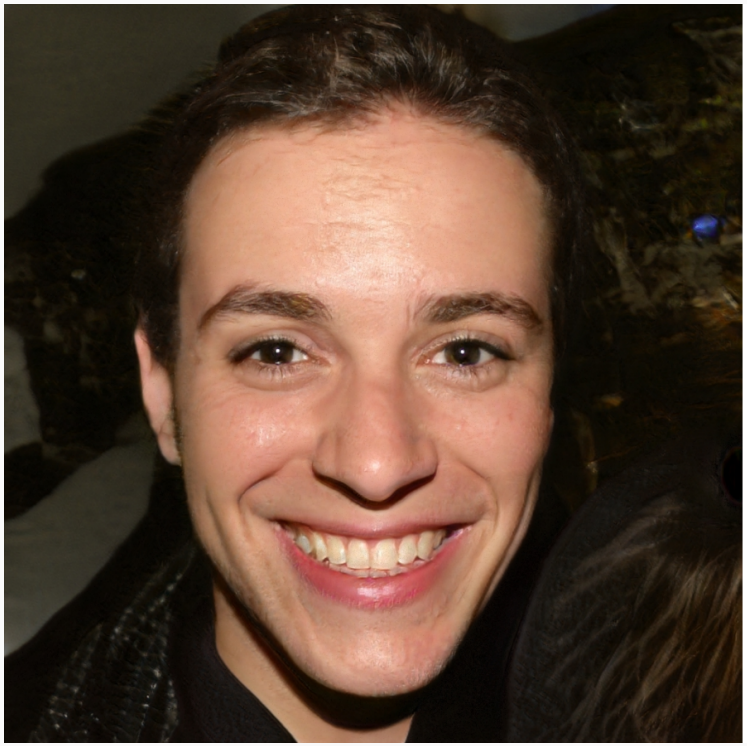
Generative Networks

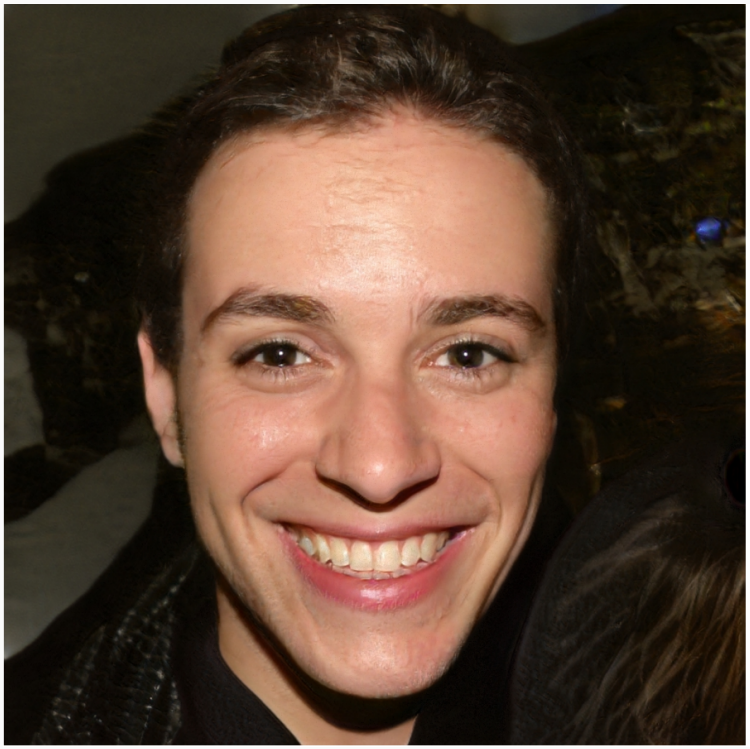
October 7, 2022

Let’s play a game: which face is real ?



Which face is real?.

Let's play a game...



Which face is real.

Motivation

Generative Networks

- A discriminative model is a way to model the conditional probability of a target Y (low-dimension) given some covariates X (high-dimension).
- Conversely, a generative model tries to model the conditional probability of X given Y (or even the joint probability $X \times Y$

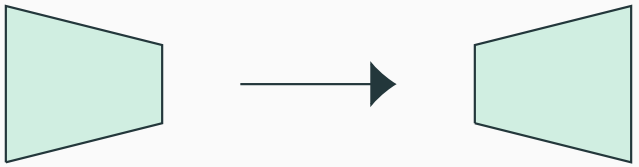
Figure 1: Sampling from $P(X|Y)$ on MNIST using a ConditionalGan (Mirza and Osindero 2014)

3 different types of generative networks

1. **GANs**: GAN provides a smart solution to model the data generation, an unsupervised learning problem, as a supervised one.
2. **VAEs**: VAE inexplicitly optimizes the log-likelihood of the data by maximizing the evidence lower bound (ELBO).
3. **Flows**: A flow-based generative model is constructed by a sequence of invertible transformations, and therefore maximizes the negative log-likelihood.

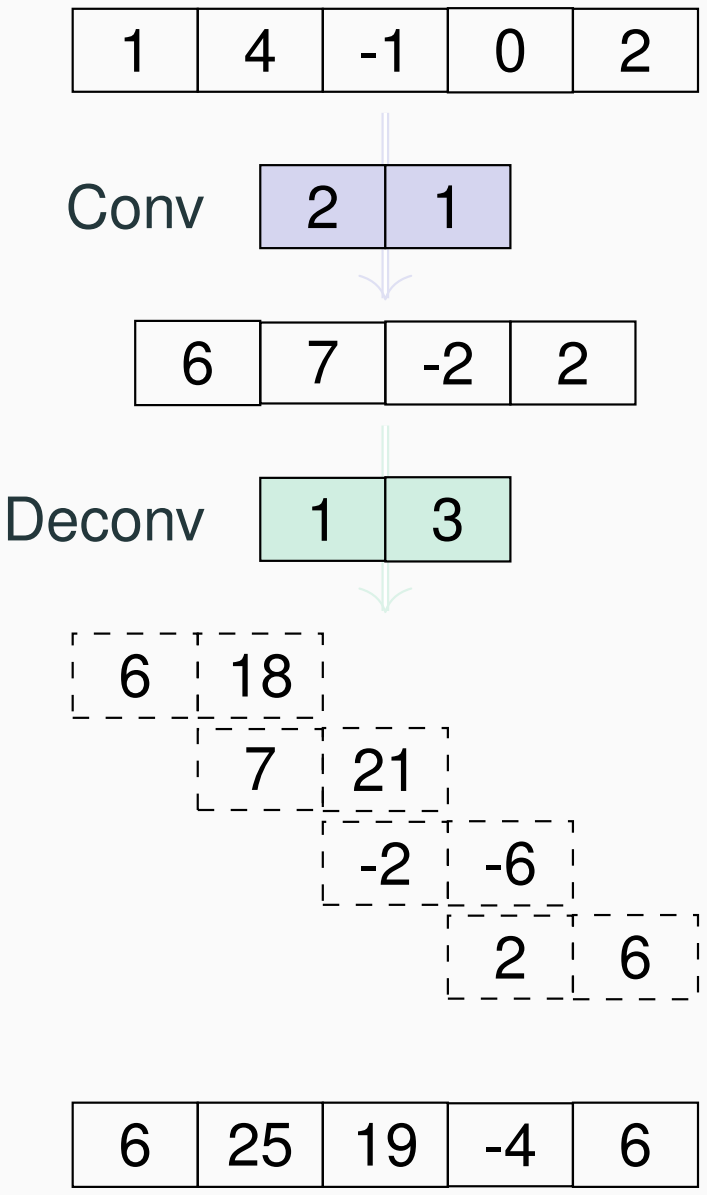
Inverse convolutions

- Our objective is to expand the signal from a low-dimension representation to an high-dimension signal space.
- In feed-forward networks, the objective was to reduce the signal dimension using for instance conv layers



- To do the opposite, we introduce the inverse convolutional operator

Inverse convolutions (1D case)



$$\begin{pmatrix} 2 & 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 2 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 3 & 0 & 0 & 0 \\ 0 & 1 & 3 & 0 & 0 \\ 0 & 0 & 1 & 3 & 0 \\ 0 & 0 & 0 & 1 & 3 \end{pmatrix}^T$$

Inverse convolutions (1D case)

Mathematically speaking,

- Convolution for a kernel k

$$(x \circledast k)_i = \sum_j x_j \cdot k_{j-i+1}$$

- If $y = x \circledast k$, then

$$\frac{\partial \ell}{\partial x} = \frac{\partial \ell}{\partial y} * k \quad ,$$

  where $*$ is similar to $\circledast$ except that the coefficients are visited in reverse order (transposed convolution).

Inverse convolutions

- Applying convolution + inverse convolution will keep the signal "roughly" unchanged
  (intuition: mass of $K \cdot K^T$ will concentrate on the diagonal)
- We can define stride, padding and dilatation similarly to regular convolution
- Since it's an upscaling operation, it can creates artifacts on the resulting image especially when stride > 1

1	1	2	1	2	1	2	1	1

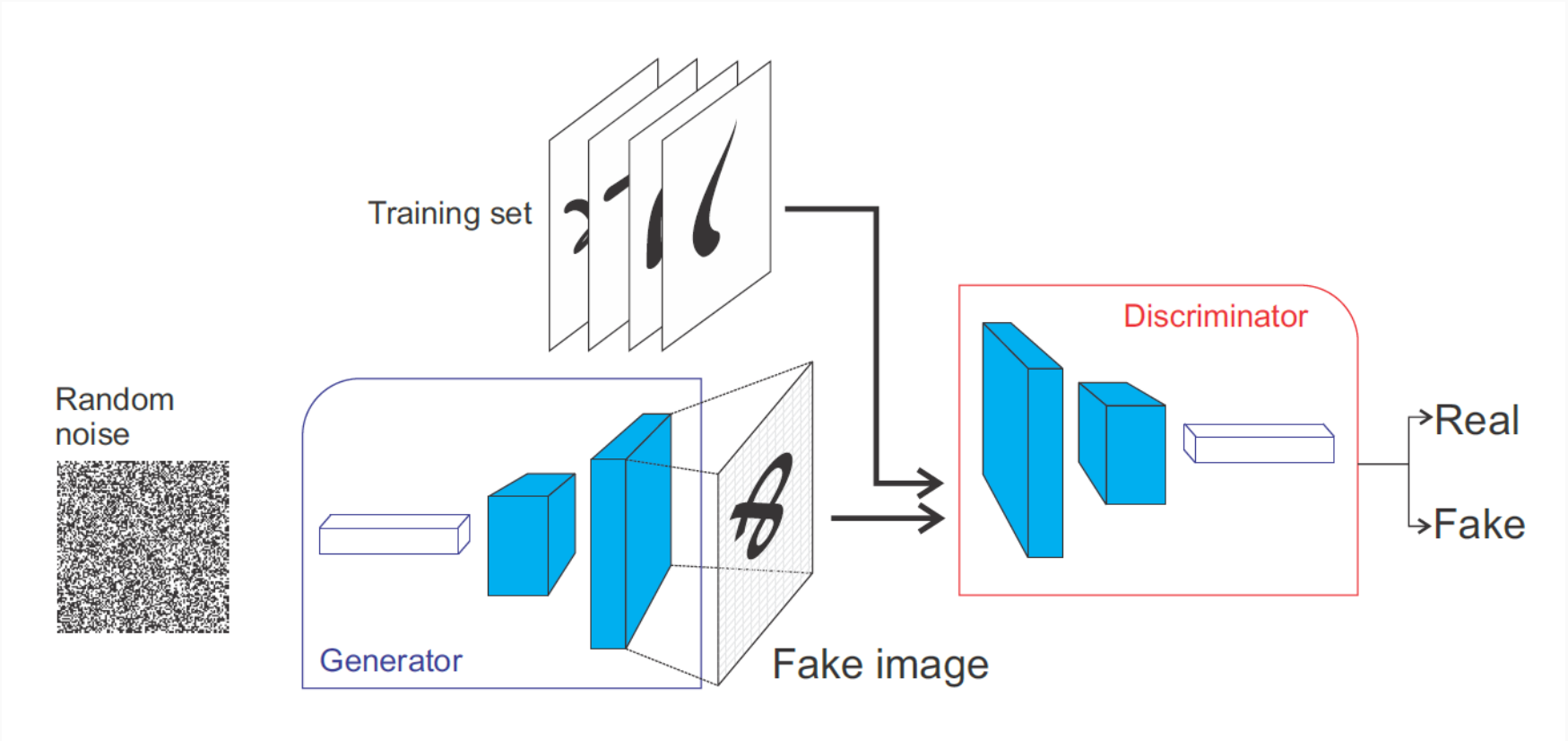
Figure 2: Result of $(1, 1, 1, 1) \circledast (1, 1, 1)$ (stride 2)

- In some cases, it's better to combine this with interpolation.
- https://distill.pub/2016/deconv-checkerboard/

Outline

Generative Adversarial Nets

GANs Goodfellow et al. 2014



Source: medium.

Motivation: generating artificial contents.

criteo AI Lab

Pros

- Simple generation.
- Work extremely well with high-dimensional data.
- Allow manifold discovering: image interpolation.

Abdal et al. 2019.

Cons

- Unknown probability density function: we cannot easily check low density areas.
- Tricky training.

Art: Edmond de Belamy.

https://en.wikipedia.org/wiki/Edmond_de_Belamy

Attacking classifiers with GANs

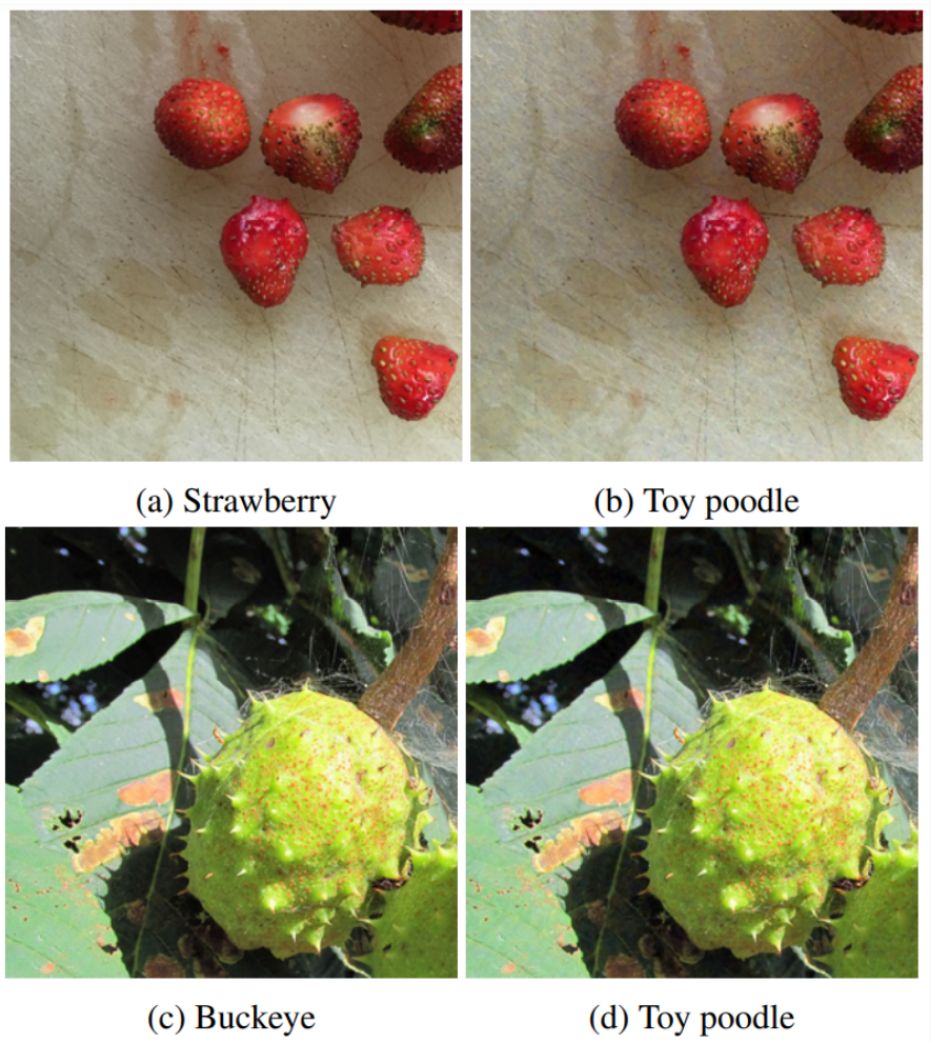


Figure 3: Xiao et al. 2018

Defending classifiers with GANs

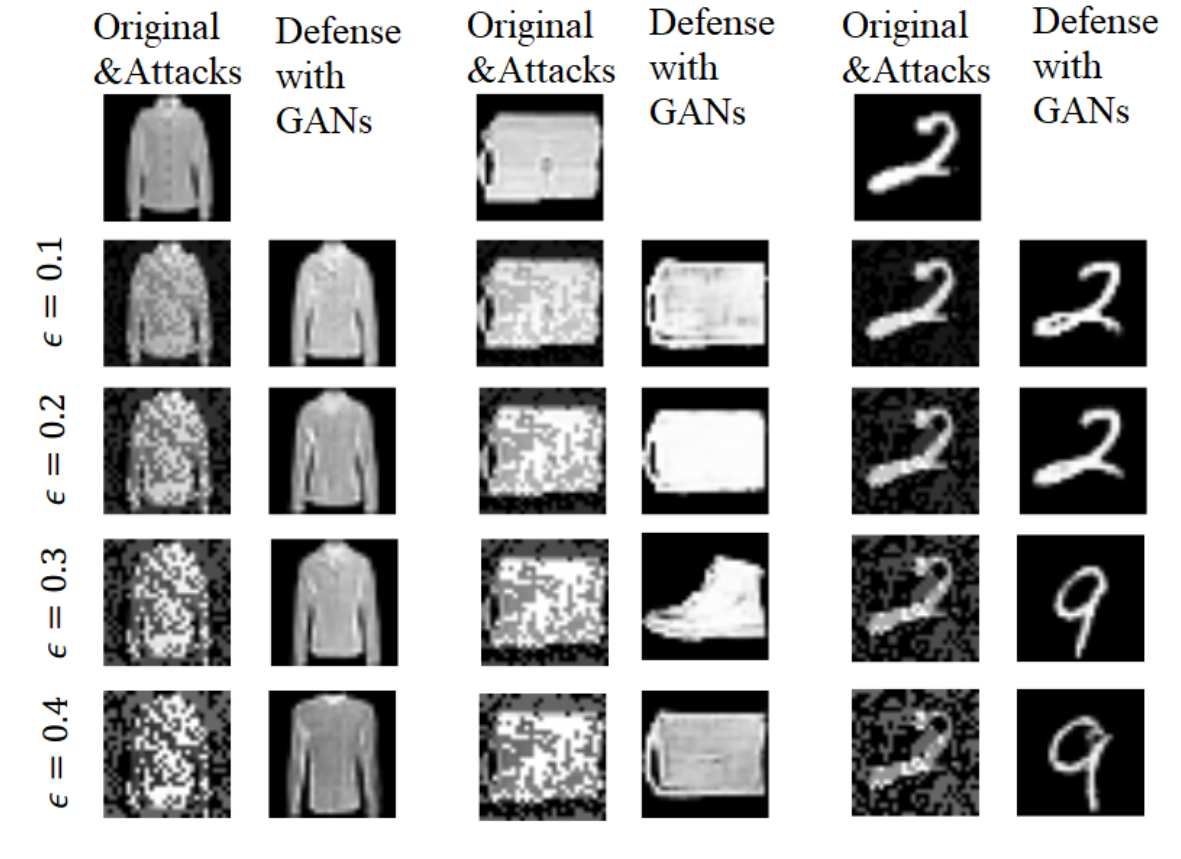


Figure 4: Samangouei et al. 2018

GANs (cted)

- Let us consider a generator G parametrized by θ and a discriminator D parametrized by ϕ
  and
  - $(x^i)_{i=1...n}$ a batch of n training images
  - $(z^i)_{i=1...n}$ a batch of n noise samples sampled from a fixed noise prior.
- The goal of the discriminator is to distinguish between $G(z)$ and x so minimize the negative log-likelihood

$$NLLH(x, z, \theta) = -\left[\sum_{i=1}^{n} log(D_\theta(x^i)) + log(1 - D_\theta(G_\phi(z^i))\right] .$$

- The goal of the generator is to minimize the log-likelihood

$$LLH(x, \phi) = \sum_{i=1}^{n} log(1 - D_\theta(G_\phi(z^i))$$

GANs: Pathological behaviors

- **Oscillation / bad convergence**
  Due to minimax game
- **Mode collapse**
  Happens when the training data is multi-modal (which is usually the case in practice): can be a good strategy for the generator to target the easiest mode of the target distribution (pullover in the example below)

GANs: Alchemy ?

Lots of "hacks" to stabilize the training

1. Normalize the inputs
2. $min\, log(1 - D)$ vs $max\, log(D)$
3. Choose the noise prior wisely
4. BatchNorm on full real / fake images
5. Avoid Sparse Gradients (ReLu -> LeakyReLu)
6. Use soft / noisy labels
7. Choose the optimizers wisely (e.g. Adam for G, SGD for D)
8. ...

(e.g. https://github.com/soumith/ganhacks)

GANs: (A bit of) theory

- Let us denote
  - μ the density of the true data
  - $\mu_G = G(\mu_{\text{noise}})$ the density of the data generated by a generator G
- Our main goal is to find G that minimizes a well-chosen distance between μ and μ_G
- **Intuition**: the performance of the best discriminator mesures this gap between μ and μ_G (the bigger the gap, the better the optimal discriminator).

Can we formalize this intuition ?

Theorem

The optimal discriminator (without regularization) D_G^* is

$$x \to \frac{\mu(x)}{\mu(x) + \mu_G(x)} \quad .$$

The corresponding loss at this point is

$$\mathcal{L}_G(D_G^*) = 2\mathbb{D}_{JS}(\mu, \mu_G) - log(4) \quad ,$$

where $\mathbb{D}_{JS}$ is the Jensen-Shannon divergence (symmetric variant of the KL-divergence).

Training the GAN $\equiv$ finding G that minimizes $\mathbb{D}_{JS}(\mu, \mu_G)$

Drawbacks of GANs

Drawbacks of orignal GANs formulation...

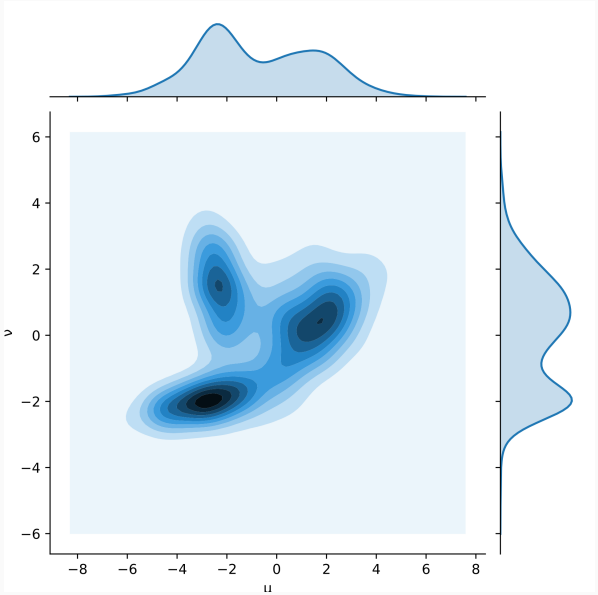
- The training process of GANs is unstable.
- Mode collapse phenomenon.
- Arjovsky, Chintala, and Bottou (2017): Wasserstein GANs.
- Authors claim that the Jensen-Shannon divergence does not allow to take into account the metric structure of the space.
- WGANs have become a standard in machine learning.

Wasserstein GANs

- They propose to go with the Wasserstein distance $\mathbb{D}_{W_1}$.

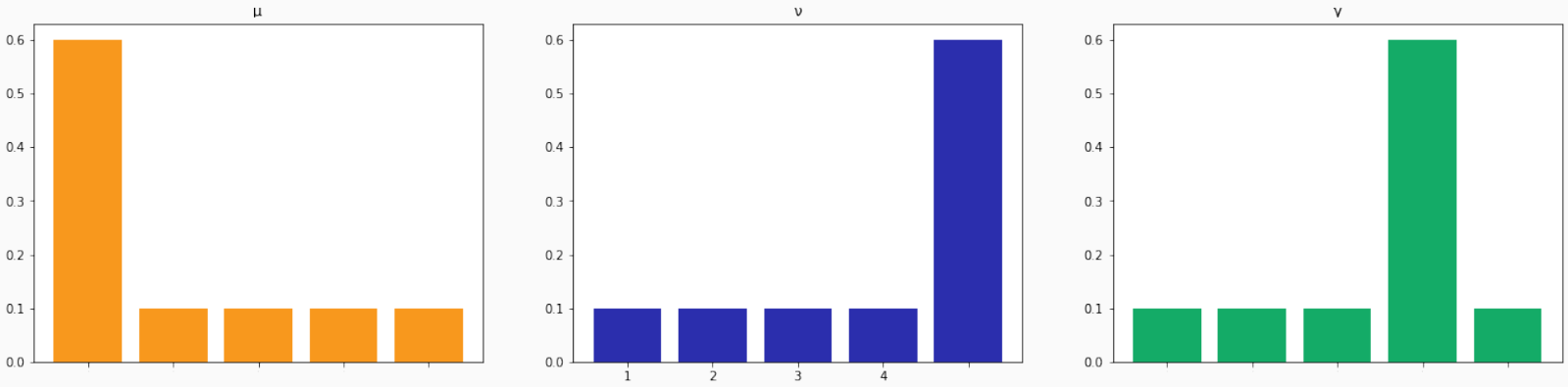
$$\mathbb{D}_{W_1}(\mu, \nu) = \inf_{\gamma \in \Gamma(\mu,\nu)} \int d(x, y) d\gamma(x, y)$$

- Continuous "earth moving distance"



Wasserstein GANs (cted)

Advantages of $\mathbb{D}_{W_1}$ over $\mathbb{D}_{JS}$?



$$\mathbb{D}_{W_1}(\mu, \nu) = 2 > \mathbb{D}_{W_1}(\mu, \gamma) = 1.5$$

$$\mathbb{D}_{JS}(\mu, \nu) = 0.20 < \mathbb{D}_{JS}(\mu, \gamma) = 0.25$$

Problem: How to compute $argmin_G \mathbb{D}_{W_1}(\mu, \mu_G)$?

Wasserstein GANs (cted)

- Using Kantorovich-Rubinstein duality theorem,

$$\mathbb{D}_{W_1}(\mu, \mu_G) = \max_{\|D|_L \leqslant 1} \left[\mathbb{E}_{X\sim\mu}[D(X)] - \mathbb{E}_{X\sim\mu_G}[D(X)]\right] \quad ,$$

  where $\|D|_L$ is the Lipschitz semi-norm equal to

$$\max_{x,y} \frac{\|D(x) - D(y)\|}{\|x - y|} \quad .$$

- We get a **new loss** for the discriminator !

Controlling the gradient of the discriminator?

The compactness requirement is classical when parameterizing GANs.

1. Weight clipping Arjovsky et al. 2017.
2. Gradient penalty Gulrajani et al. 2017.
3. Spectral normalization Miyato et al. 2018.
4. Bjorck orthonormalization.

The role of the discriminator

In practice, one has always $\mathcal{D} = \{D_\alpha : \alpha \in \Lambda\}$

$$\sup_{\alpha \in \Lambda} \Big[\mathbb{E} \log(D_\alpha(X)) + \mathbb{E} \log(1 - D_\alpha(G_\theta(Z))) \Big]$$

acts like a divergence between the distributions μ_θ and the empirical distribution μ_n.

- Neural net divergence Arora et al. 2017
- Adversarial divergence Liu et al. 2017

Other variants of GANs

1. Least squares GANs Mao et al. 2017: related to the Pearson-ξ^2 div.

$$\sup_{\alpha\in\Lambda} \sum_{i=1}^{n} (D_\alpha(X_i) - 1)^2 + \sum_{i=1}^{n} D_\alpha(G_\theta(Z_i))^2,$$

$$\inf_{\theta\in\Theta} \sum_{i=1}^{n} (D_\alpha(G_\theta(Z_i)) - 1)^2.$$

2. Nowozin et al. 2016 proposed f-GANs and showed that any f-divergence can be used for training GANs:

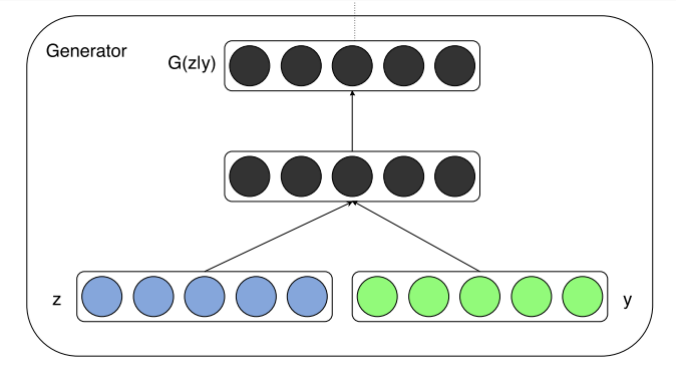
$$\inf_{\theta\in\Theta} \sup_{\alpha\in\Lambda} \mathbb{E} D_\alpha(X) - \mathbb{E}(f^\star \circ D_\alpha)(G_\theta(Z)), \quad f^\star \text{ convex conjugate.}$$

3. **WGANs**.

Text2Image Generation

Conditional GANs

- First introduced by Mirza and Osindero 2014: use additional conditioning input into your GAN (typically a label)
- The conditionning input is given both to the generator and the discriminator



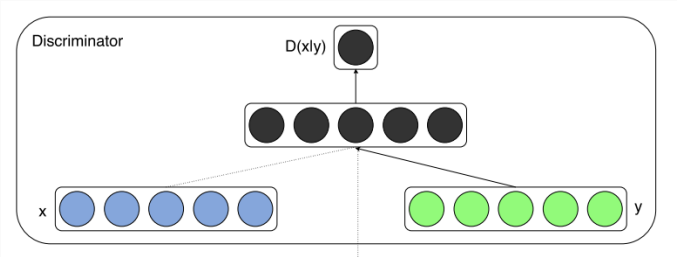
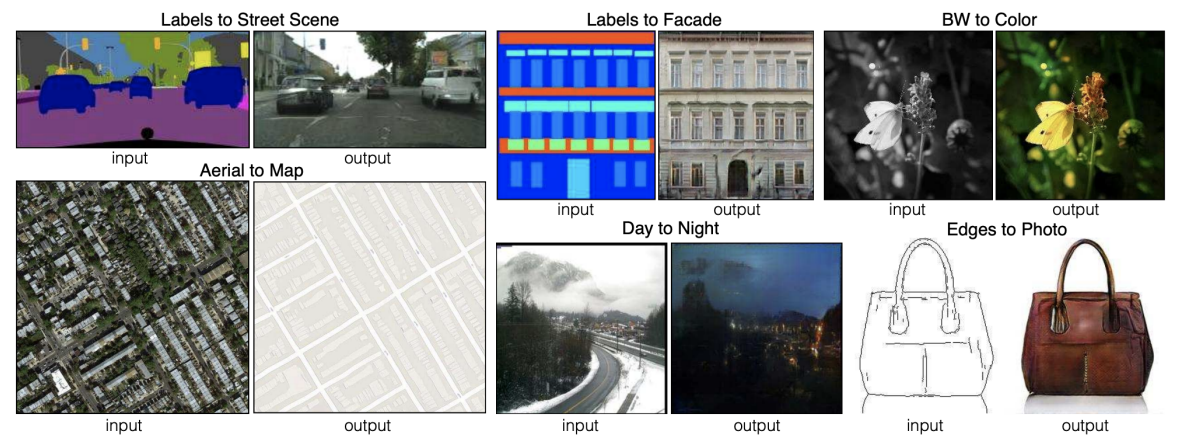


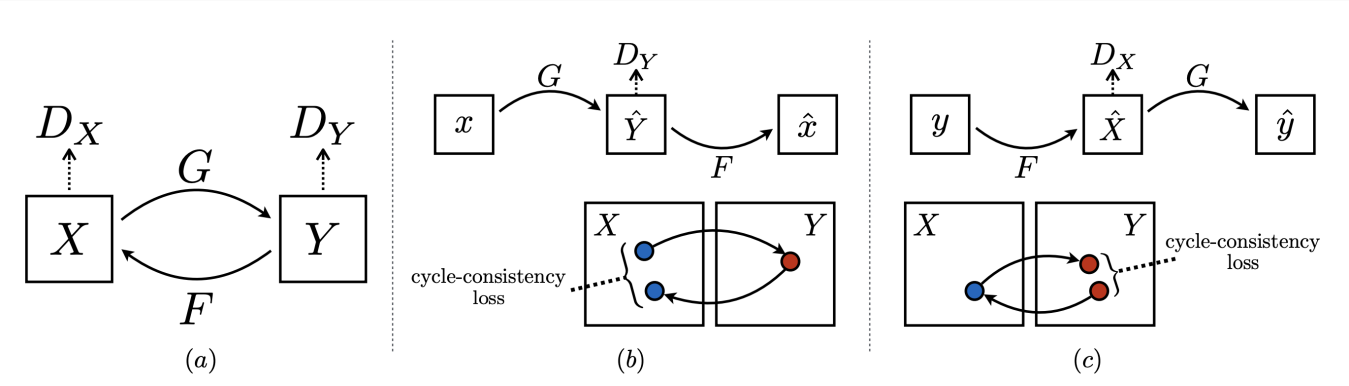
Image translation with conditional GANs

- Conditioning can be anything (text, image, . . .)
- If we have a paired dataset, we can perform some image translation by using the source image as conditionner and adding a reconstruction term to the GAN loss.
- **Pix2Pix** (Isola et al. 2017)



Going unpaired (and invertible !)

- What if we don't have a paired dataset but just two collections of images (source and target) ?
- Key idea 1: Let's make the generator G invertible and use the reconstruction loss on $G^{-1} \circ G$!
- Key idea 2: Flip source and target and repeat the process !
- **CycleGAN** (Zhu et al. 2017)



$(F = G^{-1})$

Text2Image with CLIP

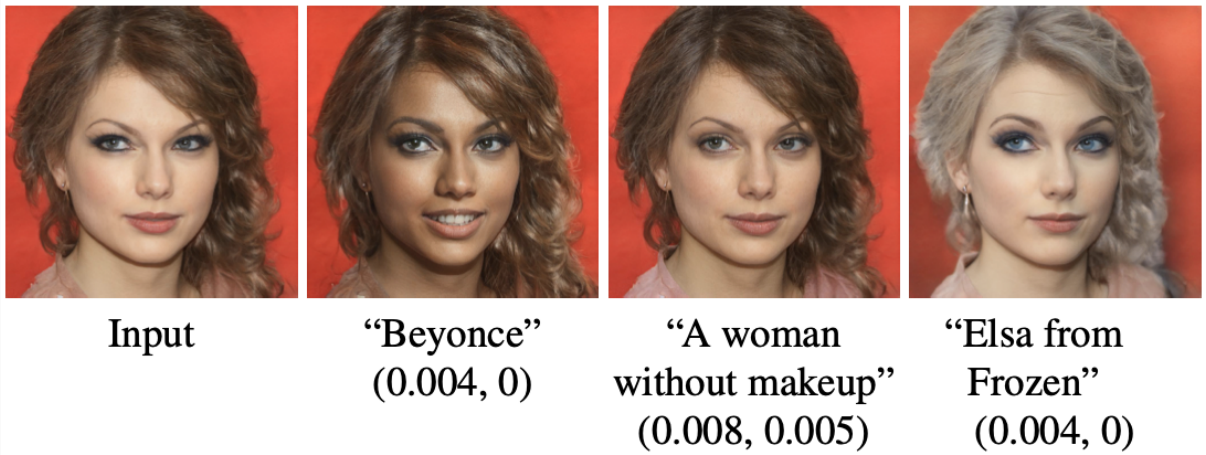
CLIP

1. CLIP jointly trains an image encoder and text encoder.
2. Trained on 400 million (image, text).
3. The training objective: given a batch of N (image, text) pairs, predicting which of the N × N possible (image, text) pairings across a batch actually occurred.

StyleCLIP: combines StyleGAN2 and CLIP:

1. latent optimization in $\mathcal{W}^+$.
2. build a latent mapper trained on one text prompt.
3. build a global mapper that takes as input a text prompt and outputs a direction in $\mathcal{S}$.

Analysis of method 1: latent optimization



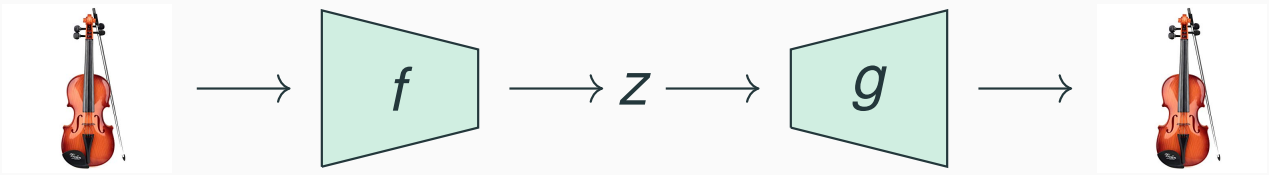
Downsides:

- sensitive to parameters
- requires a few minutes of optimization for every generation

Variational Autoencoders

Autoencoders

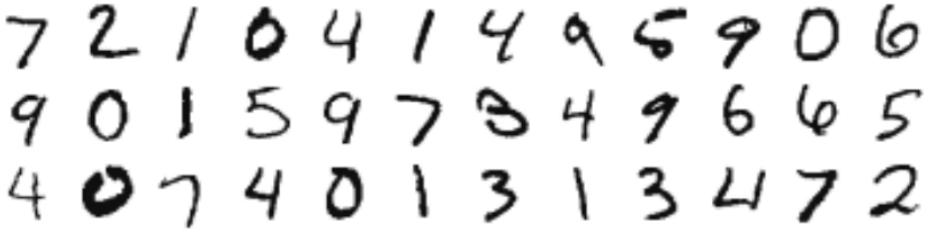
- Main idea: force a self-supervised network to compress the original representation in a low-dimensional latent space.



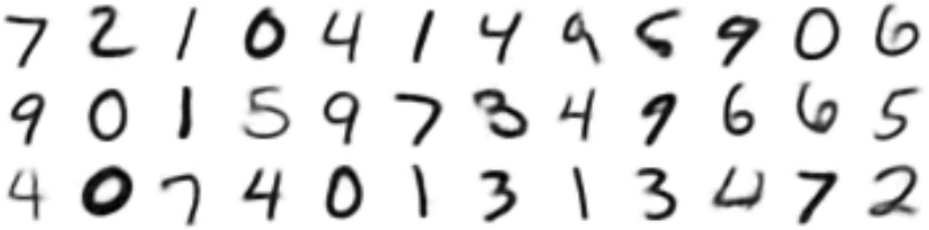
- The goal is to learn an encoder f and a decoder g such that $g \circ f$ is close to identity.
- If f and g are linear, the optimal solution is given by a PCA
- Otherwise, we can achieve better performance with deep networks

Deep Autoencoders

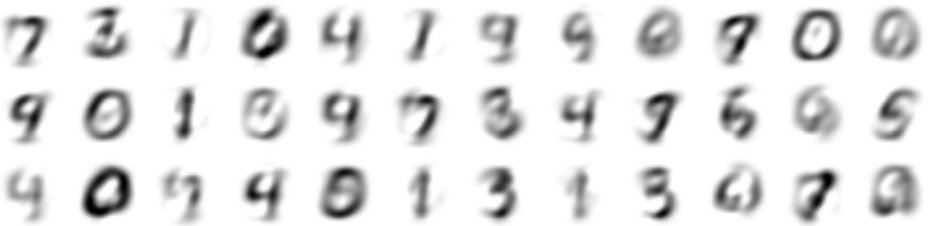
X (original samples)



$g \circ f(X)$ (CNN, $d = 8$)



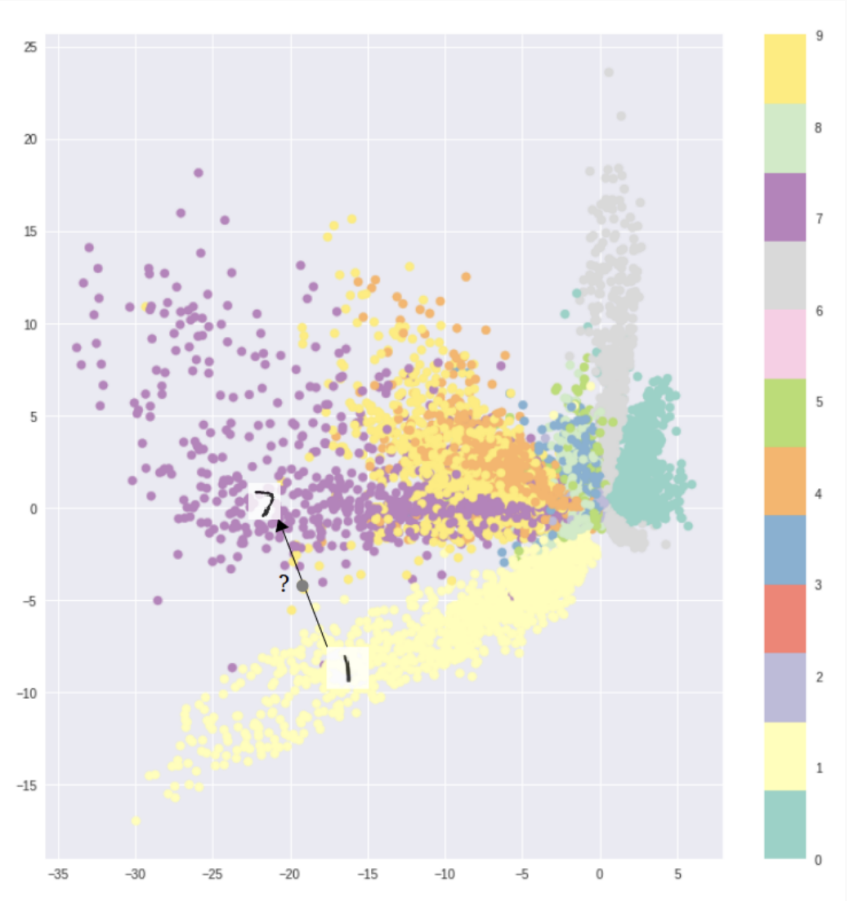
$g \circ f(X)$ (PCA, $d = 8$)



(by courtesy of François Fleuret)

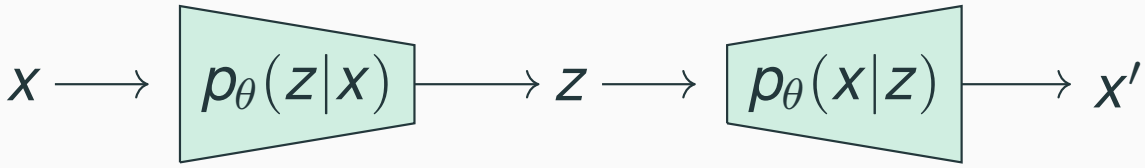
How to sample from autoencoders ?

- Simple answer: sample z in the latent space and feed it into the decoder
- However it is very likely that the encoded inputs lies in a low-dimensional manifold inside the latent space



VAE in a nutshell

- **Let us constraint the latent variable z to follow a fixed distribution from which we can sample easily**
- Let's rewrite everything with probabilities !



- $p_\theta(z|x)$ is intractable since we do not know the distribution of the true data so we approximate it by the variational distribution $q_\phi(z|x)$ that should minimize

$$\mathbb{D}_{KL}(q_\phi(z|x), p_\theta(z|x)) \quad .$$

VAE in a nutshell (cted)

Lemma

For any variational distribution q_ϕ, the (true) marginal log-likelihood $log(p_\theta(x))$ can be written as

$$\mathbb{D}_{KL}(q_\phi(z|x), p_\theta(z|x)) + \mathcal{L}_{\theta,\phi} \quad .$$

Note that:

- $\mathcal{L}_{\theta,\phi}$ is called the **variational lower bound** since $log(p_\theta(x)) \geqslant \mathcal{L}_{\theta,\phi}$
- For a fixed θ, minimizing the KL-divergence wrt ϕ is similar to **maximize** $\mathcal{L}_{\theta,\phi}$.
- For a fixed ϕ, maximizing $\mathcal{L}_{\theta,\phi}$ wrt θ, maximizes the expected log-likelihood of the data.

VAE in a nutshell

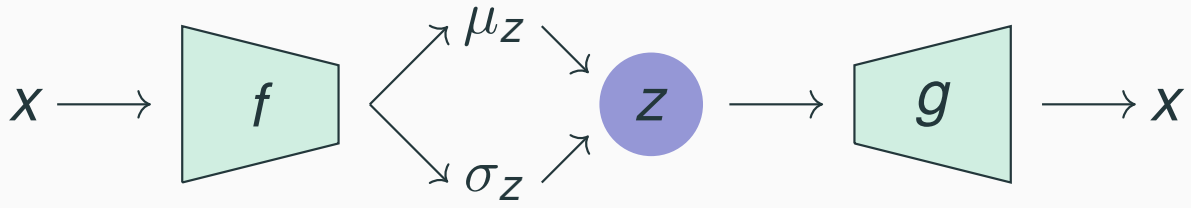
- Let's summarize ! The loss function to minimize is $-\mathcal{L}_{\theta,\phi}$ and can be rewritten as

$$-\mathbb{E}_{z\sim q_\phi(z|x)}\left[log(p_\theta(x|z))\right] + \mathbb{D}_{KL}(q_\phi(z|x)|p_\theta(z)) \quad .$$

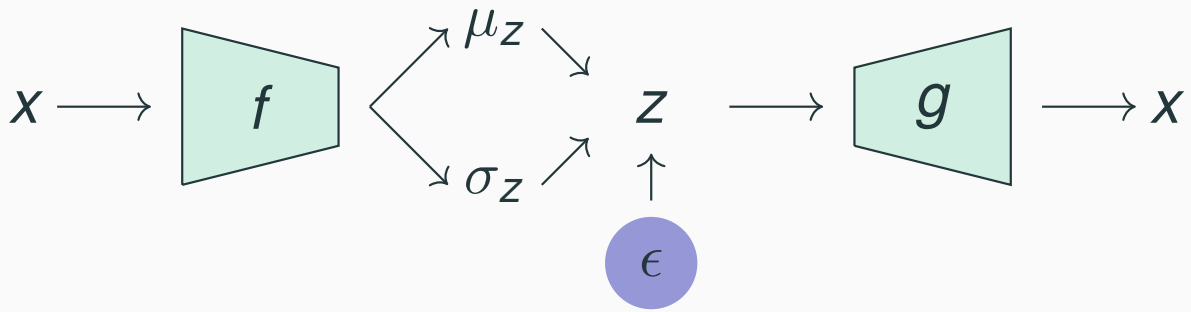
- The first term is called the reconstruction loss.
- The second term can be seen as a regularizer toward the prior distribution of the latent variable p_θ

One last problem ! How to backprop ?

- **Problem:** Impossible to backpropagate through a **stochastic node** like z



- **Solution (ex. for a Gaussian posterior):** Let's write $z = \mu_z + \sigma_z \odot \epsilon$ with $\epsilon \sim \mathcal{N}(0, 1)$ to have a differentiable path end-to-end.



Reparametrization trick

VAE vs GANs

	VAE	GAN
Modules	Encoder + Decoder	Generator + Discriminator
Training ?	Reconstruction Loss + Latent Loss	Minimax game
Stability ?	Closed-form	Need to reach a Nash equilibrium
Quality ?	Good but blurry images	High quality sharp images

Vector Quantized GANs

Vector-quantized latent space Esser et al. 2021

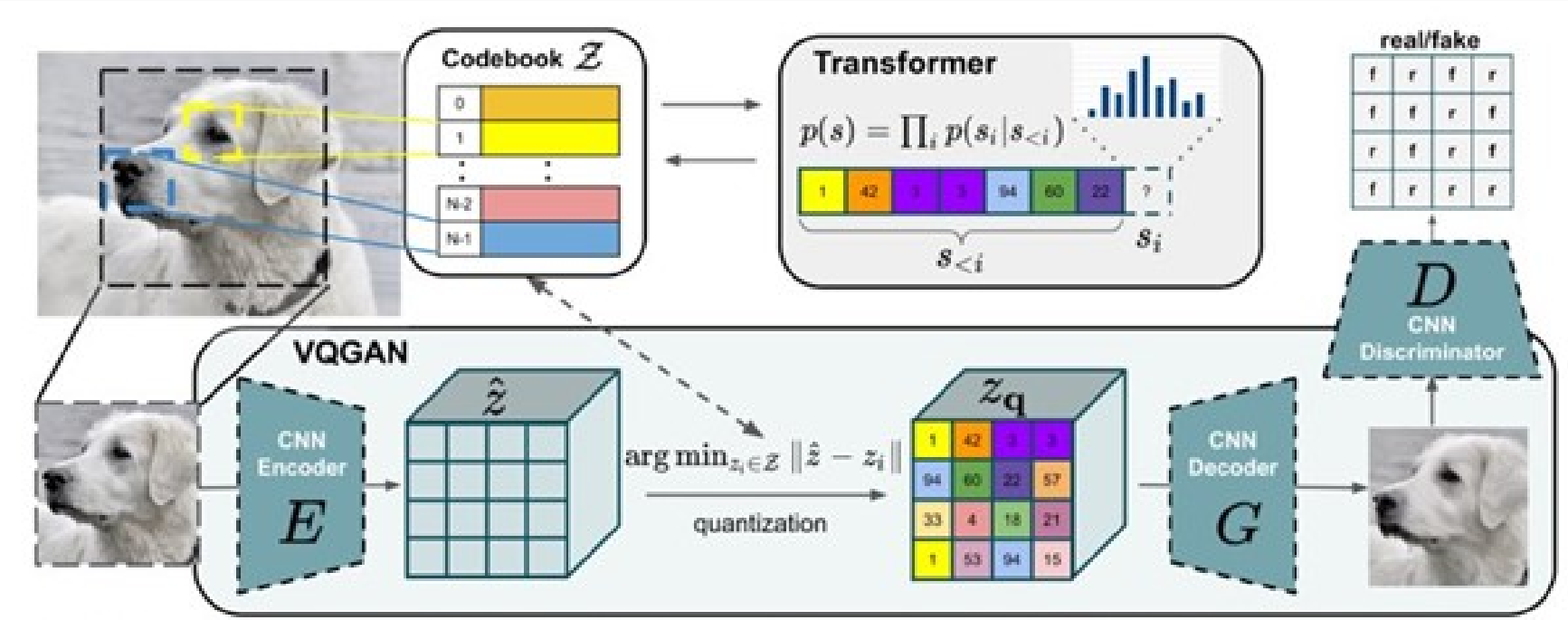


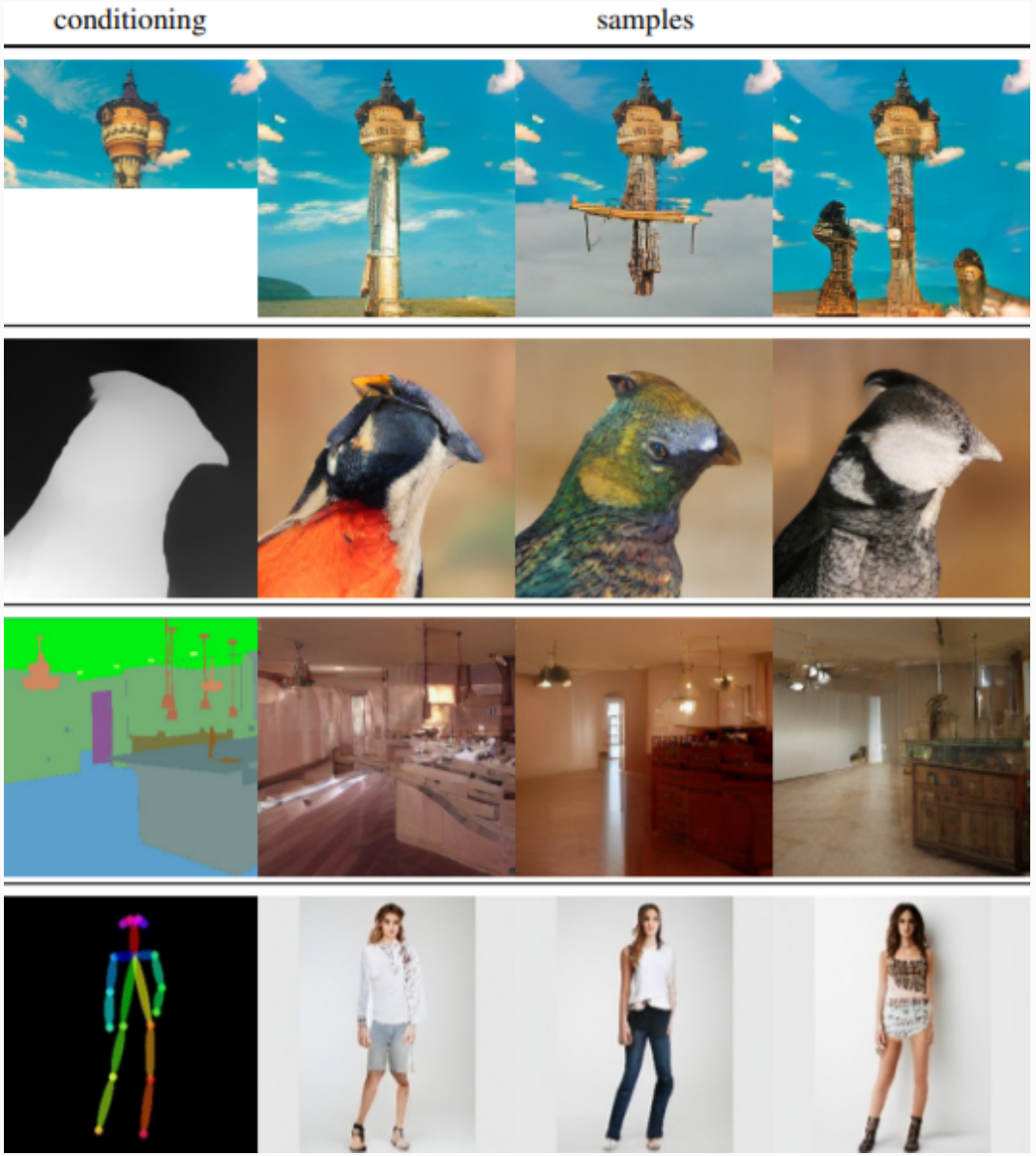
Figure 2. Our approach uses a convolutional *VQGAN* to learn a codebook of context-rich visual parts, whose composition is subsequently modeled with an autoregressive transformer architecture. A discrete codebook provides the interface between these architectures and a patch-based discriminator enables strong compression while retaining high perceptual quality. This method introduces the efficiency of convolutional approaches to transformer based high resolution image synthesis.

Figure 6

NLP and Vision are now two sides of the same c

1. VQGAN aims at training a triplet (E, D, C).
2. For any dataset of images $\mathcal{D}$, one can create a dataset of sequences $\mathcal{D}_S$.
3. One can train any language model on $\mathcal{D}_S$ (Transformers, RNN, etc...), to be able to generate likely sequences.
4. After that, use the decoder to decode them into images.

Generating images with Transformers



Properties of this vector-quantized latent space

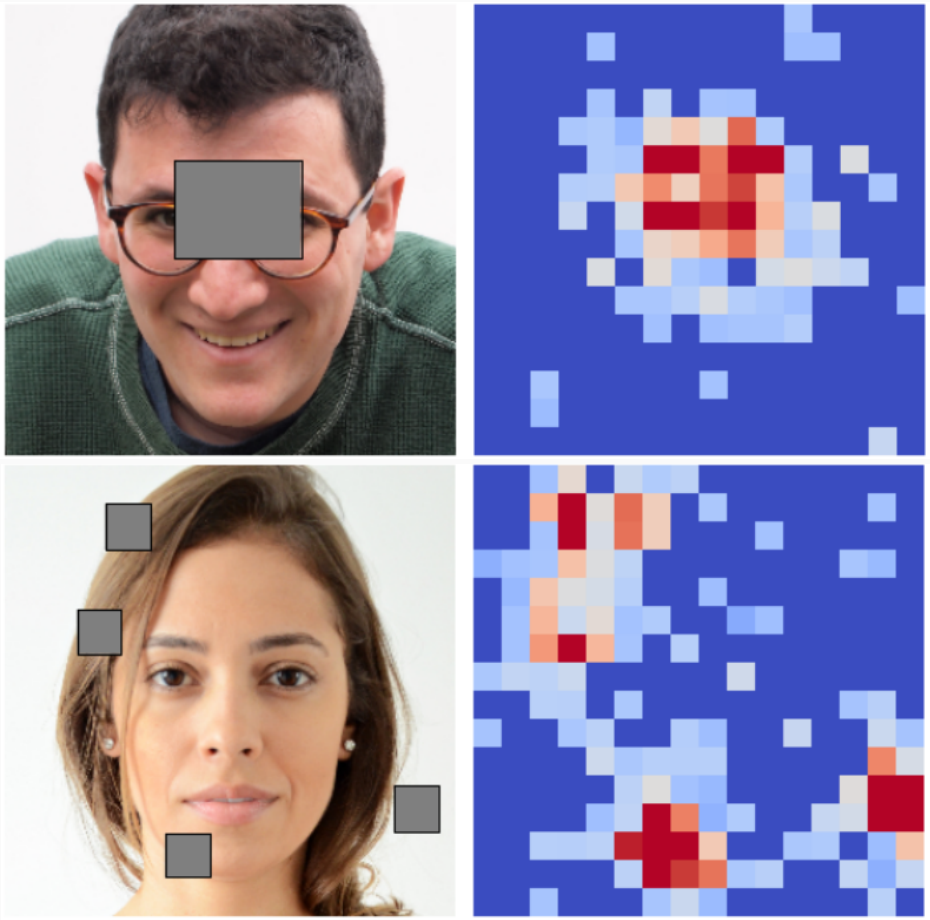
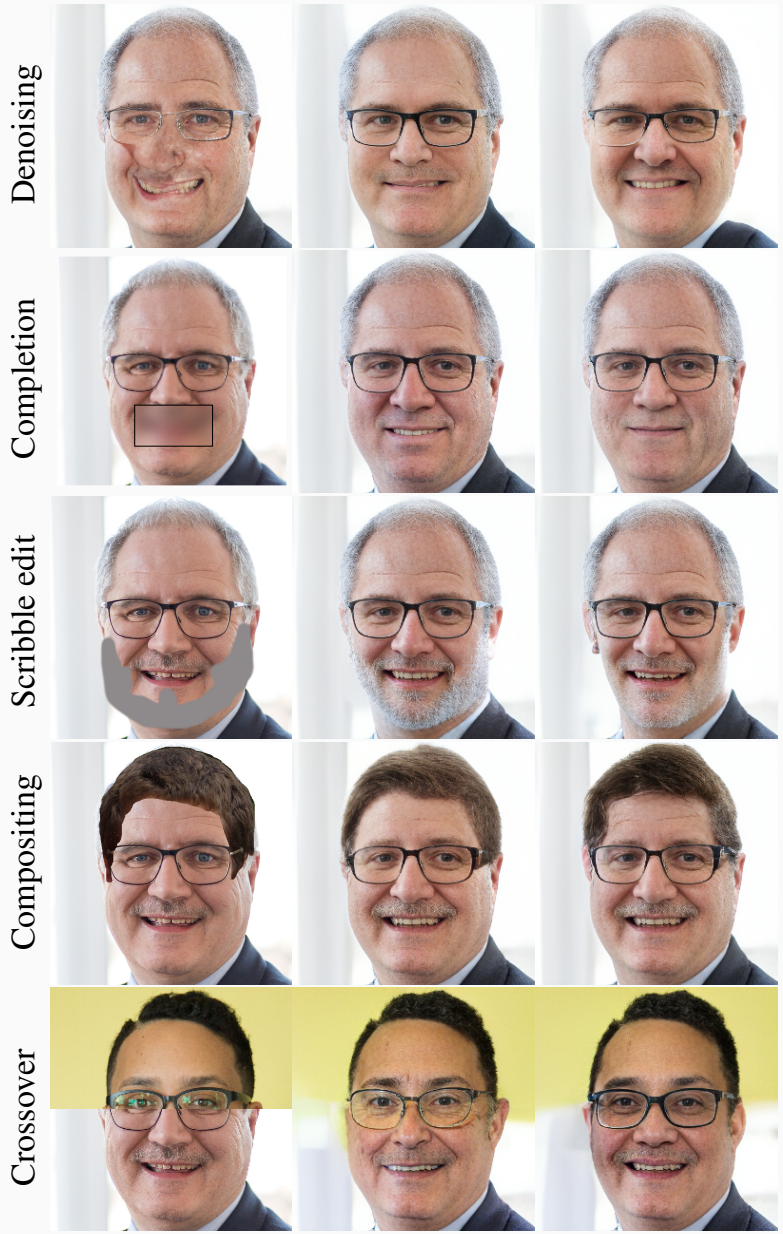


Figure 7: Each VQGAN token is strongly tied to a small spatial area in the image space. Perturbed images lead to variations of tokens in the latent space.

Properties of this vector-quantized latent space

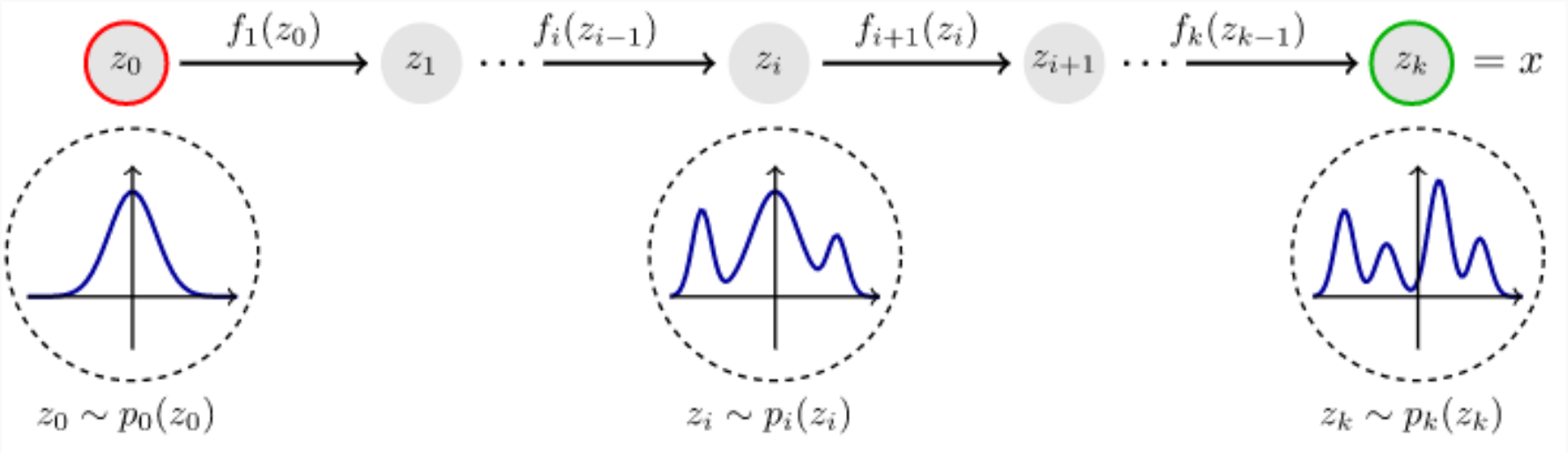
Figure 8: Each VQGAN token is strongly tied to a small spatial area in the image space. Collages of images can easily be done with collages of latent representations.

Image manipulation with VQGANs



Normalizing Flows

Normalizing Flows



Normalizing flows vs GANs

Setting

- Let $\mathbb{R}^d$ be the latent space with latent variable Z.
- Let $\mathcal{G} = \{G_\theta, \theta \in \Theta\}$ be a class of invertible functions.

Pros

- Simpler architecture & simpler loss: likelihood.
- Less prone to mode collapse. Especially, when compared to cGANs (known to be nearly deterministic).
- Super Resolution image generation Lugmayr et al. 2020.

Cons

- The input and output dimensions must be the same.
- The transformation must be invertible.

Glow: an efficient Normalizing flow architecture

kingma2018glow

References

References

- Abdal, Rameen et al. (2019). “Image2stylegan: How to embed images into the stylegan latent space?” In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4432–4441.
- Arjovsky, Martin et al. (2017). “Wasserstein gan”. In: arXiv preprint arXiv:1701.07875.
- Arora, Sanjeev et al. (2017). “Generalization and Equilibrium in Generative Adversarial Nets (GANs)”. In: CoRR abs/1703.00573. arXiv: 1703.00573. URL: http://arxiv.org/abs/1703.00573.

References ii

- Esser, Patrick et al. (2021). “Taming transformers for high-resolution image synthesis”. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Reco pp. 12873–12883.
- Goodfellow, Ian et al. (2014). “Generative adversarial nets”. In: Advances in neural information processing systems, pp. 2672–2680.
- Gulrajani, Ishaan et al. (2017). “Improved training of wasserstein gans”. In: Advances in Neural Information Processing Systems, pp. 5767–5777.
- Isola, Phillip et al. (2017). “Image-to-image translation with conditional adversarial networks”. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1125–1134.
- Issenhuth, Thibaut et al. (2021). “EdiBERT, a generative model for image editing”. In: arXiv preprint arXiv:2111.15264.

References iii

- Liu, S. et al. (2017). “Approximation and convergence properties of generative adversarial learning”. In: Advances in Neural Information Processing Systems 30. Ed. by I. Guyon et al. Red Hook: Curran Associates, Inc., pp. 5551–5559.
- Lugmayr, Andreas et al. (2020). “Srflow: Learning the super-resolution space with normalizing flow”. In: European Conference on Computer Vision. Springer, pp. 715–732.
- Mao, X. et al. (2017). “Least Squares Generative Adversarial Networks”. In: IEEE International Conference on Computer Vision.
- Mirza, Mehdi and Simon Osindero (2014). “Conditional Generative Adversarial Nets”. In: CoRR abs/1411.1784. arXiv: 1411.1784. URL: http://arxiv.org/abs/1411.1784.
- Miyato, Takeru et al. (2018). “Spectral Normalization for Generative Adversarial Networks”. In: International Conference on Learning Representations.

References iv

- Nowozin, Sebastian et al. (2016). “f-gan: Training generative neural samplers using variational divergence minimization”. In: Advances in neural information processing systems, pp. 271–279.
- Samangouei, Pouya et al. (2018). “Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models”. In: International Conference on Learning Representations.
- Xiao, Chaowei et al. (2018). “Generating adversarial examples with adversarial networks”. In: arXiv preprint arXiv:1801.02610.
- Zhu, Jun-Yan et al. (2017). “Unpaired image-to-image translation using cycle-consistent adversarial networks”. In: Proceedings of the IEEE international conference on computer vision, pp. 2223–2232.