Generative Adversarial Networks

Workshop on Learning and Control
IIT Mandi

Pramod P. Khargonekar and Deepan Muthurayan

Department of Electrical Engineering and Computer Science
University of California, Irvine

July 2019

Outline

1. Generative Adversarial Framework

2. GAN Development and Key Concepts and Ideas
   - First GAN
   - Deep Convolutional (DC) GAN
   - Further Improvements to GAN
   - Energy Based (EB) GAN
   - Auxiliary Classifier (AC)GAN
   - Conditional GANs with Projection Discriminator
   - Spectral Normalization (SN) GAN
   - Self Attention (SA) GAN
   - Other GAN Formulation

3. Application of GANs
   - Semi-supervised Learning
   - Video Prediction

Generative Adversarial Framework

Motivation

- Discriminative models do classification but do not necessarily capture the probability distribution of the data.
- Supervised learning requires large amount of labeled data. Can we generate synthetic data?
- Generative adversarial networks aim to address these issues.
- Prior alternative approaches: variational auto encoders, deep belief networks, generative stochastic networks, etc.

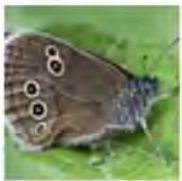
Generative Adversarial Framework

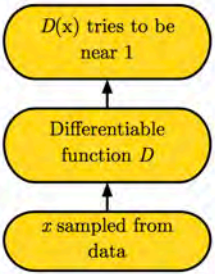
- a generative model G and a discriminative model D
- D estimates the probability that a sample came from data rather than G
- Objective of G: to generate a sample that cannot be distinguished by the discriminator D as fake
- D acts as the adversary of G

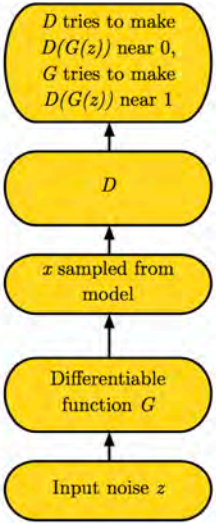
Central Idea

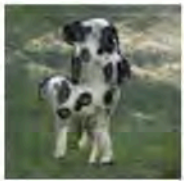
- Generative model G is like a team of counterfeiters
- Discriminative model D is like the team trying to identify the counterfeiters
- There is a competitive game among them
- Competition is expected to improve both teams and thus the generating ability of the model G

Adversarial Nets Framework









(Goodfellow et al., 2014)

Notation

- z: input to the generative model
- G: generator model; $G(z)$ is the generated output for input z
- x: input to the discriminator model
- D: discriminator model; $D(x)$ is the output of the discriminator for data x
- p_{data}: distribution of data being modeled
- p_z: distribution of the variable z
- p_g: distribution of the generated samples

Formulation

- Generative adversarial framework minmax optimization problem:

$$\min_G \max_D [\mathbb{E}_{x \sim p_{\text{data}}} \log[D(x)] + \mathbb{E}_{z \sim p_z} \log(1 - D(G(z)))] \tag{1}$$

- Inner maximization: with G given, identify a discriminator model D that maximizes recognition of samples from the data distribution and rejection of samples generated by G
- Outer minimization: identify a generator model G that minimizes the performance of the optimal D

Two Player Min-Max Game

- GA framework $\equiv$ Two player Min-Max game
- A solution exists to the game and is given by:
  - G recovers the training data, i.e., the distribution of data generated by G is the same as the distribution of the training data
  - D equals $1/2$ everywhere

Theoretical Results

Proposition

Let a generative model G be given. Then the corresponding optimal discriminator D_g^* satisfies

$$D_g^*(x) = \frac{p_{data}(x)}{p_g(x) + p_{data}(x)} \tag{2}$$

Theorem

Global optimum for the outer optimization leads to a pair G, D such that $p_g = p_{data}$

Note that $D_g^* = 1/2$ when $p_g = p_{\text{data}}$.

Generative Adversarial Nets

- Function G replaced by a neural network
- Function D replaced by a neural network
- The resulting generative-adversarial system is referred to as *Adversarial Nets*
- Main advantage of Adversarial Nets over other generative models (deep belief networks, autoencoders, etc.): simple training and generation

GAN Development and Key Concepts and Ideas

GAN Development and Key Concepts and Ideas

- First GAN
- Deep Convolutional (DC) GAN
- Improvements to First GAN
- Energy based GAN
- Conditional GANs
  - Auxiliary Classifier GAN
  - Conditional GANs with Projection Discriminator
- Spectral Normalization
- Self Attention GAN
- OT GAN

First GAN

Generative Adversarial Nets, Goodfellow et al., NIPS 2014

- NN architecture for the two functions D and G: multi-layer perceptron
- Algorithm: Minibatch stochastic gradient descent iterative training

Theorem

If G and D have enough capacity, and at each step of Algorithm, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to optimize the criterion

$$\mathbb{E}_{x\sim p_{data}}[\log D_g^*(x)] + \mathbb{E}_{x\sim p_g}[\log(1 - D_g^*(x))] \tag{3}$$

then p_g converges to p_{data}.

Deep Convolutional (DC) GAN

Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, Radford, Metz, and Chintala, 2015.

- The first GAN used multilayer perceptrons
- Image synthesis: it was observed to generate samples that were incomprehensible, wobbly or didn't make sense
- Can CNNs which have been successful in image classification be used instead?
- Early attempts to combine CNNs and GANs were unsuccessful, ex: LAPGAN, 2015
- DCGAN: specific deep convolutional architecture that works for adversarial nets

DCGAN features

- All convolutional net (no intermediate max pooling layers as in CNNs)
- Top layers are not fully connected
- Batch normalization training (input to each layer is normalized to zero mean and unit variance) for both G and D
- RELU (rectified linear unit) activation in all layers of generator except output layer, which uses tanh
- LEAKY-RELU in all layers of discriminator. [Leaky RELU has a small slope for negative values, instead of zero as in RELU.]

Further Improvements to GAN

Improved Techniques for Training GANs, Salimans et al. (2016)

- Gradient descent algorithms not guaranteed to converge for min-max games
- Salimans et al. (2016) proposed several additional modifications to improve convergence:
  - Feature matching
  - Minibatch discrimination
  - Historical averaging
  - One sided smoothing
  - Virtual batch normalization
- Achieved the state-of-the-results then for semi-supervised learning

Feature Matching

- The generator is trained to match hidden layer features of the discriminator instead of trying to directly maximize the discriminator's output.
- Generator optimization problem:

$$\min_G \|\mathbb{E}_{x\sim p_{\text{data}}}[f(x)] - \mathbb{E}_{z\sim p_z}[f(G(z))]\|_2 \tag{4}$$

  where $f(x)$ denotes the hidden layer features of the discriminator
- Discriminator optimization problem:

$$\max_D \mathbb{E}_{x\sim p_{\text{data}}} \log[D(x)] + \mathbb{E}_{z\sim p_z} \log(1 - D(G(z))) \tag{5}$$

Minibatch Discrimination

- A well known failure mode in the training of a GAN is the so-called "generator collapse". Generator collapse results in generated outputs that are very similar.
- Minibatch Discrimination: A hidden layer of the discriminator is augmented with a vector of values that measures the closeness of the current input and other samples in the minibatch
- Minibatch Discrimination is observed to improve convergence.

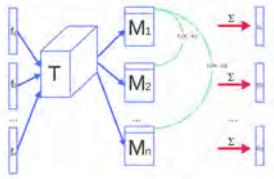


Figure 1: Figure sketches how minibatch discrimination works. Features $\mathbf{f}(\boldsymbol{x}_i)$ from sample $\boldsymbol{x}_i$ are multiplied through a tensor T, and cross-sample distance is computed.

source: Salimans et. al. 2016

Historical Averaging

- Discriminator's cost is modified to include: $\left\|\theta_d - \frac{1}{T}\sum_i \theta_d[i]\right\|_2$
- Generator's cost is modified to include: $\left\|\theta_g - \frac{1}{T}\sum_i \theta_g[i]\right\|_2$
- This modification to the cost is observed to improve convergence to the Nash equilibrium in certain two player games
- Related to fictitious play in games

One-Sided Smoothing

- Label smoothing is an old technique going back to the 1980's. The idea is to replace the classifier output values 1 and 0 by $\alpha = 0.9$ and $\beta = 0.1$ respectively.
- Double-sided smoothing for classifiers was shown to reduce vulnerability to adversarial examples by Szegedy et. al. 2018
- Double-sided smoothing leads to the optimal discriminator:

$$D^*(x) = \frac{\alpha p_{\text{data}} + \beta p_g}{p_{\text{data}} + p_g}$$

- One-sided smoothing refers to replacing only 1 by α and leaving 0 unchanged. It is better suited for adversarial training, and in this case:

$$D^*(x) = \frac{\alpha p_{\text{data}}}{p_{\text{data}} + p_g}$$

Virtual Batch Normalization

- Batch normalization (BN): normalizes a data point by statistics collected from the mini-batch
- Virtual batch normalization (VBN): normalizes a data point by statistics collected from a fixed set of points, defined at the start of the training
- VBN
  - Retains benefit of BN
  - Reduces dependency of the training on other samples, which would have been the case with BN

Energy Based (EB) GAN

Energy-based Generative Adversarial Network, Zhao, Mathieu, and LeCun, ICLR, 2017

- Discriminator is modeled as an energy function that assigns low energy to samples from real data and high energy to sample from the generator.
- More general version of earlier probabilistic GANs. Discriminator can be modeled as a general energy function than just a probabilistic binary discriminator, as in previous versions.
- "Viewing the discriminator as an energy function allows to use a wide variety of architectures and loss functionals in addition to the usual binary classifier with logistic output."

Formulation

- Discriminator loss function,

$$\mathcal{L}_D(x, z) = D(x) + [m - D(G(z))]_+ \tag{6}$$

  m: a constant

- Generator loss function,

$$\mathcal{L}_G(z) = D(G(z)) \tag{7}$$

EB GAN - Theoretical Result

EBGAN formulation can be interpreted as a two-player game. Let us define

$$V(G, D) = \int_{x,z} \mathcal{L}_D(x, z) p_z(z) p_{\text{data}}(x) dx dz$$

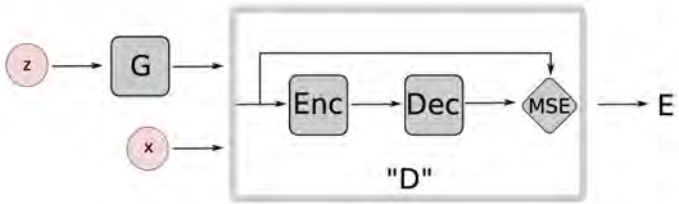
Then Zhao et al. (2017) prove the following theorem.

Theorem

If (D^*, G^*) is a Nash equilibrium of the system then $p_{g^*} = p_{data}$ almost everywhere, $m = V(D^*, G^*)$. Moreover, a Nash equilibrium exists. The Nash equilibrium of the game (D^*, G^*) is characterized by (a) $p_{g^*} = p_{data}$ almost everywhere and (b) there exists $\gamma \in [0, m]$ such that $D^*(x) = \gamma$.

Discriminator as an Auto-encoder

EBGAN provides a richer framework: discriminator can be modeled by more general architectures, for example: auto-encoder



$$D(x) = \|Dec(Enc(x)) - x\|$$

Generative Model is like a Regularizer

- Training auto-encoders requires regularization to avoid zeroing out of the function on the whole space.
- The generative model in the EBGAN framework provides the negative examples for the auto-encoder training, acting as a regularizer.

Avoiding Generator Collapse

- To avoid generator collapse, EBGAN framework uses a repelling regularizer term in the generator loss.
- Repelling regularizer term:

$$\frac{1}{N(N-1)} \sum_i \sum_j \frac{S_i^T S_j}{\|S_i\| \, \|S_j\|} \tag{8}$$

  where N is the number of sample representation in a batch and S_i corresponds to the sample.
- This is equivalent to orthoganalizing the representation samples of the encoder.
- Recall that the GAN framework uses minibatch discrimination for the same purpose.

GANs so far...

- GANs can produce convincing image samples on datasets with low variability and low resolution
- But GANs struggle to generate globally coherent, high resolution samples - particularly from datasets with high variability
- Providing side information to GANs can improve synthesis, examples:
  - Class conditional GANs
  - Bounding box information
  - Labels

Conditional GANs

Auxiliary Classifier (AC)GAN

Conditional image synthesis with auxiliary classifier GANs, Odena, Olah, and Shlens, ICML 2017

- Motivation: forcing a model to perform additional tasks is known to improve performance on the original task (Ex. (Sutskever et al., 2014; Szegedy et al., 2014; Ramsundar et al., 2016)).
- Idea: use class label in the output and input of discriminator and generator respectively
- Generator takes as input a class label c in addition to a sample of the noise variable z
- Discriminator estimates the probability of the class of the input sample in addition to whether the sample is real or fake
- This simple modification is reported to improve training and produce good results
- Achieves improved performance on CIFAR-10 dataset (to be discussed later) without incorporation of minibatch discrimination, historical averaging and one-sided smoothing

Formulation

- Generation model $G(c, z)$, where c is a sample class label and z is the input noise variable.
- Discriminator models a probability distribution over the classes and the samples
- Define

$$L_S = \mathbb{E}[\log \mathbb{P}(S = real | x \sim p_{\text{data}}] + \mathbb{E}[\log \mathbb{P}(S = fake | x \sim p_{\text{g}})] \tag{9}$$

$$L_C = \mathbb{E}[\log \mathbb{P}(C = c | x \sim p_{\text{data}}] + \mathbb{E}[\log \mathbb{P}(C = c | x \sim p_{\text{g}})] \tag{10}$$

- Generator objective: $\max [L_C - L_S]$
- Discriminator objective: $\max [L_C + L_S]$

Architecture and Training

- Generator: series of deconvolution layers that transform z, c into an image
- Discriminator: deep convolutional network with a LEAKY-RELU nonlinearity
- ImageNet has 1000 classes. ACGAN is trained to produce 100 AC-GAN models, each on images from just 10 classes, for 50000 mini-batches of size 100.

Improved GAN vs. (AC)GAN

Figure 7. Samples generated from the ImageNet dataset. (Left) Samples generated from the model in (Salimans et al., 2016). (Right) Randomly chosen samples generated from an AC-GAN. AC-GAN samples possess global coherence absent from the samples of the earlier model.

Note the wobbly nature of the images generated by Salimans et al., 2016. GAN

Source: Odena et al. 2017

Conditional GANs with Projection Discriminator

cGANs with Projection Discriminator, Miyato and Koyama, ICLR, 2018

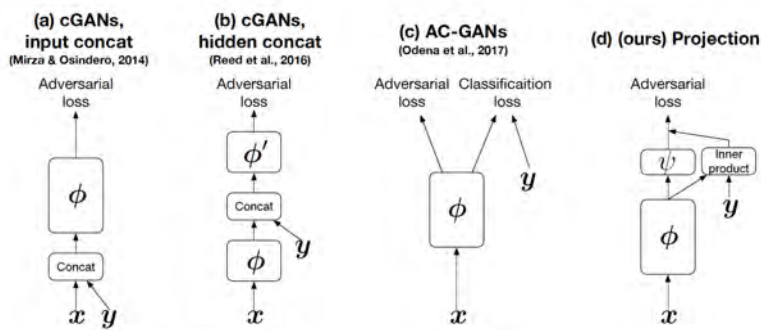
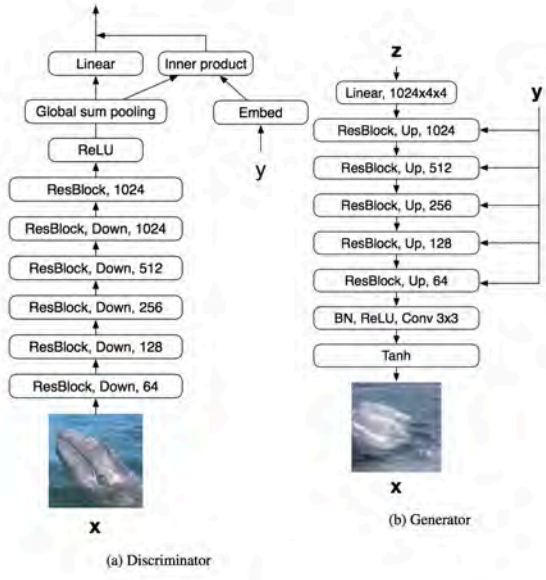


Figure 1: Discriminator models for conditional GANs

Discriminator and Generator Architecture



Justification and Motivation for Projection Discriminator

- For a given generator model, Miyato and Koyama show that the optimal solution to the discriminator objective function has a "projection structure". They show this for discrete log linear distributions in case of y being a categorical variable and unimodal distribution, e.g., Gaussian distribution, for continuous y.
- The authors state: "how this specific projection discriminator implementation influences the training of the generator is still unclear."

Loss Functions for cGAN

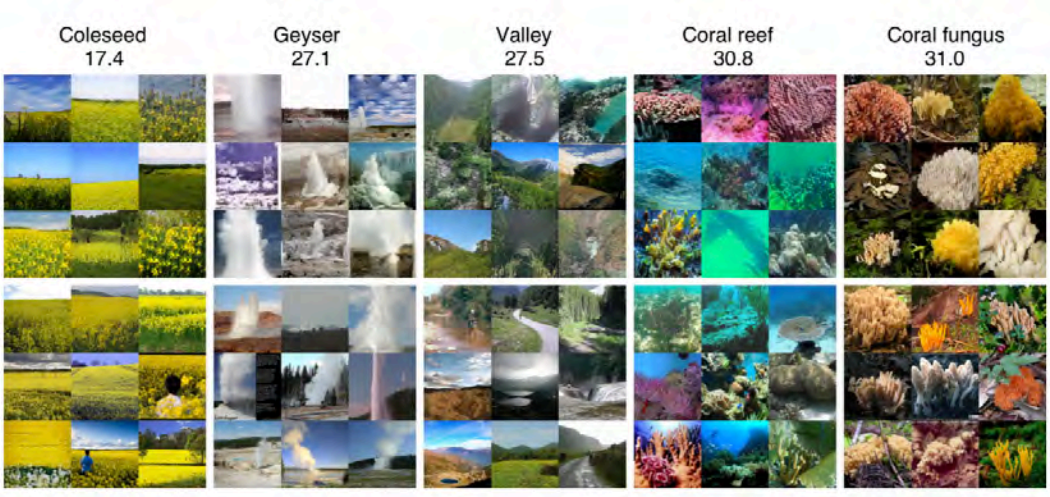
- Loss function for the Discriminator,

$$\mathcal{L}_D = -\mathbb{E}_{y\sim p_{\text{data}}}[\mathbb{E}_{x\sim p_{\text{data}}(x|y)}[\log(D(x))]] - \mathbb{E}_{y\sim p_g}[\mathbb{E}_{x\sim p_g(x|y)}[\log(1-D(x))]] \quad (11)$$

- Loss function for the Generator,

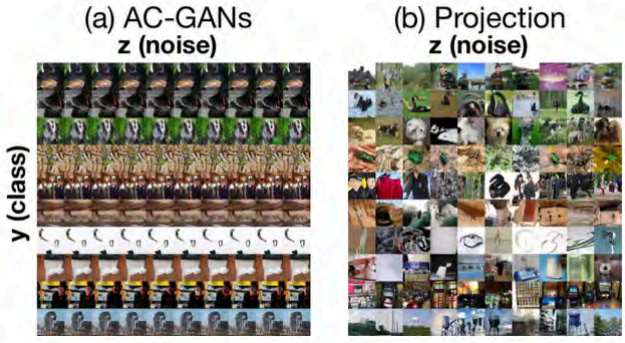
$$\mathcal{L}_G = \mathbb{E}_{y\sim p_g}[\mathbb{E}_{x\sim p_g(x|y)}[\log(1-D(x))]] \quad (12)$$

cGAN Generated Images



Source: Miyato et al. 2018

cGAN vs ACGAN



Note the generator collapse phenomenon in ACGAN

Source: Miyato et al., 2018

Image Synthesis: Measuring Performance

- Difficult to evaluate how "good" a generative model is.
- Inception Score: $\exp\left(\mathbb{E}_x \mathrm{KL}(p(y|x)||p_y)\right)$ where $p(y|x)$ is the conditional distribution of the label output of the inception model (Szegedy et al. 2015) applied to the generated image x. KL refers to the KL divergence. The score captures both the accuracy and the diversity of the generation model.
- Frechet Inception Distance (FID): Wasserstein's distance between distribution of the feature vectors generated by inception model applied to real data and the generated images

Inception and FID Scores Comparison

Table 1: Inception score and intra FIDs on ImageNet.

Method	Inception Score	Intra FID
AC-GANs	28.5±.20	260.0
concat	21.1±.35	141.2
projection	**29.7**±.61	**103.1**
*projection (850K iteration)	**36.8**±.44	**92.4**

Source: Miyato et al., 2018

Spectral Normalization (SN) GAN

Spectral Normalization (SN) GAN

Spectral Normalization for Generative Adversarial Networks, Miyato et al.,ICLR, 2018

- A significant challenge in adversarial nets is the instability of training.
- "A persisting challenge in the training of GANs is the performance control of the discriminator. In high dimensional spaces, the density ratio estimation by the discriminator is often inaccurate and unstable during the training, and generator networks fail to learn the multimodal structure of the target distribution."
- Spectral normalization is a *regularization method* for improving convergence in GANs.
- Easy to incorporate in current GAN methods.

Idea: Spectral Normalization Controls Lipschitz Constant

- A number of papers, e.g., Uehara et al., 2016; Qi, 2017; Gulrajani et al., 2017, advocate the importance of controlling the Lipschitz constant of the discriminator.
- Spectral normalization technique controls the Lipschitz constant by constraining the spectral norm of every layer of the discriminator model

Background

- Denote the function that maps a layer's input to the layer's output by g
- The Lipschitz constant for this function is given by,

$$\|g\|_{\text{lip}} = \sup_{h} \sigma(\nabla g(h))$$

  where $\sigma(.)$ is the spectral norm
- Example, for a linear function $g = Wh$,

$$g_{\text{lip}} = \sigma(W) = \sqrt{\lambda_{max}[W^T W]}$$

- For two functions g_1 and g_2, the composition $g_1 \circ g_2$ satisfies:

$$\|g_1 \circ g_2\|_{\text{lip}} \leq \|g_1\|_{\text{lip}} \|g_2\|_{\text{lip}}$$

Spectral Normalization

- Spectral normalization: rescale the weight matrix W by its spectral norm
- Denote the rescaled weights by $\overline{W}$:

$$\overline{W} = \frac{W}{\sigma(W)}$$

- Thus, if $\overline{g} = \overline{W}h$, $\overline{g}_{\text{lip}} = 1$
- Thus, for $\overline{g}_1 = \overline{W}_1 h$, $\overline{g}_2 = \overline{W}_2 h$, $\|\overline{g}_1 \circ \overline{g}_2\|_{\text{lip}} \leq 1$

Advantages

- Other normalization methods like weight normalization (scaling each column by its ℓ_2 norm) impose a much stronger constraint on the weight matrix, effectively constraining the number of features
- By comparison spectral normalization does not impose any such constraint because the normalization is only dependent on the spectral norm

Other Normalization Methods for GANs

- Orthonormal regularization: includes $\left\|W^T W - I\right\|_F$ in the objective. Restrictive on the spectrum unlike SN
- Weight normalization: scale each column of the weight matrix by its l_2 norm
- Batch normalization: normalizes based on the statistics of a batch
- Layer normalization: normalizes based on the statistics of a layer
- Gradient penalty: a penalty in the objective

Comparison of Normalization Methods

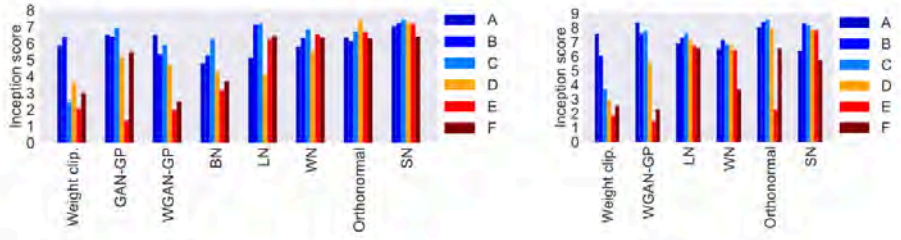


Figure 1: Inception scores on CIFAR-10 and STL-10 with different methods and hyperparameters (higher is better).

Source: Miyato et al. '18

Normalization Methods on CIFAR 10

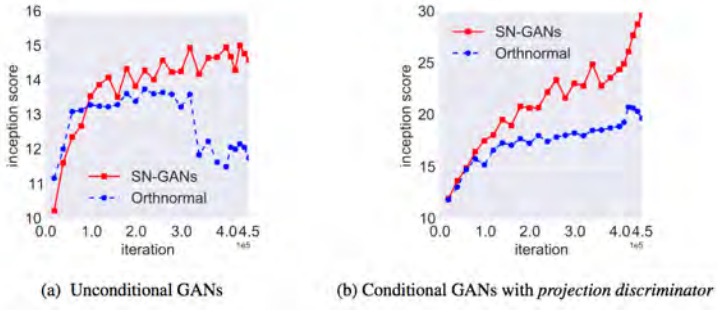
Dataset: CIFAR-10, STL-10

Method	Inception score		FID	
	CIFAR-10	STL-10	CIFAR-10	STL-10
Real data	11.24±.12	26.08±.26	7.8	7.9
-Standard CNN-				
Weight clipping	6.41±.11	7.57±.10	42.6	64.2
GAN-GP	6.93±.08		37.7	
WGAN-GP	6.68±.06	8.42±.13	40.2	55.1
Batch Norm.	6.27±.10		56.3	
Layer Norm.	7.19±.12	7.61±.12	33.9	75.6
Weight Norm.	6.84±.07	7.16±.10	34.7	73.4
Orthonormal	7.40±.12	8.56±.07	29.0	46.7
(ours) SN-GANs	7.42±.08	8.28±.09	29.3	53.1
Orthonormal (2x updates)		8.67±.08		44.2
(ours) SN-GANs (2x updates)		8.69±.09		47.5
(ours) SN-GANs, Eq.(17)	7.58±.12		25.5	
(ours) SN-GANs, Eq.(17) (2x updates)		8.79±.14		43.2
-ResNet-[5]				
Orthonormal, Eq.(17)	7.92±.04	8.72±.06	23.8±.58	42.4±.99
(ours) SN-GANs, Eq.(17)	**8.22**±.05	**9.10**±.04	**21.7**±.21	**40.1**±.50
DCGAN[†]	6.64±.14	7.84±.07		
LR-GANs[‡]	7.17±.07			
Warde-Farley et al.[*]	7.72±.13	8.51±.13		
WGAN-GP (ResNet)[††]	7.86±.08			

Source: Miyato et al. 2018

Performance on ImageNet

Dataset: ImageNet



Right: SN is combined with cGAN with Proj. Disc.

Source: Miyato et al. 2018

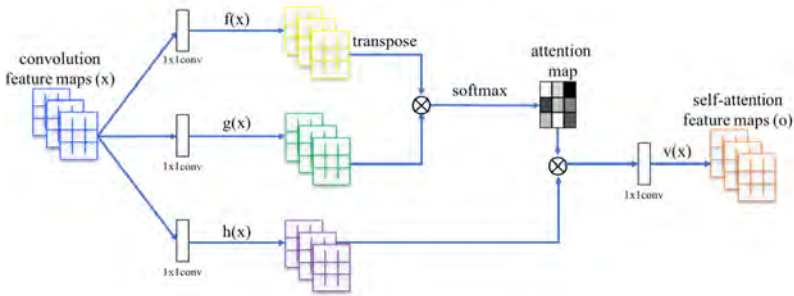
Self Attention (SA) GAN

Self-Attention GAN (SAGAN)

Self-Attention Generative Adversarial Networks, Zhang et al., 2019

- The most successful GANs use convolutional layers
- Long range dependencies are possibly captured only after passing throuh several layers and backpropagation training may not effectively capture such dependencies
- Attention (originally proposed by Vaswani et al., '17): "calculates response at a position as a weighted sum of the features at all positions, where the weights – or attention vectors – are calculated with only a small computational cost"
- Self-attention (SA) GAN: combines a self-attention mechanism and a GAN
- Impressive improvements: "SAGAN significantly outperforms prior work in image synthesis by boosting the best reported Inception score from 36.8 to 52.52 and reducing Frechet Inception distance from 27.62 to 18.65"

Self-Attention Layer



Source: Zhang et al. '19

Self-Attention Illustration

Figure 1. The proposed SAGAN generates images by leveraging complementary features in distant portions of the image rather than local regions of fixed shape to generate consistent objects/scenarios. In each row, the first image shows five representative query locations with color coded dots. The other five images are attention maps for those query locations, with corresponding color coded arrows summarizing the most-attended regions.

Source: Zhang et al. '19

Hinge Loss for GAN

Hinge loss for GAN (Lim & Ye, 2017; Tran et al., 2017; Miyato et al., 2018)

$$\mathcal{L}_D = -\mathbb{E}_{x\sim p_{\text{data}}}[\min\{0, -1 + D(x)\}] - \mathbb{E}_{z\sim p_z}[\min\{0, -1 - D(G(z))\}]$$

$$\mathcal{L}_G = -\mathbb{E}_{z\sim p_z} D(G(z))$$

SAGAN Implementation

- Both generators and discriminators are implemented with self-attention (Wang et al. 2018)
- Spectral normalization for both discriminator and generator
- Two timescale update rule (TTUR) of Heusel et al., 2017
- Hinge loss version of GAN (Lim & Ye, 2017; Tran et al., 2017; Miyato et al., 2018)

$$\mathcal{L}_D = -\mathbb{E}_{x\sim p_{\text{data}}}[\min\{0, -1 + D(x)\}] - \mathbb{E}_{z\sim p_z}[\min\{0, -1 - D(G(z))\}]$$

$$\mathcal{L}_G = -\mathbb{E}_{z\sim p_z} D(G(z))$$

Other GAN formulations

- Cramer GAN
- Optimal Transport (OT) GAN: general version of Cramer GAN
- Both formulations learn the generator model by optimizing a measure of distance (critic) between the generative distribution and the real data distribution evaluated over an adversarially learned latent space
- Statistically consistent distance metric
- Requirement: sufficient mini batch size

Convergence in OT-GAN

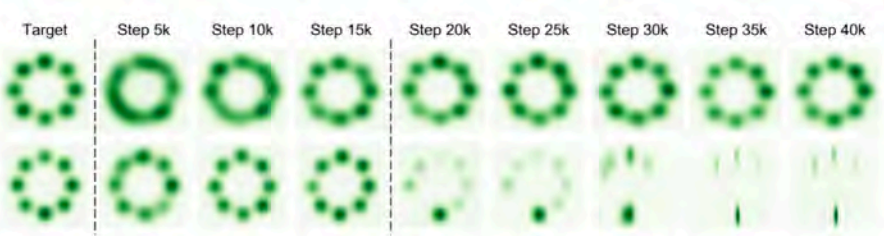


Figure 2: Results for consistency when fixing the critic on data generated from 8 Gaussian mixtures. The first column shows the data distribution. The top row shows the training results of OT-GAN using mini-batch energy distance. The bottom row shows the training result with the original GAN loss (DAN-S). The latter collapses to 3 out of 8 modes after fixing the discriminator, while OT-GAN remains consistent.

Application of GANs

Applications

- Image Synthesis
- Super Resolution
- Image 2 Image Translation
- Text 2 Image Translation
- Semi Supervised Learning
- Video Prediction
- Self Supervised Learning for Control

Image Synthesis: Datasets

- ImageNet is an image database organized according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. Currently there are an average of over five hundred images per node
- CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images
- STL-10 is a Stanford image dataset for unsupervised learning. Fewer labeled training examples compared to CIFAR-10 but a large number of unlabled examples
- MNIST dataset: database of handwritten digits

Image Synthesis: Best Models

State-of-the-art on large scale complex dataset, ImageNet

Model	Inception Score	Intra FID	FID
AC-GAN (Odena et al., 2017)	28.5	260.0	/
SNGAN-projection (Miyato & Koyama, 2018)	36.8	92.4	27.62*
SAGAN	**52.52**	**83.7**	**18.65**

Source: Zhang et al. 2019

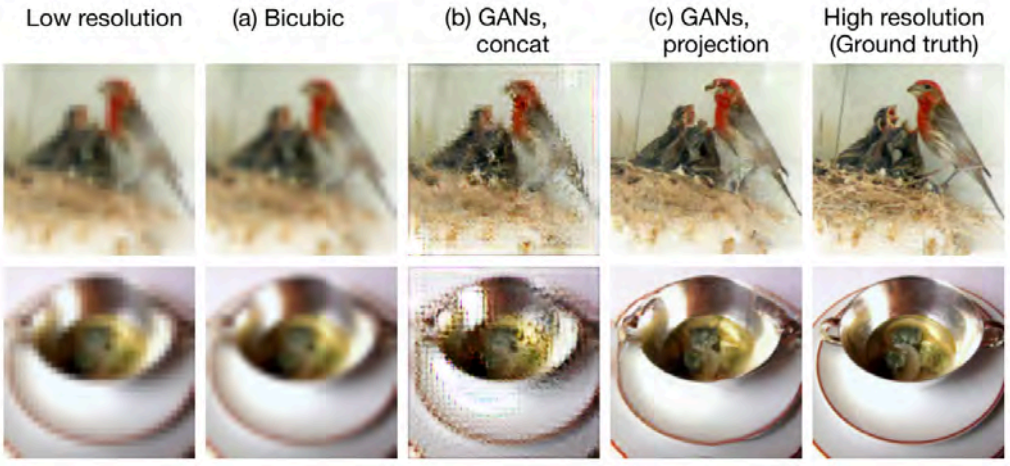
Conditional GANs on CIFAR 10

cGAN with Projection Discriminator is state-of-the-art on CIFAR 10

Method	Inception score		FID	
	C10	C100	C10	C100
(Real data)	11.24	14.79	7.60	8.94
AC-GAN	8.22	8.80	19.7	25.4
input concat	8.25	7.93	19.2	31.4
hidden concat	8.14	8.82	19.2	24.8
(ours) projection	**8.62**	**9.04**	**17.5**	**23.2**

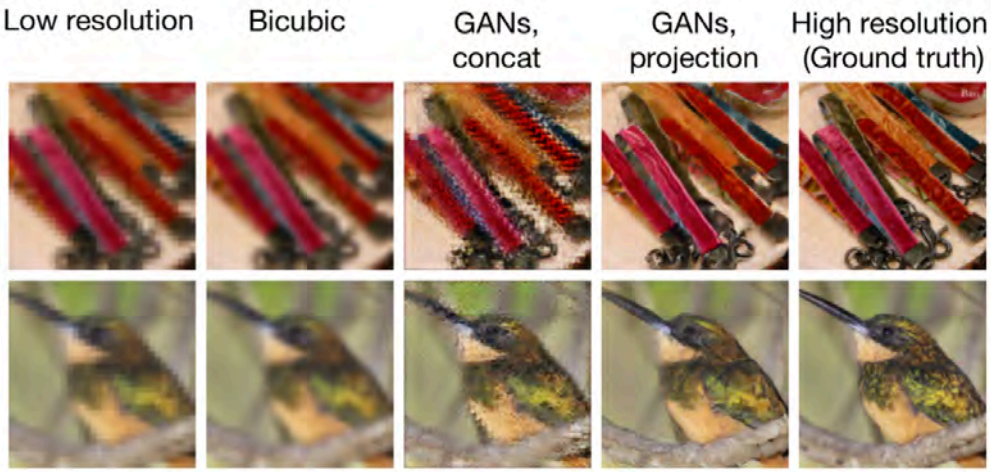
Source: Miyato et al. 2018

Super Resolution



Source: Miyato et al. 2018

Super Resolution



Source: Miyato et al. 2018

Super Resolution GAN

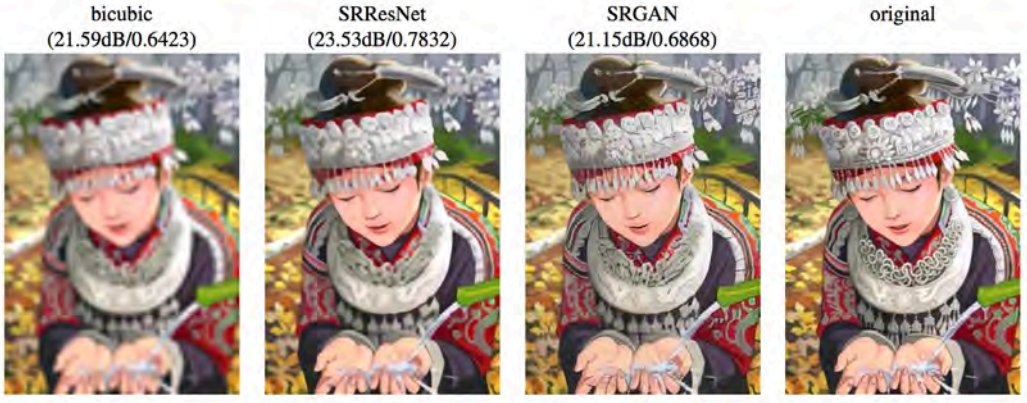
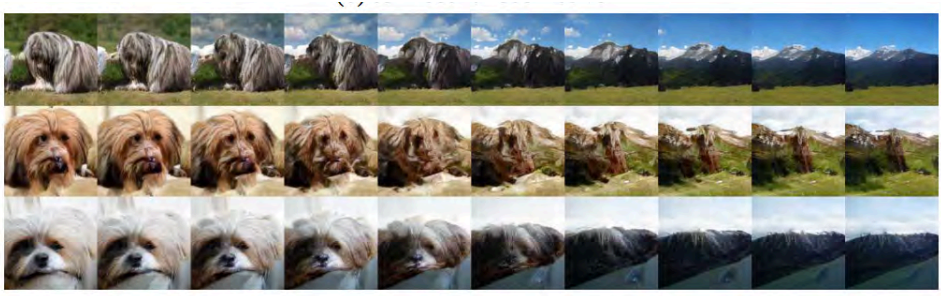


Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

Source: Ledig et al. 2017

Category Morphing



Source: Miyato et al 2018

Category Morphing

Source: Miyato et al. 2018

Image 2 Image Translation Model, Zhu et al. 2017

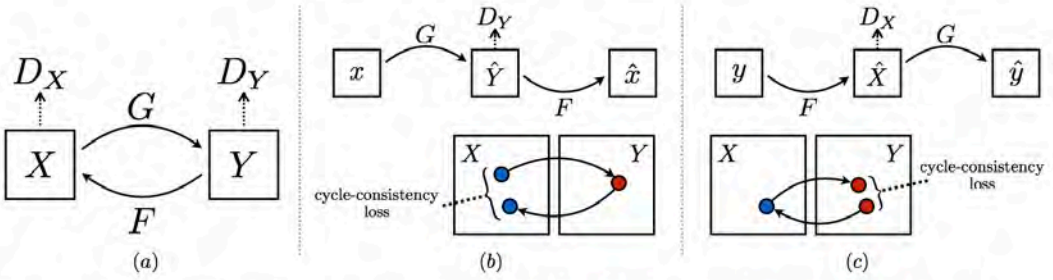


Figure 3: (a) Our model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X, F, and X. To further regularize the mappings, we introduce two "cycle consistency losses" that capture the intuition that if we translate from one domain to the other and back again we should arrive where we started: (b) forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

Source: Zhu et al. 2017

Image 2 Image Illustration

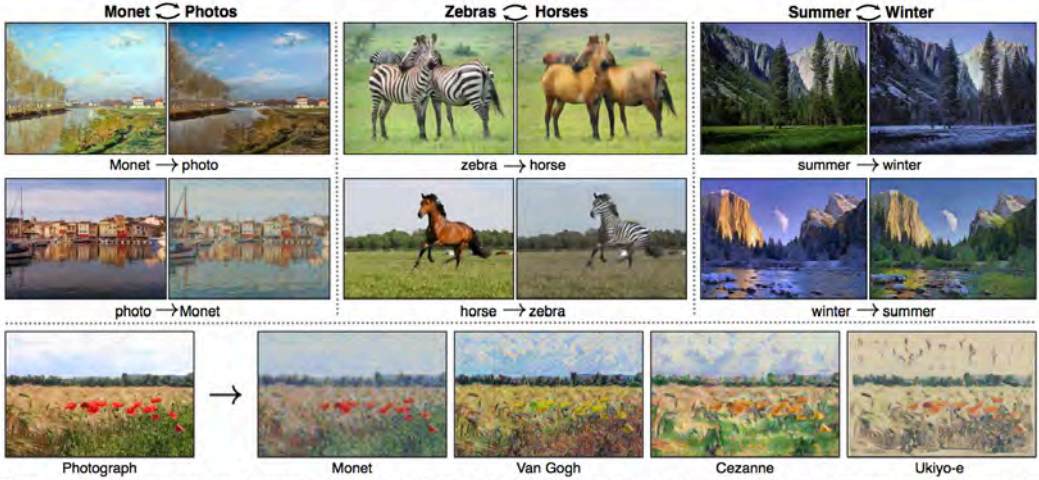


Figure 1: Given any two unordered image collections X and Y, our algorithm learns to automatically "translate" an image from one into the other and vice versa. Example application (*bottom*): using a collection of paintings of a famous artist, learn to render a user's photograph into their style.

Source: Zhu et al. 2017

Text 2 Image, Stack GAN, Zhang et al. 2017

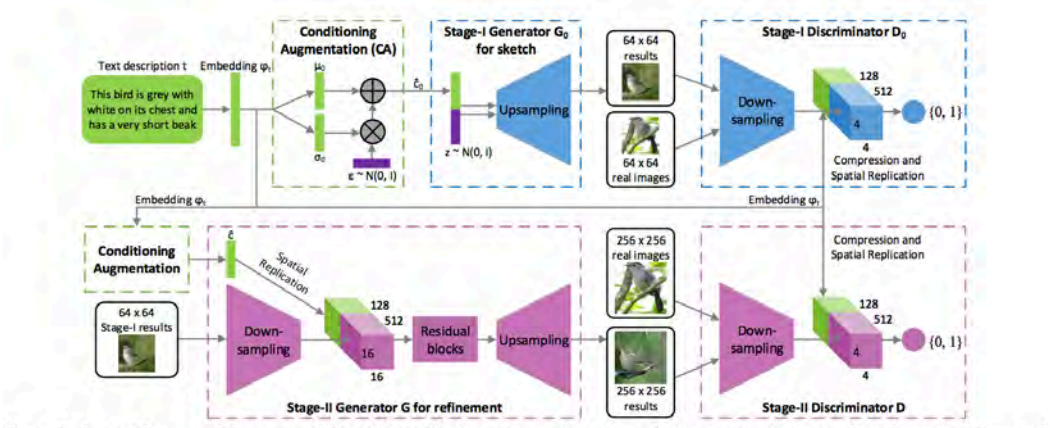


Figure 2. The architecture of the proposed StackGAN. The Stage-I generator draws a low-resolution image by sketching rough shape and basic colors of the object from the given text and painting the background from a random noise vector. Conditioned on Stage-I results, the Stage-II generator corrects defects and adds compelling details into Stage-I results, yielding a more realistic high-resolution image.

Source: Zhang et al. 2017

Text 2 Image Illustration

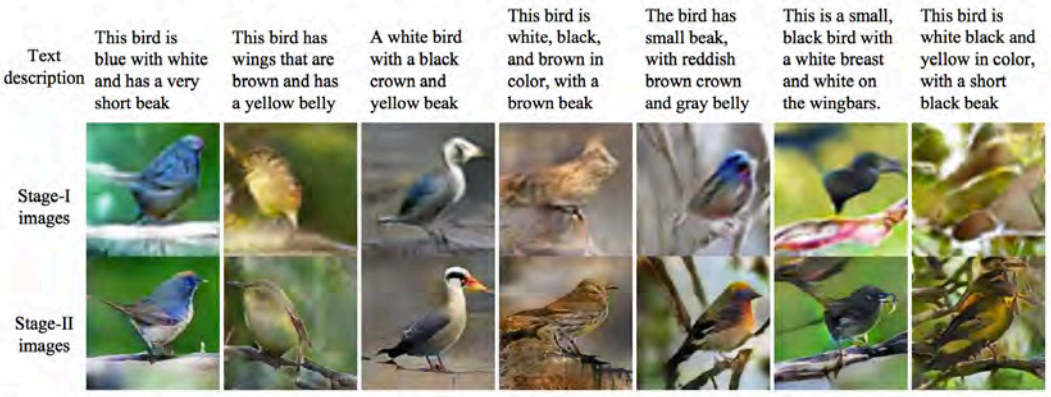
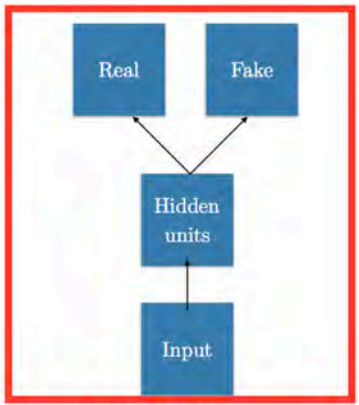


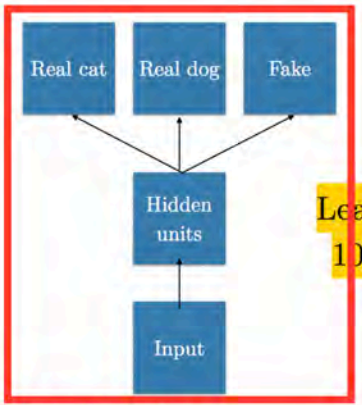
Figure 5. Samples generated by our StackGAN from unseen texts in CUB test set. Each column lists the text description, images generated from the text by Stage-I and Stage-II of StackGAN.

Source: Zhang et al. 2017

Semi-supervised Learning Framework, Salimans et al. 2016



Discriminator in Unsupervised Learning



Discriminator in Semi-supervised Learning

Learn to read with 100 labels rather than 60,000

Source: Goodfellow

Semi-supervised Learning

Performance of Improved GAN on MNIST, Salimans et al. 2016

Model	Number of incorrectly predicted test examples for a given number of labeled samples			
	20	50	100	200
DGN [22]			333 ± 14	
Virtual Adversarial [23]			212	
CatGAN [14]			191 ± 10	
Skip Deep Generative Model [24]			132 ± 7	
Ladder network [25]			106 ± 37	
Auxiliary Deep Generative Model [24]			96 ± 2	
Our model	1677 ± 452	221 ± 136	93 ± 6.5	90 ± 4.2
Ensemble of 10 of our models	1134 ± 445	142 ± 96	86 ± 5.6	81 ± 4.3

Table 1: Number of incorrectly classified test examples for the semi-supervised setting on permutation invariant MNIST. Results are averaged over 10 seeds.

Semi-supervised Learning

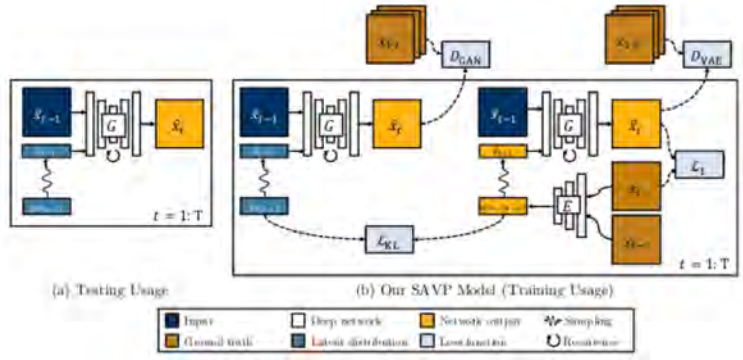
Performance of Improved GAN on CIFAR-10, Salimans et al. 2016

Model	Test error rate for a given number of labeled samples			
	1000	2000	4000	8000
Ladder network [25]			20.40±0.47	
CatGAN [14]			19.58±0.46	
Our model	21.83±2.01	19.61±2.09	18.63±2.32	17.72±1.82
Ensemble of 10 of our models	19.22±0.54	17.25±0.66	15.59±0.47	14.87±0.89

Table 2: Test error on semi-supervised CIFAR-10. Results are averaged over 10 splits of data.

Video Prediction

Stochastic Adversarial Video Prediction (SAVP); Lee et al., 2018



SAVP Framework

- GAN framework + VAE framework
- GAN loss function, $\mathcal{L}_{\text{GAN}}$:

$$\mathcal{L}_{\text{GAN}} = \mathbb{E}_{x_{1:T}}[\log(D(x_{1:T}))] + \mathbb{E}_{x_{1:T}, z_t \sim p_z(z_t)|_{t=0}^{t=T-1}}[\log(1 - D(G(z_{0:T-1})))]$$

- VAE loss funciton, $\mathcal{L}_{\text{VAE}}$:

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{x_{0:T}, z_t \sim E(x_{t:t+1})|_{t=0}^{T-1}} \left[\sum_{t=1}^{T} \|x_t - G(x_0, z_{0:t-1})\|_1 \right]$$

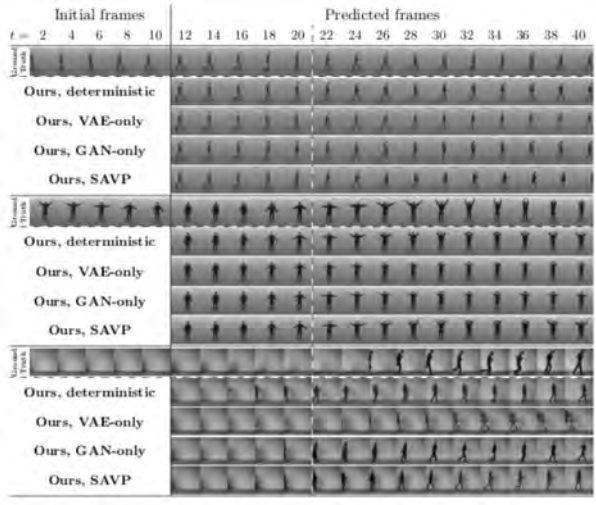
  where, $E(.)$: encoder that generates the latent variables
- Regularization term for the encoder,

$$\mathcal{L}_{KL}(E) = \mathbb{E}_{x_{0:T}} \left[\sum_{t=1}^{T} \mathcal{D}_{KL}(E(x_{t-1:t}) || p(z_{t-1})) \right]$$

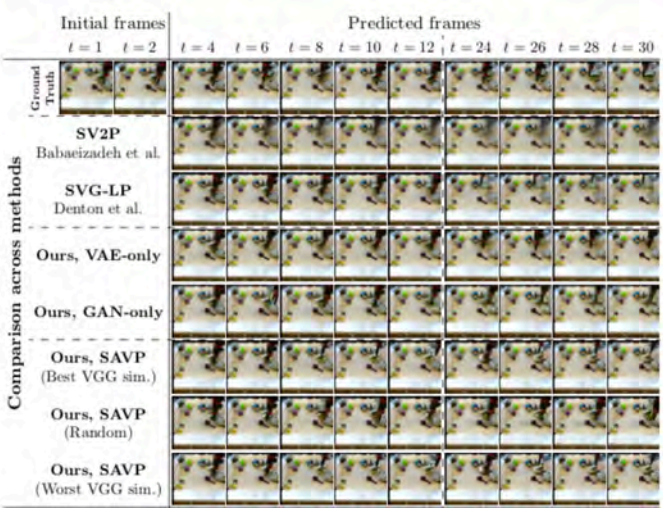
- SAVP Problem,

$$G^*, E^* = \arg\min_{G,E} \max_{D, D_{VAE}} \mathcal{L}_{\text{GAN}} + \lambda_{VAE} \mathcal{L}_{\text{VAE}} + \lambda_{KL} \mathcal{L}_{KL} + \mathcal{L}_{\text{GAN}}^{DAE}$$

Results for SAVP



Results for SAVP



Video Prediction in Semantic Segmentation Space

Predicting Deeper Into the Future of Semantic Segmentation, Luc et al, CVPR, 2017

Learns semantic-level scene dynamics through adversarial framework

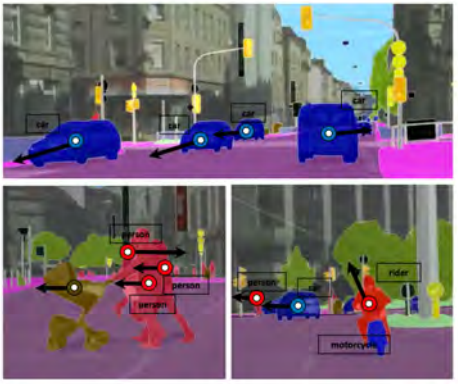
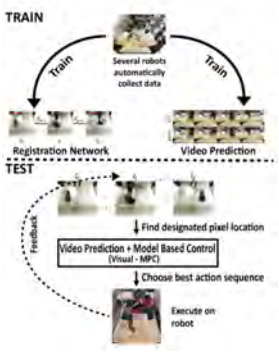


Figure 1: Our models learn semantic-level scene dynamics to predict semantic segmentations of unobserved future frames given several past frames.

Source: Luc et al '17

Self Supervised Learning for Robot Manipulation

Robustness via Retrying: Closed-Loop Robotic Manipulation with Self-Supervised Learning, Ebert et al, Conference on Robot Learning, 2018



Source: Ebert et al, '18

Self Supervised Learning for Robot Manipulation

- "Closed-loop visual MPC: Autonomously collected experience is used to train a video prediction model, as well as an image-to-image registration model, which enables the robot to keep track of its goal."
- A manipulation task is solved by a planning algorithm in conjunction with a registration model
- The planning algorithm uses a video prediction model. The video prediction module is trained by SAVP method of Lee et al
- The registration model is used for assessing the accuracy of long term predictions
- The video prediction model and the registration model are trained by autonomously collected real world image data and the training is unsupervised.

GAN for Personalized Manufacturing

Learning Beyond Human Expertise with Generative Models for Dental Restorations, Hwang et al., 2018

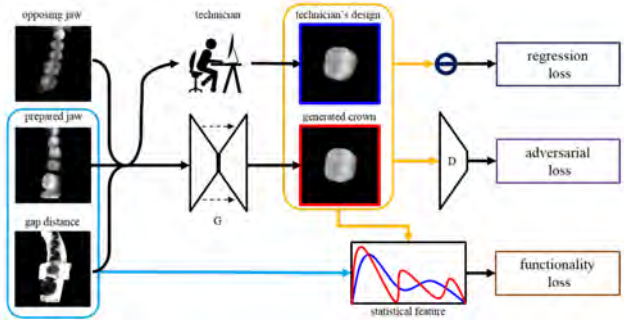


Fig. 2. Diagram illustration of the proposed model. We propose a functionality loss with space information (in the blue box) to learn the functionality of technician's designs. Please refer to Fig. 3 for the computation diagram of the functionality loss. (For all the crowns, we zoom in the images by a factor of 2 for better visualization.)

The GAN Zoo

"Every week, new GAN papers are coming out and it's hard to keep track of them all, not to mention the incredibly creative ways in which researchers are naming these GANs! So, here's a list of what started as a fun activity compiling all named GANs!"

Source: The GAN Zoo

Contact Information for Further Information

- email: pramod.khargonekar@uci.edu; khargonekar@gmail.com
- Faculty website
- LinkedIn Page
- Google Scholar page