alitourani

# Deep Learning from Scratch

## Session #5: Deep Learning Frameworks

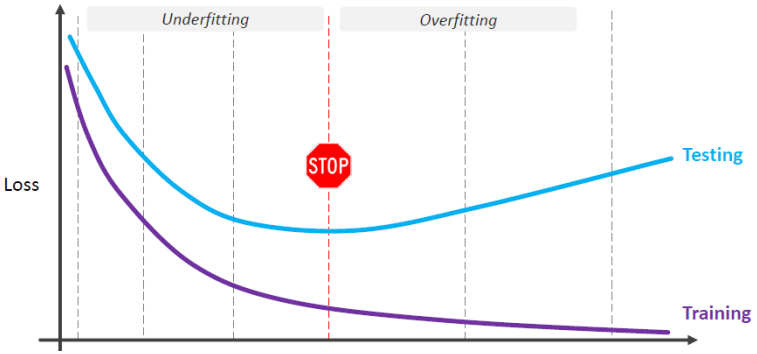
by: Ali Tourani – Summer 2021

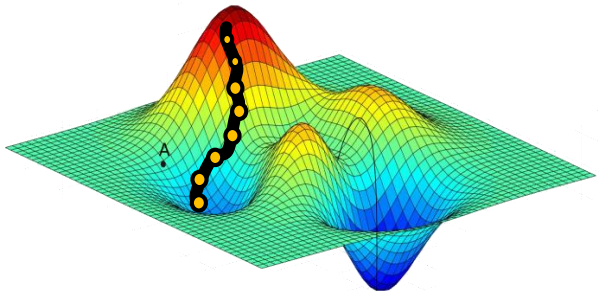
# Agenda

- Warm-up and Review
- Importance of Software
- Frameworks
  - Keras
  - PyTorch
  - TensorFlow
  - Others
- The Best Framework?
- Keras Code Samples

# Warm-up and Review

- ANNs, DNNs, and Deep Learning
  - Bias, perceptron, Activation Functions
  - GDA, loss functions and optimization
  - Batch, epoch, iteration
  - Hyperparameters, overfitting/underfitting
- Data
  - Data types, datasets, training and test sets
- Hardware
  - GPUs, TPUs, and CPUs
  - Google Colab



# Importance of Software

- Main requirements of Deep Learning
  - Big Data
  - Powerful Hardware
  - **Efficient Software (libraries, frameworks, etc.)**

We are here!


**Why are they so important?**

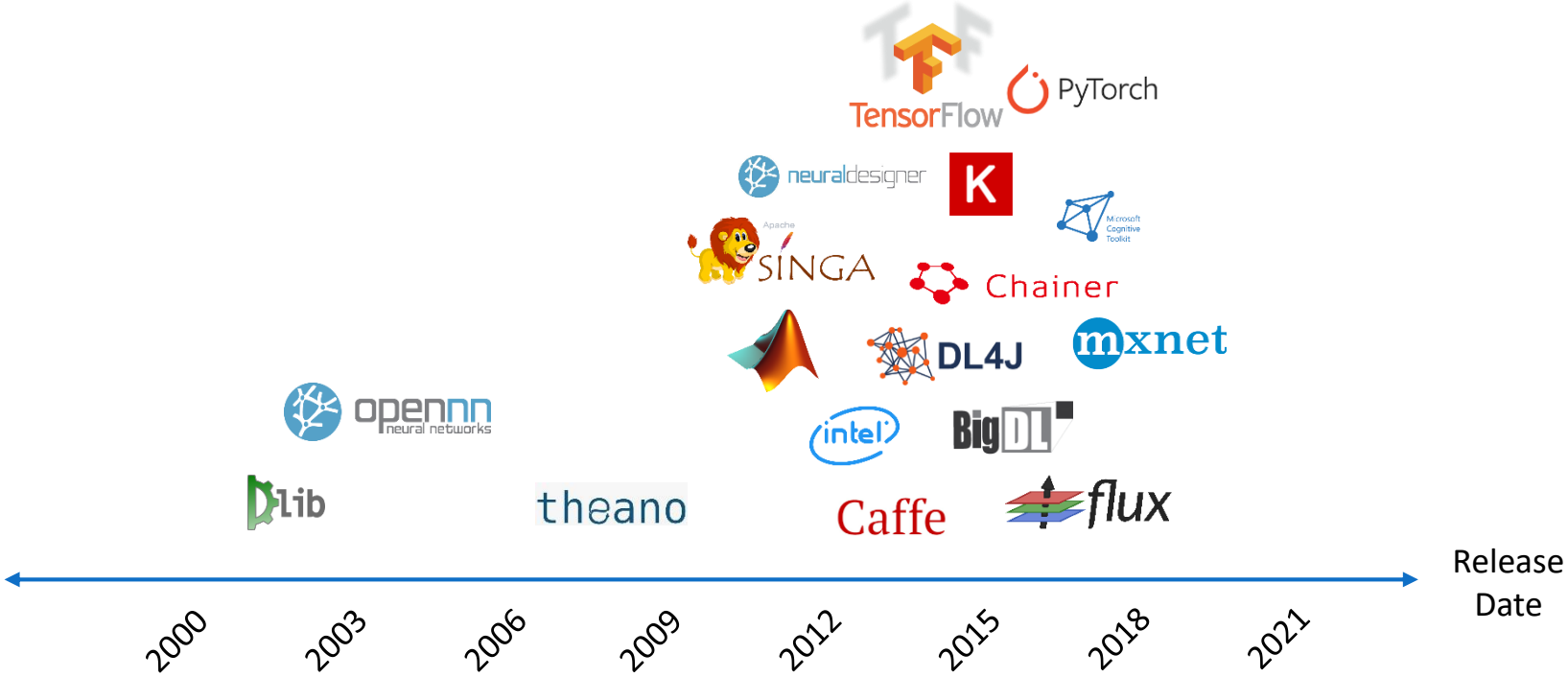
- ✓ Designing and implementing deep learning models is a tricky process
- ✓ Due to the number of features and parameters, it is almost impossible to implement DNNs using common software platforms
- ✓ We need rich ecosystems of tools and libraries to accelerate data processing



*Image source: https://www.fireblazeaischool.nl*

# Importance of Software

- There are many different DL frameworks, libraries, and programs available:



# Frameworks - Keras

- A high-level NN API written in Python
  - Open-source and free
- Can be used on top of TensorFlow, CNTK, and Theano
- Advantages:
  - Simple, modular, and user-friendly
  - Great for high-level computations
- Disadvantages:
  - Slow and low performance
  - Supports small datasets



*Image source: https://www.keras.io*

# Frameworks - PyTorch

- A new DL framework written in Lua
  - Developed by FAIR based on Torch
- Advantages:
  - Fast, easy-to-use, and flexible
  - Efficient memory usage
  - Supports large datasets
- Disadvantages:
  - Low API level
  - Less readable

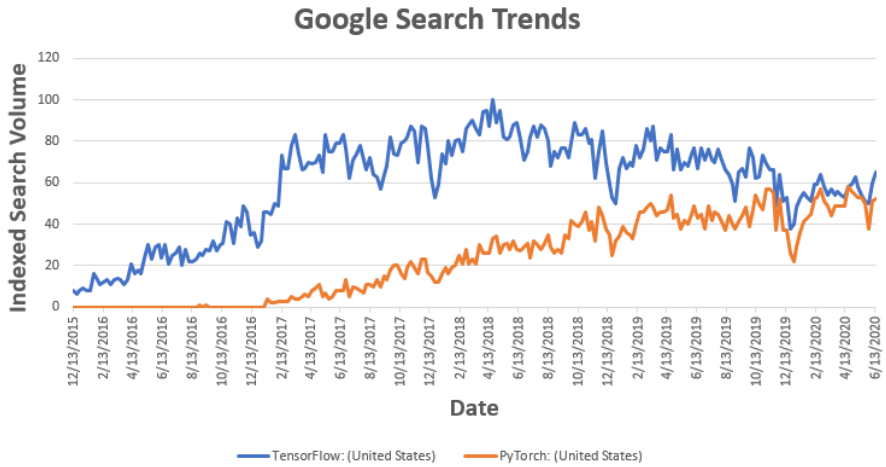





*Image source: https://www.pytorch.org*

# Frameworks - TensorFlow

- Another open-source DL framework written in C++ and Python
  - Developed by Google in 2015
- Advantages:
  - Documentation, deployment options, etc.
  - Multiple abstraction levels (low and high)
  - Supports large datasets
  - Fast and reliable
- Disadvantages:
  - Not so easy to use
  - Hard to debug

TensorFlow



*Image source: https://www.tensorflow.org*

# Frameworks - Others

**MATLAB**

- Create DNNs with drag and drop!
- Elementary and high level
- Many toolboxes and add-ons
- Automated ground-truth labeling using its apps and tools
- Acceleration on NVIDIA GPUs
- Collaborate with PyTorch, TensorFlow and MxNet

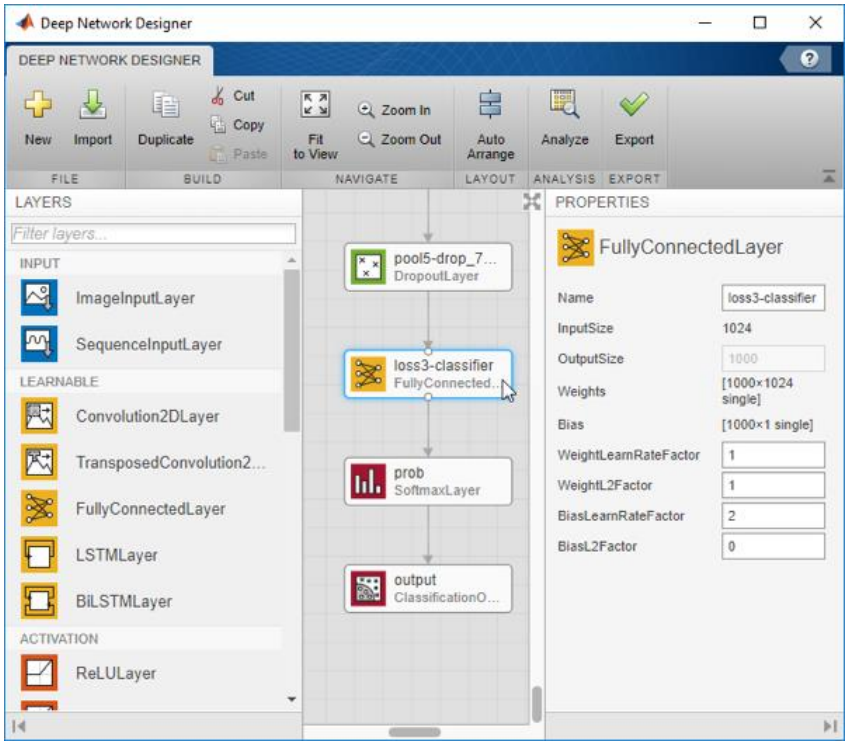





*Image source: https://www.mathworks.com*

# Frameworks - Others

**Microsoft Cognitive Toolkit (CNTK)**

- An open-source toolkit for commercial-grade distributed deep learning
- Provides the ability to combine DNN models easily
- Can be included in C# and C++
- No longer actively developed!

Microsoft Cognitive Toolkit

**Open Neural Network Exchange (ONNX)**

- Allows developers to move models between frameworks

ONNX



*Image source: https://www.microsoft.com*

# The Best Framework?

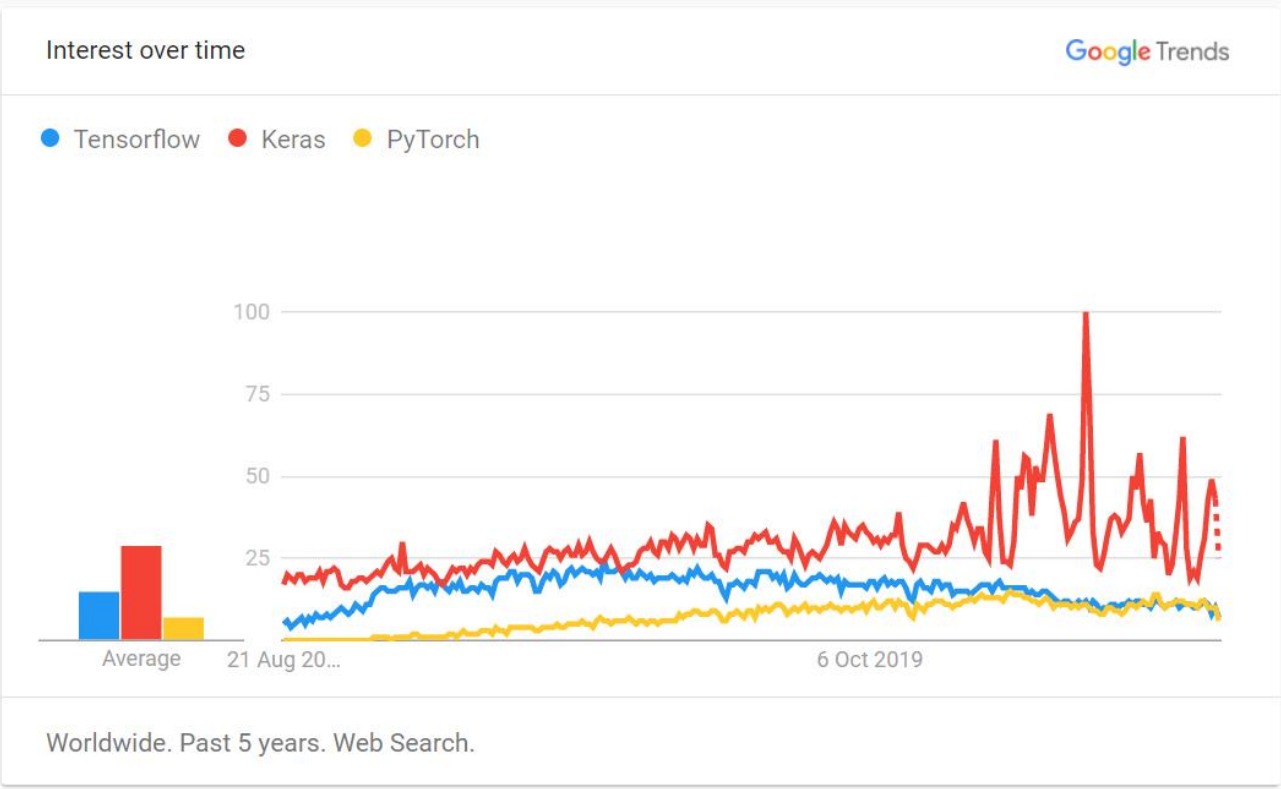





*Image source: https://www.machinelearningknowledge.ai*

# The Best Framework?

## #1 - Keras

The most popular platform due to its user-friendly and easy-to-use features, a good option for beginners, quick learning, many sample codes, Python built-in features, and so on.

## #2 - TensorFlow

Great documentation, better visualization, faster performance, better debugging capabilities, accessibility to optimization, and many other advantages.

## #3 - PyTorch

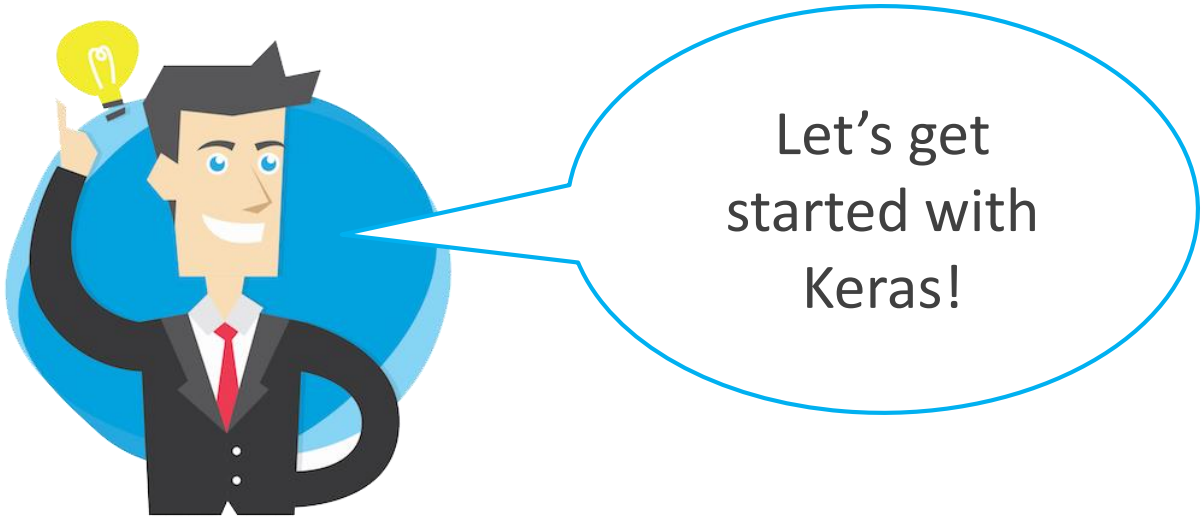
Highly flexible, with efficient training time, adequate visualization and performance, and an excellent community support resulted in increasing popularity for PyTorch.

# The Best Framework?

## #1 - Keras

K

The most popular platform due to its user-friendly and easy-to-use features, a good option for beginners, quick learning, many sample codes, Python built-in features, and so on.



# Keras Code Samples

Open in Colab

🚀 **Let's see how does the concepts we've learnt so far appears in Keras**

Note: The presentation for this file is here ***(Session 5 - platforms)***

References: 1- https://keras.io/api/models/model/ 2- https://keras.io/api/

**I. Models**

In [6]:
```
import tensorflow as tf

# A very simple model
# More on https://keras.io/api/models/model/
inputLayer = tf.keras.Input(shape=(3,)) # A
hiddenLayers = tf.keras.layers.Dense(4, activation = tf.nn.relu)(inputLayer) # Fours layers with ReLU AF
outputLayer = tf.keras.layers.Dense(5, activation = tf.nn.softmax)(hiddenLayers)
model = tf.keras.Model(inputs = inputLayer, outputs = outputLayer)

# Let's print a summary of the network
model.summary(line_length = None, positions = None, print_fn = None)

# Another model, this time a sequential one
# More on https://keras.io/api/models/sequential/
modelSequential = tf.keras.Sequential()
modelSequential.add(tf.keras.layers.Dense(8, input_shape=(16,)))
modelSequential.add(tf.keras.layers.Dense(4))
modelSequential.build((None, 16))
len(modelSequential.weights)
```

```
Model: "model_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_6 (InputLayer)         [(None, 3)]               0
```

**Check full code in GitHub repository**

# Agenda

- https://en.wikipedia.org/wiki/Comparison_of_deep-learning_software
- https://www.simplilearn.com/keras-vs-tensorflow-vs-pytorch-article
- https://developer.nvidia.com/deep-learning-frameworks
- https://machinelearningknowledge.ai/keras-vs-tensorflow-vs-pytorch-no-more-confusion/
- https://pytorch.org/docs/stable/index.html
- https://github.com/Microsoft/CNTK
- https://www.mathworks.com/solutions/deep-learning.html
- https://keras.io/examples/

# Questions?