Unsupervised learning and Generative models

Charles Ollion - Olivier Grisel

Outline

Unsupervised learning

Outline

Unsupervised learning

Autoencoders

Outline

Unsupervised learning

Autoencoders

Generative Adversarial Networks

Unsupervised learning

Unsupervised learning

Generic goal of unsupervised learning is to **find underlying structrure** in data. Specific goals include:

- clustering: group similar observations together;
- reducing the dimensionality for visualization;
- building a better representation of data for a downstream supervised task;
- learning a likelihood function, e.g. to detect anomalies;

Unsupervised learning

For complex data (text, image, sound, ...), there is plenty of hidden latent structure we hope to capture:

- **Image data**: find low dimensional semantic representations, independent sources of variation;
- **Text data**: find fixed size, dense semantic representation of data.

Unsupervised learning

For complex data (text, image, sound, ...), there is plenty of hidden latent structure we hope to capture:

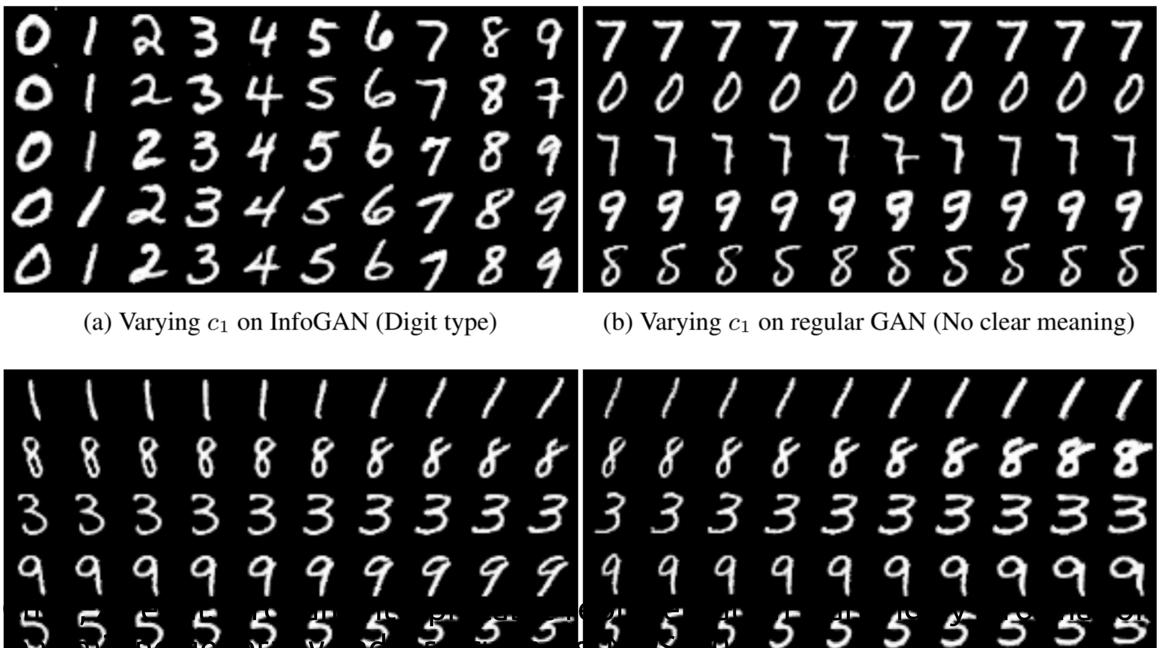
- **Image data**: find low dimensional semantic representations, independent sources of variation;
- **Text data**: find fixed size, dense semantic representation of data.

Latent space might be used to help build more efficient human labeling interfaces.

=> Goal: reduce labeling cost via active learning.

Graal of unsupervised learning

A low dimension space which captures all the **variations** of data and **disentangles** the different latent factors underlying the data.



(a) Varying c_1 on InfoGAN (Digit type)

(b) Varying c_1 on regular GAN (No clear meaning)

(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation)

(d) Varying c_3 from -2 to 2 on InfoGAN (Width)

maximizing generative adversarial nets. NIPS, 2016.

Self-supervised learning

find smart ways to **build supervision** without labels, exploiting domain knowledge and regularities

Self-supervised learning

find smart ways to **build supervision** without labels, exploiting domain knowledge and regularities

Use **text structure** to create supervision

- Word2Vec, BERT or GPT-2 language models

Self-supervised learning

find smart ways to **build supervision** without labels, exploiting domain knowledge and regularities

Use **text structure** to create supervision

- Word2Vec, BERT or GPT-2 language models

Can we do the same for other domains?

- **Image:** exploit spatial context of an object
- **Sound, video:** exploit temporal context

Self-supervised learning

find smart ways to **build supervision** without labels, exploiting domain knowledge and regularities

Use **text structure** to create supervision

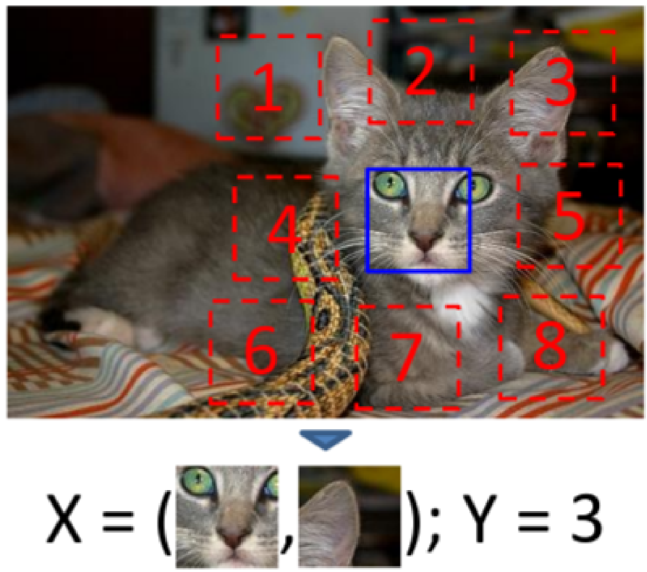
- Word2Vec, BERT or GPT-2 language models

Can we do the same for other domains?

- **Image:** exploit spatial context of an object
- **Sound, video:** exploit temporal context

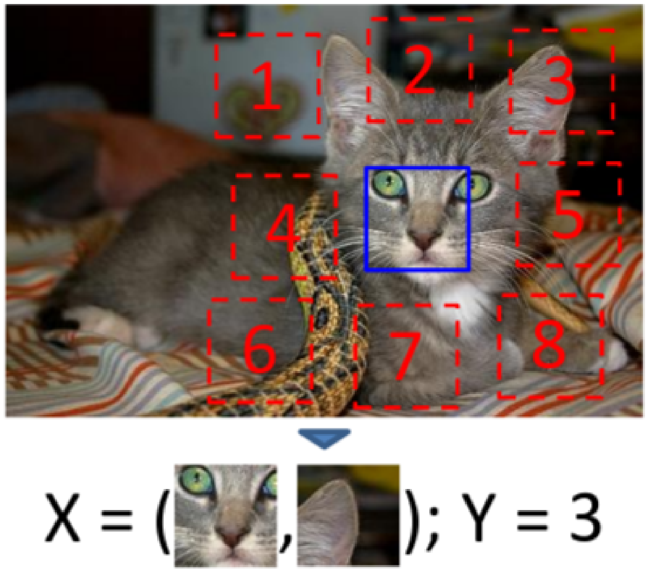
No direct **accuracy** measure: usually tested through a downstream task

Self-supervised learning



Doersch, Carl, Abhinav Gupta, and Alexei A. Efros. "Unsupervised visual representation learning by context prediction." ICCV 2015.

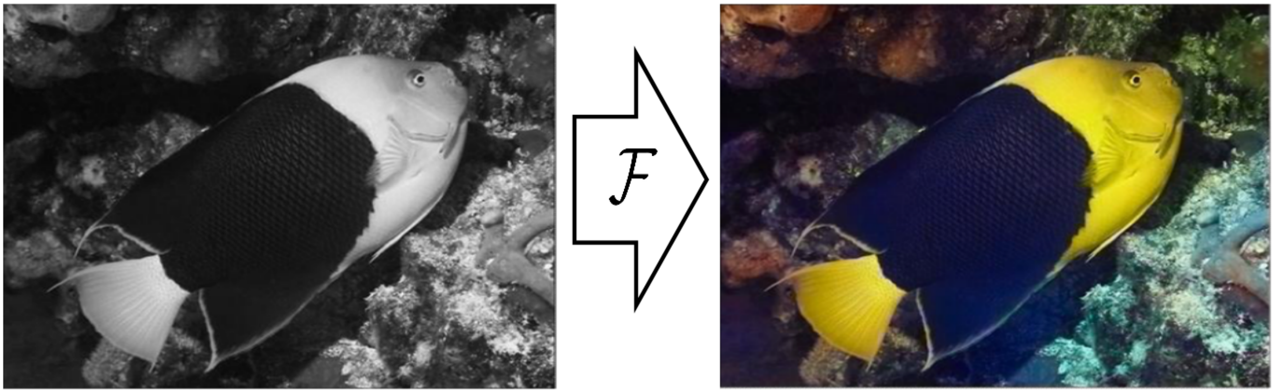
Self-supervised learning



- Predict patches arrangement in images: 8 class classifier
- Siamese architecture for the two patches + concat

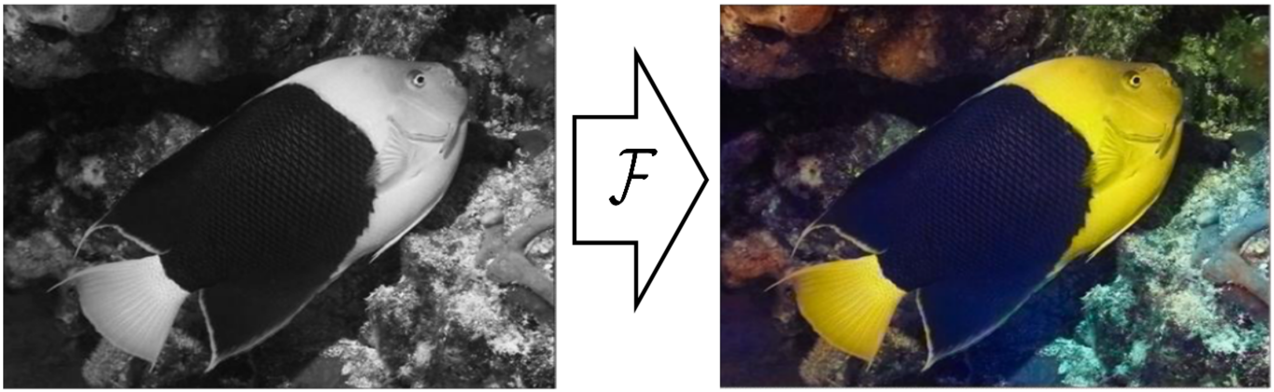
Doersch, Carl, Abhinav Gupta, and Alexei A. Efros. "Unsupervised visual representation learning by context prediction." ICCV 2015.

Self-supervised learning



Zhang et al. "Colorful Image Colorization" ECCV 2016

Self-supervised learning



- Given RGB images, generate their grayscale version
- Train a network to predict pixels color given grayscale image

Zhang et al. "Colorful Image Colorization" ECCV 2016

Self-supervised learning

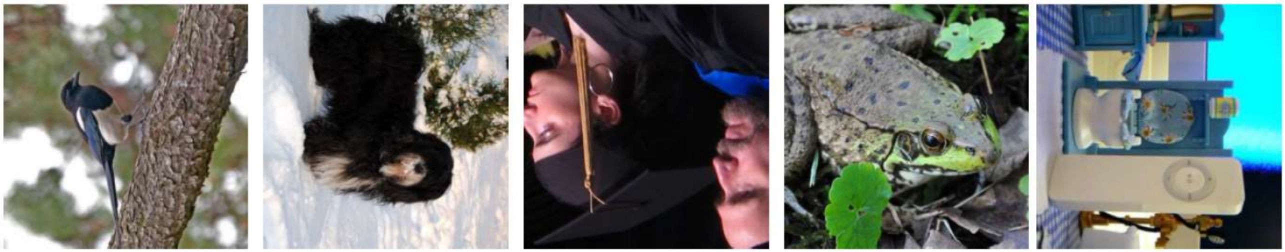
Dosovitskiy et al. "Exemplar Networks" 2014

Self-supervised learning

- Heavy augmentation of the images
- Network must predict that augmented images are similar, and another random image dissimilar

Dosovitskiy et al. "Exemplar Networks" 2014

Self-supervised

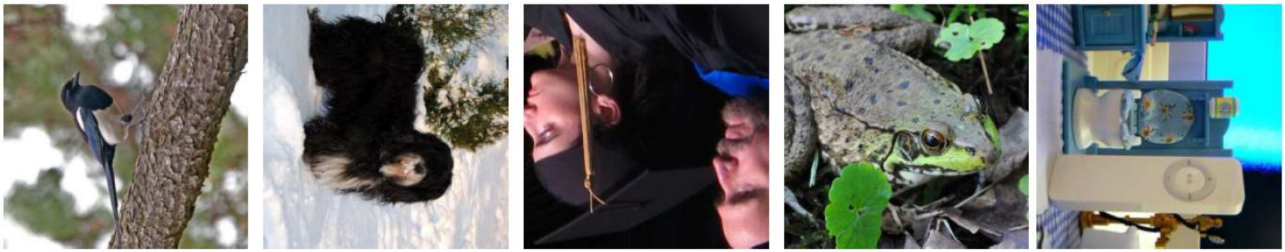


90° rotation | 270° rotation | 180° rotation | 0° rotation | 270° rotation

Figure 1: Images rotated by random multiples of 90 degrees (e.g., 0, 90, 180, or 270 degrees). The core intuition of our self-supervised feature learning approach is that if someone is not aware of the concepts of the objects depicted in the images, he cannot recognize the rotation that was applied to them.

Spyros Gidaris, Praveer Singh, Nikos Komodakis. "Unsupervised representation learning by predicting image rotations, " ICLR 2018

Self-supervised



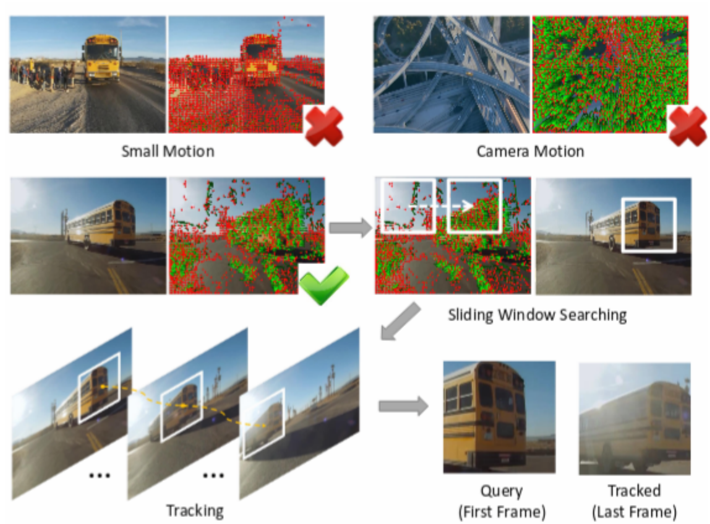
90° rotation | 270° rotation | 180° rotation | 0° rotation | 270° rotation

Figure 1: Images rotated by random multiples of 90 degrees (e.g., 0, 90, 180, or 270 degrees). The core intuition of our self-supervised feature learning approach is that if someone is not aware of the concepts of the objects depicted in the images, he cannot recognize the rotation that was applied to them.

- Generate 4 versions of the image, rotated by 0˚, 90˚, 180˚, and 270˚
- Network must predict the angle

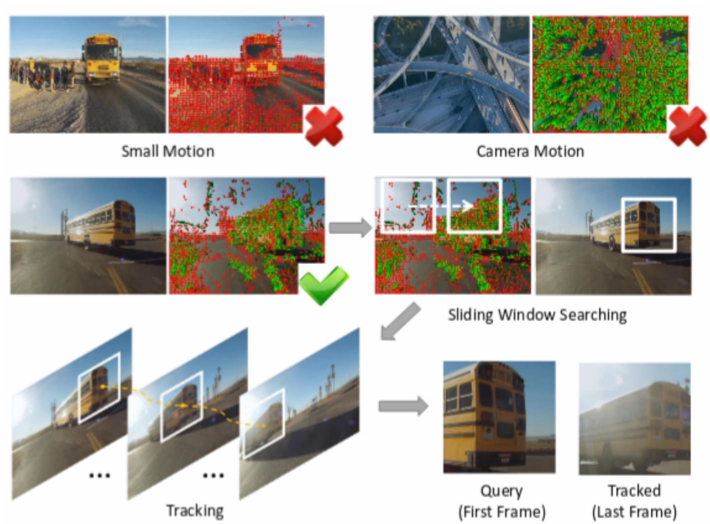
Spyros Gidaris, Praveer Singh, Nikos Komodakis. "Unsupervised representation learning by predicting image rotations, " ICLR 2018

Self-supervised from video



Wang, Xiaolong, and Abhinav Gupta. "Unsupervised learning of visual representations using videos." ICCV 2015.

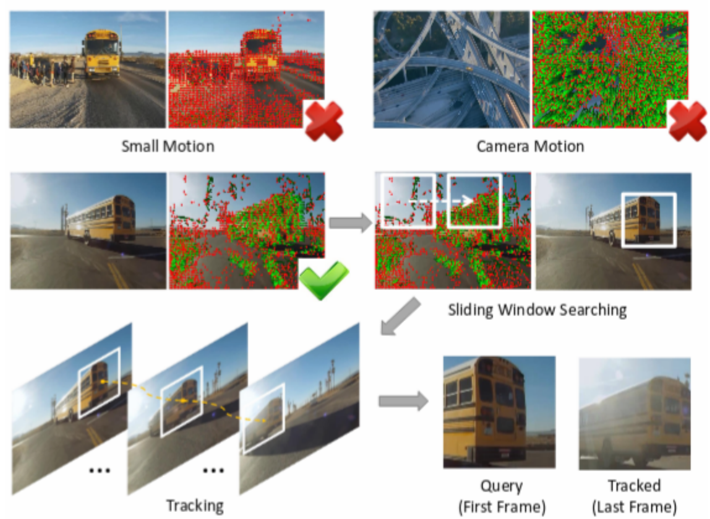
Self-supervised from video



- Collect pairs of similar objects from videos

Wang, Xiaolong, and Abhinav Gupta. "Unsupervised learning of visual representations using videos." ICCV 2015.

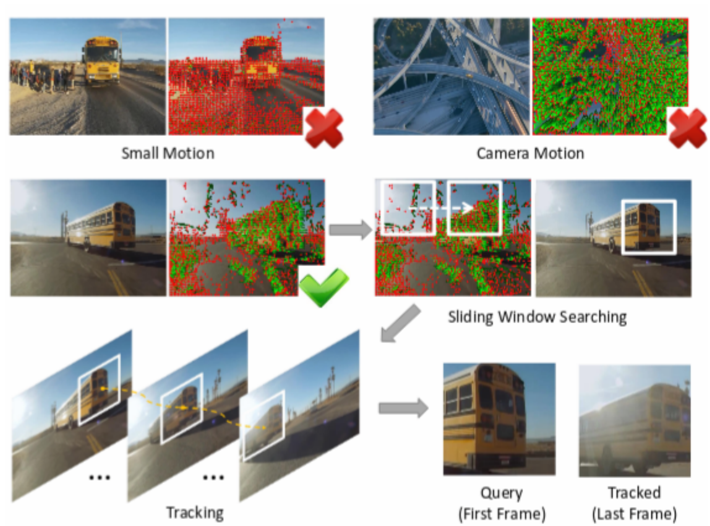
Self-supervised from video



- Collect pairs of similar objects from videos
- Train a siamese net with positive pairs = similar objects detected

Wang, Xiaolong, and Abhinav Gupta. "Unsupervised learning of visual representations using videos." ICCV 2015.

Self-supervised from video

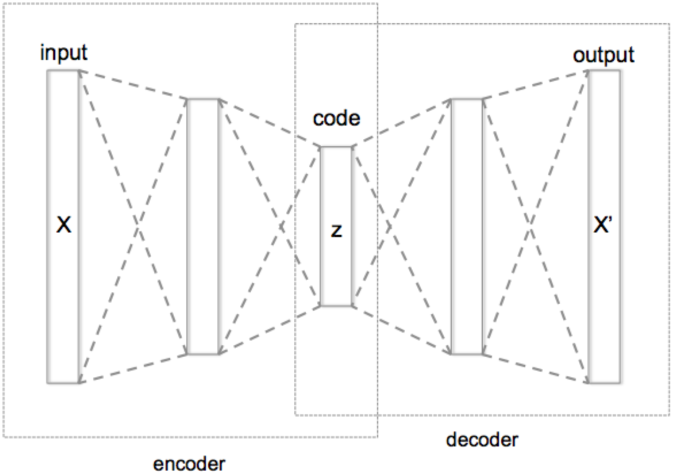


- Collect pairs of similar objects from videos
- Train a siamese net with positive pairs = similar objects detected
- Hard pairs mining: find objects with large movement

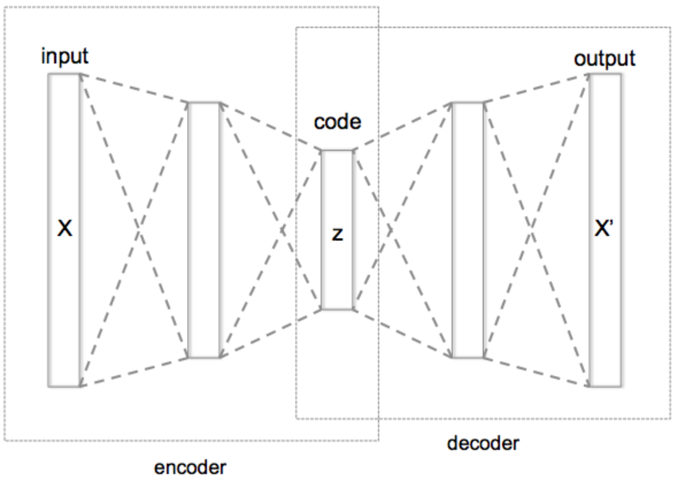
Wang, Xiaolong, and Abhinav Gupta. "Unsupervised learning of visual representations using videos." ICCV 2015.

Autoencoders

Autoencoder



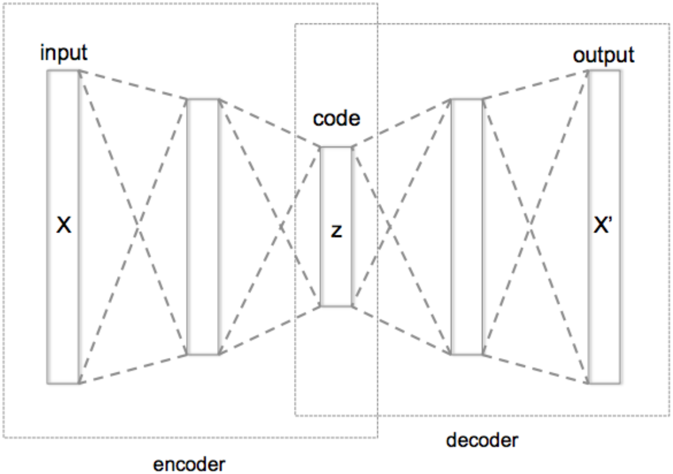
Autoencoder



Supervision : reconstruction loss of the input, usually:

$$l(x, f(x)) = ||f(x) - x||_2^2$$

Autoencoder

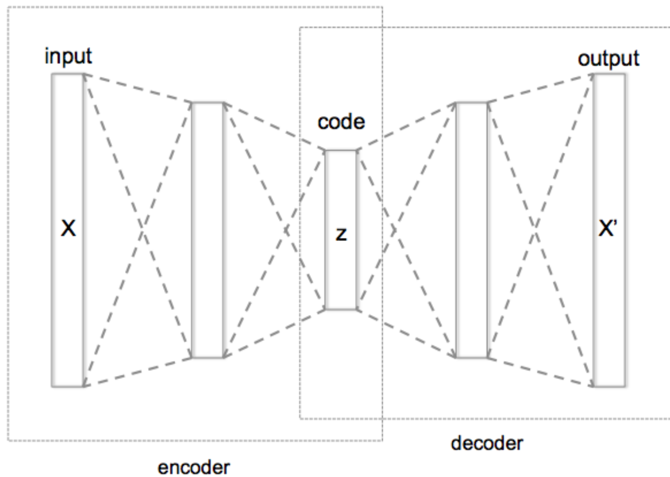


Supervision : reconstruction loss of the input, usually:

$$l(x, f(x)) = ||f(x) - x||_2^2$$

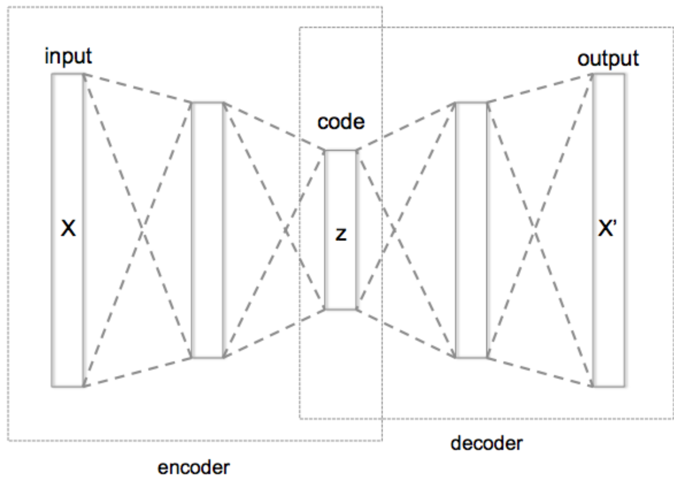
Binary crossentropy is also used

Autoencoder



Keeping the **latent code** $\mathbf{z}$ low-dimensional forces the network to learn a "smart" compression of the data, not just an identity function

Autoencoder



Keeping the **latent code $\mathbf{z}$** low-dimensional forces the network to learn a "smart" compression of the data, not just an identity function

Sparse/Denoising Autoencoder

Adding a sparsity constraint on activations:

$$||encoder(x)||_1 \sim \rho, \rho = 0.05$$

Learns sparse features, easily interpretable

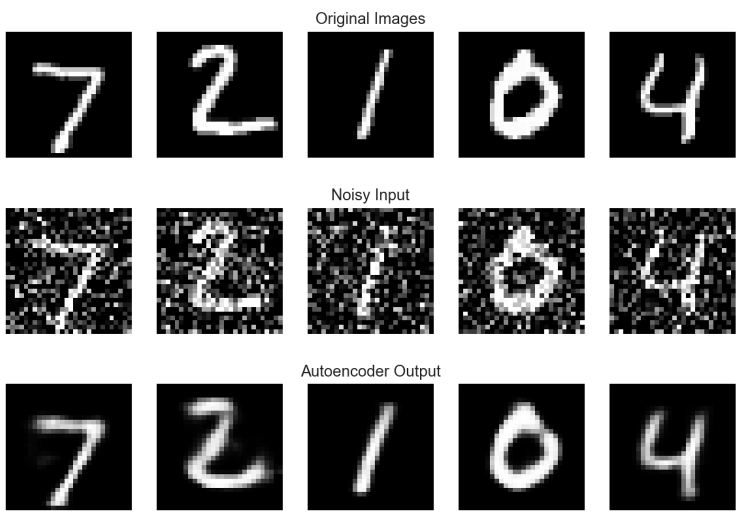
Sparse/Denoising Autoencoder

Adding a sparsity constraint on activations:

$$||encoder(x)||_1 \sim \rho, \rho = 0.05$$

Learns sparse features, easily interpretable

Denoising Autoencoder: train features for robustness to noise.



Uses and limitations

After **pre-training** use the latent code $\mathbf{z}$ as input to a classifier instead of $\mathbf{x}$

Semi-supervised learning simultaneous learning of the latent code (on a large, unlabeled dataset) and the classifier (on a smaller, labeled dataset)

Uses and limitations

After **pre-training** use the latent code $\mathbf{z}$ as input to a classifier instead of $\mathbf{x}$

Semi-supervised learning simultaneous learning of the latent code (on a large, unlabeled dataset) and the classifier (on a smaller, labeled dataset)

Other use: Use decoder $D(x)$ as a **Generative model**: generate samples from random noise

Uses and limitations

After **pre-training** use the latent code $\mathbf{z}$ as input to a classifier instead of $\mathbf{x}$

Semi-supervised learning simultaneous learning of the latent code (on a large, unlabeled dataset) and the classifier (on a smaller, labeled dataset)

Other use: Use decoder $D(x)$ as a **Generative model**: generate samples from random noise

Limitations :

- Direct autoencoder fails to capture good representations for complex data such as images
- The generative model is usually of very poor quality (very blurry for images for instance)

Reality Check

For image features, **ImageNet pretraining** is still **much better** than unsupervised models

Pretraining Method	Supervision	Pretraining time	Classification	Detection	Segmentation
ImageNet [26]	1000 class labels	3 days	**78.2%**	**56.8%**	**48.0%**
Random Gaussian	initialization	< 1 minute	53.3%	43.4%	19.8%
Autoencoder	-	14 hours	53.8%	41.9%	25.2%
Agrawal *et al*. [1]	egomotion	10 hours	52.9%	41.8%	-
Doersh *et al*. [7]	context	4 weeks	55.3%	**46.6%**	-
Wang *et al*. [39]	motion	1 week	**58.4%**	44.0%	-
Ours	context	14 hours	56.5%	44.5%	**29.7%**

- Results shown after fine-tuned the network on **Pascal VOC dataset**
- The "ours" method is feature representation based on context inpainting

Pathak, Deepak, et al. Context encoders: Feature learning by inpainting. CVPR 2016.

Variational Autoencoders

Variational Autoencoders (VAE)

Assume the data samples $\mathbf{x}^{(i)}$ are generated by the model:

$$p_{\theta^*}(\mathbf{x}, \mathbf{z}) = p_{\theta^*}(\mathbf{z}) \cdot p_{\theta^*}(\mathbf{x}|\mathbf{z})$$

Variational Autoencoders (VAE)

Assume the data samples $\mathbf{x}^{(i)}$ are generated by the model:

$$p_{\theta^*}(\mathbf{x}, \mathbf{z}) = p_{\theta^*}(\mathbf{z}) \cdot p_{\theta^*}(\mathbf{x}|\mathbf{z})$$

- $\mathbf{x}$ is an observed r.v. with values in $\mathbb{R}^n$;
- $\mathbf{z}$ is a latent r.v. with values in $\mathbb{R}^d$;
- True continuous parameters θ^* are unknown;
- Estimate parameters θ from data $\mathbf{x}^{(i)}$ by maximizing the marginal likelihood (MLE):

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{z}) \cdot p_\theta(\mathbf{x}|\mathbf{z}) \, \mathrm{d}\mathbf{z}$$

Variational Autoencoders (VAE)

Assume the data samples $\mathbf{x}^{(i)}$ are generated by the model:

$$p_{\theta^*}(\mathbf{x}, \mathbf{z}) = p_{\theta^*}(\mathbf{z}) \cdot p_{\theta^*}(\mathbf{x}|\mathbf{z})$$

- $\mathbf{x}$ is an observed r.v. with values in $\mathbb{R}^n$;
- $\mathbf{z}$ is a latent r.v. with values in $\mathbb{R}^d$;
- True continuous parameters θ^* are unknown;
- Estimate parameters θ from data $\mathbf{x}^{(i)}$ by maximizing the marginal likelihood (MLE):

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{z}) \cdot p_\theta(\mathbf{x}|\mathbf{z}) \, \mathrm{d}\mathbf{z}$$

But this high dimensional integral cannot be estimated efficently.

Variational Autoencoders (VAE)

Assume the data samples $\mathbf{x}^{(i)}$ are generated by the model:

$$p_{\theta^*}(\mathbf{x}, \mathbf{z}) = p_{\theta^*}(\mathbf{z}) \cdot p_{\theta^*}(\mathbf{x}|\mathbf{z})$$

- $\mathbf{x}$ is an observed r.v. with values in $\mathbb{R}^n$;
- $\mathbf{z}$ is a latent r.v. with values in $\mathbb{R}^d$;
- True continuous parameters θ^* are unknown;
- Estimate parameters θ from data $\mathbf{x}^{(i)}$ by maximizing the marginal likelihood (MLE):

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{z}) \cdot p_\theta(\mathbf{x}|\mathbf{z})\, \mathrm{d}\mathbf{z}$$

But this high dimensional integral cannot be estimated efficently.

Auto-Encoding Variational Bayes, Diederik P Kingma, Max Welling

Variational Autoencoders (VAE)

Introduce approximate (variational) posterior on the latent variable: $q_\phi(\mathbf{z}|\mathbf{x})$ with continuous parameters ϕ.

Variational Autoencoders (VAE)

Introduce approximate (variational) posterior on the latent variable: $q_\phi(\mathbf{z}|\mathbf{x})$ with continuous parameters ϕ.

We can rewrite the log-likelihood as:

$$log\, p_\theta(\mathbf{x}^{(i)}) = \mathrm{D}_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\theta, \phi; \mathbf{x}^{(i)})$$

with:

$$\mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) = -\mathrm{D}_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})} log\, p_\theta(\mathbf{x}^{(i)}|\mathbf{z})$$

Variational Autoencoders (VAE)

Introduce approximate (variational) posterior on the latent variable: $q_\phi(\mathbf{z}|\mathbf{x})$ with continuous parameters ϕ.

We can rewrite the log-likelihood as:

$$log\, p_\theta(\mathbf{x}^{(i)}) = \mathrm{D}_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\theta, \phi; \mathbf{x}^{(i)})$$

with:

$$\mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) = -\mathrm{D}_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})} log\, p_\theta(\mathbf{x}^{(i)}|\mathbf{z})$$

- $\mathrm{D}_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z}|\mathbf{x}^{(i)}))$ is positive but unknown. It depends on the quality of our approximation.

Variational Autoencoders (VAE)

Introduce approximate (variational) posterior on the latent variable: $q_\phi(\mathbf{z}|\mathbf{x})$ with continuous parameters ϕ.

We can rewrite the log-likelihood as:

$$log\, p_\theta(\mathbf{x}^{(i)}) = \mathrm{D}_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\theta, \phi; \mathbf{x}^{(i)})$$

with:

$$\mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) = -\mathrm{D}_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})} log\, p_\theta(\mathbf{x}^{(i)}|\mathbf{z})$$

- $\mathrm{D}_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z}|\mathbf{x}^{(i)}))$ is positive but unknown. It depends on the quality of our approximation.
- $\mathcal{L}(\theta, \phi; \mathbf{x}^{(i)})$ is a lower bound to maximize w.r.t. θ and ϕ.

Variational Autoencoders (VAE)

- Assume prior distribution for latent variable $\mathbf{z}$:

$$p_\theta(\mathbf{z}) = \mathcal{N}(0, 1)$$

Variational Autoencoders (VAE)

- Assume prior distribution for latent variable $\mathbf{z}$:

$$p_\theta(\mathbf{z}) = \mathcal{N}(0, 1)$$

- Parametrize $p_\theta(\mathbf{x}|\mathbf{z})$ by a neural network f_θ (decoder):

$$p_\theta(\mathbf{x}^{(i)}|\mathbf{z}) = \mathcal{N}(f_\theta(\mathbf{z}), 1)$$

Variational Autoencoders (VAE)

- Assume prior distribution for latent variable $\mathbf{z}$:

$$p_\theta(\mathbf{z}) = \mathcal{N}(0, 1)$$

- Parametrize $p_\theta(\mathbf{x}|\mathbf{z})$ by a neural network f_θ (decoder):

$$p_\theta(\mathbf{x}^{(i)}|\mathbf{z}) = \mathcal{N}(f_\theta(\mathbf{z}), 1)$$

- Parametrize $q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$ by a neural network with two heads μ_ϕ and σ_ϕ (encoder):

$$q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) = \mathcal{N}(\mu_\phi(\mathbf{x}^{(i)}), \sigma_\phi(\mathbf{x}^{(i)}))$$

Variational Autoencoders (VAE)

- Assume prior distribution for latent variable $\mathbf{z}$:

$$p_\theta(\mathbf{z}) = \mathcal{N}(0, 1)$$

- Parametrize $p_\theta(\mathbf{x}|\mathbf{z})$ by a neural network f_θ (decoder):

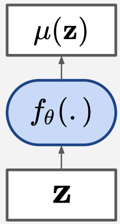
$$p_\theta(\mathbf{x}^{(i)}|\mathbf{z}) = \mathcal{N}(f_\theta(\mathbf{z}), 1)$$

- Parametrize $q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$ by a neural network with two heads μ_ϕ and σ_ϕ (encoder):

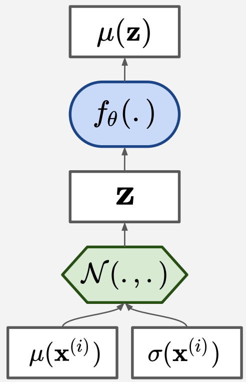
$$q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) = \mathcal{N}(\mu_\phi(\mathbf{x}^{(i)}), \sigma_\phi(\mathbf{x}^{(i)}))$$

- Reparametrization trick:

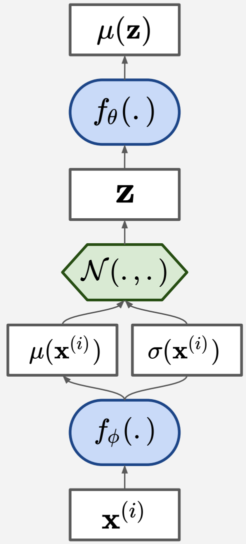
$$\mathbf{z} = \mu_\phi(\mathbf{x}^{(i)}) + \sigma_\phi(\mathbf{x}^{(i)}) \cdot \epsilon \quad \text{with} \quad \epsilon \sim \mathcal{N}(0, 1)$$



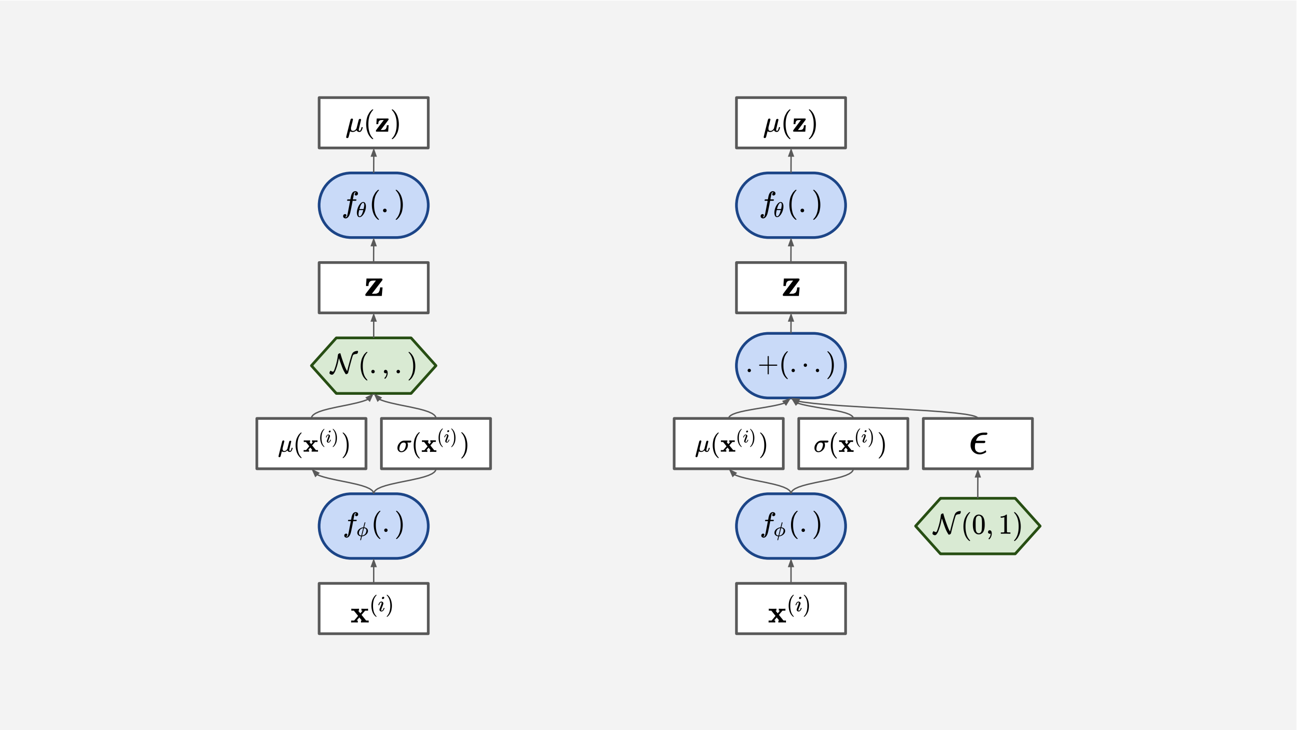
The **decoder** f_θ defines the likelihood of data.



The latent variable $\mathbf{z}$ is stochastic.

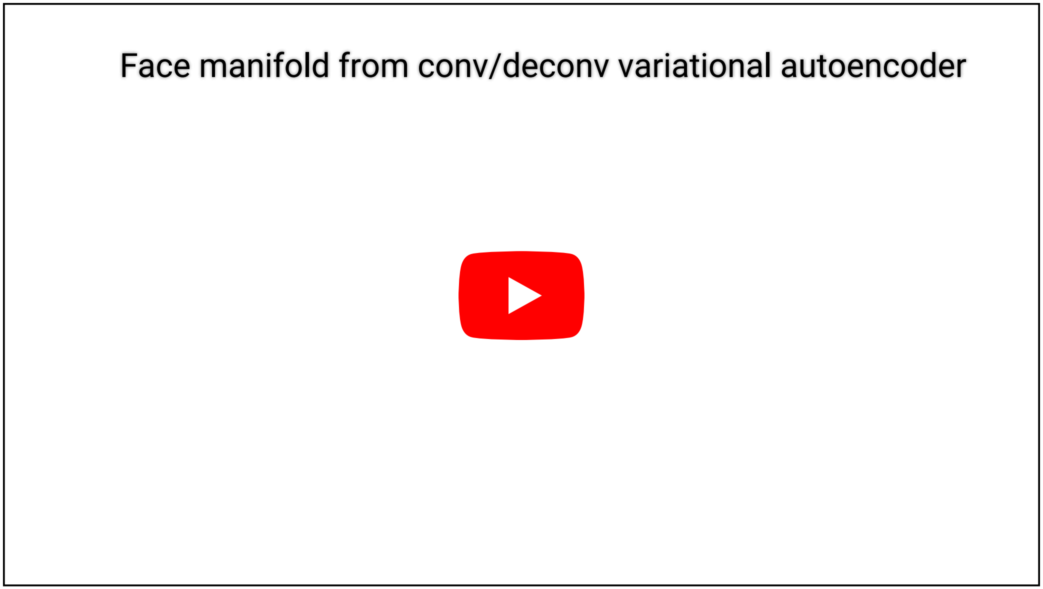


The **encoder** f_ϕ defines an approximate posterior on $\mathbf{z}$.



The reparametrization makes the objective differentiable wrt. θ & ϕ.

Conv/deconv VAEs



conv/deconv VAE trained by Alec Radford in 2015 on Labeled Faces in the Wild (LFW) dataset, 2h on single GTX 980

Variational Autoencoders (VAE)

Remarks

- Similar to Denoising AE but noise added to hidden layer;
- Motivated by a well-defined probablistic model of the generative process;
- Quite easy to train in practice.

Variational Autoencoders (VAE)

Remarks

- Similar to Denoising AE but noise added to hidden layer;
- Motivated by a well-defined probablistic model of the generative process;
- Quite easy to train in practice.

Limitations

- Is the continuous parametrization of posterior latent distribution too restrictive?
- Would a discrete latent variable make more sense?

Discrete latent variables VAE

Gumbel-Softmax / Concrete distribution VAEs

- Adapts the reparametrization trick for a discrete $\mathbf{z}$.
- Trains ok but no ground breaking applications so far.

Discrete latent variables VAE

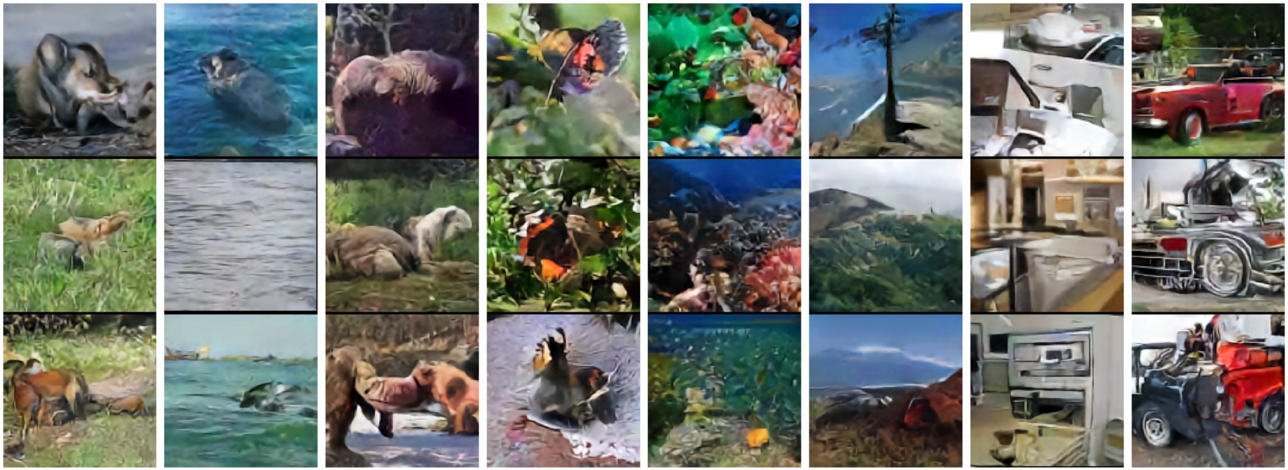
Gumbel-Softmax / Concrete distribution VAEs

- Adapts the reparametrization trick for a discrete $\mathbf{z}$.
- Trains ok but no ground breaking applications so far.

VQ-VAE

- $\mathbf{z}$ is a vector indexed in a trainable embedding matrix.
- Select $\mathbf{z}$ as embedding vector closest to encoder output.
- Approximate backprop via "gradient-copy" trick.
- Very expressive model, especially when combined with strong decoders and priors.

VQ-VAE imagenet results



[Neural Discrete Representation Learning] Aaron van den Oord, Oriol Vinyals, Koray Kavukcuoglu

VQ-VAE speech results

Speech synth demo:
https://avdnoord.github.io/homepage/vqvae/

Example reconstruction:

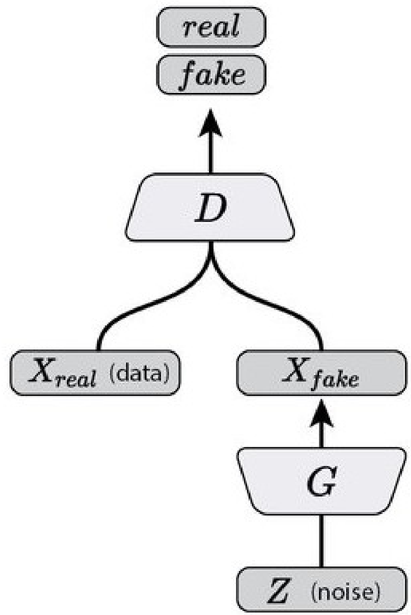
- Original:
- Reconstructed:

Reconstruction conditionned on different speaker id:

- Original:
- Reconstructed:

Generative Adversarial Networks

Generative Adversarial Networks



Alternate training of a **generative network** G and a **discrimininative network** D

Goodfellow, Ian, et al. Generative adversarial nets. NIPS 2014.

GANs

- D tries to find out which example are generated or real
- G tries to fool D into thinking its generated examples are real

GANs

- D tries to find out which example are generated or real
- G tries to fool D into thinking its generated examples are real

Sample real data $x \sim p_{data}$

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

$$D(x) \in [0, 1] \rightarrow 1$$

GANs

- D tries to find out which example are generated or real
- G tries to fool D into thinking its generated examples are real

Sample real data $x \sim p_{data}$

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

$$D(x) \in [0,1] \to 1$$

Sample $\mathbf{z}$ and generate fake data $G(\mathbf{z})$

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

$$D(G(z)) \in [0,1] \to 0$$

GANs

- D tries to find out which example are generated or real
- G tries to fool D into thinking its generated examples are real

Sample real data $x \sim p_{data}$

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

$$D(x) \in [0, 1] \rightarrow 1$$

Sample $\mathbf{z}$ and generate fake data $G(\mathbf{z})$

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

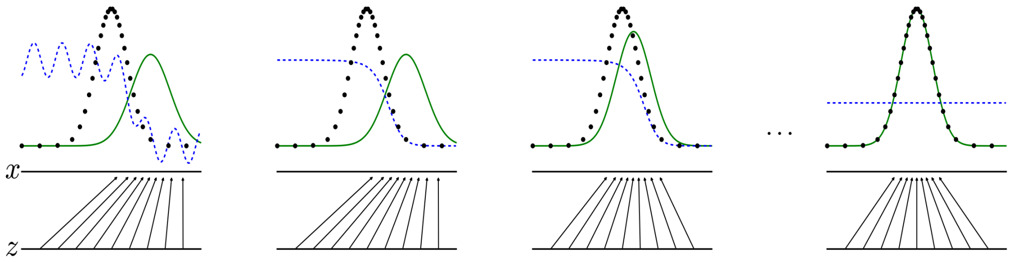
$$D(G(z)) \in [0, 1] \rightarrow 0$$

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

$$D(G(z)) \in [0, 1] \rightarrow 1$$

GANs

1D-example

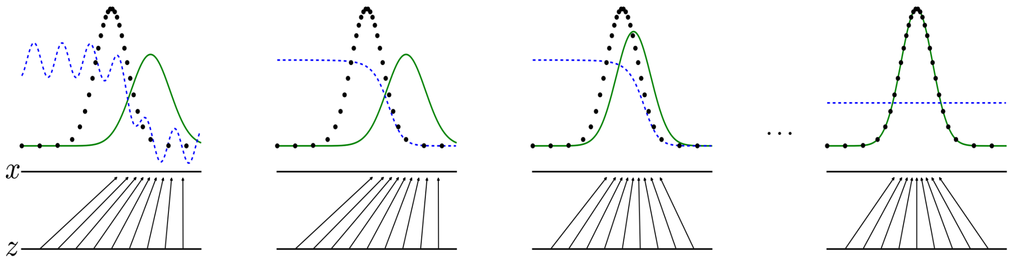


(a) (b) (c) (d)

- optimal: $D = \frac{1}{2}$, $G(\mathbf{z}) = p_{data}$

GANs

1D-example

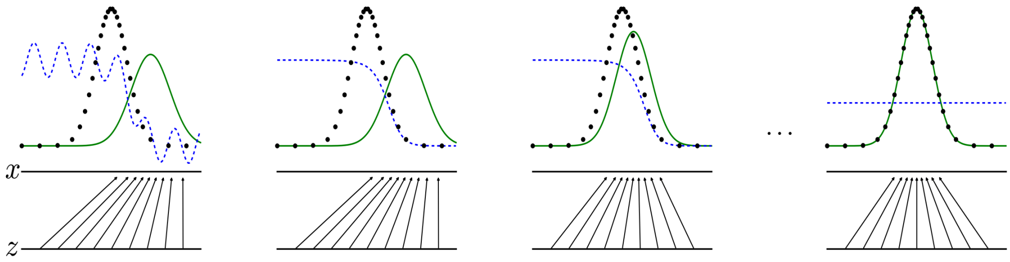


(a) (b) (c) (d)

- optimal: $D = \frac{1}{2}$, $G(\mathbf{z}) = p_{data}$
- G never "sees" training data, it is solely updated from gradients coming from D

GANs

1D-example

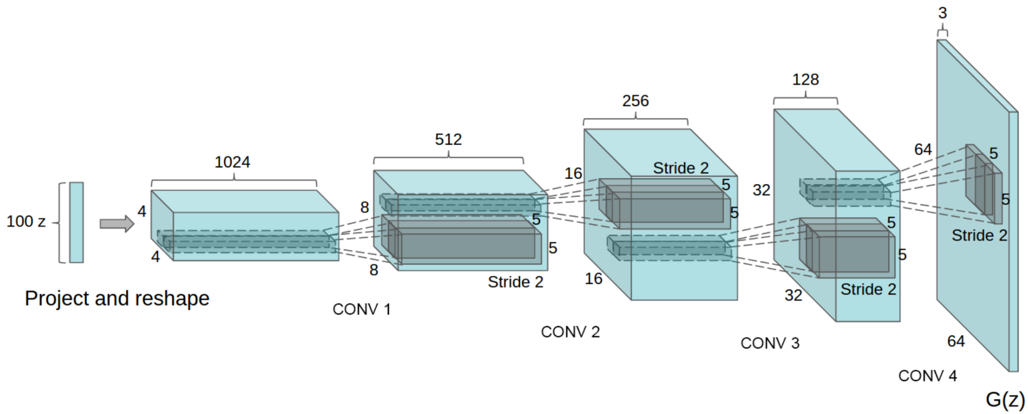


(a) (b) (c) (d)

- optimal: $D = \frac{1}{2}$, $G(\mathbf{z}) = p_{data}$
- G never "sees" training data, it is solely updated from gradients coming from D
- Naive Keras implementation:

```
d_loss = K.mean(-K.log(Dx) - K.log(1 - DGz))
g_loss = K.mean(K.log(1 - DGz))
```

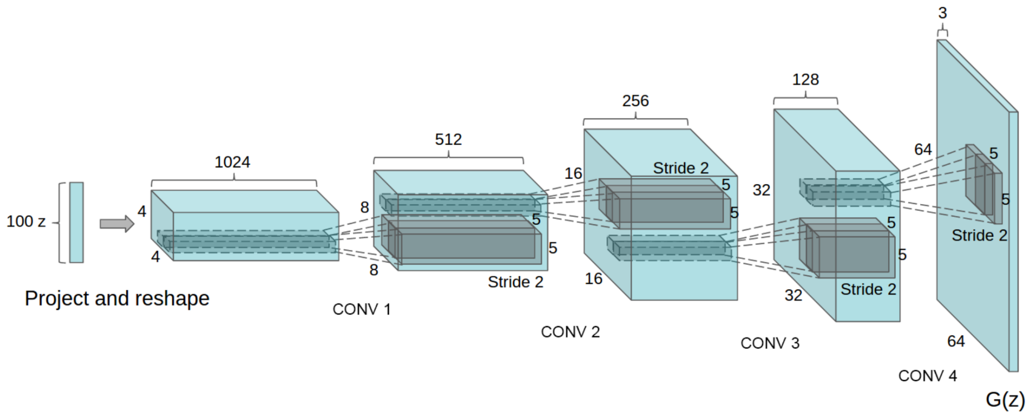
DC-GAN



- GANs training is unstable, and may suffer from **mode collapse**

Radford, Alec, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. 2015.

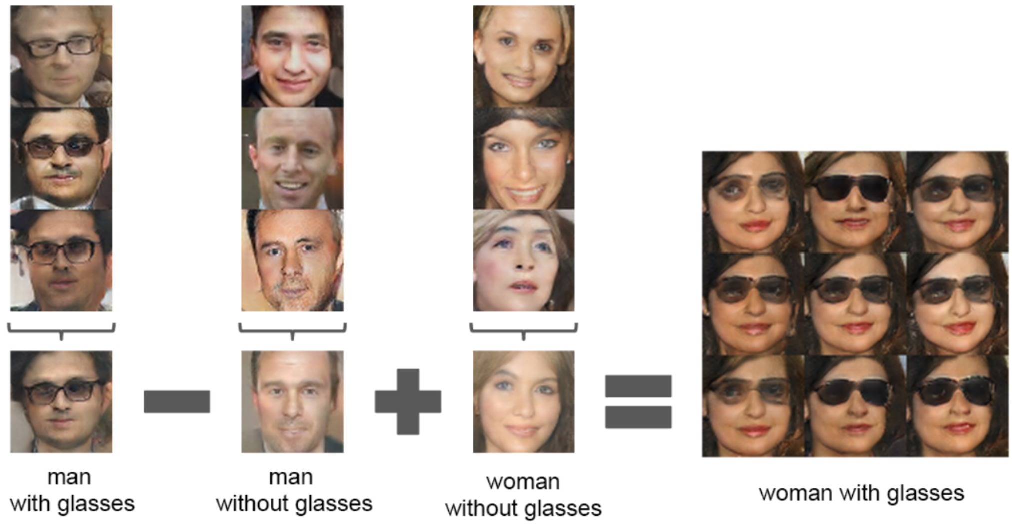
DC-GAN



- GANs training is unstable, and may suffer from **mode collapse**
- **Sensitive hyperparameters:** Use of batchnorm, strided convolutions, careful learning rates, several D updates per G updates...

Radford, Alec, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. 2015.

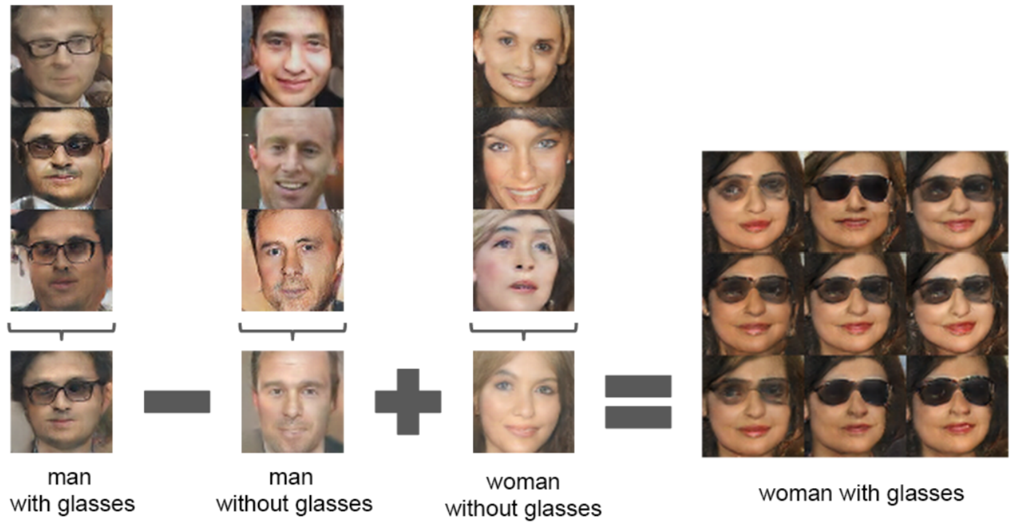
DC-GAN



- Generator generates less-blurry images than VAEs

Radford, Alec, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. 2015.

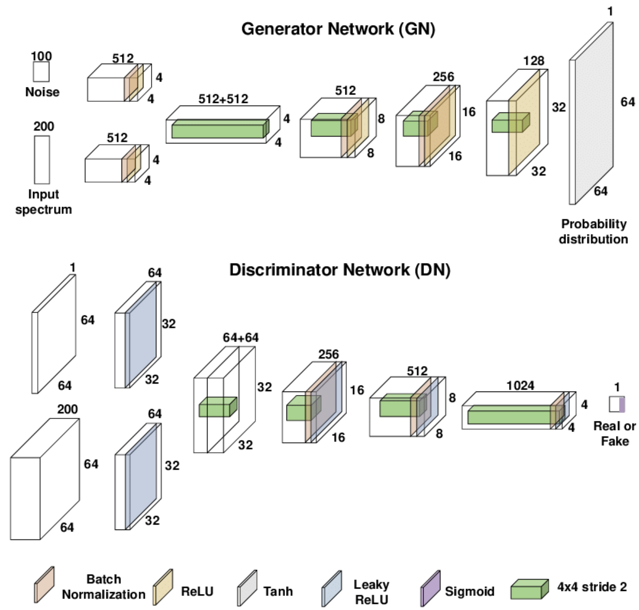
DC-GAN



- Generator generates less-blurry images than VAEs
- Latent space has some local linar properties (vector arithmetic like with Word2Vec)

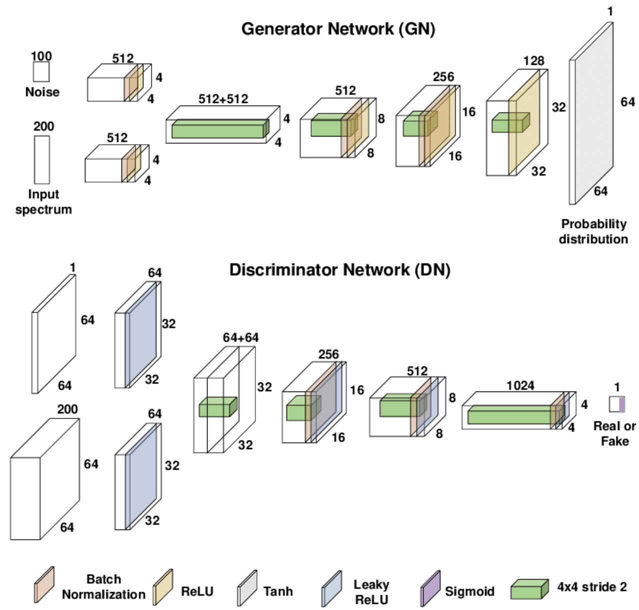
Radford, Alec, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. 2015.

cDC-GAN



Mehdi Mirza, Simon Osindero. "Conditional Generative Adversarial Nets". NeurIPS 2014.

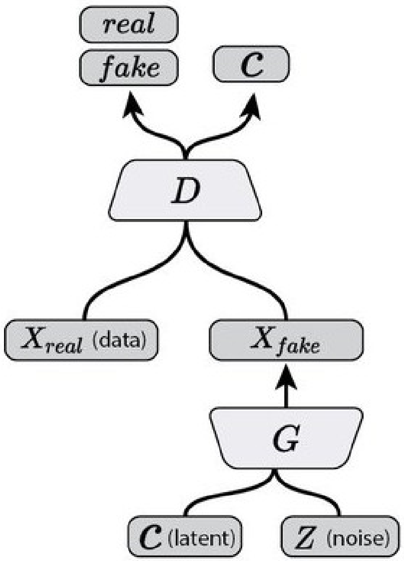
cDC-GAN



- Can generate image of the wanted class
- Simply encode for both discriminator and generator the labels and concatenate it to features

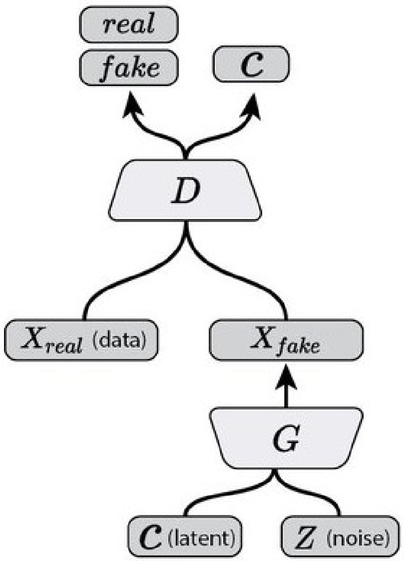
Mehdi Mirza, Simon Osindero. "Conditional Generative Adversarial Nets". NeurIPS 2014.

Info GAN



Chen, Xi, et al. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. NIPS, 2016.

Info GAN



Separate noise $\mathbf{Z}$ and latent variables $\mathbf{C}$ and maximizes the mutual information between c the and the observation $G(z, c)$

Chen, Xi, et al. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. NIPS, 2016.

Info GAN

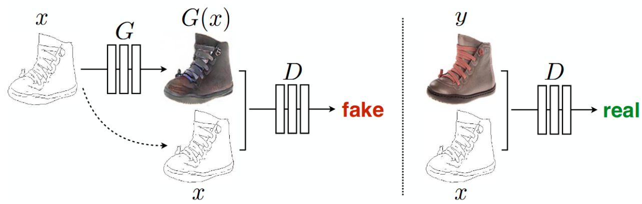
Chen, Xi, et al. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. NIPS, 2016.

Progressive growing of GANs

All images are generated by walking through the latent space

Karras, Tero, et al. Progressive growing of gans for improved quality, stability, and variation. 2017.

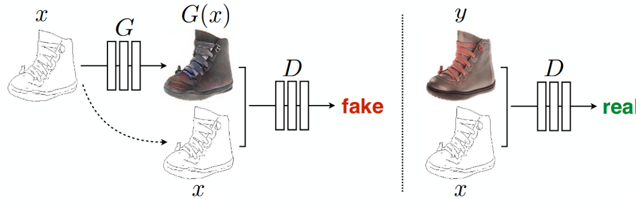
Pix2pix: Conditional GANs



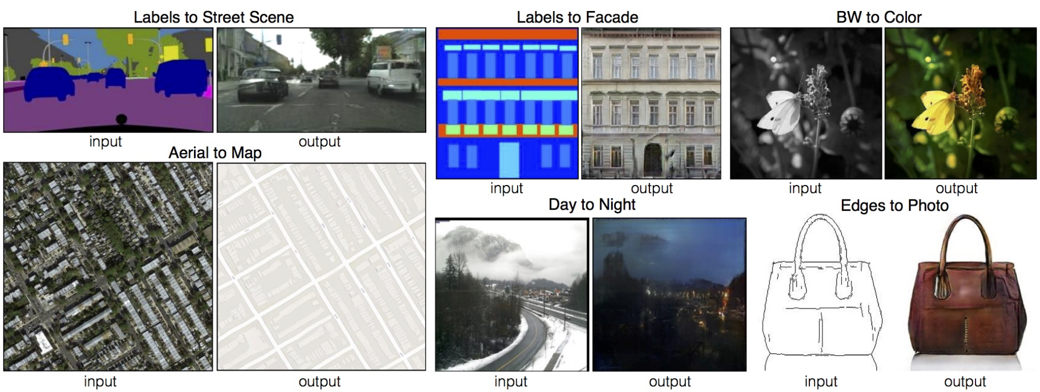
The generation no longer makes use of $\mathbf{z}$, rather is conditionned by an input $\mathbf{x}$

Isola, Phillip et al. Image-to-Image Translation with Conditional Adversarial Networks, CVPR 2017

Pix2pix: Conditional GANs

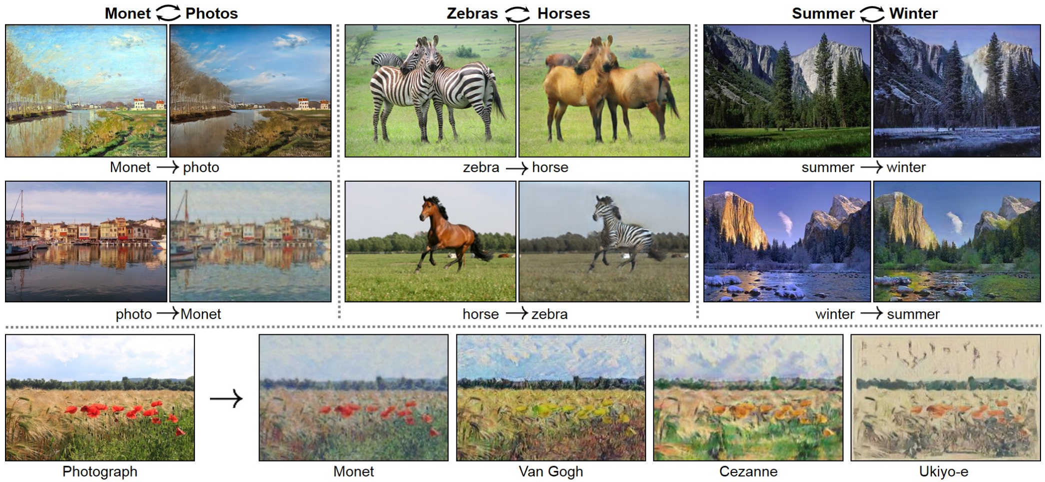


The generation no longer makes use of $\mathbf{z}$, rather is conditionned by an input $\mathbf{x}$



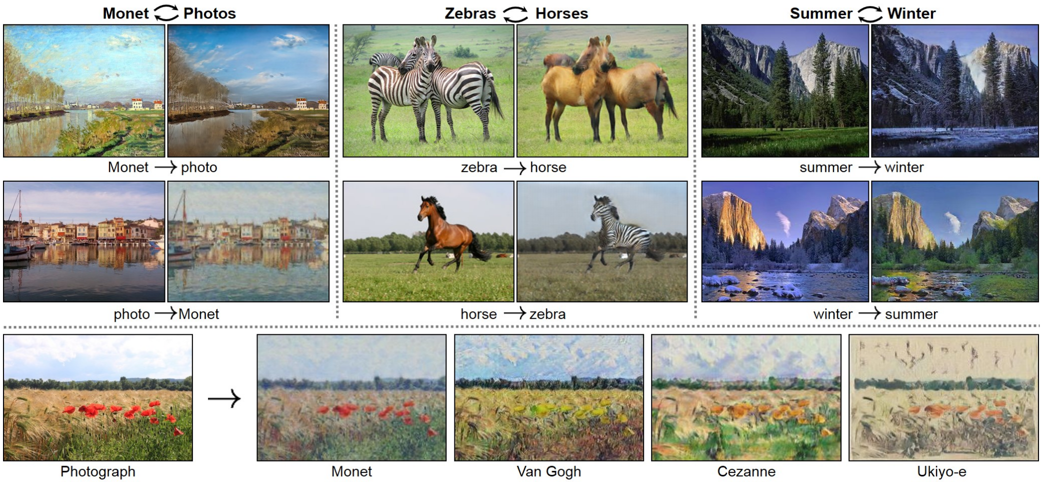
Isola, Phillip et al. Image-to-Image Translation with Conditional Adversarial Networks, CVPR 2017

Cycle GANs



Jun-Yan Zhu et al. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV 2017

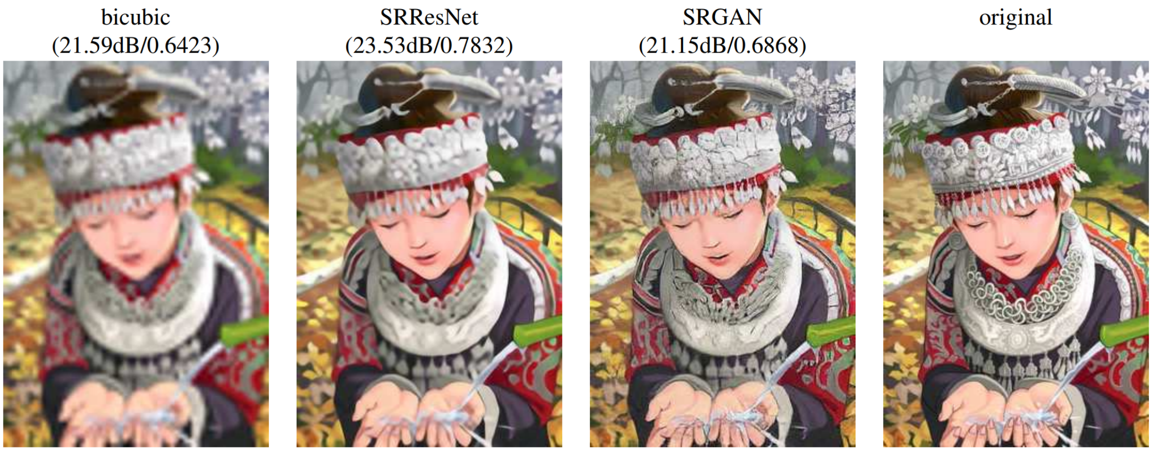
Cycle GANs



- No alignment between pairs needed, simply two different sets of images

Jun-Yan Zhu et al. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV 2017

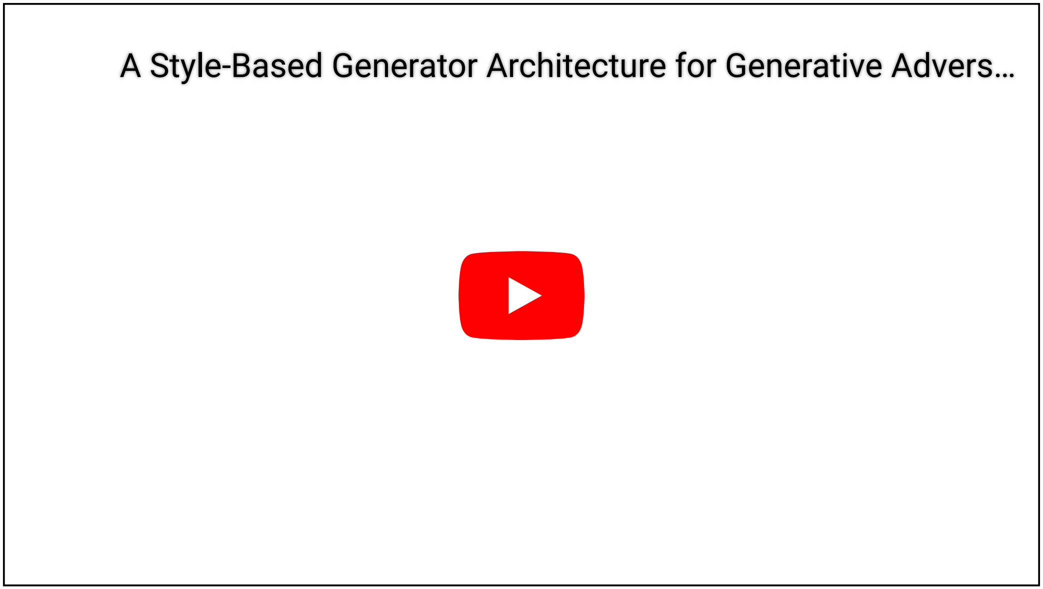
Super Resolution



"Perceptual" loss = combining pixel-wise loss mse-like loss with GAN loss

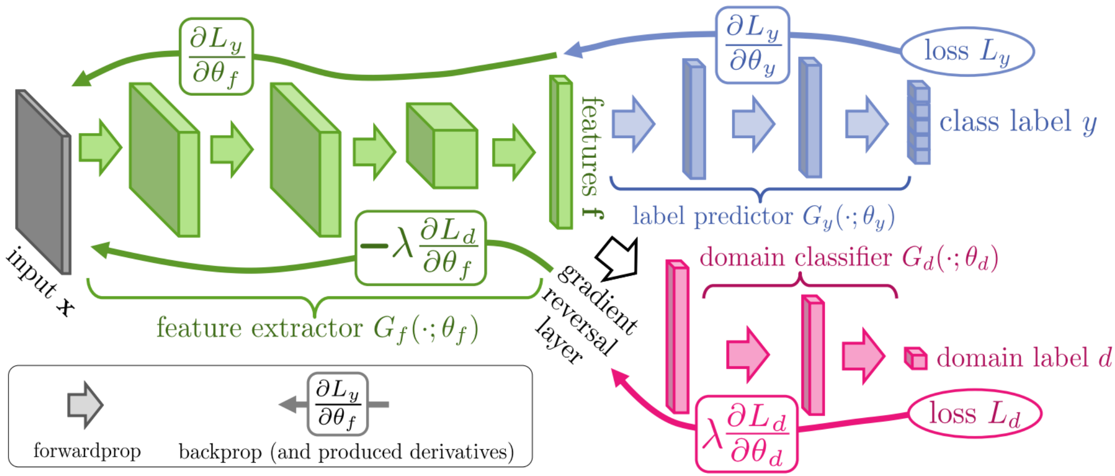
Ledig, Christian, et al. Photo-realistic single image super-resolution using a generative adversarial network. CVPR 2016.

Style GANs



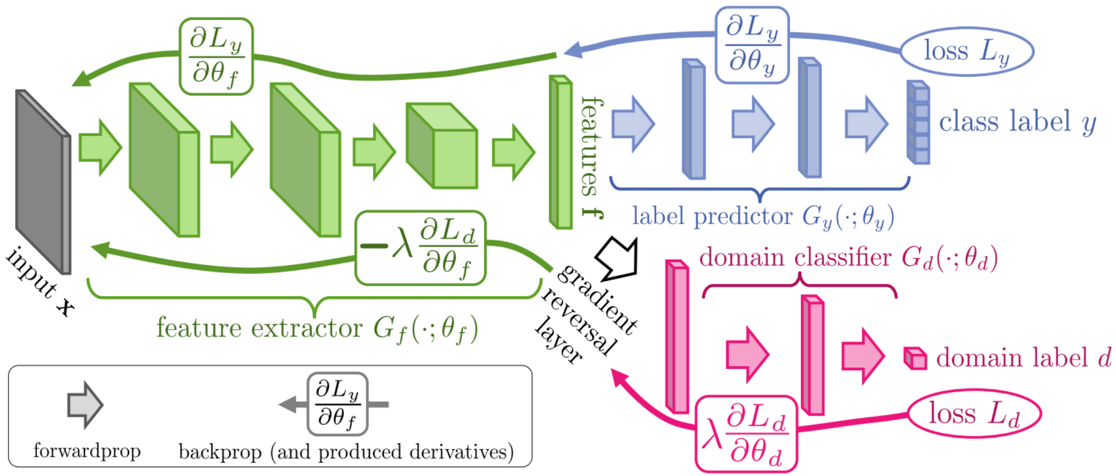
A Style-Based Generator Architecture for Generative Adversarial Networks by Tero Karras, Samuli Laine, Timo Aila, 2018 (preprint)

Domain Adversarial Training



Ganin, Yaroslav, et al. Domain-adversarial training of neural networks. JMLR 2016.

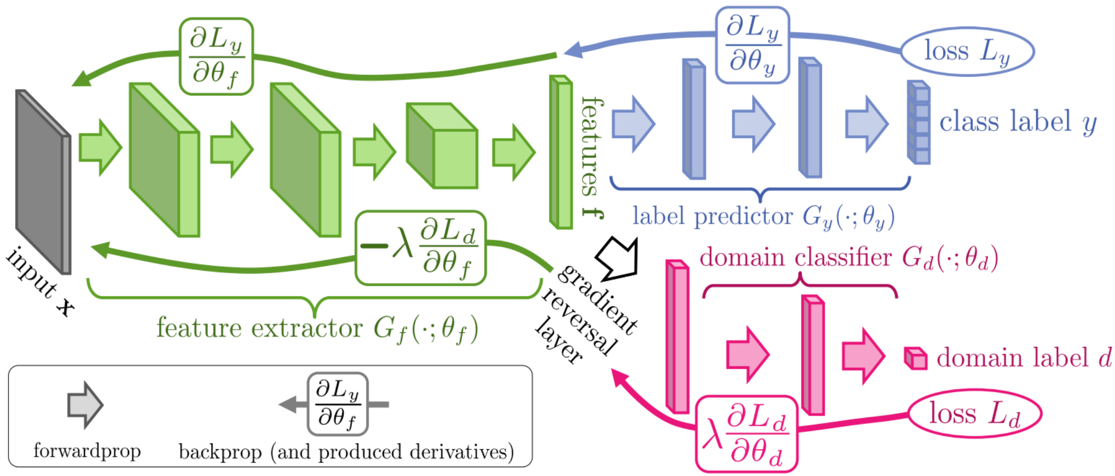
Domain Adversarial Training



- Forces the features (green) **not** to be specialised in discriminating between domains

Ganin, Yaroslav, et al. Domain-adversarial training of neural networks. JMLR 2016.

Domain Adversarial Training

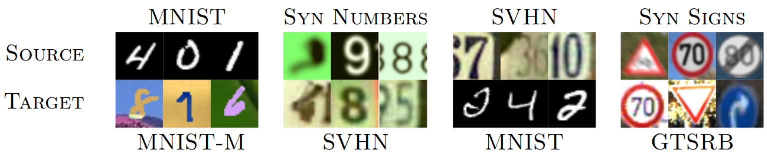


- Forces the features (green) **not** to be specialised in discriminating between domains
- Easy to implement in TensorFlow / Pytorch with a `GradientReversalLayer`

Ganin, Yaroslav, et al. Domain-adversarial training of neural networks. JMLR 2016.

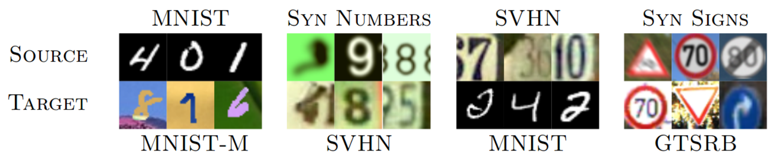
Domain Adversarial Training

- Train **labeled** source domain + **unlabeled** target domain



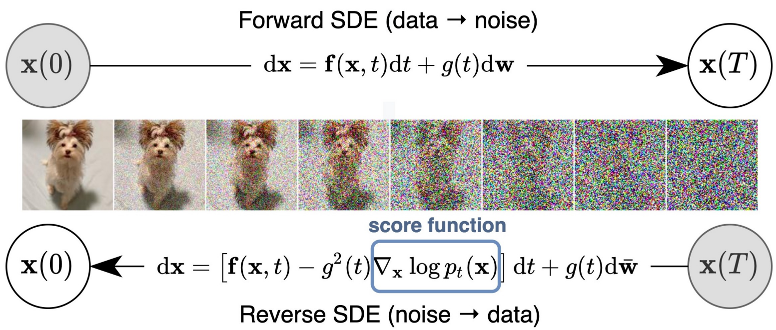
Domain Adversarial Training

- Train **labeled** source domain + **unlabeled** target domain



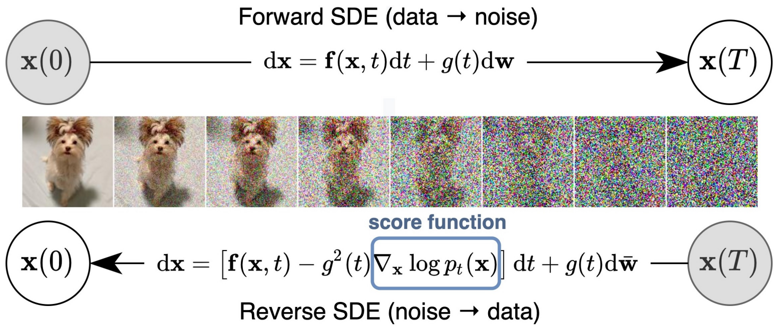
- Reprensentation tends to be **less biased** towards the domain

Score-based matching (denoising diffusion)

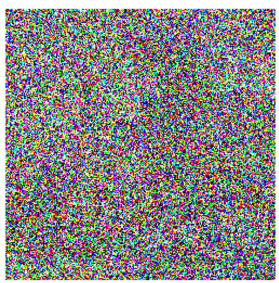


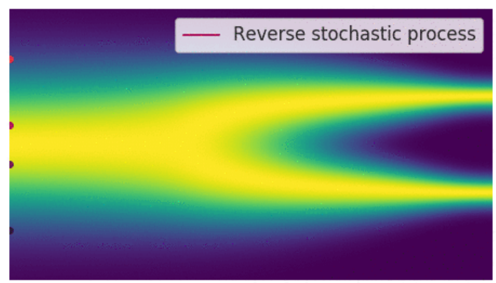
Yang Song, and S. Ermon. [Generative Modeling by Estimating Gradients of the Data Distribution](), NeurIPS 2019.
Jonathan Ho, A. Jain, P. Abbeel, [Denoising Diffusion Probabilistic Models]()

Score-based matching (denoising diffusion)



Generate images by gradually denoising random noise





Yang Song, and S. Ermon. [Generative Modeling by Estimating Gradients of the Data Distribution](), NeurIPS 2019.
Jonathan Ho, A. Jain, P. Abbeel, [Denoising Diffusion Probabilistic Models]()

Takeaways

(Reconstruction) Autoencoders

- have no direct probabilistic interpretation;
- are not designed to generate useful samples;
- encoder defines a useful latent representation.

Takeaways

(Reconstruction) Autoencoders

- have no direct probabilistic interpretation;
- are not designed to generate useful samples;
- encoder defines a useful latent representation.

VAEs

- model explicitly (a lower bound of) the likelihood;
- high quality samples from high dimensional distributions;
- encoder defines a useful latent representation;
- optimization problem is often well-behaved.

Takeaways

GANs

- likelihood-free generative models;
- high quality samples from high dimensional distributions;
- discriminator not meant be used as encoder;
- optimization problem is trickier than for VAEs (open research).

Takeaways

GANs

- likelihood-free generative models;
- high quality samples from high dimensional distributions;
- discriminator not meant be used as encoder;
- optimization problem is trickier than for VAEs (open research).

There exists other kinds of generative models:

- auto-regressive models: PixelCNN, WaveNet, RNN language models...
- can be used as prior and decoder for VAEs, generators for GANs.

Takeaways

Adversarial training is useful beyond generative models:

- domain adaptation;
- learning representations blind to sensitive attributes;
- defend against malicious inputs (adversarial examples);
- regularization by training on adversarial examples.

Takeaways

Adversarial training is useful beyond generative models:

- domain adaptation;
- learning representations blind to sensitive attributes;
- defend against malicious inputs (adversarial examples);
- regularization by training on adversarial examples.

Quality of samples from VAE and GAN depends a lot on the architectures of sub-networks.

Lab 10: back here in 15min!