Adversarial examples - vulnerability of machine learning methods and prevention

Petra Vidnerová

Institute of Computer Science
The Czech Academy of Sciences

2018

Outline

- Introduction
- Works on adversarial examples
- Our work
  - Genetic algorithm
  - Experiments on MNIST
- Ways to robustness to adversarial examples
- Deep RBF networks

Introduction

- Applying an imperceptible non-random perturbation to an input image, it is possible to arbitrarily change the machine learning model prediction.

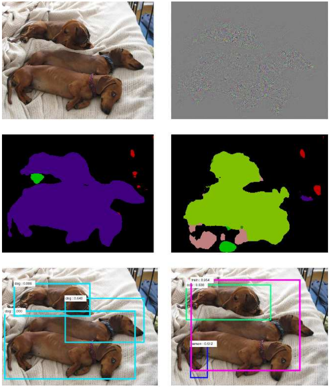
57.7% Panda 99.3% Gibbon

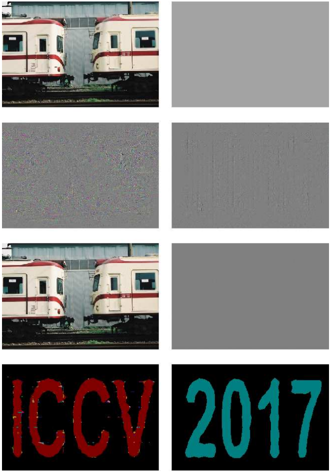
Figure from *Explaining and Harnessing Adversarial Examples* by Goodfellow et al.

- Such perturbed examples are known as *adversarial examples*. For human eye, they seem close to the original examples.
- They represent a security flaw in classifier.

Introduction

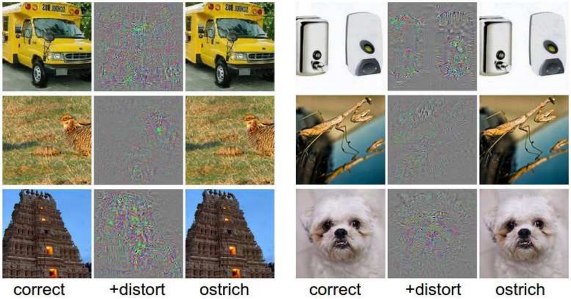
- *Adversarial Examples for Semantic Segmentation and Object Detection.*
  2017, Cihang Xie et al.





Works on adversarial examples I.

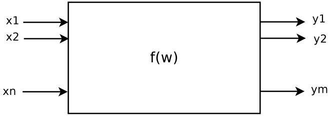
- *Intriguing properties of neural networks*. 2014,Christian Szegedy et al.



- Perturbations are found by optimising the input to maximize the prediction error (L-BFGS).

Works on adversarial examples I.

Learning



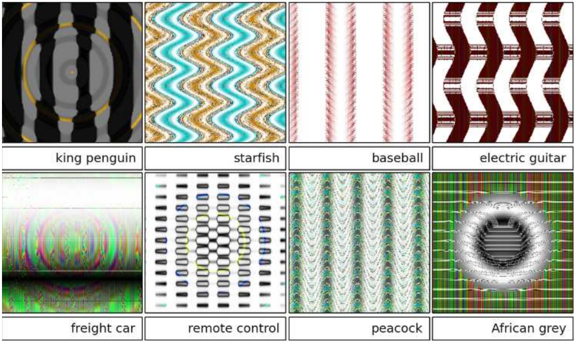
- model $f_{\vec{w}} : \mathbb{R}^n \rightarrow \mathbb{R}^m$
- error func.: $E(\vec{w}) = \sum_{i=1}^{N} e(f_{\vec{w}}(x_i), y_i) = \sum_{i=1}^{N} (f_{\vec{w}}(x_i) - y_i)^2$
- learning: $\min\limits_{\vec{w}} E(\vec{w})$

Finding adversarial example

- $\vec{w}$ is fixed, $\vec{x}$ is optimized
- minimize $||r||_2$ subject to $f(x + r) = l$ and $(x + r) \in [0, 1]^m$
- a box-constrained L-BFGS

Works on adversarial examples II.

- *Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images*
  2015,Anh Nguyen, Jason Yosinski, Jeff Clune

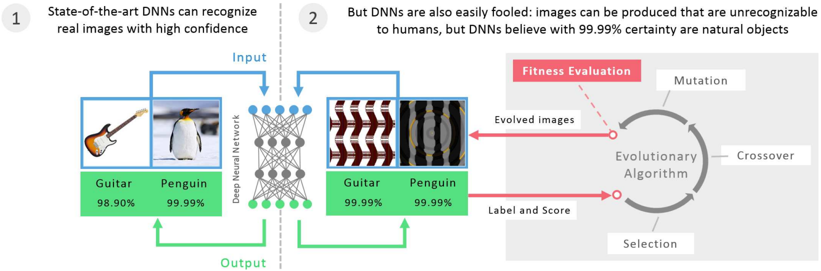


- evolutionary generated images

Works on adversarial examples II.

Compositional pattern-producing network (CPPN)

- similar structure to neural networks
- takes (x, y) as an input, outputs pixel value
- nodes: sin, sigmoid, Gaussian, and linear



Works on adversarial examples III.

- *Explaining and Harnessing Adversarial Examples* 2015,Goodfellow et al.
- linear behaviour in high dimensional spaces is sufficent to cause adversarial examples

$$\tilde{x} = x + \eta$$

$$x, \tilde{x} \text{ belong to the same class if } ||\eta||_\infty < \epsilon$$

$$w^T \tilde{x} = w^T x + w^T \eta$$

$$\text{for } \eta = \epsilon sign(w) \text{ activation increases } \epsilon mn$$

$||\eta||_\infty$ does not grow with dimensionality, but ϵmn does

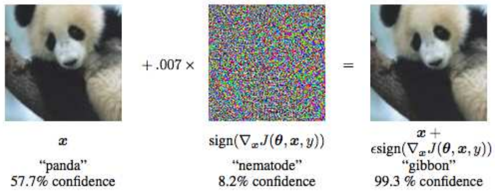
- in large dimensions small changes of the input cause large change to the output

Works on adversarial examples III.

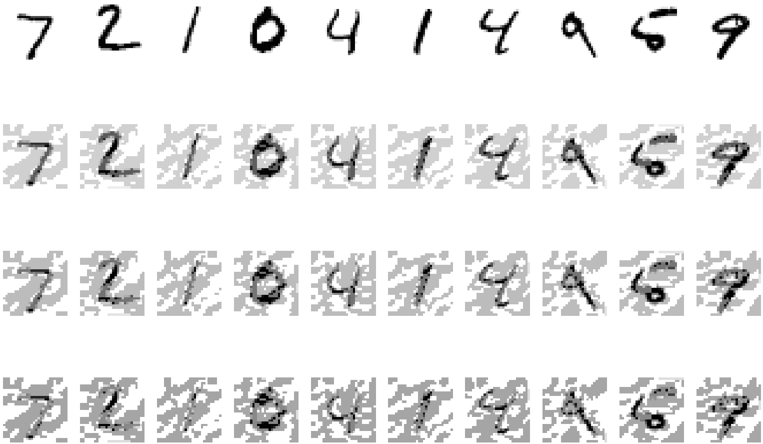
- nonlinear models: parameters θ, input x, target y, cost function $J(\theta, x, y)$
- we can linearize the cost function around θ and obtain optimal perturbation

$$\eta = \epsilon sign(\nabla_x J(\theta, x, y))$$

- adding small vector in the direction of the sign of the derivation – *fast gradient sign method*



Fast Gradient Sign Method on MNIST



Original test examples and corresponding adversarial examples crafted by FGSM with ϵ 0.2, 0.3, and 0.4.

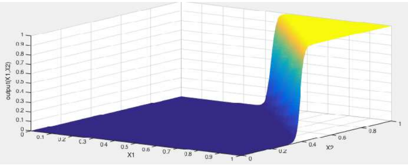
Works on adversarial examples IV.

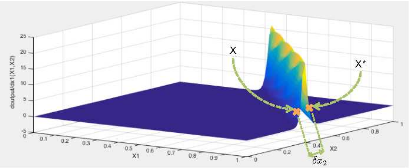
- *The Limitations of Deep Learning for Adversarial Settings* 2016, Papernot et al.

$$\arg\min_{\delta_{\mathbf{X}}} ||\delta_{\mathbf{X}}|| \text{ such that } F(\mathbf{X} + \delta_{\mathbf{X}}) = \mathbf{Y}^*$$

- *adversarial saliency maps* - identify features of the input that most significantly impact output classifications
- Motivation:

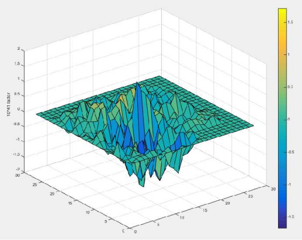
  $F(\mathbf{X}) = x_1 \text{and} x_2,$ $\qquad \frac{\delta F(\mathbf{X})}{\delta x_2}$ (forward derivative)





Works on adversarial examples IV.

Saliency Map



- misclasify $\mathbf{X}$ such that it is assigned a target class $t \neq label(\mathbf{X})$
- $F_t(\mathbf{X})$ must increase, while $F_j(\mathbf{X}), j \neq t$ decrease

$$S(\mathbf{X}, t)[i] = \begin{cases} 0 \text{ if } \frac{\delta F_t(\mathbf{X})}{\delta \mathbf{X_i}} < 0 \text{ or } \sum_{j \neq t} \frac{\delta F_j(\mathbf{X})}{\delta \mathbf{X_i}} > 0 \\ \left(\frac{\delta F_t(\mathbf{X})}{\delta \mathbf{X_i}}\right) \left| \sum_{j \neq t} \frac{\delta F_j(\mathbf{X})}{\delta \mathbf{X_i}} \right| \text{ otherwise} \end{cases}$$

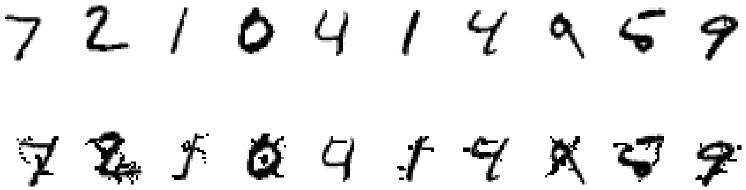
Works on adversarial examples IV.

Crafting algorithm based on Saliency Map

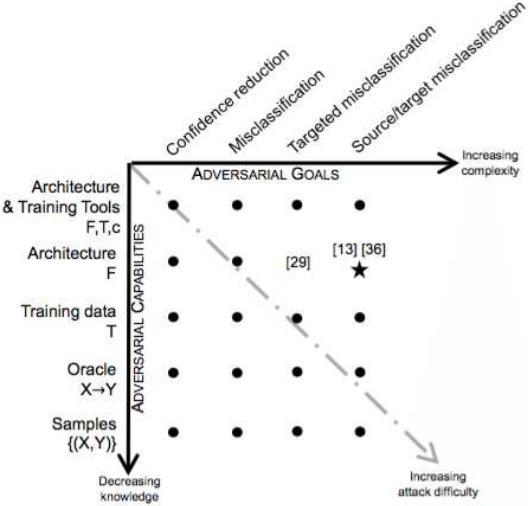
Input: $\mathbf{X}, \mathbf{Y}^*, F, \Upsilon$(maximal distortion), θ(change)

1. $\mathbf{X}^* \leftarrow \mathbf{X}$
2. $\Gamma = \{1 \ldots |\mathbf{X}|\}$
3. **while** $F(\mathbf{X}^*) \neq \mathbf{Y}^*$ and $||\delta_{\mathbf{X}}|| < \Upsilon$ **do**
4. Compute forward derivative $\nabla F(\mathbf{X}^*)$
5. $S \leftarrow saliency\_map(\nabla F(\mathbf{X}^*), \Gamma, \mathbf{Y}^*)$
6. modify $\mathbf{X}^*_{i_{max}}$ by θ, $i_{max} = \arg\max_i S(\mathbf{X}, \mathbf{Y}^*)[i]$
7. $\delta_{\mathbf{X}} \leftarrow \mathbf{X}^* - \mathbf{X}$
8. **end while**
9. **return** $\mathbf{X}^*$

Saliency Map Method on MNIST



Taxonomy of Threat Models in Deep Learning



from Papernot, et al. *The Limitations of Deep Learning in Adversarial Settings.*

Works on Adversarial Examples V.

- *Practical Black-Box Attacks against Deep Learning Systems using Adversarial Examples*
  2016, Papernot et al.
- black-box - adversaries need not know internal details of a system to compromise it
- train a local substitute DNN with a **synthetic** dataset
- The algorithm:
  1. create initial collection of data samples S_0
  2. select architecture for the substitute model
  3. substitute model training
     - labeling
     - training
     - augmentation: $S_{\rho+1} = \{\vec{x} + \lambda sgn(J_F[O(\vec{x})]) : \vec{x} \in S_\rho\} \cup S_\rho$
  4. use the substitute model to craft adversarial samples

Our work

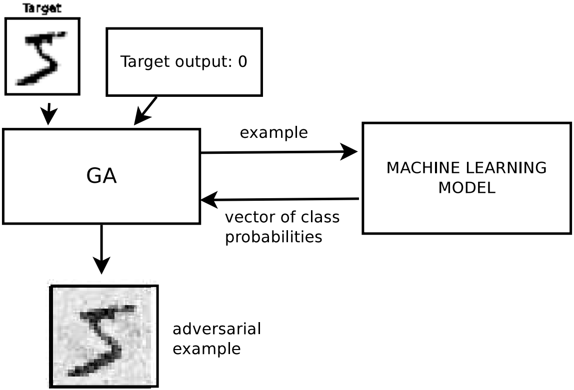
- genetic algorihms used to search for adversarial examples
- tested various machine learning models including both deep and shallow architectures
- Vidnerová, Neruda. *Vulnerability of Machine Learning Models to Adversarial Examples.* 2016
- Vidnerová, Neruda. *Evolutionary Generation of Adversarial Examples for Deep and Shallow Machine Learning Models.* 2016

Search for adversarial images

- To obtain an adversarial example for the trained machine learning model, we need to *optimize the input image with respect to model output*.
- For this task we employ a GA – robust optimisation method working with the whole population of feasible solutions.
- The population evolves using operators of selection, mutation, and crossover.
- The machine learning model and the target output are fixed.

Black box approach

- genetic algorithms to generate adversarial examples
- machine learning method is a blackbox
- applicable to all methods without the need to acess models parameters (weights)



Genetic algorithm

- *Individual:* image encoded as a vector of pixel values:

$$I = \{i_1, i_2, \ldots, i_N\},$$

  where $i_i \in < 0, 1 >$ are levels of grey and N is a size of flatten image.
- *Crossover:* operator performs a two-point crossover.
- *Mutation:* with the probability p_{mutate_pixel} each pixel is changed:

$$i_i = i_i + r,$$

  where r is drawn from Gaussian distribution.
- *Selection:* $3-$tournament

GA fitness

- The fitness function should reflect the following two criteria:
  - the individual should resemble the target image
  - if we evaluate the individual by our machine learning model, we would like to obtain a target output (i.e. misclassify it).

Thus, in our case, a fitness function is defined as:

$$f(I) = -(\quad 0.5 * cdist(I, target\_image) \tag{1}$$

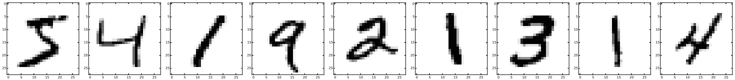
$$+ \quad 0.5 * cdist(model(I), target\_answer)), \tag{2}$$

where *cdist* is an Euclidean distance.

Dataset for our experiments

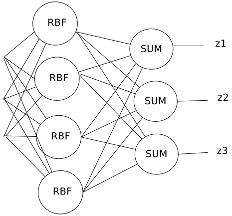
MNIST dataset

- 70000 images of handwritten digits
- 28×28 pixels
- 60000 for training, 10000 for testing



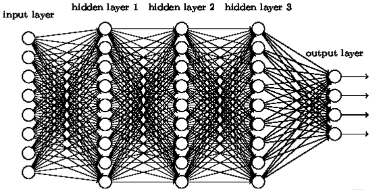
Machine learning models overview

- Shallow architectures
  - SVM — support vector machine
  - RBF — RBF network



- DT — decision tree

- Deep architectures
  - MLP — multilayer perceptron network
  - CNN — convolutional network



Support Vector Machines (SVM)

- popular kernel method
- learning based on searching for a separating hyperplane with highest margin
- one hidden layer of kernel units, linear output layer

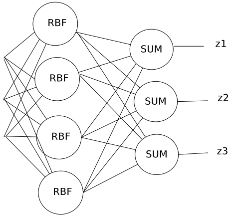
Kernels used in experiments:

- linear $\langle x, x' \rangle$
- polynomial $(\gamma \langle x, x' \rangle + r)^d$, grade 2 and 4
- Gaussian $\exp(-\gamma |x - x'|^2)$
- sigmoid $\tanh(\gamma \langle x, x' \rangle + r)$.

Implementation: SCIKIT-learn library

RBF network

- feedforward network with one hidden layer, linear output layer
- local units (typically Gaussian functions)



- our own implementation
- 1000 Gaussian units

Decision Tree (DT)

- a non-parametric supervised learning method

Implementation: SCIKIT-learn

Deep neural networks

- feedforward neural networks with multiple hidden layers between the input and output layer

Multilayer perceptrons (MLP)

- Perceptron units with *sigmoid* function
- Rectified linear unit (ReLU): $y(z) = max(0, z)$.

Implementation:

- KERAS library
- MLP — three fully connected layers, two hidden layers have 512 ReLUs each, using dropout; the output layer has 10 softmax units.

Convolutional Networks (CNN)

- *Convolutional units* perform a simple discrete convolution operation which for 2-D data can be represented by a matrix multiplication.
- *max pooling layers* that perform an input reduction by selecting one of many inputs, typically the one with maximal value

Implementation:

- KERAS library
- CNN — two convolutional layers with 32 filters and ReLUs, each, max pooling layer, fully connected layer of 128 ReLUs, and a fully connected output softmax layer.

Baseline Classification Acurracy

model	trainset	testset
MLP	1.00	0.98
CNN	1.00	0.99
RBF	0.96	0.96
SVM-rbf	0.99	0.98
SVM-poly2	1.00	0.98
SVM-poly4	0.99	0.98
SVM-sigmoid	0.87	0.88
SVM-linear	0.95	0.94
DT	1.00	0.87

Experimental Setup

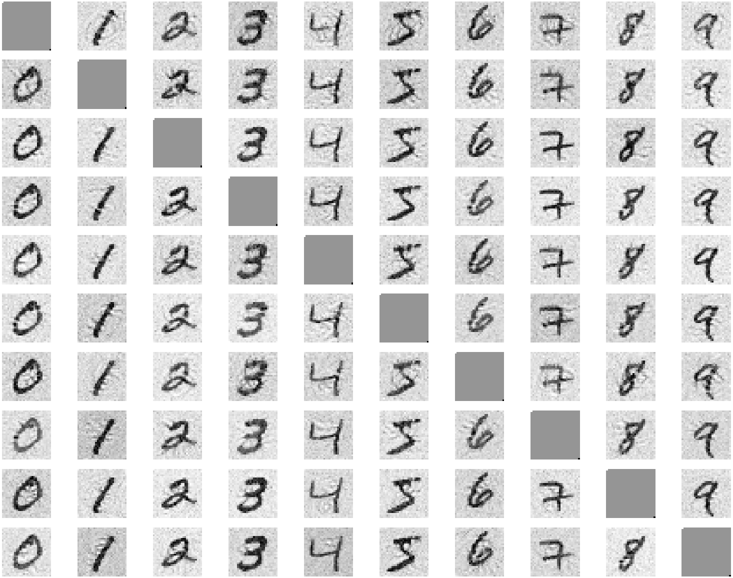
GA setup

- population of 50 individuals
- 10 000 generations
- crossover probability 0.6
- mutation probability 0.1
- DEAP framework

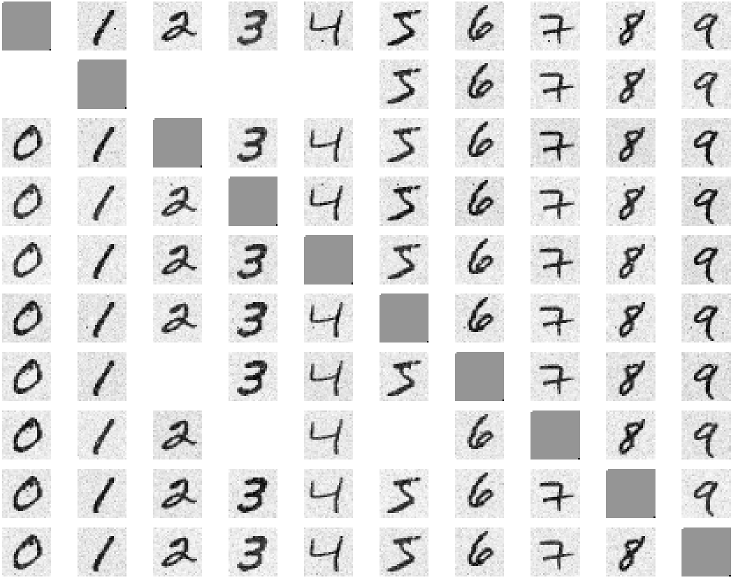
Images

- for 10 images from training set (one representant for each class)
- target: classify as zero, one, . . . , nine

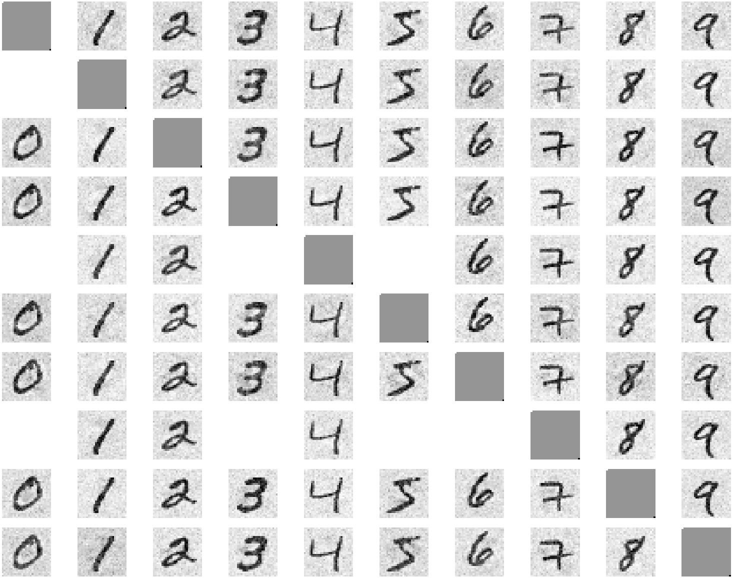
Evolved Adversarial Examples – CNN (90/90)



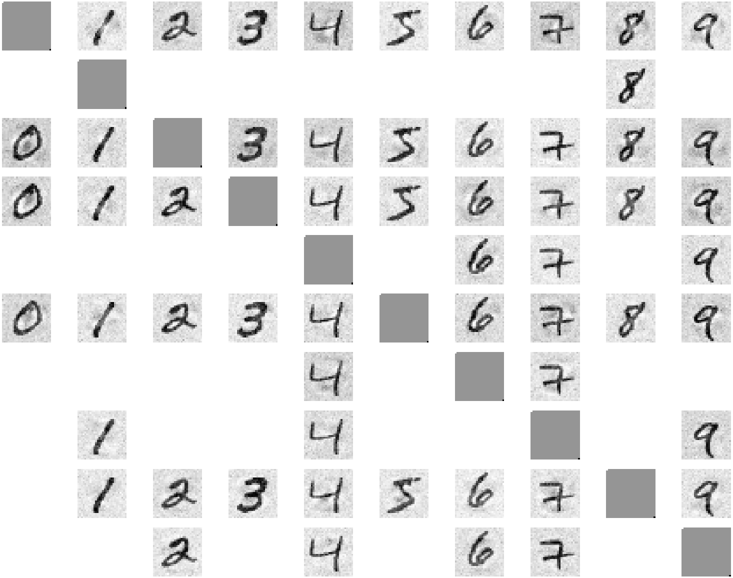
Evolved Adversarial Examples – DT (83/90)



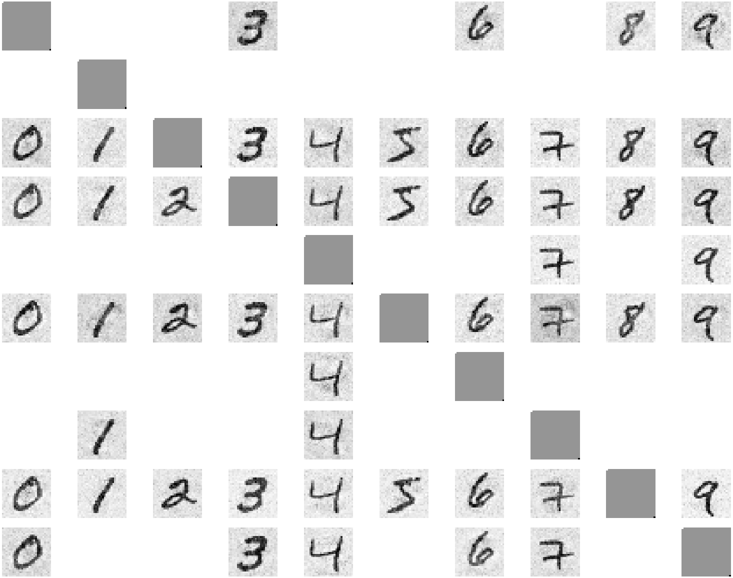
Evolved Adversarial Examples – MLP (82/90)



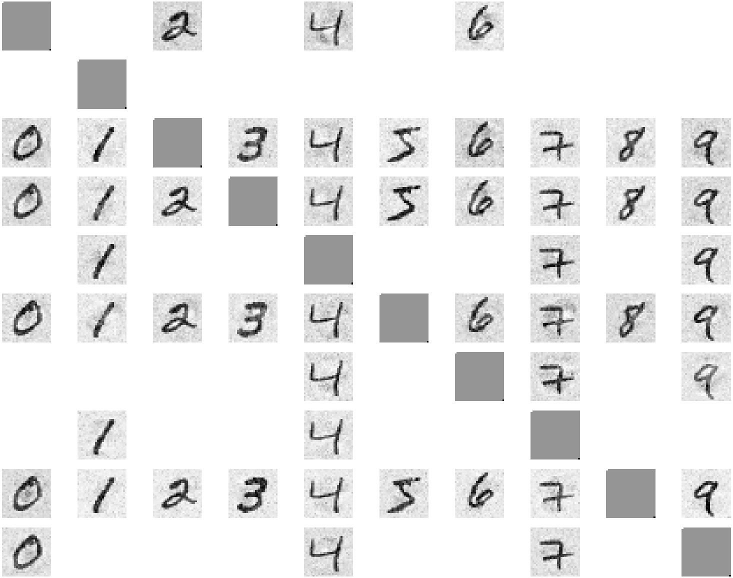
Evolved Adversarial Examples – SVM sigmoid (57/90)



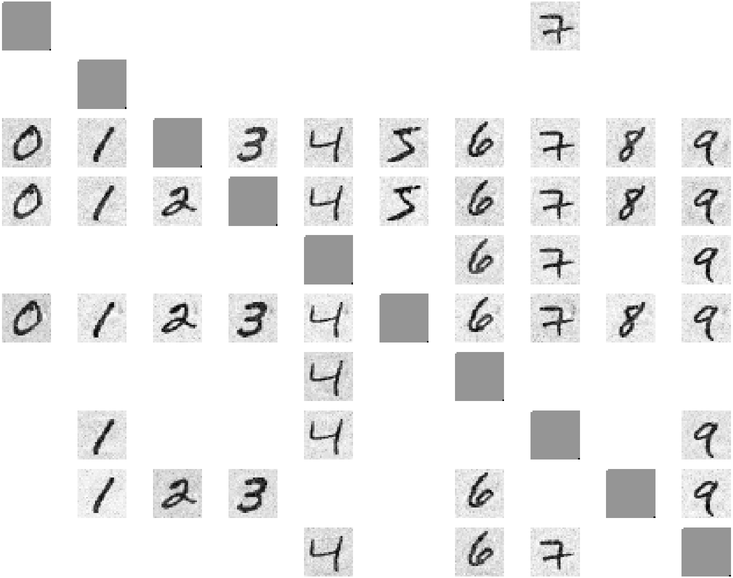
Evolved Adversarial Examples – SVM poly (50/90)



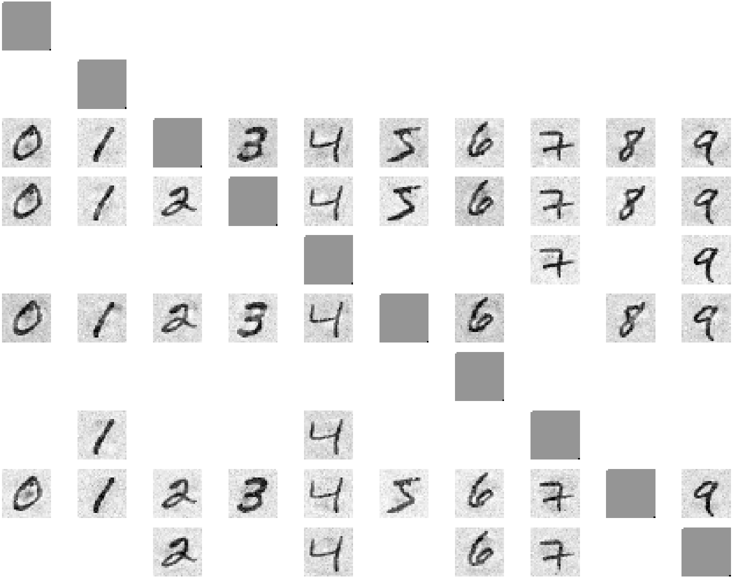
Evolved Adversarial Examples – SVM poly4 (50/90)



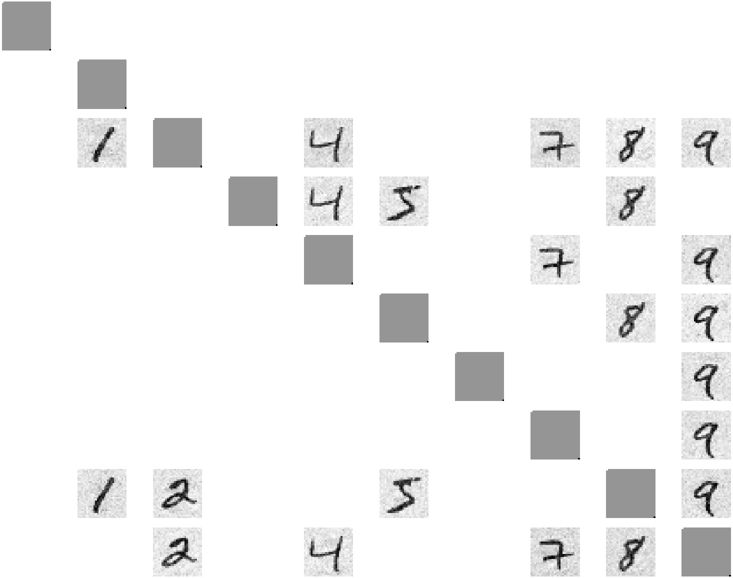
Evolved Adversarial Examples – SVM linear (43/90)



Evolved Adversarial Examples – SVM rbf (43/90)



Evolved Adversarial Examples – RBF (22/90)

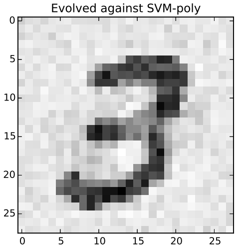


Experimental Results

- CNN, MLP, and DT were fooled in all or almost all cases
- RBF network was the most resistant model, but in 22 cases it was fooled too
- from SVMs the most vulnerable is SVM_sigmoid, most resistant is SVM_rbf and SVM_linear

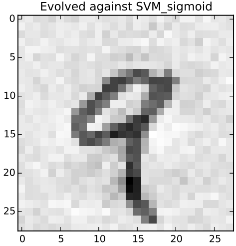
Generalization

- some adversarial examples generated for one model are also missclassified by other models



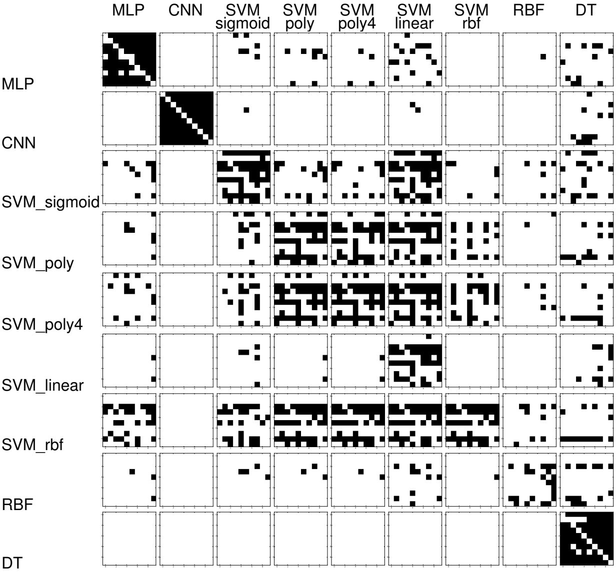
	0	1	2	3	4	5	6	7	8	9
RBF	0.32	0.02	0.17	**0.86**	-0.01	-0.09	-0.09	-0.03	-0.12	0.01
MLP	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00
CNN	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00
ENS	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00
SVM-rbf	0.00	0.00	0.00	**0.99**	0.00	0.00	0.00	0.00	0.00	0.00
SVM-poly	**0.87**	0.00	0.02	0.04	0.00	0.00	0.00	0.00	0.04	0.02
SVM-poly4	**0.38**	0.01	0.11	0.23	0.01	0.02	0.01	0.02	0.15	0.04
SVM-sigmoid	**0.55**	0.01	0.04	0.19	0.01	0.05	0.01	0.01	0.13	0.02
SVM-linear	**0.71**	0.01	0.02	0.06	0.01	0.02	0.01	0.01	0.15	0.01
DT	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00

Generalization



	0	1	2	3	4	5	6	7	8	9
CNN	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**1.00**
MLP	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**1.00**	0.00
SVM_sigmoid	0.00	0.01	0.00	0.00	0.01	0.01	0.00	0.00	**0.85**	0.11
SVM_rbf	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.98**	0.01
SVM_poly	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	**0.98**	0.02
SVM_poly4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.98**	0.01
SVM_linear	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**1.00**	0.00
RBF	0.01	0.01	0.09	0.09	-0.10	0.06	0.07	-0.02	**0.44**	0.41
DT	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Generalization Summary



Generalization — Summary

- adversarial example evolved for CNN was misclassified by other models only in few cases, and CNN never misclassified other adversarial examples than those evolved for the CNN;
- adversarial example evolved for DT was never misclassified by other models, however DT sometimes misclassifies the adversarial examples evolved for other models
- adversarial examples are often shared between various SMVs

Adversarial vs. noisy data

- We tried to learn a classifier to distinguish between adversarial examples and examples that are only noisy.

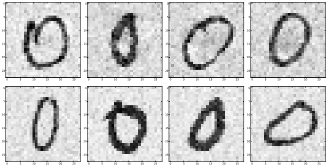


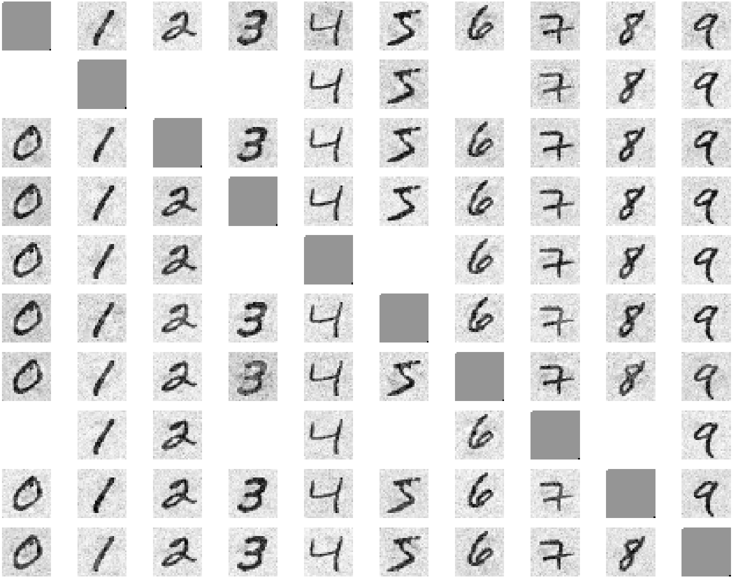
Figure: Digit zero —adversarials examples (top), noisy examples (bottom). Noisy examples were classified as zero by the MLP, adversarial examples as other class.

Adversarial vs. noisy data: results

- The data contains 22500 noisy examples and 19901 adversarial examples, and are randomly divided to training and test data (20% for test).

	precision	recall
SVM-rbf	0.888	0.843
MLP	0.923	0.912
CNN	0.964	0.925

New adversarial examples (for MLP)



Towards approaches robust to adversarial examples

- *Towards Deep Neural Network Architectures Robust To Adversarial Examples*.
  2015, Shixiang Gu, Luca Rigazio

- noise injection, Gaussian blur
- autoencoder
- deep contractive network

Gaussian blur of the input

- a recovery strategy based on additional corruption
- decrease error on adversarial data but not enough

Test error rates

	clean data			adversarial data		
blur kernel size	—	5	11	—	5	11
N100-100-10	1.8	2.6	11.3	99.9	43.5	62.8
N200-200-10	1.6	2.5	14.8	99.9	47.0	65.5
ConvNet	0.9	1.2	4.0	100	53.8	43.8

Autoencoder

- a three-hidden-layer autoencoder (784-256-128-256-784 neurons)
- trained to map adversarial examples back to the original data and original data back to itself
- autoencoders recover at least 90% of adversarial errors

	N-100-100-10	N200-200-10	ConvNet
N-100-100-10	2.3%	2.4%	5.2%
N-200-200-10	2.3%	2.2%	5.4%
ConvNet	7.7%	7.6%	2.6%

- drawback: autoencoder and classifier can be stacked to form a new feed-forward network, new adversarial examples can be generated

Deep Contractive Network

- layer-wise penalty approximately minimizing the network outputs variance with respect to perturbations in the inputs
- Deep Contractive Network (DNC) — generalization of the contractive autoencoder

$$J_{DNC}(\theta) = \sum_{i=1}^{m}(L(t^{(i)}, y^{(i)}) + \lambda|| \frac{\partial y^{(i)}}{\partial x^{(i)}} ||_2)$$

$$J_{DNC}(\theta) = \sum_{i=1}^{m}(L(t^{(i)}, y^{(i)}) + \sum_{j=1}^{H+1} \lambda_j || \frac{\partial h_j^{(i)}}{\partial h_{j-1}^{(i)}} ||_2)$$

Deep Contractive Network – Experimental Results

	DCN		original	
model	error	adv. distortion	error	adv. distortion
N100-100-10	2.3%	0.107	1.8%	0.084
N200-200-10	2.0%	0.102	1.6%	0.087
ConvNet	1.2%	0.106	0.9%	0.095

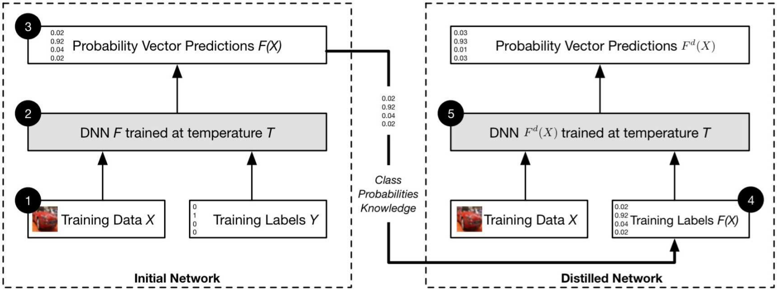
Defence to Adversarial Perturbations by Distillation

- *Distillation as a Defence to Adversarial Perturbations against Deep Neural Networks*, Papernot et al., 2016
- distillation – training procedure using knowledge transferred from a different DNN (originally to reduce computational complexity)
- used to improve resilience to adversarial samples

- distillation temperature – high temperature $\rightarrow$ probability vectors with large values for each class
- output softmax layer:

$$F(X) = \left[\frac{e^{z_i(X)/T}}{\sum_{l=0}^{N-1} e^{z_l(X)/T}} \right]_{i \in 0 \dots N-1}$$

Defence to Adversarial Perturbations by Distillation



	adversarial success rate	
	original DNN	distilled DNN
MNIST	95.89	1.34
CIFAR10	89.90	16.76

Denoising input samples

MLP	none	mean	gaussian	tv_chambolle	tv_bregman	bilateral	nl_means
legitimate	98.35	98.13	97.67	97.72	97.95	98.28	98.34
FGSM	2.87	3.83	6.94	8.11	6.25	4.52	5.34
Saliency Map	39.07	90.13	78.06	74.17	68.93	43.93	64.34

CNN	none	mean	gaussian	tv_chambolle	tv_bregman	bilateral	nl_means
legitimate	98.94	98.42	98.38	98.48	98.70	98.89	98.92
FGSM	14.80	21.52	23.17	25.65	22.53	17.86	18.54
Saliency Map	0.10	68.21	65.25	48.45	43.54	5.18	2.17

Denoising GA adversarial examples

	none	mean	gaussian	tv_chambolle	tv_bregman	bilateral	nl_means
MLP	98.49	98.27	98.14	98.16	98.26	98.41	98.44
MLP	0.00	68.15	70.77	81.35	84.41	88.04	93.74
CNN	98.77	98.20	98.35	98.50	98.59	98.71	98.75
CNN	0.00	14.29	28.57	21.43	21.43	21.43	14.29
DT	87.54	87.77	37.93	21.02	23.71	60.52	87.28
DT	0.00	27.93	17.06	12.32	15.17	20.63	17.07
SVM lin	94.87	94.41	93.96	94.14	94.40	94.78	94.85
SVM lin	0.00	7.58	42.53	69.47	54.53	62.95	67.16

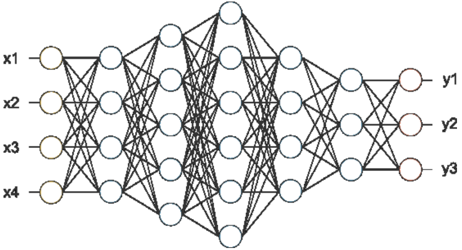
Denoising GA adversarial examples

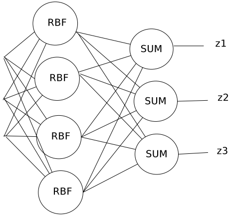
	none	mean	gaussian	tv_chambolle	tv_bregman	bilateral	nl_means
SVM poly	98,20	97,99	97,40	97,42	97,72	98,16	98,21
SVM poly	0,00	21,23	44,39	58,95	53,33	72,46	55,79
SVM poly4	98,35	97,98	97,06	97,07	97,54	98,20	98,28
SVM poly4	0,00	10,49	28,39	44,67	36,17	65,46	49,19
SVM rbf	98,57	98,33	96,52	96,90	97,52	98,39	98,53
SVM rbf	0,00	1,08	16,20	36,29	31,97	61,56	51,62
SVM sigmoid	89,11	88,81	89,84	89,62	89,94	89,28	89,17
SVM sigmoid	0,00	0,00	3,26	21,61	10,12	57,80	30,36

Deep Networks and RBF Networks

- combinations of Deep Networks and RBF Networks
- RBF layers can be also included in evolution

- RBF networks less vulnerable to adversarial examples
- Does add RBF layers to deep network help to prevent adversarial examples?





RBF Networks

- feed-forward neural networks with one hidden layer of RBF units
- local units alternative to MLP
- RBF unit:

$$y = \varphi(\xi); \quad \xi = \beta||\vec{x} - \vec{c}||^2$$

  where $\varphi : \mathbb{R} \to \mathbb{R}$ is suitable activation function, typically Gaussian $\varphi(z) = e^{-z^2}$.
- the network computes the function $\vec{f} : \mathbb{R}^n \to \mathbb{R}^m$:

$$f_s(\vec{x}) = \sum_{j=1}^{h} w_{js}\varphi\left(\frac{\| \vec{x} - c_j \|}{\beta_j}\right)$$

RBF Networks Learning

- wide range of methods

Three Step Learning

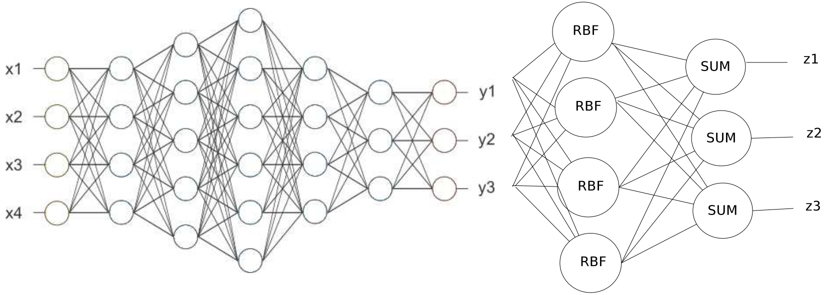
1. set the centers - approximate the distribution of training samples
   - random or uniform samples, various clustering methods
2. set the widths - cover the input space by unit's fields
   - heuristics (k-neighbours)
3. compute the output weights
   - linear system, pseudoinverse

Gradient Learning

- analogous to backpropagation for MLP

Proposed architecture DNNRBF

- stacking deep neural network and RBF network



DNNRBF learning

1. train the *DNN*
2. set the centers of *RBF* randomly, drawn from uniform distribution on $(0, 1.0)$
3. set the parameters β to the constant value
4. init the weights of RBF output layer to random small values
5. retrain the whole network DNNRBF (by back propagation)

Experiments

Architectures

- MLP
  - dense layer of 512 ReLU
  - dense layer of 512 ReLU
  - dense layer of 10 softmax units
- CNN
  - convolutional layer with 32 3x3 filters and ReLU activation
  - convolutional layer with 32 3x3 filters and ReLU activation
  - 2x2 max pooling layer
  - dense layer of 128 ReLU
  - dense layer of 10 softmax units

Experiments

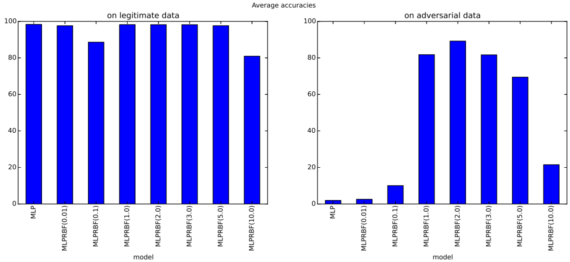
Implementation

- FGSM for crafting adversarial examples
  Cleverhans library: *cleverhans v2.0.0: an adversarial machine learning library, Nicolas Papernot, et al., arXiv preprint arXiv:1610.00768, 2017*
- Keras for MLP and CNN
  Keras, François Chollet, `https://github.com/fchollet/keras`*, 2015*
- our implementation of RBF Keras layers
  `http://github.com/PetraVidnerova/rbf_keras`

`http://github.com/PetraVidnerova/rbf_tests`

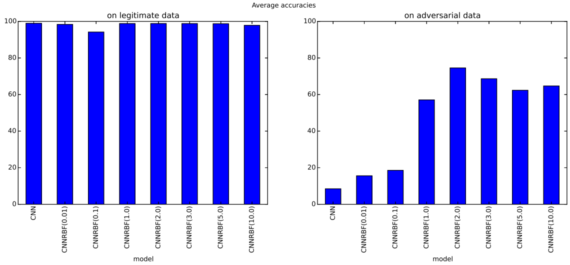
Experiments Results- MLP

	Legitimate samples				Adversarial samples			
model	mean	std	min	max	mean	std	min	max
MLP	**98.35**	**0.12**	**98.04**	**98.59**	**1.95**	**0.41**	**1.30**	**2.86**
MLPRBF(0.01)	97.62	2.43	88.44	98.65	2.56	2.09	1.16	10.71
MLPRBF(0.1)	88.61	8.56	69.91	98.36	10.04	6.45	1.71	23.10
MLPRBF(1.0)	98.23	0.10	98.08	98.48	81.77	7.84	64.18	94.06
MLPRBF(2.0)	**98.19**	**0.14**	**97.91**	**98.38**	**89.21**	**5.03**	**66.28**	**94.83**
MLPRBF(3.0)	98.18	0.14	97.88	98.45	81.66	4.38	70.13	87.23
MLPRBF(5.0)	97.64	2.09	89.34	98.36	69.47	13.26	13.01	81.95
MLPRBF(10.0)	80.94	11.82	58.57	98.33	21.49	16.32	2.48	65.11



Experiments Results - CNN

	Legitimate samples				Adversarial samples			
model	mean	std	min	max	mean	std	min	max
CNN	**98.97**	**0.07**	**98.84**	**99.13**	**8.49**	**3.52**	**3.11**	**16.43**
CNNRBF(0.01)	98.36	1.73	89.12	99.01	15.60	4.28	10.26	28.44
CNNRBF(0.1)	94.19	8.21	58.88	98.92	18.58	6.42	6.01	31.29
CNNRBF(1.0)	98.83	0.13	98.46	99.04	57.09	9.23	33.39	78.99
CNNRBF(2.0)	**98.85**	**0.13**	**98.38**	**99.09**	**74.57**	**7.69**	**53.07**	**84.67**
CNNRBF(3.0)	98.82	0.14	98.55	99.10	68.65	7.77	44.36	80.13
CNNRBF(5.0)	98.74	0.11	98.49	98.94	62.35	7.04	48.03	77.04
CNNRBF(10.0)	97.86	2.24	89.33	98.84	64.71	8.32	46.61	79.89



Experiments Results

model	Accuracy on adversarial data					
	$\epsilon = 0.2$		$\epsilon = 0.3$		$\epsilon = 0.4$	
	avg	std	avg	std	avg	std
CNN	33.85	7.58	8.49	3.52	4.34	1.71
CNNRBF	76.88	6.25	74.57	7.69	73.51	8.08
MLP	3.01	0.69	1.95	0.41	1.66	0.38
MLPRBF	90.14	4.82	89.21	5.03	88.27	5.14

Deep RBF Networks – *I don't know* scenario I.

- if maximal output < threshold answer I don't know
- threshold = 0.75

	legitimate data		
	correct	I don't know	wrong
baseline CNN	98.20	1.31	0.49
CNN + RBF	95.39	4.25	0.36

	adversarial data		
	correct	I don't know	wrong
baseline CNN	11.45	47.22	41.34
CNN + RBF	1.32	92.46	6.22

Deep RBF Networks – *I don't know* scenario II.

- threshold = 0.9

	legitimate data		
	correct	I don't know	wrong
baseline CNN	97.24	2.50	0.26
CNN + RBF	89.24	10.57	0.19

	adversarial data		
	correct	I don't know	wrong
baseline CNN	7.85	67.40	24.75
CNN + RBF	0.39	97.91	1.70

Summary

- We have proposed a GA for generating adversarial examples for machine learning models by applying minimal changes to the existing patterns.
- Our experiment showed that many machine models suffer from vulnerability to adversarial examples.
- Models with local units (RBF networks and SVMs with RBF kernels) are quite resistant to such behaviour.
- The adversarial examples evolved for one model are usually quite general – often misclassified also by other models.
- Defenses against adversarial examples are succsessfull only to some extend.

Thank you! Questions?