Introduction to Deep Learning and TensorFlow 2.0

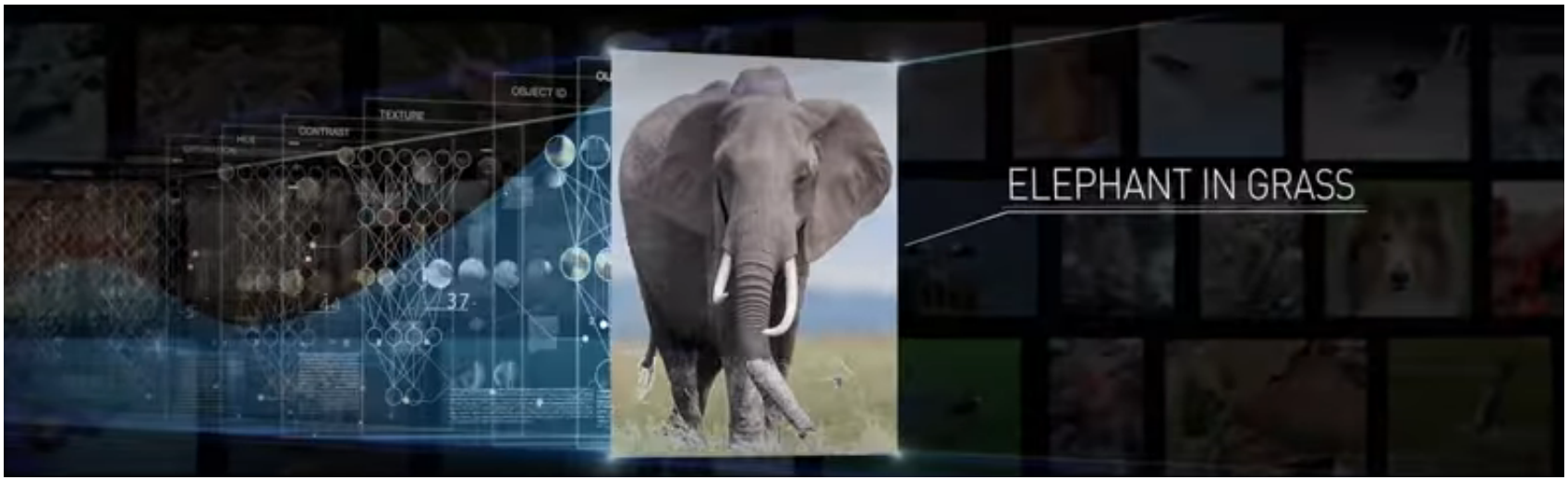
Yogesh Kulkarni

September 12, 2020

Introduction to Deep Learning

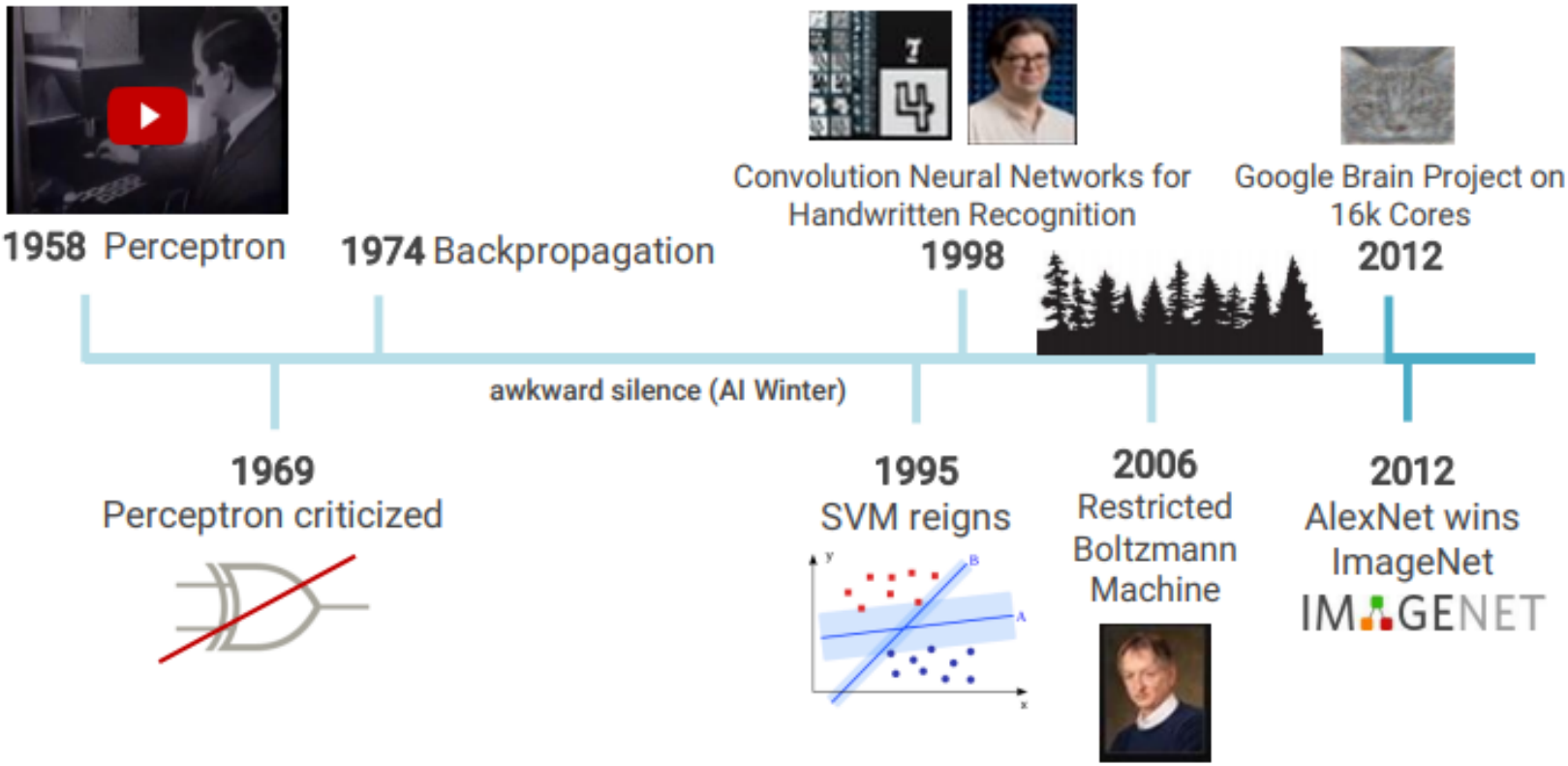
Deep Learning is

- A big bang of Artificial Intelligence.
- making Spark of innovation of most amazing break-through
- having Super human capabilities in image recognition
- making remarkable progress in text, speech, automobiles
- beating humans at Chess and Go



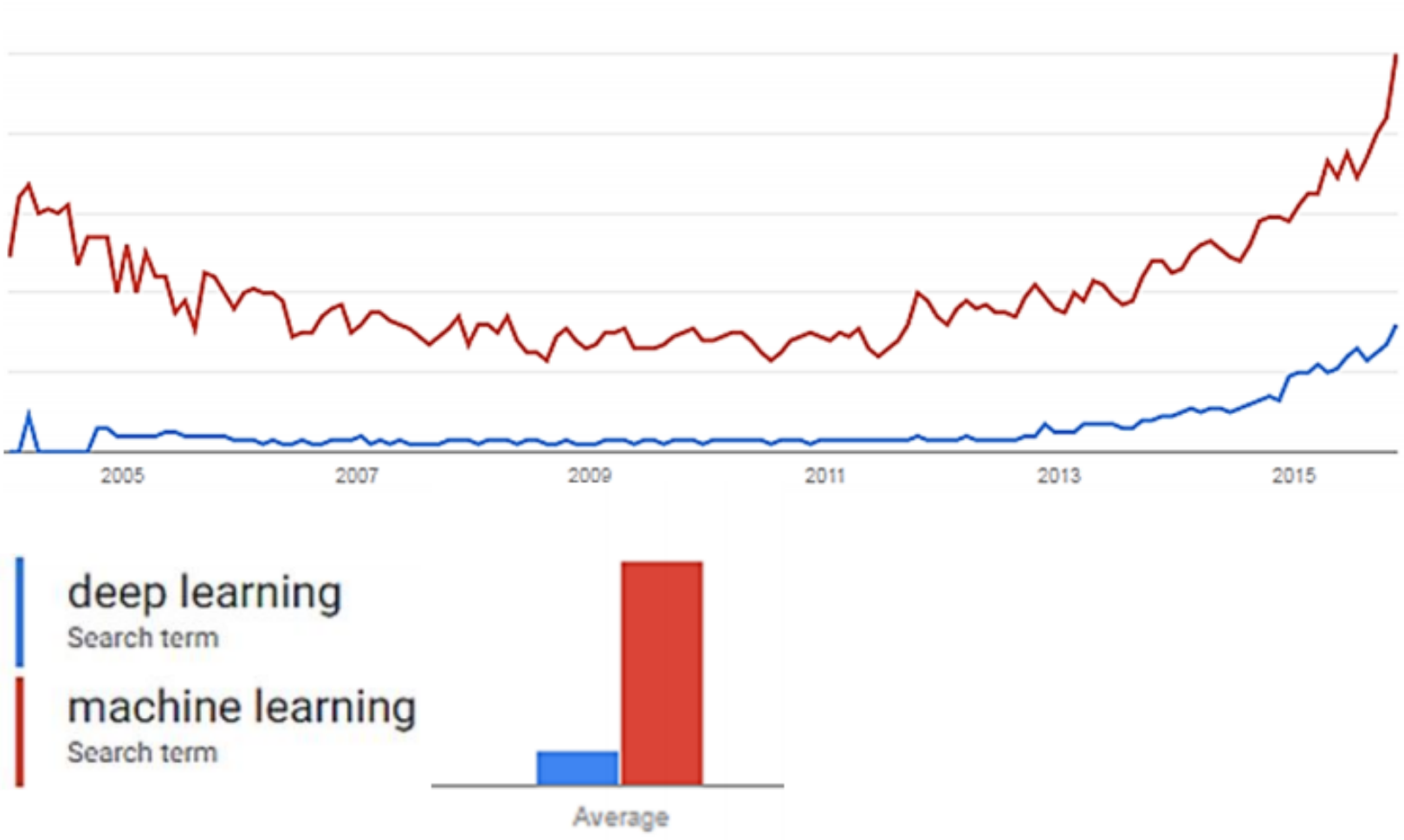
(The Deep Learning Revolution - Nvidia)

History



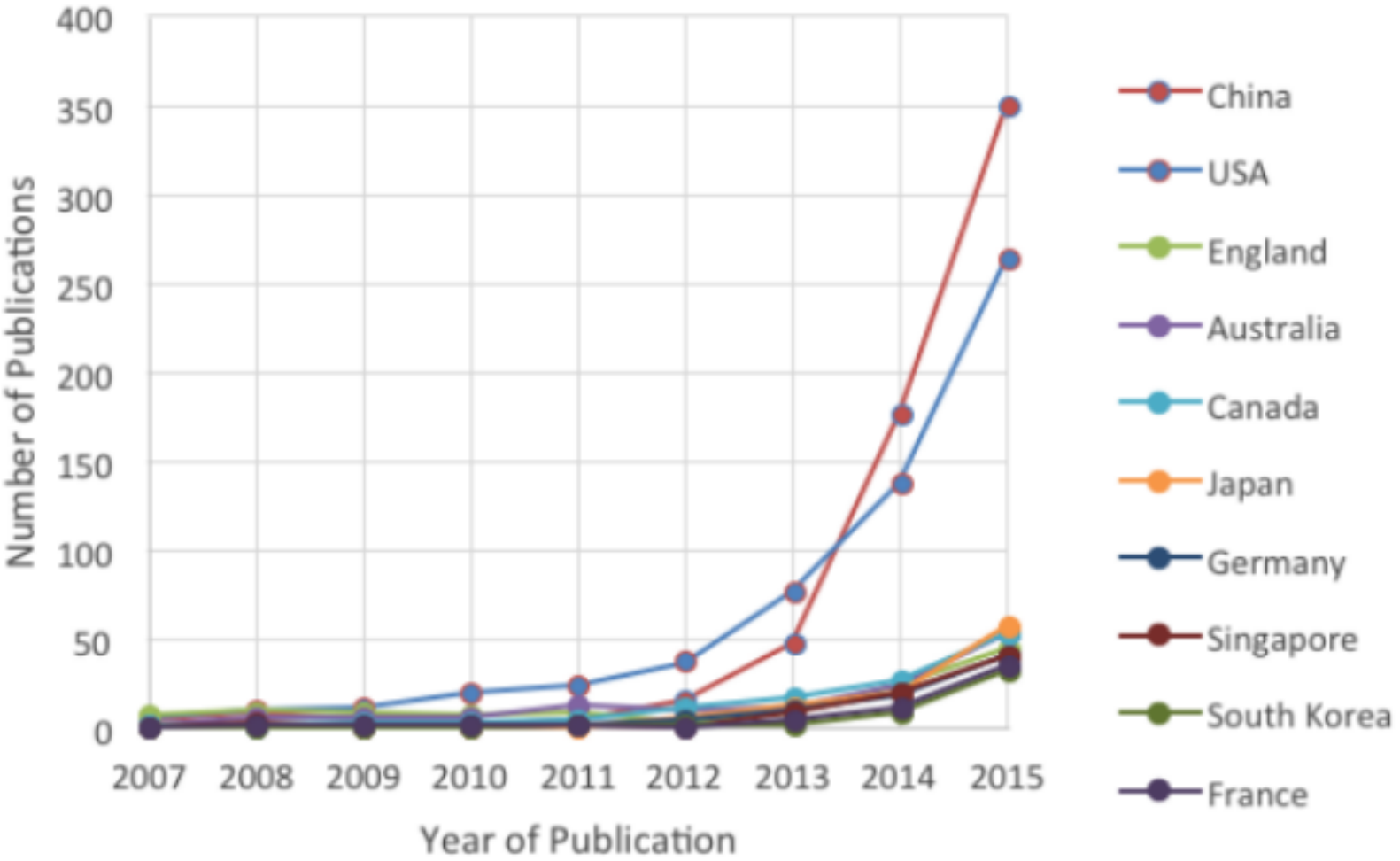
(Deep Learning - The Past, Present and Future of Artificial Intelligence - Lukas Masuch)

Interest : Google Trends



(Deep Learning - The Past, Present and Future of Artificial Intelligence - Lukas Masuch)

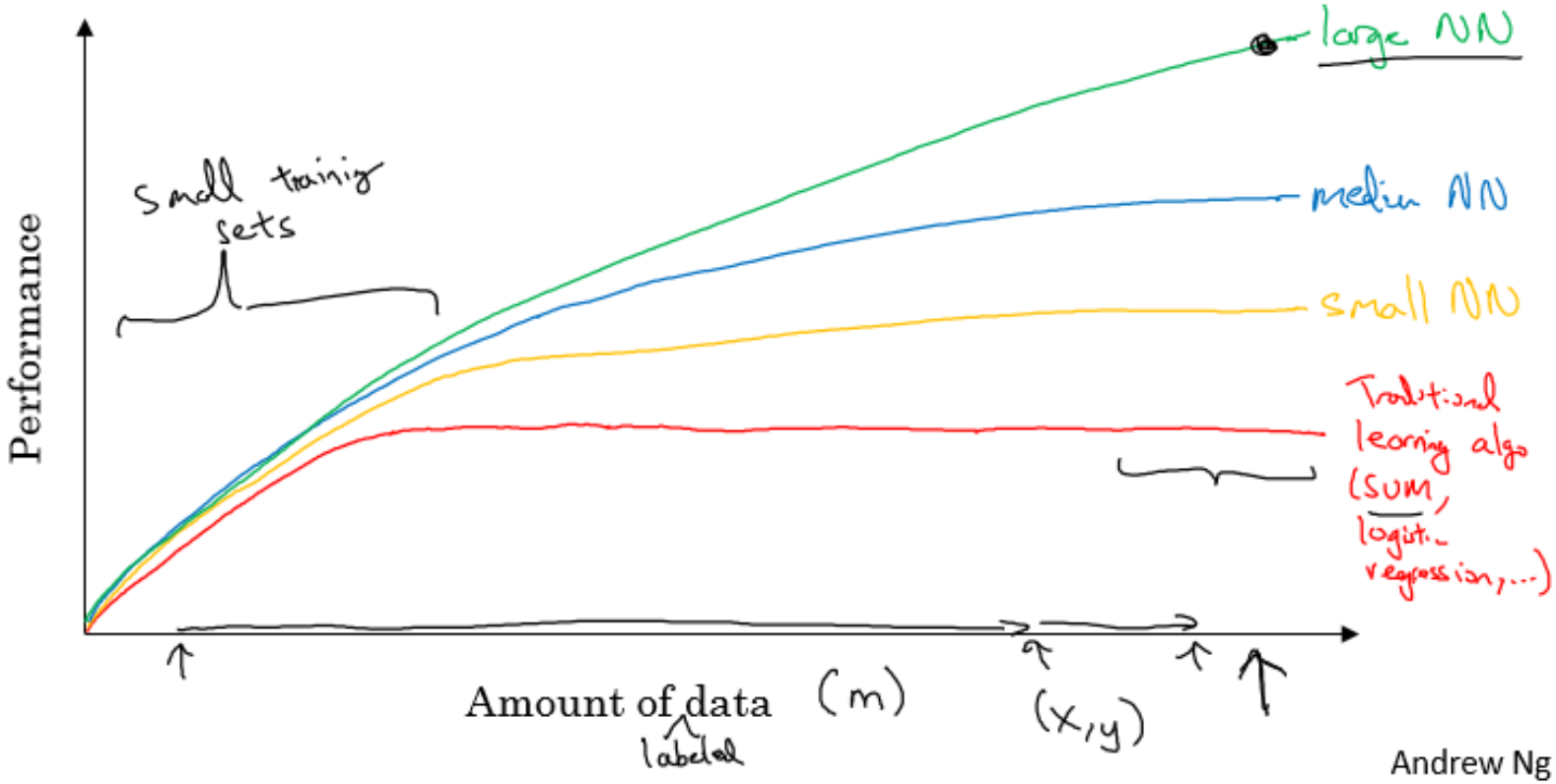
Academic Publications in Deep Learning



(Deep Learning - The Past, Present and Future of Artificial Intelligence - Lukas Masuch)

Why Deep Learning is taking off?

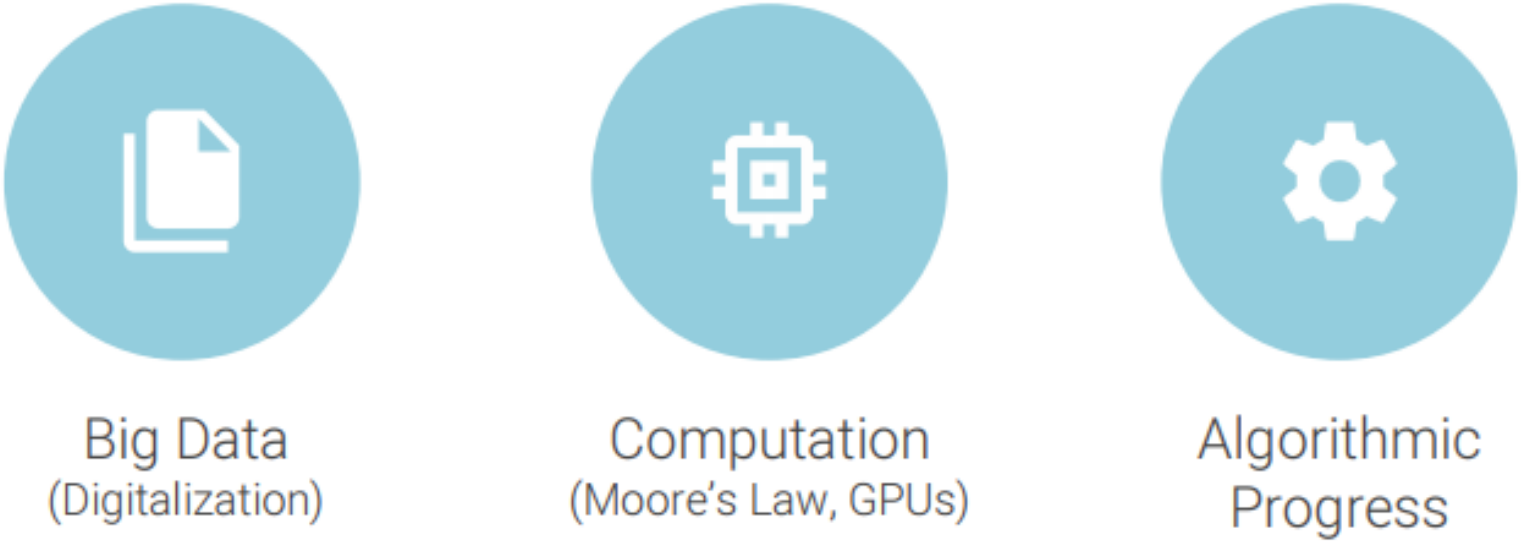
Deep learning is taking off due to a large amount of data available through the digitization of the society, faster computation and innovation in the development of neural network algorithm.



(Reference: Introduction to Neural Networks - Andrew Ng)

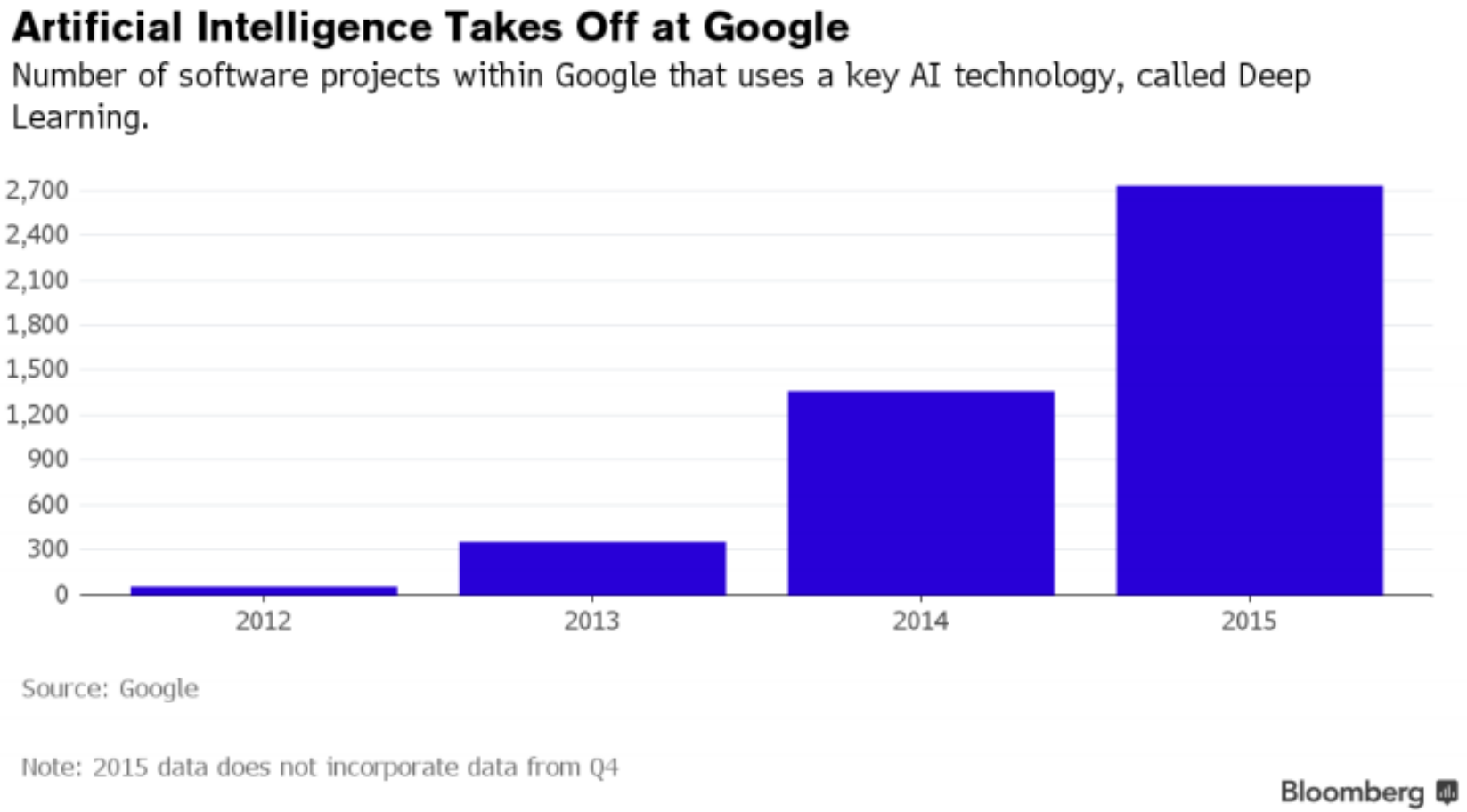
What changed?

Old wine in new bottles



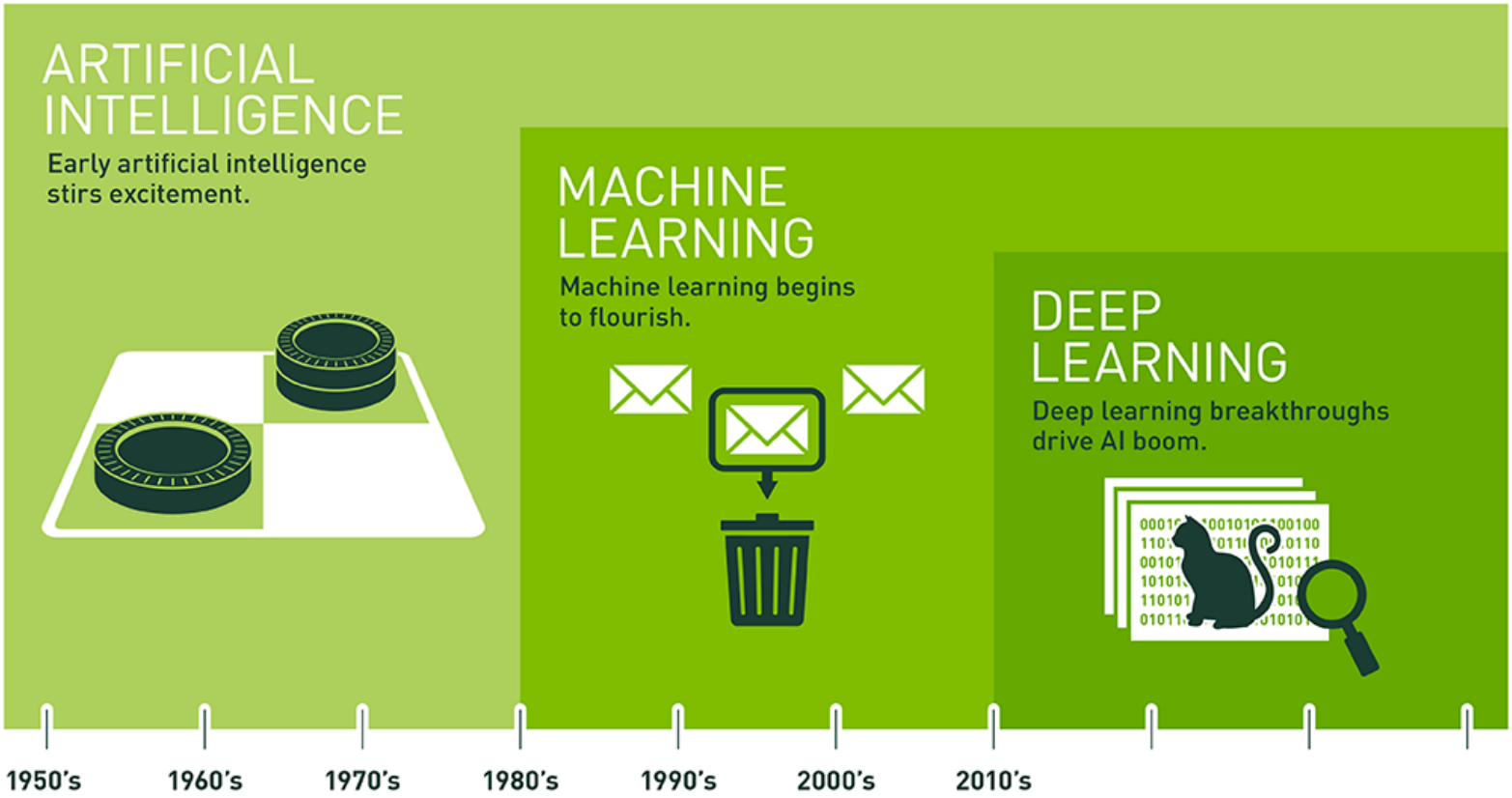
(Deep Learning - The Past, Present and Future of Artificial Intelligence - Lukas Masuch)

Deep Learning: Hype or Reality



(Deep Learning - The Past, Present and Future of Artificial Intelligence - Lukas Masuch)

AI ML DL: What's the difference?



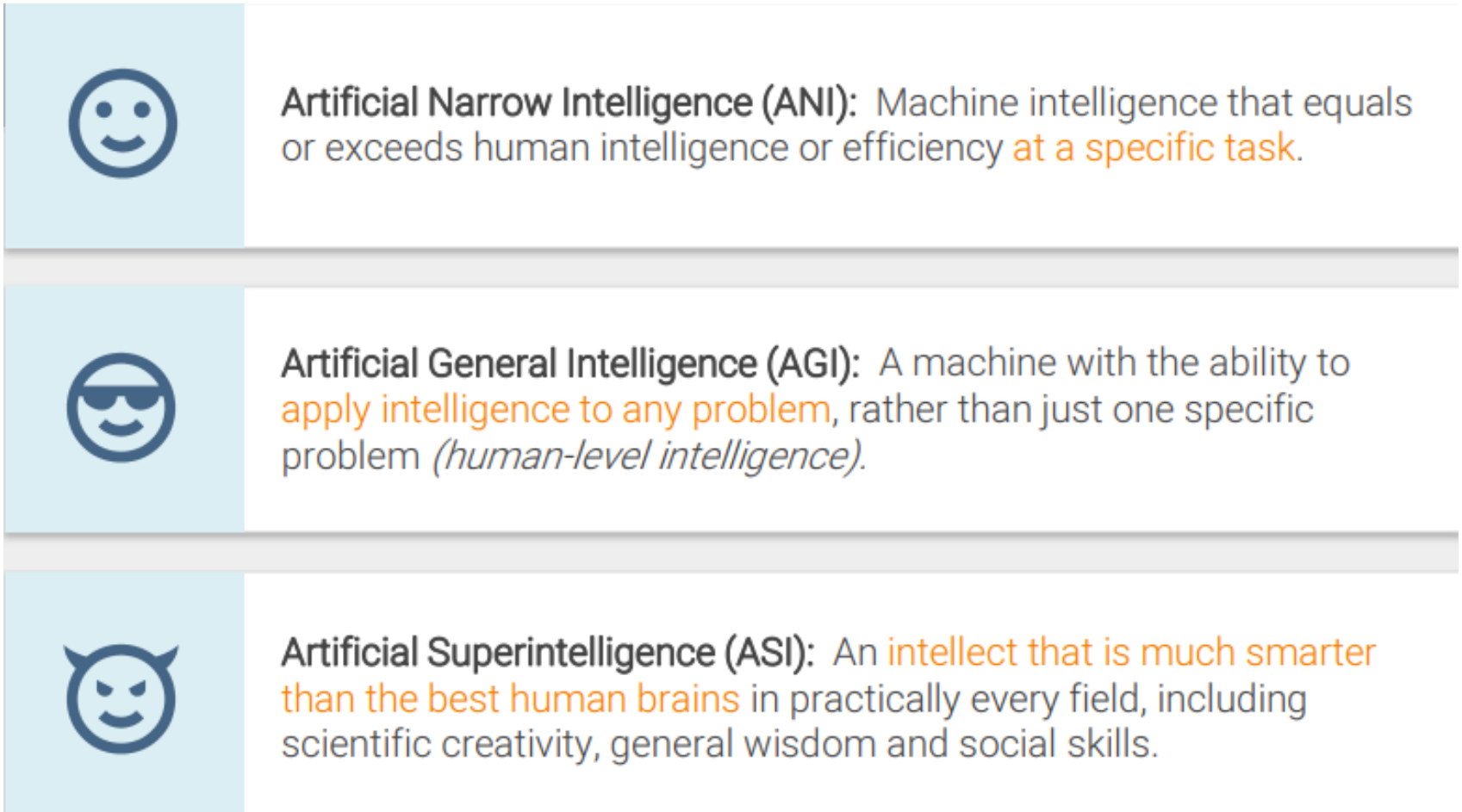
Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

(Reference: The Difference Between AI, Machine Learning, and Deep Learning - NVIDIA Blog)

AI ML DL: What's the difference?

- Artificial Intelligence: mimicking human intelligence
- Machine Learning: Automating Learning with features.
- There could be programmed (hand coded) AI, that's not Machine Learning
- Machine Learning could be for non AI activities, like automation
- Deep Learning: Neural network with no input features

Types of AI

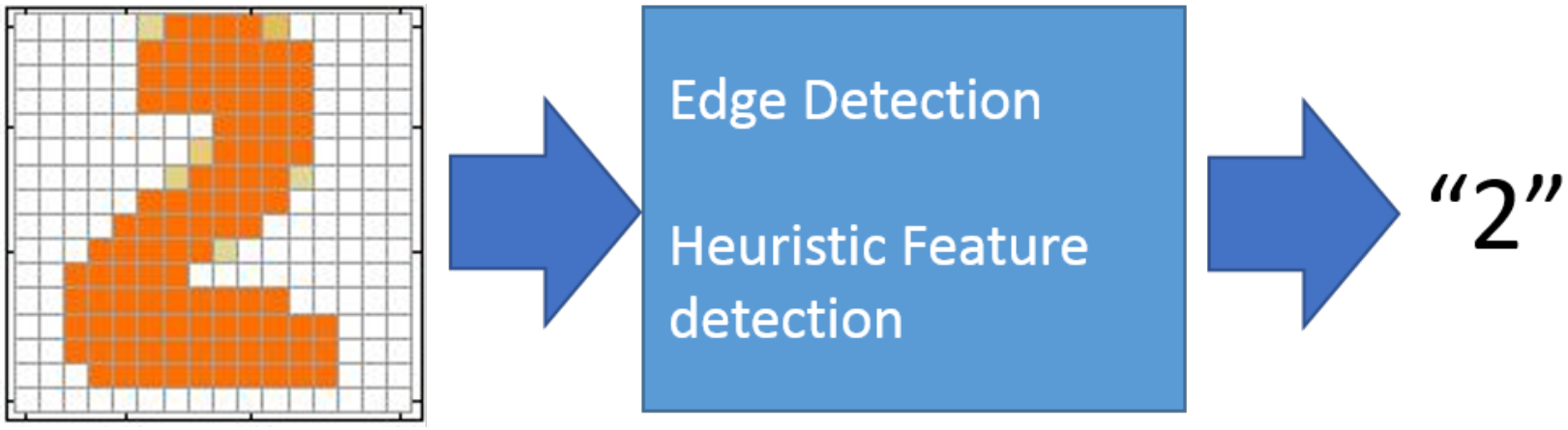


(Deep Learning - The Past, Present and Future of Artificial Intelligence - Lukas Masuch)

Paradigms of Software Development

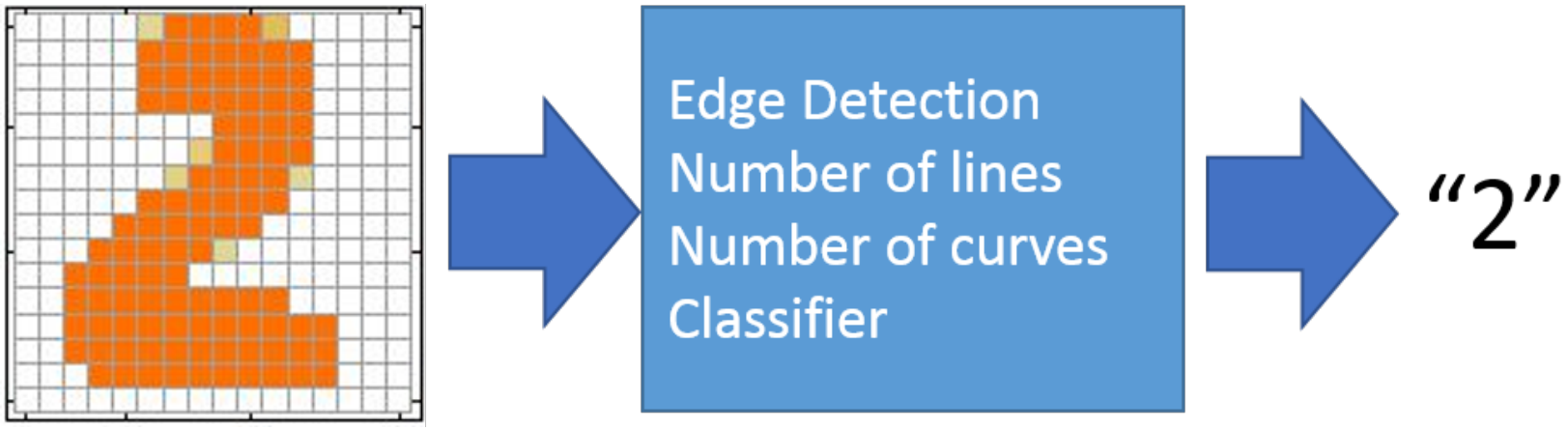
- Rule based
- Machine Learning
- Deep Learning

Rule based : Digit recognition



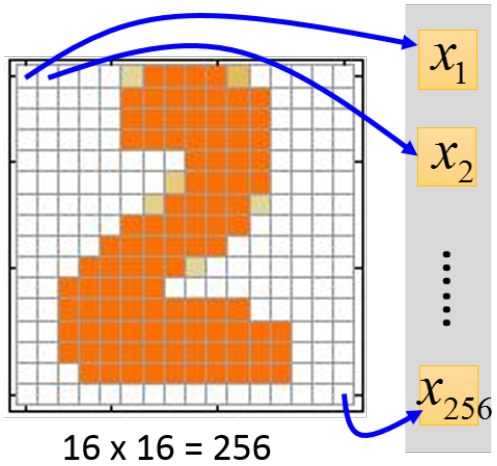
(Image Credit: Deep Learning Tutorial - Hung yi Lee)

Machine Learning : Digit recognition



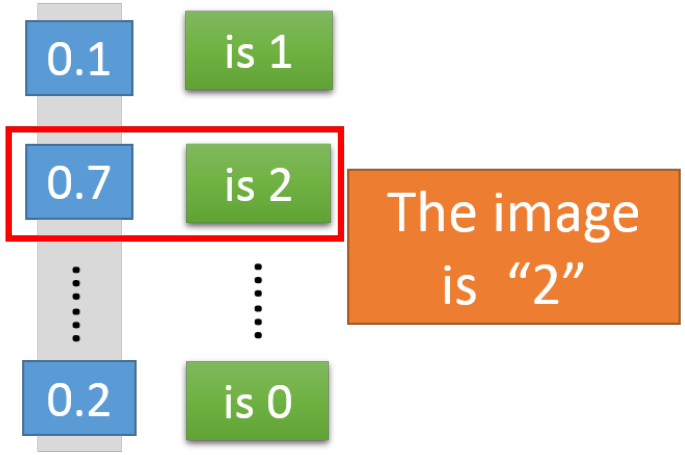
(Image Credit: Deep Learning Tutorial - Hung yi Lee)

Deep Learning : Digit recognition



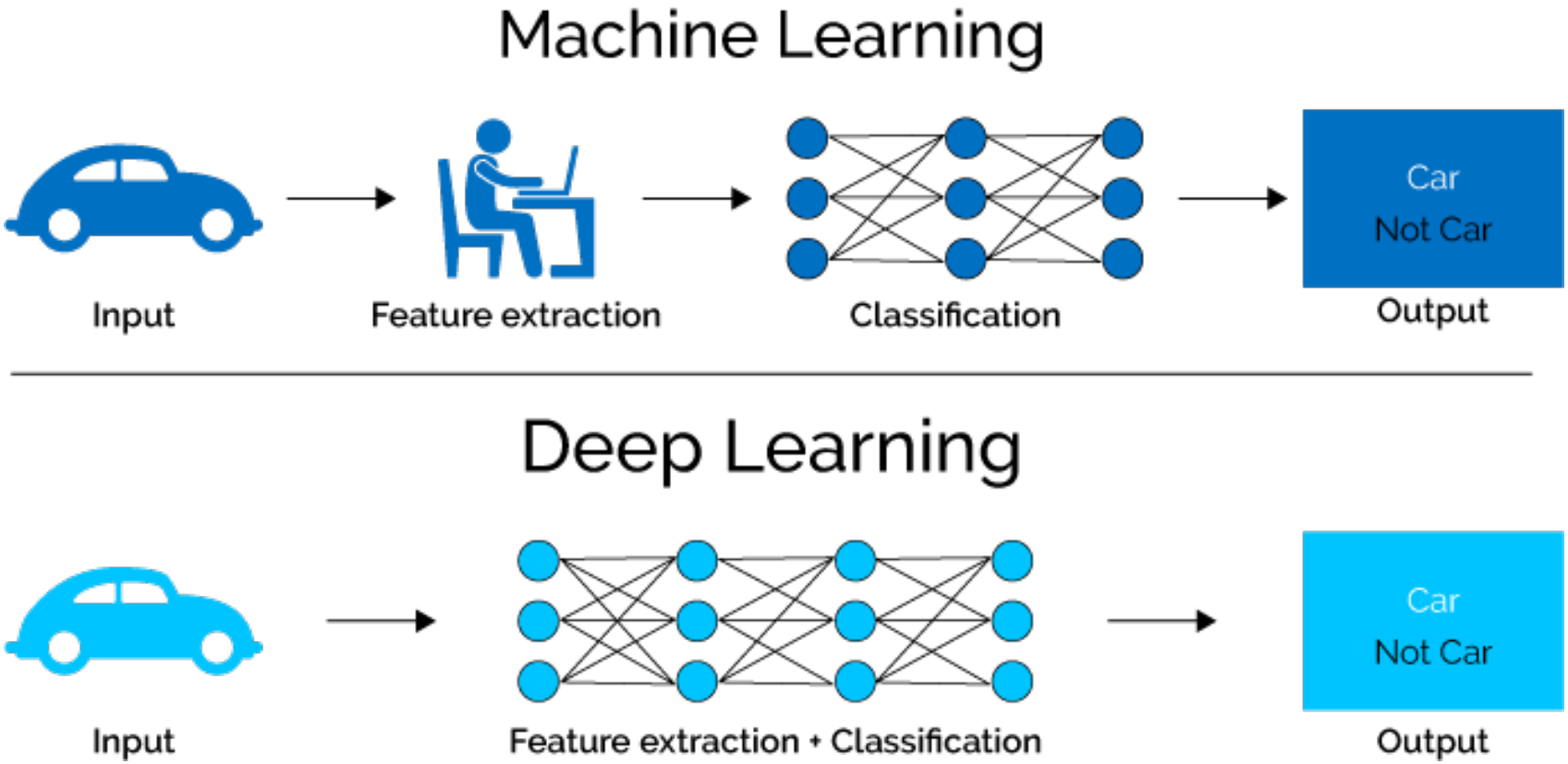
Ink → 1

No ink → 0



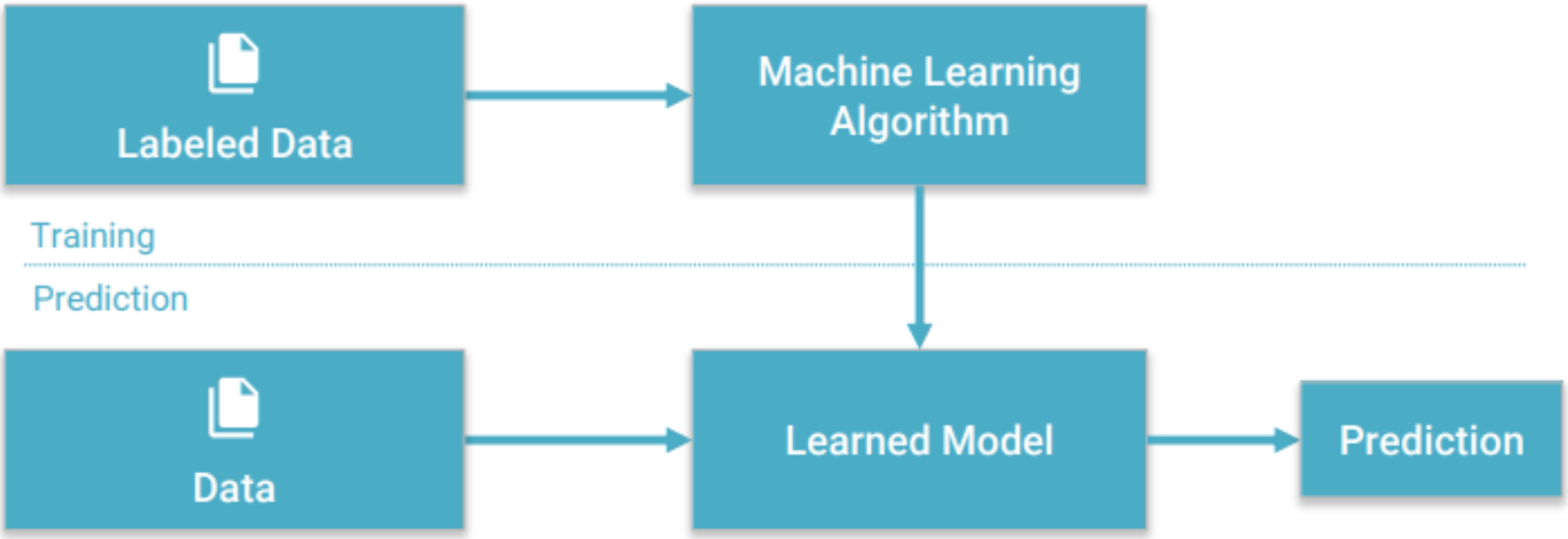
Each dimension represents the confidence of a digit.

Machine Learning Deep Learning



(Reference: https://medium.com/@xenonstack/log-analytics-with-deep-learning-and-machine-learning-20a1891ff70e)

Machine Learning

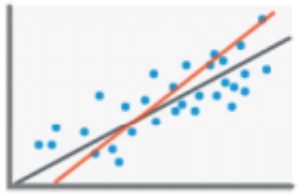


Provides various techniques that can learn from and make predictions on data

(Deep Learning - The Past, Present and Future of Artificial Intelligence - Lukas Masuch)

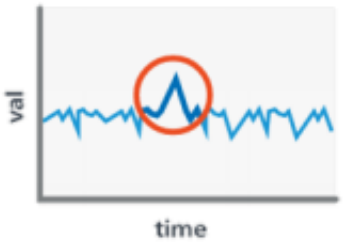
Machine Learning

Classification
(supervised – predictive)



Regression
(supervised – predictive)

Clustering
(unsupervised – descriptive)



Anomaly Detection
(unsupervised– descriptive)

(Deep Learning - The Past, Present and Future of Artificial Intelligence - Lukas Masuch)

Deep Learning

Part of the machine learning field of learning representations of data. Exceptional effective at learning patterns.

Utilizes learning algorithms that derive meaning out of data by using a hierarchy of multiple layers that mimic the neural networks of our brain.

If you provide the system tons of information, it begins to understand it and respond in useful ways.

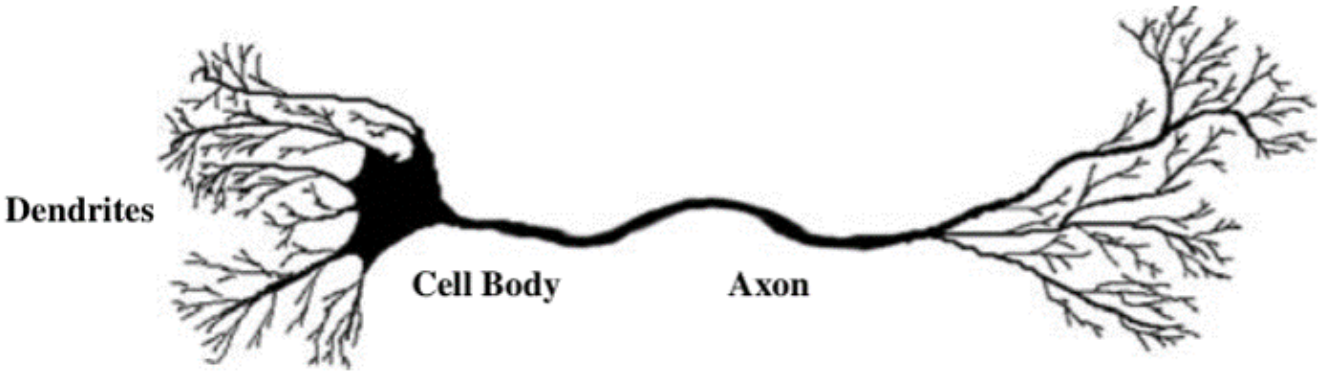
(Deep Learning - The Past, Present and Future of Artificial Intelligence - Lukas Masuch)

Deep Learning == Neural Nets

- Main idea of deep learning: transform the input space into outputs via higher level abstractions.
- Neural Net architectures are made up of perceptrons (similar to neurons)
- Each neuron carries certain transformations on inputs coming to it.
- Collection of such neurons with various types of transformations, can create desired overall transformation.

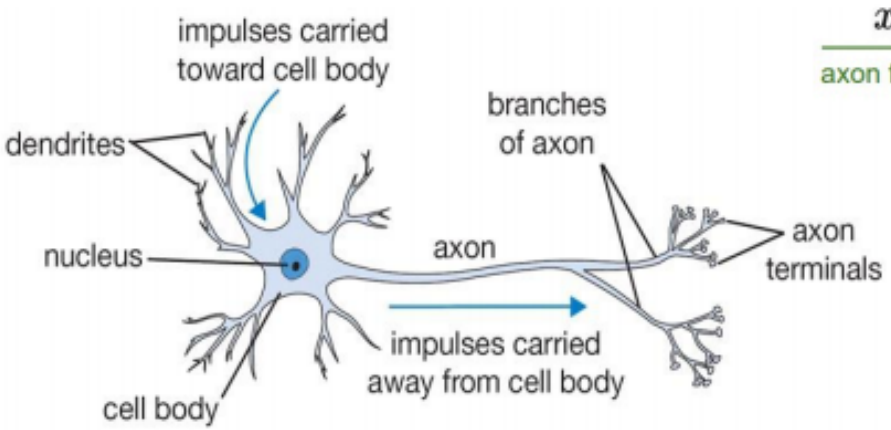
Inspiration

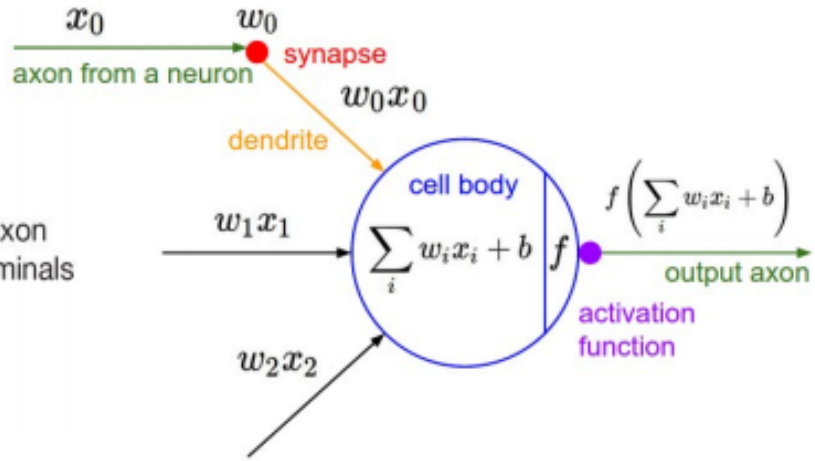
- Attempts to simulate biological neural systems
- Animal brains have complex learning systems consisting of closely interconnected sets of neurons
- Human brain contains approximately 1011 neurons
- Each connected on average to 10,000 other neurons
- Total of 1,000,000,000,000,000 = 1015 connections



Inspiration

An artificial neuron contains a nonlinear activation function and has several incoming and outgoing weighted connections.

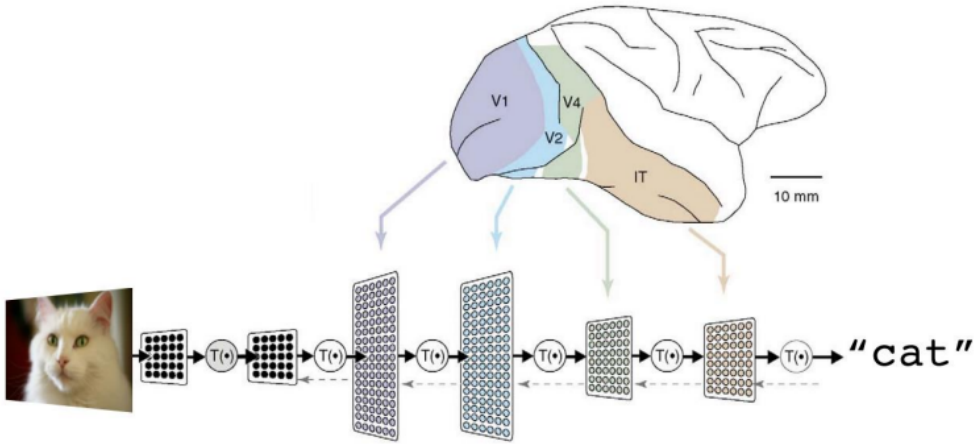




(Deep Learning - The Past, Present and Future of Artificial Intelligence - Lukas Masuch)

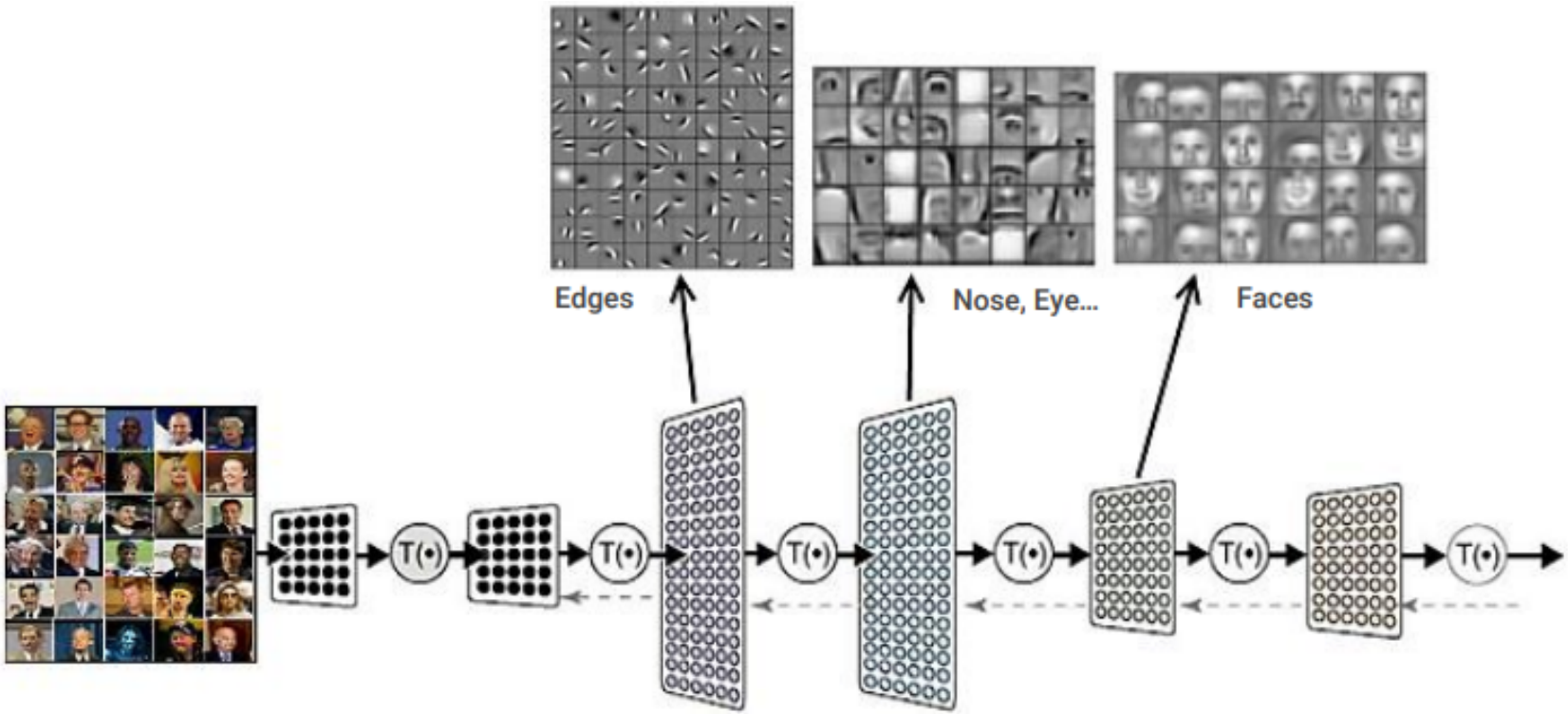
Inspiration

- A deep neural network consists of a hierarchy of layers, whereby each layer transforms the input data into more abstract representations (e.g. edge to nose to face).
- The output layer combines those features to make predictions.



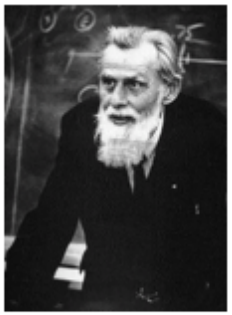
(Deep Learning - The Past, Present and Future of Artificial Intelligence - Lukas Masuch)

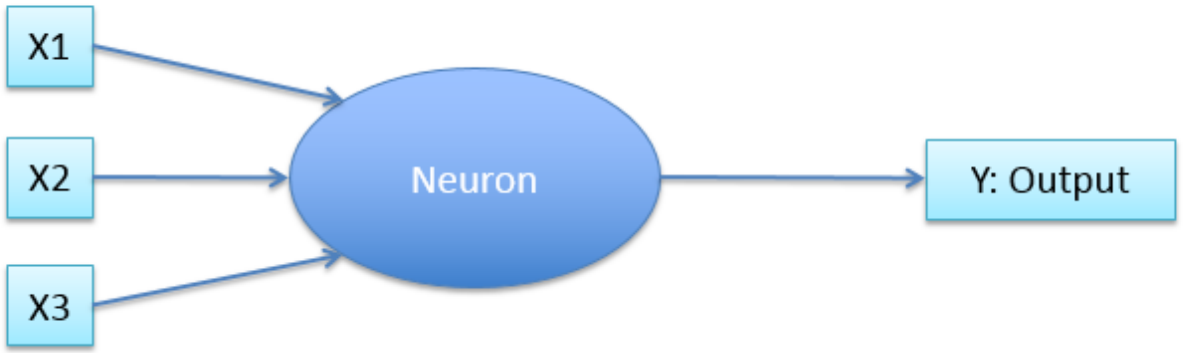
Inspiration



(Deep Learning - The Past, Present and Future of Artificial Intelligence - Lukas Masuch)

Inspiration

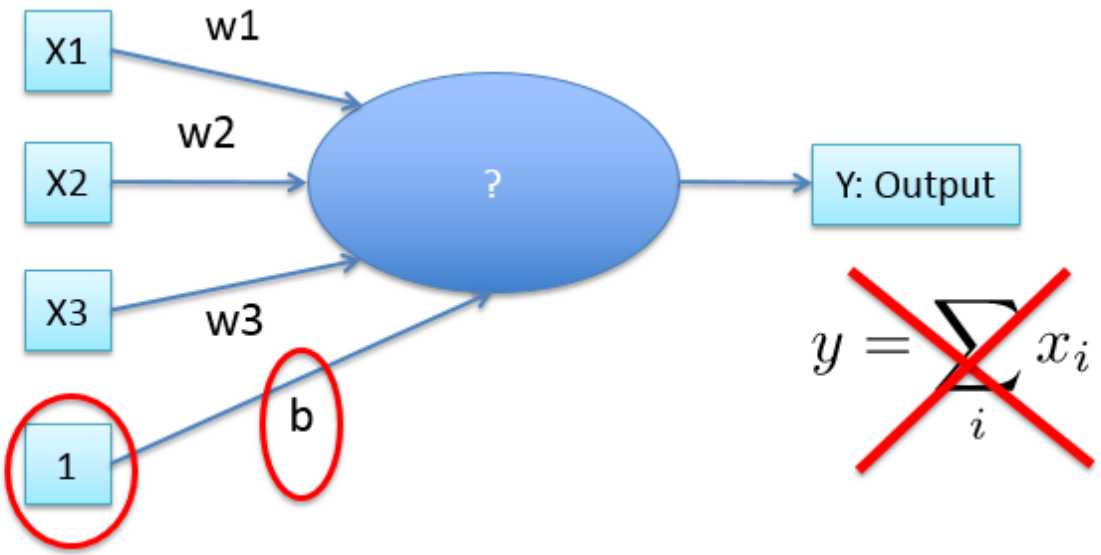


(Reference: Warren McCulloch and Walter Pitts (1943) "A Logical Calculus of the Ideas Immanent in Nervous Activity".)

Sum of the inputs

Idea 1: what if we consider f as a sum of the inputs?



Does not work. (Reference: Neural Networks - Dr. Long Tran-Thanh)

Weighted sum of the inputs

Idea 2: If we allow the possibility of weighting each input differently, we gain some expressivity

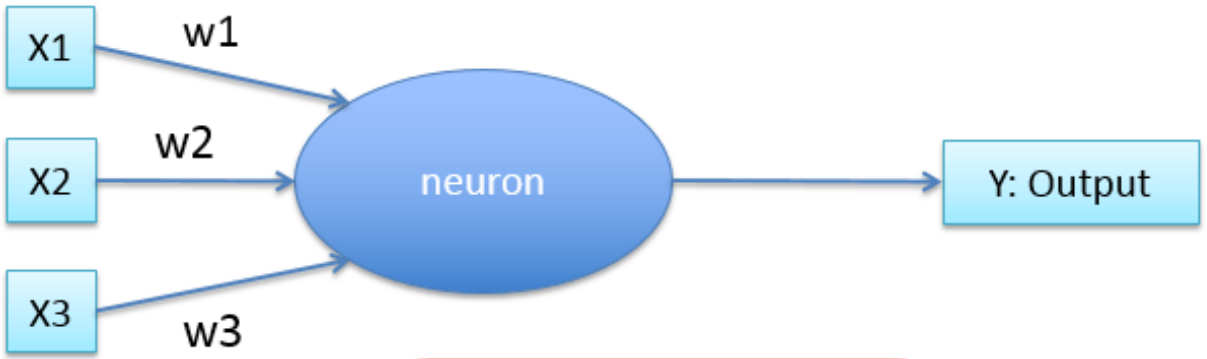
$$y = \sum_i x_i w_i + b$$

$$y = \mathbf{x}^T \mathbf{w} + b$$

Remind you of anything? (Reference: Neural Networks - Dr. Long Tran-Thanh)

Weighted sum of the inputs and function

Idea 3: Transform weighted sum via a function



$$\sum_i x_i w_i + b \longrightarrow f\left(\sum_i x_i w_i\right) = y = \begin{cases} 1 & \text{if } \sum_i x_i w_i > T \\ 0 & \text{otherwise} \end{cases}$$

f: activation function

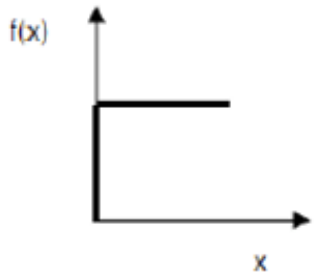
(Reference: Neural Networks - Dr. Long Tran-Thanh)

Transformation functions

Activation functions

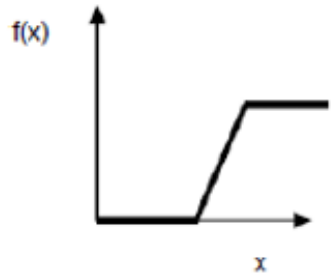
Threshold Function

$$f(x) = \begin{cases} 1 & if \ x \geq 0 \\ 0 & if \ x < 0 \end{cases}$$



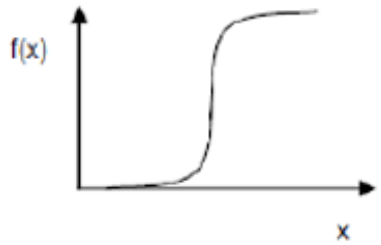
Piecewise-Linear Function

$$f(x) = \begin{cases} 1 & if \ x \geq 0.5 \\ x + 0.5 & if \ -0.5 \leq x \leq 0.5 \\ 0 & if \ x \leq 0.5 \end{cases}$$



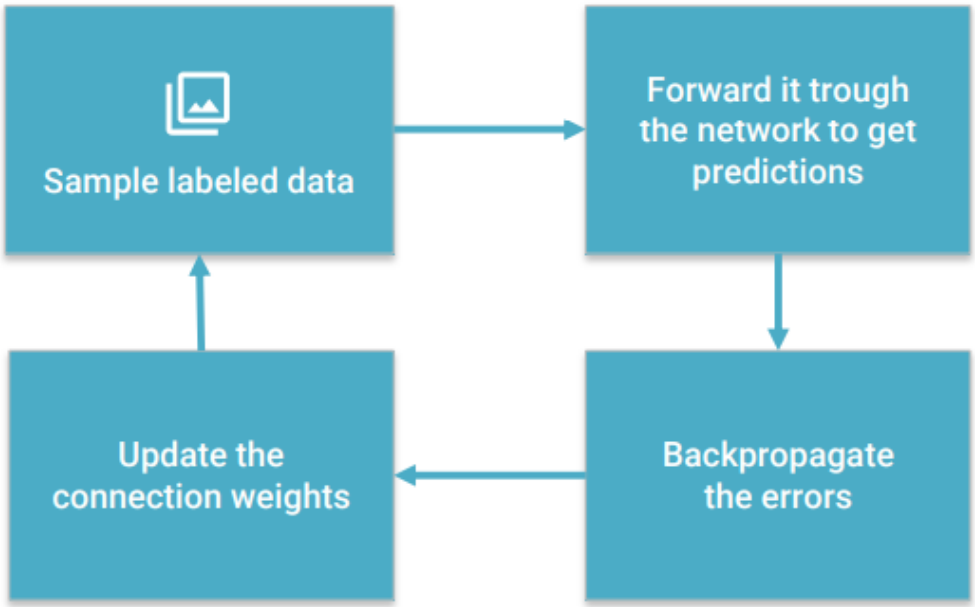
Sigmoid Function

$$f(x) = \frac{1}{1 + e^{-x}}$$



The Training Process

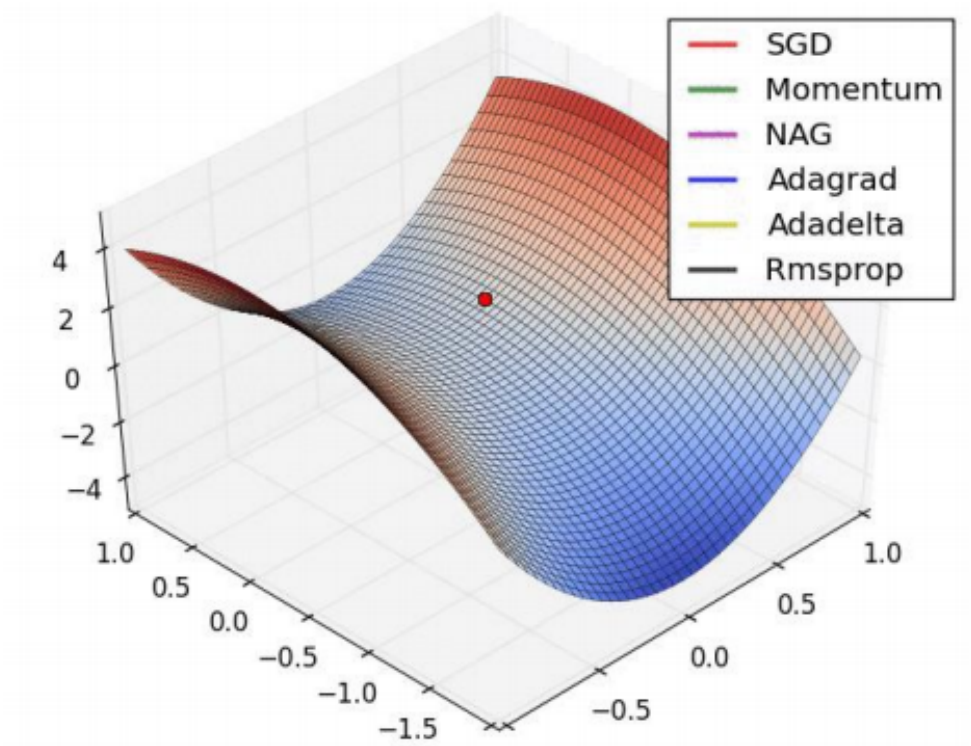
Learns by generating an error signal that measures the difference between the predictions of the network and the desired values and then using this error signal to change the weights (or parameters) so that predictions get more accurate.



(Deep Learning - The Past, Present and Future of Artificial Intelligence - Lukas Masuch)

Gradient Descent

Gradient Descent finds the (local) the minimum of the cost function (used to calculate the output error) and is used to adjust the weights.

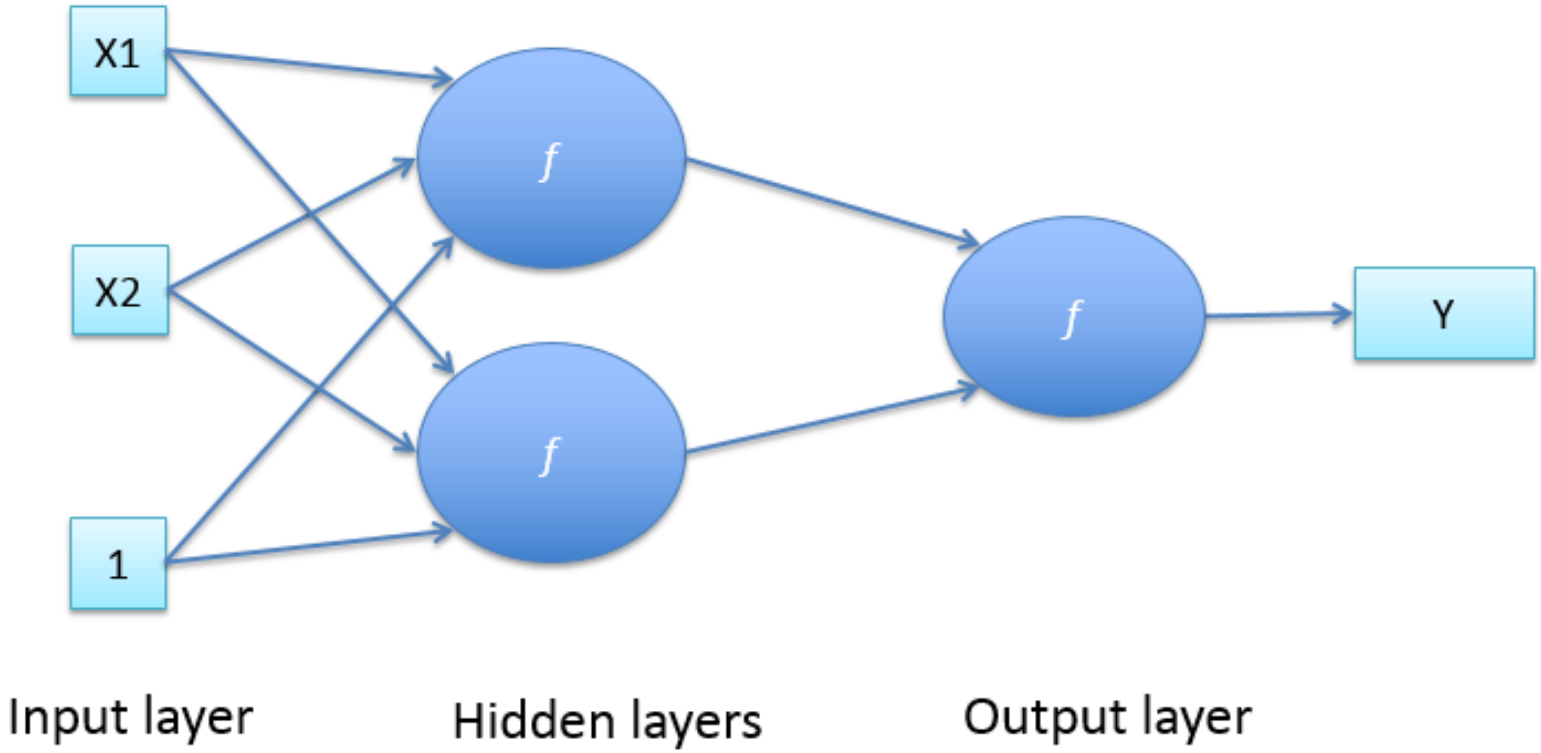


(Deep Learning - The Past, Present and Future of Artificial Intelligence - Lukas Masuch)

Why Deep?

- Many representation multi level graph architecture. Can not be done with shallow ones like SVM, NB, etc.
- Human Brain learns languages, speech, images, by sequence of areas
- Humans organize their ideas and concepts hierarchically.
- Humans first learn simpler concepts and then compose them to represent more abstract ones.
- Engineers break-up solutions into multiple levels of abstraction and processing

Example



Our black box is quite complicated now; can approximate arbitrary functions given enough hidden neurons.
(Reference: Neural Networks - Dr. Long Tran-Thanh)

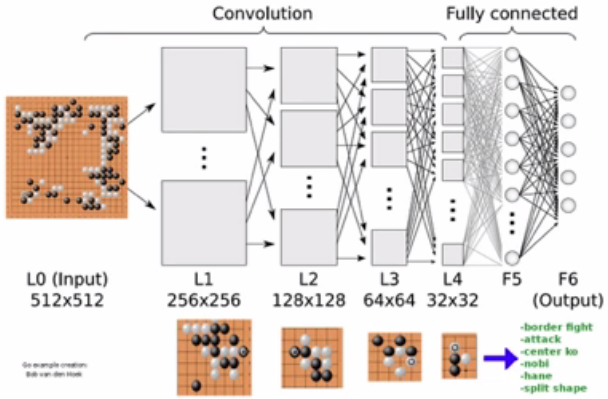
Usage Requirements

- Large data set with good quality (input-output mappings)
- Measurable and describable goals (define the cost)
- Enough computing power (AWS GPU Instance)
- Excels in tasks where the basic unit (pixel, word) has very little meaning in itself, but the combination of such units has a useful meaning.

(Deep Learning - The Past, Present and Future of Artificial Intelligence - Lukas Masuch)

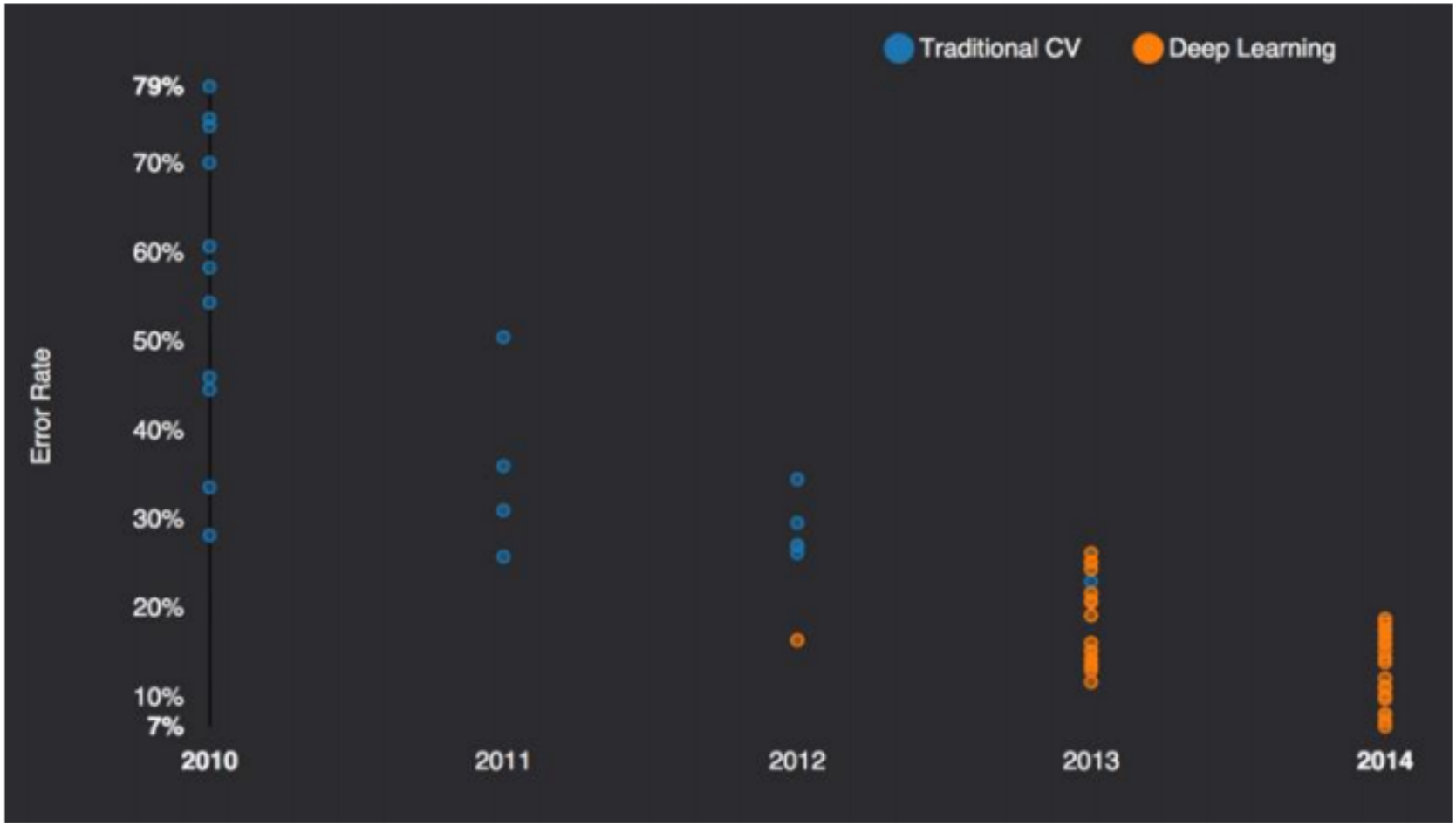
Where do you find Deep Learning?

- Speech Recognition: Siri, Alexa
- Natural Language Processing: Google Translate
- Image Recognition: Cancer detection



ImageNet: The "computer vision World Cup"

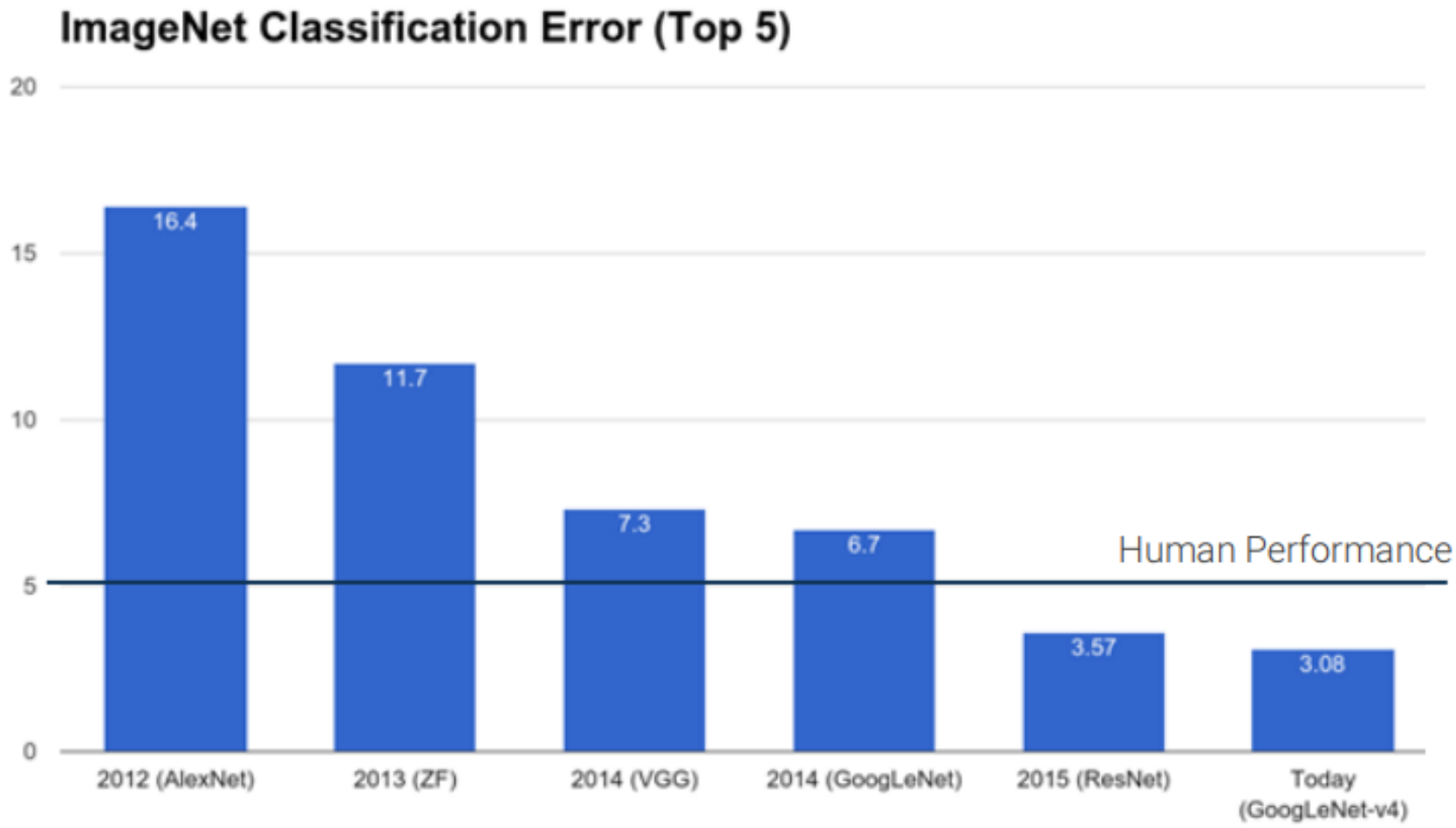
Cats and Dogs.



(Deep Learning - The Past, Present and Future of Artificial Intelligence - Lukas Masuch)

ImageNet: The "computer vision World Cup"

Cats and Dogs.



(Deep Learning - The Past, Present and Future of Artificial Intelligence - Lukas Masuch)

Heroes of Deep Learning




(Reference: Neural Networks - Dr. Long Tran-Thanh)

Big Players

Companies

(Deep Learning - The Past, Present and Future of Artificial Intelligence - Lukas Masuch)

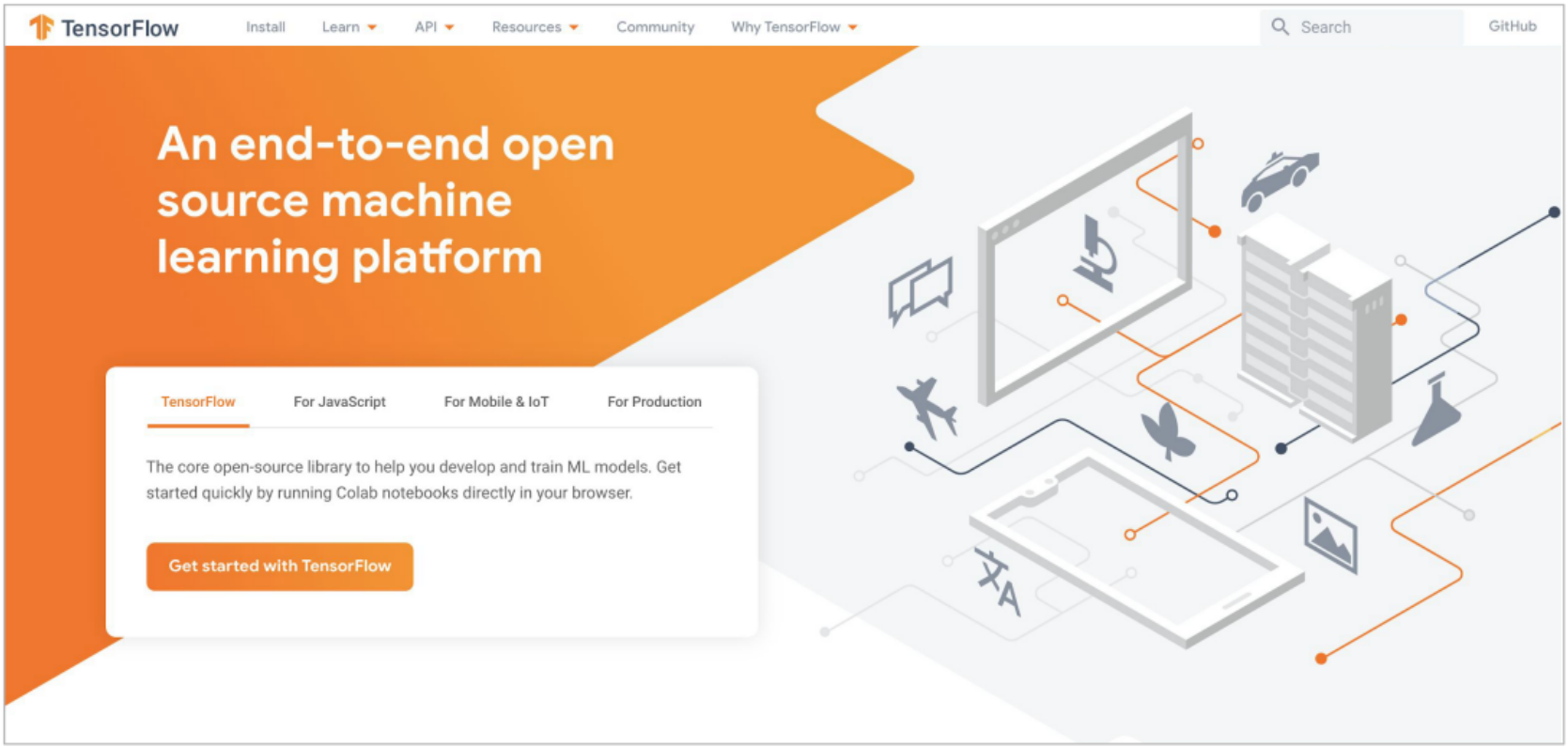
Big Players

Startups

(Deep Learning - The Past, Present and Future of Artificial Intelligence - Lukas Masuch)

Introduction to TensorFlow 2.0

Website

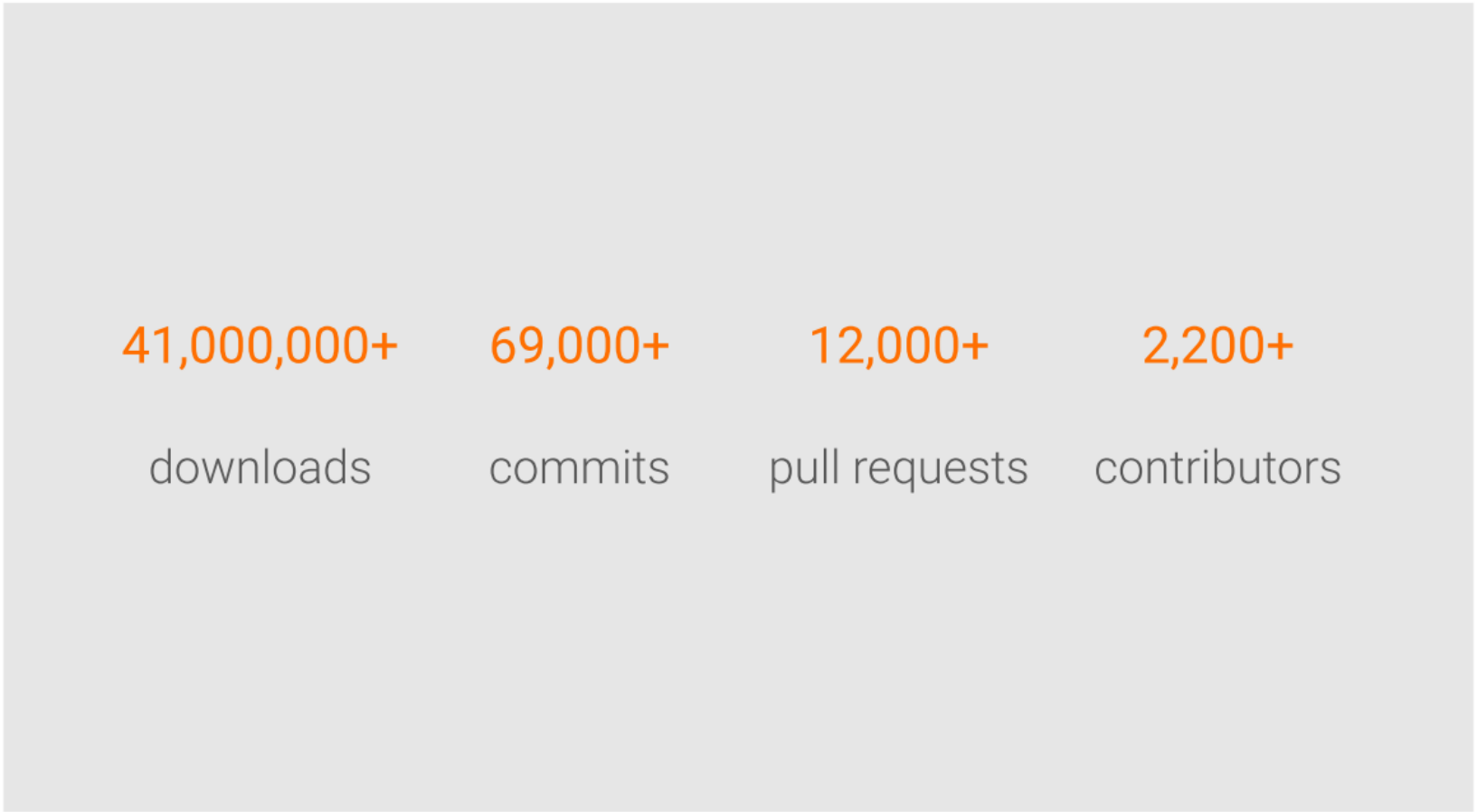


(Ref: Introduction to TensorFlow 2.0 - Brad Miro)

TensorFlow is

- Open source, Free library, with Python bindings, by Google Brain team
- Other libraries are: Caffe (Berkeley), Torch (Facebook), Cntk (Microsoft),
- Can deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API
- Flexibility: from Raspberry Pi, Android, Windows, iOS, Linux to server farms
- Till 2019 Keras was popular as a separate library (with back-end as Tensorflow) but with Tensorflow 2.0, Keras as become its default front end API.
- TensorFlow 2.0 merges keras as "tf.keras". It allows you to design, fit, evaluate deep learning models.

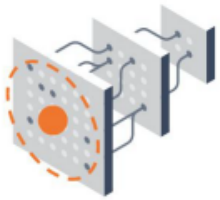
Open Source Community



As of Oct 2019 . . .

(Ref: Introduction to TensorFlow 2.0 - Brad Miro)

Tensorflow 2.0



Easy

Simplified APIs.
Focused on Keras and
eager execution

Powerful

Flexibility and performance.
Power to do cutting edge research
and scale to > 1 exaflops

Scalable

Tested at Google-scale.
Deploy everywhere

(Ref: Introduction to TensorFlow 2.0 - Brad Miro)

Deploy Anywhere

Servers

TensorFlow
Extended

Edge devices

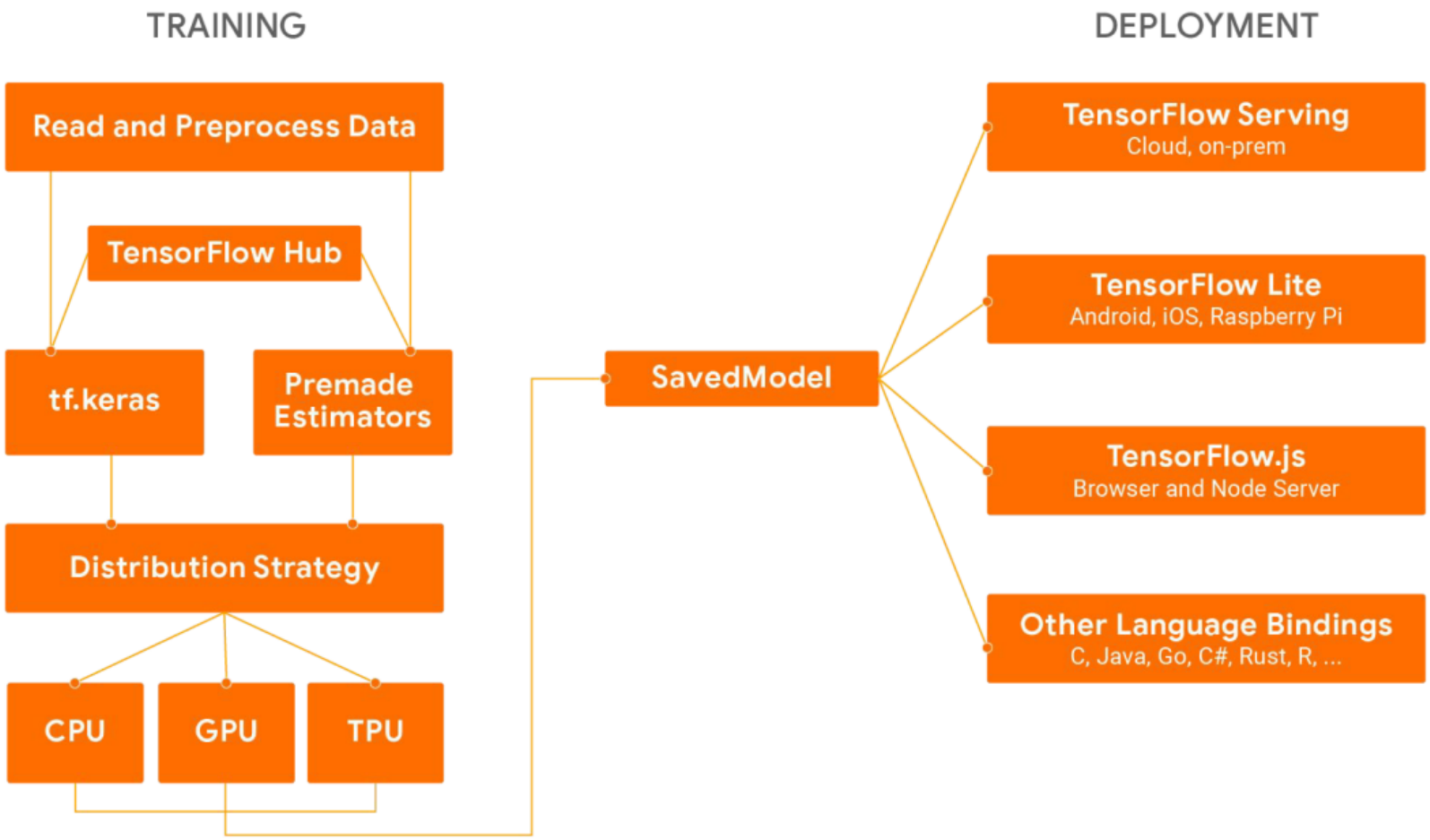
TensorFlow
Lite

JavaScript

TensorFlow
.JS

(Ref: Introduction to TensorFlow 2.0 - Brad Miro)

Training and Deployment



(Ref: Introduction to TensorFlow 2.0 - Brad Miro)

Ecosystem/Verticals

TF Probability

TF Agents

Tensor2Tensor

TF Ranking

TF Text

TF Federated

TF Privacy

...

(Ref: Introduction to TensorFlow 2.0 - Brad Miro)

Compared to TF 1.0

What's Gone

- `Session.run`
- `tf.control_dependencies`
- `tf.global_variables_initializer`
- `tf.cond, tf.while_loop`
- `tf.contrib`

(Ref: Introduction to TensorFlow 2.0 - Brad Miro)

What's New

- Eager execution by default
- `tf.function`
- Keras as main high-level api

Installation

- Have Python installed, such as Python 3.6 or higher.
- Easy way to install TensorFlow
- Linux:

```
sudo pip install tensorflow
```

- Windows:

```
pip install tensorflow
```

Installation Check

- Confirm the installation by:

```
# check version
import tensorflow
print(tensorflow.__version__)
```

- It must be 2.0 onwards
- If you get warning like below, Don't worry, just IGNORE.

```
Your CPU supports instructions that this TensorFlow binary was not
     compiled to use: AVX2 FMA
XLA service 0x7fde3f2e6180 executing computations on platform Host.
     Devices:
StreamExecutor device (0): Host, Default Version
```

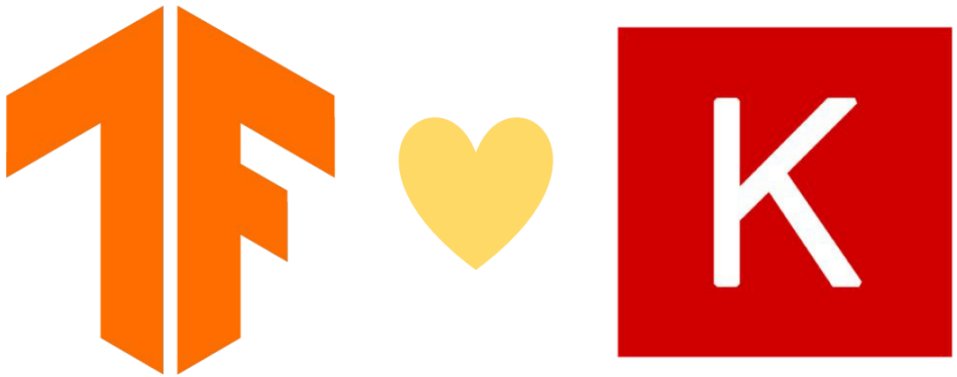
Hello World!!

```
import tensorflow as tf # Assuming TF 2.0 is installed
a = tf.constant([[1, 2],[3, 4]])
b = tf.matmul(a, a)
print(b)
# tf.Tensor( [[ 7 10] [15 22]], shape=(2, 2), dtype=int32)
print(type(b.numpy()))
# <class 'numpy.ndarray'>
```

(Ref: Introduction to TensorFlow 2.0 - Brad Miro)

Keras and tf.keras

- Fast prototyping, advanced research, and production
- keras.io = reference implementation `import keras`
- tf .keras = TensorFlow's implementation (a superset, built-in to TF, no need to install Keras separately) `from tensorflow import keras`



(Ref: Introduction to TensorFlow 2.0 - Brad Miro)

Steps to use tf.Keras

- Define the model.
- Compile the model.
- Fit the model.
- Evaluate the model.
- Make predictions.

Define the Model

- First, select the type of the model.
- Choose architecture or network topology.
- Meaning, define layers, its parameters.
- There are multiple API ways to define the model (will look at later)

```
...
# define the model
model = ...
```

Compile the Model

- Select loss function that you want to optimize, eg Cross Entropy or Mean Squared Error
- Select Optimization method, eg Adam, Stochastic Gradient Descent
- Select performance metrics to be used during Training

```
...
# compile the model
opt = SGD(learning_rate=0.01, momentum=0.9)
model.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])
```

Fit the Model

- Select Training configuration (epochs, batch size, etc)
- *Epochs: number of full cycles (forward + backward) during training
- *Batch Size: Number of samples used to estimate model error, or update the weights
- Can take minutes to hours to days depending on complexity, hardware, training samples size.
- Progress bar shows status of each epoch, performance, etc.

```
...
# fit the model
model.fit(X, y, epochs=100, batch_size=32)
```

Evaluate the Model

- Select a holdout dataset (cross validation)
- This is not used for training but just for evaluation as it has correct answers as well.

```
...
# evaluate the model
loss = model.evaluate(X, y, verbose=0)
```

Making Predictions

- Get Test set for which answers have to be found out.
- Better to save the model and later load it to make predictions.
- May choose to fit a model on all of the available data before you start using it.

```
...
# make a prediction
yhat = model.predict(X)
```

Model Definition

API styles

- The Sequential Model
  - Dead simple
  - Only for single-input, single-output, sequential layer stacks
  - Good for 70+% of use cases
- The functional API
  - Like playing with Lego bricks
  - Multi-input, multi-output, arbitrary static graph topologies
  - Good for 95% of use cases
- Model subclassing
  - Maximum flexibility
  - Larger potential error surface

Sequential Model API (Simple)

- Called "Sequential" because it involves using Sequential class and adding layers to it one-by-one, in a sequence.
- E.g. 8 inputs, one hidden layer with 10 nodes, and one output layer with one node to predict numerical value would look:

```
# example of a model defined with the sequential api
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
# define the model
model = Sequential()
model.add(Dense(10, input_shape=(8,)))
model.add(Dense(1))
```

Note:

- Input layer, per say, is NOT added. Its an argument for the first HIDDEN layer.
- Here 'input_shape' of (8,) means one sample/row is of 8 values. And such, many samples/rows can come, so left blank.

Functional Model API (Advanced)

- Need to explicitly connections between layers.
- Models may have multiple input/output paths (a word and a number)
- Input layer needs to be defined explicitly, like:

```
x_in = Input(shape=(8,))
```

- Next, a fully connected layer can be connected to the input by calling the layer and passing the input layer. This will return a reference to the output connection in this new layer.

```
x = Dense(10)(x_in)
```

- Once connected, we define a Model object and specify the input and output layers.

```
x_in = Input(shape=(8,))
x = Dense(10)(x_in)
x_out = Dense(1)(x)
# define the model
model = Model(inputs=x_in, outputs=x_out)
```

Sub-classing Model API (Very Advanced)

```
class MyModel(tf.keras.Model):
  def __init__(self, num_classes=10):
    super(MyModel, self).__init__(name='my_model')
    self.dense_1 = layers.Dense(32, activation='relu')
    self.dense_2 = layers.Dense(num_classes, activation='sigmoid')

  def call(self, inputs):
    # Define your forward pass here,
    x = self.dense_1(inputs)
    return self.dense_2(x)
```

Understanding deferred (symbolic) vs. eager (imperative)

- Deferred: Build a computation graph that gets compiled first and then once values are filled, executed later
- Eager: Model is a python exe,Execution is runtime (like Numpy)
- Deferred: Symbolic tensors don't have a value in your Python code (yet)
- Eager: tensors have a value in your Python code
- Eager: can use value-dependent dynamic topologies (tree-RNNs)

Distribution Strategy

For the sample code below . . .

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(64, input_shape=[10]),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')])
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Multi-GPU

One of the computations distribution strategy could be . . .

```
strategy = tf.distribute.MirroredStrategy()
with strategy.scope():
  model = tf.keras.models.Sequential([
      tf.keras.layers.Dense(64, input_shape=[10]),
      tf.keras.layers.Dense(64, activation='relu'),
      tf.keras.layers.Dense(10, activation='softmax')])
  model.compile(optimizer='adam',
                loss='categorical_crossentropy',
                metrics=['accuracy'])
```

TensorFlow Datasets

- audio
  - "nsynth"
- image
  - "cifar10"
  - "diabetic_retinopathy_detection"
  - "imagenet2012"
  - "mnist"
- structured
  - "titanic"
- text
  - "imdb_reviews"
  - "lm1b"
  - "squad"
- translate
  - "wmt_translate_ende"
  - "wmt_translate_enfr"
- video
  - "bair_robot_pushing_small"
  - "moving_mnist"
  - "starcraft_video"

More at tensorflow.org/datasets

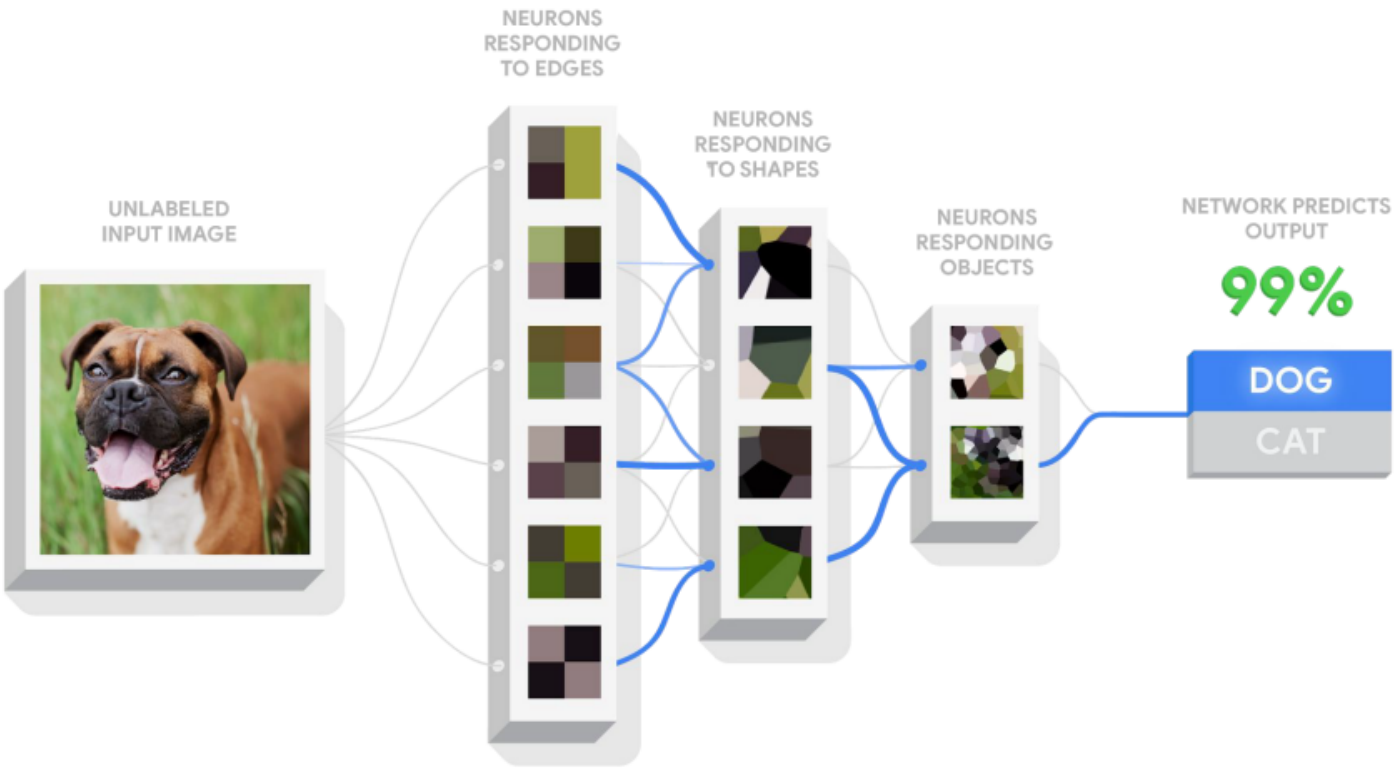
(Ref: Introduction to TensorFlow 2.0 - Brad Miro)

Terminologies

In the neural network terminology:

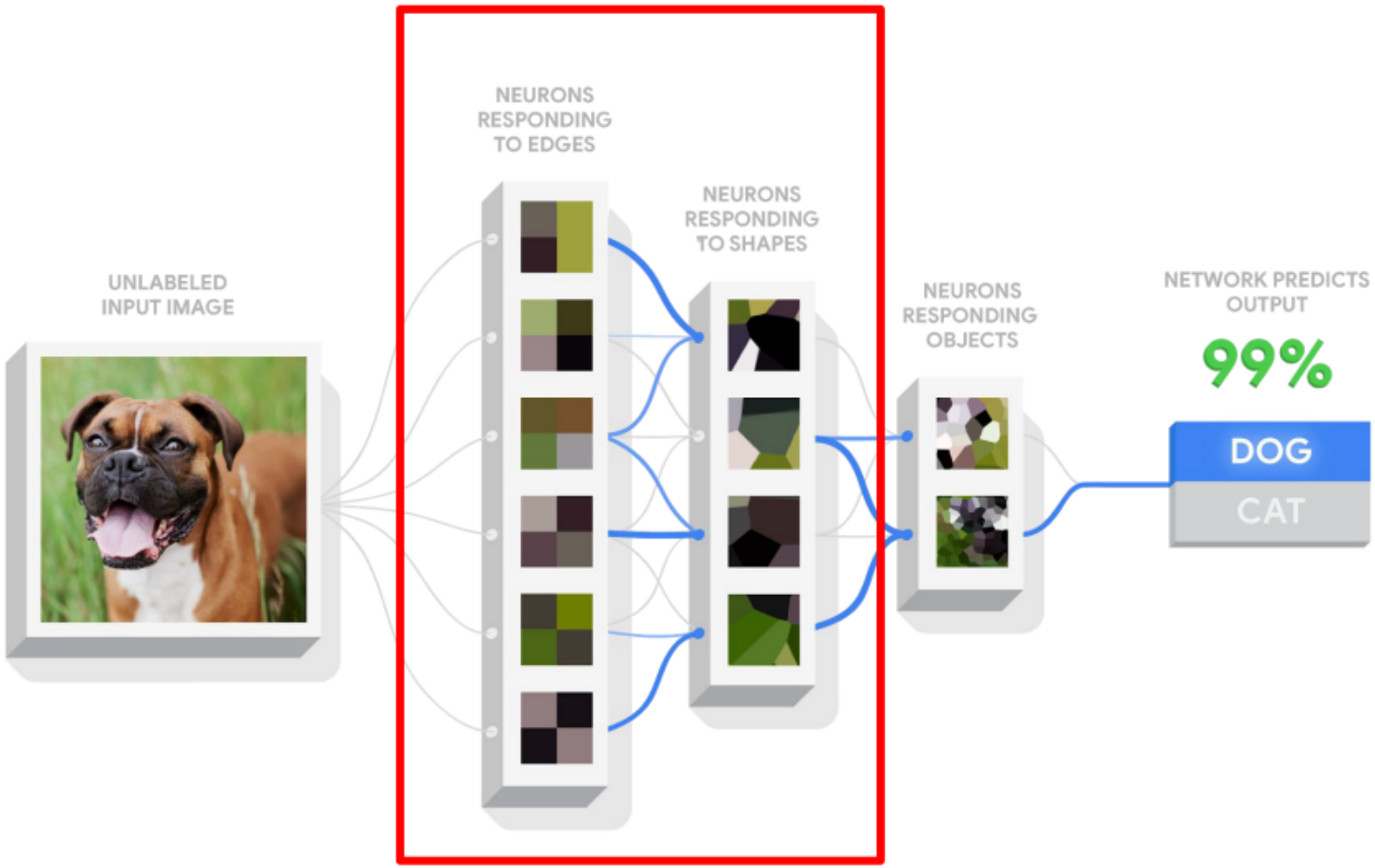
- one epoch = one forward pass and one backward pass of all the training examples
- batch size = the number of training examples in one forward/backward pass. The higher the batch size, the more memory space you'll need.
- number of iterations = number of passes, each pass using [batch size] number of examples. To be clear, one pass = one forward pass + one backward pass (we do not count the forward pass and backward pass as two different passes).
- Example: if you have 1000 training examples, and your batch size is 500, then it will take 2 iterations to complete 1 epoch.

Transfer Learning



(Ref: Introduction to TensorFlow 2.0 - Brad Miro)

Transfer Learning



(Ref: Introduction to TensorFlow 2.0 - Brad Miro)

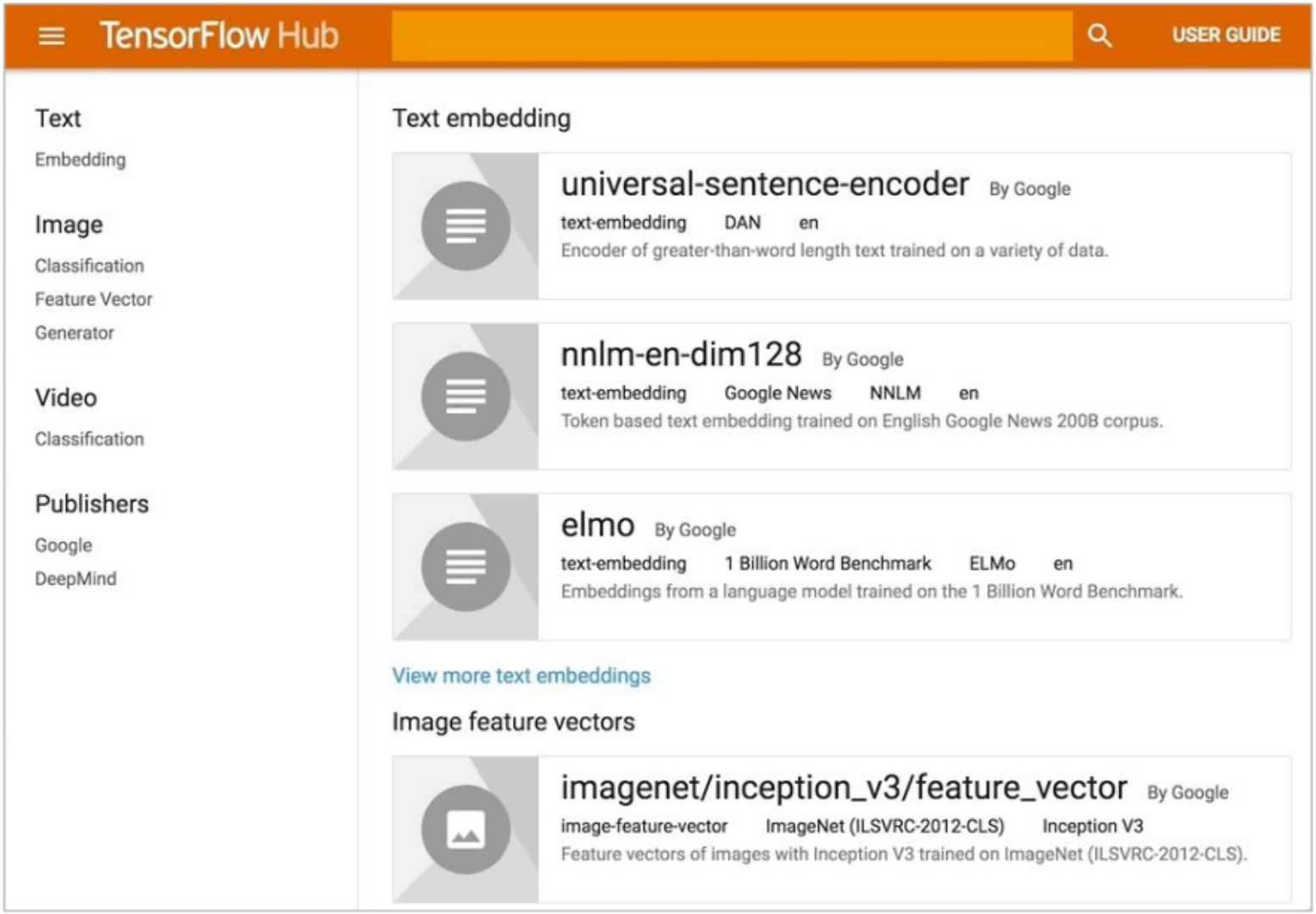
Transfer Learning

```
import tensorflow as tf
base_model = tf.keras.applications.SequentialMobileNetV2(
                 input_shape=(160, 160, 3),
                 include_top=False,
                 weights="imagenet")

base_model.trainable = False
model = tf.keras.models.Sequential([
  base_model,
  tf.keras.layers.GlobalAveragePooling2D(),
  tf.keras.layers.Dense(1)
])
# Compile and fit
```

(Ref: Introduction to TensorFlow 2.0 - Brad Miro)

Transfer Learning with TF Hub



Learning More ...

- Latest tutorials and guides at http://tensorflow.org/beta
- Book: Hands-on ML with Scikit-Learn, Keras and TensorFlow (2nd edition)

(Ref: Intro to TensorFlow 2.0 - Josh Gordon)

References

Many publicly available resources have been refereed for making this presentation. Some of the notable ones are:

- Deep Learning Tutorial Hung yi Lee
- Deep Learning Project in NLP - Yu Wang, Yale University
- "A friendly Introduction to Deep Learning and Neural Networks": Luis Serrano
- Michael Nielsen's Neural Networks and Deep Learning: `http://neuralnetworksanddeeplearning.com/`

Thanks ... yogeshkulkarni@yahoo.com