Fooling neural networks

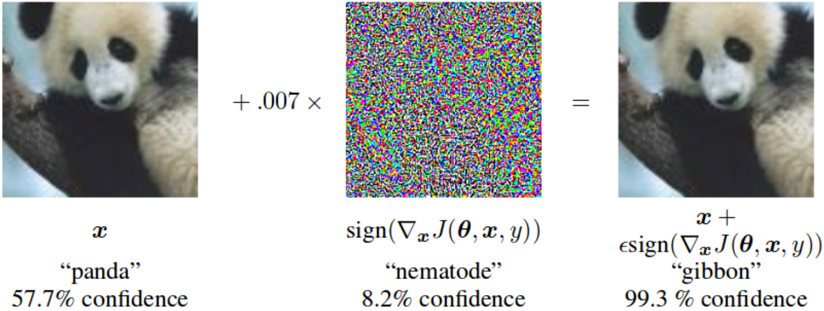
Berlin ML meetup | January 8th, 2018

Katharina Rasch | kat@krasch.io

Katharina Rasch
kat@krasch.io
`https://github.com/krasch`
`https://twitter.com/krasch_io`

PhD computer science
Previously: Data science / computer vision @ zalando
Now: Freelance data science + teaching

Today



Goodfellow et al., "Explaining and Harnessing Adversarial Examples", *ICLR*, 2015
(first identified in Szegedy et al., "Intriguing properties of neural networks", *arxiv*, 2013)

Today

- Digging into the Fast Gradient Sign Method
- Explanations?
- Defenses?
- Physical manifestations

Image classifier should tolerate small perturbations

Original image	x
Perturbation	η
Perturbed image	$\tilde{x} = x + \eta$

For small η expect classifier output $f(x) = f(\tilde{x})$

Goodfellow et al., "Explaining and Harnessing Adversarial Examples", *ICLR*, 2015

Image classifier should tolerate small perturbations

Original image x
Perturbation η
Perturbed image $\tilde{x} = x + \eta$

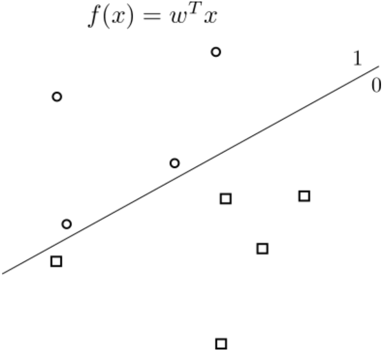
For small η expect classifier output $f(x) = f(\tilde{x})$

Every pixel should change at most $\pm\epsilon \rightarrow \|\eta\|_\infty \leq \epsilon$
$(\|\eta\|_\infty = max(|\eta|))$

for pixel values in [0,1]: $1/256 = 0.004$

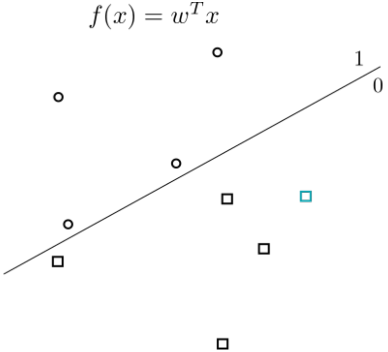
Goodfellow et al., "Explaining and Harnessing Adversarial Examples", *ICLR*, 2015

Generating adversarial examples for a linear classifier



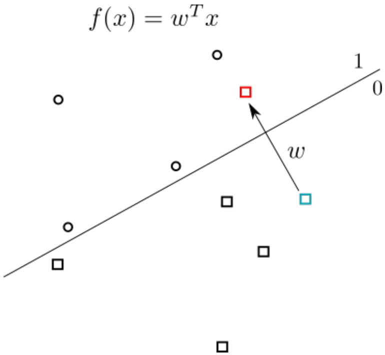
Based on Goodfellow et al., "Explaining and Harnessing Adversarial Examples", *ICLR*, 2015

Generating adversarial examples for a linear classifier



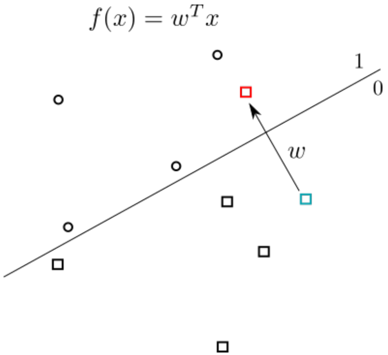
Based on Goodfellow et al., "Explaining and Harnessing Adversarial Examples", *ICLR*, 2015

Generating adversarial examples for a linear classifier



Based on Goodfellow et al., "Explaining and Harnessing Adversarial Examples", *ICLR*, 2015

Generating adversarial examples for a linear classifier

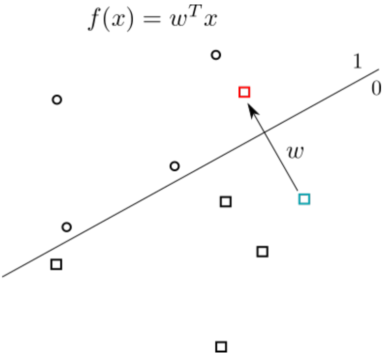


Propose

$$\eta = \epsilon * sign(w)$$

Based on Goodfellow et al., "Explaining and Harnessing Adversarial Examples", *ICLR*, 2015

Generating adversarial examples for a linear classifier



Propose

$$\eta = \epsilon * sign(w)$$

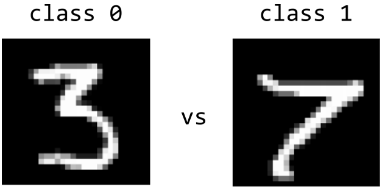
$$\rightarrow \eta = \begin{pmatrix} \pm\epsilon \\ \pm\epsilon \\ \pm\epsilon \\ \dots \end{pmatrix}$$

Based on Goodfellow et al., "Explaining and Harnessing Adversarial Examples", *ICLR*, 2015

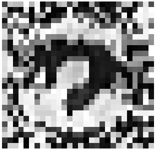
In code

```
pixel_min, pixel_max = 0.0, 1.0

def create_adversarial(image, original_class, epsilon):
    perturb = epsilon * np.sign(weights)
    if original_class == 0:
        adversarial = image + perturb
    else:
        adversarial = image - perturb
    return np.clip(adversarial, pixel_min, pixel_max)
```

Let's try it out!

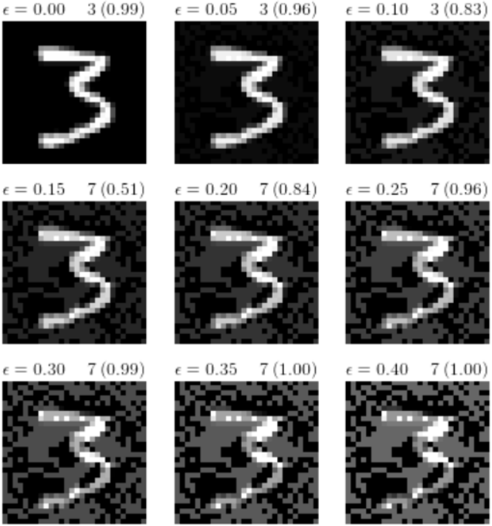


Trained logistic regression model with val accuracy ~ 97%

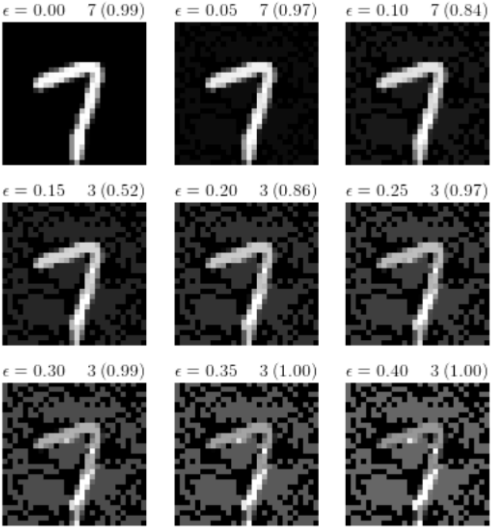


Visualisation of model weights

Turning 3 into 7



Turning 7 into 3



How about non-linear models?

Hypothesis: non-linear neural networks still designed to be very linear (for easier optimisation)
$\rightarrow$ won't be able to resist linear adversarial perturbation

Fast gradient sign method

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

(with original image x, original label y, model parameters θ, cost function $J(\theta, x, y)$)

Goodfellow et al., "Explaining and Harnessing Adversarial Examples", *ICLR*, 2015

In code

```python
def FGSM(image, original_class, epsilon):
    perturb = epsilon * np.sign(grad(image, original_class))
    adversarial = image + perturb
    return np.clip(adversarial, pixel_min, pixel_max)
```

In code

```
def FGSM(image, original_class, epsilon):
    perturb = epsilon * np.sign(grad(image, original_class))
    adversarial = image + perturb
    return np.clip(adversarial, pixel_min, pixel_max)

def targeted_FGSM(image, target_class, epsilon):
    perturb = epsilon * np.sign(grad(image, target_class))
    adversarial = image - perturb
    return np.clip(adversarial, pixel_min, pixel_max)
```

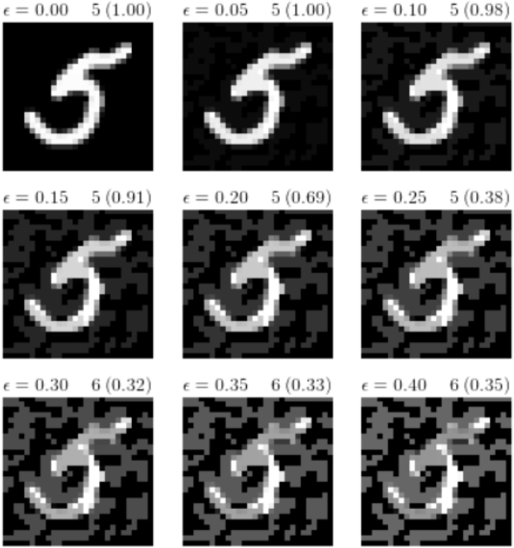
In code

```
def FGSM(image, original_class, epsilon):
    perturb = epsilon * np.sign(grad(image, original_class))
    adversarial = image + perturb
    return np.clip(adversarial, pixel_min, pixel_max)

def targeted_FGSM(image, target_class, epsilon):
    perturb = epsilon * np.sign(grad(image, target_class))
    adversarial = image - perturb
    return np.clip(adversarial, pixel_min, pixel_max)

def iterative_FGSM(image, original_class, epsilon, steps):
    epsilon = epsilon / steps
    adversarial = image
    for i in range(steps):
        adversarial = FGSM(adversarial, original_class, epsil
    return adversarial
```

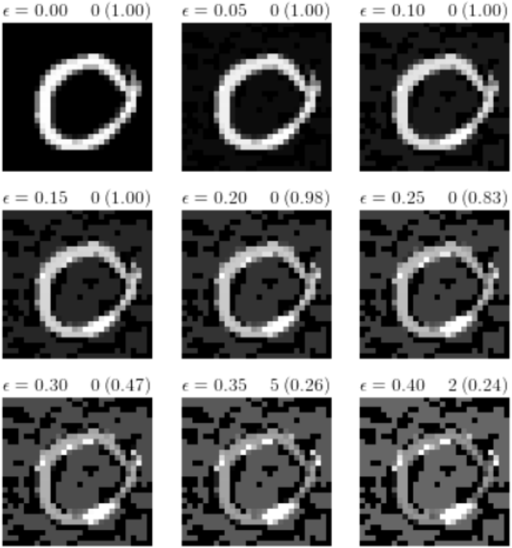
Let's try it out!

- MNIST: CNN with 2 convolutional + 2 fully connected layers
  - Validation accuracy $\sim 98\%$
- ImageNet: Pre-trained MobileNet
  - Slightly worse accuracy than VGG16 but much fewer parameters

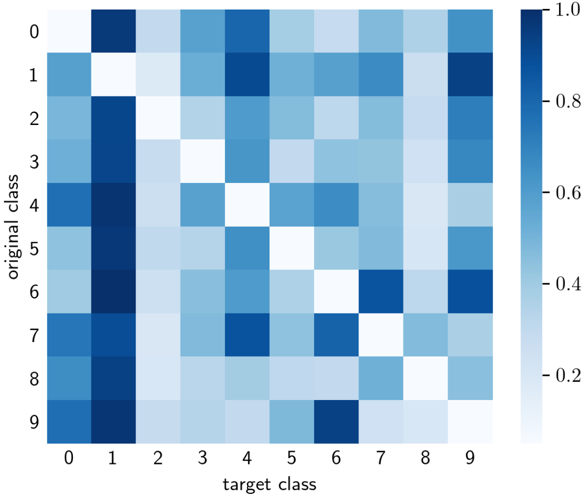
Targeted attack: turning 5 into 6



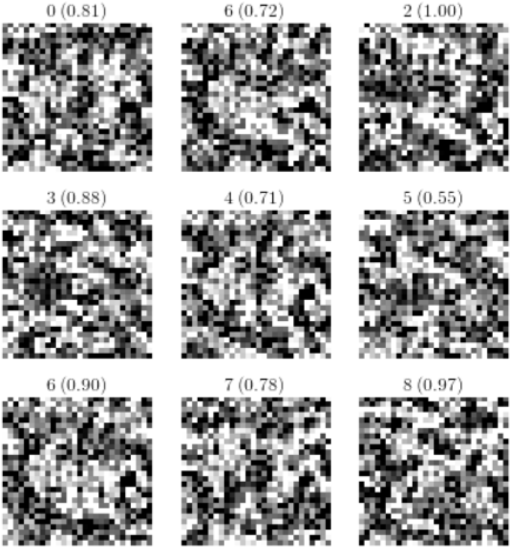
Targeted attack: turning 0 into 1



How large epsilon is needed for attack to succeed?

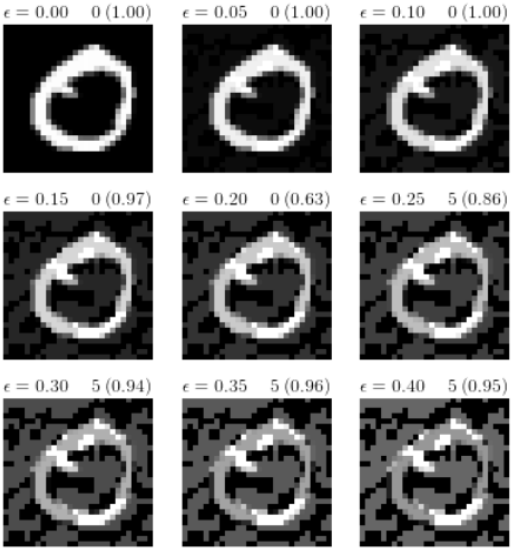


Targeted attack: turning rubbish into numbers

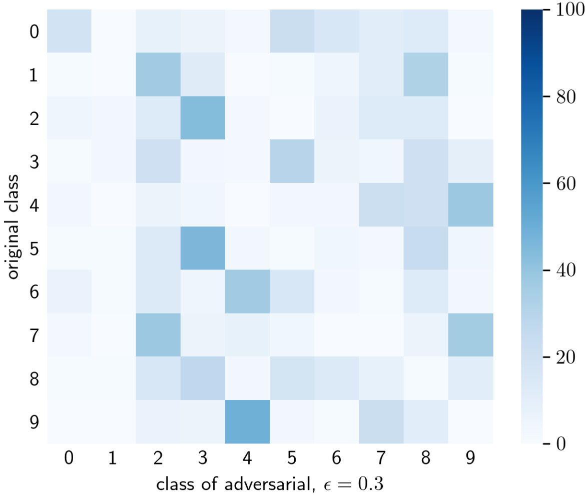



```
rubbish = np.random.uniform(low=0.0, high=1.0, size=(28*28))
```

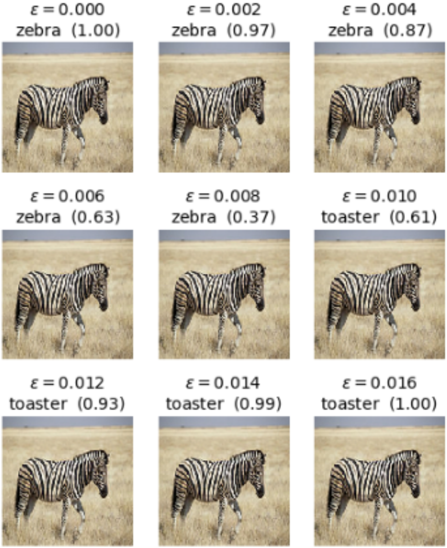
Untargeted attack: turning 0 into something



Where do untargeted attacks lead?



Iterative targeted attack: turning a zebra into a toaster



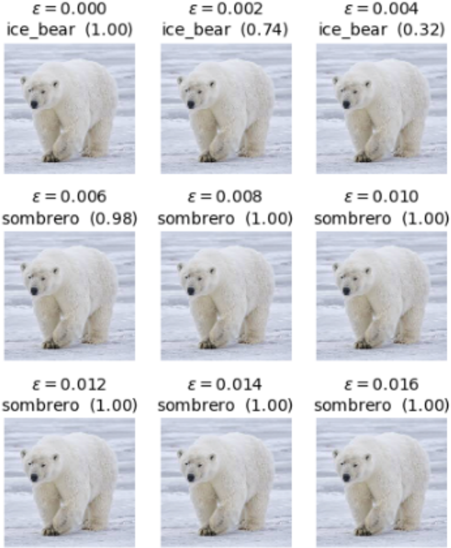
Zebra source: Rui Ornelas, `https://www.flickr.com/photos/fotos_dos_ornelas`

Iterative targeted attack: turning a zebra into a toaster

Original

Adversarial with $\epsilon = 0.012$

Iterative targeted attack: turning a polar bear into a sombrero

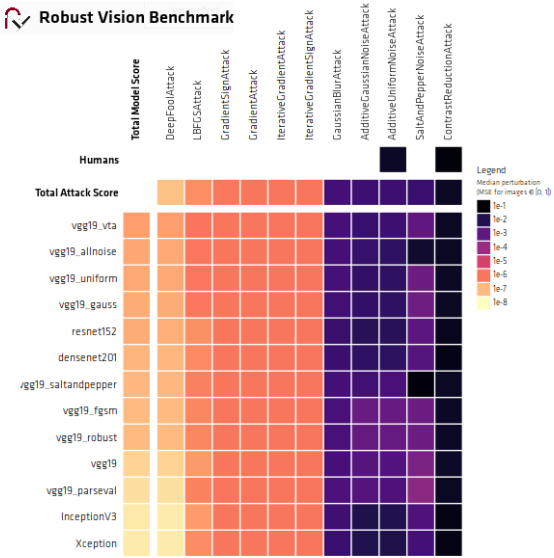


Polar bear source: Alan Wilson, `http://www.naturespicsonline.com`

FGSM is of course not the only method

In particular, have a look at: Moosavi-Dezfooli et al., "DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks", *CVPR*, 2016

Many different ones are implemented in
`https://github.com/bethgelab/foolbox`



https://robust.vision/benchmark/leaderboard/

Adversarial examples are transferable -> enables black box attacks

	RMSD	ResNet-152	ResNet-101	ResNet-50	VGG-16	GoogLeNet
-ResNet-152	30.68	38%	76%	70%	97%	76%
-ResNet-101	30.76	75%	43%	69%	98%	73%
-ResNet-50	30.26	84%	81%	46%	99%	77%
-VGG-16	31.13	74%	78%	68%	24%	63%
-GoogLeNet	29.70	90%	87%	83%	99%	11%

Table 3: The matching rate of targeted adversarial images generated using the optimization-based approach. The first column indicates the average RMSD of the generated adversarial images. Cell (i, j) indicates that percentage of the targeted adversarial images generated for the ensemble of the four models except model i (row) is predicted as the target label by model j (column). In each row, the minus sign "−" indicates that the model of the row is not used when generating the attacks.

Liu et al., "Delving into Transferable Adversarial Examples and Black-box Attacks", *ICLR*, 2017

Generating universal perturbations

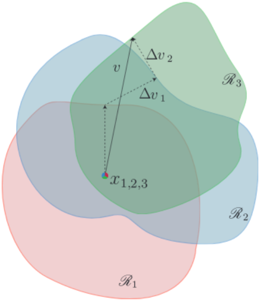


Figure 2: Schematic representation of the proposed algorithm used to compute universal perturbations. In this illustration, data points x_1, x_2 and x_3 are super-imposed, and the classification regions $\mathscr{R}_i$ (i.e., regions of constant estimated label) are shown in different colors. Our algorithm proceeds by aggregating sequentially the minimal perturbations sending the current perturbed points $x_i + v$ outside of the corresponding classification region $\mathscr{R}_i$.

Moosavi-Dezfooli et al., "Universal adversarial perturbations", *CVPR*, 2017

Universal perturbations are trippy

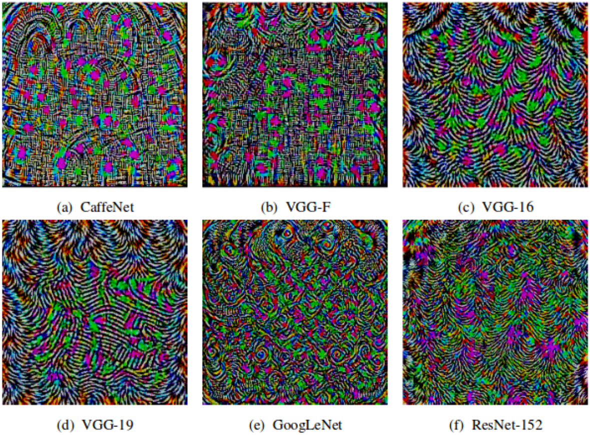


Figure 4: Universal perturbations computed for different deep neural network architectures. Images generated with $p = \infty$, $\xi = 10$. The pixel values are scaled for visibility.

Moosavi-Dezfooli et al., "Universal adversarial perturbations", *CVPR*, 2017

Universal perturbations are also transferable

	VGG-F	CaffeNet	GoogLeNet	VGG-16	VGG-19	ResNet-152
VGG-F	**93.7%**	71.8%	48.4%	42.1%	42.1%	47.4 %
CaffeNet	74.0%	**93.3%**	47.7%	39.9%	39.9%	48.0%
GoogLeNet	46.2%	43.8%	**78.9%**	39.2%	39.8%	45.5%
VGG-16	63.4%	55.8%	56.5%	**78.3%**	73.1%	63.4%
VGG-19	64.0%	57.2%	53.6%	73.5%	**77.8%**	58.0%
ResNet-152	46.3%	46.3%	50.5%	47.0%	45.5%	**84.0%**

Table 2: Generalizability of the universal perturbations across different networks. The percentages indicate the fooling rates. The rows indicate the architecture for which the universal perturbations is computed, and the columns indicate the architecture for which the fooling rate is reported.

Moosavi-Dezfooli et al., "Universal adversarial perturbations", *CVPR*, 2017

Universal attacks lead to common classes

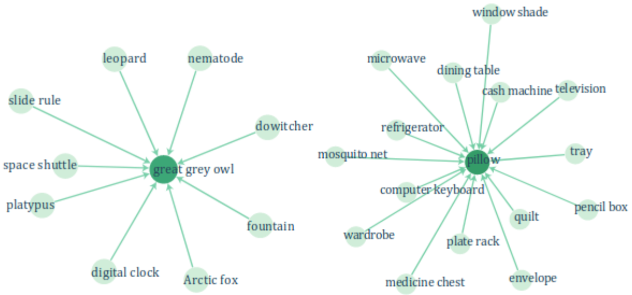


Figure 7: Two connected components of the graph $G = (V, E)$, where the vertices are the set of labels, and directed edges $i \to j$ indicate that most images of class i are fooled into class j.

Moosavi-Dezfooli et al., "Universal adversarial perturbations", *CVPR*, 2017

Explanations?

Voices against the "Networks are too linear" explanation

Tanay et al., "A Boundary Tilting Persepective on the Phenomenon of Adversarial Examples", *arxiv only*, 2016

Sabour et al., "Adversarial manipulation of deep representations", *ICLR*, 2016

Adversarial examples live in pockets in the input space

"To escape the adversarial pockets completely we have to add a noise considerably stronger than the original distortion used to reach them in the first place: adversarial regions are not isolated."

Tabacof et al., "Exploring the space of adversarial images", *IJCNN*, 2016

Other types of models are susceptible as well (results on MNIST)

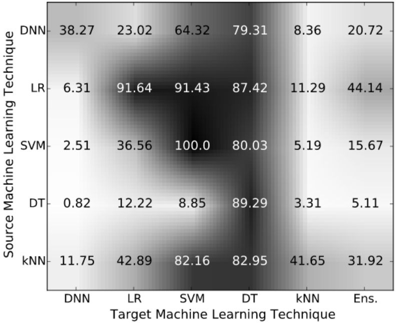


Figure 3: cross-technique Transferability matrix: cell (i, j) is the percentage of adversarial samples crafted to mislead a classifier learned using machine learning technique i that are misclassified by a classifier trained with technique j.

Papernot et al., "Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples", *arvix only*, 2016

Defenses?

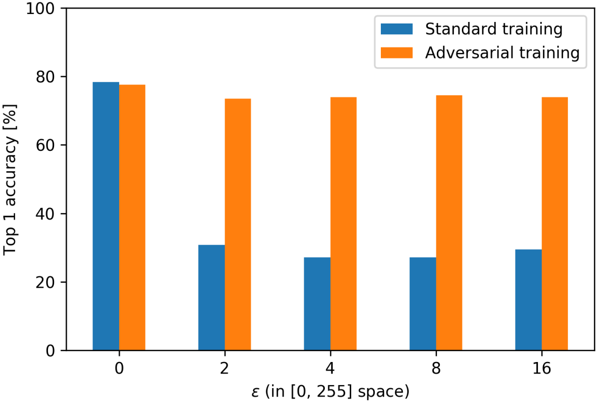
Augment training data with adversarial examples

Algorithm 1 Adversarial training of network N.
Size of the training minibatch is m. Number of adversarial images in the minibatch is k.

Randomly initialize network N
repeat
    Read minibatch $B = \{X^1, \ldots, X^m\}$ from training set
    Generate k adversarial examples $\{X^1_{adv}, \ldots, X^k_{adv}\}$ from corresponding clean examples $\{X^1, \ldots, X^k\}$ using current state of the network N
    Make new minibatch $B' = \{X^1_{adv}, \ldots, X^k_{adv}, X^{k+1}, \ldots, X^m\}$
    Do one training step of network N using minibatch B'
until training converged

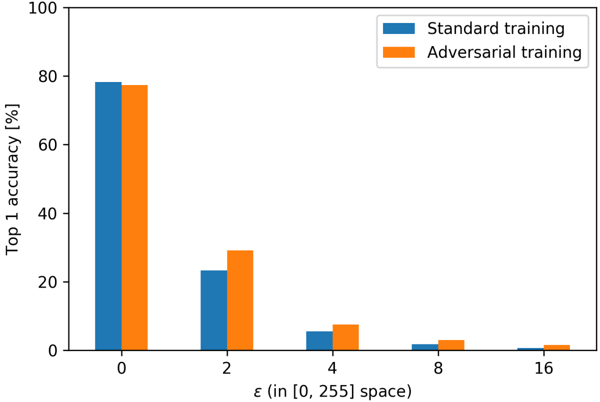
Szegedy et al., "Intriguing properties of neural networks", *arxiv*, 2013
Kurakin et al., "Adversarial Machine Learning at Scale", *ICLR*, 2017

Adversarial training protects against one-step methods (e.g. FGSM)



Plotting results from Kurakin et al., "Adversarial Machine Learning at Scale", *ICLR*, 2017

Adversarial training fails against iterative methods (e.g. Iterative Least Likely FGSM)



Plotting results from Kurakin et al., "Adversarial Machine Learning at Scale", *ICLR*, 2017

JPG compression destroys smaller perturbations

$\epsilon = 0.012 + JPG \rightarrow$ 'zebra'

$\epsilon = 0.1 + JPG \rightarrow$ 'toaster'

Dziugaite et al., "A study of the effect of JPG compression on adversarial images", *arxiv only*, 2016

Detecting adversarial examples

Many methods proposed, e.g. using detector network before classification network

...

Carlini et al., "Adversarial examples are not easily detected: Bypassing ten detection methods.", *AISEC*, 2017

Detecting adversarial examples

Many methods proposed, e.g. using detector network before classification network

...

Carlini et al., "Adversarial examples are not easily detected: Bypassing ten detection methods.", *AISEC*, 2017
(also "offer a note of caution about evaluating solely on MNIST; it appears that MNIST has somewhat different security properties than CIFAR")

Physical manifestations

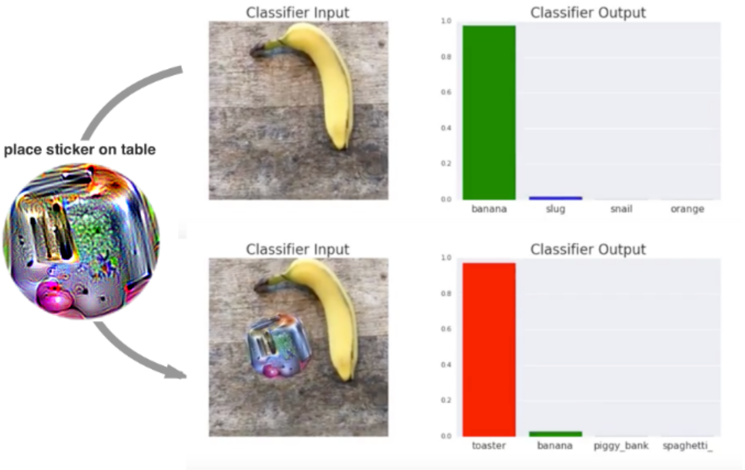
Impersonator glasses

(a) (b) (c) (d)

Figure 4: Examples of successful impersonation and dodging attacks. Fig. (a) shows S_A (top) and S_B (bottom) dodging against DNN_B. Fig. (b)–(d) show impersonations. Impersonators carrying out the attack are shown in the top row and corresponding impersonation targets in the bottom row. Fig. (b) shows S_A impersonating Milla Jovovich (by Georges Biard / CC BY-SA / cropped from https://goo.gl/GlsWlC); (c) S_B impersonating S_C; and (d) S_C impersonating Carson Daly (by Anthony Quintano / CC BY / cropped from https://goo.gl/VfnDct).

Sharif et al., "Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition", *Computer and Communications Security*, 2016

Adversarial patch



Brown et al., "Adversarial patch", *arxiv only*, Dec 2017

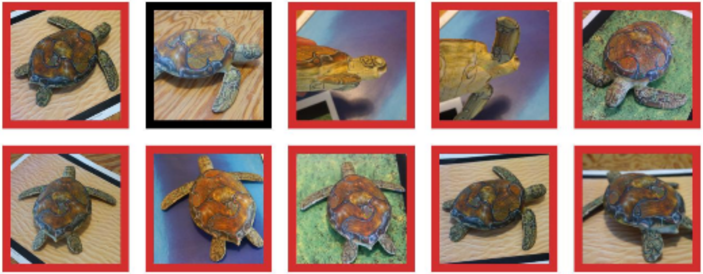
Let's try it out!

toaster (0.97)
pencil sharpener (0.007)
soap dispenser (0.004)

cockroach (0.15)
loafer (0.13)
cowboy boot (0.03)

toaster (0.74)
soap dispenser (0.03)
punching bag (0.02)

Adversarial turtle



classified as turtle　classified as rifle　classified as other

Figure 1: Randomly sampled poses of a **single** 3D-printed turtle adversarially perturbed to classify as a rifle at every viewpoint by an ImageNet classifier. An unperturbed model is classified correctly as a turtle 100% of the time. See `https://youtu.be/YXy6oX1iNoA` for a video where every frame is fed through the classifier: the turtle is consistently classified as a rifle.

Athalye et al., "Synthesizing Robust Adversarial Examples", *arxiv only*, Oct 2017, `https://youtu.be/YXy6oX1iNoA`

Take home

- Generating adversarial examples is easy
- Understanding why they exist is hard
- Defenses are mostly band-aids and quickly broken
- Opportunities (“Probleme sind nur dornige Chancen”?)

References I

.

Athalye, Anish, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. "Synthesizing Robust Adversarial Examples". In: *arxiv only* (Oct 2017). URL: http://arxiv.org/abs/1707.07397.

Brown, Tom B., Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. "Adversarial patch". In: *arxiv only* (Dec 2017). URL: https://arxiv.org/abs/1712.09665.

Carlini, Nicholas and David Wagner. "Adversarial examples are not easily detected: Bypassing ten detection methods." In: *AISEC* (2017). URL: https://arxiv.org/abs/1705.07263.

Dziugaite, Gintare Karolina, Zoubin Ghahramani, and Daniel M. Roy. "A study of the effect of JPG compression on adversarial images". In: *arxiv only* (2016). URL: http://arxiv.org/abs/1608.00853.

Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: *ICLR* (2015). URL: http://arxiv.org/abs/1412.6572.

References II

Kurakin, Alexey, Ian J. Goodfellow, and Samy Bengio. "Adversarial Machine Learning at Scale". In: *ICLR* (2017). URL: http://arxiv.org/abs/1611.01236.

Liu, Yanpei, Xinyun Chen, Chang Liu, and Dawn Song. "Delving into Transferable Adversarial Examples and Black-box Attacks". In: *ICLR* (2017). URL: http://arxiv.org/abs/1611.02770.

Moosavi-Dezfooli, Seyed-Mohsen, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. "Universal adversarial perturbations". In: *CVPR* (2017). URL: http://arxiv.org/abs/1610.08401.

Moosavi-Dezfooli, Seyed-Mohsen, Alhussein Fawzi, and Pascal Frossard. "DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks". In: *CVPR* (2016). URL: https://arxiv.org/abs/1511.04599.

References III

Papernot, Nicolas, Patrick D. McDaniel, and Ian J. Goodfellow. "Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples". In: *arvix only* (2016). URL: http://arxiv.org/abs/1605.07277.

Sabour, Sara, Yanshuai Cao, Fartash Faghri, and David J Fleet. "Adversarial manipulation of deep representations". In: *ICLR* (2016).

Sharif, Mahmood, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. "Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition". In: *Computer and Communications Security* (2016). URL: http://doi.acm.org/10.1145/2976749.2978392.

Szegedy, Christian, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. "Intriguing properties of neural networks". In: *arxiv* (2013). URL: http://arxiv.org/abs/1312.6199.

References IV

Tabacof, P. and E. Valle. "Exploring the space of adversarial images". In: *IJCNN* (2016).

Tanay, Thomas and Lewis D. Griffin. "A Boundary Tilting Persepective on the Phenomenon of Adversarial Examples". In: *arxiv only* (2016). URL: http://arxiv.org/abs/1608.07690.