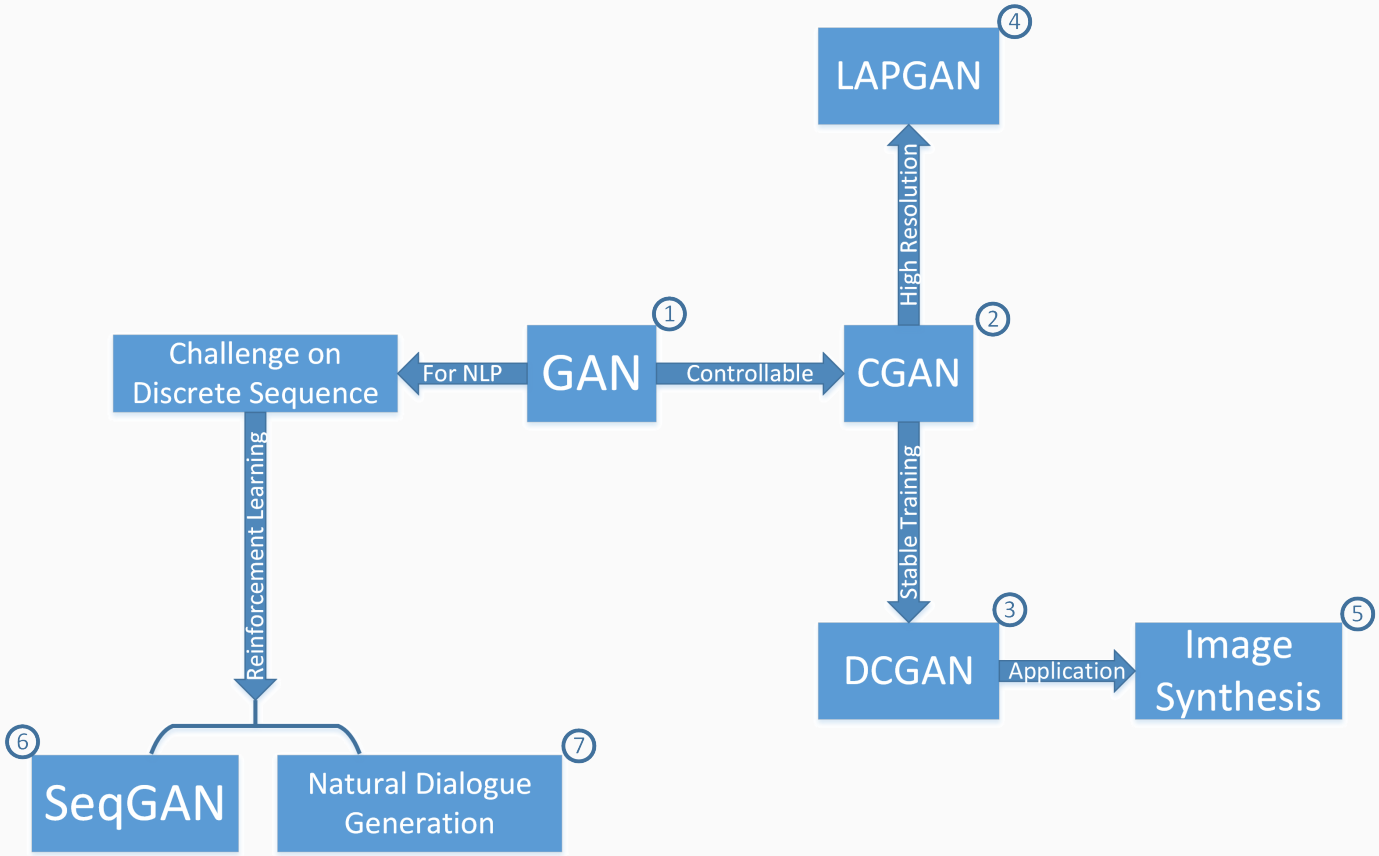
A Brief Review on GANs

From Computer Vision to Natural Language Processing

Bothan Shi
botianshi@bit.edu.cn

Apr, 13, 2017

Outline


LAPGAN
4
High Resolution
Challenge on Discrete Sequence
For NLP
GAN
1
Controllable
CGAN
2
Reinforcement Learning
Stable Training
DCGAN
3
Application
Image Synthesis
5
6
SeqGAN
Natural Dialogue Generation
7

Background Information

Generative Model and Discriminative Model

Generative Model and Discriminative Model

What is generative model?

- Modelling the generation distribution of data.
- Done by modelling a joint probability distribution $p(x, y)$.

What is discriminative model?

- Modelling the dependence of an unobserved variable y on an observed variable x.
- Done by modelling the conditional probability distribution $p(y|x)$.
- For tasks such as classification and regression that do not require the joint distribution, discriminative models can yield superior performance.

generative models modelling the data, discriminative models modelling the difference between data.

Generative Model and Discriminative Model

Why we need generative model?

- We can get a discriminative model via a generative model by $p(y|x) = \frac{p(x,y)}{p(x)}$, the other way not.
- We can **sample new data** from generative model whereas discriminative model cannot. (AMAZING!)

Is is easy to get a generative model?

- It is hard to estimate an probabilistic model.
- We need enough observed data + robust model.

How can we get a generative model?

- Gaussian mixture model and other types (GMM)
- Hidden Markov Model (HMM)
- Latent Dirichlet Allocation(LDA)
- Restricted Boltzmann Machine (RBM)
- **Generative Adversarial Networks (GAN)**

Generative Adversarial Nets

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu,
David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio.
NIPS 2014
arXiv: 1406.2661

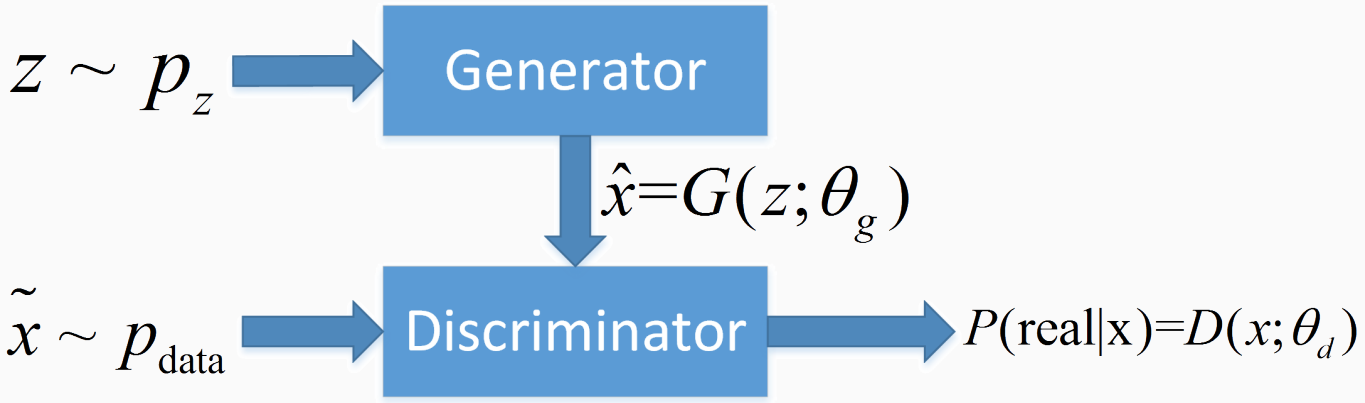
Introduction of GAN

- New framework for estimating generative models via an adversarial process.
- Train two models simultaneously:
  - A generative model G that captures the data distribution.
  - A discriminative model D that estimates the probability that a sample came from the training data rather than G.
- A minimax two-player zero-sum game.
- Unique solution exists (ensuring convergence): G recovering the training data distribution; $D = \frac{1}{2}$ everywhere.

Introduction of GAN

- The purpose of generative model is to learn data distribution p_g.
- We have a noise variable $p_z(z)$ first, and sample a $z \sim p_z(z)$.
- Synthesized data $\hat{x} = G(z; \theta_g)$. θ_g is the parameter of the generative model.
- If the θ_g is well trained, we will get a generative model $p_g \approx p_{\text{data}}$. ($p_g$ have learned the same mapping function as p_{data})
- In this case, we can generate new ”fake” data realistic enough to confuse human as well as the discriminator.
- AMAZING! Our model has learned the distribution of data and can generate new sample!
- For example: produce ”reasonable” images.

Adversarial nets



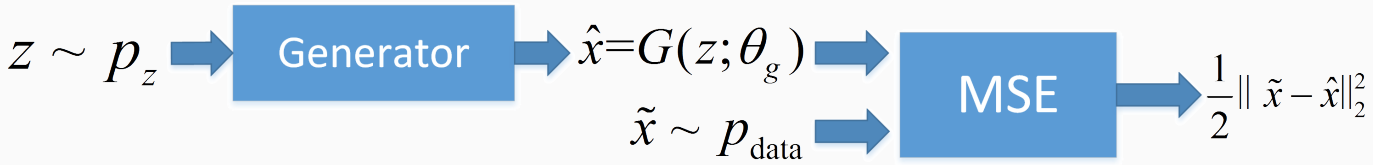
We train D to maximize the probability of assigning the correct label to both training examples and samples from G.

We simultaneously train G to minimize $\log(1 - D(G(\boldsymbol{z})))$. In other words, D and G play the following two-player minimax game with value function $V(G, D)$:

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

Adversarial nets

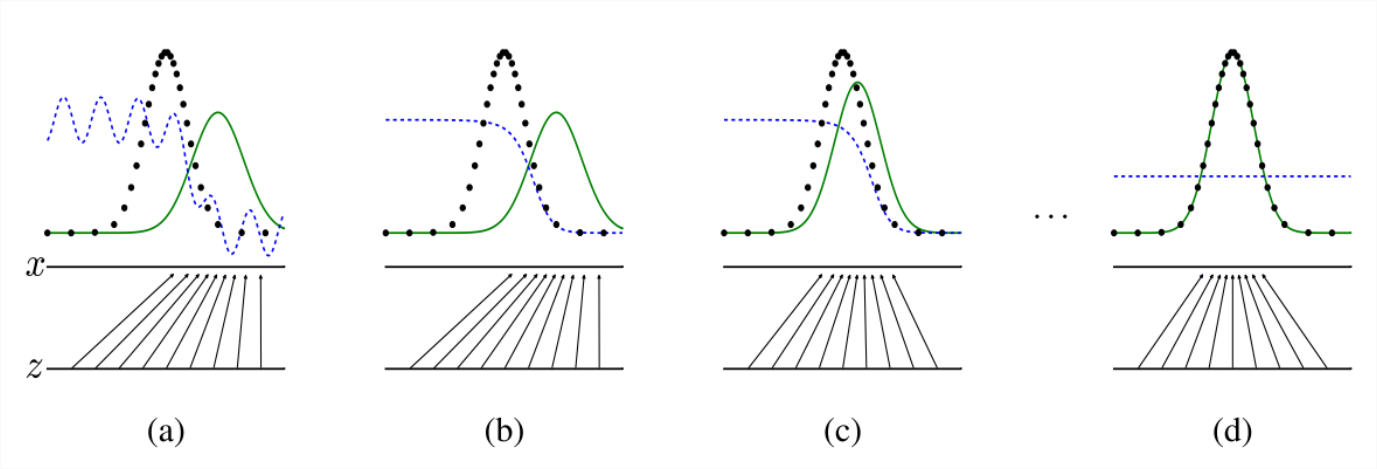
Why we need the discriminator? Why can't we define the loss function as the MSE of the real image and the image produced by the generator directly?



If we only have the generator, this model can be regarded as an unsupervised learning problem. But if we want the output image of the generator as similar as possible to the training sample, the generator will finally remember all the training data and it will not be able to produce any "new" images.

Adversarial nets

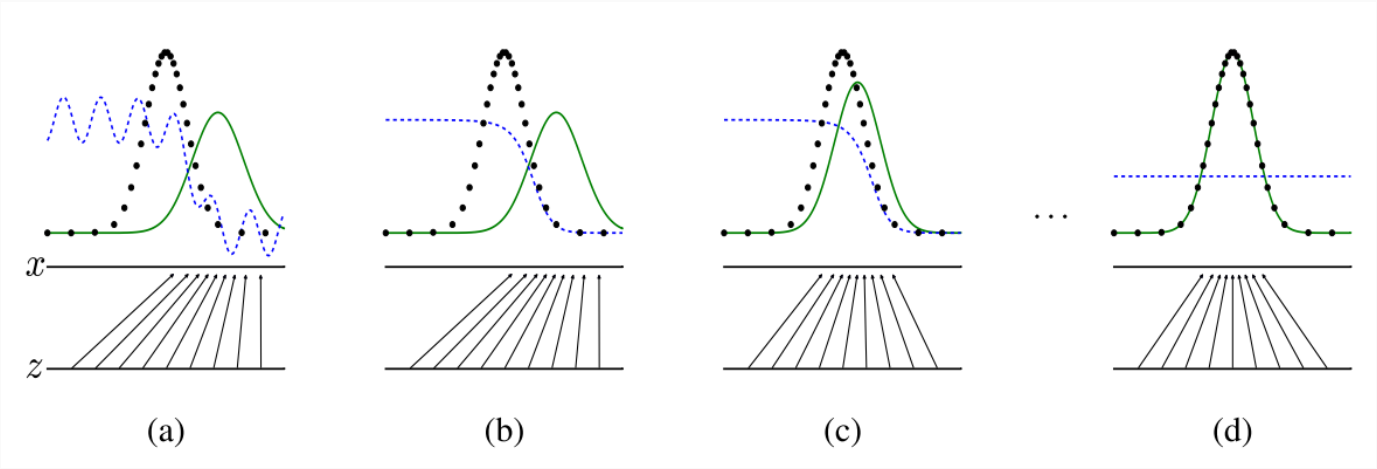
Here is a less formal, more pedagogical explanation of the approach:



Generative adversarial nets are trained by simultaneously updating the **d**iscriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) p_g from those of the **g**enerative distribution $p_g(G)$ (green, solid line).

Adversarial nets

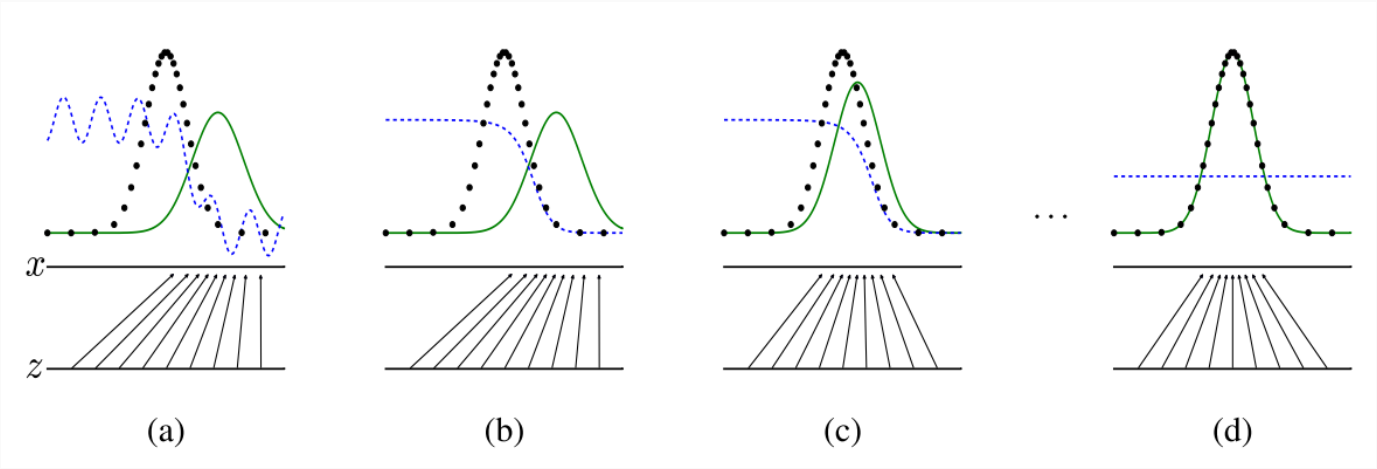
Here is a less formal, more pedagogical explanation of the approach:



The lower horizontal line is the domain from which $\boldsymbol{z}$ is sampled, in this case uniformly. The horizontal line above is part of the domain of $\boldsymbol{x}$. The upward arrows show how the mapping $\boldsymbol{x} = G(\boldsymbol{z})$ imposes the non-uniform distribution p_g on transformed samples.

Adversarial nets

Here is a less formal, more pedagogical explanation of the approach:

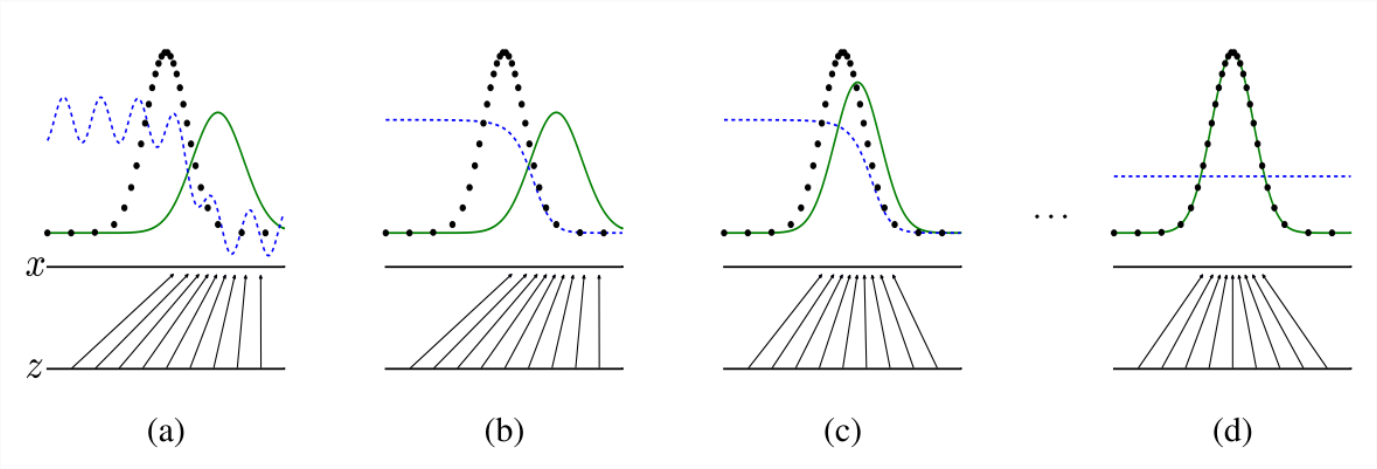


(a) Consider an adversarial pair near convergence: p_g is similar to p_{data} and D is a partially accurate classifier.

(b) In the inner loop of the algorithm D is trained to discriminate samples from data, converging to $D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x)+p_g(x)}$

Adversarial nets

Here is a less formal, more pedagogical explanation of the approach:

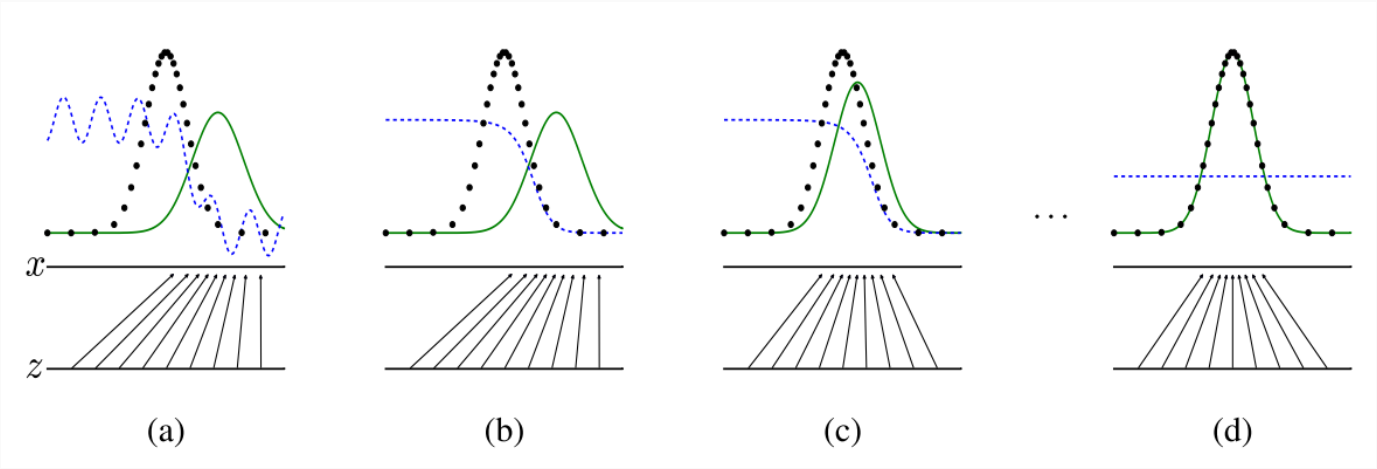


(c) After an update to G, gradient of D has guided $G(\boldsymbol{z})$ to flow to regions that are more likely to be classified as data.

(d) After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{\text{data}}$.

Adversarial nets

Here is a less formal, more pedagogical explanation of the approach:



The discriminator is unable to differentiate between the two distributions. i.g. $D(x) = \frac{1}{2}$

Adversarial nets

- There are several trifles we need to deal with. The first is insufficient gradient in the early learning.
- Let's take a look at the loss function again:

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

- In practice, this equation may not provide sufficient gradient for G to learn well.
- Early in learning, when G is poor, D can reject samples with high confidence because they are clearly different from the training data.
- In this case, $\log(1 - D(G(z)))$ saturates.
- Rather than training G to minimize $\log(1 - D(G(\boldsymbol{z})))$ we can train G to maximize $\log D(G(\boldsymbol{z}))$.

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

$$\max_G \mathbb{E}_{\boldsymbol{z}\sim p_z}[\log D(G(\boldsymbol{z}))]$$

Adversarial nets

- The second trifle is that we need to prevent the overfit of G.
- We should conduct the iterative training: optimize the discriminator k times and then train the generator once.

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets.

for number of training iterations **do**
  for k steps **do**
    Sample minibatch of m noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$
    Sample minibatch of m examples $x^{(1)}, \ldots, x^{(m)}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$
    Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right]$$

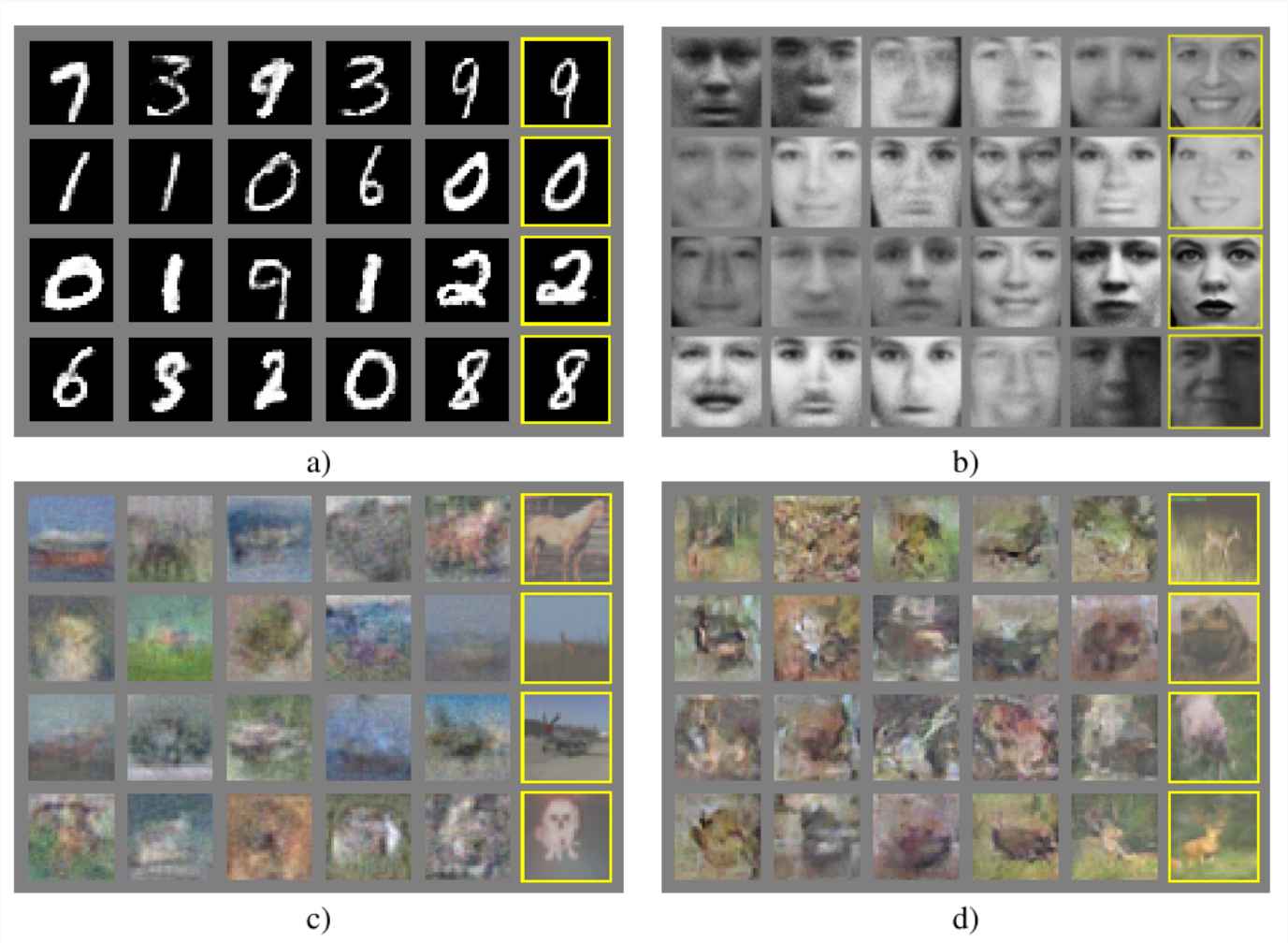
  end for
  Sample minibatch of m noise samples $z^{(1)}, \ldots, z^{(m)}$ from noise prior $p_g(\boldsymbol{z})$.
  Update the generator by descending its stochastic gradient:

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

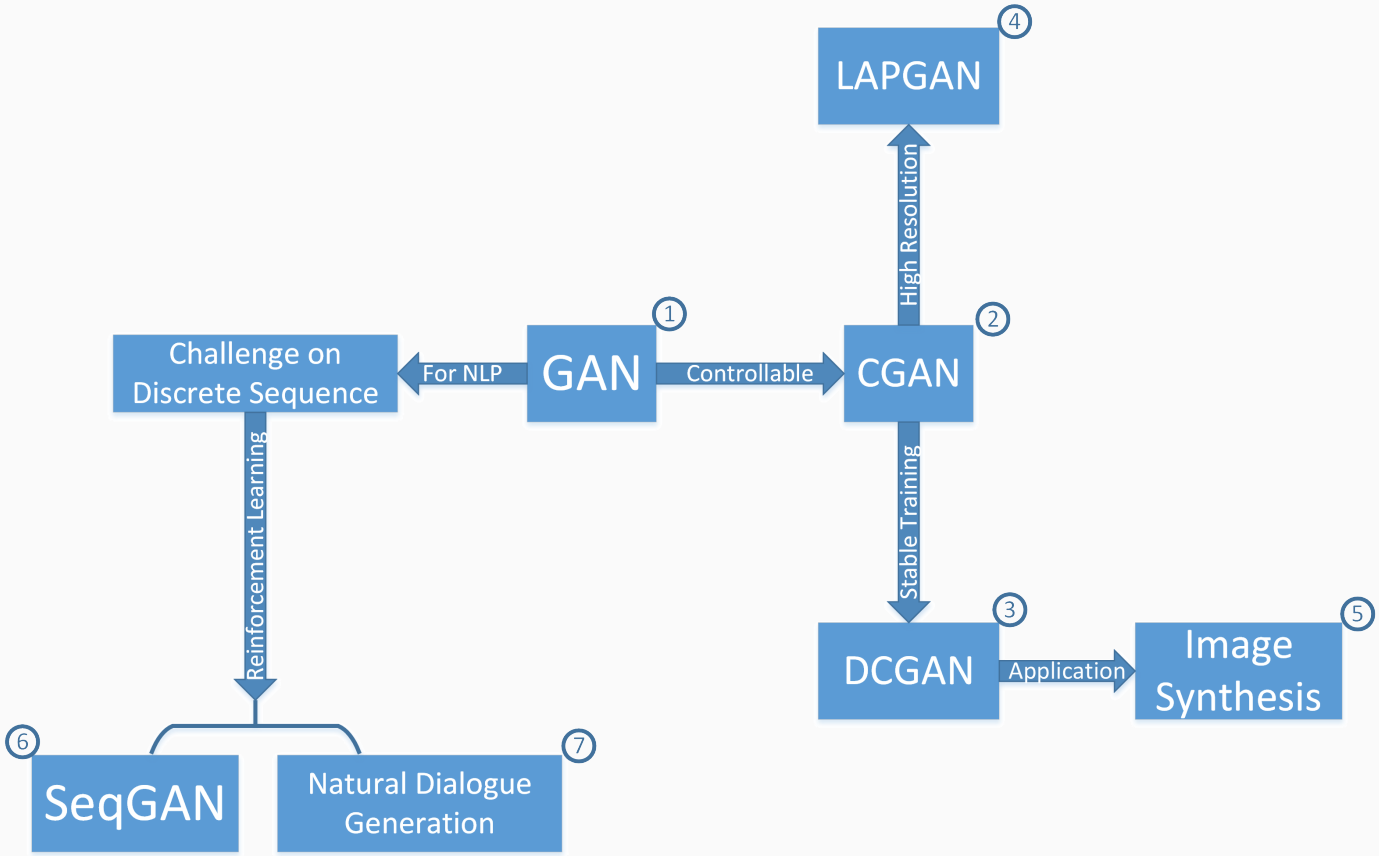
end for

Experiments

- MNIST.
- Toronto Face Database(TFD).
- CIFAR-10.



Outline


LAPGAN
4
High Resolution
Challenge on Discrete Sequence
For NLP
GAN
1
Controllable
CGAN
2
Reinforcement Learning
Stable Training
DCGAN
3
Application
Image Synthesis
5
6
SeqGAN
7
Natural Dialogue Generation

Conditional Generative Adversarial Nets

Mehdi Mirza, Simon Osindero
arXiv: 1411.1784

Introduction

- Generative adversarial nets were recently introduced as an alternative framework for training generative models.
- In an unconditioned generative model, there is no control on models of the data being generated.
- By conditioning the model on additional information it is possible to direct the data generating process.
- Such conditioning could be based on class labels, on some part of data, or even on data from different modality.

Introduction

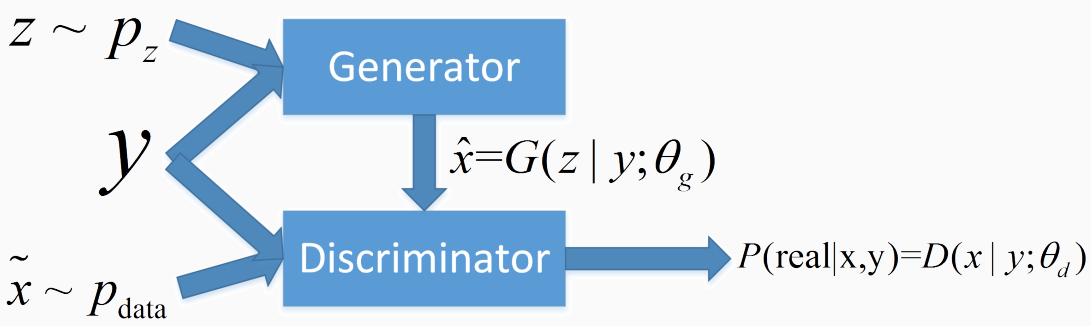
- Optimization object of unconditioned generative model:

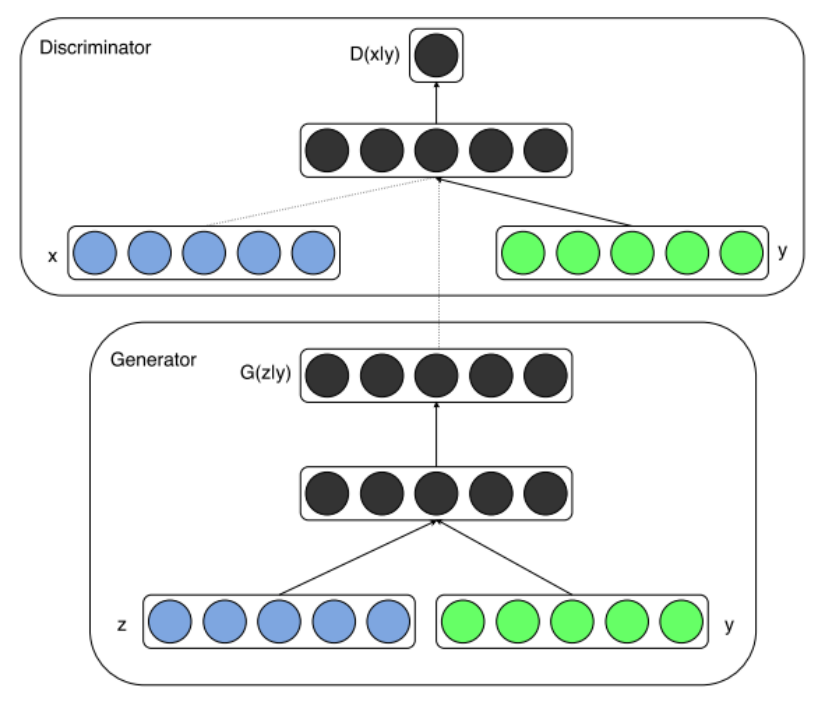
$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}}\left[\log D(x)\right] + \mathbb{E}_{z\sim p_z(z)}\left[\log(1-D(G(z)))\right].$$

- Optimization object of conditioned generative model:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}}\left[\log D(x|y)\right] + \mathbb{E}_{z\sim p_z(z)}\left[\log(1-D(G(z|y)|y))\right].$$

Structure

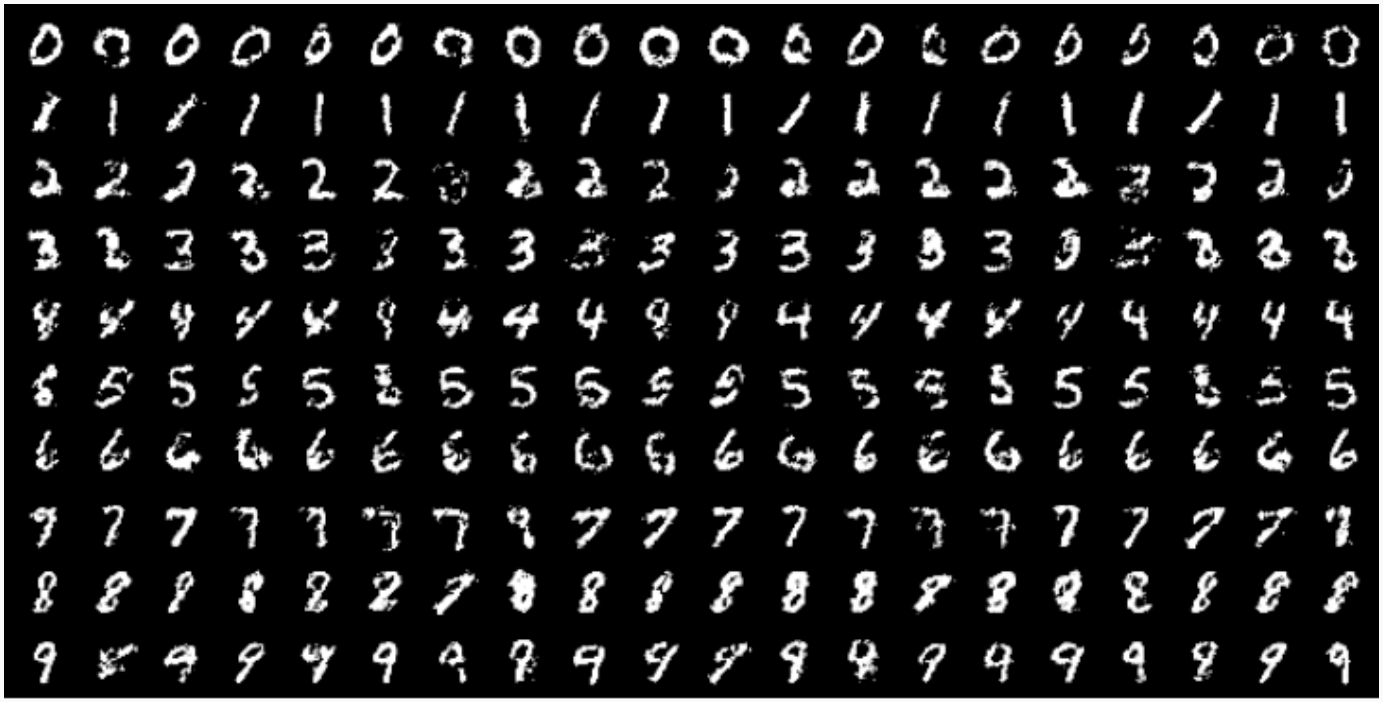

$z \sim p_z$
y
$\tilde{x} \sim p_{\text{data}}$
Generator
$\hat{x}=G(z \mid y;\theta_g)$
Discriminator
$P(\text{real}|\text{x,y})=D(x \mid y;\theta_d)$


Discriminator
D(xly)
x
y
Generator
G(zly)
z
y

Experiment

- Experiment on MNIST
- y is the 10 classes of digit represent in one-hot vector.

Figure: Generated MNIST digits, each row conditioned on one label



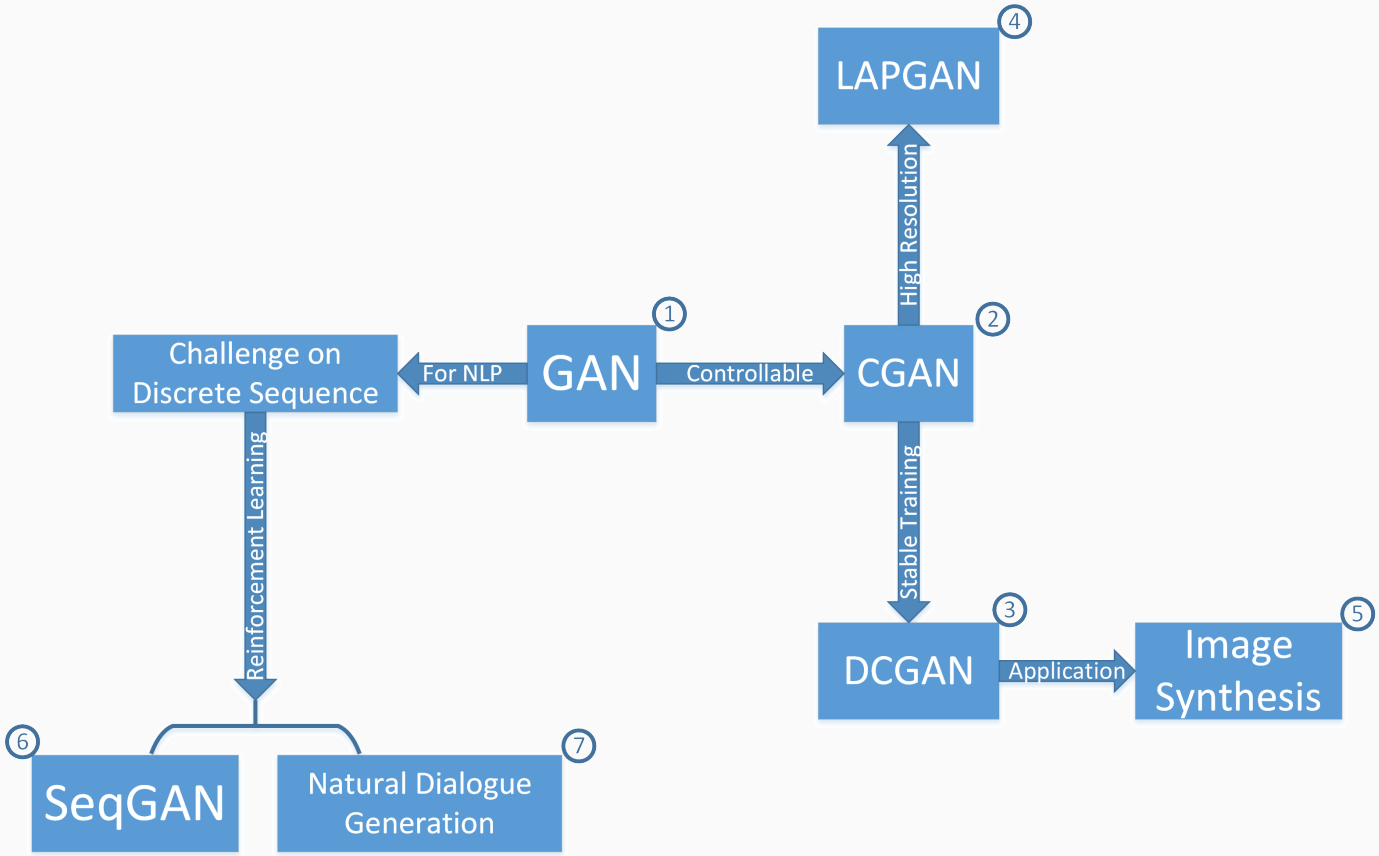
Experiment (Multi-modal)

- Pretrained image feature extractor and skip-gram model for word representation.
- Trained a convolutional image feature extractor on full ImageNet dataset.
- Trained a skip-gram model for word representation: a R^{200} word vector on YFCC100M(Yahoo Flickr Creative Common 100M) dataset meta-data.
- Conducted experiment on MIR Flickr 25,000 dataset, extract the image and tags features using the convolution model and embedding model.

Experiment (Multi-modal)

	User tags + annotations	Generated tags
	montanha, trem, inverno, frio, people, male, plant life, tree, structures, transport, car	taxi, passenger, line, transportation, railway station, passengers, railways, signals, rail, rails
	food, raspberry, delicious, homemade	chicken, fattening, cooked, peanut, cream, cookie, house made, bread, biscuit, bakes
	water, river	creek, lake, along, near, river, rocky, treeline, valley, woods, waters
	people, portrait, female, baby, indoor	love, people, posing, girl, young, strangers, pretty, women, happy, life

Outline


LAPGAN
4
High Resolution
Challenge on Discrete Sequence
For NLP
GAN
1
Controllable
CGAN
2
Reinforcement Learning
Stable Training
DCGAN
3
Application
Image Synthesis
5
6
SeqGAN
Natural Dialogue Generation
7

Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

Alec Radford, Luke Metz, Soumith Chintala
arXiv: 1511.06434

Introduction

- Shortcoming of GANs:
  - unstable to train
  - result ing generators that produce nonsensical outputs.
- Very limited published research in trying to understand and visualize what GANs learn, and the intermediate representations of multi-layer GANs.
- Contributions of this paper:
  - a set of constraints -> stable to train.
  - trained discriminators for image classification task -> competitive performance.
  - visualize the filters -> show that the filters learns to draw objects.
  - vector arithmetic properties -> semantic manipulation.

Approach

- Architecture guidlines for stable Deep Convolution GANs:
  - Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
  - Use batch normalization in both the generator and the discriminator.
  - Remove fully connected hidden layers for deeper architectures.
  - Use ReLU activation in generator for layers except for the output. which uses Tanh.
  - Use LeakyReLU activation in the discriminator for all layers.

Structure

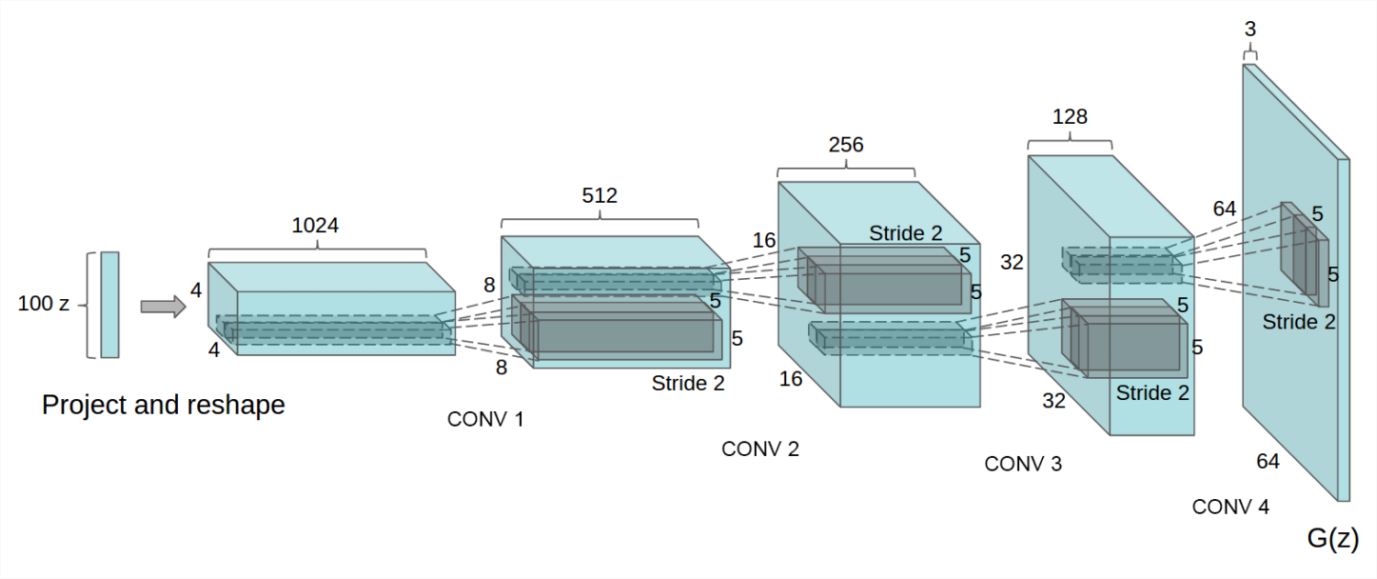


Figure: DCGAN generator used for LSUN scene modeling.

During the training, many technics are adopted including dropout and momentum.

Experiment

Empirical Validation of DCGANs Capabilities

- Classifying CIFAR-10 using GANs as a Feature Extractor
- To evaluate the quality of the representations learnt by DCGANs for supervised tasks:
  - the model was trained on ImageNet-1k.
  - use the discriminator's convolutional features from all layers.
  - maxpooling each layers representation to produce a 4×4 spatial grid.
  - flattened and concatenated to form a 28672 dimensional vector.

Model	Accuracy	Accuracy (400 per class)	max # of features units
1 Layer K-means	80.6%	63.7% ($\pm$0.7%)	4800
3 Layer K-means Learned RF	82.0%	70.7% ($\pm$0.7%)	3200
View Invariant K-means	81.9%	72.6% ($\pm$0.7%)	6400
Exemplar CNN	84.3%	77.4% ($\pm$0.2%)	1024
DCGAN (ours) + L2-SVM	82.8%	73.8% ($\pm$0.4%)	512

Empirical Validation of DCGANs Capabilities

- for StreetView House Numbers dataset (SVHN)

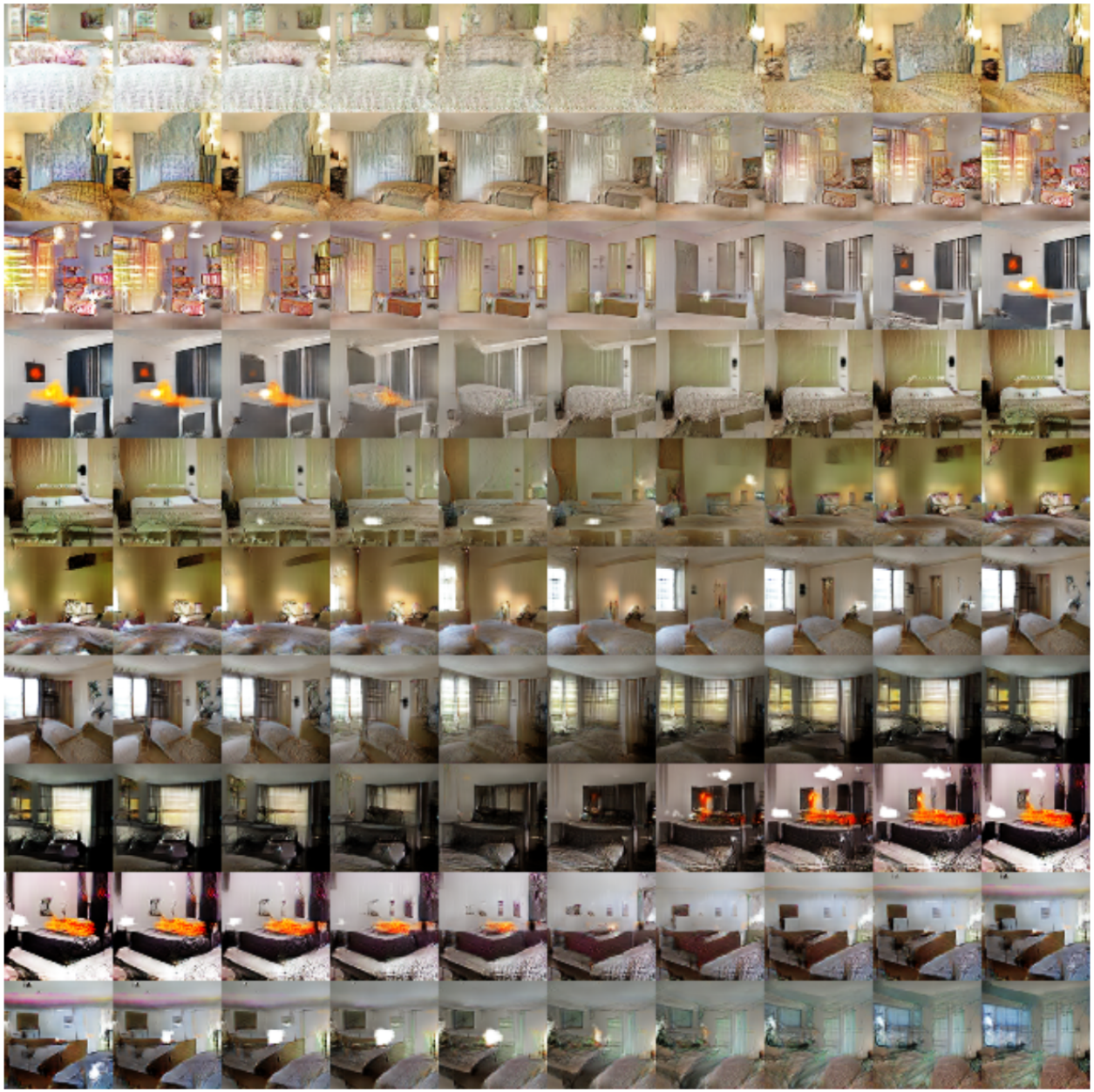
Model	error rate
KNN	77.93%
TSVM	66.55%
M1+KNN	65.63%
M1+TSVM	54.33%
M1+M2	36.02%
SWWAE without dropout	27.83%
SWWAE with dropout	23.56%
DCGAN (ours) + L2-SVM	22.48%
Supervised CNN with the same architecture	28.87% (validation)

- L2-SVM classifier on the top of same feature extraction extraction pipeline used for CIFAR-10.
- Achieves state of the art (for classification using 1000 labels) at 22.48% test error.
- Additionally, authors trained a purely supervised CNN with the same architecture on the same data -> CNN architecture used in DCGAN is not the key contributing factor of the model's performance.

Investigating and Visualizing the Internals of the Networks

- How to prove that the model learned feature representation instead of the mapping from a noise to a sample?
- If model learned the semantic representation, we can conduct small image transforms on feature space.
- The first experiment was to understand the landscape of the latent space.
- If walking in this latent space results in semantic changes to the image generations (such as object being added and removed), we can reason that the model has learned relevant and interesting representations.

Investigating and Visualizing the Internals of the Networks



Investigating and Visualizing the Internals of the Networks

- Unsupervised DCGAN trained on a large image dataset can also learn a hierarchy of features that are interesting.

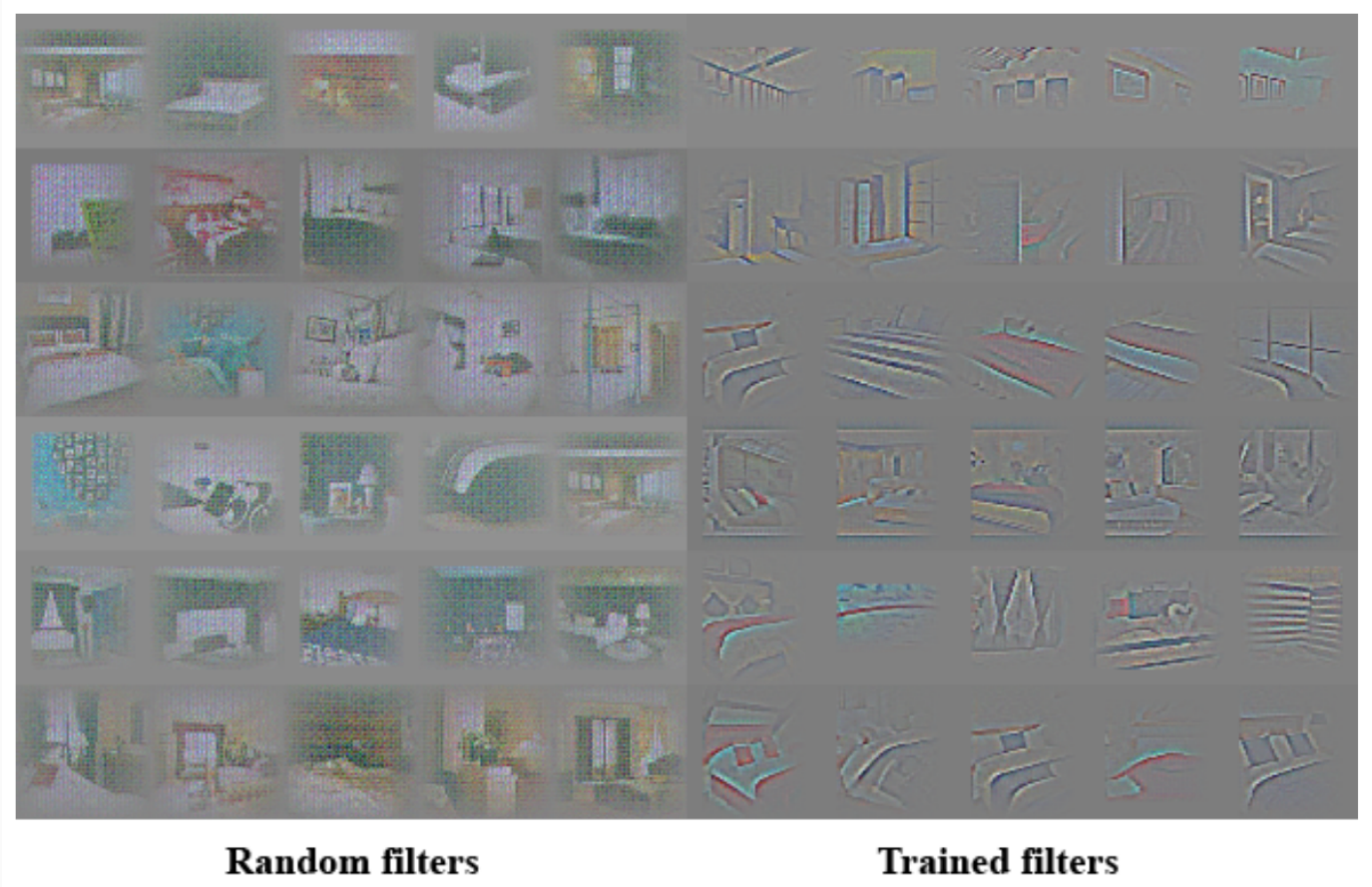


Figure: Features learnt by the discriminator.

Investigating and Visualizing the Internals of the Networks

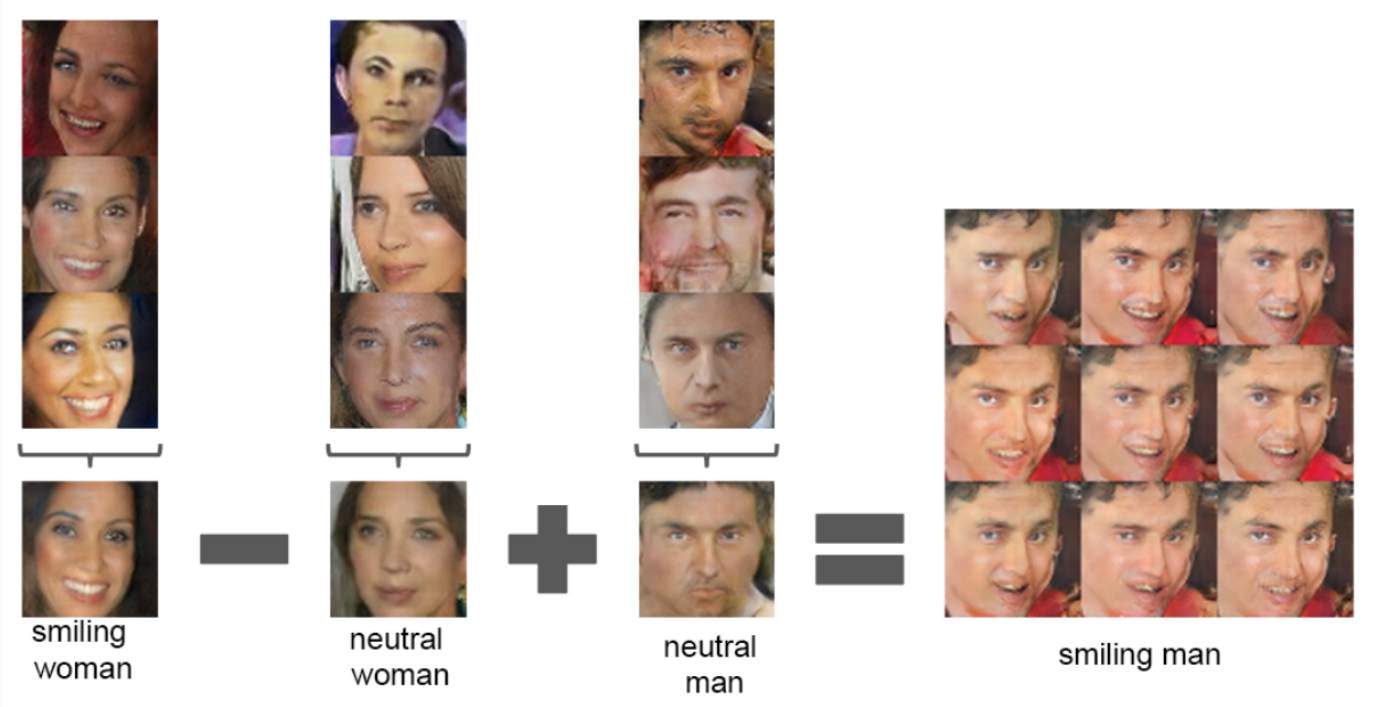
- Manipulating the generator representation: Forgetting to draw certain objects
- What representations the generator learns? Scene components? Beds? Windows? Lamps? Doors?
- If so, we can attempt to remove some component by manipulating the latent representation:
  - 150 samples, 52 window bounding boxes were drawn manually.
  - On the second highest convolution layers features, logistic regression was fit to predict whether a feature activation was on a window (or not).
  - by using the criterion that activations inside the drawn bounding boxes are positive and random samples from the same images are negatives.
  - All feature maps with weights greater than zero (200 in total) were dropped from all spatial locations.
  - Then, random new samples were generated with and without the feature map removal.

Investigating and Visualizing the Internals of the Networks

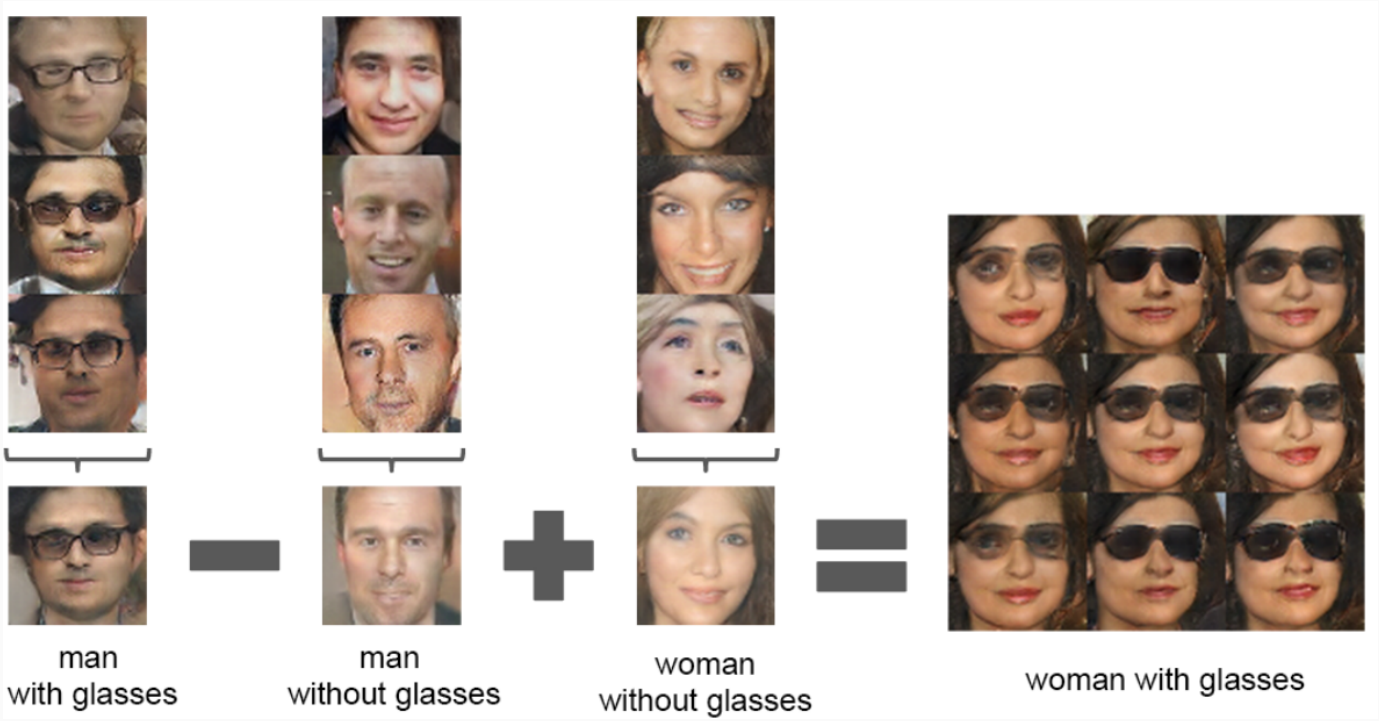
- Top row: un-modified samples from model.
- Bottom row: the same samples generated with dropping out ”window” filters.
- Some windows are removed, others are transformed into objects with similar visual appearance such as doors and mirrors.

Investigating and Visualizing the Internals of the Networks

- Representation of words:
  vector(”King”)-vector(”Man”)+vector(”Woman”)≈vector(”Queen”).
- Similar structure emerges in the *Z* representation of generators.
- Experiments working on only single samples per concept were unstable -> averaging the *Z* vector for three examples.



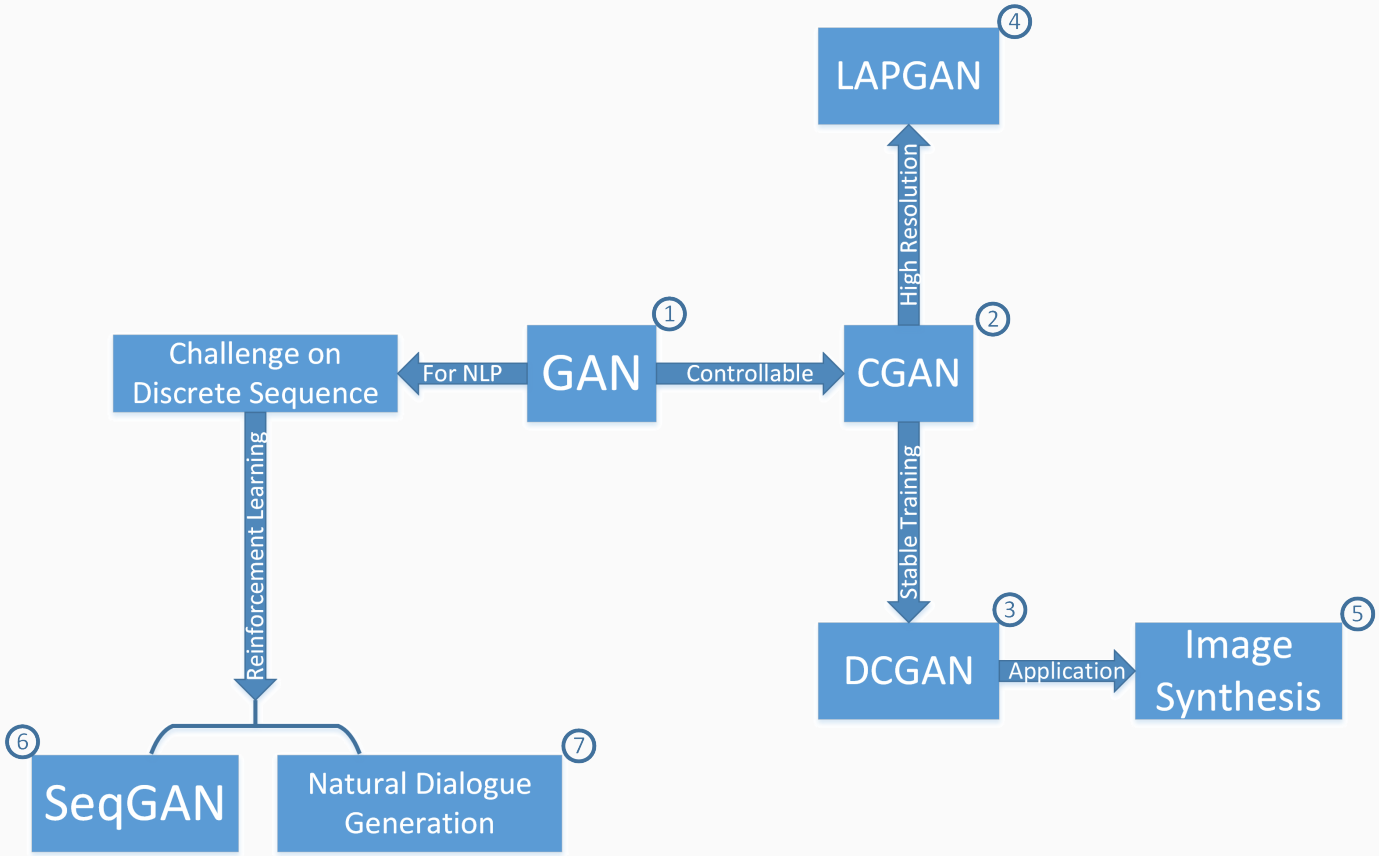
Investigating and Visualizing the Internals of the Networks



Investigating and Visualizing the Internals of the Networks

Figure: A ”turn” vector was created from four averaged samples of faces looking left vs looking right. By adding interpolations along this axis to random samples.

Outline


LAPGAN
4
High Resolution
Challenge on Discrete Sequence
For NLP
GAN
1
Controllable
CGAN
2
Reinforcement Learning
Stable Training
DCGAN
3
Application
Image Synthesis
5
6
SeqGAN
7
Natural Dialogue Generation

Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

Emily Denton, Soumith Chintala, Arthur Szlam, Rob Fergus
NIPS 2015
arXiv: 1506.05751

Introduction

- Building a good generative model of natural images has been a fundamental problem.
- However, images are complex and high dimensional, making them hard to model well.
- For the generation of high-resolution image, most existing approaches modelling entire scene instead generate image patches.
- Multi-scale structure of natural images, building a series of generative models.
- Each captures image structure at a particular scale of a Laplacian pyramid.

Laplacian Pyramid

- The main idea of Laplacian Pyramid is to consider the residue of the "image" and the "size-recovered down-sampled image".
- If I is a $j \times j$ image.
- Define $d(.)$ is down sampling. $d(I)$ will be a $j/2 \times j/2$ image.
- Define $u(,)$ is up sampling. $u(I)$ will be a $2j \times 2j$ image.
- First build a Gaussian pyramid $\mathcal{G}(I) = [I_0, I_1, \ldots, I_K]$.
- $I_0 = I$ and I_k is k repeated applications of $d(,)$. e.g. $I_2 = d(d(I))$. K is the number of levels in the pyramid.
- The coefficient h_k at each level k of the Laplacian pyramid $\mathcal{L}(I)$ are constructed by taking the difference between adjacent levels in the Gaussian pyramid:

$$h_k = \mathcal{L}_k(I) = \mathcal{G}_k(I) - u(\mathcal{G}_{k+1}(I)) = I_k - u(I_{k+1})$$

- If we obtain, we can reconstruct the image from I_K to I_0 by:

$$I_k = u(I_{k+1}) + h_k$$

Laplacian Generative Adversarial Networks (LAPGAN)

- LAPGAN followed these ideas by training a series of generators to build coefficients $\tilde{h}_k$ at each scale.
- We can "draw" a high-resolution image by these coefficients by:

$$\tilde{I}_k = u(\tilde{I}_{k+1}) + \tilde{h}_k = u(\tilde{I}_{k+1}) + G_k(z_k, u(\tilde{I}_{k+1}))$$

- The generative models $G_0, \ldots, G_k$ are trained using the CGAN with the condition $u(\tilde{I}_{k+1})$ at each level of the pyramid.

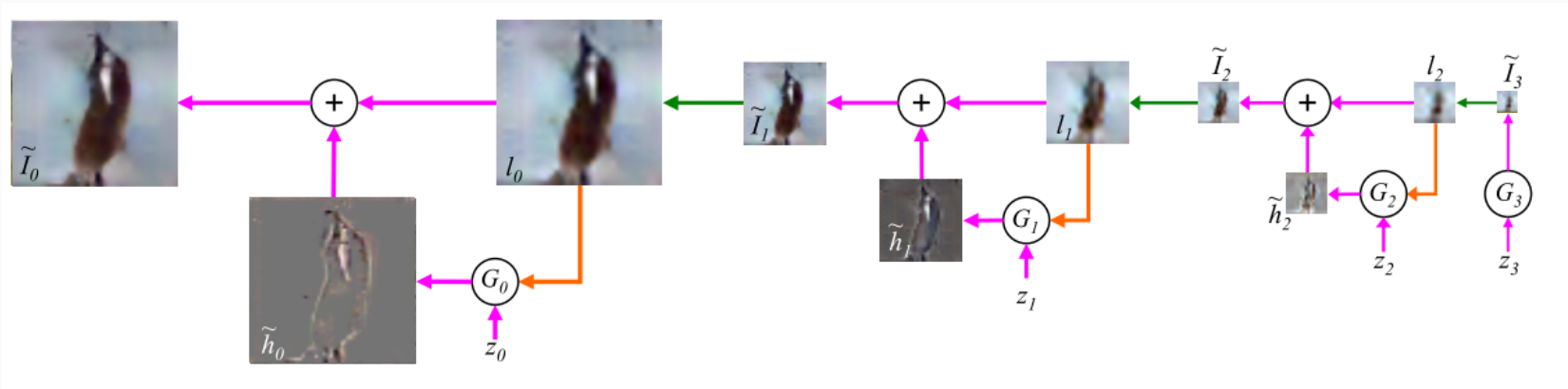
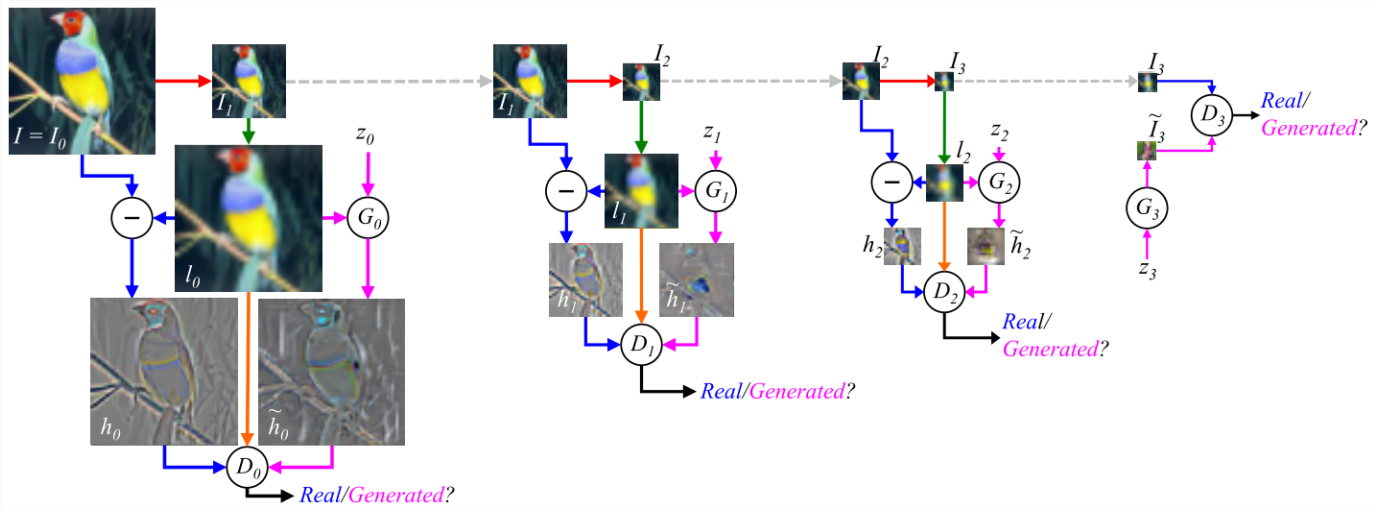


Figure: The structure of the generator of LAPGAN

- We will be able to calculate $\tilde{h}_k = G_k(z_k, u(l_{k+1}))$ and generate the high-solution realistic image if we have these GANs.

Laplacian Generative Adversarial Networks (LAPGAN)

- So, how do we get these GANs? (Training approach)



- The training start with a 64×64 input image I from trining set:
  1. take $I_0 = I$ and down-sample it by a factor of two to produce I_1.
  2. up-sample I_1 by a factor of two, giving a low-pass version l_0 of I_0.
  3. In the real case (blue arrows), we compute high-pass $h_0 = I_0 - l_0$.
  4. In the generated case (megenta arrows), the generative network G_0 receives as input a random noise vector z_0 and l_0. It outputs a generated high-pass image $\tilde{h}_0 = G_0(z_0, l_0)$.
  5. In both the real/generated cases, D_0 also receives l_0 as condition.
  6. At level 3, I_3 is an 8×8 image, simple enough to be modeled directly with a standard GANs G_3 and D_3

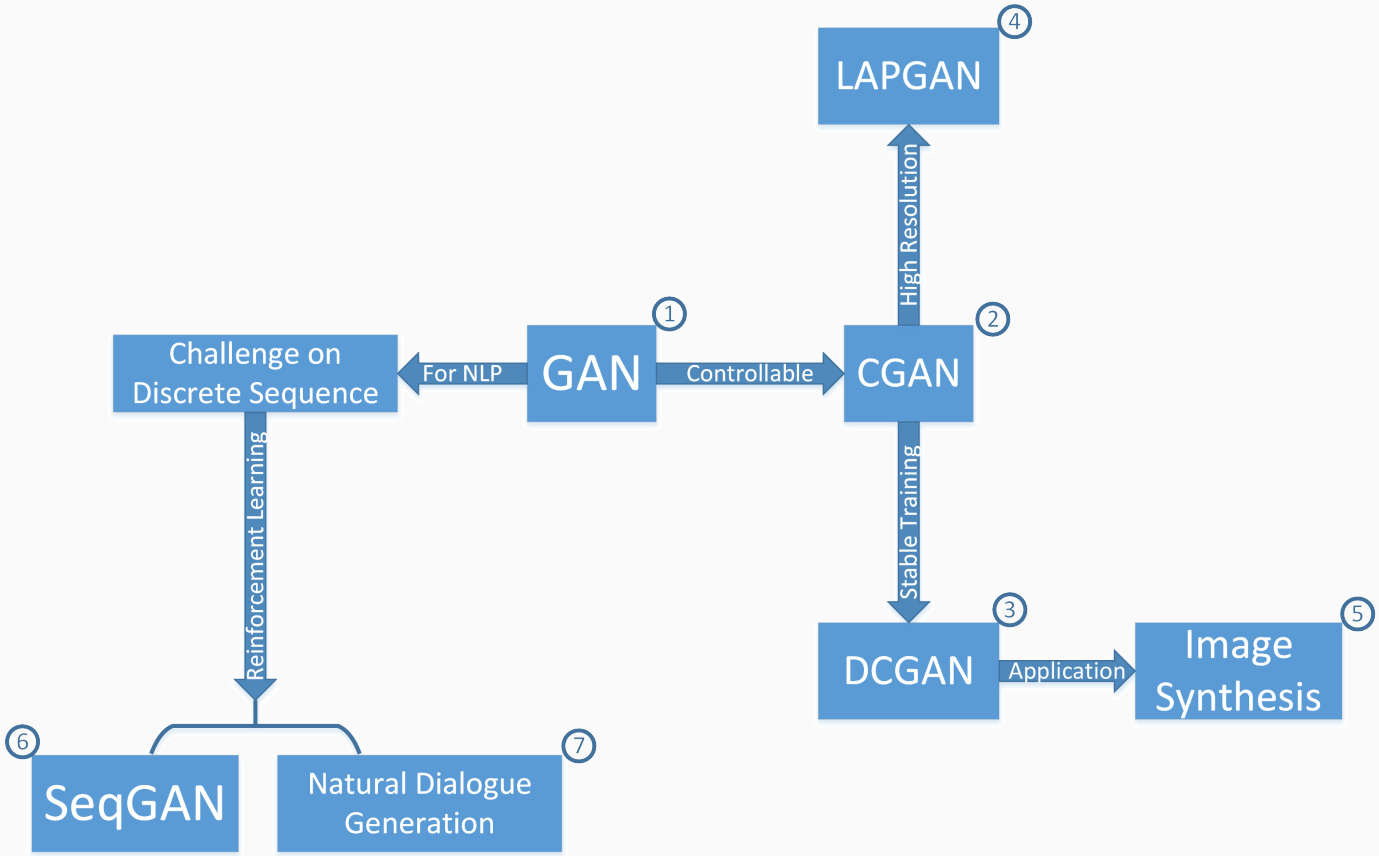
Laplacian Generative Adversarial Networks (LAPGAN)

- The key idea in this work is breaking the generation into successive refinements.
- Give up any ”global” manipulation.
- Never make any attempt to train a network to discriminate between the output of a cascade and a real image and instead focus on making each step plausible.
- Furthermore, the independent training of each pyramid level has the advantage that it is far more difficult for the model to memorize training examples — a hazard when high capacity deep networks are used. (over-fit in generative model).

Experiment

Figure: STL 10 Samples:
(a) Random 96 $\times$ 96 samples from our LAPGAN model.
(b) Coarse-to-fine generation chain.

Outline

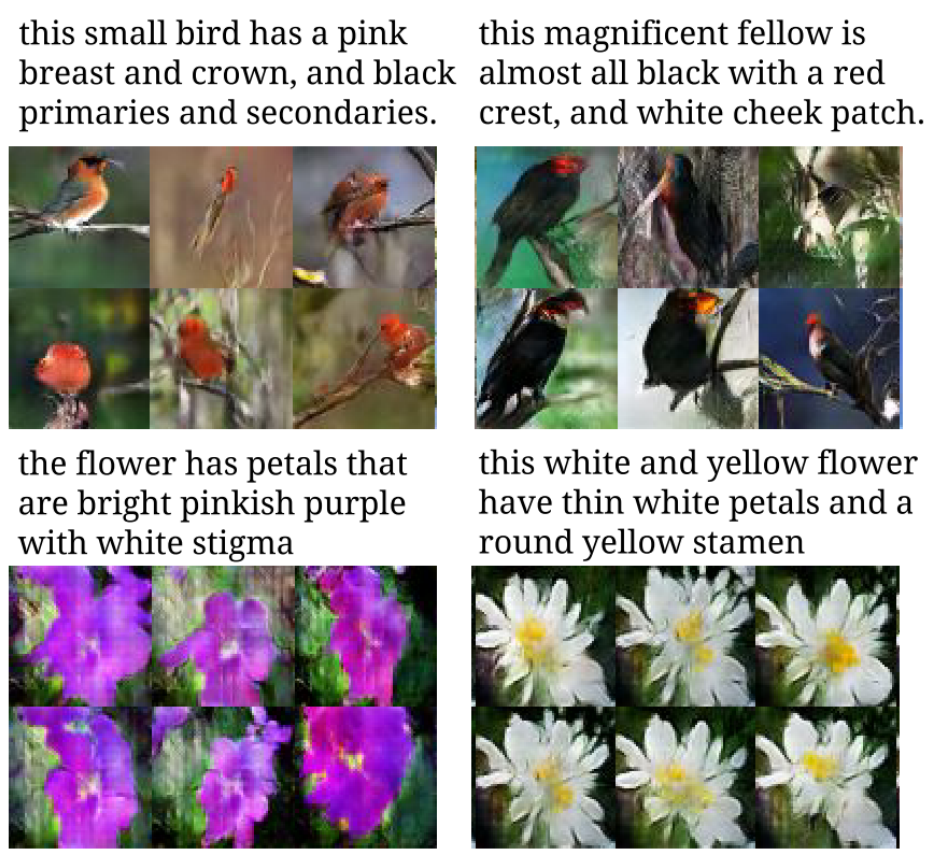

LAPGAN
4
High Resolution
Challenge on Discrete Sequence
For NLP
GAN
1
Controllable
CGAN
2
Reinforcement Learning
Stable Training
DCGAN
3
Application
Image Synthesis
5
6
SeqGAN
Natural Dialogue Generation
7

Generative Adversarial Text to Image Synthesis

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, Honglak Lee
ICML2016+JMLR Vol.48
arXiv: 1605.05396

Introduction

- Papers we mentioned earlier are all about ”generate realistic images”.
- Can we translat text in the form of single-sentence human-written descriptions directly into image pixels?



Introduction

- This work aims to learn a mapping directly from words and characters to image pixels.
- To solve this challenging problem requires solving two sub problems:
    1. learn a text feature representation that captures the important visual details
    2. use these features to synthesize a compelling image that a human might mistake for real.
- Fortunately, deep learning has enabled enormous progress in both sub problems - natural language representation and image synthesis.
- Current task will be build on these previous works.

Method

- Approach is to train a deep convolutional generative adversarial network (DCGAN) conditioned on text features encoded by a hybird character-level convolutional-recurrent neural network.
- Both the generator network G and the discriminator network D perform feed-forward inference conditioned on the text feature.
- The generator network is denoted G: $\mathrm{R}^Z \times \mathrm{R}^T \rightarrow \mathrm{R}^D$.
- The discriminator as D: $\mathrm{R}^D \times \mathring{R}^T \rightarrow 0, 1$.
- T is the dimension of the text description embedding, D is the dimension of the image, and Z is the dimension of the noise input to G.