Graph Autoencoders (GAE) & Graph Variational Autoencoders (VGAE)

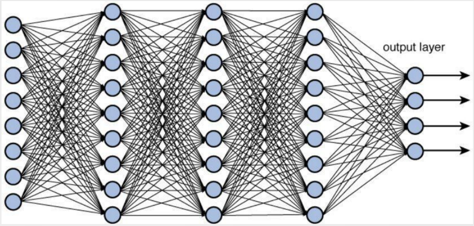
Antonio Longa[1,2]

MobS[1] Lab, Fondazione Bruno Kessler,Trento, Italy
SML[2] Lab, University of Trento, Italy

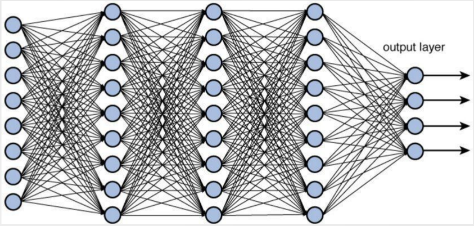
TABLE OF CONTENTS

01 Autoencoders

What does a Deep neural network do?



01 Autoencoders

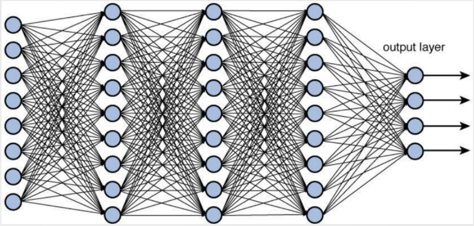


What does a Deep neural network do?

It learns **important features** from the input.

01 Autoencoders

What does a Deep neural network do?



It learns **important features** from the input.

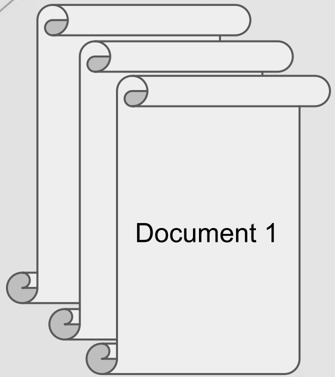
Features that allow to do a specific task on the data.
I.e classification, regression, generalization etc

01 Autoencoders

Can we compress our input data?

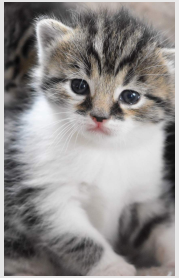
01 Autoencoders

Can we compress our input data?



[10,1,6,0.4,11,4]

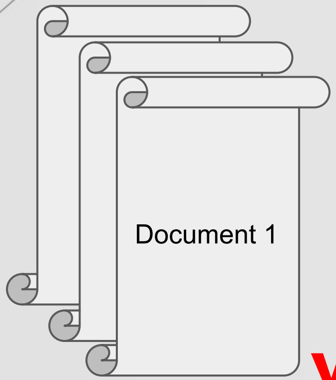
Low dimensional vector



[74,22,45,1,12,4,4,4]

01 Autoencoders

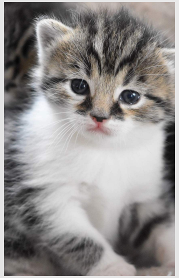
Can we compress our input data?



[10,1,6,0.4,11,4]

Low dimensional vector

Yes, with Autoencoders



[74,22,45,1,12,4,4,4]

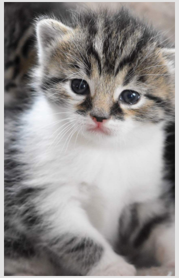
01 Autoencoders

Autoencoders are Neural networks that works in an **unsupervised** manner

01 Autoencoders

Autoencoders are Neural networks that works in an **unsupervised** manner

We do **not need labeled data**



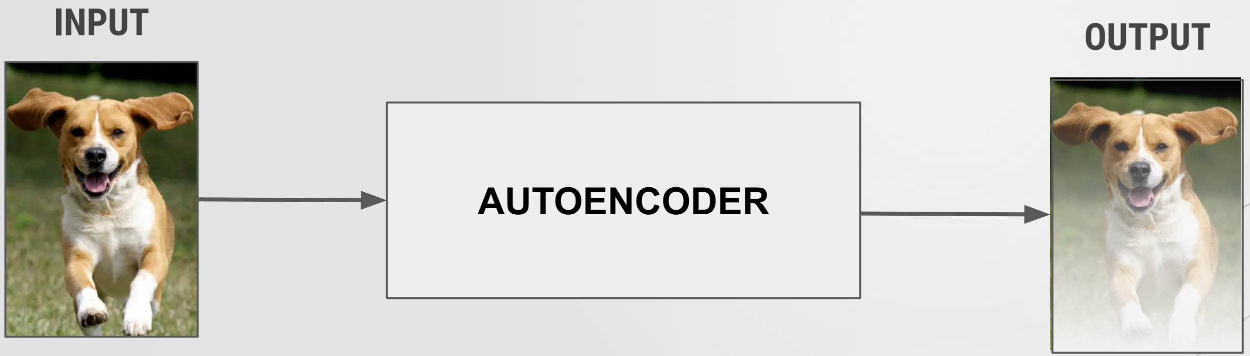
01 Autoencoders

How can they **work** without any labeled data?

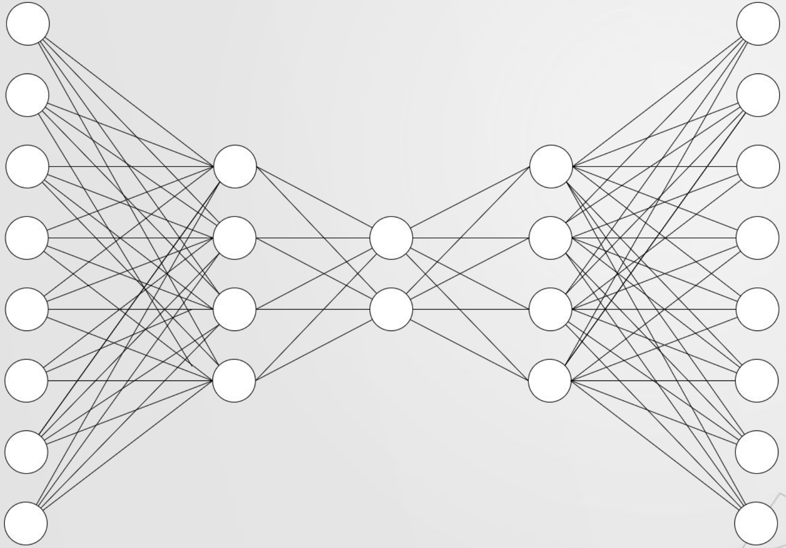
01 Autoencoders

How can they **work** without any labeled data?

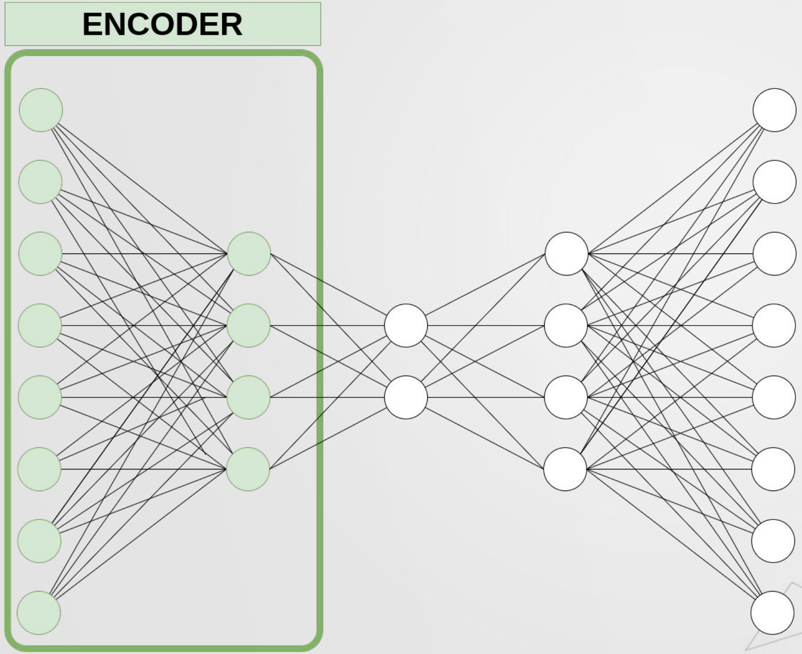
The idea is to **reconstruct** the **input**



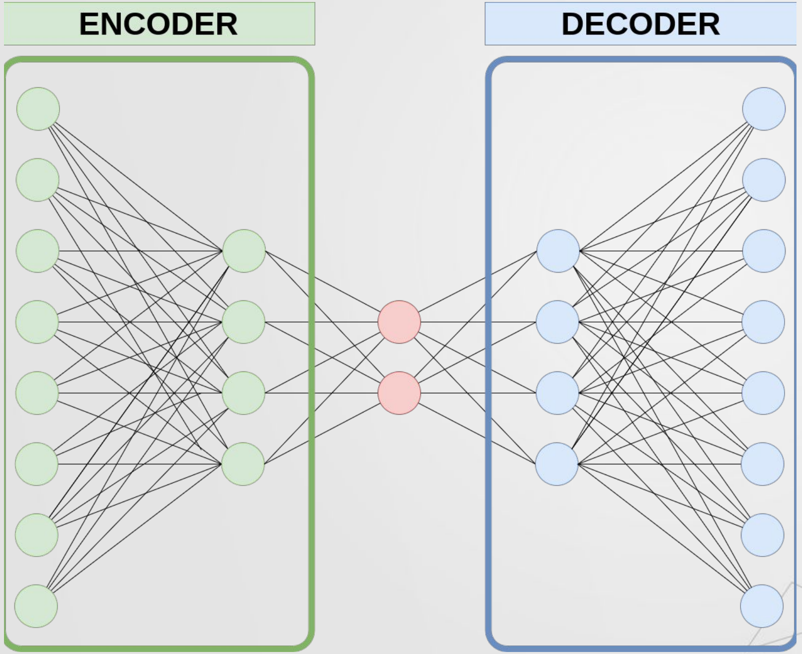
01 Autoencoders



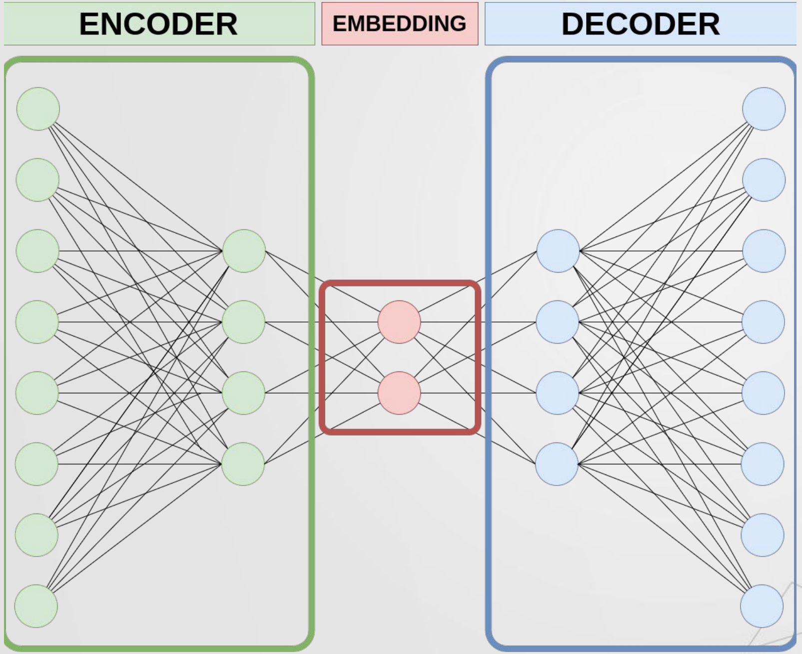
01 Autoencoders



01 Autoencoders

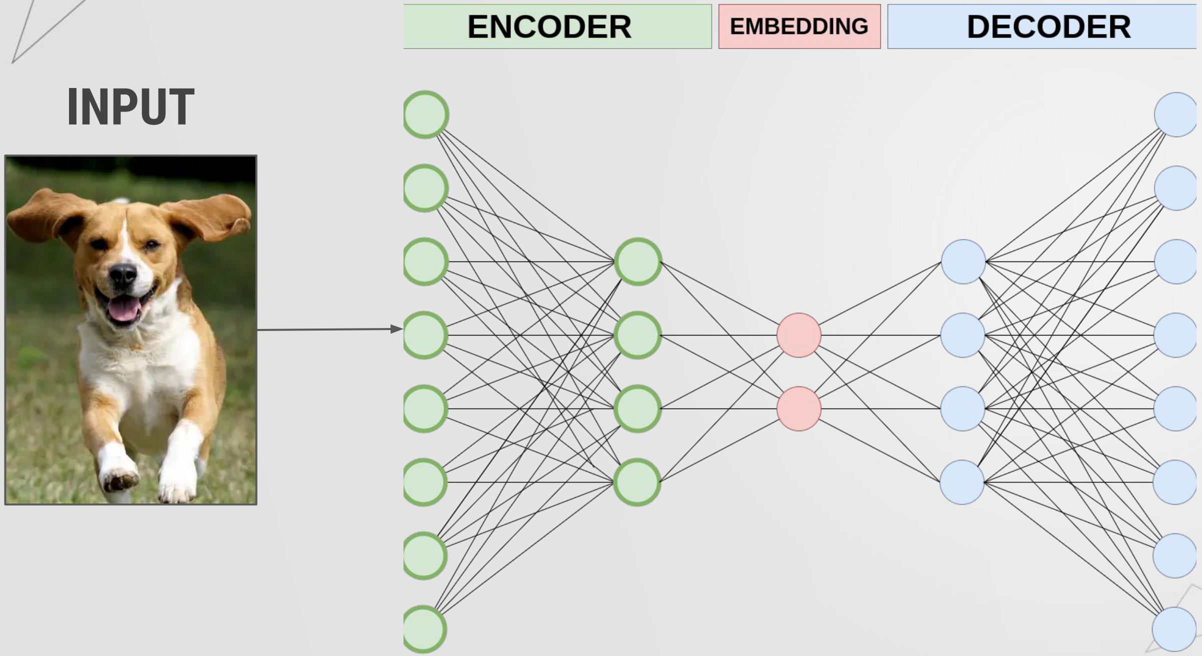


01 Autoencoders

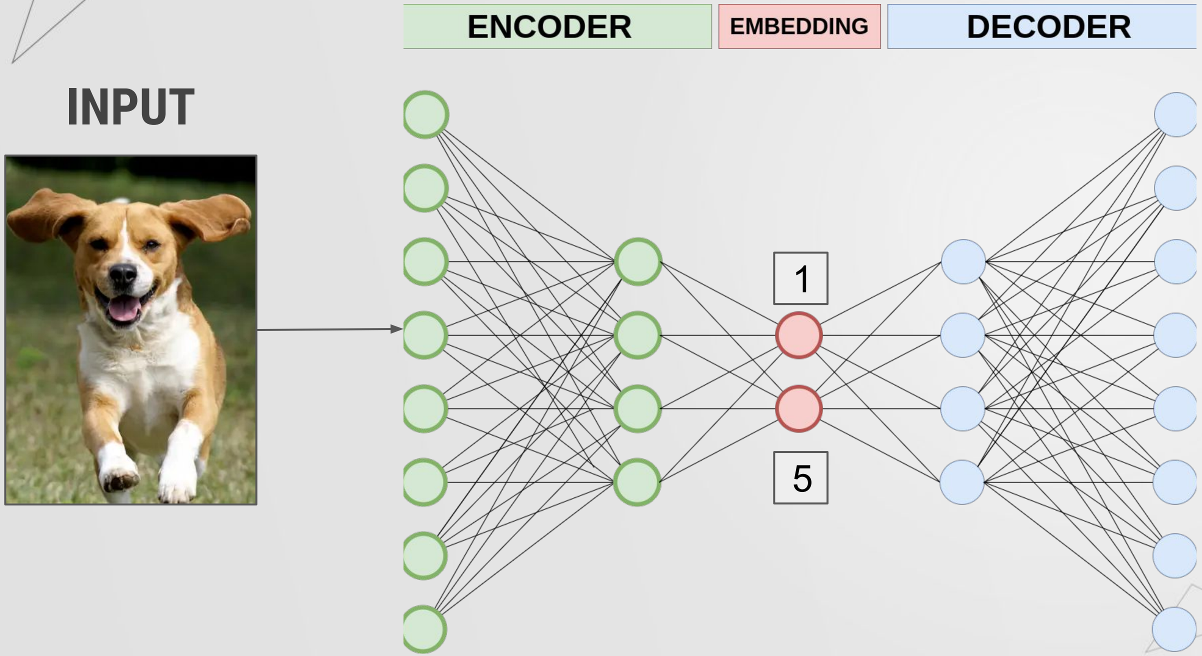


01 Autoencoders

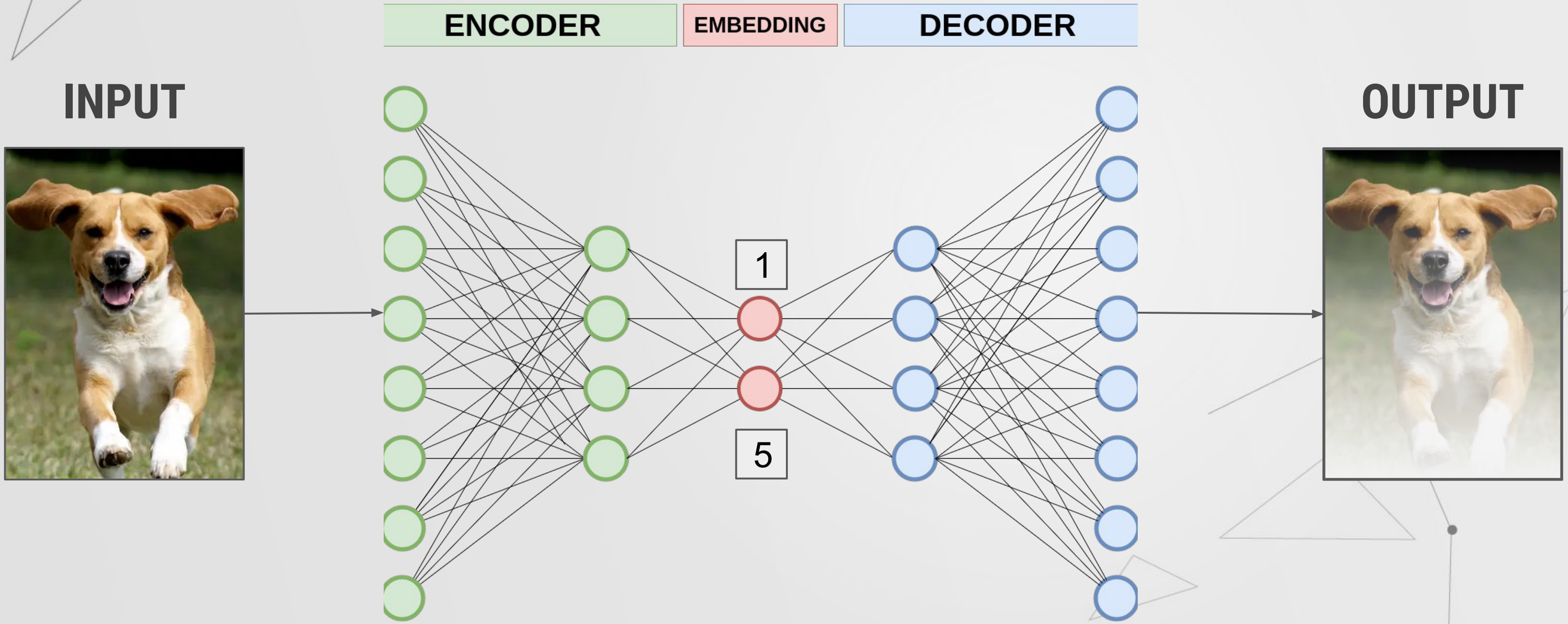
INPUT



01 Autoencoders



01 Autoencoders



01 Autoencoders

LOSS = similarity (

INPUT

,

OUTPUT

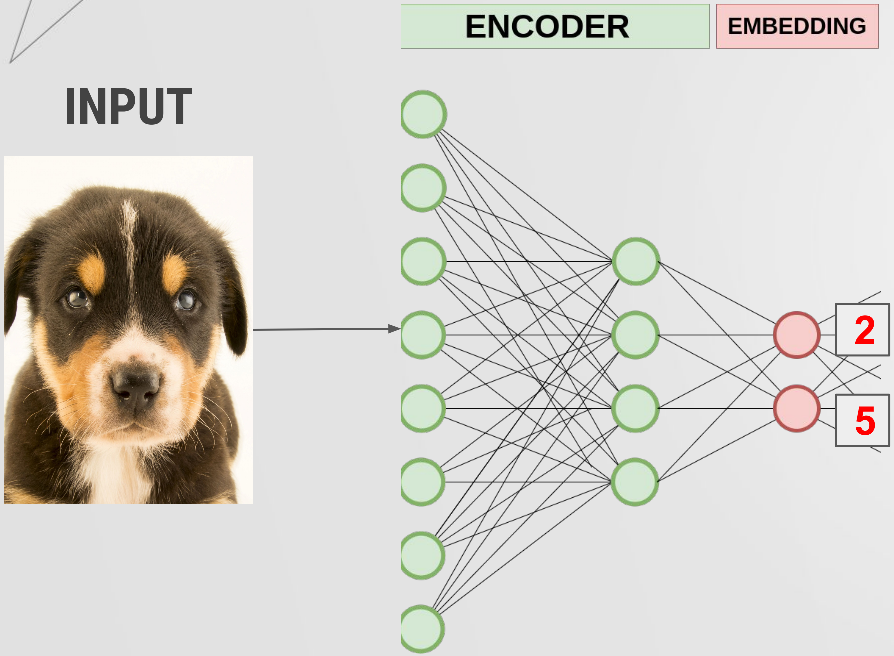
)

01 Autoencoders

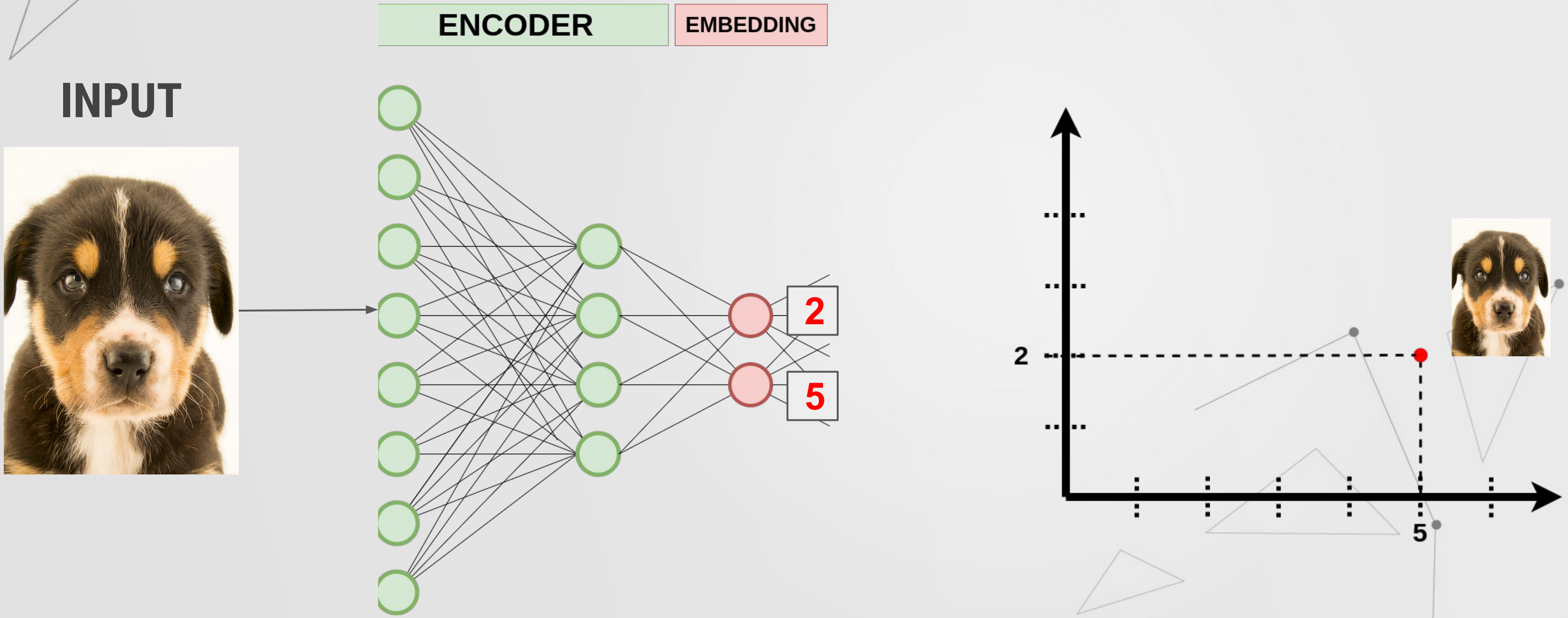
How to use an autoencoder?

01 Autoencoders

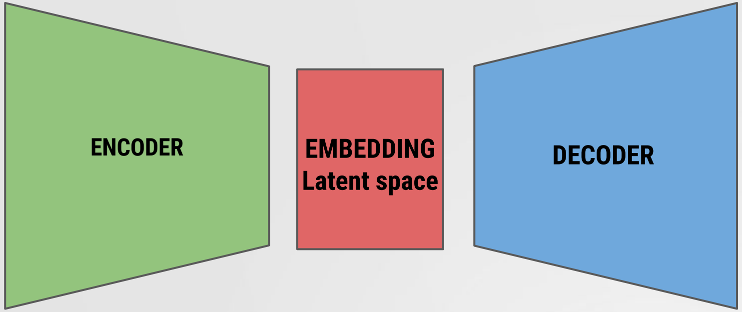
How to use an autoencoder?



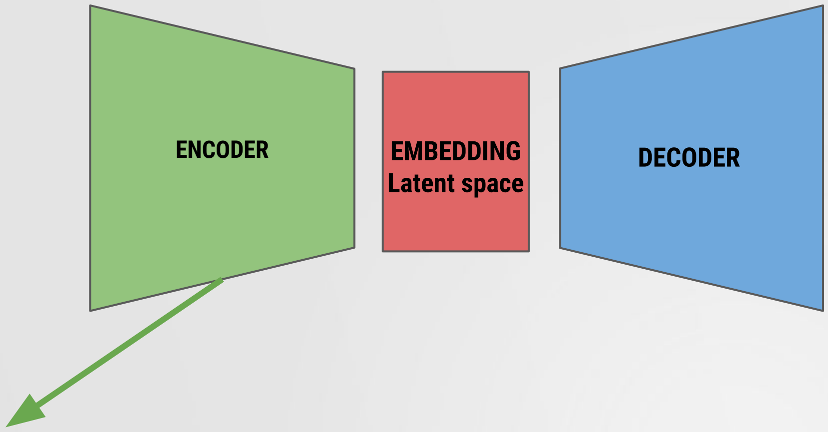
How to use an autoencoder?



02 Graph Autoencoders (GAE) Theory

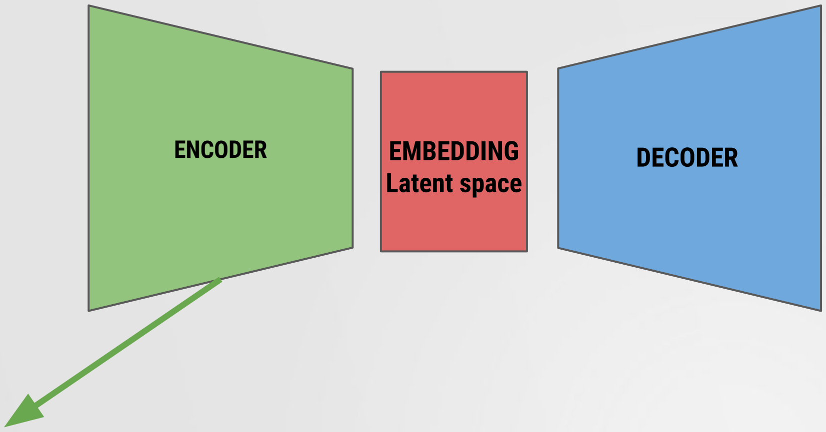


02 Graph Autoencoders (GAE) Theory



ONE convolutional Graph neural network:

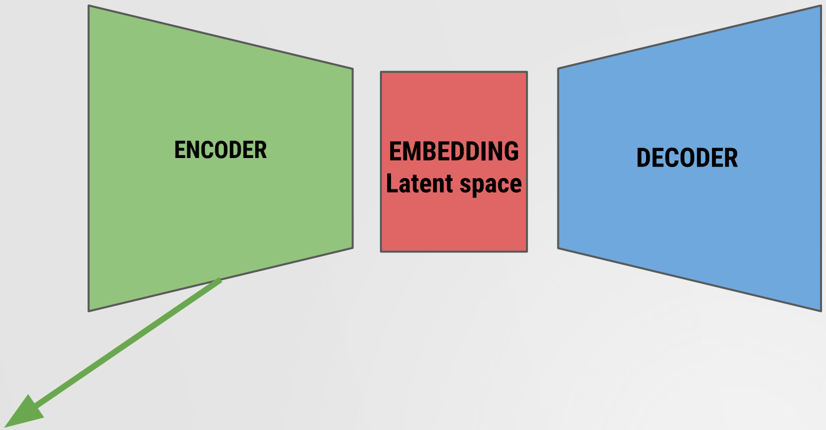
02 Graph Autoencoders (GAE) Theory



ONE convolutional Graph neural network:

- produces a low dimensional embedding representation

02 Graph Autoencoders (GAE) Theory



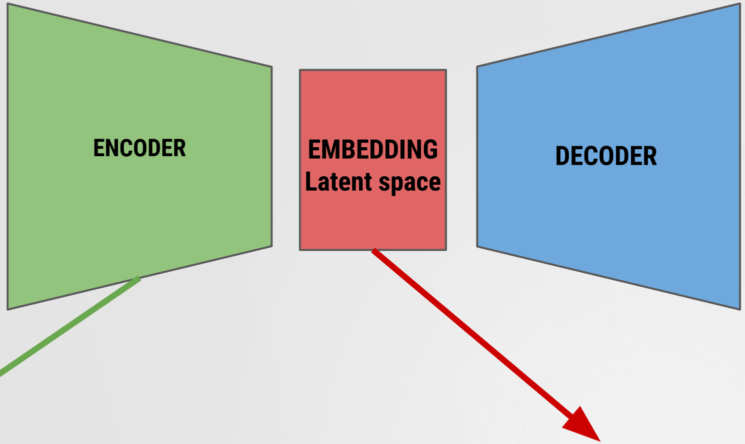
ONE convolutional Graph neural network:

- produces a low dimensional embedding representation

$$\bar{X} = GCN(A, X) = ReLU(\tilde{A}XW_0)$$

$$\text{with } \tilde{A} = D^{-1/2}AD^{-1/2}$$

02 Graph Autoencoders (GAE) Theory



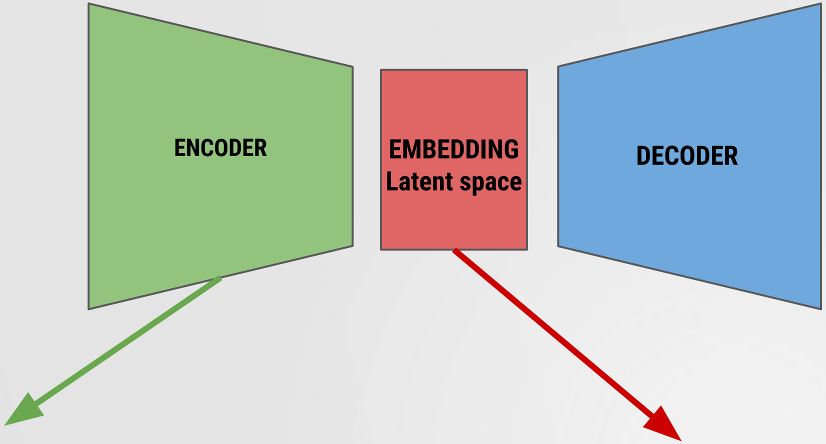
ONE convolutional Graph neural network:

- produces a low dimensional embedding representation

$$\bar{X} = GCN(A, X) = ReLU(\tilde{A}XW_0)$$

$$\text{with } \tilde{A} = D^{-1/2} A D^{-1/2}$$

02 Graph Autoencoders (GAE) Theory



ONE convolutional Graph neural network:

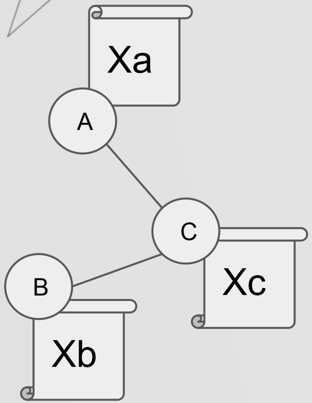
- produces a low dimensional embedding representation

$$Z = \bar{X}$$

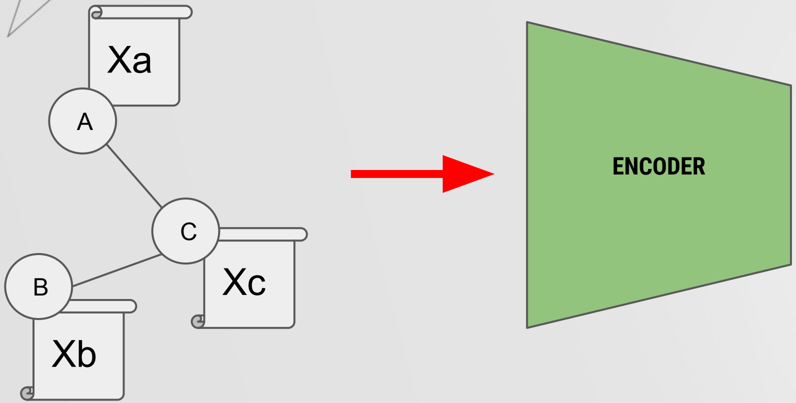
$$\bar{X} = GCN(A, X) = ReLU(\tilde{A}XW_0)$$

$$\text{with } \tilde{A} = D^{-1/2}AD^{-1/2}$$

02 Graph Autoencoders (GAE) Theory

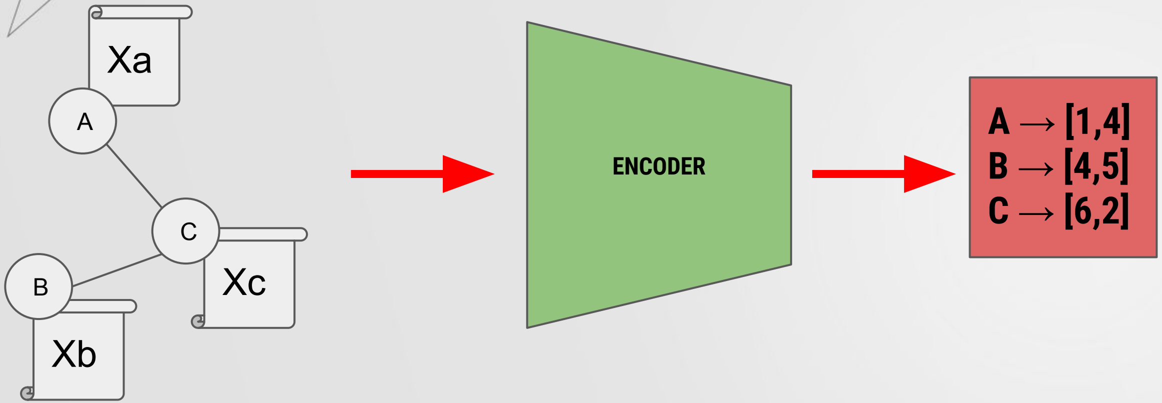


02 Graph Autoencoders (GAE) Theory

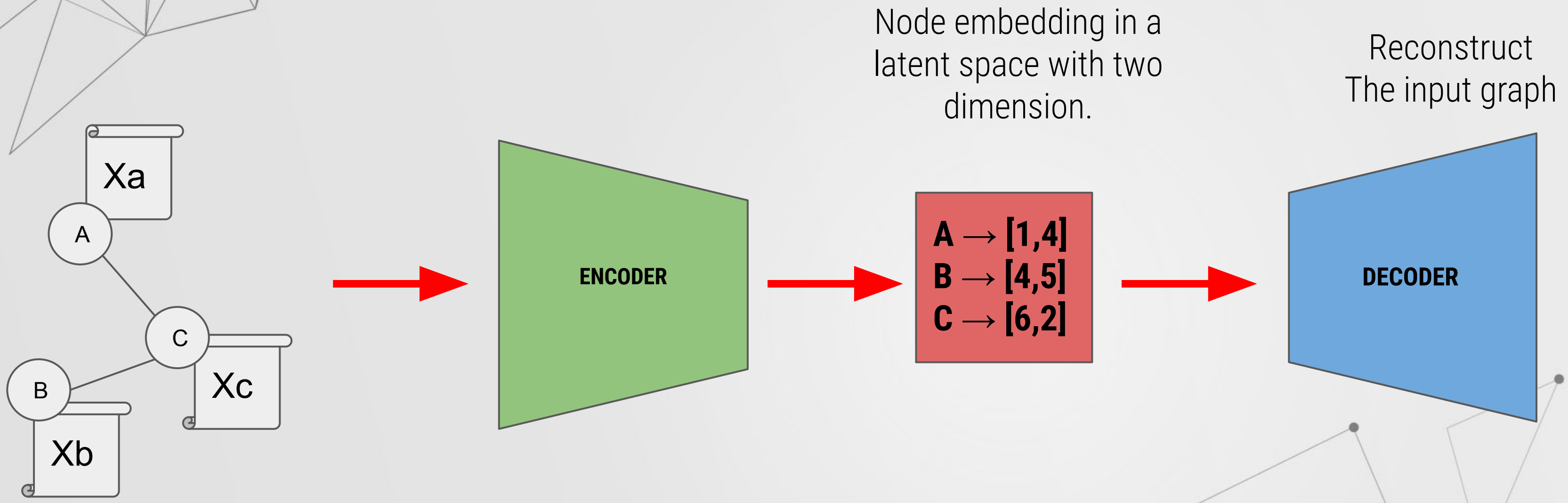


02 Graph Autoencoders (GAE) Theory

Node embedding in a latent space with two dimension.



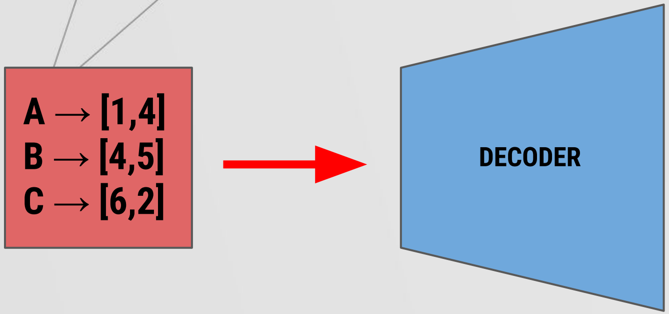
02 Graph Autoencoders (GAE) Theory



02 Graph Autoencoders (GAE) Theory

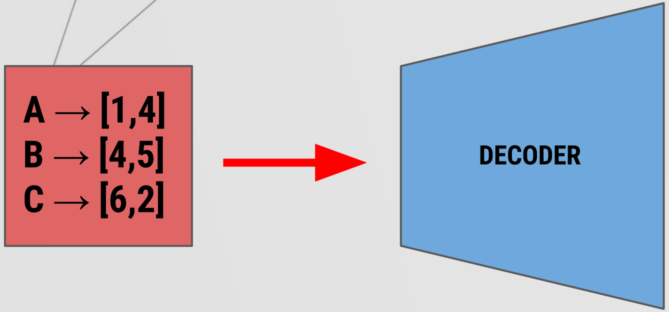
Reconstruct
The input graph

Inner product
Between latent variable Z



02 Graph Autoencoders (GAE) Theory

Reconstruct
The input graph



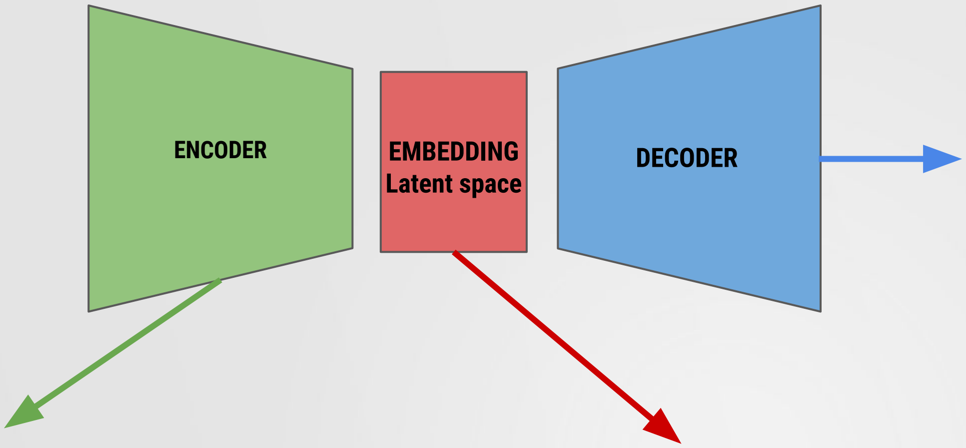
Inner product
Between latent variable Z

$$Adj_{(A,B)} = \text{sigmoid}([1,4] * [4,5]^{T})$$

$$Adj_{(B,C)} = \text{sigmoid}([4,5] * [6,2]^{T})$$

……

02 Graph Autoencoders (GAE) Theory



$$Z = \bar{X}$$

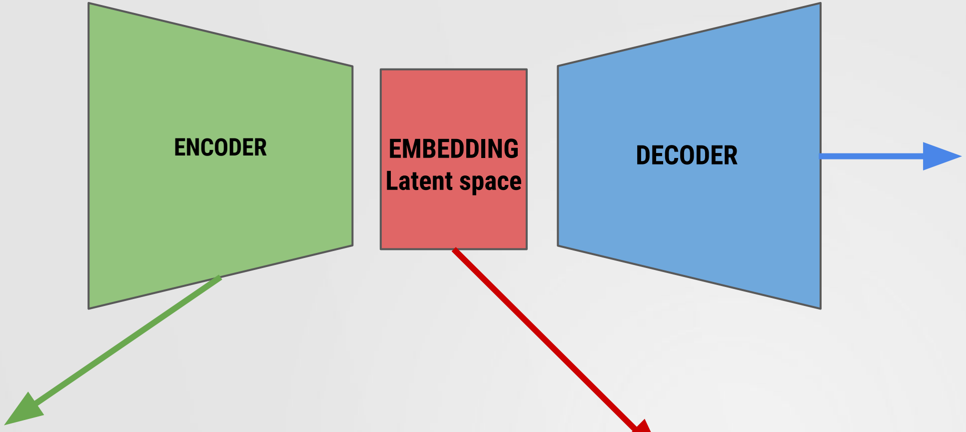
ONE convolutional Graph neural network:

- produces a low dimensional embedding representation

$$\bar{X} = GCN(A, X) = ReLU(\tilde{A}XW_0)$$

$$\text{with } \tilde{A} = D^{-1/2}AD^{-1/2}$$

02 Graph Autoencoders (GAE) Theory



So far we have an **embedding** in a latent space **for each node** of the graph.

$$Z = \bar{X}$$

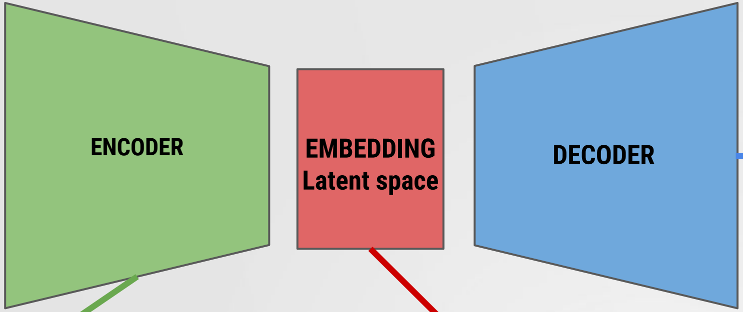
ONE convolutional Graph neural network:

- produces a low dimensional embedding representation

$$\bar{X} = GCN(A, X) = ReLU(\tilde{A}XW_0)$$

$$\text{with } \tilde{A} = D^{-1/2}AD^{-1/2}$$

02 Graph Autoencoders (GAE) Theory



So far we have an **embedding** in a latent space **for each node** of the graph.

We want to **reconstruct** the adjacency matrix **A**

ONE convolutional Graph neural network:

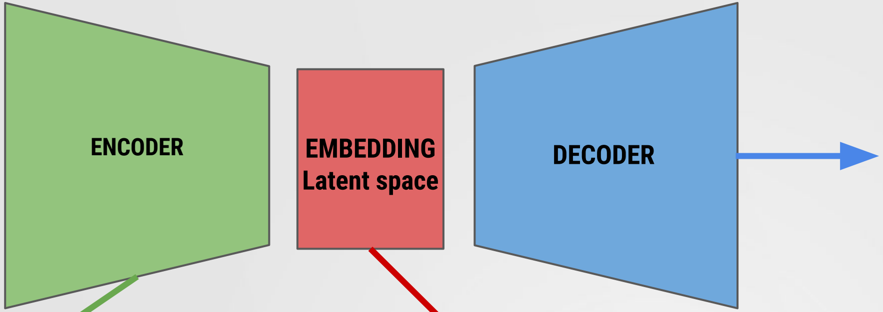
- produces a low dimensional embedding representation

$$Z = \bar{X}$$

$$\bar{X} = GCN(A, X) = ReLU(\tilde{A} X W_0)$$

$$\text{with } \tilde{A} = D^{-1/2} A D^{-1/2}$$

02 Graph Autoencoders (GAE) Theory



So far we have an **embedding** in a latent space **for each node** of the graph.

We want to **reconstruct** the adjacency matrix **A**

Inner product
Between latent variable Z

$$\hat{A} = \text{logistic sigmoid}\left(zz^T\right)$$

ONE convolutional Graph neural network:

- produces a low dimensional embedding representation

$$Z = \bar{X}$$

$$\bar{X} = GCN(A, X) = ReLU(\tilde{A}XW_0)$$

$$\text{with } \tilde{A} = D^{-1/2} A D^{-1/2}$$

03 Graph Autoencoders (GAE) Practice

CLASS **GAE** (**encoder**, **decoder**=None) [source]

The Graph Auto-Encoder model from the "Variational Graph Auto-Encoders" paper based on user-defined encoder and decoder models.

PARAMETERS

- **encoder** (*Module*) – The encoder module.
- **decoder** (*Module, optional*) – The decoder module. If set to `None`, will default to the `torch_geometric.nn.models.InnerProductDecoder`. (default: `None`)

decode (*args, **kwargs) [source]

Runs the decoder and computes edge probabilities.

encode (*args, **kwargs) [source]

Runs the encoder and computes node-wise latent variables.

recon_loss (**z**, **pos_edge_index**, **neg_edge_index**=None) [source]

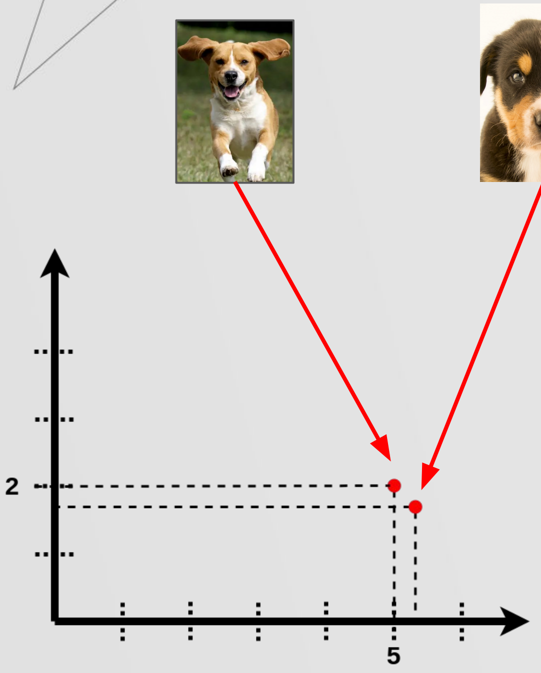
Given latent variables `z`, computes the binary cross entropy loss for positive edges `pos_edge_index` and negative sampled edges.

PARAMETERS

- **z** (*Tensor*) – The latent space $\mathbf{Z}$.
- **pos_edge_index** (*LongTensor*) – The positive edges to train against.
- **neg_edge_index** (*LongTensor, optional*) – The negative edges to train against. If not given, uses negative sampling to calculate negative edges. (default: `None`)

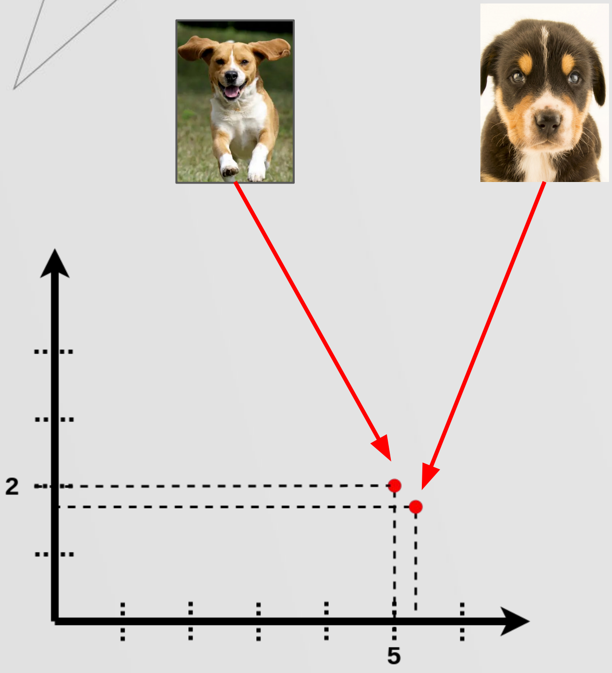
Jupyter Notebook

Autoencoder (encoder)

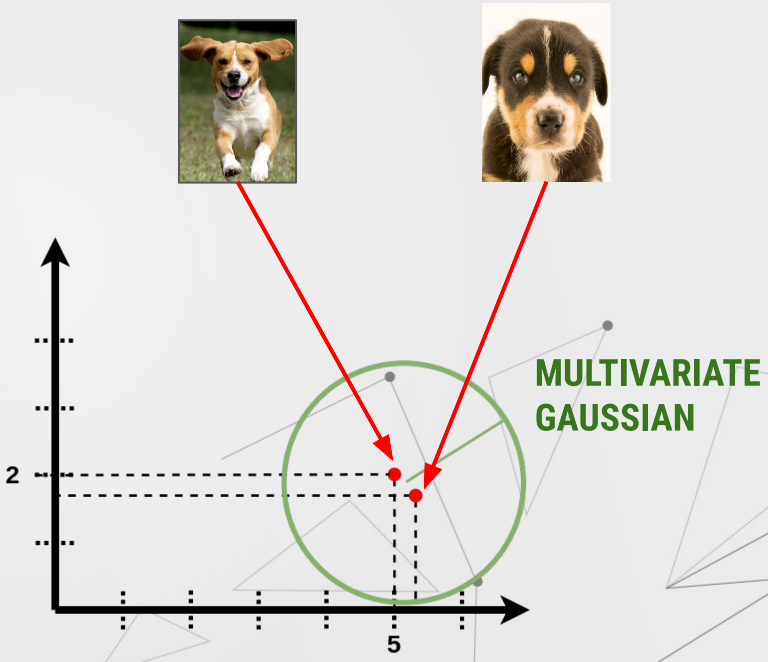


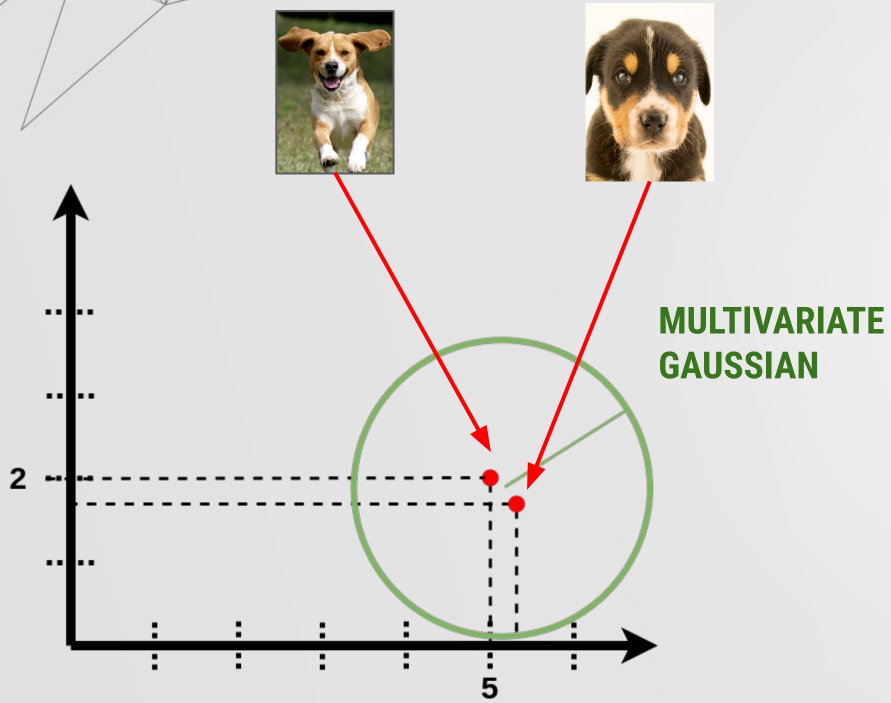
04 Variational Autoencoders

Autoencoder (encoder)



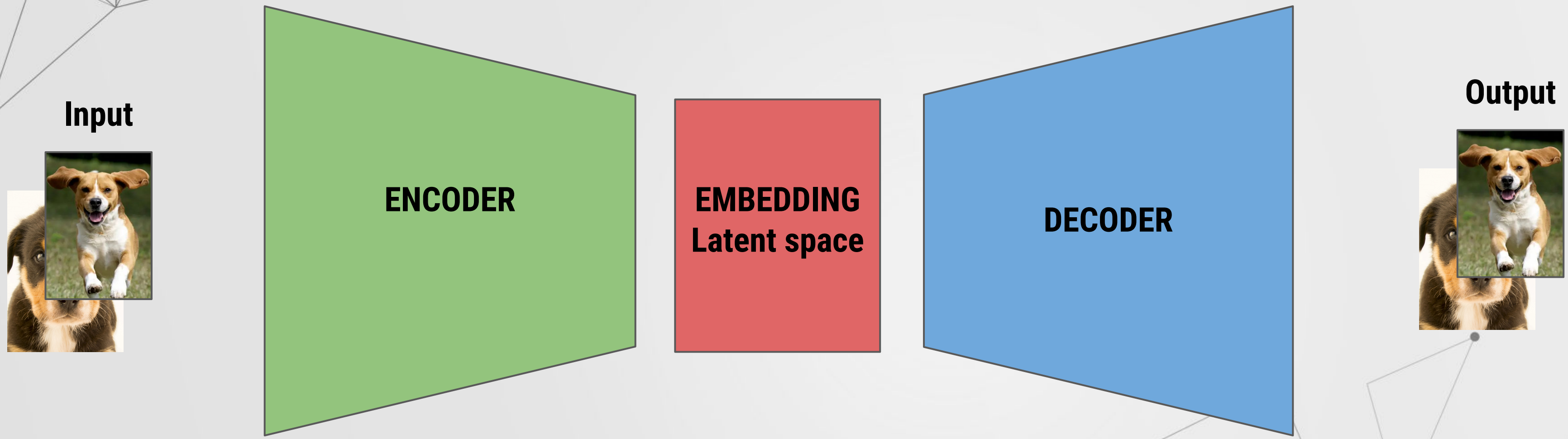
Variational Autoencoder (encoder)



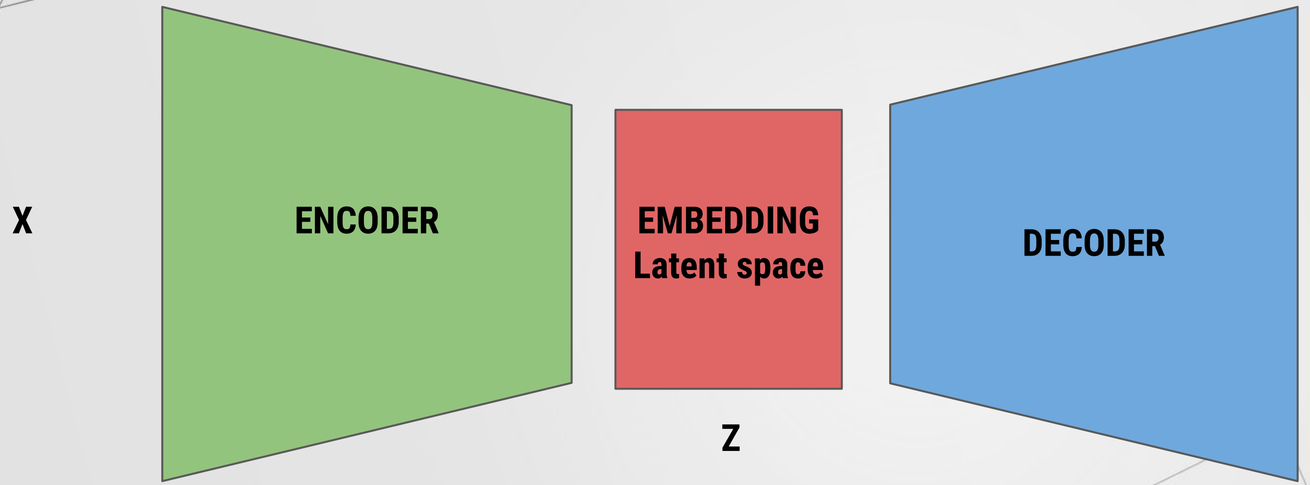


$$\mu, \sigma^2$$

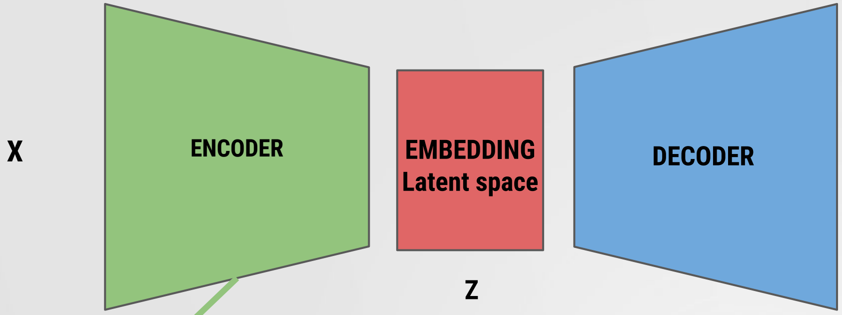
04 Variational Autoencoders



04 Variational Autoencoders

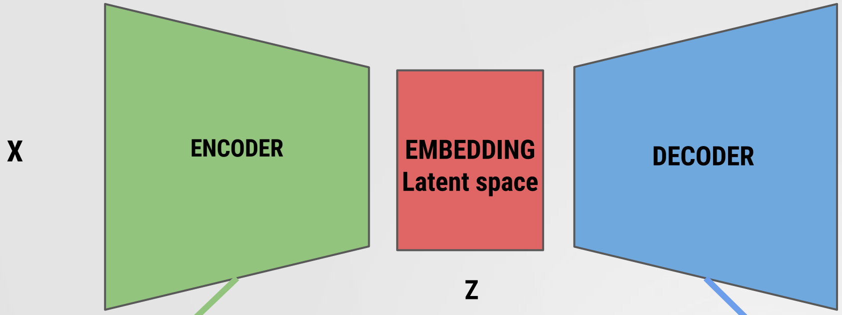


04 Variational Autoencoders



$$p(z|x)$$

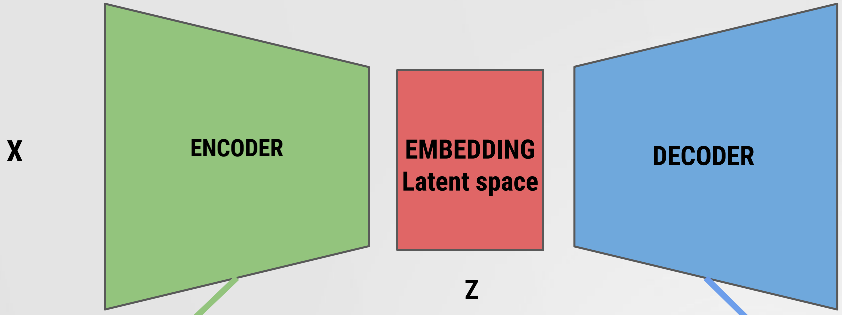
04 Variational Autoencoders



$$p(z|x)$$

$$q(x|z)$$

04 Variational Autoencoders

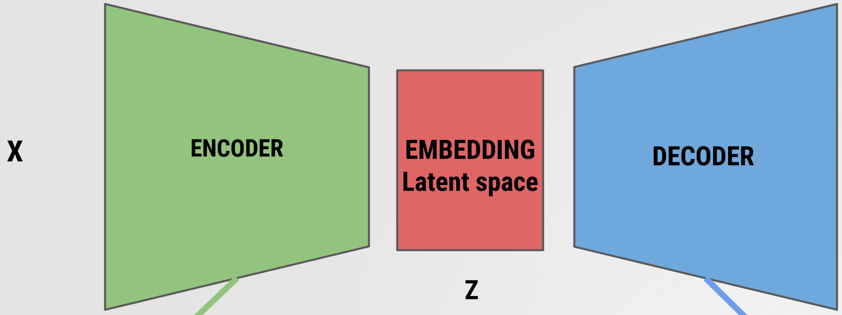


$$p(z|x)$$

$$q(x|z) = \frac{q(z|x)q(x)}{\boxed{q(z)}}$$

Multivariate Gaussian

04 Variational Autoencoders



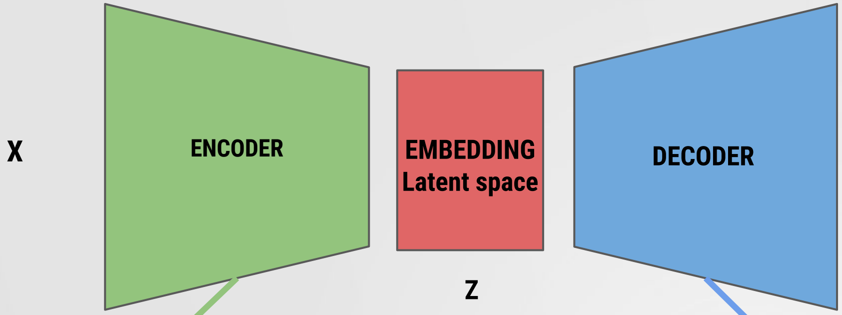
$$p(z|x)$$

Distribution of the input

$$q(x|z) = \frac{q(z|x)\boxed{q(x)}}{\boxed{q(z)}}$$

Multivariate Gaussian

04 Variational Autoencoders



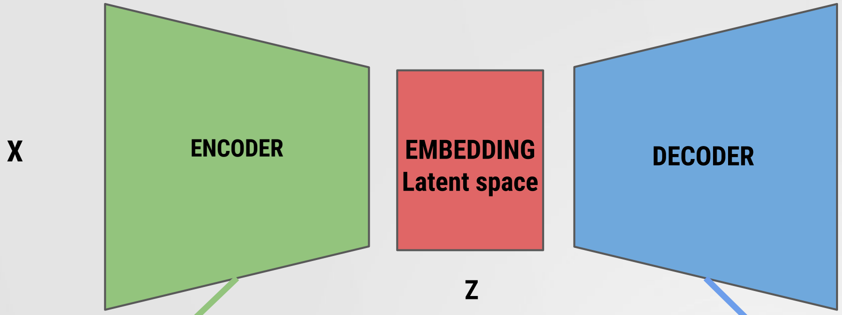
$$p(z|x)$$

Distribution of the input

$$q(x|z) = \frac{\boxed{q(z|x)}\boxed{q(x)}}{\boxed{q(z)}}$$

Multivariate Gaussian

04 Variational Autoencoders



$$\boxed{p(z|x)}$$

Distribution of the input

$$q(x|z) = \frac{\boxed{q(z|x)}\boxed{q(x)}}{\boxed{q(z)}}$$

Multivariate Gaussian

04 Variational Autoencoders

$$p(z|x) \quad q(z|x)$$

As much similar as possible…

$$p(z|x) \quad q(z|x)$$

As much similar as possible...

KL-Divergence → measures the distance between distributions

$$KL(q(z|x)||p(z|x))$$

$$p(z|x) \quad q(z|x)$$

As much similar as possible...

KL-Divergence → measures the distance between distributions

$$KL(q(z|x)||p(z|x))$$

$$\min KL(q(z|x)||p(z|x))$$

04 Variational Autoencoders

$$p(z|x) \quad q(z|x)$$

As much similar as possible...

KL-Divergence → measures the distance between distributions

$$KL(q(z|x)||p(z|x))$$

$$\min KL(q(z|x)||p(z|x))$$

$$\min KL(q(z|x)||p(z|x))$$

Is it done?

04 Variational Autoencoders

$$\min KL(q(z|x)||p(z|x))$$

Is it done?

We cannot compute q(z|x)

LOSS = similarity (

INPUT

,

OUTPUT

)

$$Loss = -E_{p(z|x)} \log q(z|x) + KL(p(z|x)||q(z))$$

04 Variational Autoencoders

$$Loss = \boxed{-E_{p(z|x)} \log q(z|x)} + KL(p(z|x)||q(z))$$

Variational Lower Bound [Reconstruction error]

How well the network is able to reconstruct the input?

04 Variational Autoencoders

$$Loss = \boxed{-E_{p(z|x)} \log q(z|x)} + \boxed{KL(p(z|x)||q(z))}$$

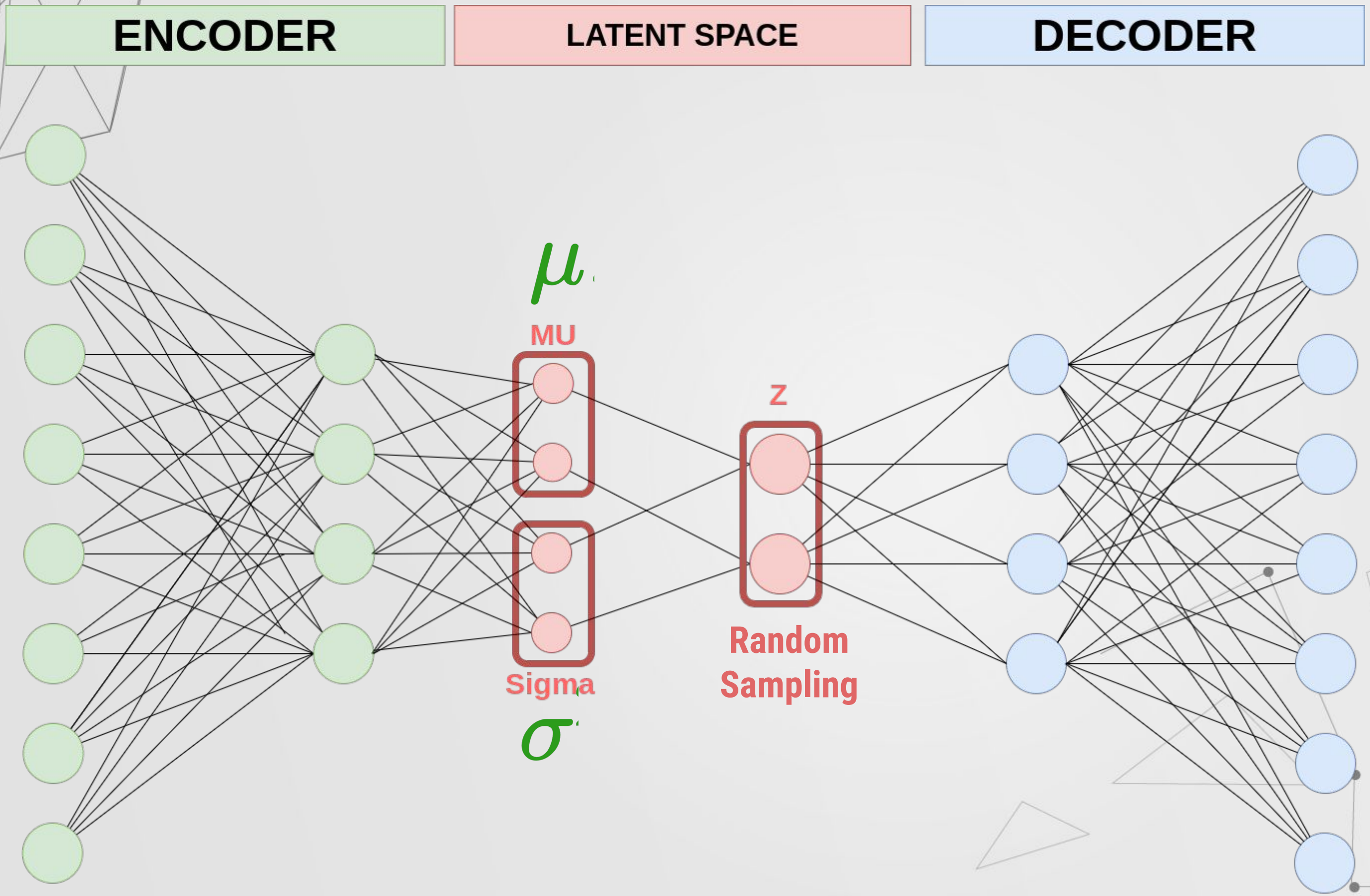
Variational Lower Bound [Reconstruction error]

How well the network is able to reconstruct the input?

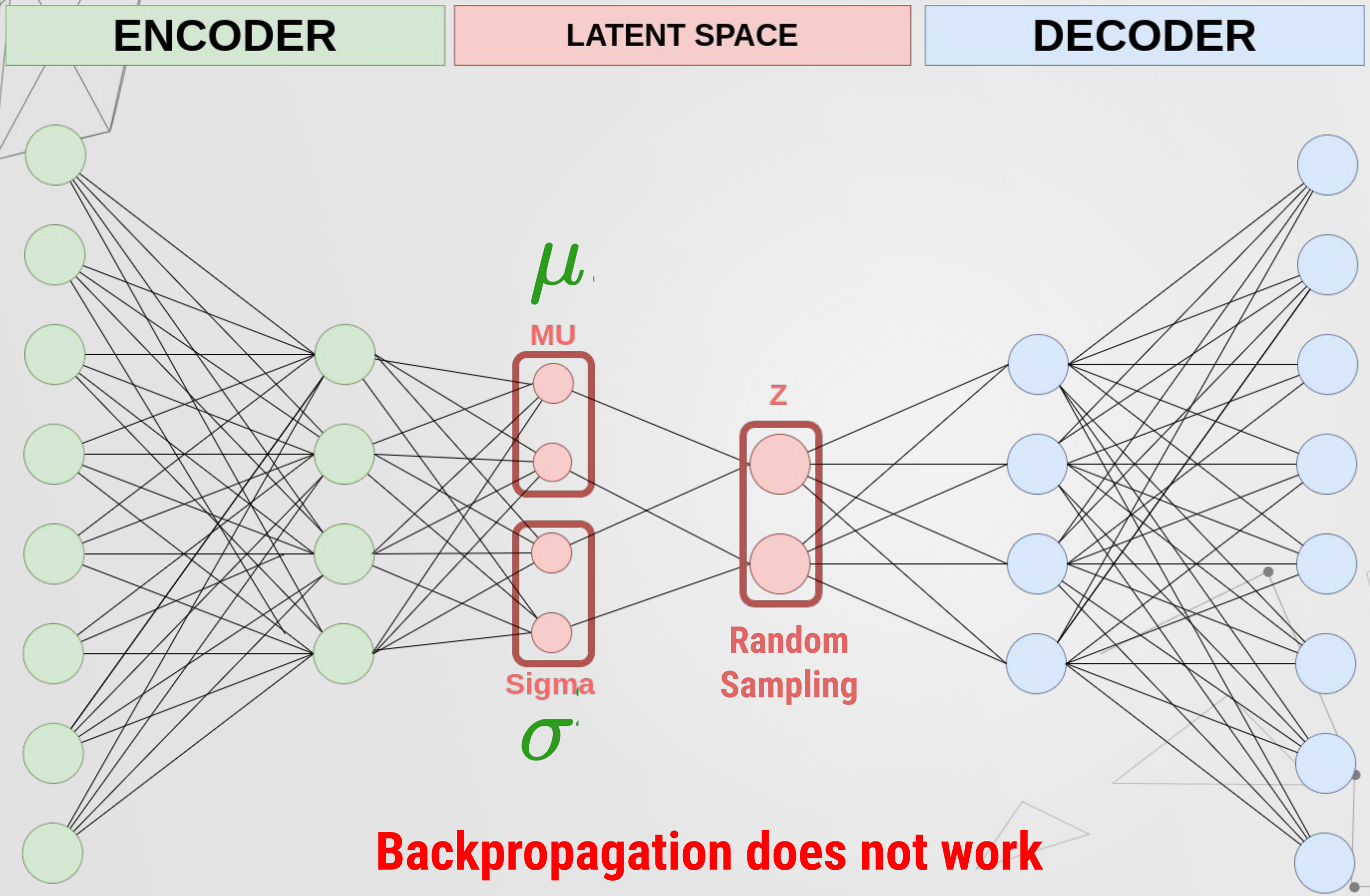
Regularizer

Keep the distributions of q(z|x) and p(z|x) as much similar as possible.

04 Variational Autoencoders

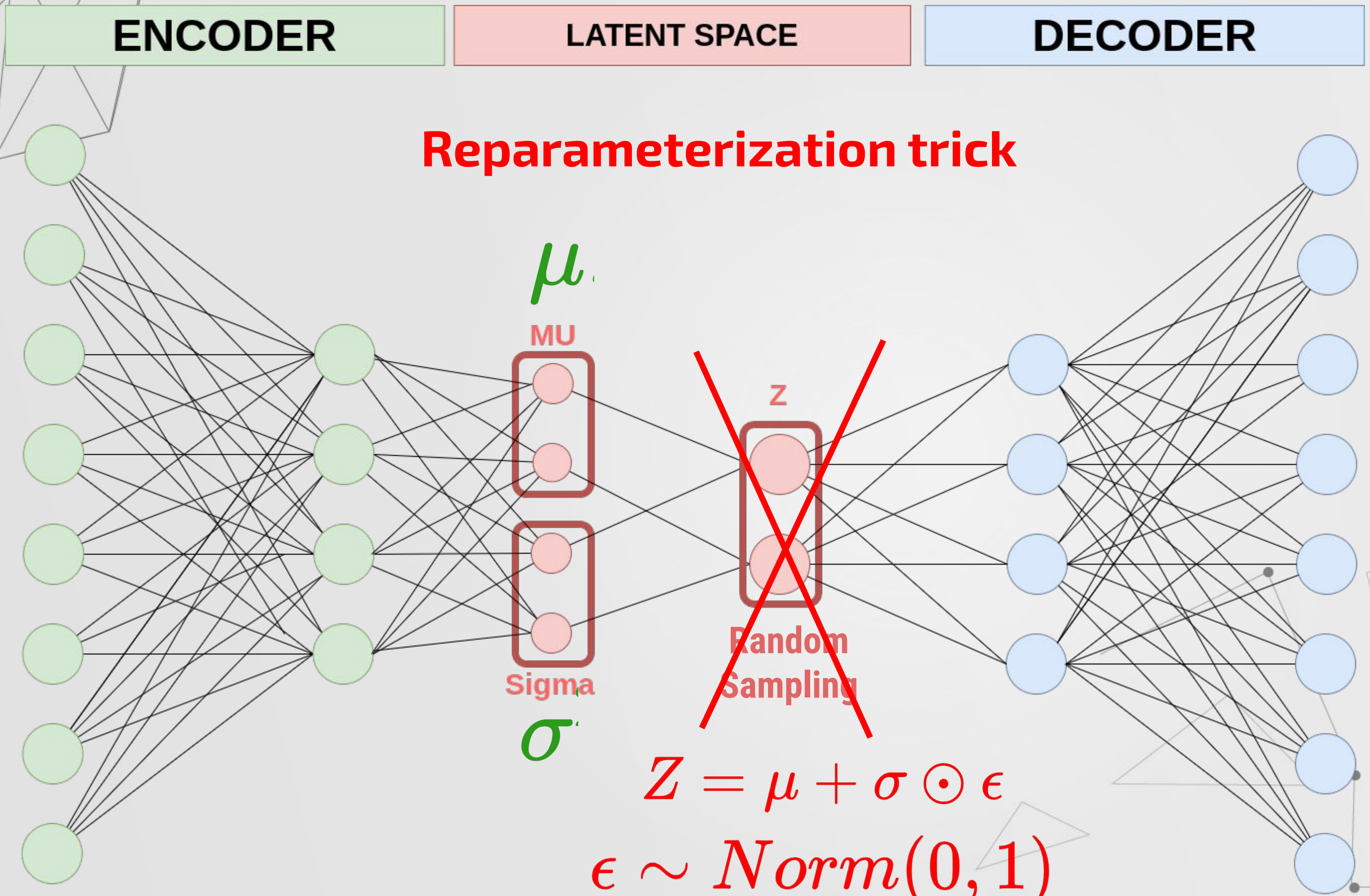


04 Variational Autoencoders



Backpropagation does not work

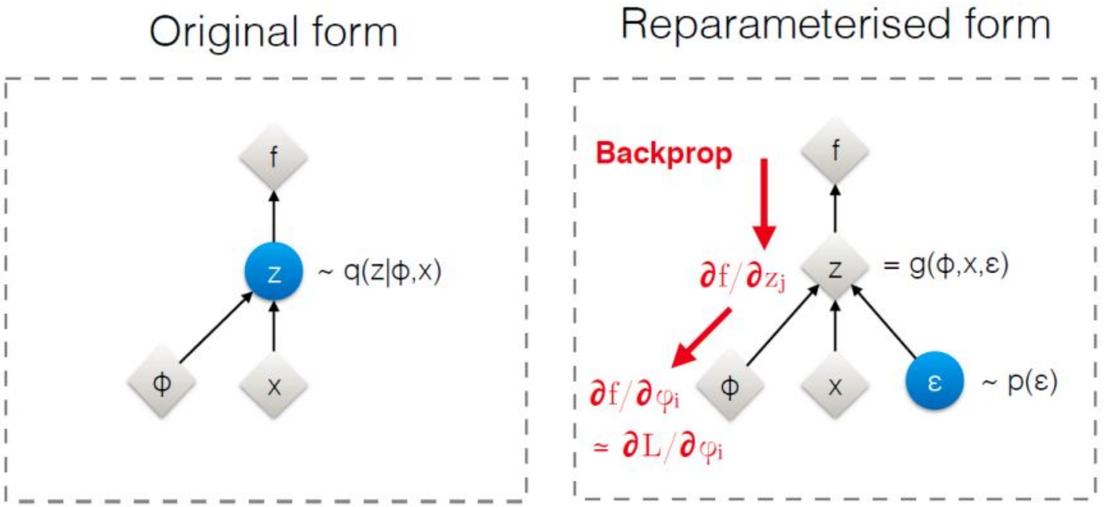
04 Variational Autoencoders



$$Z = \mu + \sigma \odot \epsilon$$

$$\epsilon \sim Norm(0, 1)$$

Reparameterization trick



: Deterministic node

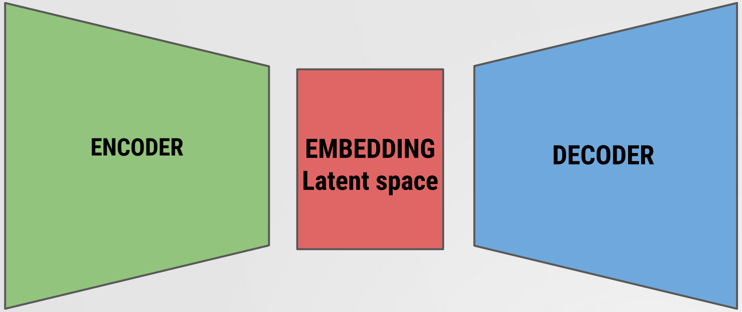
: Random node

[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]

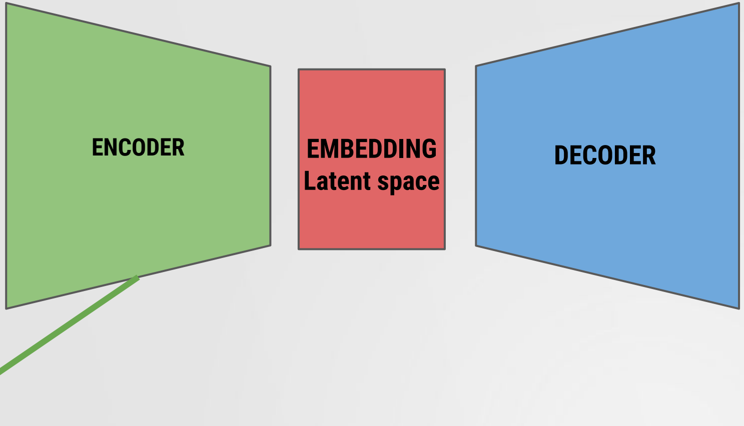
Kingma & Welling, NIPS workshop 2015

05 **Graph Variational Autoencoders (GVAE)** Theory

05 Graph Variational Autoencoders (GVAE) Theory

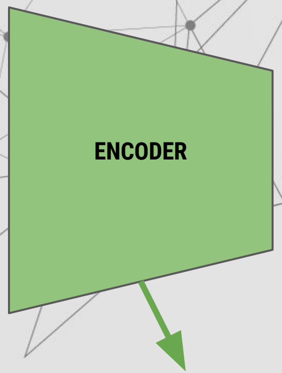


05 Graph Variational Autoencoders (GVAE) Theory



TWO convolutional Graph neural networks:

05 Graph Variational Autoencoders (GVAE) Theory

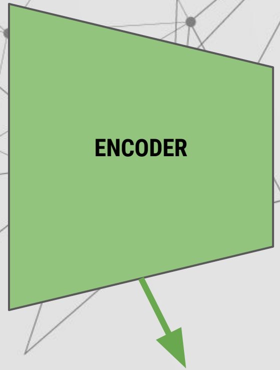


TWO convolutional Graph neural networks:

GCN 1: produces an low dimensional embedding representation

GCN 2: generates μ and $log\,\sigma^2$

05 Graph Variational Autoencoders (GVAE) Theory



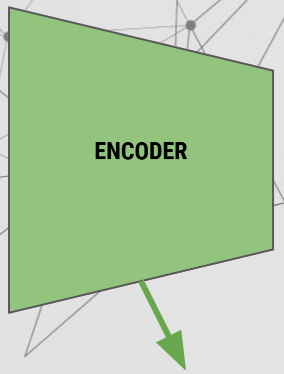
TWO convolutional Graph neural networks:

$$\bar{X} = GCN(A, X)$$

GCN 1: produces an low dimensional embedding representation

GCN 2: generates μ and $log\,\sigma^2$

05 Graph Variational Autoencoders (GVAE) Theory



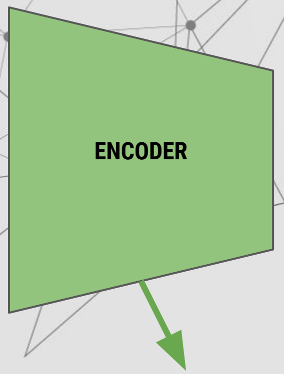
TWO convolutional Graph neural networks:

$$\bar{X} = GCN(A, X) = ReLU(\tilde{A}XW_0)$$

GCN 1: produces an low dimensional embedding representation

GCN 2: generates μ and $log\,\sigma^2$

05 Graph Variational Autoencoders (GVAE) Theory



TWO convolutional Graph neural networks:

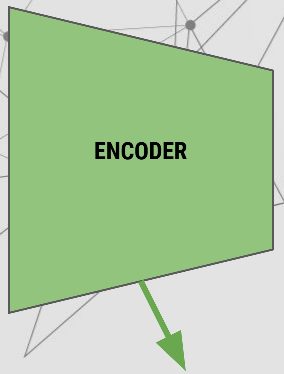
$$\bar{X} = GCN(A, X) = ReLU(\tilde{A}XW_0)$$

GCN 1: produces an low dimensional embedding representation

$$\text{with } \tilde{A} = D^{-1/2}AD^{-1/2}$$

GCN 2: generates μ and $log\, \sigma^2$

05 Graph Variational Autoencoders (GVAE) Theory



TWO convolutional Graph neural networks:

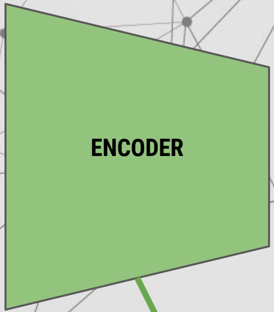
$$\bar{X} = GCN(A, X) = ReLU(\tilde{A}XW_0)$$

GCN 1: produces an low dimensional embedding representation

$$\text{with } \tilde{A} = D^{-1/2} A D^{-1/2}$$

GCN 2: generates $\mu$and $log\, \sigma^2$

05 Graph Variational Autoencoders (GVAE) Theory



TWO convolutional Graph neural networks:

$$\bar{X} = GCN(A, X) = ReLU(\tilde{A}XW_0)$$

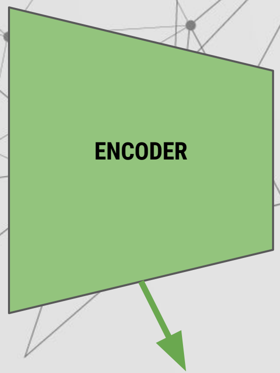
GCN 1: produces an low dimensional embedding representation

$$\text{with } \tilde{A} = D^{-1/2}AD^{-1/2}$$

GCN 2: generates μ and $log\,\sigma^2$

$$\mu = GCN_{\mu}(X, A) = \tilde{A}\bar{X}W_1$$

05 Graph Variational Autoencoders (GVAE) Theory



TWO convolutional Graph neural networks:

$$\bar{X} = GCN(A, X) = ReLU(\tilde{A}XW_0)$$

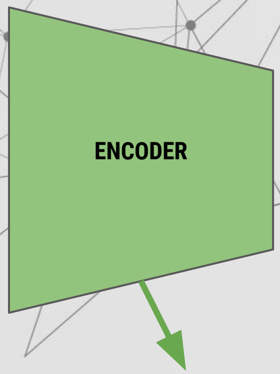
GCN 1: produces an low dimensional embedding representation

$$\text{with } \tilde{A} = D^{-1/2}AD^{-1/2}$$

GCN 2: generates μ and $log\,\sigma^2$

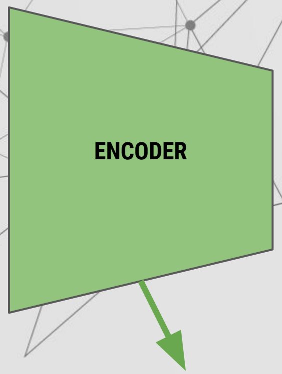
$$log\,\sigma^2 = GCN_\sigma(X, A) = \tilde{A}\bar{X}W_1$$

$$\mu = GCN_\mu(X, A) = \tilde{A}\bar{X}W_1$$



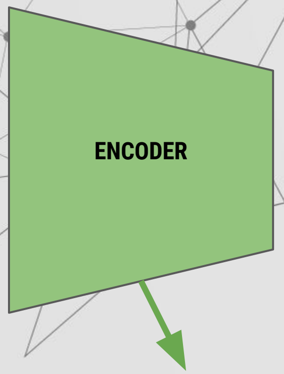
$$GCN(A, X) = \tilde{A} ReLU(\tilde{A} X W_0) W_1$$

$$\text{with } \tilde{A} = D^{-1/2} A D^{-1/2}$$



$$GCN(A, X) = \tilde{A}\overbrace{ReLU(\tilde{A}XW_0)}^{\text{1° GCN}}W_1$$

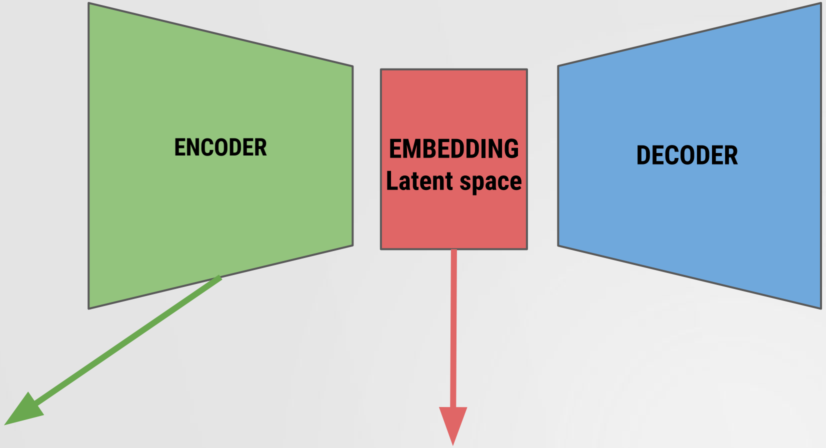
$$\text{with } \tilde{A} = D^{-1/2}AD^{-1/2}$$



$$GCN(A,X) = \overbrace{\tilde{A}\,\overbrace{ReLU(\tilde{A}XW_0}^{1°\ GCN})W_1}^{2°\ GCN}$$

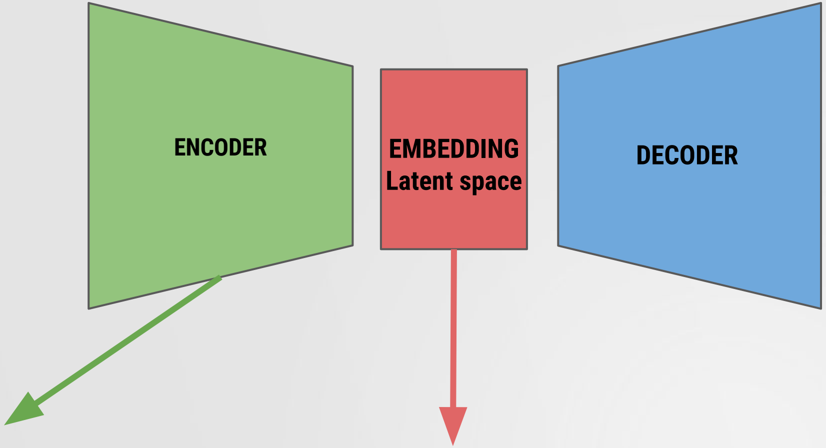
$$\text{with } \tilde{A} = D^{-1/2} A D^{-1/2}$$

05 Graph Variational Autoencoders (GVAE) Theory



$$GCN(A, X) = \tilde{A}ReLU(\tilde{A}XW_0)W_1$$

05 Graph Variational Autoencoders (GVAE) Theory



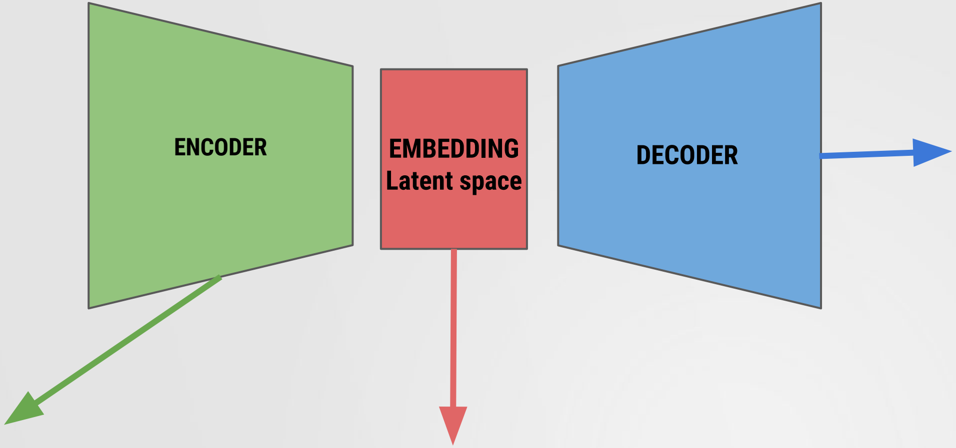
$$GCN(A, X) = \tilde{A} ReLU(\tilde{A} X W_0) W_1$$

Reparameterization trick

$$Z = \mu + \sigma \odot \epsilon$$

$$\epsilon \sim Norm(0, 1)$$

05 Graph Variational Autoencoders (GVAE) Theory



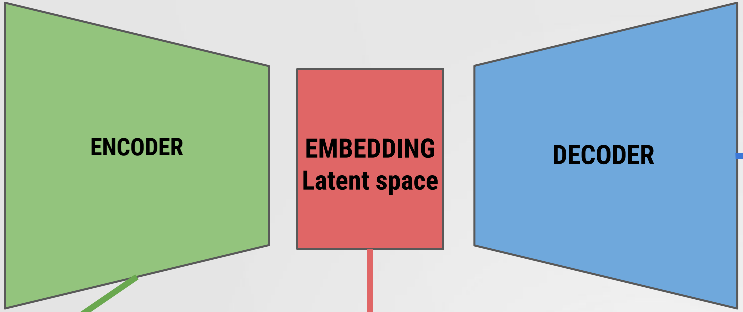
$$GCN(A, X) = \tilde{A}ReLU(\tilde{A}XW_0)W_1$$

Reparameterization trick

$$Z = \mu + \sigma \odot \epsilon$$

$$\epsilon \sim Norm(0, 1)$$

05 Graph Variational Autoencoders (GVAE) Theory



Inner product
Between latent variable Z

$$\hat{A} = \text{logistic sigmoid}\left(zz^T\right)$$

$$GCN(A, X) = \tilde{A}ReLU(\tilde{A}XW_0)W_1$$

Reparameterization trick

$$Z = \mu + \sigma \odot \epsilon$$
$$\epsilon \sim Norm(0, 1)$$

Jupyter Notebook