Introduction

CISC 7026 - Introduction to Deep Learning

Steven Morad

University of Macau

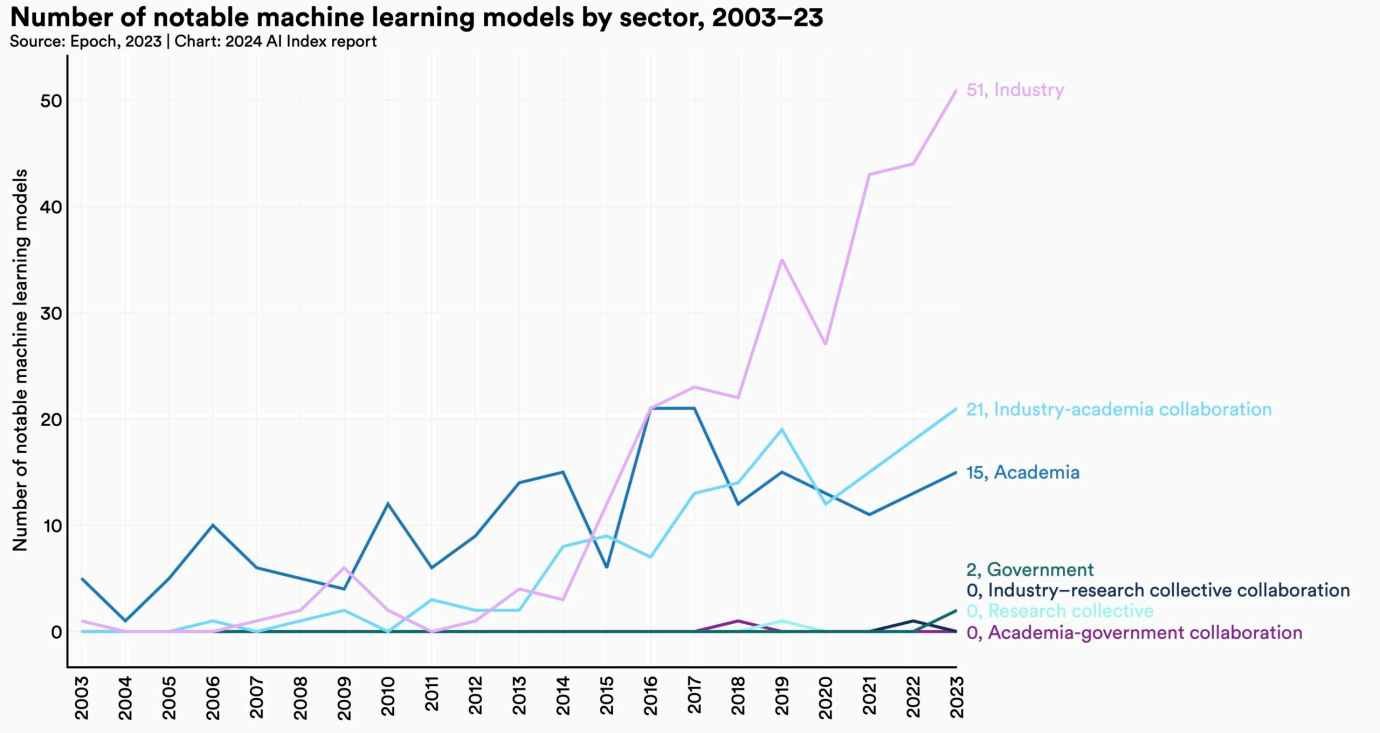
Outline

Deep Learning Successes 2
What is Deep Learning? 15
Terminology: AI, ML, DL 22
Machine Learning 24
Course Timeline 33
Programming 35
Coding 45

Deep Learning Successes

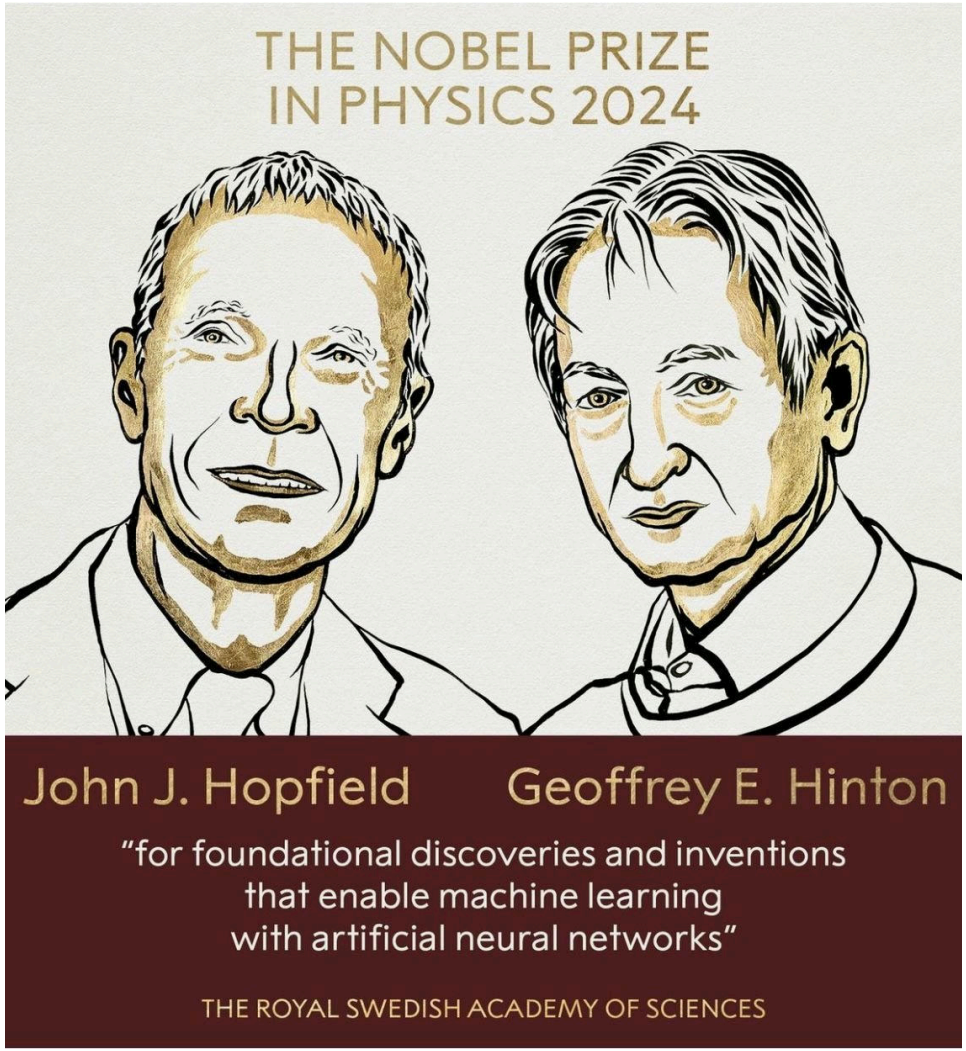
Deep Learning Successes

Deep learning is becoming very popular worldwide



Credit: Stanford University 2024 AI Index Report

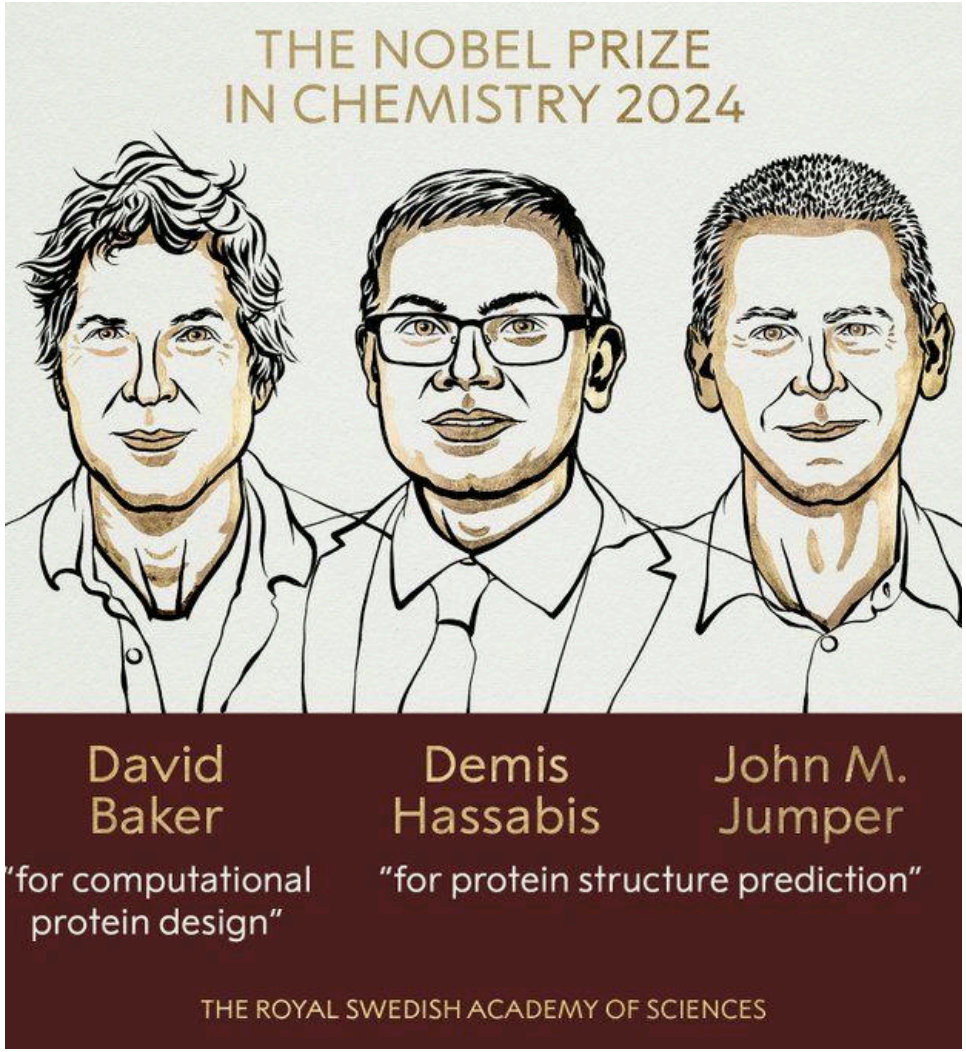
Deep Learning Successes



John Hopfield and Geoffrey Hinton won the 2024 Nobel prize in physics

They won the prize for their work on deep learning

Deep Learning Successes



Hassabis (DeepMind), Jumper, and Baker won the 2024 Nobel prize in chemistry

Also for their work on deep learning

Deep Learning Successes

Science fiction is now possible through deep learning

It creates art

Create an image expressing how far deep learning has come

Here is the image depicting the advancements in deep learning. It showcases a futuristic cityscape with various elements symbolizing the integration of AI and deep learning into society. Let me know if you need any adjustments or further details!

Deep Learning Successes

It beat the world champions at difficult video games like DotA 2

https://youtu.be/eHipy_j29Xw?si=iM8QVB6_P-ROUU1Y

Deep Learning Successes

It is learning to use tools and break rules

https://youtu.be/kopoLzvh5jY?si=keH4i8noY4zUVNrP

Deep Learning Successes

It is operating fully autonomous taxis in many cities

https://www.youtube.com/watch?v=Zeyv1bN9v4A

Deep Learning Successes

Maybe it is doing your homework, then explaining itself

Write a stack implementation in python

Certainly! Below is a simple stack implementation in Python using a list. The stack will support typical operations such as push, pop, peek, and checking if the stack is empty.

python Copy code

```python
class Stack:
    def __init__(self):
        self.items = []

    def is_empty(self):
        return len(self.items) == 0

    def push(self, item):
        self.items.append(item)

    def pop(self):
        if self.is_empty():
            raise IndexError("pop from empty stack")
        return self.items.pop()

    def peek(self):
        if self.is_empty():
            raise IndexError("peek from empty stack")
        return self.items[-1]

    def size(self):
        return len(self.items)

    def __str__(self):
        return str(self.items)

# Example usage:
if __name__ == "__main__":
    stack = Stack()
    stack.push(1)
    stack.push(2)
    stack.push(3)
    print("Stack after pushes:", stack)
    print("Peek:", stack.peek())
    print("Pop:", stack.pop())
    print("Stack after pop:", stack)
    print("Is stack empty?", stack.is_empty())
    print("Stack size:", stack.size())
```

Explanation:

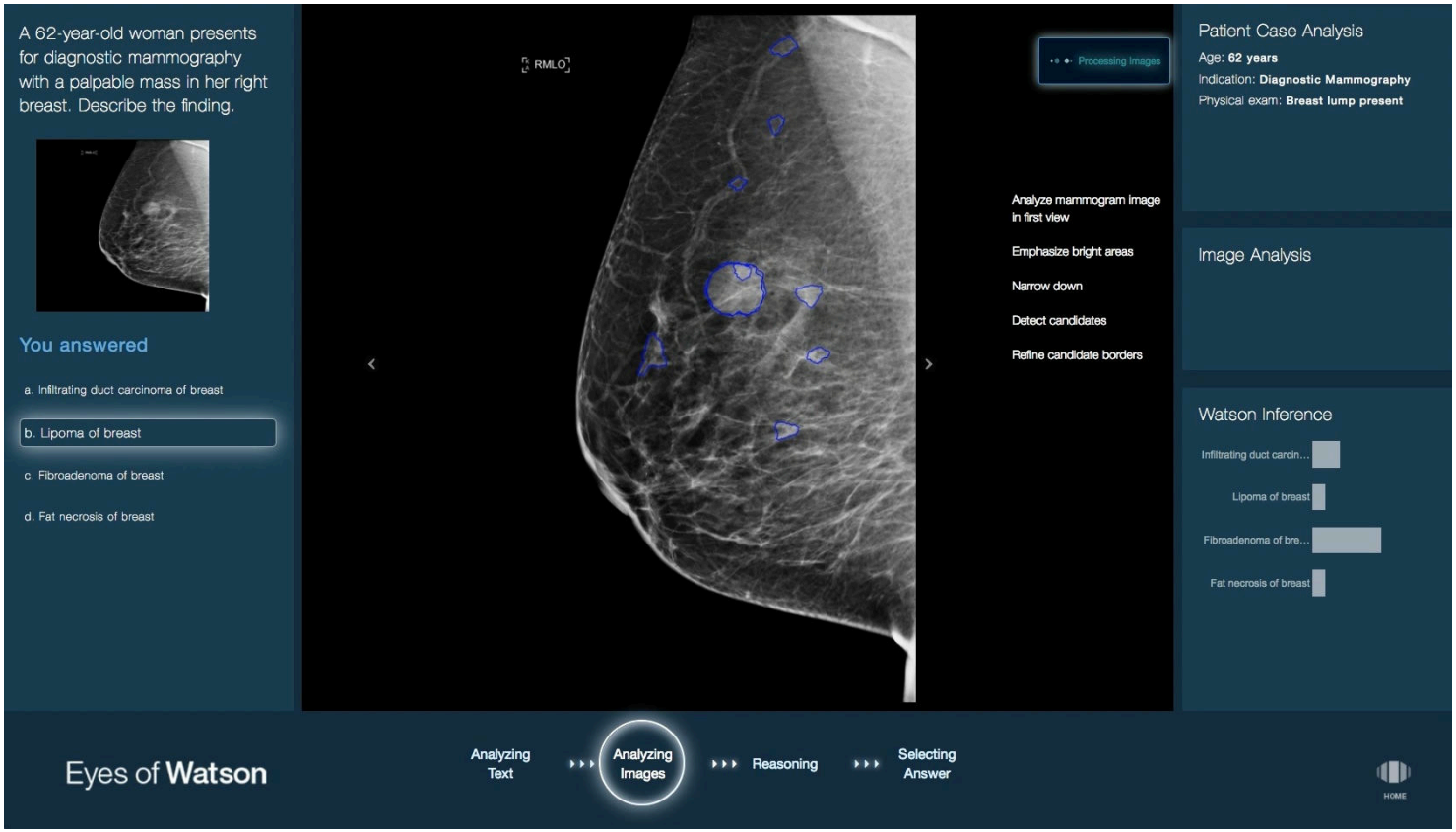
1. **Initialization**: The stack is initialized with an empty list `self.items`.
2. **is_empty**: Checks if the stack is empty by returning `True` if the length of `self.items` is 0.

Deep Learning Successes

It is making you lose money in the stock market

Deep Learning Successes

It is telling your doctor if you have cancer



Deep Learning Successes

We are solving more and more problems using deep learning

Deep learning is creeping into our daily lives

In many cases, deep models outperform humans

Our deep models keep improving as we get more data

Opinion: In the next 10 years, our lives will look very different

Deep Learning Successes

Throughout this course, you will be training your own deep models

After the course, you will be experts at deep learning

Deep learning is a powerful tool

Like all powerful tools, deep learning can be used for evil

Request:

Before you train a deep model, consider if it is good or bad for the world

What is Deep Learning?

What is Deep Learning?

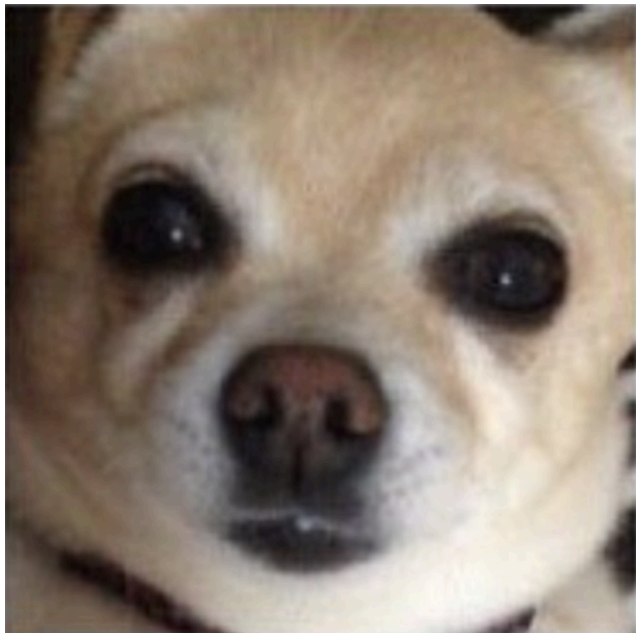
How does deep learning work?

It consists of four parts:

1. Dataset
2. Deep neural network
3. Loss function
4. Optimization

What is Deep Learning?

The **dataset** provides a set inputs and associated outputs



Dog

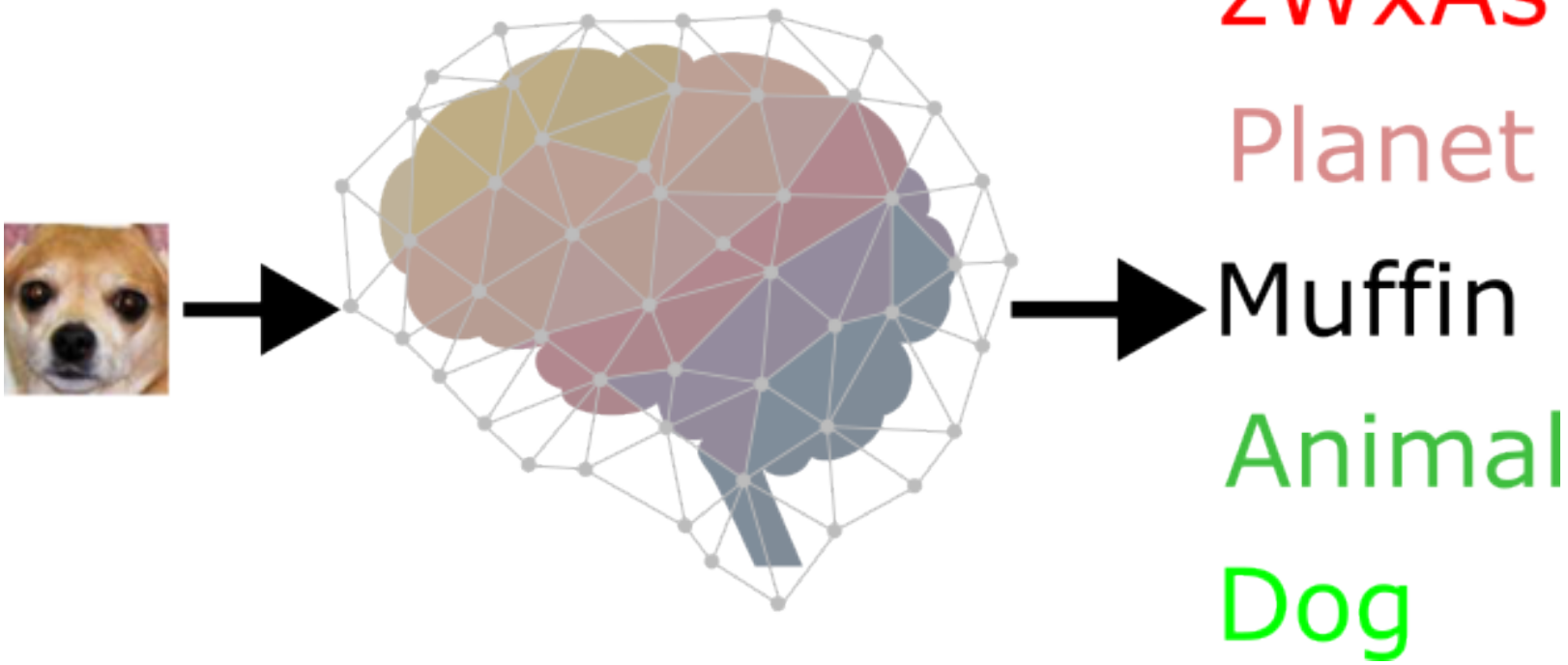
Muffin

What is Deep Learning?

The **deep neural network** maps inputs to outputs

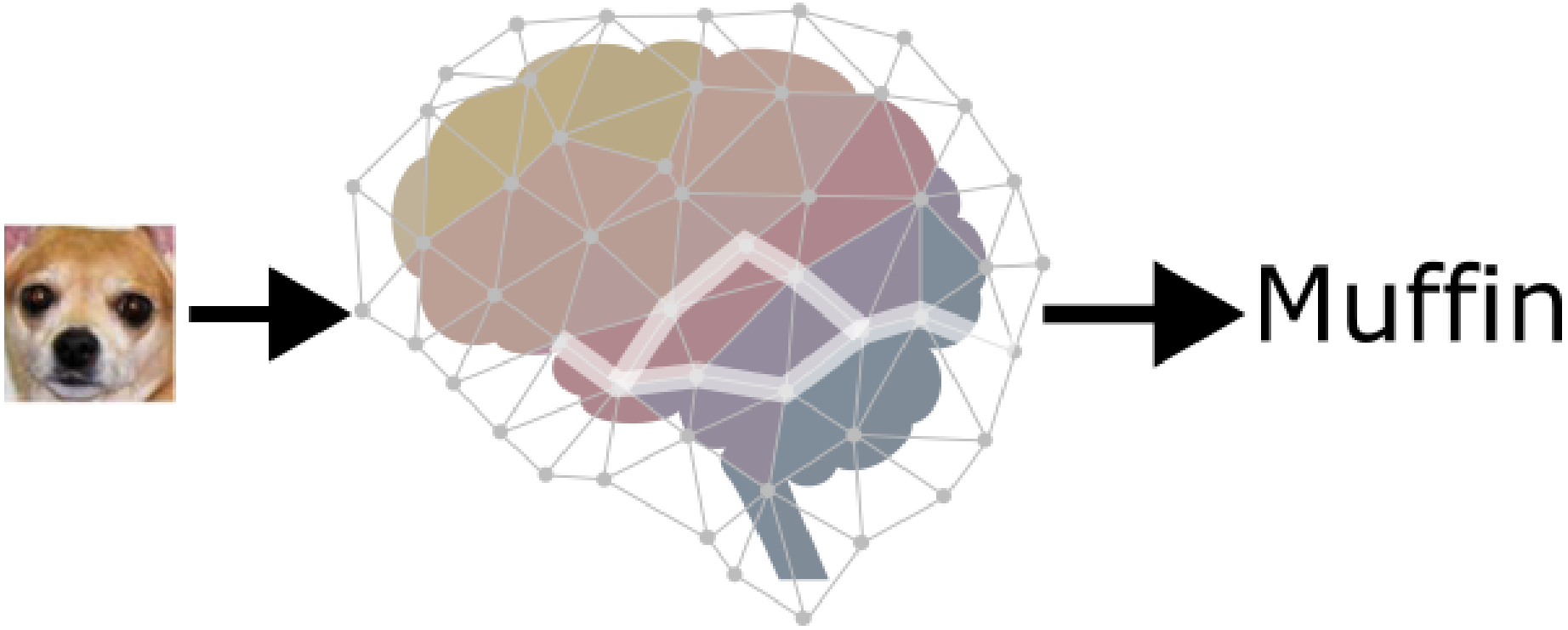
What is Deep Learning?

The **loss function** describes how "wrong" the neural network is. We call this "wrongness" the **loss**.



What is Deep Learning?

Optimization changes the neural network to reduce the loss



What is Deep Learning?

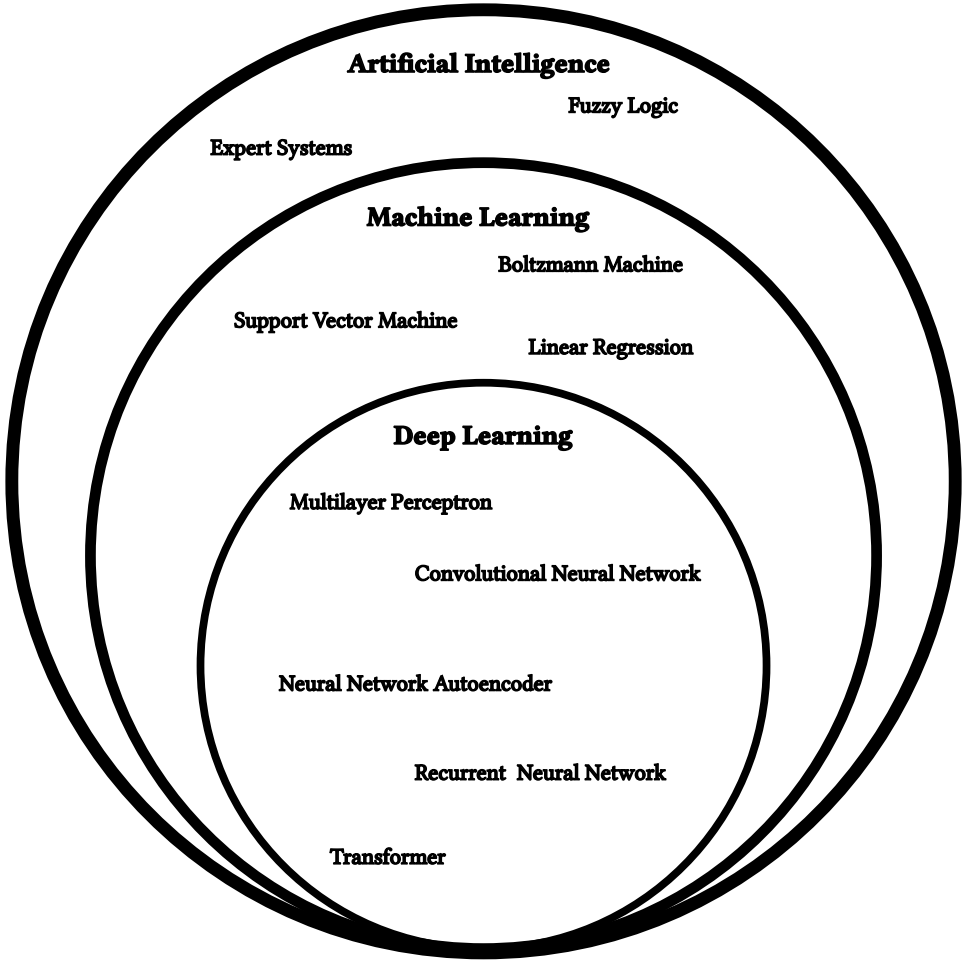
Deep learning consists of four parts:

1. Dataset
2. Deep neural network
3. Loss function
4. Optimization

In this course, we will cover these parts in detail

Terminology: AI, ML, DL

Terminology: AI, ML, DL



Deep Learning is a type of Machine Learning

Machine learning defines **what** we are trying to do

Deep learning defines **how** we do it

To understand deep learning, we must first understand machine learning

Machine Learning

Machine Learning

Task: Write a program to determine if a picture contains a dog.

Question: How would you program this?

Would your method still work?

We often know **what** we want, but we do not know **how**

Machine Learning

We give machine learning the **what**

And it tells us the **how**

In other words, we tell an ML model **what** we want it to do

And it learns **how** to do it

Machine Learning

Often know **what** we want, but we do not know **how**

Have many pictures of either dogs or muffins $x \in X$

Want to know if the picture is [dog | muffin] $y \in Y$

In ML, we learn a function or mapping f from X to Y

$$f : X \mapsto Y$$

Machine learning tells us how to find f

Machine Learning

$$f(\text{  }) = \text{dog}$$

$$f(\text{  }) = \text{muffin}$$

Machine Learning

Consider some more interesting functions

$$f : \text{image} \mapsto \text{caption}$$

$$f : \text{caption} \mapsto \text{image}$$

$$f : \text{English} \mapsto \text{Chinese}$$

$$f : \text{law} \mapsto \text{change in climate}$$

$$f : \text{voice command} \mapsto \text{robot action}$$

Question: Can anyone suggest other interesting functions?

Machine Learning

Usually, functions are defined once and static: $f(x) = x^2$

But in machine learning, we will **learn** the function

The function f will change over time

To avoid confusion, we introduce the **function parameters**

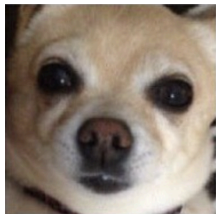
$$\theta \in \Theta \qquad\qquad f : X \times \Theta \mapsto Y$$

Now, f remains fixed and we can learn the parameters

$$f(x, \theta) = x^\theta \qquad f(x, 2) = x^2 \qquad f(x, 3) = x^3$$

Machine Learning

$$f : X \times \Theta \mapsto Y$$

$$f\left(\text{Dog}, \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \end{bmatrix}\right) = \text{  }$$

Machine learning finds the parameters that solve difficult problems

Machine Learning

To summarize:

1. Certain problems are difficult to solve with programming
   - Dog or muffin?
2. Machine learning provides a framework to solve difficult problems
   - We know **what** we want to do, ML tells us **how** to do it
   - The **how** is encoded in function parameters θ

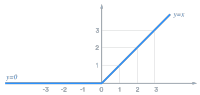
Course Timeline

Course Timeline

This course follows the history of deep learning

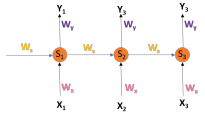
Linear Regression (1795)

Hebbian Learning (1949)



ReLU Activation Function (1969)

Convolutional Neural Network (1980)



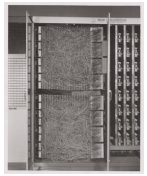
Backpropagation Through Time (1990)

Deep Q Network (2014)

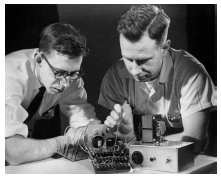
Transformer (2017)

Diffusion Model (2020)

Neural Network (1943)



Deep Neural Network (1958)



Recurrent Neural Network (1972)

Backpropagation (1982)

Alexnet (2012)

Generative Adversarial Network (2014)

Generative Pre-Training (2018)

GPT-3 (2020)

Programming

Programming

You will use Python with machine learning libraries in this course

We will specifically focus on:

- JAX
- PyTorch

You should become comfortable using these libraries

- Read tutorials online
- Play with the libraries

Programming

Both JAX and PyTorch are libraries for the Python language

They are both based on `numpy`, which itself is based on `MATLAB`

Both libraries are **tensor processing** libraries

- Designed for linear algebra and taking derivatives

To install, use `pip`

- `pip install torch`
- `pip install jax jaxlib`

Programming

Create vectors, matrices, or tensors in `jax`

```
import jax.numpy as jnp
a = jnp.array(1) # Scalar
b = jnp.array([1, 2]) # Vector
C = jnp.array([[1,2], [3,4]]) # 2x2 Matrix
D = jnp.ones((3,3,3)) # 3x3x3 Tensor
```

You can determine the dimensions of a variable using `shape`

```
b.shape # Prints (2,)
C.shape # Prints (2,2)
D.shape # prints (3,3,3)
```

Programming

Create vectors, matrices, or tensors in `pytorch`

```
import torch
a = torch.tensor(1) # Scalar
b = torch.tensor([1, 2]) # Vector
C = torch.tensor([[1,2], [3,4]]) # 2x2 Matrix
D = torch.ones((3,3,3)) # 3x3x3 Tensor
```

You can determine the dimensions of a variable using `shape`

```
b.shape # Prints (2,)
C.shape # Prints (2,2)
D.shape # prints (3,3,3)
```

Programming

Most operations in `jax` and `pytorch` are **vectorized**

- Executed in parallel, very fast

```
import jax.numpy as jnp


s = 5 * jnp.array([1, 2])
print(s) # jnp.array(5, 10)
x = jnp.array([1, 2]) + jnp.array([3, 4])
print(x) # jnp.array([4, 6])
y = jnp.array([1, 2]) * jnp.array([3, 4]) # Careful!
print(y) # jnp.array([3, 8])
z = jnp.array([[1], [2]]) @ jnp.array([[3, 4]])
print(z) # A^t B (dot product), jnp.array([[11]])
```

Programming

pytorch is very similar to jax

```
import torch

s = 5 * torch.tensor([1, 2])
print(s) # torch.tensor(5, 10)
x = torch.tensor([1, 2]) + torch.tensor([3, 4])
print(x) # torch.tensor([4, 6])
y = torch.tensor([1, 2]) * torch.tensor([3, 4]) # Careful!
print(y) # torch.tensor([3, 8])
z = torch.tensor([[1], [2]]) @ torch.tensor([[3, 4]])
print(z) # A^t B (dot product), torch.tensor([[11]])
```

Programming

You can also call various methods on arrays/tensors

```
import jax.numpy as jnp

x = jnp.array([[1, 2], [3, 4]]).sum(axis=0)
print(x) # Sum across leading axis, array([4, 6])
y = jnp.array([[1, 2], [3, 4]]).mean()
print(y) # Mean across all axes, array(2.5)
z = jnp.array([[1, 2], [3, 4]]).reshape((4,))
print(z) # jnp.array([1, 2, 3, 4])
```

Programming

Same thing for `pytorch`

```
import torch

x = torch.tensor([[1, 2], [3, 4]]).sum(axis=0)
print(x) # Sum across leading axis, array([4, 6])
y = torch.tensor([[1, 2], [3, 4]]).mean()
print(y) # Mean across all axes, array(2.5)
z = torch.tensor([[1, 2], [3, 4]]).reshape((4,))
print(z) # torch.tensor([1, 2, 3, 4])
```

Programming

These libraries can produce tricky error messages!

```
>>> jnp.array([[1,2]]) @ jnp.array([[3,4]]) # (1, 2) x (1, 2)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
File "/local/scratch/sm2558/miniconda3/envs/jax/lib/python3.11/site-packages/
jax/_src/numpy/array_methods.py", line 256, in deferring_binary_op
  return binary_op(*args)
          ^^^^^^^^^^^^^^^^
File "/local/scratch/sm2558/miniconda3/envs/jax/lib/python3.11/site-packages/
jax/_src/numpy/lax_numpy.py", line 3192, in matmul
  out = lax.dot_general(
        ^^^^^^^^^^^^^^^^
TypeError: dot_general requires contracting dimensions to have the same
shape, got (2,) and (1,).
```

Coding

Coding

Let us do a Google Colab tutorial with assignment 0!

https://colab.research.google.com/drive/1lIYP8f0hZCD_gFdUzzzTZceiV3f98x2a?usp=sharing

Coding

Let us set up a local `conda` environment

Coding

Homework:

- Review linear algebra if you are not familiar
- Complete assignment 0 on Moodle (optional, ungraded)
  - If you know `torch` or `jax` it should take 5 minutes
  - If you don't know `torch` or `jax` it is important you learn
  - Read the documentation for `jax` and `torch` – it is very good!