technische universität
dortmund

Uncertainty quantification in machine learning

Adversarial Examples can be Effective Data Augmentation for Unsupervised Machine Learning

Waldemar Voos

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl für Data Science and Data Engineering
(LS 9)
https://ls9-www.cs.tu-dortmund.de/

Contents

Chapter 1

Introduction

1.1 Motivation

Adversial examples are a common technique to evaluate and improve the robustness of Machine Learning models. In this paper, we create adversial examples with our new supervision-free framework for unsupervised and supervised learning tasks. In order to create adversial examples, we make use of the new MinMax algorithm developed in this paper. The algorithm works with a per-sample mutual information neural estimator (MINE). This is a neural network which can measure the similarity of two different data samples. Apart from approving robustness of machine learning models, we even can improve performance, accuracy and visual quality in comparison to other frameworks, too.

1.2 Adversial Examples

An adversial example 1.1 is a perturbed data sample leading to misclassification. A perturbed data sample is a native data sample added with noise. A human being is still able to classify correctly, but the machine learning model usually cannot without adversial defense techniques.

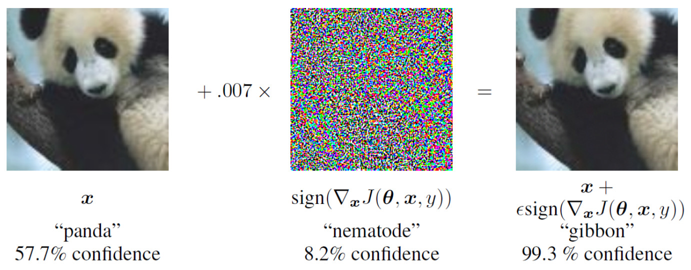

+ .007 ×
=
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

Figure 1.1: Adversial Example ([C15])

1.3 Mutual Information Neural Estimator (MINE)

A MINE compounds mutual information (MI) and a convolutional neural network (CNN).

1.3.1 Mutual Information (MI)

MI 1.1 calculates the dependency between two random variables X and Z with the joint distribution $\mathbb{P}_{XZ}$ and the marginal distributions $\mathbb{P}_X$ and $\mathbb{P}_Z$ as input parameters to the Kullback-Leibler (KL-) divergence. If I(X,Y) $=$ 0, X and Y are independent and we cannot conclude any information from Y out of X. If I(X,Y) $>$ 0 there is at least some dependency between the data.

For the MINE algorithm, we use a dual representation 1.2 from [D21]. With that representation, we are able to calculate the maximal lower bound of the mutual information efficiently [C-M21].

$$I(X,Y) = D_{KL}(\mathbb{P}_{XZ}||\mathbb{P}_X \otimes \mathbb{P}_Z) \tag{1.1}$$

$$D_{KL}(\mathbb{P}||\mathbb{Q}) \geq \underset{T\in\mathcal{F}}{sup}\, \mathbb{E}_{\mathbb{P}}[T] - log(\mathbb{E}_{\mathbb{Q}}[e^T]) \tag{1.2}$$

1.3.2 Artificial Neural Networks

In the picture 1.2 we see a special case of an artificial neural network - a multilayer perceptron (MLP). The blue lines should represent the forward calculation and the red dashed lines represent the backpropagation process, in which the minimum of the error function is calculated by the gradient descent. In the paper it will be the projected gradient descent, because we have to consider constraints and projection.

The green circles with a blue touch represent neurons 1.3. An artificial neuron gets parameterized inputs. Those will be summed up and forwarded to an activation function. This activation function - which will be a RELU in the paper's approach - will activate the neuron, if it satisfies the threshold.

1.3.3 Convolutional Neural Network (CNN)

CNNs 1.4 have several layers. In the convolution layer we pass our filters which recognize features and save them in feature maps. Our MINE algorithm primarily use the first convolution layer of our inputs, since it delivers the best results.

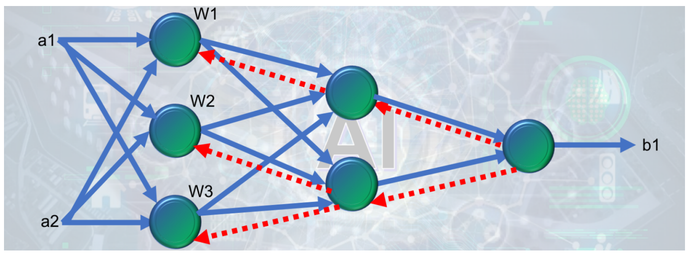


Figure 1.2: MLP with Backpropagation (https://vinodsblog.com/2019/02/17/deep-learning-backpropagation-algorithm-basics/)

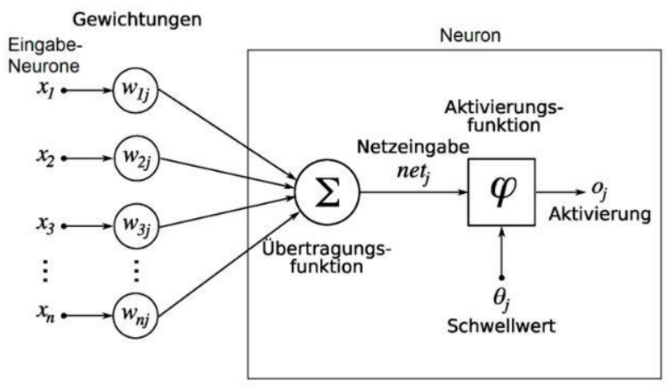


Figure 1.3: https://datasolut.com/neuronale-netzwerke-einfuehrung/

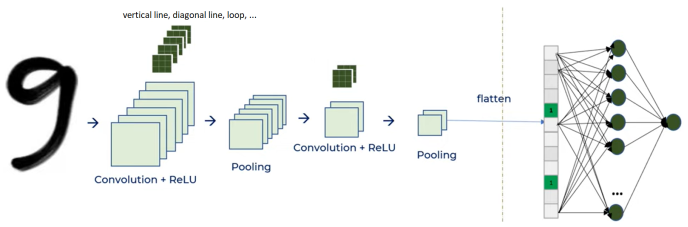


Figure 1.4: Convolutional Neural Network (https://www.youtube.com/watch?v=zfiSAzpy9NM)

```
1: **Require:** input sample $x$, perturbed sample $x+\delta$, 1st convolution layer output $conv(\cdot)$, MI neural estimator $I(\theta)$
2: Initialize neural network parameters $\theta$
3: Get $\{conv(x)_k\}_{k=1}^{K}$ and $\{conv(x+\delta)_k\}_{k=1}^{K}$ via $1^{st}$ convolution layer
4: **for** $t$ in $T_I$ iterations **do**
5:     Take $K$ samples from the joint distribution: $\{conv(x)_k, conv(x+\delta)_k\}_{k=1}^{K}$
6:     Shuffle $K$ samples from $conv(x+\delta)$ marginal distribution: $\{conv(x+\delta)_{(k)}\}_{k=1}^{K}$
7:     Evaluate the mutual information estimate $I(\theta) \leftarrow \frac{1}{K}\sum_{k=1}^{K} T_\theta(conv(x)_k, conv(x+\delta)_k) - \log\left(\frac{1}{K}\sum_{k=1}^{K}\exp[T_\theta(conv(x)_k, conv(x+\delta)_{(k)})]\right)$
8:     $\theta \leftarrow \theta + \nabla_\theta I(\theta)$
9: **Return** $I(\theta)$
```

Figure 1.5: MINE algorithm ([C-M21])

1.4 MINE Algorithm

While computing MI can be difficult, this neural network 1.5 maximizes the lower bound of mutual information of two random variables. For that reason, we use the Donsker-Varadhan representation 1.2.

Our inputs are a native sample x and the perturbed sample x $+ \delta$. x $+ \delta$ is the adversial example candidate passed from our MinMax Algorithm 2.1. In the beginning of our algorithm, we initialize the network parameters called θ. After that we calculate the first convolution layer of both samples, the native and perturbed to extract our K feature maps. K is a number we pass manually. The loop has a manually passed number of iterations. Like we have seen in 1.3.1, we need the joint distribution and the marginal distribution. In line 5 and 6 of the algorithm, we calculate their auxiliary distributions. Then we calculate the lower bound of mutual information estimate with the neural network and finally update the parameters θ.

Chapter 2

MinMax Algorithm

The gist of the MinMax algorithm is to optimize the c and δ value and so return an adversial example with the highest mutual information in supervised tasks leading to misclassification and the lowest mutual information in unsupervised tasks leading to lower reconstruction error. Those calculations are done in line four and eight of the algorithm 2.1.

δ, which is the grade of perturbation, should be maximized in supervised tasks and minimized in unsupervised tasks.

So, in the supervised tasks we would like to have most similar examples leading to misclassification, because we can still have very good visual quality.

In our unsupervised tasks we would like to have most dissimilar examples leading to generalization errors, but still having lower reconstruction loss, for example in autoencoder tasks. This switch is done by a simple change of sign.

After that δ should be projected to a feasible set of $\delta \in [-\epsilon,\epsilon]$, where $\epsilon \in [0.1, 1.0]^d$ and x + $\delta \in [0, 1]^d$. In 2.1 we can see a corresponding representation of this convex optimization. Those constraints derive from the L_p-Norm perturbation. We use this to be able to easily compare to other algorithms.

In the inner part, we have to maximize the variable c $\geq$ 0. This variable is multiplicated with the attack success evaluation function $f_x^+(x + \delta)$, where $f_x^+(x + \delta) \leq 0$ means that we found an adversial example. f^+ means, that you pass the result of the function to a RELU function. One reason why other approaches are less successful in our test environment is, that they have a constant c value and just use the L_p-Norm perturbation. In the paper's approach, we are going to optimize the c in every iteration.

The use of the projected gradient descent is done for optimization of c and δ. In the paper, it is also proved, that the algorithm converges.

```
Algorithm 1: MinMax Attack Algorithm
1: Require: data sample $x$, attack criterion $f_x(\cdot)$, step sizes $\alpha$ and $\beta$, perturbation bound $\epsilon$, # of iterations $T$
2: Initialize $\delta_0 = 0$, $c_0 = 0$, $\delta^* =$ null, $I^*_\Theta = -\infty$, $t = 1$
3: for $t$ in $T$ iterations do
4:     $\delta_{t+1} = \delta_t - \alpha \cdot (c \cdot \nabla f^+_x(x + \delta_t) - \nabla I_\Theta(x, x + \delta_t))$
5:     Project $\delta_{t+1}$ to $[-\epsilon, \epsilon]$ via clipping
6:     Project $x + \delta_{t+1}$ to $[0, 1]$ via clipping
7:     Compute $I_\Theta(x, x + \delta_{t+1})$
8:     Perform $c_{t+1} = (1 - \frac{\beta}{t^{1/4}}) \cdot c_t + \beta \cdot f^+_x(x + \delta_{t+1})$
9:     Project $c_{t+1}$ to $[0, \infty]$
10:    if $f_x(x+\delta_{t+1}) \leq 0$ and $I_\Theta(x, x+\delta_{t+1}) > I^*_\Theta$ then
11:        update $\delta^* = \delta_{t+1}$ and $I^*_\Theta = I_\Theta(x, x + \delta_{t+1})$
12: Return $\delta^*$, $I^*_\Theta$
```

Figure 2.1: MinMax Algorithm ([C-M21])

$$\underset{\delta:x+\delta\in[0,1]^d,\delta\in[-\epsilon,\epsilon]^d}{MIN} \; \underset{c\geq 0}{MAX} \; F(\delta, c) \triangleq c \cdot f^+_x(x + \delta) - I_\Theta(x, x + \delta) \tag{2.1}$$

Chapter 3

Evaluation

3.1 Supervised Adversial Examples

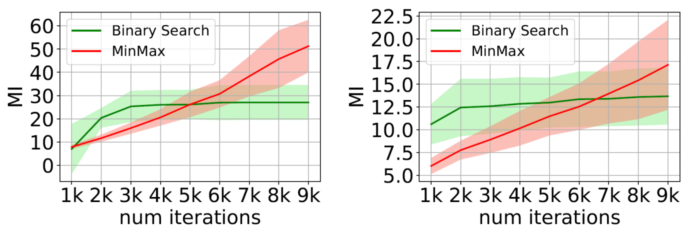
We compare supervised adversial examples with our new MinMax algorithm and common techniques so far - the penalty-based ones. We use them on two different datasets. The wellknown MNIST with grayscale numbers and CIFAR-10 with coloured pictures of 10 classes like trucks, ships, etc.. We observe strongly better MI on both datasets MNIST and CIFAR-10 3.1 with our approach. The attack success rate is on both sides 100% in that test set up.

We can observe 3.2 how the MI is developing over the number of iterations. Penalty-based attacks (green) like in [D17] pretty quickly reaches their peak, whereas MinMax improves over the number of iterations and gets better on both datasets. For some reason, the question if the performance is still useful after so many iterations is not answered. Also, it would be very interesting, if the MI would increase even after 9000 iterations and if or when it saturates.

Moreover, we can see 3.3 why a better mutual information is useful. With the higher mutual information in our framework, we get significantly better visual quality against the common adversial techniques, the Projected Gradient Descent attack. PGD has very good visual quality when using a small number for ϵ. But this leads to a low attack success rate.

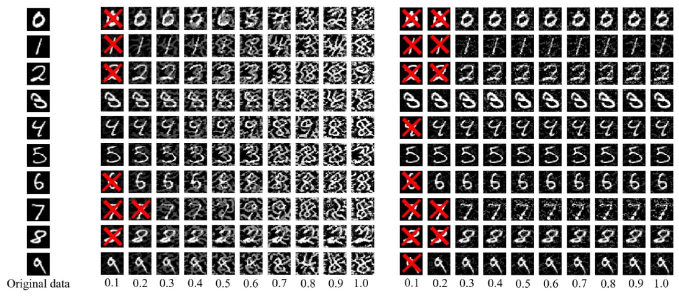
	MNIST		CIFAR-10	
	ASR	MI	ASR	MI
Penalty-based	100%	28.28	100%	13.69
MinMax	100%	**51.29**	100%	**17.14**

Figure 3.1: ASR and MI over 1000 Adversial Examples ([C-M21])



(a) MNIST (b) CIFAR-10

Figure 3.2: Mean and Standard Deviation of Supervised Adversial Attacks ([C-M21])



(a) (b) PGD attack (c) MinMax attack

Figure 3.3: Mean and Standard Deviation of Supervised Adversial Attacks ([C-M21])

3.2 Unsupervised Adversial Examples

3.2.1 Data Reconstruction

In data reconstruction tasks using autoencoders, we can observe lower reconstruction loss with our framework than with the original model or other common techniques. Furthermore, the generated unsupervised adversial examples serve as data augmentation, which improves the reconstruction loss tremendously. The evaluation 3.4 shows that on both data sets - MNIST and SVHN - we perform better than other common unsupervised adversial example techniques. With sparse autoencoders, we improve the reconstruction error on MNIST dataset up to 56.7% and up to 73.5% on SVHN dataset. Also, the attack success rate is usually higher in comparison to the other techniques.

3.2.2 Representation Learning

Using the state-of-the-art feature selection method and its test set in representation learning - concrete autoencoder [J19] - we can improve the reconstruction error generating UAEs for data augmentation with our framework up to 11% 3.5. The probable reason for the change for the worse in the dataset of Coil-20 is the result of the low attack success rate followed by too fewer features recognized by [J19].

MNIST									
	Reconstruction Error (test set)					ASR (training set)			
Autoencoder	Original	MINE-UAE	L_2-UAE	GA ($\sigma = 0.01$)	GA ($\sigma = 10^{-3}$)	MINE-UAE	L_2-UAE	GA ($\sigma = 0.01$)	GA ($\sigma = 10^{-3}$)
Sparse	0.00561	**0.00243** (↑ 56.7%)	0.00348 (↑ 38.0%)	0.00280±2.60e-05 (↑ 50.1%)	0.00280±3.71e-05 (↑ 50.1%)	100%	99.18%	54.10%	63.95%
Dense	0.00258	**0.00228** (↑ 11.6%)	0.00286 (↓ 6.0%)	0.00244±0.00014 (↑ 5.4%)	0.00238±0.00012 (↑ 7.8%)	92.99%	99.94%	48.53%	58.47%
Convolutional	0.00294	**0.00256** (↑ 12.9%)	0.00364 (↓ 23.8%)	0.00301±0.00011 (↓ 2.4%)	0.00304±0.00015 (↓ 3.4%)	99.86%	99.61%	68.71%	99.61%
Adversarial	0.04785	**0.04581** (↑ 4.3%)	0.06098 (↓ 27.4%)	0.05793±0.00501 (↓ 21%)	0.05544±0.00567 (↓ 15.86%)	98.46%	43.54%	99.79%	99.83%
SVHN									
Sparse	0.00887	**0.00235** (↑ 73.5%)	0.00315 (↑ 64.5%)	0.00301±0.00137 (↑ 66.1%)	0.00293±0.00078 (↑ 67.4%)	100%	72.16%	72.42%	79.92%
Dense	0.00659	**0.00421** (↑ 36.1%)	0.00550 (↑ 16.5%)	0.00858±0.00232 (↓ 30.2%)	0.00860±0.00190 (↓ 30.5%)	99.99%	82.65%	92.3%	93.92%
Convolutional	0.00128	**0.00095** (↑ 25.8%)	0.00121 (↑ 5.5%)	0.00098 ± 3.77e-05 (↑ 25.4%)	0.00104±7.41e-05 (↑ 18.8%)	100%	56%	96.40%	99.24%
Adversarial	0.00173	**0.00129** (↑ 25.4%)	0.00181 (↓ 27.4%)	0.00161±0.00061 (↑ 6.9%)	0.00130±0.00037 (↑ 24.9%)	94.82%	58.98%	97.31%	99.85%

Figure 3.4: Reconstruction Loss of Unsupervised Adversial Examples ([C-M21])

	Reconstruction Error (test set)		Accuracy (test set)		ASR
Dataset	Original	MINE-UAE	Original	MINE-UAE	MINE-UAE
MNIST	0.01170	**0.01142** (↑ 2.4%)	94.97%	95.41%	99.98%
Fashion MMIST	0.01307	**0.01254** (↑ 4.1%)	84.92%	85.24%	99.99%
Isolet	0.01200	**0.01159** (↑ 3.4%)	81.98%	82.93%	100%
Coil-20	**0.00693**	0.01374 (↓ 98.3%)	98.96%	96.88%	9.21%
Mice Protein	0.00651	**0.00611** (↑ 6.1%)	89.81%	91.2%	40.24%
Activity	0.00337	**0.00300** (↑ 11.0%)	83.38%	84.45%	96.52%

Figure 3.5: Reconstruction Error in Representation Learning with UAE Data Augmentation ([C-M21])

3.2.3 Contrastive Learning

Also in contrastive learning, we can observe an improvement of contrastive loss and accuracy with UAE data augmentation generated by our framework for the state-of-the-art algorithm - SimCLR [G18].

Chapter 4

Future Work

In the paper, the framework was compared to various other frameworks in supervised learning or was serving as a data augmentation tool in unsupervised learning to improve the according loss function and accuracy.

We tested in very limited test sets and environments. To realize how good our framework is, we have to test it in reality with real life examples. One future work could be using the framework in autonomous driving to manipulate real world ML models integrated in nowadays cars, to see, if the framework is comparable to the state-of-the-art ones in those scenarios.

List of Figures

Bibliography

[C15] Goodfellow I. J.; Shlens J.; Szegedy C. “EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES”. In: *https://arxiv.org/pdf/1412.6572.pdf* (2015).

[D17] Carlini N.; Wagner D. “Towards Evaluating the Robustness of Neural Networks”. In: *https://arxiv.org/pdf/1608.04644.pdf* (2017).

[G18] Chen T.; Kornblith S.; Norouzi M.; Hinton G. “A Simple Framework for Contrastive Learning of Visual Representations”. In: *http://proceedings.mlr.press/v119/chen20j/che* (2018).

[J19] Abid A.; Balin M. F.; Zou J. “Concrete Autoencoders: Differentiable Feature Selection and Reconstruction”. In: *http://proceedings.mlr.press/v97/balin19a/balin19a.pdf* (2019).

[C-M21] Hsu C.-Y.; Chen P.-Y.; Lu S.; Liu S.; Yu C.-M. “Adversarial Examples can be Effective Data Augmentation for Unsupervised Machine Learning”. In: *https://arxiv.org/pdf/2103.* (2021).

[D21] Belghazi M. I.; Baratin A.; Rajeswar S.; Ozair S.; Bengio Y.; Courville A.; Hjelm R D. “Mutual Information Neural Estimation”. In: *https://arxiv.org/pdf/1801.04062.pdf* (2021).