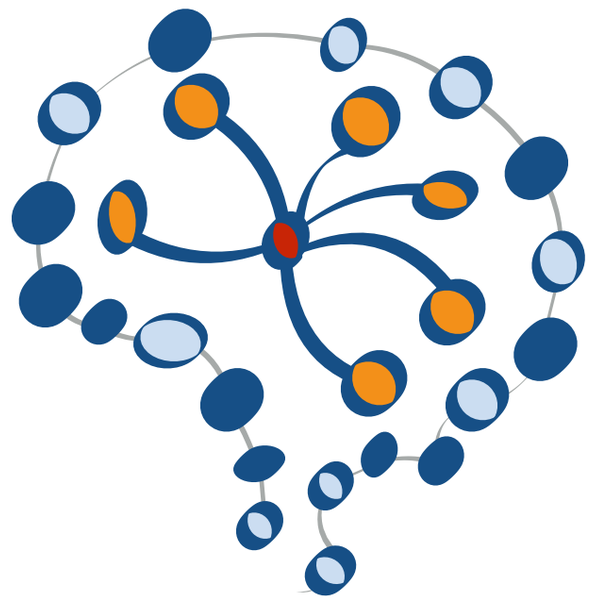
STAT 453: Introduction to Deep Learning and Generative Models

Sebastian Raschka

http://stat.wisc.edu/~sraschka/teaching



Lecture 16

Introduction to Autoencoders

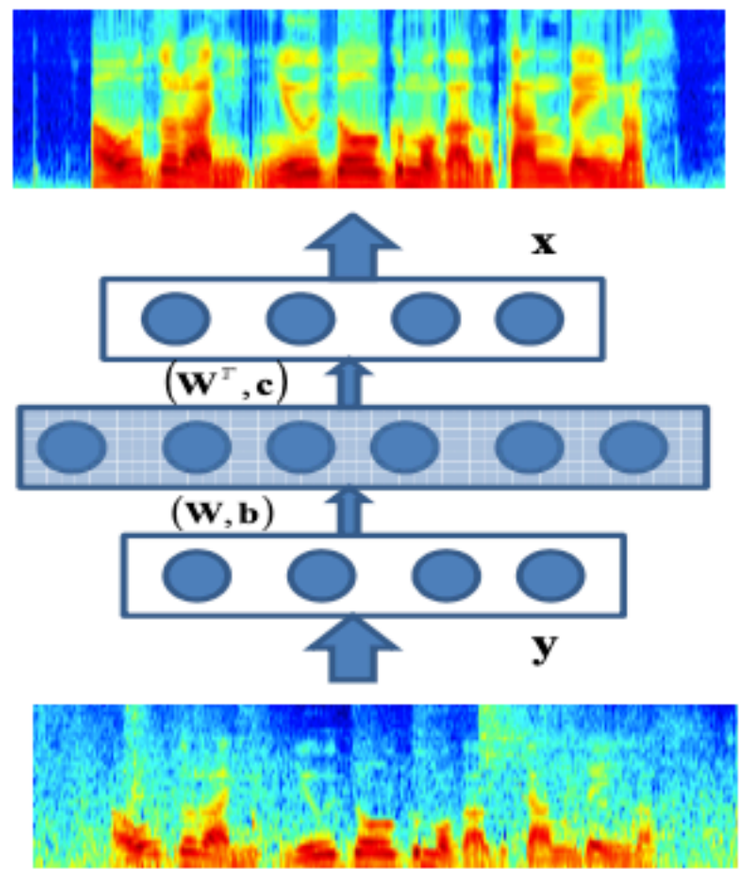
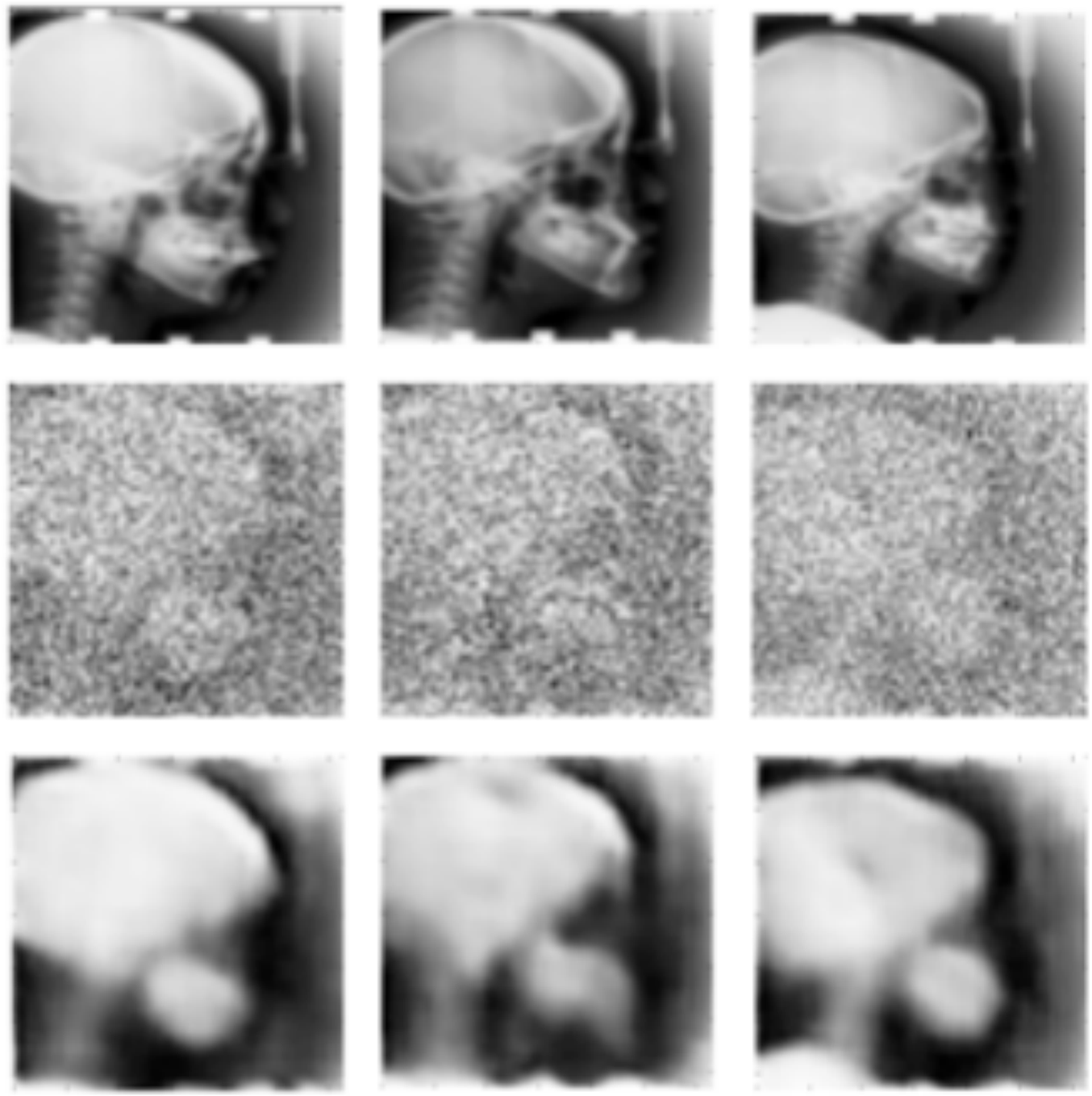


Figure 1: Training neural autoencoder with noisy-clean speech pairs.

Lu, X., Tsao, Y., Matsuda, S., & Hori, C. (2013, August). Speech enhancement based on deep denoising autoencoder. In *Interspeech* (pp. 436-440).



Gondara, L. (2016, December). Medical image denoising using convolutional denoising autoencoders. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)* (pp. 241-246). IEEE.

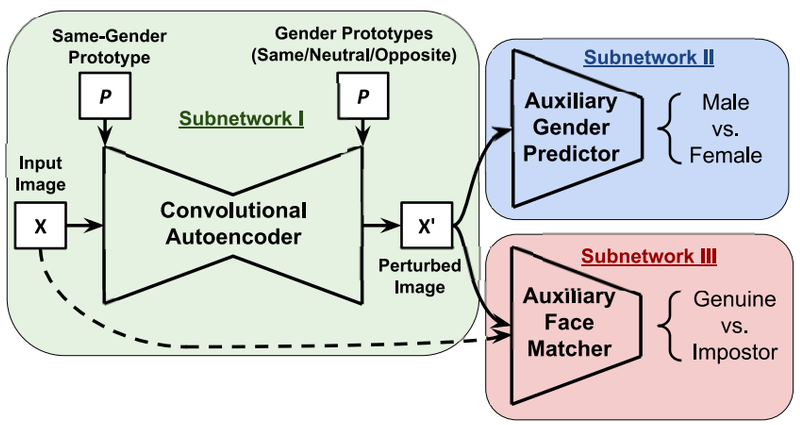
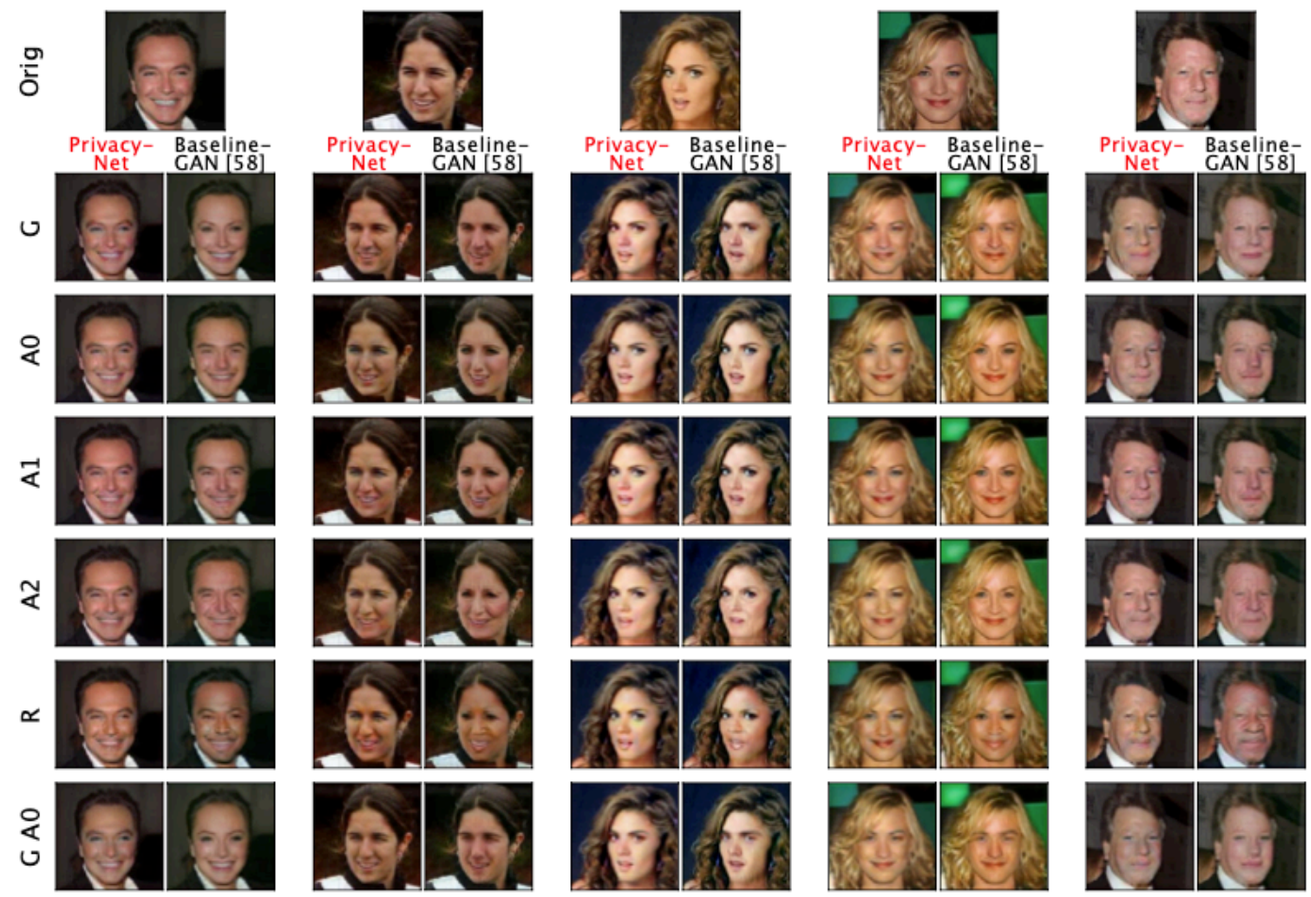


Figure 1. Schematic representation of the semi-adversarial neural network architecture designed to derive perturbations that are able to confound gender classifiers while still allowing biometric matchers to perform well. The overall network consists of three sub-components: a convolutional autoencoder (subnetwork I), an auxiliary gender classifier (subnetwork II), and an auxiliary matcher (subnetwork III).

Vahid Mirjalili, Sebastian Raschka, Anoop Namboodiri, and Arun Ross (2018) *Semi-adversarial networks: Convolutional autoencoders for imparting privacy to face images.* Proc. of 11th IAPR International Conference on Biometrics (ICB 2018), Gold Coast, Australia.



"About half (52%) of U.S. adults said they decided recently not to use a product or service because they were worried about how much personal information would be collected about them."

https://www.pewresearch.org/fact-tank/2020/04/14/half-of-americans-have-decided-not-to-use-a-product-or-service-because-of-privacy-concerns/

Lecture Overview

1. Dimensionality Reduction
2. Fully-connected Autoencoders
3. Convolutional Autoencoders
4. A Convolutional Autoencoder in PyTorch
5. Other Types of Autoencoders

Feature Extraction & Dimensionality Reduction

1. **Dimensionality Reduction**
2. Fully-connected Autoencoders
3. Convolutional Autoencoders
4. A Convolutional Autoencoder in PyTorch
5. Other Types of Autoencoders

Unsupervised Learning

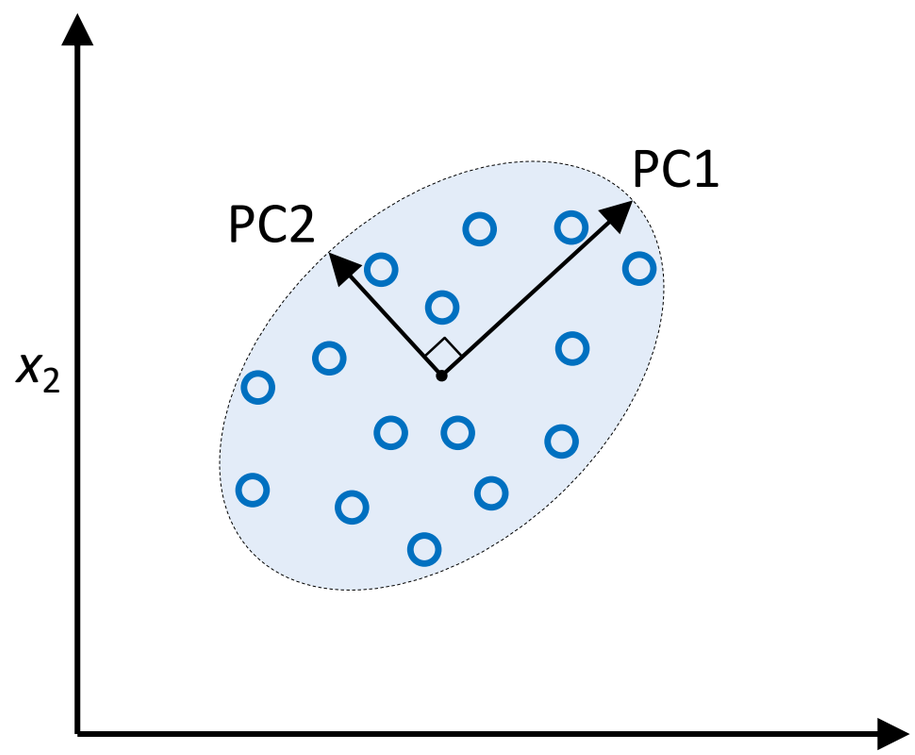
Working with datasets *without* considering a/the *target* variable

Some Applications and Goals:

- Finding hidden structures in data
- Data compression
- Clustering
- Retrieving similar objects
- Exploratory data analysis
- Generating new examples

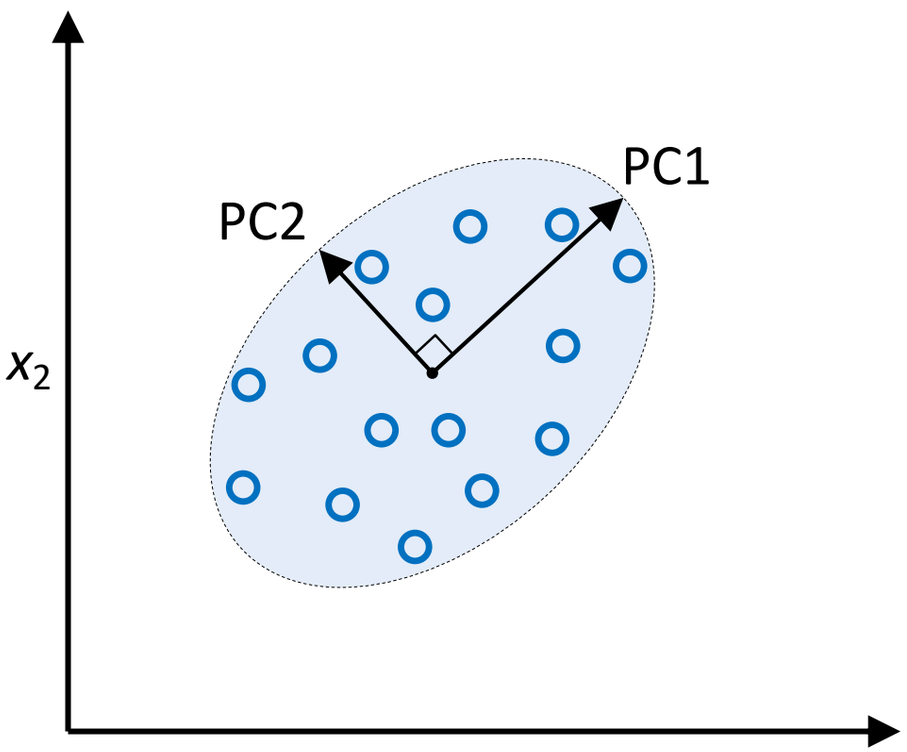
Principal Component Analysis (PCA)

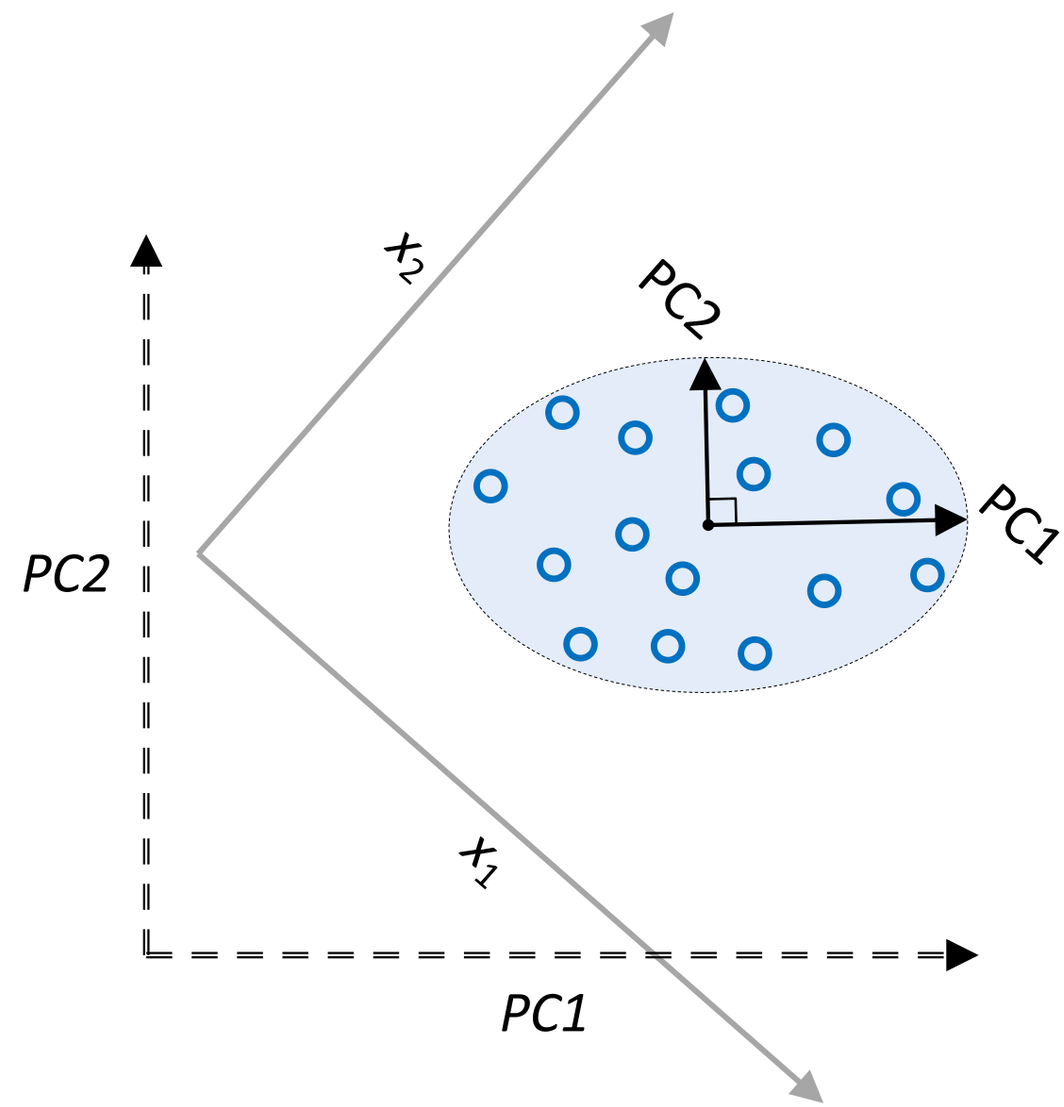
1) Find directions of maximum variance



Principal Component Analysis (PCA)

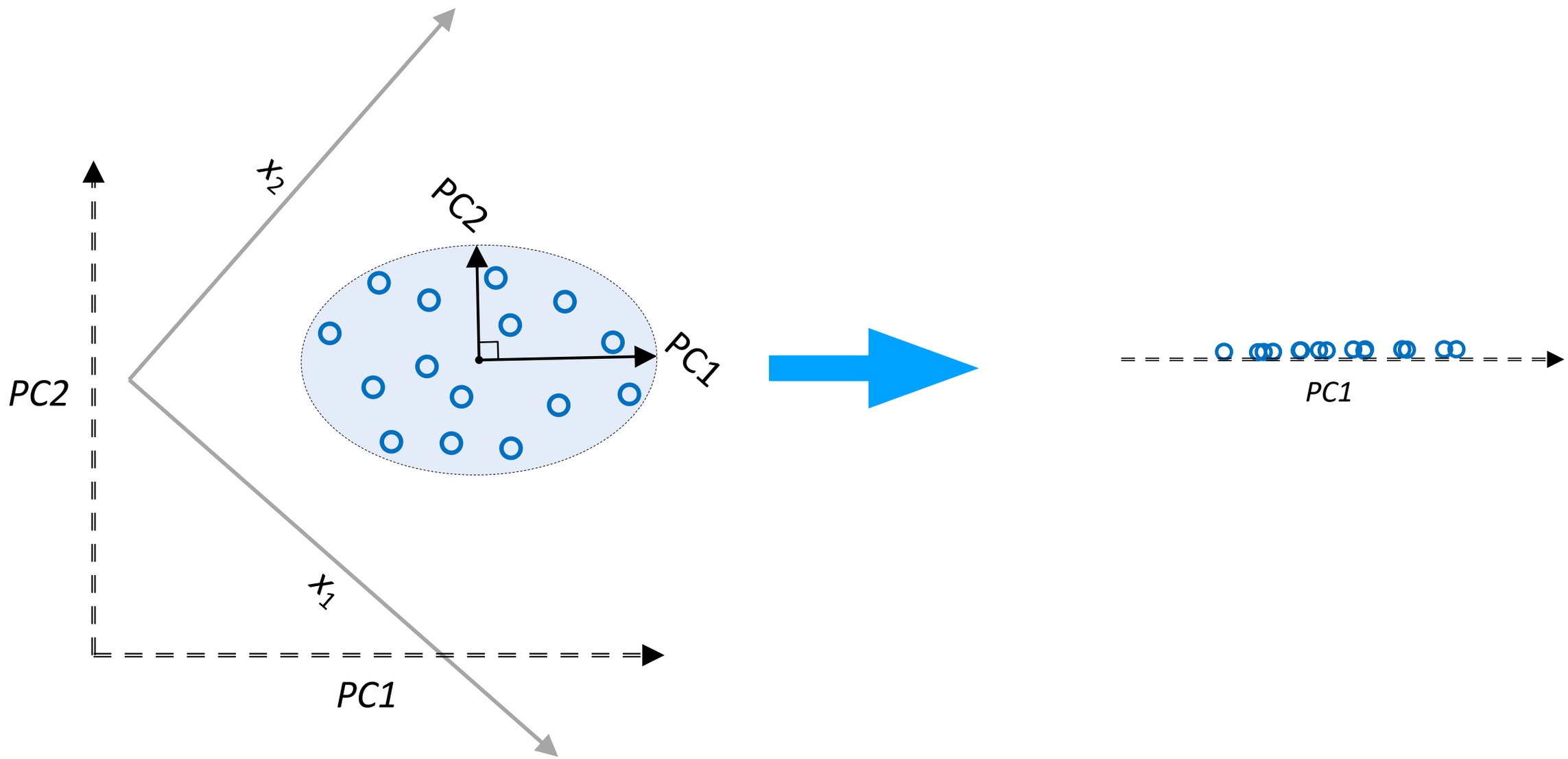
2) Transform features onto directions of maximum variance





Principal Component Analysis (PCA)

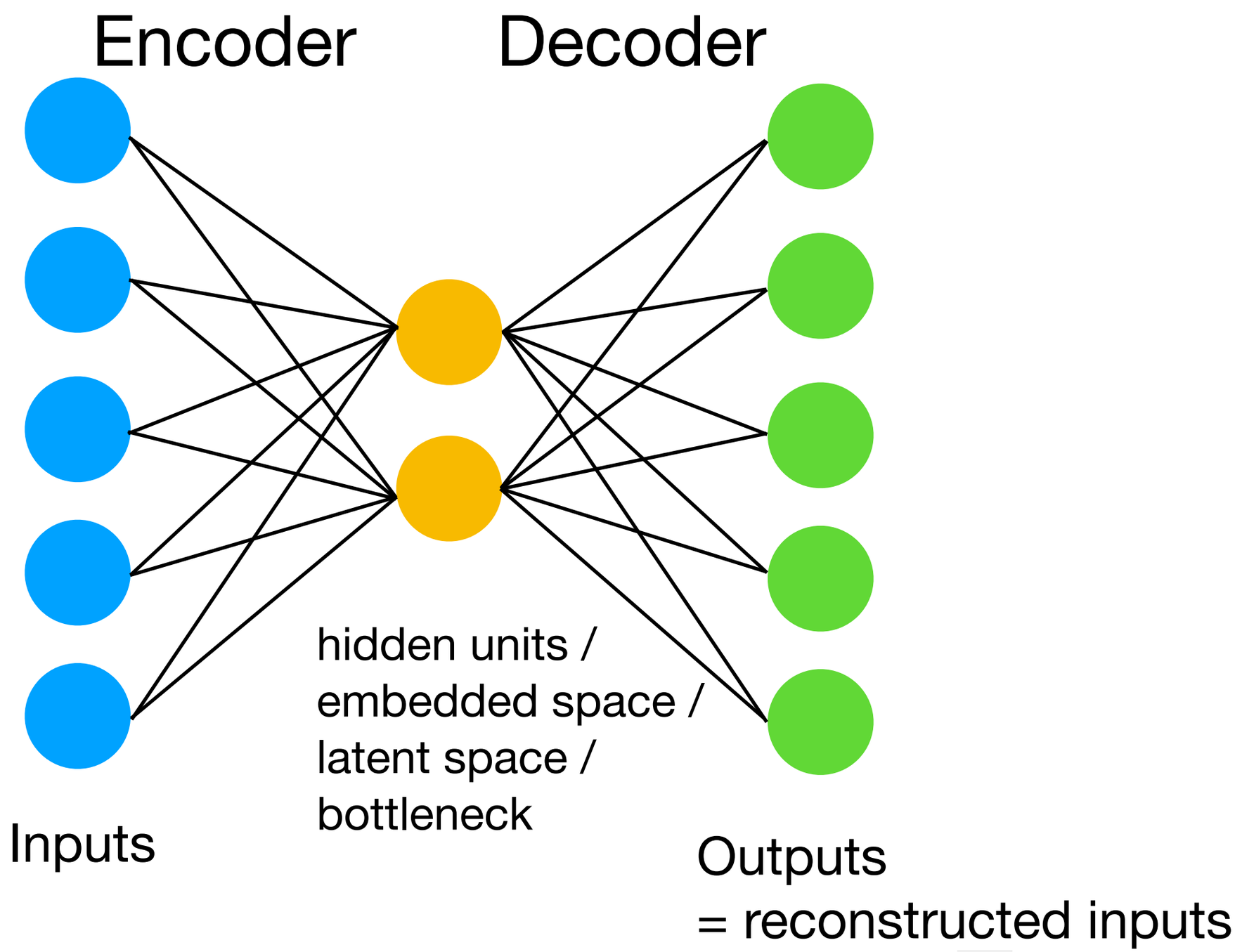
3) Usually consider a subset of vectors of most variance (dimensionality reduction)



An Hourglass-Shaped Multilayer Perceptron

1. Dimensionality Reduction
2. **Fully-connected Autoencoders**
3. Convolutional Autoencoders
4. A Convolutional Autoencoder in PyTorch
5. Other Types of Autoencoders

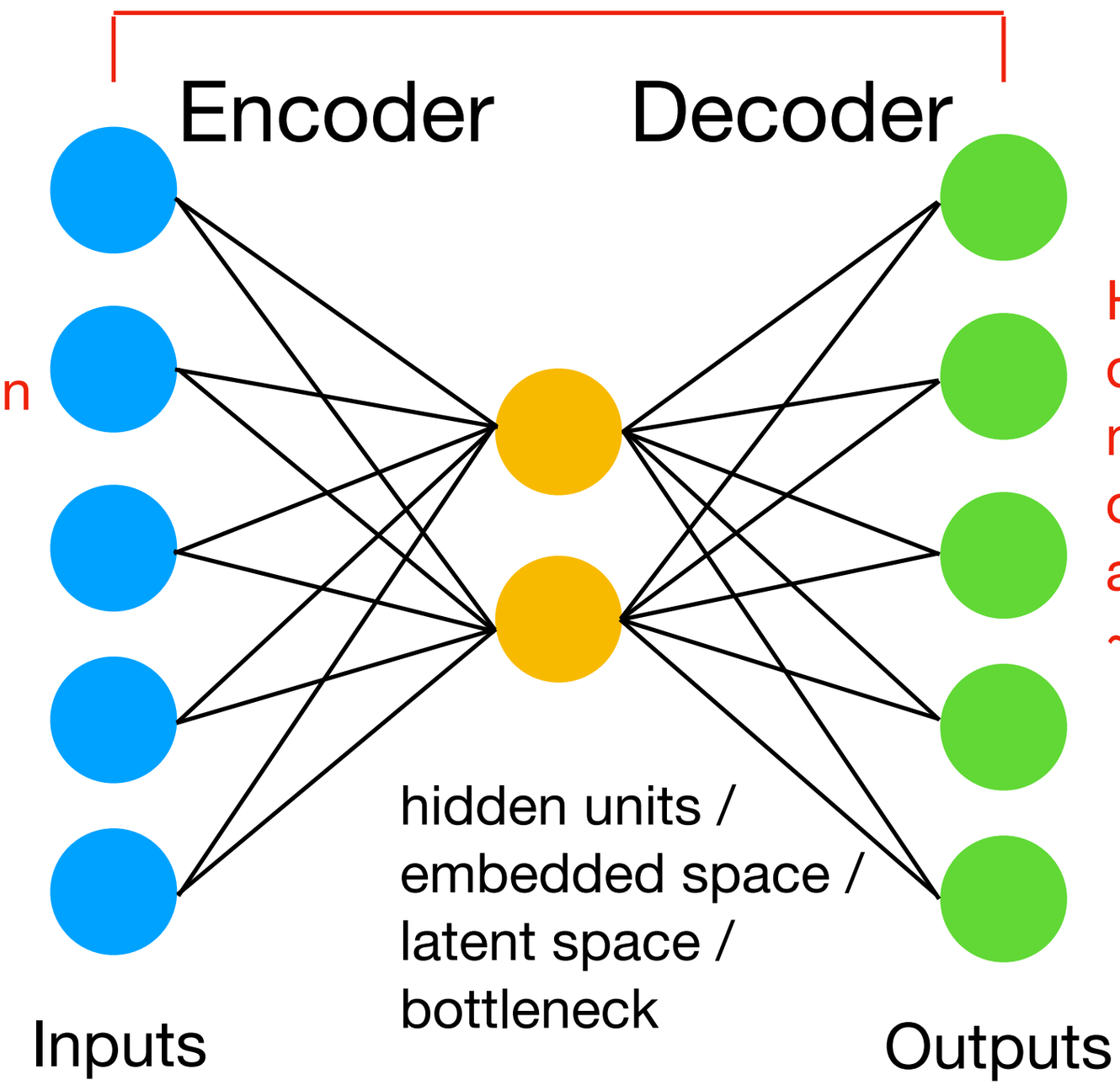
A Basic Fully-Connected (Multilayer-Perceptron) Autoencoder



A Basic Fully-Connected (Multilayer-Perceptron) Autoencoder

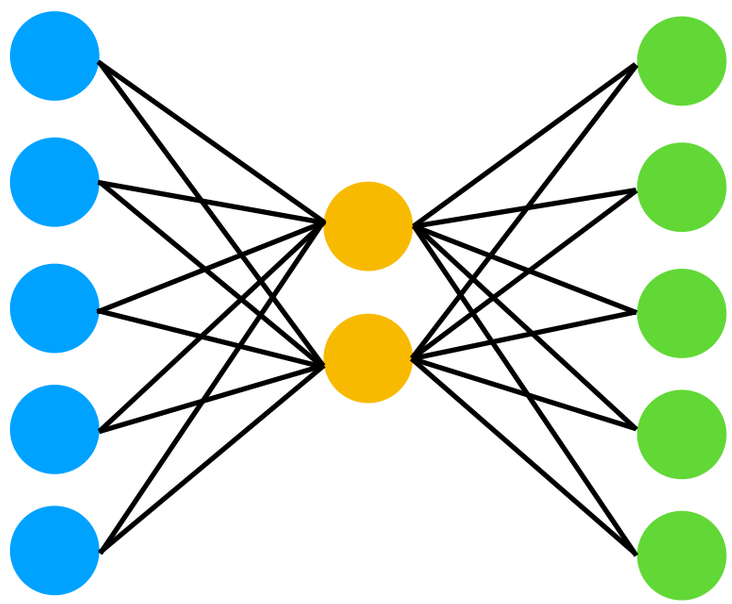
$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||_2^2 = \sum_i (x_i - x_i')^2$$

If we don't use non-linear activation functions and minimize the MSE, this is very similar to PCA



However, the latent dimensions will not necessarily be orthogonal and will have ~ same variance

A Basic Fully-Connected (Multilayer-Perceptron) Autoencoder

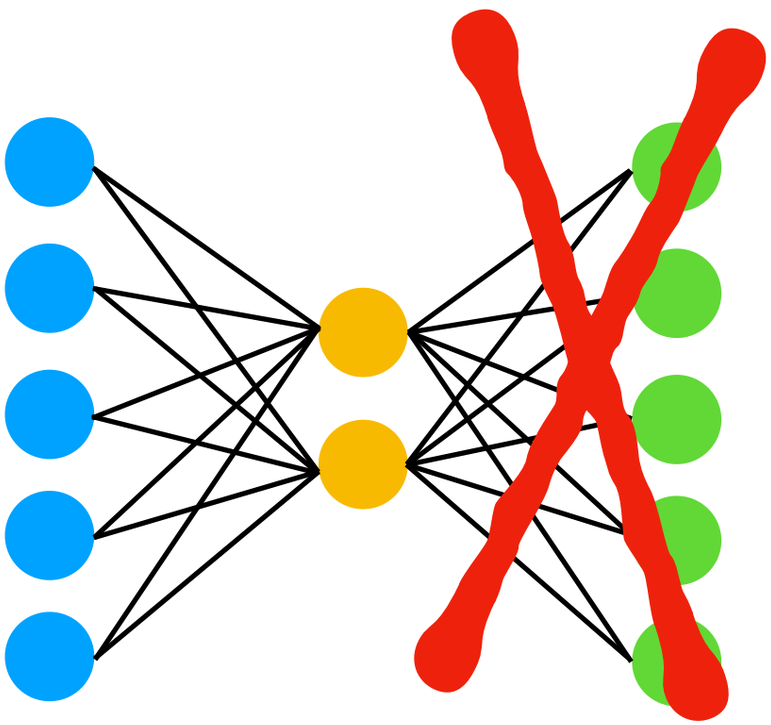


<u>Question:</u>

If we can achieve the same with PCA, which is essentially a kind of matrix factorization that is more efficient than Backprop + SGD, why bother with autoencoders?

Potential Autoencoder Applications

After training, disregard this part

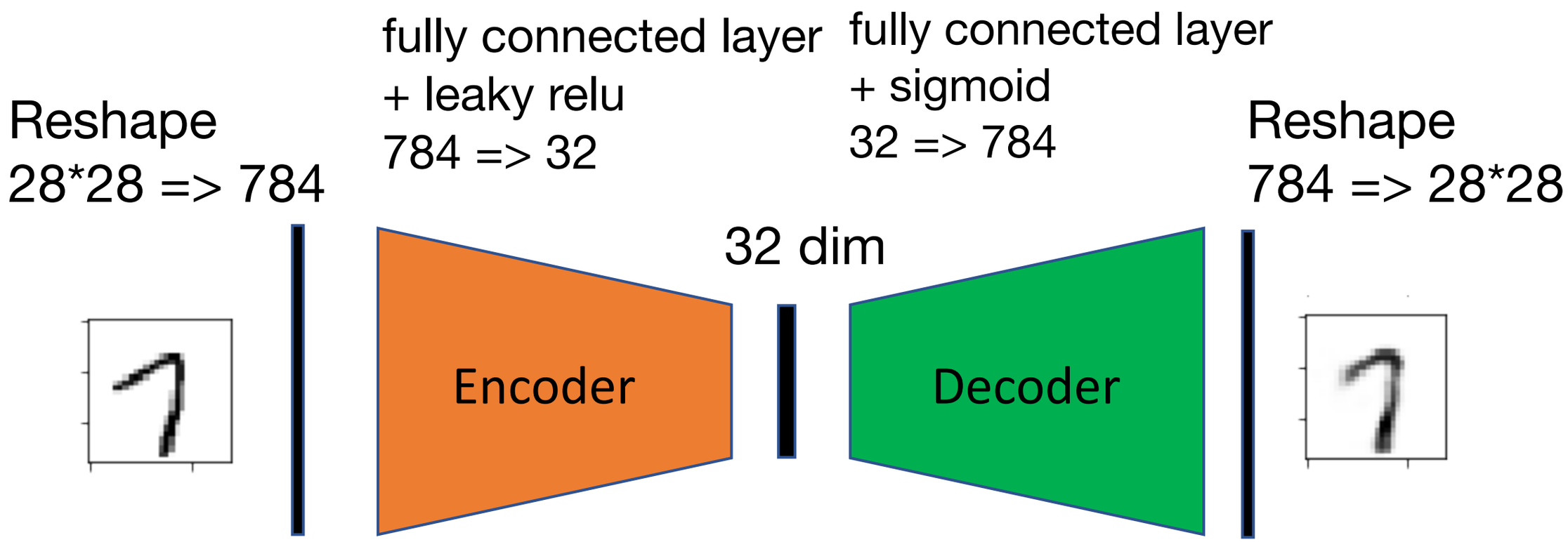


Use embedding as input to classic machine learning methods (SVM, KNN, Random Forest, ...)

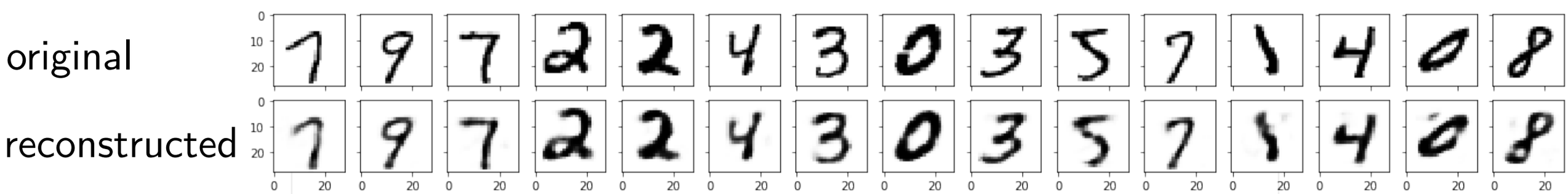
Or, similar to transfer learning, train autoencoder on large image dataset, then fine tune encoder part on your own, smaller dataset and/or provide your own output (classification) layer

Latent space can also be used for visualization (EDA, clustering), but there are better methods for that

A Simple Autoencoder



https://github.com/rasbt/deeplearning-models/blob/master/pytorch_ipynb/autoencoder/ae-basic.ipynb



Convolutional Autoencoders
& Transposed Convolutions / Deconvolutions

1. Dimensionality Reduction
2. Fully-connected Autoencoders
3. **Convolutional Autoencoders**
4. A Convolutional Autoencoder in PyTorch
5. Other Types of Autoencoders

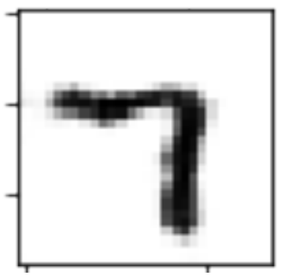
A Convolutional Autoencoder

1 or more
convolutional layers

1 or more
"de"convolutional layers

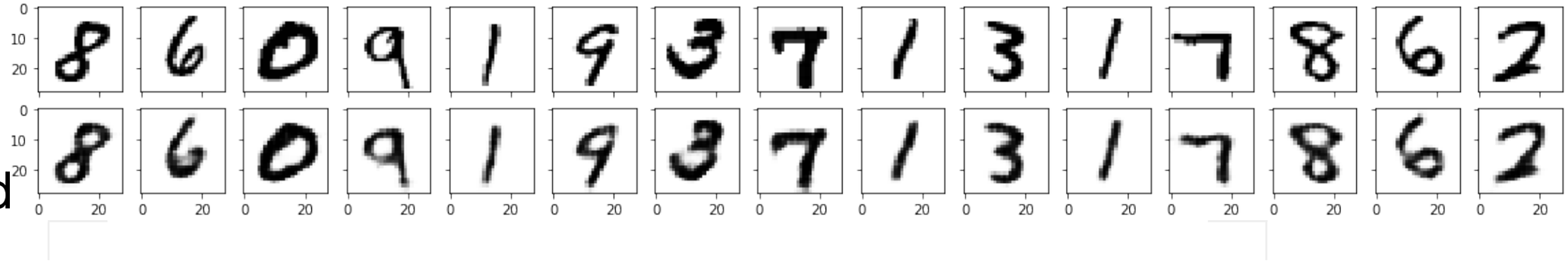
Encoder

Decoder



original

reconstructed



Transposed Convolution

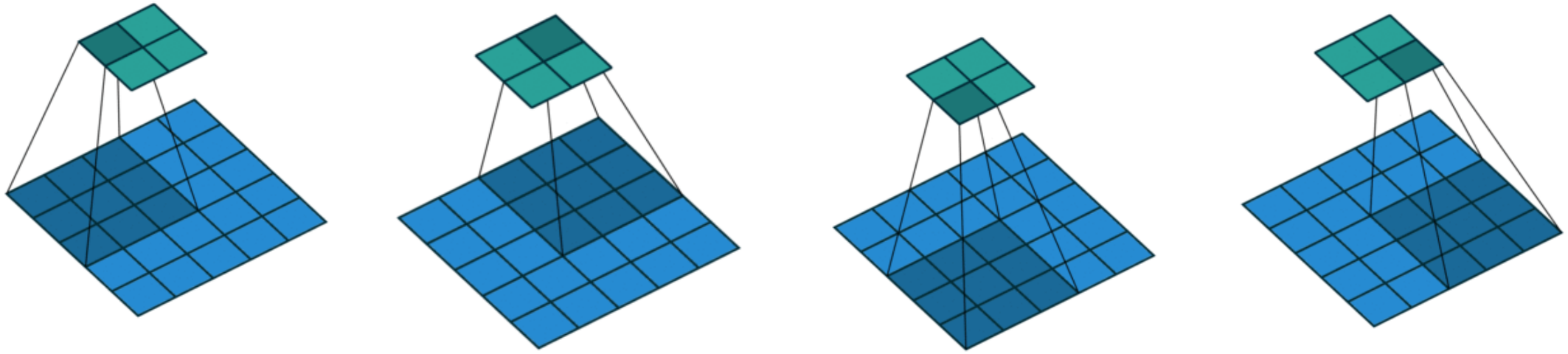
- Allows us to increase the size of the *output* feature map compared to the *input* feature map

- Synonyms:
  - often also (incorrectly) called "deconvolution" (mathematically, deconvolution is defined as the inverse of convolution, which is different from transposed convolutions)
  - the term "unconv" is sometimes also used
  - fractionally strided convolution is another (better?) term for that

Regular Convolution:

output

input

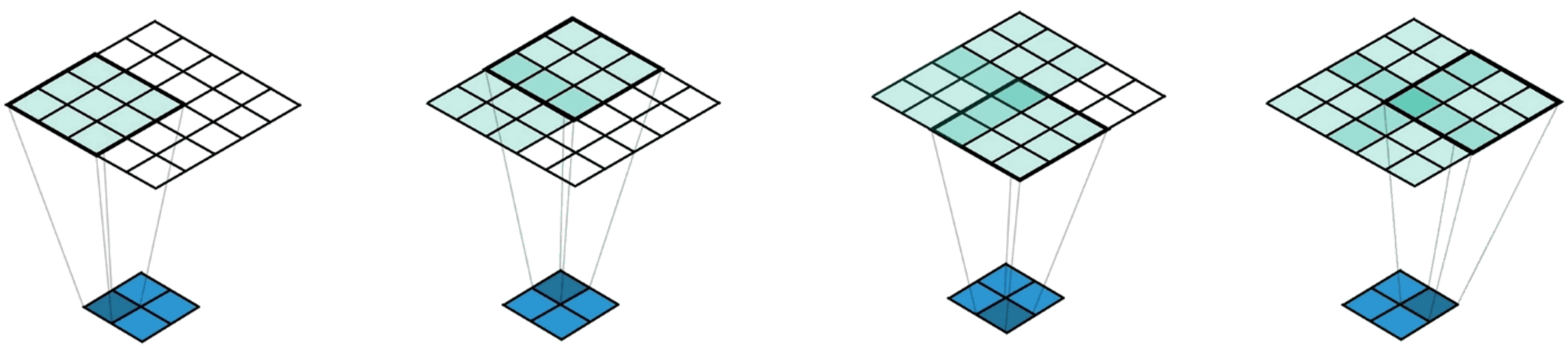


Dumoulin, Vincent, and Francesco Visin. "A guide to convolution arithmetic for deep learning." *arXiv preprint arXiv:1603.07285* (2016).

Transposed Convolution (stride = 2)

output

input

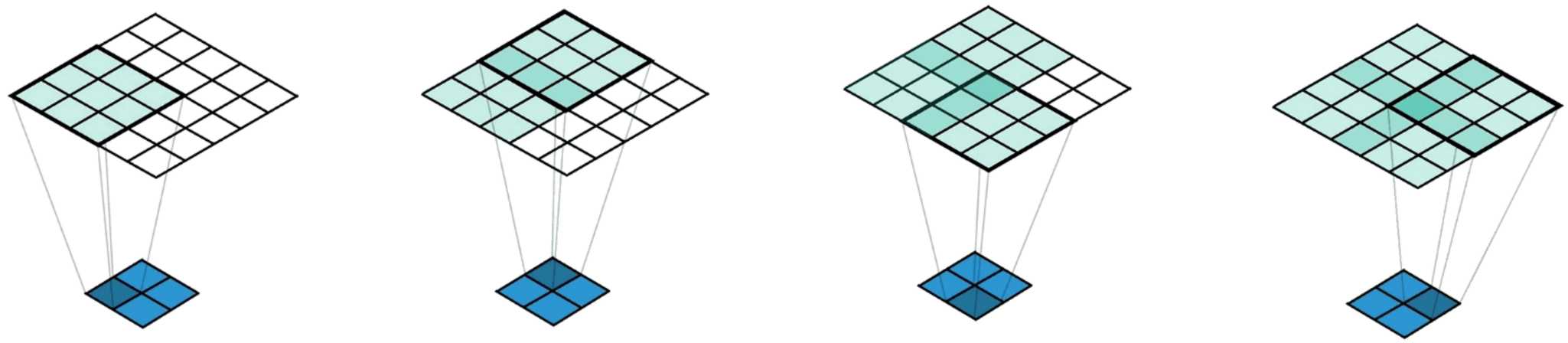


A Conv2DTranspose with 3x3 kernel and stride of 2x2 applied to a 2x2 input to give a 5x5 output. (https://medium.com/apache-mxnet/transposed-convolutions-explained-with-ms-excel-52d13030c7e8)

Transposed Convolution (3x3 kernel, stride=2)

output

input

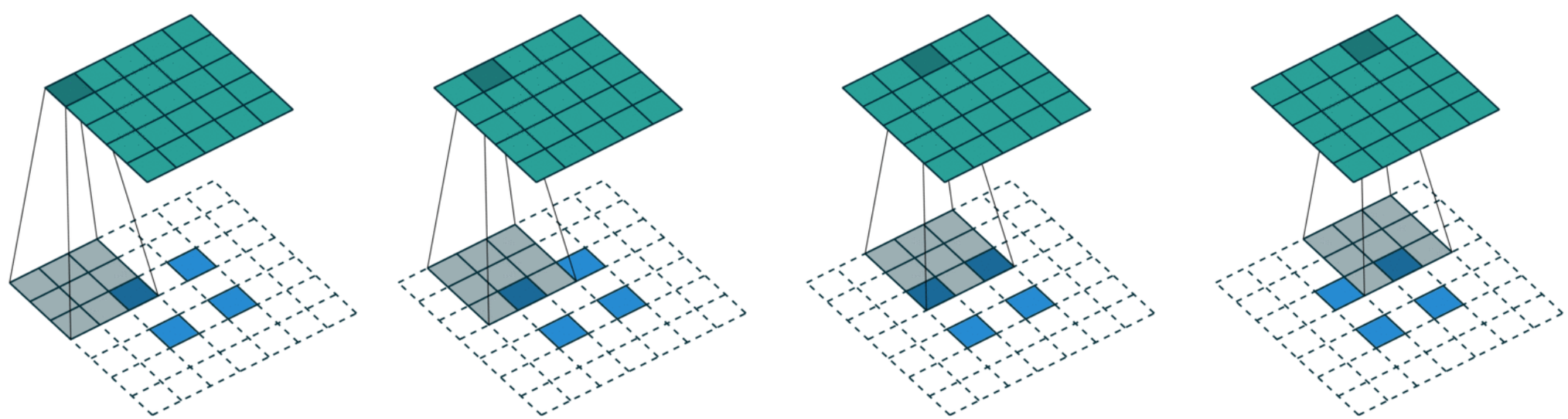


A Conv2DTranspose with 3x3 kernel and stride of 2x2 applied to a 2x2 input to give a 5x5 output.
(https://medium.com/apache-mxnet/transposed-convolutions-explained-with-ms-excel-52d13030c7e8)

Transposed Convolution (emulated with direct convolution):

output

input



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

Regular Convolution: (stride = 1)

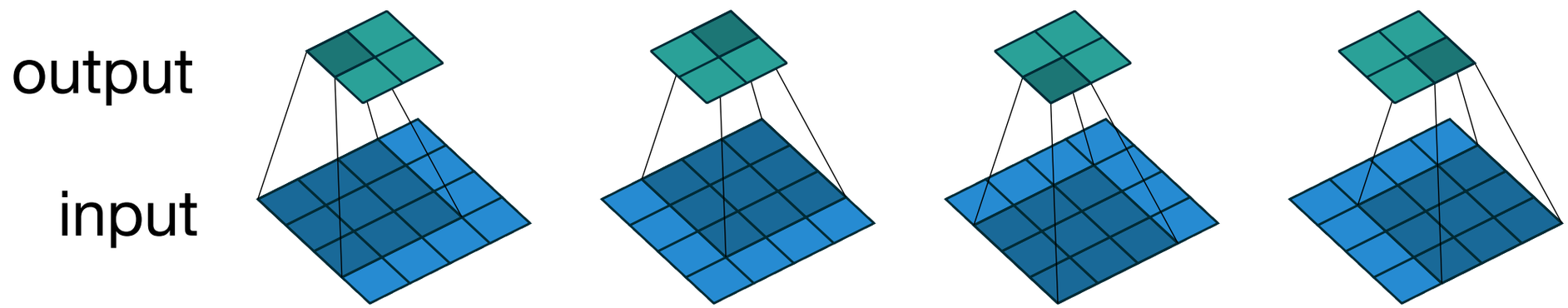
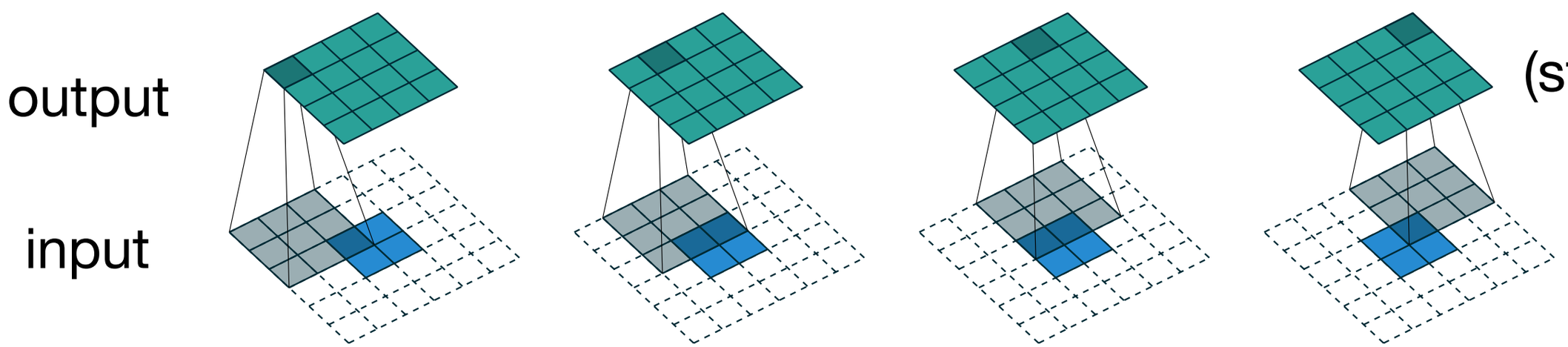


Figure 2.1: (No padding, unit strides) Convolving a 3×3 kernel over a 4×4 input using unit strides (i.e., $i = 4$, $k = 3$, $s = 1$ and $p = 0$).

Transposed Convolution (emulated with direct convolution):



(stride = 1)

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

Transposed Convolution

```python
import torch
```

```python
torch.manual_seed(123)
a = torch.rand(4).view(1, 1, 2, 2)

conv_t = torch.nn.ConvTranspose2d(in_channels=1,
                                  out_channels=1,
                                  kernel_size=(3, 3),
                                  padding=0,
                                  stride=1)

# output = s(n-1)+k-2p = 1*(2-1)+3-2*0 = 4
conv_t(a)
```

```
tensor([[[[-0.2863, -0.2766, -0.1478, -0.3274],
          [-0.3522, -0.5356, -0.1591, -0.2911],
          [-0.3054, -0.4644, -0.3286, -0.2444],
          [-0.2332, -0.2557, -0.1876, -0.3970]]]],
       grad_fn=<ThnnConvTranspose2DBackward>)
```

$$\text{output} = s(n-1) + k - 2p$$

?

```python
torch.manual_seed(123)
a = torch.rand(16).view(1, 1, 4, 4)

conv_t = torch.nn.ConvTranspose2d(in_channels=1,
                                  out_channels=1,
                                  kernel_size=(3, 3),
                                  padding=0,
                                  stride=1)

# output = s(n-1)+k-2p = 1*(4-1)+3-2*0 = 6
conv_t(a).size()
```

```
torch.Size([1, 1, 6, 6])
```

```python
torch.manual_seed(123)
a = torch.rand(64).view(1, 1, 8, 8)

conv_t = torch.nn.ConvTranspose2d(in_channels=1,
                                  out_channels=1,
                                  kernel_size=(3, 3),
                                  padding=0,
                                  stride=1)

# output = s(n-1)+k-2p = 1*(8-1)+3-2*0 = 10
conv_t(a).size()
```

```
torch.Size([1, 1, 10, 10])
```

Deconvolution and Checkerboard Artifacts

AUGUSTUS ODENA
Google Brain

VINCENT DUMOULIN
Université de Montréal

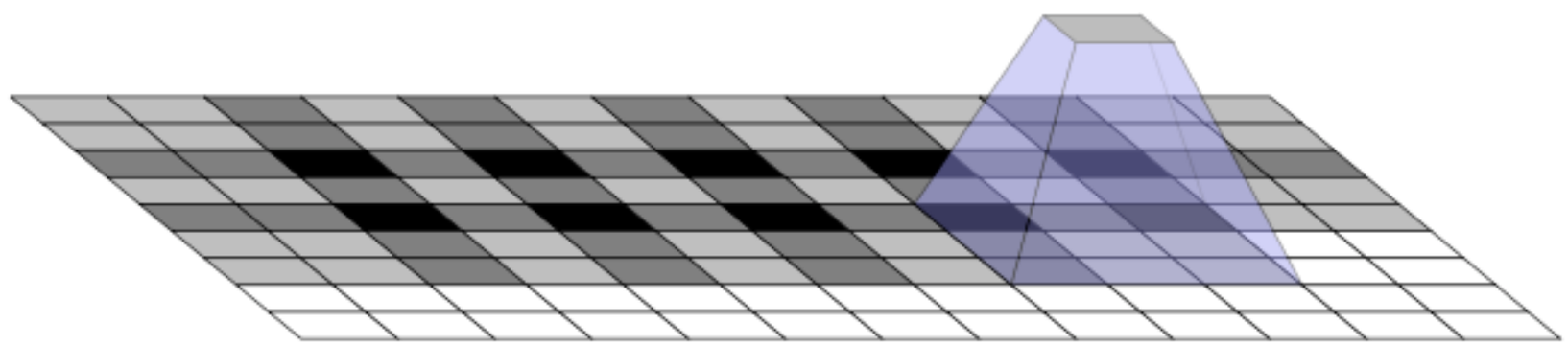
CHRIS OLAH
Google Brain

Oct. 17
2016

Citation:
Odena, et al., 2016

https://distill.pub/2016/deconv-checkerboard/

A good interactive article highlighting the dangers of transposed conv.

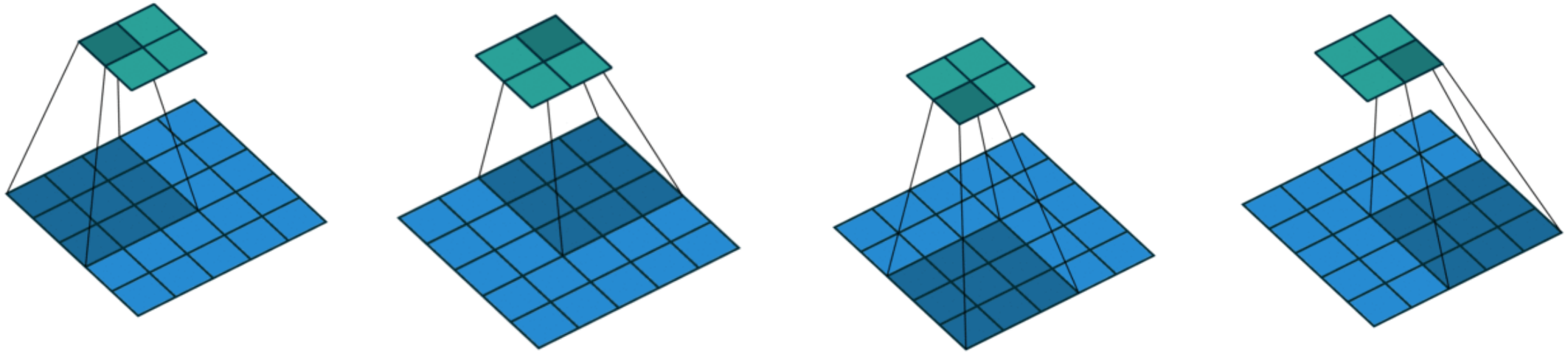


In short, recommends replacing transposed conv.
by upsampling (interpolation) followed by regular convolution

Regular Convolution:

output

input

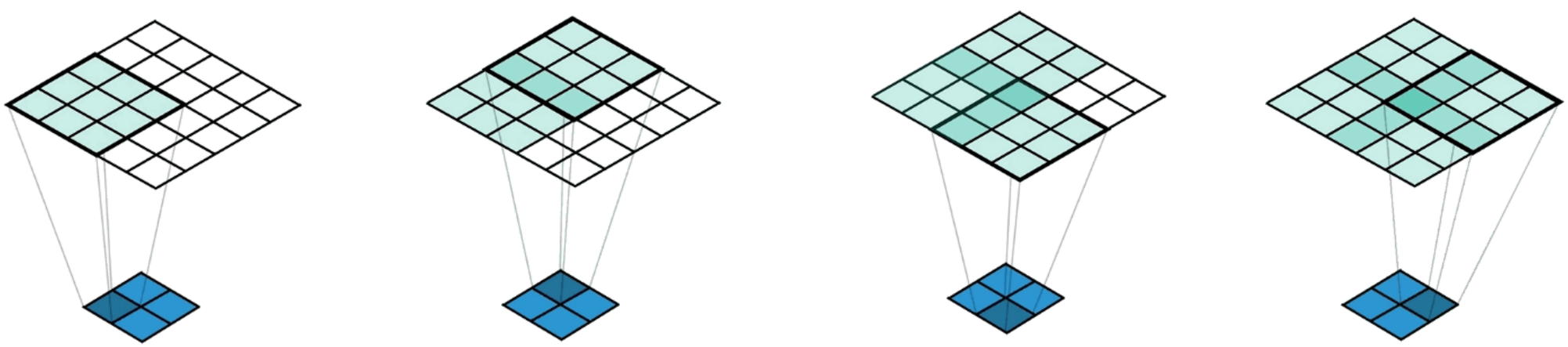


Dumoulin, Vincent, and Francesco Visin. "A guide to convolution arithmetic for deep learning." *arXiv preprint arXiv:1603.07285* (2016).

Transposed Convolution (stride = 2)

output

input



A Conv2DTranspose with 3x3 kernel and stride of 2x2 applied to a 2x2 input to give a 5x5 output. (https://medium.com/apache-mxnet/transposed-convolutions-explained-with-ms-excel-52d13030c7e8)

Implementing a Convolutional Autoencoder for Handwritten Digits

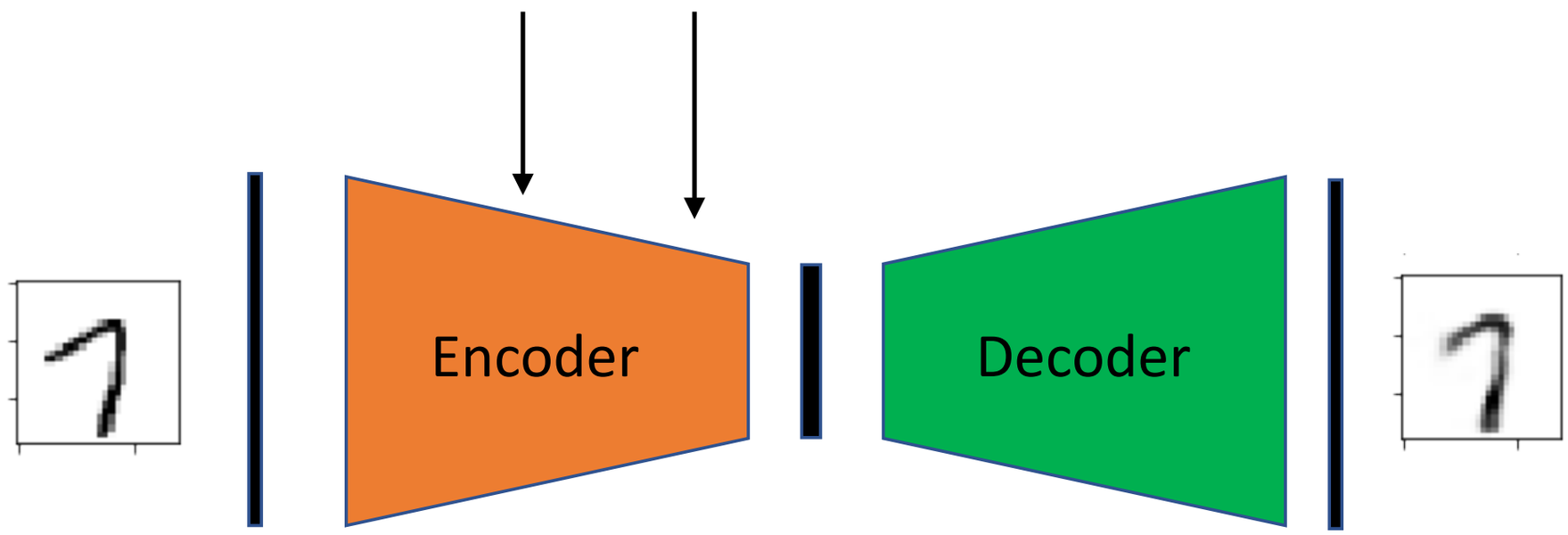
1. Dimensionality Reduction
2. Fully-connected Autoencoders
3. Convolutional Autoencoders
4. **A Convolutional Autoencoder in PyTorch**
5. Other Types of Autoencoders

Beyond "Regular" Fully-Connected or Convolutional Autoencoders

1. Dimensionality Reduction
2. Fully-connected Autoencoders
3. Convolutional Autoencoders
4. A Convolutional Autoencoder in PyTorch
5. **Other Types of Autoencoders**

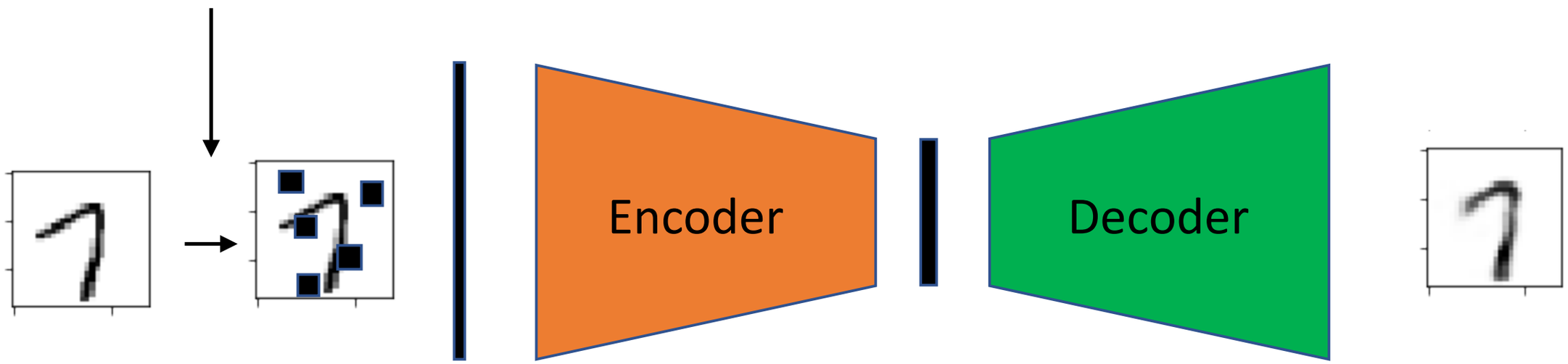
Autoencoders and Dropout

Add dropout layers to force networks to learn redundant features



Denoising Autoencoder

Add dropout after the input, or add noise to the input to learn to denoise images

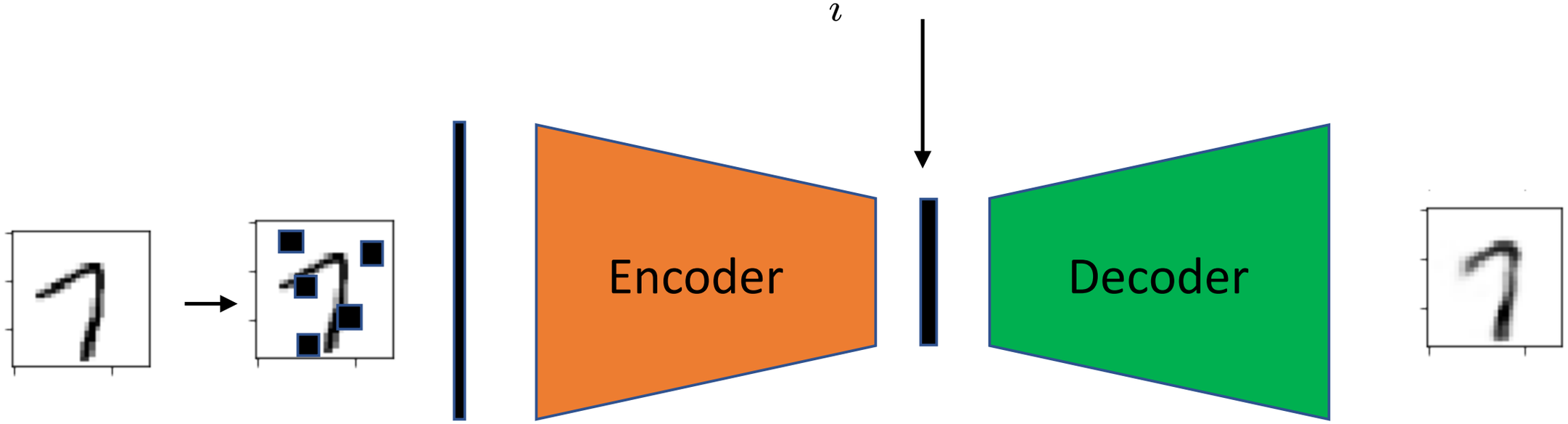


Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. (2008, July). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning* (pp. 1096-1103). ACM.

Sparse Autoencoder

Add L1 penalty to the loss to learn sparse feature representations

$$\sum_i |Enc_i(\mathbf{x})|$$



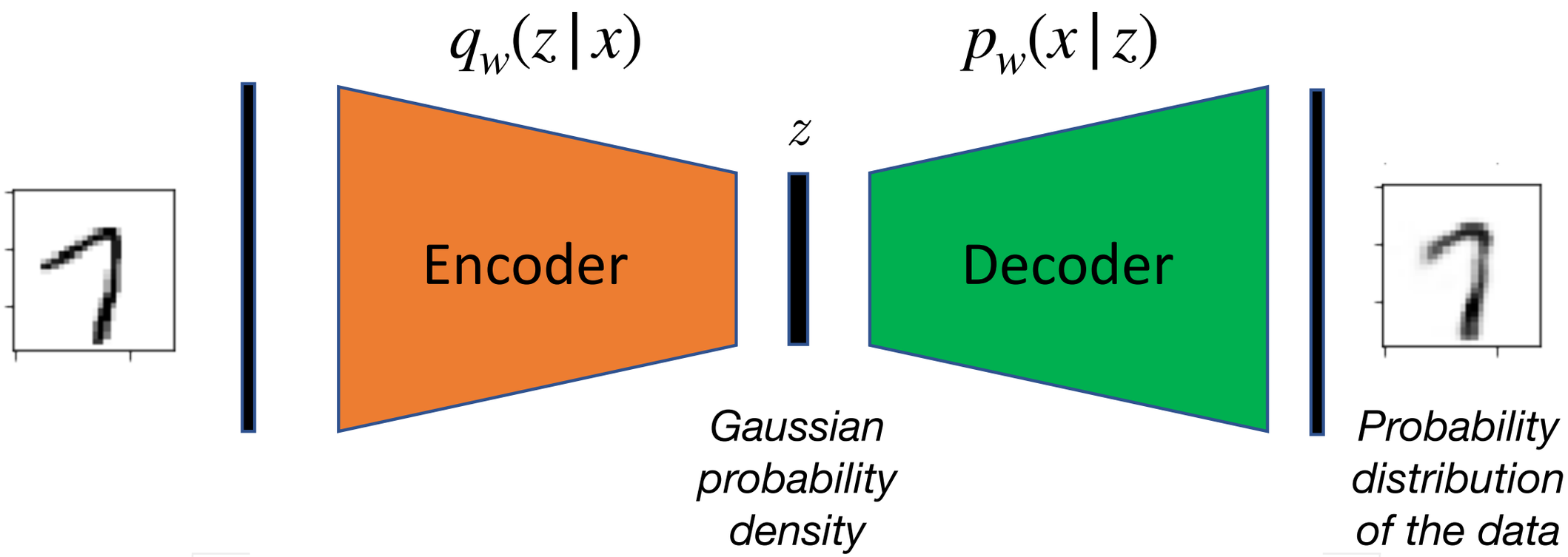
$$\mathcal{L} = ||\mathbf{x} - Dec(Enc(\mathbf{x}))||_2^2 + \sum_i |Enc_i(\mathbf{x})|$$

Variational Autoencoder

$$L^{[i]} = -\mathbb{E}_{z \sim q_w(z \mid x^{[i]})} \left[\log p_w \left(x^{[i]} \mid z \right) \right] + \mathbf{KL} \left(q_w \left(z \mid x^{[i]} \right) \| p(z) \right)$$

Expected neg. log likelihood term; wrt to encoder distribution

Kullback-Leibler divergence term where $p(z) = \mathcal{N}\left(\mu = 0, \sigma^2 = 1\right)$



Kingma, D. P., & Welling, M. (2013). Auto-encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*.