Autoencdoer & matrix completion

Lecture 21

Changho Suh

October 6, 2021

Outline

1. Figure out what matrix completion (MC) is.

2. Explore a connection to fusion learning.

3. Investigate a variant of autoencoder that plays a significant role for matrix completion:

   Denoising autoencoder

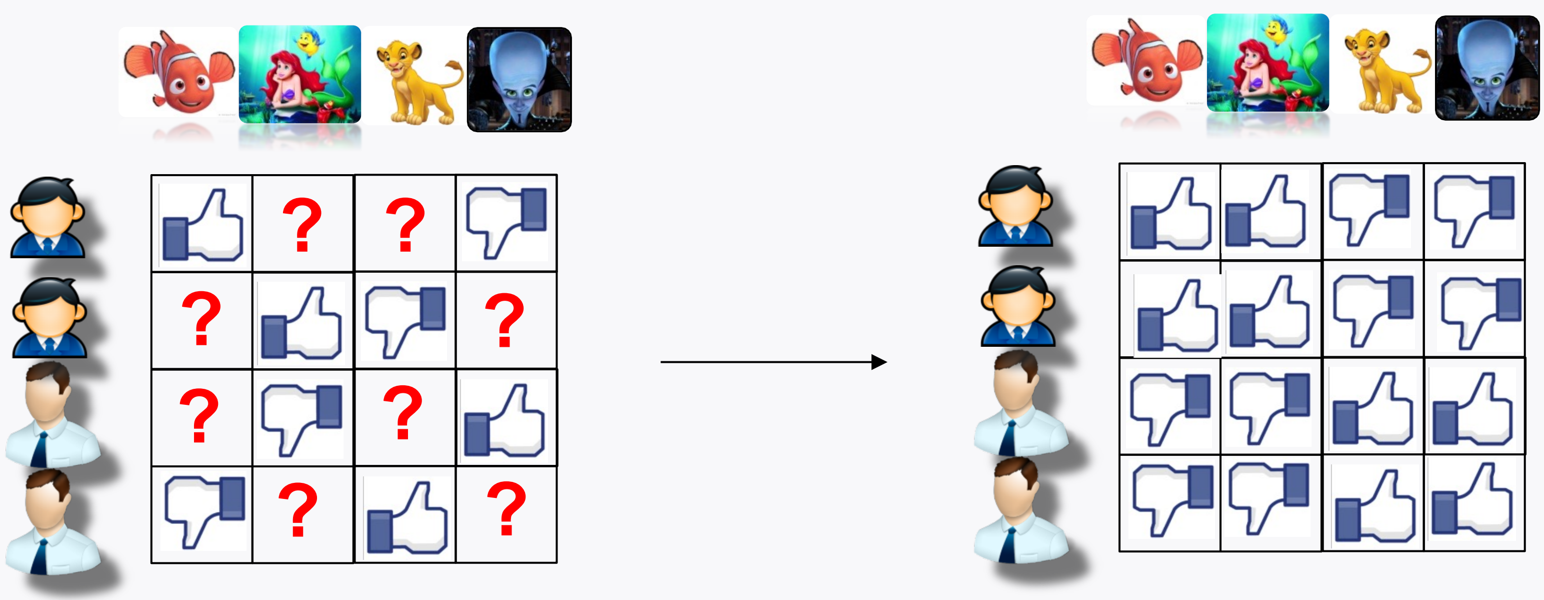
4. Study one recent MC techinique which leverages the denoising autoencoder.

Matrix completion

1.0	2.5	*
1.5	*	0.1
*	2.5	0.5

1.0	2.5	0.5
1.5	-2.1	0.1
2.0	2.5	0.5

One killer application: Recommender system



The key property that MC exploits

Low rank structure of many interested matrices

Example:

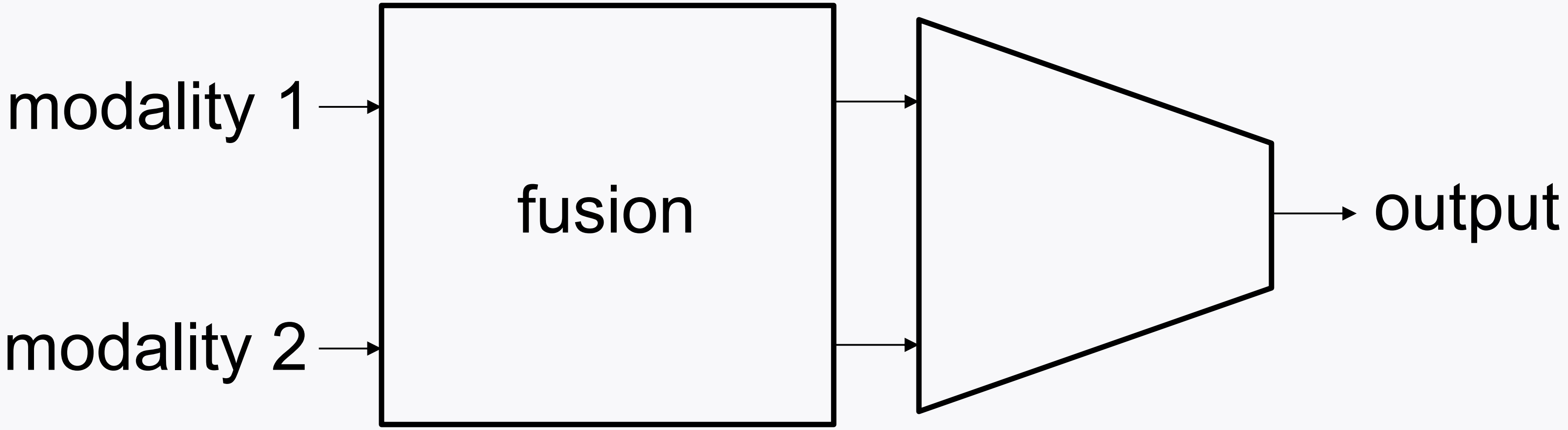
1	2	*
*	*	3
*	2	3

$$\xrightarrow{\textbf{rank = 1}}$$

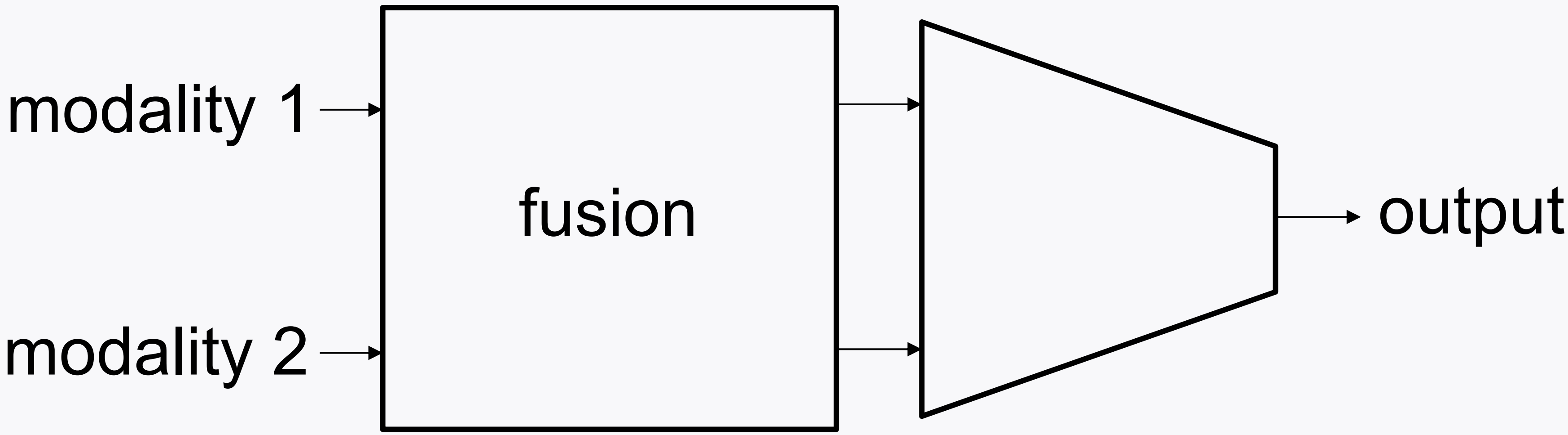
1	2	3
1	2	3
1	2	3

Connection to fusion learning?

Fusion learning: A learning methodology that exploits different types of input simultaneously



A challenge in fusion learning



Often: We have missing data!

→ Small # of examples available for all types of data

One natural way to address the challenge

Estimate missing entries!

One promeninent way to do this:

Matrix completion

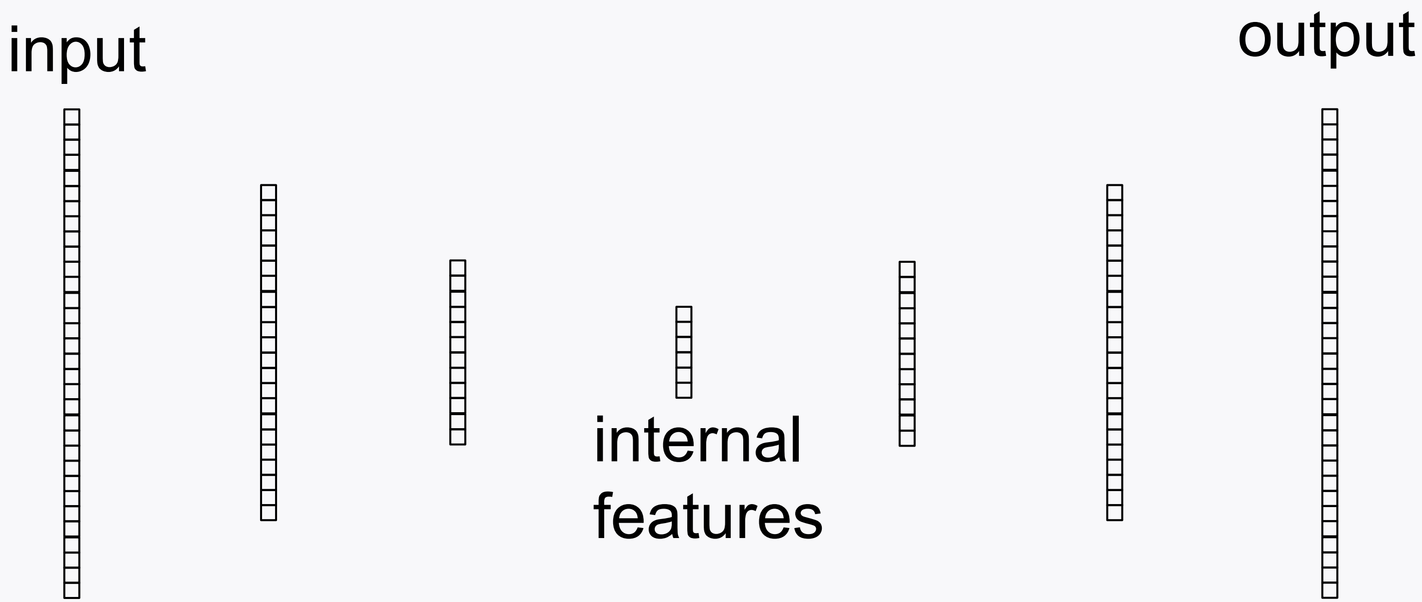
Methods for matrix completion

One recent method is via **autoencoder**.

More specifically, it is based on a variant of autoencoder:

Denoising autoencoder (DAE)

Denoising autoencoder (DAE)

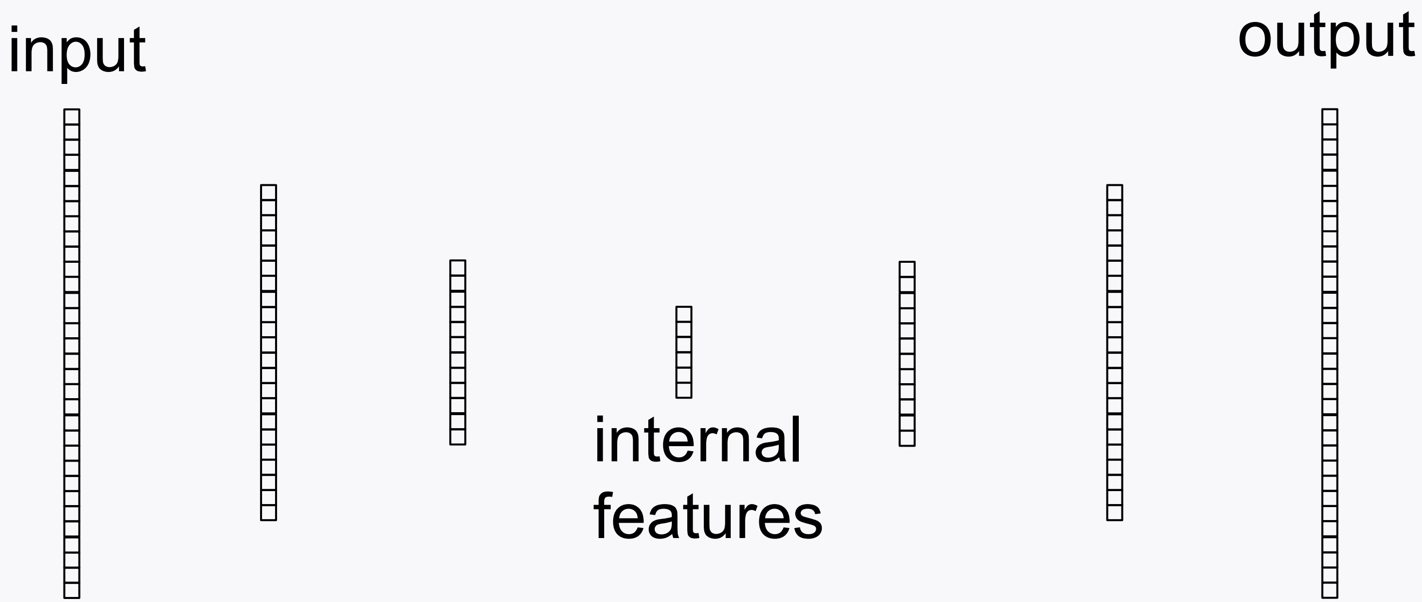


The motivation behind DAE has nothing to do with matrix completion.

Instead it is inspired by the key role of autoencoder:

Dimensionality reduction!

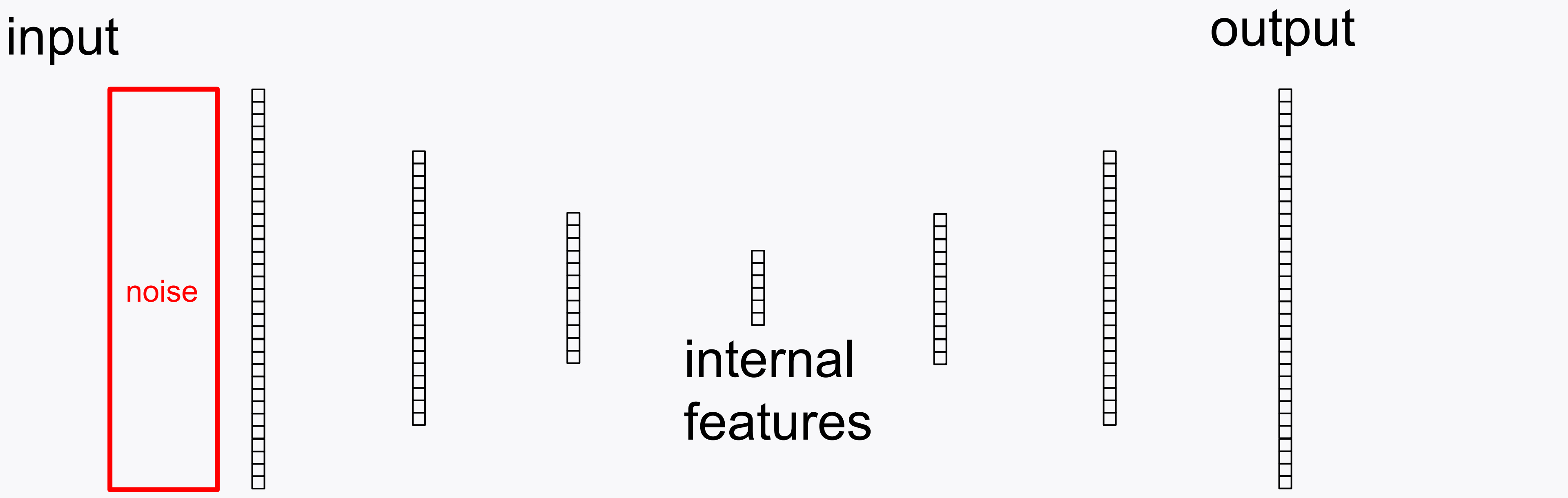
Motivation: Denoising autoencoder



What we want: Internal features well capture *key patterns* of the input.

One way to encourage this is to make the network **robust against noise**.

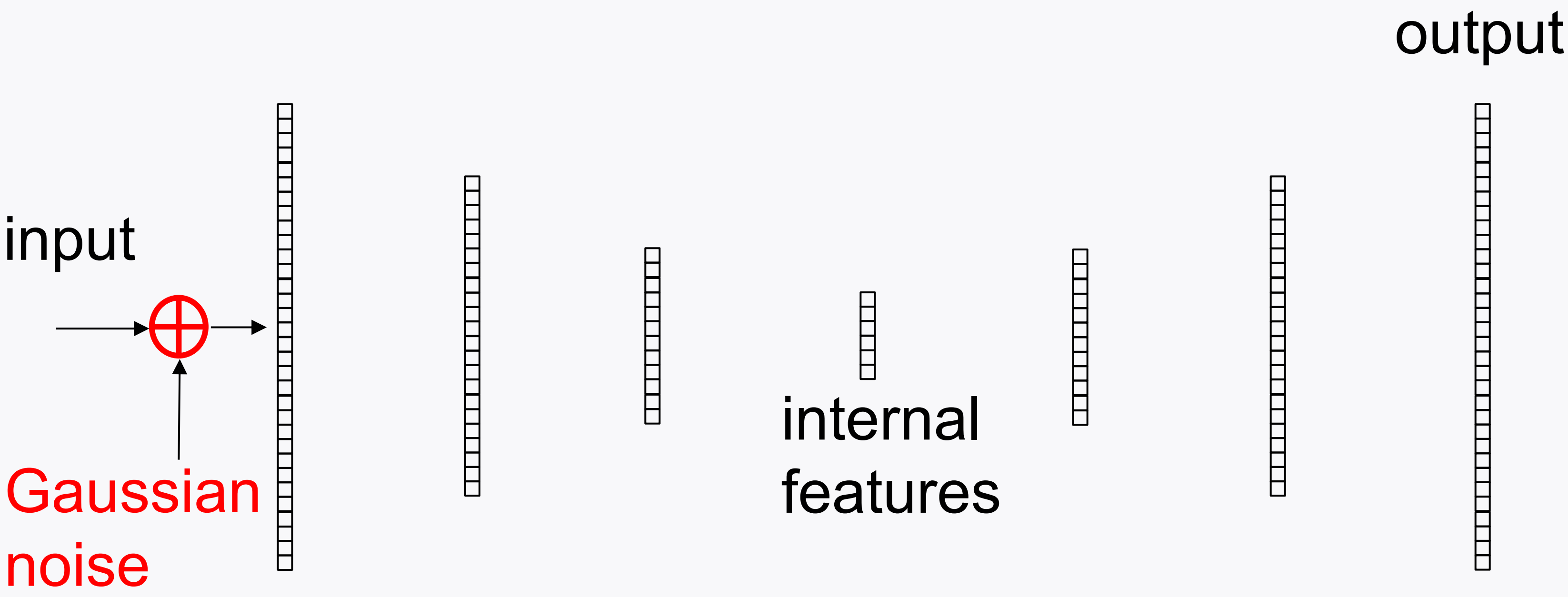
Denoising autoencoder



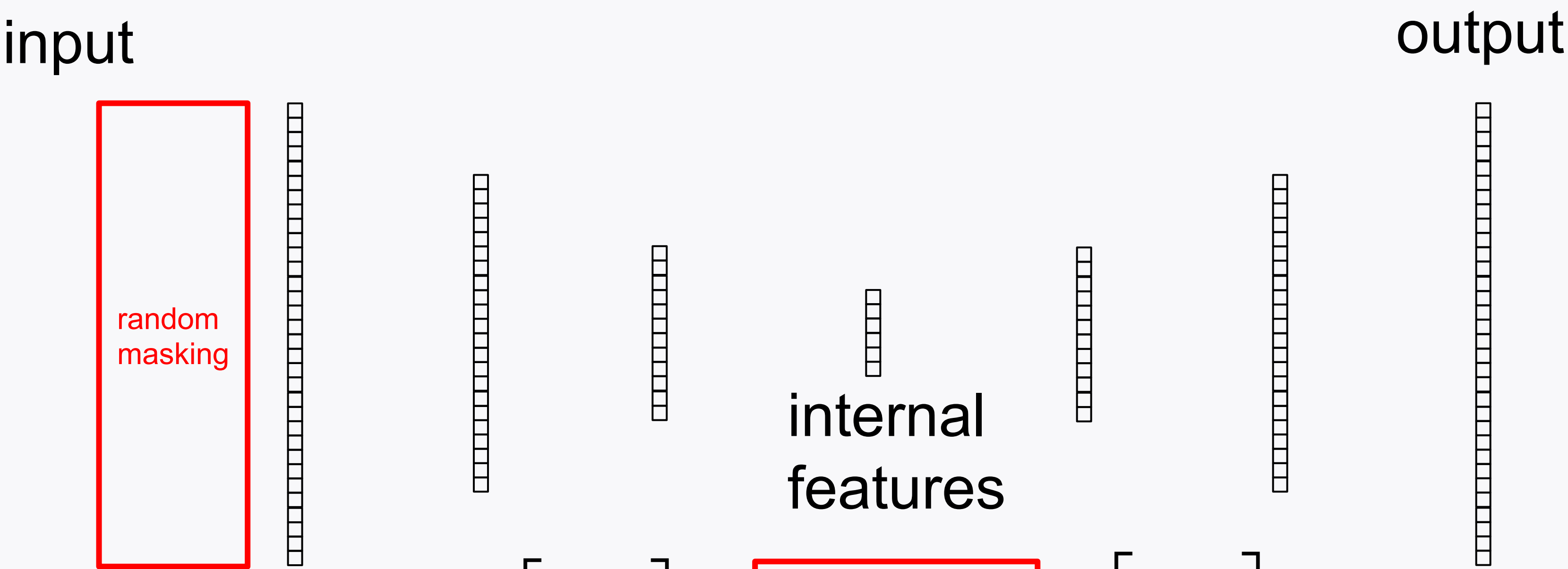
Typically noise is applied to the input.

There are two types depending on the noise pattern.

Type I: Additive Gaussian noise



Type II: Random masking

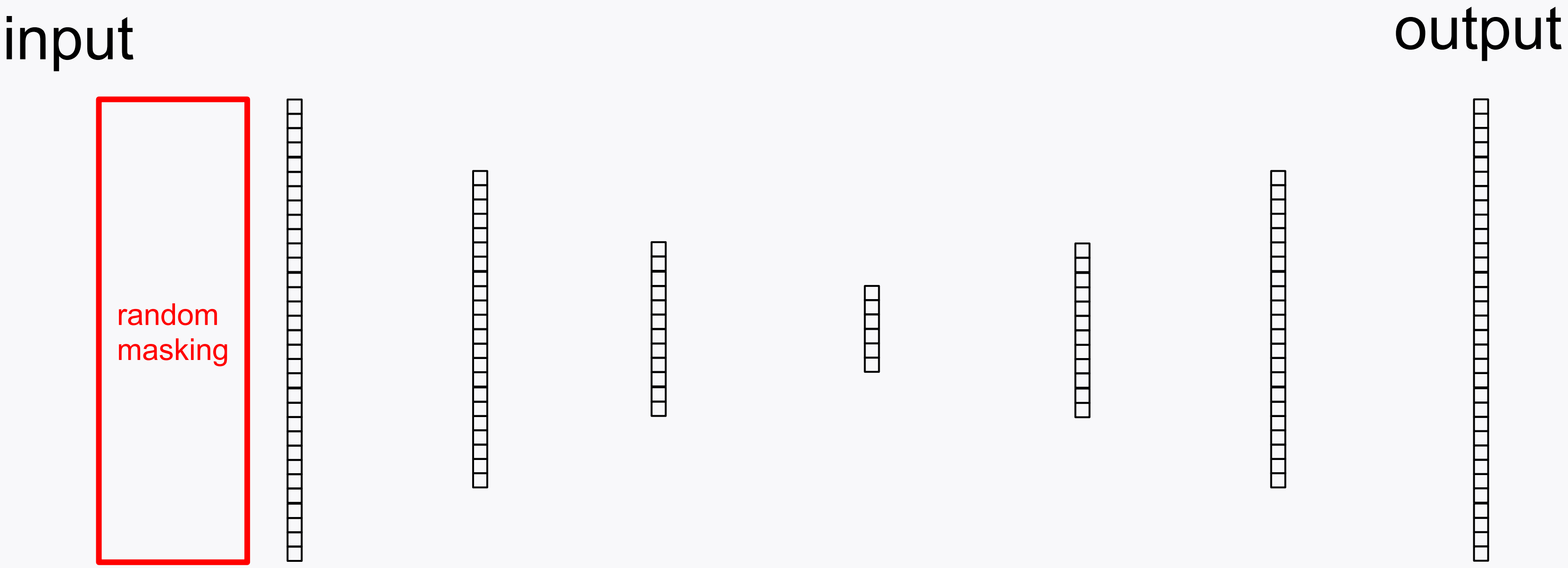


$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix}$$

random masking

$$\begin{bmatrix} x_1 \\ 0 \\ 0 \\ x_4 \\ x_5 \\ 0 \\ x_7 \\ 0 \end{bmatrix}$$

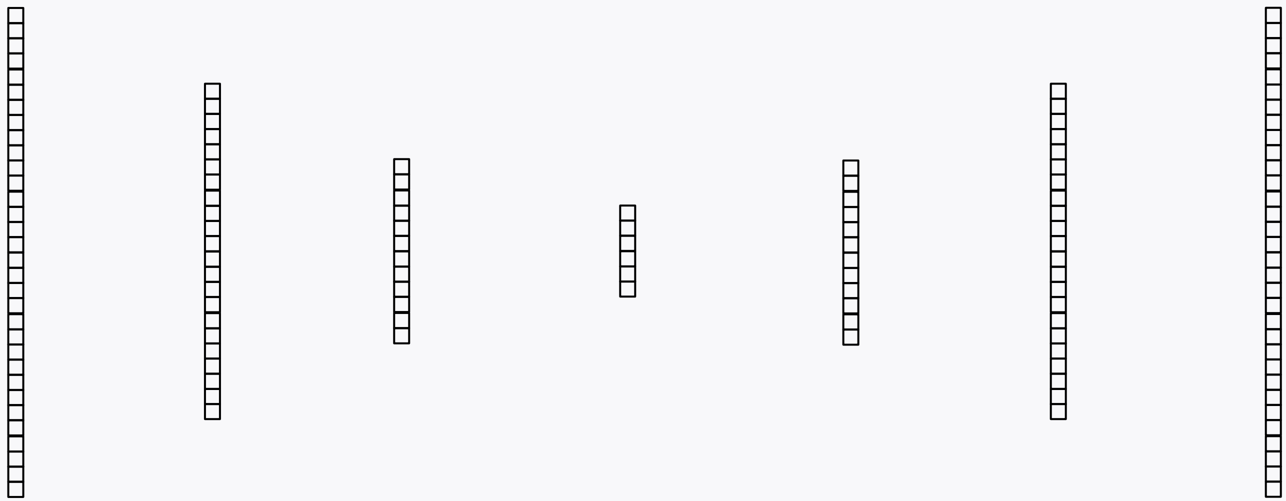
Type II: Random masking



Turns out: Gives an inspiration to matrix completion

Connection to matrix completion

$$x^{(i)} = \begin{bmatrix} x_1^{(i)} \\ 0 \\ 0 \\ x_4^{(i)} \\ x_5^{(i)} \\ 0 \\ x_7^{(i)} \\ 0 \end{bmatrix}$$



$\hat{x}^{(i)}$

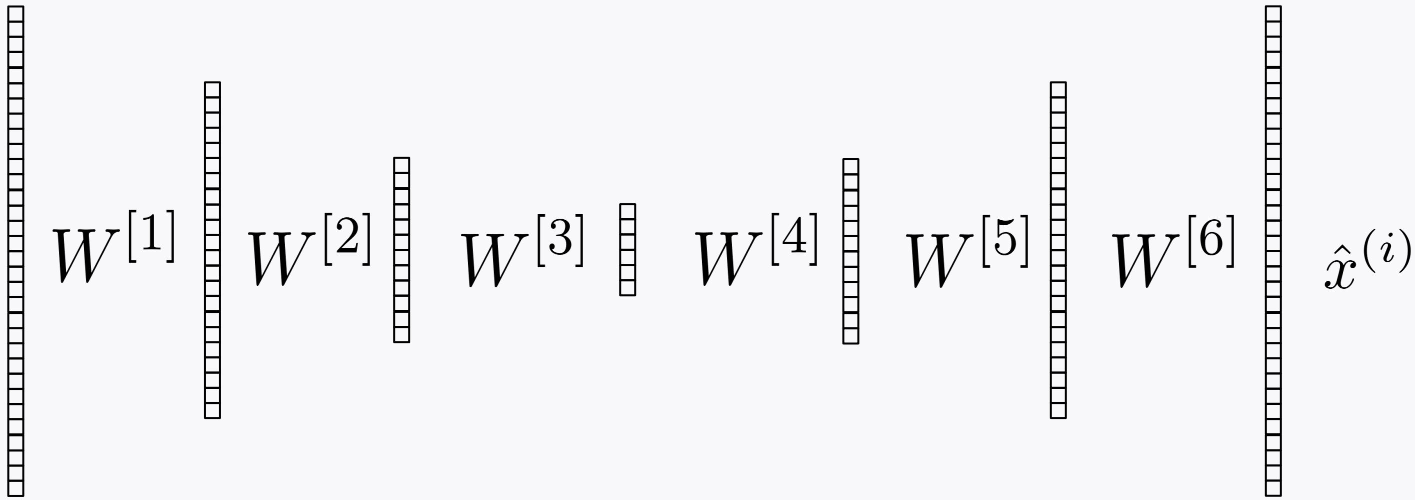
Take each example with missing entries as input to AE.

Consider output $\hat{x}^{(i)}$ as the fully-populated version.

Expect: Missing entries would be well reconstructed thanks to good performance of DAE.

How to train AE?

$$x^{(i)} = \begin{bmatrix} x_1^{(i)} \\ 0 \\ 0 \\ x_4^{(i)} \\ x_5^{(i)} \\ 0 \\ x_7^{(i)} \\ 0 \end{bmatrix}$$



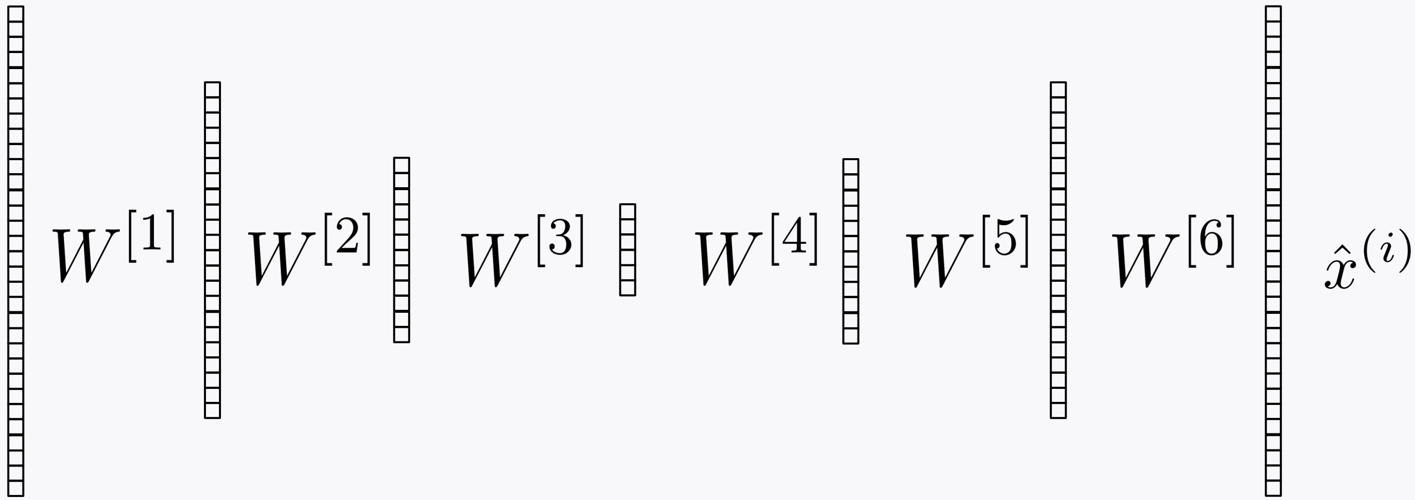
Naive method:

$$\min_{W^{[1]},\ldots,W^{[6]}} \frac{1}{m} \sum_{i=1}^{m} \sum_{j:(i,j)\in\Omega} \left(x_j^{(i)} - \hat{x}_j^{(i)}\right)^2$$

set of pair indices for observed entries

How to train AE?

$$x^{(i)} = \begin{bmatrix} x_1^{(i)} \\ 0 \\ 0 \\ x_4^{(i)} \\ x_5^{(i)} \\ 0 \\ x_7^{(i)} \\ 0 \end{bmatrix}$$



Instead: Can employ the **standard** method w/ or w/o tying weights.

What is next?

Recall: Autoencoder can serve as a generative model.

There is a more powerful generative model based on:

Generative Adversarial Networks (**GANs**)

Prior to GANs, a classical method was often employed:

Restricted Boltzmann Machines (**RBMs**)

Look ahead

Will study:

1. Generative Adversarial Networks (**GANs**)

2. Restricted Boltzmann Machines (**RBMs**)