Machine Learning (CE 40717)

Fall 2024

Ali Sharifi-Zarchi

CE Department
Sharif University of Technology

November 26, 2024

Unlabeled Data Usage in Deep Learning

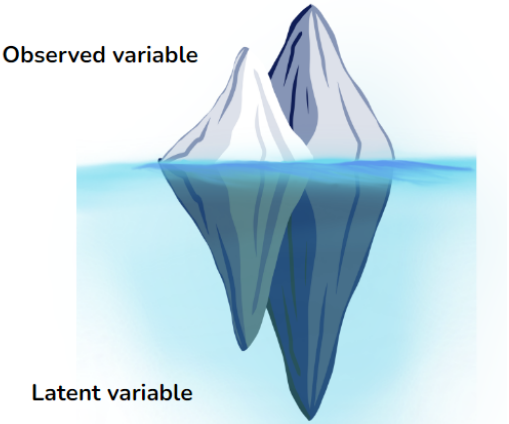
How can we use unlabeled data in our deep neural networks?

Image: Source

What Is Latent Variable?

- **Definition:** Latent variables are variables that are not directly observed but are inferred from other variables that are observed (measured).
- **Origin:** Derived from the Latin word *latēre* meaning "hidden".
- **Examples:**
  - Personality traits (in psychology)
  - Economic factors like inflation (in economics)
  - Topic of a document (in natural language processing)

What Is Latent Variable?



Latent Variables in Deep Learning

We can train deep neural networks using unlabeled data to infer latent variables in order to:

- Compress data
- Generate new data
- Denoise Data
- Using latent variables as features for various tasks like classification
- ...

Autoencoder: Background

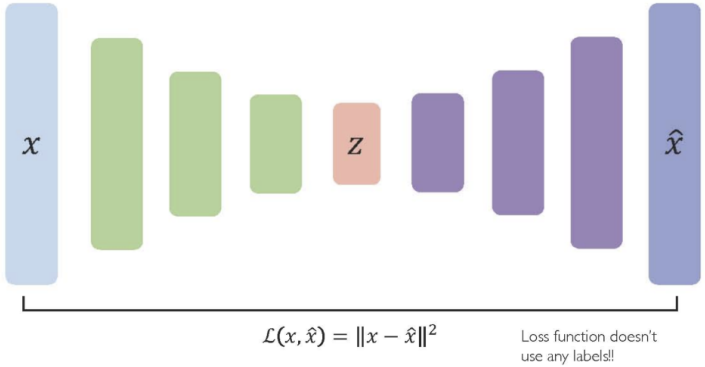
Using latent variables as a foundation, an unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data.

Why do we care about a low-dimensional z?

"Encoder" learns mapping from the data, x, to a low-dimensional latent space, z

Autoencoders: Background

How can we learn this latent space?
Train the model to use these features to **reconstruct the desired (usually original) data**.

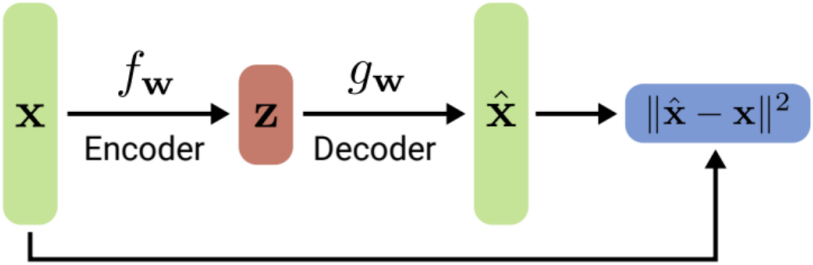


"Decoder" learns mapping back from latent z, to a reconstructed observation, x

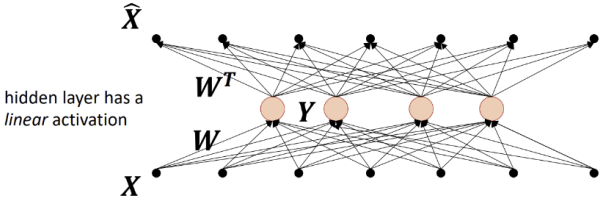
Autoencoders

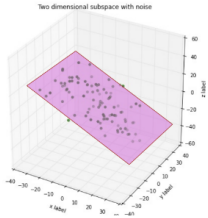
1. These models map between **observation space** $\mathbf{x} \in \mathbb{R}^D$ and **latent space** $\mathbf{z} \in \mathbb{R}^Q$:
   - **Encoder** $f_w : \mathbf{x} \mapsto \mathbf{z}$ The "Analysis" net which computes the hidden representation
   - **Decoder** $g_w : \mathbf{z} \mapsto \mathbf{x}$ The "Synthesis" which recomposes the data from the hidden representation

2. Models are linear or non-linear, deterministic or stochastic, with/without encoder.



Autoencoder Without a Non-linearity



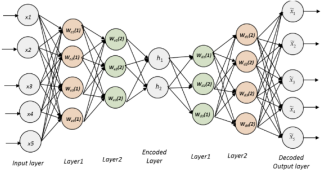


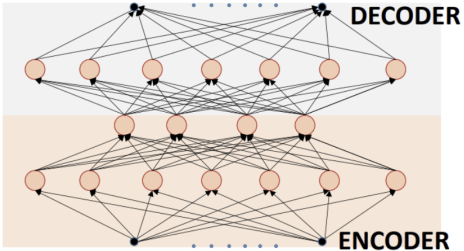
$\mathbf{Y} = \mathbf{WX}$ $\quad$ $\hat{\mathbf{X}} = W^T\mathbf{Y}$ $\quad$ $E = \|\mathbf{X} - W^T W\mathbf{X}\|^2$ $\quad$ Find W to minimize Avg$[E]$

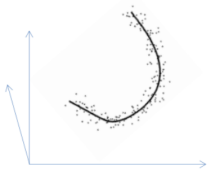
- **This is similar to PCA**
  - The output of the hidden layer will be in the principal subspace
    - Even if the recomposition weights are different from the "analysis" weights

"Deep" AE

- **Deep nonlinear autoencoders** learn to project the data, not onto a subspace, but onto a nonlinear manifold.
- This manifold is the **image of the decoder**.
- This is a kind of nonlinear dimensionality reduction.

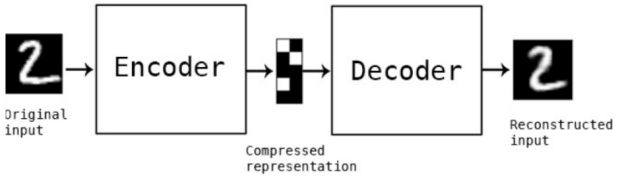






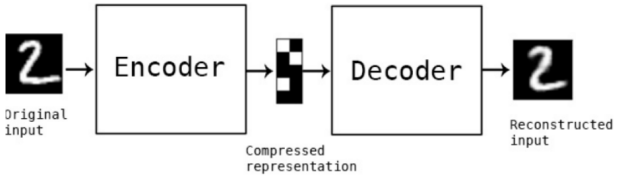
AE vs PCA

As we said, **PCA** can be described as



$$\min_W \sum \|\hat{x} - x\|^2$$
$$W^T W = I$$
$$\min_W \sum \|W^T W x - x\|^2$$

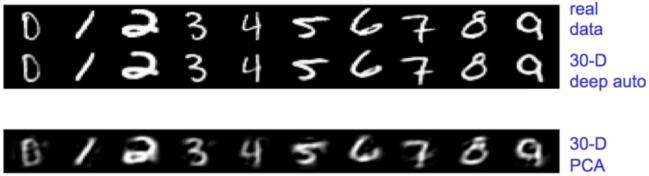
And also **Autoencoders** can be thought of as a **non-linear PCA**.

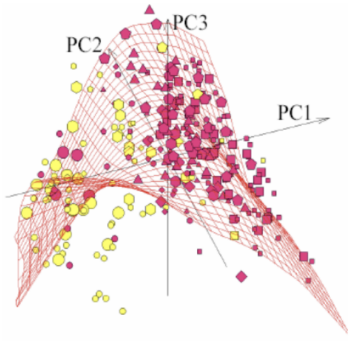


$$\min_{h,g} \sum \|\hat{x} - x\|^2$$
$$\min_{h,g} \sum \|g(f(x)) - x\|^2$$

AE vs PCA

Nonlinear autoencoders can learn more powerful codes for a given dimensionality, compared with linear autoencoders (PCA).





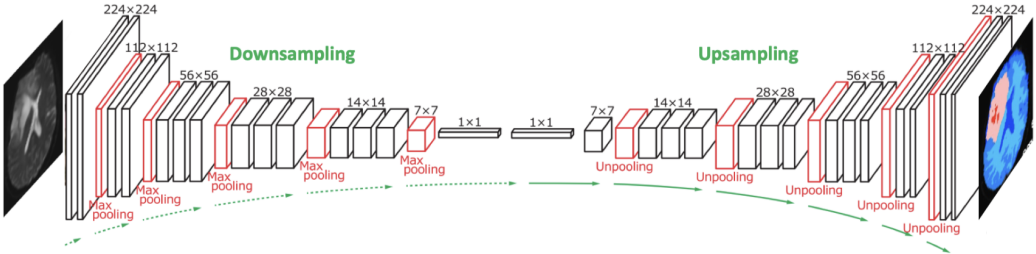
Autoencoder Key Characteristics

- **Specialized for Specific Data Type**
  - Autoencoders excel at recognizing patterns in the type of data they were trained on, but may fail on different data types.
  - Example: A model trained on animal images will not perform well when applied to landscapes.

- **Tailored Compression vs. General Algorithms**
  - Unlike generic compression techniques (MP3, JPEG), which apply universal rules, autoencoders customize their approach to fit the training data structure.

- **Loss of Detail in Output**
  - Autoencoders don't retain all the original information, leading to some degradation in the reconstructed data.
  - This makes them "lossy" – similar to formats like MP3 or JPEG, which prioritize size over perfect fidelity.

Fully Convolutional AE

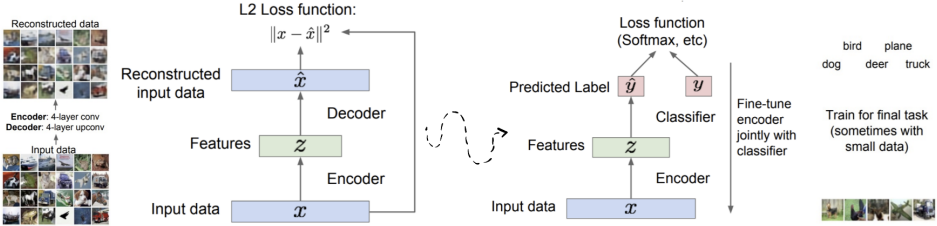


We will discuss more about **"Downsampling"** and **"Upsampling"** in the next section

Image from this link

Autoencoders As Pretrained Models

- **Feature Extraction:** Encoder **learns** data patterns and **compresses** them into useful features.
- **Supervised Model Initialization:** Use the **pretrained encoder** to initialize a supervised learning model.
- **Fine-Tuning:** Train the encoder and classifier **together** to improve performance.
- **Useful with Small Data:** Effective when training on **small datasets** by leveraging pretrained features.



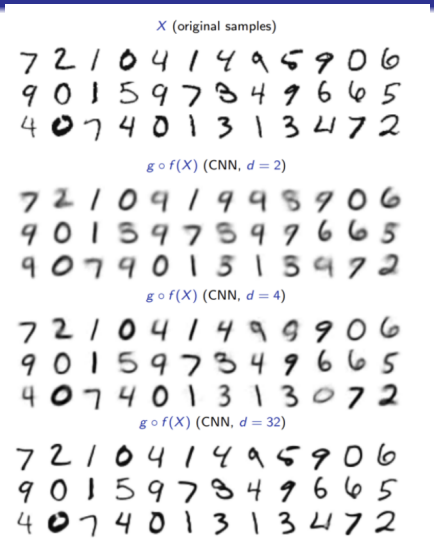
AE Performance & Bottleneck Size (MNIST Example)

- **Reconstruction Error and Bottleneck Size:**
  - A good autoencoder should minimize reconstruction error.
  - The error depends heavily on the size of the z (the dimension of the latent space).
- **Experiment Setup:**
  - CNN-based AE trained on MNIST with different bottleneck sizes:
    - $(2, 1, 1)$, $(4, 1, 1)$ & $(32, 1, 1)$

AE Performance & Bottleneck Size (MNIST Example) - Results

- **Results:**
  - With $d = 2$: Reconstruction is still good; a classifier could identify digits decently.
  - With $d = 32$: Reconstruction is nearly perfect; despite a compression ratio of ~ 0.04 (from 32 to 28x28).

- **Larger latent spaces** generally lead to **better reconstructions**, as more variables allow more information to be preserved during compression.



AE's: Undercomplete Autoencoders

1. An autoencoder whose code dimension is less than the input dimension is called undercomplete.
2. Learning an undercomplete representation forces the autoencoder to capture the most salient features of the training data.
3. The learning process is described simply as minimizing a loss function

$$L(x, g(f(x)))$$

where L is a loss function penalizing $g(f(x))$ for being dissimilar from x, such as the mean squared error.

AE's: Overcomplete Autoencoders

1. consider encoder f and decoder g.

$$X \in \mathbb{R}^d \quad h \in \mathbb{R}^k$$

2. When $k > d$, the autoencoder is called Overcomplete Autoencoders.

3. There are other ways we can constrain the reconstruction of an autoencoder than to impose a hidden layer of smaller dimension than the input.

4. Regularized Autoencoders use a loss function that encourages the model to have some properties besides reproducing inputs.
   - Sparsity representation (Sparse Autoencoders)
   - Smallness of derivative of representation (Contractive Autoencoders)
   - Robustness to noise or to missing inputs (Denoising Autoencoders)

AE's: Sparse Autoencoders (Intuition)

Idea: Can we describe the input with a small set of *"attributes"*?
This might be a more *compressed* and *structured* representation.

1. NOT structured - *"dense": most values non-zero*

```
Pixel (0,0):  #FE057D
Pixel (0,1):  #FD0263
Pixel (0,2):  #E1065F
```

Idea: "Sparse" representations are going to be more structured!

2. Very structured! - *"sparse": most values are zero*

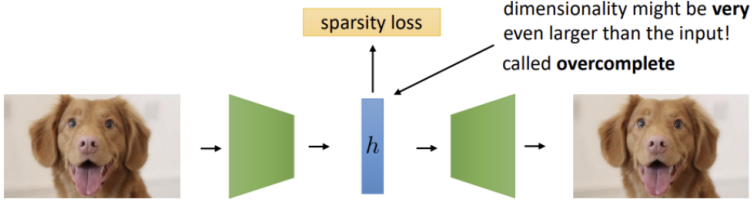
```
has_ears:  1
has_wings:  0
has_wheels:  0
```

There are many possible "attributes,"& most images don't have most of the attributes.

Aside:

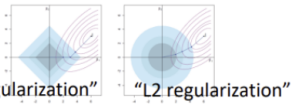
This idea originated in neuroscience, where researchers believe that the brain uses **sparse** representations (see "sparse coding").

AE's: Sparese Autoencoder (Big Picture)



ℓ_1 norm of h
gradient is always $\text{sgn}(h_j)$

simple sparsity loss: $\sum_{j=1}^{D} |h_j|$



There are other kinds of sparsity losses/models:

- Lifetime sparsity
- Spike and slab models

AE's: Sparse Autoencoders (Details)

1. Sparse Autoencoders try to minimize the following function.

$$L(x, g(f(x))) + \Omega(h)$$

2. The first term is loss for copying inputs.
3. The second term is sparsity penalty.
4. In general neural networks, we are trying to find the **maximum likelihood:** $p(x|\theta)$.
5. We often use $\log p(x|\theta)$ for simplification, from which we can get the loss function without regularization.
6. What about MAP (Maximum a posteriori)?

$$p(\theta|x) \propto p(x|\theta) \times p(\theta)$$

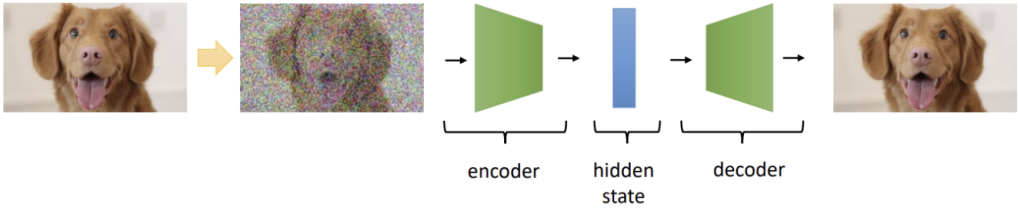
7. Maximizing the log of the above function yields:

$$\text{maximize } (\log p(x|\theta) + \log p(\theta))$$

8. The first term is the loss function, and the second term is the regularization penalty.

AE's: Denoising Autoencoders (Intuition)

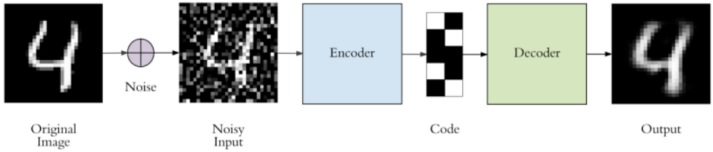
Idea: a good model that has learned meaningful structure should "fill in the blanks"



There are many variants on this basic idea, and this is one of the most widely used simple autoencoder designs.

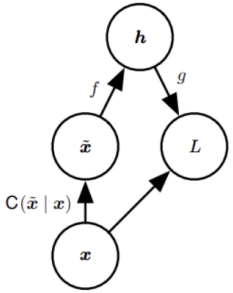
AE's: Denoising Autoencoders (Details)

1. **The denoising autoencoder (DAE)** is an autoencoder that receives a corrupted data point as input and is trained to predict the original, uncorrupted data point as its output.
2. **Traditional autoencoders minimize** $L(x, g(f(x)))$
   - L is a loss function penalizing $g(f(x))$ for being dissimilar from x, such as L_2 norm of difference: mean squared error.
3. **A DAE minimizes** $L(x, g(f(\tilde{x})))$
   - $\tilde{x}$ is a copy of x that is corrupted by some form of noise.
   - The autoencoder must undo this corruption rather than simply copying their input.



AE's: Denoising Autoencoders (Training Procedure)

1. The DAE training procedure is

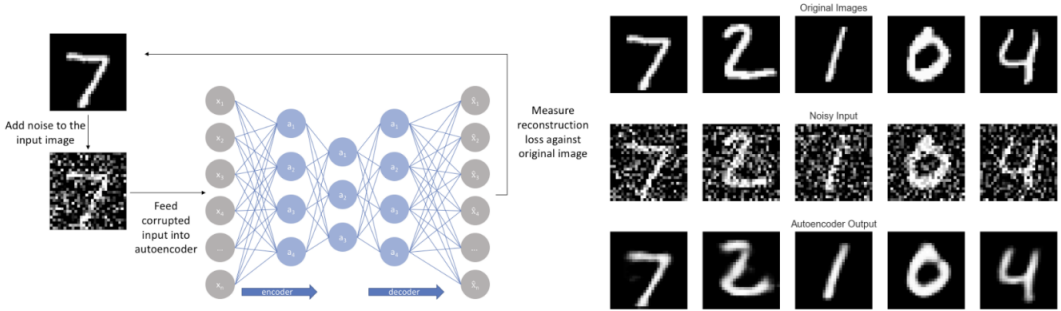


2. We introduce a corruption process $C(\tilde{x}|x)$.

AE's: Denoising Autoencoders (Training Procedure)

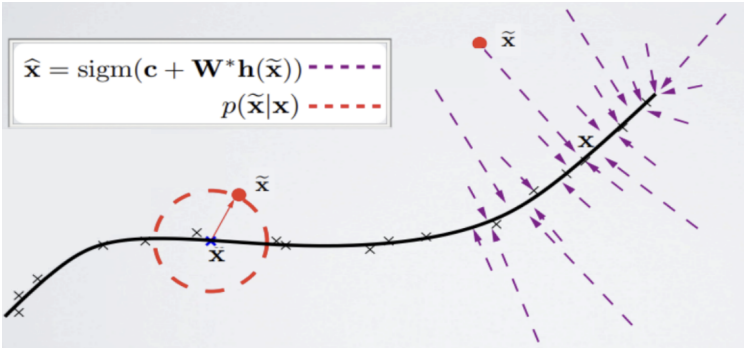
1. We introduce a corruption process $C(\tilde{x}|x)$.
2. The autoencoder then learns a reconstruction distribution $p_{\text{reconstruct}}(x|\tilde{x})$ estimated from training pairs $(x, \tilde{x})$ as follows:
   - Sample a training example x_i from the training data.
   - Sample a corrupted version $\tilde{x}_i$ from $C(\tilde{x}_i|x = x_i)$.
   - Use $(x, \tilde{x})$ as a training example for estimating the autoencoder reconstruction distribution $p_{\text{reconstruct}}(x|\tilde{x}) = p_{\text{decoder}}(x|h)$ with h the output of encoder $f(\tilde{x})$ and p_{decoder} typically defined by a decoder $g(h)$.
   - Typically we can simply perform gradient-based approximate minimization on the negative log-likelihood $-\log p_{\text{decoder}}(x|h)$.

AE's: Denoising Autoencoders (Result On MNIST)



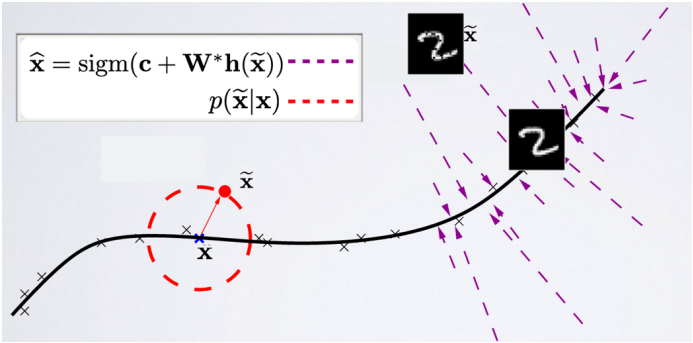
AE's: Denoising Autoencoders (It Isn't Lazy Network!)

- DAEs won't simply memorize the inputs and outputs (More robustness)
- Intuitively, a DAE learns a projection from a neighborhood of our training data back onto the training data



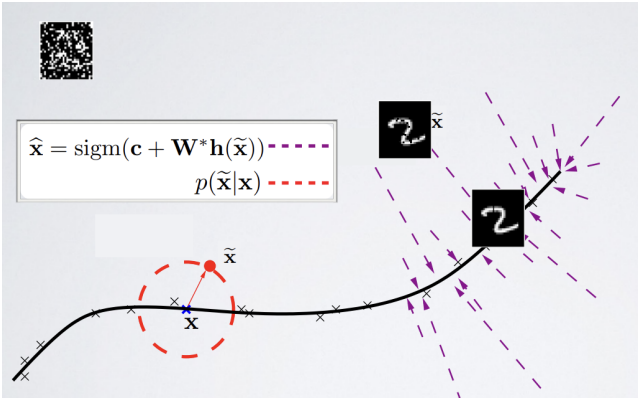
AE's: Denoising Autoencoders (It Isn't Lazy Network!)

- DAEs won't simply memorize the inputs and outputs (More robustness)
- Intuitively, a DAE learns a projection from a neighborhood of our training data back onto the training data



AE's: Denoising Autoencoders (It Isn't Lazy Network!)

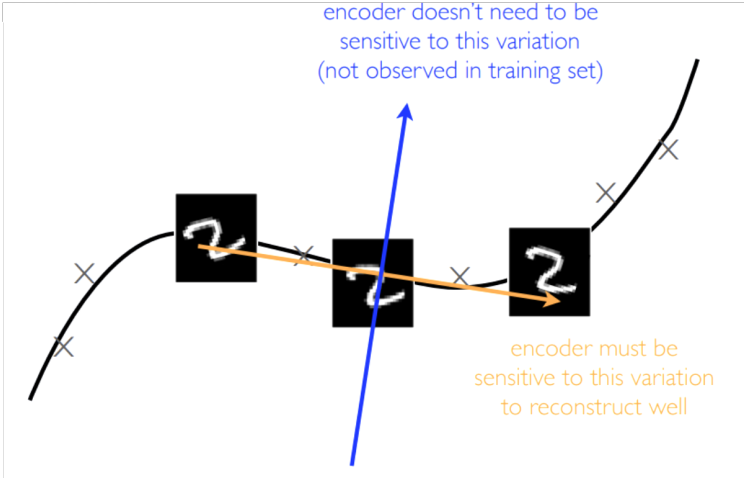
- DAEs won't simply memorize the inputs and outputs (More robustness)
- Intuitively, a DAE learns a projection from a neighborhood of our training data back onto the training data



AE's: Contractive Autoencoder (Intuition)

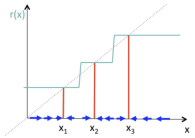
1. Contractive autoencoders are explicitly encouraged to learn a manifold through their loss function (Alain and Bengio 2014).
2. Desirable property: Points close to each other in input space maintain that property in the latent space.
3. Method to avoid uninteresting solutions
4. Add an explicit term in the loss that penalizes that solution
5. We wish to extract features that only reflect variations observed in the training set
6. We would like to be invariant to other variations

AE's: Contractive Autoencoder (Intuition)



AE's: Contractive Autoencoder (Details)

1. Contractive autoencoder has an explicit regularizer on $h = f(x)$, encouraging the derivatives of f to be as small as possible.
2. This will be true if $f(x) = h$ is continuous, has small derivatives.



3. We can use the Frobenius Norm of the Jacobian Matrix as a regularization term:

$$\Omega(f, x) = \lambda \left\| \frac{\partial f(x)}{\partial x} \right\|_F^2$$

4. These autoencoders are called *contractive* because they contract the neighborhood of input space into a smaller, localized group in latent space.
5. **Exercise:** What is the difference between DAE and CAE?

AE's: Contractive Autoencoder (Details)

- New loss function:

$$\underbrace{l\left(f\left(x^{(t)}\right)\right)}_{\texttt{autoencoder reconstruction}} + \lambda \underbrace{\|\nabla_{x^{(t)}} h(x^{(t)})\|_F^2}_{\texttt{Jacobian of encoder}}$$

- where, for binary observations:

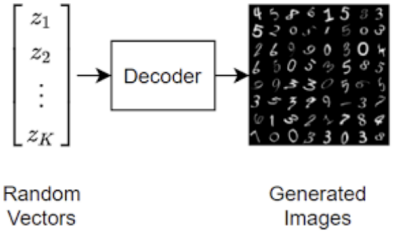
$$l\left(f\left(x^{(t)}\right)\right) = -\Sigma_k \left(x_k^{(t)} \log\left(\hat{x}_k^{(t)}\right) + \left(1 - x_k^{(t)}\right) \log\left(1 - \hat{x}_k^{(t)}\right)\right) \quad \Big\} \ \texttt{encoders keeps good information}$$

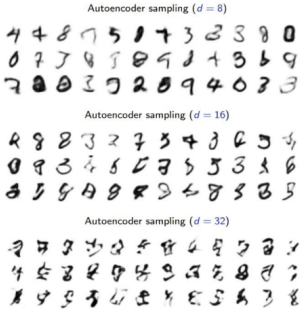
$$\|\nabla_{x^{(t)}} h(x^{(t)})\|_F^2 = \Sigma_j \Sigma_k \left(\frac{\partial h\left(x^{(t)}\right)_j}{\partial x_k^{(t)}}\right)^2 \quad \Big\} \ \texttt{encoder throws away all information}$$

$\rightarrow$ encoders keep only good information

Can The Decoder Of AE Be Used For Data Generation?

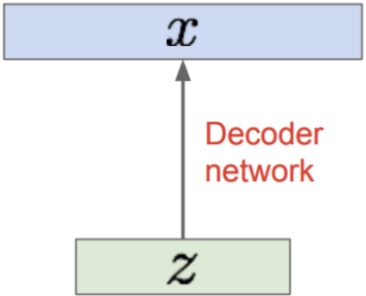
- Autoencoders are effective data compressors but up to now do not yet show **data generation capabilities**.
- We already trained an AE on MNIST; then we use **random vectors** as input to the Decoder to generate new data.
  - Is the **size** of the bottleneck affecting the **quality** of the generated sample?

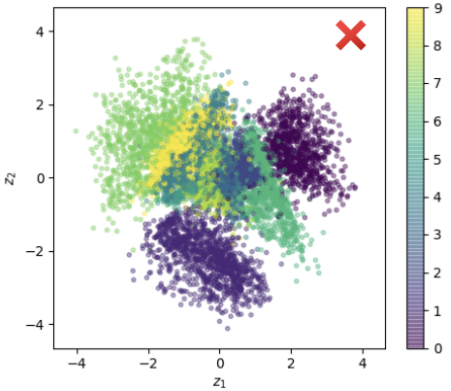




Can The Decoder Of AE Be Used For Data Generation?

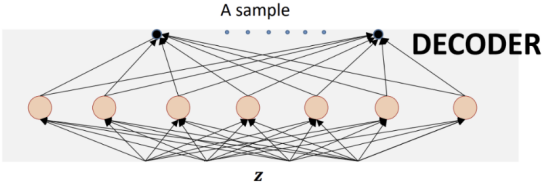
Is **any random vector** we feed into decoder networks results in a valid output?

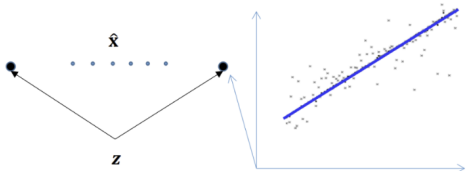




Can The Decoder Of AE Be Used For Data Generation?

- **Latent Variation:** Varying z generates outputs along a learned manifold.
- **Manifold Constraint:** Data lies within patterns learned from training.
- **Generation:** Outputs follow the training data distribution.



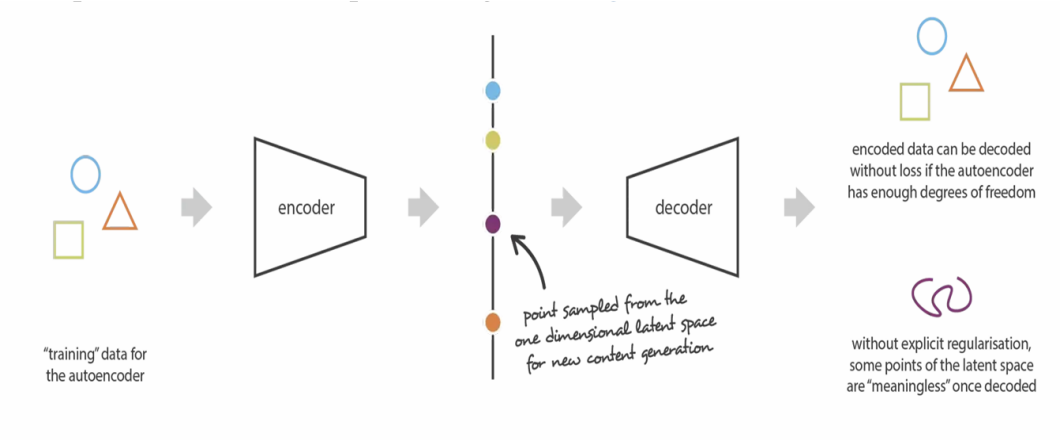


Latent Space Properties

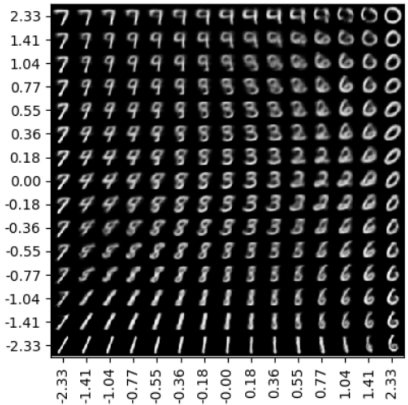
- For the generation purpose, we need the latent space to show the following properties:
  - **Continuity:** Similar points in the latent space will be similar after decoding.
  - **Completeness:** Samples of the latent space lead to meaningful content after decoding.

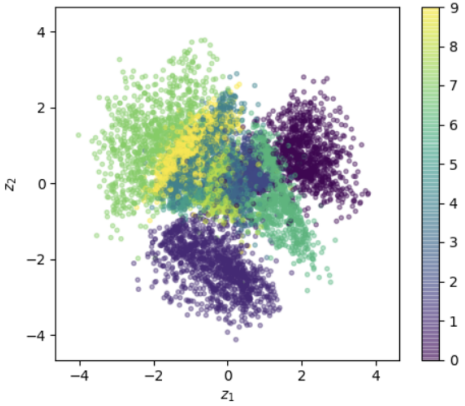
Latent Space Properties: Continuity

Latent Space Properties: Completeness



AE Latent Space - MNIST



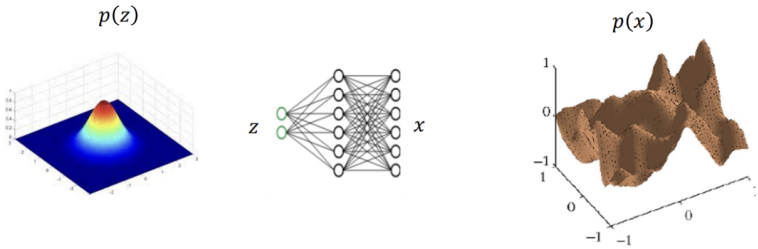


What is the problem?

Latent Variable

- We define a simple distribution on the latent space, i.e., $p(z)$ and a decoder network $p_\theta(\mathbf{x}|\mathbf{z})$ to map the latent variable to the data space.

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})\, p(\mathbf{z})\, d\mathbf{z}$$



AE's: Variational Autoencoder (Intuition)

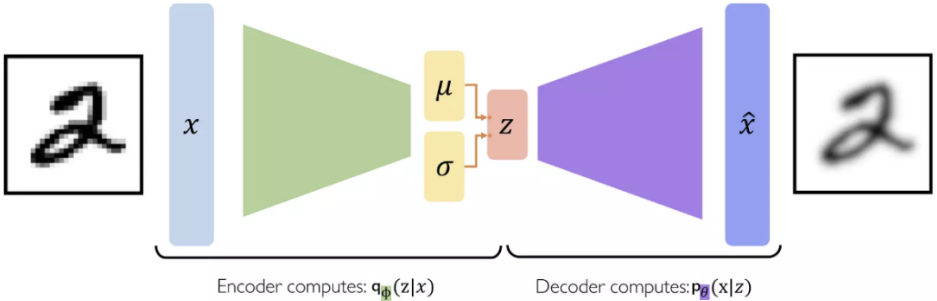
The key change that they added was that instead of having an encoder that maps points in X to points in Z, VAE's map points in X to distributions in Z. This allows a point in X to be indirectly mapped to several close neighbour points in Z.

Simple AE: input x $\xrightarrow{\text{Encoding}}$ latent representation $z = e(x)$ $\xrightarrow{\text{Decoding}}$ input reconstruction $d(z)$

Variational AE: input x $\xrightarrow{\text{Encoding}}$ latent distribution $p(z|x)$ $\xrightarrow{\text{Sampling}}$ sampled latent representation $z \sim p(z|x)$ $\xrightarrow{\text{Decoding}}$ input reconstruction $d(z)$

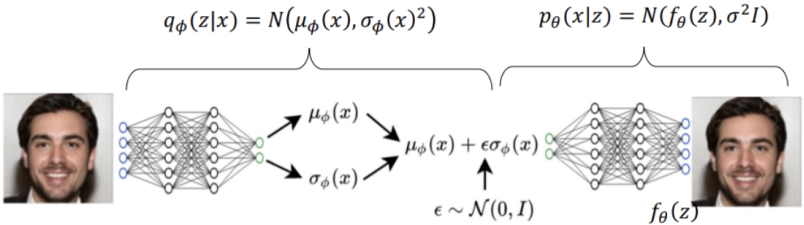
AE's: Variational Autoencoder (Big Picture)

The key change that they added was that instead of having an encoder that maps points in X to points in Z, VAE's map points in X to distributions in Z. This allows a point in X to be indirectly mapped to several close neighbour points in Z.



$$\mathcal{L}(\phi, \theta, x) = \text{(reconstruction loss)} + \text{(regularization term)}$$

AE's: Variational Autoencoder (Details)



VAE loss: $\mathscr{L}(x, \theta, \phi) =$ $\underbrace{\|x^{(i)} - \hat{x}^{(i)}\|^2}_{\text{reconstruction loss}}$ $+ \underbrace{KL\left(q_\phi(z|x^{(i)}) \,\|\, \mathcal{N}(0, I)\right)}_{\text{regularization term}}$

$\downarrow$ $f_\theta(z) = f_\theta\left(\mu_\phi(x^{(i)}) + \epsilon\sigma_\phi(x^{(i)})\right)$

$\downarrow$ Common choice of prior on the latent distribution.

AE's: Variational Autoencoder (Details)

- Encourage encodings to be evenly distributed around the center of the latent space
- Penalize the network when it tries to memorize data

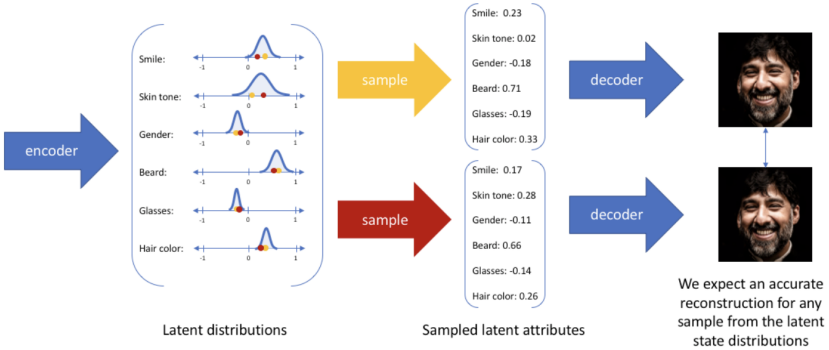
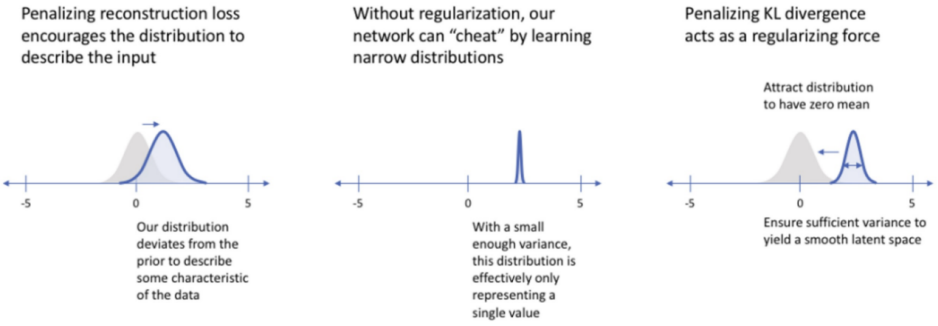


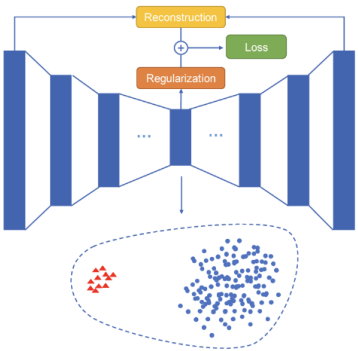
Image: Source

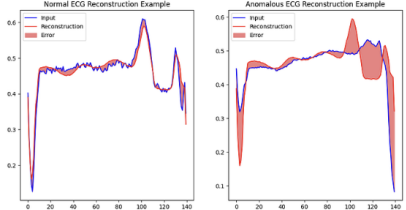
AE's: Variational Autoencoder (Details)

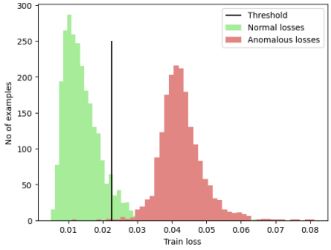


The VAE objectives arranges data on a compact manifold (we can sample from) in a continuous smooth way.

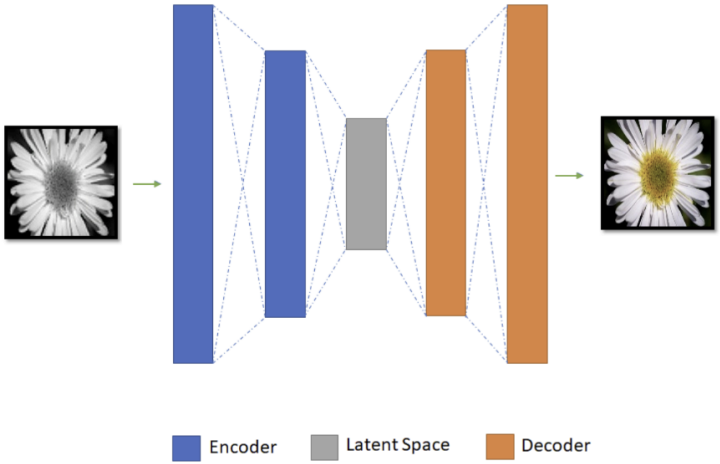
AE other Applications: Anomaly Detection



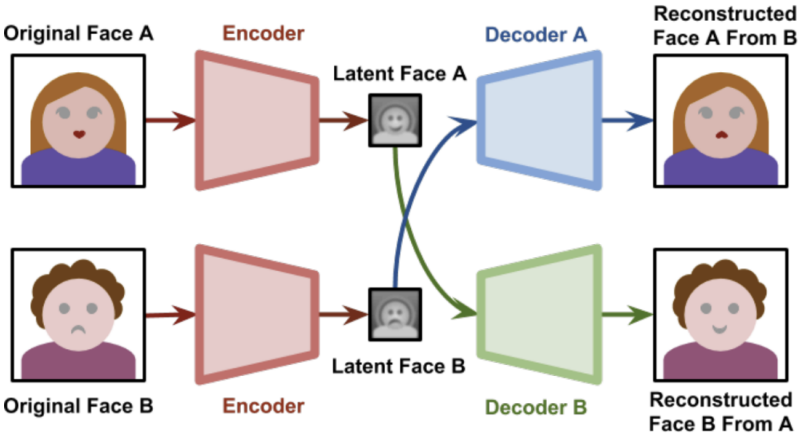




AE other Applications: Image Colorization



AE other Applications: Fake Lough



Slides by: Amirhossein Izadi

- F.-F. Li, J. Wu, and R. Gao, “CS231n” Lecture slides, 2024, Stanford
- A. Amini, “6s191” Lecture slides, 2024, MIT.
- M. Soleymani, “Deep Learning” Lecture slides, 2024, Sharif University of Technology.
- H. Beigy, “Deep Learning” Lecture slides, 2022, Sharif University of Technology.
- S. Levine, “CS W182/282A” Lecture slides, 2024, UC Berkeley
- Y. Maziane, “Diffusion models: Seek of information and structure in latent space,” 2022, University of Liège.
- G. Buzzard, “Mathematical Aspects of Neural Networks” Lecture slides, 2019, Purdue University.

Any Questions?