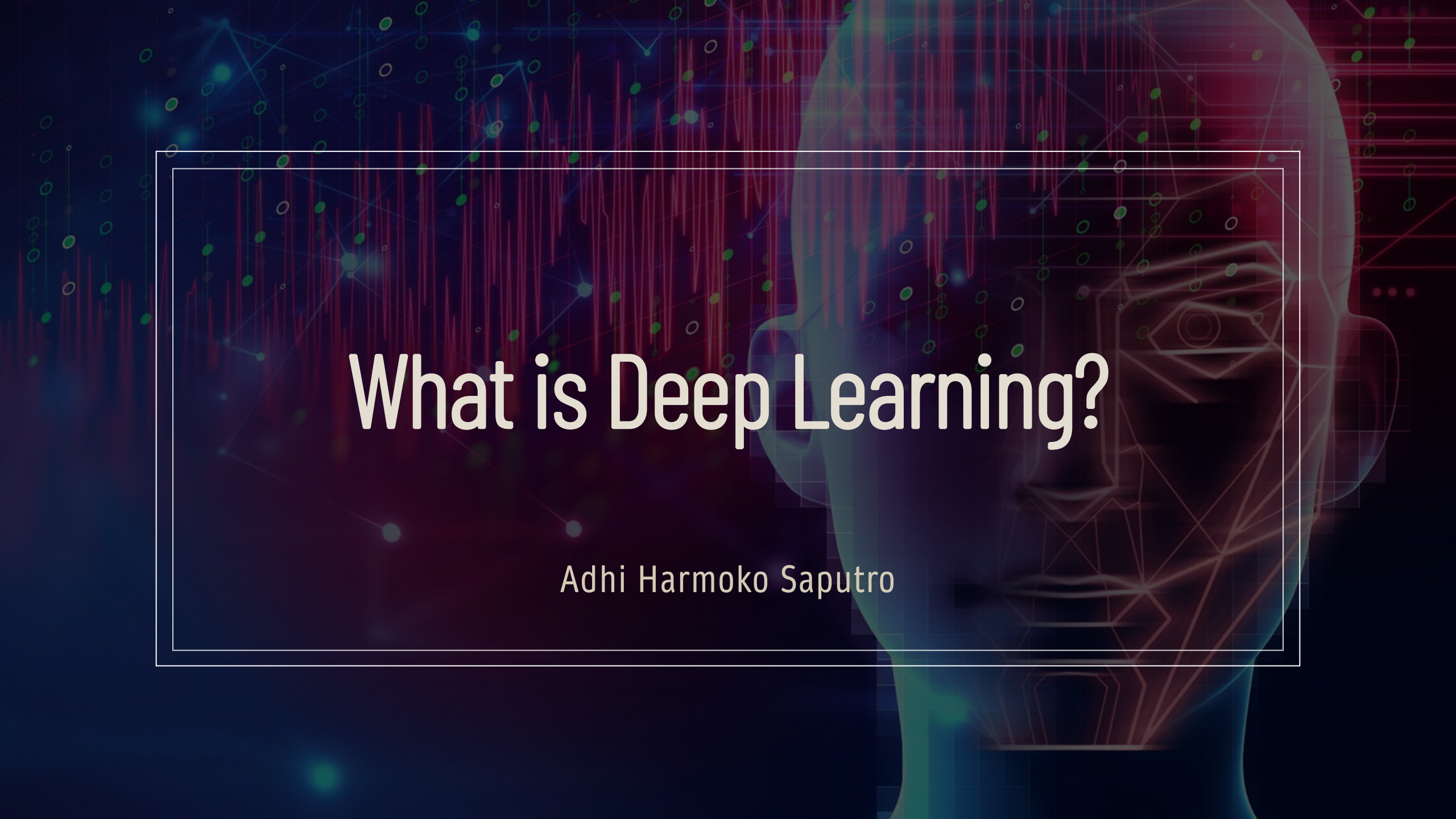
Artificial Intelligence

Adhi Harmoko Saputro



What is Deep Learning?

Adhi Harmoko Saputro

ARTIFICIAL INTELLIGENCE,

Machine Learning, and Deep Learning Non Exhaustive

LIST OF SECTORS TRANSFORMED

Marketing

- Hyper personalisation
- Recommendation systems
- Smarter targeted advertising

Finance

- Regtech KYC – using face detection
- Investment Portfolio Optimization & ETF replication for index tracking
- Fraud detection
- Payments – pay with face
- Trader oversight
- Customer Experience and support

Transport

- Autonomous Cars
- Autonomous Drones

Insurance

- Customer Experience and support
- Automated claims

Industry

- Automation
- Robotics

Retail

- Robots in Warehouses
- Payments pay with face
- Invetory Optimization
- Recommendations
- Visual Search
- Virtual Stylist

Security

- Cameras with intrusion detection
- Behavioural Detection
- Crowd counting
- Cyber in fighting malware

- KYC
- Fraud Prevention

The edge (on device) is the future

AI will move into the real world via intelligent devices that we will interact on a real time basis throughout the day

www.dls.ltd
@deeplearn007

Source: https://7wdata.be/internet-of-things/a-guide-to-real-world-ai-machine-learning-use-cases/

Artificial Intelligence

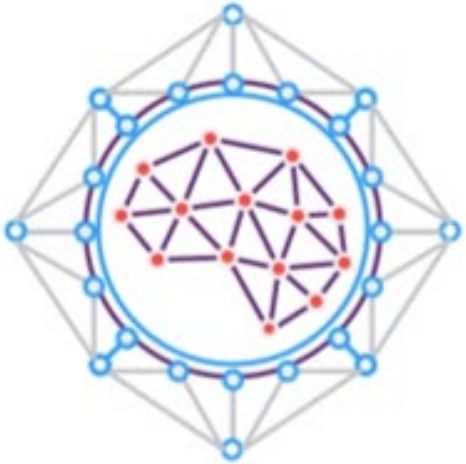
Early artificial intelligence stirs excitement

Machine Learning

Machine learning begins to flourish

Deep Learning

Deep learning breakthroughs drive AI boom

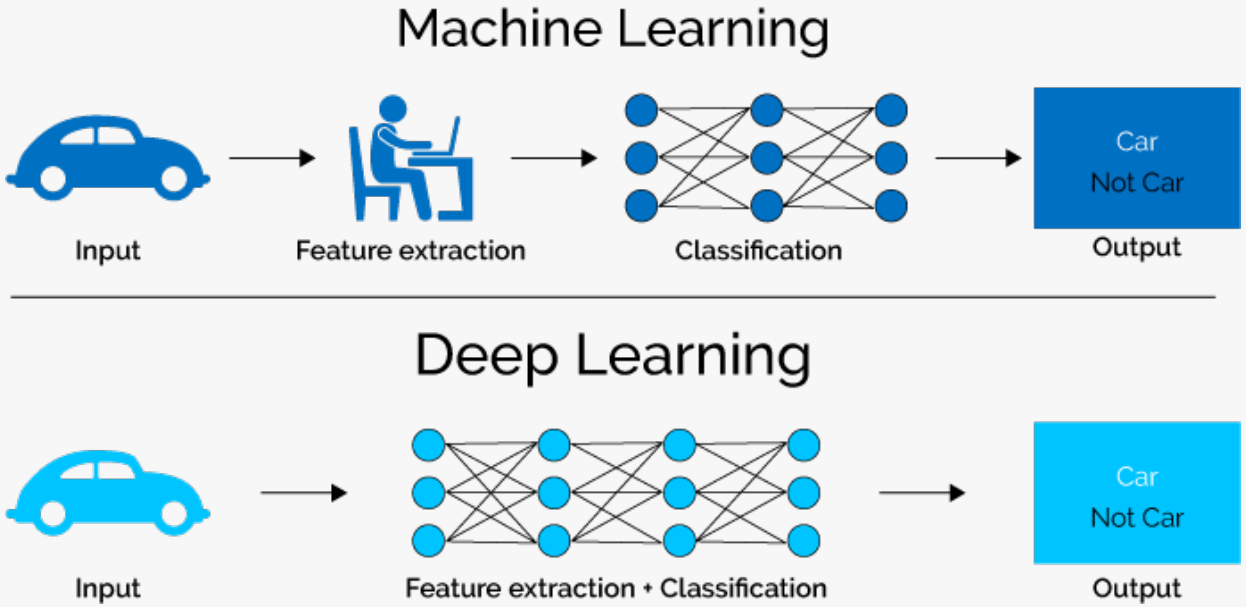


1950's 1960's 1970's 1980's 1990's 2000's 2010's 2020's 2030's

Source: https://www.kaggle.com/kaggle-survey-2020

Machine Learning v.s. Deep Learning

- A Machine Learning subfield of learning representations of data
  - Exceptional effective at learning patterns
- Deep Learning algorithms attempt to learn (multiple levels of) representation by using a hierarchy of multiple layers
- If you provide the system tons of information
  - it begins to understand it and respond in useful ways



What Deep Learning need?

- Input data points
  - If the task is speech recognition, these data points could be sound files of people speaking
  - If the task is image tagging, they could be pictures
- Examples of the expected output
  - In a speech-recognition task, these could be human-generated transcripts of sound files
  - In an image task, expected outputs could be tags such as “dog,” “cat,” and so on
- A way to measure whether the algorithm is doing a good job
  - This is necessary in order to determine the distance between the algorithm’s current output and its expected output
  - The measurement is used as a feedback signal to adjust the way the algorithm works
  - This adjustment step is what we call learning

Learning Rules & Data Representations

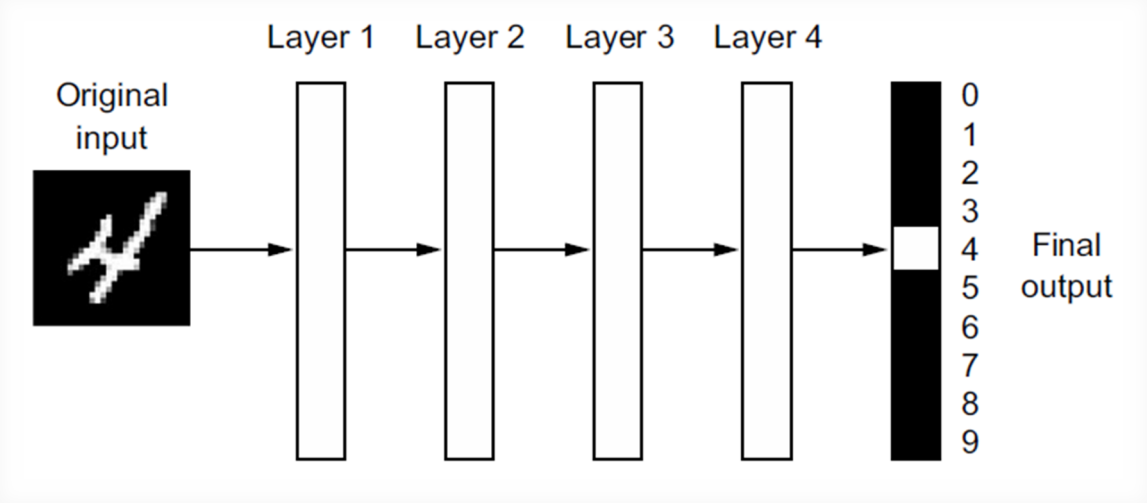
- The central problem in “Machine” Learning is to meaningfully transform data
  - Learn useful representations of the input data at hand
  - Tepresentations that get us closer to the expected output

- “Machine” learning models are all about finding appropriate representations for their input data
  - Transformations of the data that make it more amenable to the task at hand

- Learning describes an automatic search process for data transformations
  - Produce useful representations of some data, guided by some feedback signal
  - Representations that are amenable to simpler rules solving the task at hand

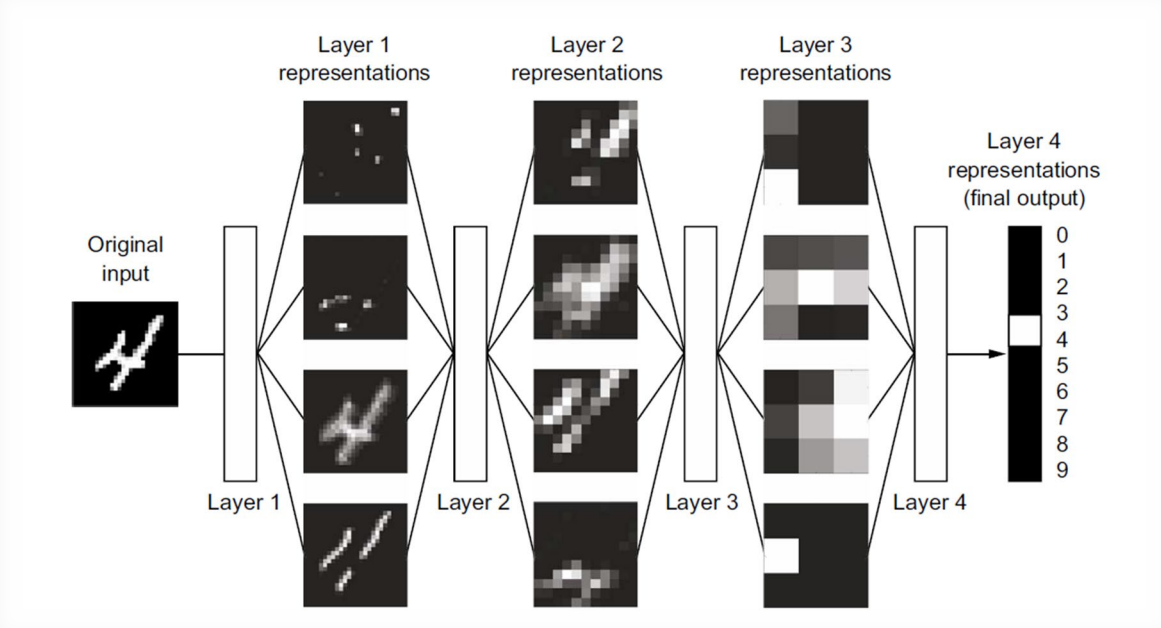
The “Deep” in “Deep Learning”

- The “deep” in “deep learning” isn’t a reference to any kind of deeper understanding achieved by the approach
  - It stands for this idea of successive layers of representations
  - The number of layers contribute to a model of the data is called the depth of the model
  - Modern deep learning often involves tens or even hundreds of successive layers of representations, and they’re all learned automatically from exposure to training data
  - These layered representations are learned via models called neural networks, structured in literal layers stacked on top of each other



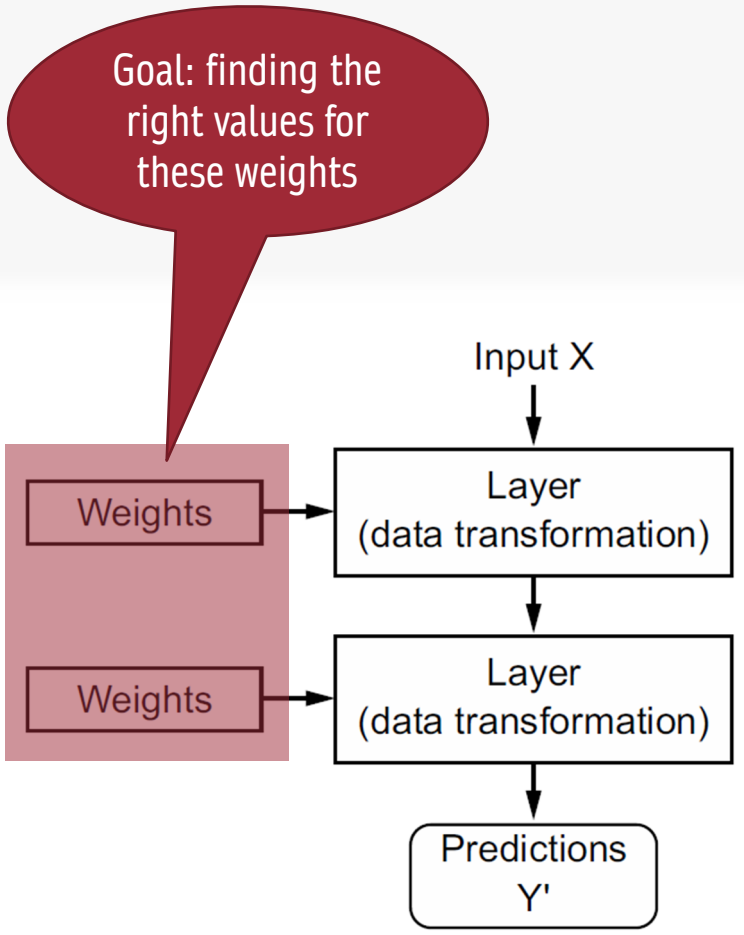
Mathematical Framework

- Deep Learning is a mathematical framework for learning representations from data
  - The network transforms the digit image into representations
    - Increasingly different from the original image and increasingly informative about the final result
  - A deep network is a multistage information distillation process
    - Information goes through successive filters and comes out increasingly purified



How Deep Learning Works

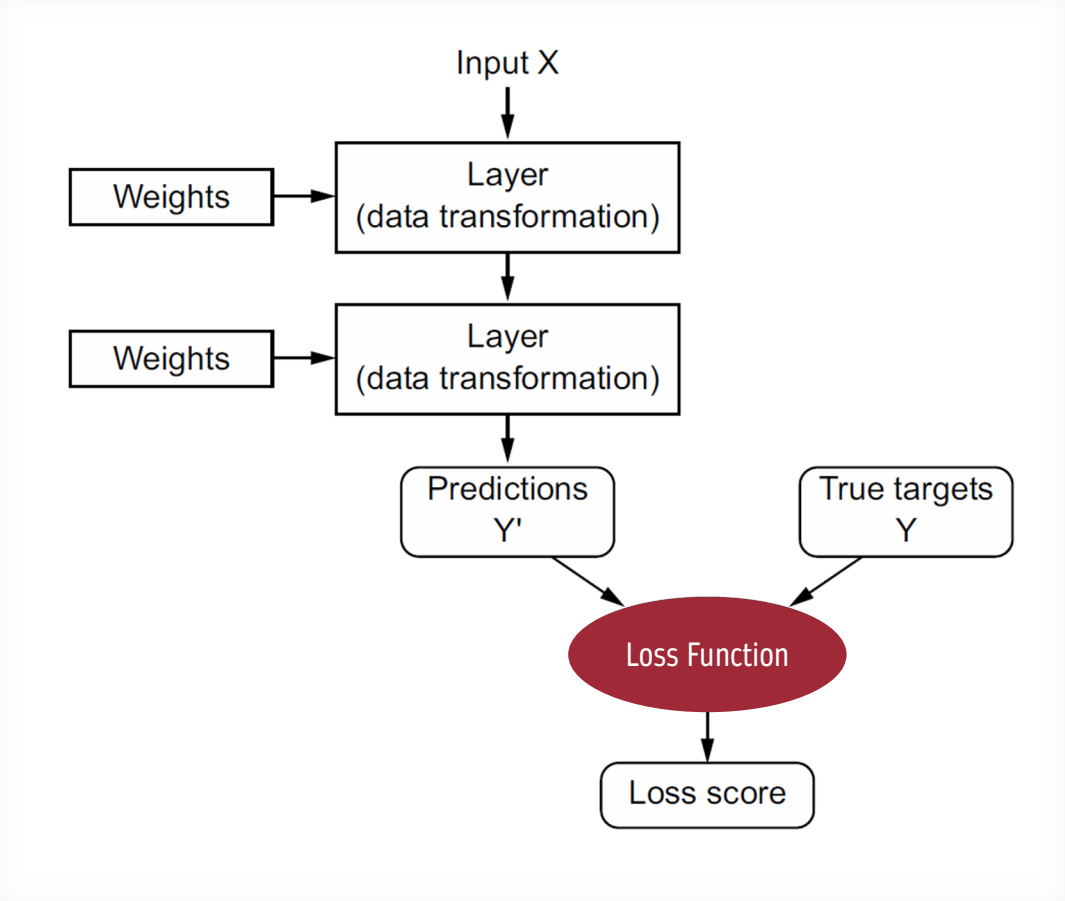
- Deep neural networks do the input-to-target mapping via a deep sequence of simple data transformations (layers) by observing many examples of input and targets
- The specification of what a layer does to its input data is stored in the layer's weights
  - The transformation implemented by a layer is parameterized by its weights
  - Learning means finding a set of values for the weights of all layers in a network, such that the network will correctly map example inputs to their associated targets
- A deep neural network can contain tens of millions of parameters
  - Finding the correct values for all of them may seem like a daunting task
  - Modifying the value of one parameter will affect the behavior of all the others



Loss Function

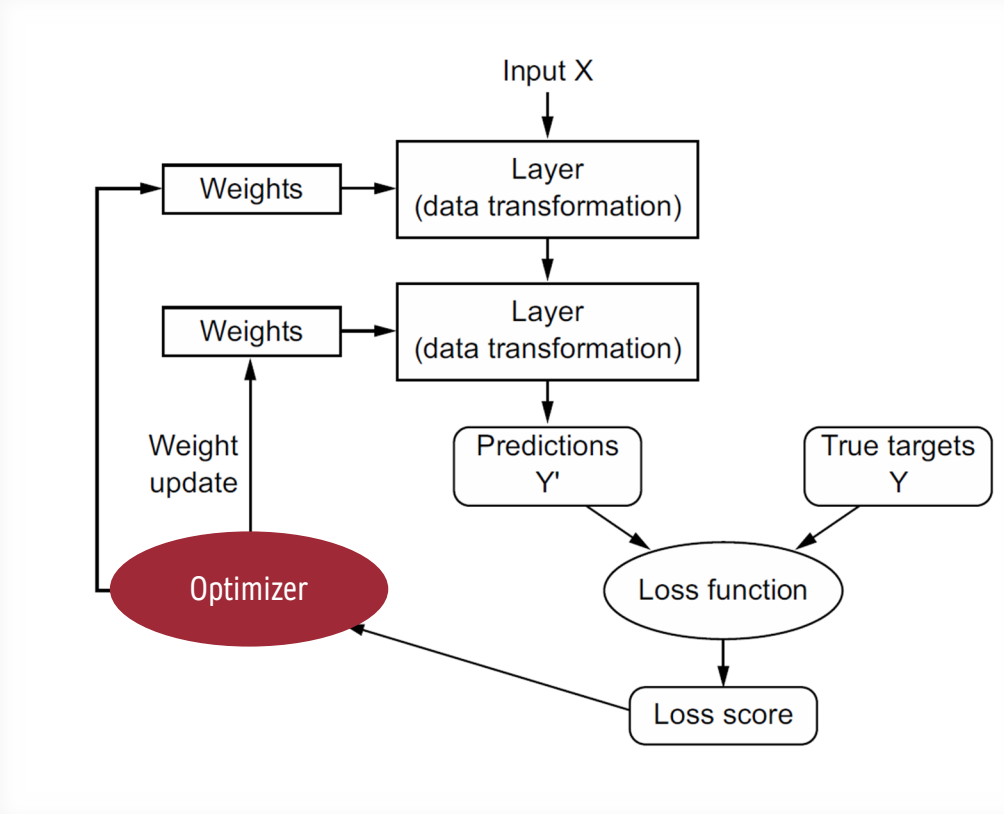
- The loss function of the network: measure how far the output is from what you expected
  - Compare the predictions of the network and the true target and computes a distance score, capturing how well the network has done on this specific example

- Use the loss score as a feedback signal to adjust the value of the weights a little, in a direction that will lower the loss score for the current example



Optimizer

- The optimizer: adjust the value of the weights a little using loss score
- The training loop: the weights are adjusted a little in the correct direction, and the loss score decreases
  - Repeated a sufficient number of times
  - Typically tens of iterations over thousands of examples
- A trained network: a network with a minimal loss is one for which the outputs are as close as they can be to the targets



Other Importants Terminology

- Epoch – The number of times the algorithm runs on the whole training dataset
- Sample – A single row of a dataset
- Batch – It denotes the number of samples to be taken to for updating the model parameters
- Learning Rate – It is a parameter that provides the model a scale of how much model weights should be updated
- Cost Function/Loss Function – A cost function is used to calculate the cost that is the difference between the predicted value and the actual value
- Weights/ Bias – The learnable parameters in a model that controls the signal between two neurons

Before Deep Learning

Probabilistic Modelling

- The application of the principles of statistics to data analysis
- The earliest forms of machine learning
- Ex.: Naive Bayes Algorithm, Logistic Regression

Early Neural Networks

- Investigated in toy forms
- Ex.: Backpropagation Algorithm

Kernel methods

- Group of classification algorithms
- Ex.: Support Vector Machine

Flowchart-like structures

- Flowchart-like structures that let you classify input data points or predict output values given inputs
- Ex.: Decision trees, random forests, gradient boosting machines

What makes Deep Learning Different

- It offered better performance for many problems
- Makes problem-solving much easier
  - It completely automates what used to be the most crucial step in a machine learning workflow: Feature Engineering
  - Completely automates in finding good layers of representations for the data
  - Joint feature Learning
    - The model adjusts one of its internal features
    - All other features that depend on it automatically adapt to the change, without requiring human intervention
  - Everything is supervised by a single feedback signal:
    - Every change in the model serves the end goal
  - The incremental, layer-by-layer way in which increasingly complex representations are developed
  - The intermediate incremental representations are learned jointly, each layer being updated to follow both the representational needs of the layer above and the needs of the layer below

Feature Engineering

Shallow Learning

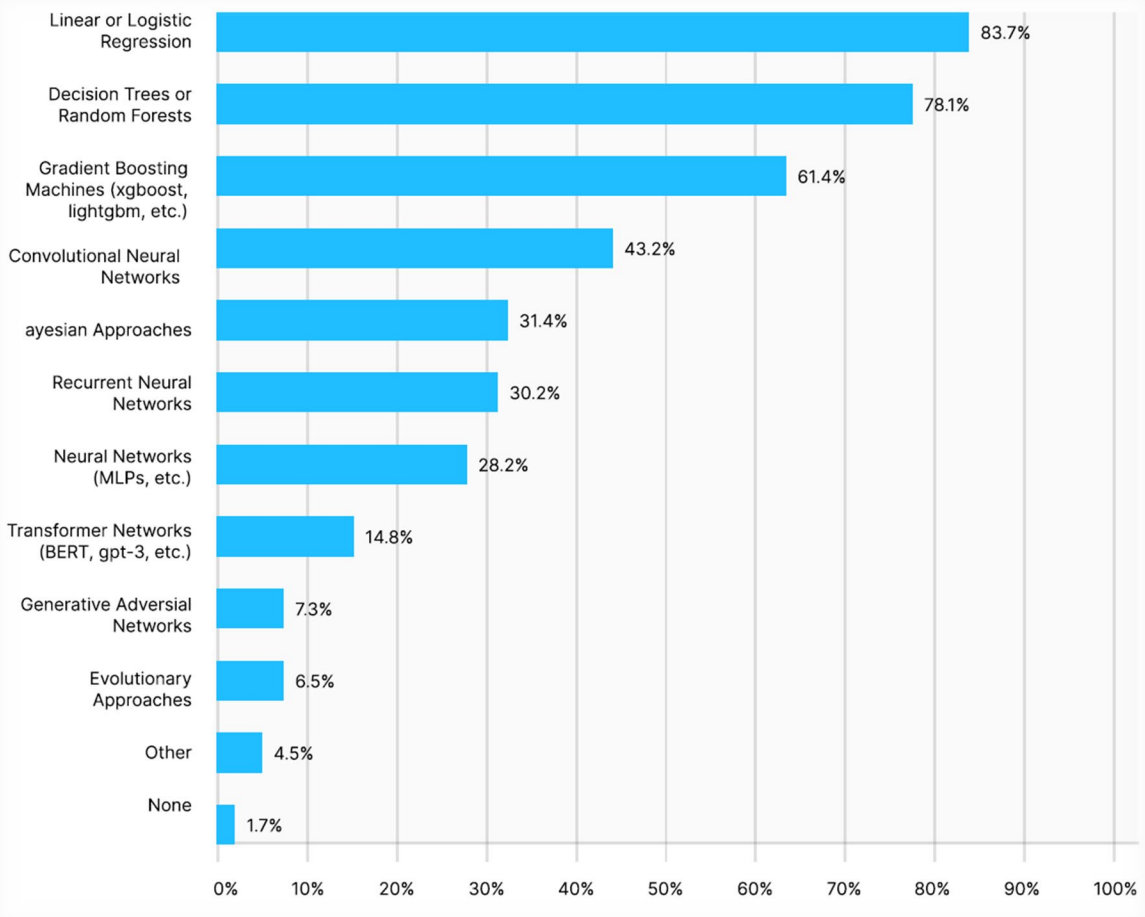
- Transforming the input data into one or two successive representation spaces
- Simple transformations such as high-dimensional non-linear projections

Deep Learning

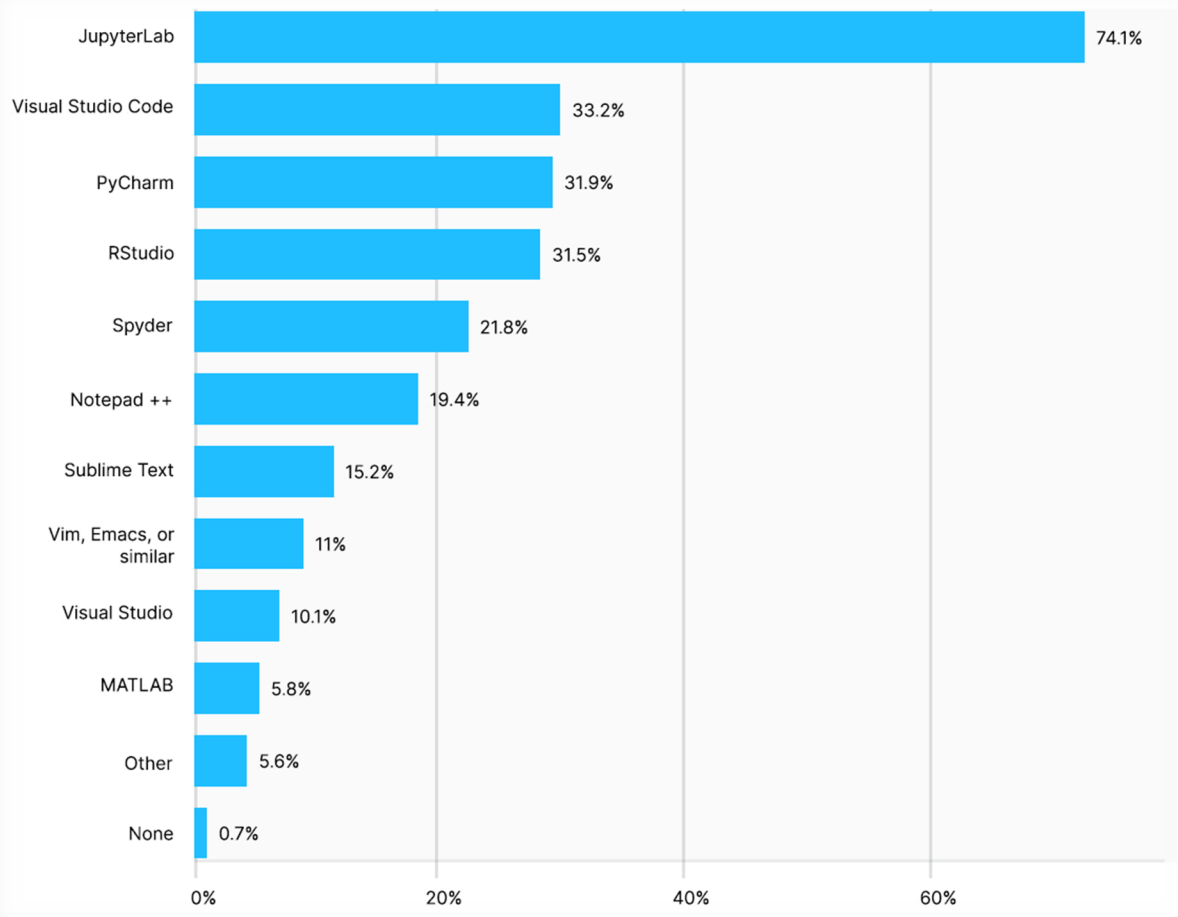
- Learn all features in one pass
- Replacing sophisticated multistage pipelines with a single, simple, end-to-end deep learning model

Survey

Methods & Algorithms



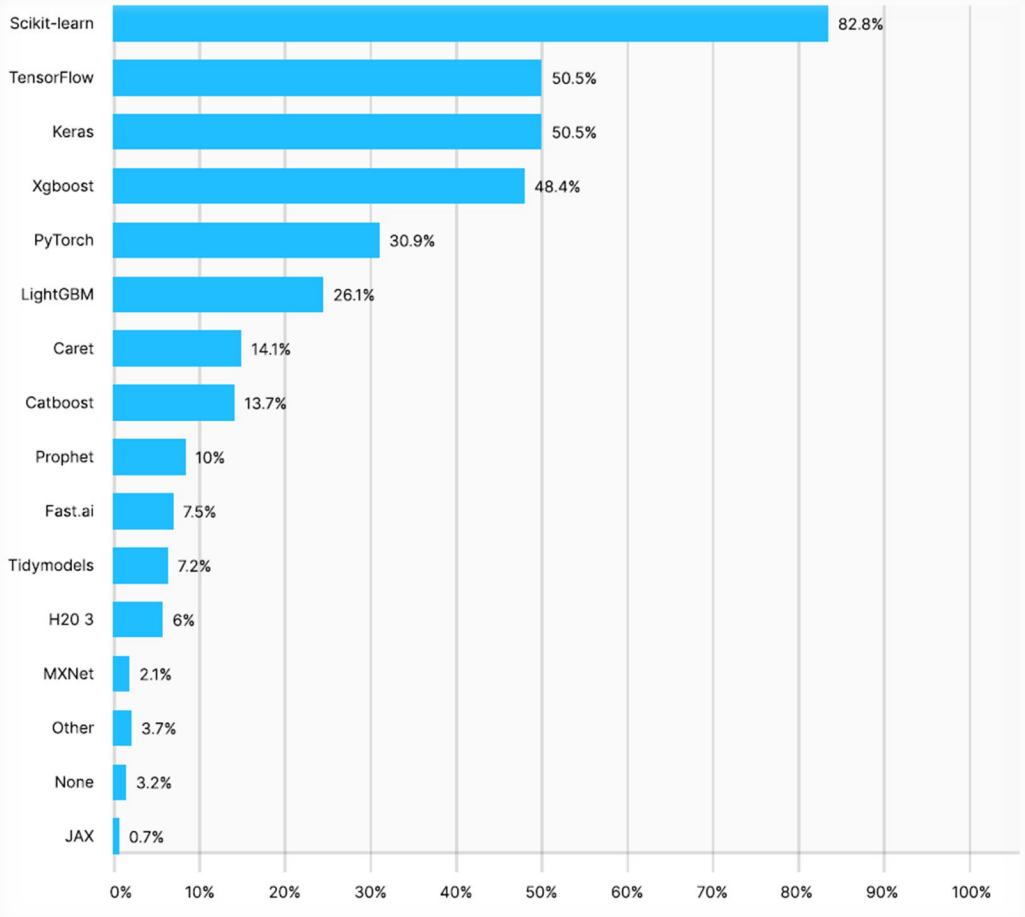
IDE Technology



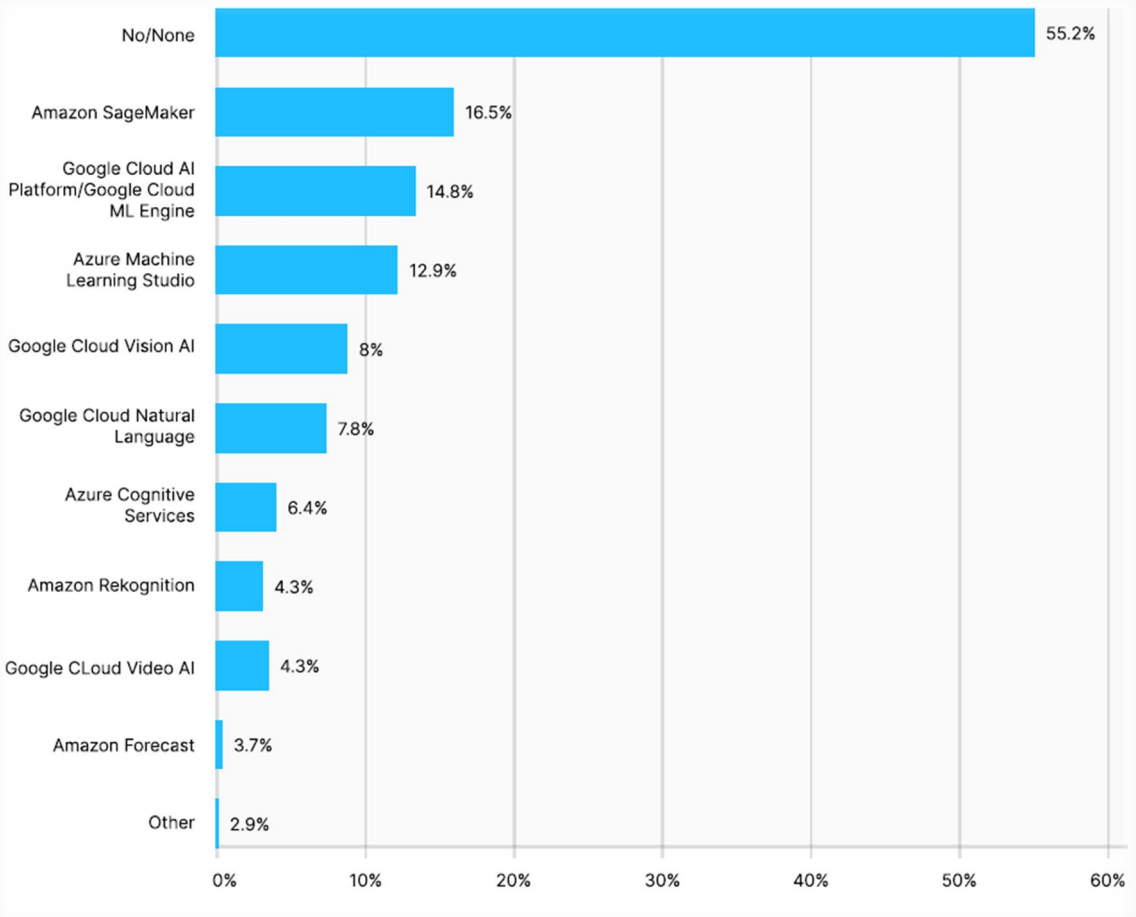
Source: https://www.kaggle.com/kaggle-survey-2020

Survey

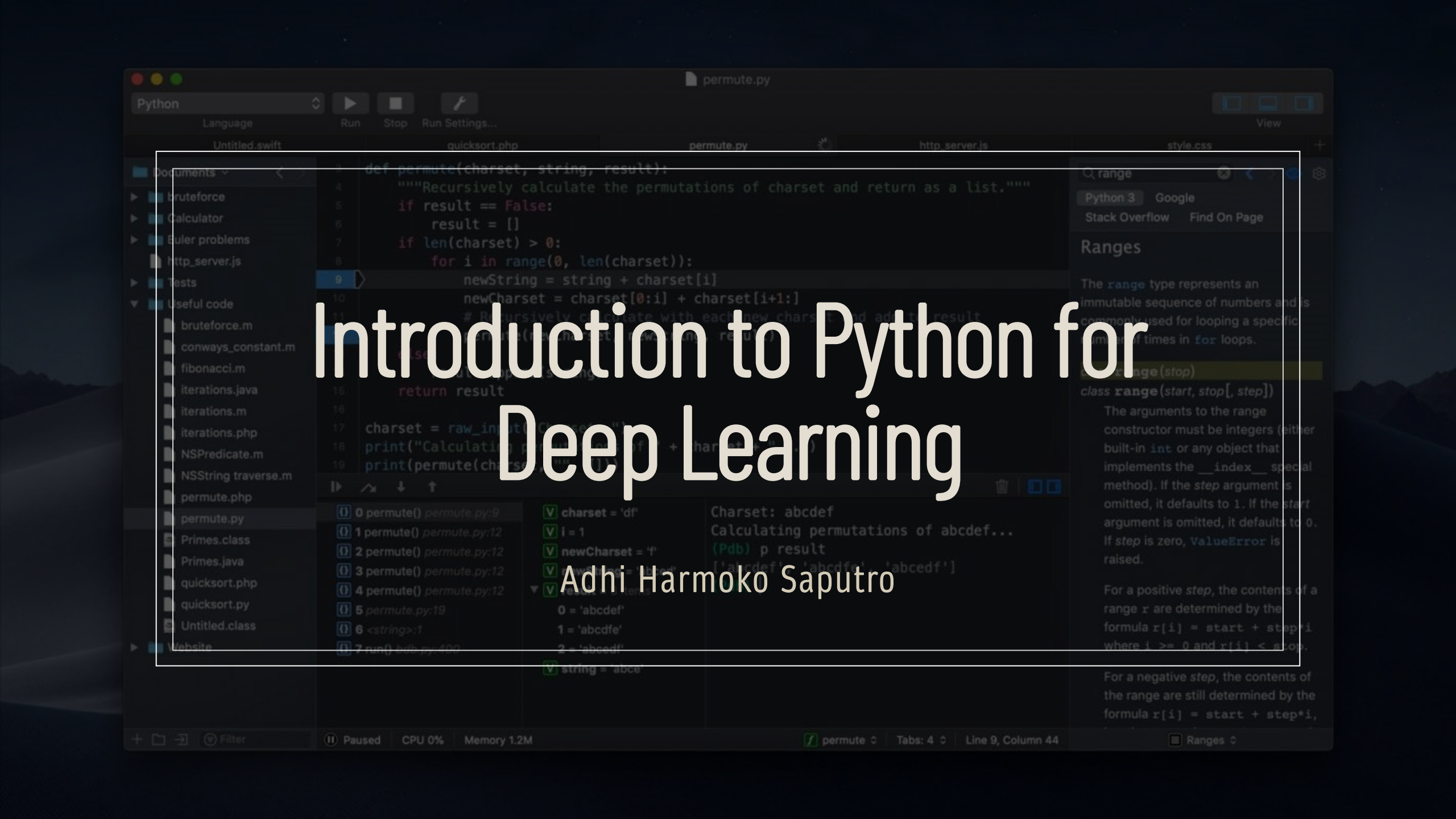
ML Framework



Enterprise ML Tools



Source: https://www.kaggle.com/kaggle-survey-2020

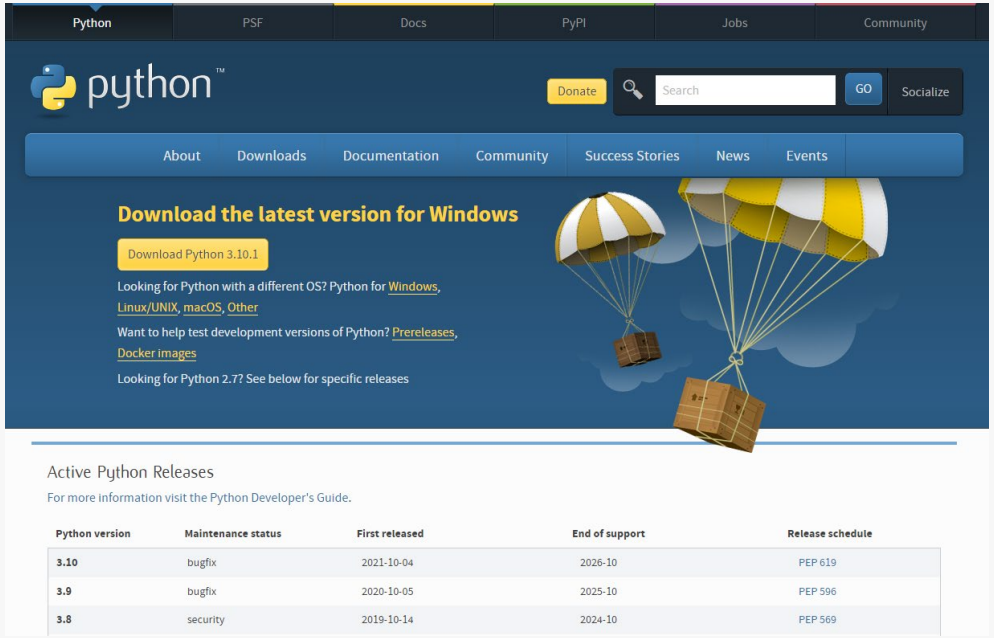


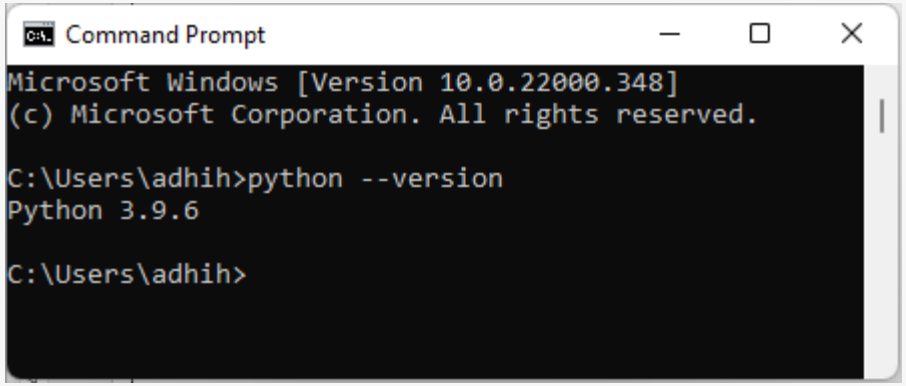
Introduction to Python for Deep Learning

Adhi Harmoko Saputro

Python

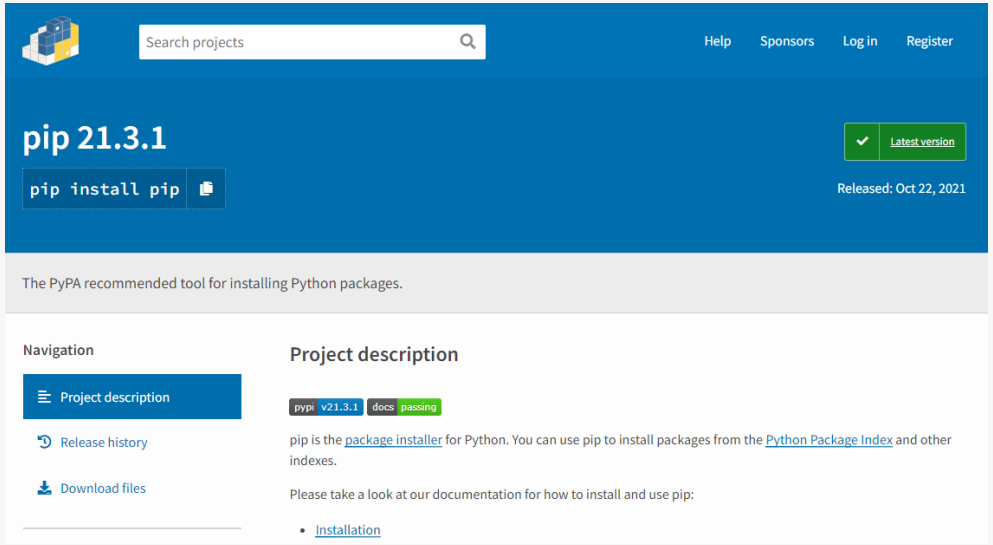
- Python is a cross-platform language by which the same program is written once and used by different operating systems
  - A general-purpose language that works on different platforms such as desktop, mobile, web, and embedded systems
- The official Python website is https://www.python.org/
  - Python 3 can be downloaded from the downloads page python.org/downloads

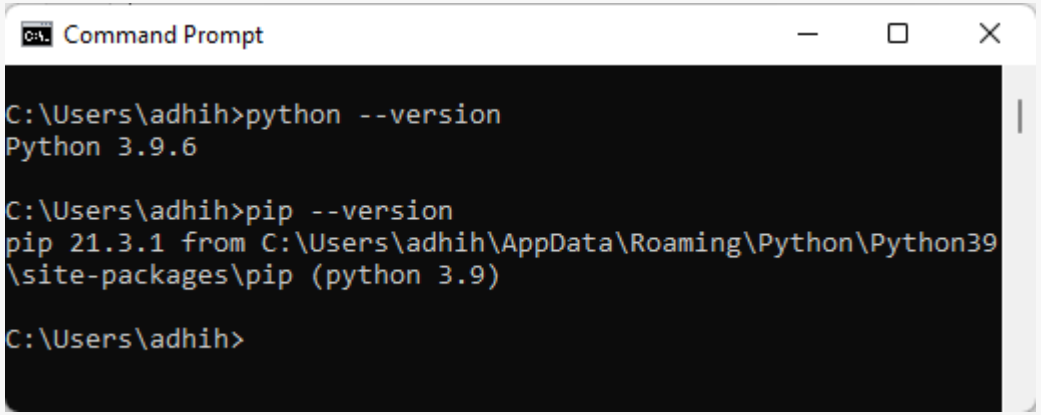




Python Package Installer

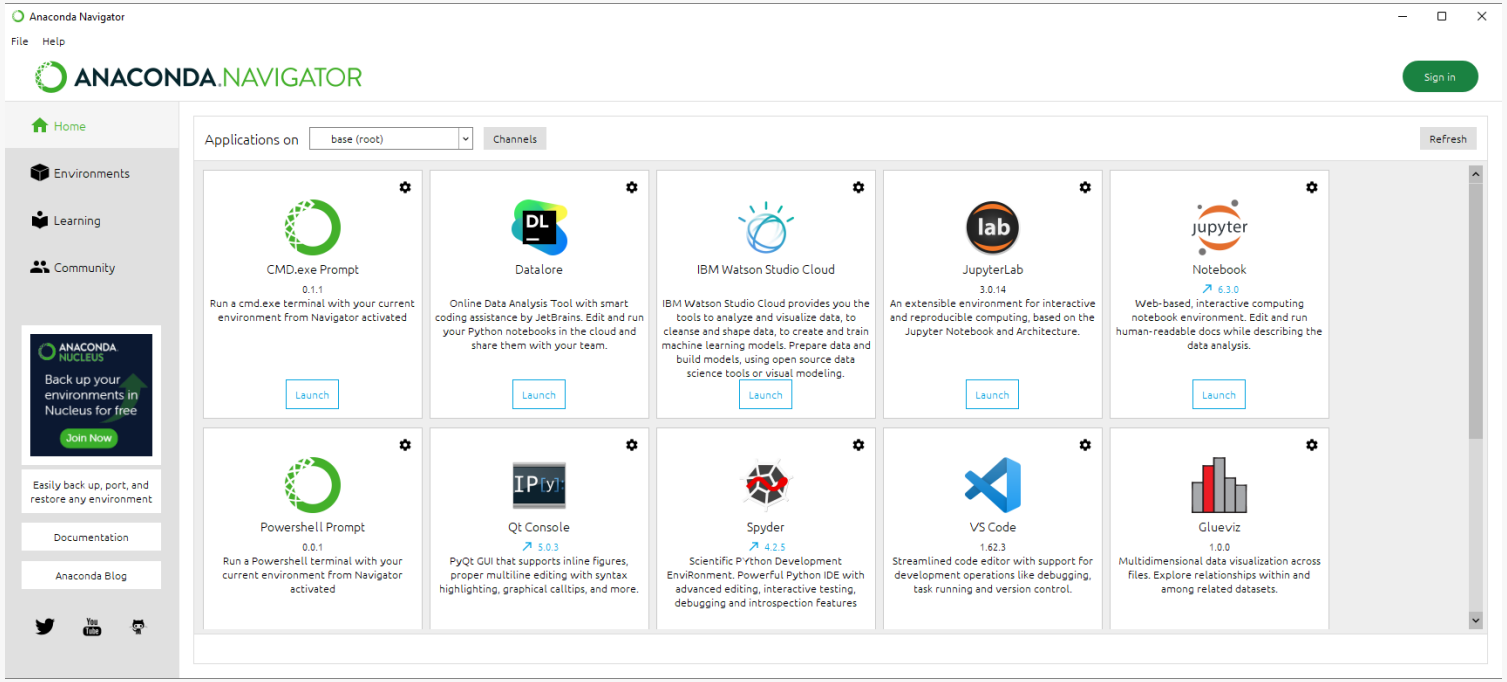
- pip is the package installer for Python (https://pypi.org/project/pip/)
  - Use pip to install packages from the Python Package Index and other indexes
  - How to install: https://pip.pypa.io/en/stable/installation/
  - How to use: https://pip.pypa.io/en/stable/getting-started/
    - pip install < library-name >





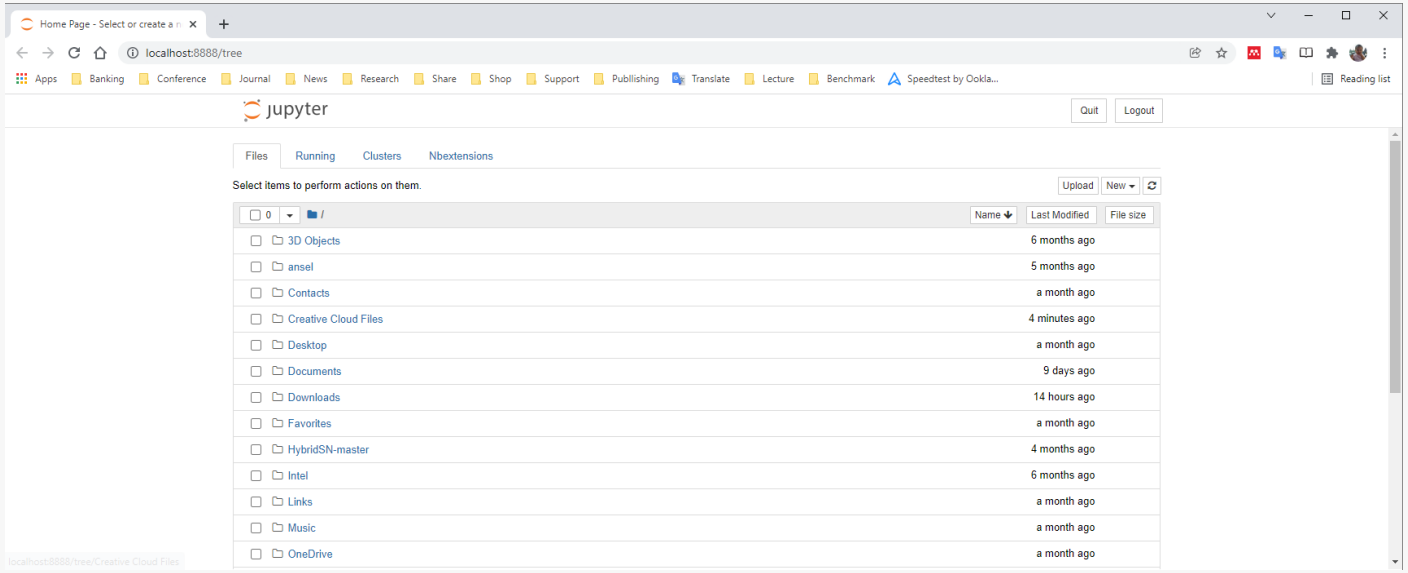
Anaconda

- Anaconda (https://www.anaconda.com/) is a distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment
  - The distribution includes data-science packages suitable for Windows, Linux, and macOS



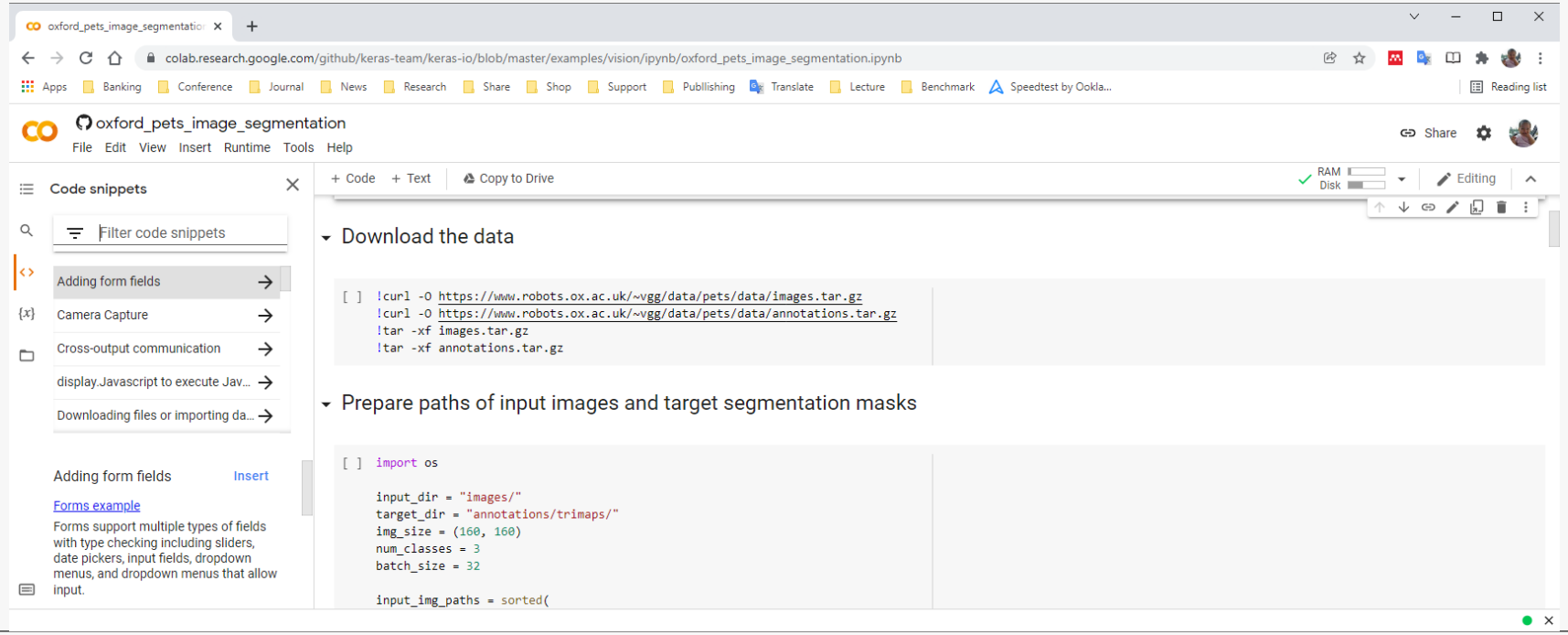
Jupyter Notebooks

- Jupyter notebooks are a great way to run deep learning experiments—in particular, the many code examples in this book
  - A notebook is a file generated by the Jupyter Notebook app (https://jupyter.org) that can edit in your browser
  - It mixes the ability to execute Python code with rich text-editing capabilities for annotating what you're doing
  - A notebook also allows you to break up long experiments into smaller pieces that can be executed independently, development interactively and rerun your code selectively



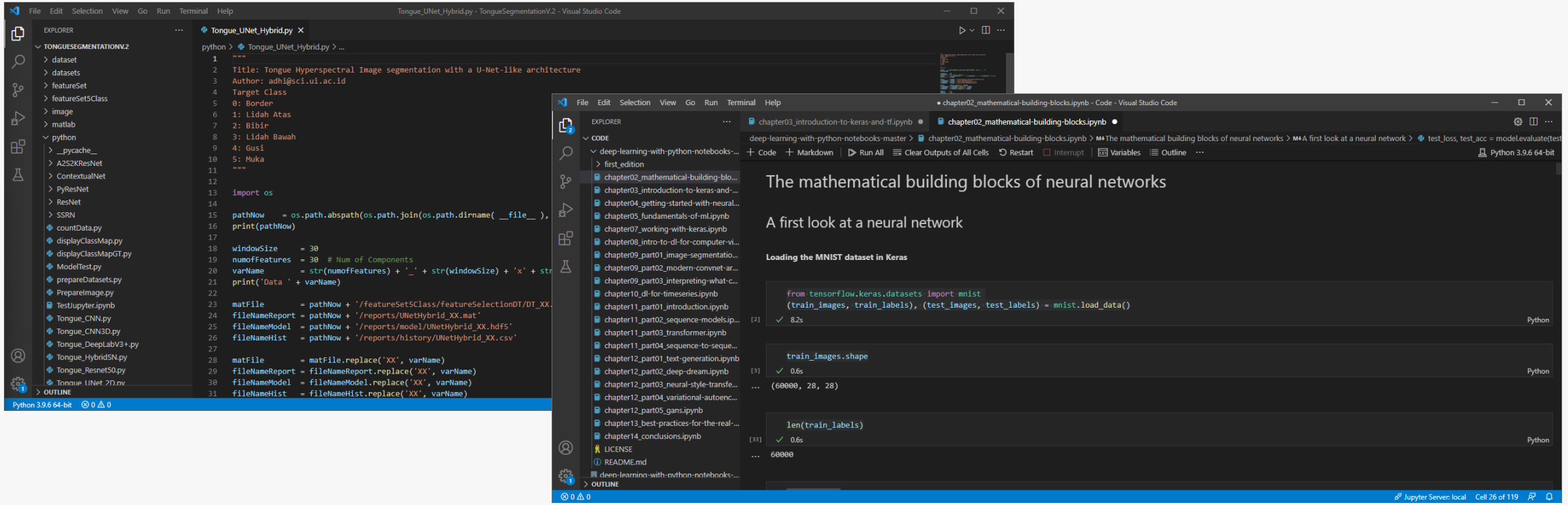
Google Colab

- Colaboratory (or Colab for short) is a free Jupyter notebook service that requires no installation and runs entirely in the cloud
  - Effectively, it's a web page that lets you write and execute Keras scripts right away
  - It gives you access to a free (but limited) GPU runtime and even a TPU runtime, so you don't have to buy your own GPU.
- To get started with Colab (https://colab.research.google.com) and click the New Notebook button



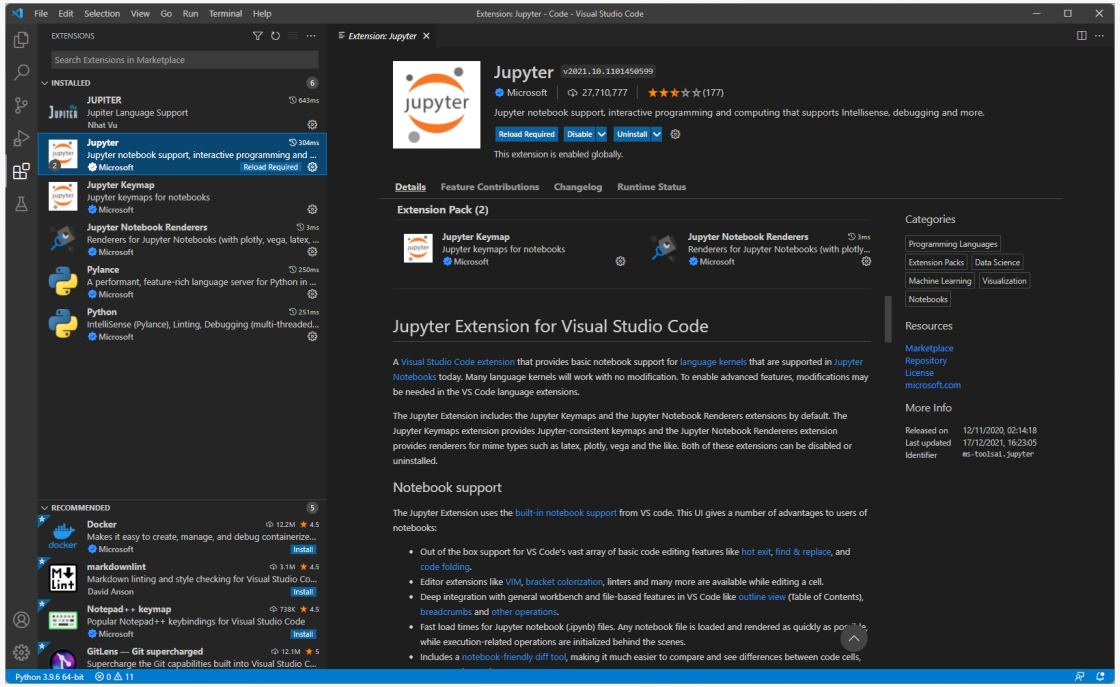
Visual Studio Code

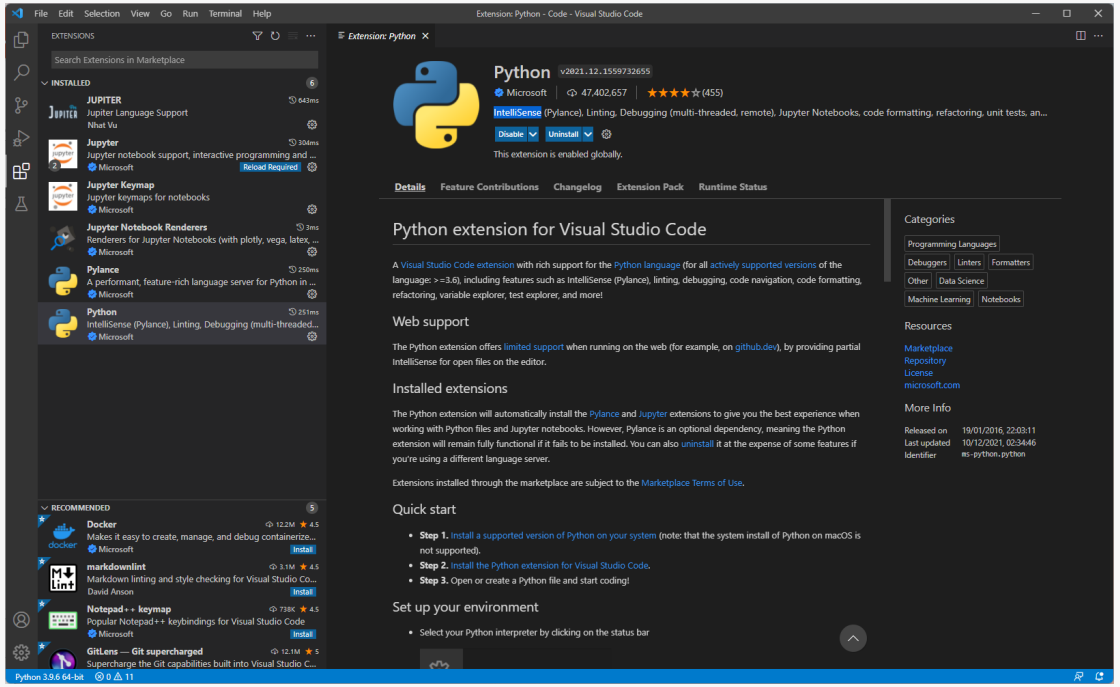
- Visual Studio Code (https://code.visualstudio.com/) is a source-code editor made by Microsoft for Windows, Linux and macOS
  - Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git



Visual Studio Code

- Install Visual Studio Code Extension: https://code.visualstudio.com/docs/editor/extension-marketplace

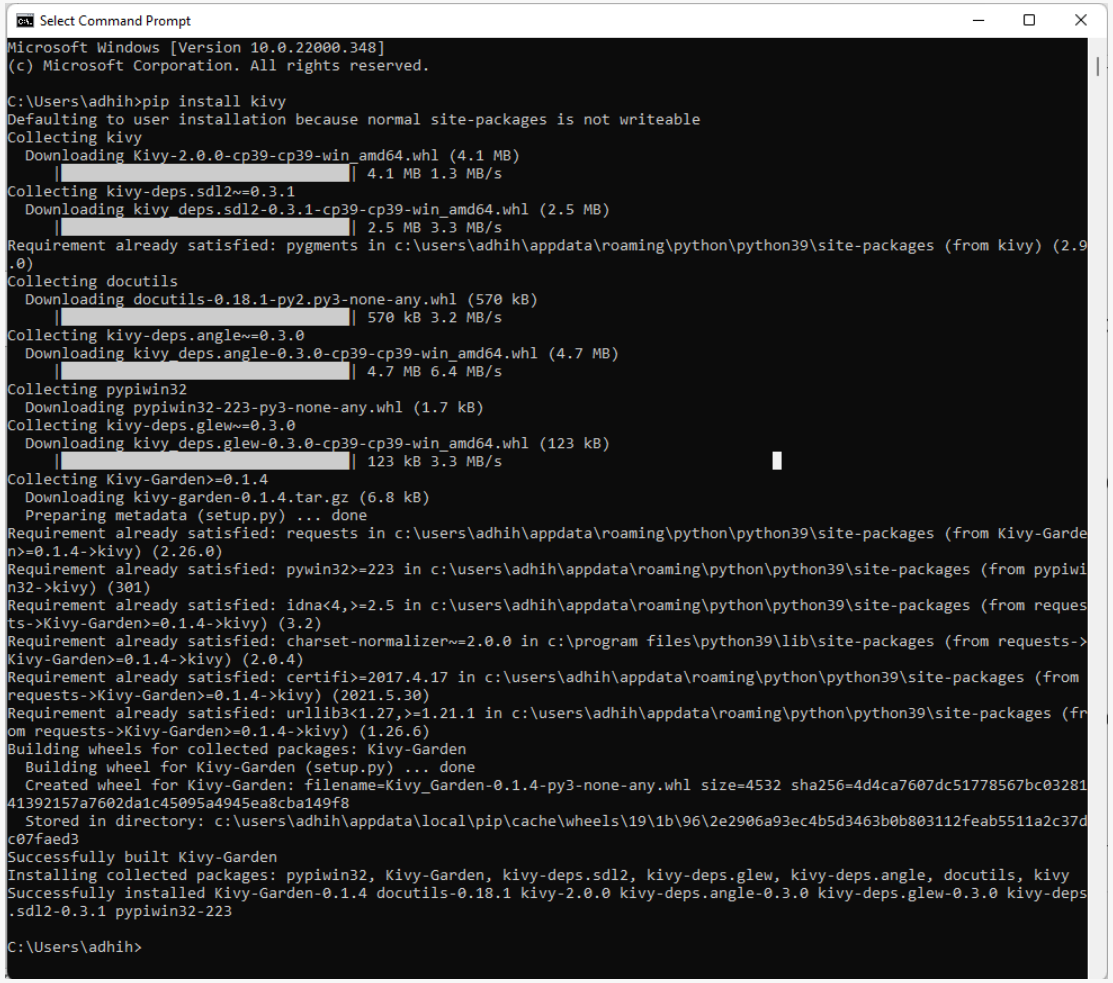




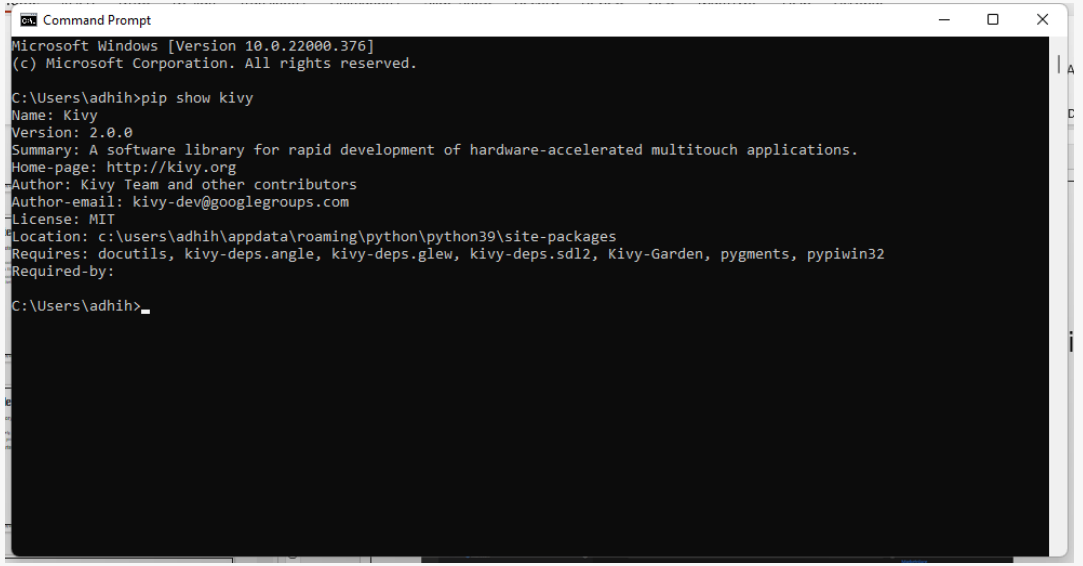
Installing Required Libraries

- NumPy
  - A library that supports the ndarray data type to process numerical arrays in Python that very popular library in data science
- Matplotlib
  - Used to create visualizations
- Cython
  - A superset of Python to support the use of C code within Python, which speeds-up the Python execution
- Virtualenv
  - Creates virtual environments where Python and its libraries can be installed
- Kivy
  - A cross-platform library for building native user interfaces
- KivyMD
  - A number of widgets that are compatible with Kivy and approximate Google's Material Design spec
- Buildozer
  - A tool for building the Kivy applications for mobile devices

Installing Required Libraries



Example script to install Kivy: pip install kivy



Example script to check Kivy: pip show kivy

Data Representations

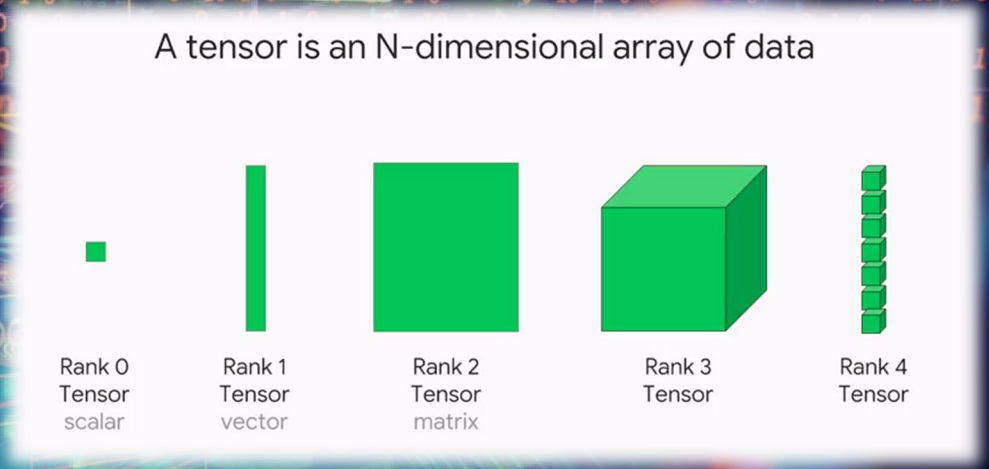
All current Machine / Deep Learning systems use tensors as their basic data structure

A tensor is a container for data which can house data in N dimensions

- Usually numerical data

Tensors are a generalization of matrices to an arbitrary number of dimensions

- In the tensors, a dimension is often called an axis



MNIST Dataset

- Grayscale images of handwritten digits (28 × 28 pixels) that has 10 categories (0 through 9)
  - It's a set of 60,000 training images, plus 10,000 test images, assembled by the National Institute of Standards and Technology
  - Each such matrix is a grayscale image, with coefficients between 0 and 255
  - Access in http://yann.lecun.com/exdb/mnist/

Loading the MNIST Dataset in Keras

- The MNIST dataset comes preloaded in Keras, in the form of a set of four NumPy arrays

- The images are encoded as NumPy arrays, and the labels are an array of digits, ranging from 0 to 9
  - The images and labels have a one-to-one correspondence

```
from tensorflow.keras.datasets import mnist

(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

Scalars (Rank-0 Tensors)

- A tensor that contains only one number is called a scalar (or scalar tensor, or rank-0 tensor, or 0D tensor)
  - In NumPy, a float32 or float64 number is a scalar tensor (or scalar array)
  - Can display the number of axes of a NumPy tensor via the ndim attribute;
  - A scalar tensor has 0 axes (ndim == 0)
  - The number of axes of a tensor is also called its rank

```
import numpy as np
x = np.array(12)
x
array(12)


x.ndim
0
```

Vectors (Rank-1 Tensors)

- An array of numbers is called a vector, or rank-1 tensor, or 1D tensor
  - A rank-1 tensor is said to have exactly one axis

- The vector that has five entries, is called a 5-dimensional vector

- A 5D Vector v.s. A 5D Tensor
  - A 5D vector has only one axis and has five dimensions along its axis
  - A 5D tensor has five axes (and may have any number of dimensions along each axis)

```
x = np.array([12, 3, 6, 14, 7])
x
array([12, 3, 6, 14, 7])


x.ndim
1
```

Matrices (Rank-2 Tensors)

- An array of vectors is a matrix, or rank-2 tensor, or 2D tensor
  - A matrix has two axes (often referred to as rows and columns)
  - Can visually interpret a matrix as a rectangular grid of numbers

- Rows: the entries from the first axis
- Columns: the entries from the second axis

- [5, 78, 2, 34, 0] is the first row of x
- [5, 6, 7] is the first column of x

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

Rank-3 & Higher-rank Tensors

- If you pack such matrices in a new array
  - Obtain a rank-3 tensor (or 3D tensor)
  - Can visually interpret as a cube of numbers

- By packing rank-3 tensors in an array
  - Can create a rank-4 tensor, and so on
  - In deep learning, generally manipulate tensors with ranks 0 to 4
  - May go up to 5 if process video data

```
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
```

Key Attributes

- Number of axes (rank)
  - For instance, a rank-3 tensor has three axes, and a matrix has two axes
  - This is also called the tensor's ndim in Python libraries such as NumPy or TensorFlow

- Shape
  - This is a tuple of integers that describes how many dimensions the tensor has along each axis
  - For instance, the previous matrix example has shape (3, 5), and the rank-3 tensor example has shape (3, 3, 5)
  - A vector has a shape with a single element, such as (5,), whereas a scalar has an empty shape, ()

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

train_images.ndim
3



train_images.shape
(60000, 28, 28)
```

Key Attributes

- Data type (usually called dtype in Python libraries)
  - This is the type of the data contained in the tensor
    - for instance, a tensor's type could be float16, float32, float64, uint8, and so on
  - In TensorFlow, you are also likely to come across string tensors

```
train_images.dtype
dtype('uint8')
```

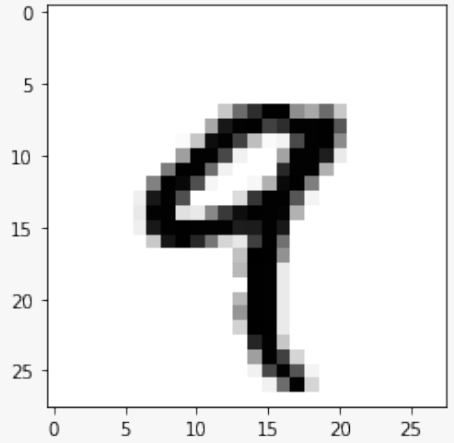
Display the Data

- Display the fourth digit in the rank-3 tensor, using the Matplotlib library
  - A well-known Python data visualization library

```
import matplotlib.pyplot as plt
digit = train_images[4]
plt.imshow(digit, cmap=plt.cm.binary)
plt.show()
```

- The corresponding label is the integer 9

```
train_labels[4]
9
```



Manipulating Tensors in NumPy

- Selecting specific elements in a tensor is called tensor slicing
  - Specifies a start index and stop index for the slice along each tensor axis
  - Example: selects digits #10 to #100 (#100 isn't included) and puts them in an array of shape (90, 28, 28)

```
my_slice = train_images[10:100]
my_slice.shape
(90, 28, 28)
```

```
my_slice = train_images[10:100, :, :]
my_slice.shape
(90, 28, 28)
```

```
my_slice = train_images[10:100, 0:28, 0:28]
my_slice.shape
(90, 28, 28)
```

- Select slices between any two indices along each tensor axis
  - For instance, in order to select 14 × 14 pixels in the bottom-right corner of all images

```
my_slice = train_images[:, 14:, 14:]
my_slice.shape
(60000, 14, 14)
```

Manipulating Tensors in NumPy

- Possible to use negative indices
  - Much like negative indices in Python lists, indicate a position relative to the end of the current axis
  - Example: Crop the images to patches of 14 × 14 pixels centered in the middle

```
my_slice = train_images[:, 7:-7, 7:-7]
my_slice.shape
(60000, 14, 14)
```

The Notion of Data Batches

- The first axis (axis 0, because indexing starts at 0) in all data tensors, come across in deep learning will be the samples axis (sometimes called the samples dimension)
  - The first axis (axis 0) is called the batch axis or batch dimension
  - In the MNIST example, “samples” are images of digits
- Deep learning models don’t process an entire dataset at once
  - Break the data into small batches
  - Example: one batch of our MNIST digits, with a batch size of 128

```
batch = train_images[:128]
batch.shape
(128, 28, 28)


batch = train_images[128:256]
batch.shape
(128, 28, 28)


n = 3
batch = train_images[128 * n:128 * (n + 1)]
batch.shape
(128, 28, 28)
```

Real-world Examples of Data Tensors

Vector Data

- Rank-2 tensors of shape (samples, features), where each sample is a vector of numerical attributes (“features”)

Timeseries Data or Sequence Data

- Rank-3 tensors of shape (samples, timesteps, features), where each sample is a sequence (of length timesteps) of feature vectors

Images

- Rank-4 tensors of shape (samples, height, width, channels), where each sample is a 2D grid of pixels, and each pixel is represented by a vector of values (“channels”)

Video

- Rank-5 tensors of shape (samples, frames, height, width, channels), where each sample is a sequence (of length frames) of images

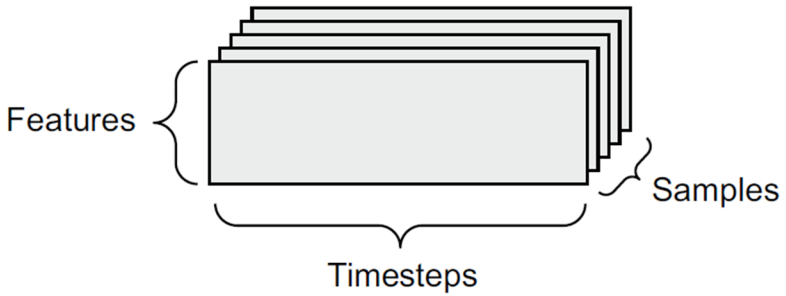
Vector Data

- In such a dataset, each single data point can be encoded as a vector
  - A batch of data will be encoded as a rank-2 tensor (that is, an array of vectors)
  - The first axis is the samples axis and the second axis is the features axis

- Examples:
  - An actuarial dataset of people
    - Consider each person's age, gender, and income
    - Each person can be characterized as a vector of 3 values
    - An entire dataset of 100,000 people can be stored in a rank-2 tensor of shape (100000, 3)
  - A dataset of text documents
    - Represent each document by the counts of how many times each word appears in it (out of a dictionary of 20,000 common words)
    - Each document can be encoded as a vector of 20,000 values (one count per word in the dictionary)
    - An entire dataset of 500 documents can be stored in a tensor of shape (500, 20000)

Timeseries Data or Sequence Data

- Whenever time matters in your data (or the notion of sequence order)
  - It makes sense to store it in a rank-3 tensor with an explicit time axis
  - Each sample can be encoded as a sequence of vectors (a rank-2 tensor)
  - A batch of data will be encoded as a rank-3 tensor



Timeseries Data or Sequence Data

- The time axis is always the second axis (axis of index 1) by convention
- Examples:
  - A dataset of stock prices
    - Every minute, store the current price of the stock, the highest price in the past minute, and the lowest price in the past minute.
    - Every minute is encoded as a 3D vector, an entire day of trading is encoded as a matrix of shape (390, 3) (there are 390 minutes in a trading day), and 250 days' worth of data can be stored in a rank-3 tensor of shape (250, 390, 3)
    - Each sample would be one day's worth of data
  - A dataset of tweets
    - Encode each tweet as a sequence of 280 characters out of an alphabet of 128 unique characters
    - Each character can be encoded as a binary vector of size 128 (an all-zeros vector except for a 1 entry at the index corresponding to the character)
    - Each tweet can be encoded as a rank-2 tensor of shape (280, 128), and a dataset of 1 million tweets can be stored in a tensor of shape (1000000, 280, 128)

Image Data

- Images typically have three dimensions: height, width, and color depth
  - Grayscale images (like MNIST digits) have only a single color channel
  - Could be stored in rank-2 tensors, by convention image tensors are always rank-3, with a one-dimensional color channel for grayscale images
  - A batch of 128 grayscale images of size 256 × 256, could be stored in a tensor of shape (128, 256, 256, 1), and a batch of 128 color images could be stored in a tensor of shape (128, 256, 256, 3)

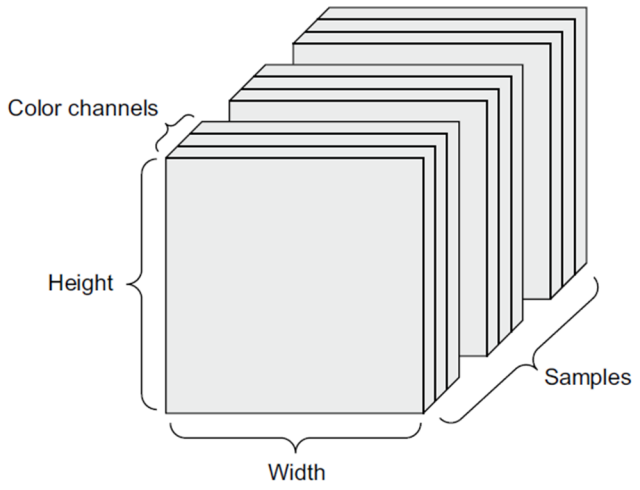


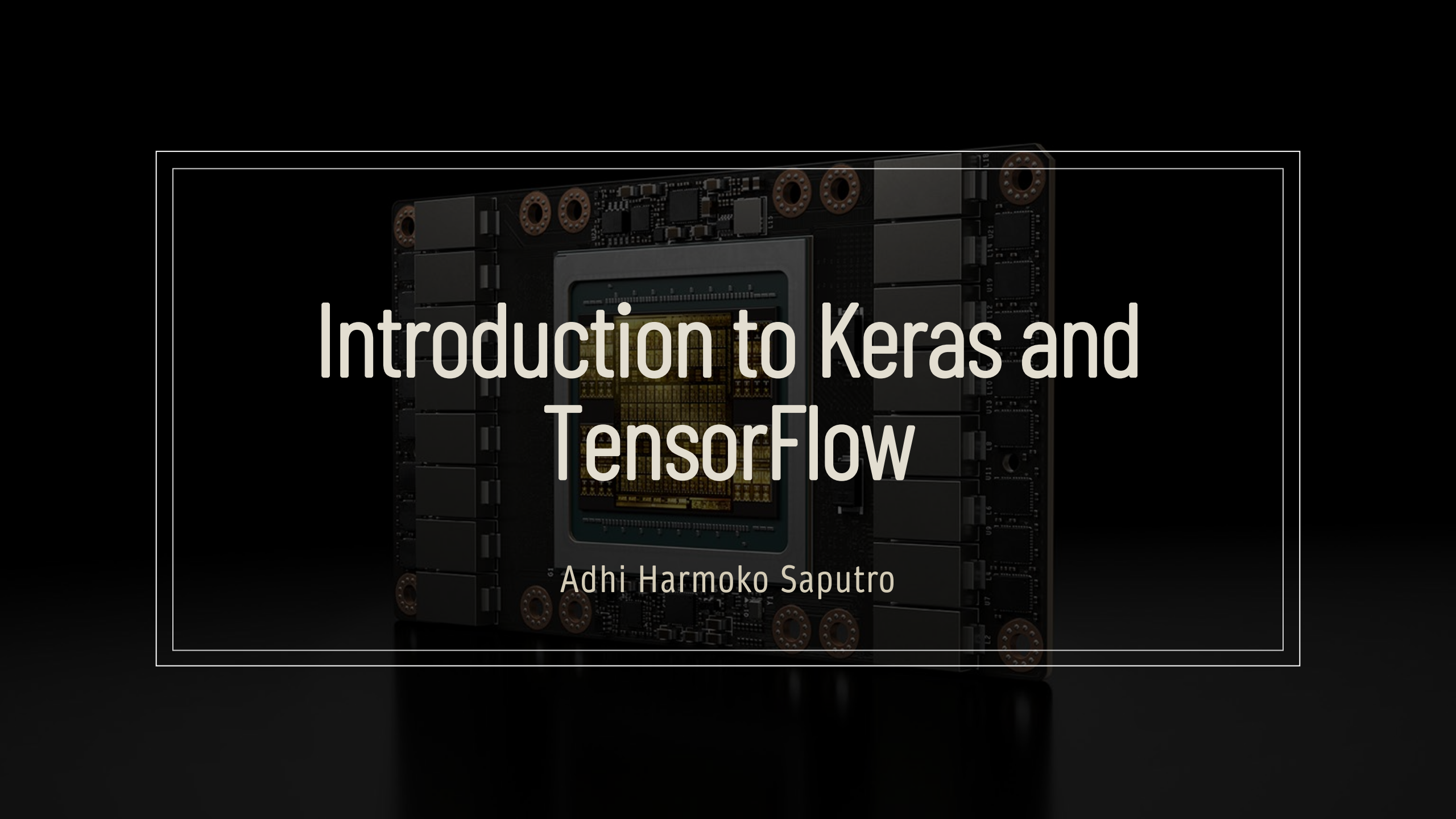
Image Data

- There are two conventions for shapes of image tensors:
  - The channels-last convention (which is standard in TensorFlow)
    - Places the color-depth axis at the end: (samples, height, width, color_depth)
  - The channels-first convention (which is increasingly falling out of favor)
    - Places the color depth axis right after the batch axis: (samples, color_depth, height, width)
    - With the channels-first convention, the previous examples would become (128, 1, 256, 256) and (128, 3, 256, 256).

- The Keras API provides support for both formats

Video Data

- Video data is one of the few types of real-world data
  - Need rank-5 tensors
  - A sequence of frames, each frame being a color image
  - Each frame can be stored in a rank-3 tensor (height, width, color_depth), a sequence of frames can be stored in a rank-4 tensor (frames, height,width, color_depth)
  - A batch of different videos can be stored in a rank-5 tensor of shape (samples, frames, height, width, color_depth)
- Example:
  - A 60-second, 144 × 256 YouTube video clip sampled at 4 frames per second would have 240 frames
  - A batch of four such video clips would be stored in a tensor of shape (4, 240, 144, 256, 3)
  - A total of 106,168,320 values
    - If the dtype of the tensor was float32, each value would be stored in 32 bits, so the tensor would represent 405 MB



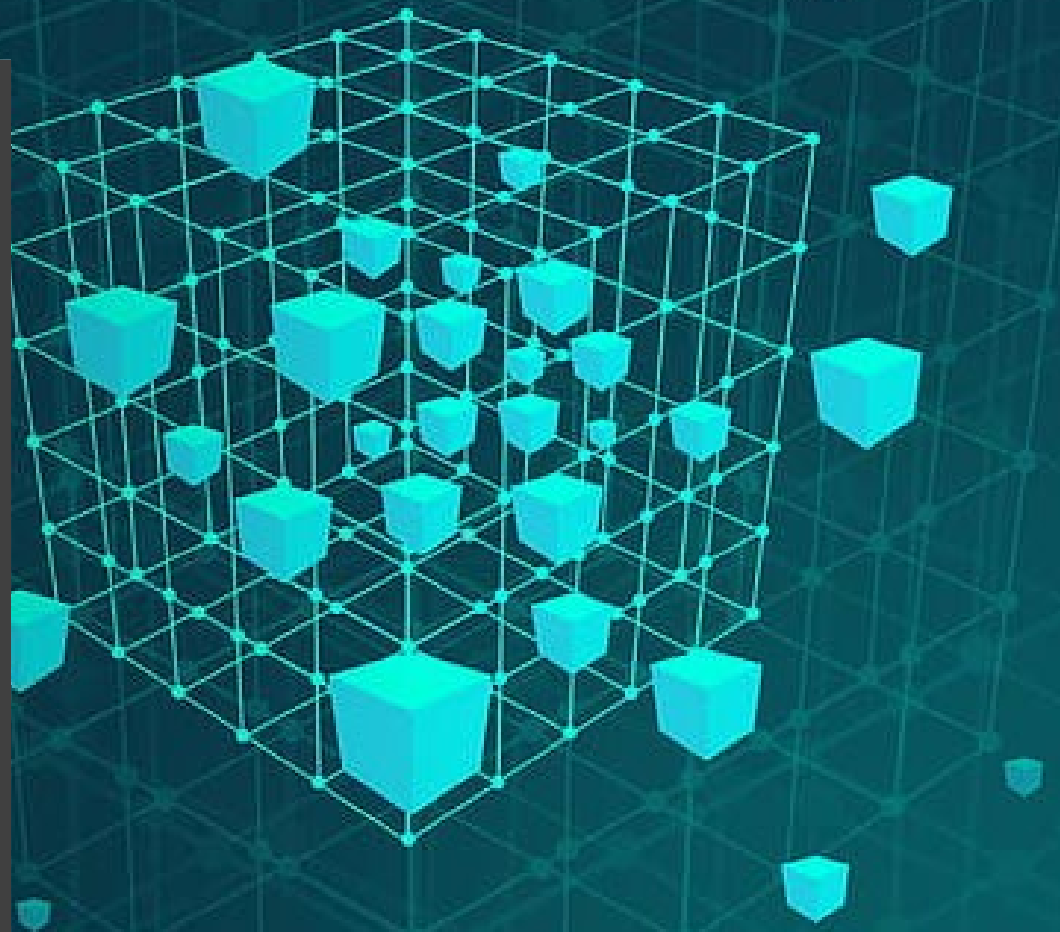
Introduction to Keras and TensorFlow

Adhi Harmoko Saputro

What's TensorFlow?

TensorFlow (https://www.tensorflow.org/) is a Python-based, free, open source machine learning platform, developed primarily by Google

- To enable engineers and researchers to manipulate mathematical expressions over numerical tensors



What's TensorFlow?

- It can automatically compute the gradient of any differentiable expression, making it highly suitable for both machine or deep learning
- It can run not only on CPUs, but also on GPUs and TPUs, highly parallel hardware accelerators
- Computation defined in TensorFlow can be easily distributed across many machines
- TensorFlow programs can be exported to other runtimes, such as C++, Java-Script (for browser-based applications), or TensorFlow Lite (for applications running on mobile devices or embedded devices), etc
  - This makes TensorFlow applications easy to deploy in practical settings

TensorFlow Platform

- It's important to keep in mind that TensorFlow is much more than a single library
  - It's really a platform, home to a vast ecosystem of components, some developed by Google and some developed by third parties
  - Example:
    - TF-Agents for reinforcement-learning research
    - TFX for industry-strength machine learning workflow management

- TensorFlow scales fairly well
  - Oak Ridge National Lab have used it to train a 1.1 exaFLOPS extreme weather forecasting model on the 27,000 GPUs of the IBM Summit supercomputer
  - Google has used Tensor-Flow to develop very compute-intensive deep learning applications, such as the chessplaying and Go-playing agent AlphaZero

What's Keras?

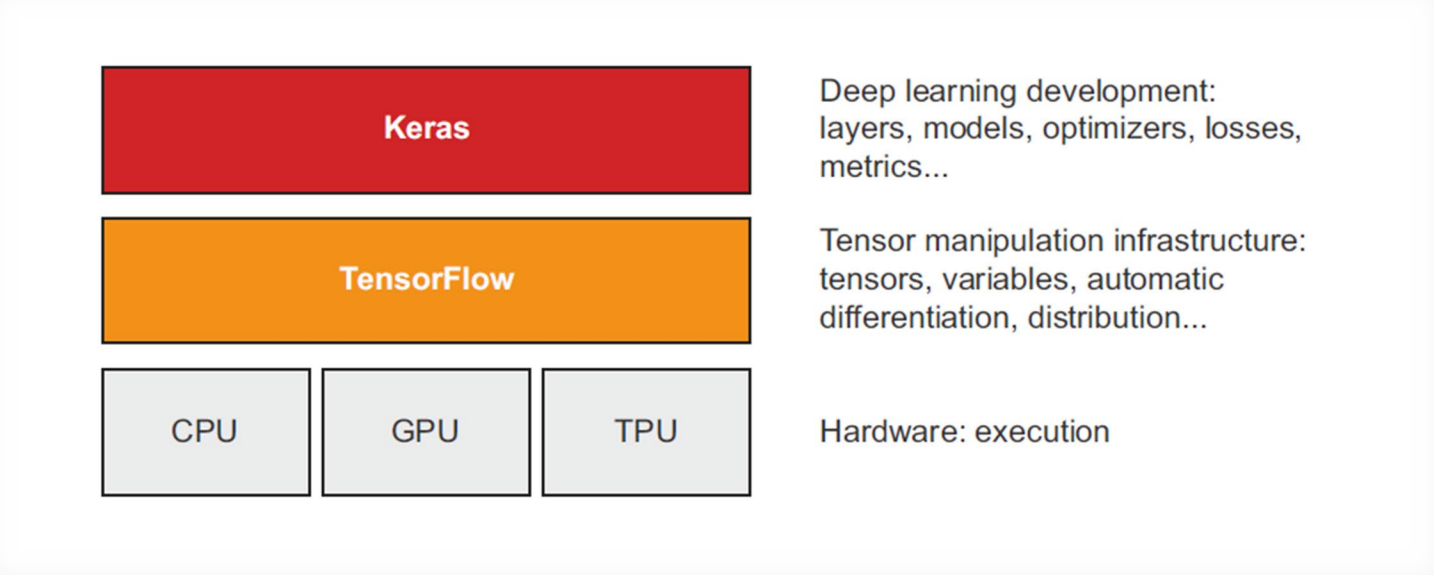
Keras (https://keras.io/) is a deep learning API for Python, built on top of TensorFlow, that provides a convenient way to define and train any kind of deep learning model

What's Keras?

- Simple -- but not simplistic
  - Keras reduces developer cognitive load to free you to focus on the parts of the problem that really matter.
- Flexible -- Keras adopts the principle of progressive disclosure of complexity
  - Simple workflows should be quick and easy, while arbitrarily advanced workflows should be possible via a clear path that builds upon what you've already learned.
- Powerful -- Keras provides industry-strength performance and scalability
  - It is used by organizations and companies including NASA, YouTube, or Waymo

What's Keras?

- Initially developed for research, with the aim of enabling fast deep learning experimentation
- Keras can run on top of different types of hardware through TensorFlow
  - GPU, TPU, or plain CPU—and can be seamlessly scaled to thousands of machines
- Keras and TensorFlow: TensorFlow is a low-level tensor computing platform, and Keras is a high-level deep learning API



What's Keras?

- Keras has well over a million users as of late 2021, ranging from academic researchers, engineers, and data scientists at both startups and large companies to graduate students and hobbyists

- Keras is used at Google, Netflix, Uber, CERN, NASA, Yelp, Instacart, Square, and hundreds of startups working on a wide range of problems across every industry
  - YouTube recommendations originate from Keras models.
  - The Waymo self-driving cars are developed with Keras models
  - Keras is also a popular framework on Kaggle, the machine learning competition website, where most deep learning competitions have been won using Keras

- Keras doesn't force you to follow a single "true" way of building and training models
  - Enables a wide range of different workflows, from the very high level to the very low level, corresponding to different user profiles
  - An array of ways to build models and an array of ways to train them, each representing a certain trade-off between usability and flexibility

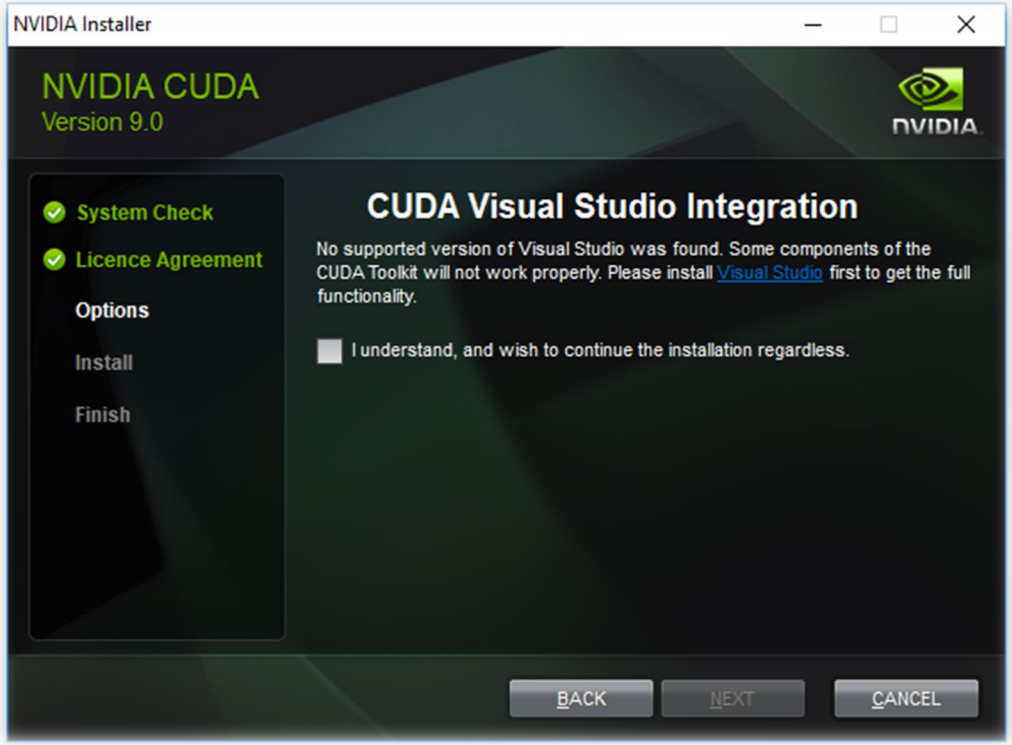
Setting Up a Deep Learning Workspace

Run deep learning code on a modern NVIDIA GPU rather than your computer's CPU

- Buy and install a physical NVIDIA GPU on your workstation
- Use GPU instances on Google Cloud or AWS EC2
- Use the free GPU runtime from Colaboratory, a hosted notebook service offered by Google

Setting Up a Deep Learning Workspace

- Installing CUDA, cuDNN and GPU support on Windows 10
  - Guide: https://towardsdatascience.com/installing-tensorflow-with-cuda-cudnn-and-gpu-support-on-windows-10-60693e46e781
  - Cuda Toolkits Documentations: https://docs.nvidia.com/cuda/cuda-installation-guide-microsoft-windows/index.html



TensorFlow Installation

- Install TensorFlow with pip (https://www.tensorflow.org/install/pip)
  - pip install tensorflow

- TensorFlow GPU Support (https://www.tensorflow.org/install/gpu)
  - NVIDIA® GPU drivers
  - CUDA® Toolkit
  - CUPTI
  - cuDNN SDK 8.1.0
  - TensorRT 7

Check TensorFlow Installation

- Check TensorFlow installation using pip command

- Check TensorFlow & Cuda installation using Python

```
pip show tensorflow



# importing the tensorflow package
import tensorflow as tf

tf.test.is_built_with_cuda()




tf.config.list_physical_devices('GPU’)
```

Check Keras Installation

- Check Keras using pip
- Check Keras using Command Line or Windows Terminal
- To use Keras, will need to have the TensorFlow package installed
  - See at https://keras.io/getting_started/

```
pip show keras


# Command Line
python -c 'import keras; print(keras.__version__)’

# Windows Terminal
python -c "import keras; print(keras.__version__)"



from tensorflow import keras
```

THANK YOU!

Adhi Harmoko Saputro