Lecture 15

A Short Introduction to Autoencoders

STAT 479: Deep Learning, Spring 2019

Sebastian Raschka

http://stat.wisc.edu/~sraschka/teaching/stat479-ss2019/

Unsupervised Learning

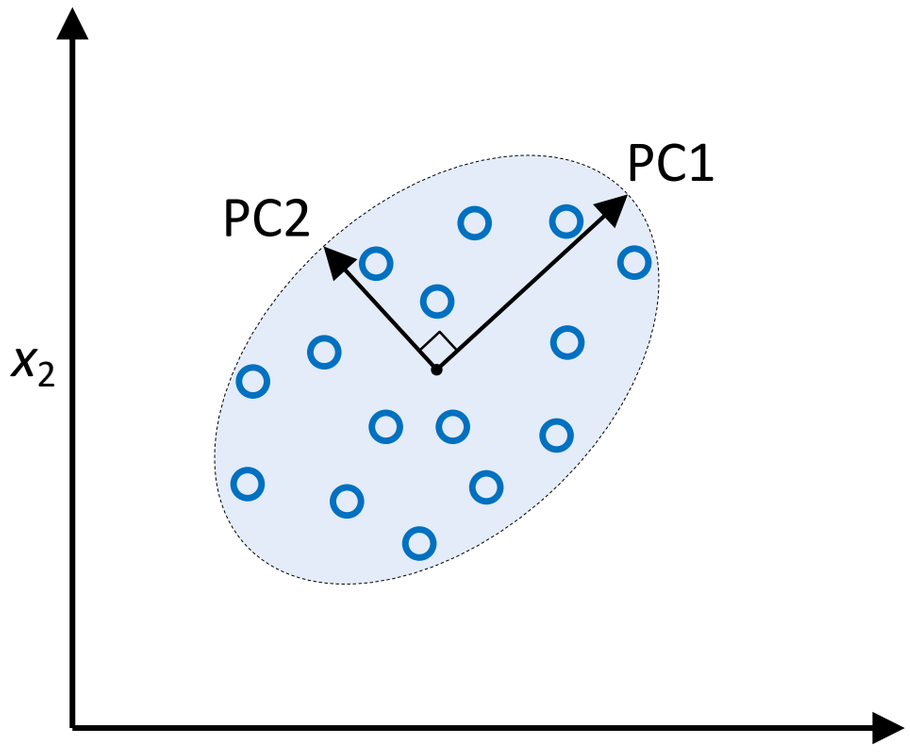
Working with datasets *without* considering a/the *target* variable

Some Applications and Goals:

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

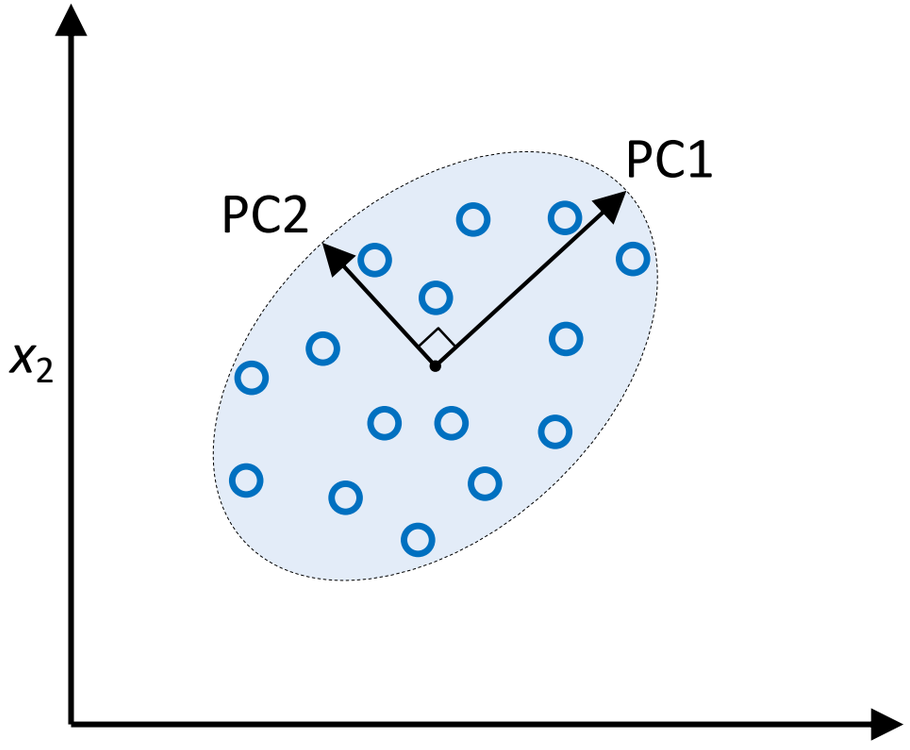
Principal Component Analysis (PCA)

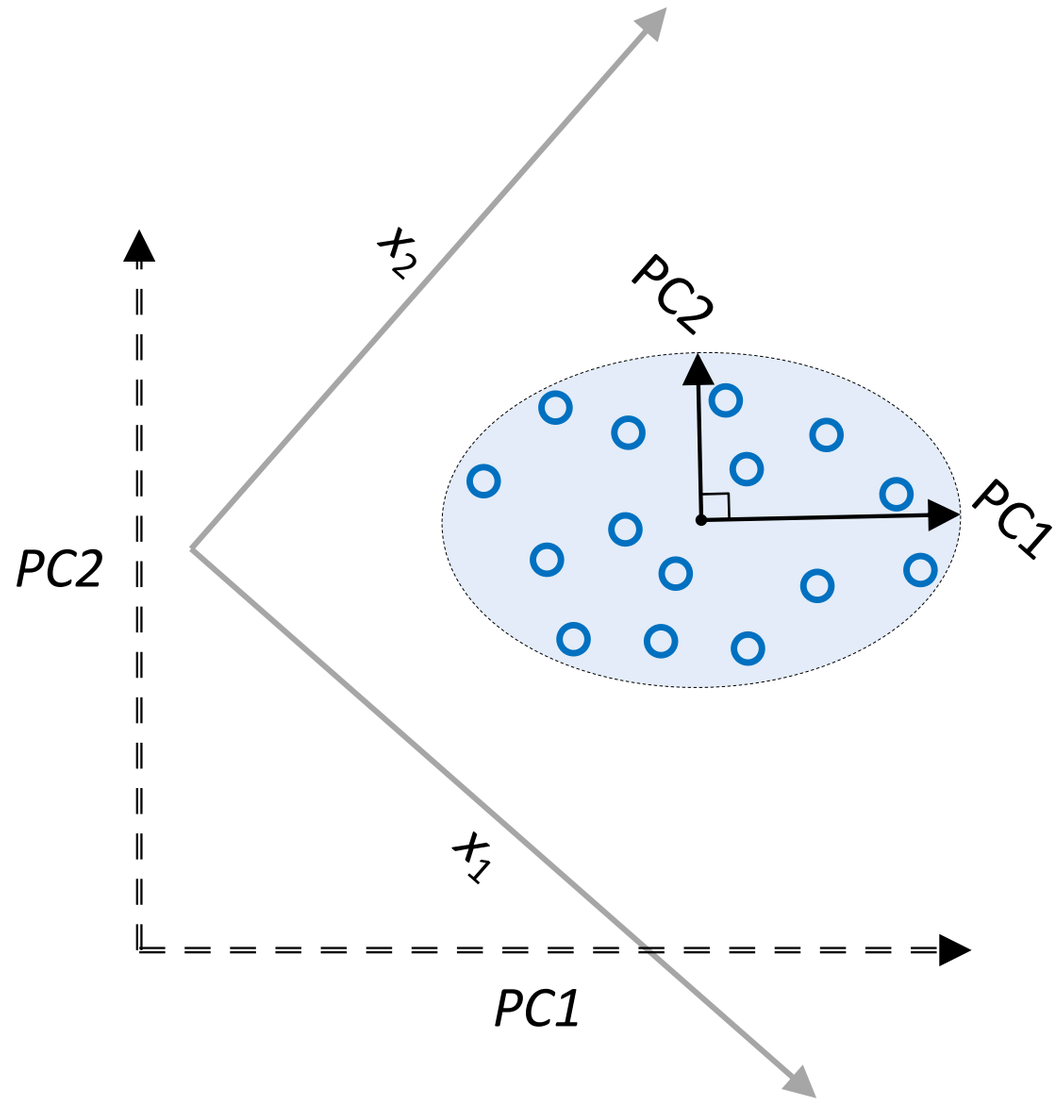
1) Find directions of maximum variance



Principal Component Analysis (PCA)

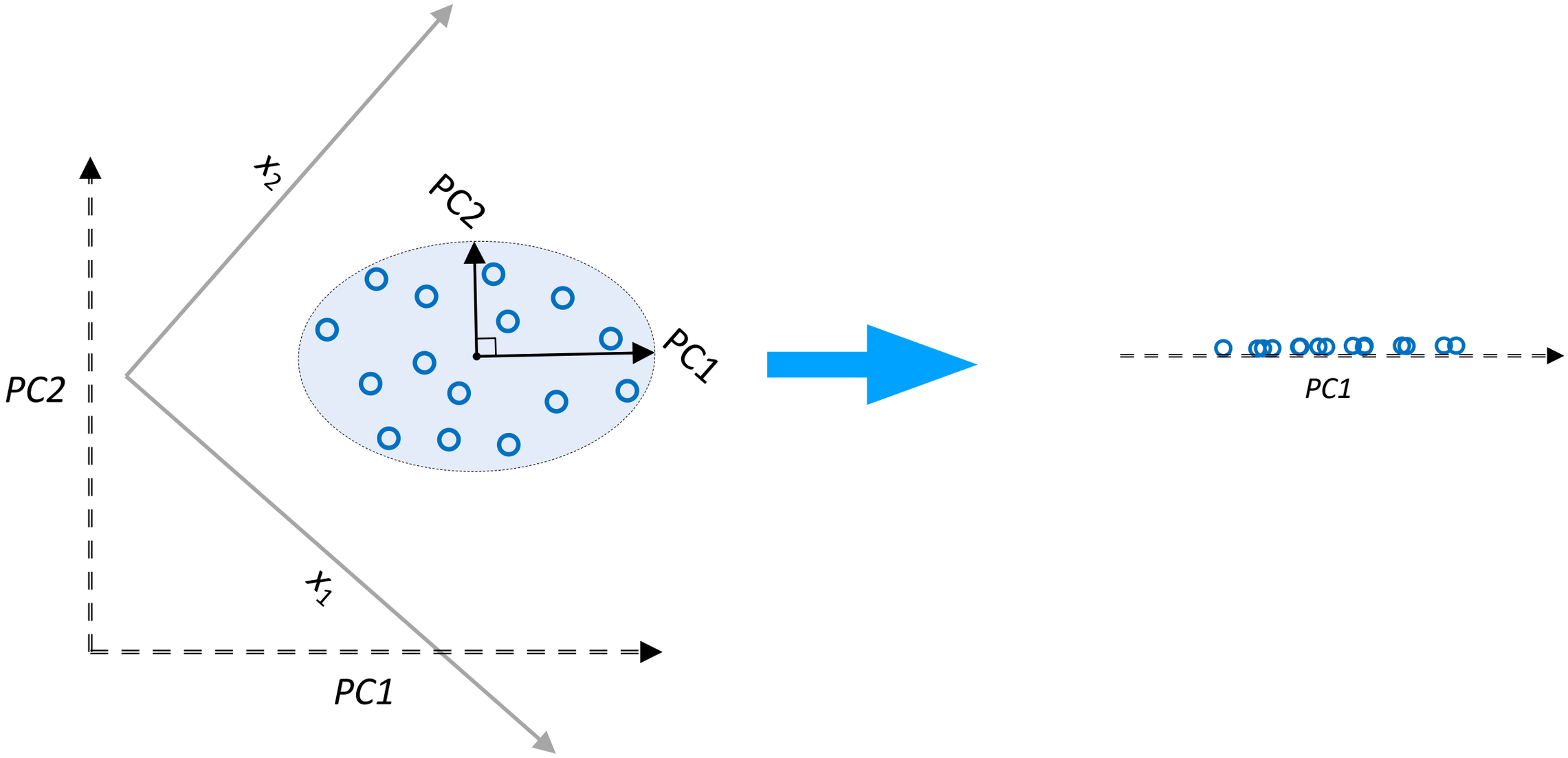
2) Transform features onto directions of maximum variance





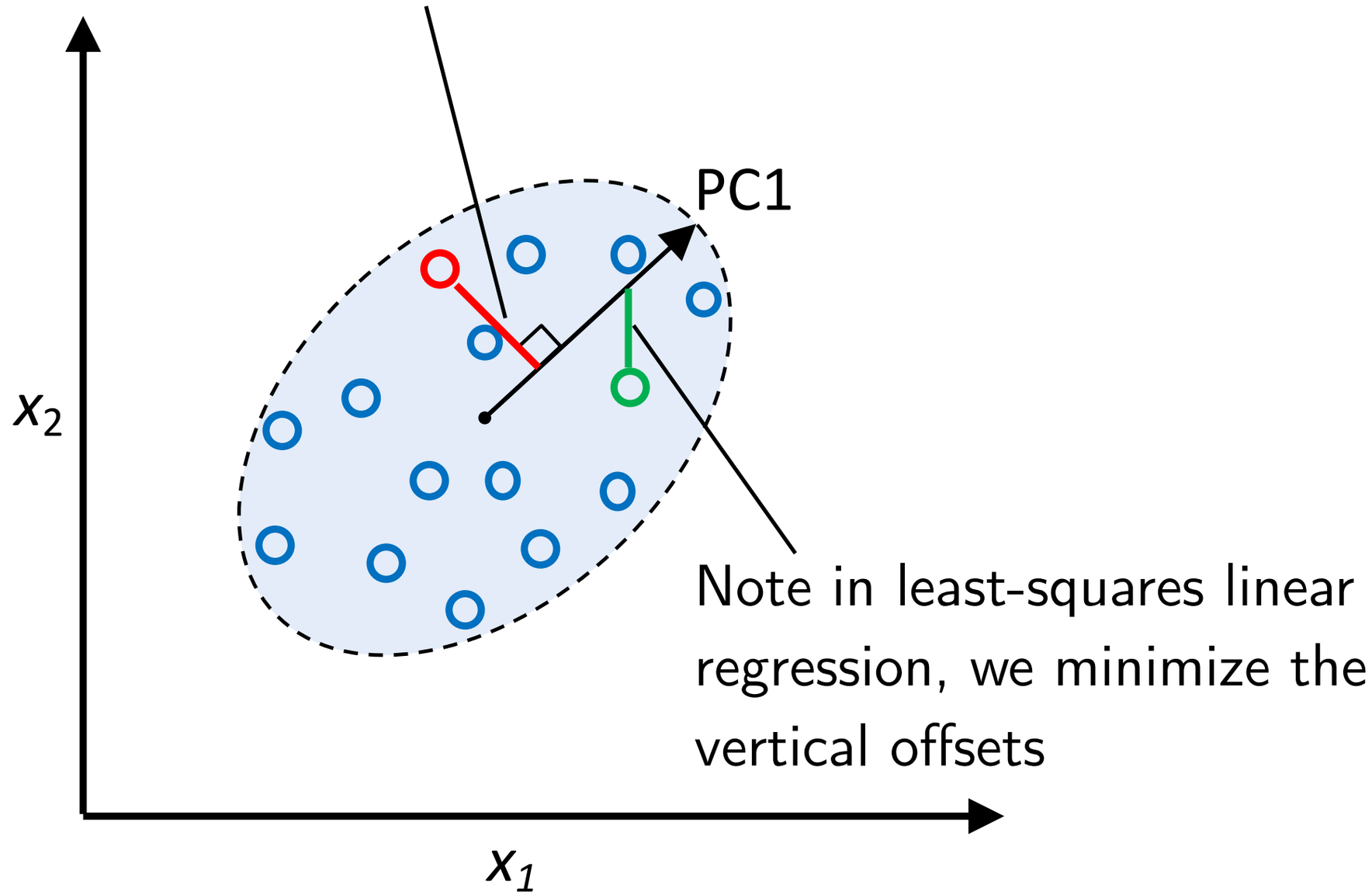
Principal Component Analysis (PCA)

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

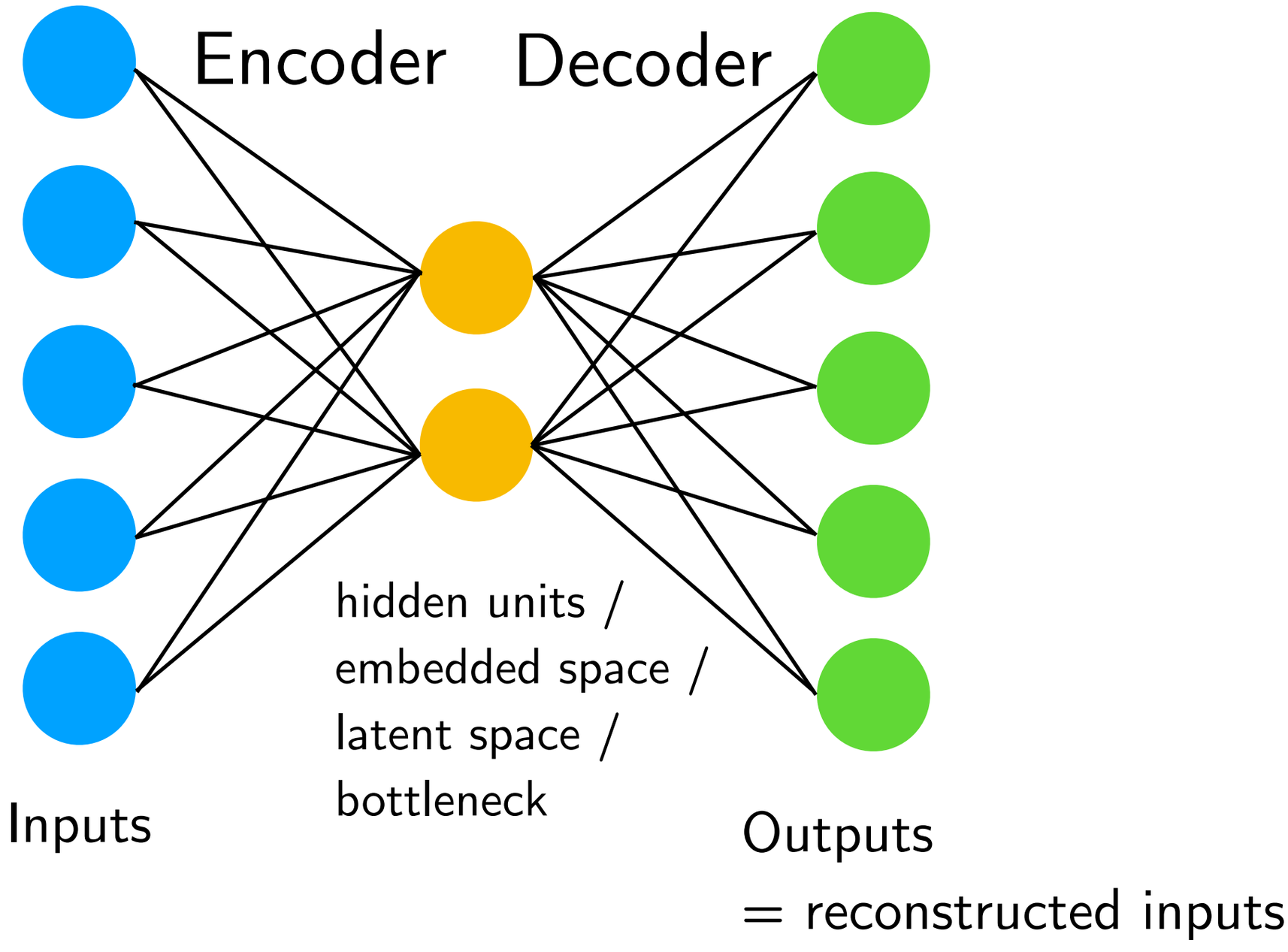


Principal Component Analysis (PCA)

Another view on PCA is minimizing the squared offsets

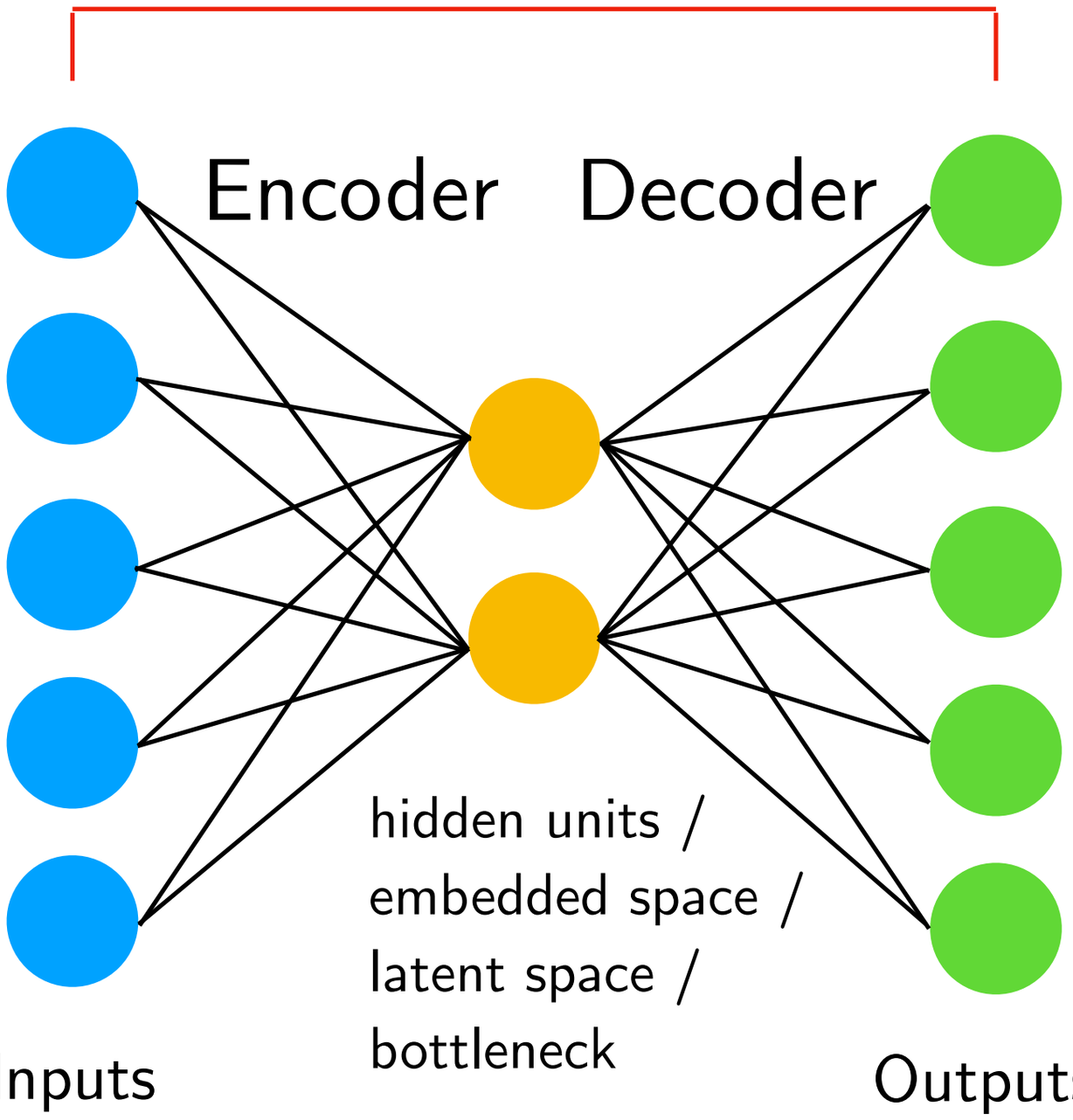


A Basic Fully-Connected (Multilayer-Perceptron) Autoencoder



A Basic Fully-Connected (Multilayer-Perceptron) Autoencoder

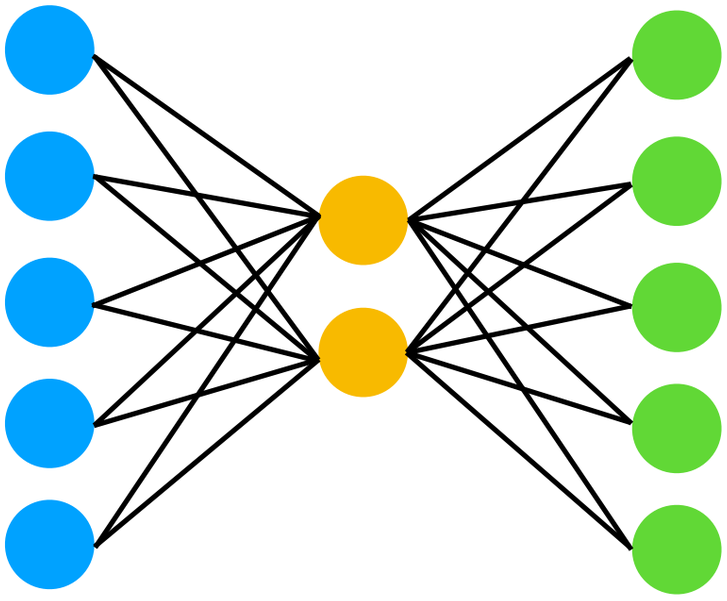
$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||_2^2 = \sum_i (x_i - x_i')^2$$



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

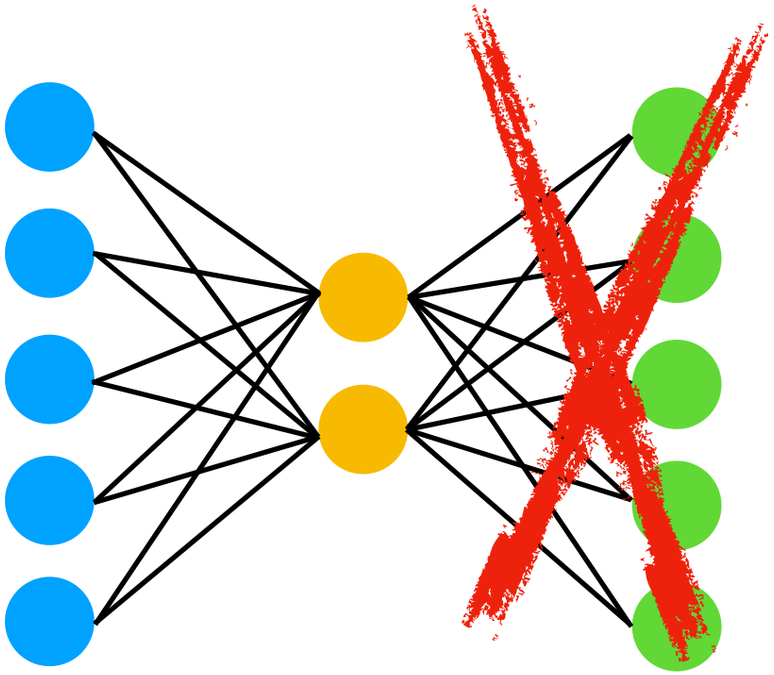


Question:

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, ...

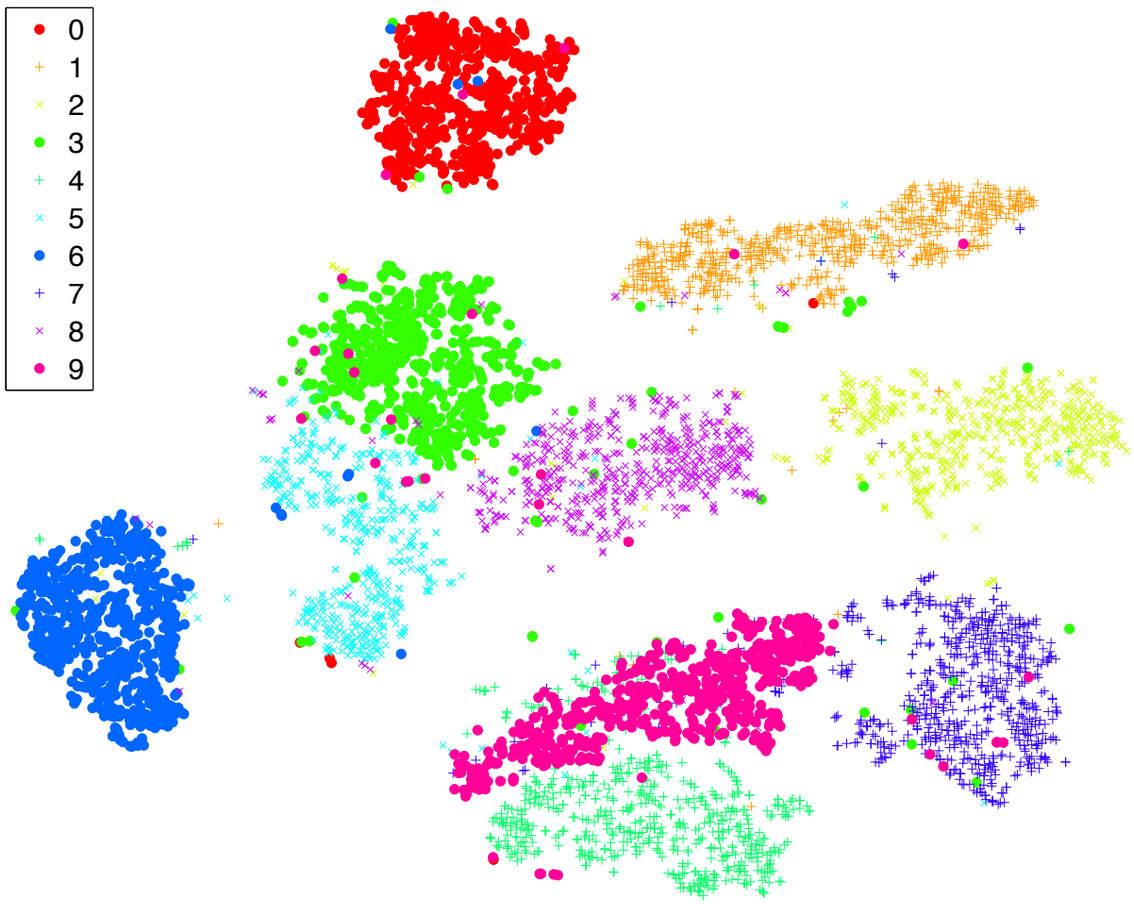
Or, similar to transfer learning, train autoencoder on large image dataset, then fine tune encoder part on your own, smaller datset and/or provide your own output (classification) layer

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

t-Distributed Stochastic Neighbor Embedding (t-SNE)

Another way to learn embeddings ...

(t-SNE is only meant for visualization not for preparing datasets!)



(a) Visualization by t-SNE.

Note that MNIST has
28 x 28 = 784 dimensions

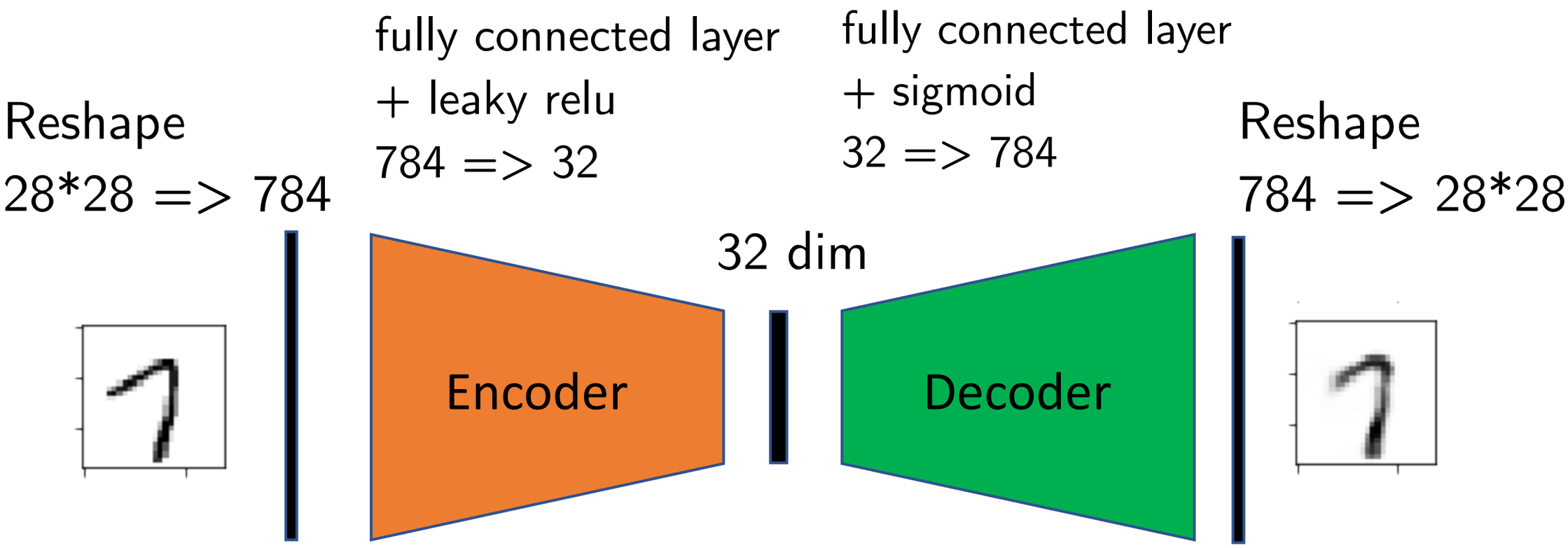
Shown are 6000 images from MNIST projected in 2D

For details, see

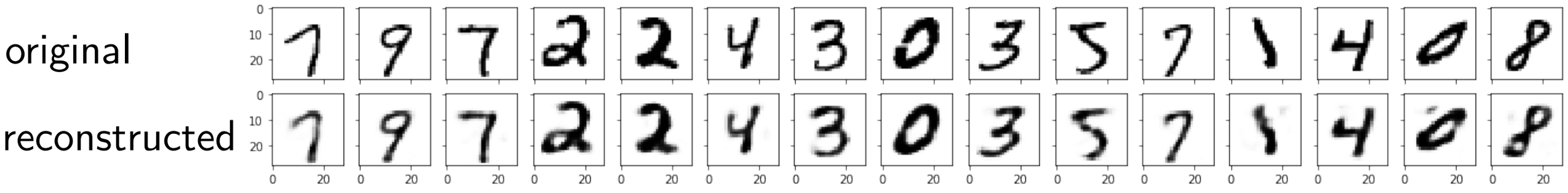
https://github.com/rasbt/stat479-machine-learning-fs18/blob/master/14_feat-extract/14_feat-extract_slides.pdf

Maaten, L. V. D., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*, *9*(Nov), 2579-2605.

A Simple Autoencoder



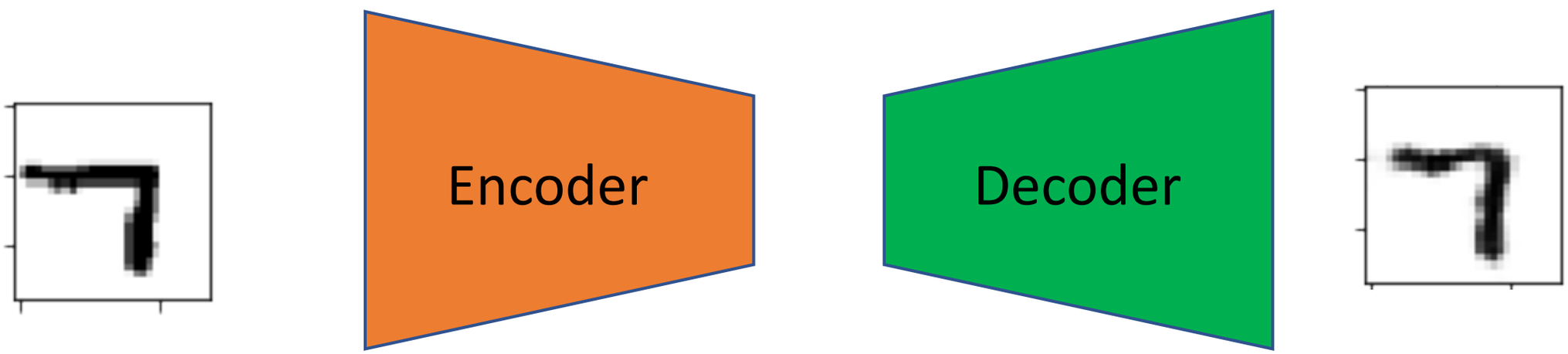
https://github.com/rasbt/stat479-deep-learning-ss19/blob/master/L15_autoencoder/code/ae-simple.ipynb



A Convolutional Autoencoder

1 or more
convolutional layers

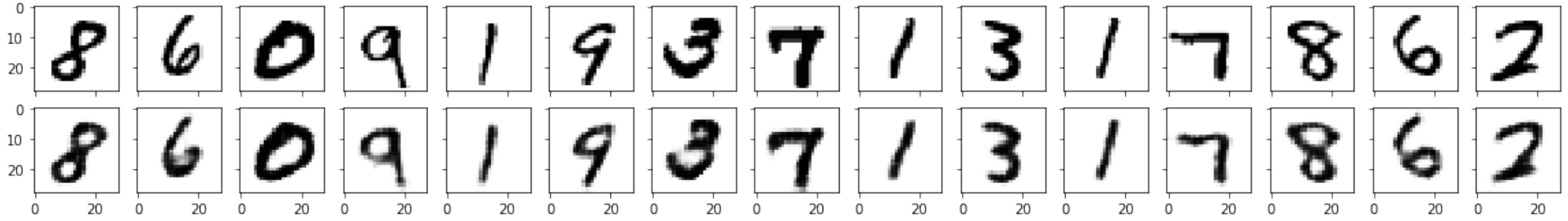
1 or more
"de"convolutional layers



https://github.com/rasbt/stat479-deep-learning-ss19/blob/master/L15_autoencoder/code/ae-conv.ipynb

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" or sometimes (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 term for that

Regular Convolution:

output

input

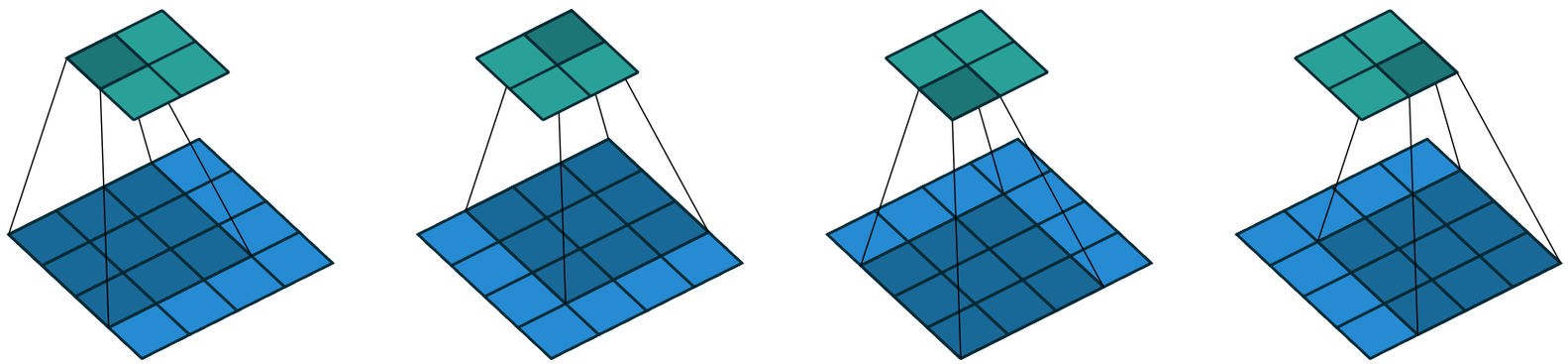
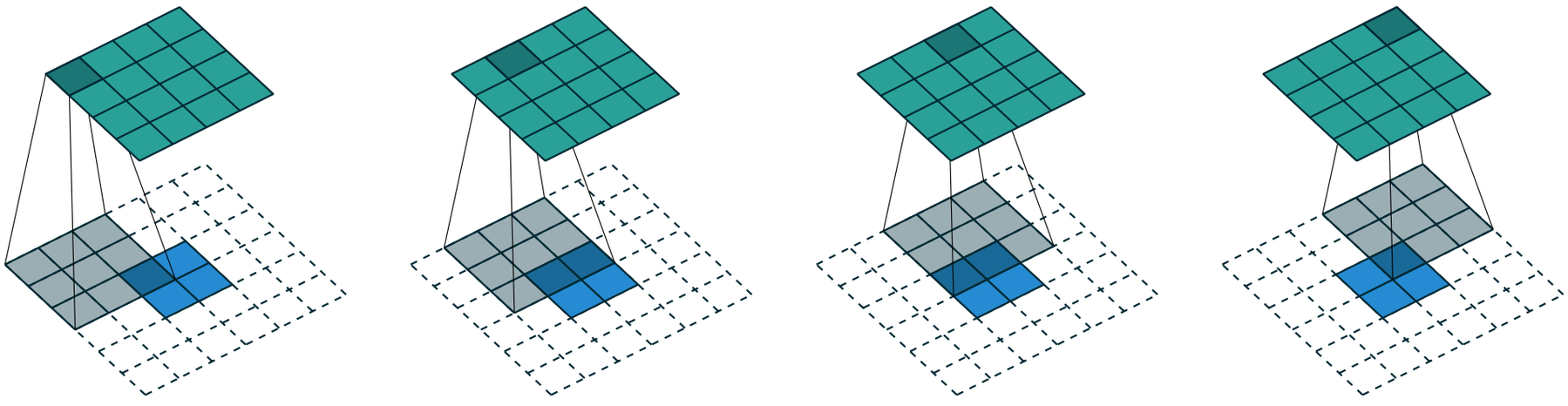


Figure 2.1: (No padding, unit strides) Convolving a 3×3 kernel over a 4×4 input using unit strides (i.e., $i = 4$, $k = 3$, $s = 1$ and $p = 0$).

Transposed Convolution (emulated with direct convolution):

output

input



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

Transposed Convolution

```
import torch
```

```
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>)
```

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

?

```
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])
```

```
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])
```

Transposed Convolution

strides: in transposed convolutions, we stride over the output;

hence, larger strides will result in larger outputs
(opposite to regular convolutions)

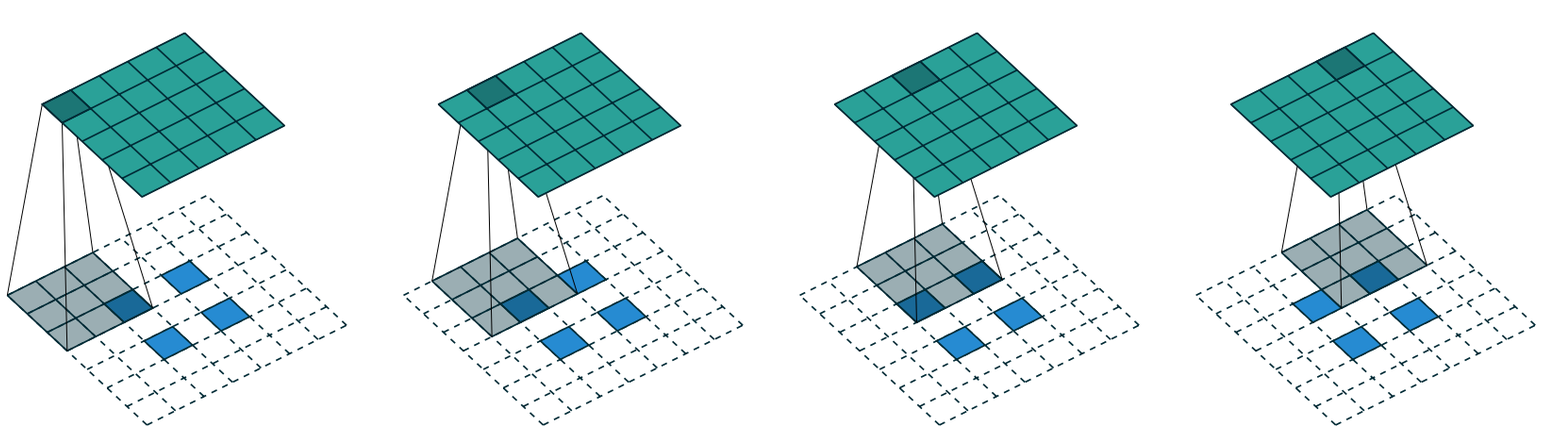
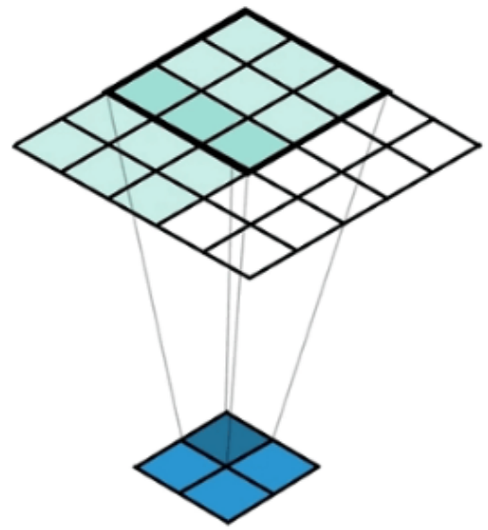


Figure 4.5: The transpose of convolving a 3×3 kernel over a 5×5 input using 2×2 strides (i.e., $i = 5$, $k = 3$, $s = 2$ and $p = 0$). It is equivalent to convolving a 3×3 kernel over a 2×2 input (with 1 zero inserted between inputs) padded with a 2×2 border of zeros using unit strides (i.e., $i' = 2$, $\tilde{i}' = 3$, $k' = k$, $s' = 1$ and $p' = 2$).

You can think of it as this:



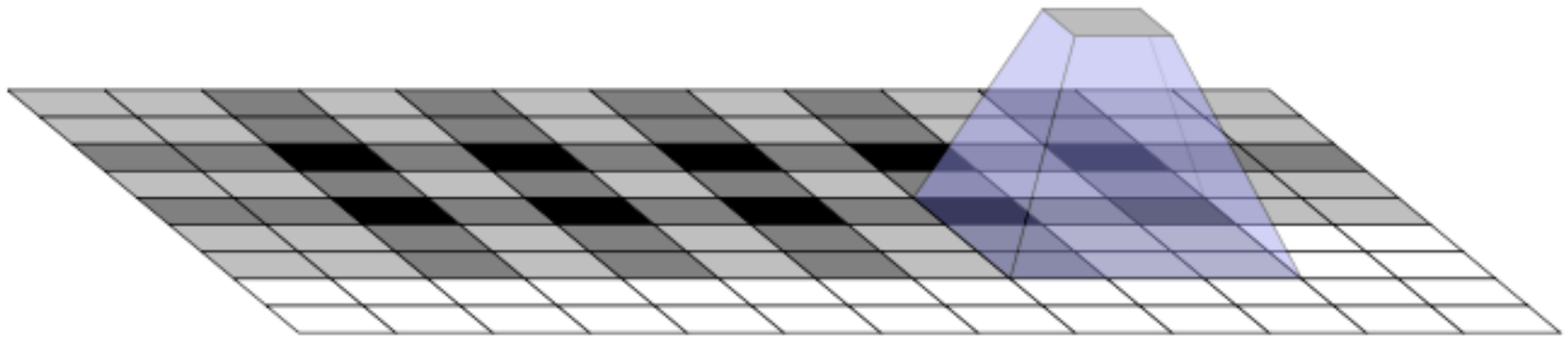
Dumoulin, Vincent, and Francesco Visin. "A guide to convolution arithmetic for deep learning." *arXiv preprint arXiv:1603.07285* (2016).

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.

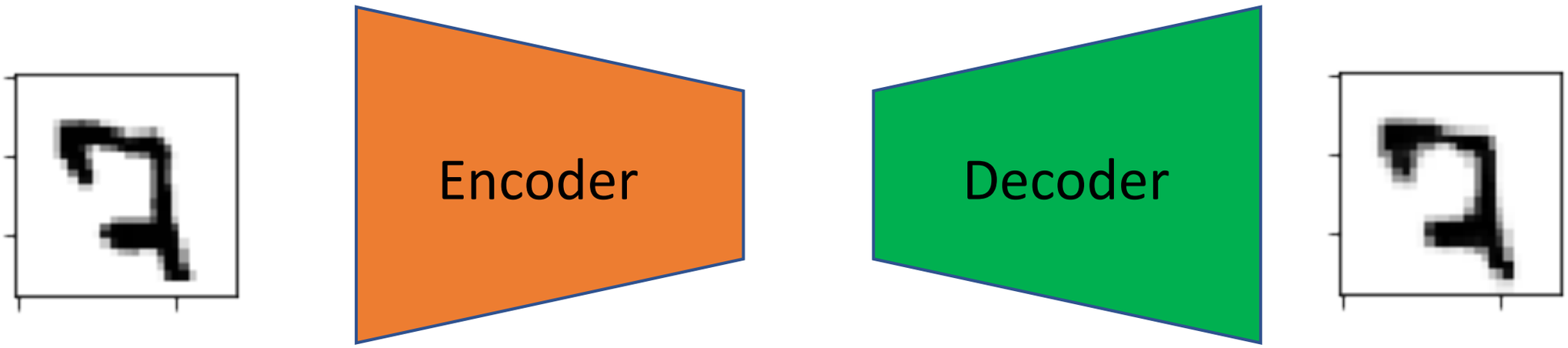


In short, it is recommended to replace transposed conv.
by upsampling (interpolation) followed by regular convolution

A Convolutional Autoencoder with Nearest Neighbor Interpolation

Convolutional layers
+ maxpooling

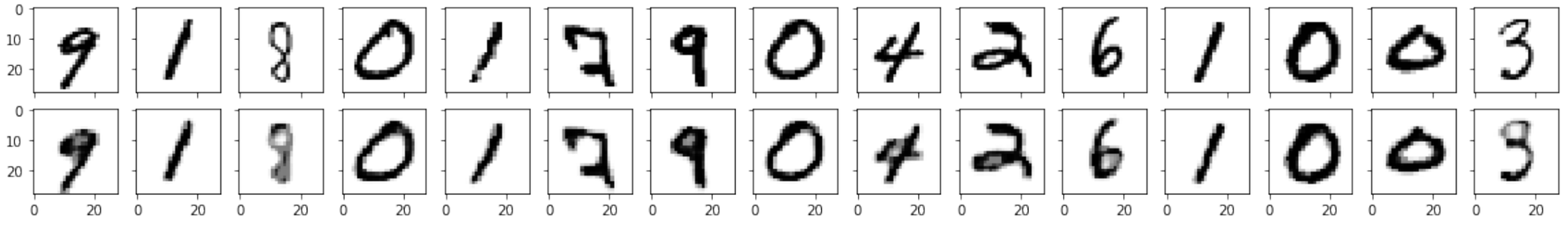
Convolutional layers
+ interpolation



https://github.com/rasbt/stat479-deep-learning-ss19/blob/master/L15_autoencoder/code/ae-interpolate.ipynb

original

reconstructed



Transposed Convolution

padding: in transposed convolutions, we pad the output; hence, larger padding will result in smaller output maps
(opposite to regular convolutions)

for a 2x2 input (previous slide) the output would be 3x3

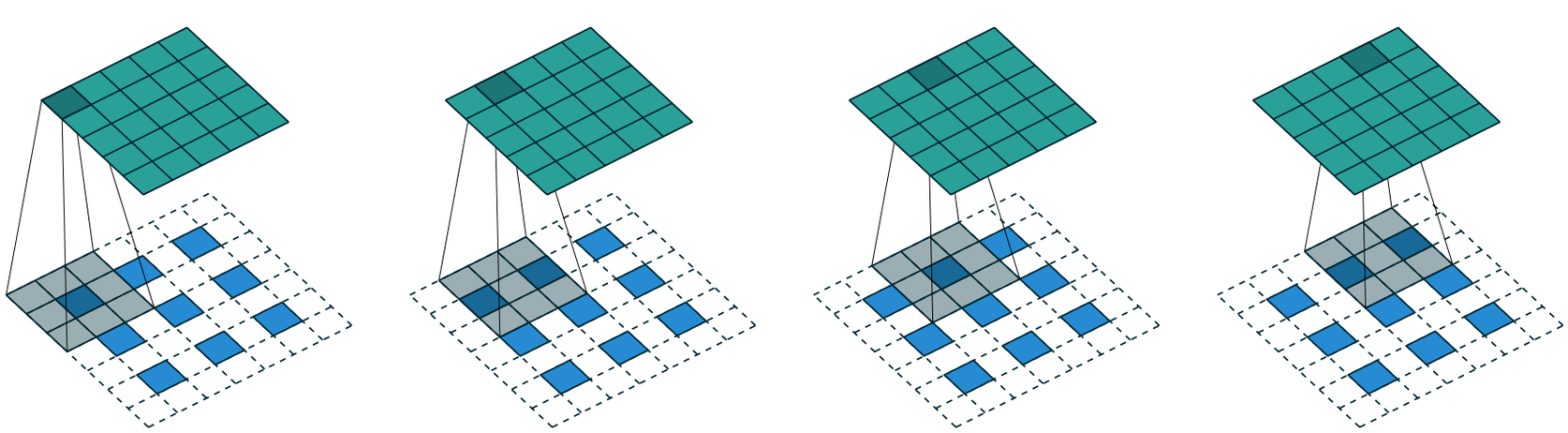
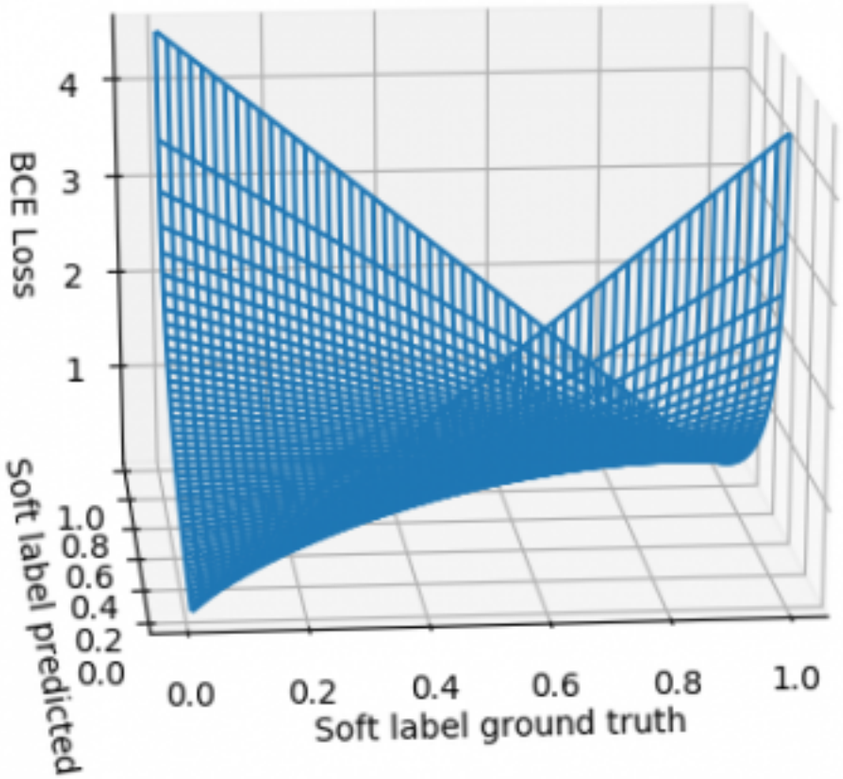


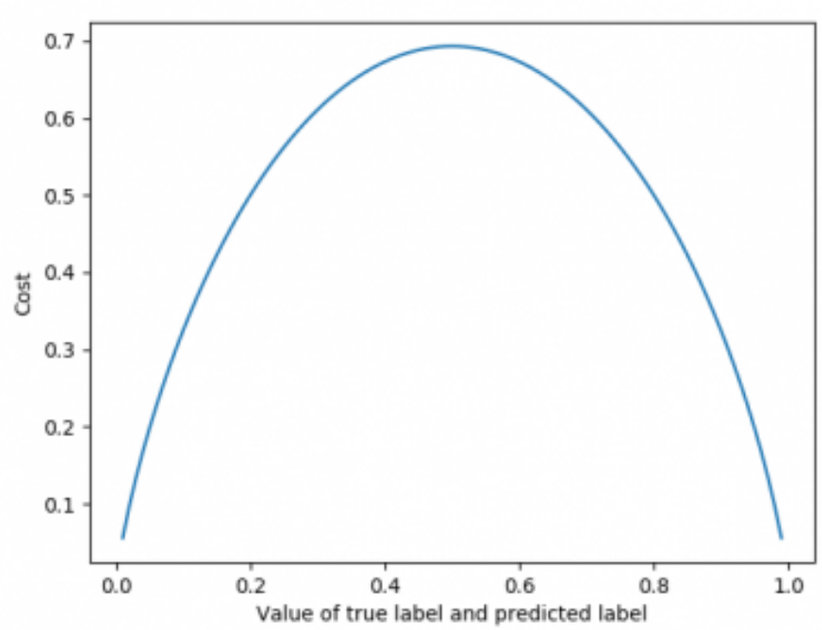
Figure 4.6: The transpose of convolving a 3×3 kernel over a 5×5 input padded with a 1×1 border of zeros using 2×2 strides (i.e., $i = 5$, $k = 3$, $s = 2$ and $p = 1$). It is equivalent to convolving a 3×3 kernel over a 3×3 input (with 1 zero inserted between inputs) padded with a 1×1 border of zeros using unit strides (i.e., $i' = 3$, $\tilde{i}' = 5$, $k' = k$, $s' = 1$ and $p' = 1$).

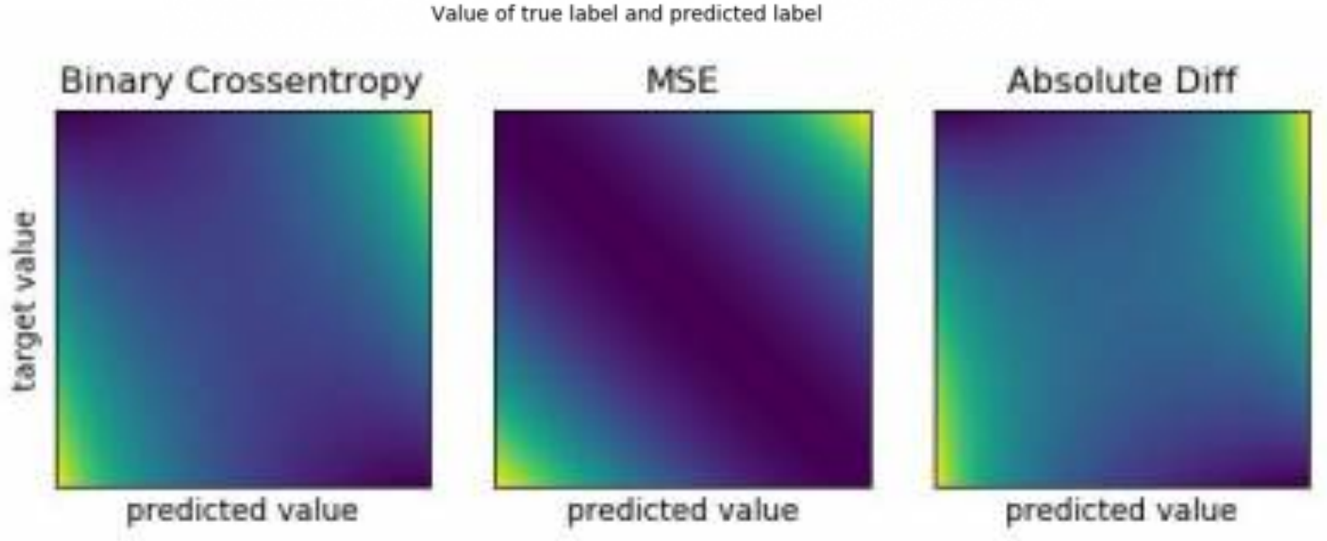
Dumoulin, Vincent, and Francesco Visin. "A guide to convolution arithmetic for deep learning." *arXiv preprint arXiv:1603.07285* (2016).

Side-note: Often you will also see binary cross entropy used as a loss function instead of MSE

I find BCE for the autoencoder reconstruction loss problematic,
but it does seem to work well in practice ... (because it is not symmetric)

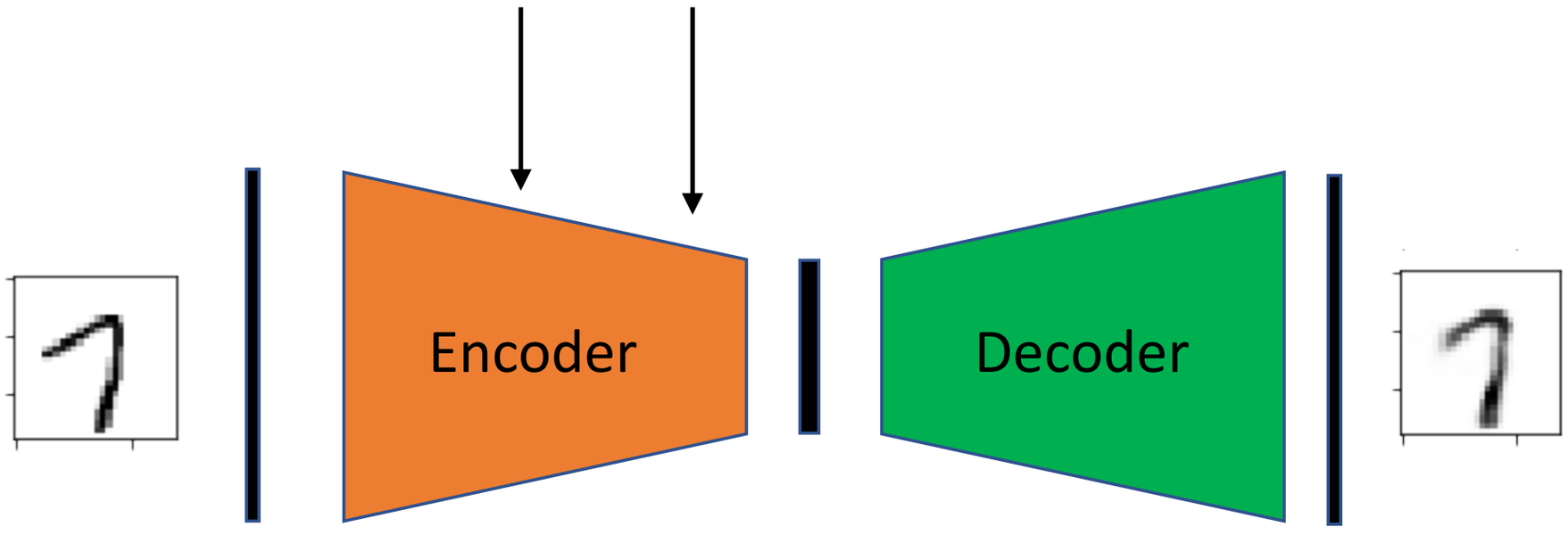






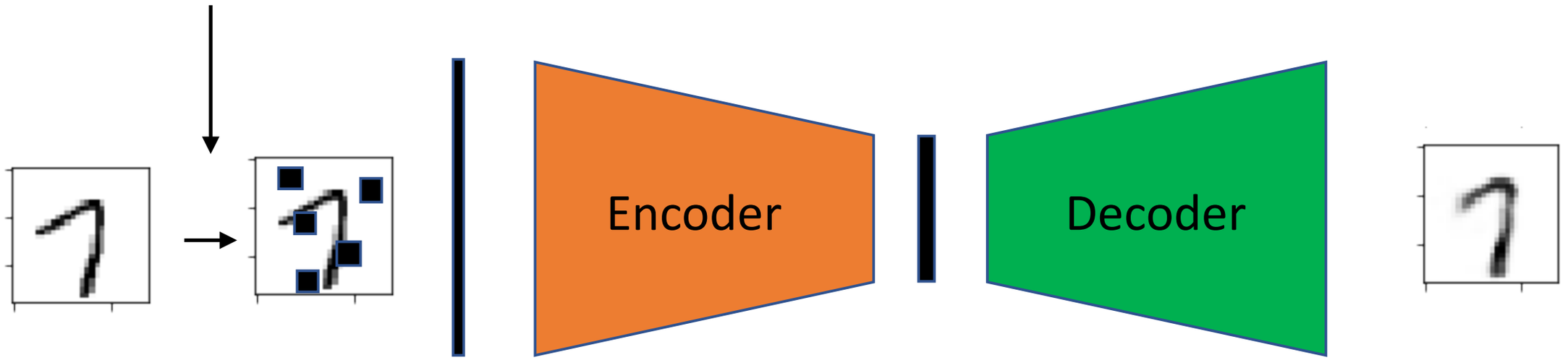
Autoencoders and Dropout

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

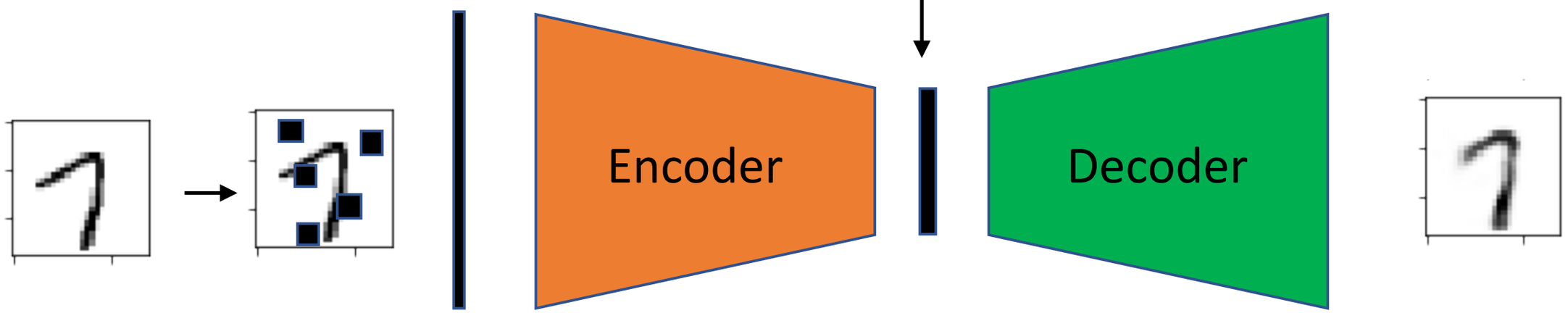


Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. (2008, July). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning* (pp. 1096-1103). ACM.

Sparse Autoencoder

Add L1 penalty to the loss to learn sparse feature representations

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



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