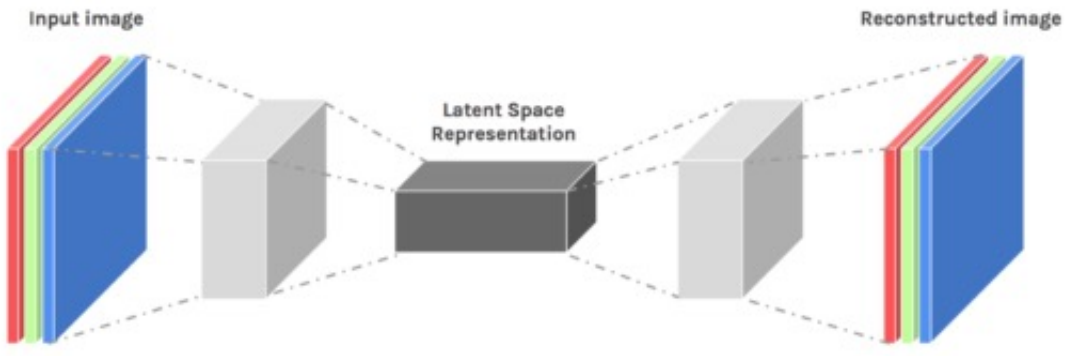
Autoencoders

- Autoencoders are designed to reproduce their input, especially for images.
  - Key point is to reproduce the input from a learned encoding.



https://www.edureka.co/blog/autoencoders-tutorial/

Autoencoders

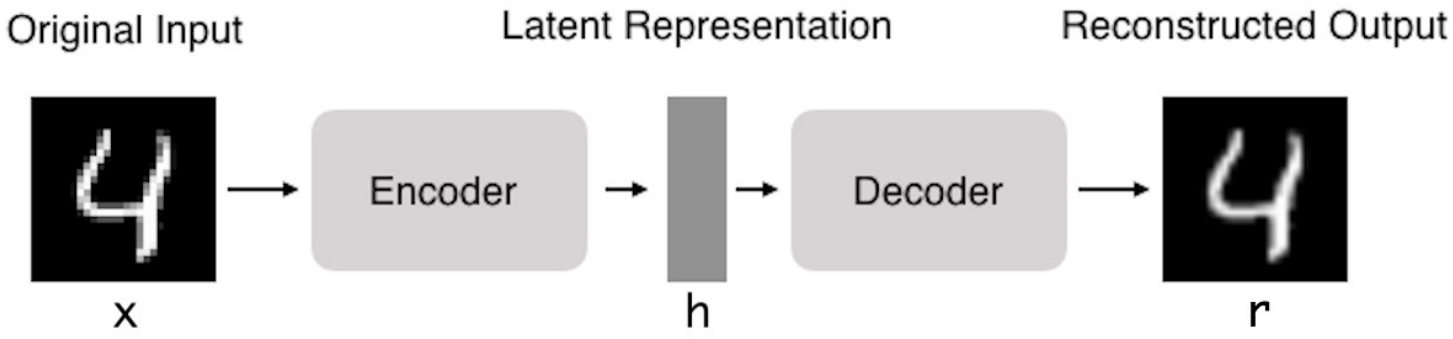
- Compare PCA/SVD
  - PCA takes a collection of vectors (images) and produces a usually smaller set of vectors that can be used to approximate the input vectors via linear combination.
  - Very efficient for certain applications.
  - Fourier and wavelet compression is similar.

- Neural network autoencoders
  - Can learn nonlinear dependencies
  - Can use convolutional layers

https://www.edureka.co/blog/autoencoders-tutorial/

Autoencoders: structure

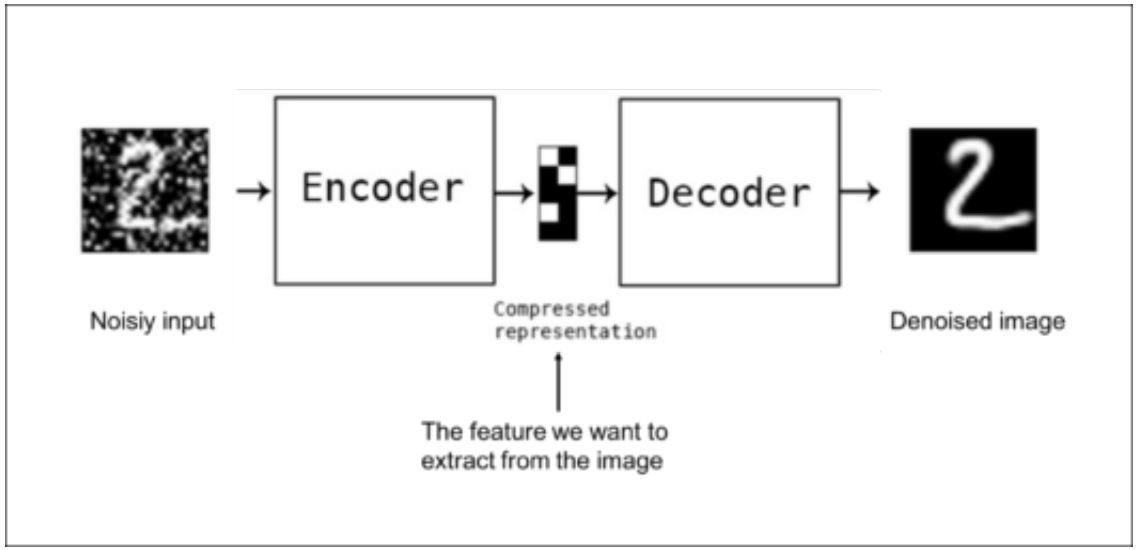
- Encoder: compress input into a latent-space of usually smaller dimension. h = f(x)
- Decoder: reconstruct input from the latent space. r = g(f(x)) with r as close to x as possible



https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f

Autoencoders: applications

- Denoising: input clean image + noise and train to reproduce the clean image.



https://www.edureka.co/blog/autoencoders-tutorial/

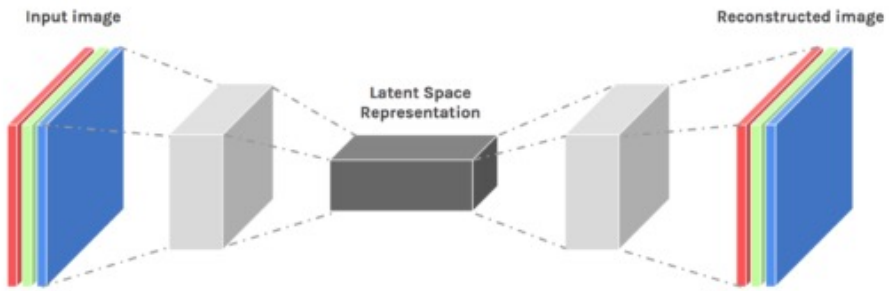
Autoencoders: Applications

- Watermark removal

https://www.edureka.co/blog/autoencoders-tutorial/

Bottleneck layer (undercomplete)

- Suppose input images are nxn and the latent space is m < nxn.
- Then the latent space is not sufficient to reproduce all images.
- Needs to learn an encoding that captures the important features in training data, sufficient for approximate reconstruction.



Simple bottleneck

- input_img = Input(shape=(784,))
- encoding_dim = 32
- encoded = Dense(encoding_dim, activation='relu')(input_img)
- decoded = Dense(784, activation='sigmoid')(encoded)
- autoencoder = Model(input_img, decoded)

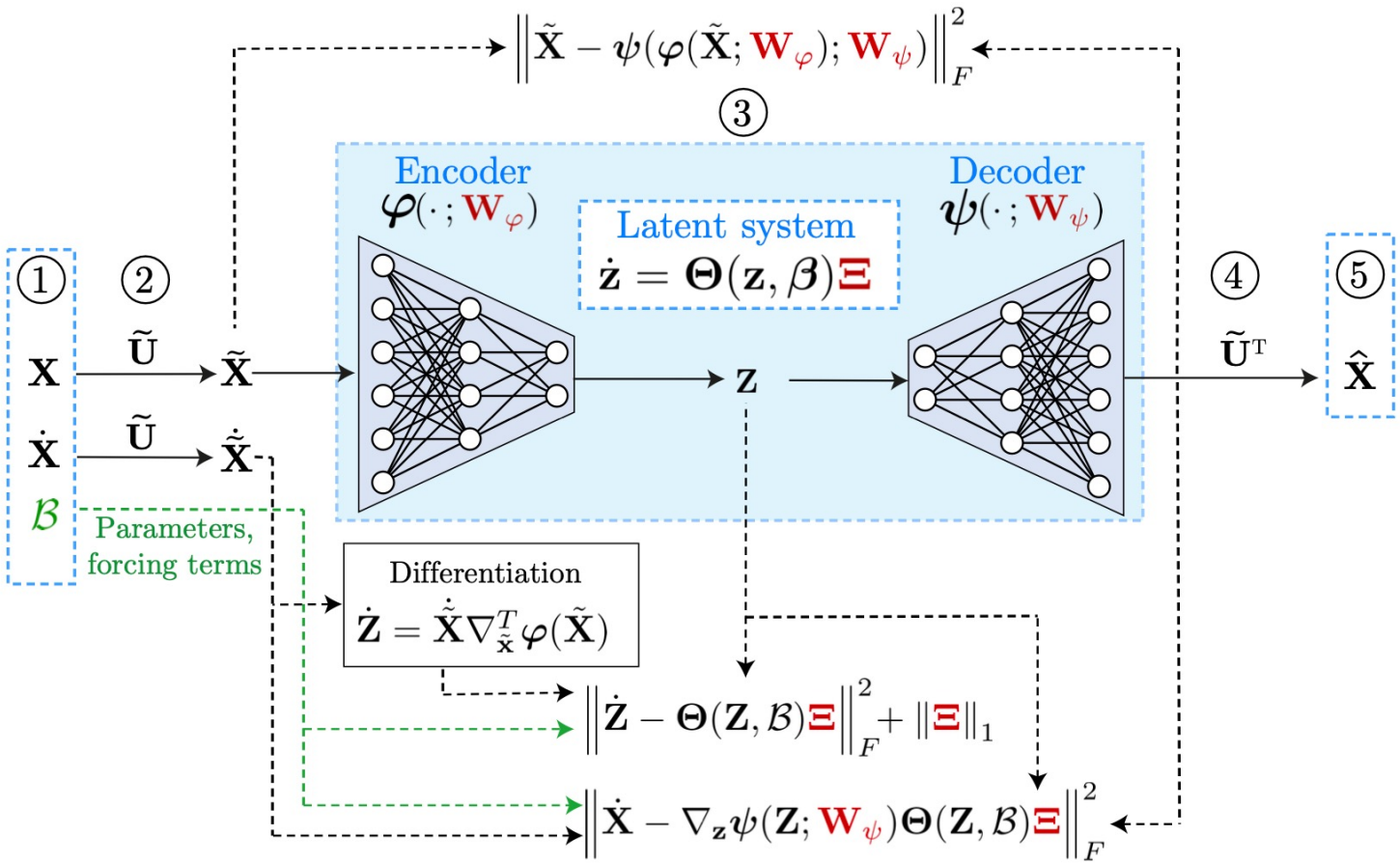
- Maps 28x28 images into a 32 dimensional vector.
- Can also use more layers and/or convolutions.

https://blog.keras.io/building-autoencoders-in-keras.html

Denoising autoencoders

- Basic autoencoder trains to minimize the loss between x and the reconstruction g(f(x)).
- Denoising autoencoders train to minimize the loss between x and g(f(x+w)), where w is random noise.
- Same possible architectures, different training data.
- Kaggle has a dataset on damaged documents.

https://blog.keras.io/building-autoencoders-in-keras.html


$\left\|\tilde{\mathbf{X}} - \psi(\varphi(\tilde{\mathbf{X}}; \mathbf{W}_\varphi); \mathbf{W}_\psi)\right\|_F^2$
3
Encoder
$\varphi(\cdot\,; \mathbf{W}_\varphi)$
Latent system
$\dot{\mathbf{z}} = \boldsymbol{\Theta}(\mathbf{z}, \boldsymbol{\beta})\boldsymbol{\Xi}$
Decoder
$\psi(\cdot\,; \mathbf{W}_\psi)$
1
2
4
5
$\mathbf{X}$
$\dot{\mathbf{X}}$
$\mathcal{B}$
$\tilde{\mathbf{U}}$
$\tilde{\mathbf{X}}$
$\dot{\tilde{\mathbf{X}}}$
$\mathbf{Z}$
$\tilde{\mathbf{U}}^{\mathrm{T}}$
$\hat{\mathbf{X}}$
Parameters, forcing terms
Differentiation
$\dot{\mathbf{Z}} = \dot{\tilde{\mathbf{X}}} \nabla_{\tilde{\mathbf{x}}}^T \varphi(\tilde{\mathbf{X}})$
$\left\|\dot{\mathbf{Z}} - \boldsymbol{\Theta}(\mathbf{Z}, \mathcal{B})\boldsymbol{\Xi}\right\|_F^2 + \|\boldsymbol{\Xi}\|_1$
$\left\|\dot{\mathbf{X}} - \nabla_{\mathbf{z}} \psi(\mathbf{Z}; \mathbf{W}_\psi)\boldsymbol{\Theta}(\mathbf{Z}, \mathcal{B})\boldsymbol{\Xi}\right\|_F^2$