



**ChatGPT**

# Introduction to Deep Learning (with PyTorch)

Andreas Makris

23/10/2024

# SORA by OpenAI

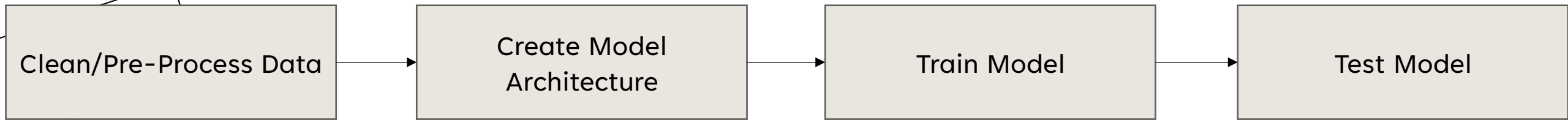




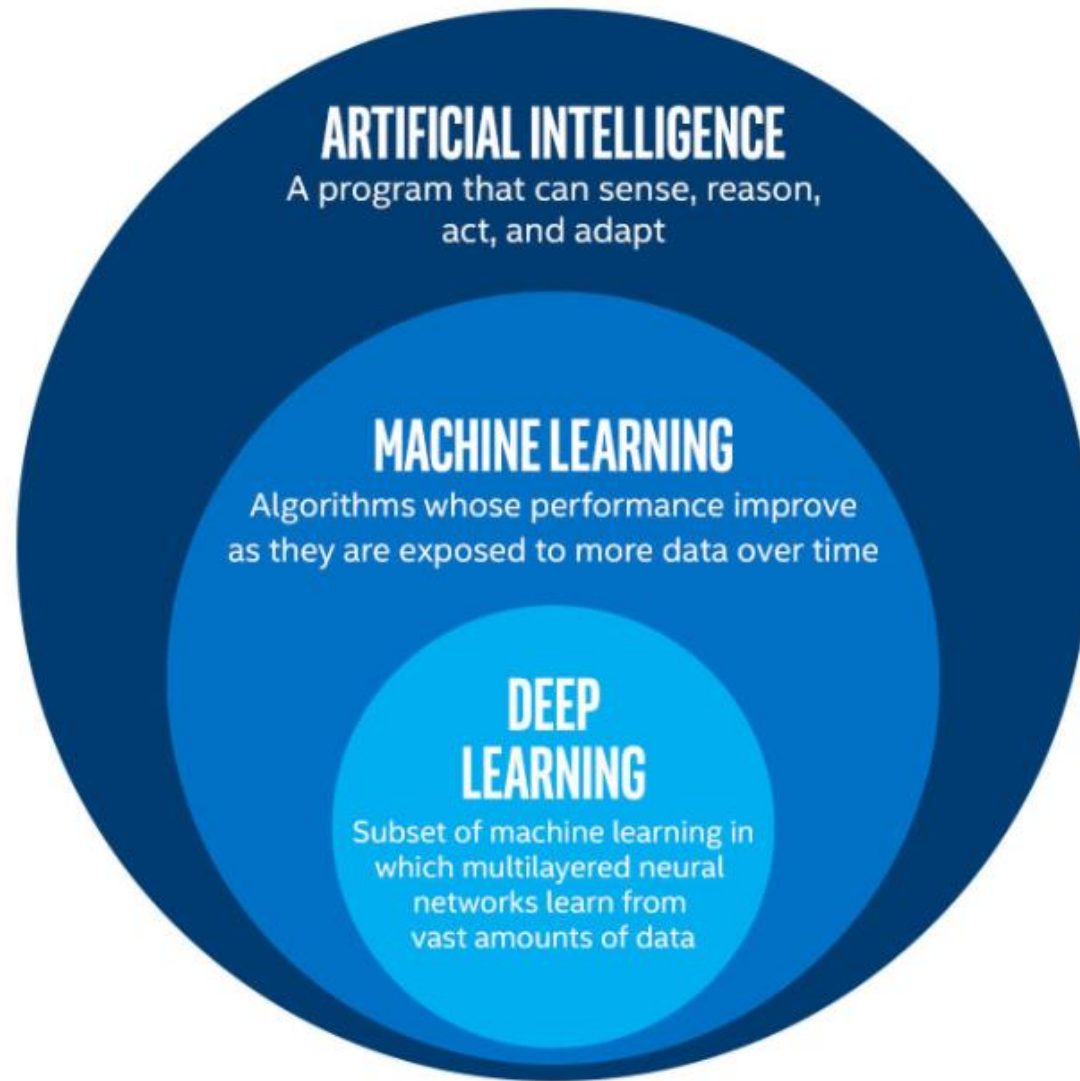
# Outline

- Introduction to Deep Learning
- Coding with PyTorch

# Deep Learning Pipeline



# What is Deep Learning?

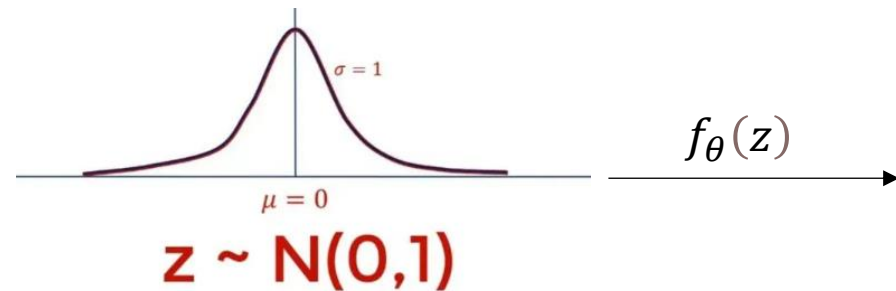


<https://mylearningsinai.ml.wordpress.com/wp-content/uploads/2018/05/ai-ml-dl-image.png>

# Yes, but what are Neural Networks?

$$h^{(l)} = a(h^{(l-1)}W^{(l)} + B^{(l)})$$

- Neural Networks are incredibly flexible **function approximators**
- Neural Networks contain multiple layers. They often multiply the input from the previous layer with a matrix, add some bias and use an activation function.
- Their flexibility rises from million/billion of parameters
- How do we learn these parameters?



How many  
parameters do  
you have?

$f_{\theta}(z)$

I have 1.76 trillion  
parameters

# Training the Model

- To estimate the parameters, use a **loss function** that we **minimize** using **backpropagation** and **gradient descent**
- Need the Neural Network to perform **differentiable operations** and the loss functions to be differentiable
- Supervised learning  $\rightarrow$  (x,y) pairs
- Unsupervised learning  $\rightarrow$  Only x
- Self-Supervised learning  $\rightarrow$  e.g. create (x,y) pairs using only x

Task	Error type	Loss function
Regression	Mean-squared error	$\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$
	Mean absolute error	$\frac{1}{n} \sum_{i=1}^n  y_i - \hat{y}_i $
Classification	Cross entropy = Log loss	$-\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] =$

$$\mathbb{E}_{q(\mathbf{z})} [\log p_{\theta}(\mathbf{x}|\mathbf{z})] - \text{KL}[q(\mathbf{z})||p(\mathbf{z})]$$

$$\theta^* = \arg \min_{\theta} \sum_{i=1}^N \sigma_i^2 \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{p_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x})} [\| \mathbf{s}_{\theta}(\tilde{\mathbf{x}}, \sigma_i) - \nabla_{\tilde{\mathbf{x}}} \log p_{\sigma_i}(\tilde{\mathbf{x}} | \mathbf{x}) \|_2^2]$$

$$\mathcal{L}(\theta, \phi) := \mathbb{E}_{p_{\text{data}}(\mathbf{x})} [\log D_{\phi}(\mathbf{x})] + \mathbb{E}_{p_{\theta}(\mathbf{x})} [\log(1 - D_{\phi}(\mathbf{x}))].$$



# Testing the Neural Network

- How well does it approximate our function?
- Test dataset that was NOT trained on
- MSE, Accuracy, FID score, Visual Inspection, Perplexity



# Case Study – Image Classification

- Classify handwritten digits (0 to 9)
- Our inputs will have dimensions [B, 1, 28, 28]
- What should the output dimensions be?
- What should our architecture be to be consistent with the output?
- What should the loss function be?
- How should we evaluate the model?



$$\xrightarrow{f_{\theta}(x)} 9$$

# Case Study – Image Classification

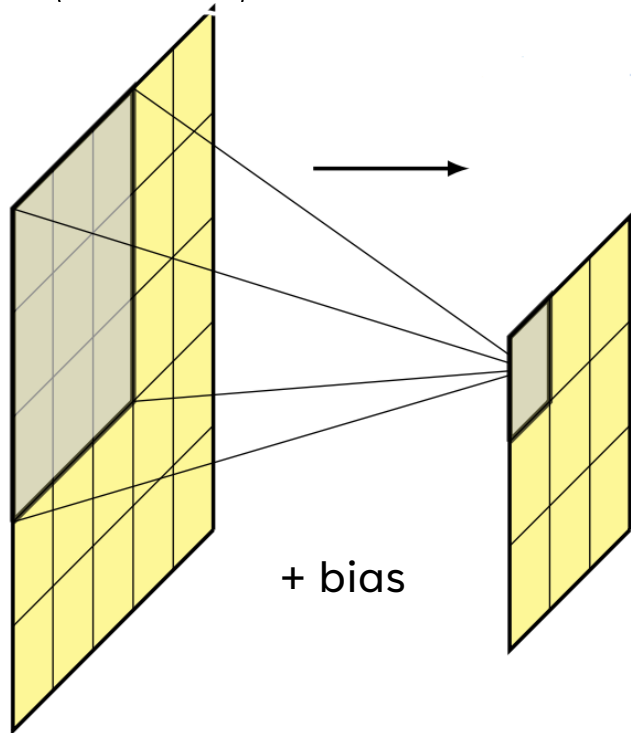
- Classify handwritten digits (0 to 9)
- Our inputs will have dimensions [B, 1, 28, 28]
- What should the output dimensions be?
- [B, 10]
- What should our architecture be to be consistent with the output?
- What should the loss function be?
- How should we evaluate the model?



$$\xrightarrow{f_{\theta}(x)} 9$$

# Case Study – Image Classification

- What should our architecture be to be consistent with the output?
- Use CNN with Convolutional, MaxPool and Linear layers.











12	20	30	0
8	12	2	0
34	70	37	4
112	100	25	12

$2 \times 2$  Max-Pool

20	30
112	37

<https://computersciencewiki.org/index.php/File:MaxpoolSample2.png>  
<https://bishopbook.com>

# Image Convolution

			
	<p>11111 11111 11111 11111 11111</p>		<p>11111 11411 14841 11411 11111</p>
			<p>11111 22222 44444 22222 11111</p>
			
	<p>11 111 22 222 00 000 -2-2-2-2-2 -1-1-1-1-1</p>		<p>120-2-1 120-2-1 120-2-1 120-2-1 120-2-1</p>
			<p>-1-1 -1-1-1 -1-1 -1-1-1 -1-1 24-1-1 -1-1 -1-1-1 -1-1 -1-1-1</p>

# Examples of 2D image filters



(wikipedia)

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$



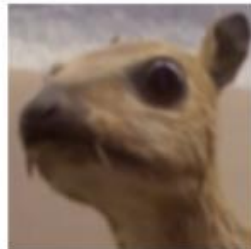
Edge Detection

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$



Sharpen

$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$



Gaussian Blur

Remember: in CNNs all learned through backpropagation, dependant on the task!

Slide credit: Smola, Li 2019

DEEP LEARNING: BELTRAMO & SMOLA

# Case Study – Image Classification

- Classify handwritten digits (0 to 9)
- Our inputs will have dimensions [B, 1, 28, 28]
- What should the output dimensions be?
- [B, 10]
- What should our architecture be to be consistent with the output?
- CNN
- What should the loss function be?
- Cross Entropy Loss
- How should we evaluate the model?

$$L = -\frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C y_c^{(i)} \log(\hat{y}_c^{(i)})$$

# Case Study – Image Classification

- Classify handwritten digits (0 to 9)
- Our inputs will have dimensions [B, 1, 28, 28]
- What should the output dimensions be?
- [B, 10]
- What should our architecture be to be consistent with the output?
- CNN
- What should the loss function be?
- Cross Entropy Loss
- How should we evaluate the model?
- Accuracy

$$L = -\frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C y_c^{(i)} \log(\hat{y}_c^{(i)})$$



# Time to Code!

*What I cannot create, I do not understand.*

~Richard Feynman, 1988

Why PyTorch? <https://paperswithcode.com/trends>

<https://colab.research.google.com/drive/1KQnXlb9QEHn0MgPq6i118GP6Cyf4QfNH>