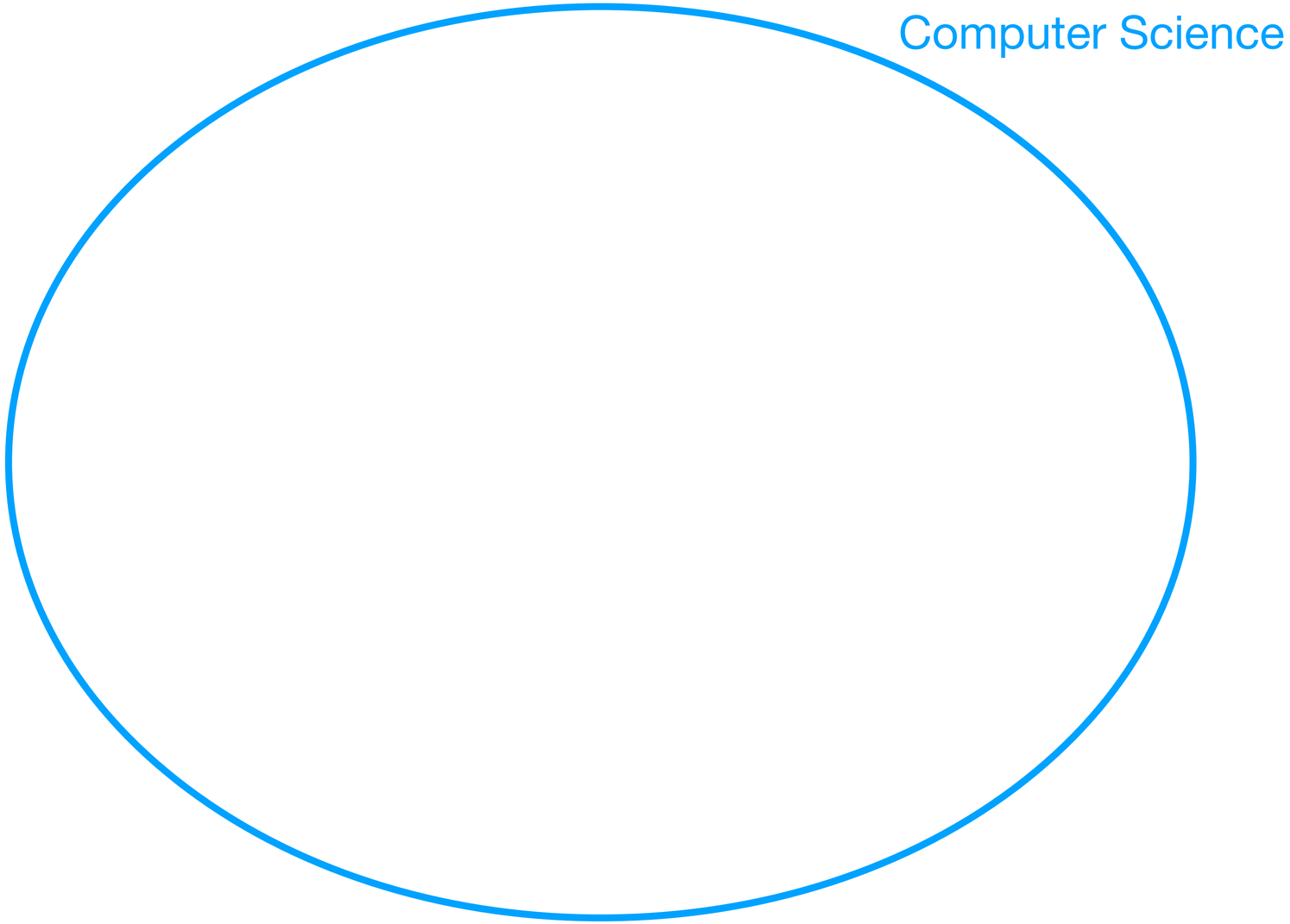
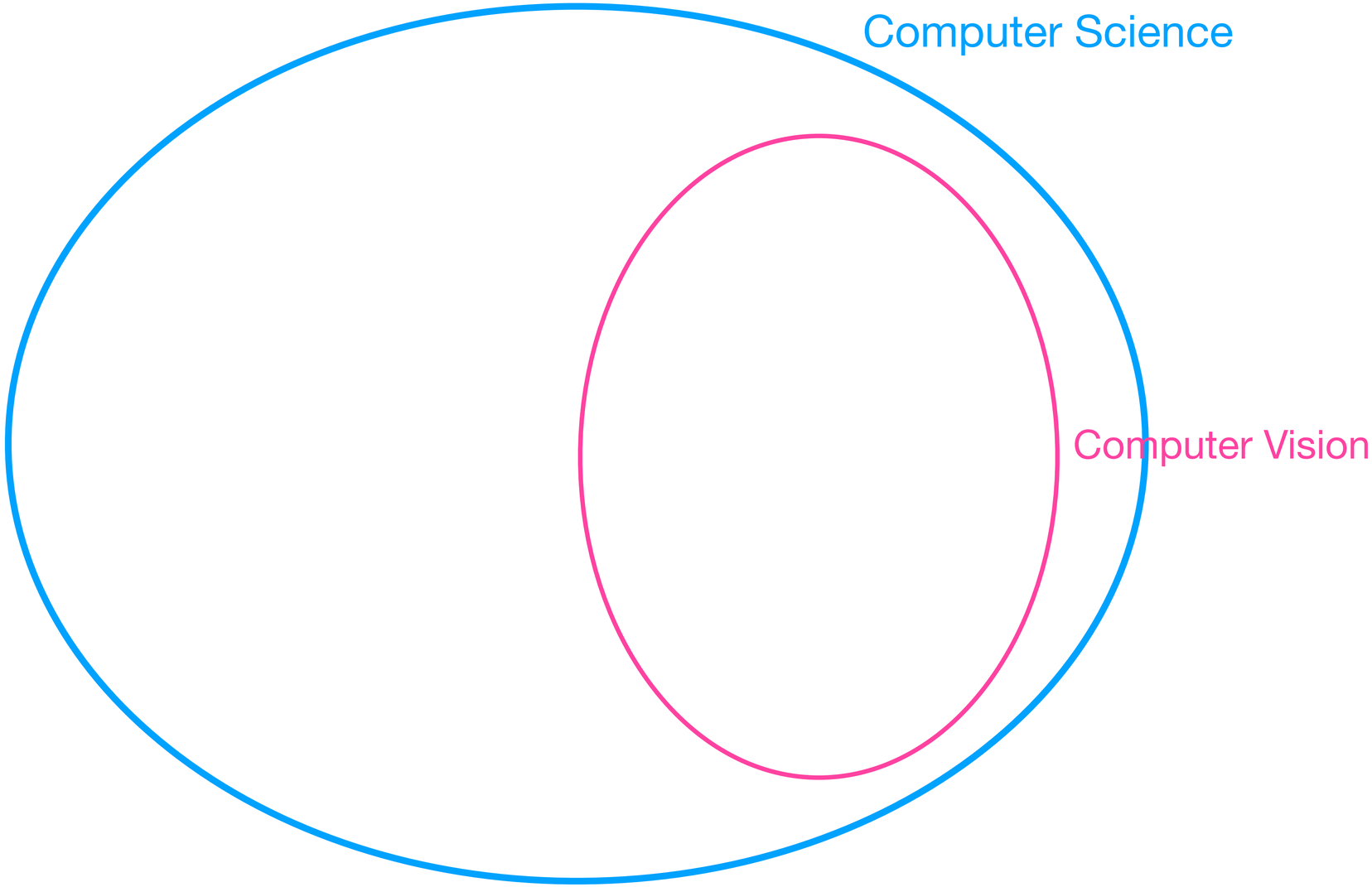
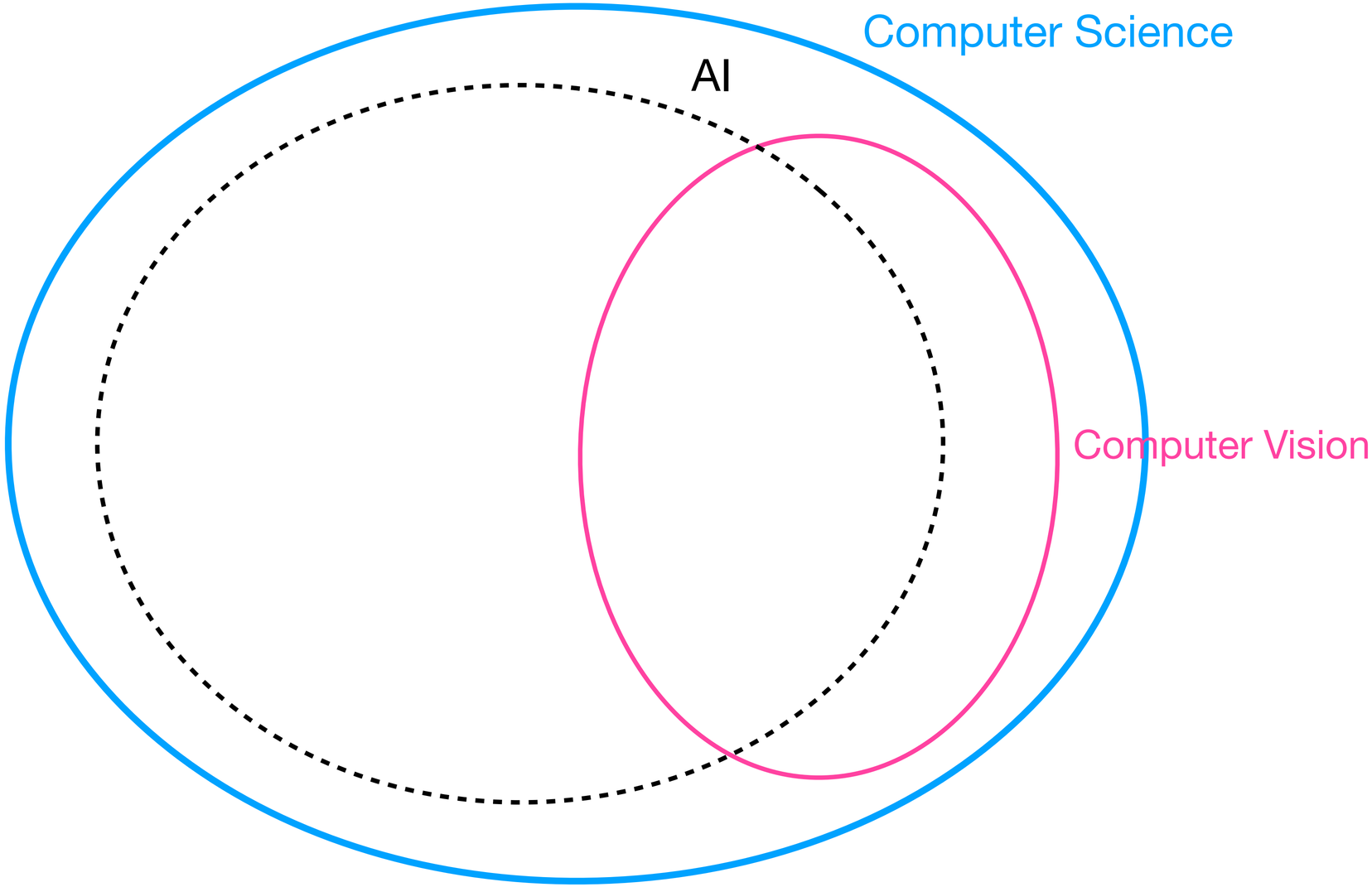
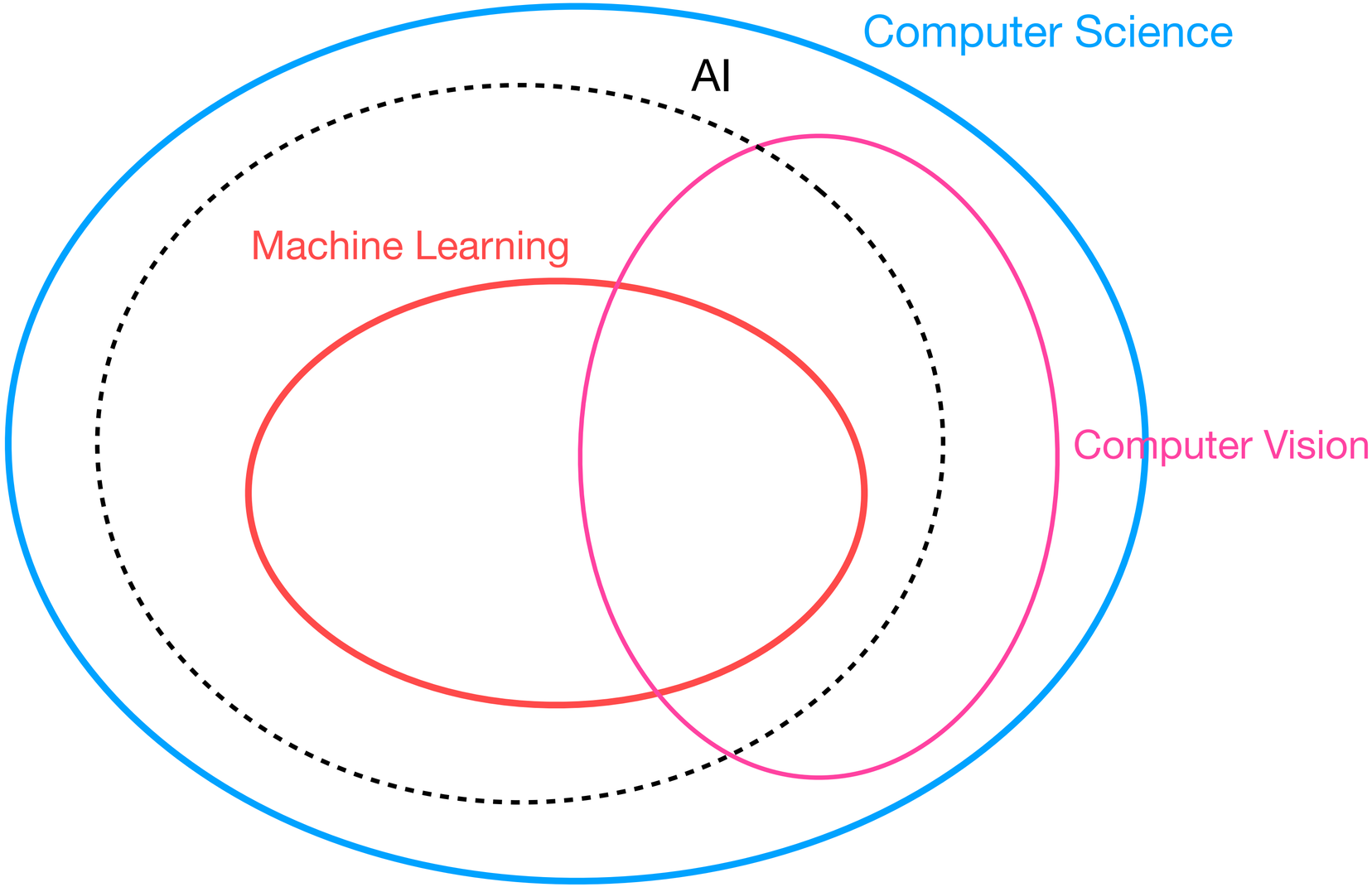
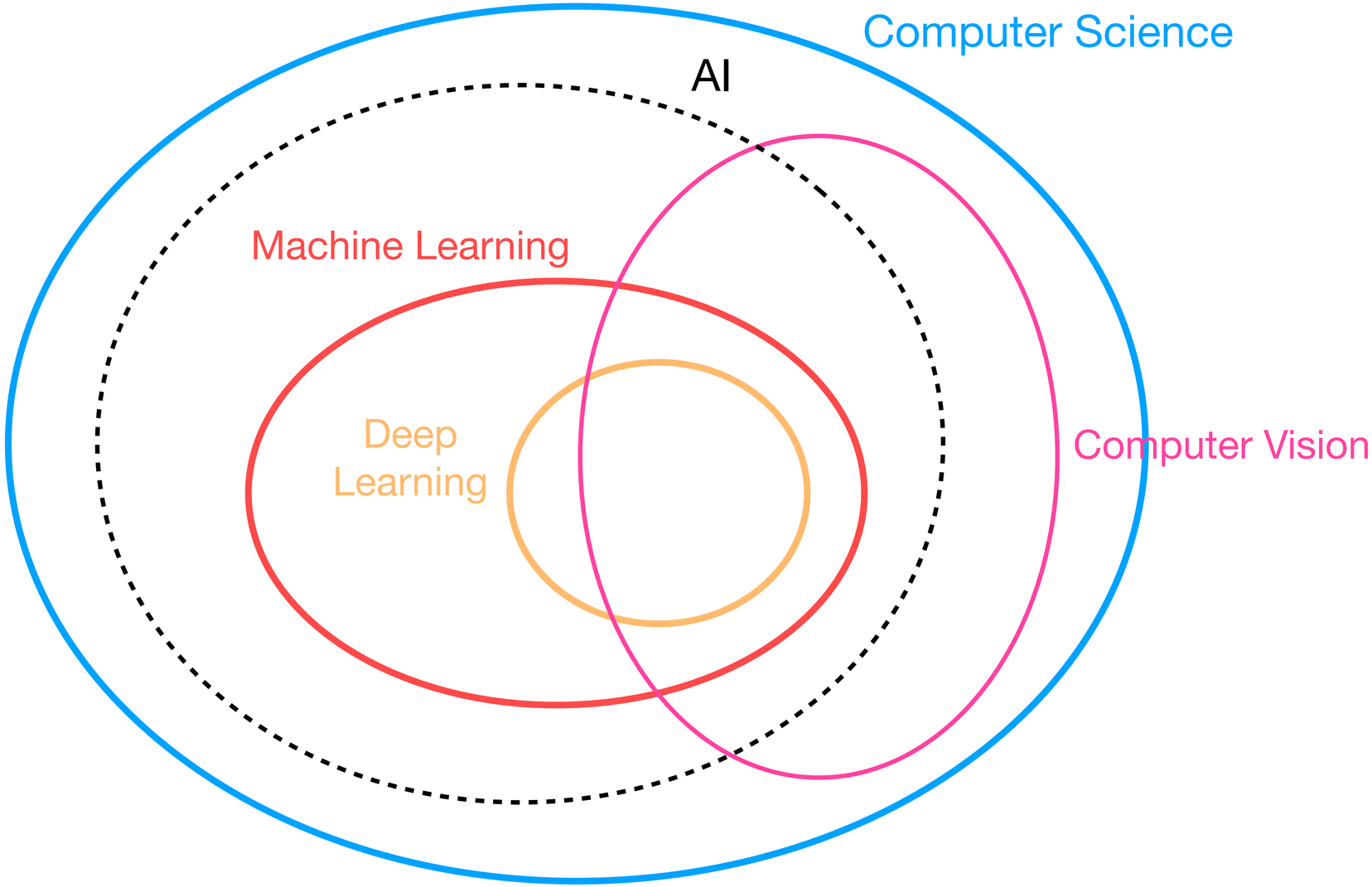
A (very) General Introduction to Deep Learning

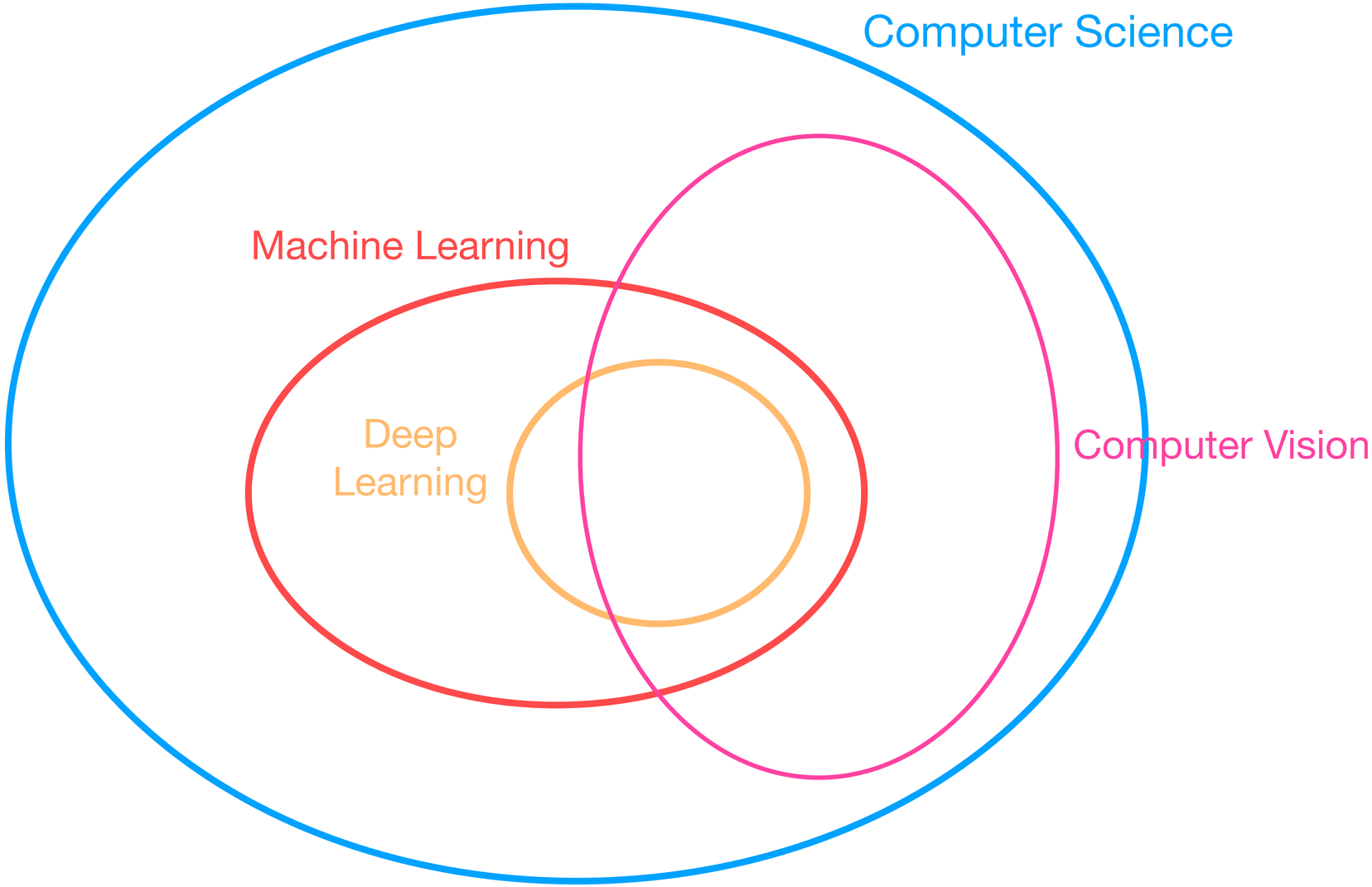

Computer Science


Computer Science
Computer Vision

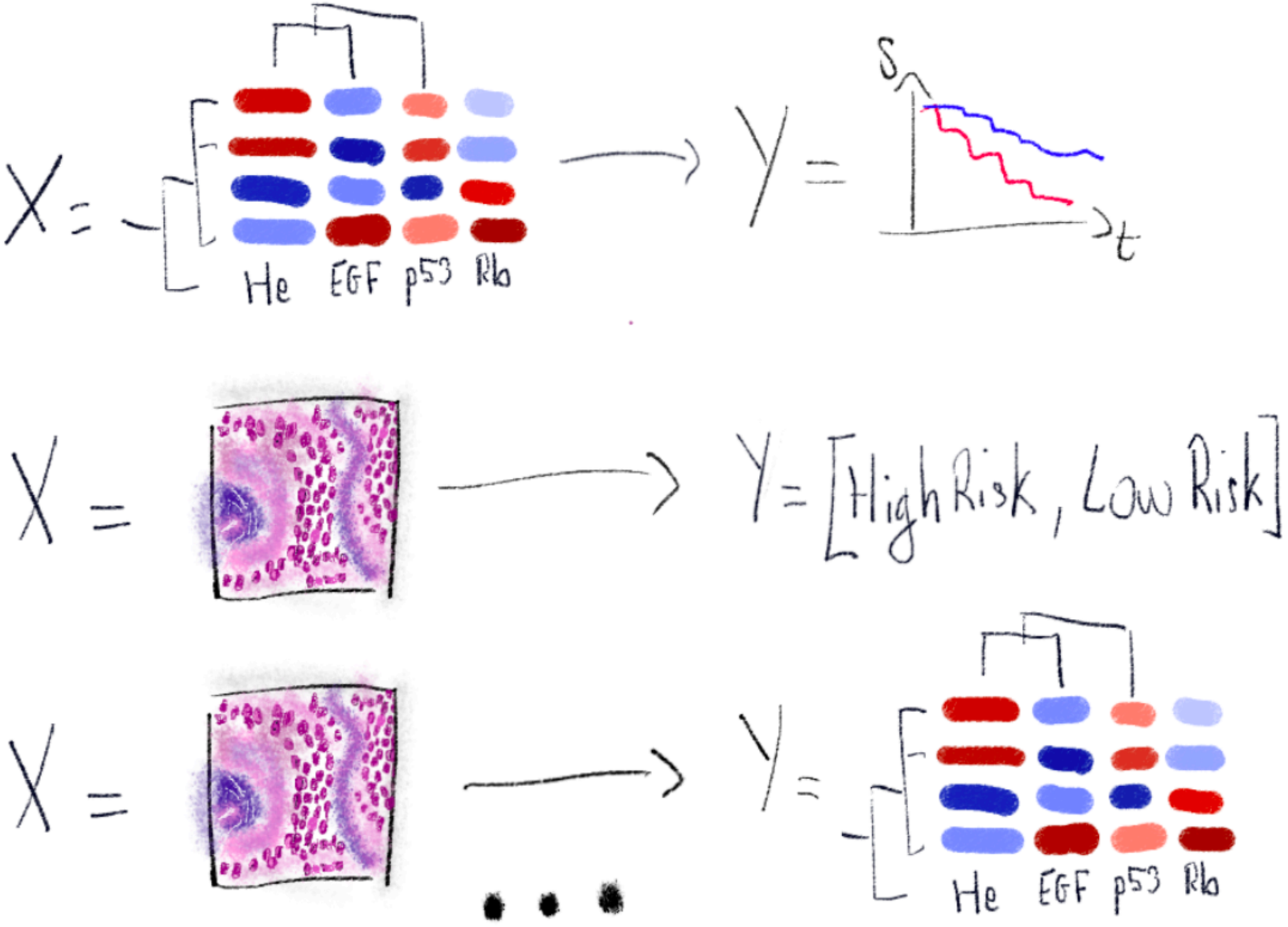

Computer Science
AI
Computer Vision


Computer Science
AI
Machine Learning
Computer Vision

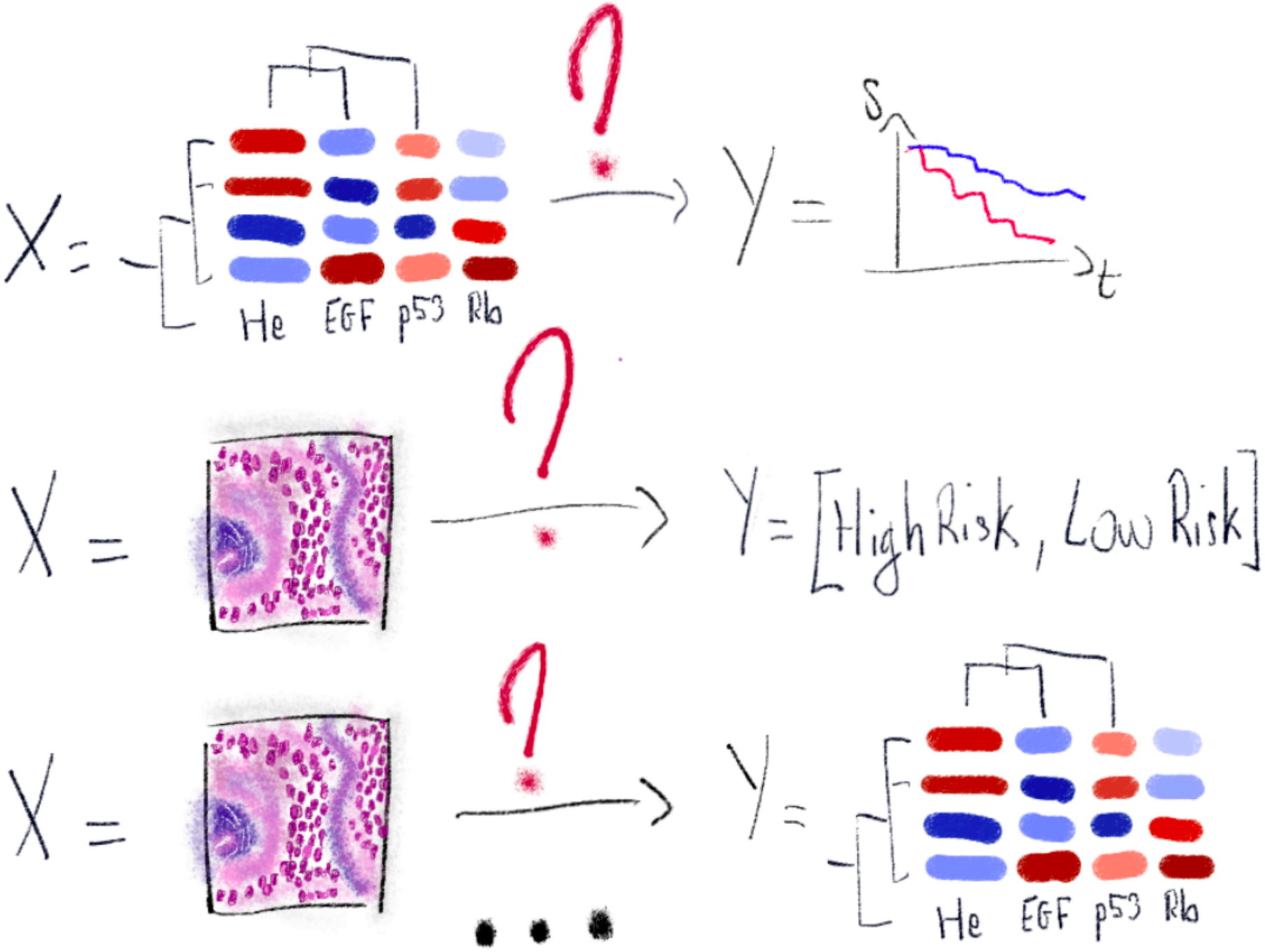

Computer Science
AI
Machine Learning
Deep
Learning
Computer Vision


Computer Science
Machine Learning
Deep
Learning
Computer Vision

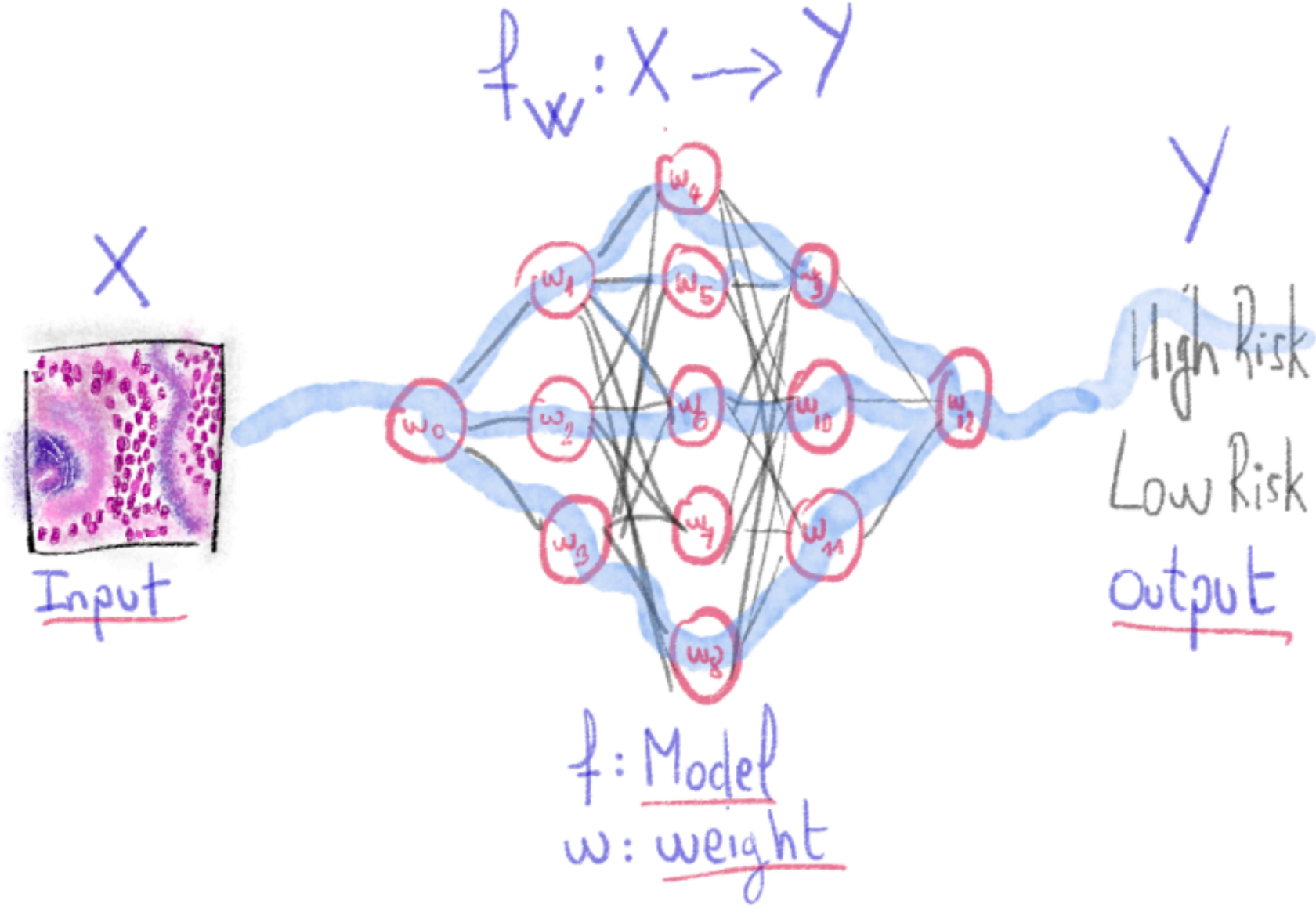
What do you have ? What do you want to do ?



How ?

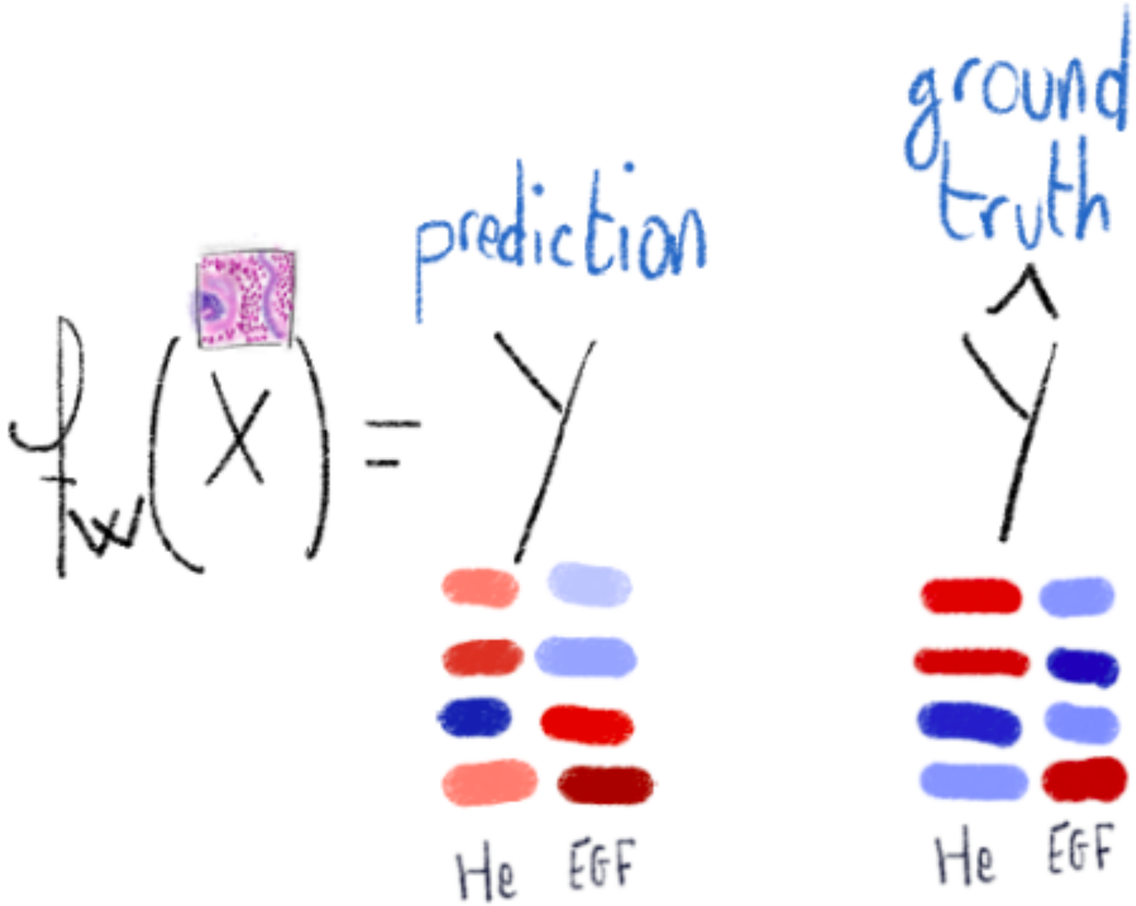


Define a model that you can train

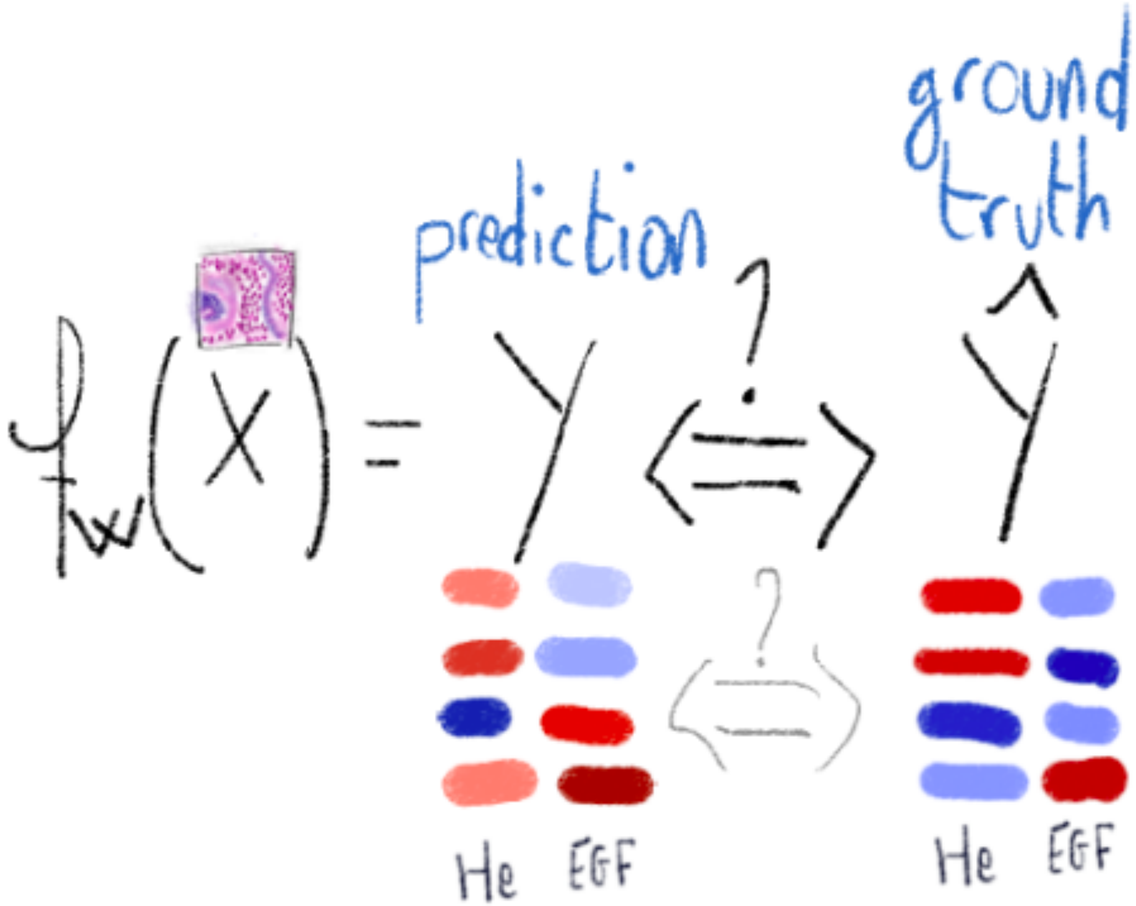


Training means that you need to have some known labels

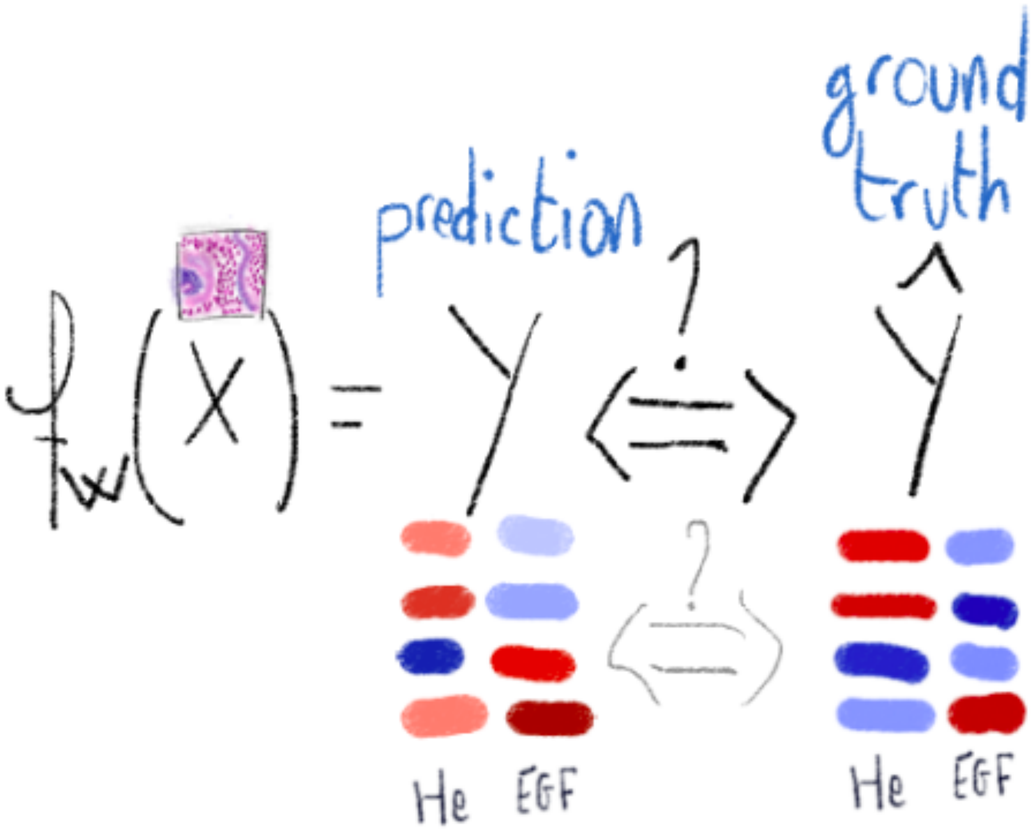
With wich you’re able to compare your model’s prediction



How ?

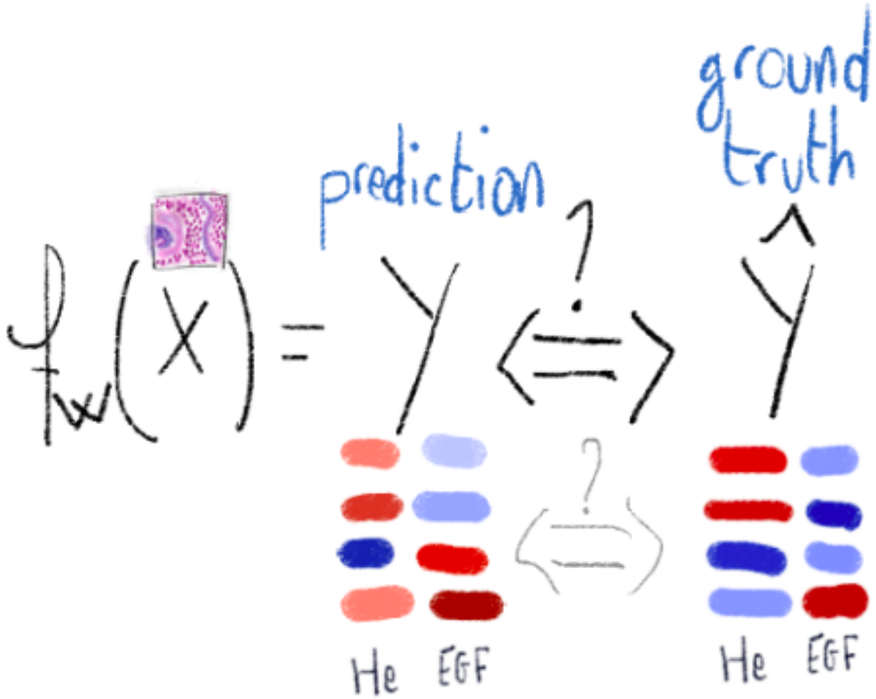


By defining an error function, or objective function, or **Loss**



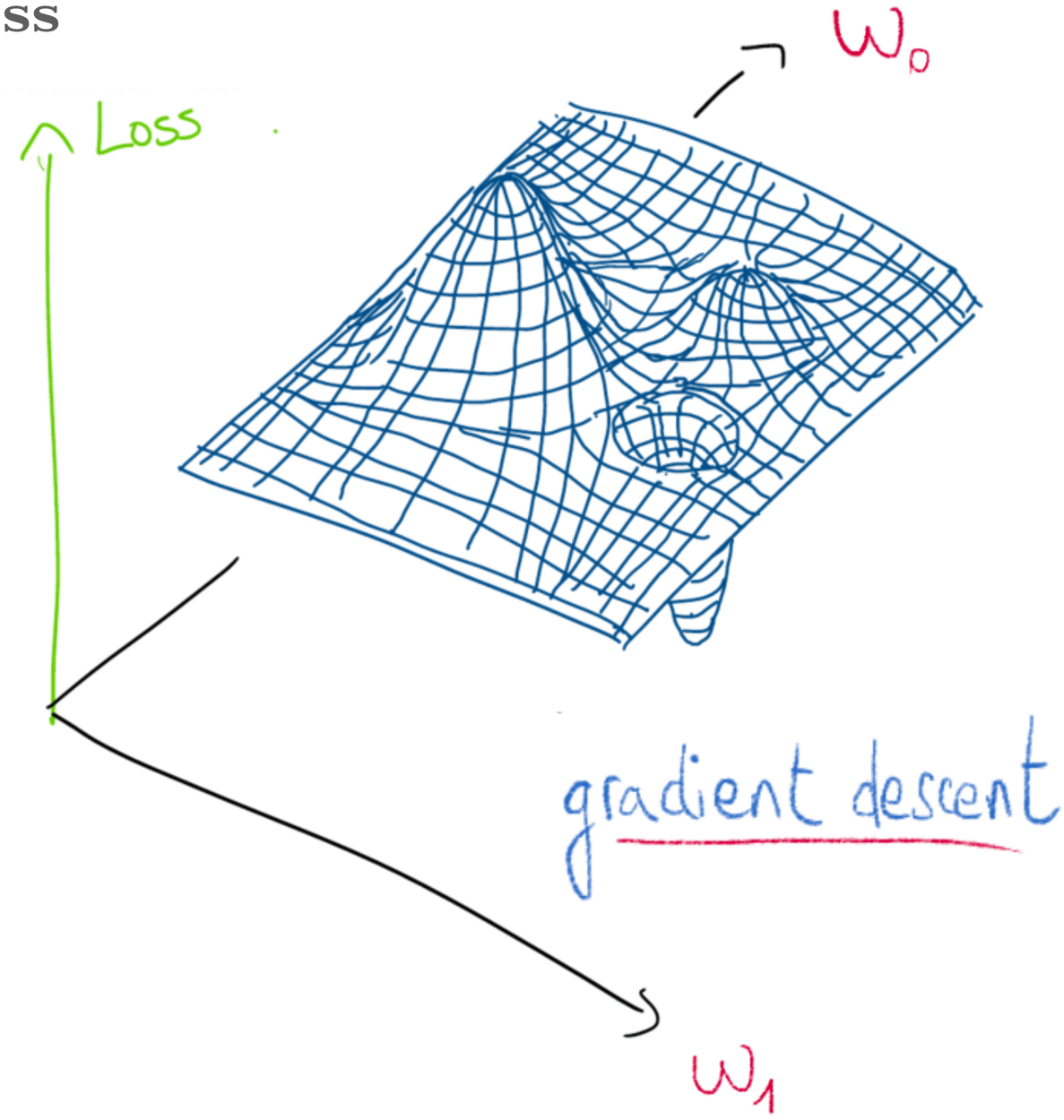
$$\underset{\text{Loss}}{\mathcal{L}} = |Y - \hat{Y}|$$

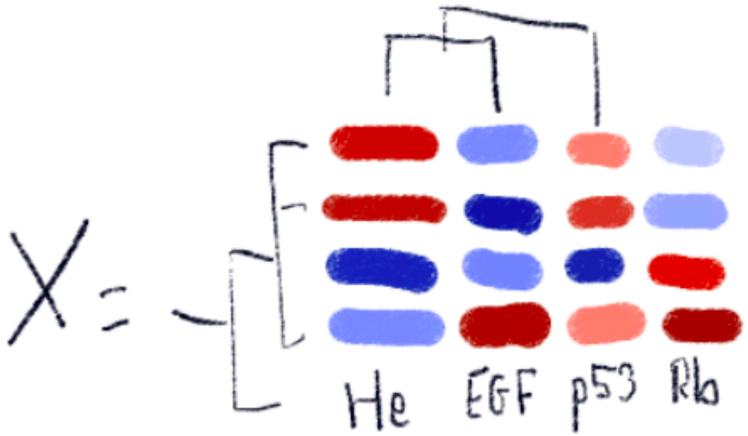
With this, you're going to be able to adapt your model such that it learns how to decreases the Loss



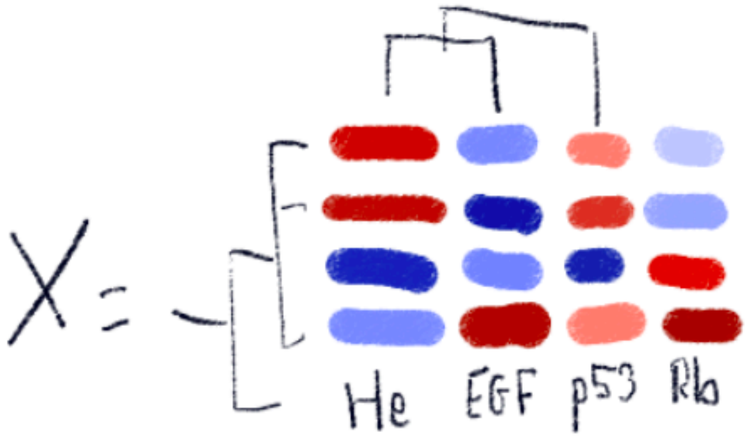
$$\mathcal{L} = |Y - \hat{Y}|$$

Loss

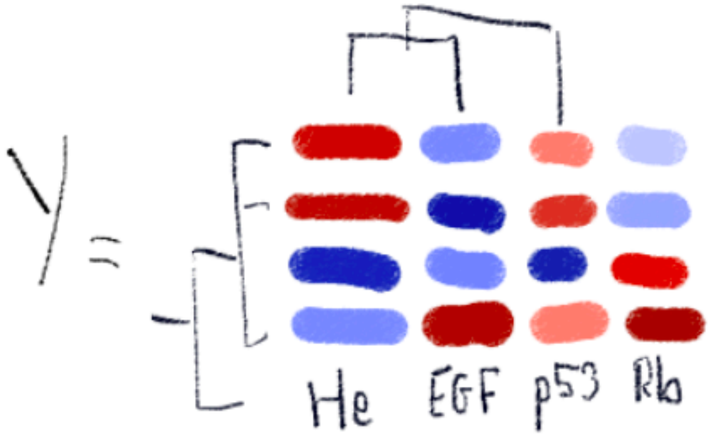




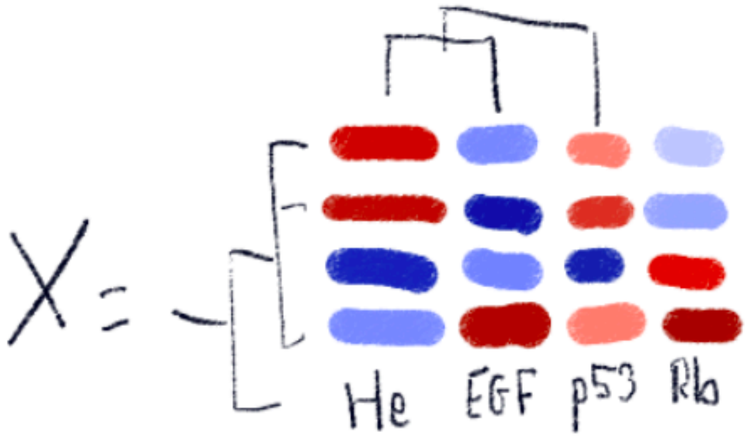
Input Data



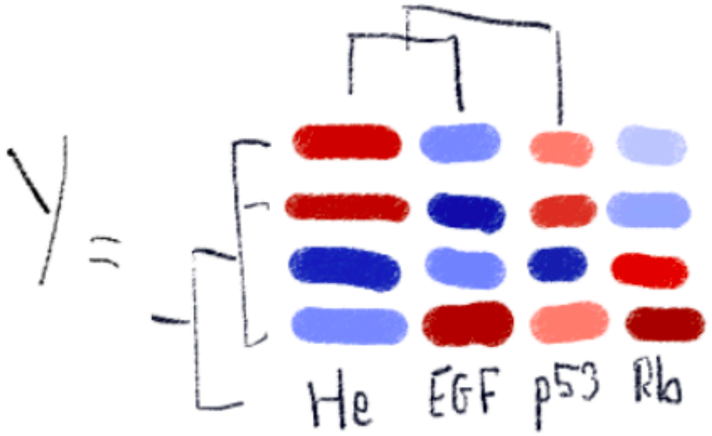
Input Data



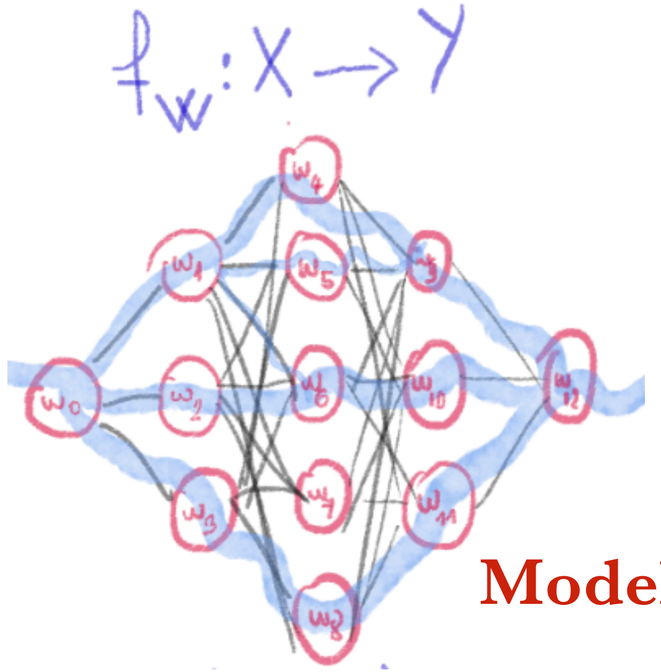
Ground Truth



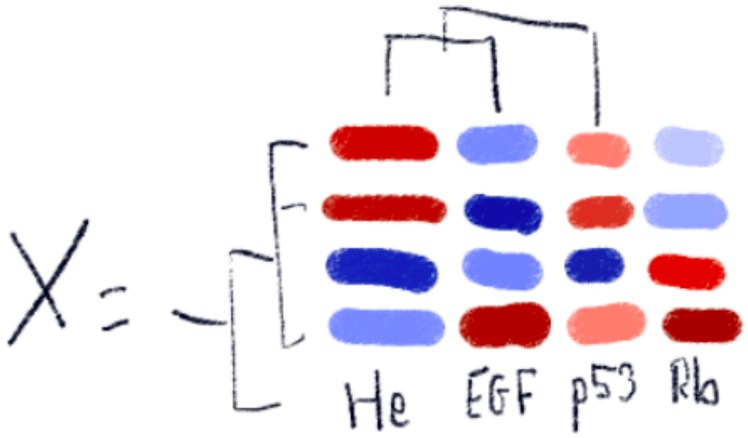
Input Data



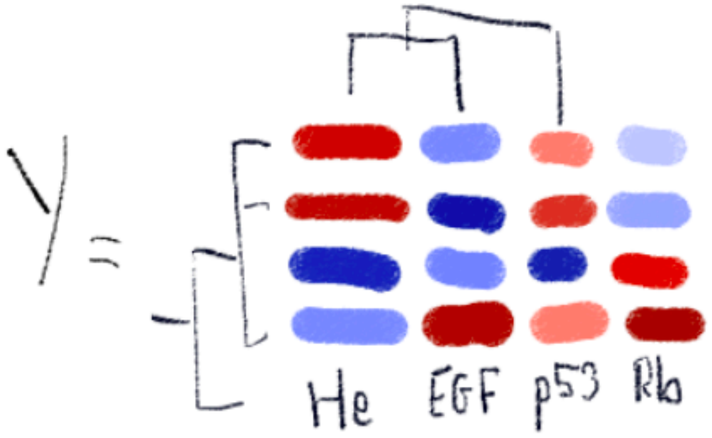
Ground Truth



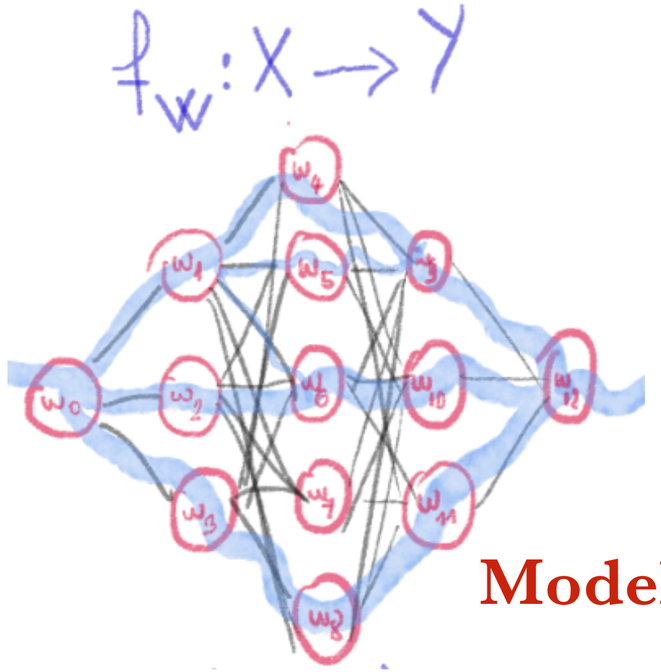
Model



Input Data



Ground Truth

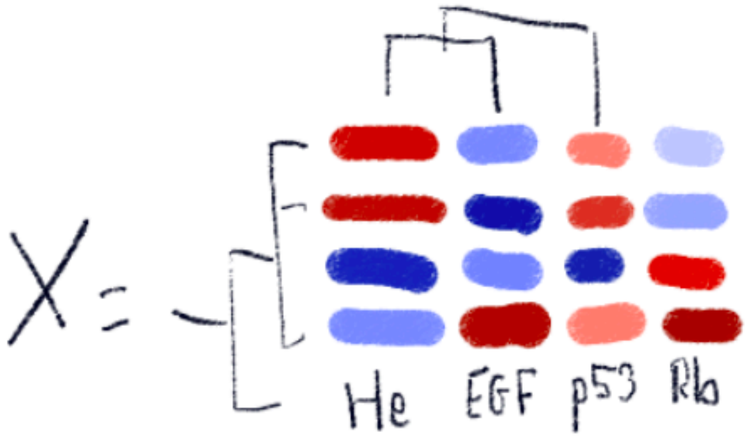


Model

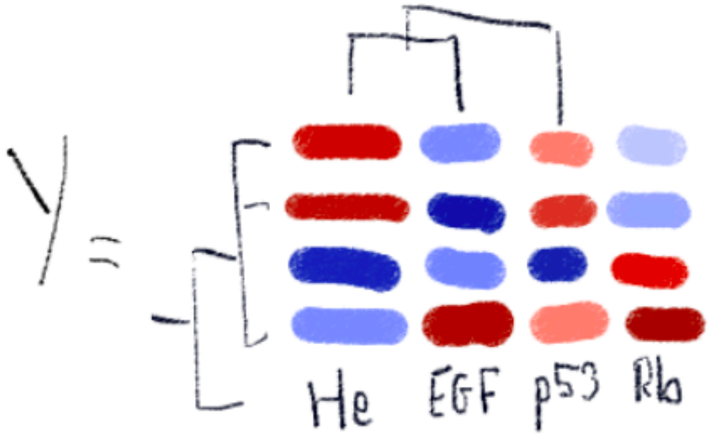
$$\mathcal{L} = |Y - \hat{Y}|$$

Loss

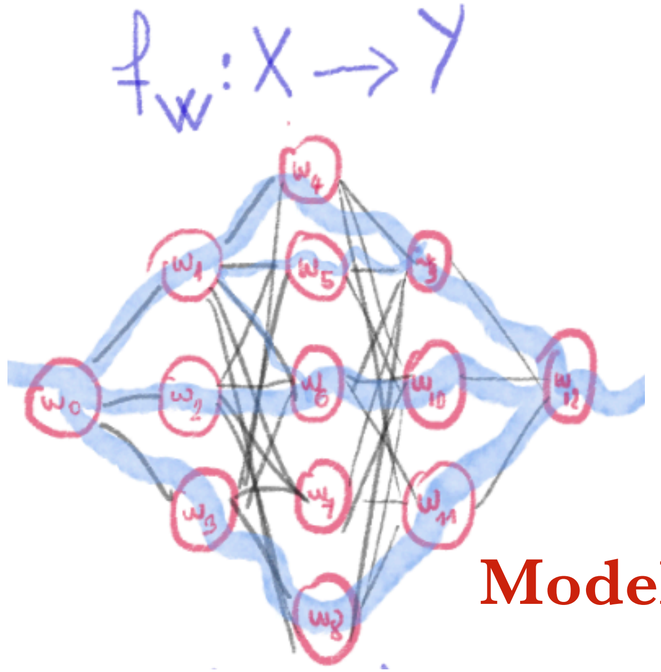
Loss Function



Input Data



Ground Truth

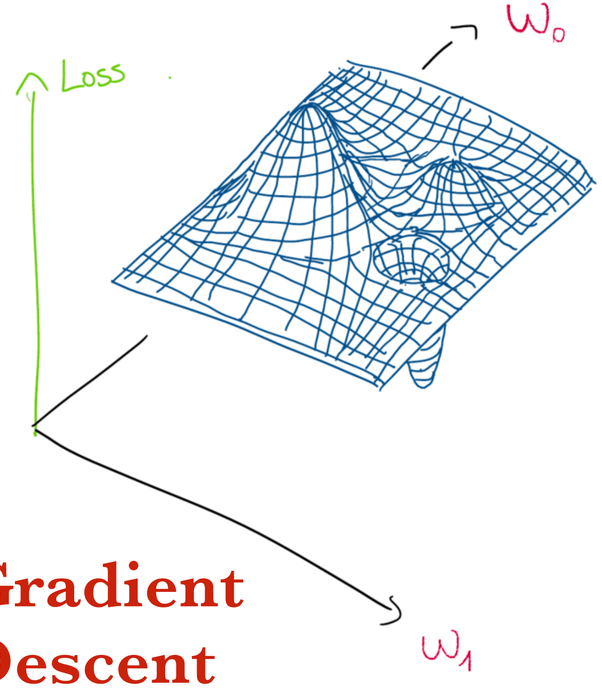


Model

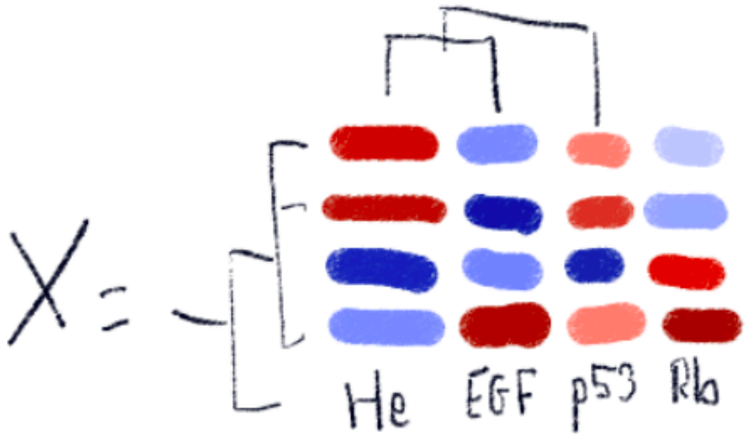
$$\mathcal{L} = |Y - \hat{Y}|$$

Loss

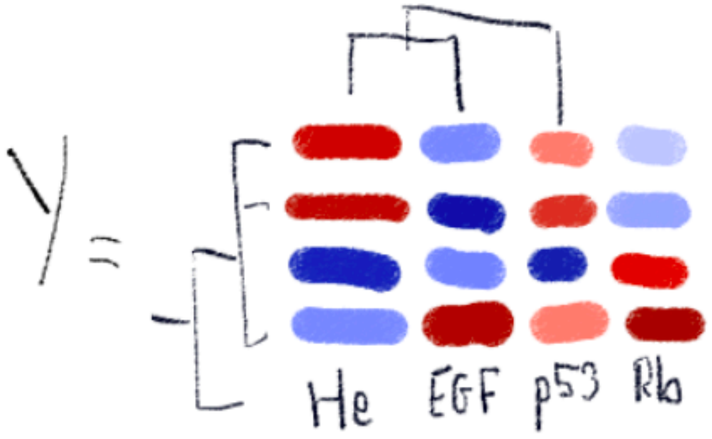
Loss Function



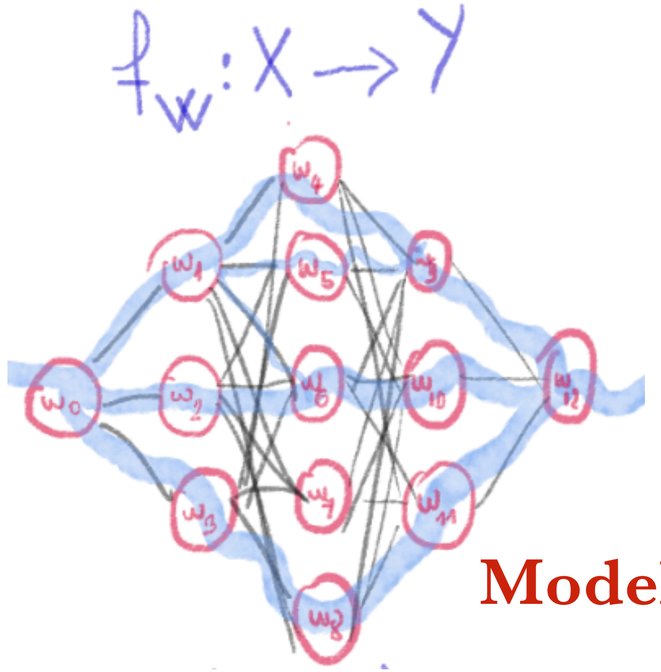
Gradient Descent



Input Data



Ground Truth

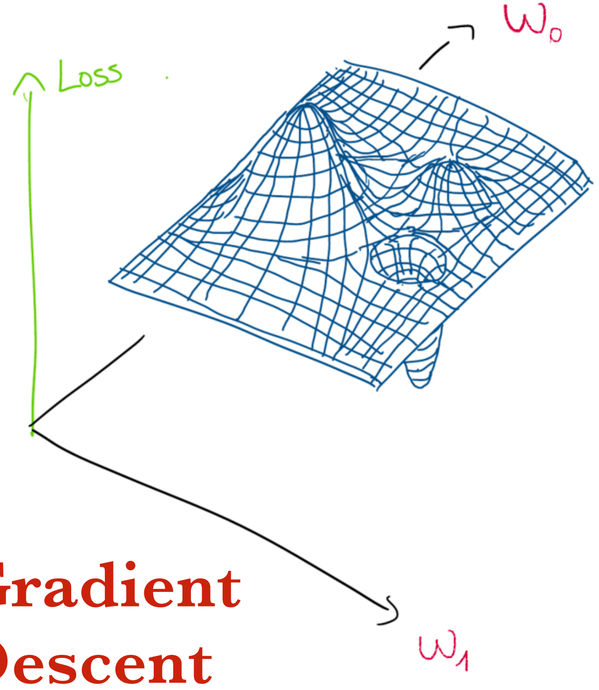


Model

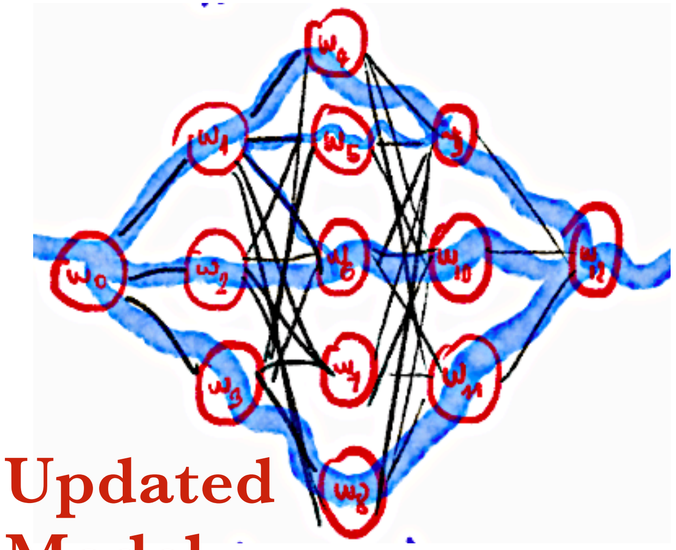
$$\mathcal{L} = |Y - \hat{Y}|$$

Loss

Loss Function



Gradient Descent



Updated Model

➡ Define your **dataset**, input X, ground truth/labels Y

➡ Define your **architecture/model**, $f_{\mathscr{W}} : x \rightarrow \hat{y}$

➡ Define the **loss** of your model: $\mathscr{L} = \hat{y} - y$

Hyper-parameters definition

➡ **Predict** a batch of data: $\hat{y} = f(x)$

➡ Compute the **loss** on the batch

➡ Use **stochastic gradient descent** to update your weights:

➡ Repeat until **convergence**

Training

➡ Test the **accuracy** on some test data

Testing

➡ Define your **dataset**, input X, ground truth/labels Y

➡ Define your **architecture/model**, $f_{\mathscr{W}} : x \rightarrow \hat{y}$

➡ Define the **loss** of your model: $\mathscr{L} = \hat{y} - y$

Hyper-parameters definition

➡ **Predict** a batch of data: $\hat{y} = f(x)$

➡ Compute the **loss** on the batch

➡ Use **stochastic gradient descent** to update your weights: $\mathscr{W}_{t+1} = \mathscr{W}_t + \lambda \frac{\partial \mathscr{L}}{\partial \mathscr{W}_t}$

➡ Repeat until **convergence**

Training

➡ Test the **accuracy** on some test data

Testing

➡ Define your **dataset**, input X, ground truth/labels Y

➡ Define your **architecture/model**, $f_{\mathscr{W}} : x \rightarrow \hat{y}$

➡ Define the **loss** of your model: $\mathscr{L} = \hat{y} - y$

Hyper-parameters definition

➡ **Predict** a batch of data: $\hat{y} = f(x)$

➡ Compute the **loss** on the batch

➡ Use **stochastic gradient descent** to update your weights:

➡ Repeat until **convergence**

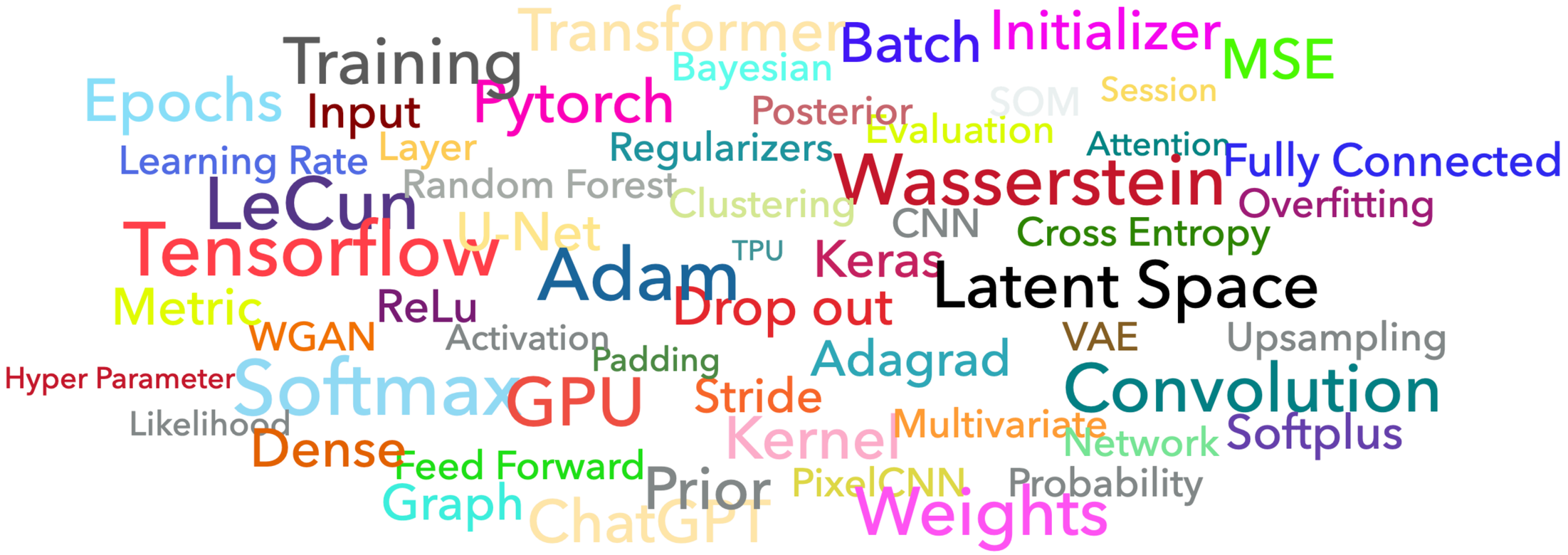
Training

➡ Test the **accuracy** on some test data

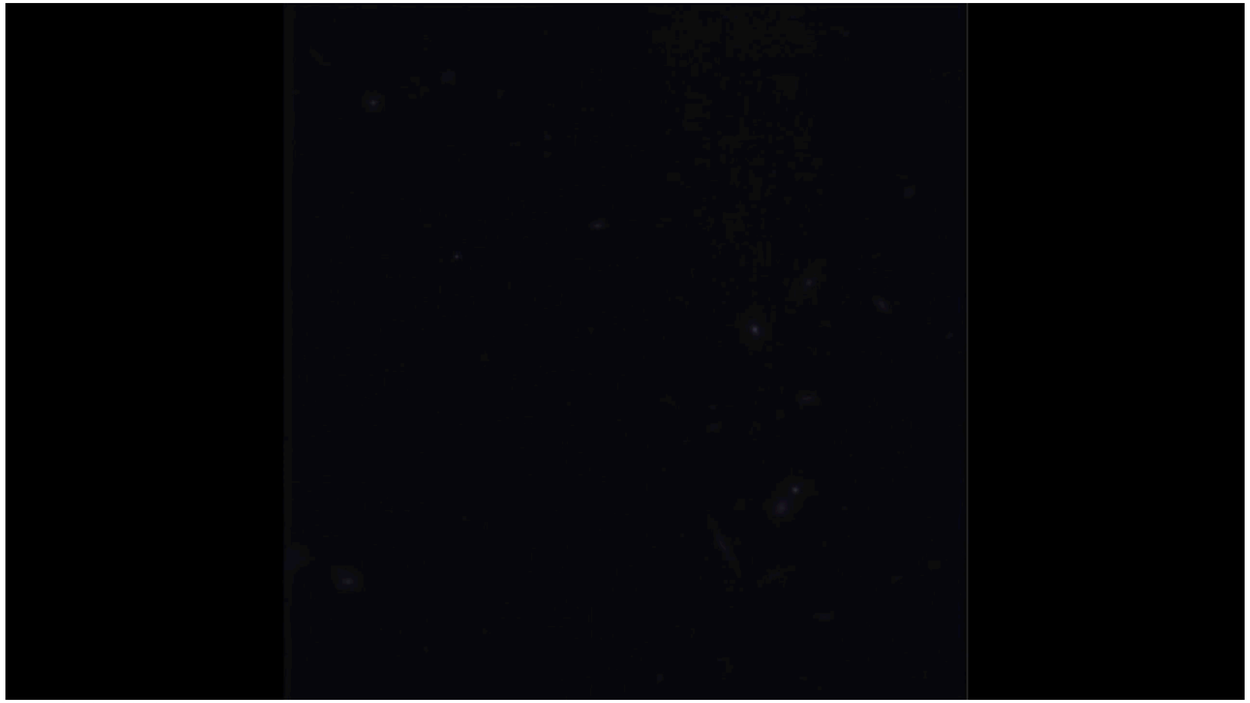
Testing

From here, you have basically all the steps to build models such as Chat-GPT…

What you'll have to do, is to increase the complexity of each of the steps. Let's explore some of them.

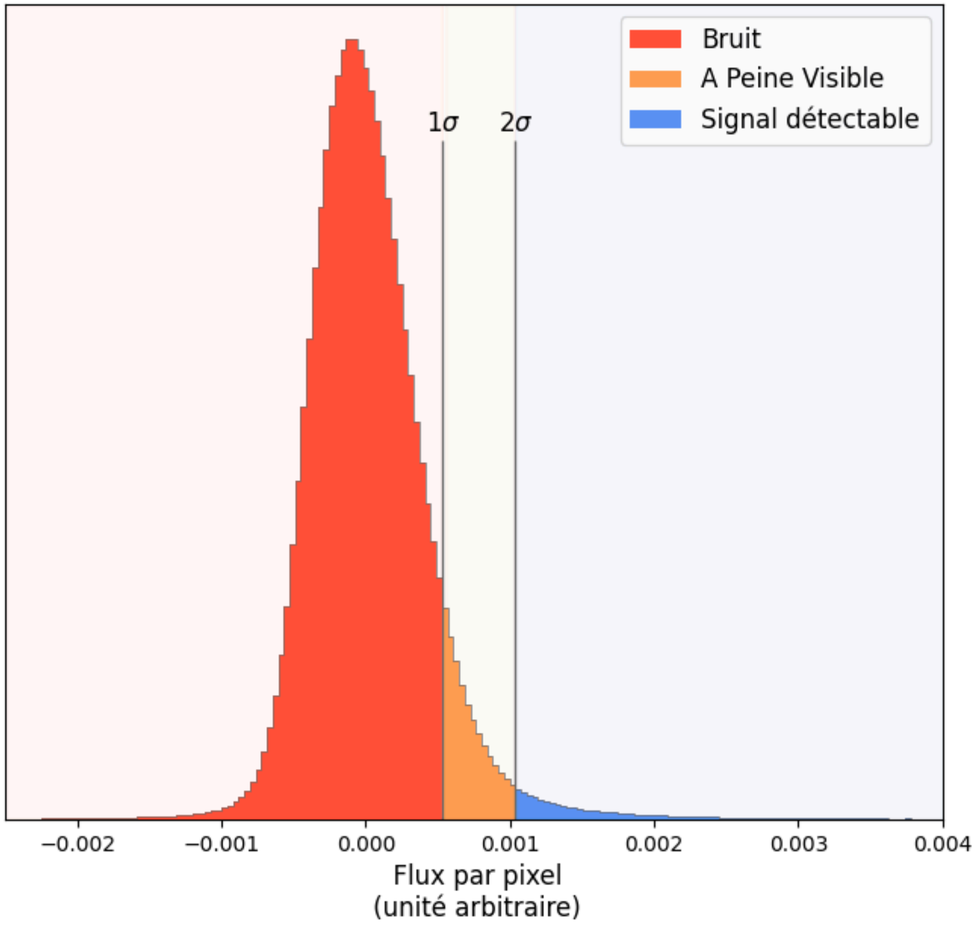


1- Your Data



sorry for the galaxies…

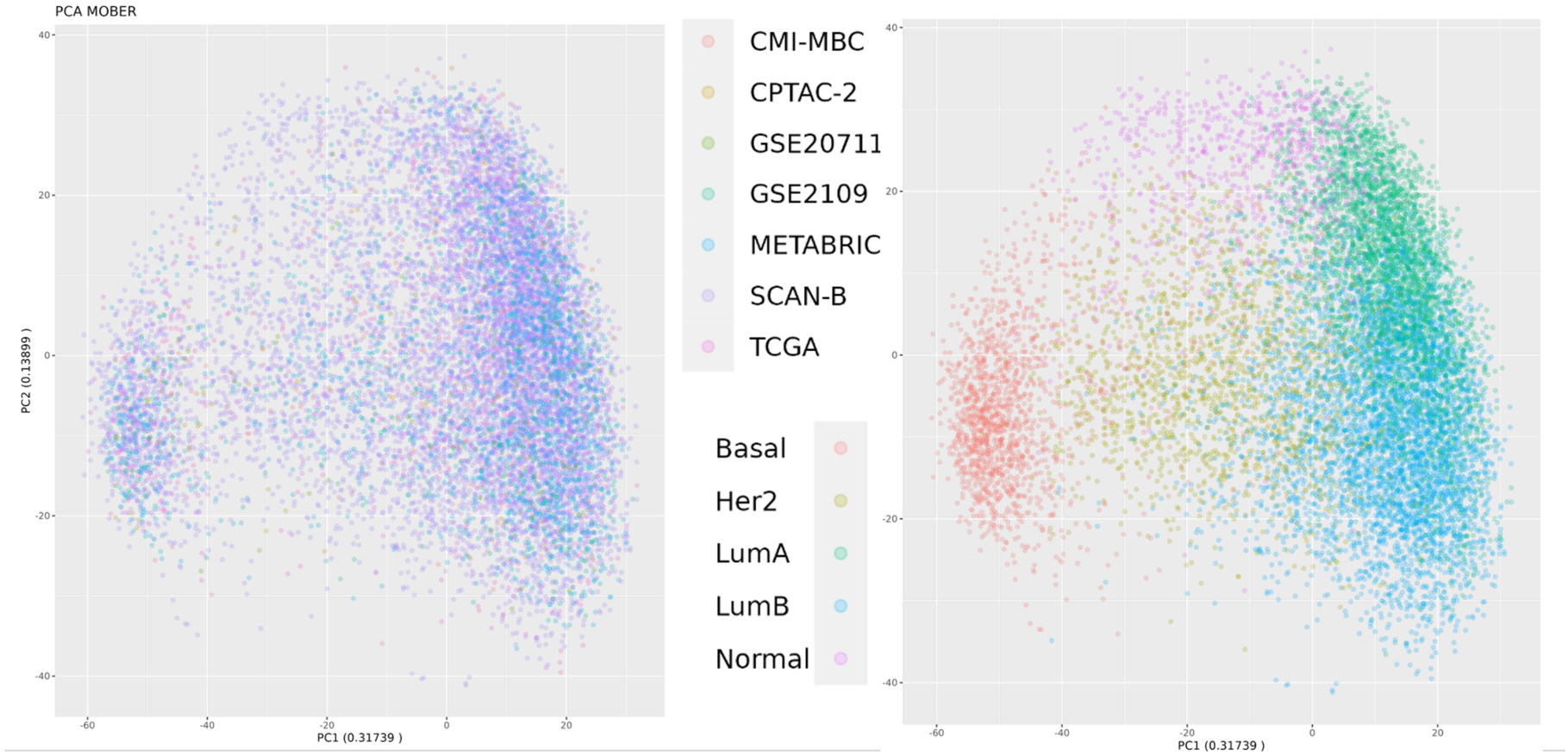
pardon my french…



Normalisation / dynamic range / cleaning / gold sample…

1- Your Data

Courtesy to Arnau

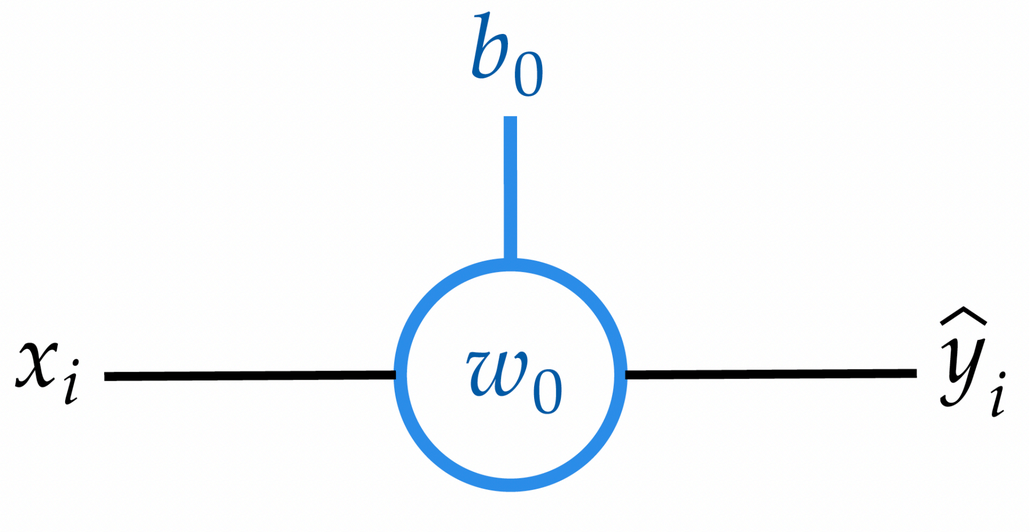


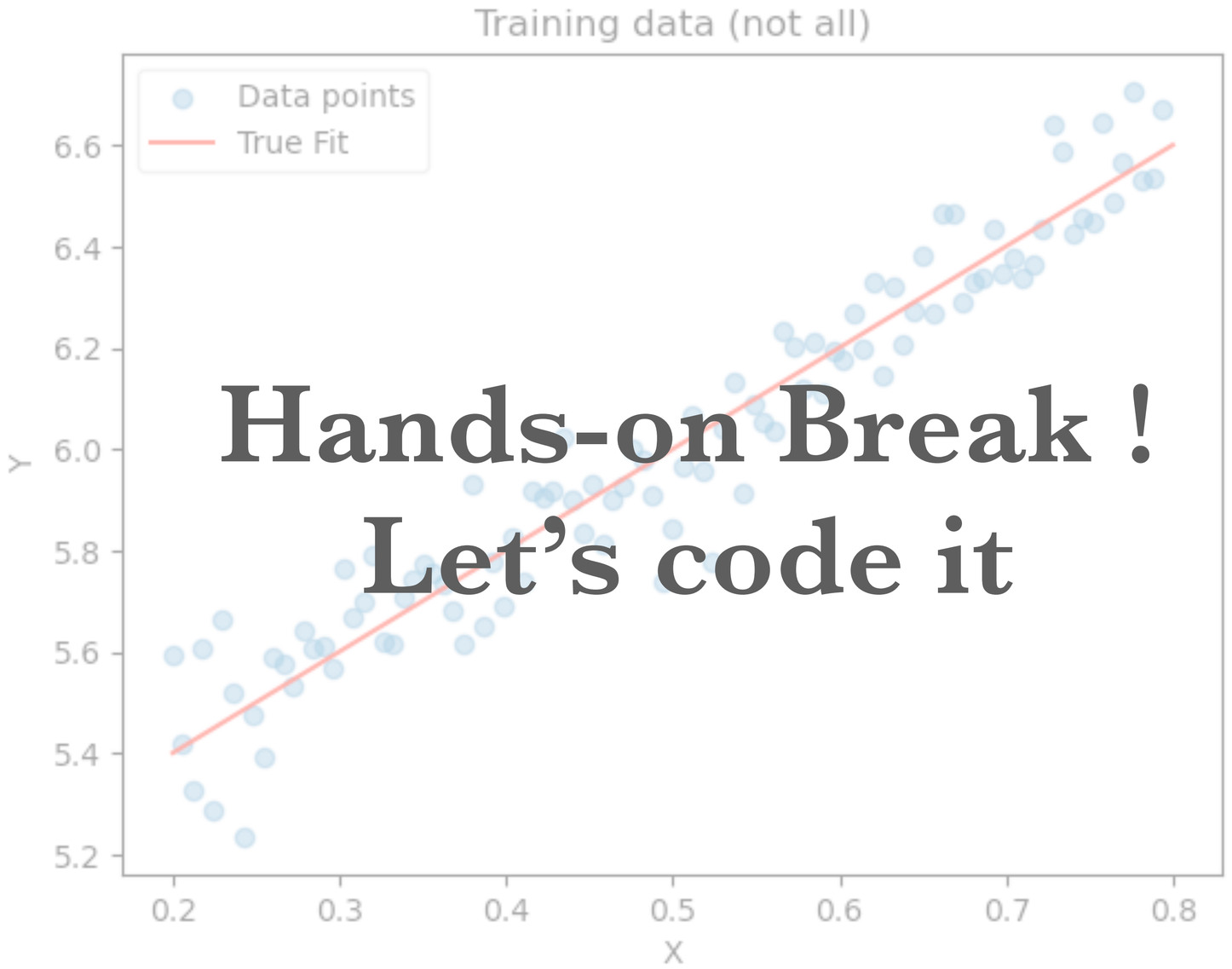
Batch effect / Normalisation / dynamic range / cleaning / gold sample…

2- Your Model

$$f(x) = w_0 x + b_0$$

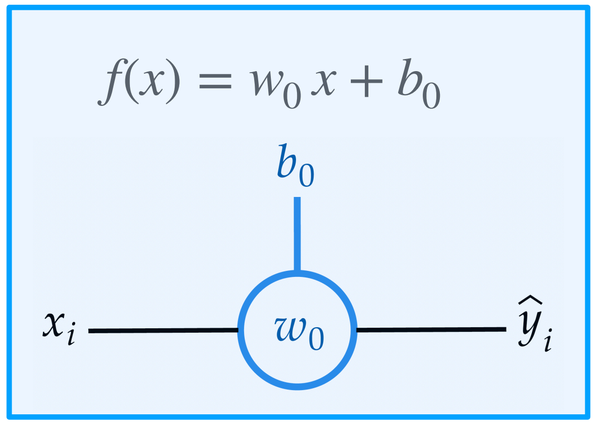
Neuron (almost)



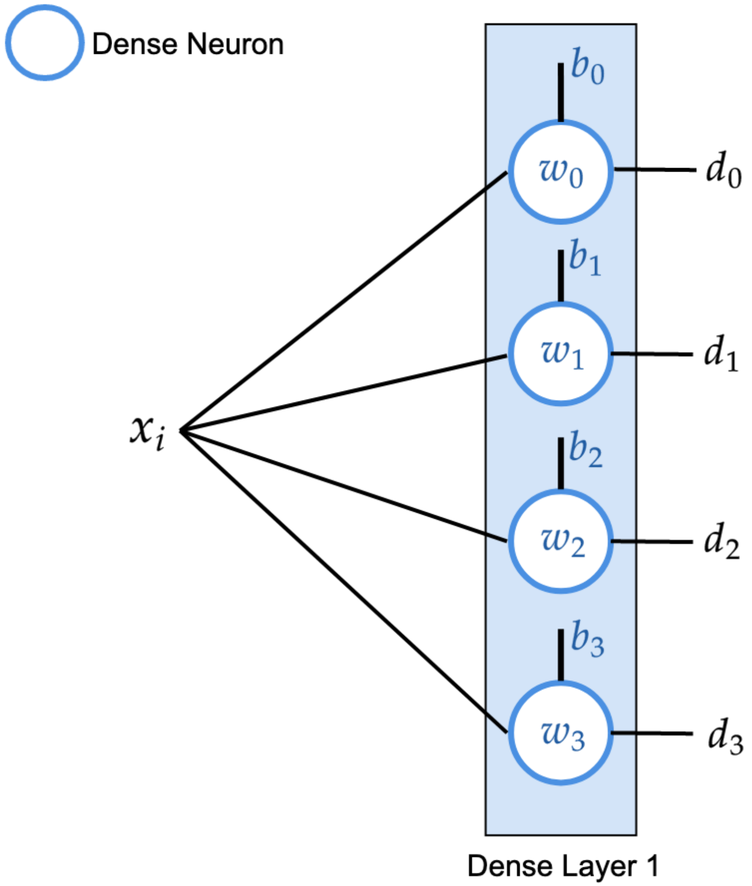


Hands-on Break !
Let’s code it

2- Your Model

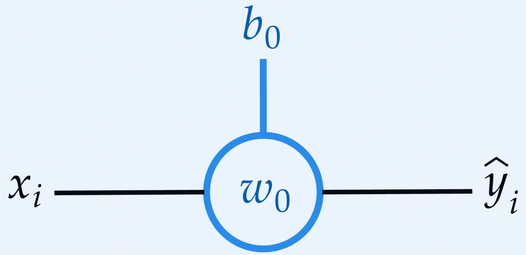


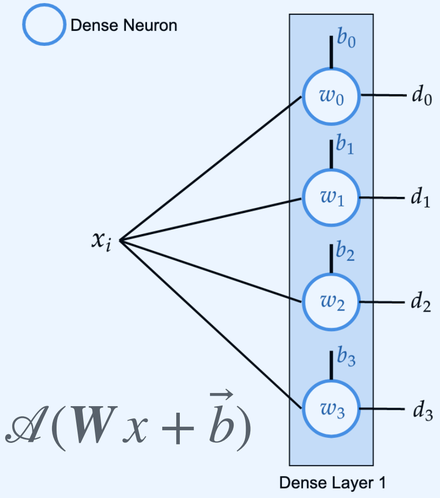
$$f(x) = \mathscr{A}(\boldsymbol{W}\,x + \vec{b})$$



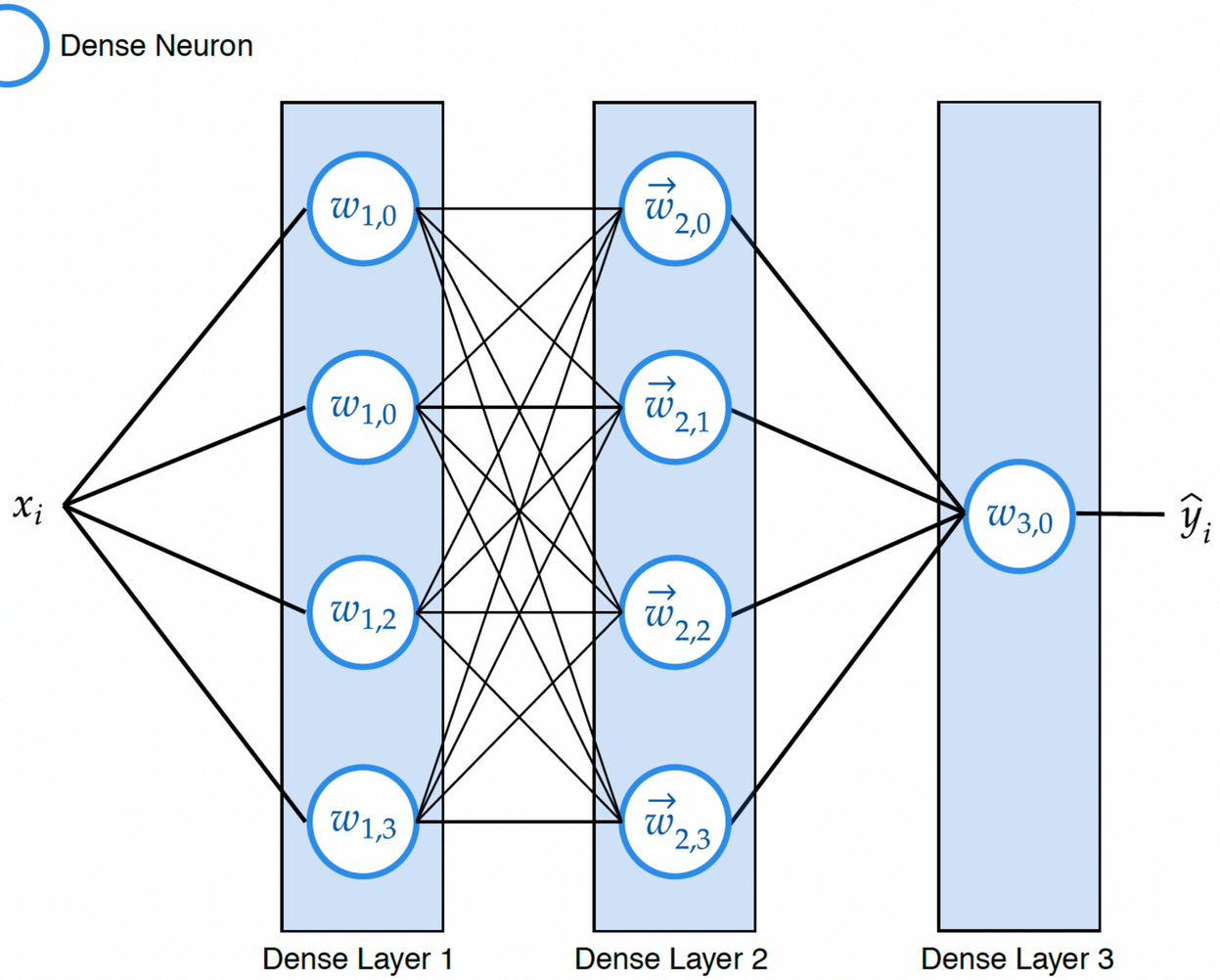
2- Your Model

$$f(x) = w_0\, x + b_0$$



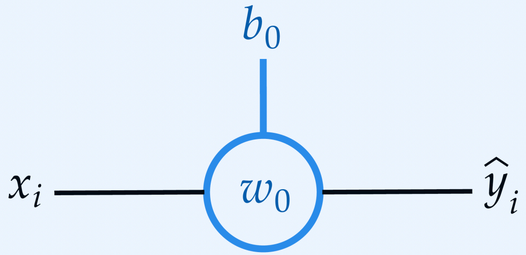


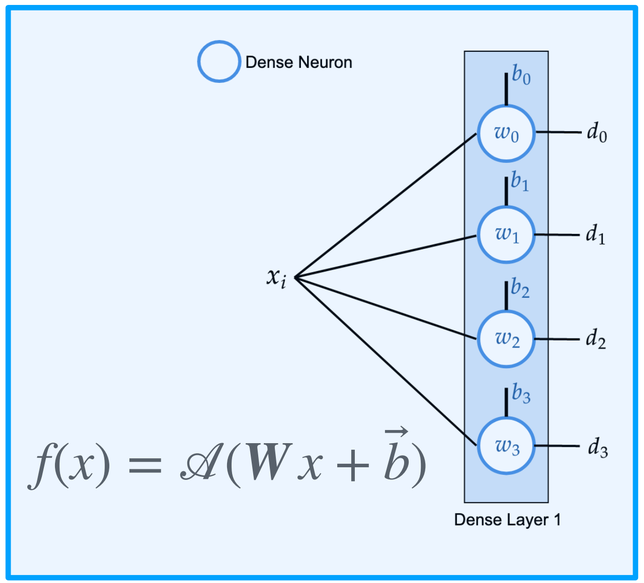
$$f(x) = \mathcal{A}(W\, x + \vec{b})$$

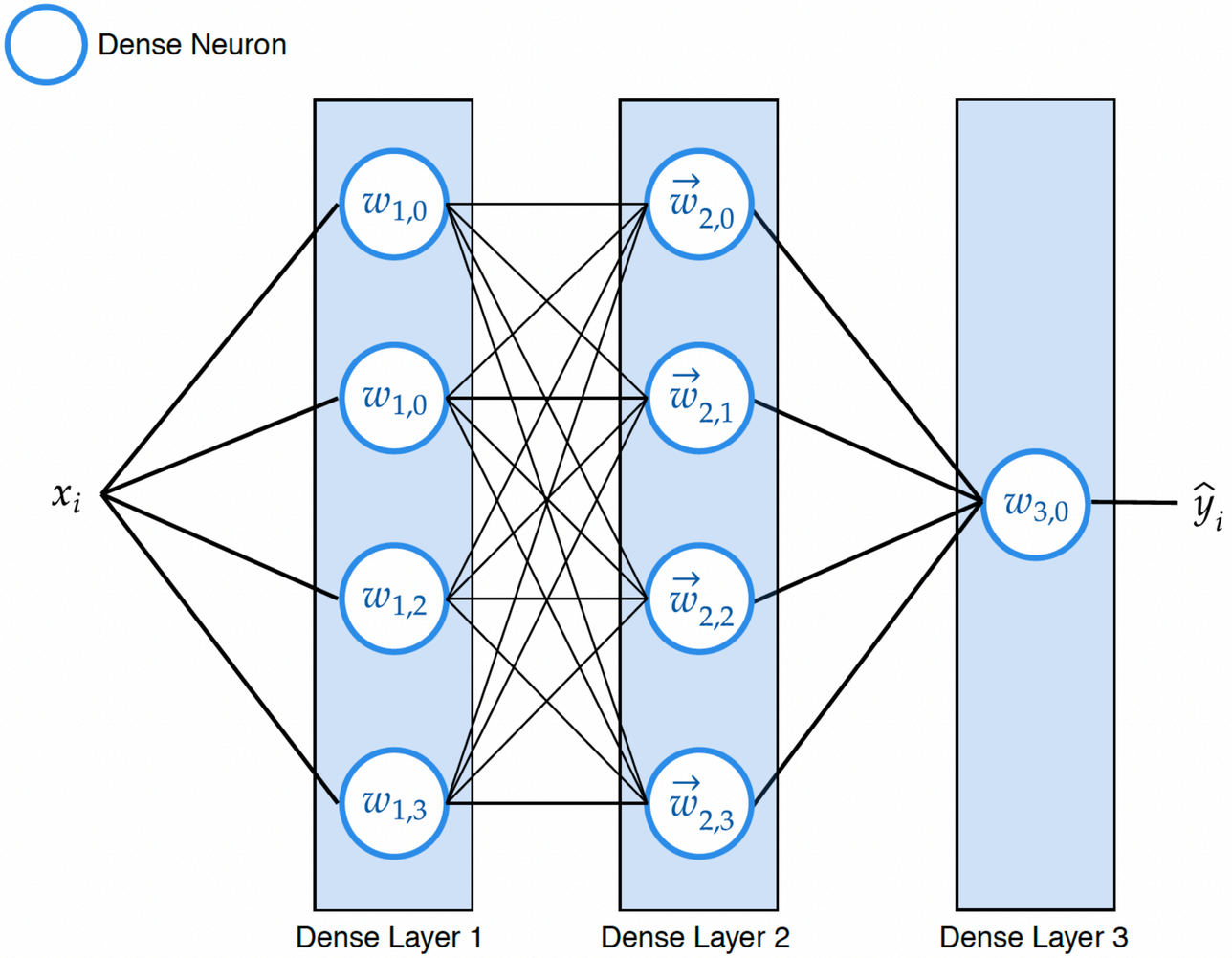


2- Your Model

$$f(x) = w_0\, x + b_0$$



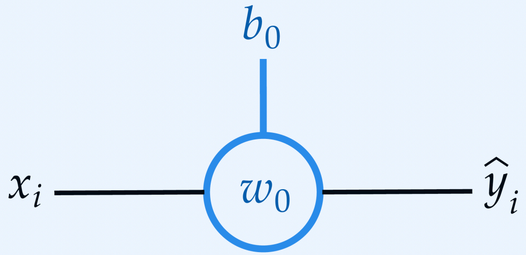


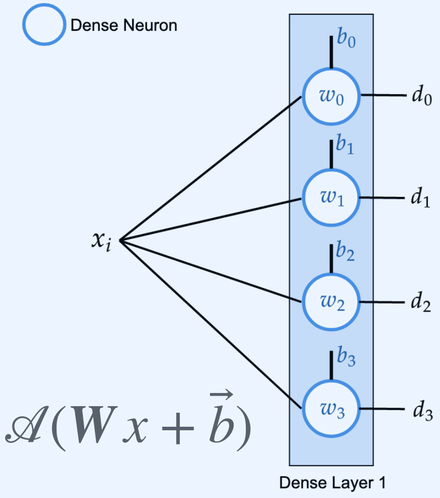


$$f(x) = \mathcal{A}\left(\boldsymbol{W_1}\,\mathcal{A}\left(\boldsymbol{W_1}\,\mathcal{A}(\boldsymbol{W_0} + \vec{b_0}) + \vec{b_1}\right) + \vec{b_2}\right)$$

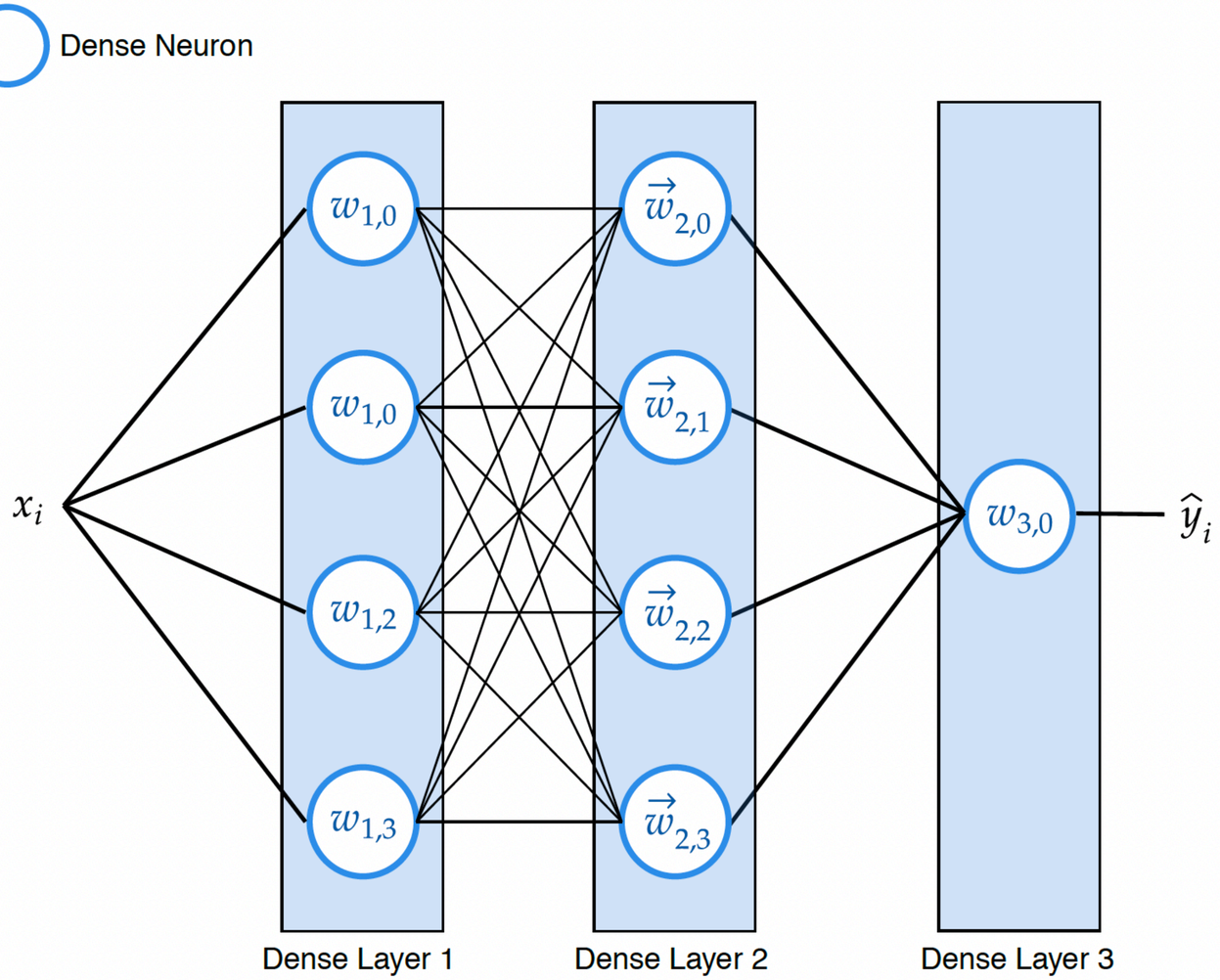
2- Your Model

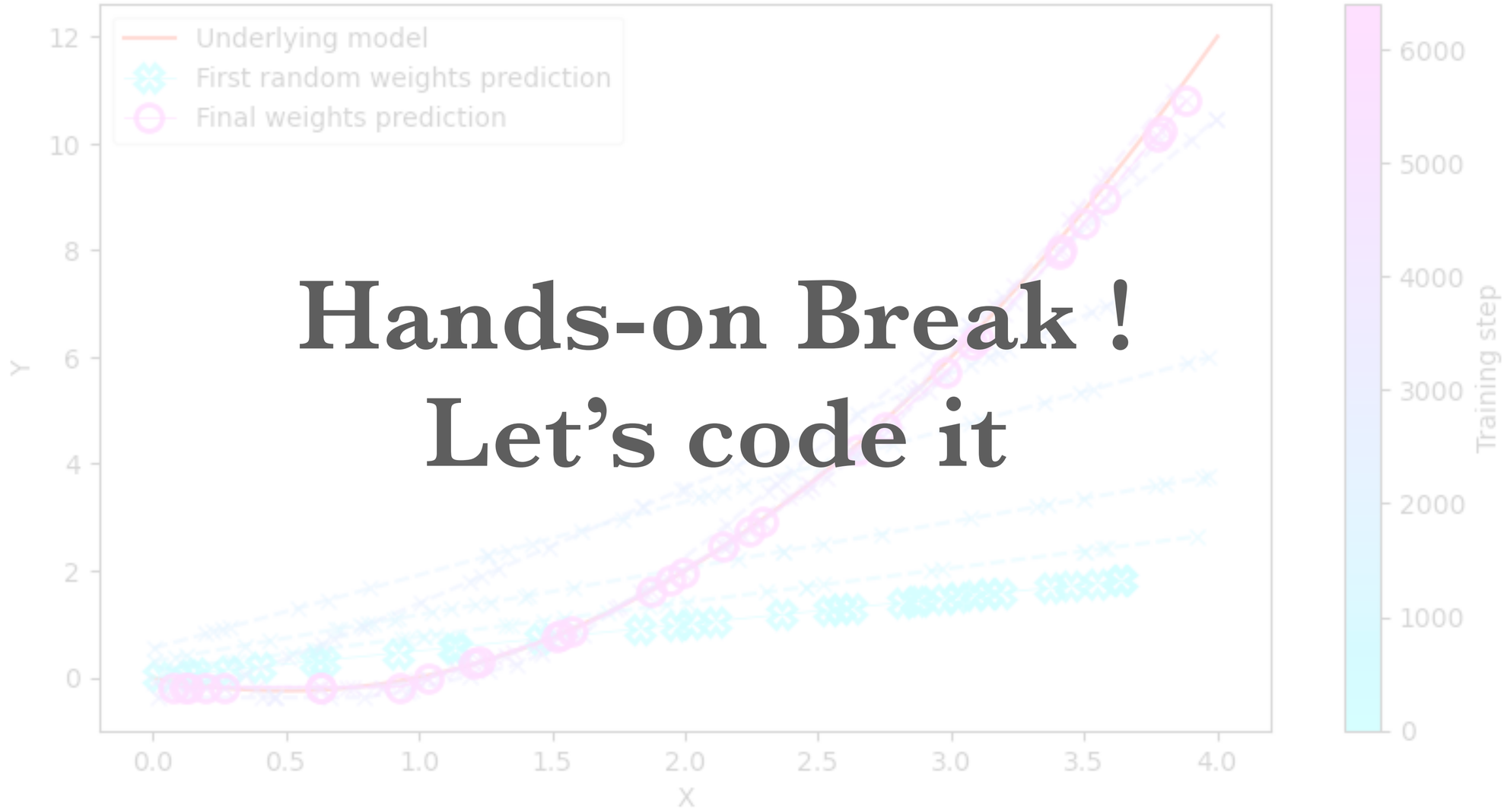
$$f(x) = w_0\, x + b_0$$





$$f(x) = \mathcal{A}(W\, x + \vec{b})$$

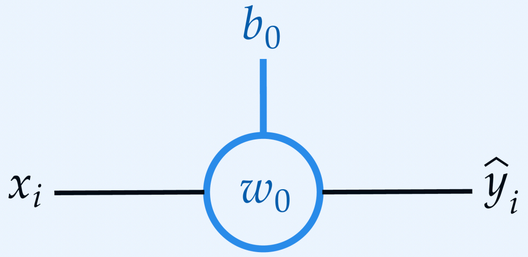


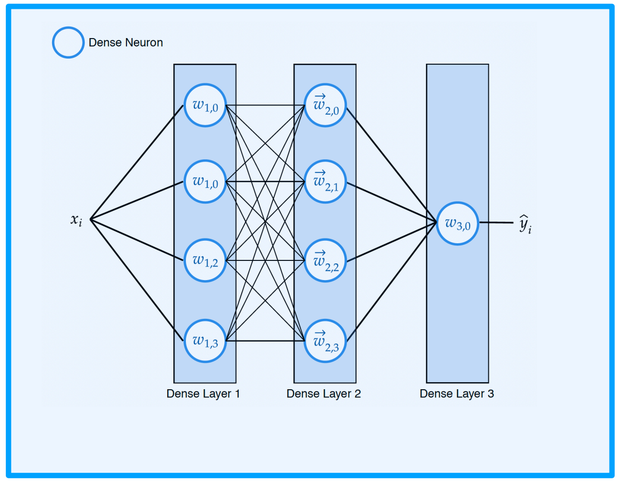


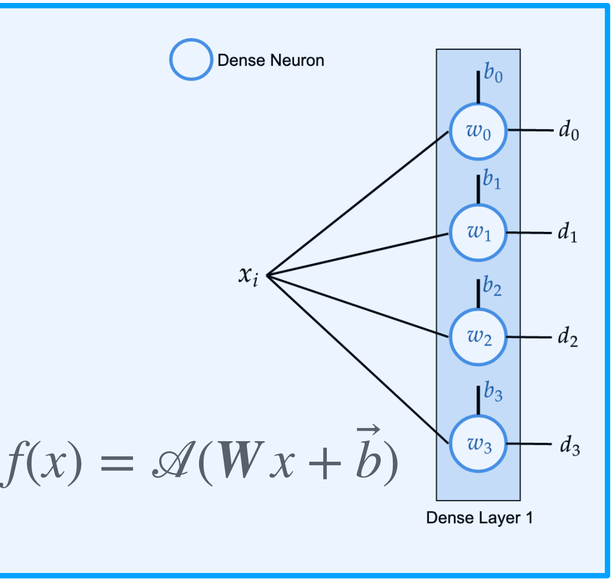
Hands-on Break !
Let’s code it

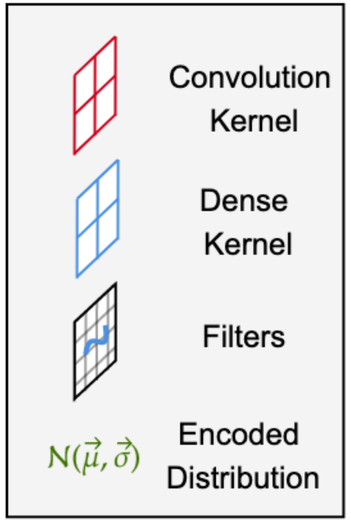
2- Your Model

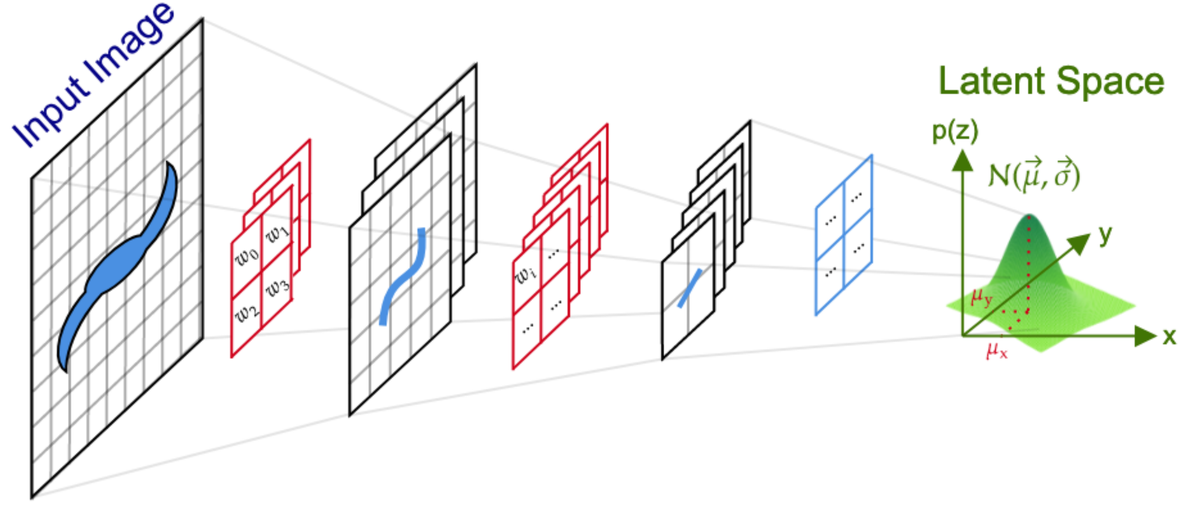
$$f(x) = w_0\, x + b_0$$





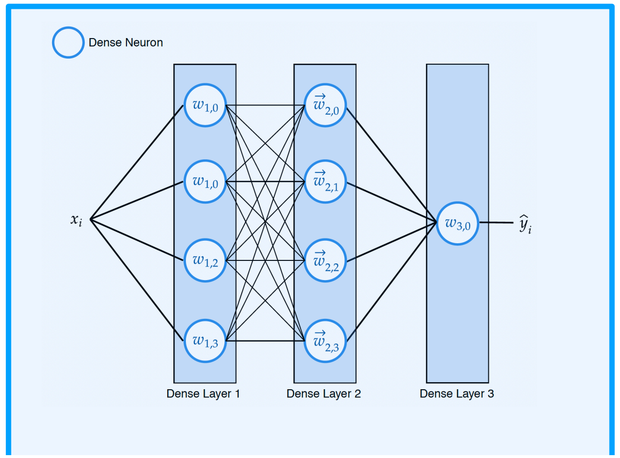


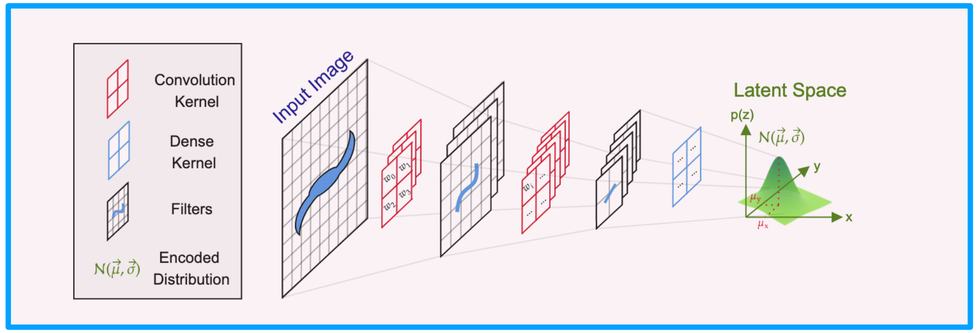


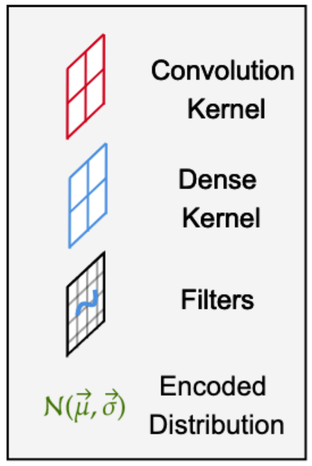


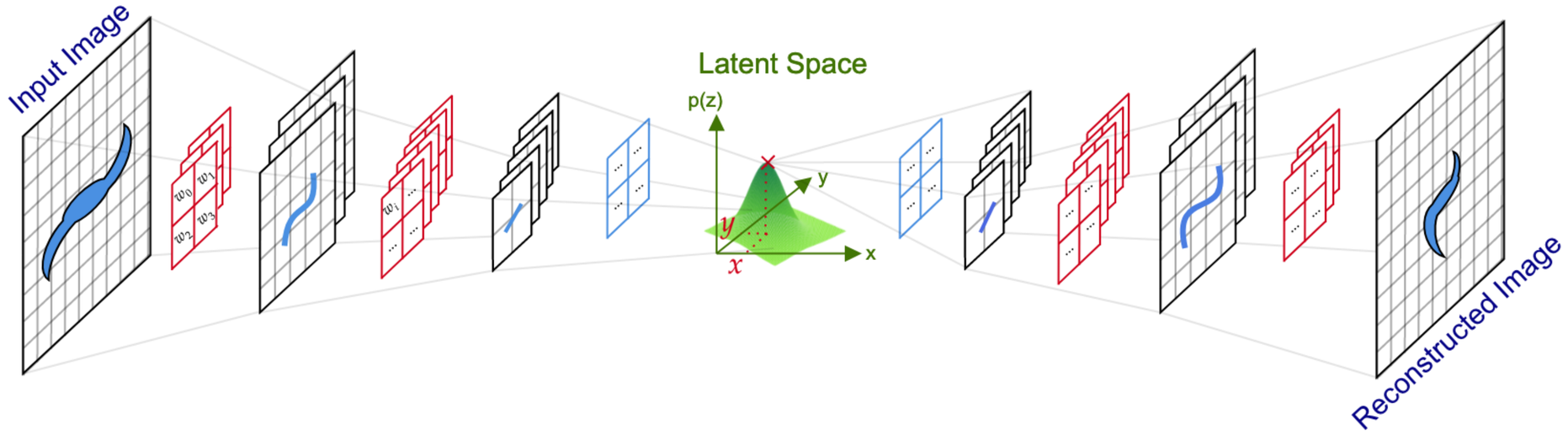
2- Your Model

$$f(x) = w_0\, x + b_0$$

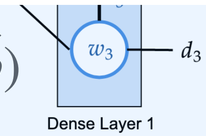






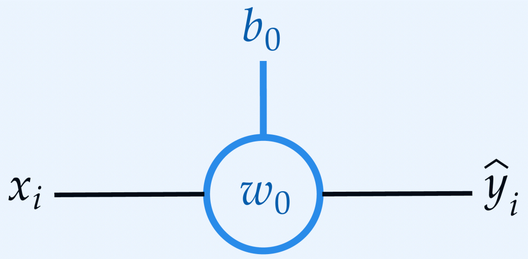


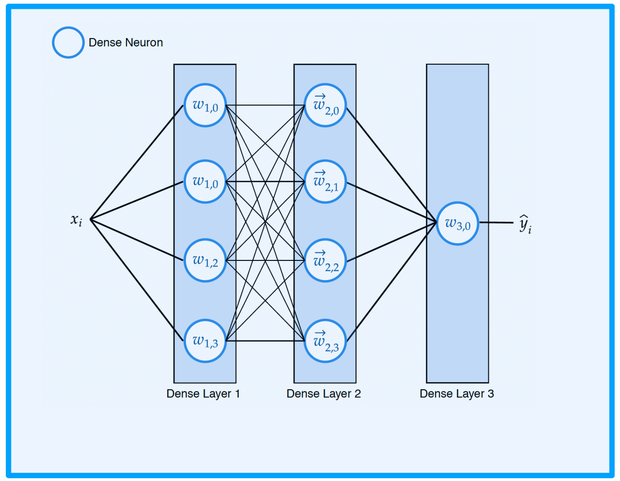
$$f(x) = \mathcal{A}(W\, x + \vec{b})$$

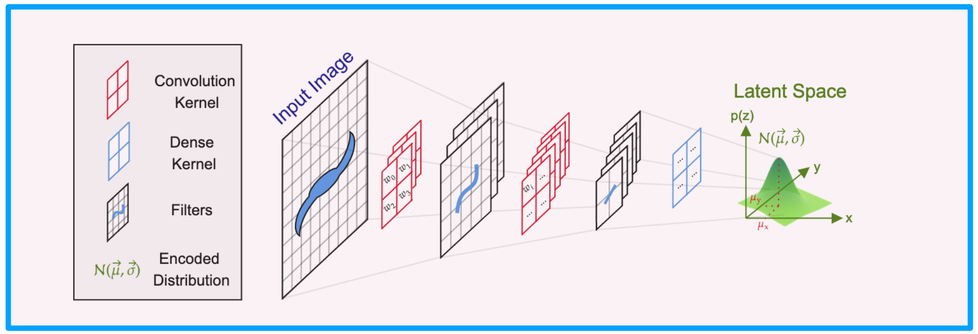


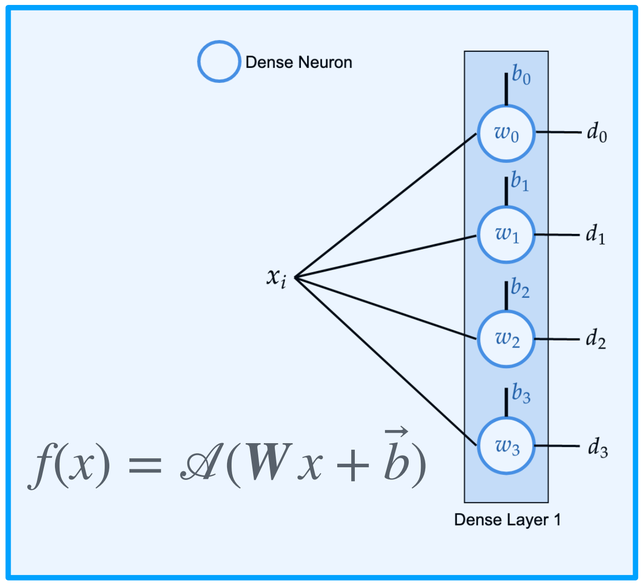
2- Your Model

$$f(x) = w_0\, x + b_0$$

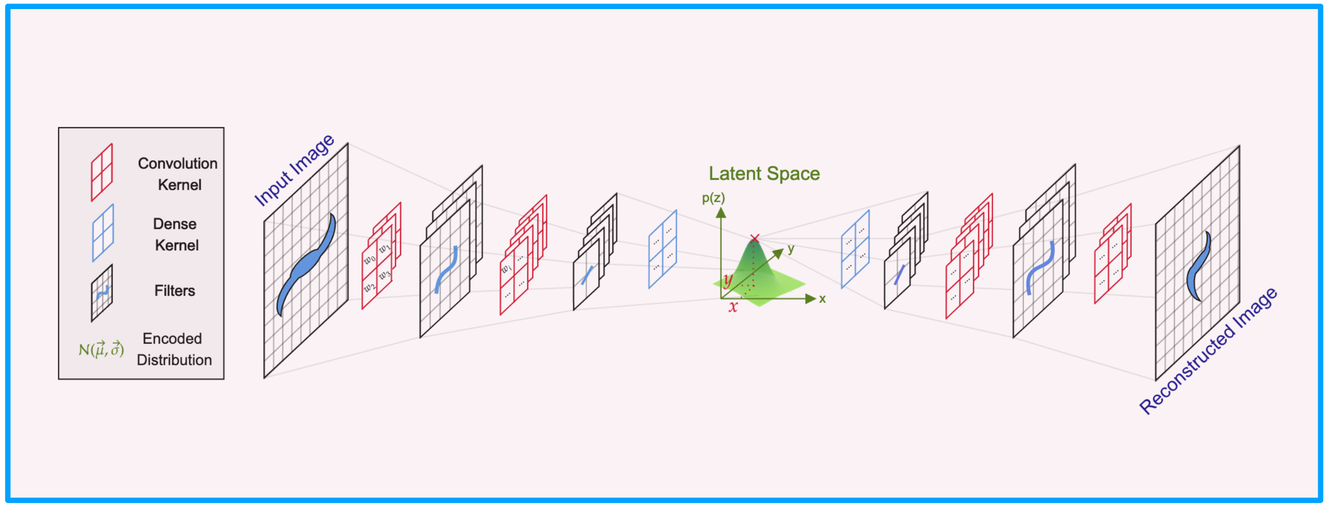






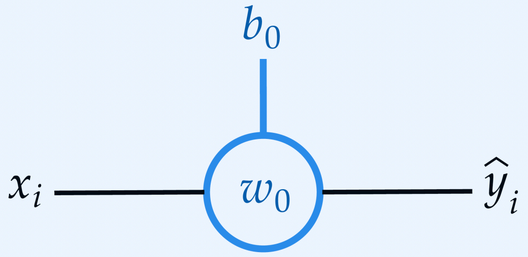


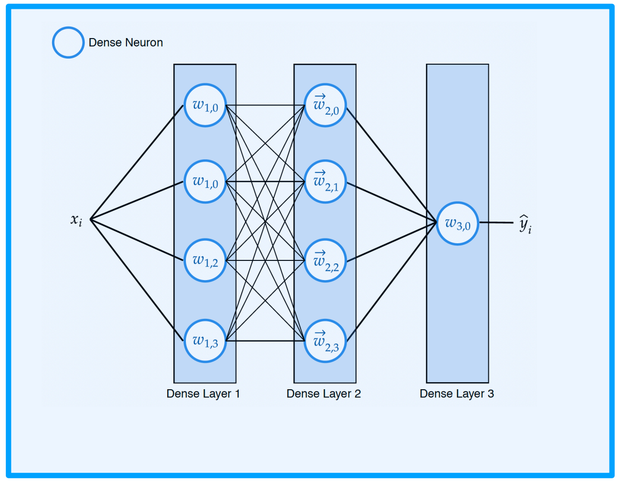
$$f(x) = \mathcal{A}(W\, x + \vec{b})$$

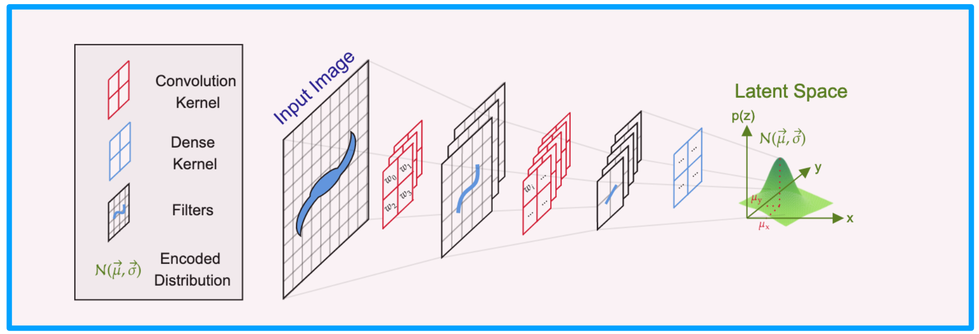


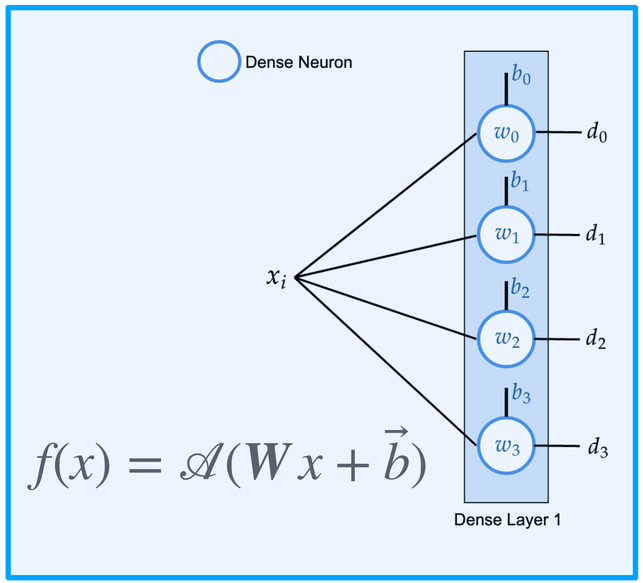
2- Your Model

$$f(x) = w_0\, x + b_0$$

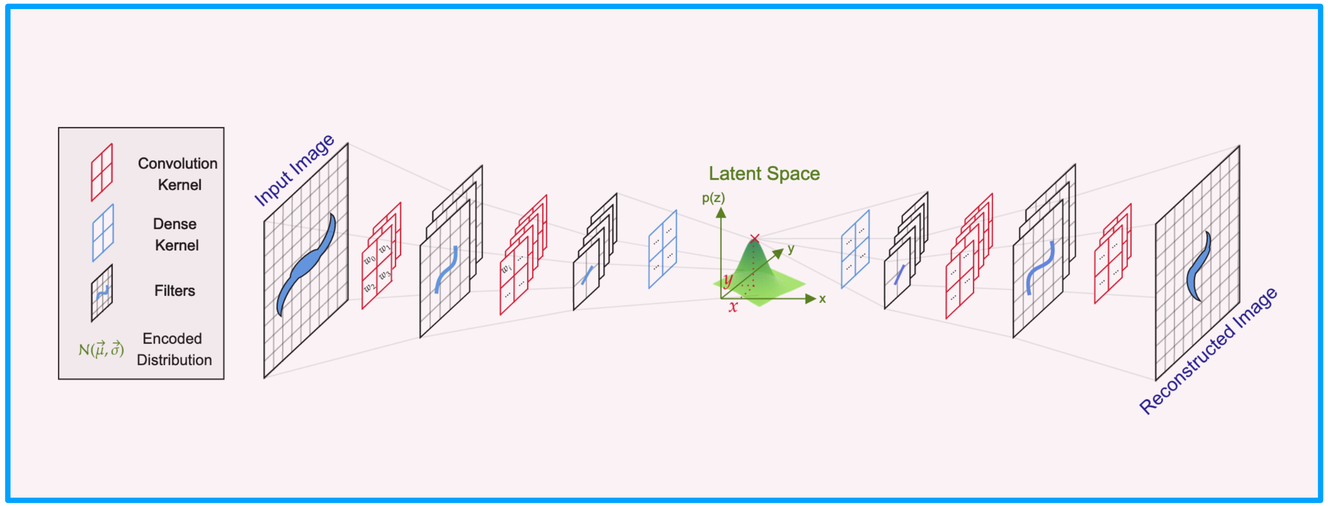








$$f(x) = \mathcal{A}(W\, x + \vec{b})$$



• • •

3- Your Loss

Add more examples

$$\mathscr{L} = Y - \hat{Y}$$

3- Your Loss

$$\mathscr{L} = Y - \hat{Y}$$

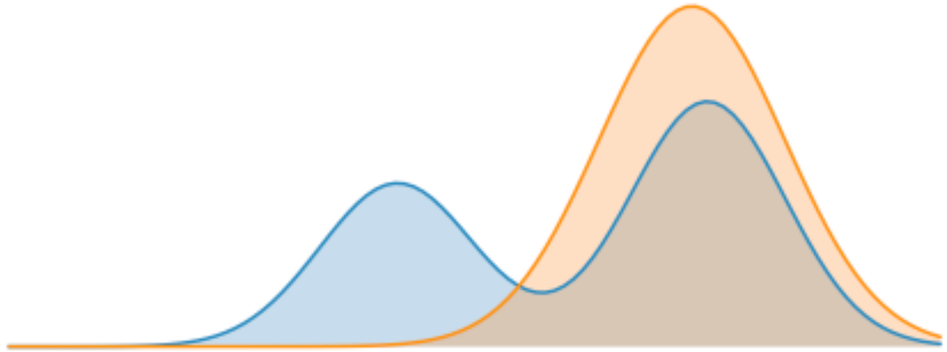
$$\mathscr{L} = |Y - \hat{Y}|^2$$

3- Your Loss

$$\mathscr{L} = Y - \hat{Y}$$

$$\mathscr{L} = |Y - \hat{Y}|^2$$

$$\mathscr{L} = \mathscr{P}(\hat{Y})$$

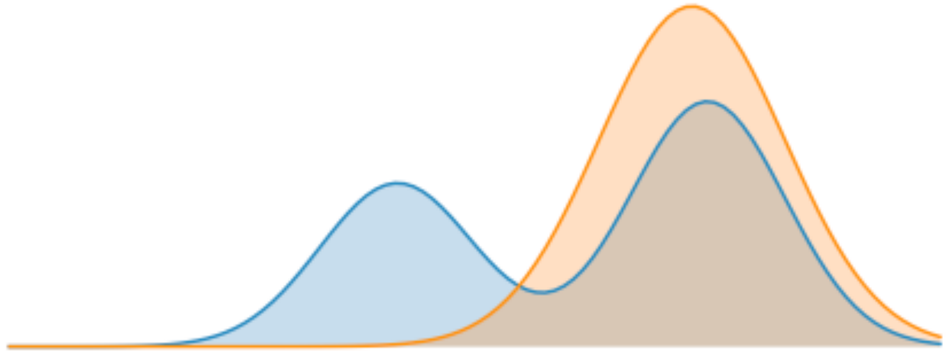


3- Your Loss

$$\mathscr{L} = Y - \hat{Y}$$

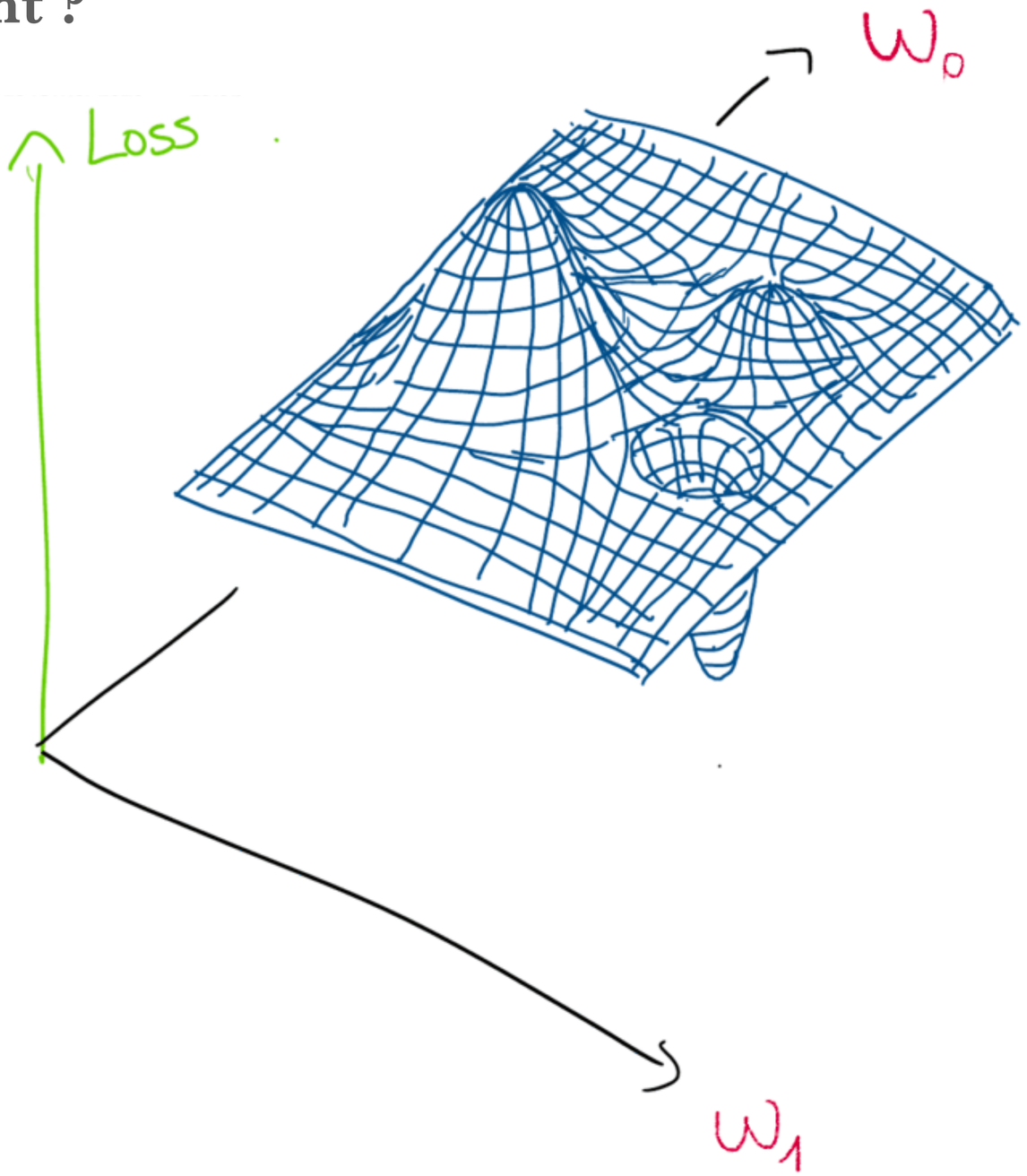
$$\mathscr{L} = |Y - \hat{Y}|^2$$

$$\mathscr{L} = \mathscr{P}(\hat{Y})$$



• • •

4- Gradient Descent ?



How I see the use of DL in academic science…

- Do you really need it ?
- Don’t randomly use a model found in the net.
- We are scientists, not pure data scientists: use your knowledge to inform your model: input, loss, tests…
- Check 10 times your results, with tests, robustness, interpretation…
- Be carefull on how you present your methods: clarity, trust, pedagogy…