Autoencoders

CS8004: Deep Learning and Applications

Slides based on lecture slides of Dr Guy Golan

Unsupervised Learning

- The goal is to discover (important) intrinsic features of the data to obtain subgroups among the observed data.
- Data samples in a subgroup are expected to have greater similarity or degree of closeness than those falling in different subgroups.
- Examples include
  - X-Ray images of COVID-19 patients, other Pneumonia affected patients, and normal persons.
  - Customers characterized and grouped by their browsing and purchase histories.
  - Organising your photo files in groups containing road scenes, selfies, group photos of family and friends, and natural scenes.

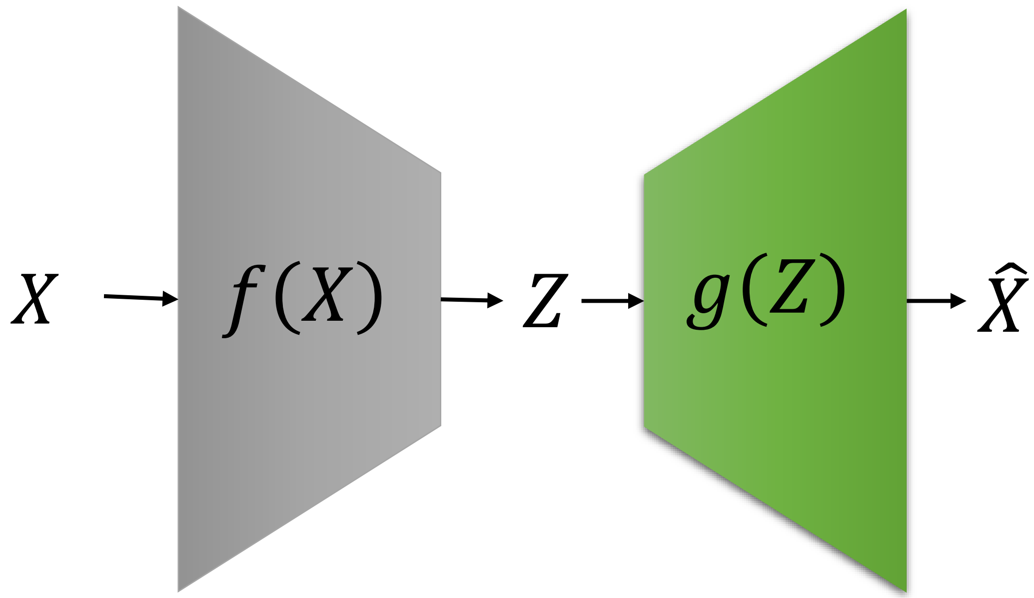
Why Unsupervised learning ?

- Most of the data is unlabelled.
- The contents of a database may not be known. It may help gain insight into data characteristics.
- Examples:
  - News – fake or authentic
  - sentiments expressed through messages on a Twitter account, positive, negative, provocative or neutral ?
  - Similar image retrieval from a large set of images.
- For data labelling human intervention is needed, which is time consuming, costly and is prone to errors.

Neural Networks for Unsupervised Learning

- Autoencoders: A basic autoencoder is a neural network that is trained to reproduce its input.
- Main objective is to capture the features of its input in a code, that can be used to exactly reproduce the original input.

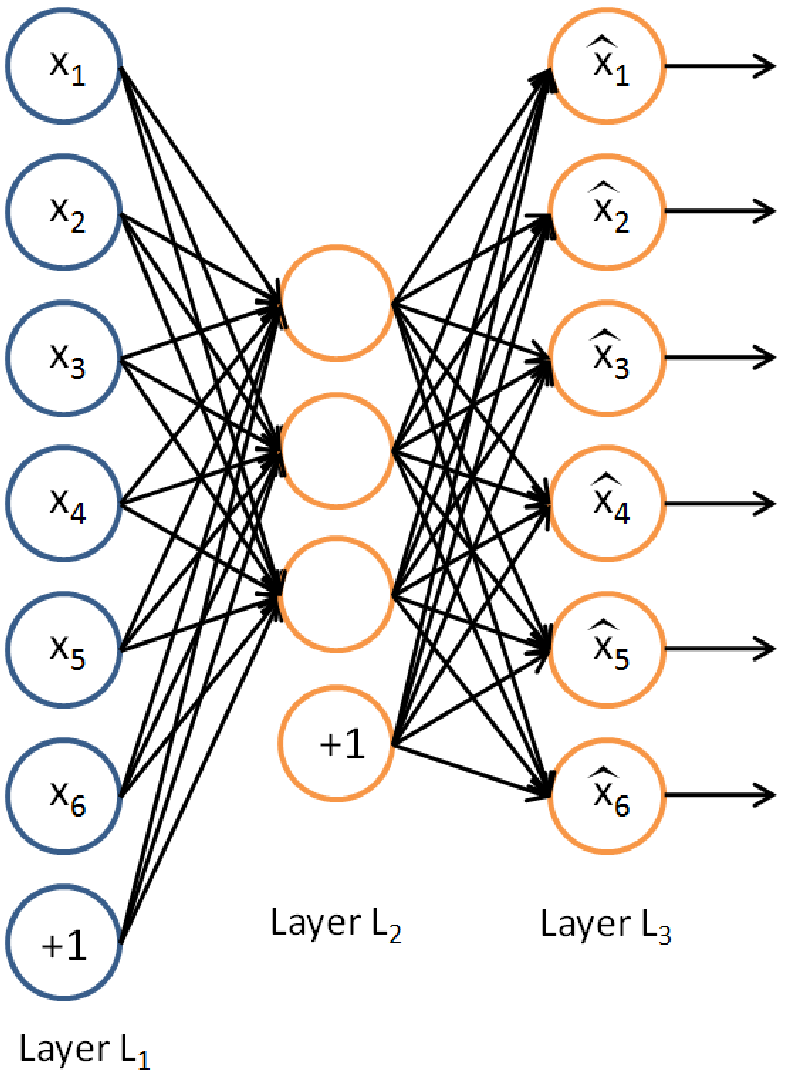
  - $Encode(X) = Z\ (\,latent\ vector);$
  - $Decode(Z) = \hat{X}$
  - $Minimize\ distance(X, \hat{X})$



Autoencoders

- The aim of an autoencoder (AE) is to learn a representation (encoding) for a dataset, typically for dimensionality reduction.
- When the NN is trained to learn data representation, it ignores noise present in the data samples.
- Along with dimensionality reduction, the NN also learns to generate the data from its encoded vector Z. This is the reason it is called an Autoencoder.

An Autoencoder Neural Network



Given data X, objective is to learn the functions f (encoder) and g (decoder) where:

$$f(X) = s\left(W^{E}X + b^{\mathrm{E}}\right) = Z$$

$$g(Z) = s\left(W^{\mathrm{D}}\mathrm{Z} + b^{\mathrm{D}}\right) = \hat{X}$$

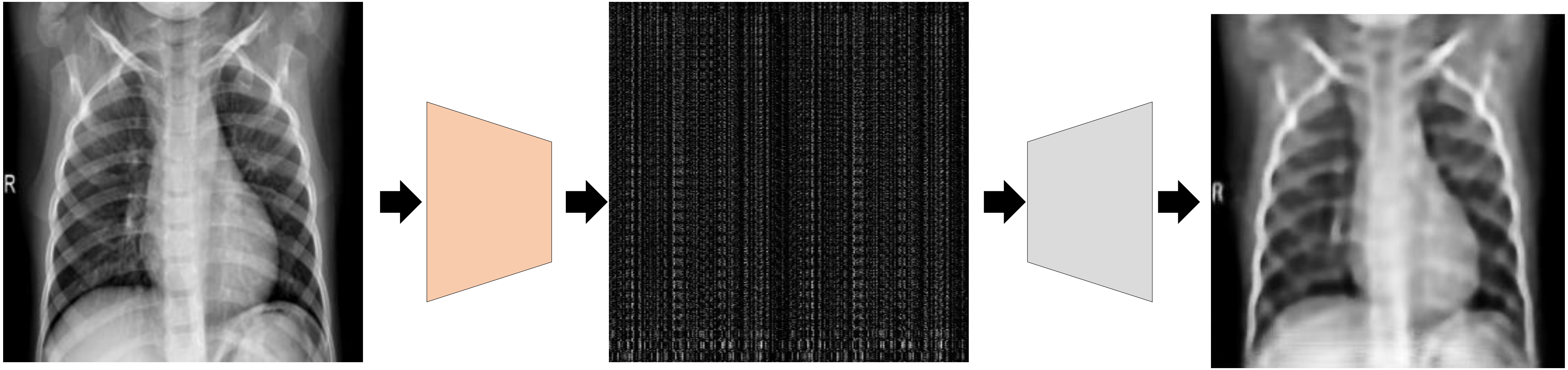
Such that $\quad h(X) = g\left(f(X)\right) = \hat{X}$

approximates X , i.e. $\|h(X) - X\|$ is minimized.

Here h is an **approximation** of the identity function. Activation function in the hidden layer helps extract non-linear data characteristics.

Example: Chest X- Ray

Chest X-ray



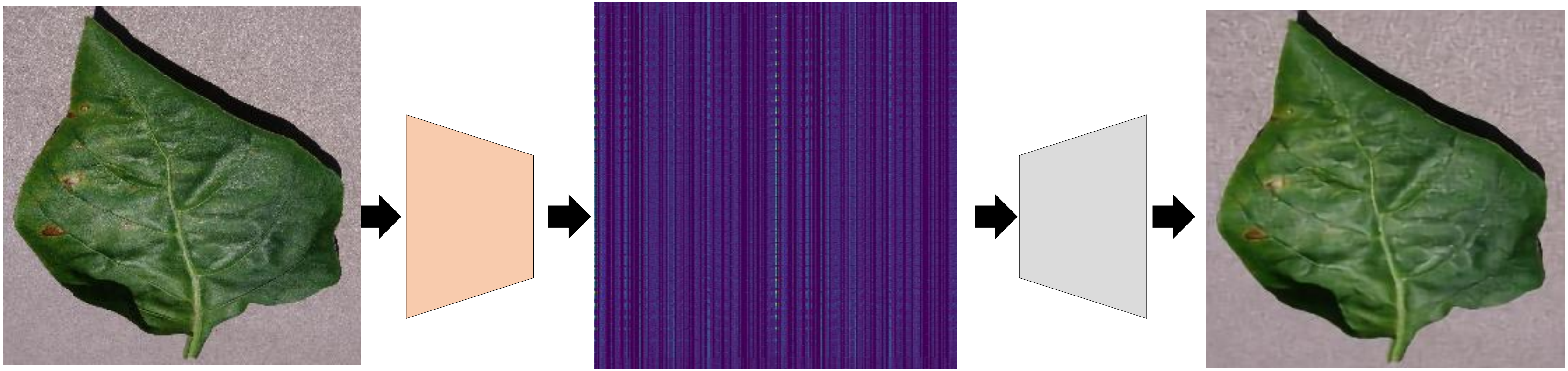
Original Image

Latent Representation

Reconstructed Image

Example: Plant Leaf

Plant leaf



Original Image

Latent Representation

Reconstructed Image

Example: Brain MRI

Brain MRI Image

Original Image

Latent Representation

Reconstructed Image

Example

- Let the input vector be $X = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}$
- Let the encoder contain only 1 layer having 2 units (neuron).
- Let the encoder weight matrix and bias be as follows.

$$W = \begin{bmatrix} 1 & 2 \\ 1 & 0 \\ -1 & 0 \end{bmatrix}, b = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Then $W^T X + b = ?$

Apply ReLU activation to get the output (latent representation)

Example (Contd.)

- Let the decoder weight matrix and bias be as follows.

$$W = \begin{bmatrix} 2 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, b = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Then $W^T X + b = ?$

Apply ReLU again. And then find out the error of reconstruction.

Origin and Applications

- “The idea of autoencoders has been part of the historical landscape of neural networks for decades (LeCun, 1987; Bourlard and Kamp, 1988; Hinton and Zemel,1994). Traditionally, autoencoders were used for dimensionality reduction or feature learning.”

  Ian Goodfellow, Yoshua Bengio and Aaron Courville, Deep Learning, MIT Press.

- More recently, autoencoders are extensively used for a variety of different applications including generative modeling.

Applications of Autoencoder

- Feature & Representation learning
- Dimensionality reduction
- Clustering
- Data compression
- Denoising, data augmentation, segmentation etc.
- Generative models

Training the AE

- The AE is trained just like a conventional NN using **Gradient Descent.**

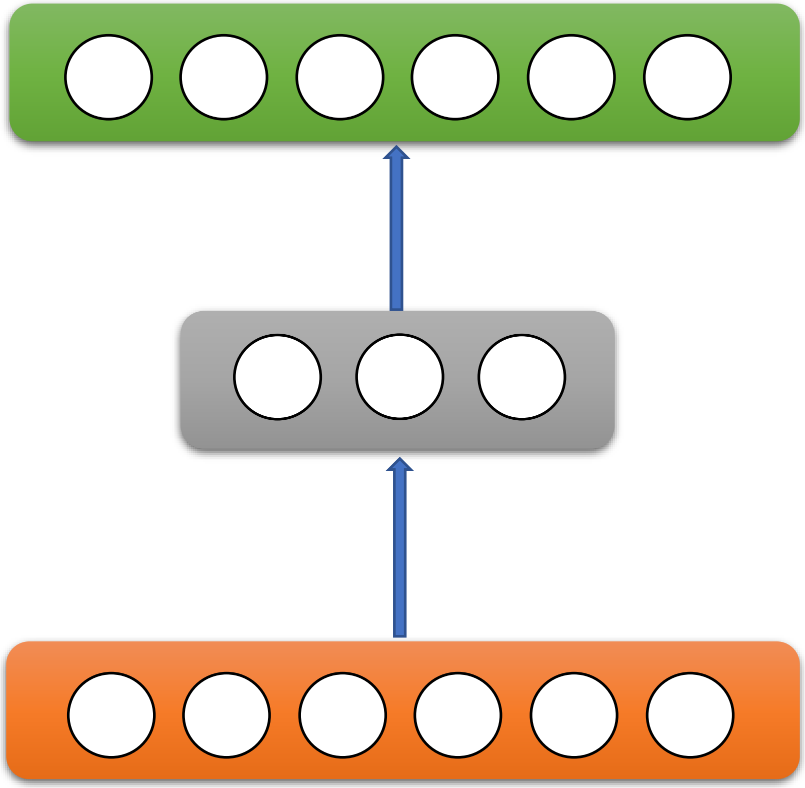
- To train an AE for reconstructing the input, the loss function is taken as

  - $L(\mathrm{X}, \hat{X}) = \|\mathrm{X} - \hat{X}\|^2$

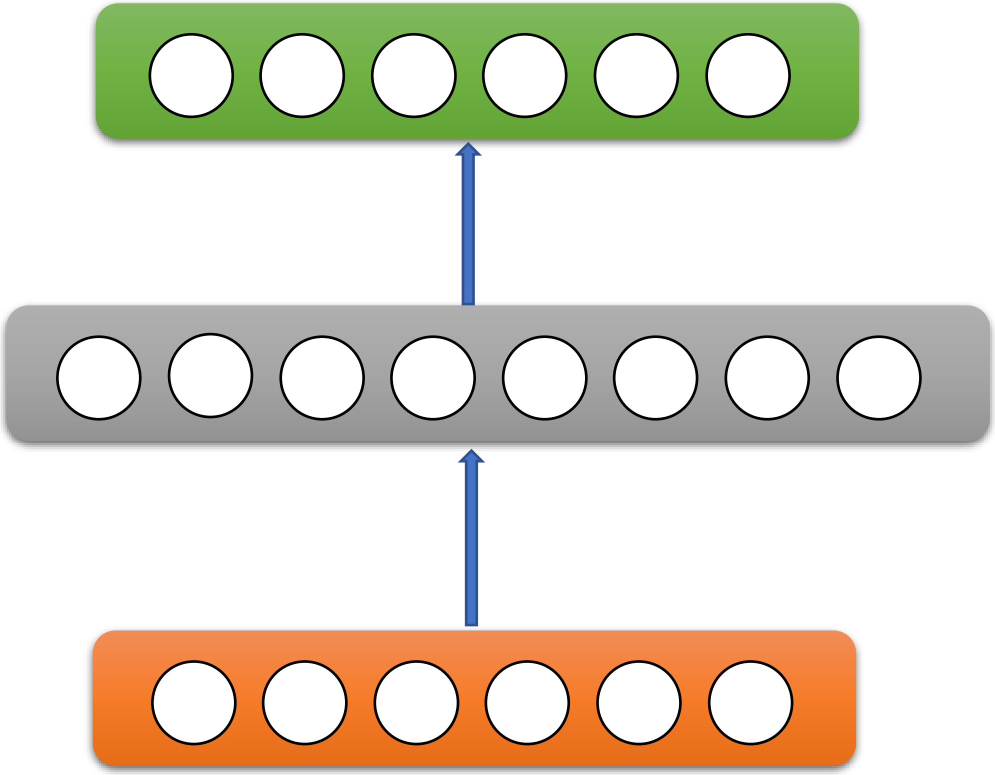
- If the input is a bit vector or a vector of bit probabilities, then cross entropy loss can be used

  - $H(p, \hat{p}) = -\sum_X p(X) \log \hat{p}(X)$

Undercomplete AE and overcomplete AE



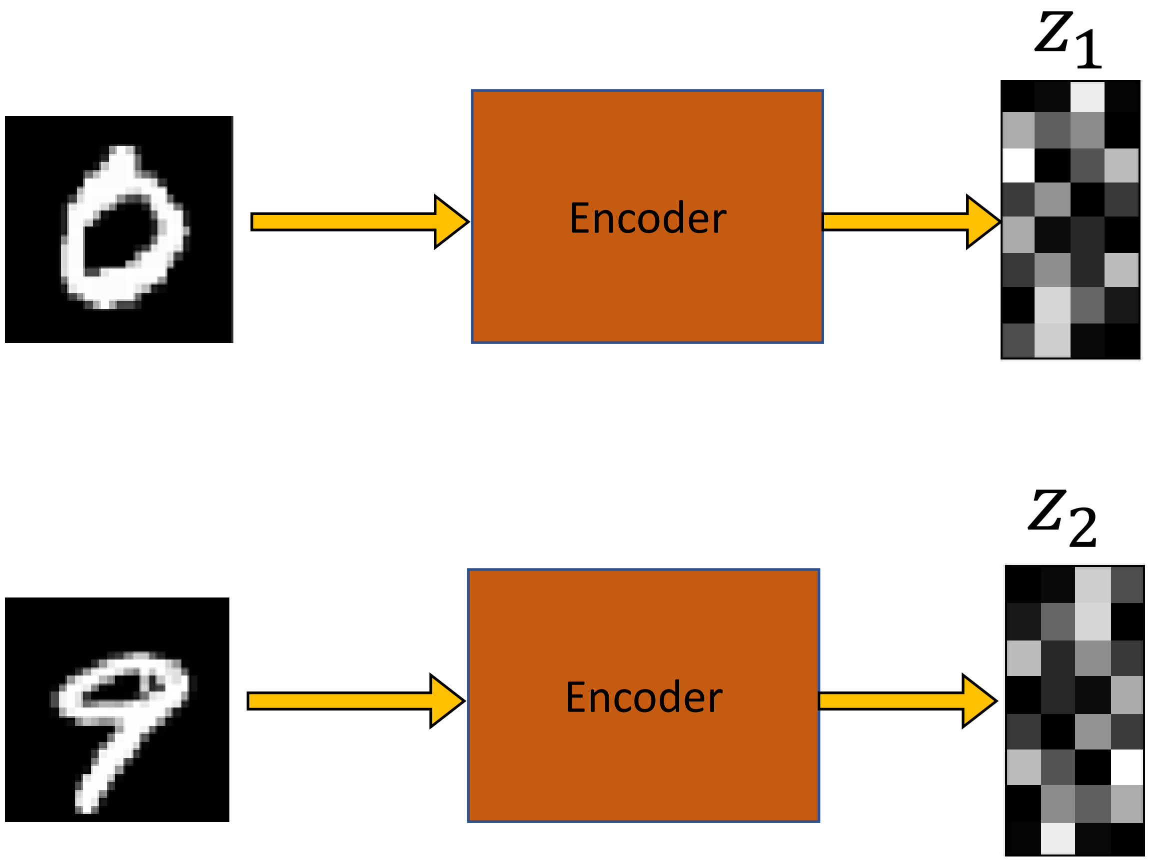
Undercomplete
AE compresses the input.



Overcomplete AE
learns sparse representation

Source: lecture slides of Dr. Guy Golan

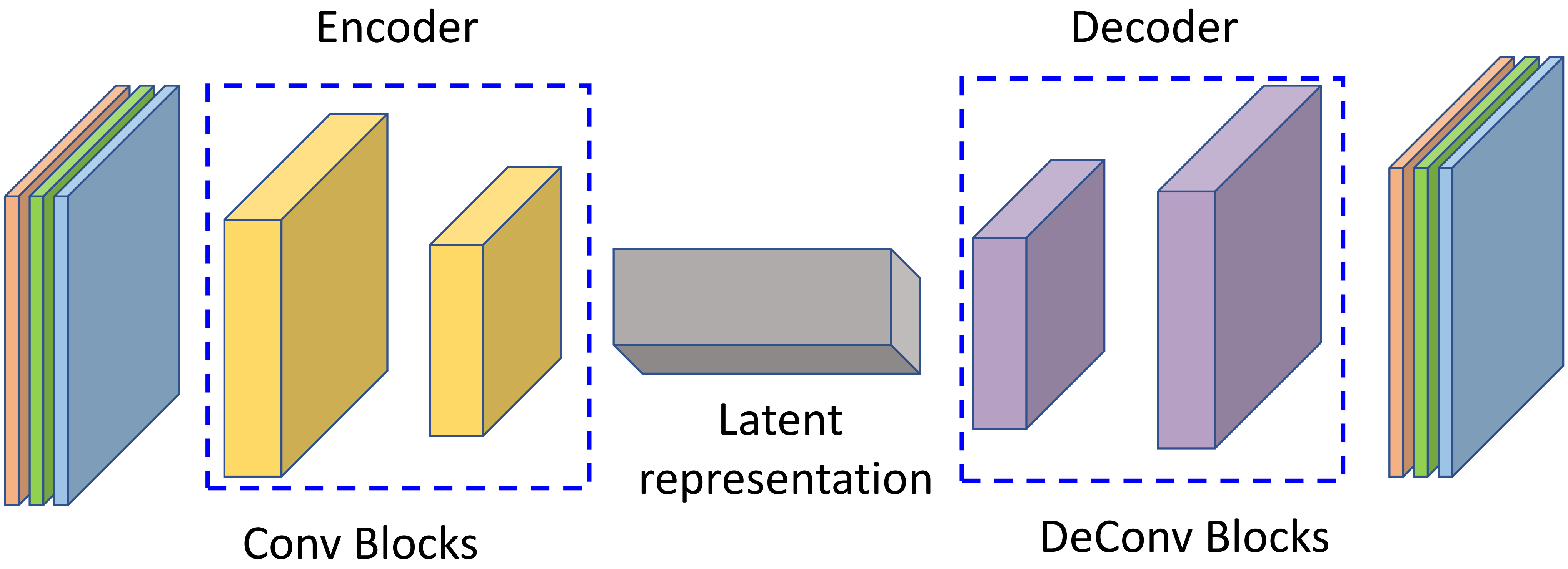
Simple latent space in Keras



Source: lecture slides of Dr. Guy Golan

Convolutional AE

- Contains convolution and deconvolution blocks.
- May contain pooling and upsampling layers.



An Example

Image (6 x 6) → Convolution 2d with ReLU, SAME Padding → Max Pooling → Convolution 2d with ReLU → UpSampling

0	1	4	2	8	2
5	3	0	1	0	5
1	5	3	2	9	3
9	8	5	7	8	7
7	2	7	9	9	4
7	1	8	7	5	3

Image (6 x 6)

*

Convolution 2d

1	0	1
0	-1	1
1	1	0

Filter (3 x 3)
(Random values)

6	11	1	7	-5	3
0	7	12	16	20	15
16	20	16	19	15	12
11	10	18	29	22	15
10	27	26	28	21	12
-4	21	10	14	11	6

Output of Convolution 2d

→

6	11	1	7	0	3
0	7	12	16	20	15
16	20	16	19	15	12
11	10	18	29	22	15
10	27	26	28	21	12
0	21	10	14	11	6

ReLU Max(0,x)

→

11	16	20
20	29	22
27	28	21

Max-pooling
(2 x 2)

An Example (Contd.)

11	16	20
20	29	22
27	28	21

Convolution 2d

*

1	1	0
0	1	1
1	0	1

Filter (3 x 3)
(Random values)

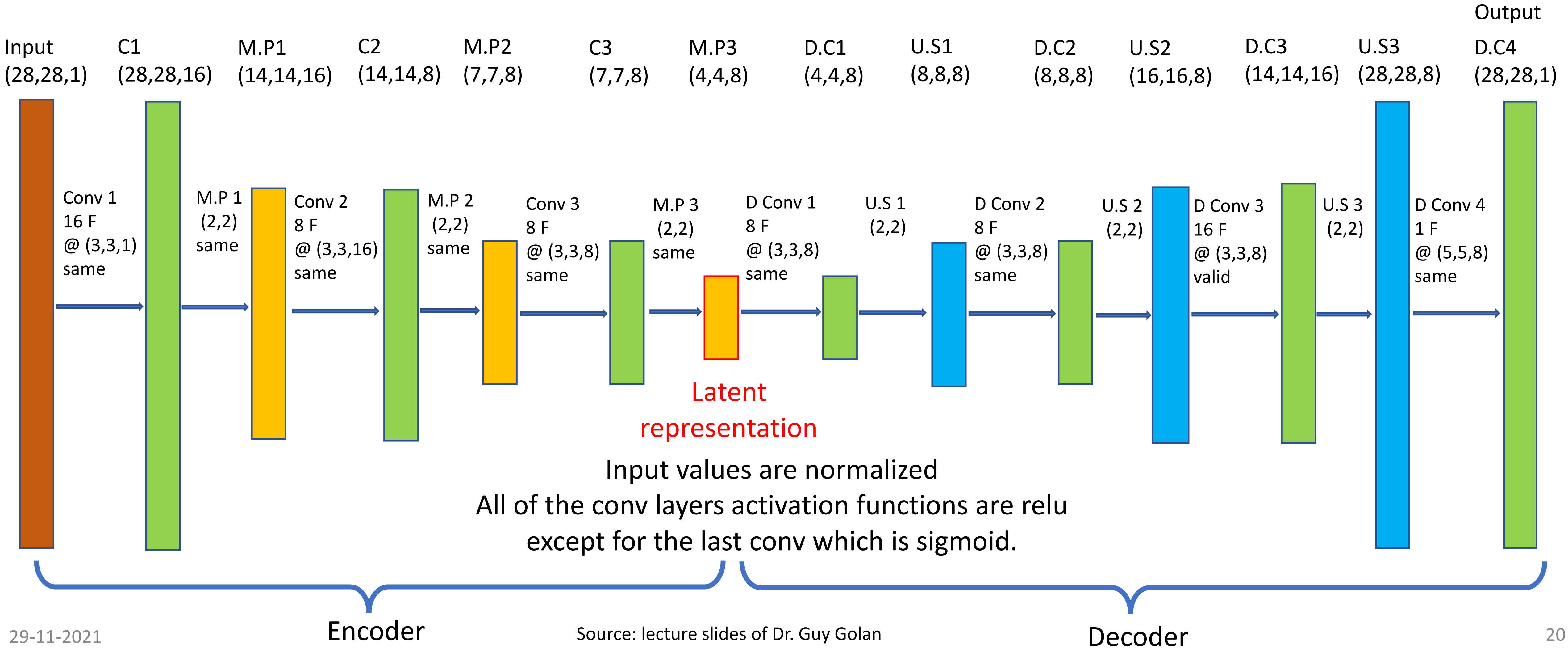
56	78	49
88	126	86
75	98	72

Upsampling
(2 x 2)

56	56	78	78	49	49
56	56	78	78	49	49
88	88	126	126	86	86
88	88	126	126	86	86
75	75	98	98	72	72
75	75	98	98	72	72

Image

Another Example: Convolutional AE

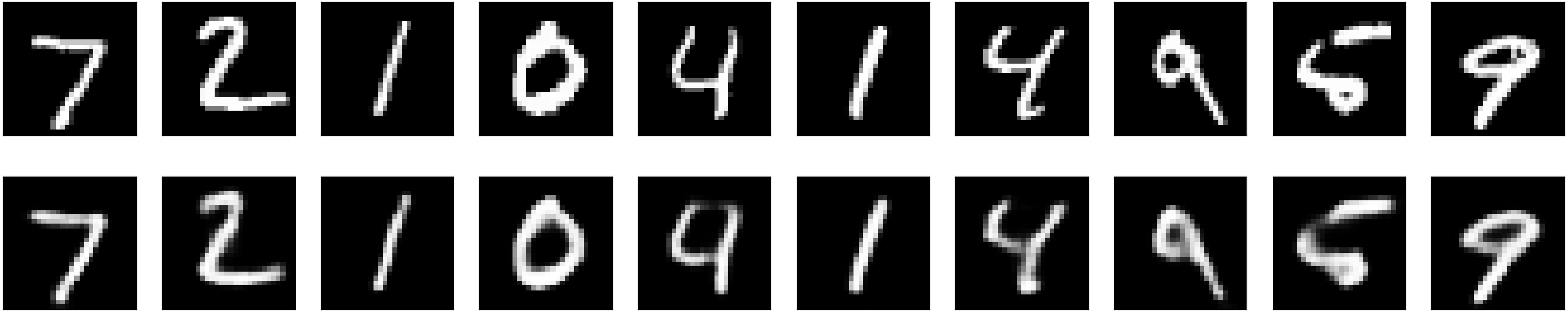


Input values are normalized
All of the conv layers activation functions are relu
except for the last conv which is sigmoid.

Source: lecture slides of Dr. Guy Golan

Convolutional AE – Keras Example Results

- 50 epochs.
- 88% accuracy on validation set.



Source: lecture slides of Dr. Guy Golan

Deconvolution Layer

- Mathematical meaning of deconvolution is inverse of convolution operation.
- If convolution on X produces Z, then the output of a deconvolution on Z should ideally produce X.
- In Deep learning however, deconvolution is not so popular.
- Transposed convolution or fractionally strided convolution are commonly used to spatially reconstruct the dimension of an input on which a convolution is applied.

Transposed Convolution

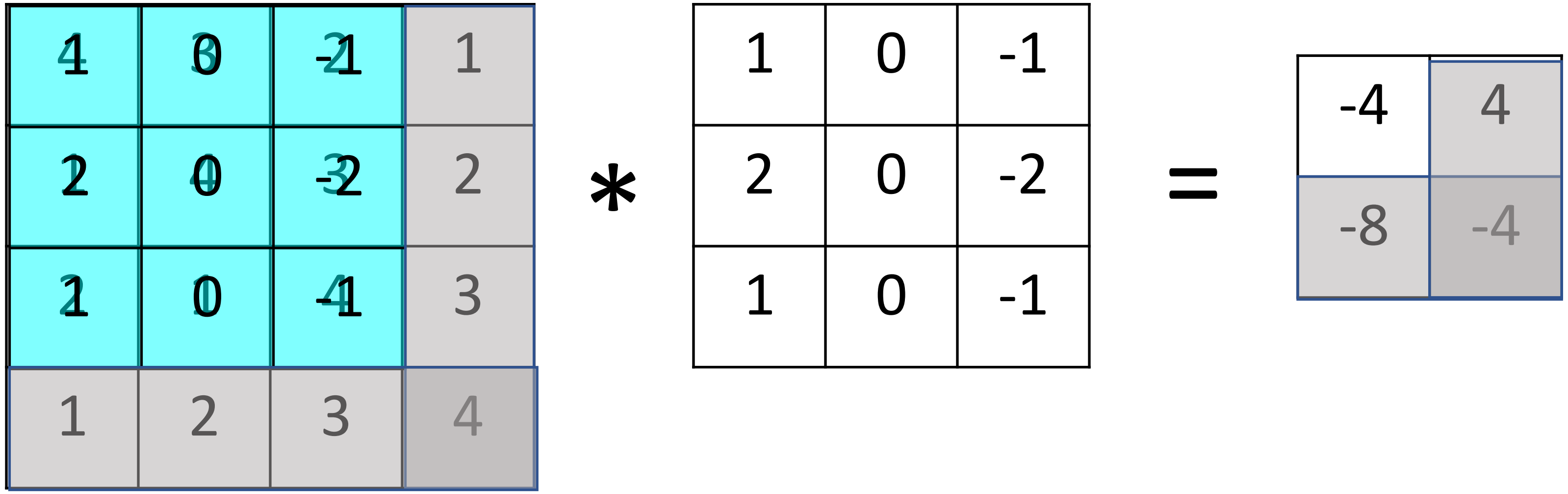
$$\begin{bmatrix} 4 & 3 & 2 & 1 \\ 1 & 4 & 3 & 2 \\ 2 & 1 & 4 & 3 \\ 1 & 2 & 3 & 4 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} -4 & 4 \\ -8 & -4 \end{bmatrix}$$

Apply Convolution

After convolution an input with 16 features is converted to an output with 4 features.

Question: How can we get a vector of dimension 16 from an input vector of dimension 4?

Transposed Convolution Revisited



When a filter is applied on an image block, contribution of other pixel values is 0 (zero).

You can think of the filter size to be the same as that of the input image. It has nonzero values only at 9 positions, and 0 elsewhere.

Transposed Convolution Revisited

$$\begin{bmatrix} 4 & 3 & 2 & 1 \\ 1 & 4 & 3 & 2 \\ 2 & 1 & 4 & 3 \\ 1 & 2 & 3 & 4 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} -4 & 4 \\ -8 & -4 \end{bmatrix}$$

$$\begin{bmatrix} 4 \\ 3 \\ 2 \\ 1 \\ 1 \\ 4 \\ 3 \\ 2 \\ 2 \\ 1 \\ 4 \\ 3 \\ 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} * \begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \\ 2 \\ 0 \\ -2 \\ 0 \\ 1 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = -4$$

Transposed Convolution…

- First column of the transposed (Trans) convolution (Conv) matrix is just constructed.
- Similarly construct the second column, third and fourth columns.
- Once the Trans-Conv matrix is created, multiply it with the output of the convolution.
- The output will be a 16-d vector.
- Rearrange it into a 4×4 array.
- That is the output of a transposed convolution

1	0	0	0
0	1	0	0
-1	0	0	0
0	-1	0	0
2	0	1	0
0	2	0	1
-2	0	-1	0
0	-2	0	-1
1	0	2	0
0	1	0	2
-1	0	-2	0
0	-1	0	-2
0	0	1	0
0	0	0	1
0	0	-1	0
0	0	0	-1

-4	4
-4	-4
-8	
-4	

Other Upsampling Operations

Nearest Neighbour

4	3
2	1

4	4	3	3
4	4	3	3
2	2	1	1
2	2	1	1

Other Upsampling Operations

Bilinear Interpolation: Linear interpolation in both the directions

40	30
20	10

→

40	37	32	30
34	31	27	25
25	22	17	15
20	18	12	10

Regularization

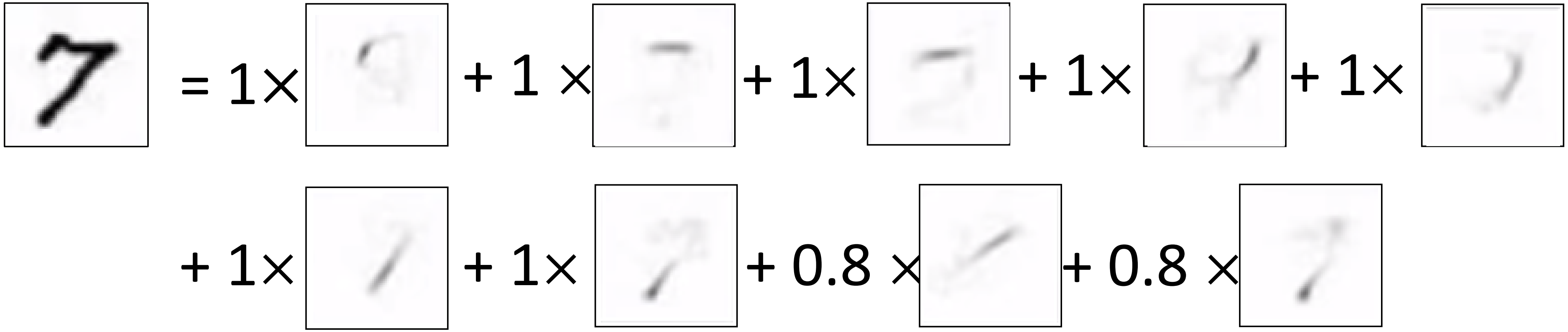
- Motivation: We would like to learn meaningful features **without** altering the code's dimensions (Overcomplete or Undercomplete).

- The solution: imposing other constraints on the network.

Source: lecture slides of Dr. Guy Golan

Sparsely Regulated Autoencoders

- We want our learned features to be as **sparse** as possible.
- With sparse features we can generalize better.



Source: lecture slides of Dr. Guy Golan

Sparsely Regulated Autoencoders

- Suppose $a_j^{(\mathrm{Bn})}$ is defined to be the activation of the jth hidden unit (bottleneck) of the autoencoder.

- Let $a_j^{(\mathrm{Bn})}(x)$ be the activation of this specific node on a given input x.

Source: lecture slides of Dr. Guy Golan

Sparsely Regulated Autoencoders

- Further let,

$$\hat{\rho}_j = \frac{1}{m}\sum_{i=1}^{m}\left[a_j^{(\mathrm{Bn})}\left(x^{(i)}\right)\right]$$

- be the average activation of hidden unit j (over the training set).
- Apply the constraint: $\hat{\rho}_j = \rho$ where ρ is a "sparsity parameter", that is typically small.
- This constraint forces the average activation of each neuron j to be close to ρ.

Source: lecture slides of Dr. Guy Golan

Sparsely Regulated Autoencoders

To penalize $\hat{\rho}_j$ for deviating from ρ, a penalty term needs to be added in the objective function.

An example of a penalty term :

$$\sum_{j=1}^{Bn} KL(\rho|\hat{\rho}_j)$$

where $KL(\rho|\hat{\rho}_j)$ is a Kullback-Leibler divergence function.

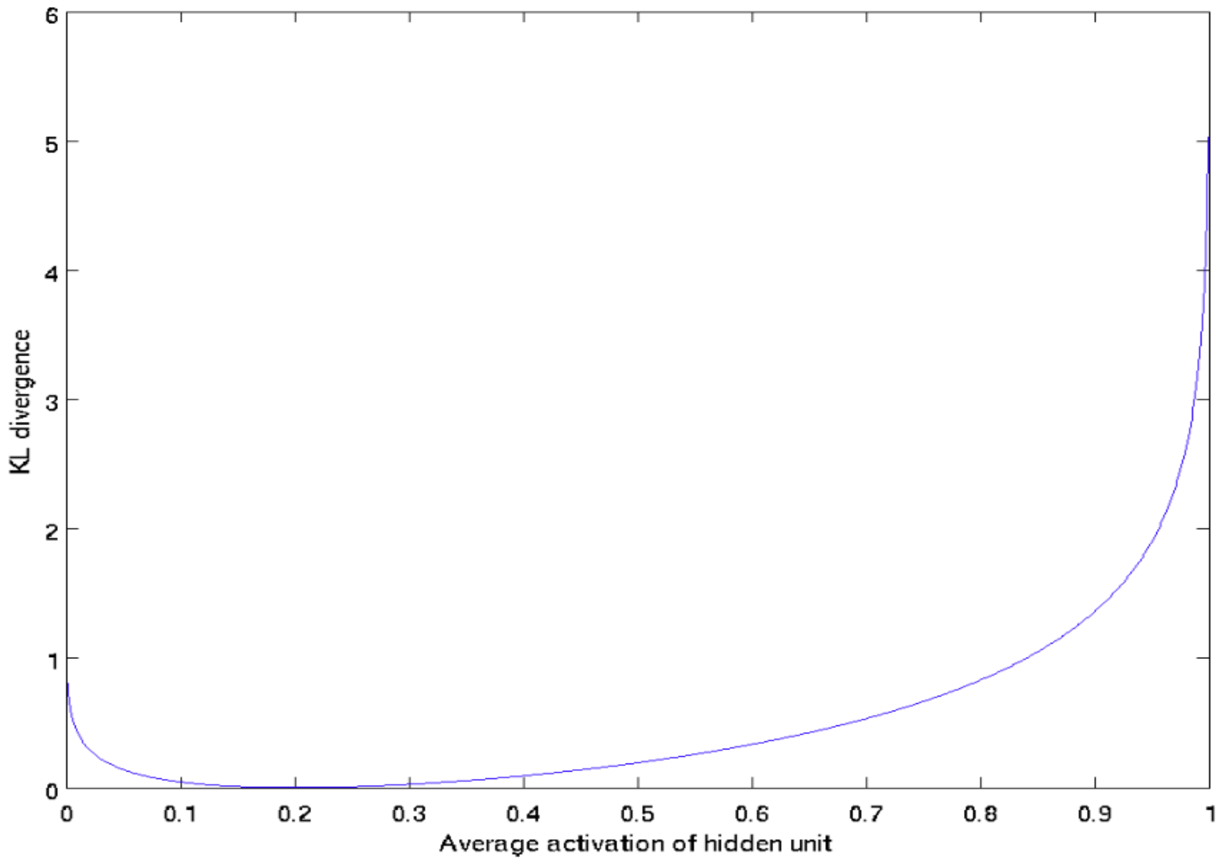
Source: lecture slides of Dr. Guy Golan

Sparsely Regulated Autoencoders

- KL divergence measures how different two data distributions are.
- It has the property:

$KL(\rho|\hat{\rho}_j)$ = 0 if $\hat{\rho}_j = \rho$

If the difference between its two arguments is large, the KL-divergence is also large and increases monotonically.



$\rho = 0.2$

Source: lecture slides of Dr. Guy Golan

Sparsely Regulated Autoencoders

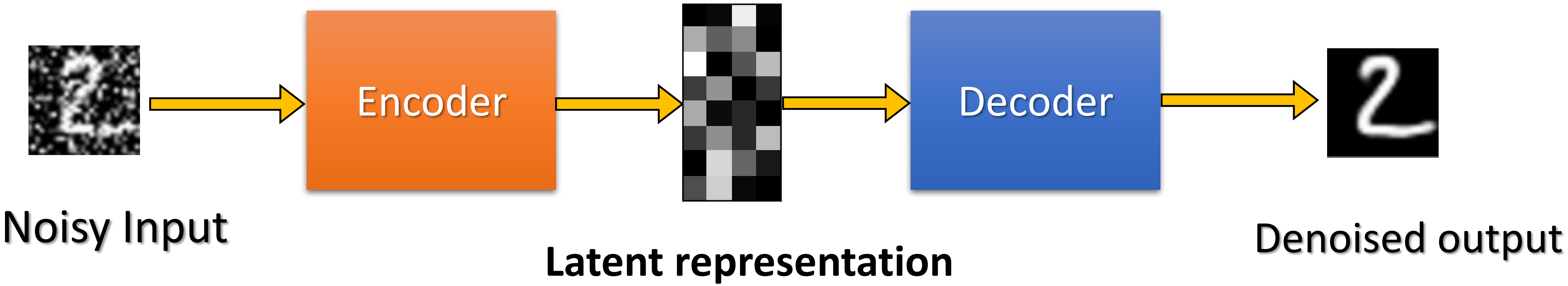
- The modified cost functions for sparsely regulated autoencoder is given by
- $J_S(W,b) = J(W,b) + \beta \sum_{j=1}^{Bn} KL\left(p|\hat{\rho}_j\right)$

- Note: We need to know $\hat{\rho}_j$ before hand, so we have to compute a forward pass on all the training set.

Source: lecture slides of Dr. Guy Golan

Denoising Autoencoders

- The idea is to reduce the effect of *corruption* process stochastically applied to the input.
- To encode the input but to NOT mimic the identity function.
- To create a robust model that eliminates noise



Source: lecture slides of Dr. Guy Golan

Denoising Autoencoder Training

- Suppose X is an original input without noise and $\tilde{X}$ is its noisy version.
- Suppose the output of the AE is $\hat{X}$. i.e.,

$$\hat{X} = Encoder(Deconder(\tilde{X}\ )$$

- Instead of trying to mimic the identity function by minimizing the reconstruction loss $L(\tilde{X}, \hat{X})$, the denosing autoencoder minimizes the loss function $L(\ X, \hat{X})$,

Denoising Autoencoder Training

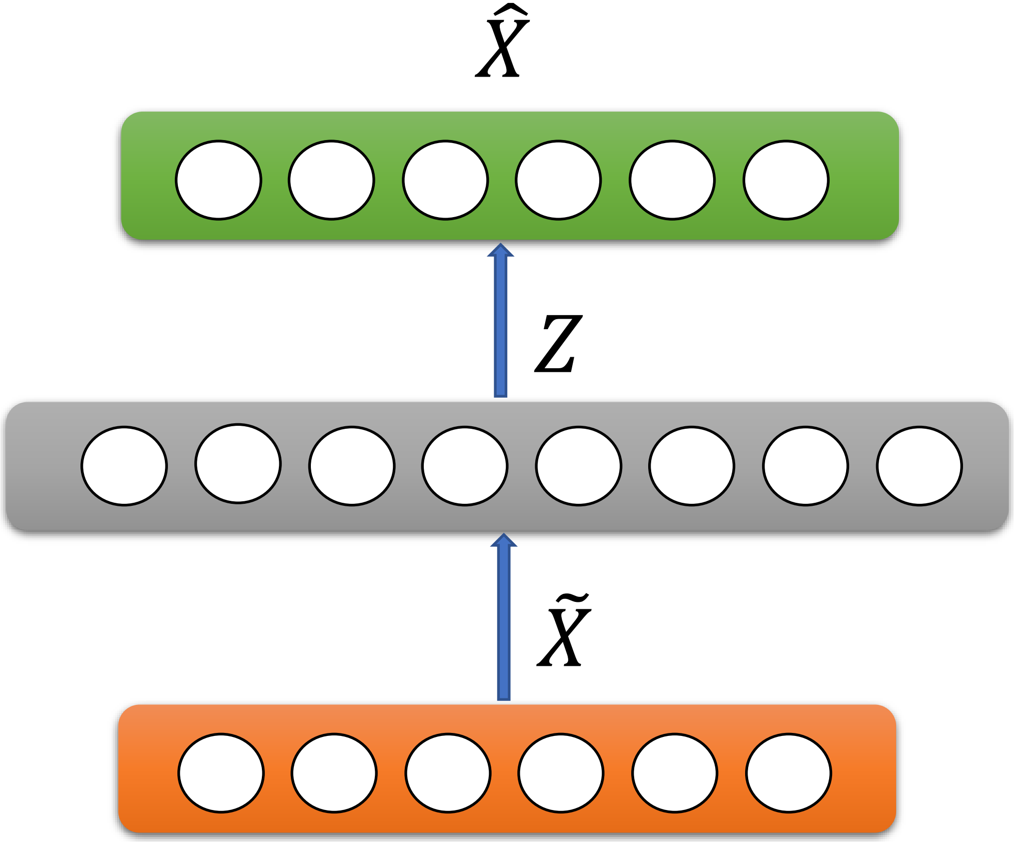
Add randomly some noise in input images say, Gaussian additive noise.

(a)

X

(b)

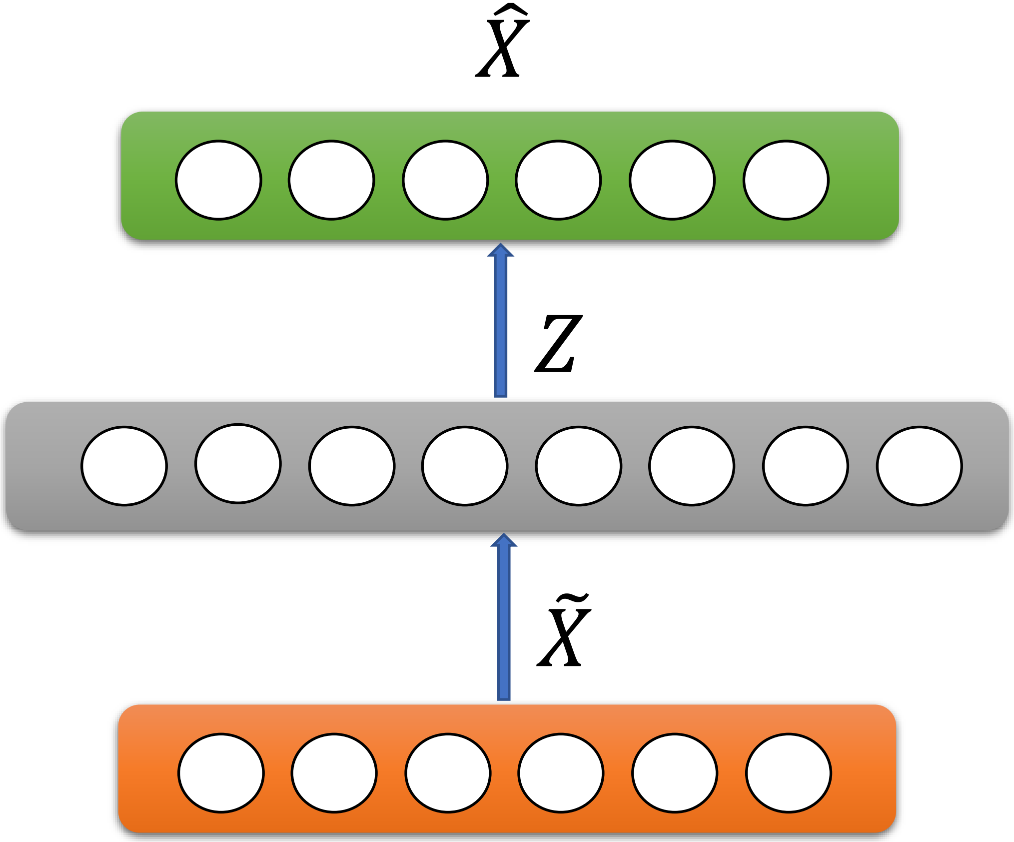
$\tilde{X}$



Denoising Autoencoder Training

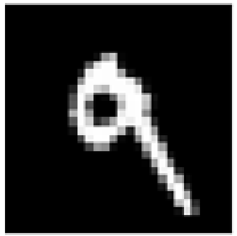
Find out the loss $L(X, \hat{X})$,
Apply gradient descent to
minimize the loss function.

(a) X (b) $\tilde{X}$



Denoising Autoencoder Training: An Example

Take some input x



Apply Noise

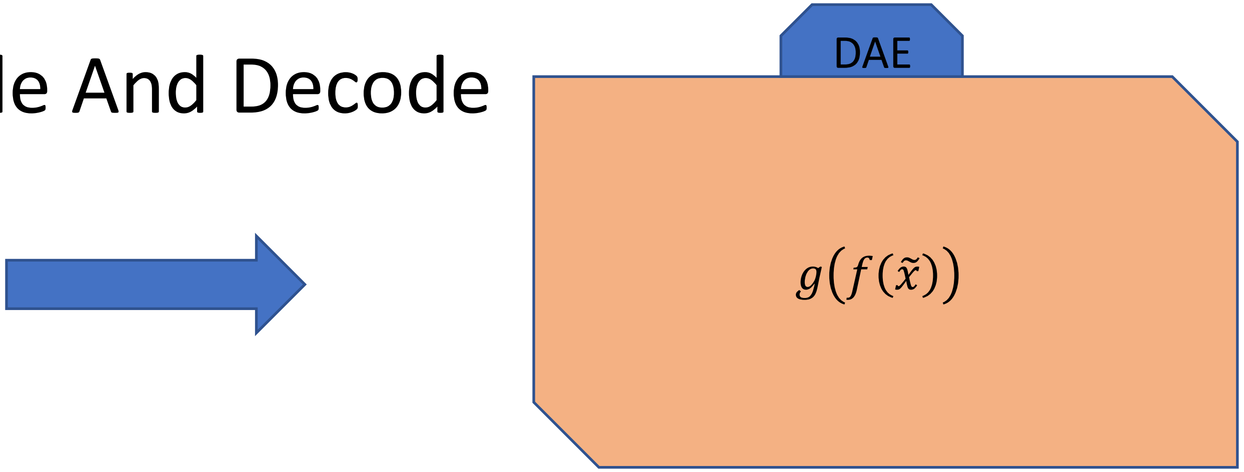
$\tilde{x}$

Source: lecture slides of Dr. Guy Golan

Denoising Autoencoder Training: An Example

$\tilde{x}$

Encode And Decode

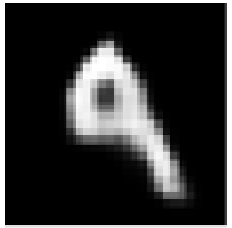


Denoising Autoencoder Training: An Example

DAE

$g\left(f(\tilde{x})\right)$

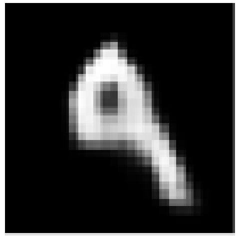
$\hat{x}$



Source: lecture slides of Dr. Guy Golan

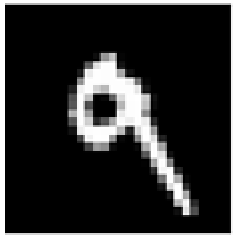
Denoising Autoencoder Training: An Example

$\hat{x}$



Compare

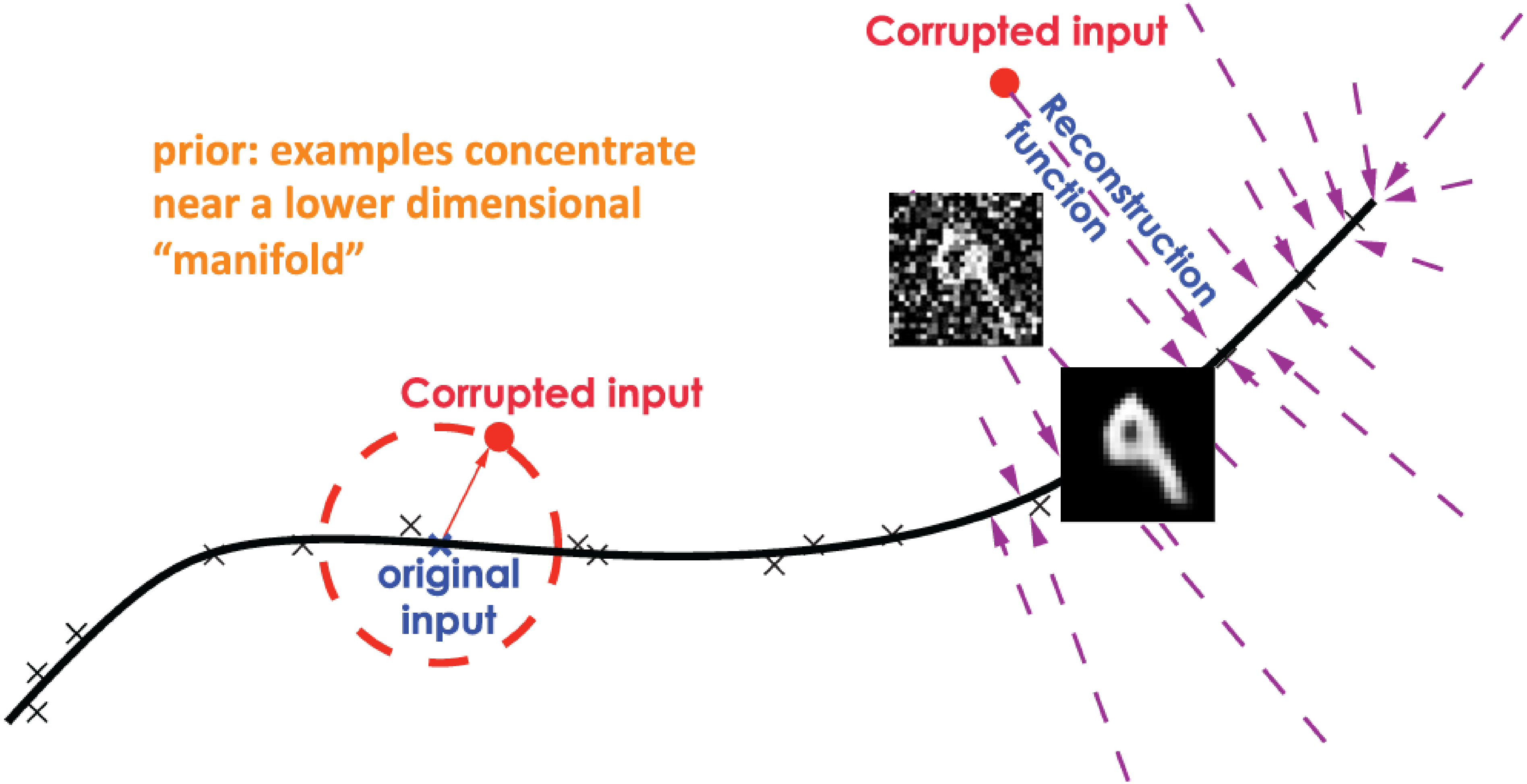
x



Source: lecture slides of Dr. Guy Golan

Denoising autoencoders

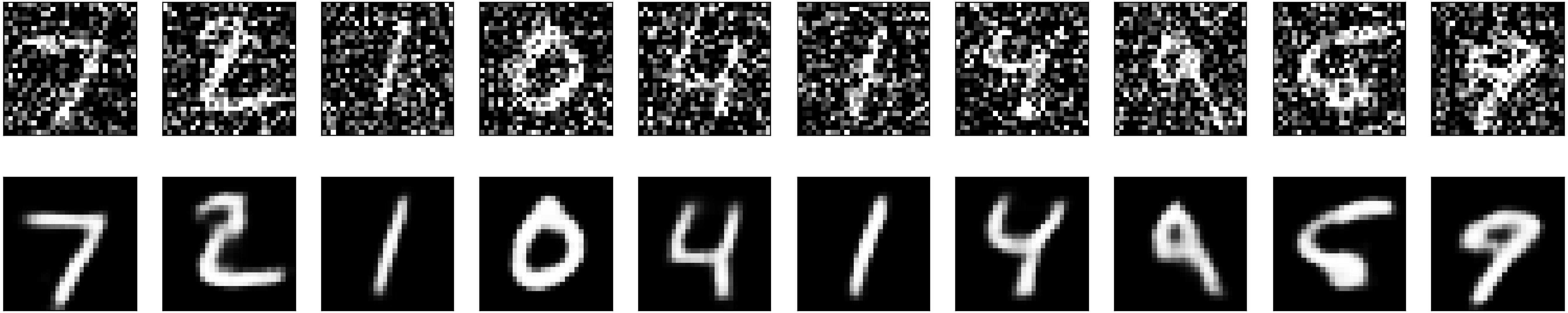
prior: examples concentrate near a lower dimensional "manifold"



Source: lecture slides of Dr. Guy Golan

Denoising convolutional AE – Keras

50 epochs.
Noise factor 0.5
92% accuracy on validation set.



Source: lecture slides of Dr. Guy Golan

Stacked Autoencoder

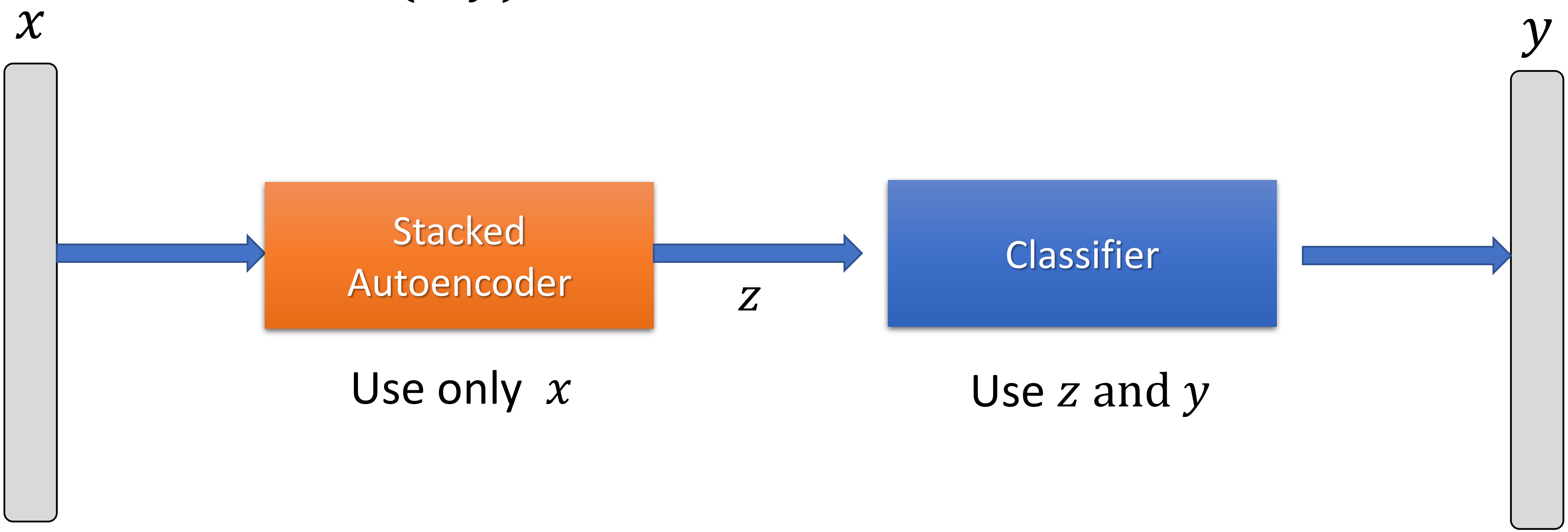
- To learn feature extraction for some application.
- Example: we can build a deep supervised classifier where the feature extraction is done by the stacked autoencoder.
- Building a SAE is done in two steps.

  Step 1. Train each AE layer one by one. Each layer learns to extract best features of its input, so that it could reconstruct the input.

  Step 2. Once all the layers of AE are trained, use the latent representation of the encoder as an input to a classifier. Any classifier like SVM, Fully connected NN can be used.

Stacked AE

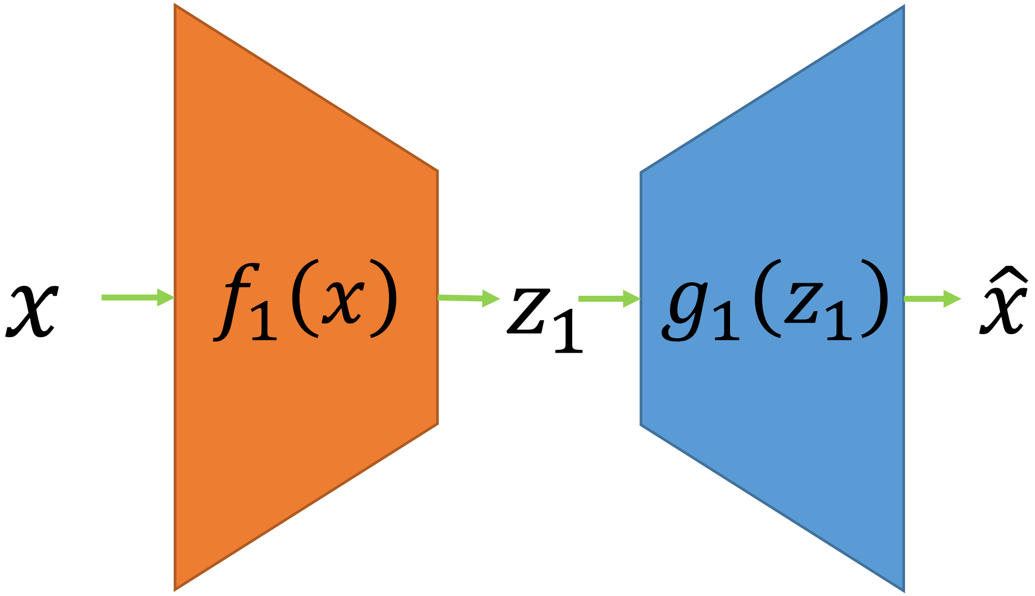
Labelled data (x, y)



Source: lecture slides of Dr. Guy Golan

Stacked AE Training

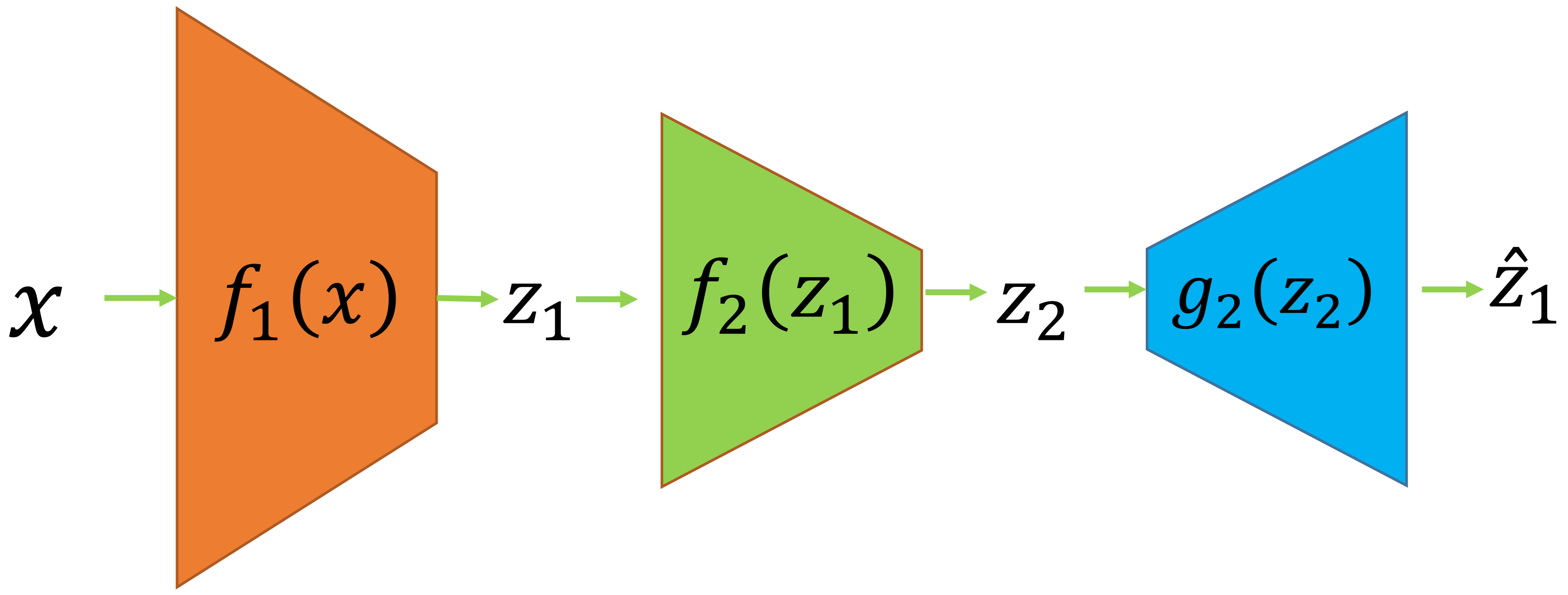
First Layer Training (AE 1)



Source: lecture slides of Dr. Guy Golan

Stacked AE Training

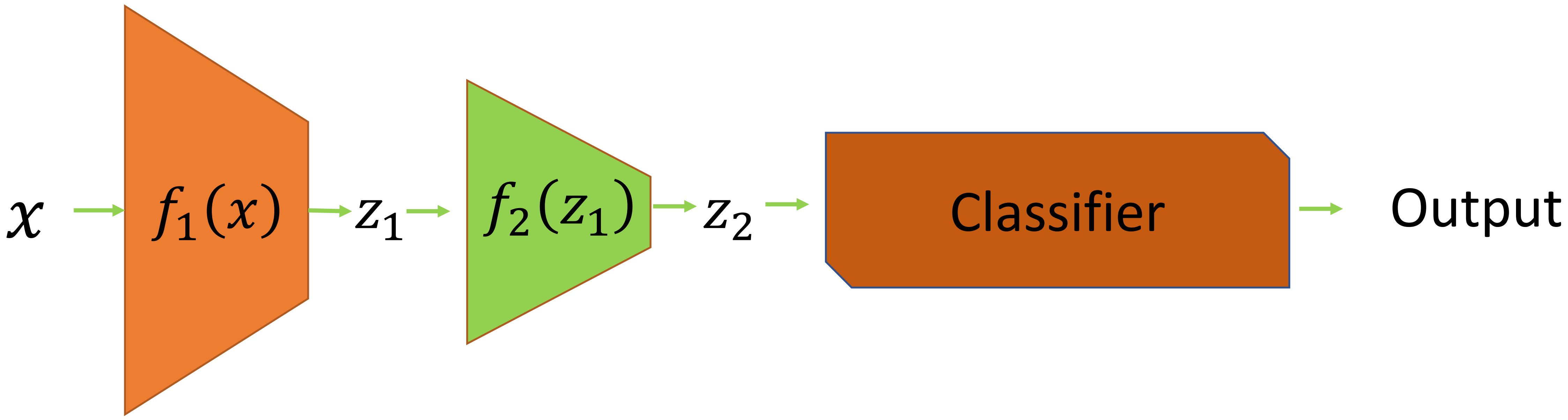
Second Layer Training (AE 2)



Source: lecture slides of Dr. Guy Golan

Stacked AE Training

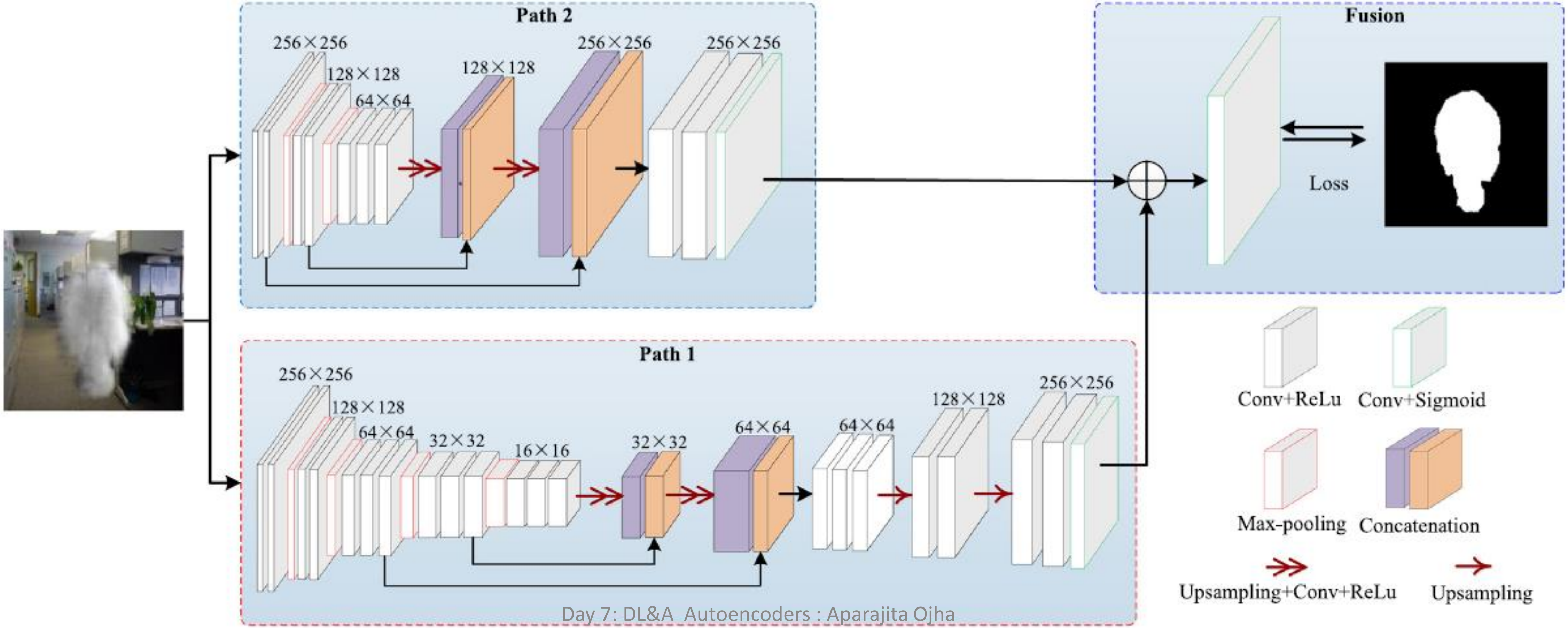
Add any classifier



Source: lecture slides of Dr. Guy Golan

Application in Image Segmentation: Encoder-Decoder Architecture

Feiniu Yuan et al., Deep smoke segmentation, Neurocomputing 357 (2019) 248–260



Deep Smoke Segmentation: Encoder-Decoder Architecture

- The first path aims at gaining global context information for generation of a coarse smoke segmentation map.
- The network takes a single RGB image as input and produces a prediction map with the same size of the input.
- First five VGG16 blocks chosen for speeding up the training process.
- Total 13 Conv layers and 4 Max Pooling layers used in the encoder of the first path.
- To learn multi-scale features and retain detailed spatial information, the network depth is increased and skip connections between the encoding and decoding phases of the network are also added.
- asymmetric structure in the decoding phase. 9 Conv layers and 4 upsampling layers introduced in the decoder.

Deep Smoke Segmentation: Encoder-Decoder Architecture

- The second path is a shallow network that aims at capturing rich local information and smoke details.
- First three blocks of VGG16 with two max-pooling layers are used in the second path.

Methods	mIoU(%)	mMse
FCN-8 s [16]	64.03	0.3221
SegNet [38]	56.94	0.3976
Static Map Detection [19]	62.88	0.3209
Text-Block FCN [40]	66.67	0.3021
Deeplab v1 [49]	68.41	0.2981
LRN [64]	66.43	0.3069
DeepSmoke	**71.04**	**0.2745**

Feiniu Yuan et al., Deep smoke segmentation, Neurocomputing 357 (2019) 248–260

Deep Smoke Segmentation Results

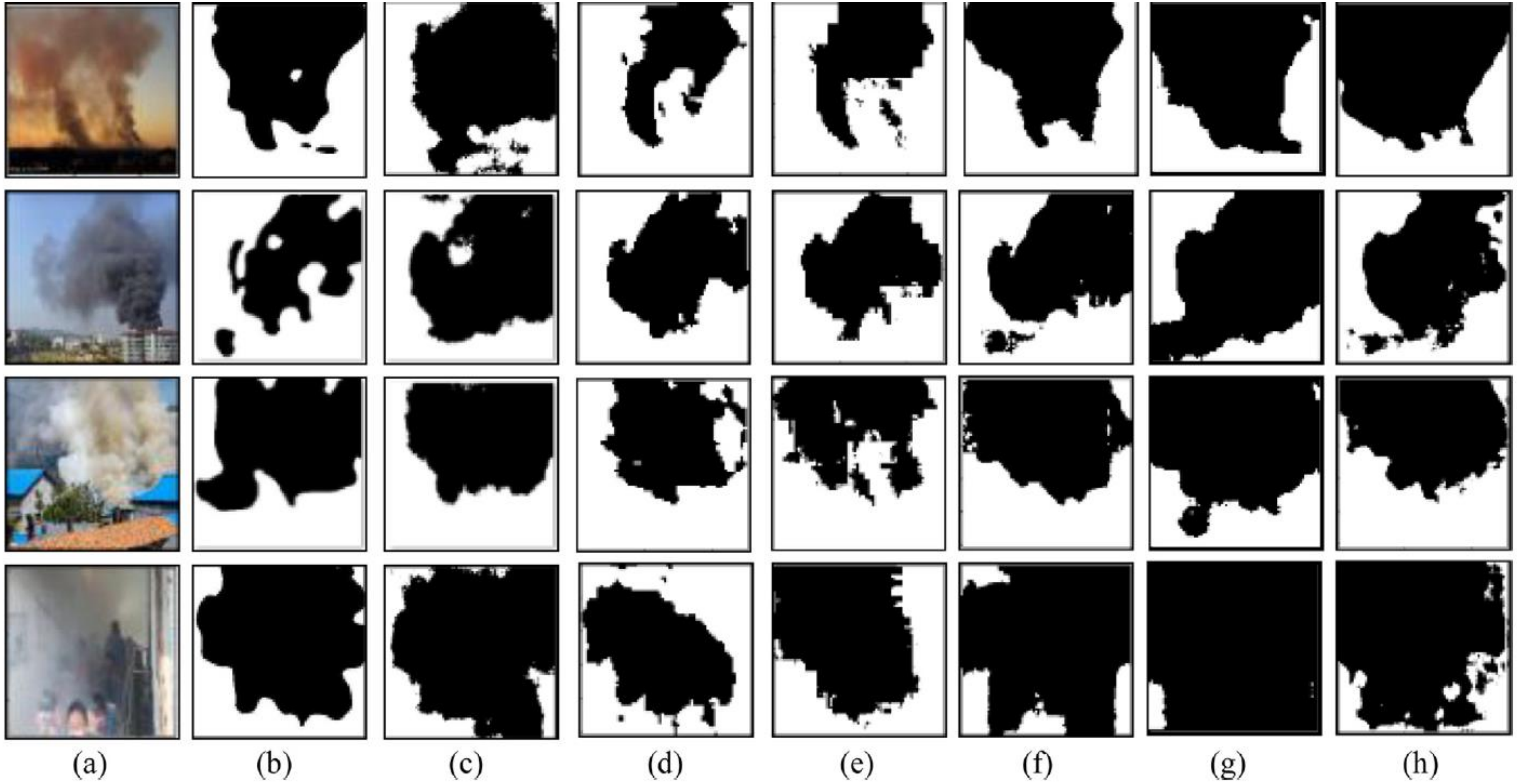


Fig. 6. Segmentation results of real smoke images. (a) Real images. (b) FCN. (c) SegNet. (d) SMD. (e) TBFCN. (f) Deeplab v1. (g)LRN. (h) DeepSmoke

Feiniu Yuan et al., Deep smoke segmentation, Neurocomputing 357 (2019) 248–260

Research Topics on Autoencoders

- Application of Autoencoders in clustering large image datasets
- Semantic segmentation
- Image Denoising
- Image dehazing
- Generating synthetic samples (new samples) : Generative Modeling

References

1. https://arxiv.org/pdf/1206.5538.pdf
2. http://www.deeplearningbook.org/contents/autoencoders.html
3. http://deeplearning.net/tutorial/dA.html
4. http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/
5. http://ufldl.stanford.edu/wiki/index.php/Stacked_Autoencoders
6. http://www.jmlr.org/papers/volume11/vincent10a/vincent10a.pdf
7. https://codeburst.io/deep-learning-types-and-autoencoders-a40ee6754663
8. Feiniu Yuan et al., Deep smoke segmentation, Neurocomputing 357 (2019) 248–260
9. Many slides are adapted from a presentation / lecture slides of Dr. Guy Golan

Thanks!

- Q/A time

Aparajita Ojha
IIITDM Jabalpur
Phone: +91-761-2794221(O)
aojha@iiitdmj.ac.in