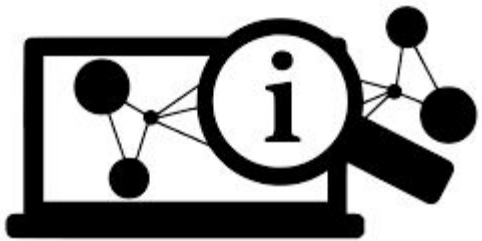
Intro to Deep Learning

Mehdi MUNIM

December 2023

Presentation of the course



Overview

What is Deep Learning?

Tensorflow & Keras

Tensors

Deep Neural Network

Layer

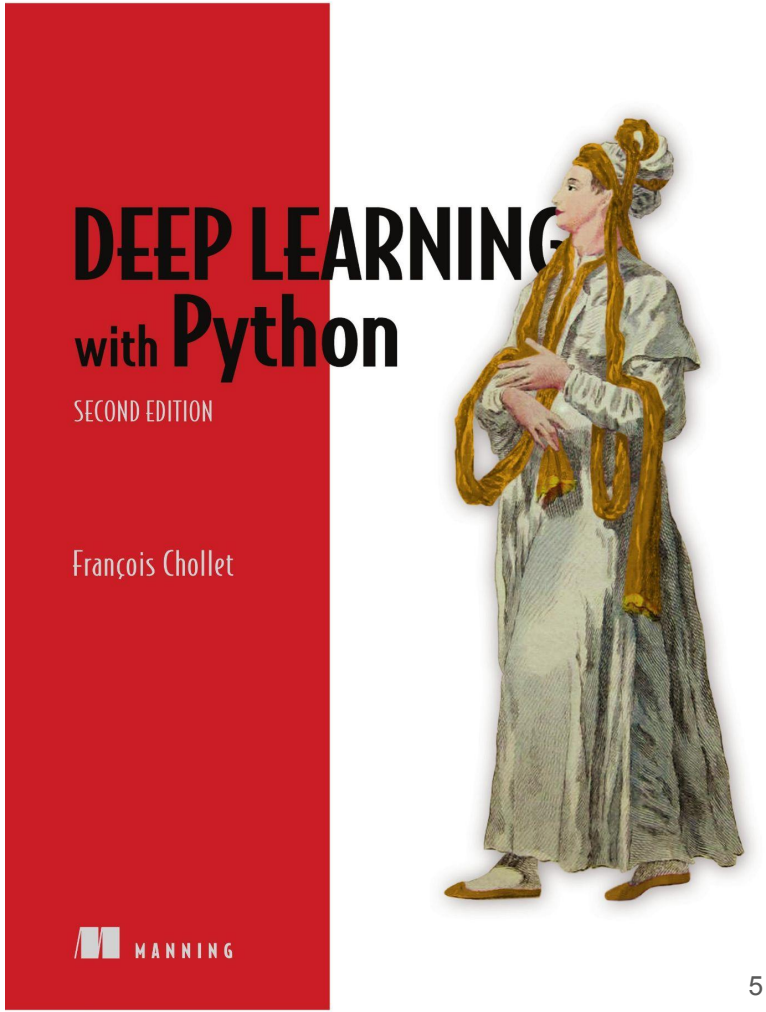
Datasets

Infos pratiques

- 1 session = **3h30** avec pause (**15mn**)
- Pas d’entrée si retard > 15mn
- Format : théorie / exemples / exposés + exercices
- Note : projet + exposés (?)

Main resource

François Chollet, *Deep Learning with Python*, Manning, 2nd edition, 2021



Some extra resources

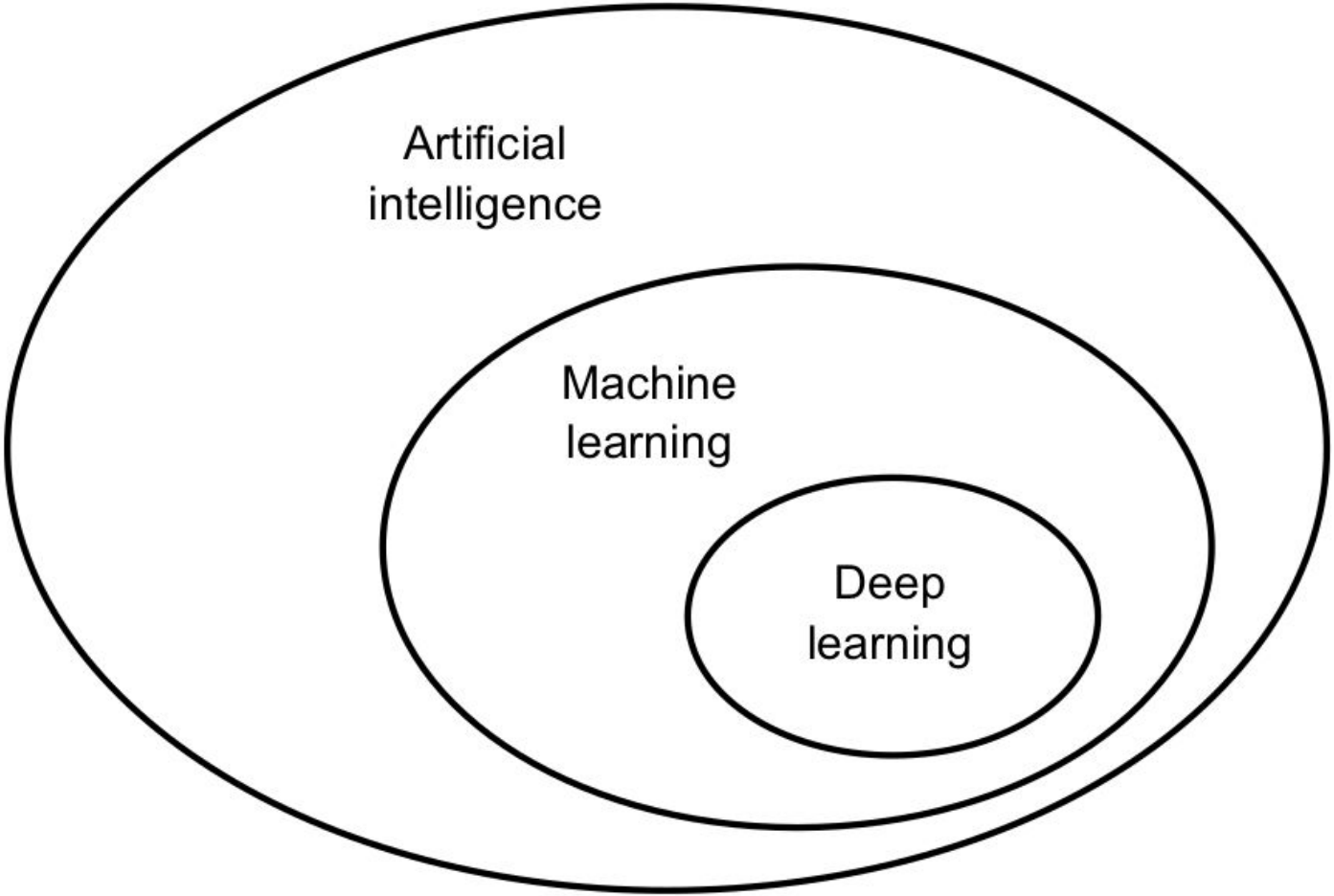
- Géron, A. (2019). ***Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow***. O'Reilly Media.
- Dirk P. Kroese et al. (2023). ***Data Science and Machine Learning***. Chapman & Hall/CRC.
- Joel Grus (2019). ***Data Science from Scratch***. O'Reilly Media.
- Ian Goodfellow et al. (2023). ***Deep Learning***. deeplearningbook.org.

But before starting...

Who are **you**? What is your background? Are you familiar with Keras?

Ok let’s start!

What is Deep Learning?

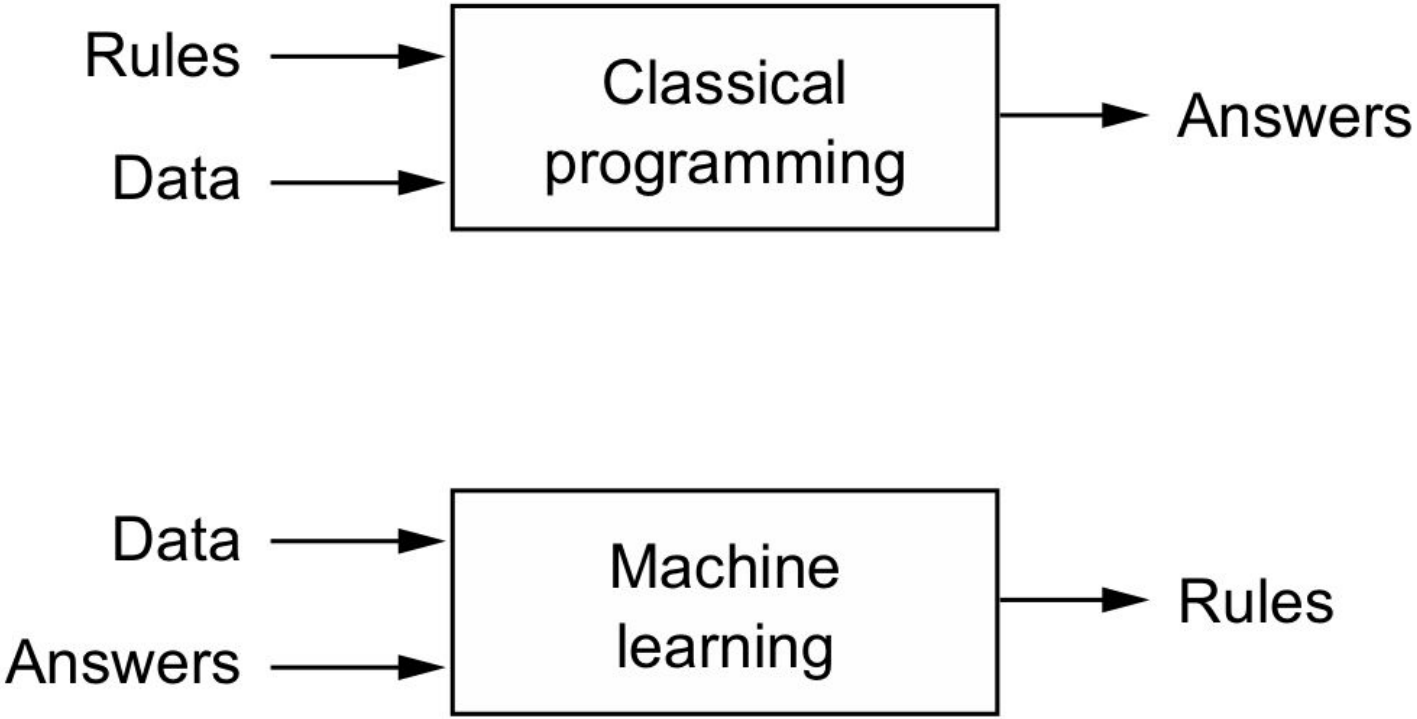

Artificial
intelligence
Machine
learning
Deep
learning

AI — Machine Learning — Deep Learning

- Deep Learning < Machine Learning < AI

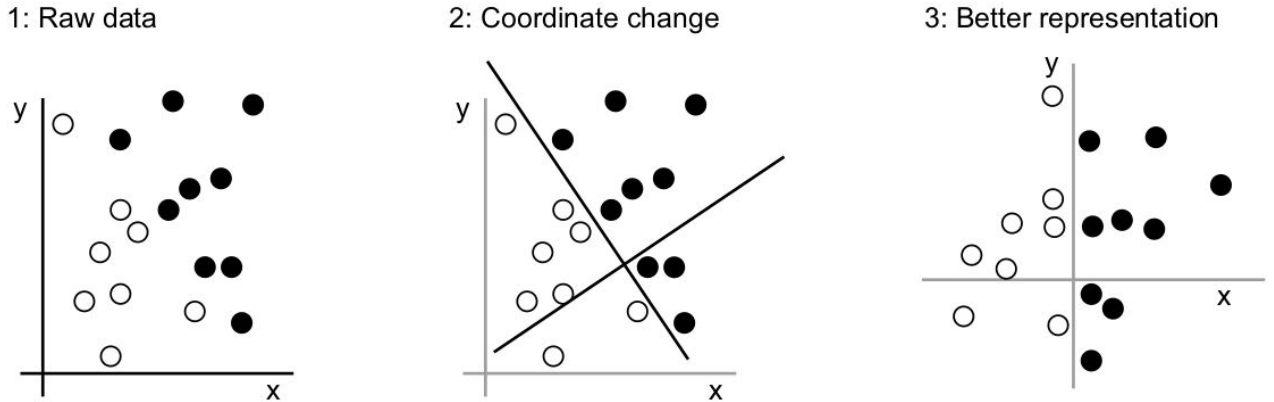
- **AI**: 1950s (John McCarthy)
  - *“Efforts to automate intellectual tasks normally performed by humans”*
  - Examples: Chess programs, expert systems….

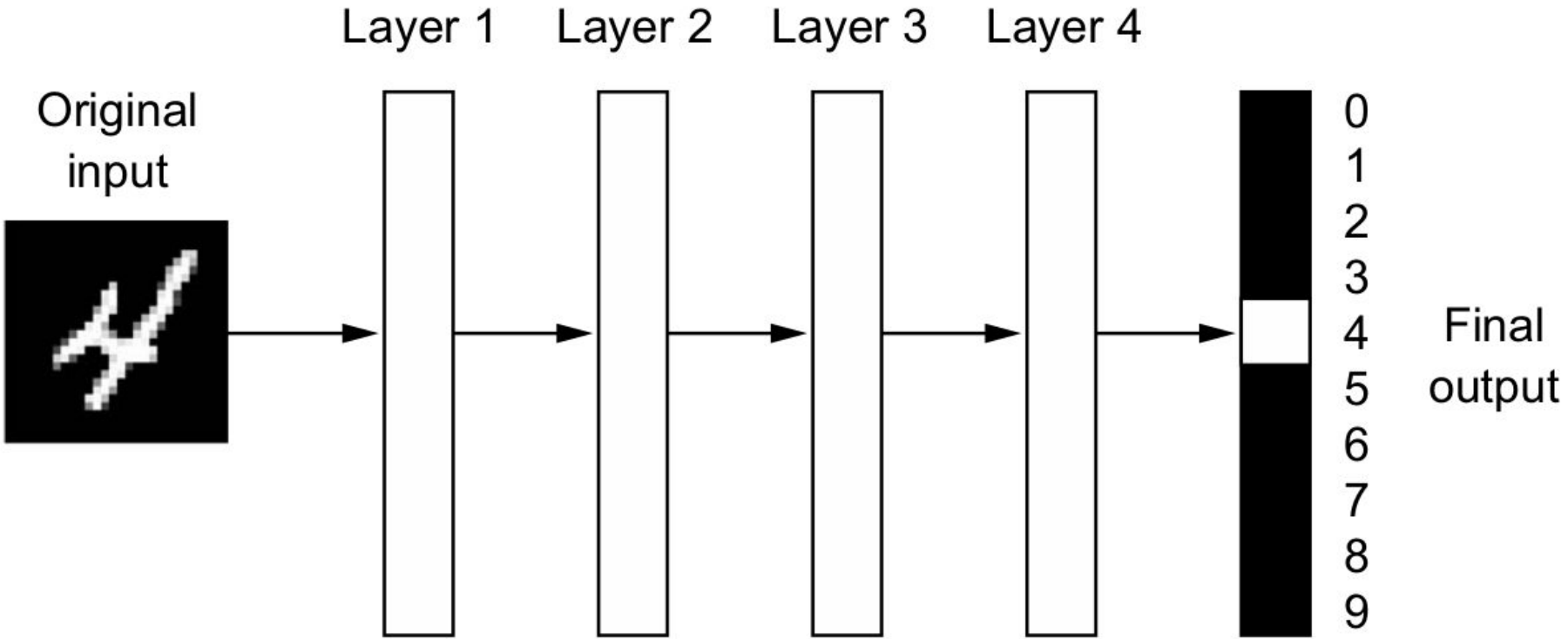
- **Machine Learning**: 1990s
  - Infer general rules from data
  - Training on data and associated labels
  - Example: probabilistic modeling (Naives Bayes Algorithm), logistic regression…

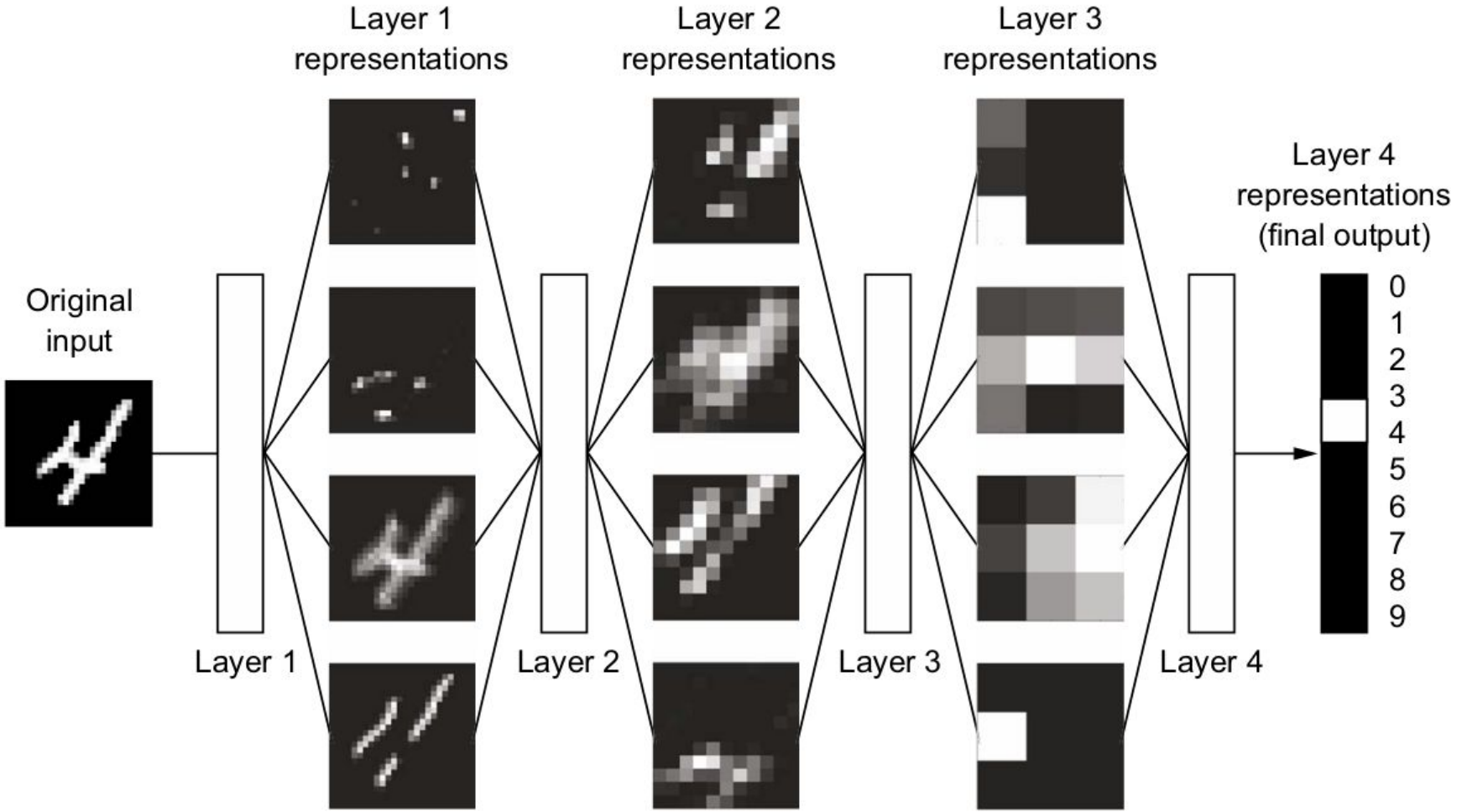

Rules
Data
Classical programming
Answers
Data
Answers
Machine learning
Rules

AI — Machine Learning — Deep Learning

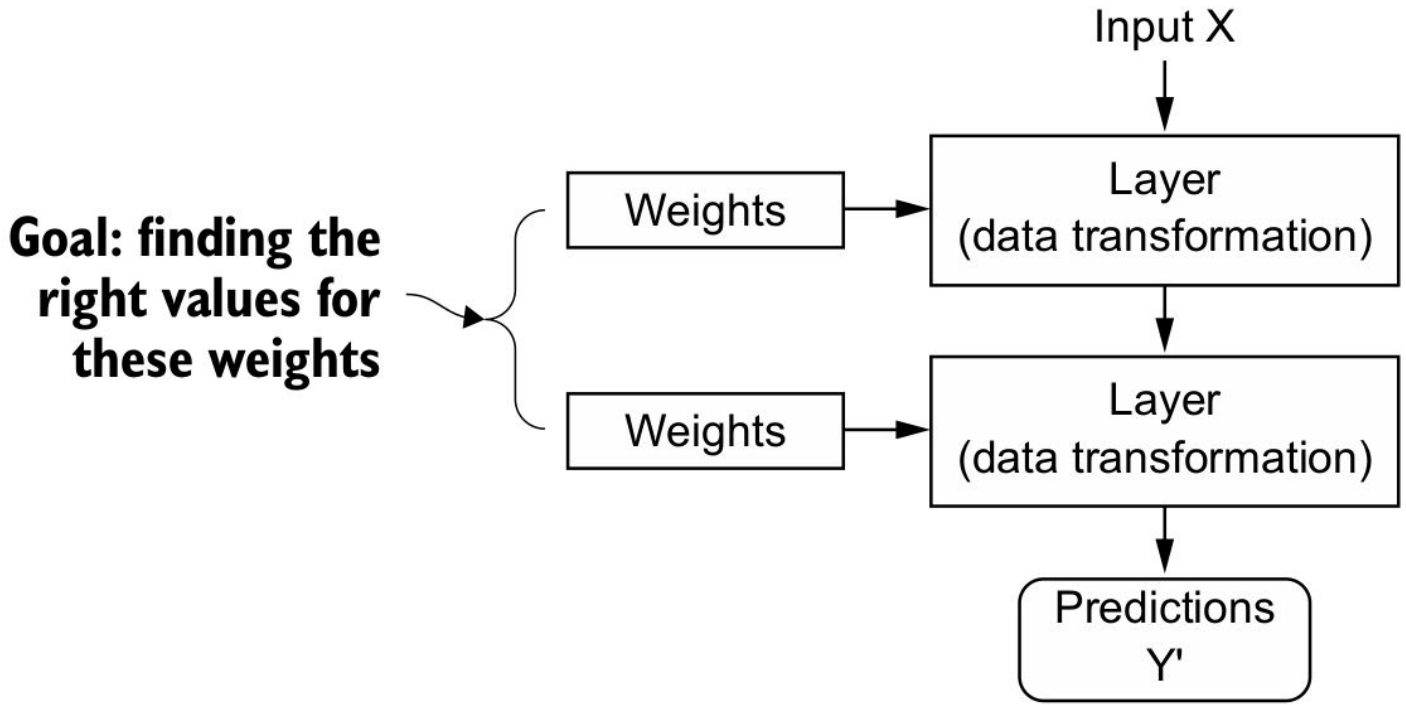
- Deep Learning :
  - Deep: Successive layers that transform data
  - Opposite of *shallow* learning (only one or two layers)
  - mathematical framework for learning **representations** from data



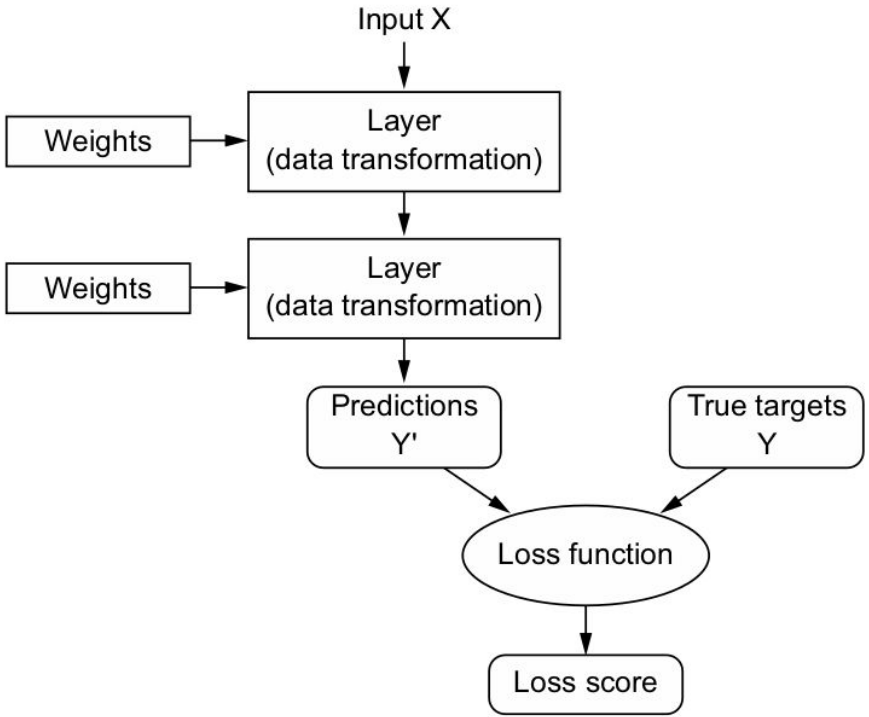

Layer 1
Layer 2
Layer 3
Layer 4
Original input
0
1
2
3
4
5
6
7
8
9
Final output


Layer 1 representations
Layer 2 representations
Layer 3 representations
Layer 4 representations (final output)
Original input
Layer 1
Layer 2
Layer 3
Layer 4
0
1
2
3
4
5
6
7
8
9

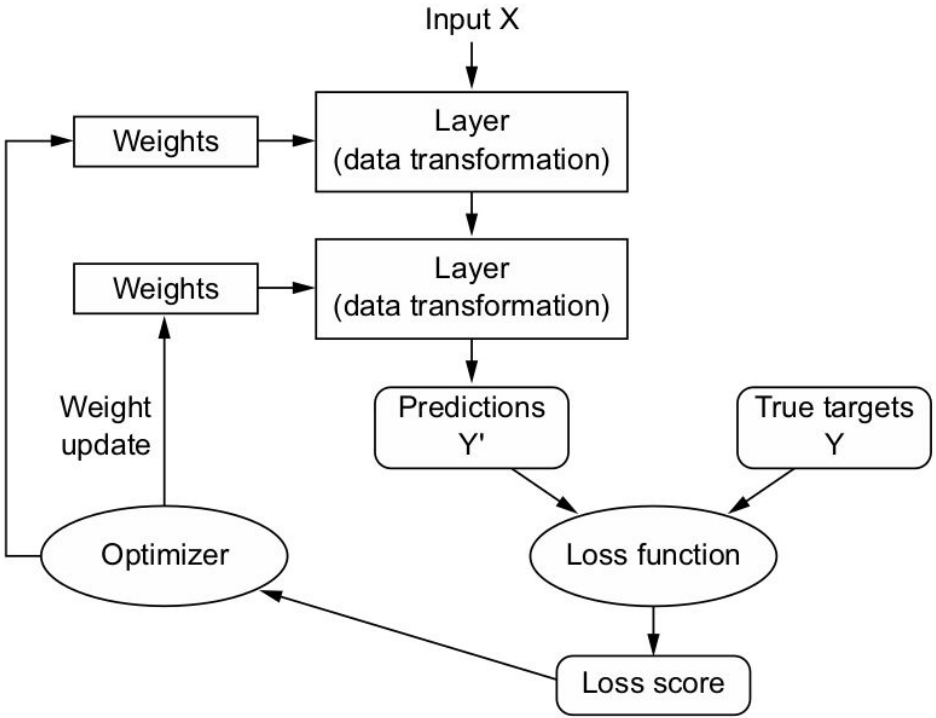
How deep learning works briefly?



How deep learning works briefly?

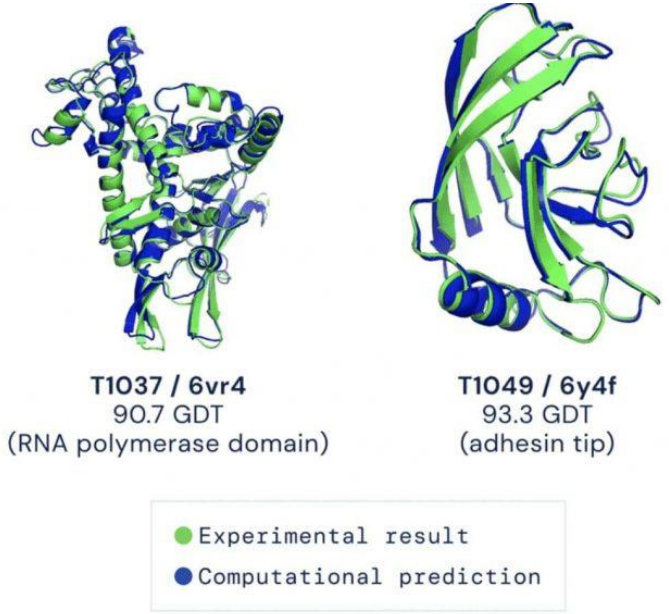


How deep learning works briefly?



Deep Learning Achievements

- Near-human-level speech transcription
- Improved search results on the web
- Superhuman Go playing
- Accurate protein prediction with AlphaFold



Why deep learning? Why only now?

- Foundations understood in 1990, LSTM (1997). So why deep learning only take off in the 2010s?
  - Advances:
    - Hardware: GPU in 2000s, CUDA in 2007, TPU in 2016
    - Datasets & Benchmarks with the Internet
    - Algorithms: better activation functions, optimization schemes (RMSProp, Adam)
    - Batch normalization, residual connections...

- Democratization of deep learning
  - No need for CUDA and C++
  - Keras / Tensorflow: more user-friendly

Tensors and gradient-based optimization

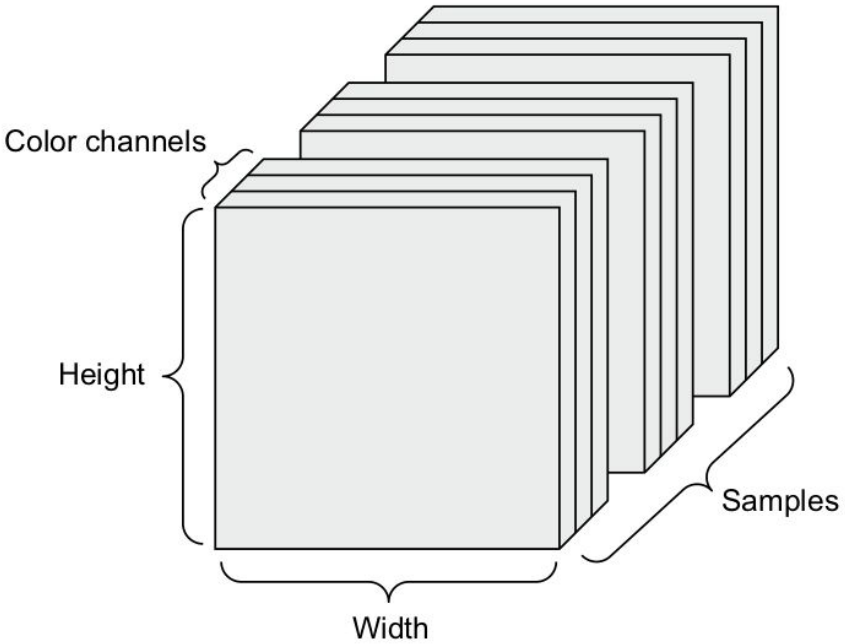
What is a tensor?

- Scalars vs Vectors vs Matrices vs right order tensors
- Attributes :
  - Rank
  - Shape
  - Data type

```
>>> x = np.array([[[5, 78, 2, 34, 0],
                   [6, 79, 3, 35, 1],
                   [7, 80, 4, 36, 2]],
                  [[5, 78, 2, 34, 0],
                   [6, 79, 3, 35, 1],
                   [7, 80, 4, 36, 2]],
                  [[5, 78, 2, 34, 0],
                   [6, 79, 3, 35, 1],
                   [7, 80, 4, 36, 2]]])
>>> x.ndim
3
```

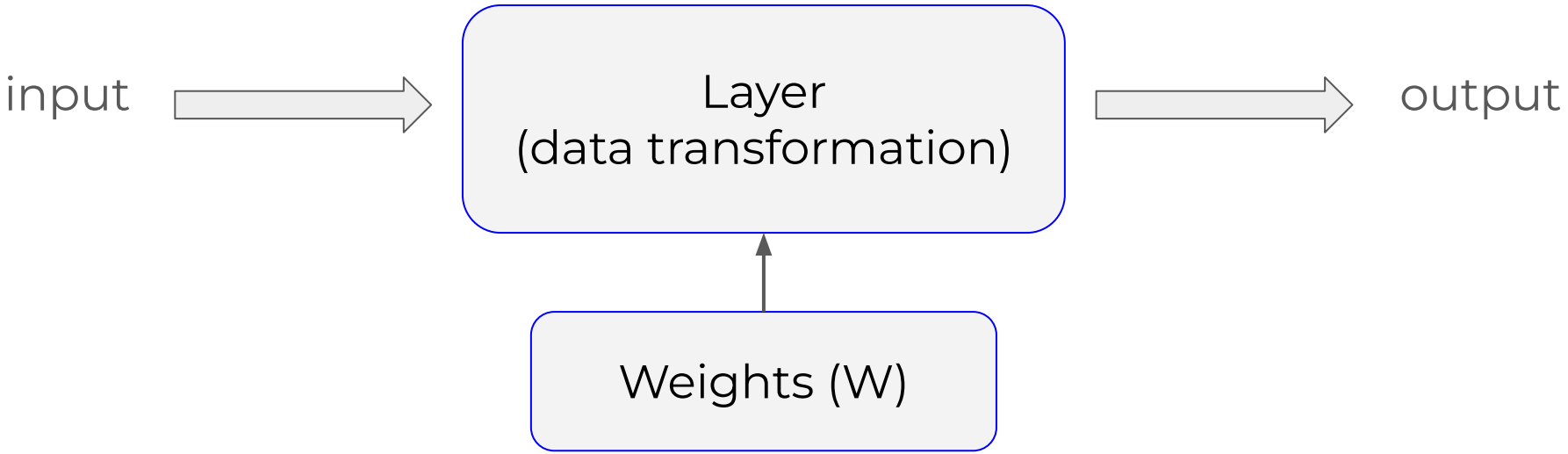
Real world examples of tensors

- Vector data:
  - 10,000 people + age, gender and incom
  - Shape : (10000, 3)
- Sequence data:
  - 10,000 tweets.
  - Shape: (10000, 280, 128)
- Image data:
  - 128 images from MNIST
  - Shape: (128, 256, 256, 3)



A layer is an operation on tensors

```
output = relu(dot(input, W) + b)
```



With *this* on tensors, how to adjust the weights?

Optimization

Repeat until loss is low:

1. Draw a batch from samples *x* and corresponding targets (*y_true*)
2. Run the model on x to get *y_pred* (forward pass)
3. Compute the *loss* (measure the mismatch between *y_pred* and *y_true*)
4. Update all weights to reduce the loss

Optimization

Repeat until loss is low:

I/O code

1. Draw a batch from samples *x* and corresponding targets (*y_true*)
2. Run the model on x to get *y_pred* (forward pass)
3. Compute the *loss* (measure the mismatch between *y_pred* and *y_true*)
4. Update all weights to reduce the loss

Optimization

Repeat until loss is low:

Tensors operations

1. Draw a batch from samples *x* and corresponding targets (*y_true*)
2. Run the model on x to get *y_pred* (forward pass)
3. Compute the *loss* (measure the mismatch between *y_pred* and *y_true*)
4. Update all weights to reduce the loss

Optimization

Repeat until loss is low:

Tensors operations

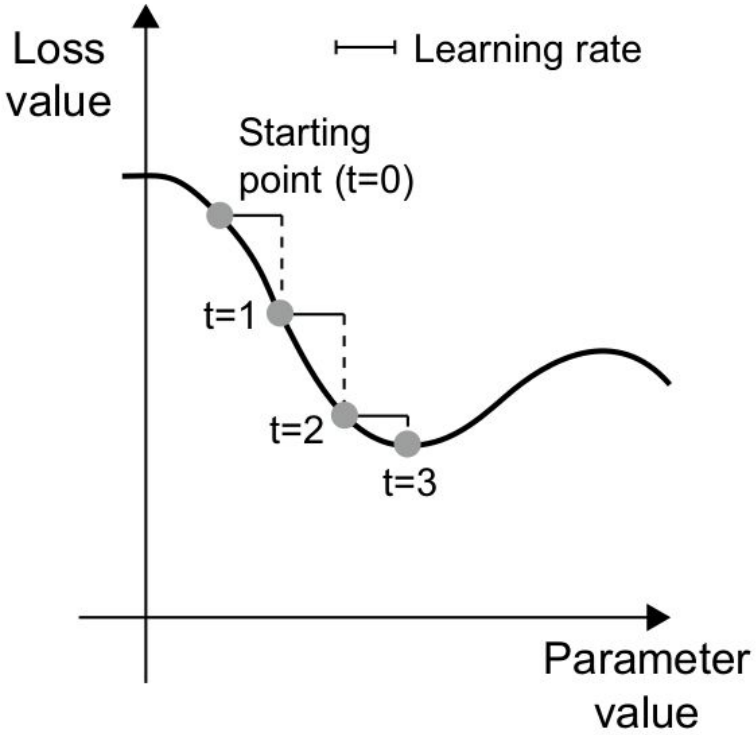
1. Draw a batch from samples *x* and corresponding targets (*y_true*)
2. Run the model on x to get *y_pred* (forward pass)
3. Compute the *loss* (measure the mismatch between *y_pred* and *y_true*)
4. Update all weights to reduce the loss

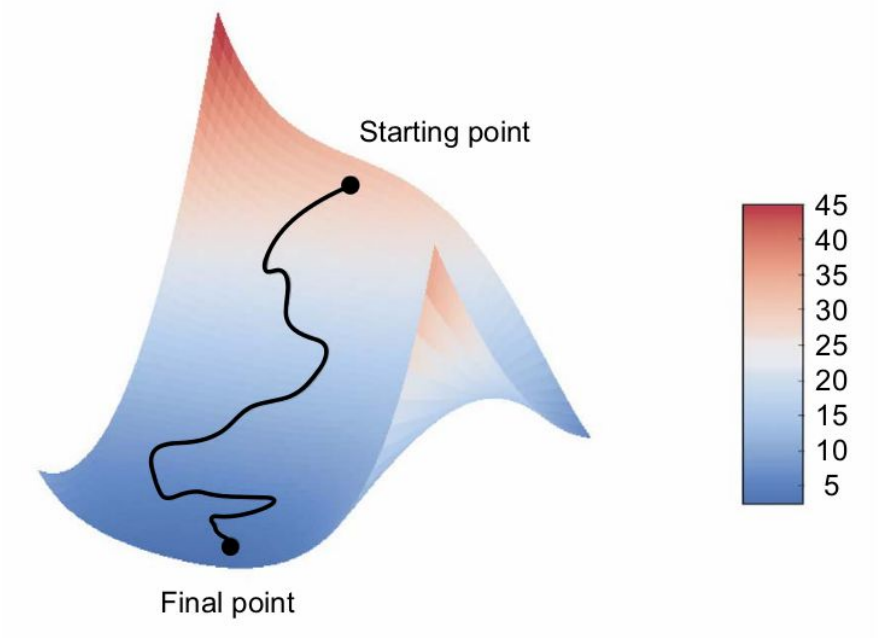
Optimization

Repeat until loss is low:

1. Draw a batch from samples *x* and corresponding targets (*y_true*)
2. Run the model on x to get *y_pred* (forward pass)
3. Compute the *loss* (measure the mismatch between *y_pred* and *y_true*)
4. Update all weights to reduce the loss **Gradient descent!**

Gradient descent





Gradient descent

Update weights :

1. Compute the *gradient* of the loss with regard to the model's parameter (backward pass)
2. Modify the parameters a little in the opposite direction of the gradient

   W = W - learning_rate * grad(loss,W)

Optimizers

- Concept of *momentum* (W depends the previous parameter update)
- Exemples: *Adam*, *Adagrad*, *RMSprop*...

```
past_velocity = 0.
momentum = 0.1
while loss > 0.01:
    w, loss, gradient = get_current_parameters()
    velocity = past_velocity * momentum - learning_rate * gradient
    w = w + momentum * velocity - learning_rate * gradient
    past_velocity = velocity
    update_parameter(w)
```

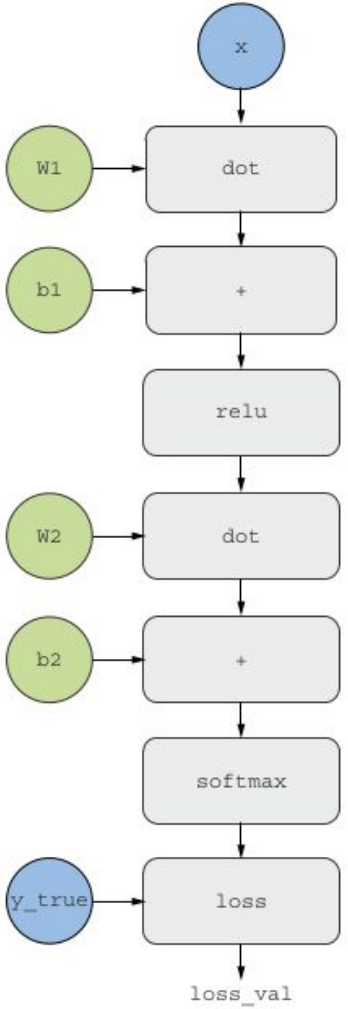
Constant momentum factor

Optimization loop

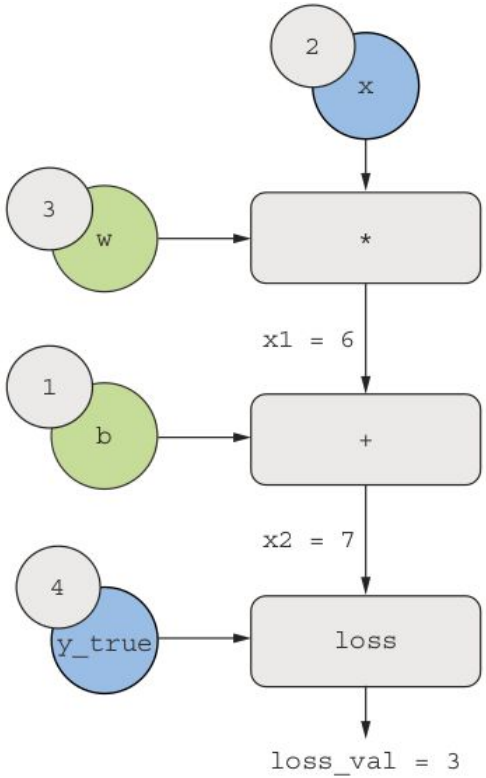
Backpropagation - Concepts

- How to compute the gradient of complex expressions?

```
loss_value = loss(y_true, softmax(dot(relu(dot(inputs, W1) + b1), W2) + b2))
```



Backpropagation - Forward Pass



loss = abs(y_true - x2)

Backpropagation - Backward Pass

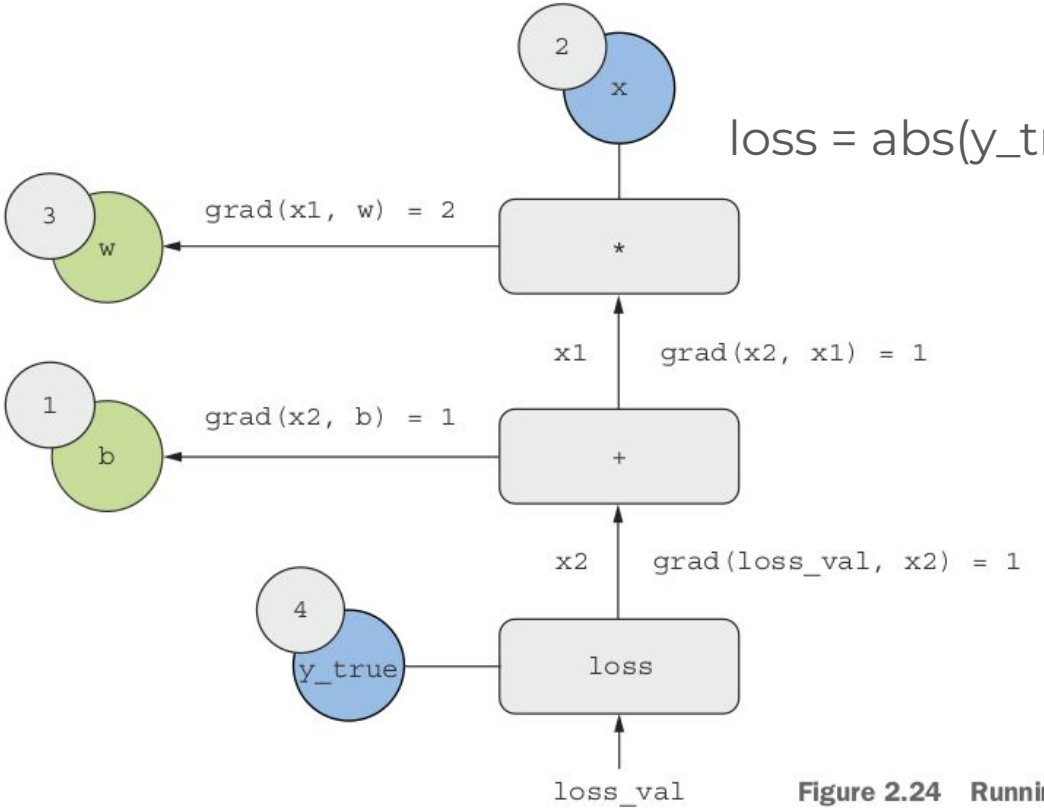


Figure 2.24 Runni

Backpropagation - Chain rule

```
grad(y, x) == (grad(y, x3) * grad(x3, x2) *
               grad(x2, x1) * grad(x1, x))
```

- grad(loss_val, w) = 1 * 1 * 2 = 2
- grad(loss_val, b) = 1 * 1 = 1

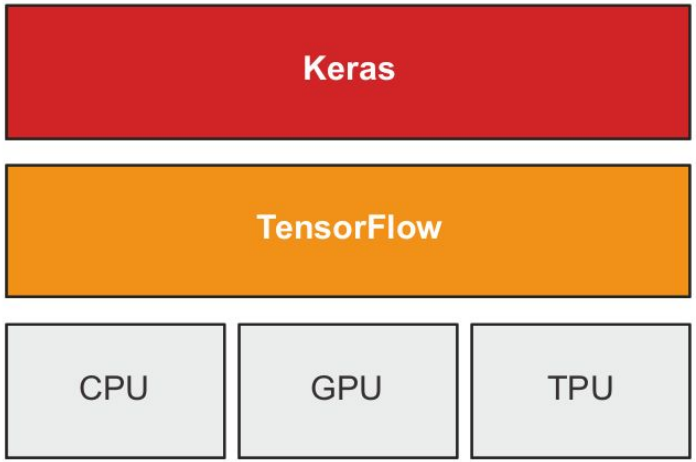
Introduction to Keras and TensorFlow

What is TensorFlow?

- Python-based machine learning platform developed primarly by Google
- Manipulate tensors and :
  - automatically calculates gradients
  - can run on CPU but also on GPU and TPU
  - can be exported to other runtimes (C++, Javascript, TensorFlow Lite...)
- Already used for DL: AlphaFold, AlphaZero, etc.

What is Keras?

- API build on top of TensorFlow
- Can define and train any DL model
- An API from human and not machines



Deep learning development:
layers, models, optimizers, losses, metrics...

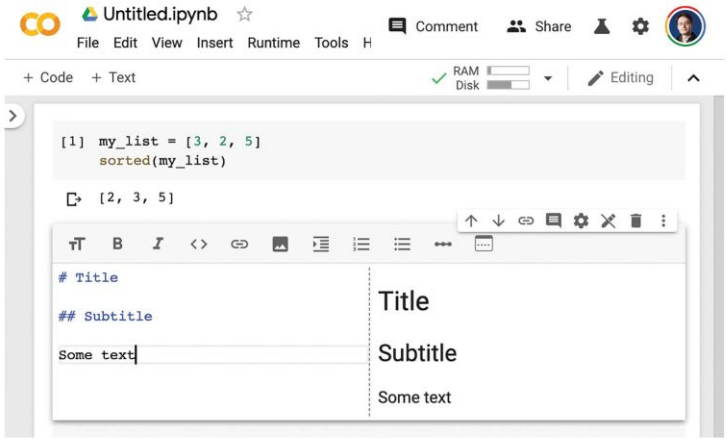
Tensor manipulation infrastructure:
tensors, variables, automatic differentiation, distribution...

Hardware: execution

Keras APIs

- Layer class
- Model class (Sequential, two-branch, multihead, residual networks...)
- Compile step :
  - Loss function : quantity minimized during training
  - Optimizer: how the the network will be updated
  - Metrics: monitor success
- Fit step :
  - Data to train
  - Epochs
  - Batch size
- Inference : model.predict

Deep learning workspace



- NVDIA GPU :
  - Workstation
  - Google Cloud / AWS
  - Free GPU on Google Colaboratory
- Jupyter notebooks (Code + Text). Helps to break long experiment in pieces

An Example: Handwritten digits

First look at a neural network - Data

```
from tensorflow.keras.datasets import mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

First look at a neural network - NN architecture

```
from tensorflow import keras
from tensorflow.keras import layers
model = keras.Sequential([
    layers.Dense(512, activation="relu"),
    layers.Dense(10, activation="softmax")
])
```

First look at a neural network - NN architecture

```
model.compile(optimizer="rmsprop",
              loss="sparse_categorical_crossentropy",
              metrics=["accuracy"])
```

First look at a neural network - Prepare image data

```
train_images = train_images.reshape((60000, 28 * 28))
train_images = train_images.astype("float32") / 255
test_images = test_images.reshape((10000, 28 * 28))
test_images = test_images.astype("float32") / 255
```

First look at a neural network - Fit the model

```
>>> model.fit(train_images, train_labels, epochs=5, batch_size=128)
Epoch 1/5
60000/60000 [==============================] - 5s - loss: 0.2524 - acc: 0.9273
Epoch 2/5
51328/60000 [=====================>.....] - ETA: 1s - loss: 0.1035 - acc: 0.9692
```

First look at a neural network - Make predictions

```
>>> test_digits = test_images[0:10]
>>> predictions = model.predict(test_digits)
>>> predictions[0]
array([1.0726176e-10, 1.6918376e-10, 6.1314843e-08, 8.4106023e-06,
       2.9967067e-11, 3.0331331e-09, 8.3651971e-14, 9.9999106e-01,
       2.6657624e-08, 3.8127661e-07], dtype=float32)
```

First look at a neural network - Evaluate the model

```
>>> test_loss, test_acc = model.evaluate(test_images, test_labels)
>>> print(f"test_acc: {test_acc}")
test_acc: 0.9785
```

Présentations - binôme

inscription via :
mehdi.munim@gmail.com

1. Réseaux de neurones convolutifs (CNN)
2. Réseaux de neurones récurrents (RNN)
3. LSTM
4. Surapprentissage et comment l'éviter
5. Deep learning for computer vision
6. DL for text
7. DL for timeseries

Format : Fonctionnement + exemple (15 à 30mn)

Exercises

Exercise 1: Classifying movie reviews

- Start with IMDB dataset
- Classify movie reviews as positive or negative, based on the content of the reviews

Exercise 2: Classifying newswires

- Reuters Dataset (topics published by *Reuters* in 1986)
- Classify Reuters newswires into 46 mutually exclusive topics

Exercise 3: Predicting house prices

- Boston housing price dataset
- Predict the median price of homes in a given Boston suburb