GANs for AML

Shoukun Sun

October 28, 2020

Generative Modeling

- Question: can we build a model to approximate a data distribution?
- Formally we are given $x \sim p_{data}(x)$ and a finite sample from this distribution

$$X = \{x | x \sim p_{data}(x)\}, |X| = n$$

- Problem: can we find a model such that

$$p_{model}(x; \theta) \approx p_{data}(x)$$

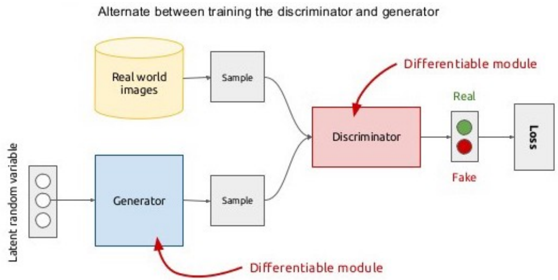
Basic of GANs

Generative Adversarial Networks (GANs) is a framework for estimating generative models via an adversarial process. This process simultaneously train two models:

- a generative model G that captures the data distribution;
- a discriminative model D that judges if a sample comes from training data rather than G.

These two model contest with each other in the zero-sum game.

Training GANs



Training GANs

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

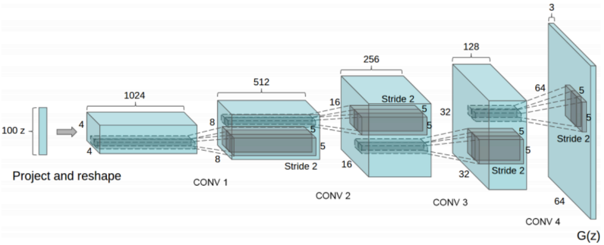
- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

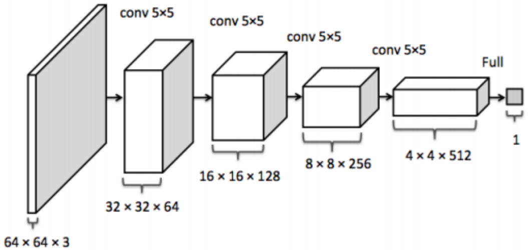
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

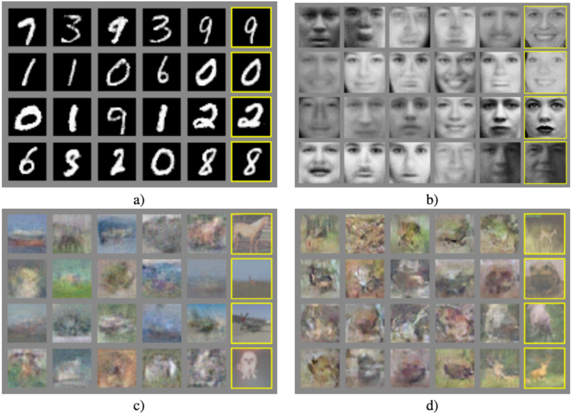
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Training GANs





Examples



The application of GANs is not limited to images, but can also be extended to text and music.

Challenges of Training GANs

- Vanishing Gradient
  If the D is too good, G training can fail due to vanishing gradients.
- Mode Collapse
  The generator produces the same output (or a small set of outputs).
- Failure to Converge
  GANs frequently fail to converge as its complexity.

Figure: Mode Collapse

Pix2pix

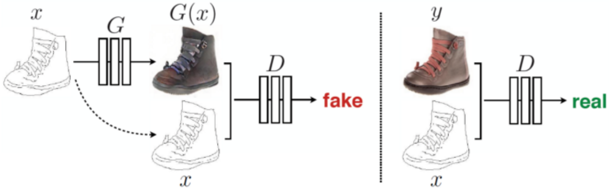


Figure: Pix2pix process

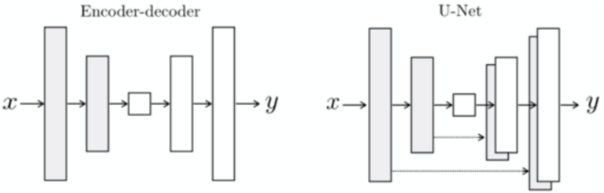
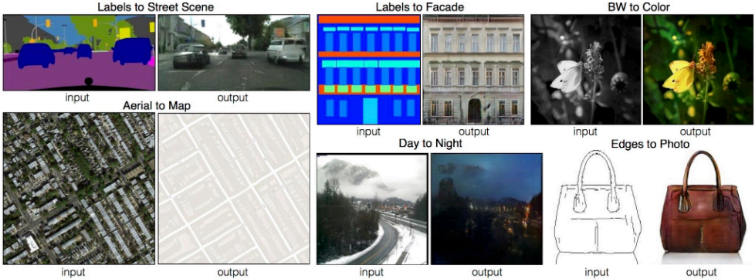


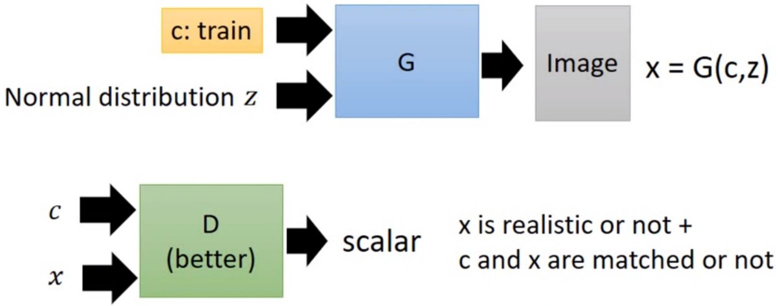
Figure: Generator

Pix2pix Examples

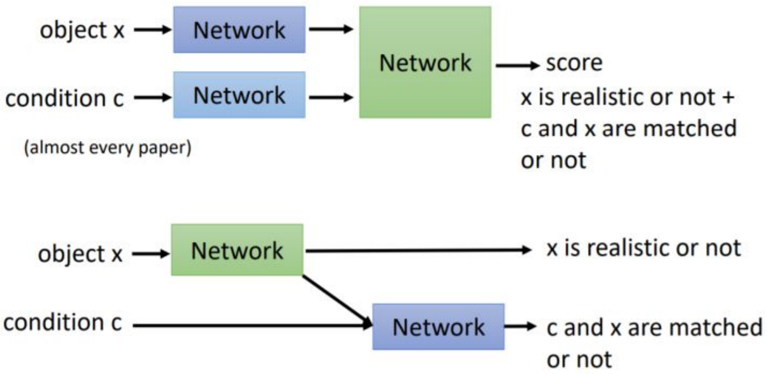


Online demo: `https://affinelayer.com/pixsrv/`

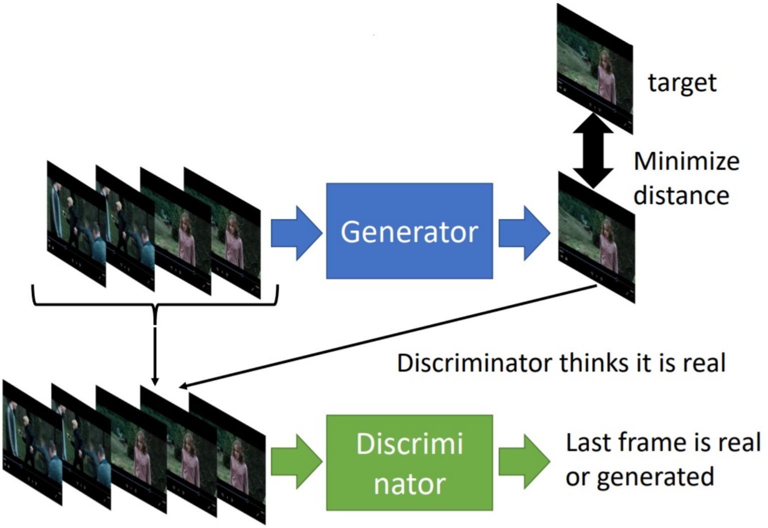
Conditional GAN



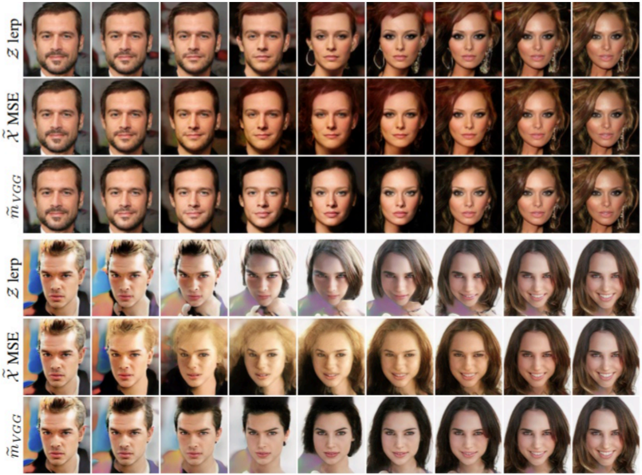
Conditional GAN Architecture



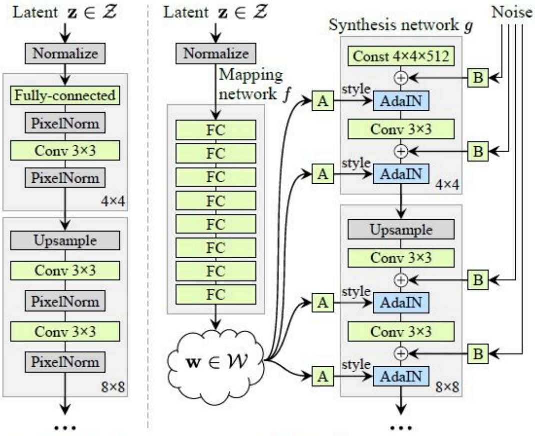
Conditional GAN Application



StyleGAN



StyleGAN

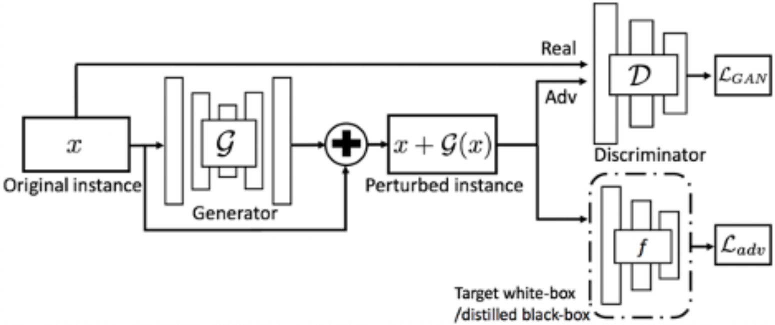


(a) Traditional (b) Style-based generator

AdvGAN

- Title: Generating Adversarial Examples with Adversarial Networks.
- Semi-whitebox;black-box
  Semi-whitebox: once the generator is trained, it can generate perturbations efficiently for any instance, no need to access the classifier.
- Time consuming while training; efficiently while generating perturbations.

Training AdvGAN



Results of AdvGAN

Results of AdvGAN

	FGSM	Opt.	Trans.	AdvGAN
Run time	0.06s	>3h	-	<0.01s
Targeted Attack	✓	✓	Ens.	✓
Black-box Attack			✓	✓

Table 1: Comparison with the state-of-the-art attack methods. Run time is measured for generating 1,000 adversarial instances during test time. Opt. represents the optimization based method, and Trans. denotes black-box attacks based on transferability.

Model	MNIST(%) A	MNIST(%) B	MNIST(%) C	CIFAR-10(%) ResNet	CIFAR-10(%) Wide ResNet
Accuracy (p)	99.0	99.2	99.1	92.4	95.0
Attack Success Rate (w)	97.9	97.1	98.3	94.7	99.3
Attack Success Rate (b-D)	93.4	90.1	94.0	78.5	81.8
Attack Success Rate (b-S)	30.7	66.6	87.3	10.3	13.3

Table 2: Accuracy of different models on pristine data, and the attack success rate of adversarial examples generated against different models by AdvGAN on MNIST and CIFAR-10. p: pristine test data; w: semi-whitebox attack; b-D: black-box attack with dynamic distillation strategy; b-S: black-box attack with static distillation strategy.

Data	Model	Defense	FGSM	Opt.	**AdvGAN**
MNIST	A	Adv.	4.3%	4.6%	**8.0%**
		Ens.	1.6%	4.2%	**6.3%**
		Iter.Adv.	4.4%	2.96%	**5.6%**
	B	Adv.	6.0%	4.5%	**7.2%**
		Ens.	2.7%	3.18%	**5.8%**
		Iter.Adv.	**9.0%**	3.0%	6.6%
	C	Adv.	2.7%	2.95%	**18.7%**
		Ens.	1.6%	2.2%	**13.5%**
		Iter.Adv.	1.6%	1.9%	**12.6%**
CIFAR10	ResNet	Adv.	13.10%	11.9%	**16.03%**
		Ens.	10.00%	10.3%	**14.32%**
		Iter.Adv	22.8%	21.4%	**29.47%**
	Wide ResNet	Adv.	5.04%	7.61%	**14.26%**
		Ens.	4.65%	8.43%	**13.94 %**
		Iter.Adv.	14.9%	13.90%	**20.75%**

Table 3: Attack success rate of adversarial examples generated by AdvGAN in semi-whitebox setting, and other white-box attacks under defenses on MNIST and CIFAR-10.

Defense	MNIST FGSM	MNIST Opt.	MNIST **AdvGAN**	CIFAR-10 FGSM	CIFAR-10 Opt.	CIFAR-10 **AdvGAN**
Adv.	3.1%	3.5%	**11.5%**	13.58%	10.8%	**15.96%**
Ens.	2.5%	3.4%	**10.3%**	10.49%	9.6%	**12.47%**
Iter.Adv.	2.4%	2.5%	**12.2%**	22.96%	21.70%	**24.28%**

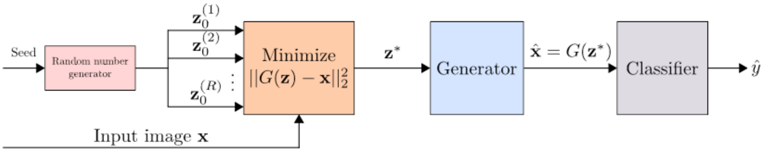
Table 4: Attack success rate of adversarial examples generated by different black-box adversarial strategies under defenses on MNIST and CIFAR-10

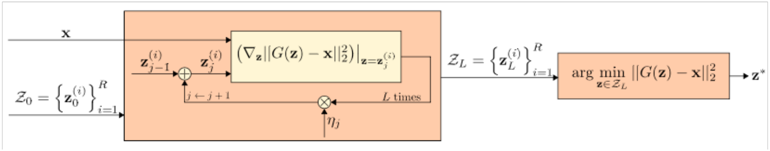
Defense-GAN

- Title: Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models
- 'denoise' adversarial examples
- Defense-GAN is trained to model the distribution of unperturbed images.
- Defense-GAN can be used with **any classification** model and **does not modify the classifier** structure or training procedure.

Training Defense-GAN

Obtain a G on training dataset first.





Results of Defense-GAN

Table 1: Classification accuracies of different classifier and substitute model combinations using various defense strategies on the MNIST dataset, under FGSM black-box attacks with $\epsilon = 0.3$. Defense-GAN has $L = 200$ and $R = 10$.

Classifier/ Substitute	No Attack	No Defense	**Defense-GAN-Rec**	**Defense-GAN-Orig**	MagNet	Adv. Tr. $\epsilon = 0.3$	Adv. Tr. $\epsilon = 0.15$
A/B	0.9970	0.6343	<u>0.9312</u>	0.9282	0.6937	**0.9654**	0.6223
A/E	0.9970	0.5432	0.9139	0.9221	0.6710	**0.9668**	<u>0.9327</u>
B/B	0.9618	0.2816	<u>0.9057</u>	**0.9105**	0.5687	0.2092	0.3441
B/E	0.9618	0.2128	<u>0.8841</u>	**0.8892**	0.4627	0.1120	0.3354
C/B	0.9959	0.6648	<u>0.9357</u>	0.9322	0.7571	**0.9834**	0.9208
C/E	0.9959	0.8050	0.9223	0.9182	0.6760	**0.9843**	<u>0.9755</u>
D/B	0.9920	0.4641	<u>0.9272</u>	**0.9323**	0.6817	0.7667	0.8514
D/E	0.9920	0.3931	**0.9164**	<u>0.9155</u>	0.6073	0.7676	0.7129

Results of Defense-GAN

Table 4: Classification accuracies of different classifier models using various defense strategies on the MNIST (top) and F-MNIST (bottom) datasets, under FGSM, RAND+FGSM, and CW white-box attacks. Defense-GAN has $L = 200$ and $R = 10$.

Attack	Classifier Model	No Attack	No Defense	**Defense-GAN-Rec**	MagNet	Adv. Tr. $\epsilon = 0.3$
FGSM $\epsilon = 0.3$	A	0.997	0.217	**0.988**	0.191	0.651
	B	0.962	0.022	**0.956**	0.082	0.060
	C	0.996	0.331	**0.989**	0.163	0.786
	D	0.992	0.038	**0.980**	0.094	0.732
RAND+FGSM $\epsilon = 0.3, \alpha = 0.05$	A	0.997	0.179	**0.988**	0.171	0.774
	B	0.962	0.017	**0.944**	0.091	0.138
	C	0.996	0.103	**0.985**	0.151	0.907
	D	0.992	0.050	**0.980**	0.115	0.539
CW ℓ_2 norm	A	0.997	0.141	**0.989**	0.038	0.077
	B	0.962	0.032	**0.916**	0.034	0.280
	C	0.996	0.126	**0.989**	0.025	0.031
	D	0.992	0.032	**0.983**	0.021	0.010

Attack	Classifier Model	No Attack	No Defense	**Defense-GAN-Rec**	MagNet	Adv. Tr. $\epsilon = 0.3$
FGSM $\epsilon = 0.3$	A	0.934	0.102	**0.879**	0.089	0.797
	B	0.747	0.102	**0.629**	0.168	0.136
	C	0.933	0.139	**0.896**	0.110	0.804
	D	0.892	0.082	**0.875**	0.099	0.698
RAND+FGSM $\epsilon = 0.3, \alpha = 0.05$	A	0.934	0.102	**0.888**	0.096	0.447
	B	0.747	0.131	**0.661**	0.161	0.119
	C	0.933	0.105	**0.893**	0.112	0.699
	D	0.892	0.091	**0.862**	0.104	0.626
CW ℓ_2 norm	A	0.934	0.076	**0.896**	0.060	0.157
	B	0.747	0.172	**0.656**	0.131	0.118
	C	0.933	0.063	**0.896**	0.084	0.107
	D	0.892	0.090	**0.875**	0.069	0.149