50] J. Li, W. Monroe, and D. Jurafsky, “Understanding neural networks through representation erasure,” arXiv preprint arXiv:1612.08220, 2016.
- [51] Z. Zhao, D. Dua, and S. Singh, “Generating natural adversarial examples,” arXiv preprint arXiv:1710.11342, 2017.
- [52] G. E. Dahl, J. W. Stokes, L. Deng, and D. Yu, “Large-scale malware classification using random projections and neural networks,” in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3422–3426.
- [53] Z. Yuan, Y. Lu, Z. Wang, and Y. Xue, “Droid-sec: deep learning in android malware detection,” in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 371–372.
- [54] J. Saxe and K. Berlin, “Deep neural network based malware detection using two dimensional binary program features,” in *Malicious and Unwanted Software (MALWARE), 2015 10th International Conference on*. IEEE, 2015, pp. 11–20.
- [55] R. Sun, X. Yuan, P. He, Q. Zhu, A. Chen, A. Gregio, D. Oliveira, and L. Xiaolin, “Learning fast and slow: Propedeutica for real-time malware detection,” arXiv preprint arXiv:1712.01145, 2017.
- [56] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, “Adversarial examples for malware detection,” in *European Symposium on Research in Computer Security*. Springer, 2017, pp. 62–79.

- [57] H. S. Anderson, A. Kharkar, B. Filar, and P. Roth, “Evading machine learning malware detection,” Black Hat, 2017.
- [58] W. Hu and Y. Tan, “Generating adversarial malware examples for black-box attacks based on gan,” arXiv preprint arXiv:1702.05983, 2017.
- [59] W. Xu, Y. Qi, and D. Evans, “Automatically evading classifiers,” in Network and Distributed System Security Symposium (NDSS), 2016.
- [60] H. S. Anderson, J. Woodbridge, and B. Filar, “Deepdga: Adversarially tuned domain generation and detection,” in Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security (AISec). ACM, 2016, pp. 13–21.
- [61] A. Kurakin and I. Goodfellow, “Adversarial examples in the physical world,” in Proceedings of International Conference on Learning Representations (ICLR), 2016, pp. 1-14.
- [62] Kurakin, A.; Goodfellow, I.; Bengio, S. Adversarial machine learning at scale. arXiv 2016, arXiv:1611.01236.
- [63] Dong, Y.; Liao, F.; Pang, T.; Su, H.; Hu, X.; Li, J.; Zhu, J. Boosting adversarial attacks with momentum. arXiv 2017, arXiv:1710.06081.
- [64] Miyato, T.; Maeda, S.i.; Ishii, S.; Koyama, M. Virtual adversarial training: A regularization method for supervised and semi-supervised learning. IEEE Trans. Pattern Anal. Mach. Intell. 2018.
- [65] Su, J.; Vargas, D.V.; Kouichi, S. One pixel attack for fooling deep neural networks. arXiv 2017, arXiv:1710.08864.

- [66] N. Carlini and D. Wagner, "Towards Evaluating the Robustness of Neural Networks," in Proceedings of IEEE Symposium on Security and Privacy (EuroS&P), 2017, pp. 39-57.
- [67] S.M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: a simple and accurate method to fool deep neural networks," in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2574-2582.
- [68] S. M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 86-94.
- [69] S. Sarkar, A. Bansal, U. Mahbub, and R. Chellappa, UPSET and ANGRI: Breaking High Performance Image Classifiers, arXiv preprint arXiv:1707.01159, 2017.
- [70] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel, Backpropagation applied to handwritten zip code recognition. Neural computation, vol. 1m no. 4, pp. 541-551, 1989.
- [71] A. Krizhevsky, Learning multiple layers of features from tiny image, 2009.
- [72] Samangouei, P.; Kabkab, M.; Chellappa, R. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. arXiv 2018, arXiv:1805.06605.
- [73] S. Baluja, I. Fischer, Adversarial Transformation Networks: Learning to Generate Adversarial Examples, arXiv preprint arXiv:1703.09387, 2017.

- [74] J. Hayes, G. Danezis, Machine Learning as an Adversarial Service: Learning Black-Box Adversarial Examples, arXiv preprint arXiv:1708.05207, 2017.
- [75] Y. Liu, X. Chen, C. Liu, and D. Song, “Delving into Transferable Adversarial Examples and Black-box Attacks,” 2016, arXiv:1611.02770.
- [76] Eykholt, K.; Evtimov I.; Fernandes E.; Li B.; Rahmati A.; Xiao C.; Prakash A.; Kohno T.; Song D. Robust physical-world attacks on deep learning visual classification. In the IEEE Conference on Computer Vision and Pattern Recognition (CVPR); IEEE: Piscataway, NJ, USA, 2018, pp. 1625–1634.
- [77] Mogelmose A., Trivedi M., Moeslund T. Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey. *Trans. Intell. Transport. Syst.* 2012, 3, 1484–1497.
- [78] Stallkamp J., Schlipsing M., Salmen J., Igel C. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Netw.* 2012, 32, 323–332.
- [79] Gu, T.; Dolan-Gavitt, B.; Garg, S. Badnets: Identifying vulnerabilities in the machine learning model supply chain. arXiv 2017, arXiv:1708.06733.
- [80] Melis, M.; Demontis, A.; Biggio, B.; Brown, G.; Fumera, G.; Roli, F. Is deep learning safe for robot vision? adversarial examples against the icub humanoid. In *Proceedings of the 2017 IEEE International Conference on Computer Vision Workshop (ICCVW)*, Venice, Italy, 22–29 October 2017; pp. 751–759.
- [81] Xu, X.; Chen, X.; Liu, C.; Rohrbach, A.; Darell, T.; Song, D. Can you fool AI with adversarial examples on a visual Turing test? arXiv 2017, arXiv:1709.08693.

- [82] Zhou, Z.; Tang, D.; Wang, X.; Han, W.; Liu, X.; Zhang, K. Invisible Mask: Practical Attacks on Face Recognition with Infrared. arXiv 2018, arXiv:1803.04683.
- [83] Yann, L.; Corinna, C.; Christopher, J. MNIST. Available online: <http://yann.lecun.com/exdb/mnist/>
- [84] Huang, R.; Xu, B.; Schuurmans, D.; Szepesvári, C. Learning with a strong adversary. arXiv 2015, arXiv:1511.03034.
- [85] Tramèr, F.; Kurakin, A.; Papernot, N.; Goodfellow, I.; Boneh, D.; McDaniel, P. Ensemble adversarial training: Attacks and defenses. arXiv 2017, arXiv:1705.07204.
- [86] Hosseini, H.; Chen, Y.; Kannan, S.; Zhang, B.; Poovendran, R. Blocking transferability of adversarial examples in black-box learning systems. arXiv 2017, arXiv:1703.04318.
- [87] Dziugaite, G.K.; Ghahramani, Z.; Roy, D.M. A study of the effect of jpg compression on adversarial images. arXiv 2016, arXiv:1608.00853.
- [88] Das, N.; Shanbhogue, M.; Chen, S.T.; Hohman, F.; Chen, L.; Kounavis, M.E.; Chau, D.H. Keeping the bad guys out: Protecting and vaccinating deep learning with jpeg compression. arXiv 2017, arXiv:1705.02900.
- [89] Wang, Q.; Guo, W.; Zhang, K.; Ororbia, I.; Alexander, G.; Xing, X.; Liu, X.; Giles, C.L. Learning adversary-resistant deep neural networks. arXiv 2016, arXiv:1612.01401.

- [90] Biggio, B.; Nelson, B.; Laskov, P. Support vector machines under adversarial label noise. In Proceedings of the Asian Conference on Machine Learning, Taoyuan, Taiwan, 13–15 November 2011; pp. 97–112.
- [91] Lyu, C.; Huang, K.; Liang, H.N. A unified gradient regularization family for adversarial examples. In Proceedings of the 2015 IEEE International Conference on Data Mining (ICDM), Atlantic City, NJ, USA, 14–17 November 2015; pp. 301–309.
- [92] Zhao, Q.; Griffin, L.D. Suppressing the unusual: Towards robust cnns using symmetric activation functions. arXiv 2016, arXiv:1603.05145.
- [93] Rozsa, A.; Gunther, M.; Boulton, T.E. Towards robust deep neural networks with BANG. arXiv 2016, arXiv:1612.00138.
- [94] Papernot, N.; McDaniel, P. Extending defensive distillation. arXiv 2017, arXiv:1705.05264.
- [95] Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In Advances in Neural Information Processing Systems, Proceedings of the Annual Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013; Curran Associates, Inc.: Red Hook, NY, USA, 2014; pp. 2672–2680.
- [96] Meng, D.; Chen, H. Magnet: A two-pronged defense against adversarial examples. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 135–147.

- [97] Liao, F.; Liang, M.; Dong, Y.; Pang, T.; Zhu, J.; Hu, X. Defense against adversarial attacks using high-level representation guided denoiser. arXiv 2017, arXiv:1712.02976.
- [98] A. Nayebi, and S. Ganguli, Biologically inspired protection of deep networks from adversarial attacks, arXiv preprint arXiv:1703.09202, 2017.
- [99] D. Krotov, and J. J. Hopfield. Dense Associative Memory is Robust to Adversarial Inputs, arXiv preprint arXiv:1701.00939, 2017.
- [100] G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer, “Reluplex: An efficient smt solver for verifying deep neural networks,” arXiv preprint arXiv:1702.01135, 2017.
- [101] D. Gopinath, G. Katz, C. S. Pasareanu, and C. Barrett, “Deepsafe: A data-driven approach for checking adversarial robustness in neural networks,” arXiv preprint arXiv:1710.00486, 2017.
- [102] T. Miyato, A. M. Dai, and Ian Goodfellow, Adversarial Training Methods for Semi-Supervised Text Classification, arXiv preprint arXiv:1605.07725, 2016.
- [103] S. Zheng, Y. Song, T. Leung, and I. Goodfellow, Improving the robustness of deep neural networks via stability training, In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4480-4488, 2016.
- [104] Akhtar, N.; Liu, J.; Mian, A. Defense against Universal Adversarial Perturbations. arXiv 2017, arXiv:1711.05929.

- [105] C. Guo, M. Rana, M. Cisse, L. Maaten, Countering Adversarial Images using Input Transformations, arXiv preprint arXiv:1711.00117, 2017.
- [106] R. Shin and D. Song, JPEG-resistant adversarial images, In Machine Learning and Computer Security Workshop, 2017.
- [107] A. N. Bhagoji, D. Cullina, C. Sitawarin, P. Mittal, Enhancing Robustness of Machine Learning Systems via Data Transformations, arXiv preprint arXiv:1704.02654, 2017.
- [108] W. Xu, D. Evans, and Y. Qi, Feature Squeezing Mitigates and Detects Carlini/Wagner Adversarial Examples, arXiv preprint arXiv:1705.10686, 2017.
- [109] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, Imagenet: A large-scale hierarchical image database. In IEEE Conference on Computer Vision and Pattern Recognition, pp. 248-255, 2009.
- [110] V. Zantedeschi, M. Nicolae, A. Rawat, Efficient Defenses Against Adversarial Attacks, arXiv preprint arXiv:1707.06728, 2017.
- [111] A. S. Ross, F. Doshi-Velez, Improving the Adversarial Robustness and Interpretability of Deep Neural Networks by Regularizing their Input Gradients, arXiv preprint arXiv:1711.09404, 2017.
- [112] H. Drucker, Y. Le Cun, Improving generalization performance using double backpropagation, IEEE Transactions on Neural Networks vol. 3, no. 6, pp. 991-997, 1992.

- [113] U. Shaham, Y. Yamada, S. Negahban, Understanding Adversarial Training: Increasing Local Stability of Neural Nets through Robust Optimization, arXiv preprint arXiv:1511.05432, 2016.

- [114] L. Nguyen, A. Sinha, A Learning and Masking Approach to Secure Learning, arXiv preprint arXiv:1709.04447, 2017.

- [115] Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. arXiv 2015, arXiv:1503.02531.

- [116] Y. Bengio, Learning deep architectures for AI, Foundations and trends in Machine Learning vol. 2, no. 1, pp. 1-127, 2009.

- [117] Rifai, S.; Vincent, P.; Muller, X.; Glorot, X.; Bengio, Y. Contractive auto-encoders: Explicit invariance during feature extraction. In Proceedings of the 28th International Conference on International Conference on Machine Learning, Bellevue, WA, USA, 28 June–2 July 2011; pp. 833–840.

- [118] W. Bai, C. Quan, and Z. Luo, Alleviating adversarial attacks via convolutional autoencoder, In International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), pp. 53-58, 2017.

- [119] Kovacevic, J.; Chebira, A.; et al. An introduction to frames. Found. Trends Signal Process. 2008, 2, 1–94.

- [120] W. Brendel and M. Bethge. Comment on “Biologically inspired protection of deep networks from adversarial attacks”, arXiv preprint arXiv:1704.01547, 2017.

- [121] S. S. Liew, M. Khalil-Hani, and R. Bakhteri, Bounded activation functions for enhanced training stability of deep neural networks on visual pattern recognition problems, *Neurocomputing*, vol. 216, pp. 718-734, 2016.
- [122] J. Jin, A. Dundar, and E. Culurciello, Robust convolutional neural networks under adversarial noise, *arXiv preprint arXiv:1511.06306*, 2015.
- [123] Z. Sun, M. Ozay, T. Okatani, HyperNetworks with statistical filtering for defending adversarial examples, *arXiv preprint arXiv:1711.01791*, 2017.
- [124] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards Deep Learning Models Resistant to Adversarial Attacks, *arXiv preprint arXiv:1706.06083*, 2017.
- [125] N. Carlini, G. Katz, C. Barrett, D. L. Dill, Ground-Truth Adversarial Examples, *arXiv preprint arXiv:1709.10207*, 2017.
- [126] T. Na, J. H. Ko, S. Mukhopadhyay, Cascade Adversarial Machine Learning Regularized with a Unified Embedding, *arXiv preprint arXiv:1708.02582*, 2017.
- [127] T. Strauss, M. Hanselmann, A. Junginger, H. Ulmer, Ensemble Methods as a Defense to Adversarial Perturbations Against Deep Neural Networks, *arXiv preprint arXiv:1709.03423*, 2017.
- [128] N. Kardan, and K. O. Stanley, Mitigating fooling with competitive overcomplete output layer neural networks, In *International Joint Conference on Neural Networks* pp. 518-525, 2017.

- [129] Q. Wang, W. Guo, I. I. Ororbia, G. Alexander, X. Xing, L. Lin, C. L. Giles, X. Liu, P. Liu, and G. Xiong, Using non-invertible data transformations to build adversary-resistant deep neural networks, arXiv preprint arXiv:1610.01934, 2016.
- [130] Q. Wang, W. Guo, K. Zhang, I. I. Ororbia, G. Alexander, X. Xing, C. L. Giles, and X. Liu, Adversary Resistant Deep Neural Networks with an Application to Malware Detection, arXiv preprint arXiv:1610.01239, 2016.
- [131] T. Lee, M. Choi, S. Yoon, Manifold Regularized Deep Neural Networks using Adversarial Examples, arXiv preprint arXiv:1511.06381, 2016.
- [132] J. Z. Kolter, E. Wong, Provable defenses against adversarial examples via the convex outer adversarial polytope, arXiv preprint arXiv:1711.00851, 2017.
- [133] A. Raghunathan, J. Steinhardt, P. Liang, Certified Defenses against Adversarial Examples, arXiv preprint arXiv:1801.09344. 2018.
- [134] X. Huang, M. Kwiatkowska, S. Wang, M. Wu, Safety Verification of Deep Neural Networks, In 29th International Conference on Computer Aided Verification, pages 3-29, 2017.
- [135] M. Wicker, X. Huang, and M. Kwiatkowska, Feature-Guided Black-Box Safety Testing of Deep Neural Networks, In 24th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, 2018.
- [136] A. G. Ororbia II, C. L. Giles and D. Kifer, Unifying Adversarial Training Algorithms with Flexible Deep Data Gradient Regularization, arXiv preprint arXiv:1601.07213, 2016.

- [137] J. Lu, T. Issaranon, D. Forsyth, SafetyNet: Detecting and Rejecting Adversarial Examples Robustly, arXiv preprint arXiv:1704.00103, 2017.
- [138] J. H. Metzen, T. Genewein, V. Fischer, B. Bischoff, On Detecting Adversarial Perturbations, arXiv preprint arXiv:1702.04267, 2017.
- [139] X. Li, F. Li, Adversarial Examples Detection in Deep Networks with Convolutional Filter Statistics, In Proceedings of International Conference on Computer Vision, 2017.
- [140] A. Nguyen, J. Yosinski, and J. Clune, Deep neural networks are easily fooled: High confidence predictions for unrecognizable images, In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 427-436, 2015.
- [141] K. Grosse, P. Manoharan, N. Papernot, M. Backes, P. McDaniel, On the (Statistical) Detection of Adversarial Examples, arXiv preprint arXiv:1702.06280, 2017.
- [142] Z. Gong, W. Wang, and W.s. Ku, “Adversarial and Clean Data Are Not Twins,” 2017, arXiv:1704.04960.
- [143] D. Hendrycks and K. Gimpel, “Early Methods for Detecting Adversarial Images,” in Proceedings of International Conference on Learning Representations (ICLR), 2016, pp. 1-9.
- [144] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, “Detecting Adversarial Samples from Artifacts,” 2017, arXiv:1703.00410.

- [145] I. Sutskever, G. Hinton, A. Krizhevsky, and R. R. Salakhutdinov, "Dropout : A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929-1958, 2014.
- [146] W. He, J. Wei, X. Chen, N. Carlini, D. Song, Adversarial Example Defenses: Ensembles of Weak Defenses are not Strong, *arXiv preprint arXiv:1706.04701*, 2017.
- [147] M. Abbasi, C. Gagne, Robustness to adversarial examples through an ensemble of specialists, *arXiv preprint arXiv:1702.06856*, 2017.
- [148] B. Liang, H. Li, M. Su, X. Li, W. Shi, X. Wang, Detecting Adversarial Examples in Deep Networks with Adaptive Noise Reduction, *arXiv preprint arXiv:1705.08378*, 2017.
- [149] T. Gebhart, P. Schrater, Adversary Detection in Neural Networks via Persistent Homology, *arXiv preprint arXiv:1711.10056*, 2017.
- [150] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," *AISEC*, 2017.
- [151] N. Carlini and D. Wagner. "Magnet and efficient defenses against adversarial attacks are not robust to adversarial examples," *arXiv preprint arXiv:1711.08478*, 2017.
- [152] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Towards proving the adversarial robustness of deep neural networks," *arXiv preprint arXiv:1709.02802*, 2017.

- [153] Hongge Chen, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, and Cho-Jui Hsieh. 2017. Attacking Visual Language Grounding with Adversarial Examples: A Case Study on Neural Image Captioning. arXiv preprint arXiv:1712.02051 (2017).
- [154] Xiaojun Xu, Xinyun Chen, Chang Liu, Anna Rohrbach, Trevor Darell, and Dawn Song. 2017. Can you fool AI with adversarial examples on a visual Turing test? arXiv preprint arXiv:1709.08693 (2017).
- [155] Christian Cosgrove, Alan L. Yuille. Adversarial Examples for Edge Detection: They Exist, and They Transfer. arXiv preprint arXiv:1906.00335 (2019).
- [156] Andrew Slavin Ross, Finale Doshi-Velez. Improving the Adversarial Robustness and Interpretability of Deep Neural Networks by Regularizing their Input Gradients. arXiv preprint arXiv:1711.09404 (2017).
- [157] Anish Athalye, Nicholas Carlini, David Wagner. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. arXiv preprint arXiv:1802.00420 (2018).
- [158] Rishi Rajalingham, Elias B. Issa, Pouya Bashivan, Kohitij Kar, Kailyn Schmidt and James J. DiCarlo. Large-Scale, High-Resolution Comparison of the Core Visual Object Recognition Behavior of Humans, Monkeys, and State-of-the-Art Deep Artificial Neural Networks. *Journal of Neuroscience* 15 August 2018, 38 (33) 7255-7269.
- [159] Ji Lin, Chuang Gan, Song Han. Defensive Quantization: When Efficiency Meets Robustness. arXiv preprint arXiv:1904.08444 (2019).

- [160] Aamir Mustafa, Salman H. Khan, Munawar Hayat, Jianbing Shen, Ling Shao. Image Super-Resolution as a Defense Against Adversarial Attacks. arXiv preprint arXiv:1901.01677 (2019).
- [161] Fawzi Alhussein, Fawzi Omar, Frossard Pascal. Fundamental limits on adversarial robustness. Proceedings of ICML, Workshop on Deep Learning. 2015.
- re
- [162] Isaac Caswell, Chuanqi Shen, Lisa Wang. Tech. Report. Loopy neural nets: Imitating feedback loops in the human brain. 2016
- [163] Xin Li, Zequn Jie, Jiashi Feng, Changsong Liu, Shuicheng Yan. Learning with Rethinking: Recurrently Improving Convolutional Neural Networks through Feedback. arXiv preprint arXiv:1708.04483 (2017).
- [164] Xiaoyong Yuan, Pan He, Qile Zhu, Xiaolin Li. Adversarial Examples: Attacks and Defenses for Deep Learning. IEEE Transactions on Neural Networks and Learning Systems (Volume: 30 , Issue: 9, Sept. 2019).
- [165] Jiliang Zhang, Chen Li. Adversarial Examples: Opportunities and Challenges. arXiv preprint arXiv:1809.04790 (2018).
- [166] Qiu, Shilin & Liu, Qihe & Zhou, Shijie & Wu, Chunjiang. Review of Artificial Intelligence Adversarial Attack and Defense Technologies. Applied Sciences. 9. 909. 10.3390/app9050909 (2019).
- [167] Naveed Akhtar, Ajmal Mian. Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey. arXiv preprint arXiv:1801.00553 (2018).

- [168] Nicholas Carlini, David Wagner. MagNet and "Efficient Defenses Against Adversarial Attacks" are Not Robust to Adversarial Examples. arXiv preprint arXiv:1711.08478 (2017).
- [169] Xingjun Ma, Bo Li, Yisen Wang, Sarah M. Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E. Houle, James Bailey. Characterizing Adversarial Subspaces Using Local Intrinsic Dimensionality. arXiv preprint arXiv:1801.02613 (2018).
- [170] Thomas Robert, Nicolas Thome, Matthieu Cord. HybridNet: Classification and Reconstruction Cooperation for Semi-Supervised Learning. arXiv:1807.11407 (2018).
- [171] Wieland Brendel, Matthias Bethge. Approximating CNNs with Bag-of-local-Features models works surprisingly well on ImageNet. arXiv:1904.00760 (2019).
- [172] Aravindh Mahendran, Andrea Vedaldi. Understanding Deep Image Representations by Inverting Them. arXiv:1412.0035 (2014).
- [173] Abhimanyu Dubey, Laurens van der Maaten, Zeki Yalniz, Yixuan Li, Dhruv Mahajan. Defense Against Adversarial Images using Web-Scale Nearest-Neighbor Search. arXiv:1903.01612 (2019).
- [174] Nicolas Papernot, Patrick McDaniel. Deep k-Nearest Neighbors: Towards Confident, Interpretable and Robust Deep Learning. arXiv:1803.04765 (2018).
- [175] Shiwei Shen, Guoqing Jin, Ke Gao, Yongdong Zhang. APE-GAN: Adversarial Perturbation Elimination with GAN. arXiv:1707.05474 (2017).

- [176] Dana H. Ballard. Modular learning in neural networks. *AAAI'87 Proceedings of the sixth National conference on Artificial intelligence*. Volume 1, Pages 279-284.
- [177] N. Papernot, I. Goodfellow, R. Sheatsley, R. Feinman, and P. McDaniel. cleverhans v1.0.0: an adversarial machine learning library. arXiv preprint arXiv:1610.00768, 2016.
- [178] S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet, “Adversarial manipulation of deep representations,” International Conference on Learning Representations, 2016.
- [179] Y. Gal, Uncertainty in deep learning. Ph.D. dissertation, PhD thesis, University of Cambridge, 2016.
- [180] Vit LISTIK. Image reconstruction from ResNet semantic feature vector. POSTER 2018.
- [181] Olaf Ronneberger, Philipp Fischer, Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv preprint arXiv: 1505.04597, 2015.
- [182] Bian Y, Wang J, Jun JJ, Xie XQ. Deep Convolutional Generative Adversarial Network (dcGAN) Models for Screening and Design of Small Molecules Targeting Cannabinoid Receptors. *Molecular Pharmaceutics*. 2019 Nov;16(11):4451-4460.
- [183] Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybernetics* 36, 193–202 (1980).

- [184] J. Weng, N. Ahuja and T. S. Huang, Cresceptron: a self-organizing neural network which grows adaptively. IJCNN International Joint Conference on Neural Networks, Baltimore, MD, USA, 1992, pp. 576-581 vol.1, doi: 10.1109/IJCNN.1992.287150.
- [185] P. J. Werbos, Backpropagation through time: what it does and how to do it. in Proceedings of the IEEE, vol. 78, no. 10, pp. 1550-1560, Oct. 1990, doi: 10.1109/5.58337.
- [186] Andrej Karpathy. What I learned from competing against a ConvNet on ImageNet. <http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/>
- [187] Samuel Dodge and Lina Karam. A Study and Comparison of Human and Deep Learning Recognition Performance Under Visual Distortions. arXiv preprint arXiv: 1705.02498, 2017.
- [188] Juyang Weng. Natural and Artificial Intelligence: Introduction to Computational Brain-Mind. BMI Press, 2012.
- [189] Xu Jin, Xie Ming. Iterative Template Matching Strategy for Visual Target Detection by Unmanned Surface Vehicle. In: Liu H., Kubota N., Zhu X., Dillmann R. (eds) *Intelligent Robotics and Applications*. Lecture Notes in Computer Science, vol 9246. Springer, Cham. 2015.