

# **Deep learning for medical imaging school**

## Hands-on session

# Autoencoders

By  
Pierre-Marc Jodoin, Nathan Painchaud

With the support of  
Thomas Grenier

Setup your  
**SaturnCloud**  
environment



Why Saturn Cloud ▾ Partners ▾ Resources ▾ Docs Plans & Pricing Enterprise Login

[Get a Technical Demo](#)

First, login on SaturnCloud.io

# Move your data science team into the cloud without having to switch tools

Saturn Cloud provides customizable, ready-to-use cloud environments for collaborative data teams. Run analyses, train models, deploy APIs, and more

[Get a Technical Demo](#)

[View pricing](#)

Powering 100,000+ data science users and teams



snowflake



kaggle

Cellarity



CUSTOM INK

MERCURY INSURANCE



**Create a Jupyter server for the autoencoder hands on session if you haven't done so already.**

2 FREE HOURS LEFT

Resets June 30

UPGRADE TO PRO

community

Saturn Cloud

USER pierremarcjodoin

Create an Organization

Resources

Secrets

Git Repositories

Images

Shared Folders

Create a Resource

Resources are self-contained compute environments that you can customize with your preferred hardware and software. Multiple resources can run at the same time. ([learn more](#))

New Python Server

Use JupyterLab, or connect PyCharm or VS Code

New R Server

Write and run code in the R IDE

New Deployment

Host an app or API

New Job

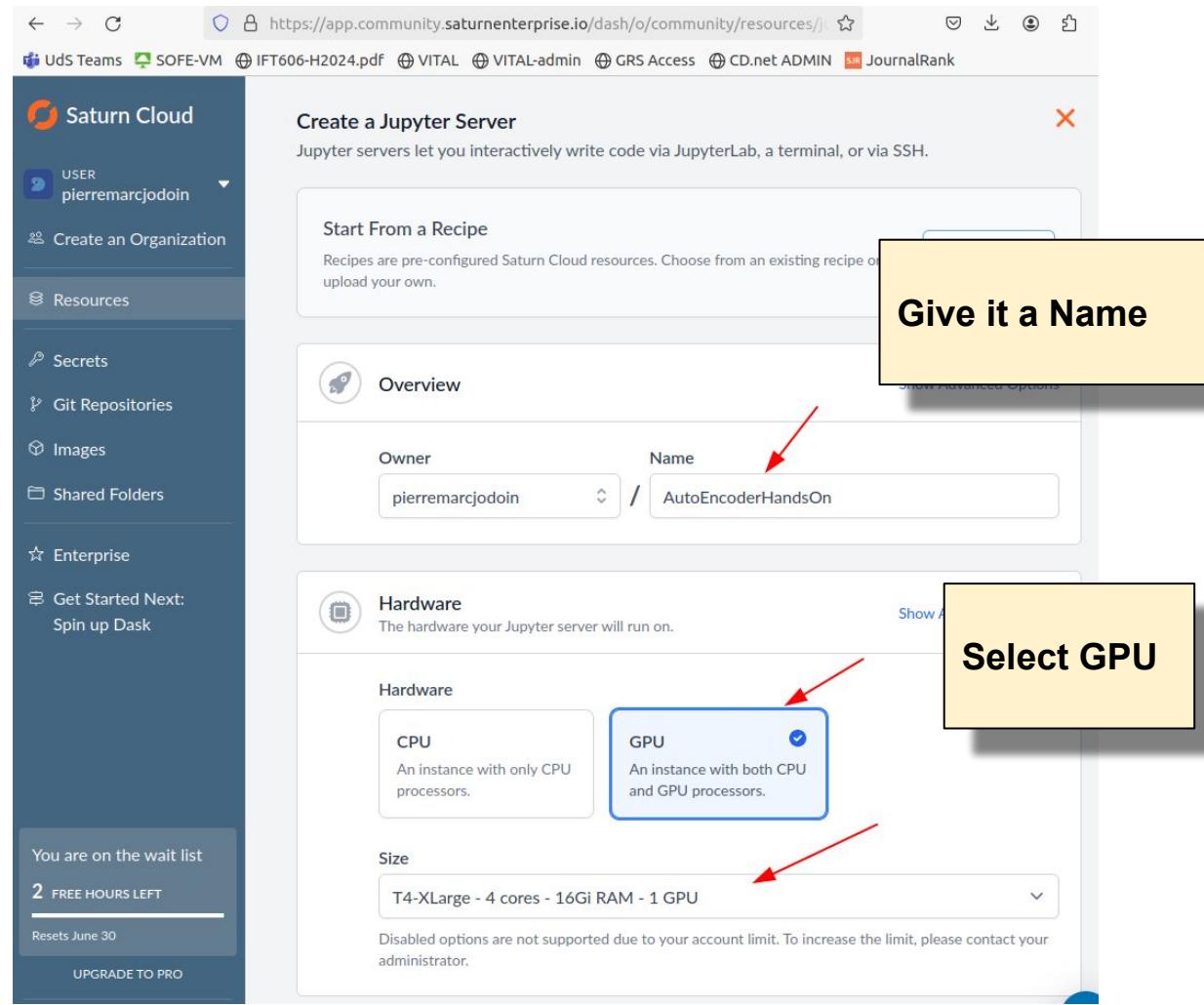
Run a task on a schedule or on command

aws Launch in your AWS account

The most secure option for your code and data

Or use one of our pre-configured resource templates

Large Language Models



Saturn Cloud

USER pierremarcjodoin

Create an Organization

Resources

Secrets

Git Repositories

Images

Shared Folders

Enterprise

Get Started Next:  
Spin up Dask

## Environment

The software your Jupyter server will use. This includes libraries, packages, environment variables, and other attributes.

Show Advanced Options

### Write torchvision for pip install

#### Extra Packages

Extra packages are installed every time the resource starts up - right before the start script. Use spaces to separate packages.

If you find yourself adding the same packages to lots of resources, you may want to permanently add packages to a custom image instead. (?)

Conda

Pip

Apt

torchvision

This is a requirements.txt

If enabled, this should be valid requirements file describing all packages to be installed by `pip` . such as the content produced by `pip freeze` . See the `pip` documentation for details.

The packages together will run the following script:

```
apt-get install htop zip unzip  
pip install torchvision
```

USER  
pierremarcjodoin ▾

Create an Organization

Resources

Secrets

Git Repositories

Images

Shared Folders

Enterprise

Get Started Next:  
Spin up Dask

## Environment



The software your Jupyter server will use. This includes libraries, packages, environment variables, and other attributes.

[Show Advanced Options](#)

### Image

saturncloud/saturn-python-pytorch

### Version

2023.09.01

### Extra Packages

Extra packages are installed every time the resource starts up - right before the start script. Use spaces to separate packages.

If you find yourself adding the same packages to lots of resources, you may want to permanently add packages to a custom image instead. [\(?\)](#)

 Conda Pip Apt

htop zip unzip

The packages together will run the following:

```
apt-get install htop zip unzip  
pip install torchvision
```

**Write these for apt install**

The screenshot shows the Saturn Cloud web interface. On the left, a sidebar lists user information (pierremarcjodoin), organization creation, resources, secrets, git repositories, images, shared folders, enterprise options, and a 'Get Started Next' section. A yellow callout box at the bottom left points to the 'Create' button with the text 'Click Create'. The main area is titled 'Create a New Server'. It includes tabs for Conda, Pip, and Apt, with Apt selected. A red circle highlights the command 'htop zip unzip' in the 'Apt' tab. Below it, a note says 'The packages together will run the following script:' followed by the commands 'apt-get install htop zip unzip' and 'pip install torchvision'. A yellow callout box over this section contains the text 'Write these for apt install'. The 'Git Repositories' section shows a dropdown for adding a repository and a 'New Git Repository' button. The 'Additional features' section includes an 'Allow SSH Connections' checkbox (unchecked) and a 'Shutdown After' dropdown set to '1 hour'. A red arrow points to the 'Create' button at the bottom, which is highlighted with a red circle.

Saturn Cloud

USER  
pierremarcjodoin

Create an Organization

Resources

Secrets

Git Repositories

Images

Shared Folders

Enterprise

Get Started Next:  
Spin up Dask

Click Create

on the wait list

HOURS LEFT

Resets June 30

UPGRADE TO PRO

community  
v2024.05.01-2

Conda

Pip

Apt

htop zip unzip

The packages together will run the following script:

```
apt-get install htop zip unzip  
pip install torchvision
```

**Write these for apt install**

Git Repositories

Add a git repository

New Git Repository

Remote URL

No Git Repositories Added.

Actions

Additional features

Show Advanced Options

Allow SSH Connections

Use SSH to directly connect to the server, including through VSCode, PyCharm, and other tools (?)

Shutdown After

1 hour

Disabled options are not supported by your current plan. Contact support if you have any questions.

Create

Cancel

USER  
pierremarcjodoin

Create an Organization

Resources

Secrets

Git Repositories

Images

Shared Folders

Enterprise

Get Started Next:  
Spin up Dask

community / Resources / pierremarcjodoin / AutoEncoderHandsOn\_test2

## JUPYTER SERVER



pierremarcjodoin / AutoEncoderHandsOn\_test2

3bbf45557caf492e9172a91dedc6c966

Edit

Overview

Environment

Secrets and Roles

Git Repos

Shared Folders

## Jupyter Server

stopped

T4-XLarge - 4 cores - 16Gi RAM - 1 GPU - 2Gi Disk

Auto Shutoff: 1 hour

Spot Instance: No

URLs: (server not running)

SSH URL: (not enabled) (?)

Start

Click Start

Jupyter Lab

 USER  
pierremarcjodoin Create an Organization Resources Secrets Git Repositories Images Shared Folders Enterprise Get Started Next:  
Spin up Dask

## JUPYTER SERVER



pierremarcjodoin / testAutoEncoder

4dc560a08e884d1da98c4206cf56226b

 Edit Overview Environment Secrets and Roles Git Repos Shared Folders

## Jupyter Server

pending

debug

T4-XLarge - 4 co

Auto Shutoff: 1 hour

Spot Instance: No

URLs:

<https://w-pierr-testautoencoder-4dc560a08e884d1da98c4206cf56226b.community.saturnenterprise.io>

SSH URL: (server not running) (?)



Provisioned Hardware

Pulled Image

Set Up Environment

Executing Start Script

Ready

 Jupyter Lab

Saturn Cloud

pierremarcjodoin

Resources

Credentials

Git Repositories

Images

Enterprise

Get Started Next:  
Spin up Dask

HOURS REMAINING  
[Upgrade for More](#)

Jupyter Dask	30 hrs 3 hrs
-----------------	-----------------

Resources / pierremarcjodoin / AutoEncoderHandsOn2

JUPYTER SERVER

pierremarcjodoin / AutoEncoderHan...

1fc514f792564bc383f7cbda6237181f

Recipe Logs Edit Delete

Overview Environment Git Repos Share

Jupyter Server running

T4-XLarge - 4 cores - 16 GB RAM - 1 GPU - 10Gi Disk

Metrics

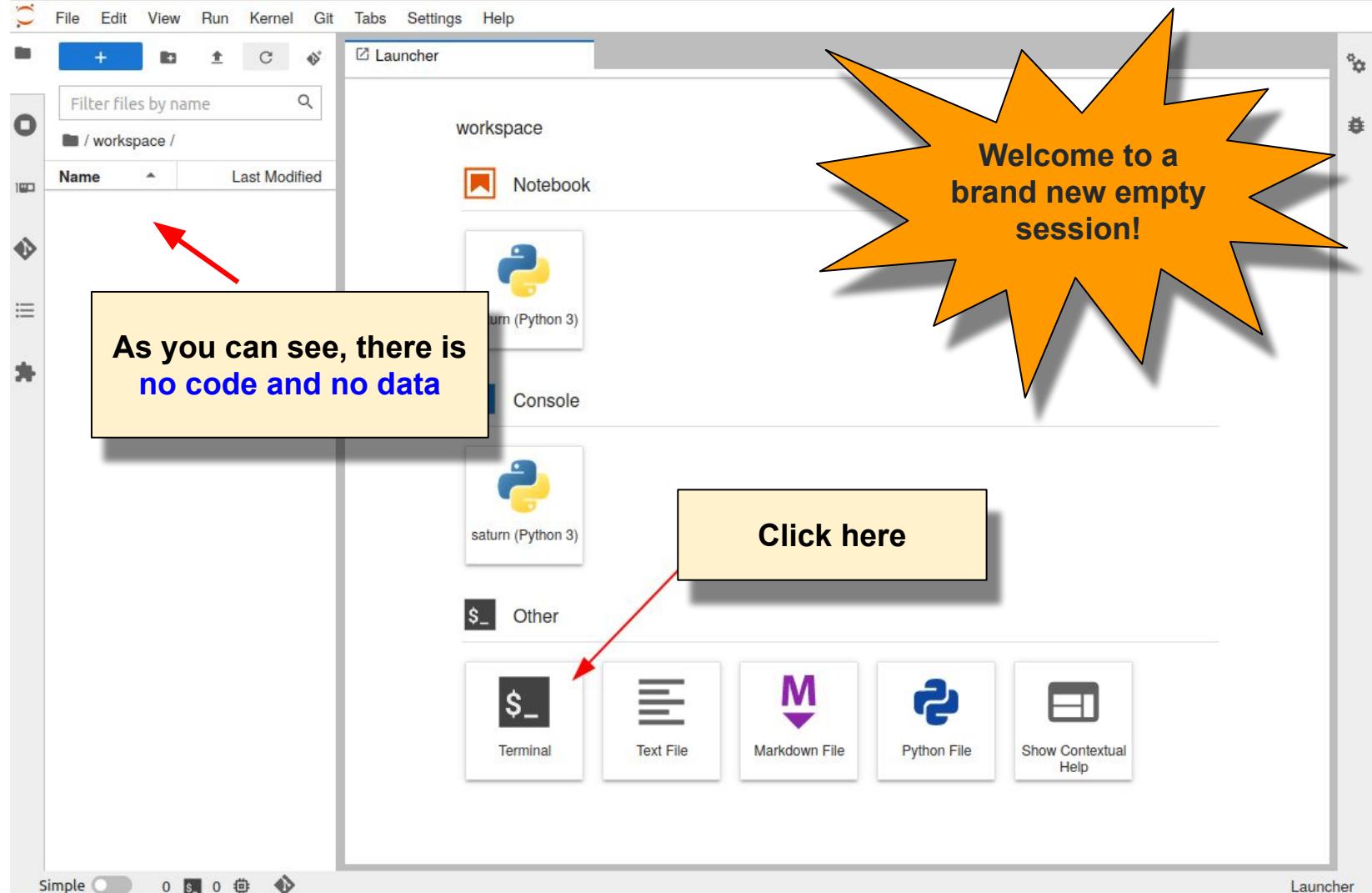
Auto Shutoff: 1 hour  
Spot Instance: No  
SSH URL: (not enabled) (?)

Click here

Stop

Jupyter Lab

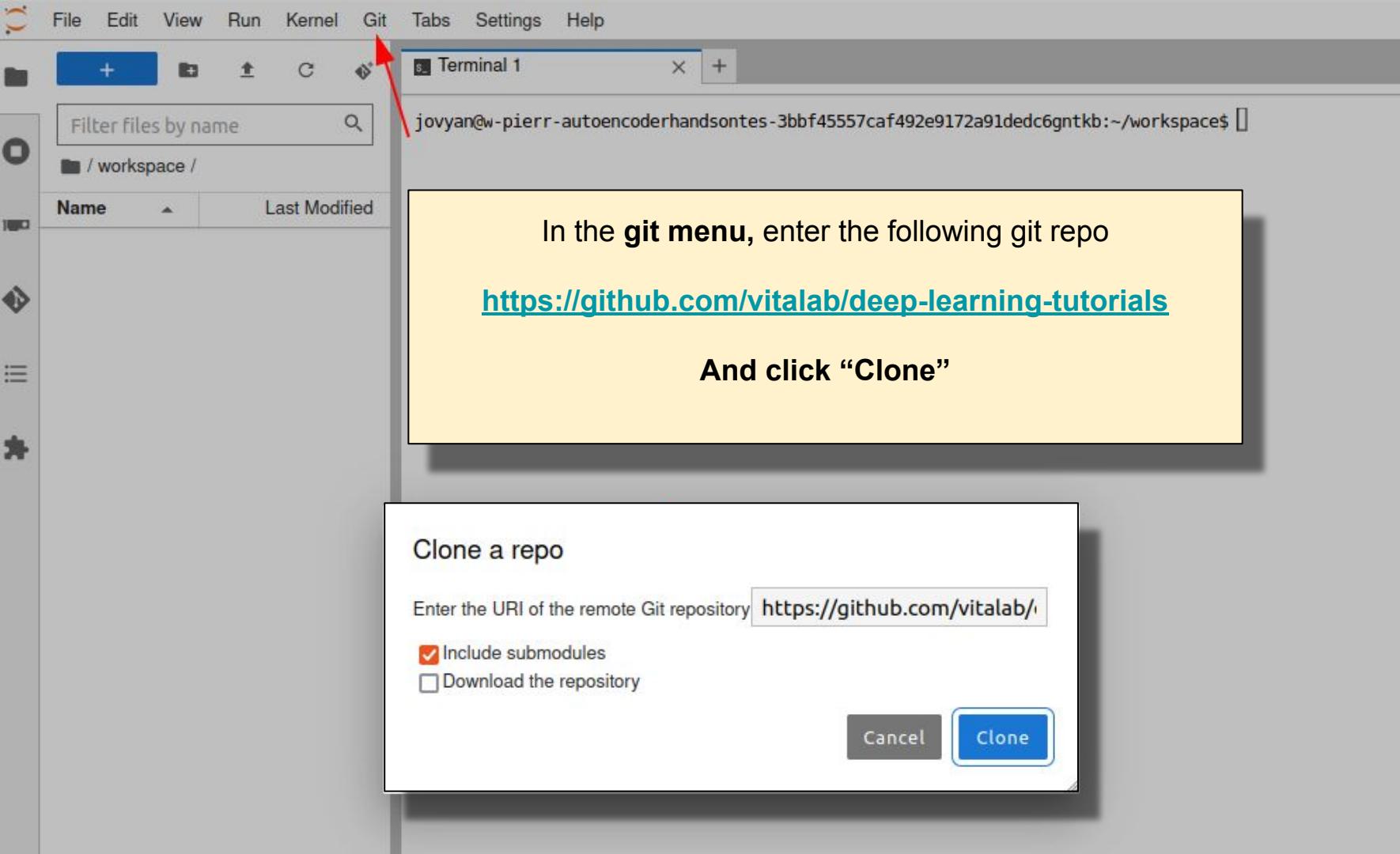
A yellow callout box with the text "Click here" is positioned over the "Stop" button. A red arrow points from a red oval at the bottom of the screen to the "Jupyter Lab" button.



# Lets download the code

```
        this._config.interval = this._config.interval || 1000;
    }

    var transitionDuration = 0;
    $(activeElement).one(Util.Transition.end, function() {
        $(nextElement).removeClass(ClassNames.ACTIVE);
        $(activeElement).removeClass(ClassNames.ACTIVE);
        _this4._isSliding = false;
        setTimeout(function () {
            return $_.this4._element.trigger('slideend');
        }, 0);
    }).emulateTransitionEnd(transitionDuration);
} else {
    $(activeElement).removeClass(ClassNames.ACTIVE);
    $(nextElement).addClass(ClassNames.ACTIVE);
}
```





Filter files by name

/ workspace /

Name Last M

deep-learning-tutorials second

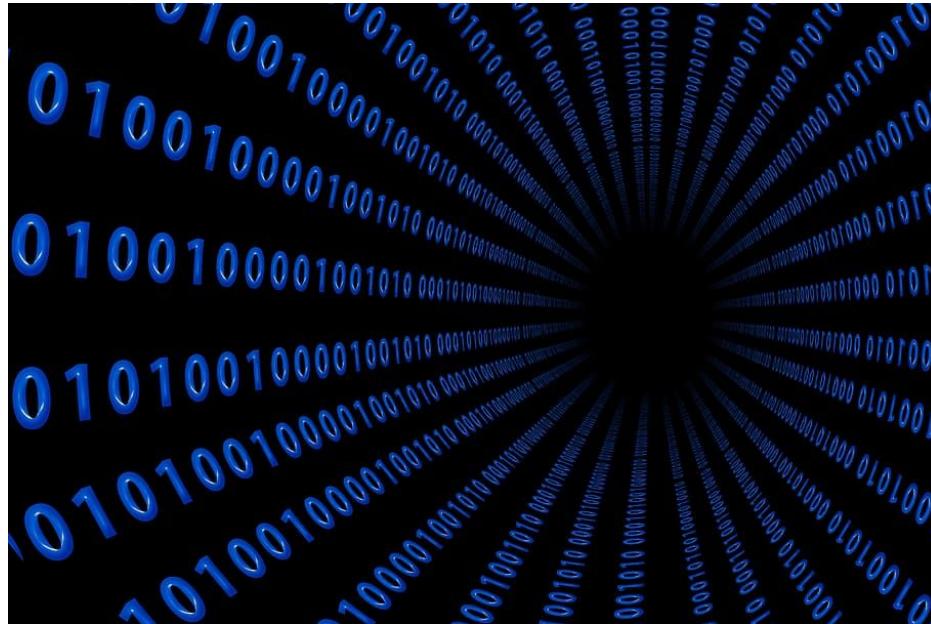
Terminal 1

jovyan@w-pierr-autoencoderhandsontes-3bbf45557caf492e9172a91dedc6gntkb:~/workspace\$

The code should be on the left.

Click here

# Lets download the data



The screenshot shows a Jupyter Notebook interface. The top navigation bar includes File, Edit, View, Run, Kernel, Git, Tabs, Settings, and Help. The Git tab is currently selected. On the left, there's a sidebar with icons for file operations like new, open, and save, followed by a 'Filter files by name' input field and a tree view of the workspace directory. The workspace contains files like src, tutorials, dev.txt, environment.yml, LICENSE, pyproject.toml, README.md, requirements.txt, setup.cfg, and setup.py. A tooltip or message box is overlaid on the right side of the screen, containing the text "Download the following file (data.tar.gz) on your computer" and a blue hyperlink: <https://drive.google.com/file/d/1H5pTOYjcSFR6B5GhA0sEPW0wgPVfBq8S/view?usp=sharing>.

File Edit View Run Kernel Git Tabs Settings Help

Filter files by name  
/ workspace / dev

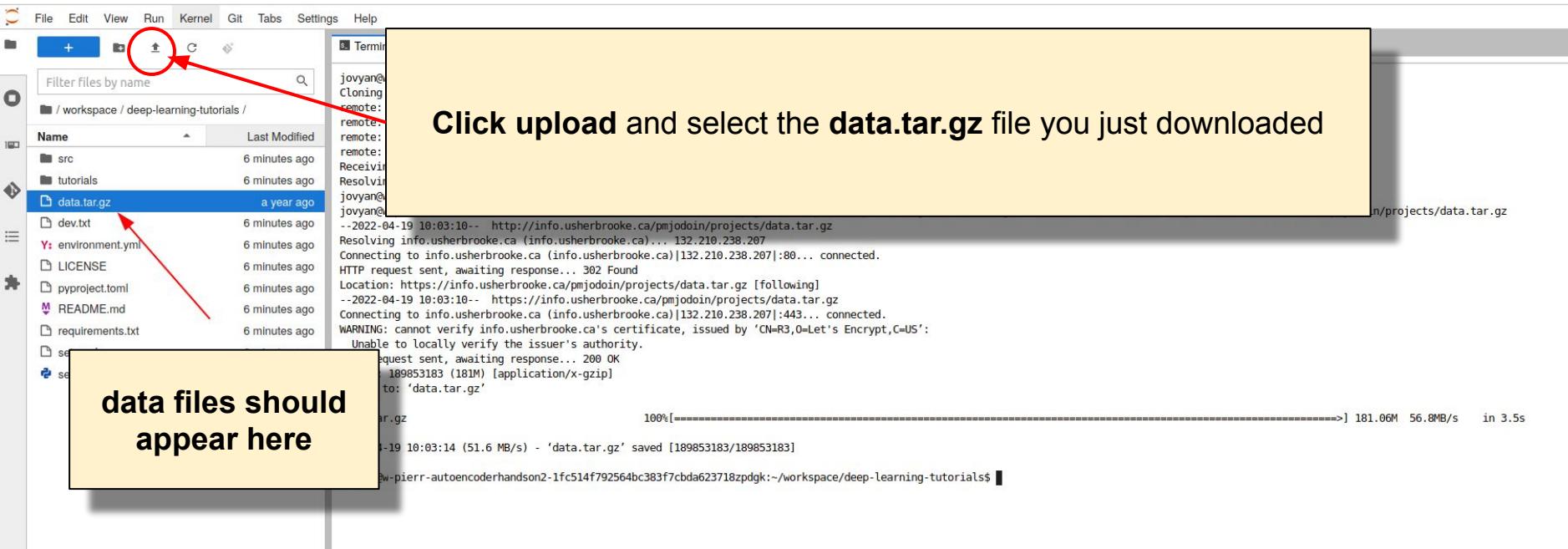
Name

- src
- tutorials
- dev.txt
- environment.yml
- LICENSE
- pyproject.toml
- README.md
- requirements.txt
- setup.cfg
- setup.py

5 minutes ago  
5 minutes ago  
5 minutes ago  
4 minutes ago  
5 minutes ago  
5 minutes ago

Download the following file (**data.tar.gz**) on your computer

<https://drive.google.com/file/d/1H5pTOYjcSFR6B5GhA0sEPW0wgPVfBq8S/view?usp=sharing>



File Edit View Run Kernel Git Tabs Settings Help

Terminal 1 requirements.txt

```
jovyan@w-pierr-autoencoderhandson2-1fc514f792564bc383f7cbda623718zpdgk:~/workspace$ git clone https://github.com/vitalab/deep-learning-tutorials.git
Cloning into 'deep-learning-tutorials'...
remote: Enumerating objects: 280, done.
remote: Counting objects: 100% (198/198), done.
remote: Compressing objects: 100% (149/149), done.
remote: Total 280 (delta 134), reused 84 (delta 49), pack-reused 82
Receiving objects: 100% (280/280), 66
Resolving deltas: 100% (156/156), don
jovyan@w-pierr-autoencoderhandson2-1fc514f792564bc383f7cbda623718zpdgk:~/workspace$ cd deep-learning-tutorials
jovyan@w-pierr-autoencoderhandson2-1fc514f792564bc383f7cbda623718zpdgk:~/workspace/deep-learning-tutorials$ ls
data  src  tutorials  data.tar.gz  dev.txt  environment.yml  LICENSE  pyproject.toml  README.md  requirements.txt  setup.cfg  setup.py
jovyan@w-pierr-autoencoderhandson2-1fc514f792564bc383f7cbda623718zpdgk:~/workspace/deep-learning-tutorials$ cd deep-learning-tutorials
jovyan@w-pierr-autoencoderhandson2-1fc514f792564bc383f7cbda623718zpdgk:~/workspace/deep-learning-tutorials$ tar -xvzf data.tar.gz
jovyan@w-pierr-autoencoderhandson2-1fc514f792564bc383f7cbda623718zpdgk:~/workspace/deep-learning-tutorials$ ls
data/MNIST/raw/
data/MNIST/raw/train-labels-idx1-ubyte
data/MNIST/raw/train-images-idx3-ubyte.gz
data/MNIST/raw/train-labels-idx1-ubyte.gz
data/MNIST/raw/t10k-labels-idx1-ubyte
data/MNIST/raw/t10k-images-idx3-ubyte.gz
data/MNIST/raw/t10k-labels-idx1-ubyte
data/MNIST/raw/train-images-idx3-ubyte
data/MNIST/raw/t10k-labels-idx1-ubyte.gz
data/MNIST/processed/
data/MNIST/processed/test.pt
data/MNIST/processed/training.pt
jovyan@w-pierr-autoencoderhandson2-1fc514f792564bc383f7cbda623718zpdgk:~/workspace/deep-learning-tutorials$
```

Then in the terminal, type

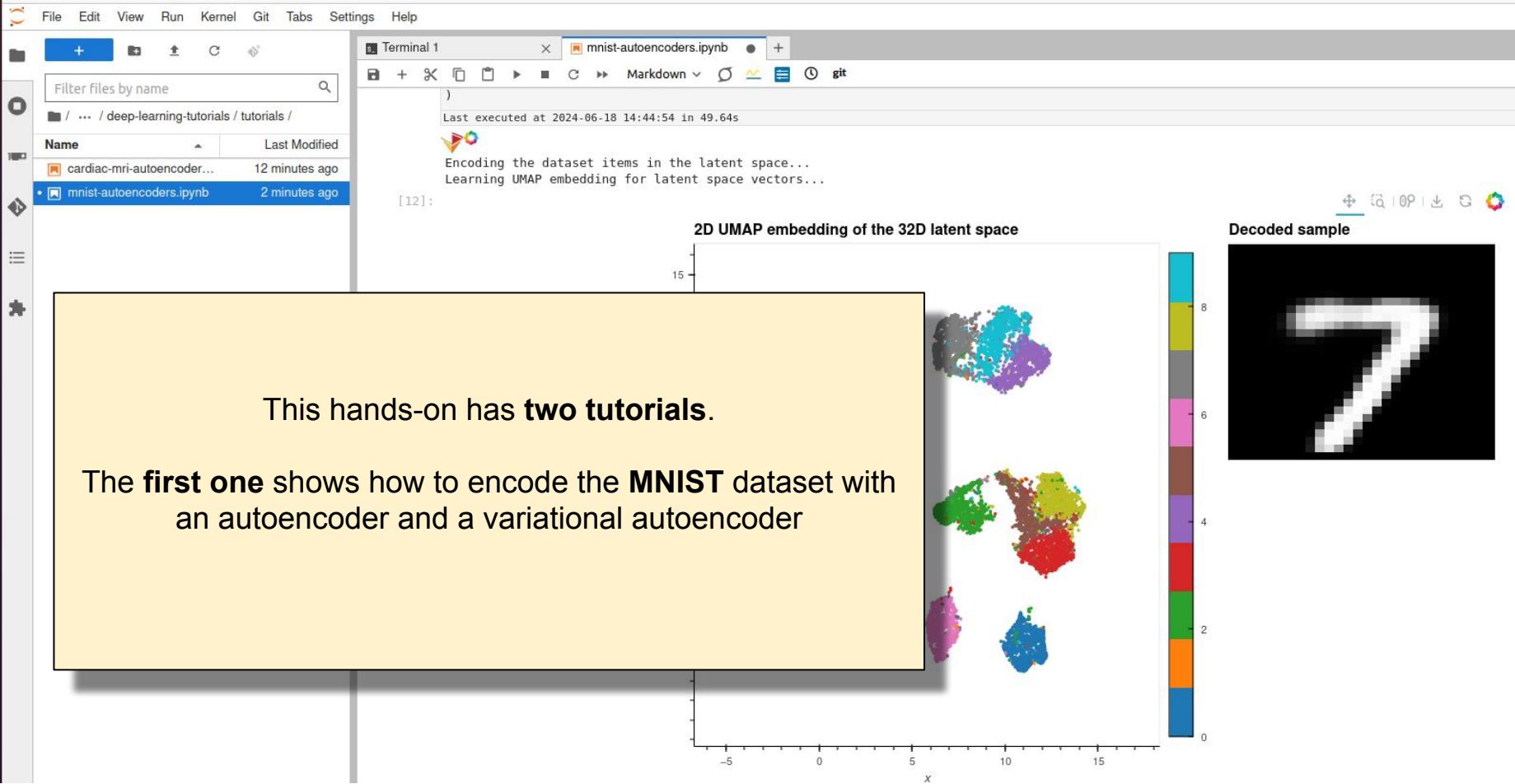
**cd deep-learning-tutorials**

followed by

**tar -xvzf data.tar.gz**



# Autoencoders' Hands on



File Edit View Run Kernel Tabs Settings Help

mnist-autoencoders.ipynb cardiac-mri-autoencoders.ipynb latent\_space.py

Notebook

Filter files by name

/ tutorials /

Name Modified

cardiac-mri-autoencoders.ipynb 3 min. ago

mnist-autoencoders.ipynb 2 min. ago

[14]: from src.visualization.latent\_space import explore\_latent\_space

explore\_latent\_space(  
 acdc\_val,  
 num\_classes=4).float()[:, :latent\_space\_size],

This hands-on has **two** tutorials.

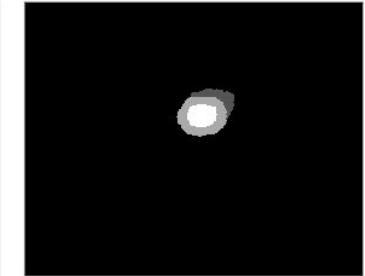
The **second one** shows how to encode a **cardiac shapes** (ACDC dataset) with an autoencoder and a variational autoencoder.

Please see the last pages of this document for **more details on cardiac shapes**

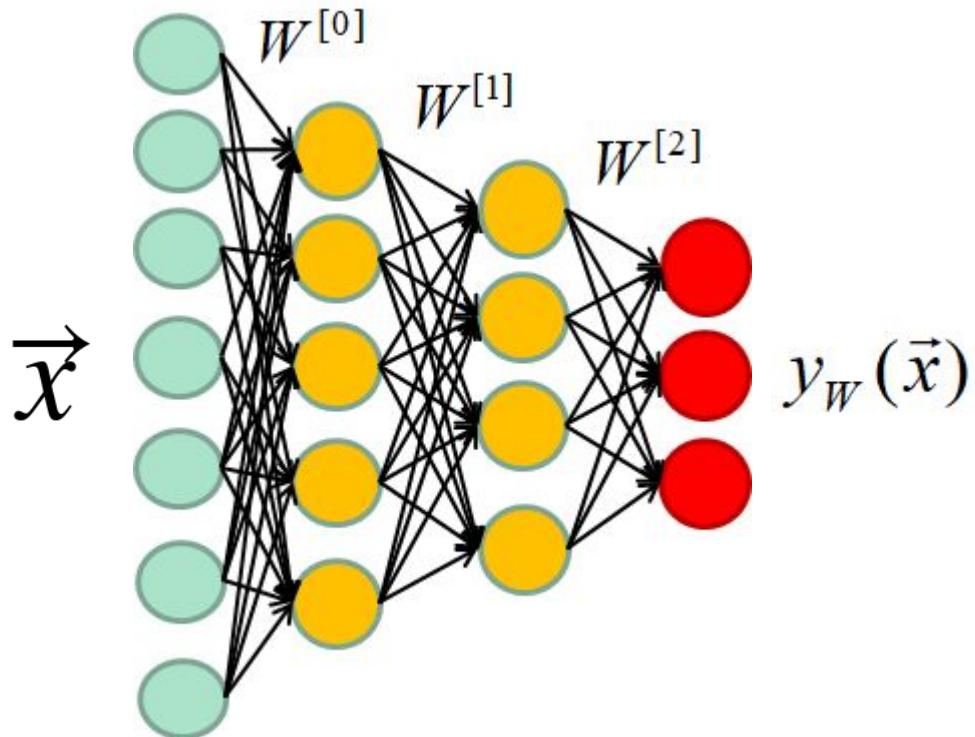
the 3D latent space



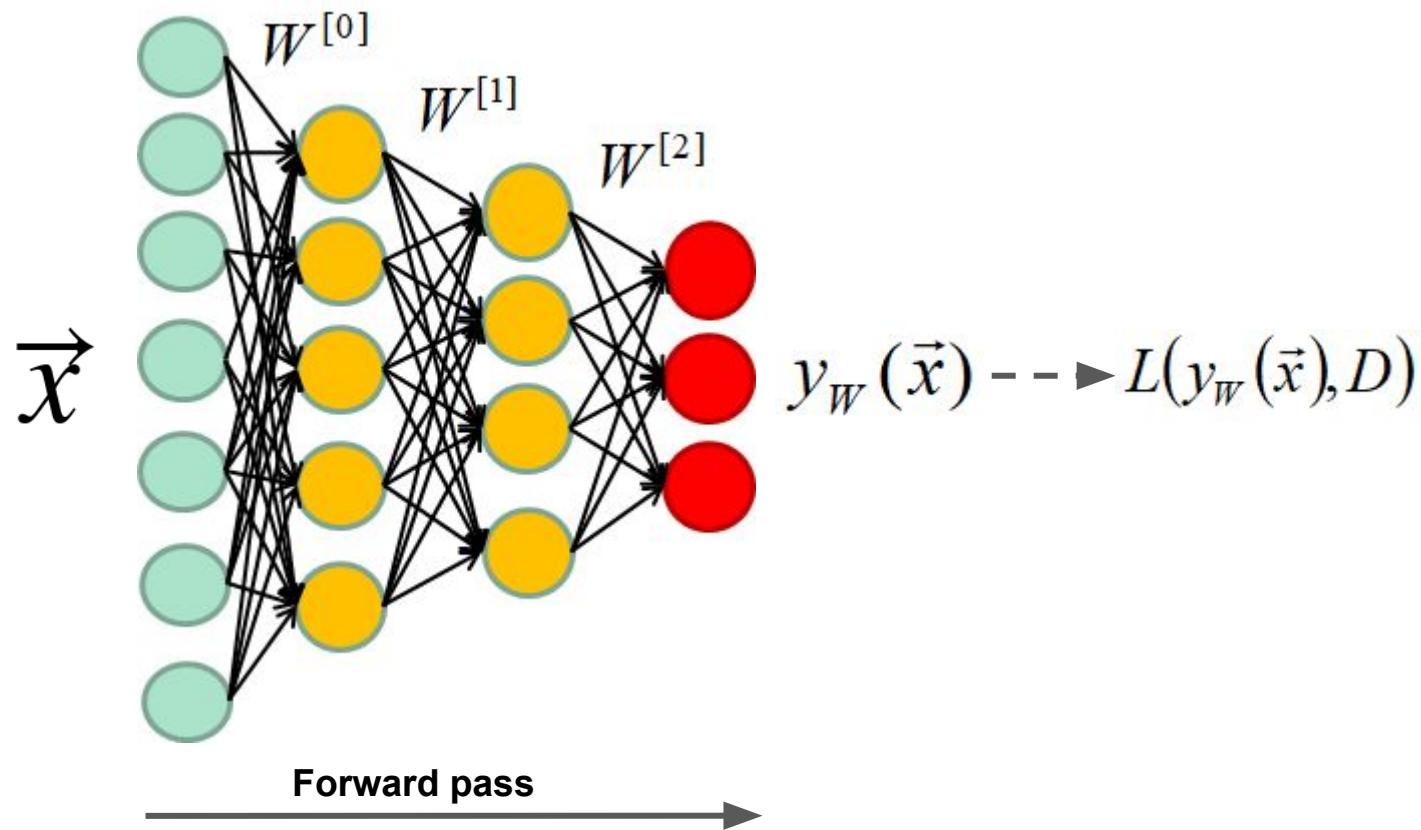
Decoded sample



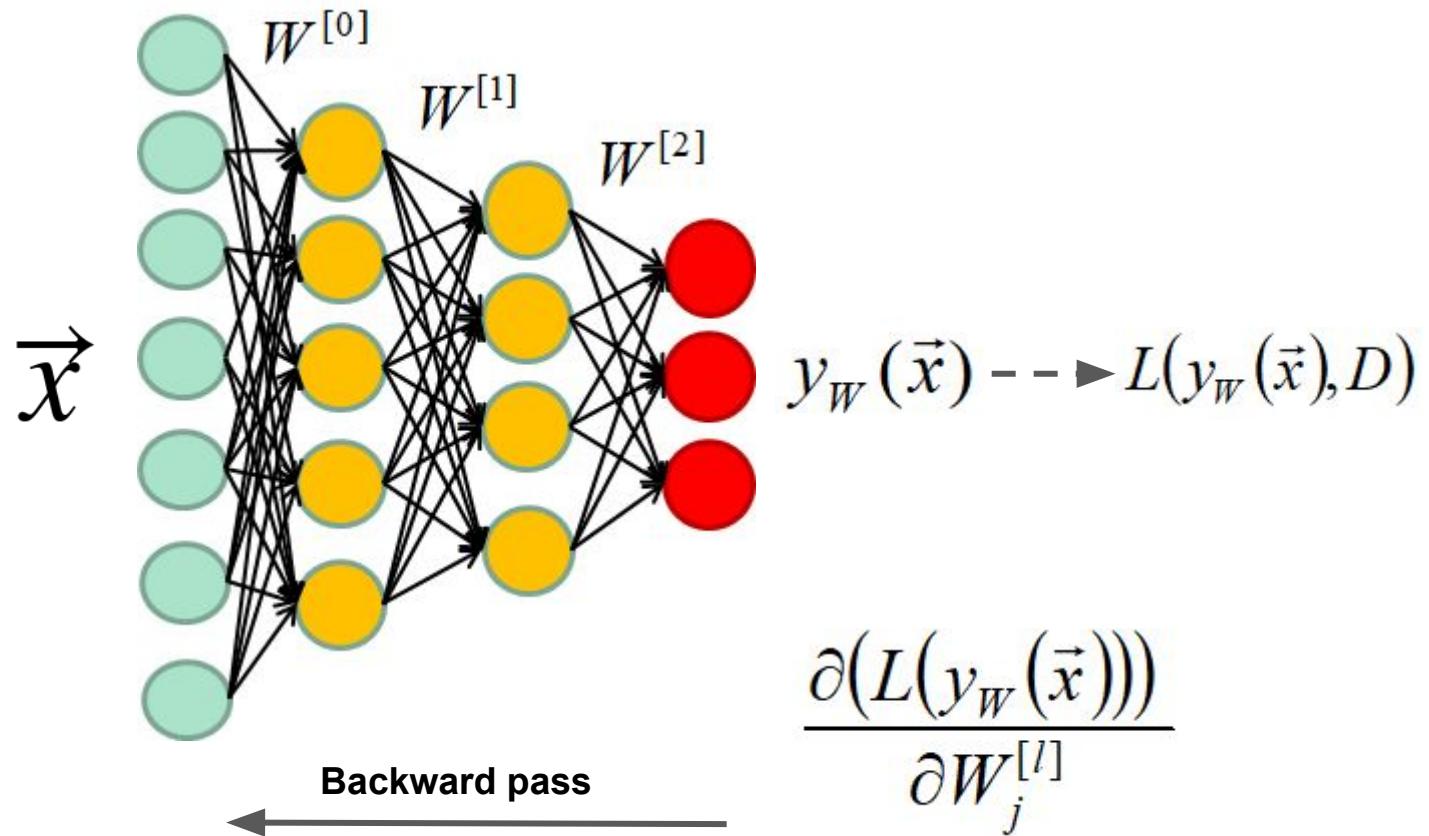
# Autoencoders (recap)



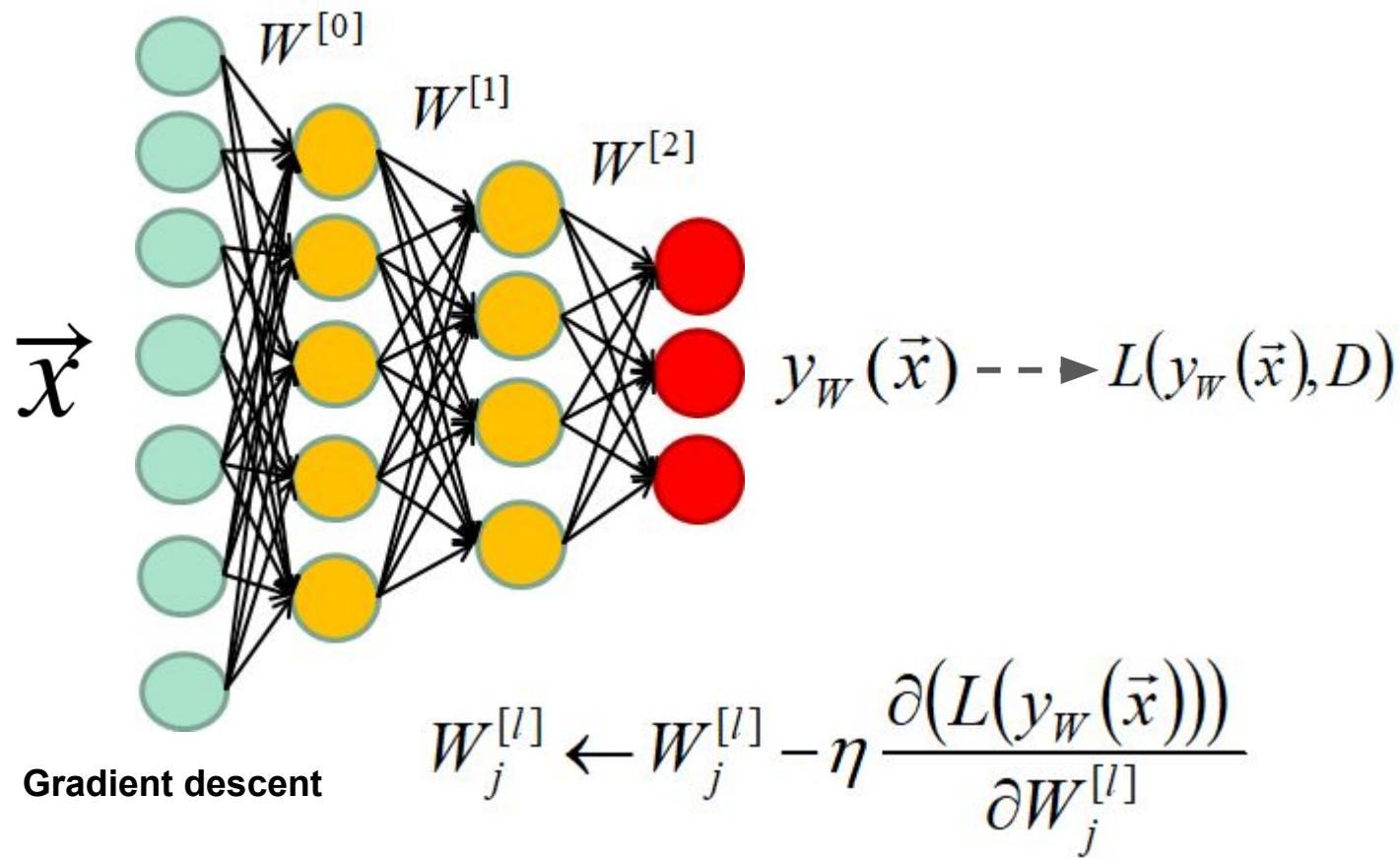
Annotated dataset:  $D = \{(\vec{x}_1, t_1), (\vec{x}_2, t_2), \dots, (\vec{x}_N, t_N)\}$

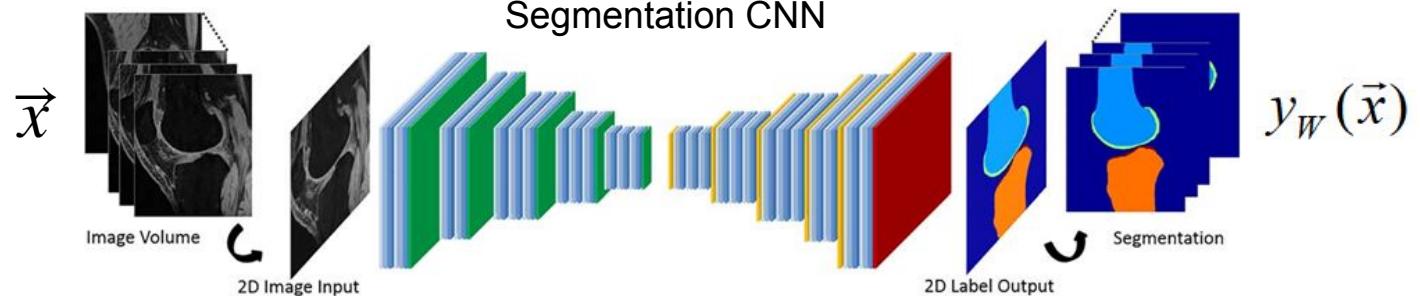
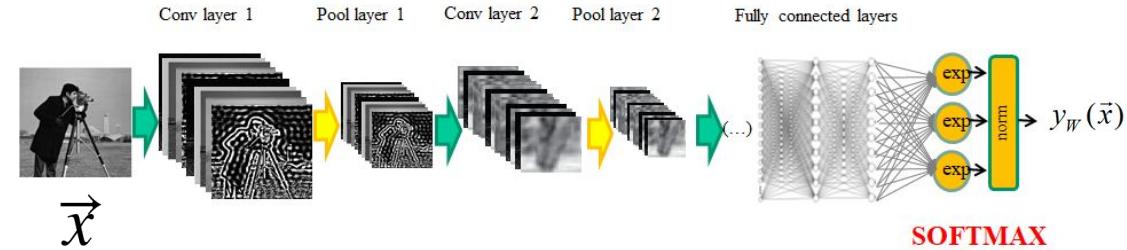
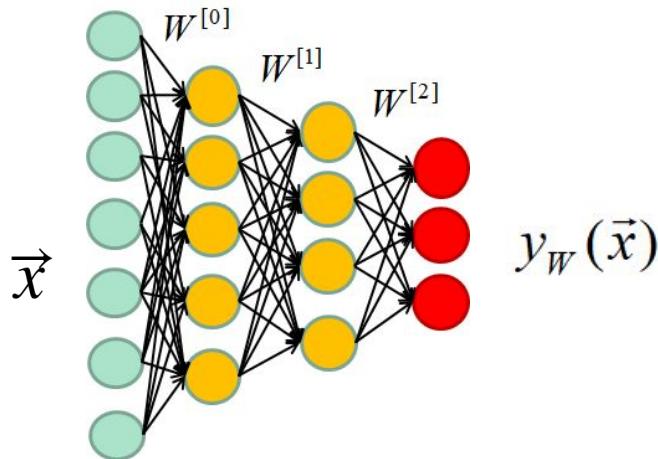


Annotated dataset:  $D = \{(\vec{x}_1, t_1), (\vec{x}_2, t_2), \dots, (\vec{x}_N, t_N)\}$



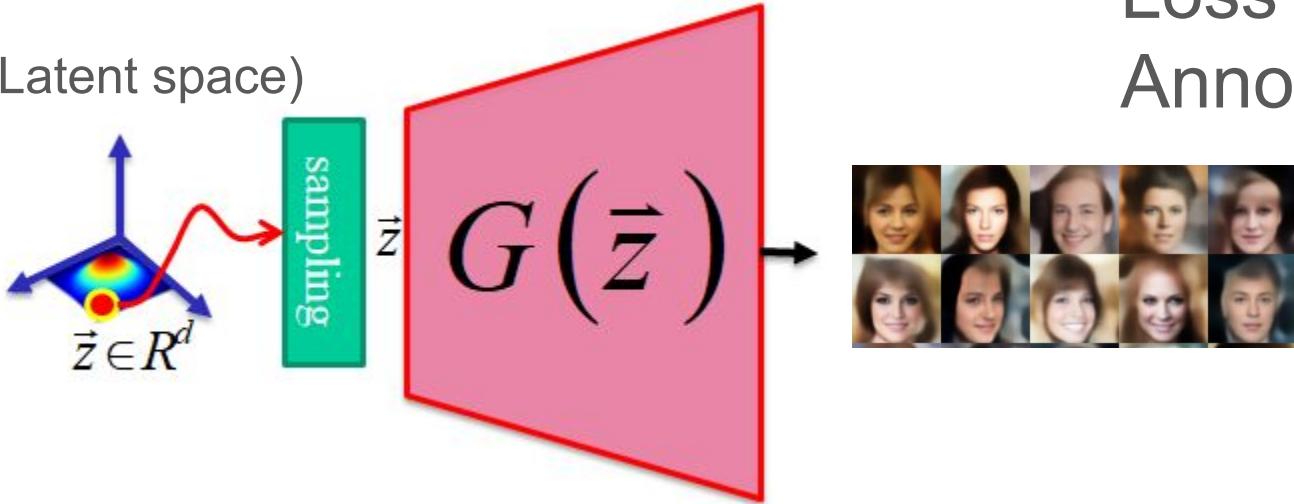
Annotated dataset:  $D = \{(\vec{x}_1, t_1), (\vec{x}_2, t_2), \dots, (\vec{x}_N, t_N)\}$





# GAN

(Latent space)

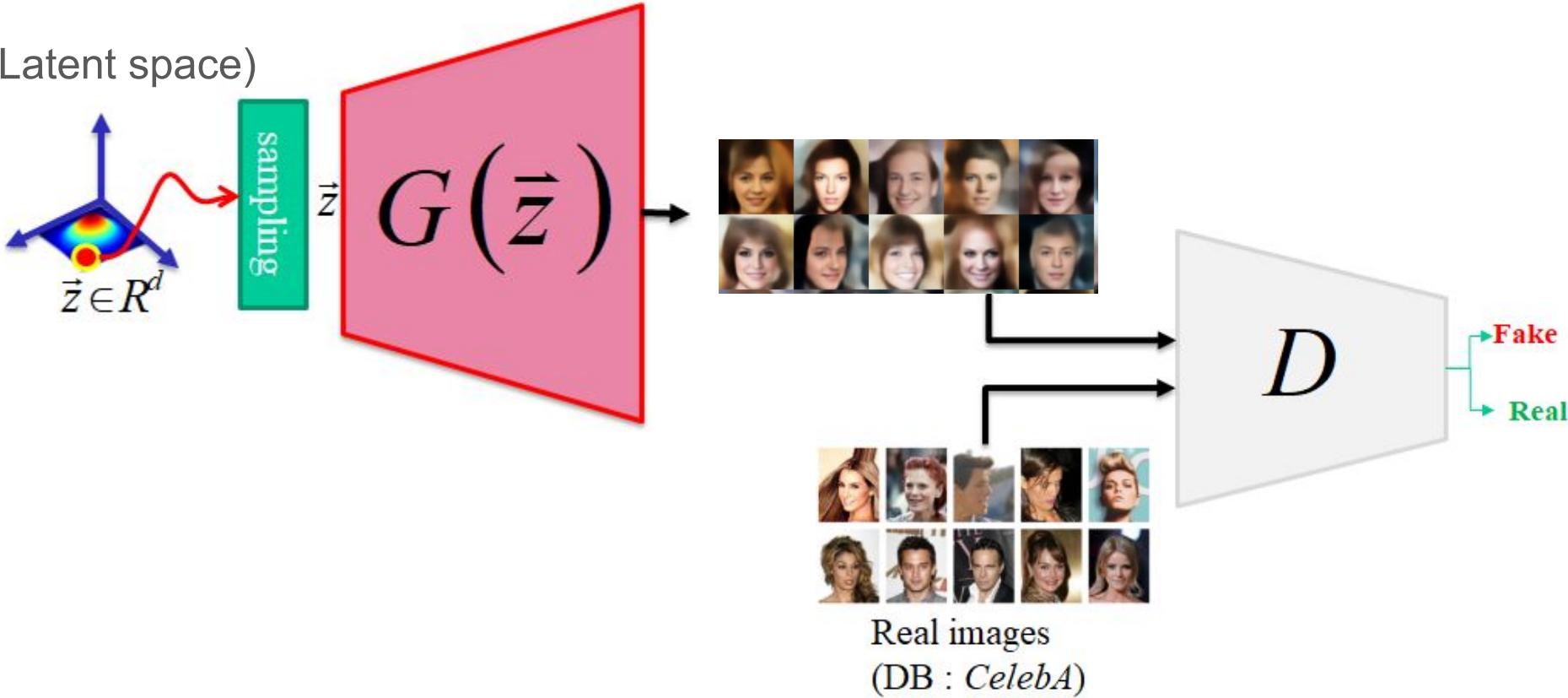


Loss?

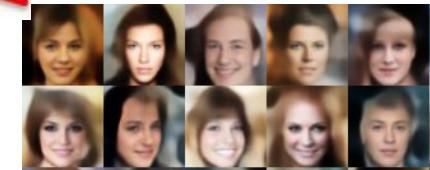
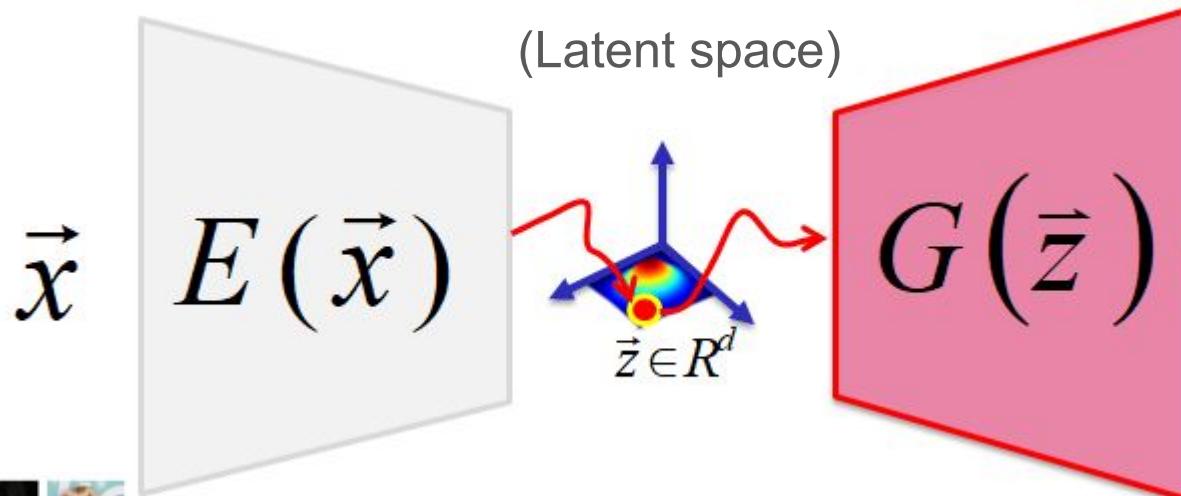
Annotated dataset?

# GAN

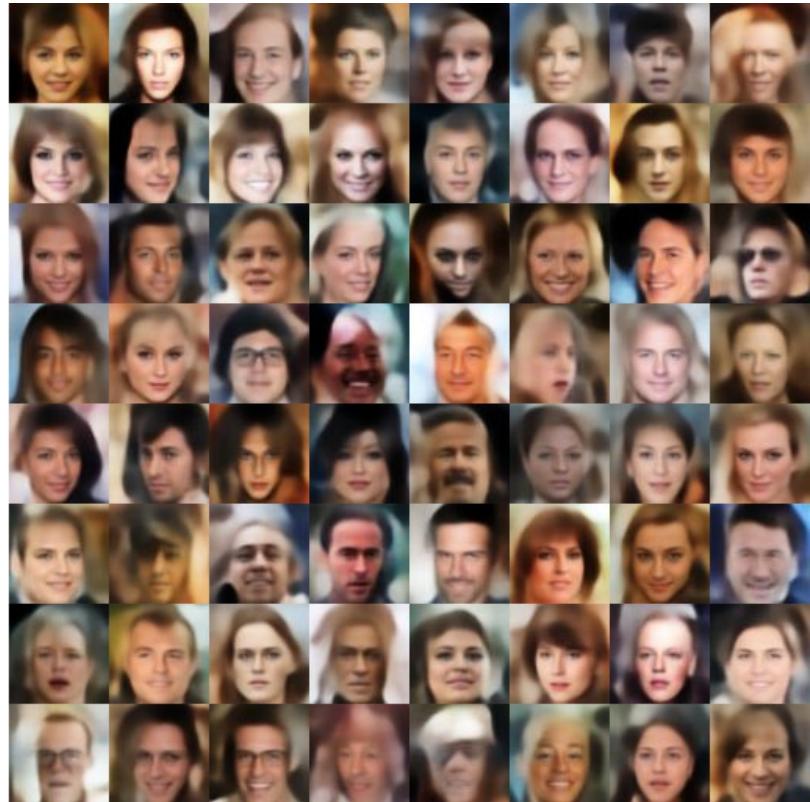
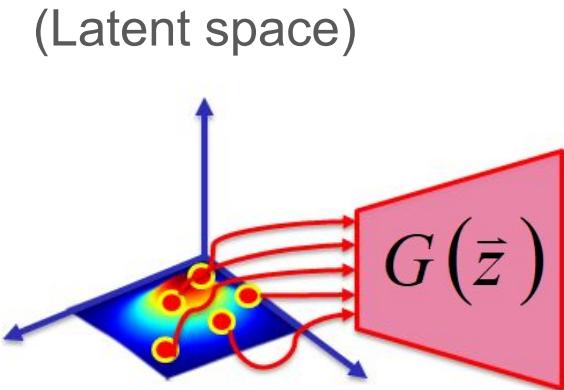
(Latent space)



# Autoencoders



# Autoencoders (once training is over)



# Summary

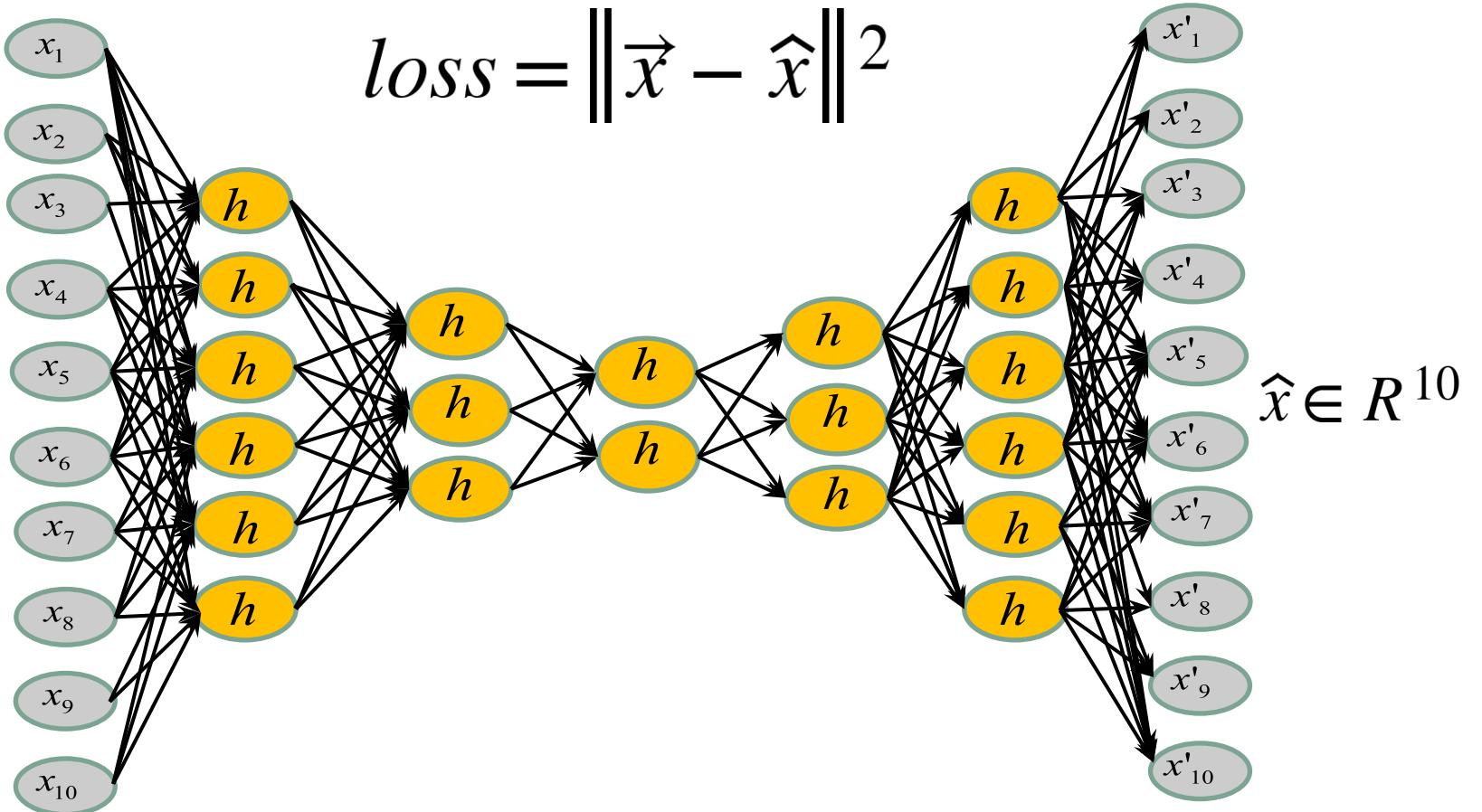
1. What are autoencoders and variational autoencoders?
2. How do they apply to MNIST (grayscale images)?
3. How do they apply to segmentation maps (ACDC cardiac labels)?

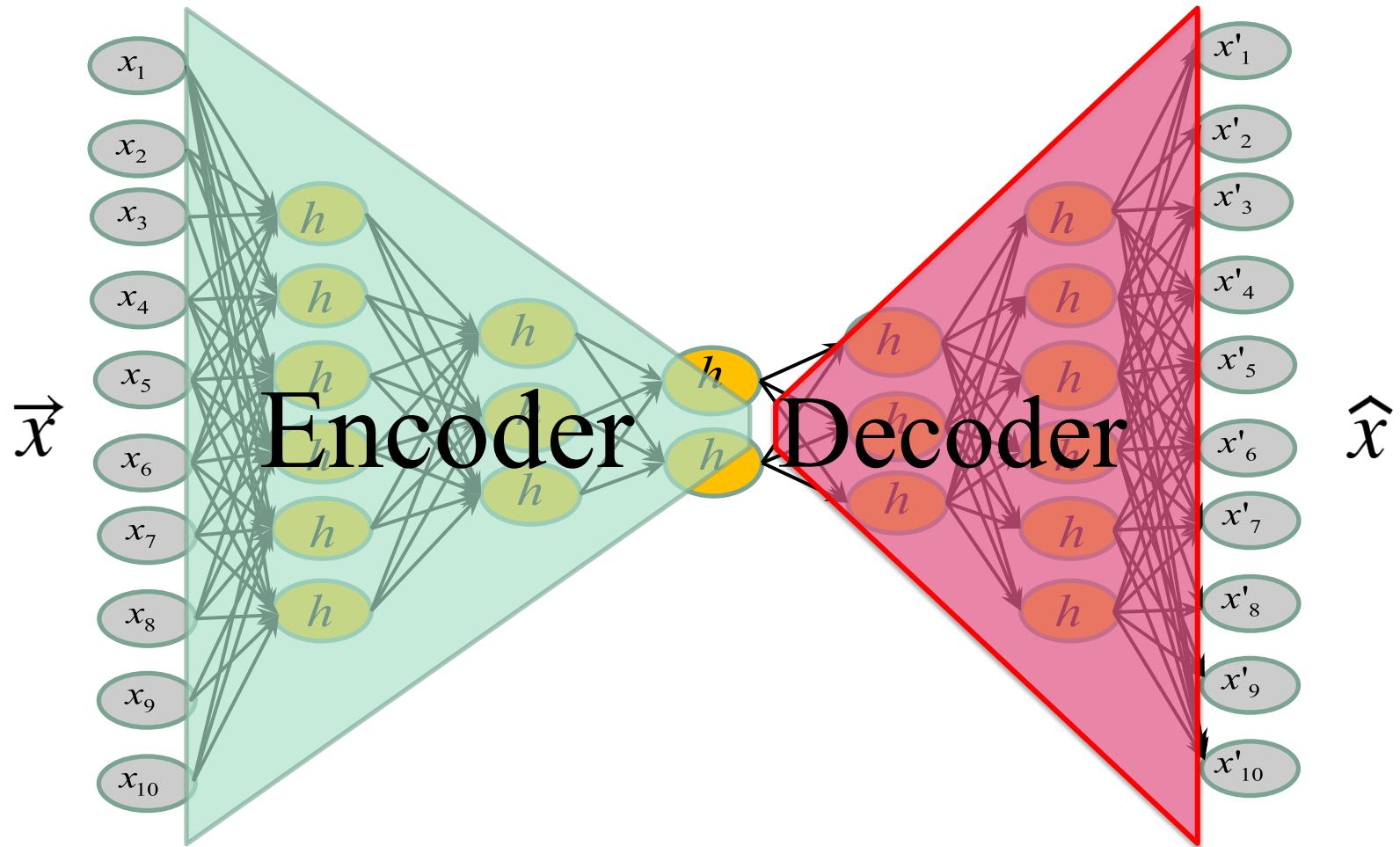
**Goal** : learn the latent representation of a set of data

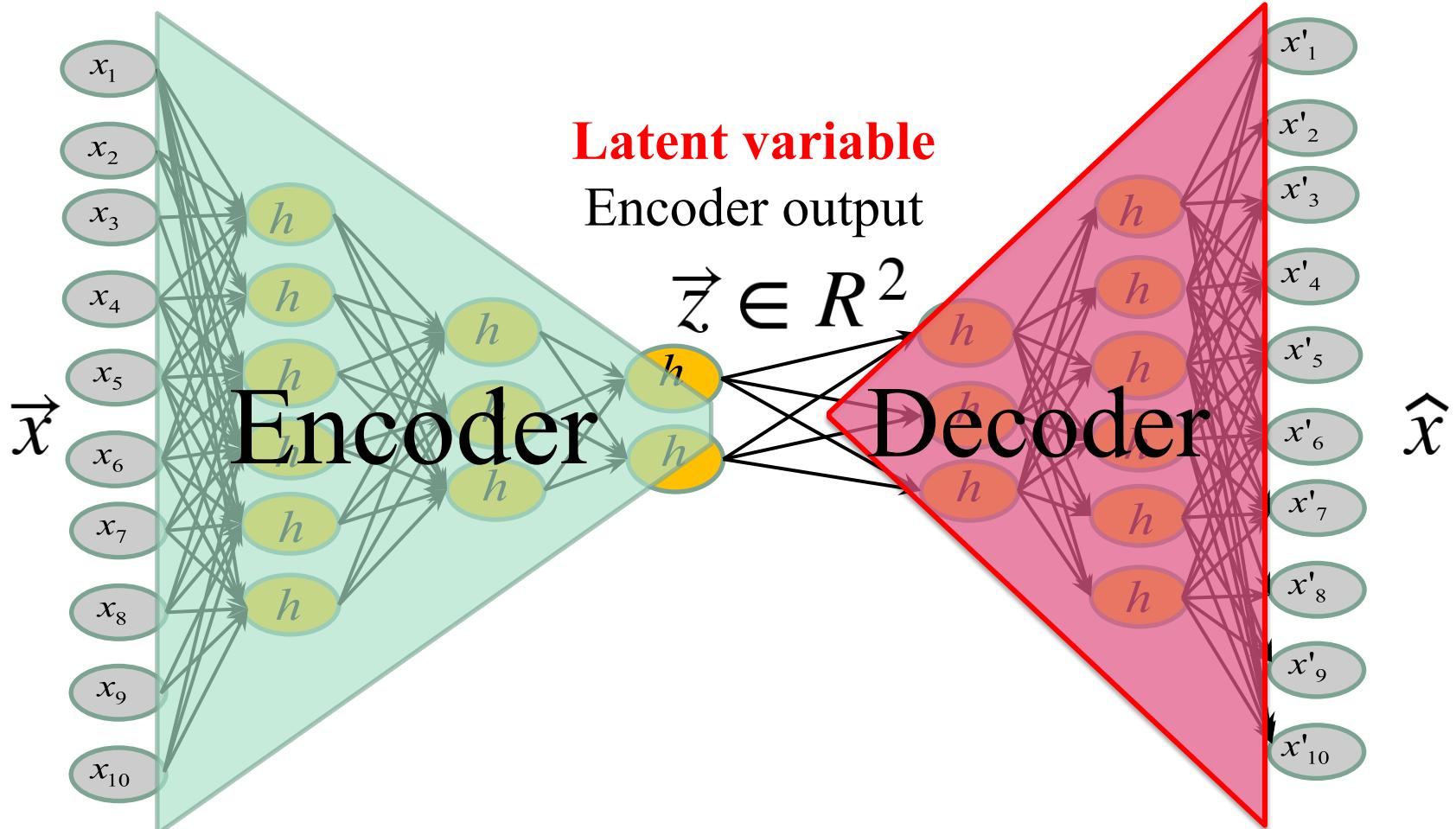
**How** : by training a Neural Net to output its own ...  
input!

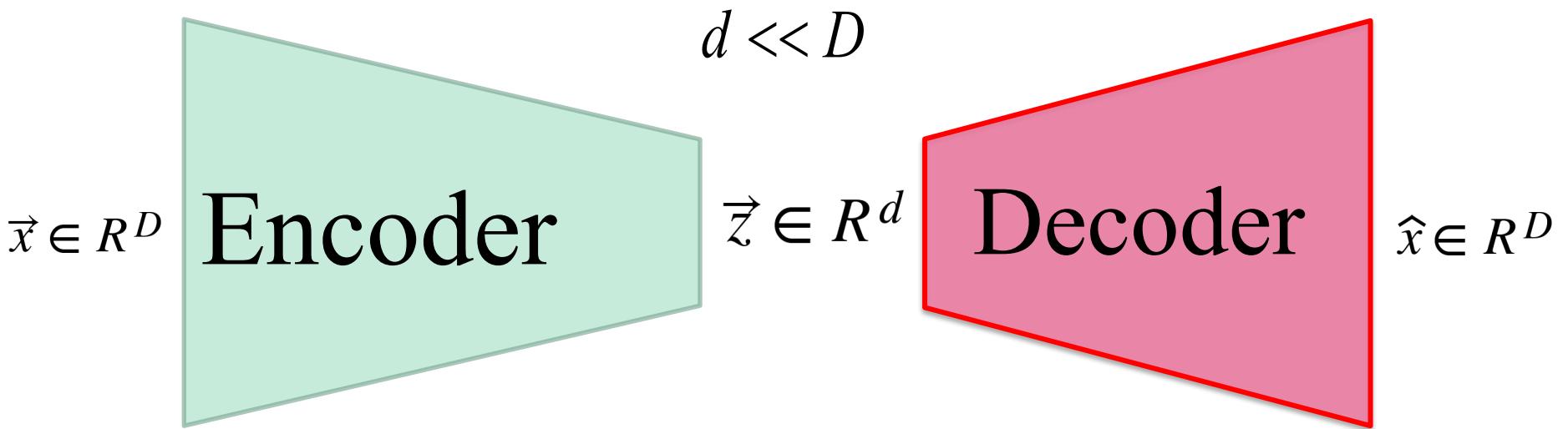
**Note** : if you are familiar with AE and VAE, you may skip the next couple of slides.

$$loss = \|\vec{x} - \hat{x}\|^2$$

 $\vec{x} \in R^{10}$ 

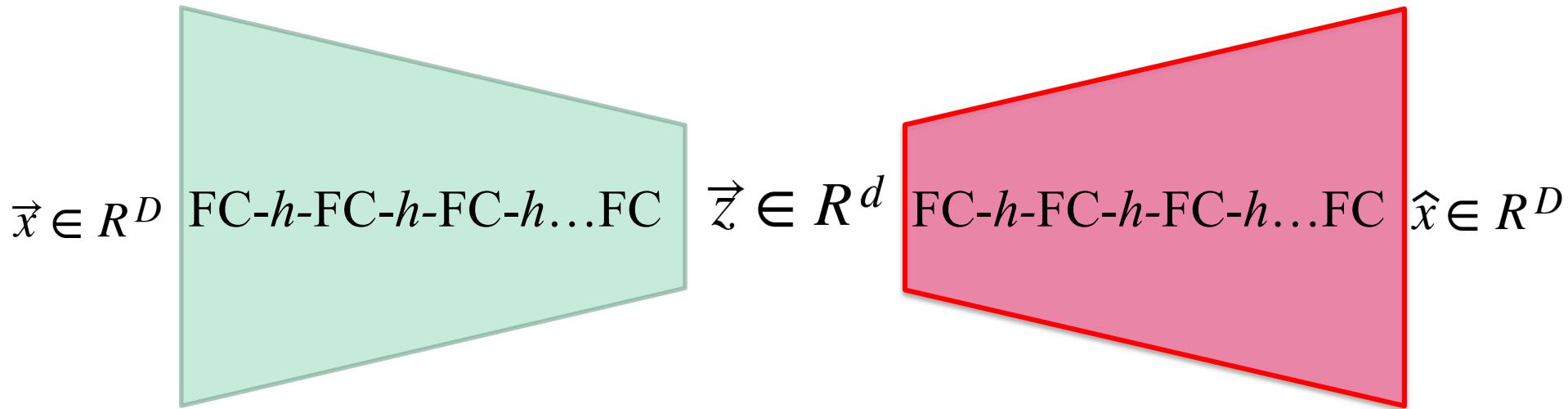






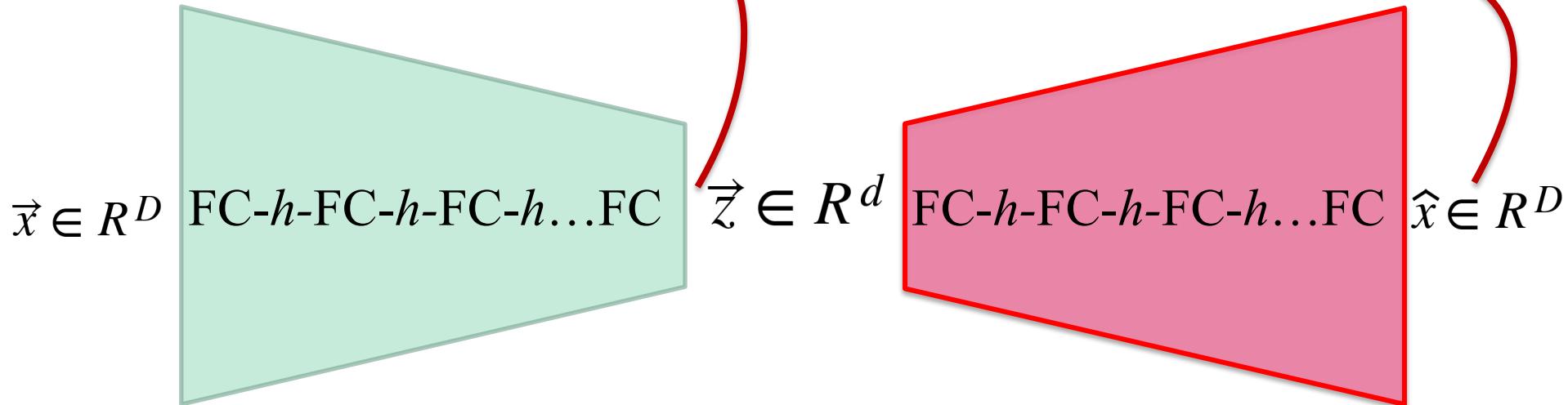
# Fully-Connected Layers

$h$  : activation function



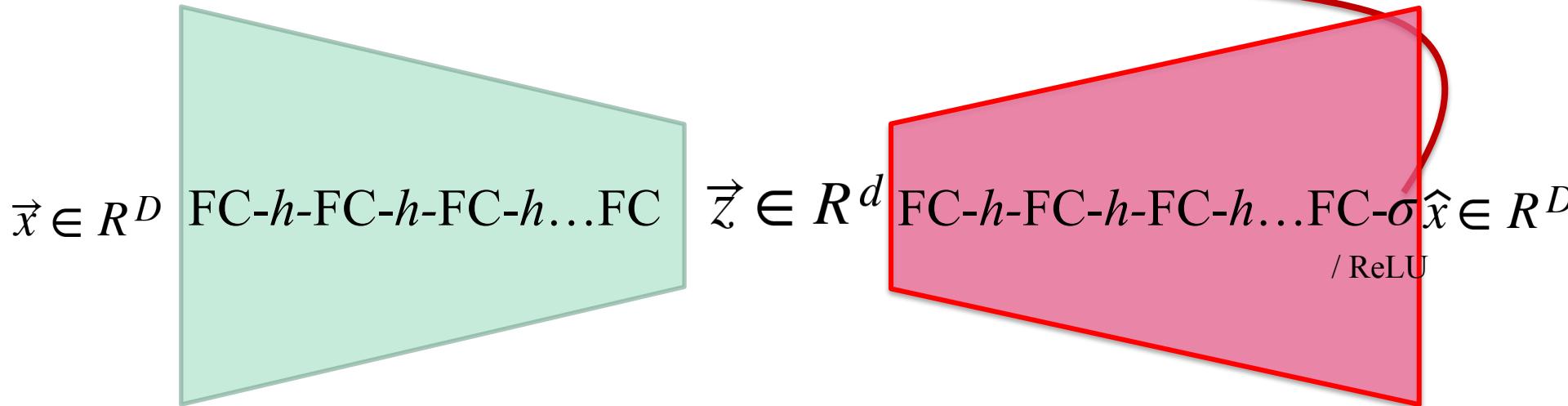
Very often...

No activation function at the output  
of the encoder and the decoder



See **why?**

Sometimes **sigmoid** to predict pixel values between 0 and 1 or **ReLU** when the pixel values can be large but never negative.

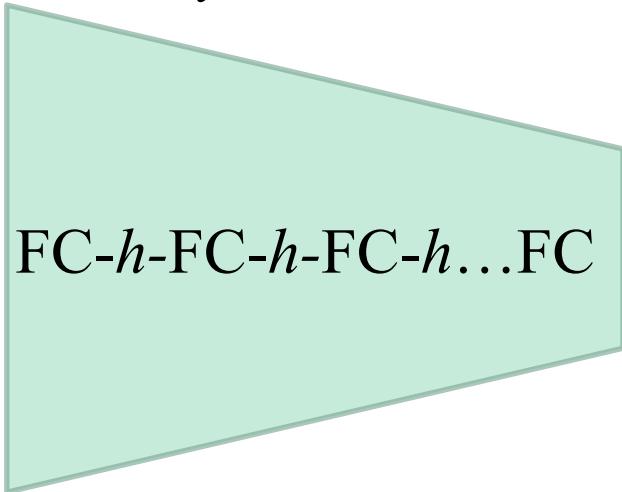


The number of neurons

## Decrease (or stay the same)

from a layer to another

$$\vec{x} \in R^D \quad \text{FC}-h\text{-FC}-h\text{-FC}-h\ldots\text{FC}$$

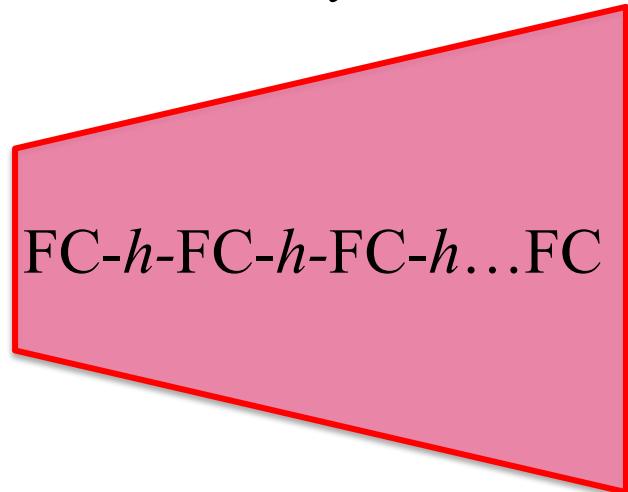


The number of neurons

## Increase (or stay the same)

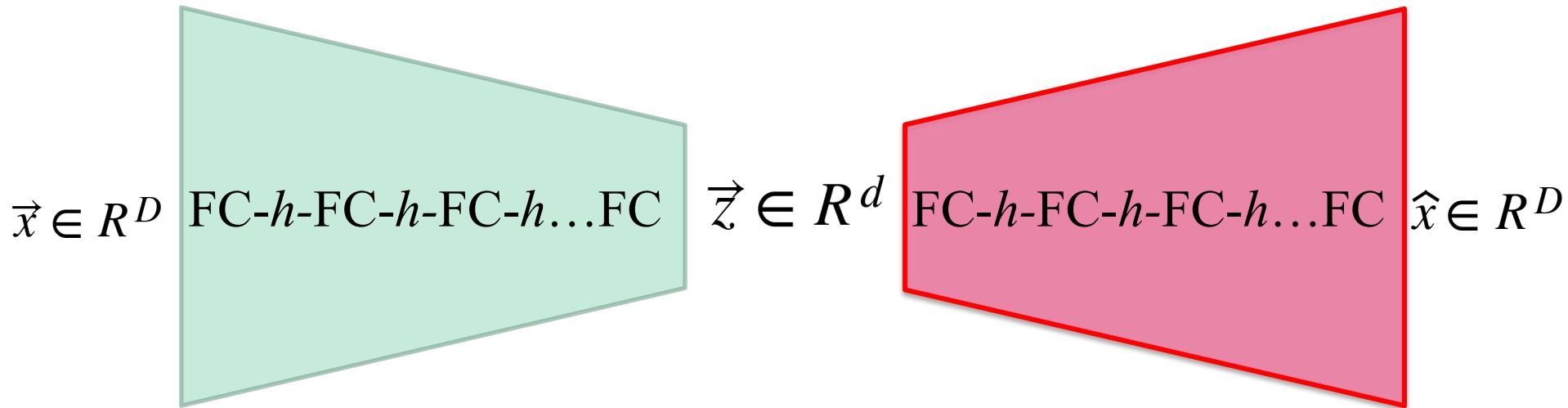
from a layer to another

$$\vec{z} \in R^d \quad \text{FC}-h\text{-FC}-h\text{-FC}-h\ldots\text{FC} \quad \hat{x} \in R^D$$

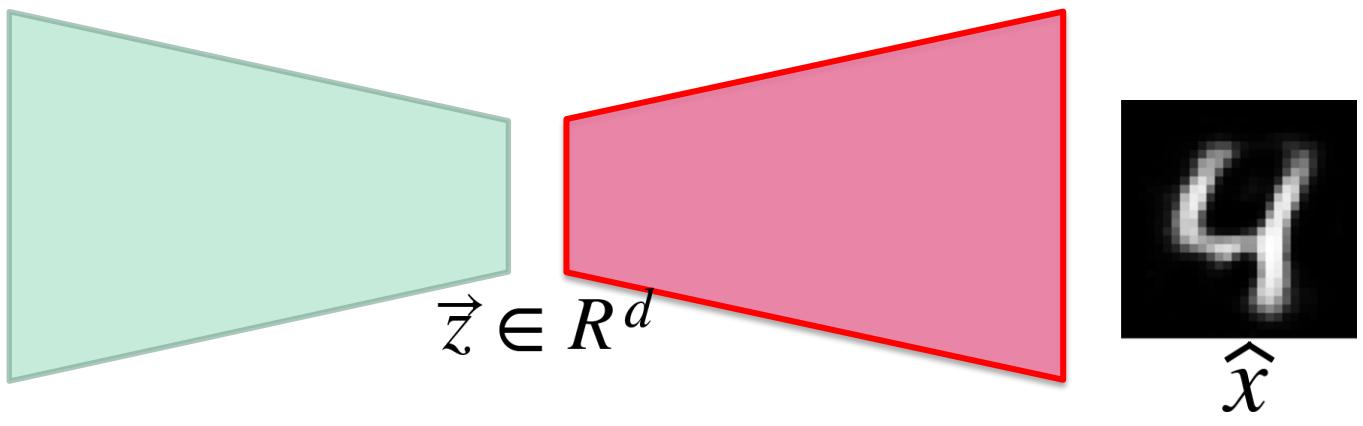
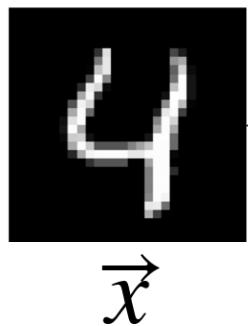


Very often...

The structure of the encoder is the **dual** of that of the decoder



## Basic image-based autoencoder



# Simple MNIST Autoencoder

```
class autoencoder(nn.Module):

    def __init__(self):
        super(autoencoder, self).__init__()

        self.encoder = nn.Sequential(
            nn.Linear(28 * 28, 128), nn.ReLU(True),
            nn.Linear(128, 64), nn.ReLU(True),
            nn.Linear(64, 12), nn.ReLU(True),
            nn.Linear(12, 2))

        self.decoder = nn.Sequential(
            nn.Linear(2, 12), nn.ReLU(True),
            nn.Linear(12, 64), nn.ReLU(True),
            nn.Linear(64, 128), nn.ReLU(True),
            nn.Linear(128, 28 * 28))
```

Latent space 2D

```
def forward(self, x):
    z = self.encoder(x)
    x_prime = self.decoder(z)
    return x_prime
```

# Simple MNIST Autoencoder

```
class autoencoder(nn.Module):  
  
    def __init__(self):  
  
        super(autoencoder, self).__init__()  
  
        self.encoder = nn.Sequential(  
  
            nn.Linear(28 * 28, 128), nn.ReLU(True),  
            nn.Linear(128, 64), nn.ReLU(True),  
            nn.Linear(64, 12), nn.ReLU(True),  
            nn.Linear(12, 2))  
  
        self.decoder = nn.Sequential(  
  
            nn.Linear(2, 12), nn.ReLU(True),  
            nn.Linear(12, 64), nn.ReLU(True),  
            nn.Linear(64, 128), nn.ReLU(True),  
            nn.Linear(128, 28 * 28))
```

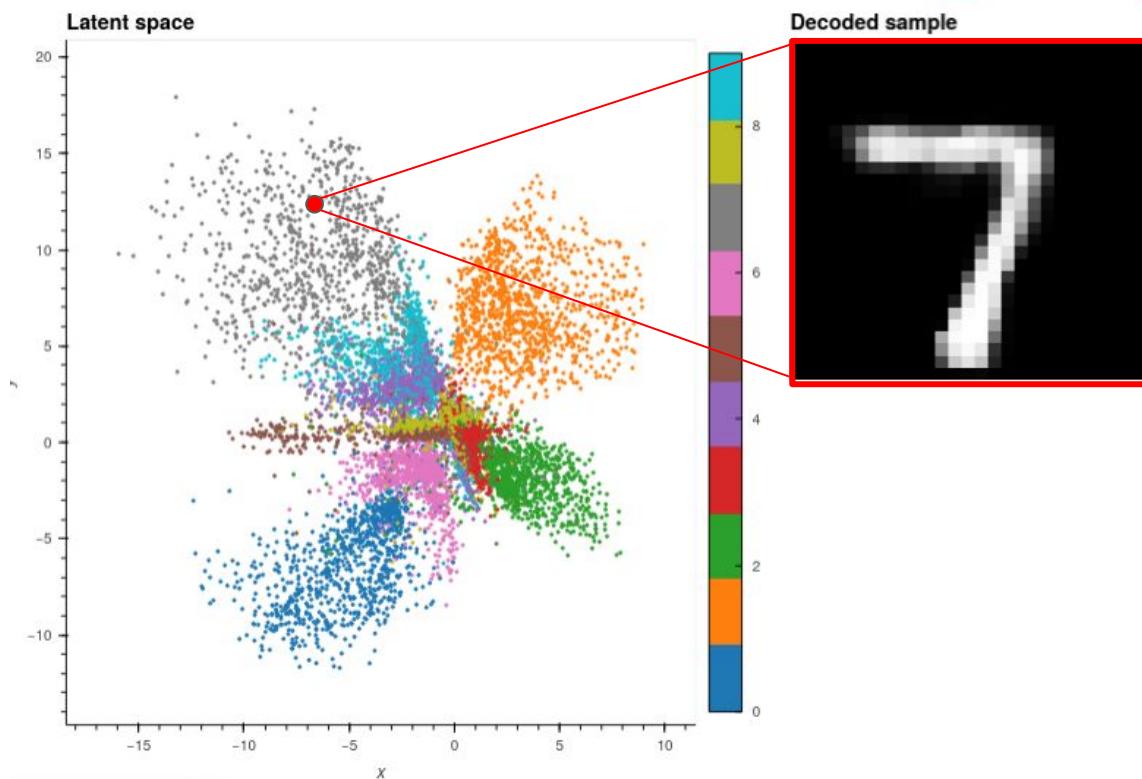
symmetry



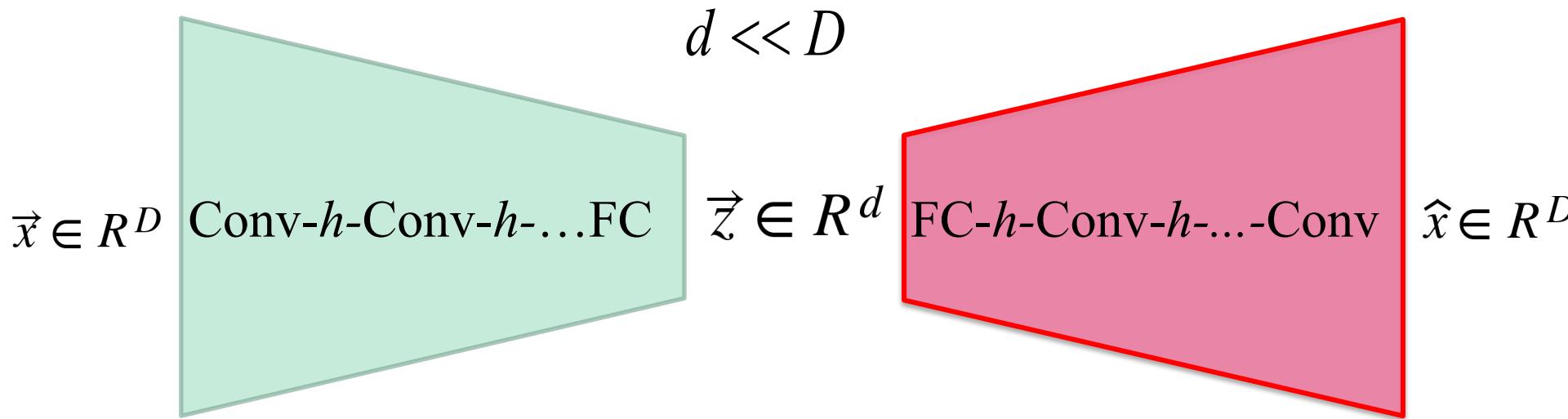
```
def forward(self, x):  
  
    z = self.encoder(x)  
  
    x_prime = self.decoder(z)  
  
    return x_prime
```

# MNIST latent space (for 1000 images)

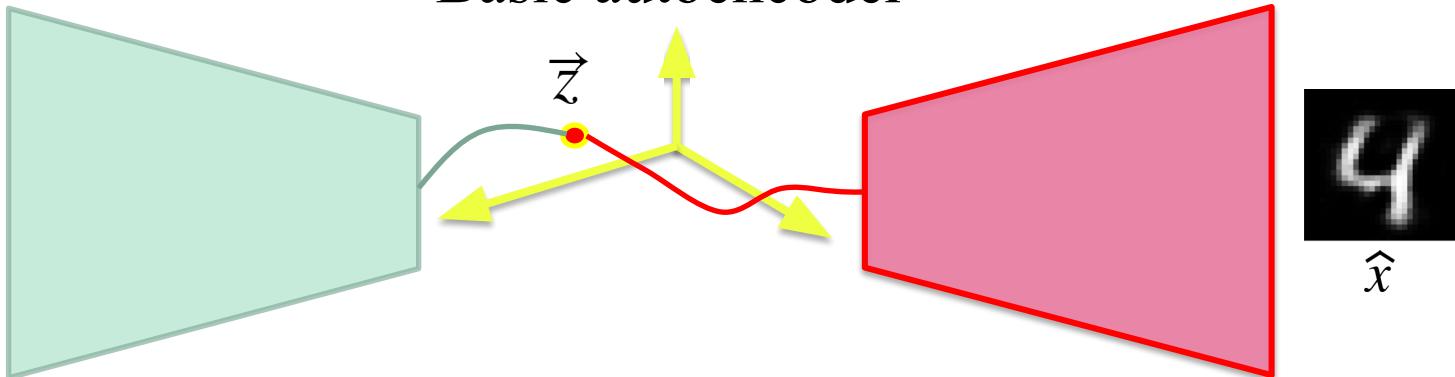
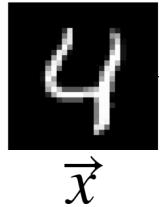
Each 2D point corresponds to an image



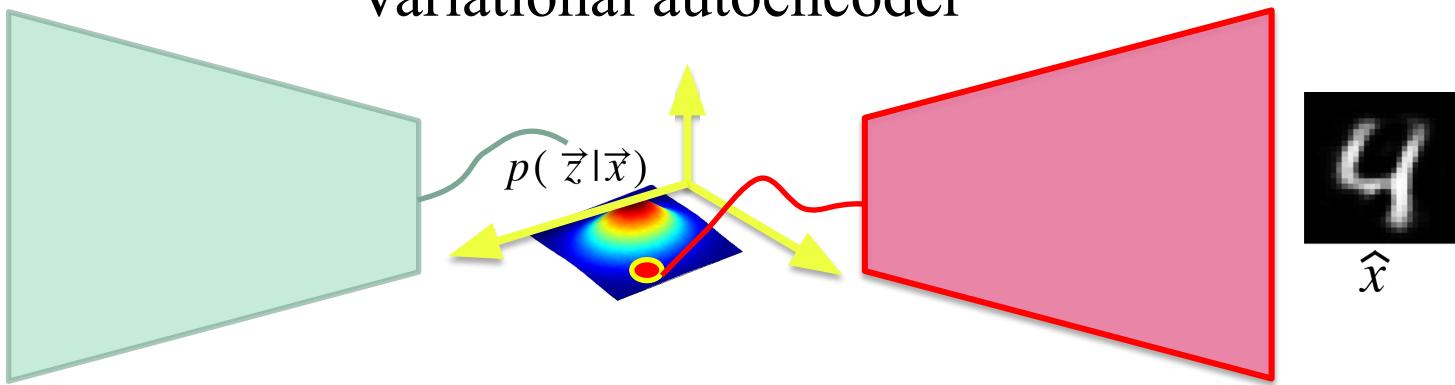
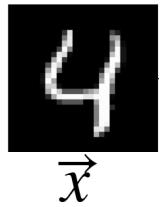
# Conv layers



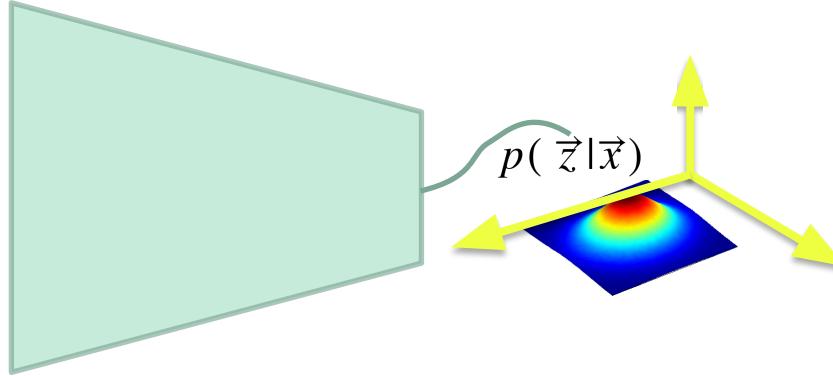
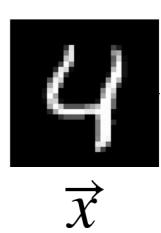
## Basic autoencoder



## Variational autoencoder

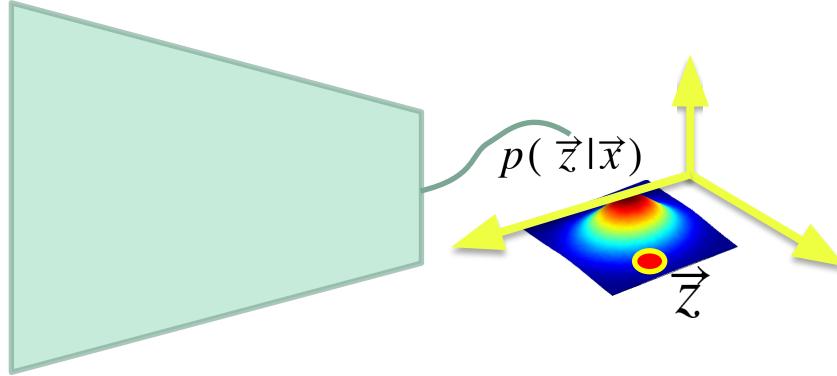
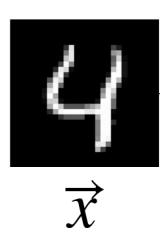


# Variational autoencoder



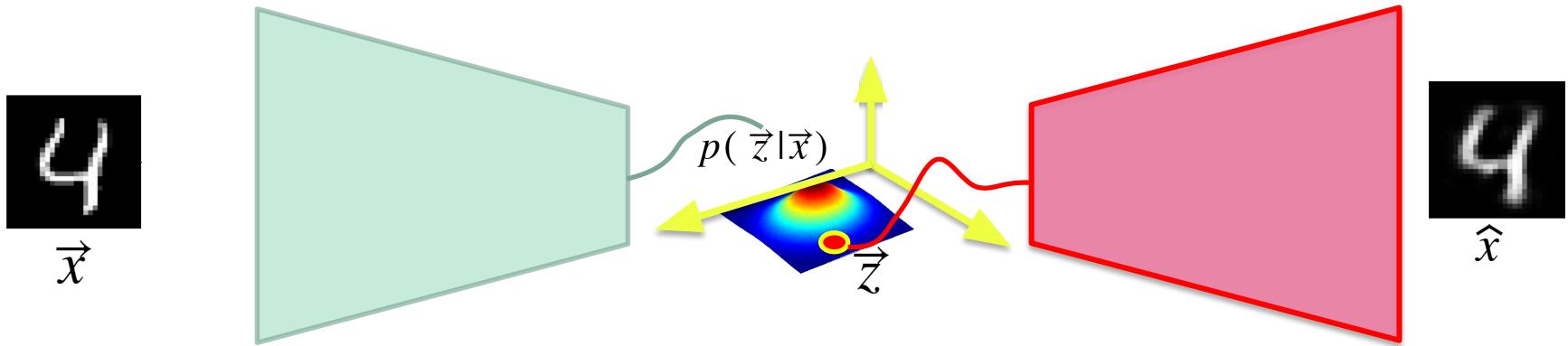
The encoder outputs a **distribution**  $p(\vec{z}|\vec{x})$  and not just a **vector**  $\vec{z}$

# Variational autoencoder



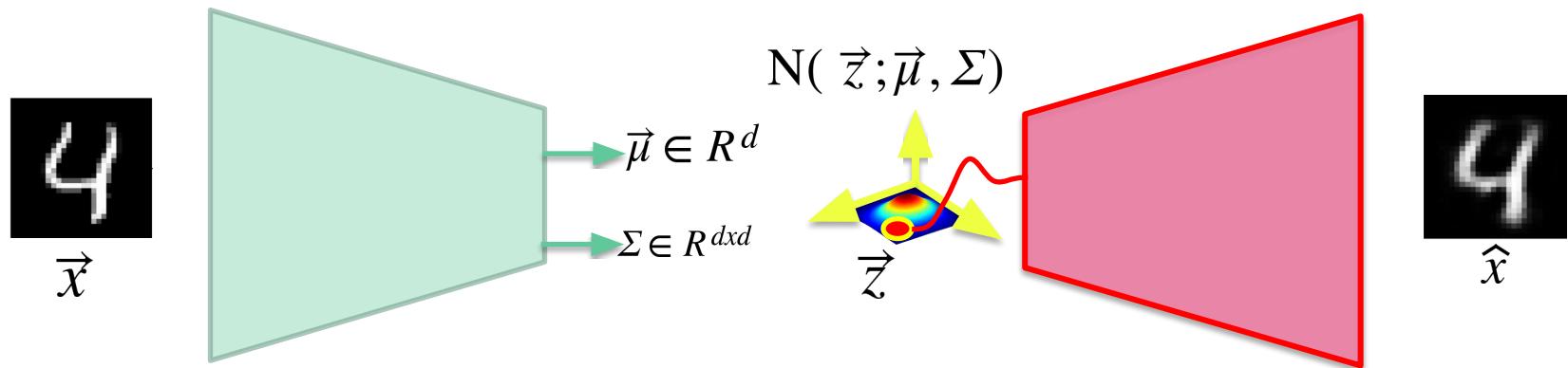
Random sample  $\vec{z} \sim P(\vec{z}|\vec{x})$

# Variational autoencoder

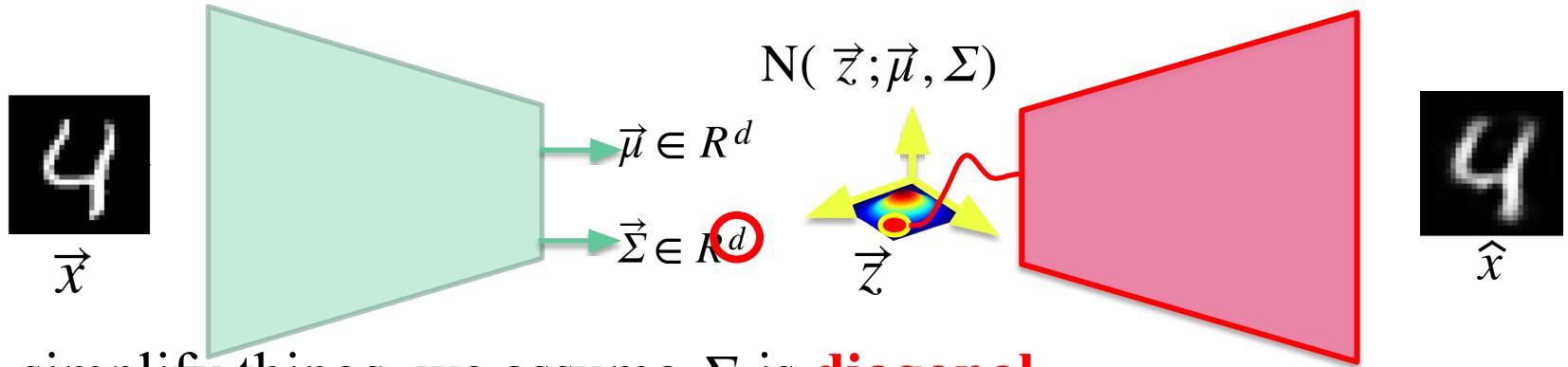


...and rebuild  $\hat{x}$

$$p(\vec{z}|\vec{x}) \sim Gaussian$$



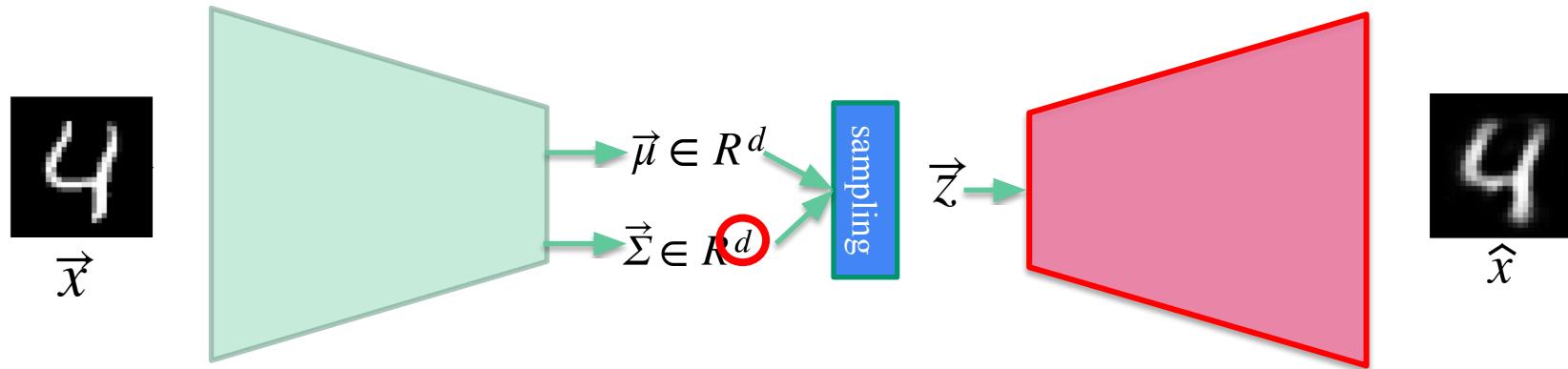
$$p(\vec{z} | \vec{x}) \sim Gaussian$$



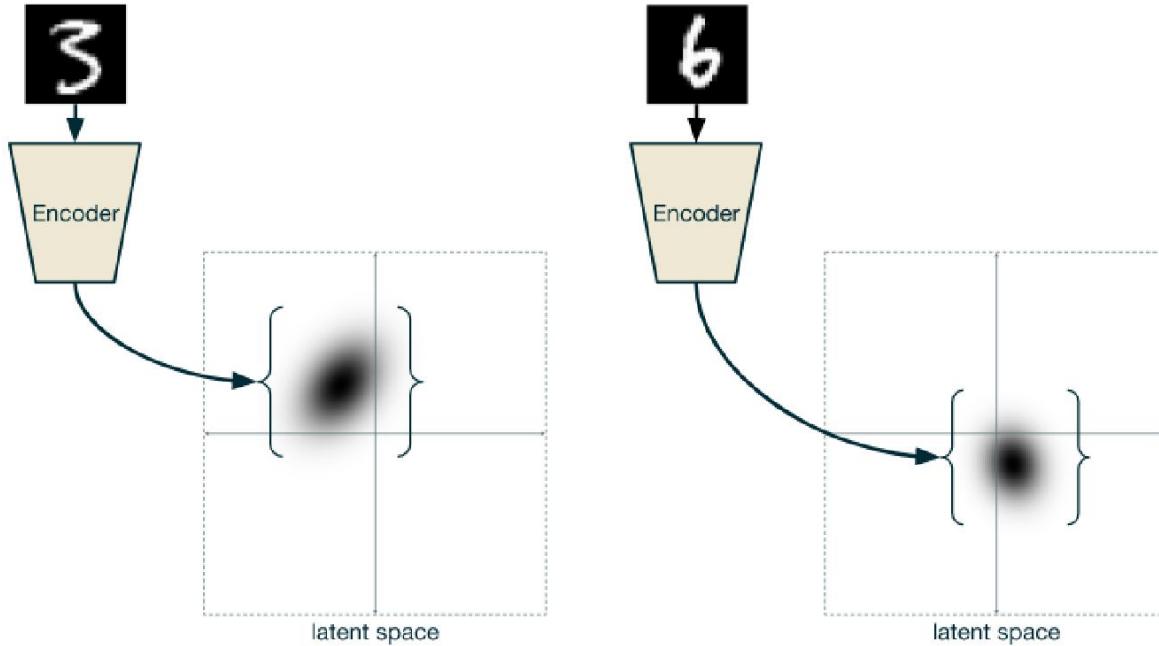
To simplify things, we assume  $\Sigma$  is **diagonal**

$$\Sigma = \begin{pmatrix} \sigma_1^2 & 0 & 0 & 0 \\ 0 & \sigma_1^2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \sigma_d^2 \end{pmatrix}$$

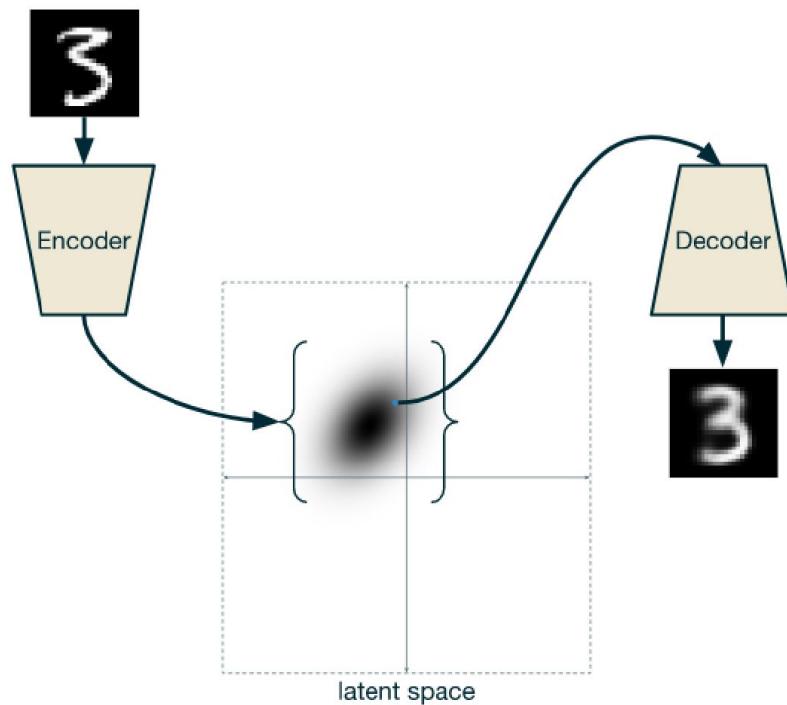
# Variational autoencoder



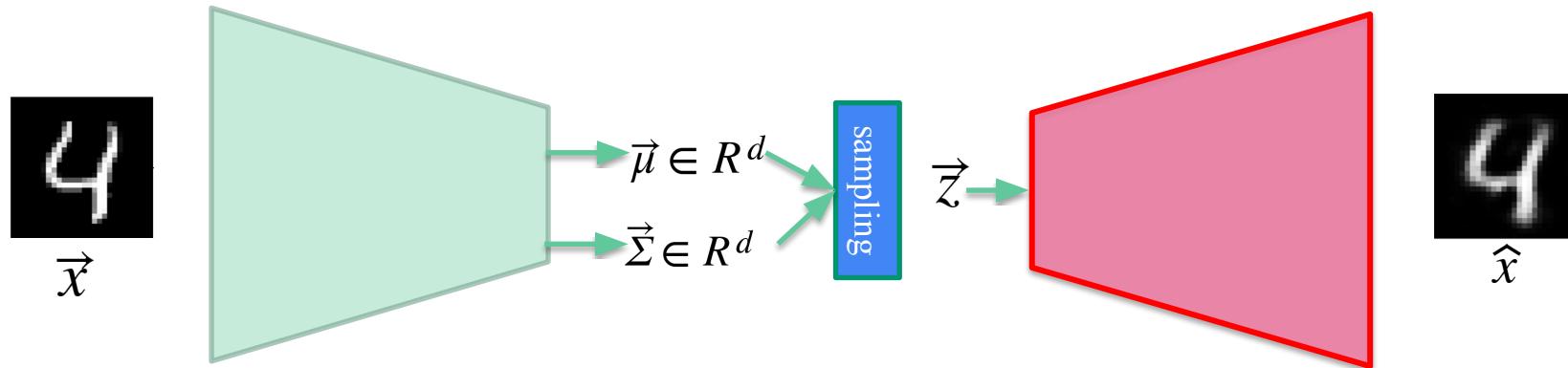
# Another way of seeing things...



# Another way of seeing things...



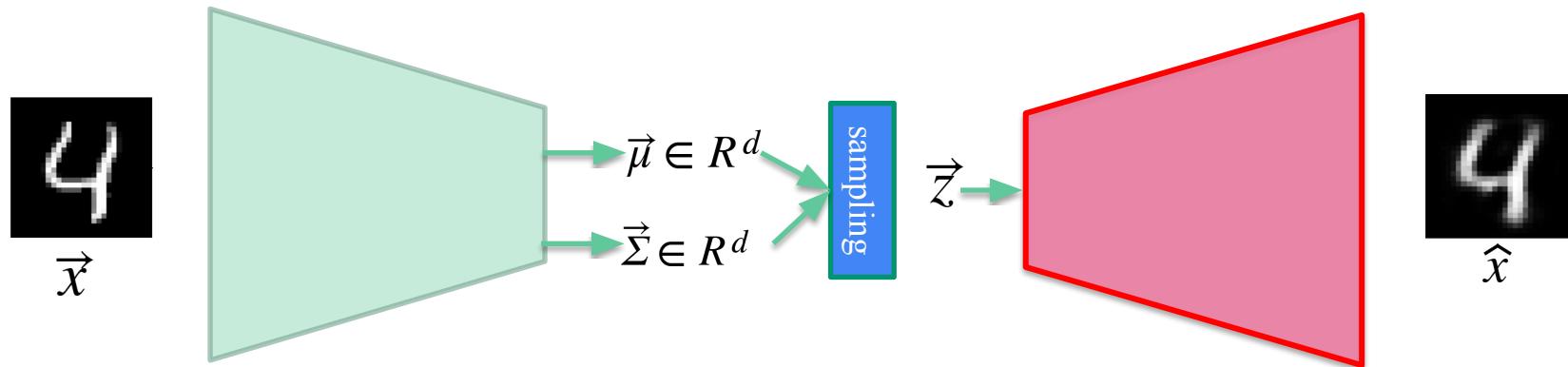
# *ELBO loss : Evidence Lower Bound loss*



$$Loss = (\vec{x} - \hat{x})^2 + \lambda KL\left( N(\vec{z}; \vec{0}, \vec{1}), N(\vec{z}; \vec{\mu}, \vec{\Sigma}) \right)$$

Loss decoder      Loss encoder

# Other loss (in case the output is binary)

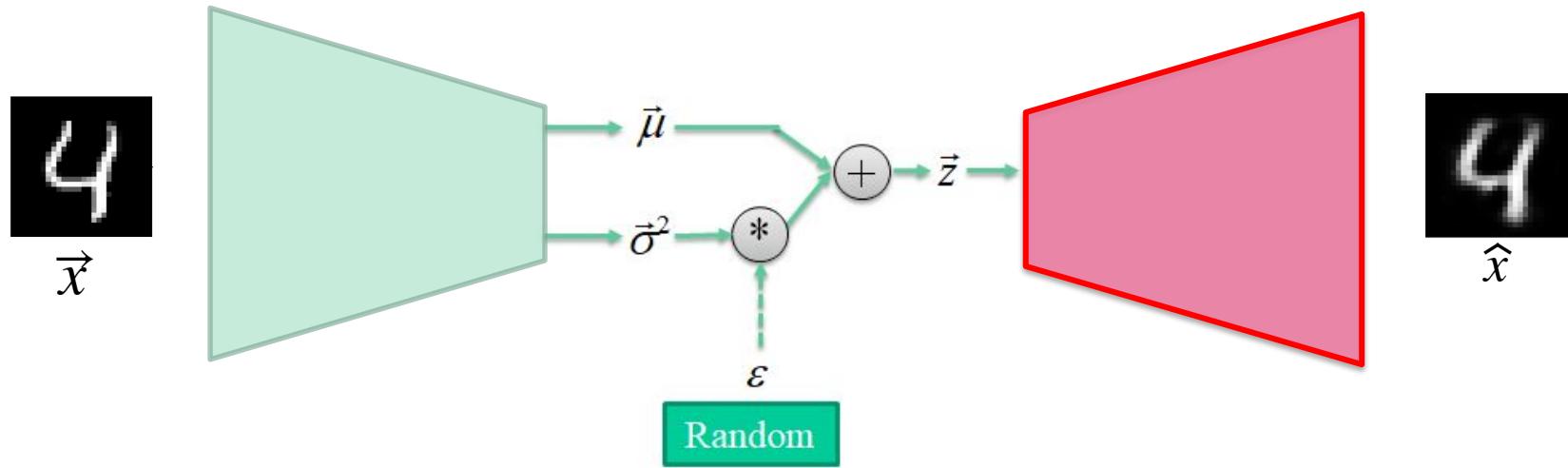


$$Loss = CrossEntropy(\vec{x}, \hat{x}) + \lambda KL\left( N(\vec{z}; \vec{0}, \vec{1}), N(\vec{z}; \vec{\mu}, \vec{\Sigma}) \right)$$

Loss decoder

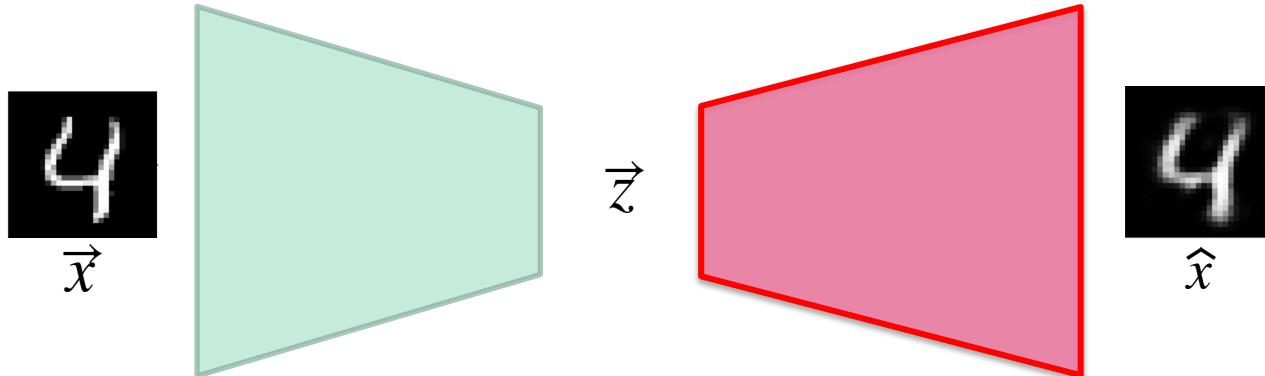
Loss encoder

# Reparametrization trick instead of sampling

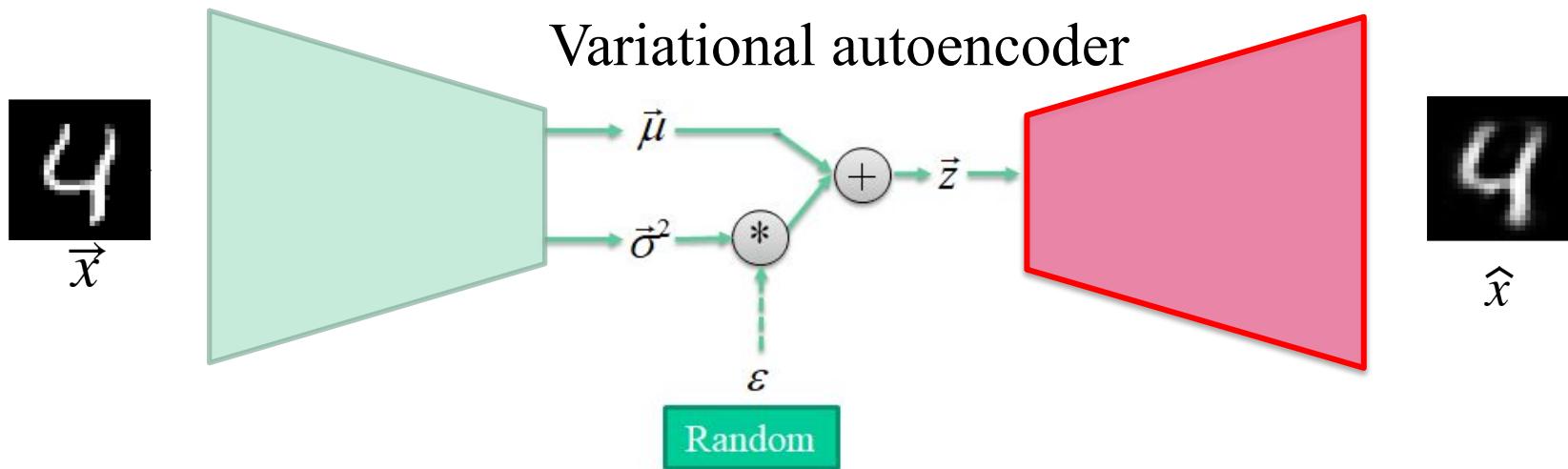


# MNIST

## Basic autoencoder

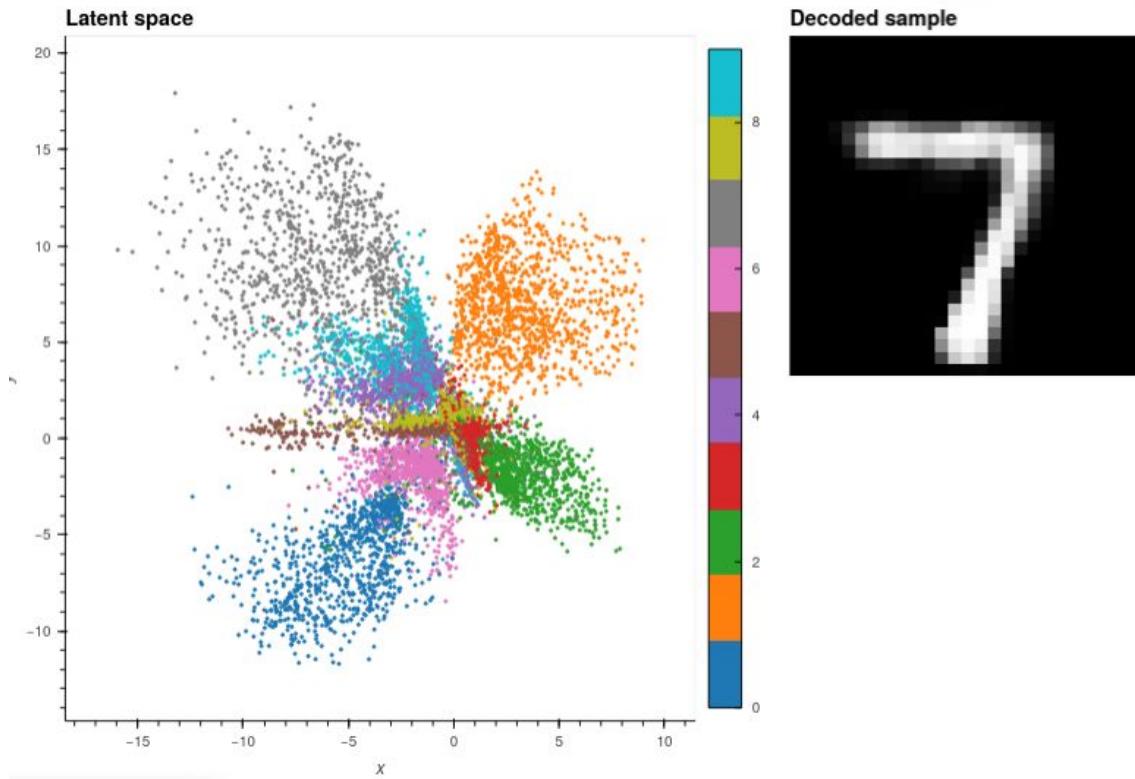


## Variational autoencoder



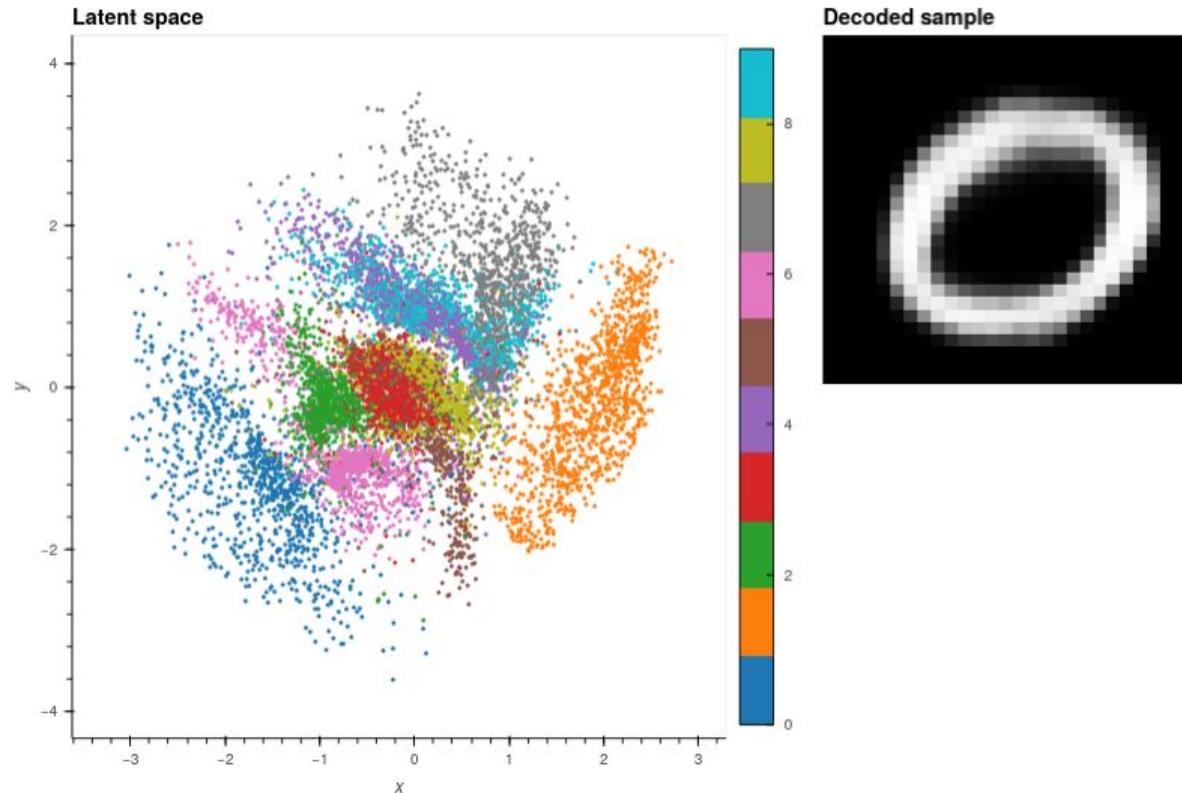
# MNIST

Visualize the latent space (autoencoder)



# MNIST

Visualize the latent space (variational autoencoder)



# MNIST

Code is in the file:

**mnist-autoencoders.ipynb**

# Automated Cardiac Diagnosis Challenge



150 exams (all from different patients) divided into 5 groups:

- dilated cardiomyopathy (DCM),
- hypertrophic cardiomyopathy (HCM),
- myocardial infarction (MINF),
- abnormal right ventricle (RV)
- patients without cardiac disease (NOR).

website: *creatis.insa-lyon.fr/Challenge/acdc*

The screenshot shows the homepage of the ACDC challenge. At the top, there's a banner featuring a grayscale MRI scan of a heart. The text on the banner reads "Automated Cardiac Diagnosis Challenge (ACDC)" and "MICCAI challenge 2017". Below the banner, it says "In conjunction with the STACOM workshop" and "Go to evaluation platform".

The main content area has a sidebar on the left with links: Overview, Contest, Participation, Databases, Evaluation, Code, Paper Submission, and Contact. The "Overview" link is highlighted in orange.

The main content area has a title "Overview" in orange. Below it are three tabs: "General context" (highlighted in blue), "Scientific interests", and "Organizers".

The "General context" tab contains the following text:

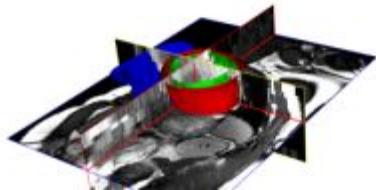
The goal of this contest is two-fold:

- compare the performance of automatic methods on the segmentation of the left ventricular endocardium and epicardium as the right ventricular endocardium for both end diastolic and end systolic phase instances;
- compare the performance of automatic methods for the classification of the examinations in five classes (normal case, heart failure with infarction, dilated cardiomyopathy, hypertrophic cardiomyopathy, abnormal right ventricle).

This will be done using a common database of 3D cine-MRI images acquired from 150 patients (30 per pathology plus 30 healthy subjects) and the associated manual references based on the analysis of a clinical expert.

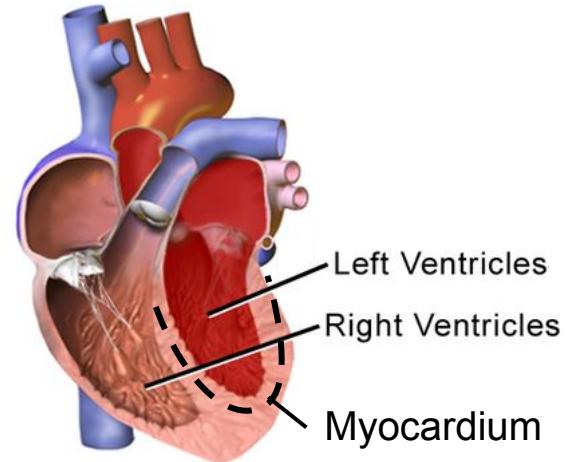
To the right, there's a "NEWS" section with three items:

- 4th of April 2017**  
The full training dataset (100 patients) is available
- 13th of March 2017**  
Challenge accepted by the MICCAI Satellite Committee
- 10th of January 2017**  
The online evaluation platform is operational



# Cardiac anatomy

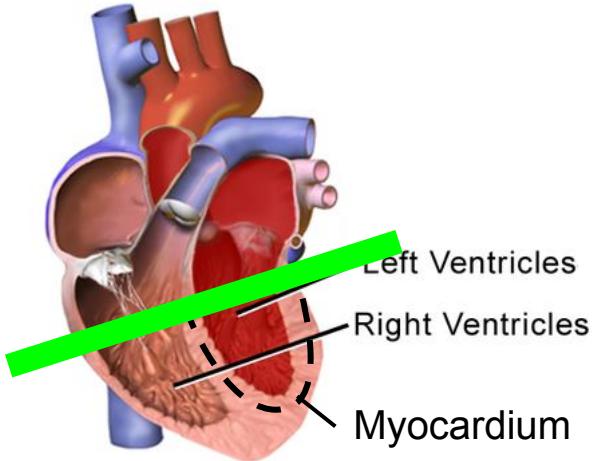
## Normal Heart



Chambers relax and fill,  
then contract and pump.

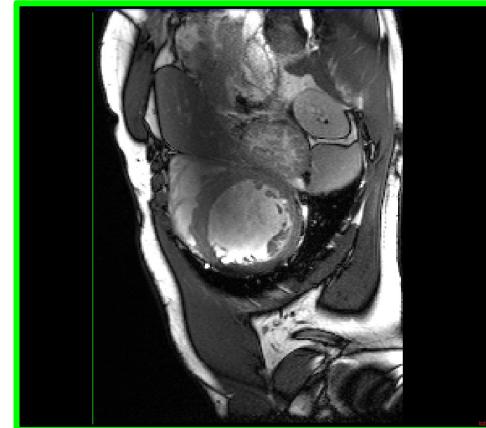
# Cardiac anatomy

## Normal Heart



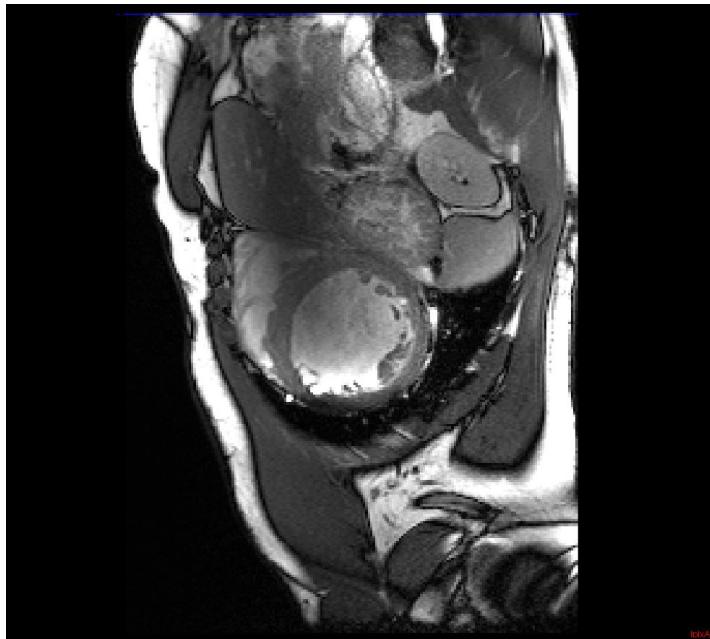
Chambers relax and fill,  
then contract and pump.

Cross-section : **short axis view**

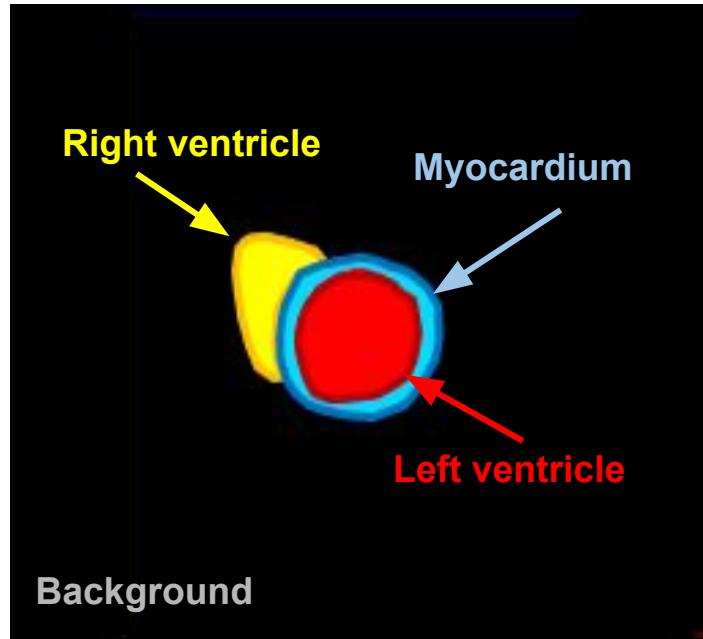




Short axis cardiac MRI

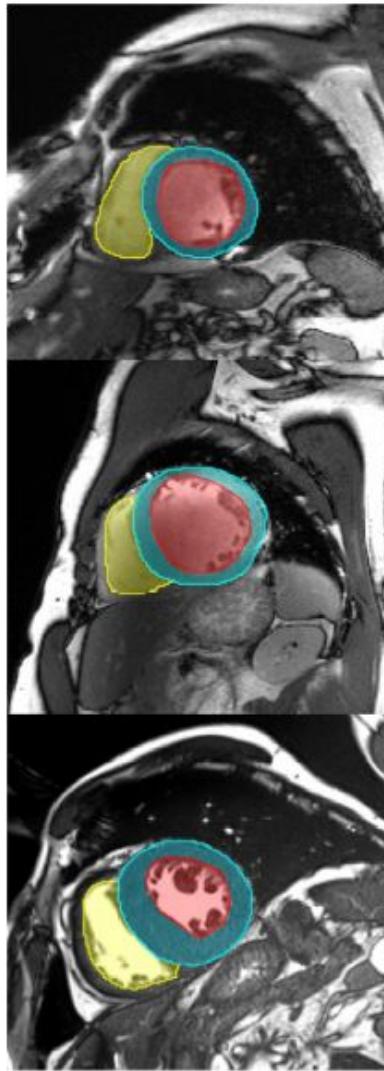


Short axis segmentation map

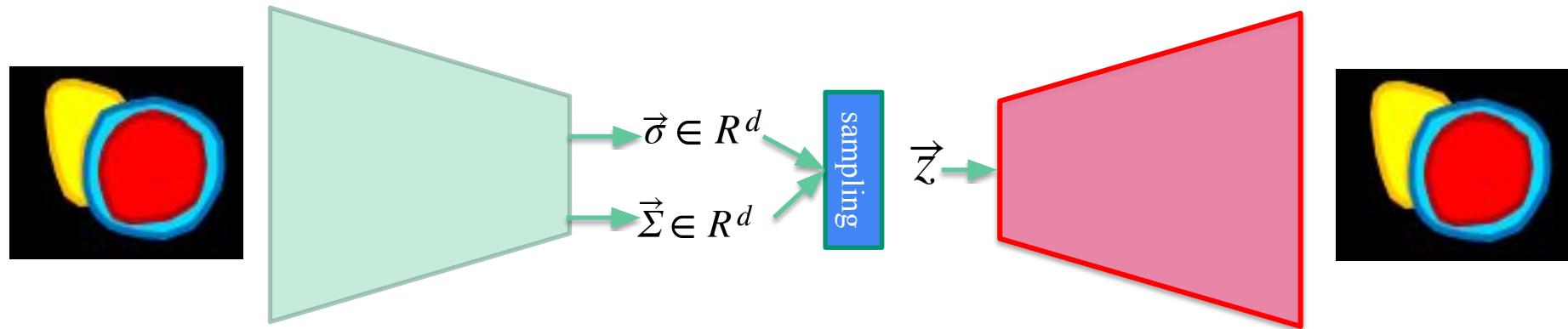




Other examples:



## Convolutional Variational autoencoder



**ACDC**

Code is in the file:

**cardiac-mri-autoencoders.ipynb**

Thank you



Image

saturncloud/saturn-pytorch

Version

2022.03.01

Spot Instance

This requests the server of specified size as Spot Instance. Spot Instances are less expensive, but may be shut down at any time (with a two-minute warning). ([?](#))

Working Directory

/home/jovyan/workspace

Extra Packages

Extra packages are installed every time

If you find yourself adding the same packages to lots of resources, you may want to permanently add packages to a custom image instead. ([?](#))

**Write opencv-python torchvision==0.11.2 for pip install**  
**Write opencv-python torchvision==0.18.1 for pip install**

Conda Install

Pip Install

Apt Packages

opencv-python torchvision==0.11.2

The packages together will run the following script:

```
apt-get install htop zip unzip python3-opencv  
pip install opencv-python torchvision==0.11.2
```

Shutdown After

1 hour

Disabled options are not supported in the Free plan. Upgrade to Pro for unlimited resources.

Advanced Settings (optional) ▾

Save

Cancel

**Image**

saturncloud/saturn-pytorch

**Version**

2022.03.01

**Spot Instance**  
This requests the server of specified size as Spot Instance. Spot Instances are less expensive, but may be shut down at any time (with a two-minute warning). (?)

**Working Directory**

/home/jovyan/workspace

**Extra Packages**  
Extra packages are installed every time the resource starts up - right before the start script. Use spaces to separate packages.  
If you find yourself adding the same packages to lots of resources, you may want to permanently add packages to a custom image instead. (?)

Conda Install      Pip Install      **Apt Packages**

htop zip unzip python3-opencv

The packages together will run the following script:  
`apt-get install htop zip unzip python3-opencv  
pip install opencv-python`

**Shutdown After**

1 hour

Disabled options are not supported in this version.

**Advanced Settings (optional)**

**And click Create**

**Create**      **Cancel**

**Write these for apt install**

**htop zip unzip python3-opencv**

**And click Create**

**Create**      **Cancel**

 **Saturn Cloud**

@ pierremarcjodoin ▾

- Resources
- Credentials
- Git Repositories
- Images
- Enterprise
- Get Started Next:  
Spin up Dask

HOURS REMAINING  
 Upgrade for More

Jupyter Dask	30 hrs
	3 hrs

Resources / pierremarcjodoin / AutoEncoderHandsOn

## JUPYTER SERVER

pierremarcjodoin / AutoEncoderHan...

04fd2176712f4c02a0e70719577c284b

 Recipe

 Logs

 Edit

 Delete

 Overview

 Environment

 Git Repos

 Share

Jupyter Server stopped

T4-XLarge - 4 cores - 16 GB RAM - 1 GPU - 10Gi Disk

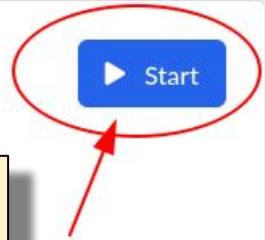
### Metrics

Auto Shutoff: 1 hour

Spot Instance: No

SSH URL: (not enabled) [\(?\)](#)

Click [Start](#)



Jupyter Lab

 **Saturn Cloud**

@ pierremarcjodoin ▾

Resources / pierremarcjodoin / AutoEncoderHandsOn2

JUPYTER SERVER

pierremarcjodoin / AutoEncoderHan...

1fc514f792564bc383f7cbda6237181f

Overview

Credentials

Git Repositories

Images

Enterprise

Get Started Next:  
Spin up Dask

HOURS REMAINING  
 Upgrade for More

Jupyter	30 hrs
Dask	3 hrs



You will see a progression bar.  
Please wait a minute or so



Jupyter Server 

T4-XLarge - 4 cores - 16 GB RAM - 1 GPU - 100GB DISK

 Start

 Stop

#### Metrics

Auto Shutoff: 1 hour

Spot Instance: No

SSH URL: (not enabled) [\(?\)](#)



Jupyter Lab

The screenshot shows the Jupyter Notebook interface with a 'Launcher' tab selected. The left sidebar displays a file tree with a single folder named 'workspace'. A red arrow points from a callout box to the folder icon. The main workspace shows a 'Notebook' icon and a 'Console' icon, both associated with 'saturn (Python 3)'. A large orange starburst graphic on the right says 'Welcome to a brand new empty session!'. A red arrow points from another callout box to the 'Terminal' icon at the bottom. The bottom navigation bar includes icons for Simple, Run, Kernel, Git, Tabs, Settings, Help, and a 'Launcher' tab.

File Edit View Run Kernel Git Tabs Settings Help

Launcher

Simple 0 0 0 0

+ Filter files by name / workspace /

Name Last Modified

As you can see, there is no code and no data

Welcome to a brand new empty session!

Notebook

Console

saturn (Python 3)

Other

Terminal Text File Markdown File Python File Show Contextual Help

# Lets download the code

```
        this._config.interval = this._config.interval || 1000;
    }

    var transitionDuration = 0;
    $(activeElement).one(Util.Transition.end, function() {
        $(nextElement).removeClass(ClassNames.ACTIVE);
        $(activeElement).removeClass(ClassNames.ACTIVE);
        _this4._isSliding = false;
        setTimeout(function () {
            return $_.this4._element.trigger('slideend');
        }, 0);
    }).emulateTransitionEnd(transitionDuration);
} else {
    $(activeElement).removeClass(ClassNames.ACTIVE);
    $(nextElement).addClass(ClassNames.ACTIVE);
}
```

In the terminal, type

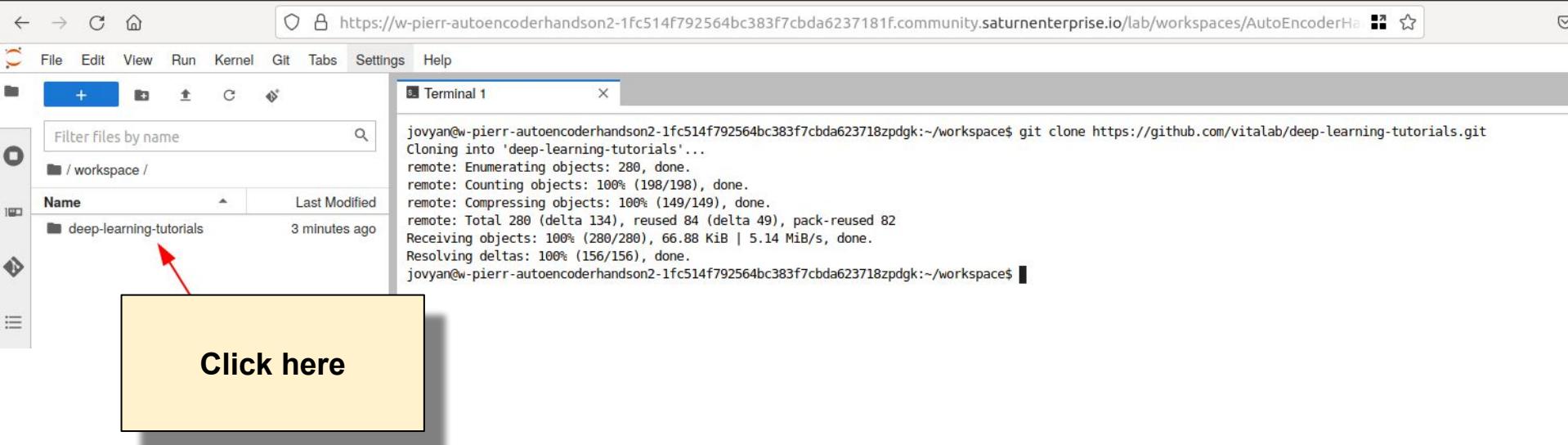
```
git clone https://github.com/vitalab/deep-learning-tutorials.git
```

The screenshot shows a Jupyter Notebook interface. On the left is a file browser pane with a sidebar containing icons for files, notebooks, and other workspace items. The main area has a terminal window titled "Terminal 1". The terminal output shows a command being run:

```
jovyan@w-pierr-autoencoderhandson2-1fc514f792564bc383f7cbda6237181f:~/workspace$ git clone https://github.com/vitalab/deep-learning-tutorials.git
```

A red box highlights the command in the terminal.

This command allows to **download the code** from a public GitHub repository



The screenshot shows a Jupyter Notebook interface with a terminal window and a file browser.

**File Browser:** On the left, there is a sidebar with icons for file operations like creating, deleting, and cloning. Below it is a search bar labeled "Filter files by name". Under "workspace /", there is a table with two rows:

Name	Last Modified
deep-learning-tutorials	3 minutes ago

A red arrow points to the "deep-learning-tutorials" entry in the file browser.

**Terminal Window:** On the right, titled "Terminal 1", the output of the following command is shown:

```
jovyan@w-pierr-autoencoderhandson2-1fc514f792564bc383f7cbda623718zpdgk:~/workspace$ git clone https://github.com/vitalab/deep-learning-tutorials.git
Cloning into 'deep-learning-tutorials'...
remote: Enumerating objects: 280, done.
remote: Counting objects: 100% (198/198), done.
remote: Compressing objects: 100% (149/149), done.
remote: Total 280 (delta 134), reused 84 (delta 49), pack-reused 82
Receiving objects: 100% (280/280), 66.88 KiB | 5.14 MiB/s, done.
Resolving deltas: 100% (156/156), done.
jovyan@w-pierr-autoencoderhandson2-1fc514f792564bc383f7cbda623718zpdgk:~/workspace$
```

A yellow box with the text "Click here" is overlaid on the left side of the terminal window.

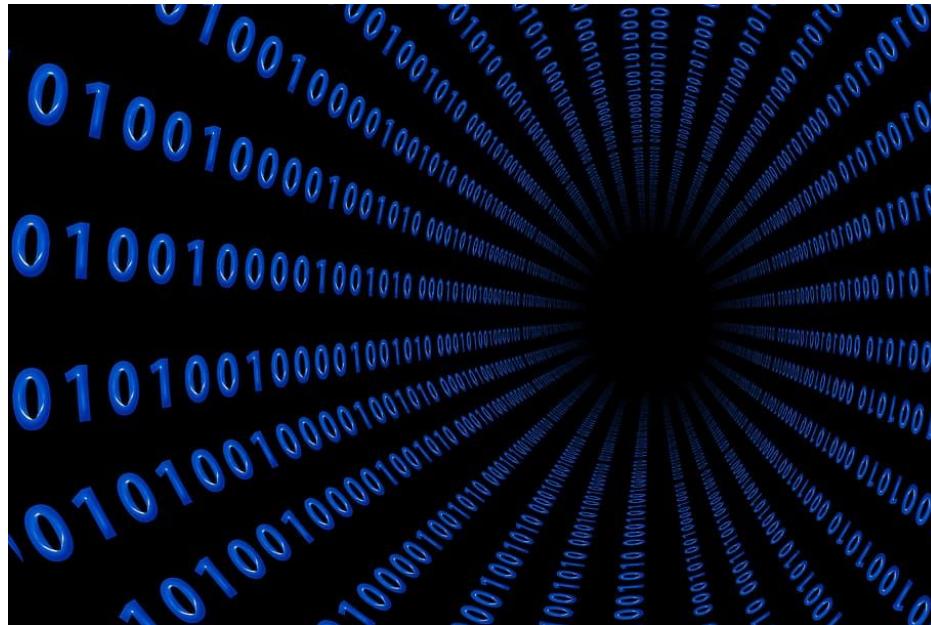
The screenshot shows a Jupyter Notebook interface with the following elements:

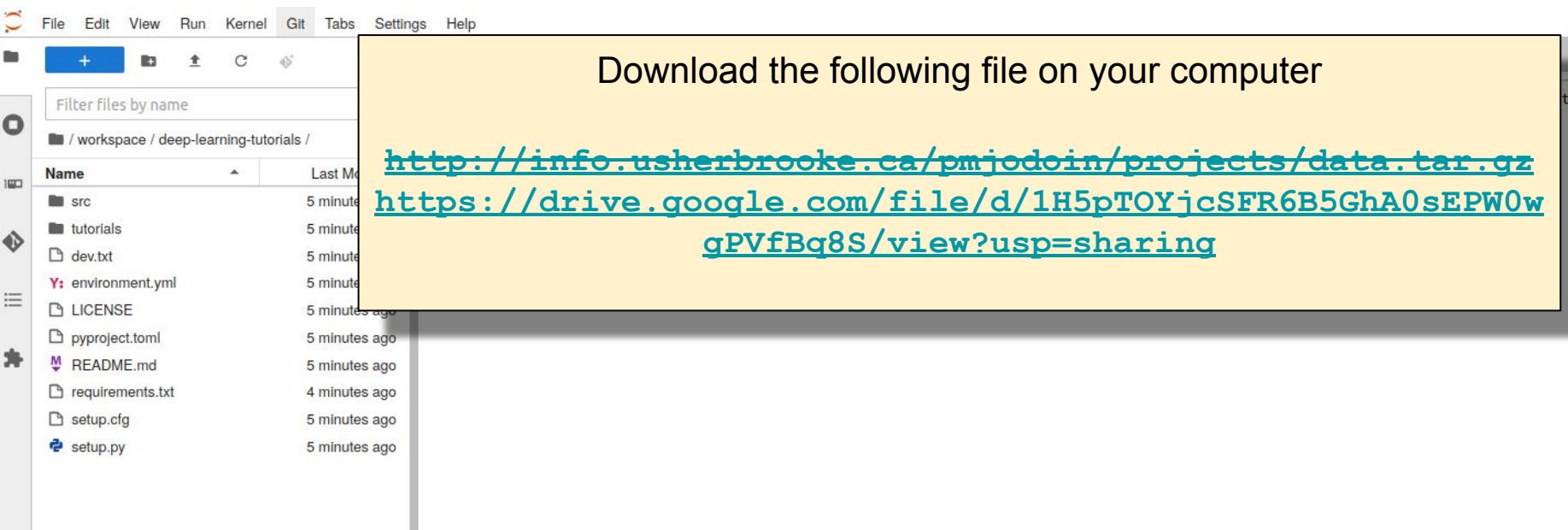
- File Bar:** File, Edit, View, Run, Kernel, Git, Tabs, Settings, Help.
- File Browser:** A sidebar on the left showing the directory structure: / workspace / deep-learning-tutorials /. The contents are listed below, sorted by Name (with Last Modified as the secondary key).

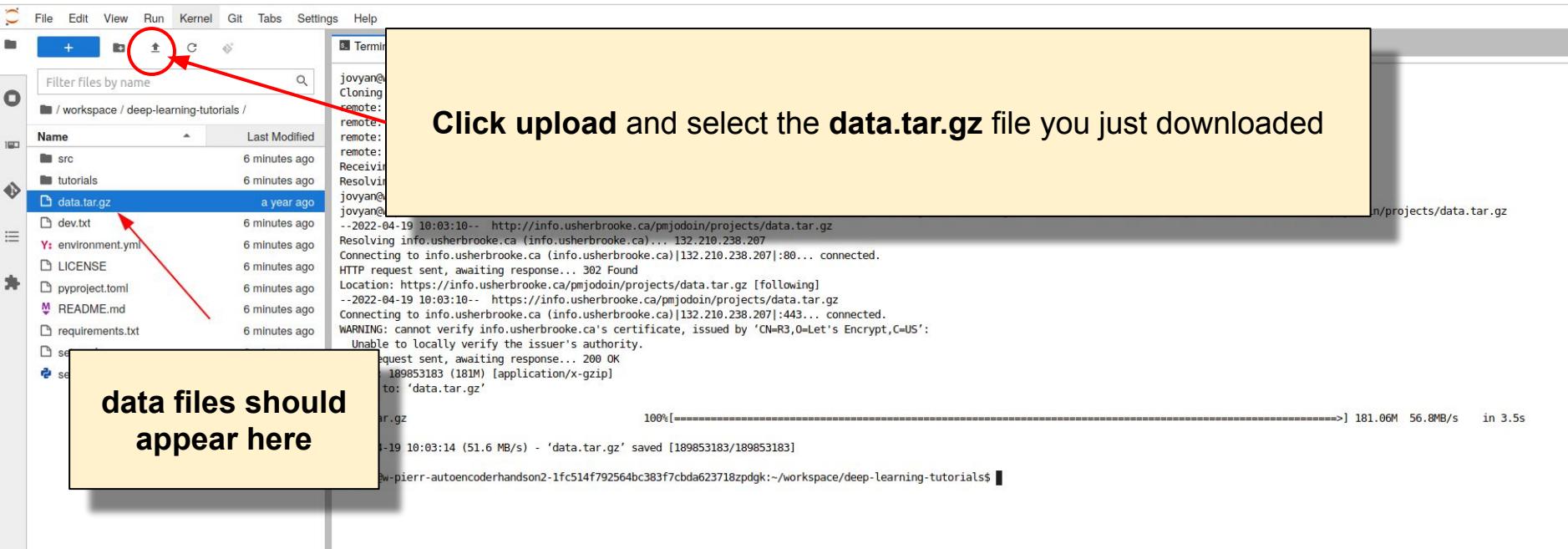
Name	Last Modified
src	5 minutes ago
tutorials	5 minutes ago
dev.txt	5 minutes ago
Y: environment.yml	5 minutes ago
LICENSE	5 minutes ago
pyproject.toml	5 minutes ago
M README.md	5 minutes ago
requirements.txt	4 minutes ago
setup.cfg	5 minutes ago
P setup.py	5 minutes ago
- Terminal:** A tab labeled "Terminal 1" showing the output of a git clone command:

```
jovyan@w-pierr-autoencoderhandson2-1fc514f792564bc383f7cbda623718zpdgk:~/workspace$ git clone https://github.com/vitalab/deep-learning-tutorials.git
Cloning into 'deep-learning-tutorials'...
remote: Enumerating objects: 280, done.
remote: Counting objects: 100% (198/198), done.
remote: Compressing objects: 100% (149/149), done.
remote: Total 280 (delta 134), reused 84 (delta 49), pack-reused 82
Receiving objects: 100% (280/280), 66.88 KiB | 5.14 MiB/s, done.
Resolving deltas: 100% (156/156), done.
jovyan@w-pierr-autoencoderhandson2-1fc514f792564bc383f7cbda623718zpdgk:~/workspace$
```
- Text Overlay:** A yellow rectangular box with a black border and shadow contains the text "Great! The code is there!".

# Lets download the data







File Edit View Run Kernel Git Tabs Settings Help

Terminal 1 requirements.txt

```
jovyan@w-pierr-autoencoderhandson2-1fc514f792564bc383f7cbda623718zpdgk:~/workspace$ git clone https://github.com/vitalab/deep-learning-tutorials.git
Cloning into 'deep-learning-tutorials'...
remote: Enumerating objects: 280, done.
remote: Counting objects: 100% (198/198), done.
remote: Compressing objects: 100% (149/149), done.
remote: Total 280 (delta 134), reused 84 (delta 49), pack-reused 82
Receiving objects: 100% (280/280), 66
Resolving deltas: 100% (156/156), don
jovyan@w-pierr-autoencoderhandson2-1fc514f792564bc383f7cbda623718zpdgk:~/workspace$ cd deep-learning-tutorials
jovyan@w-pierr-autoencoderhandson2-1fc514f792564bc383f7cbda623718zpdgk:~/workspace/deep-learning-tutorials$ ls
data  src  tutorials  data.tar.gz  dev.txt  environment.yml  LICENSE  pyproject.toml  README.md  requirements.txt  setup.cfg  setup.py
jovyan@w-pierr-autoencoderhandson2-1fc514f792564bc383f7cbda623718zpdgk:~/workspace/deep-learning-tutorials$ cd deep-learning-tutorials
jovyan@w-pierr-autoencoderhandson2-1fc514f792564bc383f7cbda623718zpdgk:~/workspace/deep-learning-tutorials$ tar -xvzf data.tar.gz
jovyan@w-pierr-autoencoderhandson2-1fc514f792564bc383f7cbda623718zpdgk:~/workspace/deep-learning-tutorials$ ls
data/MNIST/raw/
data/MNIST/raw/train-labels-idx1-ubyte
data/MNIST/raw/train-images-idx3-ubyte.gz
data/MNIST/raw/train-labels-idx1-ubyte.gz
data/MNIST/raw/t10k-labels-idx1-ubyte
data/MNIST/raw/t10k-images-idx3-ubyte.gz
data/MNIST/raw/t10k-labels-idx1-ubyte
data/MNIST/raw/train-images-idx3-ubyte
data/MNIST/raw/t10k-labels-idx1-ubyte.gz
data/MNIST/processed/
data/MNIST/processed/test.pt
data/MNIST/processed/training.pt
jovyan@w-pierr-autoencoderhandson2-1fc514f792564bc383f7cbda623718zpdgk:~/workspace/deep-learning-tutorials$
```

Then in the terminal, type

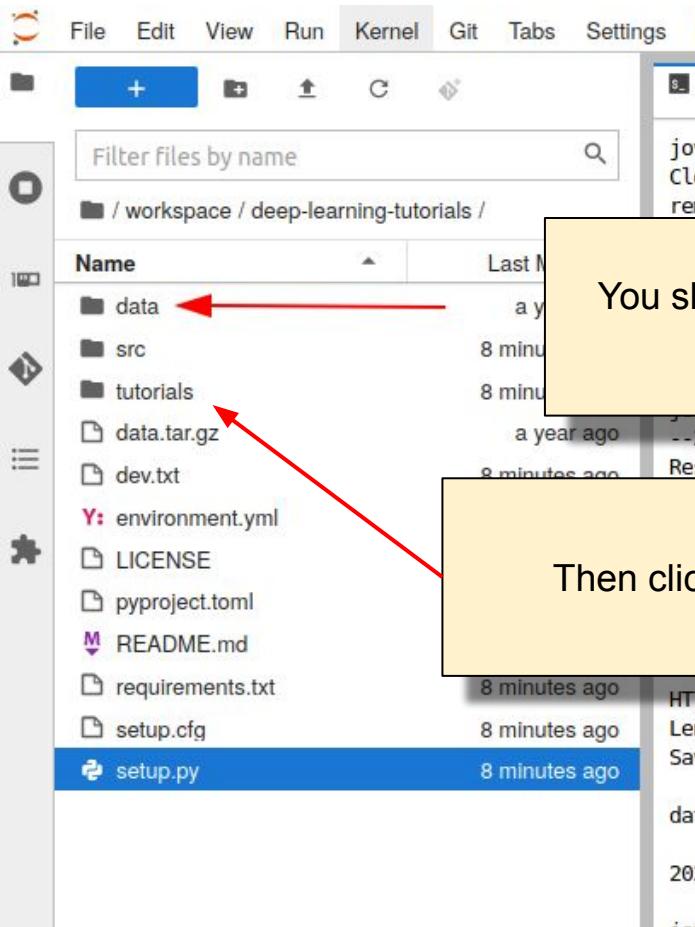
**cd deep-learning-tutorials**

followed by

**tar -xvzf data.tar.gz**

In/projects/data.tar.gz

1.06M 56.8MB/s in 3.5s



You should have the following  
**data folder**

Then click on **tutorials**

The screenshot shows a Jupyter Notebook environment with the following components:

- File Explorer:** On the left, it displays a file tree under "/ ... / deep-learning-tutorials / tutorials /". A red arrow points to the "mnist-autoencoders.ipynb" file, which is selected and highlighted in blue.
- Kernel Tab:** The top menu bar includes "Kernel" (highlighted in blue), indicating the current active kernel.
- Terminal:** A terminal window titled "Terminal 1" shows the command: `jovyan@w-pierr-autoencoderhandson2-1fc514f792564bc383f7cbda623718zpdgk:~/workspace$ git clone https://github.com/vit...`. Below this, several lines of log output are visible, starting with "Cloning into 'deep-learning-tutorials'..." and ending with "Saving to: 'data.tar.gz'".
- Code Cell:** A code cell contains the following text:

```
These are the two Jupyter notebooks for this tutorial.  
Now you are good to go!  
Start with mnist and then move on to cardiac-mri
```