深層学習における Adversarial Training による副作用とその緩和策

先崎　佑弥[1]　大畑　幸矢[2]　松浦　幹太[1]

概要：畳み込みニューラルネットワーク（Convolutional Neural Network, CNN）は画像認識や音声認識、自然言語処理などへ応用した際に高い精度を出すことがわかってきたため注目を集めている。しかし一方で、CNN への入力データに微小な改変を加えることで出力を大きく誤らせることが可能な敵対的入力の存在が報告されており、CNN を実社会で用いる際に大きな脅威となることが予想される。この問題に対して頑健な識別器を構成するテクニックとして、敵対的入力も学習用データに加えて学習する「敵対的訓練（Adversarial Training）」と呼ばれる手法が提案されており、敵対的入力に対する耐性を向上させることが確認されている。本稿ではこの敵対的訓練における問題点を指摘し、その対策法を提案する。具体的には、CNN に対して敵対的訓練を行うと（本来高い精度で識別できるはずの）ランダムノイズが乗ったデータに対する識別率が大きく減少してしまうことを指摘する。その問題を解決するためにランダムノイズを付加した画像も教師データに加えて学習する手法を提案し、計算機実験により提案手法の有用性を実証する。

キーワード：深層学習、畳み込みニューラルネットワーク、敵対的入力、敵対的訓練

Negative Side Effect of Adversarial Training in Deep Learning and Its Mitigation

Yuya Senzaki[1]　Satsuya Ohata[2]　Kanta Matsuura[1]

Abstract: Convolutional Neural Networks (CNN) attract high attention because of its accuracy in many tasks. However, the existence of “adversarial examples”, which are the maliciously generated data to cheat classifiers, will be a large thread in practice. A novel training technique so-called “adversarial training” is widely known as a countermeasure of this problem. In this technique, we include adversarial examples in training dataset and can make CNN robust against adversarial examples. In this paper, we point out the negative aspect of the adversarial training. More concretely, we found that adversarially-trainined CNN are not robust against randomly-noised data. We show experimental results of this fact and propose its mitigation.

Keywords: Deep Learning, Convolutional Neural Network, Adversarial Examples, Adversarial Training

1. はじめに

教師あり学習の一手法である深層学習（ディープラーニング）と、その発展形である畳み込みニューラルネットワーク（Convolutional Neural Network, CNN）は、画像認識を始めとする諸分野で圧倒的なパフォーマンスを示しており、大きな注目を集めている。CNN は学術界で注目されているだけでなく、囲碁のコンピュータソフト [28] がプロ棋士を上回る力を見せたり、国内でも半導体製造における生産性向上 [34] やロボット動作の高度化 [26] といった分野で既に産業応用されており、人間の意識や生活を変化させ始めていると言える。CNN が社会で広く利用されるようになると悪意を持つ者にとっては CNN 自体を攻撃

[1] 東京大学生産技術研究所 〒 153-8505 東京都目黒区駒場 4-6-1 {senzaki,kanta}@iis.u-tokyo.ac.jp
[2] 産業技術総合研究所 〒 135-0064 東京都江東区青海 2-4-7 satsuya.ohata@aist.go.jp

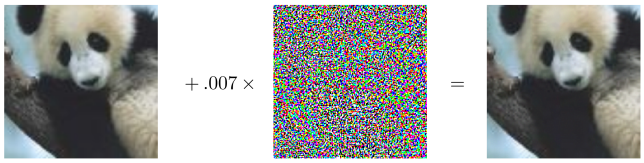

図 1　敵対的入力の一例 [12]

するインセンティブが高まることを意味するため、それ自体のセキュリティを研究する意義が大きくなる。セキュリティ研究において長らくの間、機械学習は異常検知等のためのツールであったが、（条件が揃えば）学習器それ自体に対して攻撃が可能であることが発見され、研究が活発に進められている。

本研究ではその中でも「敵対的入力（Adversarial Examples, A.E.）」と呼ばれる、正しいテストデータに対しては高い確率で正解ラベルを出力できるように訓練された学習器でも、高い確率で推定を誤ってしまう特殊な入力（テストデータ）に関して取り扱う [*1]。図 1 が A.E. の一例である。CNN は左の画像をパンダであると問題なく認識できるが、その画像に中央のノイズ画像をマスクして出来た右の画像は（人間にはほぼ変化がないように見えるが）非常に高い確率でテナガザルであると認識されてしまう [*2]。CNN を始めとする高精度な学習器は自動運転車や金融商品のアルゴリズムトレードといった分野のベースとなっていく（あるいは既になっている）ことが想定されるが、そのような分野において A.E. は人命や資産に対する脅威となりうる。

A.E. に対する対応策として知られているのが「敵対的訓練（Adversarial Training, A.T.）」と呼ばれる手法である。これは学習時から A.E. を生成し利用することで学習器を A.E. に「慣れさせ」て A.E. の悪影響を緩和するテクニックであり、学習器の A.E. 耐性を向上可能であることが計算機実験によって確認されている。しかし研究としての歴史が浅いこともあるが、この A.T. に関する評価はまだまだ不十分であるのが現状である。A.T. を取り扱った論文の多くでは、正しいテストデータと A.E. のみを用いて手法の精度を評価しているケースが多く、特に A.T. が持つ負の側面には目が向けられてこなかった。本稿ではこの A.T. が学習器に対して引き起こす副作用とその緩和策を取り扱う。

[*1] 学習器に対する攻撃としては Poisoning 攻撃 [3,4] もあるが、これは教師データに攻撃者が生成したデータを含めることで学習器の識別精度それ自体を下げる攻撃手法であり、A.E. とは異なる。

[*2] A.E. は NN 特有の性質（非線形性等）に起因するわけではなく、線形識別器に対しても存在することが知られている [12]。

1.1　関連研究

1.1.1　A.E. の生成

学習器が意図的に作成された入力に対して脆弱であることは [3,9,30] 等で示されていたが、Szegedy ら [32] が NN に対して実際にそのような入力（画像）、すなわち A.E. を生成できることを示し、この分野の研究が加速した。その後、Goodfellow ら [12] によって A.E. を高速に生成する手法である Fast Gradient Sign Method が提案された。本稿では 2.2 節で詳しく説明する。その後、Papernot ら [22] は、入力から出力への勾配をマッピングした “Saliency Map” を用いて出力を効率よく変化させられる点を探索し改変することで、改変量を少なく抑えたまま入力画像を A.E. にする手法を提案している。

1.1.2　防御手法とそれに対する攻撃

先崎・松浦 [27] は上記の Saliency Map の特性を用い、識別器とは別の検出器を用意して A.E. を検知する手法を提案している。そのようなアプローチではなく、識別器自体の構造を工夫して A.E. に対する耐性を高める研究もいくつか存在する。しかし、本節で述べるように多くの手法はうまくいっているとは言えない。

まず、Papernot ら [23] は NN のモデルサイズ削減に用いられる Distillation と呼ばれる手法を用いて A.E. の分類誤り率を 95%から 0.5%にまで減らす手法を提案した。しかしその後、Carlini と Wagner [5] によって NN への Distillation 適用に関わらず有用な攻撃アルゴリズムが提案され、A.E. に対する NN の識別誤り率は再度（100%まで）引き上げられた。また、A.E. を画像の拡大や縮小で無力化できるという主張 [17] は、それらに強い A.E. を生成する手法の提案 [1] によって（arXiv 上ではわずか数日で）破られているし、2017 年に入ってから提案された 10 個の新しい防御手法をすべて破ったと主張している論文 [6] もある。以上のように防御はあまりうまくいっているとは言えないのが現状であるが、各層における勾配の巨大化を抑えることで NN の頑健性を高める手法 [8]、NN の頑健性をより正確に測る手法 [19] などの研究が進んでおり、今後の進展が期待される。

A.T. は NN の訓練法を変更するアプローチの一つで、ここまで述べてきたネットワーク構造を変更するアプローチに比べれば NN の頑健性向上に成功していると言える。A.T. は Goodfellow ら [12] によって導入され、Fawzi ら [11]、Miyato ら [18]、Kurakin ら [14] によってその考察・改良・大規模データでの有効性評価が進められた。また、アンサンブル学習（性質の異なる複数の識別器を用いて一つの決定を下すこと）を A.T. と組み合わせた結果 [35] もあるが、弱い防御手法はアンサンブルしても効果が薄いという、Dropout 効果から導かれる直感とは異なる結果 [13] が知られており、今後のさらなる解析が待たれている。以上で見てきたように、A.T. を行った際の A.E. 以外の入力データ

	Normal	Random	Adversarial
通常の訓練	◎	○	×
敵対的訓練（A.T.）	◎	△	○
提案する訓練手法	◎	◎	○

表 1 提案手法の効果（Normal は通常データ、Random はランダムノイズが乗ったデータ、Adversarial は A.E. の意味。記号 ◎/○/△/× は各データに対する学習手法の識別精度を表す。）

耐性を取り扱った研究は存在しない。

1.1.3 応用

実社会応用を意識した研究も少しずつ始まっている。例えば、入力層に対して Dropout を行うことで敵対的マルウェアに耐性を持たせるマルウェア検知手法 [36]、音声で操作できるデバイスを人間には聞こえない音を用いて操作する攻撃とそれに対する防御手法 [7]、攻撃条件の緩和 [15]、交通標識の誤認識 *3 実証 [10]、CAPTCHA（歪んだ画像などを用いて人間と機械とを区別するテスト）の安全性を高めるために A.E. を用いる提案 [21] などがある。

1.2 本研究の貢献

本研究の貢献は次の 2 点である。

(1) A.T. に関して得られた新たな知見を報告する。通常の教師データのみを用いた（A.T. ではない）訓練を行った CNN は、通常のテストデータ及びランダムな（＝意図的でない）ノイズが乗ったデータに対して高い識別率を示すが、A.E. の識別率は低い。一方で A.T. を行うと通常のテストデータの識別率を保ったまま A.E. の識別率を大きく向上させることが可能であるが、ランダムなノイズが乗ったデータに対する識別率は大きく毀損してしまうことが判明した。既存研究においてはランダムなノイズが乗ったデータの識別率を示していないため、この事実は判明していなかった。ランダムなノイズが画像に加わることは現実的にも起こりうると考えられ、また攻撃者にとっても生成が容易なため、CNN に対する大きな脅威となる。

(2) 上記の問題を緩和する新たな訓練手法を提案する。提案手法の効果を既存手法と比較すると表 1 のようになる。A.T. においては通常の教師データと A.E. を訓練に用いるが、提案手法ではそれに加えてランダムなノイズが乗ったデータも訓練に用いる。比較的直感的な手法ではあるが、パラメータを様々に変えながら実験した結果、A.E. だけでなくランダムなノイズが乗った画像に対しても高い識別率を示す CNN を構成出来た。

1.3 本稿の構成

2 章では本稿で必要となる背景知識や用語を説明する。3 章では我々が発見した A.T. の副作用を述べ、それを緩和するための手法を提案する。4 章では実験によってその提案手法を評価する。5 章は本稿のまとめである。

*3 標識にわずかな細工をすることによって確率 1 で停止標識を速度制限標識であると誤認識させられること等が示されている。

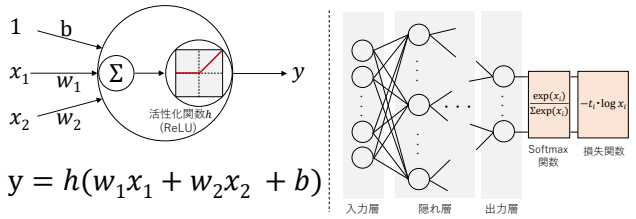

図 2 NN において各ニューロンが行う計算（左）と全体図（右）

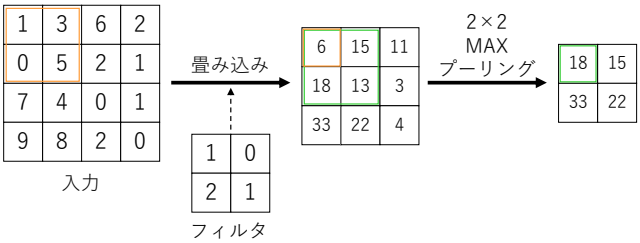

図 3 CNN における畳み込み層とプーリング層

2. 準備

本章では本研究の背景知識及び用語を説明する。

2.1 （畳み込み）ニューラルネットワーク

畳み込みニューラルネットワーク（CNN）の基本形は（単純）パーセプトロンを重ねた構造の前に畳み込み層を加えたものであり、以下でその要素技術を簡単に説明する。より詳しくはチュートリアル [31] などを参照されたい。

2.1.1 パーセプトロンとニューラルネットワーク

パーセプトロン（複数の信号それぞれに対応する重みを乗算し、その和と閾値の大小を比較して 0 か 1 をを出力するアルゴリズム）を多層化したものが NN の基本である。図 2 に示すように、活性化関数 (図 2 の h) を通した値が層の出力になる、分類問題においては最終層に Softmax 関数と呼ばれる（NN の出力を確率として解釈できるようにするための）層が入るなどの特徴があり、NN の活性化関数としてはランプ関数（Rectified Linear Unit, ReLU）[20] がよく用いられている。教師データを用いて NN の出力を計算し、損失関数（コスト関数）と呼ばれる正解ラベルとの差分（誤差）を計算する関数の出力を用いて NN の重みを自動的に更新する作業を NN の（誤差逆伝搬法による）学習、あるいは訓練と呼ぶ。具体的なアルゴリズムとしては（確率的）勾配降下法などが挙げられる。更新の速度を決定する学習率と呼ばれるパラメータは（基本的に）手動で決定する必要がある。

2.1.2 畳み込みニューラルネットワーク（CNN）

CNN とは通常の NN に畳み込み層とプーリング層を追加したもので、その概略は図 3 の通りである。畳み込み層においては入力と（学習によって更新される）フィルタの

積和演算を行う。通常の NN においては入力は 1 次元であったが CNN の入力とフィルタはともに 2 次元のデータ（チャネルまで考えればより高次元）を考慮するため、近いピクセルは似たような値である可能性が高いといった、画像の持つ空間的情報を活用することができる。プーリング層においては $N \times N$ の領域を 1 つに集約すること（最大値を取る MAX プーリングが一般的）で、微小な位置変化に対する頑健性の獲得が期待できる。

2.1.3 NN に関する種々のテクニック

学習の高速化や過学習（教師データに過剰に適応してしまい逆に汎化性能が低下してしまっている状態）の抑制等を目的として、多くの手法が開発されている。具体的には、ニューロンをある割合でランダムに欠落させ、表現力が制限された毎回微妙に異なる NN で学習させることで過学習を抑制しつつ、構造の異なるネットワークによるアンサンブル効果による精度向上が期待できる Dropout [29]、学習時にモーメントの考え方等を利用して収束を高速化する Adam [2] 等があるが、本稿では深く立ち入らない。

2.1.4 データセット

機械学習による画像認識のために、学習やテストに用いる画像とその正解ラベルをまとめた様々なデータセットが存在する。本稿では、その中でも最も基本的なデータセットのひとつである MNIST [16] を用いて実験や評価を行った。これは 0 から 9 までの手書き数字のデータセットであり、数百人によって書かれた数字がそれぞれ 28×28 ピクセルの画像になっている。また、このデータセットには学習用データとして 60000 枚、テスト用データとして 10000 枚の画像が含まれている。

2.2 Adversarial Examples

1 節で述べたように、学習器への入力に微小な改変を加えるだけで意図的に出力を誤らせることが可能であるという事実が明らかになっており、このような入力を Adversarial Examples (A.E.) [32] と呼ぶ。ここでは、攻撃者のモデルと A.E. の生成手法について簡単に説明する。

2.2.1 攻撃者のモデル

後述するように、A.E. の作成には対象とする学習器の内部情報が必要となる。よって本稿では、対象とする学習済みネットワークの損失関数やネットワーク構造、重みの値などすべての情報にアクセスできる、いわゆるホワイトボックス攻撃が可能な攻撃者を想定する。ただし注意すべきなのは、内部情報を秘匿すれば A.E. への実用的な対抗策として十分であるとは言えない点である。なぜなら A.E. には Transferability [24, 25] と呼ばれる性質、すなわち同じ機能を実現しようとしている学習器であれば、ある学習器に対し作成した A.E. がネットワーク構成や初期値、学習用データセットなどが異なる別の学習器に対しても高い確率で有効に作用してしまうという性質があることが判明

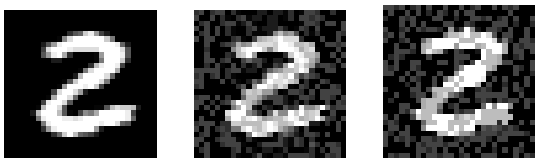

図 4 改変を行った MNIST の例。左から、(a) オリジナル、(b) ランダムノイズを乗せた画像、(c)A.E.。(b) と (c) は加えられたノイズが意図的なものであるかどうか人間の目ではほとんど区別できないことがわかる。

しているからである。この性質により、ブラックボックスな対象を攻撃する際にはその対象に対する代替学習器を構築し、それに対して A.E. を作成することで間接的に元の学習器を攻撃するといったシナリオが現実的な脅威として考えられる。

2.2.2 Fast Gradient Sign Method

A.E. の作成手法はこれまでいくつも提案されているが、その中でも初期からある手法でシンプルかつ高速なため基本となっているのが Fast Gradient Sign Method [12] である。この手法では、与えられたひとつの学習器と入力 $\mathbf{x}$ に対し改変量 $\boldsymbol{\eta}$ を求め、A.E. $\mathbf{x}_{adv}$ を

$$\mathbf{x}_{adv} = \mathbf{x} + \boldsymbol{\eta} \tag{1}$$

と定義する。ここで $\boldsymbol{\eta} = \epsilon \mathrm{sign}(\nabla_x J(\boldsymbol{\theta}, \mathbf{x}, y))$ である。$\boldsymbol{\theta}$ はモデルのパラメータ、$\mathbf{x}, y$ はそれぞれモデルへの入力と出力、$J(\boldsymbol{\theta}, \mathbf{x}, y)$ はニューラルネットワークのコスト関数、sign は $x > 0$ のとき 1、$x = 0$ のとき 0、$x < 0$ のとき -1 で定義される符号関数である。ϵ は入力に対して加える改変の度合いのパラメータである。すなわちこの手法では、通常の学習プロセスにおいて重みの更新に利用される損失関数の勾配を直接入力への改変に利用している。これにより、$\boldsymbol{\eta}$ を計算する際に指定したラベル y に対応する損失関数の値が上昇するため y であると推論されにくくなり、学習器の認識を誤らせる A.E. となる [*4]。この手法により生成した A.E. の例を図 4(c) に示す。

2.3 ランダムノイズ画像

画像に加えるランダムなノイズとしては様々なパターンが考えられるが、本稿では画像 $\mathbf{x}$ に対し以下のように定義される画像をランダムノイズが乗った画像 $\mathbf{x}_{rnd}$ とする。

$$\mathbf{x}_{rnd} = \mathbf{x} + \zeta \mathrm{sign}(\{r_i\}_{i=1..n}) \tag{2}$$

ただし r_i は正規分布 $\mathcal{N}(0, 1)$ に従う変数、n は $\mathbf{x}$ のサイズ、ζ は乗せるノイズの強度を決めるパラメータである。$\mathbf{x}_{rnd}$ の例を図 4(b) に示す。

2.4 Adversarial Training

A.E. への対抗策として提案されているのが、学習プロ

[*4] A.E. $\mathbf{x}_{adv}$ を $\mathbf{x}_{adv} = \mathbf{x} - \boldsymbol{\eta}$ と定義すれば任意のクラスに分類されるような A.E. を作成することも可能である。

	Normal	Random	Adversarial
通常	0.9898	0.9511	0.0637
A.T.	0.9882	0.5299	0.9467

表 2 通常の方法で訓練した学習器と A.T. を適用した学習器による MNIST の認識精度の違い

セスの中で A.E. を生成し教師データとして利用する Adversarial Training (A.T.) という手法である。この手法では、ネットワークの損失関数 $\tilde{J}(\boldsymbol{\theta}, \mathbf{x}, y)$ を以下のように定義する。

$$\tilde{J}(\mathbf{x}, y) = \alpha \cdot J(\boldsymbol{\theta}, \mathbf{x}_{adv}, y) + (1 - \alpha) \cdot J(\boldsymbol{\theta}, \mathbf{x}, y)$$

すなわち単純に A.E. を生成し学習用データセットとして扱うのではなく、毎回その時点でのネットワークに対して A.E. を生成しそれに対して計算した損失関数を通常の損失関数と混ぜることで A.E. を学習に利用する。ここでの α は割合を表すパラメータである。

3. A.T. の副作用とその緩和策

本章ではまず 2.4 節で説明した A.T. の副作用を述べた後、その緩和手法を提案する。

3.1 A.T. の副作用

A.E. の認識率を大きく向上させる A.T. であるが、一方で通常の CNN であればほぼ問題にならない程度のランダムノイズが画像に加わることで認識精度が低下してしまう副作用があることが判明した。これを表した実験結果を表 2 に示す。 これまで説明してきたように通常の学習手法だと A.E. の認識精度は大きく下がるが、CNN の特徴であるノイズへの頑強性によりランダムノイズを乗せた画像の認識率はかなり高いことがわかる。一方で、A.T. をした場合、意図的に認識率を下げるように改変された入力でも正しく認識できるようになっているが、逆にランダムにノイズを乗せた画像の認識精度が大きく低下してしまっている。このような問題は、我々の知る限りほとんど議論されてきていない。しかし CNN の実社会での応用を考える際、シナリオにもよるが A.E. による攻撃よりも入力になんらかのノイズが乗ってしまう場合のほうが多いと考えられるため、A.E. に対する耐性を保ちつつもランダムノイズが加わった画像を問題なく識別可能な学習器の構築手法を考えることには大きな意義がある。

3.2 副作用の緩和策

A.T. による副作用は、A.E. を学習に使用することで学習用データセットに偏りが生じてしまうことによるものだと推察される。よってこれを緩和するために、学習にランダムノイズを乗せたデータも利用するよう A.T. を拡張した手法を提案する。この手法はそれぞれ式 (1)、(2) で定義

層	構成
Convolution + ReLU	5x5x32
MaxPooling	2x2
Convolution + ReLU	5x5x64
MaxPooling	2x2
Fully Connected + ReLU	1024
Softmax	10

表 3 実験に用いた学習器の構成

される $\mathbf{x}_{adv}$、$\mathbf{x}_{rnd}$ を用いて以下の式で表される。

$$\tilde{J}(\mathbf{x}, y) = \alpha \cdot J(\boldsymbol{\theta}, \mathbf{x}_{adv}, y) + \beta \cdot J(\boldsymbol{\theta}, \mathbf{x}_{rnd}, y) + (1 - \alpha - \beta) \cdot J(\boldsymbol{\theta}, \mathbf{x}, y) \quad (3)$$

ここで、α は A.E. の割合を、β は入力にランダムなノイズを加えたものの割合を表し、$\alpha, \beta, \alpha + \beta \in [0, 1]$ である。つまり 2.4 節で説明した A.T. の式 (3) にランダムノイズの乗ったデータによる損失関数の項を追加することで学習時により多くのデータを利用するようになるということである。これらのパラメータを適切に設定することができれば、データセットの偏りが是正されどの種類のデータに対しても認識精度が向上すると考えられる。

4. 提案手法の評価

提案手法の有効性とパラメータの変動がもたらす影響を確かめ、最適なパラメータ値を探るために 2 つの実験を行った。1 つめの実験では、式 (3) のパラメータ α と β を変動させて学習器の識別率を測定した。2 つめの実験では、式 (1)、(2) のパラメータ ϵ と ζ を変動させて提案手法で学習を行った学習器を用い、テストデータの認識率を測ったものである。実験においては MNIST の学習用データ 60000 枚、テスト用データ 10000 枚を用いた。テスト用データに加えるランダムノイズとテスト用データから生成する A.E. はそれぞれ ϵ=ζ=0.25 で評価を行った。使用した NN の構成については表 3 の通りで、学習率及び Dropout 率はそれぞれ典型的な値である $1e-4$ 及び 0.5 を用い、10 エポック訓練した。フレームワークには Tensorflow [33] を用いた。

4.1 実験 1：パラメータ α, β

式 (3) のパラメータ α と β、すなわち A.E. とランダムノイズを乗せた画像の損失関数の割合を 0 から 1 まで変動させた結果を表 4, 5, 6 に示す。ただし $\alpha + \beta \in [0, 1]$ であるので、$\alpha + \beta > 1$ の部分は値を持たない。

まず何も改変を加えていないテストデータの認識精度を表した表 4 を見ると、ほとんどの場合通常の学習手法 (表中の α=β=0 に対応) と比べ遜色ない、場合によってはわずかに改善された精度になっていることがわかる。全体的な傾向としては、$\alpha > 0$ かつ $\alpha + \beta = 1$ でなければ大きく精度が落ちることはないとみなすことができる。

次にテストデータに ζ=0.25 でランダムノイズを加えた

データの認識精度を表した表 5 を見ると、パラメータの値によっては大きく精度が落ちていることが見て取れる。さらに、3 章で述べたように従来の A.T.(表中の $\alpha > 0$ かつ $\beta = 0$ に対応) では、α の値をどのように設定してもこのテストデータで高い認識率を出すことができないということがわかる。逆に β に少しでも値を持たせるだけでかなり精度が改善する。全体的には、$\beta > 0$ のとき α の値が大きくなるにつれ認識精度はわずかに下がっていく、β を大きくすると精度が上がる、などの傾向が見られる。

表 6 は、それぞれの学習器に対し ϵ=0.25 で生成した A.E. の認識精度である。α, β をどのように設定しても通常データほどは認識精度が上がらないことがわかる。最終的にはこれを元データの精度に比肩させることが目標である。α に注目すると、α は大きな値を持つほど認識精度が上がる傾向があることがわかる。これはランダムノイズの乗ったデータの認識精度とは逆の傾向であり、これらはトレードオフの関係にあると予想される。またこちらはランダムノイズの乗ったデータのときと同様に、α が 0 であるか否かで認識精度が大きく異なっている。これは A.T. の有用性を示していると言える。β については、$\alpha > 0$ においては小さければ小さいほど精度が高いという結果になっている。

4.2 実験 2：パラメータ ϵ, ζ

提案手法の中で用いている $\mathbf{x}_{adv}$, $\mathbf{x}_{rnd}$ のパラメータである、式 (1)、(2) で定義されるパラメータ ϵ と ζ をそれぞれ 0.05 から 0.5 まで変動させた結果を表 7, 8, 9 に示す。すなわち、学習時に生成する $\mathbf{x}_{adv}$, $\mathbf{x}_{rnd}$ の改変度合いを変えて学習を行った結果である。ここで α, β の値は、通常の A.T. においては $\alpha = 0.5$ 程度の値がよく用いられること、及び β は 0 でさえなければランダムノイズの乗ったデータに対する識別率を高く保てるという実験 1 の結果を踏まえ、$\alpha = 0.4$, $\beta = 0.1$ とした。

表 7 は通常のテストデータの認識精度である。ϵ の値が大きくなるとわずかに精度が下がる傾向があるが、全体的に ϵ や ζ の値に関わらず高い認識率であるといえる。

一方でランダムノイズの乗った画像の認識精度を表した表 8 ではパラメータによってはかなり認識精度が下がっている。具体的には ζ の値が小さく ϵ の値が大きいときである。これはより少ないノイズを乗せたデータで A.T. を行っている場合であり、従来の A.T. に近いため精度が下がっていると考えられる。

同様に A.E. の認識精度を示した表 9 でもほぼ A.T. を行っていないことに対応する ϵ の値が小さい場合に認識精度が下がっているが、逆に ϵ の値が大き過ぎても精度が下がっており ϵ が 0.2 から 0.3 程度のときに高い精度を出している。ϵ の値さえ適切であれば ζ の値の変動は認識精度にそこまで大きく影響していないため、ϵ の値の設定が肝要であることがわかる。また、これはテストデータの A.E. を ϵ=0.25 で生成していることも影響していると考えられる。紙面の都合で詳細は省略するが、ϵ=0.15 でテストデータの A.E. を生成すると認識精度のピークは ϵ が 0.15 から 0.25 程度のときになることが確認できた。すなわち、学習時に生成した A.E. と似た強度の A.E. に耐性がつくということである。よって、より一般的に A.E. への耐性を向上させるためには ϵ の値を変更しながら訓練するという手法が考えられるが、この手法の評価は今後の課題である。

5. まとめ

本稿では、A.E. への対抗策である A.T. を行った際に起こる副作用を指摘し、その対策として A.T. の拡張を提案した。また、提案手法を計算機実験によって評価し、その有効性とパラメータの変動が及ぼす影響を示した。

謝辞 本研究の一部は JSPS 科研費 17KT0081 の助成を受けた。また、本論文の第 2 著者は JST CREST（JPMJCR1688）の支援を受けている。

参考文献

[1] A. Athalye, I. Sutskever. Synthesizing Robust Adversarial Examples. arXiv:1707.07397, 2017.
[2] J. Ba, D. Kingma. Adam: A Method for Stochastic Optimization. ICLR2015, 2015.
[3] M. Barreno, B. Nelson, A.D. Joseph, J.D. Tygar. The Security of Machine Learning. Machine Learning 81(2), pp.121-148, 2010.
[4] B. Biggio, B. Nelson, P. Laskov. Poisoning Attacks against Support Vector Machines. ICML2012, 2012.
[5] N. Carlini, D.A. Wagner. Towards Evaluating the Robustness of Neural Networks. IEEE S&P2017, pp.39-57.
[6] N. Carlini, D.A. Wagner. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. arXiv:1705.07263, 2017.
[7] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D.A. Wagner, W. Zhou. Hidden Voice Commands. UsenixSecurity2016, pp.513-530, 2016.
[8] M. Cisse, P. Bojanowski, E, Grave, Y. Dauphin, N. Usunier. Parseval Networks: Improving Robustness to Adversarial Examples. ICML2017, pp.854-863, 2017.
[9] N.N. Dalvi, P.M. Domingos, Mausam, S.K. Sanghai, D. Verma. Adversarial Classification. KDD2004, pp.99-108, 2004.
[10] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, D. Song. Robust Physical-World Attacks on Machine Learning Models. arXiv:1707.08945, 2017.
[11] A. Fawzi, S. Moosavi-Dezfooli, P. Frossard. Robustness of Classifiers: From Adversarial to Random Noise. NIPS2016, pp.1624-1632, 2016.
[12] I.J Goodfellow, J. Shlens, C. Szegedy. Explaining and Harnessing Adversarial Examples. ICLR2015, 2015.
[13] W. He, J. Wei, X. Chen, N. Carlini, D. Song. Adversarial Example Defenses: Ensembles of Weak Defenses are not Strong. arXiv:1706.04701, 2017.
[14] A. Kurakin, I.J. Goodfellow, S. Bengio. Adversarial Machine Learning at Scale. ICLR2017, 2017.
[15] A. Kurakin, I.J. Goodfellow, S. Bengio. Adversarial Examples in the Physical World. ICLR2017, 2017.
[16] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner. Gradient-

		α										
		0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
β	1.0	0.9892	-	-	-	-	-	-	-	-	-	-
	0.9	0.9897	0.9666	-	-	-	-	-	-	-	-	-
	0.8	0.9902	0.9871	0.9616	-	-	-	-	-	-	-	-
	0.7	0.9896	0.9892	0.9887	0.9704	-	-	-	-	-	-	-
	0.6	0.9902	0.9913	0.9894	0.9874	0.9690	-	-	-	-	-	-
	0.5	0.9892	0.9896	0.9884	0.9874	0.9861	0.9709	-	-	-	-	-
	0.4	0.9900	0.9907	0.9898	0.9894	0.9876	0.9860	0.9471	-	-	-	-
	0.3	0.9894	0.9913	0.9900	0.9878	0.9881	0.9882	0.9835	0.9625	-	-	-
	0.2	0.9895	0.9909	0.9901	0.9888	0.9875	0.9875	0.9864	0.9833	0.9667	-	-
	0.1	0.9871	0.9894	0.9886	0.9899	0.9888	0.9882	0.9869	0.9851	0.9834	0.9572	-
	0.0	0.9889	0.9894	0.9899	0.9890	0.9884	0.9887	0.9851	0.9840	0.9854	0.9803	0.9195

表 4 α, β の値を変えて学習した学習器の MNIST 認識精度。($\epsilon = \zeta = 0.25$)

		α										
		0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
β	1.0	0.9854	-	-	-	-	-	-	-	-	-	-
	0.9	0.9836	0.9861	-	-	-	-	-	-	-	-	-
	0.8	0.9866	0.9861	0.9853	-	-	-	-	-	-	-	-
	0.7	0.9861	0.9852	0.9840	0.9849	-	-	-	-	-	-	-
	0.6	0.9855	0.9856	0.9856	0.9827	0.9825	-	-	-	-	-	-
	0.5	0.9834	0.9849	0.9850	0.9811	0.9804	0.9807	-	-	-	-	-
	0.4	0.9870	0.9842	0.9820	0.9829	0.9801	0.9798	0.9791	-	-	-	-
	0.3	0.9847	0.9847	0.9845	0.9808	0.9817	0.9746	0.9768	0.9743	-	-	-
	0.2	0.9843	0.9812	0.9816	0.9804	0.9748	0.9764	0.9705	0.9703	0.9700	-	-
	0.1	0.9818	0.9806	0.9763	0.9760	0.9677	0.9720	0.9706	0.9658	0.9622	0.9648	-
	0.0	0.9618	0.4962	0.4422	0.3871	0.4071	0.4775	0.4895	0.4588	0.4316	0.4617	0.6414

表 5 α, β の値を変えて学習した学習器の、ランダムノイズが乗った MNIST の認識精度。($\epsilon = \zeta = 0.25$)

		α										
		0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
β	1.0	0.1597	-	-	-	-	-	-	-	-	-	-
	0.9	0.1502	0.8508	-	-	-	-	-	-	-	-	-
	0.8	0.1538	0.8358	0.8934	-	-	-	-	-	-	-	-
	0.7	0.1540	0.8546	0.8932	0.8998	-	-	-	-	-	-	-
	0.6	0.1392	0.8641	0.8950	0.9081	0.9199	-	-	-	-	-	-
	0.5	0.1306	0.8701	0.8998	0.9160	0.9058	0.9116	-	-	-	-	-
	0.4	0.1566	0.8919	0.9100	0.9206	0.9180	0.9278	0.9317	-	-	-	-
	0.3	0.1260	0.8843	0.9158	0.9305	0.9280	0.9320	0.9223	0.9240	-	-	-
	0.2	0.1165	0.8944	0.9203	0.9222	0.9232	0.9326	0.9348	0.9322	0.9352	-	-
	0.1	0.0986	0.9109	0.9200	0.9302	0.9276	0.9379	0.9364	0.9412	0.9443	0.9457	-
	0.0	0.0797	0.9251	0.9268	0.9305	0.9363	0.9447	0.9413	0.9420	0.9482	0.9435	0.9380

表 6 α, β の値を変えて学習した学習器の、A.E. 認識精度。($\epsilon = \zeta = 0.25$)

		ζ									
		0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
ϵ	0.50	0.9873	0.9874	0.9865	0.9866	0.9894	0.9874	0.9878	0.9886	0.9879	0.9852
	0.45	0.9885	0.9866	0.9868	0.9888	0.9875	0.9872	0.9880	0.9865	0.9880	0.9893
	0.40	0.9853	0.9879	0.9878	0.9894	0.9881	0.9864	0.9894	0.9887	0.9882	0.9881
	0.35	0.9868	0.9875	0.9874	0.9882	0.9876	0.9875	0.9887	0.9895	0.9872	0.9879
	0.30	0.9882	0.9872	0.9856	0.9889	0.9883	0.9888	0.9880	0.9880	0.9878	0.9865
	0.25	0.9878	0.9895	0.9869	0.9874	0.9887	0.9888	0.9884	0.9881	0.9880	0.9886
	0.20	0.9885	0.9905	0.9890	0.9898	0.9886	0.9895	0.9891	0.9888	0.9896	0.9889
	0.15	0.9904	0.9909	0.9906	0.9908	0.9909	0.9906	0.9913	0.9906	0.9908	0.9900
	0.10	0.9909	0.9907	0.9920	0.9921	0.9911	0.9909	0.9905	0.9910	0.9914	0.9919
	0.05	0.9910	0.9915	0.9907	0.9915	0.9916	0.9920	0.9919	0.9911	0.9902	0.9915

表 7 ϵ, ζ の値を変えて学習した学習器の、MNIST 認識精度。(α=0.4, β=0.1)

		ζ									
		0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
ϵ	0.50	0.5209	0.8705	0.9482	0.9722	0.9806	0.9784	0.9798	0.9760	0.9731	0.9737
	0.45	0.4491	0.8425	0.9570	0.9782	0.9779	0.9797	0.9773	0.9762	0.9748	0.9729
	0.40	0.4351	0.7892	0.9458	0.9717	0.9763	0.9745	0.9759	0.9759	0.9726	0.9748
	0.35	0.4777	0.8329	0.9500	0.9686	0.9736	0.9741	0.9748	0.9750	0.9711	0.9710
	0.30	0.4952	0.7258	0.9282	0.9688	0.9722	0.9749	0.9756	0.9746	0.9790	0.9754
	0.25	0.5361	0.7294	0.9121	0.9637	0.9725	0.9785	0.9767	0.9794	0.9786	0.9779
	0.20	0.5568	0.7261	0.9360	0.9653	0.9772	0.9801	0.9829	0.9800	0.9828	0.9836
	0.15	0.7675	0.8938	0.9617	0.9818	0.9834	0.9848	0.9848	0.9855	0.9853	0.9864
	0.10	0.9220	0.9736	0.9851	0.9821	0.9876	0.9855	0.9884	0.9856	0.9881	0.9883
	0.05	0.9771	0.9840	0.9864	0.9869	0.9882	0.9879	0.9877	0.9892	0.9866	0.9878

表 8 ϵ, ζ の値を変えて学習した学習器の、ランダムノイズが乗った MNIST の認識精度。(α=0.4, β=0.1)

		ζ									
		0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
ϵ	0.50	0.8114	0.7898	0.7580	0.7894	0.8048	0.8151	0.8478	0.8336	0.8012	0.8125
	0.45	0.8763	0.8645	0.8321	0.8417	0.8630	0.8668	0.8758	0.8828	0.8674	0.8791
	0.40	0.9016	0.8912	0.9098	0.8926	0.9178	0.9034	0.9111	0.8950	0.9088	0.9111
	0.35	0.9268	0.9309	0.9303	0.9260	0.9315	0.9368	0.9512	0.9359	0.9270	0.9378
	0.30	0.9401	0.9378	0.9450	0.9517	0.9452	0.9500	0.9373	0.9378	0.9477	0.9336
	0.25	0.9403	0.9395	0.9427	0.9452	0.9326	0.9318	0.9307	0.9349	0.9216	0.9240
	0.20	0.9007	0.9087	0.9029	0.9021	0.8933	0.8979	0.8837	0.8849	0.8777	0.8724
	0.15	0.8161	0.8105	0.8203	0.8204	0.8224	0.8347	0.8210	0.8062	0.8060	0.8166
	0.10	0.7068	0.6925	0.7041	0.7000	0.6914	0.7214	0.6818	0.6880	0.6865	0.6946
	0.05	0.4131	0.3819	0.3496	0.4185	0.4424	0.3741	0.4337	0.4111	0.3726	0.3931

表 9 ϵ, ζ の値を変えて学習した学習器の、A.E. 認識精度。(α=0.4, β=0.1)

based learning applied to document recognition. Proceedings of the IEEE, pp.2278-2324, 1998.

[17] J. Lu, H. Sibai, E. Fabry, D. Forsyth. NO Need to Worry about Adversarial Examples in Object Detection in Autonomous Vehicles. CVPR2017 (to appear).

[18] T. Miyato, S. Maeda, M. Koyama, K. Nakae, S. Ishii. Distributional Smoothing by Virtual Adversarial Examples. ICLR2016, 2016.

[19] S. Moosavi-Dezfooli, A. Fawzi, P. Frossard. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. CVPR2016, pp.2574-2582, 2016.

[20] V. Nair, G.E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. ICML2010, pp.807-814, 2010.

[21] M. Osadchy, J. Hernandez-Castro, S. Gibson, O. Dunkelman, D. Perez-Cabo. No Bot Expects the DeepCAPTCHA! Introducing Immutable Adversarial Examples, with Applications to CAPTCHA Generation. IEEE TIFS (to appear), 2017.

[22] N. Papernot, P.D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, A. Swami. The Limitations of Deep Learning in Adversarial Settings. IEEE EuroS&P2016, pp.372-387, 2016.

[23] N. Papernot, P.D. McDaniel, X. Wu, S. Jha, A. Swami. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. IEEE S&P2016, pp.582-597.

[24] N. Papernot, P. McDaniel, I. Goodfellow. Transferability in Machine Learning: From Phenomena to Black-box Attacks Using Adversarial Samples. arXiv:1605.07277, 2016.

[25] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, B.Z. Celik, A. Swami. Practical Black-Box Attacks against Deep Learning Systems using Adversarial Examples. arXiv:1602.02697, 2016.

[26] 株式会社 Preferred Networks, ファナック株式会社. 第 3 回日本ベンチャー大賞において経済産業大臣賞（ベンチャー企業・大企業等連携賞）を受賞. https://www.preferred-networks.jp/ja/news/pr20170220

[27] 先崎佑弥, 松浦幹太. 深層学習に対し意図的に誤判定を起こさせる入力の検知手法. SCIS2017, 1C1-6.

[28] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis. Mastering the Game of Go with Deep Neural Networks and Tree Search. Nature 529(7587), pp.484-489, 2016.

[29] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 15(1): pp.1929-1958, 2014.

[30] N. Srndic, P. Laskov. Practical Evasion of a Learning-Based Classifier: A Case Study. IEEE S&P2014, pp.197-211, 2014.

[31] Stanford University. Deep Learning Tutorial http://deeplearning.stanford.edu/tutorial

[32] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I.J Goodfellow, R. Fergus. Intriguing Properties of Neural Networks. ICLR2014, 2014.

[33] TensorFlow http://www.tensorflow.org

[34] 株式会社東芝. 2016 年度人工知能学会現場イノベーション賞・金賞を受賞－四日市工場における半導体生産性改善－. https://www.toshiba.co.jp/rdc/detail/1706_04.htm

[35] F. Tramer, A. Kurakin, N. Papernot, D. Boneh, P.D. McDaniel. Ensemble Adversarial Training: Attacks and Defenses. arXiv:1705.07204, 2017.

[36] Q. Wang, W. Guo, K. Zhang, A.G. Ororbia II, X. Xing, X. Liu, C.L. Giles. Adversary Resistant Deep Neural Networks with an Application to Malware Detection. KDD2017, pp.1145-1153, 2017.