Hands-on Introduction to Deep Learning with PyTorch

Generative Models (for Images)

Rocco Meli and Rafael Sarmiento
ETHZürich / CSCS
Lugano, February 28th - March 1st 2024

ETH zürich
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Disclaimers

Overview of *some* architectures for generative modelling (mainly for images).

- We are going to skip over a lot of maths
- Many architectures discussed here deserve their own lecture
- We are mainly trying to build some intuition and learn jargon

You can read the original papers (and official implementations) if you are interested in the nitty gritty details.

Discriminative Model

Estimate

$$p(y|\mathbf{x})$$

Probability of label y given observation $\mathbf{x}$

Generative Models

Estimate

$$p(\mathbf{x})$$

Probability of observing an observation $\mathbf{x}$

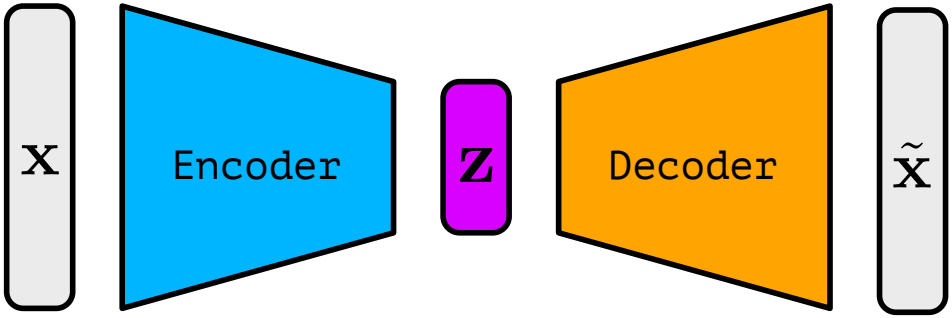
Conditional Generative Models

Estimate

$$p(\mathbf{x}|y)$$

Probability of observing an observation $\mathbf{x}$ with label y

Autoencoder



Autoencoder: Reconstruction Loss

An autoencoder is a neural network that is trained to copy its input to its output, through an information bottleneck.

$$\mathcal{L}_{\text{reconstruction}} = \mathcal{L}_2 = \sum_i ||x_i - \tilde{x}_i||^2$$

(Linear) Autoencoder and Principal Component Analysis (PCA)

The autoencoder model is tightly related with PCA: a single-layer linear autoencoder trained with $\mathcal{L}_2$ loss performs PCA.

Convolutional Autoencoder and Transposed 2D Convolution

- The convolutional encoder is trivial: same as CNNs
- How to go from a small latent space to generated images?
  - Easy with linear layers: transpose the weights matrix

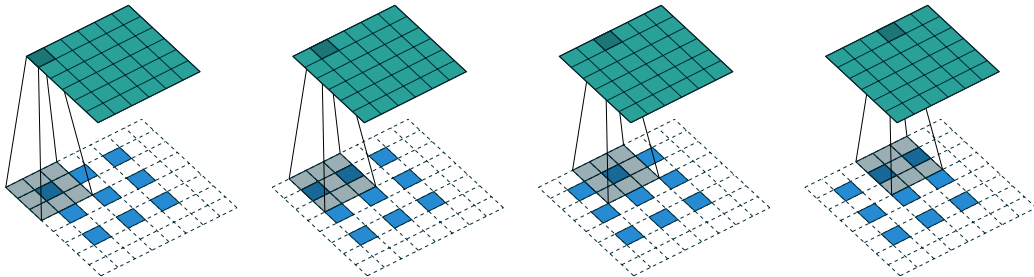
Upsampling

For the decoder, we need a way to upsample images:

- Nearest-neighbor
- Un-pooling
- Interpolation
- Transpose convolution

Convolutional Autoencoder and Transposed 2D Convolution

A transposed convolution can be interpreted as a standard convolution on dilated images.



Dumoulin, Vincent, and Francesco Visin. "A guide to convolution arithmetic for deep learning." arXiv preprint arXiv:1603.07285 (2016)

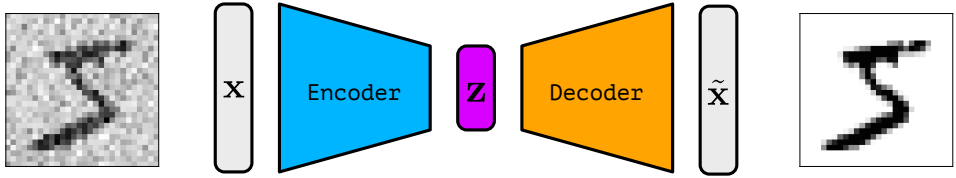
Problems with Autoencoder for Image Generation

We can sample points in latent space and use the decoder to convert them back in pixel space.

- Different classes have different distributions in latent space
- Unclear how to sample the latent space (which distribution?)
- A random point in latent space does not always decode into a meaningful image
  - No guarantee that the latent space is continuous

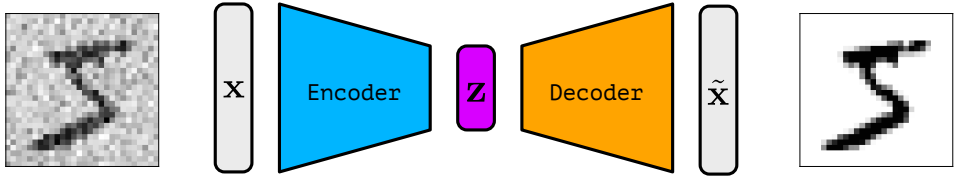
Autoencoder for De-Noising

Use noisy images as input, compute the loss with original images.



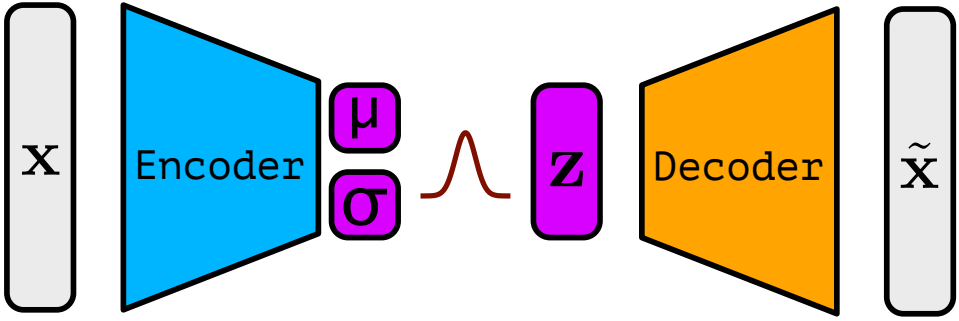
Autoencoder for De-Noising

Use noisy images as input, compute the loss with original images.



If you finish the exercises quickly, try to implement a de-nosing autoencoder!

Variational Autoencoder



The encoder computes $q_\phi(z|x)$, the decoder computes $p_\theta(x|z)$.

Variational Autoencoder

- The encoder produces the mean and the variance of a Gaussian distribution.
  - In practice produce $\log(\sigma^2)$, since variance is always positive
- The decoder produces similar images for similar points in latent space

Kullback-Leibler divergence

Given two probability distributions P and Q,

$$\mathcal{D}_{\mathrm{KL}}(P\,||\,Q) = \int_{\mathbf{x}} p(\mathbf{x}) \log \left(\frac{p(\mathbf{x})}{q(\mathbf{x})} \right) \, d\mathbf{x}$$

The KL divergence measures how P differs from the reference distribution Q

Variational Autoencoder: Loss Function

The VAE loss has two components:

$$\mathcal{L}_{\text{VAE}} = \mathcal{L}_{\text{reconstruction}} + \beta \mathcal{L}_{\text{KL}}$$

The reconstruction loss is the same as the one for an autoencoder:

$$\mathcal{L}_{\text{reconstruction}} = \sum_i ||x_i - f_\theta(z_i)||^2$$

The KL loss acts as regularization, forcing the latent space distribution to follow a Gaussian distribution:

$$\mathcal{L}_{\text{KL}} = -\mathcal{D}_{\text{KL}}(q_\phi(z_i|x_i)||p_\theta(z_i)) = -\frac{1}{2}\sum_i \left(1 + \log \sigma_i^2 - \mu_j^2 - \sigma_j^2\right)$$

Variational Autoencoder: Reparametrisation Trick

Sampling from a Gaussian (normal) distribution is problematic for backpropagation:

$$\mathbf{z} \sim \mathcal{N}\left(\mu, \sigma^2\right)$$

Gradients can't be backpropagated through the sampling layer.

Variational Autoencoder: Reparametrisation Trick

Sampling from a Gaussian (normal) distribution is problematic for backpropagation:

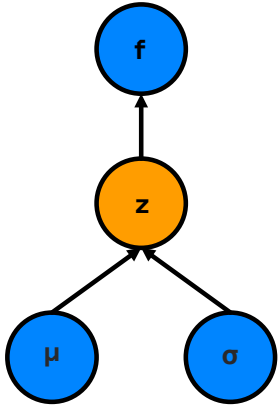
$$\mathbf{z} \sim \mathcal{N}\left(\mu, \sigma^2\right)$$

Gradients can't be backpropagated through the sampling layer.

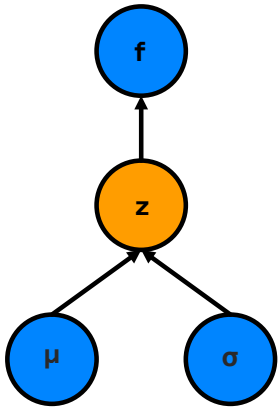
Reparametrization trick: use the properties of Gaussian distribution:

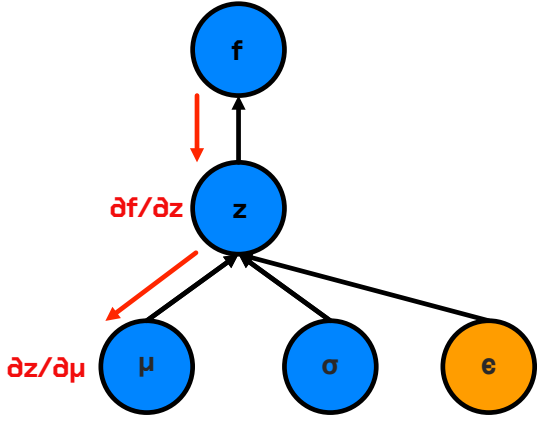
$$z = \mu + \sigma\epsilon \qquad \epsilon \sim \mathcal{N}(0, 1)$$

Variational Autoencoder: Reparametrization Trick

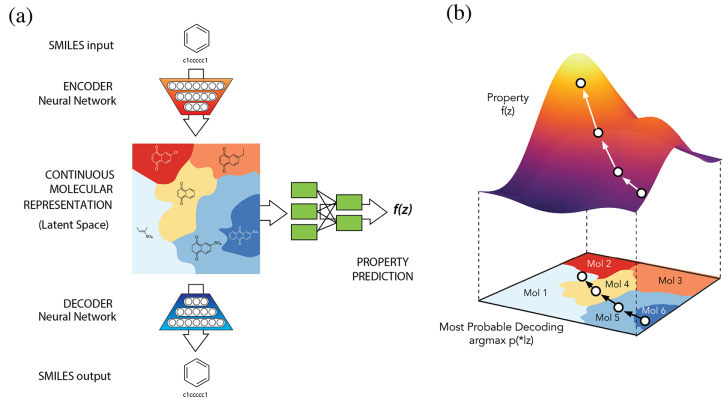


Variational Autoencoder: Reparametrization Trick



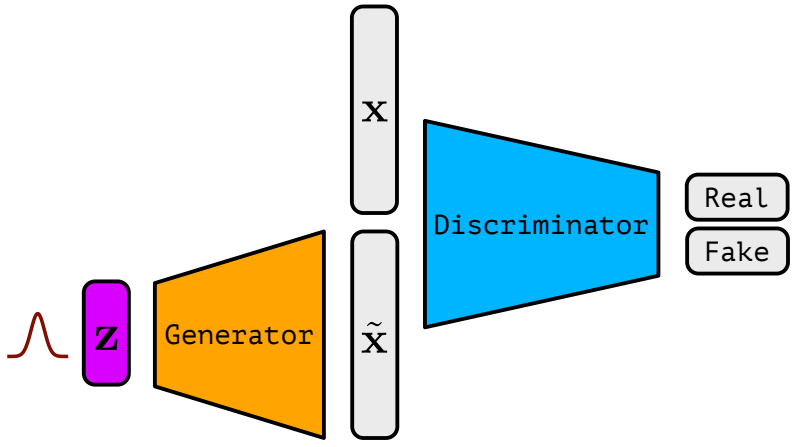


Automatic Chemical Design with VAEs



Gómez-Bombarelli, R., et al. "Automatic chemical design using a data-driven continuous representation of molecules." ACS central science 4.2 (2018): 268-276.

Generative Adversarial Model (GAN)



Training GANs: Training the Discriminator

Training the discriminator:

1. Get a batch of real images from the training set (label: 1)
2. Get a batch of fake images from the generator (label: 0)
3. Combine the two batches into a single one
4. Train the discriminator (one step) with binary cross-entropy loss

Training GANs: Training the Generator

Training the generator (discriminator is fixed):

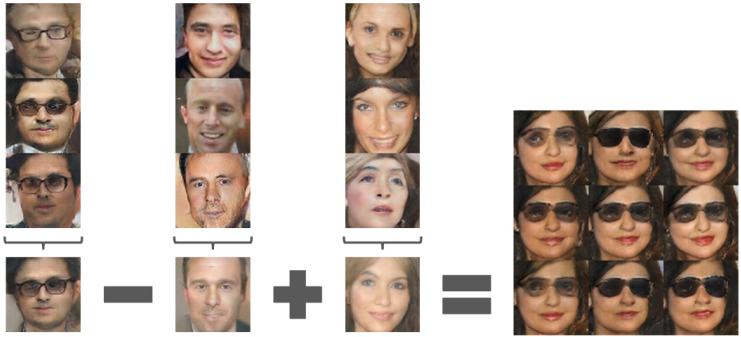
1. Get a batch of fake images from the generator
2. Set labels to 1
   - We pretend fake images are real!
3. Train the generator (one step) with binary cross-entropy loss
   - The generator weights are tweaked so that the images look more real

Challanges of Training GANs

- GANs can only reach a single Nash equilibrium
  - Generator produces realistic samples, discriminator guesses (50%/50%)
  - There is no guarantee this equilibrium will ever be reached
- Mode collapse
  - Generator output becomes less and less diverse
  - GAN cycles amongst few different classes
- Oscillating parameters leading to instabilities
- Very sensitive to hyper-parameters

Latent Space Representation (DCGAN)

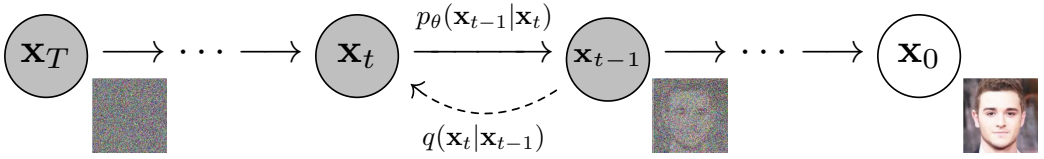
Learned representations are meaningful:



Radford, A., L. Metz, and S. Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).

Denoising Diffusion Probabilistic Model

Train a neural network to de-noise an image over a series of steps:



Forward process: add Gaussian noise to an image with $q(\mathbf{x}_t|\mathbf{x}_{t-1})$

Approximate the backward process $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ by $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ (de-noising)

Denoising Diffusion Probabilistic Model: Implementation

The implementation details of a denoising diffusion model are outside the scope of this course:

- (Modified) U-Net architecture
- Positional encodings (see tomorrow's lessons)
  - Take into account the time step
- Multi-head attention (see tomorrow's lessons)

After tomorrow, you will have most of the instruments to understand their implementation in PyTorch!

https://nn.labml.ai/diffusion/ddpm

Stable Diffusion: Text-to-Image Generative Model

A racoon with sunglasses sipping a Martini while coding a deep learning model in PyTorch on his laptop, sitting besides a crystal clear alpine lake surrounded by beautiful mountains. High quality digital art.

https://github.com/Stability-AI/generative-models

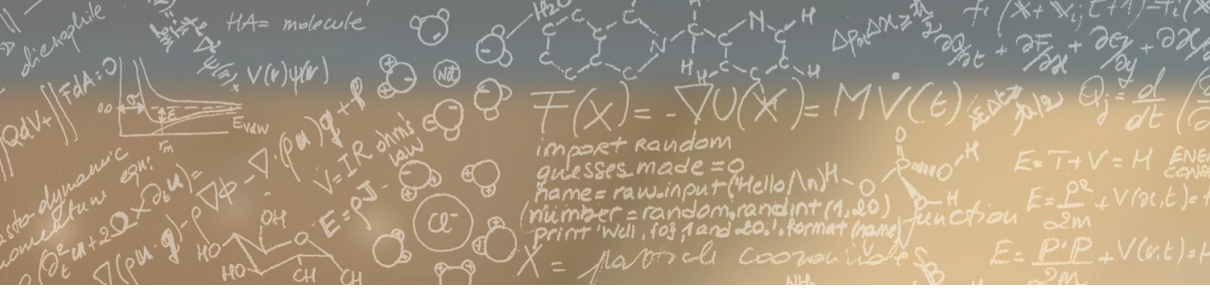
What have we missed?

A lot.

- Latent diffusion models
- Conditional generative models
- Normalizing flow models
- Energy-based models
- Text generation (see tomorrow's lectures)
- …

[lab] GANs and VAEs

- Train a variational autoencoder for the generation of MNIST handwritten digits
  - Simple feed-forward neural networks
- Train a generative adversarial network for the generation of FashionMNIST images
  - Deep convolutional GAN (DCGAN)



Thank you for your attention!

References

- Goodfellow, Ian, et al. ”Generative adversarial nets.” Advances in neural information processing systems 27 (2014).
- Radford, Alec, Luke Metz, and Soumith Chintala. ”Unsupervised representation learning with deep convolutional generative adversarial networks.” arXiv preprint arXiv:1511.06434 (2015).
- Ho, Jonathan, Ajay Jain, and Pieter Abbeel. ”Denoising diffusion probabilistic models.” Advances in neural information processing systems 33 (2020): 6840-6851.
- Nichol, Alexander Quinn, and Prafulla Dhariwal. ”Improved denoising diffusion probabilistic models.” International Conference on Machine Learning. PMLR, 2021.

Bayes' Theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Variational Autoencoder: Loss Function

The VAE loss can be written as follows:

$$\mathcal{L}_{\text{VAE}} = -\sum_i \left\{ \mathbb{E}_{z_i \sim q_\phi(z_i|x_i)} \left[\log p_\theta(x_i|z_i) \right] - \mathcal{D}_{\text{KL}}(q_\phi(z_i|x_i)||p_\theta(z_i)) \right\}$$

with

$$\begin{aligned} q_\phi(\mathbf{z}_i|\mathbf{x}_i) &\sim \mathcal{N}(\mu_{\text{encoder}}, \sigma^2_{\text{encoder}}) \\ p_\theta(x_i|z_i) &\sim \mathcal{N}(f_\theta(\mathbf{z}_i), \sigma^2_{\text{decoder}}) \\ p_\theta(z_i) &\sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \end{aligned}$$

DDPM: Forward Diffusion Process

Start with an image $\mathbf{x}_0$ and define the number of diffusion steps T.

Add small amount of Gaussian noise to the image:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) = \sqrt{1-\beta_t}\mathbf{x}_t + \sqrt{\beta_t}\epsilon_{t-1}$$

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

$$\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i \quad \alpha_i = 1 - \beta_t$$

Similar to VAE, but the forward process is un-parametrized (in VAE it is learned: $q_\phi(z|x)$)

DDPM: Diffusion Schedule

Every diffusion step can have a different β_t, according to a diffusion schedule.

- Linear (between $\beta_0 = 0.0001$ and $\beta_T = 0.02$)

$$\beta_t = \beta_0 + \frac{\beta_T - \beta_0}{T} t$$

- Cosine

$$\bar{\alpha}_t = \frac{f(t)}{f(0)} \quad f(t) = \cos\left(\frac{t}{T}\frac{\pi}{2}\right)^2 \quad \beta_t = 1 - \frac{\bar{\alpha}_t}{\alpha_{t-1}}$$

The diffusion schedule is such that $\bar{\alpha}_T \approx 0$ so that $q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$: diffused data $\mathbf{x_T}$ are normally distributed

DDPM: Reverse Diffusion Process

Approximate $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ by (learned) $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$. $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ can be approximated to be a normal distribution (if $\beta_t \ll 1$).

Train a neural network ϵ_θ to predict the noise that has been added to an image (simplified loss):

$$\mathcal{L} = ||\epsilon - \epsilon_\theta(\mathbf{x}_t, t)||^2 = ||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)||^2$$

DDPM: Denoising Diffusion Probabilistic Model: Training

1. Take a sample from the data distribution $q(\mathbf{x}_0)$ (training set)
2. Sample a noise level t uniformly on $[1, T]$
3. Generate some Gaussian noise ϵ
4. Compute the noisy sample $\mathbf{x}_t$ (based on β_t and ϵ)
5. Gradient descent step with $\nabla_\theta ||\epsilon - \epsilon_\theta(\mathbf{x}_t, t)||^2$

DDPM: Sampling

Generate random noise $\mathbf{x}_T \sim \mathcal{N}(0, 1)$

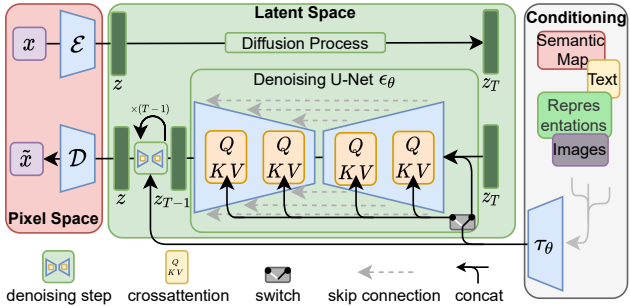
Repeat the following from T to 1:

1. Sample $\mathbf{z} \sim \mathcal{N}(0, 1)$
2. Generate the next de-noised image as

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$$

$\mathbf{x}_0$ is the generated image (de-noised)

Stable Diffusion



Rombach, Robin, et al. "High-resolution image synthesis with latent diffusion models." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022.