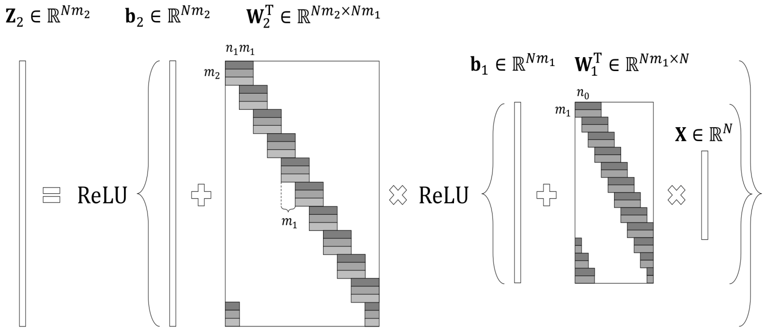
Outline for today

- Introduction to Generative Adversarial Networks
  - Inverse conv net for generating images
  - The adversarial game
  - Applications and improved training strategies, WGANs

- Adversarial attacks on deep nets
  - Sign gradient attack, DeepFool, rotations and translations
  - Universal adversarial examples, transferability between nets
  - Adversarial physical objects

CNN model through sparse coding (Papyan et al. 16'[1])

Consider a deep conv. net composed of two convolutional layers:

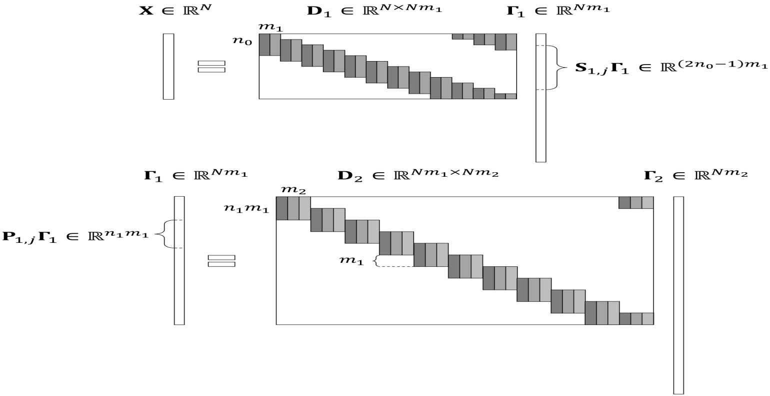

$\mathbf{Z}_2 \in \mathbb{R}^{Nm_2}$
$\mathbf{b}_2 \in \mathbb{R}^{Nm_2}$
$\mathbf{W}_2^{\mathrm{T}} \in \mathbb{R}^{Nm_2 \times Nm_1}$
$n_1 m_1$
m_2
m_1
ReLU
ReLU
$\mathbf{b}_1 \in \mathbb{R}^{Nm_1}$
$\mathbf{W}_1^{\mathrm{T}} \in \mathbb{R}^{Nm_1 \times N}$
n_0
m_1
$\mathbf{X} \in \mathbb{R}^{N}$

The forward map (note notation using transpose of $W^{(i)}$):

$$Z_2 = \sigma\left(b^{(2)} + (W^{(2)})^T \sigma\left(b^{(1)} + (W^{(1)})^T x\right)\right)$$

[1] https://arxiv.org/pdf/1607.08194.pdf

Deconvolutional NN data model (Papyan et al. 16'[2])



$\mathbf{X} \in \mathbb{R}^N$
$\mathbf{D}_1 \in \mathbb{R}^{N \times N m_1}$
$\mathbf{\Gamma}_1 \in \mathbb{R}^{N m_1}$
m_1
n_0
$\mathbf{S}_{1,j}\mathbf{\Gamma}_1 \in \mathbb{R}^{(2n_0-1)m_1}$
$\mathbf{\Gamma}_1 \in \mathbb{R}^{N m_1}$
$\mathbf{D}_2 \in \mathbb{R}^{N m_1 \times N m_2}$
$\mathbf{\Gamma}_2 \in \mathbb{R}^{N m_2}$
m_2
$n_1 m_1$
$\mathbf{P}_{1,j}\mathbf{\Gamma}_1 \in \mathbb{R}^{n_1 m_1}$
m_1

Two layer deconvolutional data model with weight matrices fixed, $W^{(i)} = D_i$, and $\Gamma_i \geq 0$ whose values compose data element X.

[2] https://arxiv.org/pdf/1607.08194.pdf

Generative deep nets (Goodfellow et al. 14'[4])

Example of a deep convolutional generator:

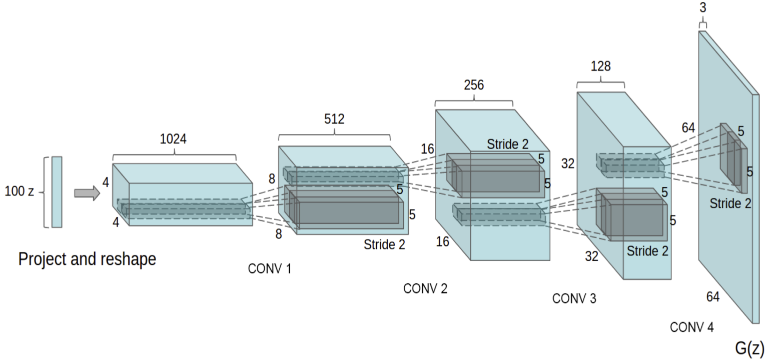


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. [3]

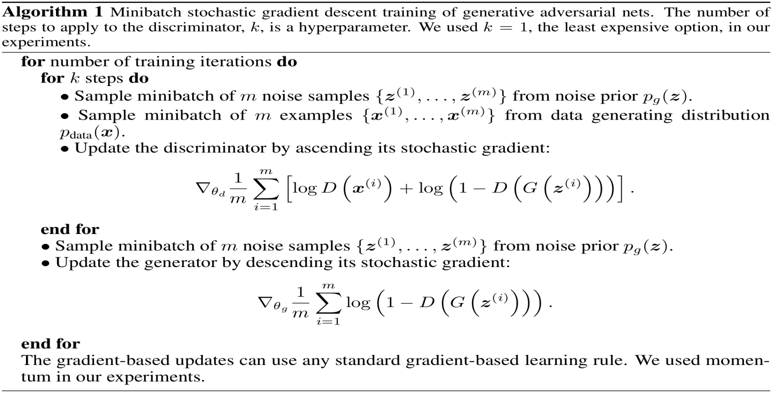
[3] https://arxiv.org/pdf/1511.06434.pdf

[4] https://arxiv.org/pdf/1406.2661.pdf

Generative deep nets (Goodfellow et al. 14'[5])

Train the two network parameters using the objective

$$\min_G \max_D n^{-1} \sum_{\mu=1}^{n} \log(D(x_\mu, y_\mu)) + p^{-1} \sum_{p} \log\left(1 - D(G(z_p), y_p)\right)$$



Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

```
for number of training iterations do
    for $k$ steps do
        • Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
        • Sample minibatch of $m$ examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
        • Update the discriminator by ascending its stochastic gradient:
            $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$
    end for
    • Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    • Update the generator by descending its stochastic gradient:
        $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$
end for
```

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

[5] https://arxiv.org/pdf/1406.2661.pdf

Generative deep nets (Radford et al. 16'[6])

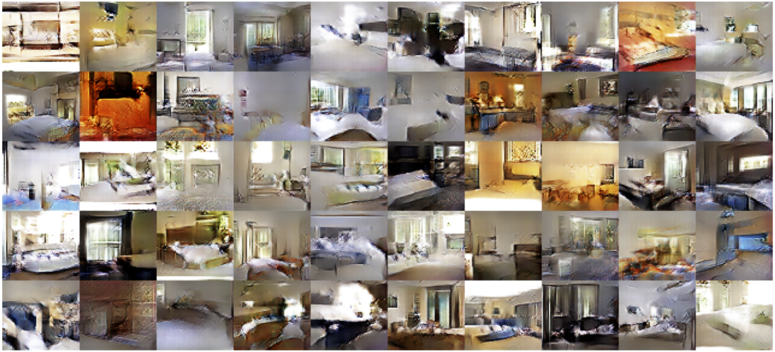


Figure 2: Generated bedrooms after one training pass through the dataset. Theoretically, the model could learn to memorize training examples, but this is experimentally unlikely as we train with a small learning rate and minibatch SGD. We are aware of no prior empirical evidence demonstrating memorization with SGD and a small learning rate.

[6] https://arxiv.org/pdf/1511.06434.pdf

Generative deep nets (Radford et al. 16'[7])

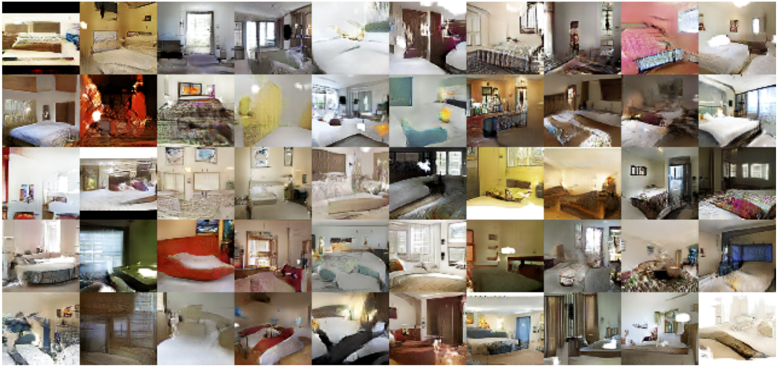
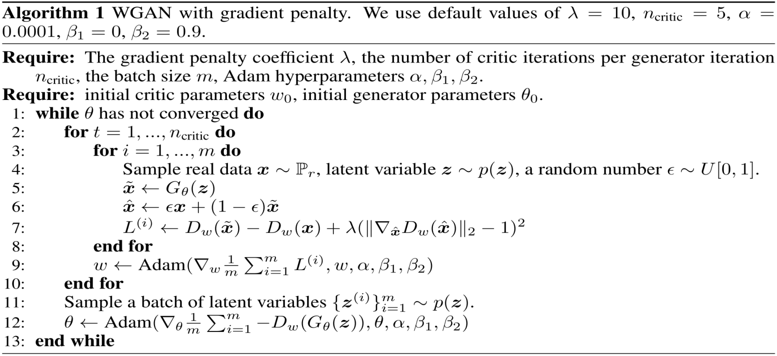


Figure 3: Generated bedrooms after five epochs of training. There appears to be evidence of visual under-fitting via repeated noise textures across multiple samples such as the base boards of some of the beds.

[7] `https://arxiv.org/pdf/1511.06434.pdf`

Wasserstein GAN (Arjovsky et al. 17'[9])

One of the central challenges with GANs is the ability to train the parameters. Improvements have been made through choice of generative architecture (DC-GAN of Radford) and through different training objective functions (W-GAN)



Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

```
Require: The gradient penalty coefficient $\lambda$, the number of critic iterations per generator iteration $n_{\text{critic}}$, the batch size $m$, Adam hyperparameters $\alpha, \beta_1, \beta_2$.
Require: initial critic parameters $w_0$, initial generator parameters $\theta_0$.
while $\theta$ has not converged do
    for $t = 1, ..., n_{\text{critic}}$ do
        for $i = 1, ..., m$ do
            Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$.
            $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
            $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
            $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
        end for
        $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$
    end for
    Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim p(\boldsymbol{z})$.
    $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
end while
```

[8]

[8] https://arxiv.org/pdf/1704.00028.pdf

[9] https://arxiv.org/pdf/1701.07875.pdf

Wasserstein GAN (Arjovsky et al. 17'[10])

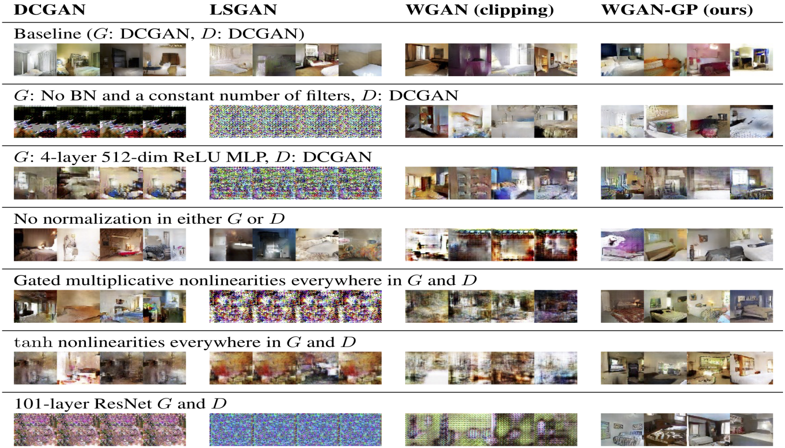

DCGAN
LSGAN
WGAN (clipping)
WGAN-GP (ours)
Baseline (G: DCGAN, D: DCGAN)
G: No BN and a constant number of filters, D: DCGAN
G: 4-layer 512-dim ReLU MLP, D: DCGAN
No normalization in either G or D
Gated multiplicative nonlinearities everywhere in G and D
tanh nonlinearities everywhere in G and D
101-layer ResNet G and D

Figure 2: Different GAN architectures trained with different methods. We only succeeded in training every architecture with a shared set of hyperparameters using WGAN-GP.

[10] https://arxiv.org/pdf/1704.00028.pdf

Wasserstein GAN (Arjovsky et al. 17'[11])

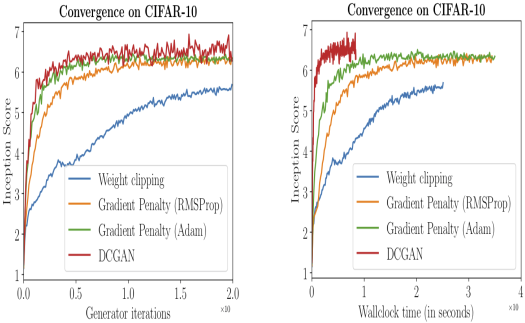


Figure 3: CIFAR-10 Inception score over generator iterations (left) or wall-clock time (right) for four models: WGAN with weight clipping, WGAN-GP with RMSProp and Adam (to control for the optimizer), and DCGAN. WGAN-GP significantly outperforms weight clipping and performs comparably to DCGAN.

[11] https://arxiv.org/pdf/1704.00028.pdf

Large scale WGAN (Karras et al. 18'[12])

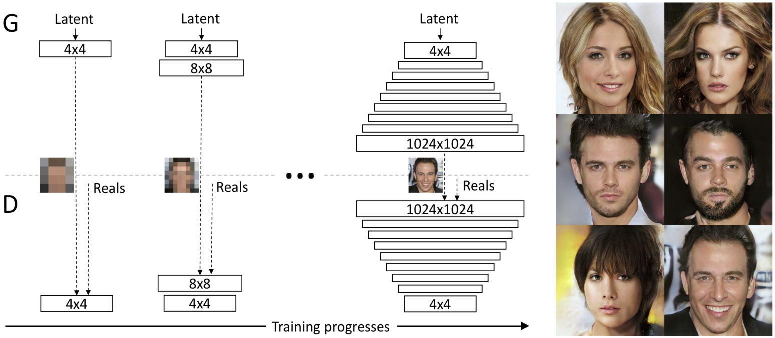


Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout the process. Here $\boxed{N \times N}$ refers to convolutional layers operating on $N \times N$ spatial resolution. This allows stable synthesis in high resolutions and also speeds up training considerably. One the right we show six example images generated using progressive growing at 1024×1024.

[12] https://arxiv.org/abs/1710.10196

Large scale WGAN (Karras et al. 18'[13])

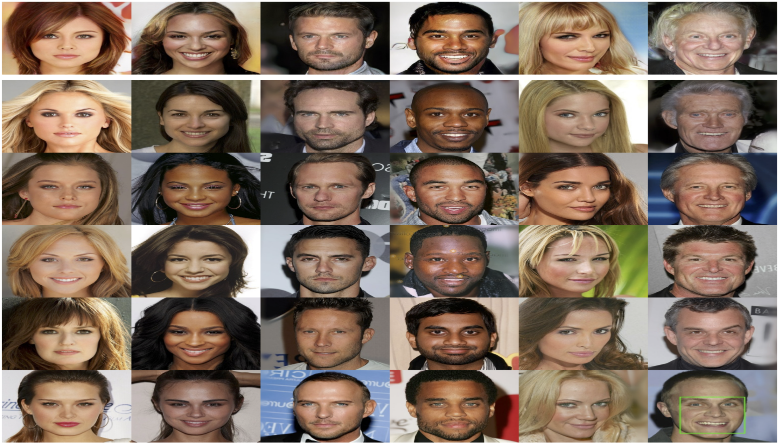
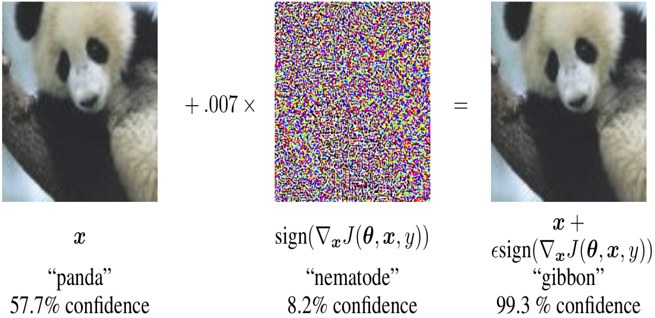


Figure 10: Top: Our CELEBA-HQ results. Next five rows: Nearest neighbors found from the training data, based on feature-space distance. We used activations from five VGG layers, as suggested by Chen & Koltun (2017). Only the crop highlighted in bottom right image was used for comparison in order to exclude image background and focus the search on matching facial features.

[13] https://arxiv.org/abs/1710.10196

Adversarial examples for deep nets (Goodfellow et al. 15[14])

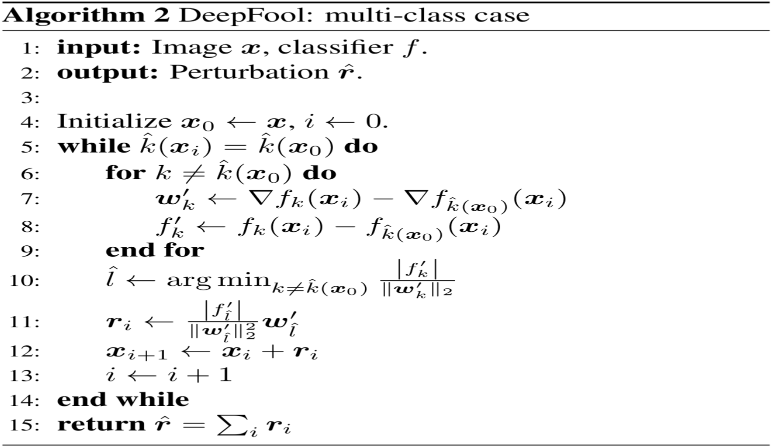


$+ .007 \times$
$=$
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

Figure 1: A demonstration of fast adversarial example generation applied to GoogLeNet (Szegedy et al., 2014a) on ImageNet. By adding an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, we can change GoogLeNet's classification of the image. Here our ϵ of .007 corresponds to the magnitude of the smallest bit of an 8 bit image encoding after GoogLeNet's conversion to real numbers.

[14] `https://arxiv.org/pdf/1412.6572.pdf`

DeepFool algorithm (Moosavi-Dezfooli et al. 15[15])



Algorithm 2 DeepFool: multi-class case

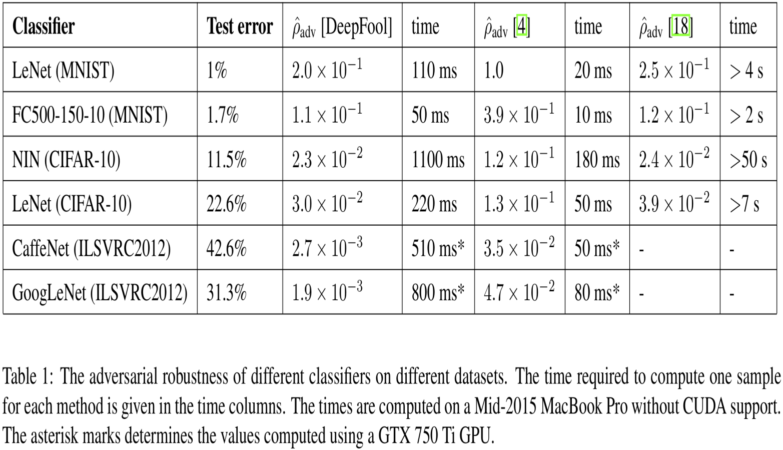
```
input: Image $\boldsymbol{x}$, classifier $f$.
output: Perturbation $\hat{\boldsymbol{r}}$.

Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$, $i \leftarrow 0$.
while $\hat{k}(\boldsymbol{x}_i) = \hat{k}(\boldsymbol{x}_0)$ do
    for $k \neq \hat{k}(\boldsymbol{x}_0)$ do
        $\boldsymbol{w}'_k \leftarrow \nabla f_k(\boldsymbol{x}_i) - \nabla f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
        $f'_k \leftarrow f_k(\boldsymbol{x}_i) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
    end for
    $\hat{l} \leftarrow \arg\min_{k \neq \hat{k}(\boldsymbol{x}_0)} \frac{|f'_k|}{\|\boldsymbol{w}'_k\|_2}$
    $\boldsymbol{r}_i \leftarrow \frac{|f'_{\hat{l}}|}{\|\boldsymbol{w}'_{\hat{l}}\|_2^2} \boldsymbol{w}'_{\hat{l}}$
    $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$
    $i \leftarrow i + 1$
end while
return $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$
```

Alternative to Goodfellow approach of
$\hat{r}(x_\mu) = \epsilon \mathrm{sign}(\mathrm{grad}_x \updownarrow(\theta; x_\mu, y_\mu))$.

[15] https://arxiv.org/pdf/1511.04599.pdf

DeepFool algorithm (Moosavi-Dezfooli et al. 15[16])



Classifier	Test error	$\hat{\rho}_{\text{adv}}$ [DeepFool]	time	$\hat{\rho}_{\text{adv}}$ [4]	time	$\hat{\rho}_{\text{adv}}$ [18]	time
LeNet (MNIST)	1%	2.0×10^{-1}	110 ms	1.0	20 ms	2.5×10^{-1}	> 4 s
FC500-150-10 (MNIST)	1.7%	1.1×10^{-1}	50 ms	3.9×10^{-1}	10 ms	1.2×10^{-1}	> 2 s
NIN (CIFAR-10)	11.5%	2.3×10^{-2}	1100 ms	1.2×10^{-1}	180 ms	2.4×10^{-2}	>50 s
LeNet (CIFAR-10)	22.6%	3.0×10^{-2}	220 ms	1.3×10^{-1}	50 ms	3.9×10^{-2}	>7 s
CaffeNet (ILSVRC2012)	42.6%	2.7×10^{-3}	510 ms*	3.5×10^{-2}	50 ms*	-	-
GoogLeNet (ILSVRC2012)	31.3%	1.9×10^{-3}	800 ms*	4.7×10^{-2}	80 ms*	-	-

Table 1: The adversarial robustness of different classifiers on different datasets. The time required to compute one sample for each method is given in the time columns. The times are computed on a Mid-2015 MacBook Pro without CUDA support. The asterisk marks determines the values computed using a GTX 750 Ti GPU.

Average relative error of adversarial example $\hat{r}(x)$ such that $f(x) \neq f(x + \hat{r}(x))$: $\hat{\rho}_{adv}(f) = |\mathcal{D}|^{-1} \sum_{x \in \mathcal{D}} \frac{\|\hat{r}(x)\|_2}{\|x\|_2}$

[16] https://arxiv.org/pdf/1511.04599.pdf

Rotations and Translations for CNNs (Engstrom et al. 18[17])

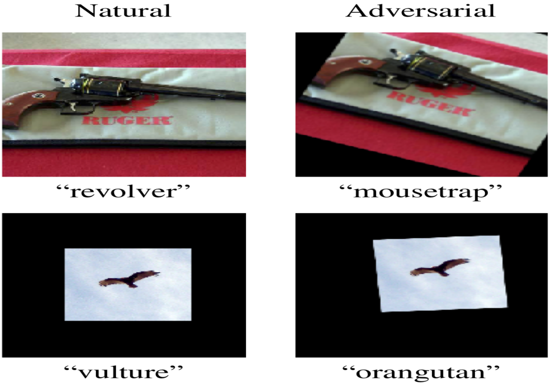


Figure 1: Examples of adversarial transformations and their predictions in the standard and "black canvas" setting.

[17] https://arxiv.org/pdf/1712.02779.pdf

Rotations and Translations for CNNs (Engstrom et al. 18[18])

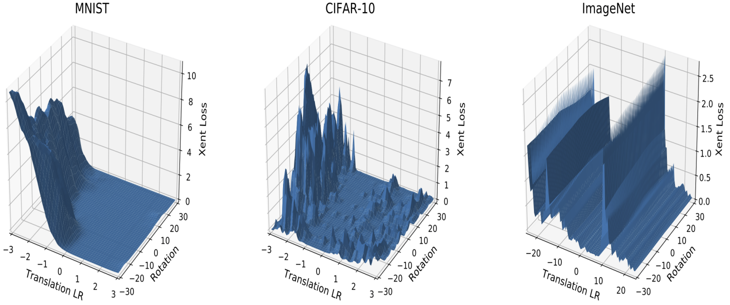


Figure 3: Loss landscape of a random example for each dataset when performing left-right translations and rotations. Translations and rotations are restricted to 10% of the image pixels and 30 deg respectively. We observe that the landscape is significantly non-concave, making rendering FO methods for adversarial example generation powerless. Additional examples are visualized in Figure 9 of the Appendix.

[18] https://arxiv.org/pdf/1712.02779.pdf

Universal adversary (Moosavi-Dezfooli et al. 16[19])

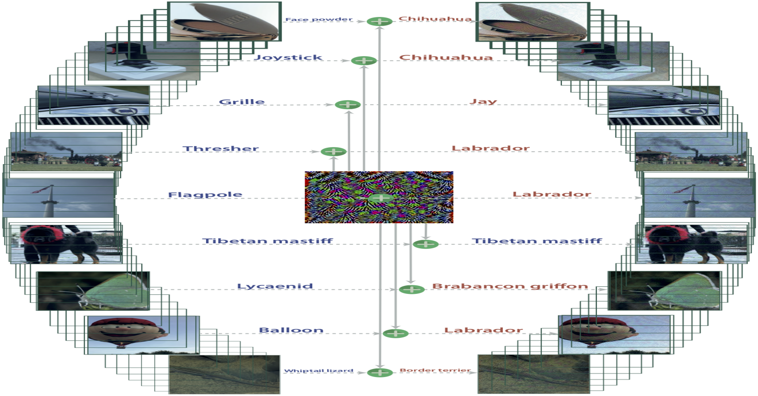
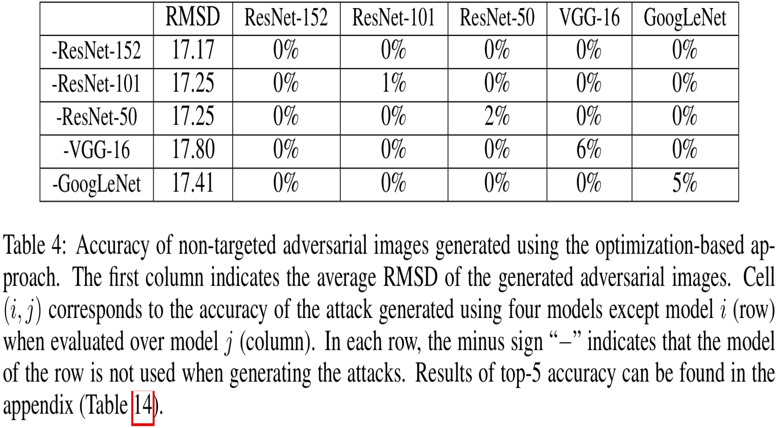


Figure 1: When added to a natural image, a universal perturbation image causes the image to be misclassified by the deep neural network with high probability. *Left images:* Original natural images. The labels are shown on top of each arrow. *Central image:* Universal perturbation. *Right images:* Perturbed images. The estimated labels of the perturbed images are shown on top of each arrow.

[19] `https://arxiv.org/pdf/1610.08401.pdf`

Transferability between nets (Liu et al. 16[20])



	RMSD	ResNet-152	ResNet-101	ResNet-50	VGG-16	GoogLeNet
-ResNet-152	17.17	0%	0%	0%	0%	0%
-ResNet-101	17.25	0%	1%	0%	0%	0%
-ResNet-50	17.25	0%	0%	2%	0%	0%
-VGG-16	17.80	0%	0%	0%	6%	0%
-GoogLeNet	17.41	0%	0%	0%	0%	5%

Table 4: Accuracy of non-targeted adversarial images generated using the optimization-based approach. The first column indicates the average RMSD of the generated adversarial images. Cell (i, j) corresponds to the accuracy of the attack generated using four models except model i (row) when evaluated over model j (column). In each row, the minus sign "–" indicates that the model of the row is not used when generating the attacks. Results of top-5 accuracy can be found in the appendix (Table 14).

RMSD is the ℓ^2 energy of the perturbation.

[20] `https://arxiv.org/pdf/1611.02770.pdf`

Transferability between nets (Liu et al. 16[21])

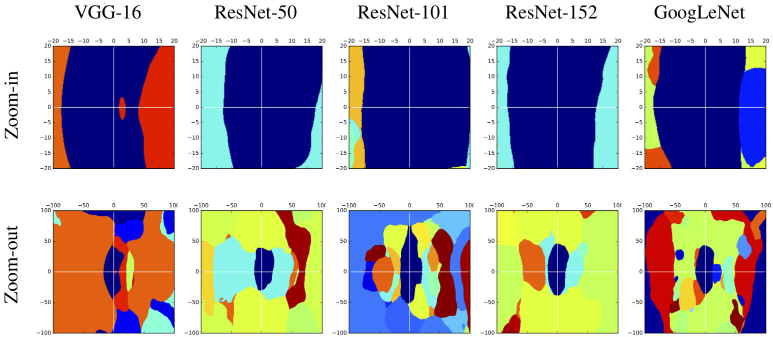


Figure 3: Decision regions of different models. We pick the same two directions for all plots: one is the gradient direction of VGG-16 (x-axis), and the other is a random orthogonal direction (y-axis). Each point in the span plane shows the predicted label of the image generated by adding a noise to the original image (e.g., the origin corresponds to the predicted label of the original image). The units of both axises are 1 pixel values. All sub-figure plots the regions on the span plane using the same color for the same label. The image is in Figure 2.

[21] https://arxiv.org/pdf/1611.02770.pdf

Adversarial physical object: Turtle (Athalye et al. 17[22])

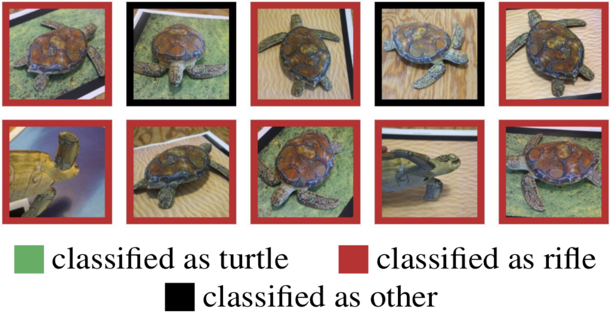


Figure 1. Randomly sampled poses of a 3D-printed turtle adversarially perturbed to classify as a rifle at every viewpoint[2]. An unperturbed model is classified correctly as a turtle nearly 100% of the time.

[22] `https://arxiv.org/pdf/1707.07397.pdf`

Adversarial graffiti (Eykholt et al. 17[23])

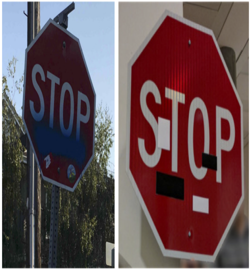
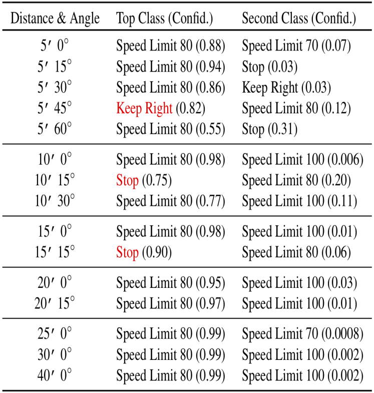


Figure 1: The left image shows real graffiti on a Stop sign, something that most humans would not think is suspicious. The right image shows our a physical perturbation applied to a Stop sign. We design our perturbations to mimic graffiti, and thus "hide in the human psyche."

Table 5: A camouflage art attack on GTSRB-CNN. See example images in Table 1. The targeted-attack success rate is 80% (true class label: Stop, target: Speed Limit 80).



Distance & Angle	Top Class (Confid.)	Second Class (Confid.)
5′ 0°	Speed Limit 80 (0.88)	Speed Limit 70 (0.07)
5′ 15°	Speed Limit 80 (0.94)	Stop (0.03)
5′ 30°	Speed Limit 80 (0.86)	Keep Right (0.03)
5′ 45°	Keep Right (0.82)	Speed Limit 80 (0.12)
5′ 60°	Speed Limit 80 (0.55)	Stop (0.31)
10′ 0°	Speed Limit 80 (0.98)	Speed Limit 100 (0.006)
10′ 15°	Stop (0.75)	Speed Limit 80 (0.20)
10′ 30°	Speed Limit 80 (0.77)	Speed Limit 100 (0.11)
15′ 0°	Speed Limit 80 (0.98)	Speed Limit 100 (0.01)
15′ 15°	Stop (0.90)	Speed Limit 80 (0.06)
20′ 0°	Speed Limit 80 (0.95)	Speed Limit 100 (0.03)
20′ 15°	Speed Limit 80 (0.97)	Speed Limit 100 (0.01)
25′ 0°	Speed Limit 80 (0.99)	Speed Limit 70 (0.0008)
30′ 0°	Speed Limit 80 (0.99)	Speed Limit 100 (0.002)
40′ 0°	Speed Limit 80 (0.99)	Speed Limit 100 (0.002)

[23] https://arxiv.org/pdf/1707.08945.pdf