Composite Functional Gradient Learning of Generative Adversarial Models

Tong Zhang

joint work with Rie Johnson

Fundamental Challenge in the AI Era

Is deep learning Alchemy or science?

- practitioners: trial and error approach move very fast!
- theoreticians: challenged by whether relevant to practice or not

Fundamental Challenge in the AI Era

Is deep learning Alchemy or science?

- practitioners: trial and error approach move very fast!
- theoreticians: challenged by whether relevant to practice or not

This work: mathematical theory with practical impact

- understand an algorithm's limitations
- lead to new algorithm addressing the limitations

Fundamental Challenge in the AI Era

Is deep learning Alchemy or science?

- practitioners: trial and error approach move very fast!
- theoreticians: challenged by whether relevant to practice or not

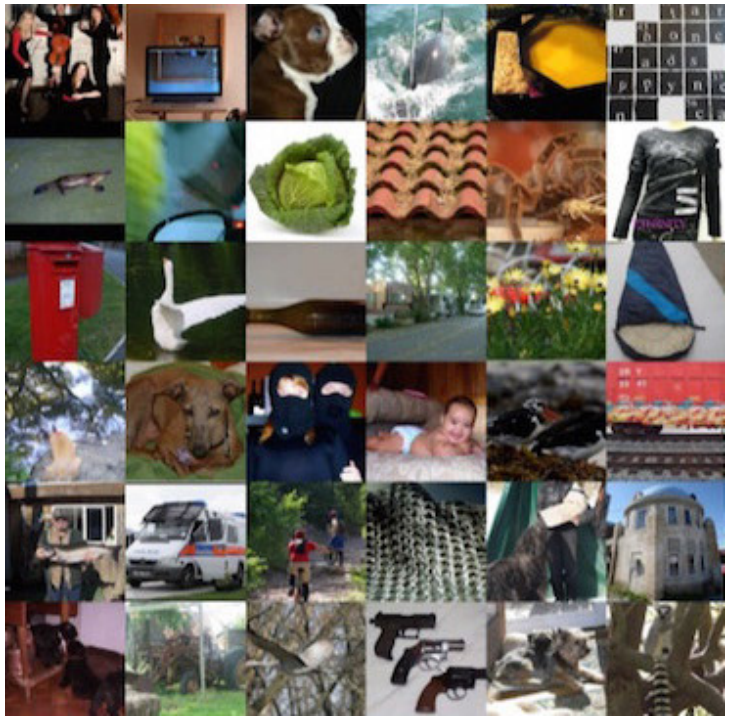
This work: mathematical theory with practical impact

- understand an algorithm's limitations
- lead to new algorithm addressing the limitations

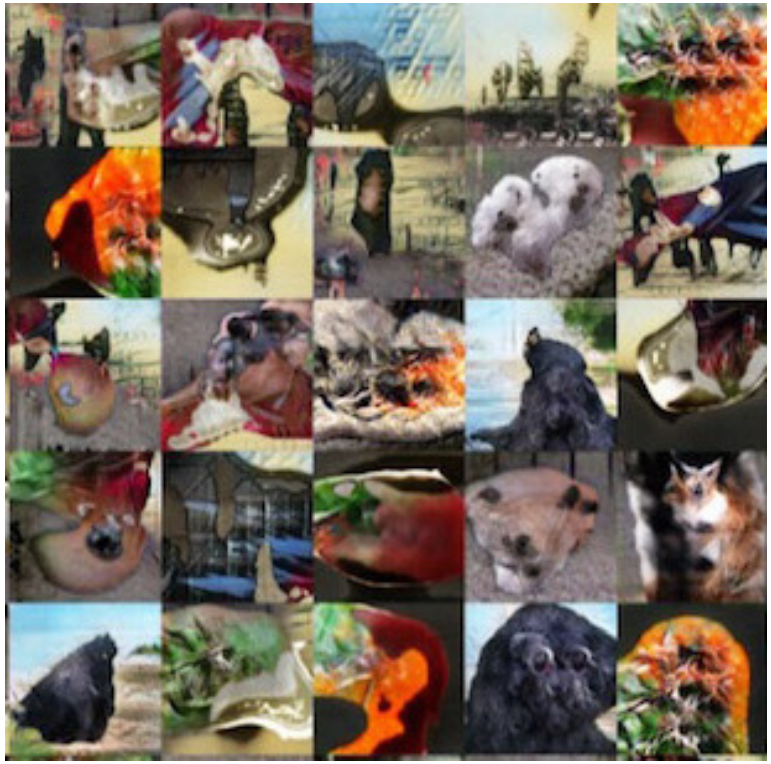
Key Message: optimization in function space makes problems easier

Problem

Generate Artificial Data from Examples



(a) Real Image



(b) Generated Image

Classical Approach

Given:

- observed data $S = \{x_1, \ldots, x_n\}$
- a family of probability density functions $p_\theta(x)$

Classical Approach

Given:

- observed data $S = \{x_1, \ldots, x_n\}$
- a family of probability density functions $p_\theta(x)$

Procedure:

- estimate $\hat{\theta}$ from S (frequentist)
- draw sample from $p_{\hat{\theta}}(x)$

Some Difficulties of the Classical Approach

Estimation: Powerful family $p_\theta(x)$ often has a form

$$p_\theta(x) \propto \exp(-f_\theta(x)),$$

- difficulty: normalization factor hard to compute.
- solution: unnormalized models

Some Difficulties of the Classical Approach

Estimation: Powerful family $p_\theta(x)$ often has a form

$$p_\theta(x) \propto \exp(-f_\theta(x)),$$

- difficulty: normalization factor hard to compute.
- solution: unnormalized models

Draw Sample:

- difficulty: Monte Carlo simulation can be slow.
- solution: SDE based Monte Carlo (recent interests)

Some Difficulties of the Classical Approach

Estimation: Powerful family $p_\theta(x)$ often has a form

$$p_\theta(x) \propto \exp(-f_\theta(x)),$$

- difficulty: normalization factor hard to compute.
- solution: unnormalized models

Draw Sample:

- difficulty: Monte Carlo simulation can be slow.
- solution: SDE based Monte Carlo (recent interests)

This talk:
an alternative data transformation approach that becomes popular

The Data Transformation Approach

Want to draw samples similar to $S = \{x_1, \ldots, x_n\}$

Procedure:

- Generate z from an initial noise distribution p_z
- Transform $z : z \rightarrow G(z)$ so that $G(z)$ has the same distribution as S

The Data Transformation Approach

Want to draw samples similar to $S = \{x_1, \ldots, x_n\}$

Procedure:

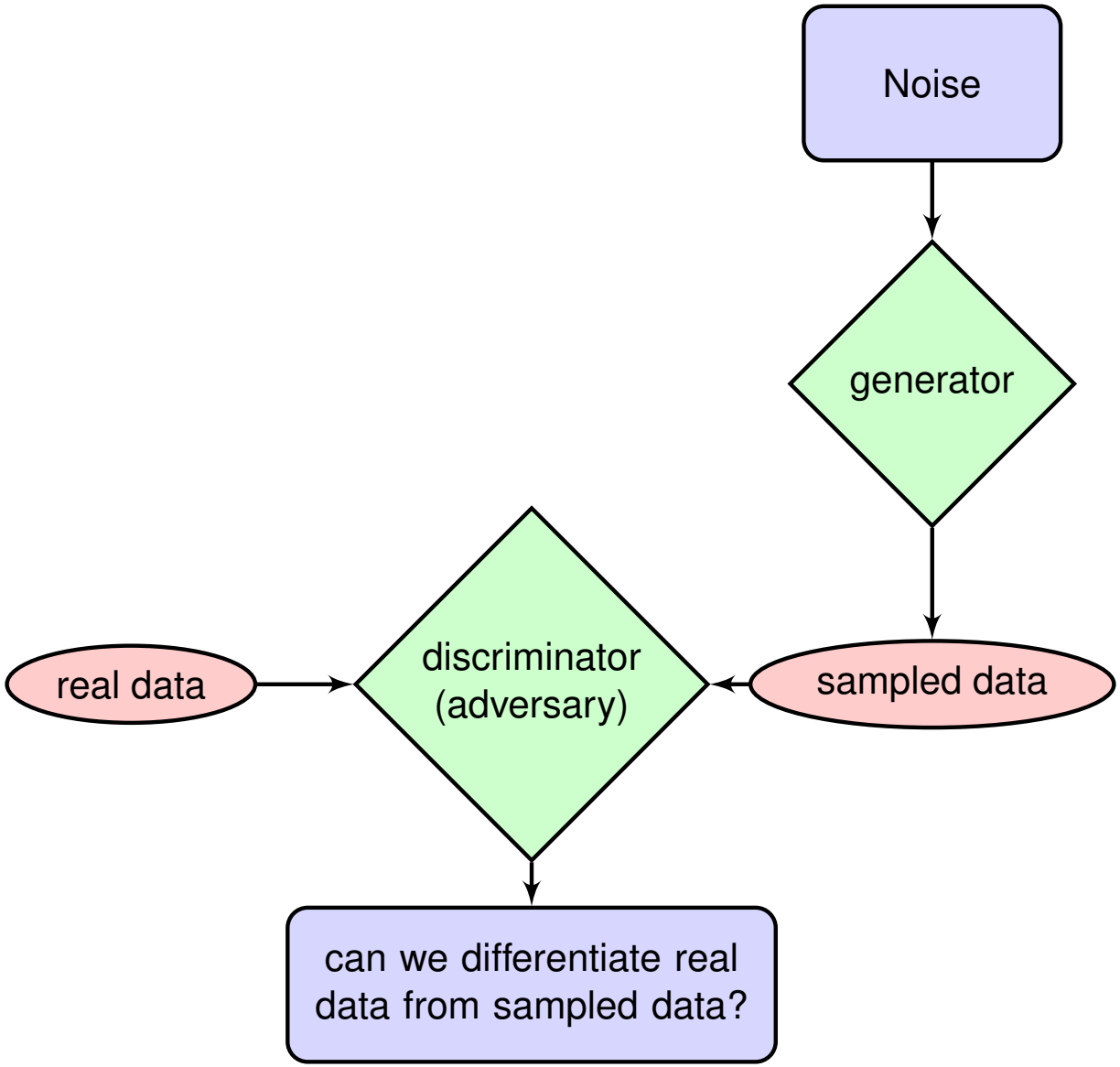
- Generate z from an initial noise distribution p_z
- Transform $z : z \rightarrow G(z)$ so that $G(z)$ has the same distribution as S

Key problem:

- How to estimate the transformation $G(z)$?

Modern answers: GAN (and related, VAE)

Generative Adversarial Network (GAN)



Mathematical Formulation of GAN

- Real data $S = \{x_1, \ldots, x_n\}$
- Noise $z_1, \ldots, z_m$
- Generator $G(\theta; z)$
- Discriminator $d(\psi; x)$

Nonconvex Minimax Saddle Point Optimization Problem:

$$\max_{\psi} \min_{\theta} \left[\frac{1}{n} \sum_i \log d(\psi; x_i) + \frac{1}{m} \sum_j \log(1 - d(\psi; G(\theta; z_j))) \right].$$

We have:

$$d(\psi; x) = \frac{1}{1 + \exp(-D(\psi; x))}$$

Solving Saddle Point Optimization

$$\max_{\psi} \min_{\theta} \left[\frac{1}{n} \sum_{i} \log d(\psi; x_i) + \frac{1}{m} \sum_{j} \log(1 - d(\psi; G(\theta; z_j))) \right].$$

Using SGD as follows.

- Update discriminator (primal variable SGD):

$$\psi \leftarrow \psi + \eta \nabla_{\psi} \left[\log d(\psi; x_i) + \log(1 - d(\psi; G(\theta; z_j))) \right]$$

- Update generator (dual variable SGD):

$$\theta \leftarrow \theta - \eta' \nabla_{\theta} \log(1 - d(\psi; G(\theta; z_j)))$$

Solving Saddle Point Optimization

$$\max_{\psi} \min_{\theta} \left[\frac{1}{n} \sum_{i} \log d(\psi; x_i) + \frac{1}{m} \sum_{j} \log(1 - d(\psi; G(\theta; z_j))) \right].$$

Using SGD as follows.

- Update discriminator (primal variable SGD):

$$\psi \leftarrow \psi + \eta \nabla_{\psi} \left[\log d(\psi; x_i) + \log(1 - d(\psi; G(\theta; z_j))) \right]$$

- Update generator (dual variable SGD):

$$\theta \leftarrow \theta - \eta' \nabla_{\theta} \log(1 - d(\psi; G(\theta; z_j)))$$

Strange phenomenon: generator update with logD trick is practically better

$$\theta \leftarrow \theta + \eta' \nabla_{\theta} \log(d(\psi; G(\theta; z_j))$$

Mathematical Theory of GAN

The population version of GAN tries to find optimal d to maximize

$$\int \log d(x) p_{\text{real}}(x) dx + \int \log(1 - d(x)) p_{\text{gen}}(x) dx,$$

where

$$p_{\text{gen}}(x) \text{ is the density of } x = G(z)$$

Mathematical Theory of GAN

The population version of GAN tries to find optimal d to maximize

$$\int \log d(x) p_{\text{real}}(x) dx + \int \log(1 - d(x)) p_{\text{gen}}(x) dx,$$

where

$$p_{\text{gen}}(x) \text{ is the density of } x = G(z)$$

The optimal d is given by

$$d_{\text{optimal}}(x) = \frac{p_{\text{real}}(x)}{p_{\text{real}}(x) + p_{\text{gen}}(x)}.$$

Generator Minimizes JS-divergence

$$\max_d \min_G \left[\mathbf{E}_x \log d(x) + \mathbf{E}_z \log(1 - d(G(z)))\right].$$

Substitute d by d_{optimal}, we get Jensen-Shanon (JS) divergence minimization:

$$\min_G \left[\int p_{\text{real}}(x) \log \frac{2p_{\text{real}}(x)}{p_{\text{real}}(x) + p_{\text{gen}}(x)} dx + \int p_{\text{gen}}(x) \log \frac{2p_{\text{gen}}(x)}{p_{\text{real}}(x) + p_{\text{gen}}(x)} dx \right],$$

where

$$p_{\text{gen}}(x) \text{ is the density of } x = G(z)$$

Some Issues of GAN

- The optimization procedure is unstable
  - one proposed improvement is WGAN (changing loss function)
- Practical implementation with logD trick is only a heuristic
  - inconsistent with the minimax formulation, implying the theory is flawed
- Minimizes JS divergence, not KL divergence

We want to design a procedure that addresses the above points.

- based on KL divergence minimization
- stable (by modifying optimization process of GAN)
- can explain the logD trick

Our Approach: Change the Optimization Problem

- Learn $G(z)$ to minimize the KL-divergence between the distributions of real data and generated data:

$$\int p_{\text{real}}(x) \log \frac{p_{\text{real}}(x)}{p_{\text{gen}}(x)} dx$$

- Procedure uses functional gradient learning greedily, similar to gradient boosting.
- Learning procedure uses functional compositions in the form

$$G_t(z) = G_{t-1}(z) + \eta_t g_t(G_{t-1}(z)), \qquad (t = 1, \ldots, T)$$

gradient descent in function space

Adversarial Learner

Given training examples $\{(x_i, y_i)\}$ (where $y_i = \pm 1$).

Assume there is an oracle adversarial learner $\mathcal{A}$ that can solve the logistic regression problem:

$$D(x) \approx \arg\min_D \sum_i \ln(1 + \exp(-D(x_i)y_i)).$$

Using

- real sample $S = \{x_1, \ldots, x_n\}$ with label $y = 1$
- generated sample $\{G(z_1), \ldots, G(z_m)\}$ with label $y = -1$

We can find approximately

$$D(x) \approx \ln \frac{p_*(x)}{p_{\text{gen}}(x)}$$

Theory (under suitable assumptions)

Theorem

Consider variable transformation $X' = X + \eta g(X)$.
Let p *be the probability density of random variable* X.
Let p' *be the probability density of random variable* X'.
Let p_* *be the probability density of the real data.*
Let $\mathcal{D}(x) := \ln \frac{p_*(x)}{p(x)}$.
Assume $D_\epsilon(x) \approx \mathcal{D}(x)$ *is learned from adversarial learner:*

$$\int p_*(x) \max(1, \|\nabla \ln p_*(x)\|) \left(|D_\epsilon(x) - \mathcal{D}(x)| + \left| e^{D_\epsilon(x)} - e^{\mathcal{D}(x)} \right| \right) dx \leq \epsilon \,.$$

Then for some constant $c > 0$*:*

$$KL(p_*||p') \leq KL(p_*||p) - \eta \int p_*(x)\, g(x)^\top \nabla D_\epsilon(x)\, dx + c\eta^2 + c\eta\epsilon.$$

Interpretation of the Theory

The result is

$$KL(p_*||p') \leq KL(p_*||p) - \eta \int p_*(x) g(x)^\top \nabla D(x)\, dx + O(\eta^2).$$

If we further take

$$g(x) = s(x)\nabla D(x),$$

where $s(x) > 0$ is a scaling factor, then

$$KL(p_*||p') \leq KL(p_*||p) - \eta \int p_*(x) s(x)\, \|\nabla D(x)\|_2^2\, dx + O(\eta^2).$$

Implication:

$$\int p_*(x) s(x)\, \|\nabla D(x)\|_2^2\, dx \to 0,$$

which means

$$D(x) = \ln \frac{p_*(x)}{p_{\text{gen}}(x)} = \text{constant}$$

Algorithm: CFG for Generative Adversarial Learning

Algorithm 1 CFG: Composite Functional Gradient Learning

Require: real data $x_1^*, \ldots, x_n^*$, initial generator $G_0(z)$

1: **for** $t = 1, 2, \ldots, T$ **do**

2: $D_t(x) \leftarrow \arg\min_D \left[\frac{1}{n}\sum_{i=1}^{n} \ln(1 + e^{-D(x_i^*)}) + \frac{1}{m}\sum_{i=1}^{m} \ln(1 + e^{D(G_{t-1}(z_i))})\right]$

3: $g_t(x) \leftarrow s_t(x)\nabla D_t(x)$ (usually $s_t(x) = 1$)

4: $G_t(z) \leftarrow G_{t-1}(z) + \eta_t g_t(G_{t-1}(z))$, for some $\eta_t > 0$.

5: **end for**

6: **return** generator $G_T(z)$

Theory: as $n \to \infty$ and $T \to \infty$

$$G_T(z) \sim p_*(\cdot)$$

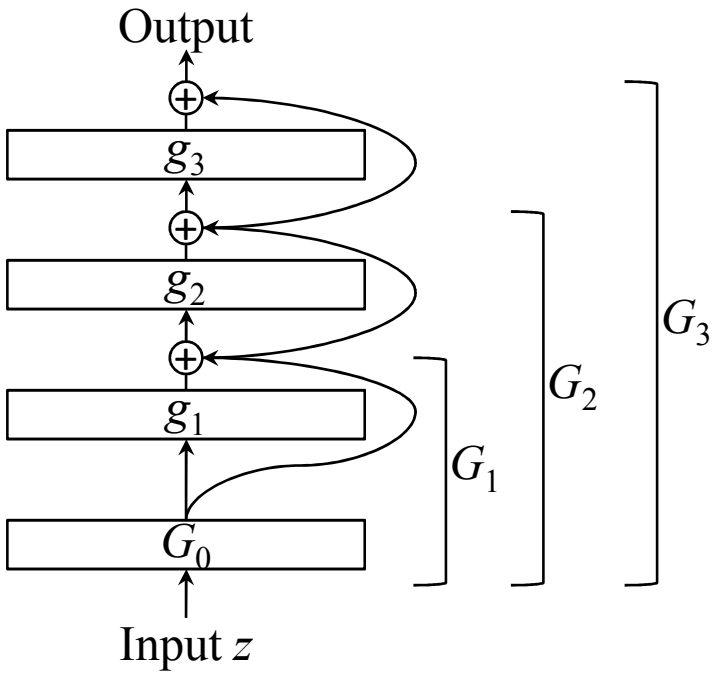
Incremental Composite Functional Gradient Learning

Algorithm 2 ICFG: Incremental Composite Functional Gradient Learning

Require: training examples S_*, prior p_z, initial generator G_0, discriminator D

1: **for** $t = 1, 2, \ldots, T$ **do**
2: **for** U steps **do**
3: Sample $x_1^*, \ldots, x_b^*$ from S_*, and $z_1, \ldots, z_b$ according to p_z.
4: Update discriminator D by SGD with minibatch gradient:
 $\nabla_{\theta_D} \frac{1}{b} \sum_{i=1}^{b} \left[\ln(1 + \exp(-D(x_i^*))) + \ln(1 + \exp(D(G_{t-1}(z_i))))\right]$
5: **end for**
6: $g_t(x) \leftarrow s_t(x)\nabla D(x)$ (most simply $s_t(x) = 1$)
7: $G_t(z) \leftarrow G_{t-1}(z) + \eta_t g_t(G_{t-1}(z))$, for some $\eta_t > 0$.
8: **end for**
9: **return** generator G_T

Graphical Illustration



Generator network automatically derived by CFG
$T = 3$ and '$\oplus$' indicates addition

Problem: network depth grows when T increases
Solution: use a fixed depth approximator

Algorithm: xICFG of GAN

Algorithm 3 xICFG: Approximate Incremental CFG Learning

Require: a set of training examples S_*, prior p_z, approximator $\widetilde{G}$ at its initial state, discriminator D.

1: **loop**
2: $S_z \leftarrow$ an input pool of the given size, sampled according to p_z.
3: $q_z \leftarrow$ the uniform distribution over S_z.
4: $G, D \leftarrow$ output of ICFG using $S_*, q_z, \widetilde{G}, D$ as input.
5: **if** some exit criteria is met **then**
6: **return** generator G
7: **end if**
8: Update the approximator $\widetilde{G}$ to minimize $\frac{1}{2}\sum_{z \in S_z} \|\widetilde{G}(z) - G(z)\|^2$
9: **end loop**

$\widetilde{G}(z)$ is a network of the fixed size

$G(z)$ is of growing but limited size (initialized from $\widetilde{G}(z)$ every time)

Approximator Network

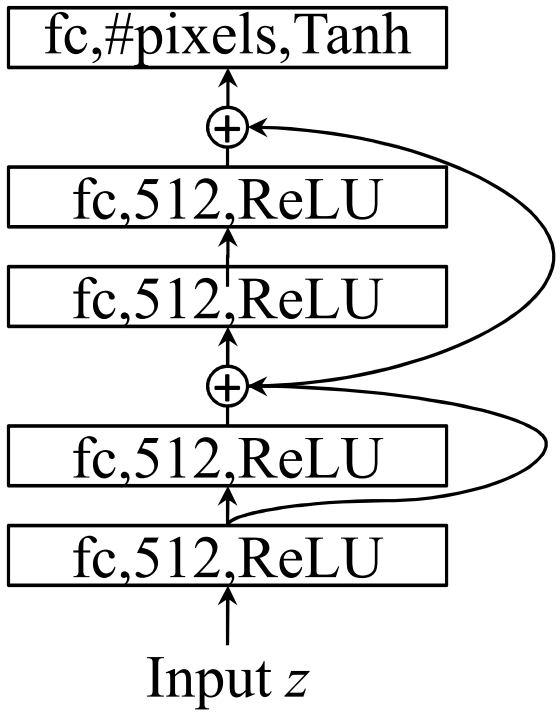


Figure: Fully-connected networks with shortcuts for use as a xICFG approximator or a GAN generator.

Compare to Original GAN (GAN0)

Algorithm 4 GAN0(Generative Adversarial Nets)

Require: training examples S_*, prior p_z, discriminator D, generator G.

1: Initialize discriminator d and generator G randomly.
2: **for** T steps **do**
3: Update discriminator D by ascending the stochastic gradient:
4: Sample $z_1, \ldots, z_b$ according to p_z.
5: Update the generator by descending the stochastic gradient:

$$\frac{1}{b}\sum_{i=1}^{b} \underbrace{(1+\exp(-D(G(z_i))))^{-1}}_{s(G(z_i))} \nabla_{\theta_G} D(G(z_i))$$

6: **end for**
7: **return** generator G

Problem: $s(G(z)) \approx 0$ when $D(G(z)) \ll 0$, which happens in the beginning.

Compare to GAN with logD trick (GAN1)

Algorithm 5 GAN1(Generative Adversarial Nets)

Require: training examples S_*, prior p_z, discriminator D, generator G.

1: Initialize discriminator d and generator G randomly.
2: **for** T steps **do**
3: Update discriminator d by ascending the stochastic gradient:
4: Sample $z_1, \ldots, z_b$ according to p_z.
5: Update the generator by descending the stochastic gradient:
$$\frac{1}{b}\sum_{i=1}^{b} \underbrace{(1+\exp(D(G(z_i))))^{-1}}_{s(S(z_i))} \nabla_{\theta_G} D(G(z_i))$$
6: **end for**
7: **return** generator G

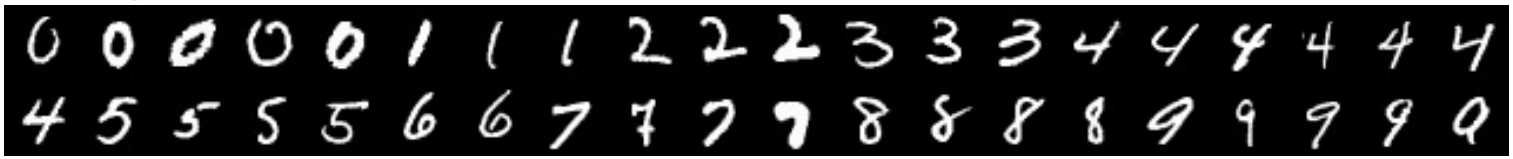
Good! $s(G(z)) \approx 1$ when $D(G(z)) \ll 0$, which happens in the beginning.

Remarks of GAN

- GAN0 is original GAN
  - SGD optimization of the minimax formulation of GAN
- GAN1 is GAN with log trick, which is a heuristics
  - cannot be explained using the original minimax formulation of GAN
  - can be explained by our theory
- xICFG
  - similar to GAN1
  - grows generator using composite function gradient
  - use approximator to reduce the network to fixed size

Data

- MNIST



- Street View House Numbers

- large-scale scene understanding (LSUN)

Evaluation

- quality *inception score*:

$$\exp(\mathbb{E}_x \mathrm{KL}(\Pr(y|x)||\Pr(y)))$$

  - high-quality images should lead to high confidence in classification, and high inception score.
- diversity *diversity score*:

$$\exp(-\mathrm{KL}(\Pr(y)||\Pr_*(y))$$

  - diversity of generated images measured by diversity score becomes large (approaching to 1) when generated class distribution mimics real class distribution.

MNIST Performance Comparisons

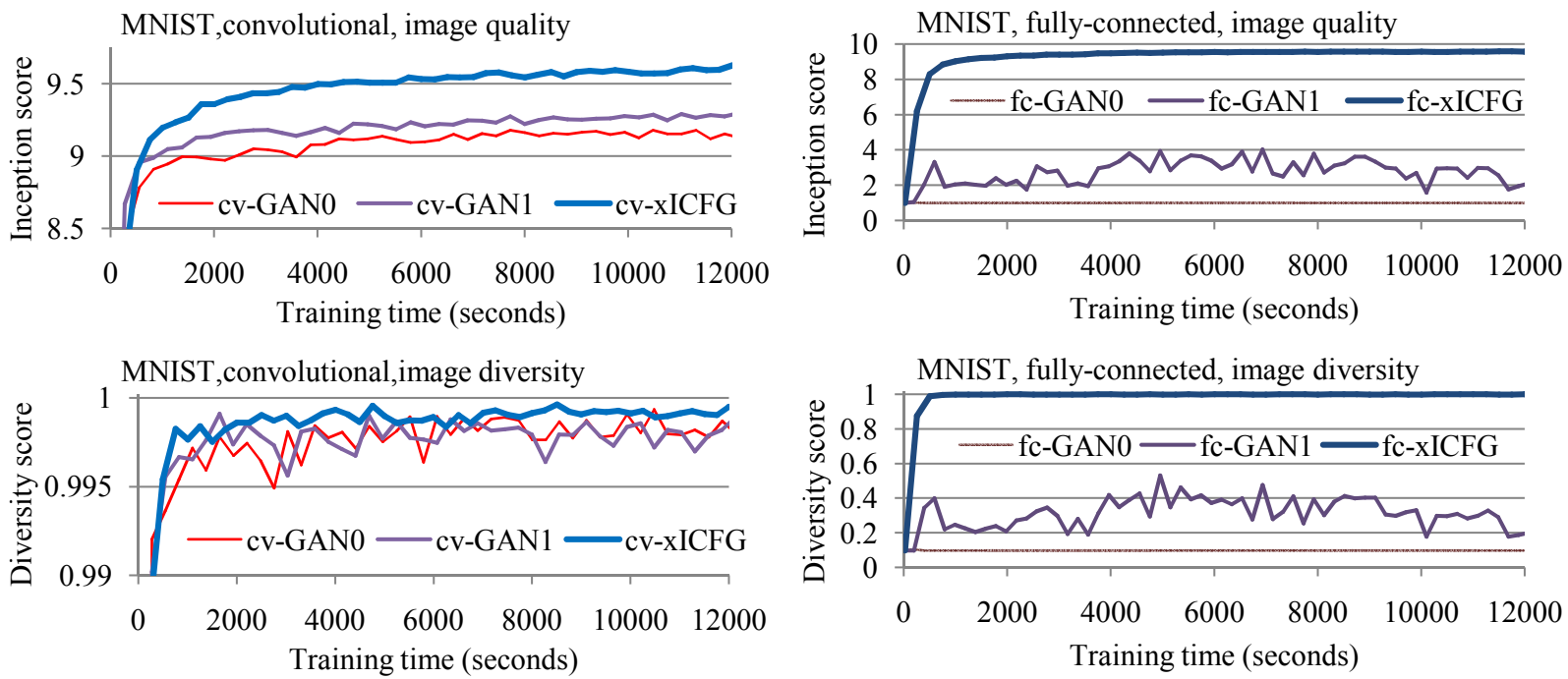
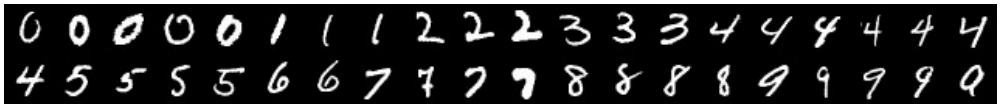


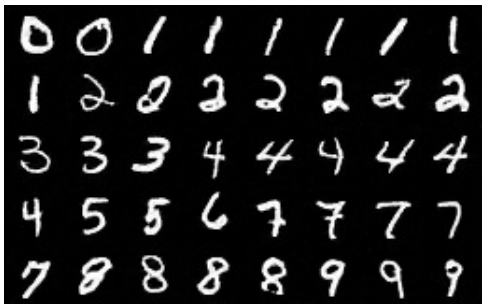
Figure: MNIST. Image quality (upper) and diversity (lower) with convolutional ('cv') approximator/generator (left) and fully-connected ('fc') approximator/generator (right).

- cv: xICFG produces higher scores than GAN.
- fc: xICFG performs well but GAN of both types fails.

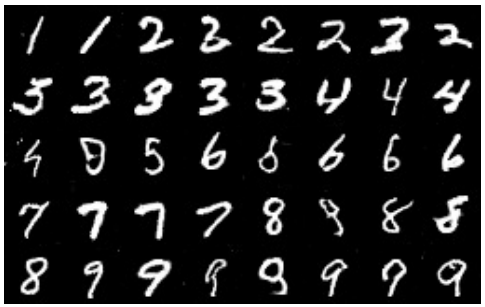
Generated Images (MNIST)



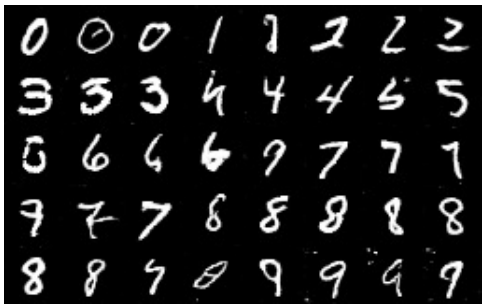
(a) Real images



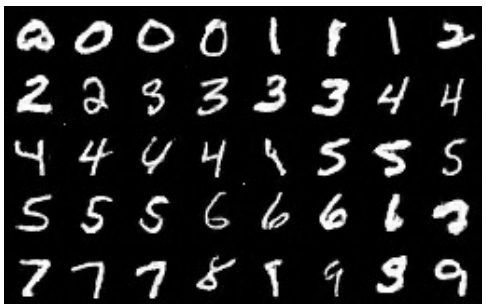
(b) cv-xICFG (9.60)



(c) cv-GAN1 (9.30)

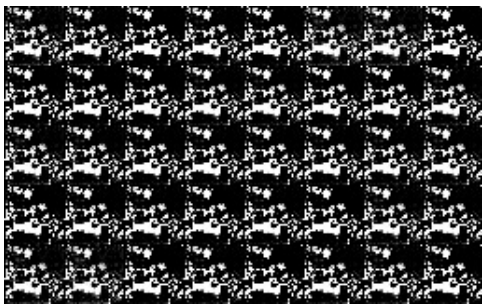


(d) cv-GAN0 (9.13)



(e) fc-xICFG (9.56)

(f) fc-GAN1 (2.57)



(g) fc-GAN0 (1.00)

Figure: MNIST. (a) Real images. (b-g) Generated images. The numbers in the parentheses are the inception scores averaged over 10K images.

House Number Performance Comparisons

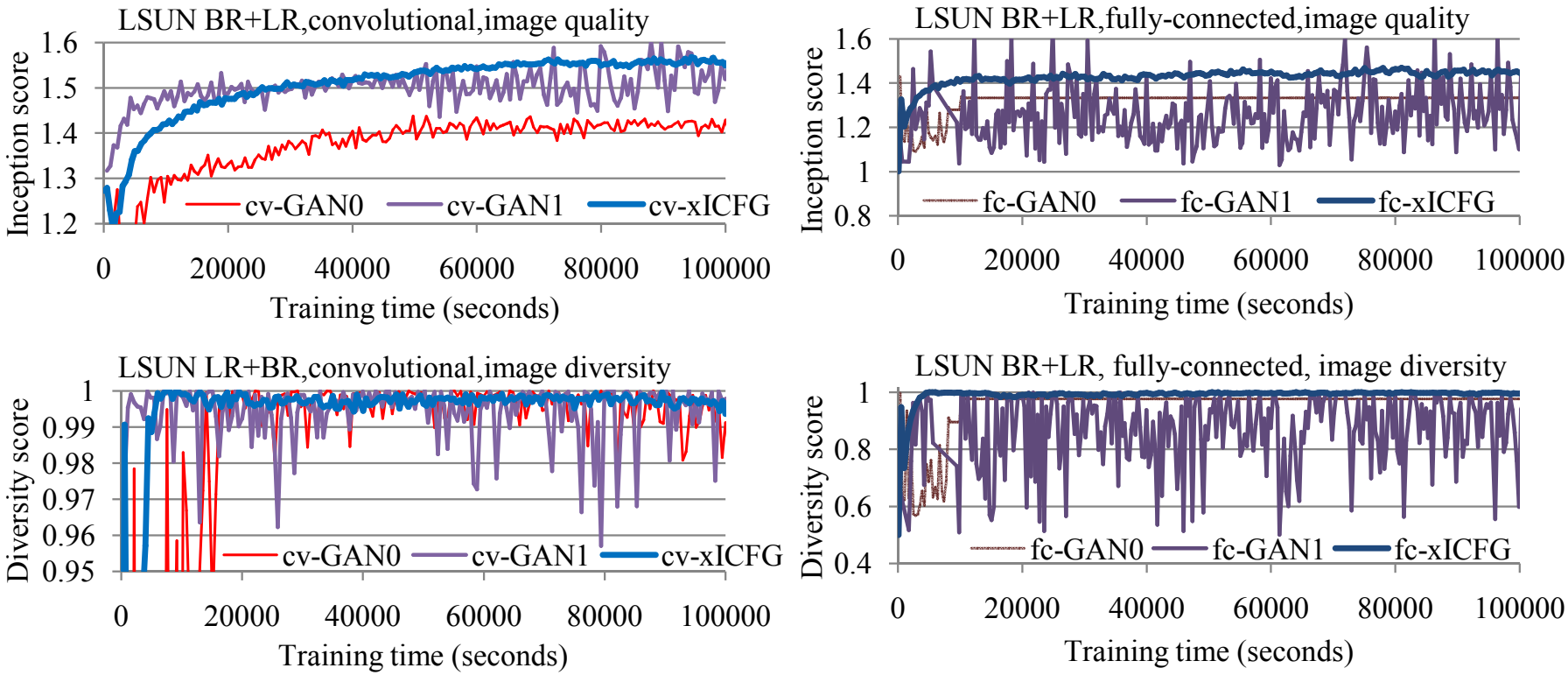
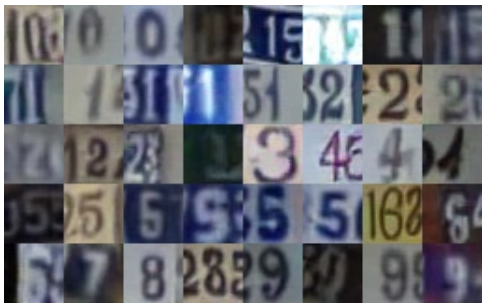


Figure: LSUN BR+LR. Image quality (upper) and diversity (lower). With the convolutional (left) and the fully-connected (right) approximator/generator.

Generated Images (House Number)

(a) Real images



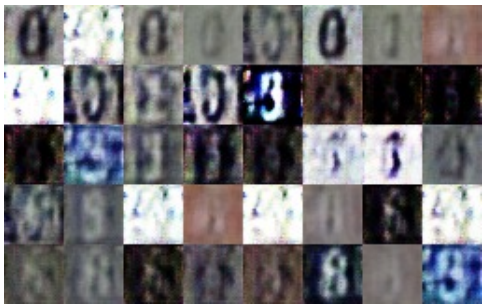
(b) cv-xICFG (8.09)

(c) cv-GAN1 (7.29)

(d) cv-GAN0 (5.90)

(e) fc-xICFG (7.65)

(f) fc-GAN1 (4.48)



(g) fc-GAN0 (3.51)

Figure: SVHN. (a) Real images. (b-g) Generated images. The numbers in the parentheses are the inception scores averaged over 10K images.

LSUN Performance Comparisons

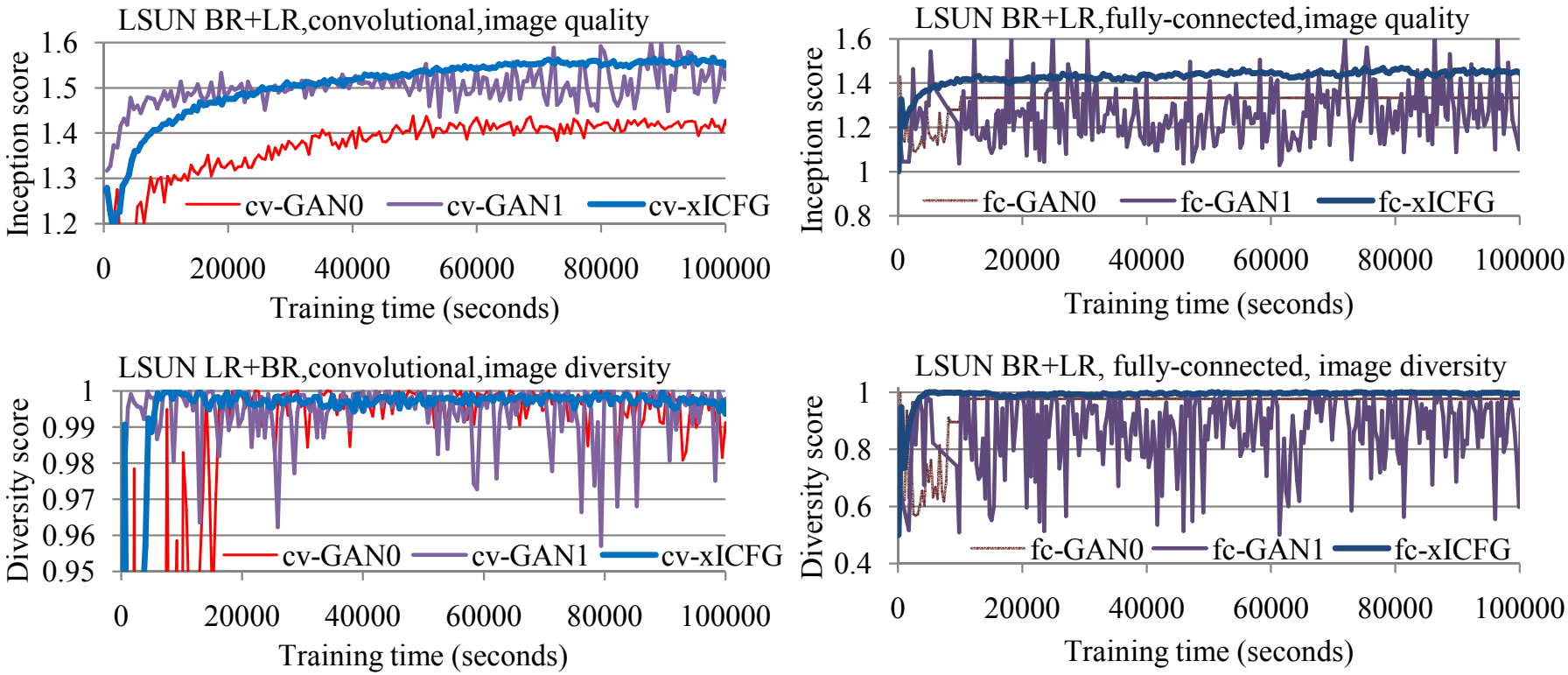


Figure: LSUN BR+LR. Image quality (upper) and diversity (lower). With the convolutional (left) and the fully-connected (right) approximator/generator.

Generated Images (LSUN)

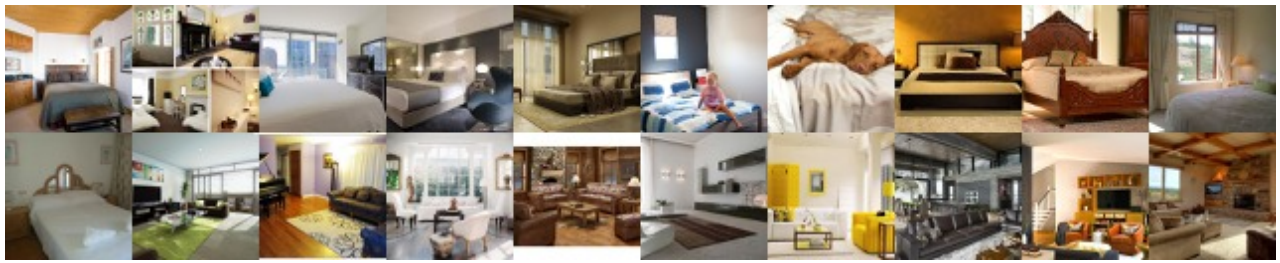
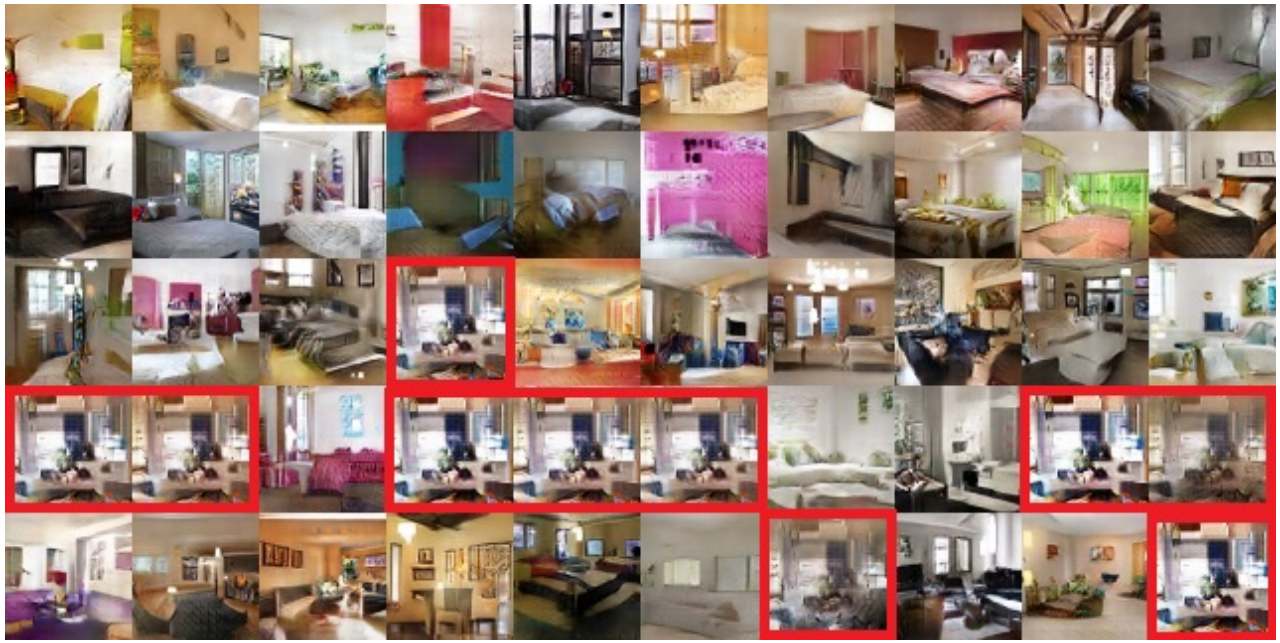


Figure: LSUN bedrooms & living rooms. Real images from the training set.

(a) cv-xICFG (1.55). High quality. No signs of modal collapse.

Generated Images (LSUN)



(a) cv-GAN1 (1.55). High quality. But it suffers from modal collapse; see the similar images marked by red.

Figure: LSUN bedrooms & living rooms. Generated using the convolutional appoximator/generator after 100K seconds of training. GAN1's modal collapse started after about 50K seconds of training.

WGAN: a variation of GAN to improve stability

WGAN uses a different loss function than log-loss

It claims to improve stability of GAN

The CFG procedure claims improved stability as well

- by making optimization easier

WGAN: a variation of GAN to improve stability

WGAN uses a different loss function than log-loss

It claims to improve stability of GAN

The CFG procedure claims improved stability as well

- by making optimization easier

How does CFG compare to WGAN?

Comparison with WGAN: convolutional networks

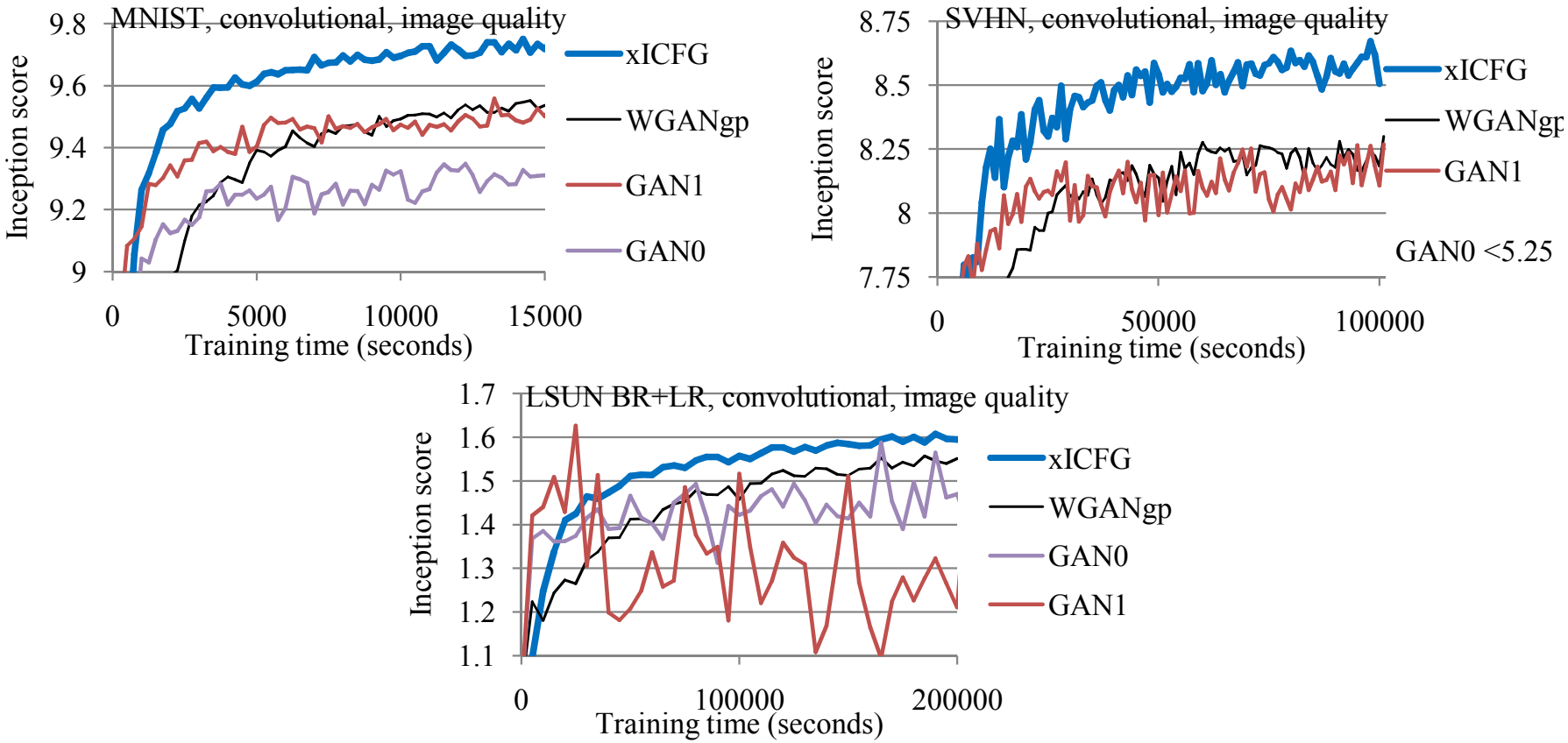


Figure: Image quality. Convolutional networks. The legends are sorted from the best to the worst. xICFG outperforms the others.

WGAN: fully-connected networks

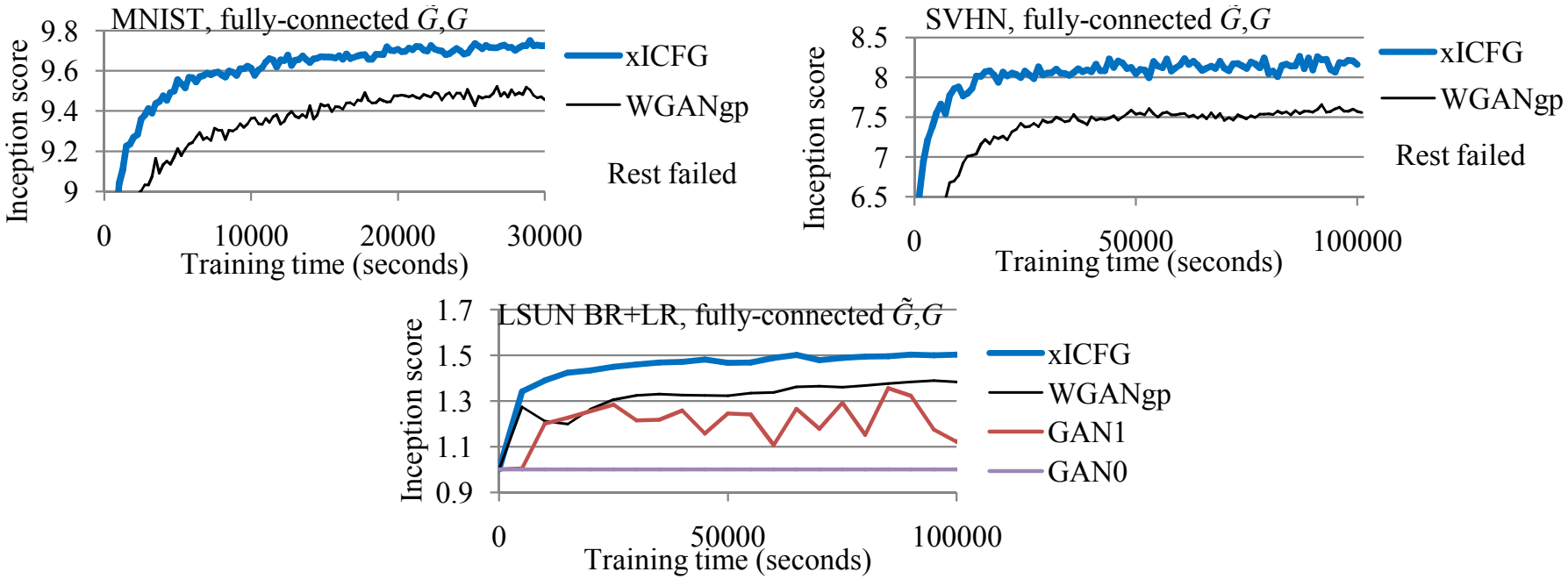


Figure: Image quality. Fully-connected approximator/generator. The legends are sorted from the best to the worst. xICFG outperforms the others.

Image Example: Creativity in LSUN (Tower & Bridge)

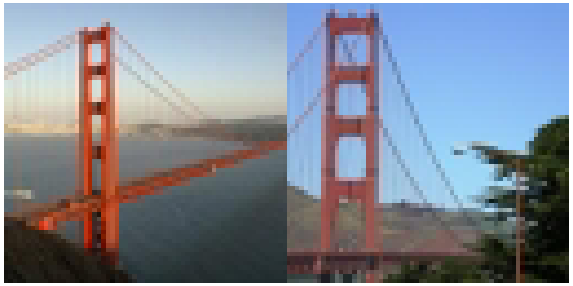
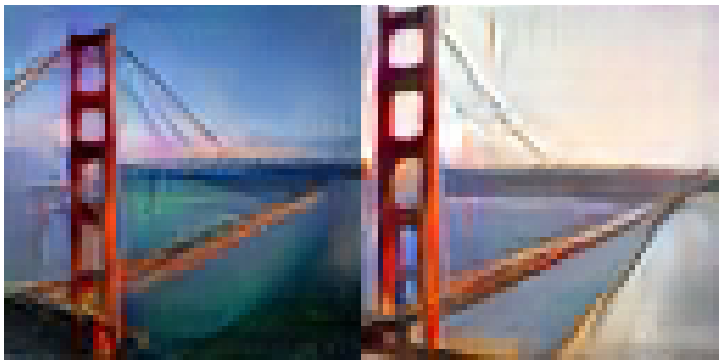
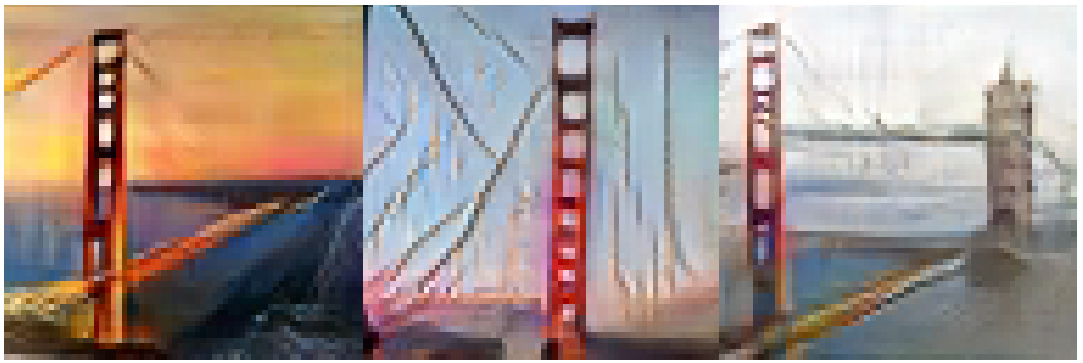


Figure: Real Golden Gate Bridge images: the red tower has 4 grids



(a) "Realistic"



(b) "Creative"

(a) Images generated by xICFG that resemble Golden Gate Bridge

(b) Images generated by xICFG that *modify* Golden Gate Bridge

Summary

In the generative adversarial learning setting:

- GAN's minimax formulation optimizes JS-divergence
- GAN's algorithm is not stable.
- the practical use of logD trick inconsistent with the minimax formulation

The optimization problem is hard and unstable

Summary

This work: change optimization, gradient descent in function space

- Learn generator $G(z)$ using CFG (composite functional gradient).
- Theory: minimizes KL-divergence
- Theory: lead to the new stable algorithm xICFG.
- Theory: explains the logD trick of GAN.
- Experiments: xICFG performs better than GAN/WGAN

Final Remarks: is deep learning Alchemy or science?

Big Theoretical Challenge

Theoretical research in the *AI Era* needs to be relevant

- either improving fundamental understanding
- or solving an important practical problem

Final Remarks: is deep learning Alchemy or science?

Big Theoretical Challenge

Theoretical research in the *AI Era* needs to be relevant

- either improving fundamental understanding
- or solving an important practical problem

Encourages problem driven theory $\rightarrow$ Practical Impact

- understanding how algorithms work and their limitations
- lead to new practical algorithms