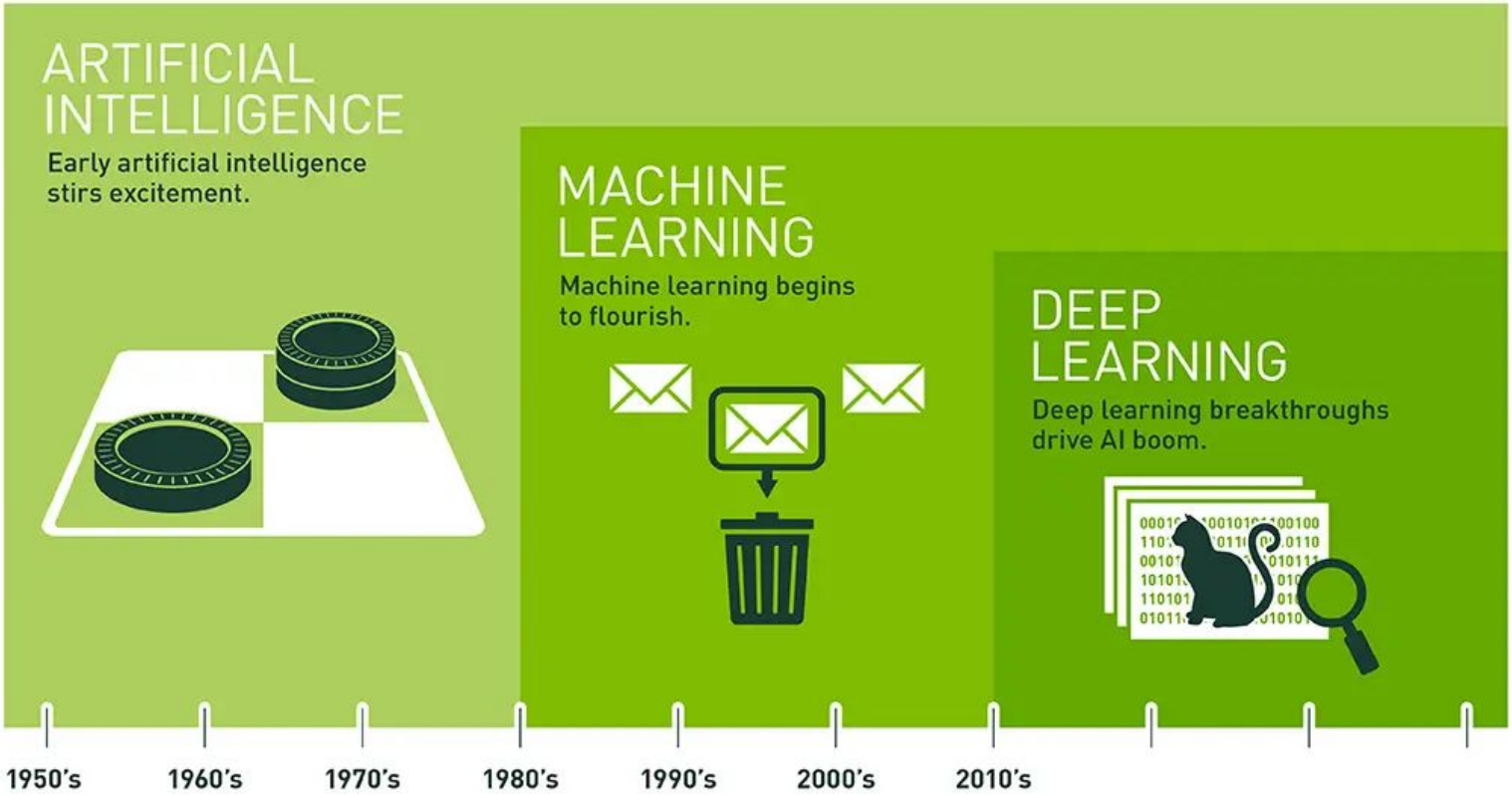
HOCHSCHULE DER MEDIEN

Deep Learning

Introduction to TensorFlow and Keras

Prof. Dr. Jan Kirenz
HdM Stuttgart

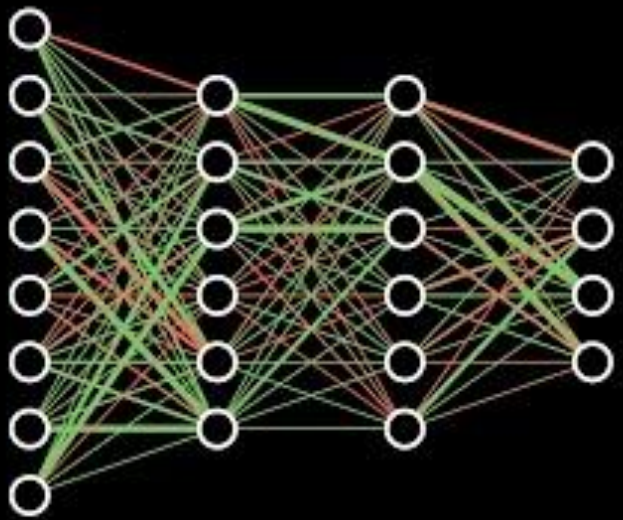


Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Source: Nvidia (2016)

Source: Moroney (2019)

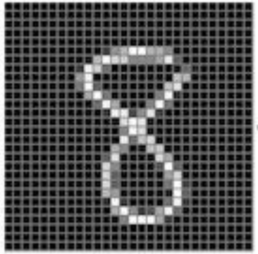
Neural Networks



From the ground up

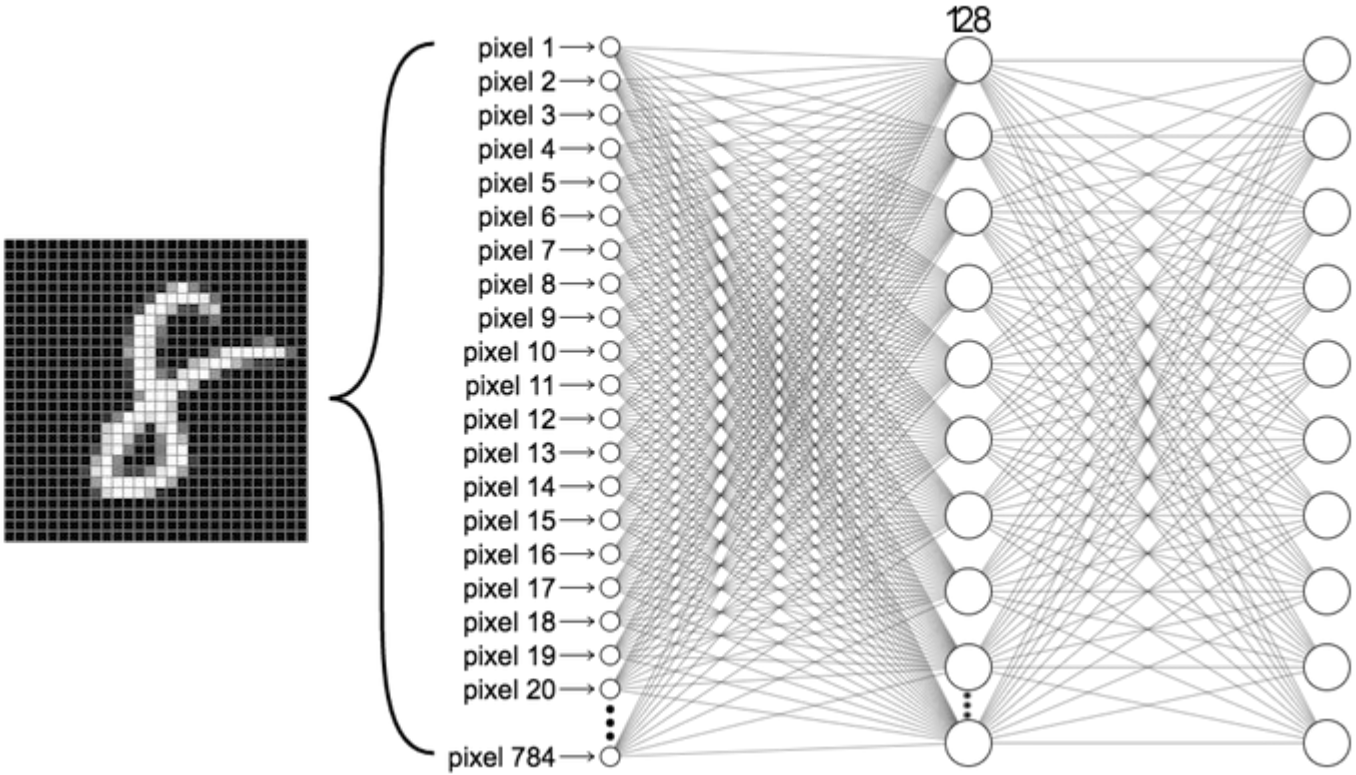
MNIST database

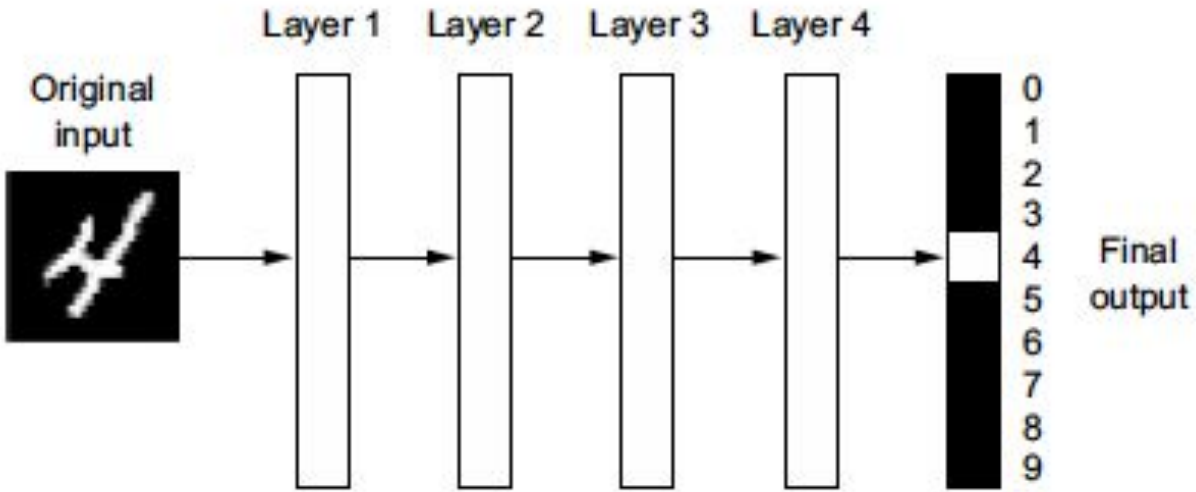
MNIST database (*Modified National Institute of Standards and Technology database*[1]) is a large database of handwritten digits that is commonly used for training various image processing systems



28 x 28
784 pixels

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	4	62	146	182	254	254	181	176	139	15	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	34	186	253	217	208	136	136	136	166	232	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	61	242	208	111	3	0	0	0	0	0	18	32	107	43	0	0	0	0	0	0
0	0	0	0	0	0	0	0	156	242	23	0	0	0	0	0	0	0	13	191	181	6	0	0	0	0	0	0
0	0	0	0	0	0	0	0	121	255	98	3	0	0	0	0	0	8	194	225	12	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	169	253	120	3	0	0	0	0	128	247	51	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	3	111	244	169	19	0	14	131	249	117	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	59	241	235	72	142	229	66	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	25	218	254	231	36	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	133	253	221	33	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	19	237	111	196	217	19	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	138	0	23	193	204	18	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	96	224	0	0	0	25	218	169	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	138	0	0	0	0	86	253	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	3	162	214	11	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	0	118	253	68	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	185	157	0	0	0	0	0	40	254	98	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	50	244	61	0	0	0	0	112	244	58	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	251	142	59	83	167	244	111	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	6	133	253	253	253	169	61	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0





Source: Chollet (2021)

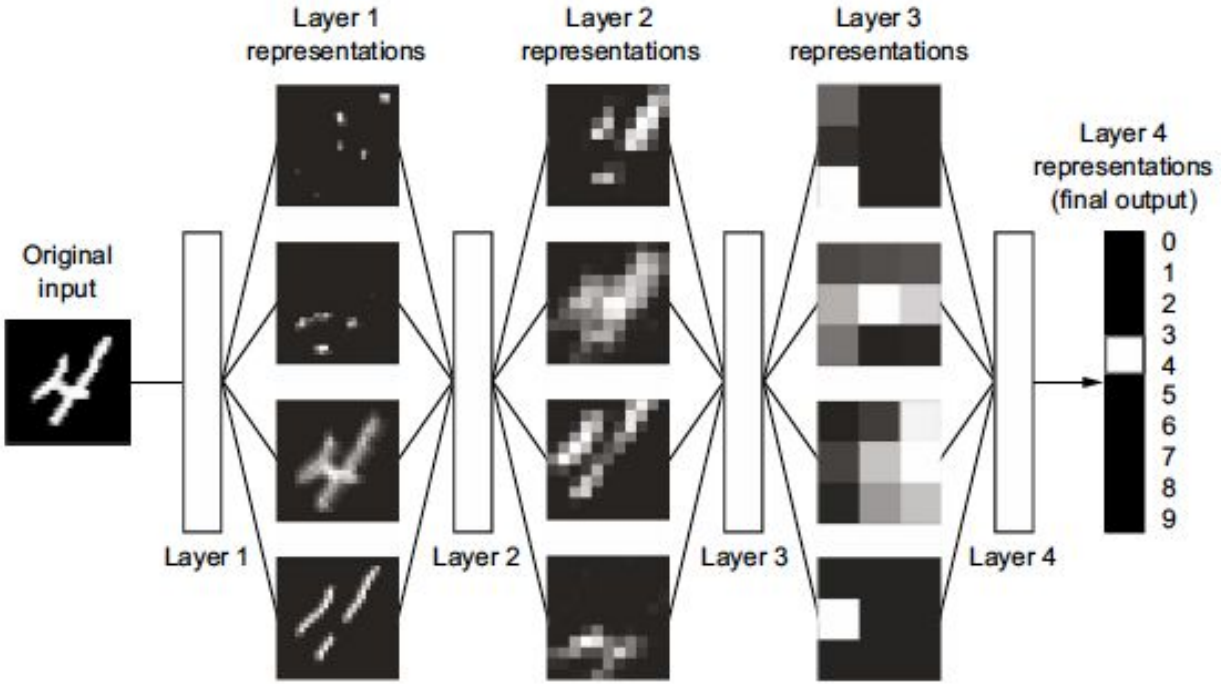
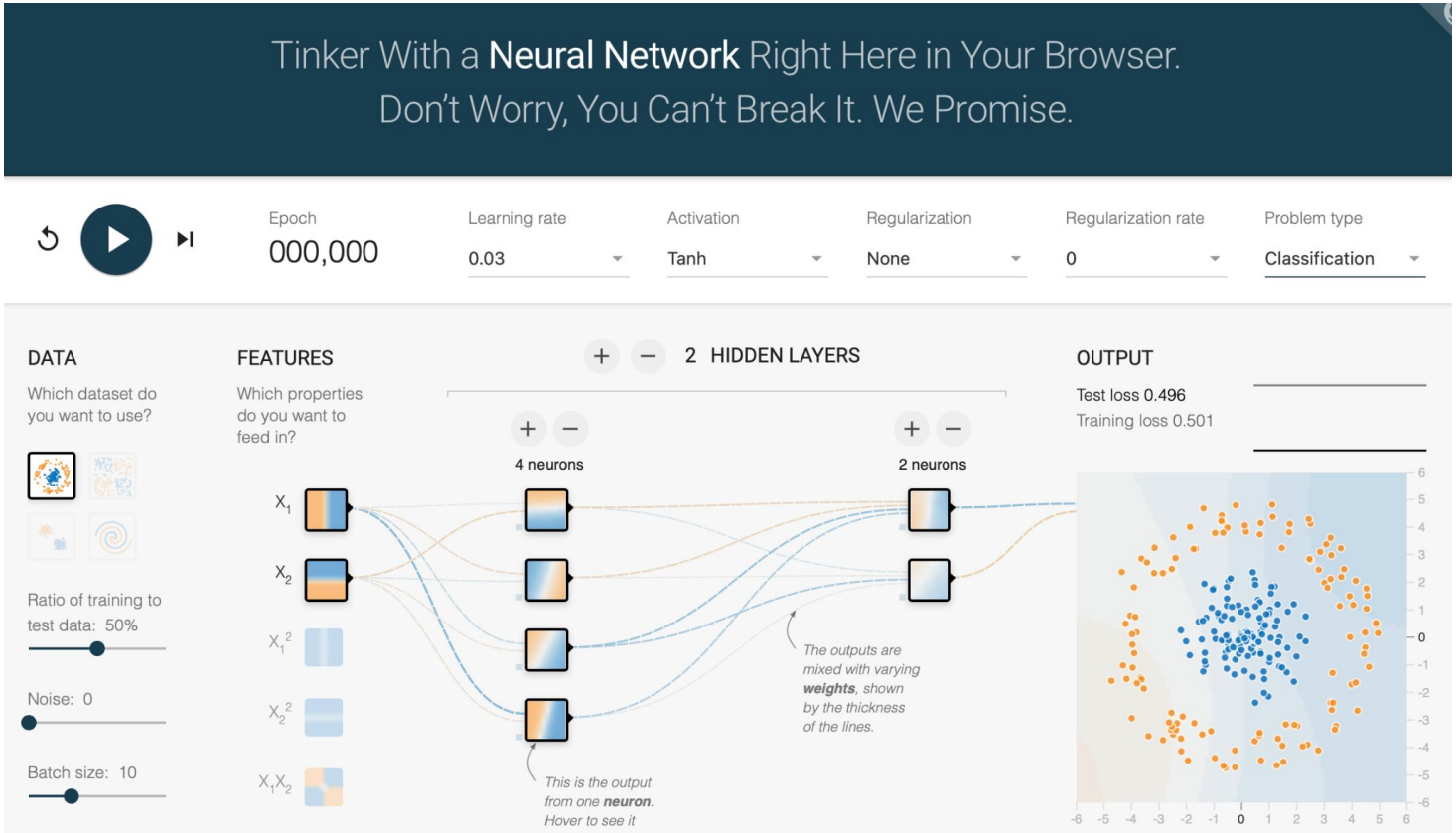


Figure 1.6 Data representations learned by a digit-classification model

Source: Chollet (2021)



A Neural Network Playgroundhttps://playground.tensorflow.org

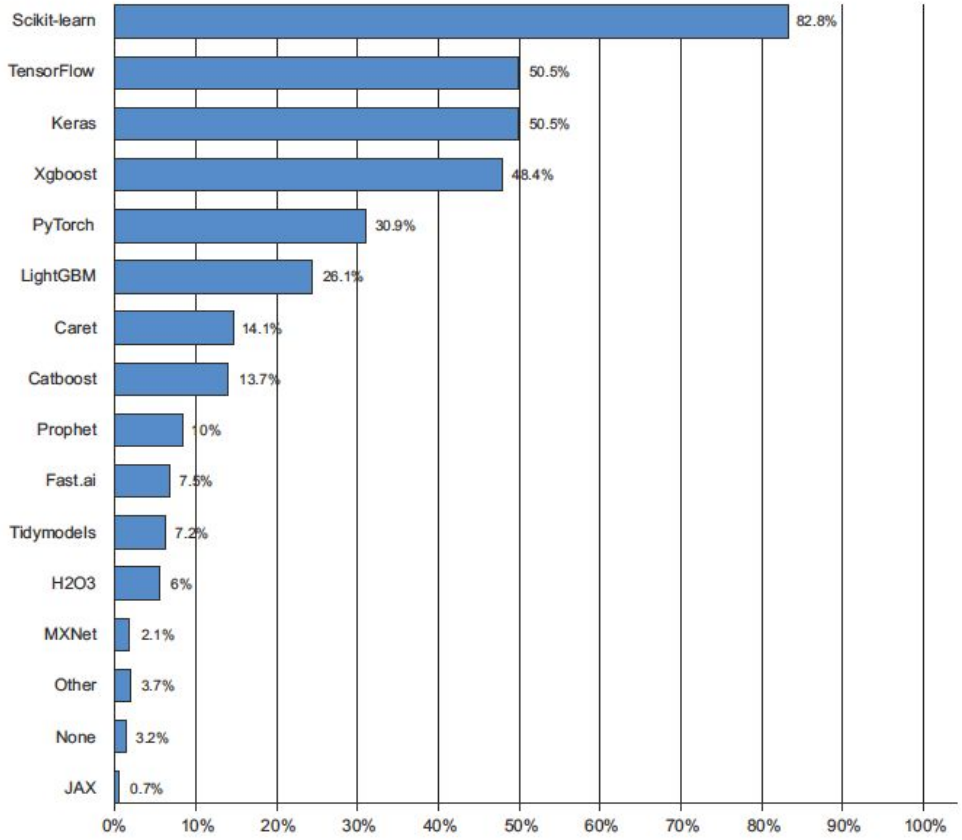
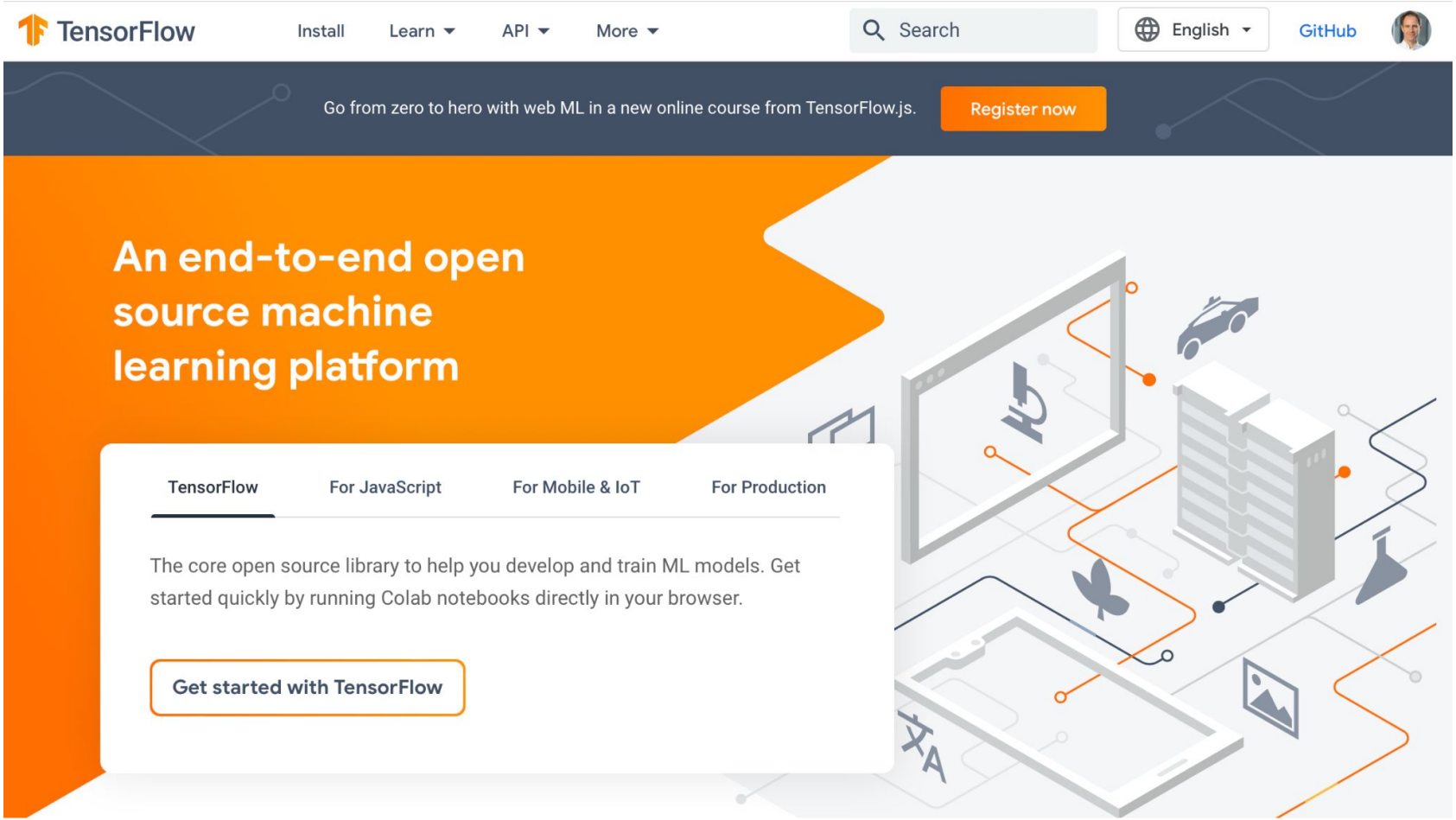


Figure 1.13 Tool usage across the machine learning and data science industry (Source: www.kaggle.com/kaggle-survey-2020)

From 2016 to 2020, the entire machine learning and data science industry has been dominated by two approaches: deep learning and gradient boosted trees.

Source: Chollet (2021)



Source: TensorFlow

Companies using TensorFlow

See case studies →

Source: TensorFlow

Keras

Simple. Flexible. Powerful.

Get started | API docs | Guides | Examples

```
from tensorflow import keras
from tensorflow.keras import layers

# Instantiate a trained vision model
vision_model = keras.applications.ResNet50()

# This is our video.encoding branch using the trained vision_model
video_input = keras.Input(shape=(100, None, None, 3))
encoded_frame_sequence = layers.TimeDistributed(vision_model)(video_input)
encoded_video = layers.LSTM(256)(encoded_frame_sequence)

# This is our text-processing branch for the question input
question_input = keras.Input(shape=(100,), dtype='int32')
embedded_question = layers.Embedding(10000, 256)(question_input)
encoded_question = layers.LSTM(256)(embedded_question)

# And this is our video question answering model:
merged = keras.layers.concatenate([encoded_video, encoded_question])
output = keras.layers.Dense(1000, activation='softmax')(merged)
video_qa_model = keras.Model(inputs=[video_input, question_input],
                             outputs=output)
```

A vast ecosystem.

Keras is a central part of the tightly-connected TensorFlow 2 ecosystem, covering every step of the machine learning workflow, from data management to hyperparameter training to deployment solutions.

Source: Keras

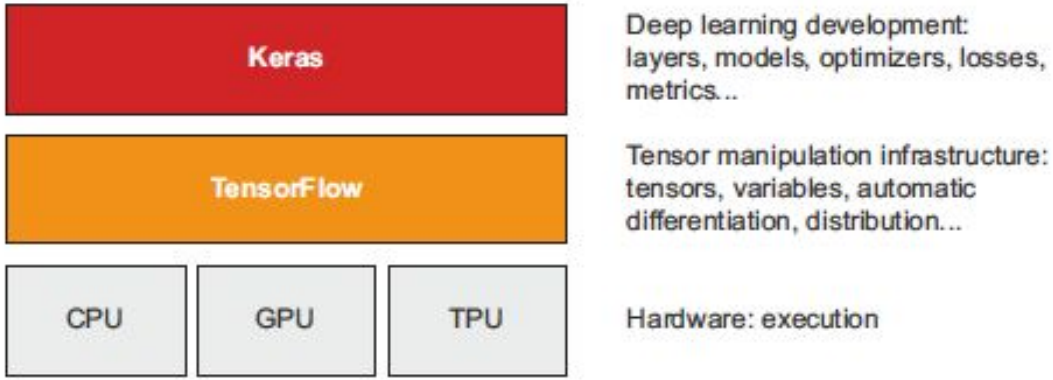


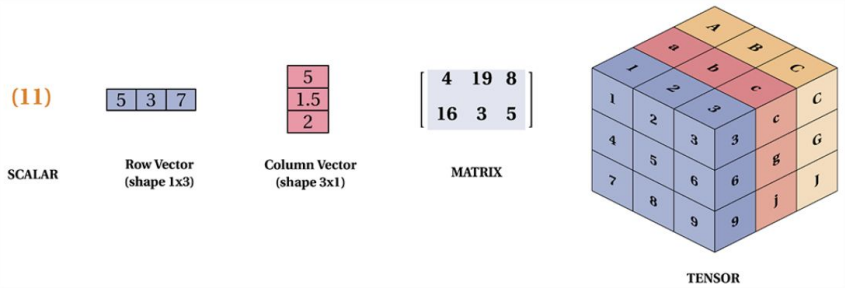
Figure 3.1 Keras and TensorFlow: TensorFlow is a low-level tensor computing platform, and Keras is a high-level deep learning API

Source: Chollet (2021)

What is a Tensor?

What is Tensor?

Tensor holds a multi-dimensional array of elements of a single data type which is very similar with numpy's ndarray. When the dimension is zero, it can be called a scalar. When the dimension is 2, it can be called a matrix. When the dimension is greater than 2, it is usually called a tensor. If you are very familiar with numpy, then understanding Tensor will be quite easy.



Introduction to Tensors

Run in Google Colab | View source on GitHub | Download notebook

A tensor is an N-dimensional array of data

Rank 0 Tensor
scalar

Rank 1 Tensor
vector

Rank 2 Tensor
matrix

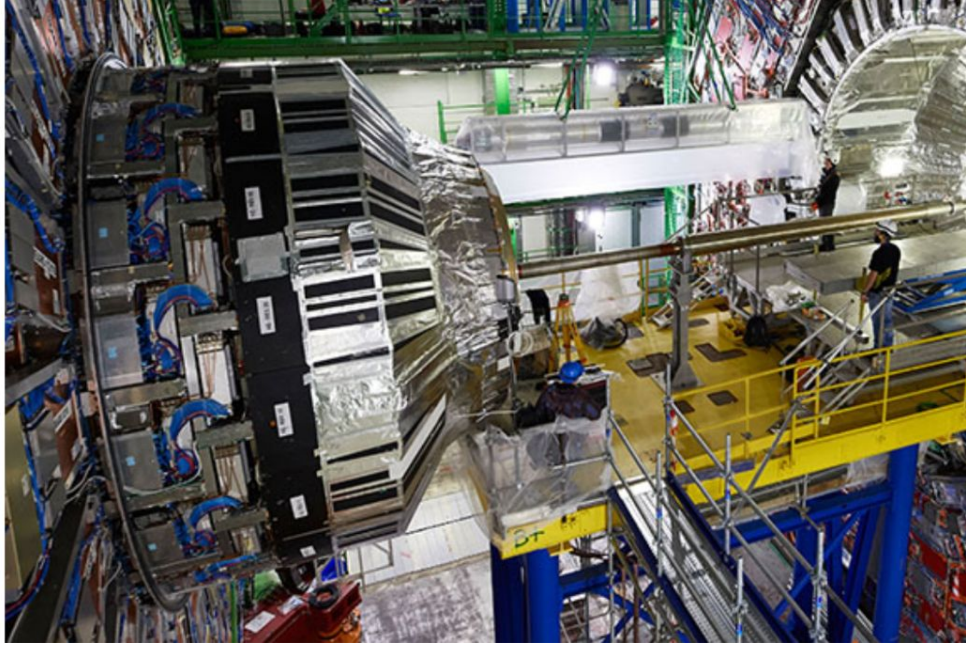
Rank 3 Tensor

Rank 4 Tensor

Source: https://medium.com/mlait/tensors-representation-of-data-in-neural-networks-bbe8a711b93b

State-of-the-art research.

Keras is used by CERN, NASA, NIH, and many more scientific organizations around the world (and yes, Keras is used at the LHC). Keras has the low-level flexibility to implement arbitrary research ideas while offering optional high-level convenience features to speed up experimentation cycles.

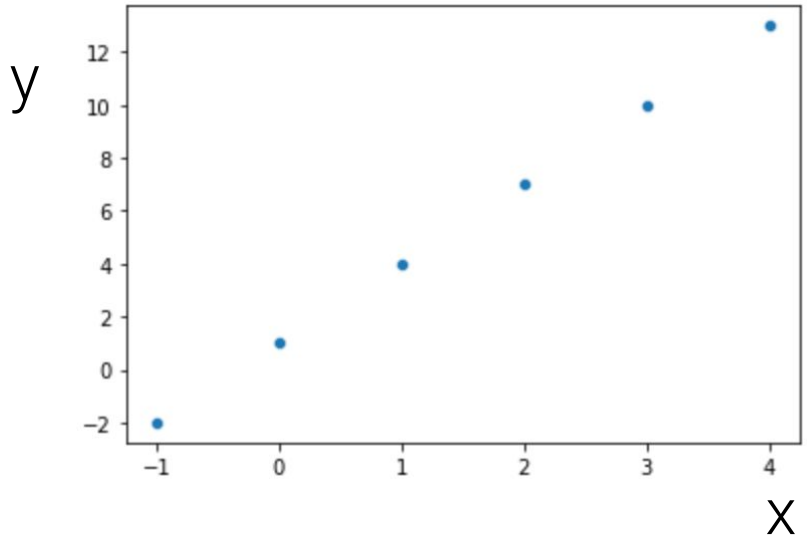


Sorce: Keras

Simple regression example with TensorFlow & Keras

Prof. Dr. Jan Kirenz

Variable	Value 1	Value 2	Value 3	Value 4	Value 5	Value 6
x	-1	0	1	2	3	4
y	-2	1	4	7	10	13



Source: Moroney (2019)

import libraries

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
```

```
# Data
x = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
y = np.array([-2.0, 1.0, 4.0, 7.0, 10.0, 13.0], dtype=float)

# Model definition
model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

# Model fitting
model.fit(x, y, epochs=50)

# Model prediction
print(model.predict([10.0]))
```

Source: Moroney (2019)

create data

```
import numpy as np
import tensorflow as tf
from tensorflow import keras

# Data
x = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
y = np.array([-2.0, 1.0, 4.0, 7.0, 10.0, 13.0], dtype=float)

# Model definition
model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

# Model fitting
model.fit(x, y, epochs=50)

# Model prediction
print(model.predict([10.0]))
```

Source: Moroney (2019)

define model architecture

```
import numpy as np
import tensorflow as tf
from tensorflow import keras

# Data
x = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
y = np.array([-2.0, 1.0, 4.0, 7.0, 10.0, 13.0], dtype=float)

# Model definition
model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

# Model fitting
model.fit(x, y, epochs=50)

# Model prediction
print(model.predict([10.0]))
```

Source: Moroney (2019)

we use a single layer

```
import numpy as np
import tensorflow as tf
from tensorflow import keras

# Data
x = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
y = np.array([-2.0, 1.0, 4.0, 7.0, 10.0, 13.0], dtype=float)

# Model definition
model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

# Model fitting
model.fit(x, y, epochs=50)

# Model prediction
print(model.predict([10.0]))
```

Source: Moroney (2019)

with one neuron

```
import numpy as np
import tensorflow as tf
from tensorflow import keras

# Data
x = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
y = np.array([-2.0, 1.0, 4.0, 7.0, 10.0, 13.0], dtype=float)

# Model definition
model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

# Model fitting
model.fit(x, y, epochs=50)

# Model prediction
print(model.predict([10.0]))
```

Source: Moroney (2019)

and only one input x

```
import numpy as np
import tensorflow as tf
from tensorflow import keras

# Data
x = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
y = np.array([-2.0, 1.0, 4.0, 7.0, 10.0, 13.0], dtype=float)

# Model definition
model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

# Model fitting
model.fit(x, y, epochs=50)

# Model prediction
print(model.predict([10.0]))
```

Source: Moroney (2019)

we compile the model

```
import numpy as np
import tensorflow as tf
from tensorflow import keras

# Data
x = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
y = np.array([-2.0, 1.0, 4.0, 7.0, 10.0, 13.0], dtype=float)

# Model definition
model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

# Model fitting
model.fit(x, y, epochs=50)

# Model prediction
print(model.predict([10.0]))
```

Source: Moroney (2019)

generate a guess for y

sgd = stochastic gradient descent

```
import numpy as np
import tensorflow as tf
from tensorflow import keras

# Data
x = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
y = np.array([-2.0, 1.0, 4.0, 7.0, 10.0, 13.0], dtype=float)

# Model definition
model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

# Model fitting
model.fit(x, y, epochs=50)

# Model prediction
print(model.predict([10.0]))
```

Source: Moroney (2019)

and calculate the error

```
import numpy as np
import tensorflow as tf
from tensorflow import keras

# Data
x = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
y = np.array([-2.0, 1.0, 4.0, 7.0, 10.0, 13.0], dtype=float)

# Model definition
model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

# Model fitting
model.fit(x, y, epochs=50)

# Model prediction
print(model.predict([10.0]))
```

Source: Moroney (2019)

```
import numpy as np
import tensorflow as tf
from tensorflow import keras

# Data
x = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
y = np.array([-2.0, 1.0, 4.0, 7.0, 10.0, 13.0], dtype=float)

# Model definition
model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

# Model fitting
model.fit(x, y, epochs=50)

# Model prediction
print(model.predict([10.0]))
```

we do this 50 times

Source: Moroney (2019)

predict y for x=10

```
import numpy as np
import tensorflow as tf
from tensorflow import keras

# Data
x = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
y = np.array([-2.0, 1.0, 4.0, 7.0, 10.0, 13.0], dtype=float)

# Model definition
model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

# Model fitting
model.fit(x, y, epochs=50)

# Model prediction
print(model.predict([10.0]))
```

Source: Moroney (2019)

Resources

The slides are based on the excellent video tutorial “Intro to Machine Learning (ML Zero to Hero - Part 1)” by Lawrence Moroney.