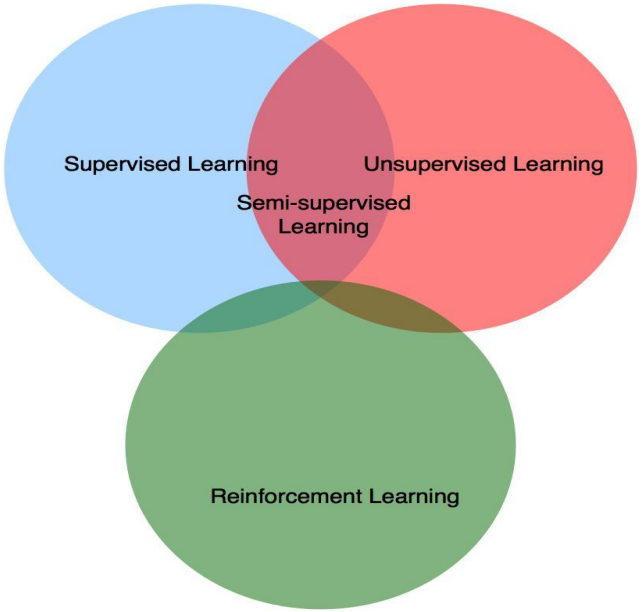
Generative Adversarial Networks (GAN)

The coolest idea in ML in the last twenty years - Yann Lecun

2017.03 Namju Kim (namju.kim@kakaobrainl.com)

Introduction

Taxonomy of ML



From David silver, Reinforcement learning (UCL course on RL, 2015).

Supervised Learning

- Find deterministic function ***f*** : $y = f(x)$, x : data, y : **label**

Supervised Learning

- How to implement the following function prototype.

```
def get_label(image):

    label_index = 0

    # TODO
    # do something to find the label index of given image

    return label_index
```

Supervised Learning

- Challenges
  - Image is **high dimensional** data
    - 224 x 224 x 3 = 150,528
  - Many **variations**
    - viewpoint, illumination, deformation, occlusion, background clutter, intraclass variation

Supervised Learning

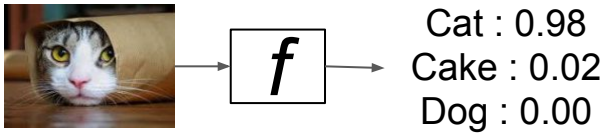
- Solution
  - **Feature** Vector
    - 150,528 → 2,048
  - Synonyms
    - Latent Vector, Hidden Vector, Unobservable Vector, Feature, Representation

Supervised Learning

```python
def get_feature(image):

    z = [0.] * 2048

    # TODO
    # do something to extract the feature of given image

    return z


def get_label(image):

    label_index = 0

    # get feature
    z = get_feature(image)

    # TODO
    # do something to find the label index of given feature

    return label_index
```

Supervised Learning

- More flexible solution
  - Get probability of the label for given data instead of label itself

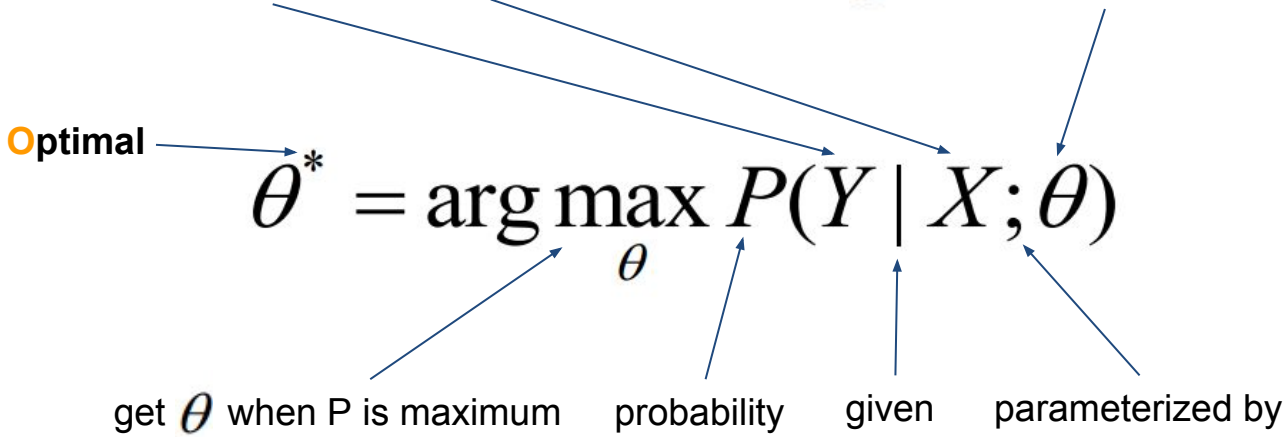


Supervised Learning

```python
def get_feature(image):

    z = [0.] * 2048

    # TODO
    # do something to extract the feature of given image

    return z


def get_prob(z):

    p = [0.] * 10

    # TODO
    # Getting the probability of given features is a simpler job,
    # if feature is good enough

    return p


def get_label(image):

    # get feature
    z = get_feature(image)

    # get prob
    p = get_prob(z)

    return np.argmax(p)
```

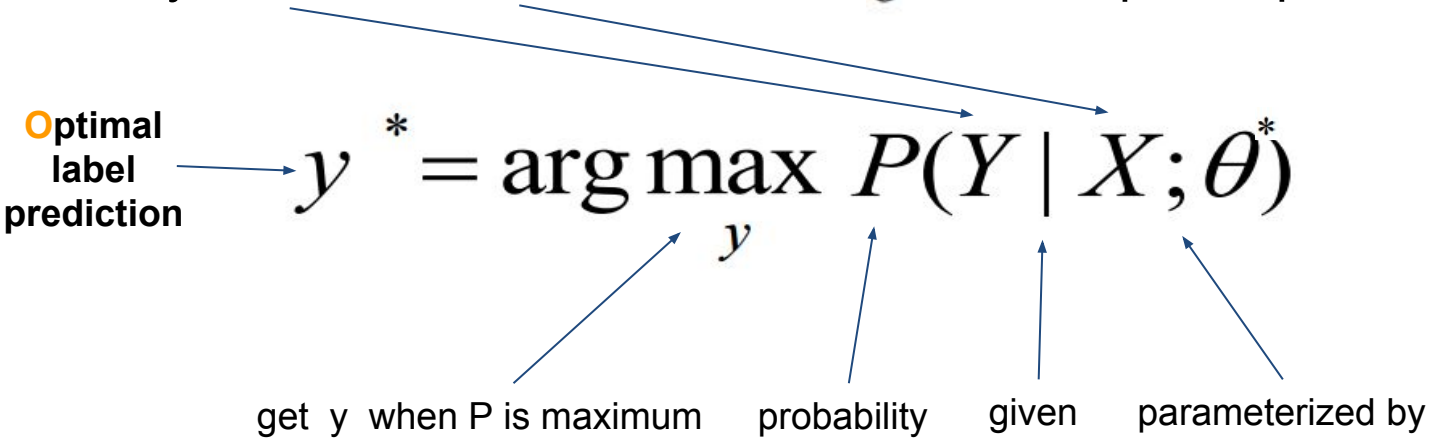
Supervised Learning

- Mathematical notation of **optimization**
  - y : label, x : data, z : latent, θ : learnable parameter



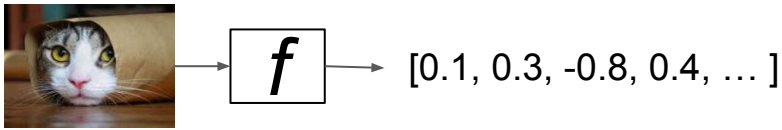
Supervised Learning

- Mathematical notation of **classifying** (greedy policy)
  - y : label, x : data, z : latent, θ^*: fixed optimal parameter



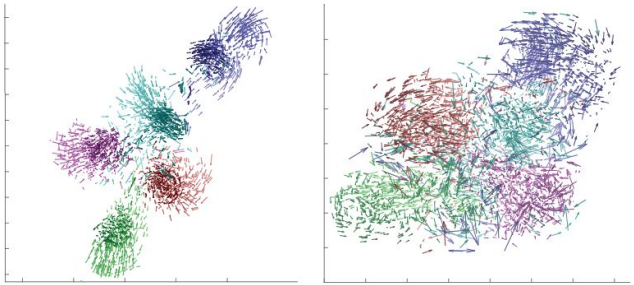
Unsupervised Learning

- Find deterministic function ***f*** : $z = f(x)$, x : data, z : **latent**



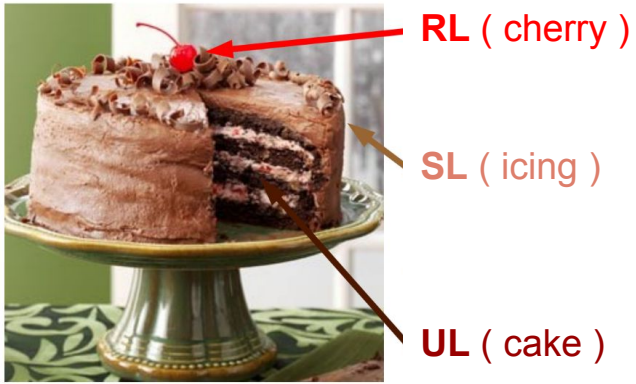
Good features

- Less redundancy
- Similar features for similar data
- High fidelity



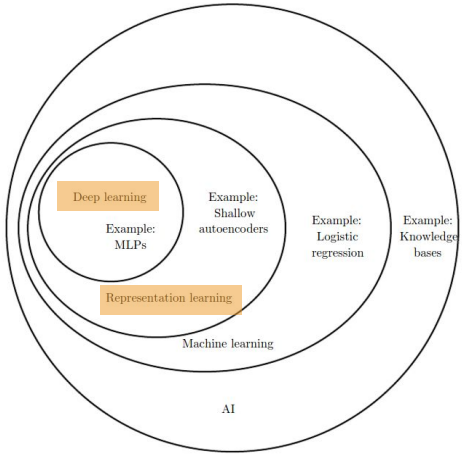
Good

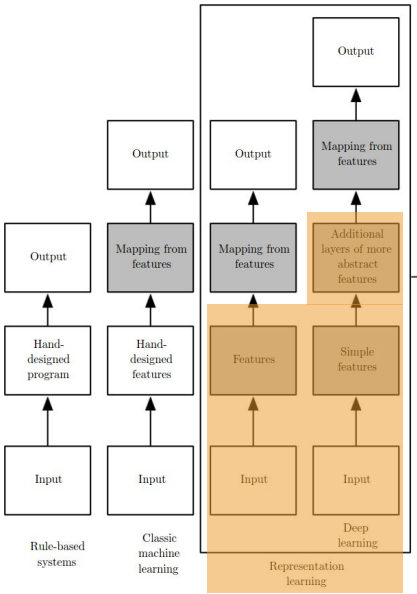
Bad



From Yann Lecun, Predictive Learning (NIPS tutorial, 2016).

Unsupervised Learning





E2E
(end-to-end)

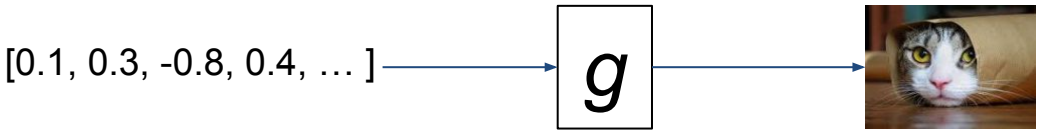
From Ian Goodfellow, Deep Learning (MIT press, 2016).

Unsupervised Learning

- More challenging than supervised learning :
  - No label or curriculum → self learning
- Some NN solutions :
  - Boltzmann machine
  - Auto-encoder or Variational Inference
  - Generative Adversarial Network

Generative Model

- Find generation function ***g*** : ***x = g(z)***, x : data, z : **latent**



Unsupervised learning vs. Generative model

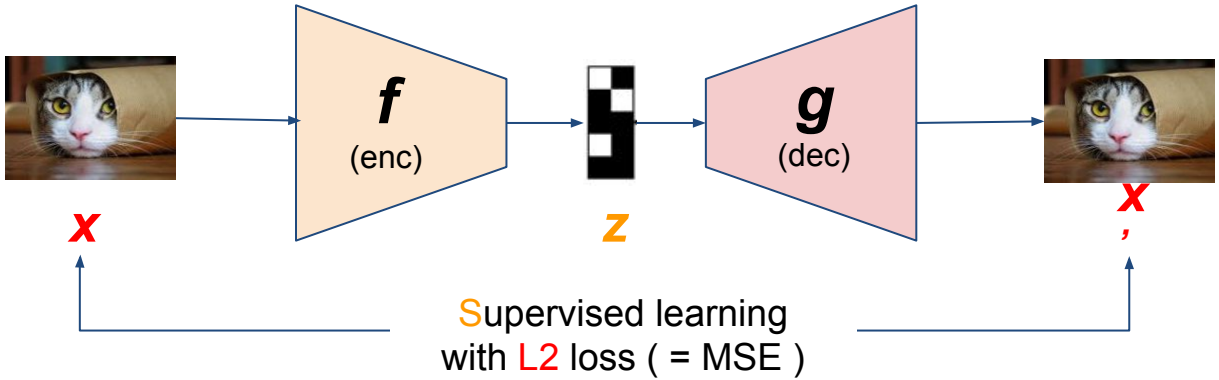
- ***z* = *f*(*x*)** *vs.* ***x* = *g*(*z*)**
- ***P*(*z*|*x*)** *vs.* ***P*(*x*|*z*)**
- ***Encoder*** *vs.* ***Decoder (Generator)***
  - P(x, z) needed. (cf : P(y|x) in supervised learning)
    - P(z|x) = P(x, z) / P(x) → P(x) is intractable (ELBO)
    - P(x|z) = P(x, z) / P(z) → P(z) is given. (prior)

$$\theta^* = \arg\max_{\theta} P(X, \boldsymbol{Z}; \theta)$$

Autoencoders

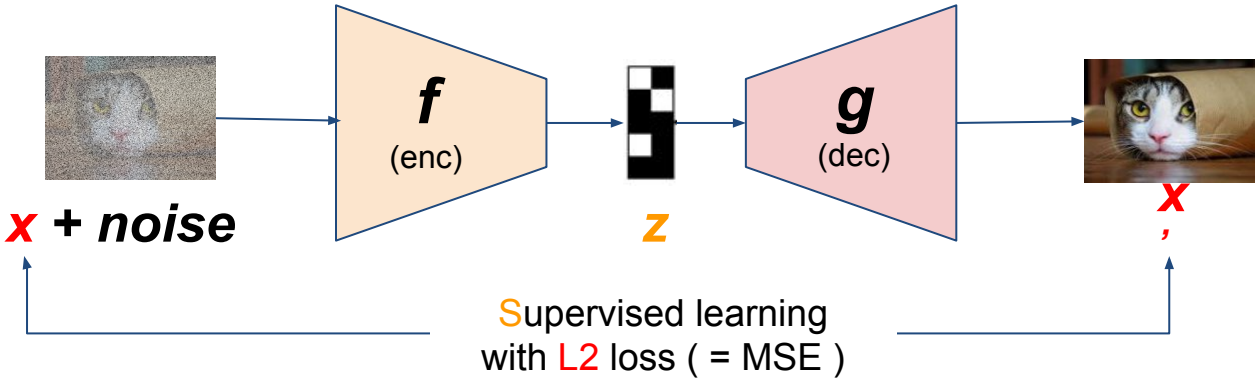
Stacked autoencoder - SAE

- Use data itself as label → Convert UL into reconstruction SL
- $z = f(x)$, $x = g(z)$ → $x = g(f(x))$
- https://github.com/buriburisuri/sugartensor/blob/master/sugartensor/example/mnist_sae.py



Denoising autoencoder - DAE

- Almost same as SAE
- Add random noise to input data → Convert UL into denoising SL
- https://github.com/buriburisuri/sugartensor/blob/master/sugartensor/example/mnist_dae.py



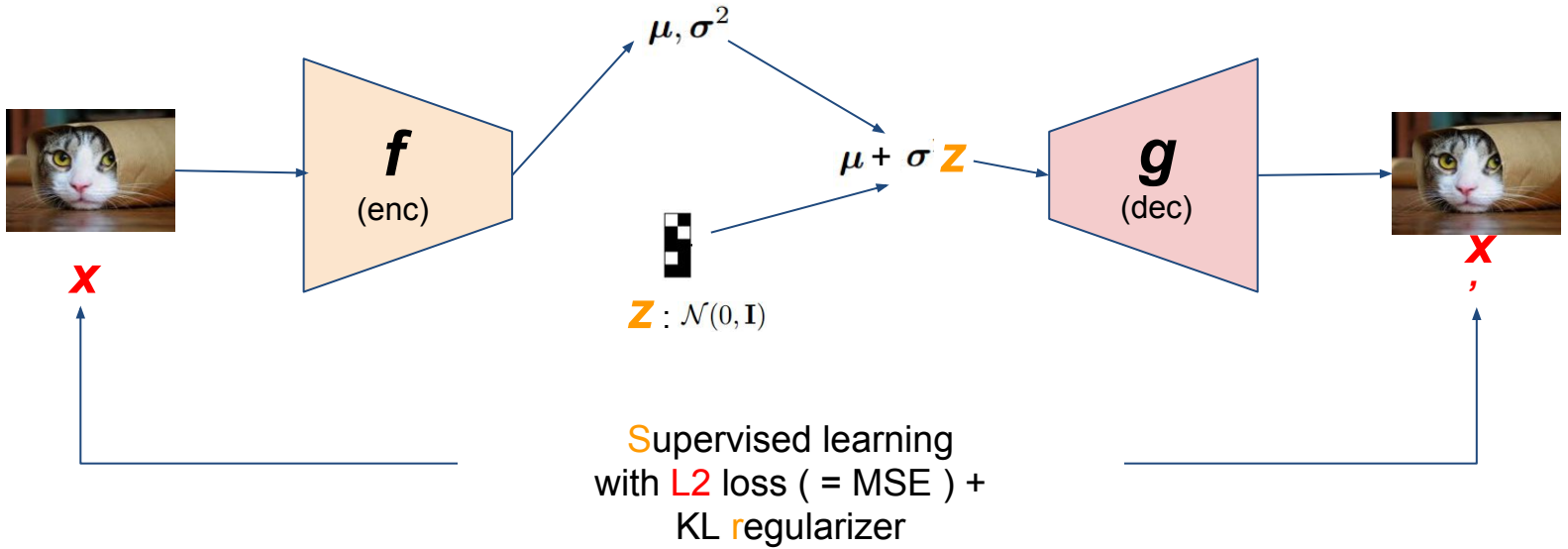
Variational autoencoder - VAE

- Kingma et al, “Auto-Encoding Variational Bayes”, 2013.

- Generative Model + Stacked Autoencoder
  - Based on **Variational approximation**
  - other approaches :
    - Point estimation (MAP)
    - Markov Chain Monte Carlo (MCMC)

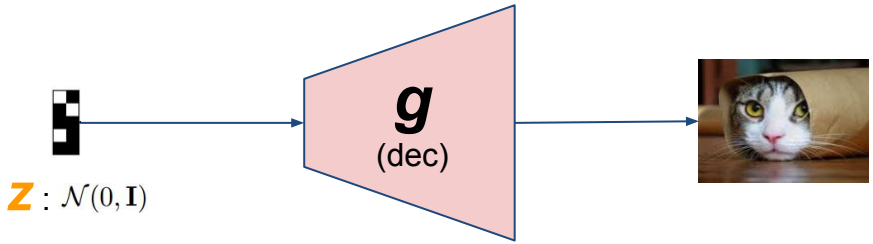
Variational autoencoder - VAE

- Training
- https://github.com/buriburisuri/sugartensor/blob/master/sugartensor/example/mnist_vae.py



Variational autoencoder - VAE

- Generating
- https://github.com/buriburisuri/sugartensor/blob/master/sugartensor/example/mnist_vae_eval.py



Variational autoencoder - VAE

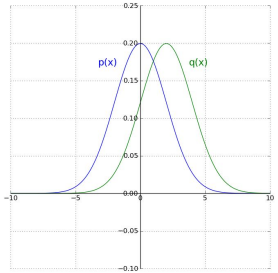
- Question : $\mu + \sigma z \neq z \rightarrow p(z|x) \neq \mathcal{N}(0, \mathbf{I})$
- Solution :
  - $p(z|x)$ is intractable but MCMC is slow.
  - Introduce auxiliary variational function $q(z|x)$
  - Approximate $q(z|x)$ to $p(z|x)$ by ELBO
  - Make q(z|x) $\simeq \mathcal{N}(0, \mathbf{I})$

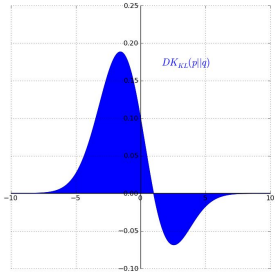
Variational autoencoder - VAE

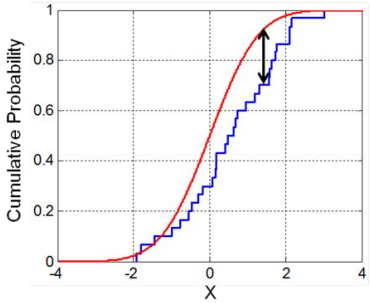
- Simply : $\mu \rightarrow 0, \quad \sigma \rightarrow 1$
- KLD Regularizer :
  - $D_{KL}((q_{\phi}(\mathbf{z})||p_{\theta}(\mathbf{z}))$
  - Gaussian case :
    - $-\frac{1}{2}\sum_{j=1}^{J}\left(1+\log((\sigma_j)^2)-(\mu_j)^2-(\sigma_j)^2\right)$
    - If we fix σ = 1, (Practical approach)
      - $\frac{1}{2}(\mu_j)^2 \rightarrow$ MSE (μ)

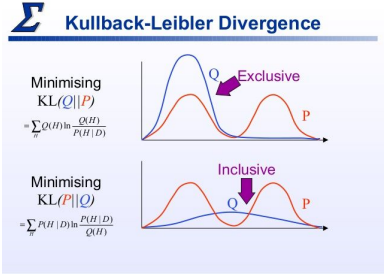
Variational autoencoder - VAE

- KLD (Kullback-Leibler divergence)
- A sort of distribution difference measure
  - cf : KS (Kolmogorov-smirov) test, JSD(Jensen-shannon Divergence)









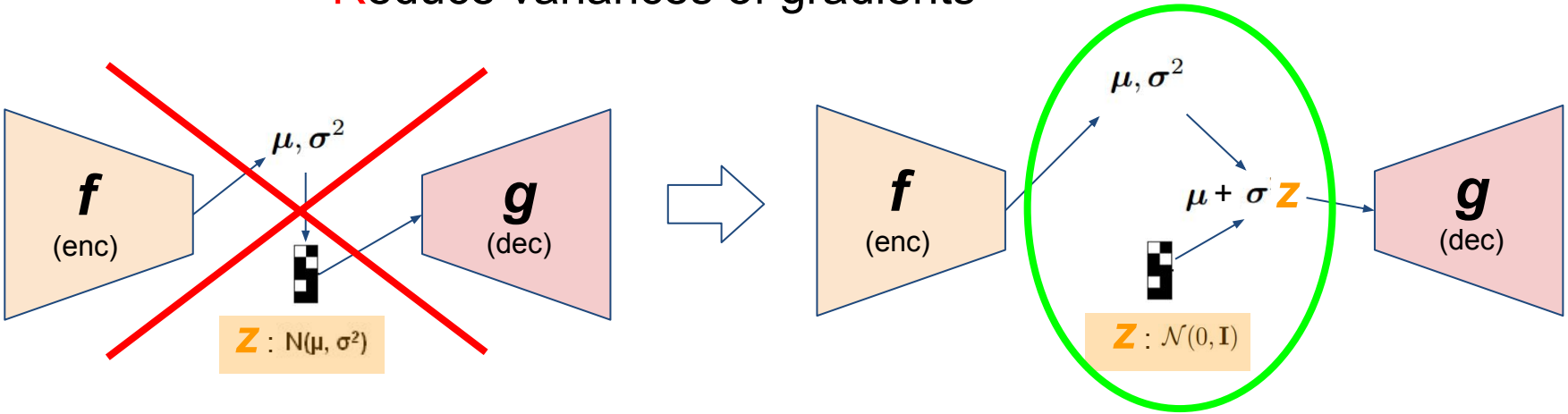
Variational autoencoder - VAE

- Find problems in the following code

```
# encoding input image
mu = encode(x)

# generate normal random number with given mu, sig
z = tf.random_normal((batch_size, num_dim), mean=mu, stddev=tf.ones_like(mu))

# decode latent
xx = decode(z)

# reconstruction loss
loss_recon = tf.reduce_mean(tf.square(x -xx), axis=[1, 2, 3])
# KLD loss
loss_kld = 0.5 * tf.reduce_mean(tf.square(mu), axis=[1])

# total loss
loss = loss_recon + loss_kld
```

Variational autoencoder - VAE

- Reparameterization trick
  - Enable back propagation
  - Reduce variances of gradients



Variational autoencoder - VAE

- This is called ‘Reparameterization trick’

```
# encoding input image
mu = encode(x)

# generate normal(0, 1) random number
r = tf.random_normal((batch_size, num_dim))

# final latent vector
z = mu + r

# decode latent
xx = decode(z)

# reconstruction loss
loss_recon = tf.reduce_mean(tf.square(x - xx), axis=[1, 2, 3])
# KLD loss
loss_kld = 0.5 * tf.reduce_mean(tf.square(mu), axis=[1])

# total loss
loss = loss_recon + loss_kld
```

Variational autoencoder - VAE

- Results

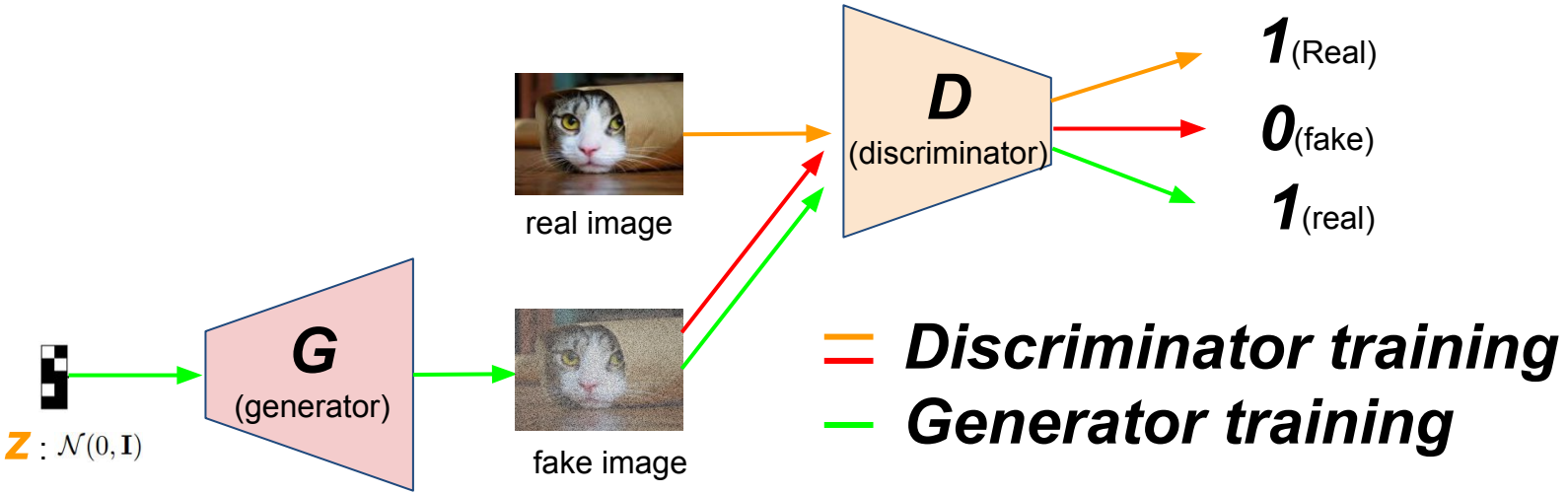
Generative Adversarial Networks

Generative Adversarial Networks - GAN

- Ian Goodfellow et al, “Generative Adversarial Networks”, 2014.
  - **Learnable** cost function
  - Mini-Max game based on Nash Equilibrium
    - Little assumption
    - High fidelity
  - Hard to training - no guarantee to equilibrium.

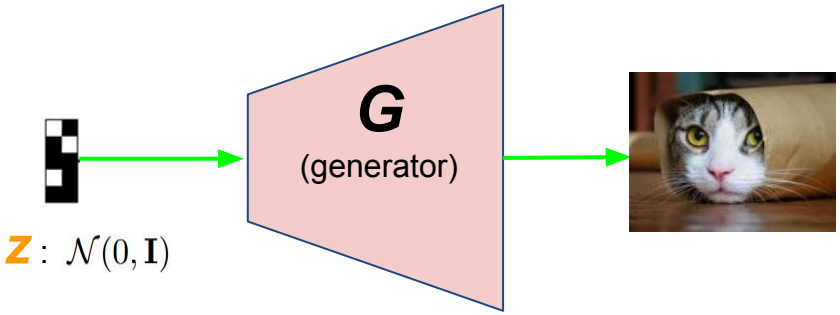
Generative Adversarial Networks - GAN

- Alternate training
- https://github.com/buriburisuri/sugartensor/blob/master/sugartensor/example/mnist_gan.py



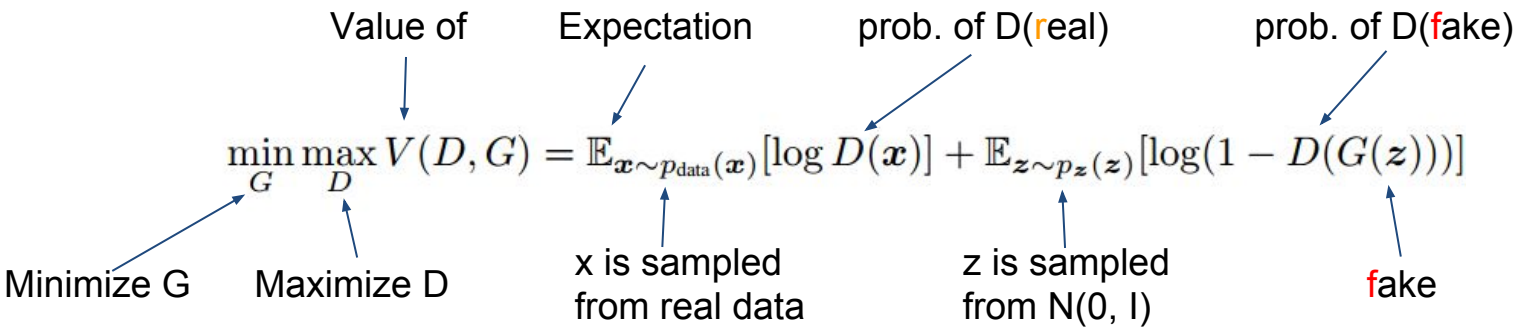
Generative Adversarial Networks - GAN

- Generating
- https://github.com/buriburisuri/sugartensor/blob/master/sugartensor/example/mnist_gan_eval.py



Generative Adversarial Networks - GAN

- Mathematical notation



Generative Adversarial Networks - GAN

- Mathematical notation - discriminator

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

Maximize prob. of D(real)

Minimize prob. of D(fake)

BCE(binary cross entropy) with label 1 for real, 0 for fake.

(Practically, **CE** will be OK. or more plausible.)

Generative Adversarial Networks - GAN

- Mathematical notation - generator

$$\min_G V(D,G) = \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] \simeq \max_G \mathbb{E}_{z\sim p_z(z)}[\log(D(G(z)))]$$

Maximize prob. of D(fake)

BCE(binary cross entropy) with label 1 for fake.

(Practically, **CE** will be OK. or more plausible.)

Generative Adversarial Networks - GAN

- Mathematical notation - equilibrium

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \longrightarrow 0.5$$

$$\begin{aligned} C(G) &= \max_D V(G, D) \\ &= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}[\log(1 - D_G^*(G(\boldsymbol{z})))] \\ &= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))] \\ &= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x} \sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] \end{aligned}$$

$$C(G) = -\log(4) + KL\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \middle\| \frac{p_{\text{data}} + p_g}{2}\right)$$

$$2 \cdot JSD\left(p_{\text{data}} \| p_g\right)$$

Jansen-Shannon divergence

Generative Adversarial Networks - GAN

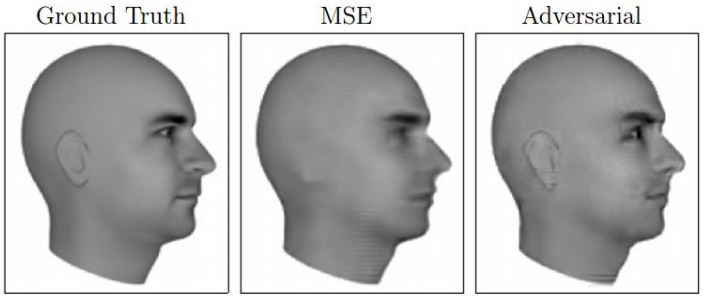
```
# generate fake image
fake = generator(tf.random_normal((batch_size, num_dim)))

# discriminator for real image
disc_real = discriminator(x)

# discriminator for fake image
disc_fake = discriminator(fake, reuse=True)

# loss for discriminator
loss_disc_real = tf.nn.sigmoid_cross_entropy_with_logits(disc_real, targets=tf.ones(batch_size))
loss_disc_fake = tf.nn.sigmoid_cross_entropy_with_logits(disc_fake, targets=tf.zeros(batch_size))
loss_disc = 0.5 * loss_disc_real + 0.5 * loss_disc_fake

# loss for generator
loss_gen = tf.nn.sigmoid_cross_entropy_with_logits(disc_fake, targets=tf.ones(batch_size))

# train ops
train_disc = tf.sg_optim(loss_disc, lr=0.0001, category='discriminator')
train_gen = tf.sg_optim(loss_gen, lr=0.001, category='generator')

# alternate training
sess.run(train_disc)[0]  # training discriminator
sess.run(train_gen)[0]   # training generator
```

Generative Adversarial Networks - GAN

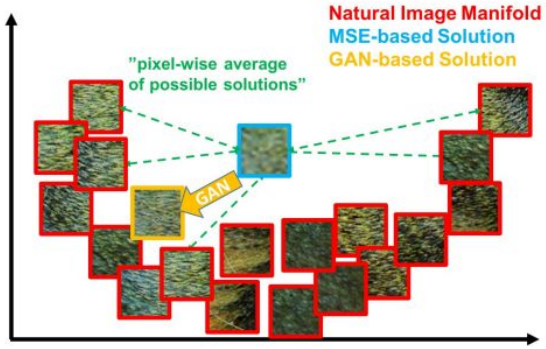
- Result

Generative Adversarial Networks - GAN

- Why blurry and why sharper ?



From Ian Goodfellow, Deep Learning (MIT press, 2016)



From Christian et al, Photo-realistic single image super-resolution using generative adversarial networks, 2016

Training GANs

Pitfall of GAN

- No guarantee to equilibrium
  - **Mode collapsing**
  - Oscillation
  - No indicator when to finish
- All generative model
  - Evaluation metrics (Human turing test)
  - Overfitting test (Nearest Neighbor) → GAN is robust !!!
  - Diversity test
  - Wu et al, On the quantitative analysis of decoder-based generative model, 2016

DCGAN

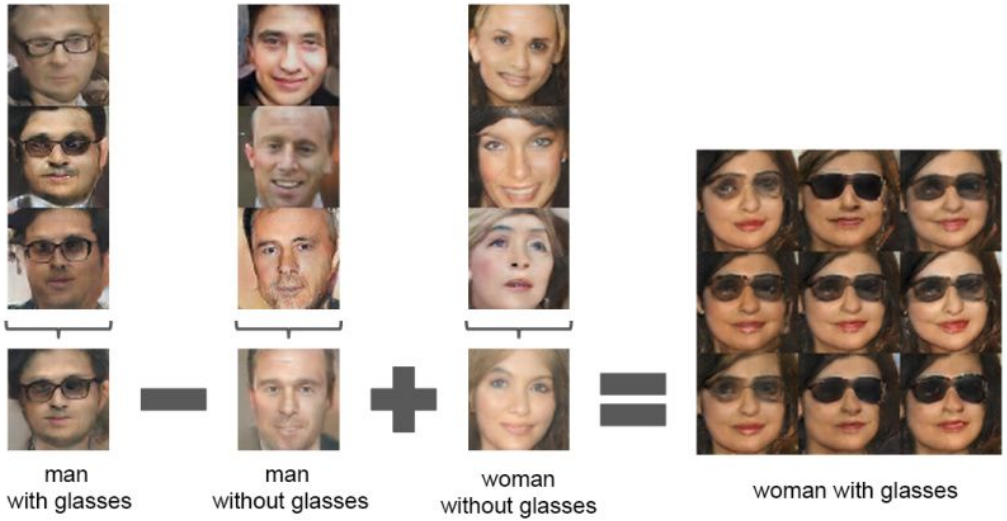
- Radford et al, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 2015
  - Tricks for gradient flow
    - Max pooling → Strided convolution or average pooling
    - Use LeakyReLU instead of ReLU
  - Other tricks
    - Use batch normal both generator and discriminator
    - Use Adam optimizer (lr = 0.0002, a = 0.9, b=0.5)

DCGAN

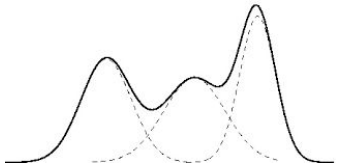
- Results

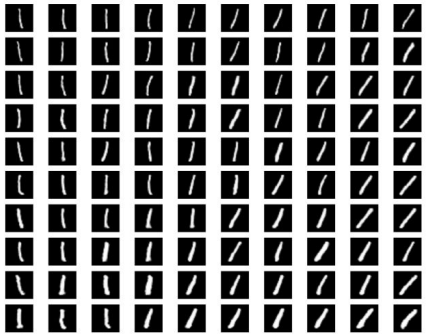
DCGAN

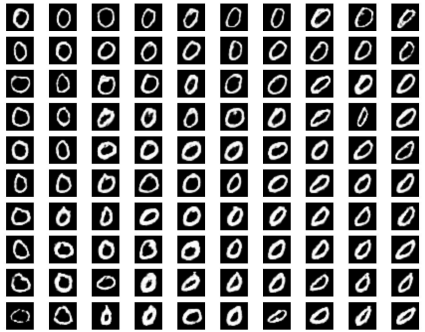
- Latent vector arithmetic

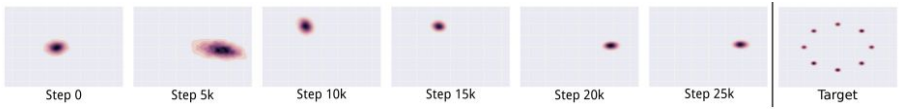


Pitfall of GAN









mode collapsing

oscillating

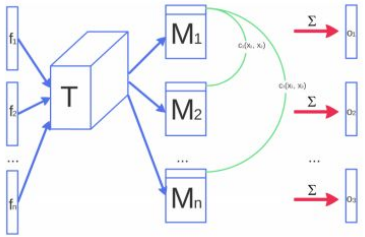
Escaping mode collapsing

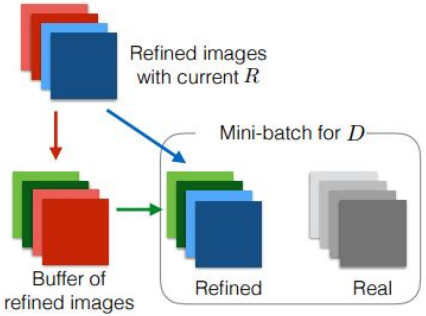
- Minibatch discrimination (*Salimans et al, 2016)* ← slow
- Replay memory (Shrivastava et al, 2016) ← controversial
- Pull-away term regularizer (Zhao et al, 2016) ← not so good
- Unrolled GAN (Metz et al, 2016) ← not so good

Escaping mode collapsing

- Discriminator ensemble (Durugkar, 2016) ← not tested
- Latent-Image pair discrimination (Donahue et al, 2016, Dumoulin et al, 2016) ← Good (additional computing time)
- InfoGAN (Chen et al, 2016) ← Good

Escaping mode collapsing





$$f_{PT}(S) = \frac{1}{N(N-1)} \sum_i \sum_{j \neq i} \left(\frac{S_i^{\mathsf{T}} S_j}{\|S_i\| \|S_j\|} \right)^2.$$

$$\theta_D^0 = \theta_D$$

$$\theta_D^{k+1} = \theta_D^k + \eta^k \frac{\mathrm{d}f\left(\theta_G, \theta_D^k\right)}{\mathrm{d}\theta_D^k}$$

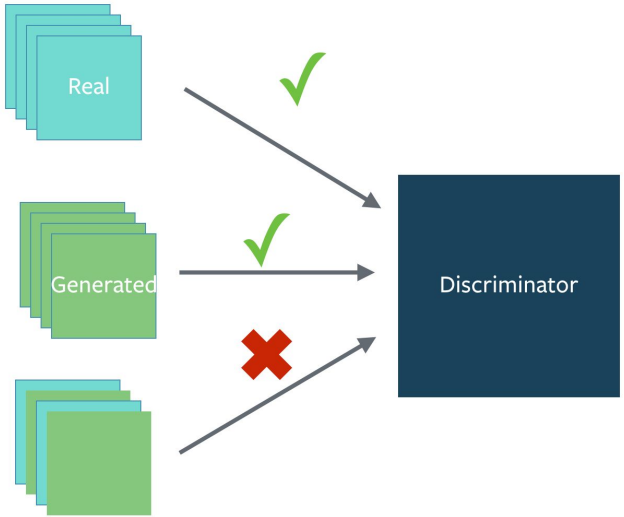
$$\theta_D^* = \lim_{k \to \infty} \theta_D^k,$$

Other tricks

- *Salimans et al, Improved Techniques for Training GANs, 2016*
- *https://github.com/soumith/ganhacks*
  - One-sided label smoothing ← not sure
  - Historical averaging ← Unrolled GAN is better
  - Feature matching ← working
  - Use noise or dropout into real/fake image ← working
  - Don't balance D and G loss ← I agree

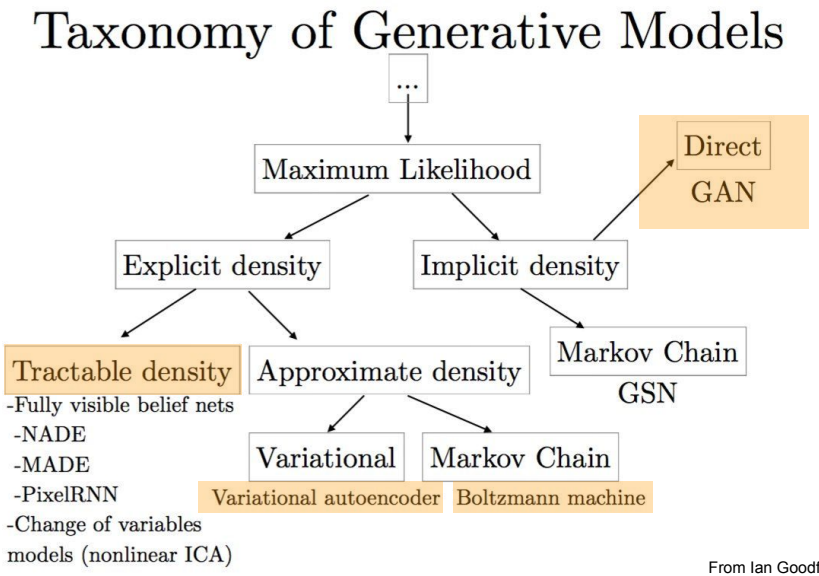
Batch normalization in GAN

- *Use in G but cautious in D*
- *Do not use at last layer of G and first layer of D*
- *IMHO, **don't use BN in D***
- *Reference BN or Virtual BN (Not sure)*



Variants of GANs

Taxonomy of Generative Models



SOTA !!! →

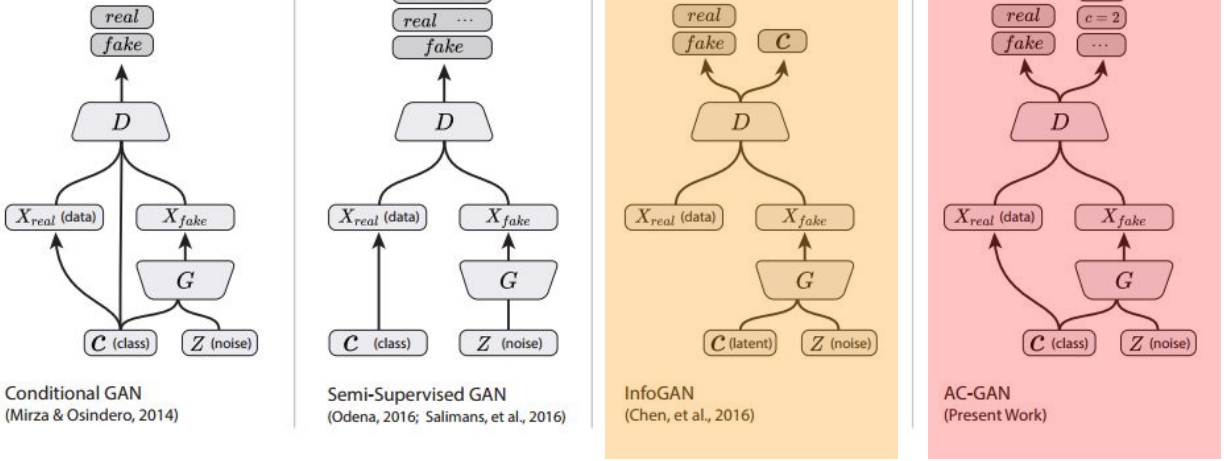
AR based model :
PixelRNN
WaveNet

From Ian Goodfellow, NIPS 2016 Tutorial : Generative Adversarial Netorks, 2016

Taxonomy of GANs

SOTA without Label

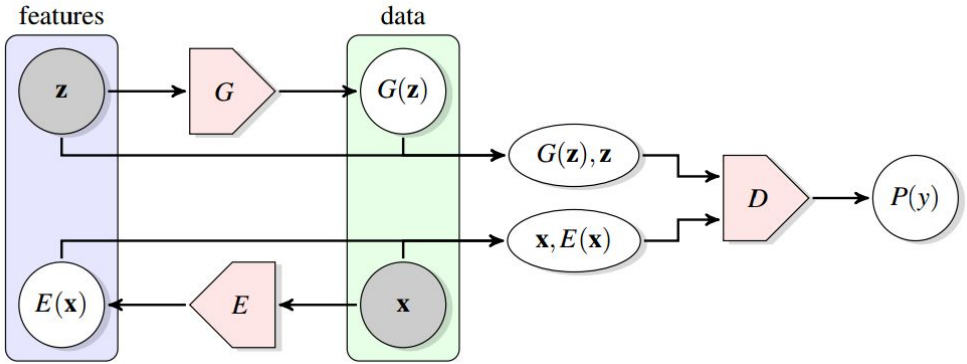
SOTA with Label



From Ordena et al, Conditional Image Synthesis With Auxiliary Classifier GANs, 2016

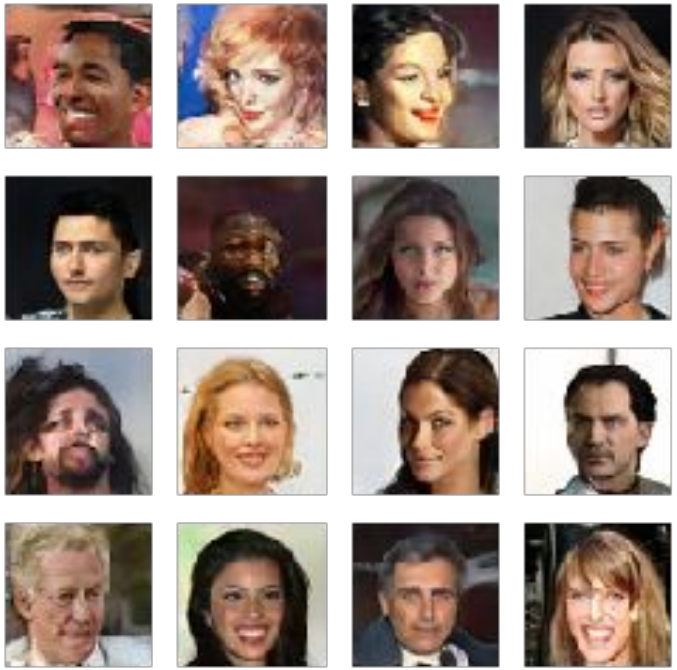
AFL, ALI

- Donahue et al, Adversarial Feature Learning, 2016
- Domoulin et al, Adversarial Learned Inference, 2016



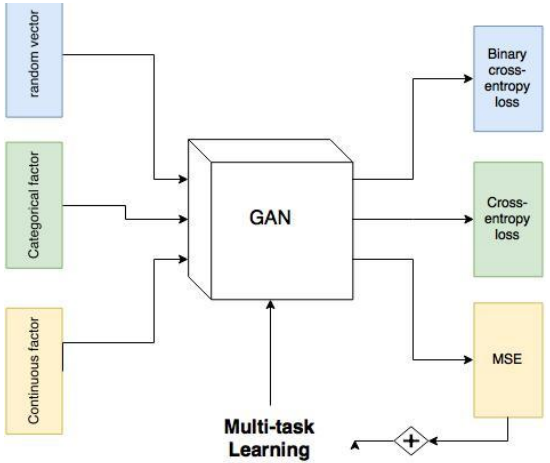
AFL, ALI

- Results



InfoGAN

- Chen et al, InfoGAN : Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets, 2016



InfoGAN

- Results

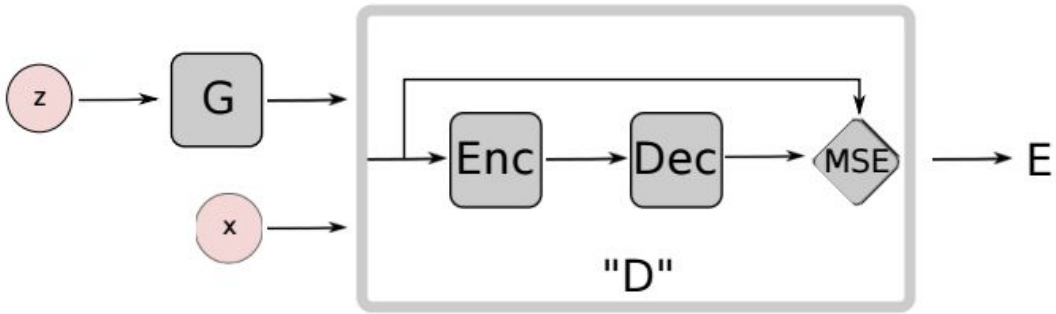
(a) Azimuth (pose) (b) Presence or absence of glasses

(c) Hair style (d) Emotion

- My implementation : https://github.com/buriburisuri/sugartensor/blob/master/sugartensor/example/mnist_info_gan.py

EBGAN

- Junbo et al, Energy-based generative adversarial networks, 2016



EBGAN

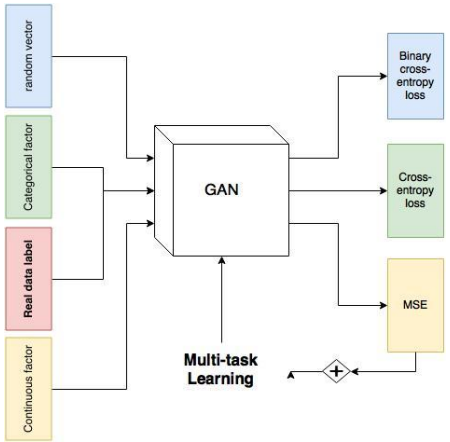
- Results

EBGAN

- My implementation : https://github.com/buriburisuri/ebgan

ACGAN

- Ordena et al, Conditional Image Synthesis with Auxiliary Classifier GANs, 2016

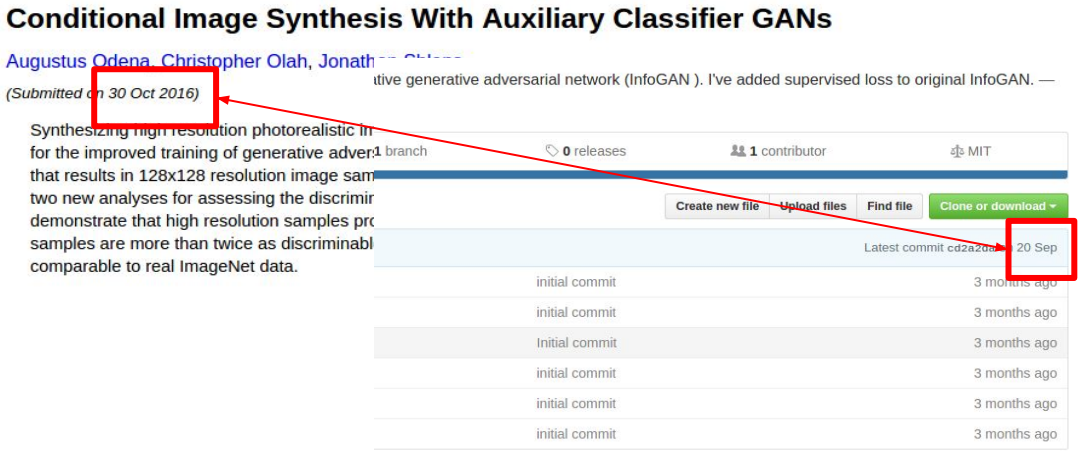


ACGAN

- Results

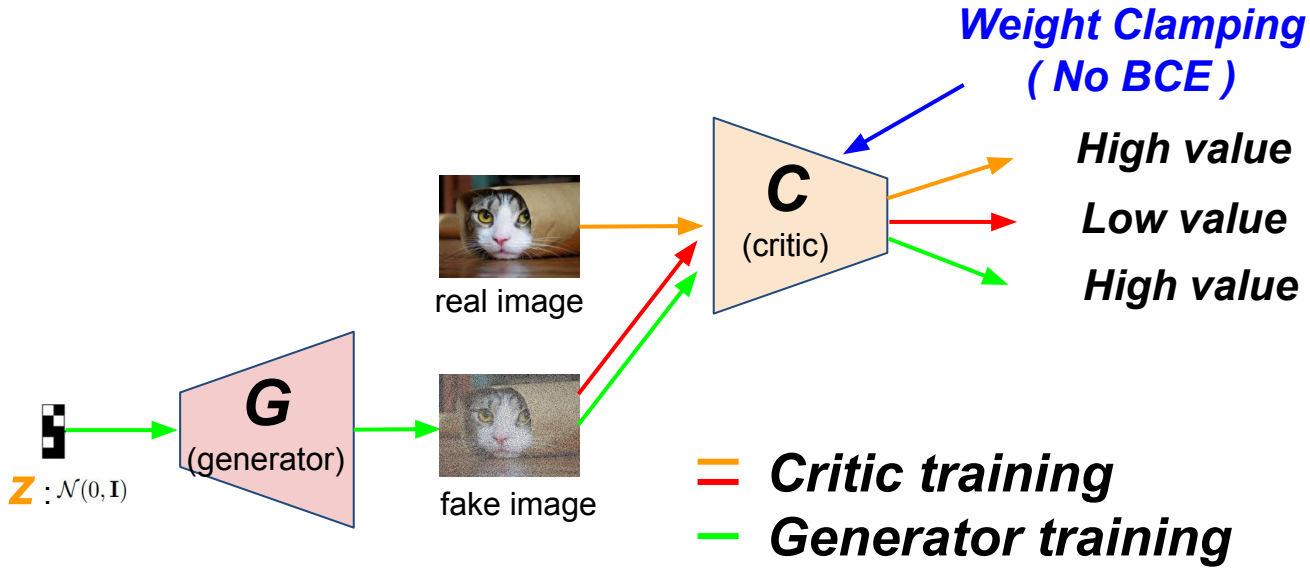
ACGAN

- My implementation : https://github.com/buriburisuri/ac-gan



WGAN

- Martin et al, Wasserstein GAN, 2017



WGAN

- Martin et al, Wasserstein GAN, 2017
  - JSD → Earth Mover Distance(=Wasserstein-1 distance)
  - Prevent the gradient vanishing by using weak distance metrics
  - Provide the parsimonious training indicator.

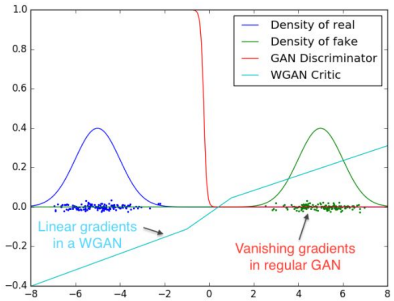
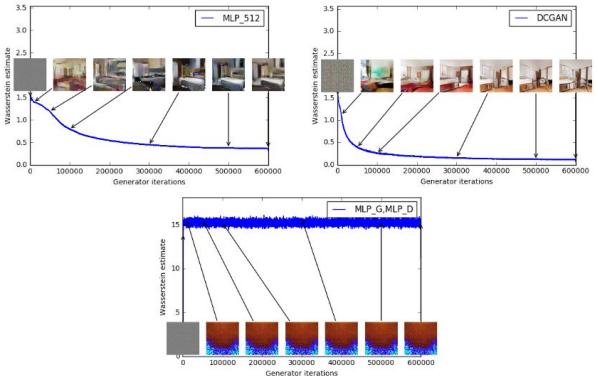
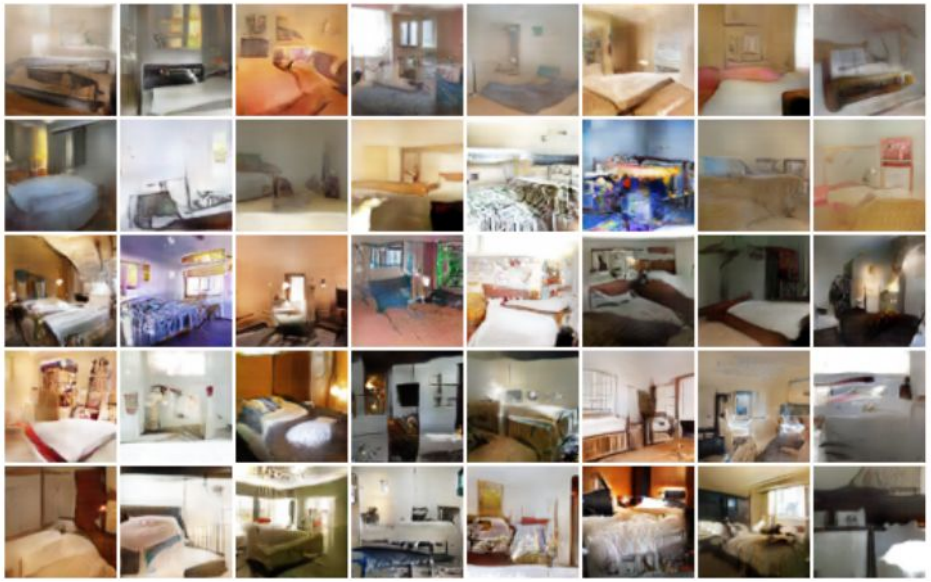


Figure 2: Optimal discriminator and critic when learning to differentiate two Gaussians. As we can see, the traditional GAN discriminator saturates and results in vanishing gradients. Our WGAN critic provides very clean gradients on all parts of the space.



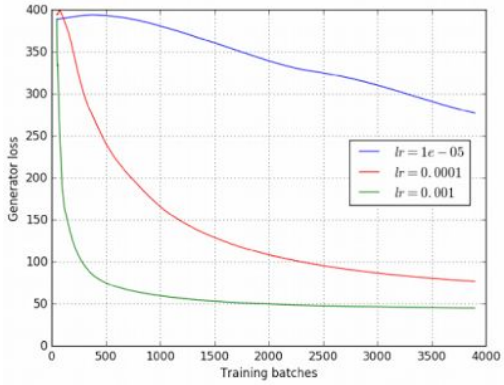
WGAN

- Results



BGAN

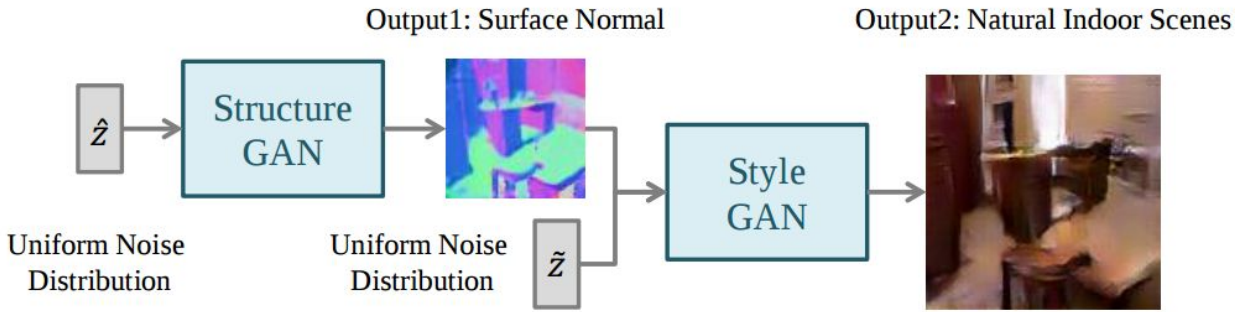
- R Devon et al, Boundary-Seeking GAN, 2017
  - Importance Sampling + Boundary Seeking
  - **Discrete** GAN training possible
  - Stabilized continuous GAN training



Application of GANs

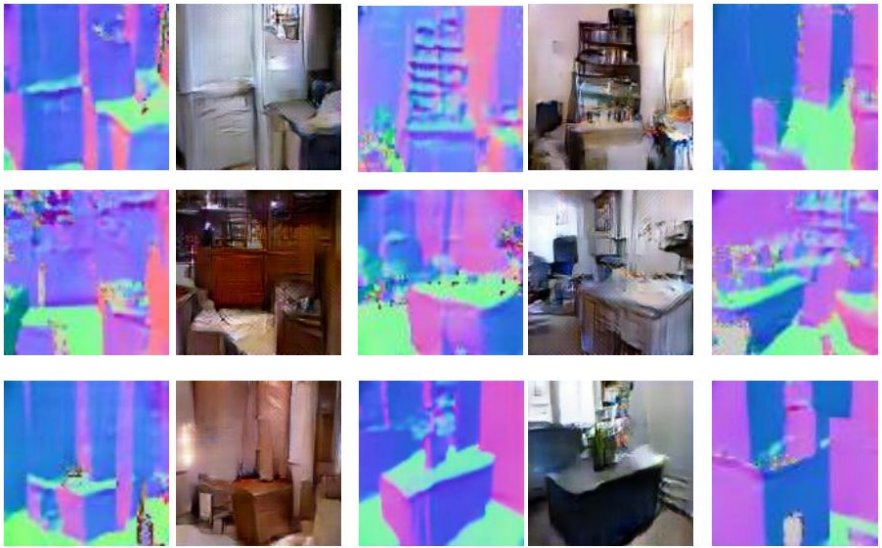
Generating image

- Wang et al, Generative Image Modeling using Style and Structure Adversarial Networks, 2016



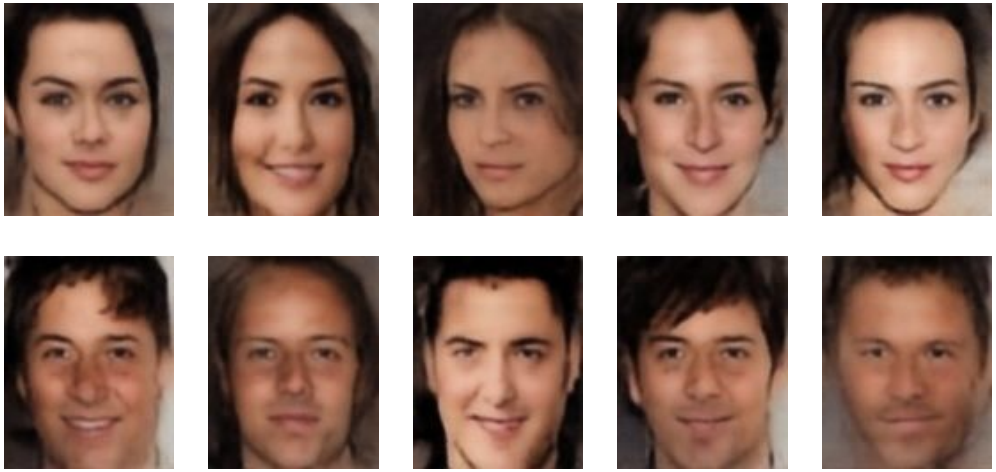
Generating image

- Wang et al : Results



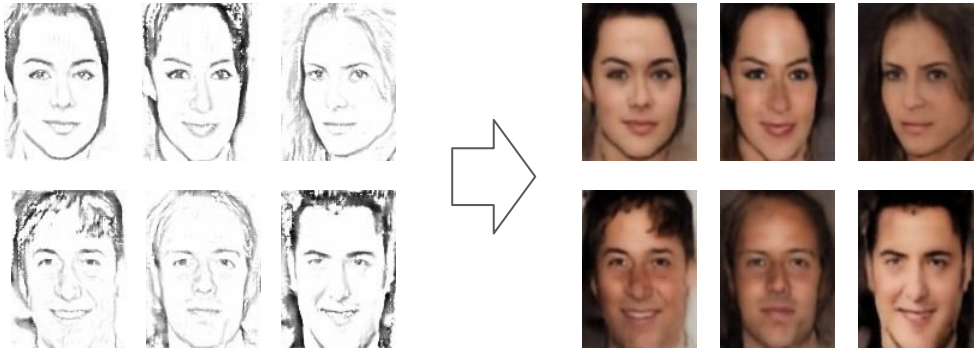
Generating image

- Jamonglab, 128x96 face image generation, 2016.11



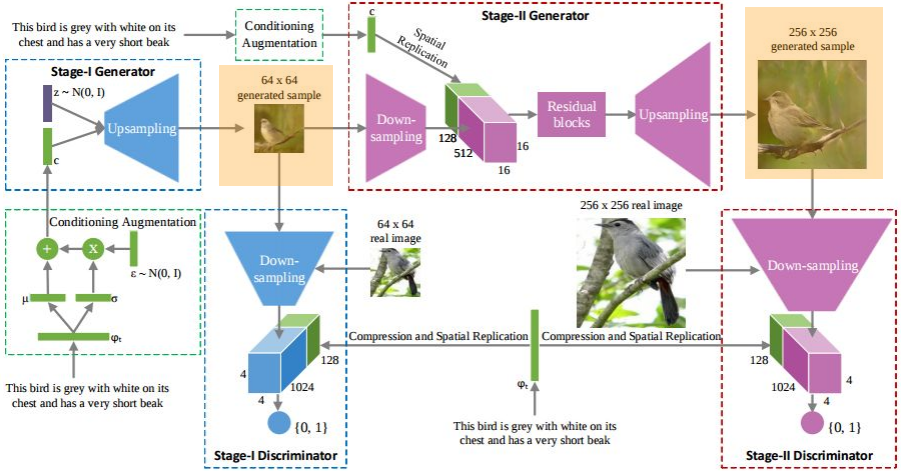
Generating image

- Two-step generation : Sketch → Color
- Binomial random seed instead of gaussian



Generating image

- Zhang et al, StackGAN : Text to Photo-realistic Image Synthesis with Stacked GANs, 2016

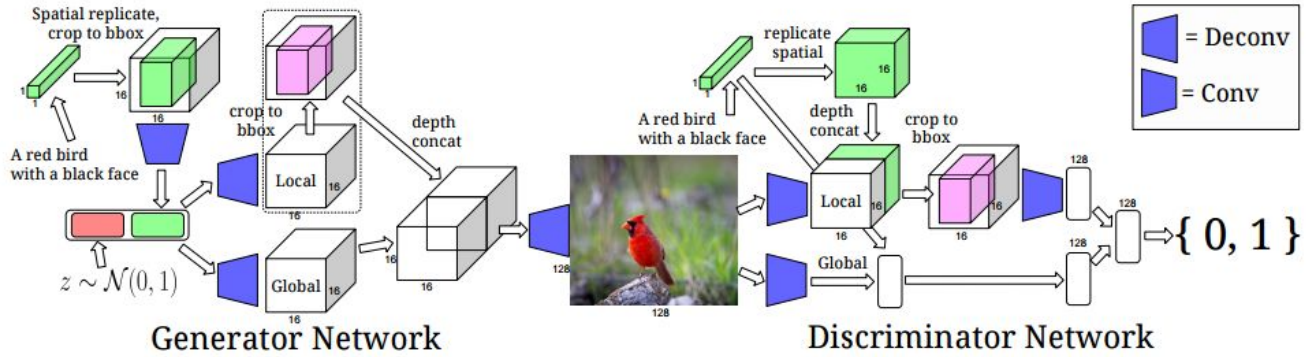


Generating image

- Zhang et al : Results

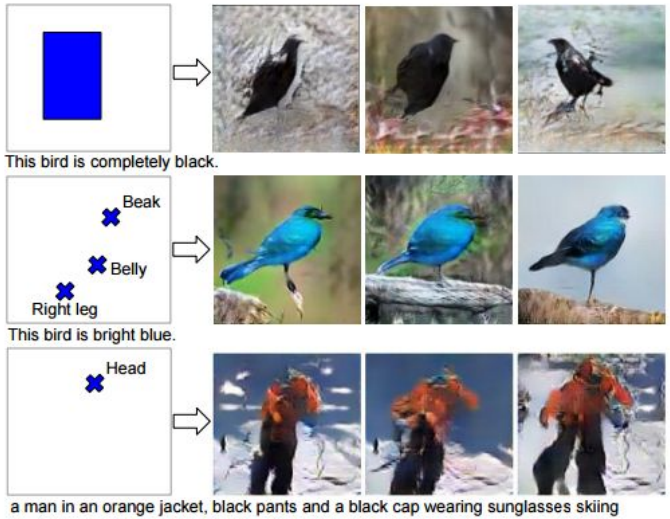
Generating image

- Reed et al, Learning What and Where to Draw, 2016



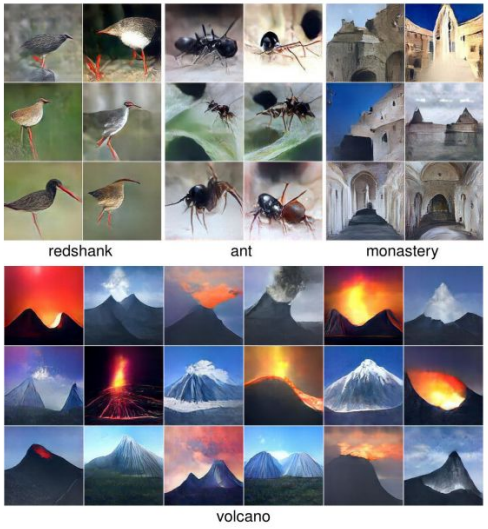
Generating image

- Reed et al : Results



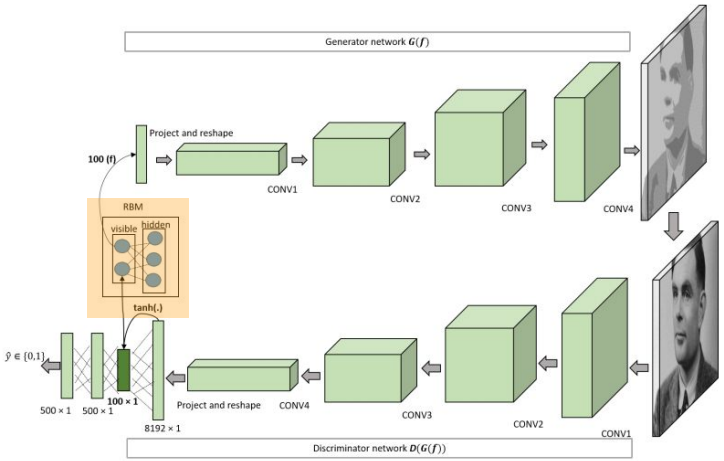
Generating image

- Nguyen et al, Plug & Play Generative Networks, 2016
  - Complex langevin **sampling** via DAE



Generating image

- Arici et al, Associative Adversarial Networks, 2016
  - $p(z|x) \neq \mathcal{N}(0, \mathbf{I})$

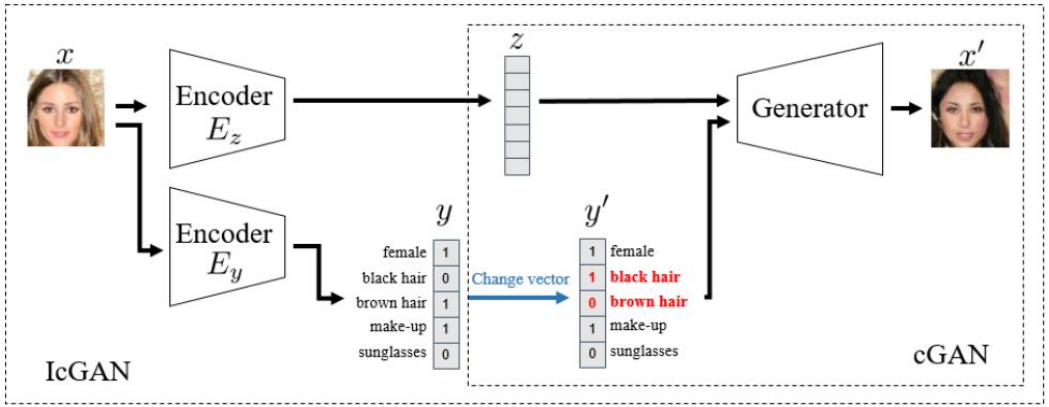


Generating image

- Arici et al : Results

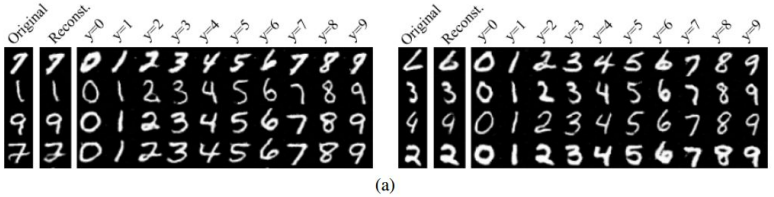
Translating image

- Perarnau et al, Invertible Conditional GANs for image editing, 2016



Translating image

- Perarnau et al : results

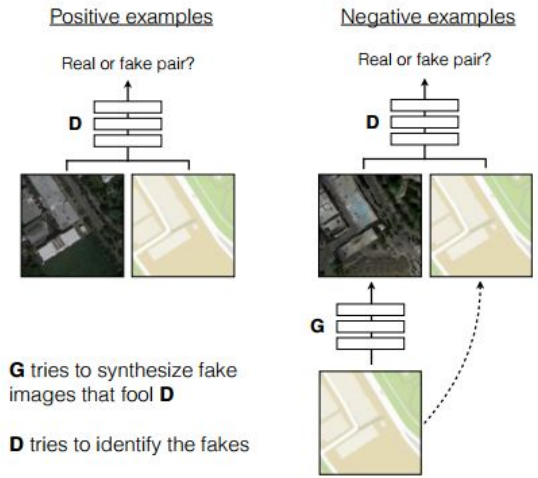


(a)

(b)

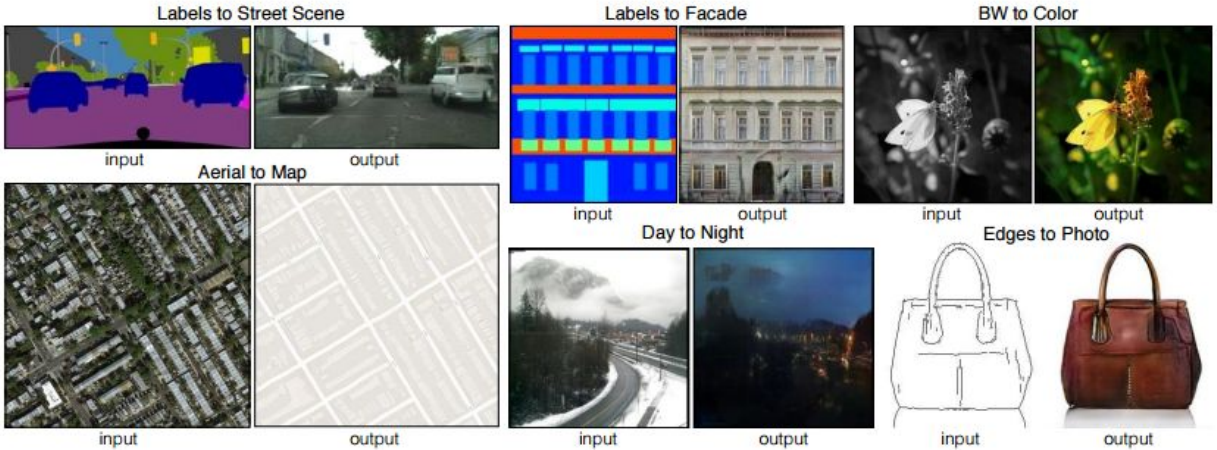
Translating image

- Isola et al, Image-to-image translation with convolutional adversarial networks, 2016



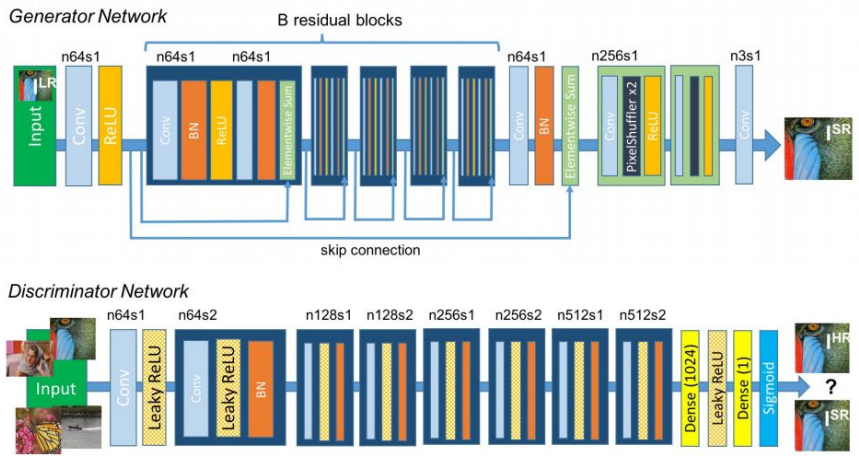
Translating image

- Isola et al : Results



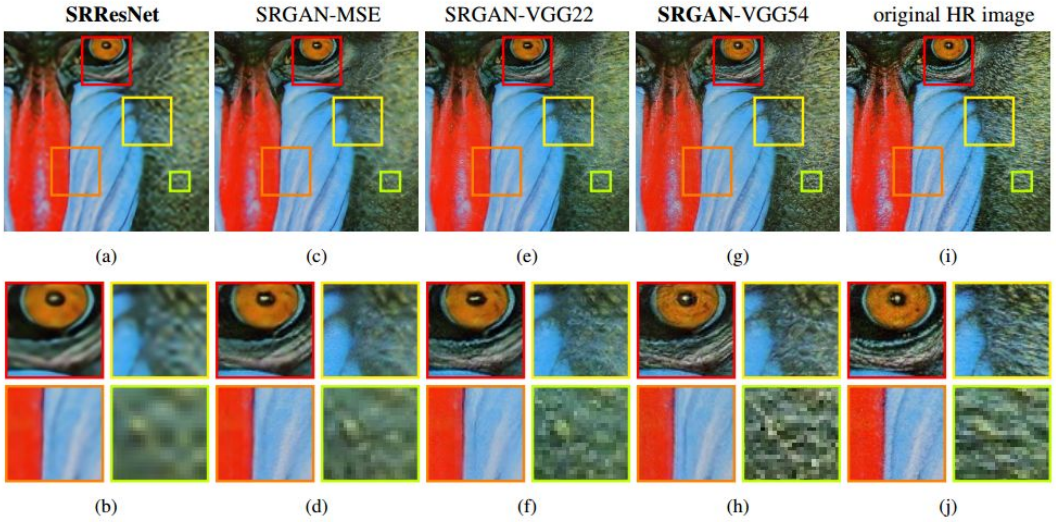
Translating image

- Ledig et al, Photo-Realistic Single Image Super-Resolution Using a GAN, 2016



Translating image

- Ledig et al : Results



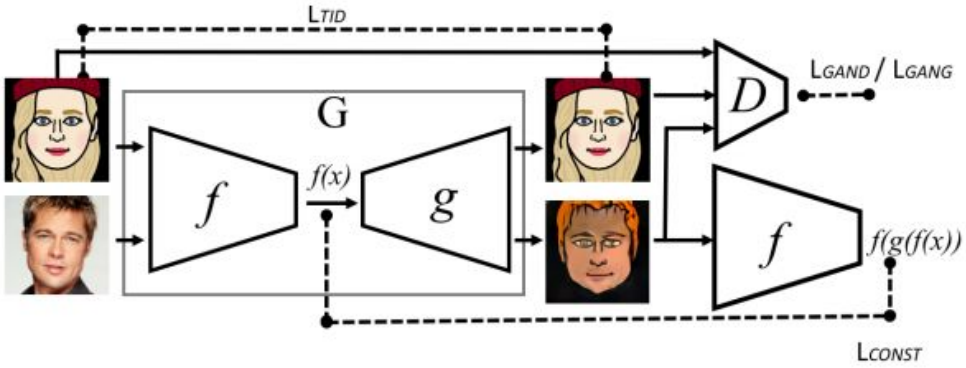
Translating image

- Sajjadi et al, EnhanceNet : Single Image Super-Resolution through Automated Texture Synthesis, 2016
  - SRGAN + Gram metrics loss

State of the art by PSNR | Our result

Domain adaptation

- Taigman et al, Unsupervised cross-domain image generation, 2016

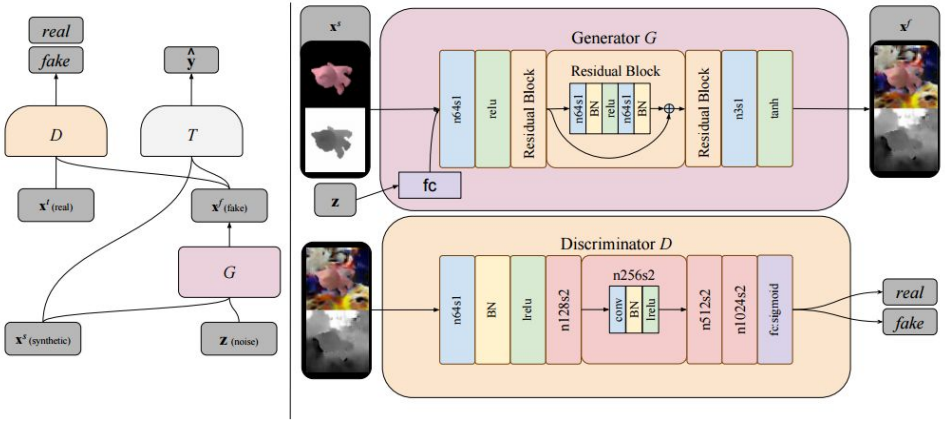


Domain adaptation

- Taigman et al, Unsupervised cross-domain image generation, 2016

Domain adaptation

- Bousmalis et al, Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks, 2016



Domain adaptation

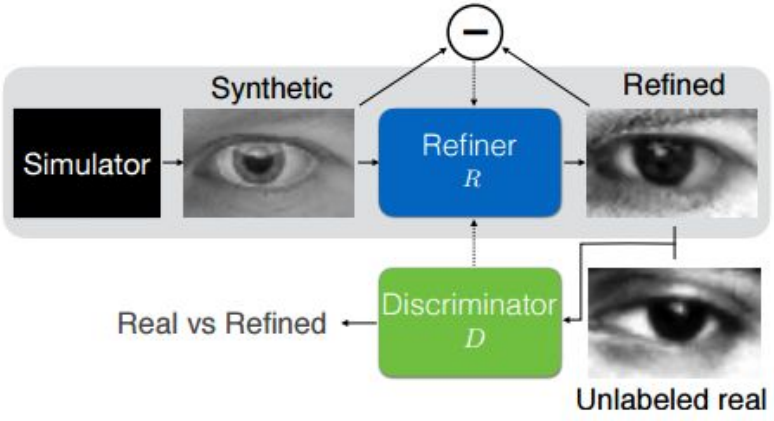
- Bousmalis et al : Results

(a) Image examples from the Linemod dataset.

(b) Examples generated by our model, trained on Linemod.

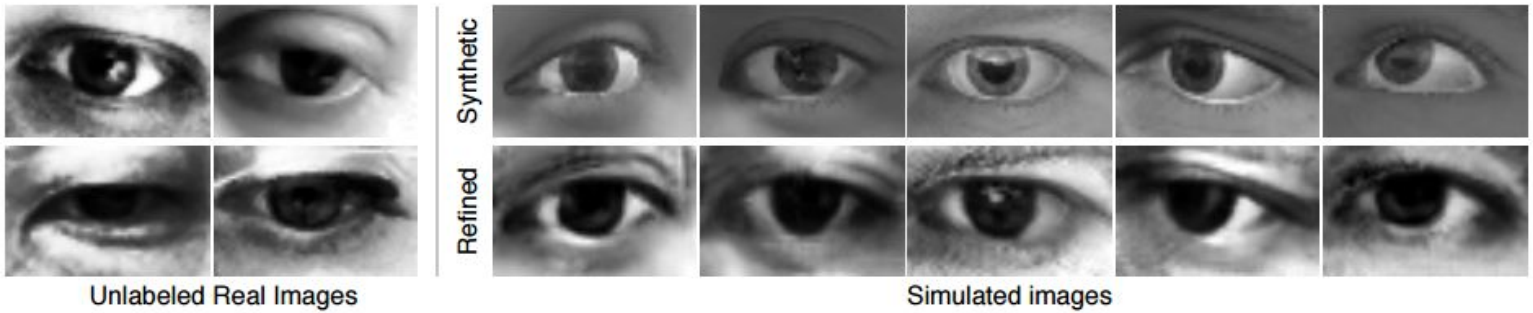
Domain adaptation

- Shrivastava et al, Learning from simulated and unsupervised images through adversarial training, 2016



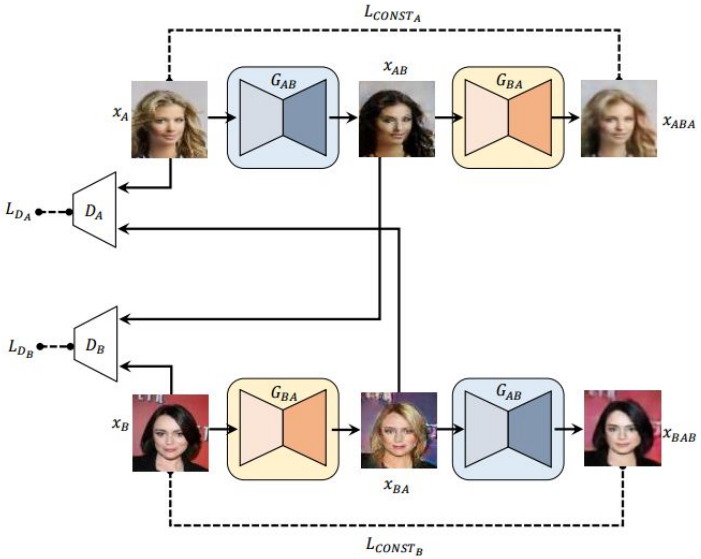
Domain adaptation

- Shrivastava et al : Results



Domain adaptation

- Taeksoo Kim et al, Learning to Discover Cross-Domain Relations with Generative Adversarial Networks, 2017

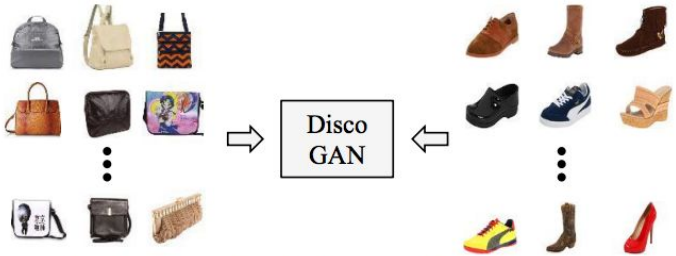


Domain adaptation

- Extended Taigman's idea

  using smart bijective mapping

(b) Handbag images (input) & **Generated** shoe images (output)

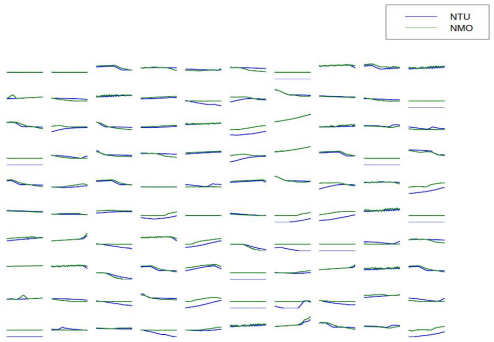


(a) Learning cross-domain relations **without any extra label**

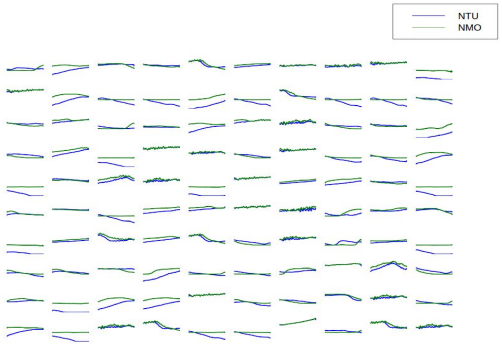
(c) Shoe images (input) & **Generated** handbag images (output)

Unsupervised clustering

- Jamonglab : https://github.com/buriburisuri/timeseries_gan
  - Using InfoGAN

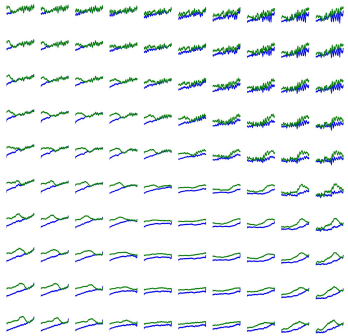


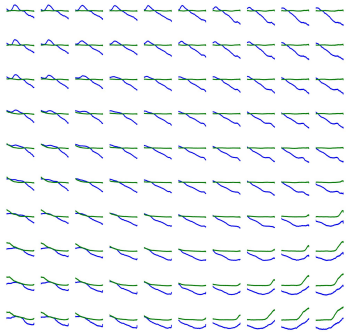
Real

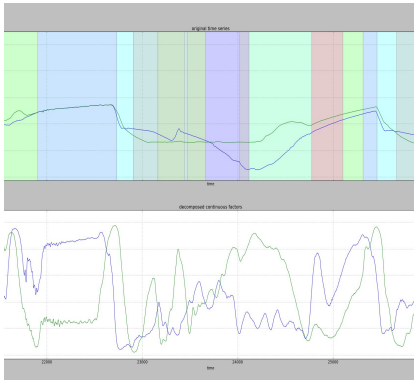


Fake

Unsupervised clustering

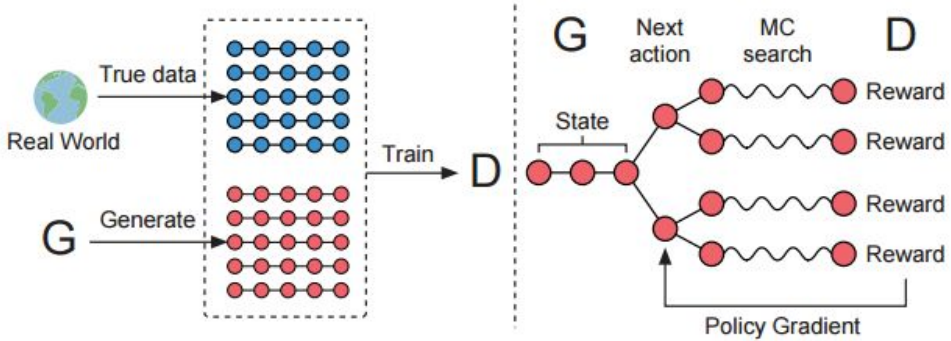






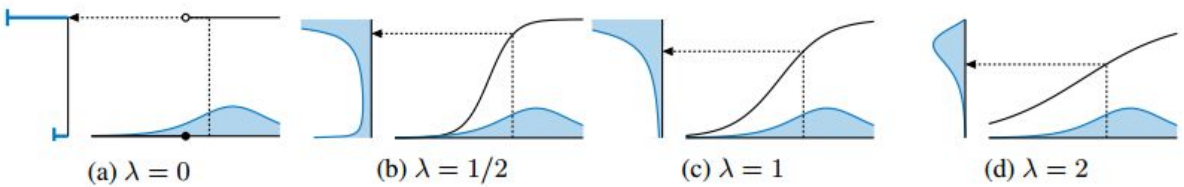
Text generation

- Yu et al, SeqGAN : Sequence Generative Adversarial Nets with Policy Gradient, 2016



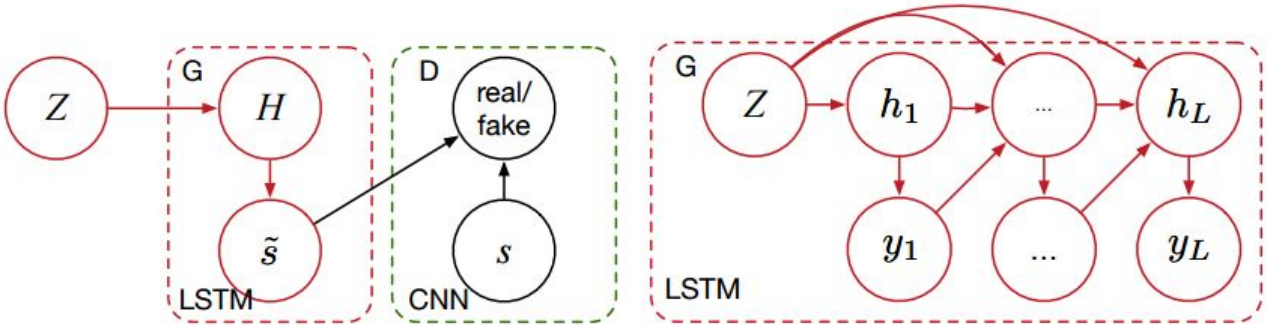
Text generation

- Maddison et al, The concrete distribution : A continous relaxation of discrete random variables, 2016
- Kusner et al, GANs for sequences of discrete elements with the Gumbel-softmax distribution, 2016



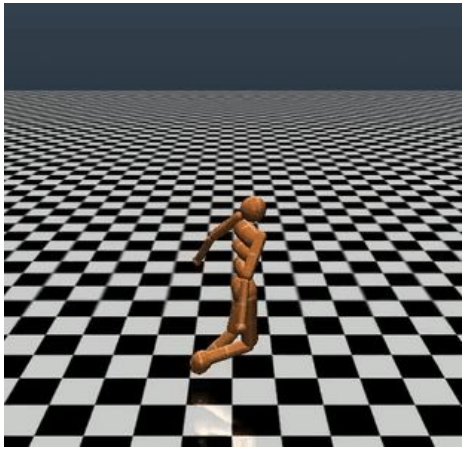
Text generation

- Zhang et al, Generating text via Adversarial Training, 2016



Imitation learning (IRL or Optimal control)

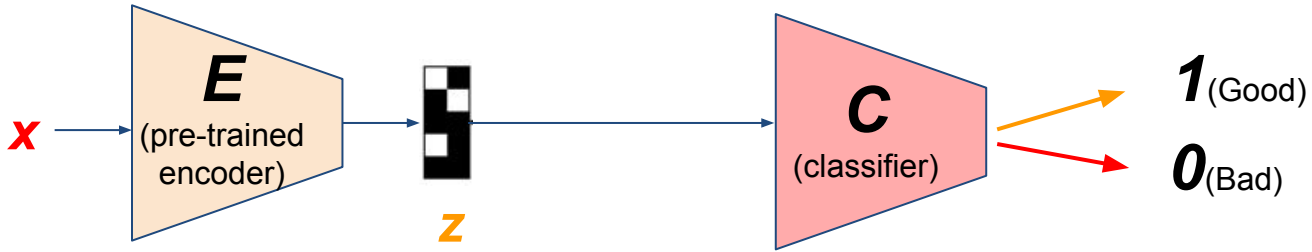
- Ho et al, Model-free imitation learning with policy optimization, 2016
- Finn et al, A connection between generative adversarial networks, Inverse Reinforcement Learning, and Energy-based models, 2016



Future of GANs

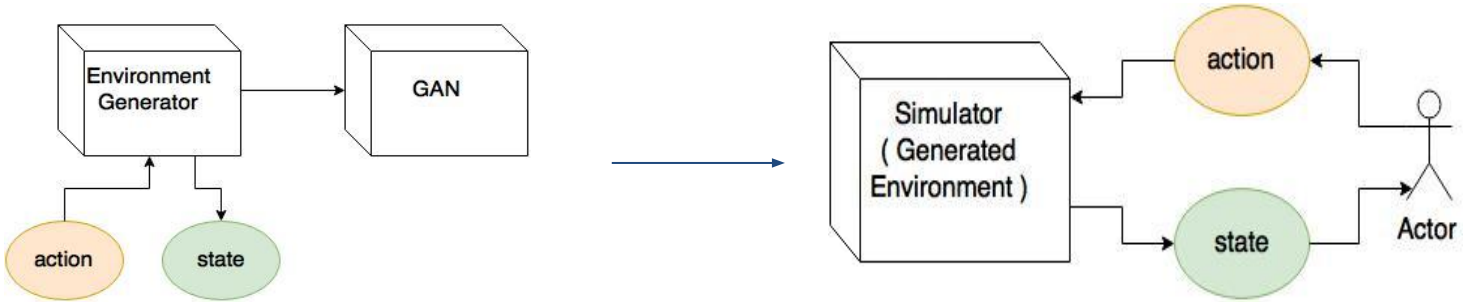
Pre-trained unified features

- Unified features by unsupervised learning
- Pre-trained and shipped on mobile chip as default



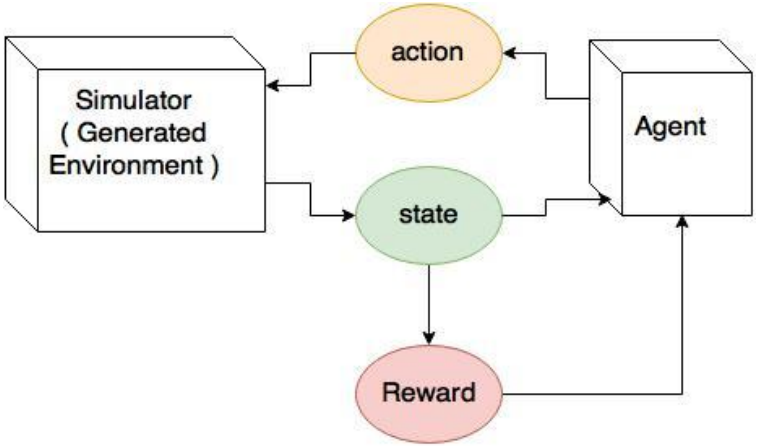
Personalization at scale

- Environment (User) simulator



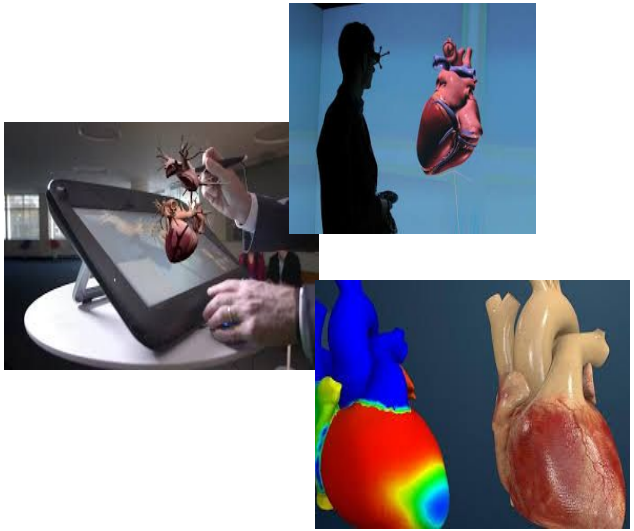
Personalization at scale

- Intelligent agent using RL



Personalization at scale

- For example, cardiac simulator using GAN



Conclusion

- Unsupervised learning is next frontier in AI
  - Same algos, multiple domains
- Creativity and experience are no longer human temperament

Towards Artificial General Intelligence

Thank you.

http://www.slideshare.net/ssuser77ee21/generative-adversarial-networks-70896091