Welcome to Deep Learning programming!

Keras for Deep Learning Research

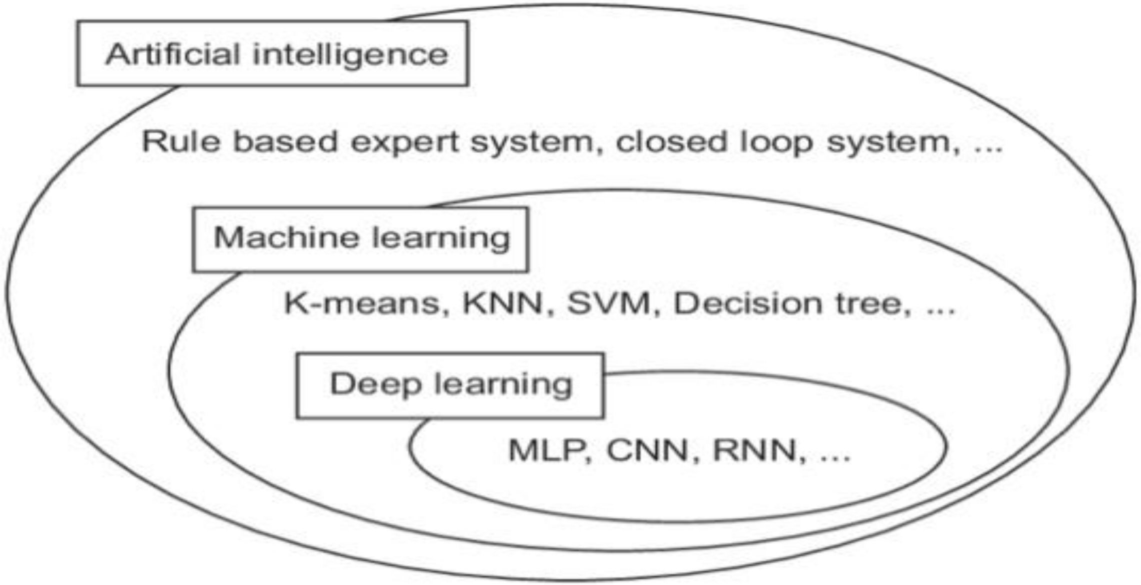
UMKC

Feedback is greatly appreciated!

Objective

- What is Deep learning
- Application of Deep learning
- A brief introduction to how Deep Learning works
- Introduction to Keras library
- Basic operations in Keras
- Use case: diabetes prediction

Venn diagram

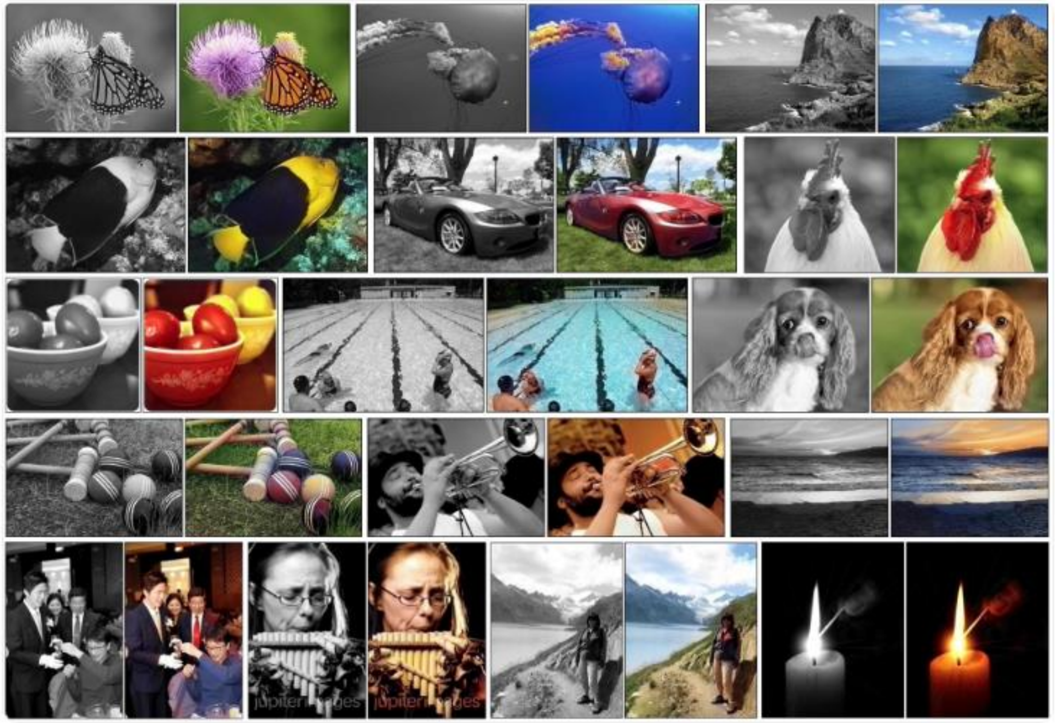

Artificial intelligence
Rule based expert system, closed loop system, ...
Machine learning
K-means, KNN, SVM, Decision tree, ...
Deep learning
MLP, CNN, RNN, ...

What is Deep Learning?

- Deep Learning is the "set of algorithms in machine learning that attempt to learn in multiple levels, corresponding to different levels of abstraction" (much-debated definition)

What can Deep Learning do?

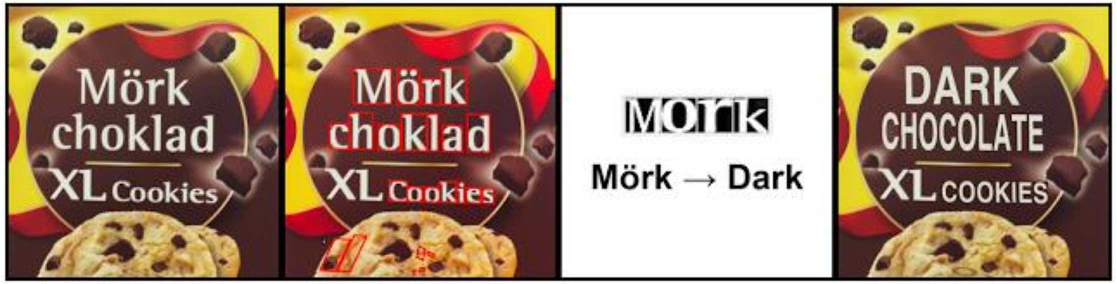
What can Deep Learning do?



Colorization of Black and White Photographs

https://machinelearningmastery.com/inspirational-applications-deep-learning/

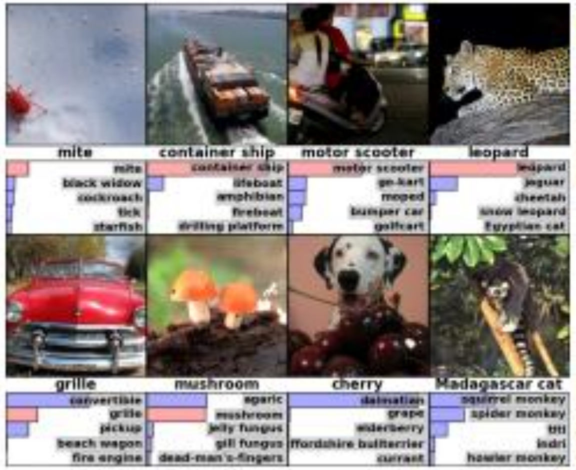
What can Deep Learning do?



Instant Visual Translation

https://machinelearningmastery.com/inspirational-applications-deep-learning/

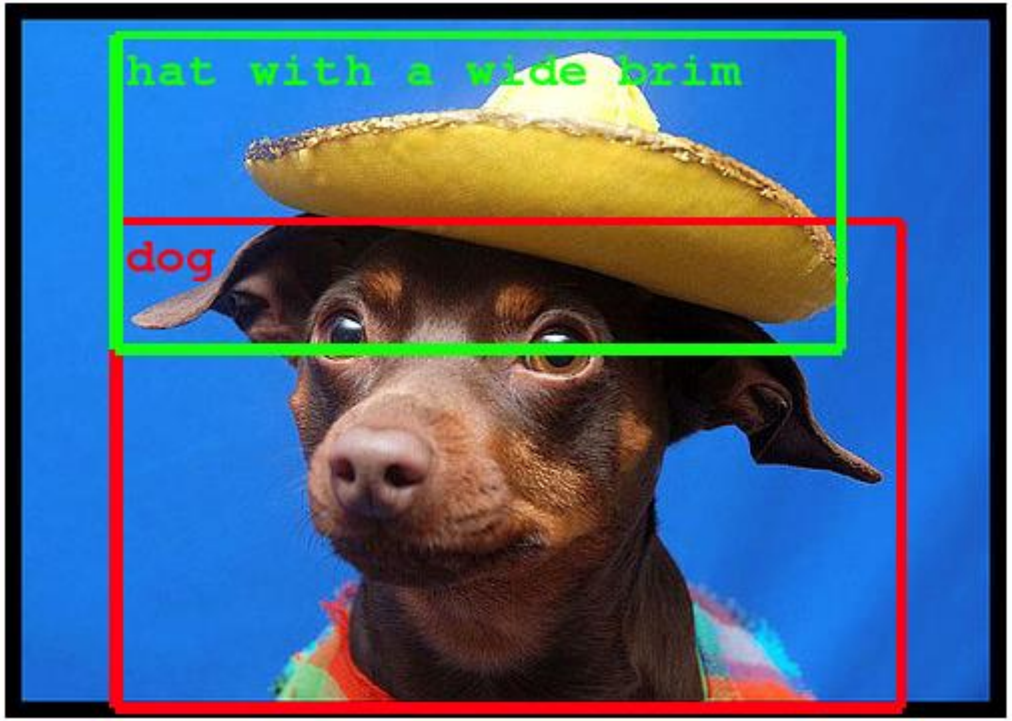
What can Deep Learning do?



Example of Object Classification

https://machinelearningmastery.com/inspirational-applications-deep-learning/

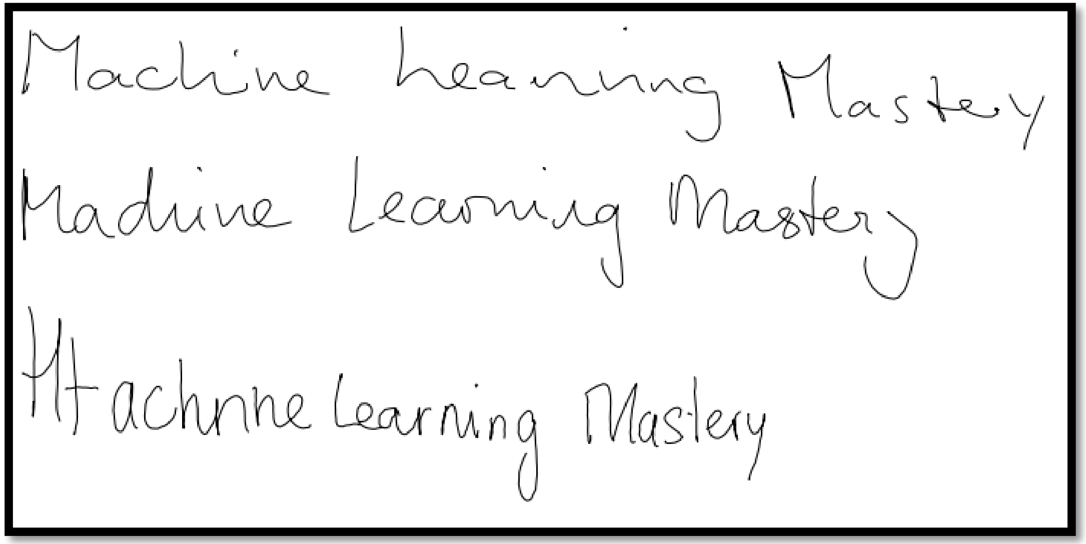
What can Deep Learning do?



Example of Object Detection within Photographs

https://machinelearningmastery.com/inspirational-applications-deep-learning/

What can Deep Learning do?



Sample of Automatic Handwriting Generation

https://machinelearningmastery.com/inspirational-applications-deep-learning/

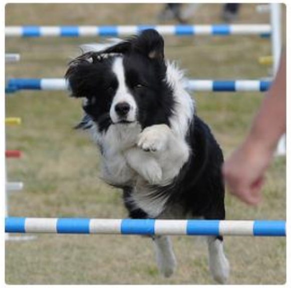
What can Deep Learning do?

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"girl in pink dress is jumping in air."


"black and white dog jumps over bar."

"young girl in pink shirt is swinging on swing."

Automatic Image Caption Generation

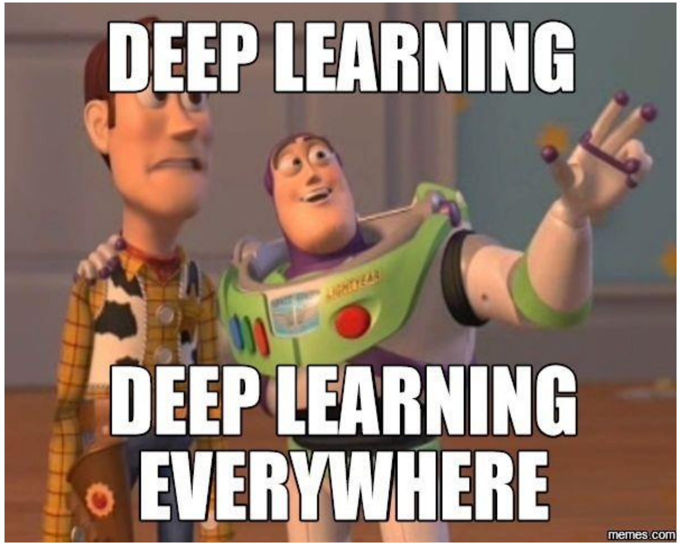
https://machinelearningmastery.com/inspirational-applications-deep-learning/

What can Deep Learning do?

What can Deep Learning do?

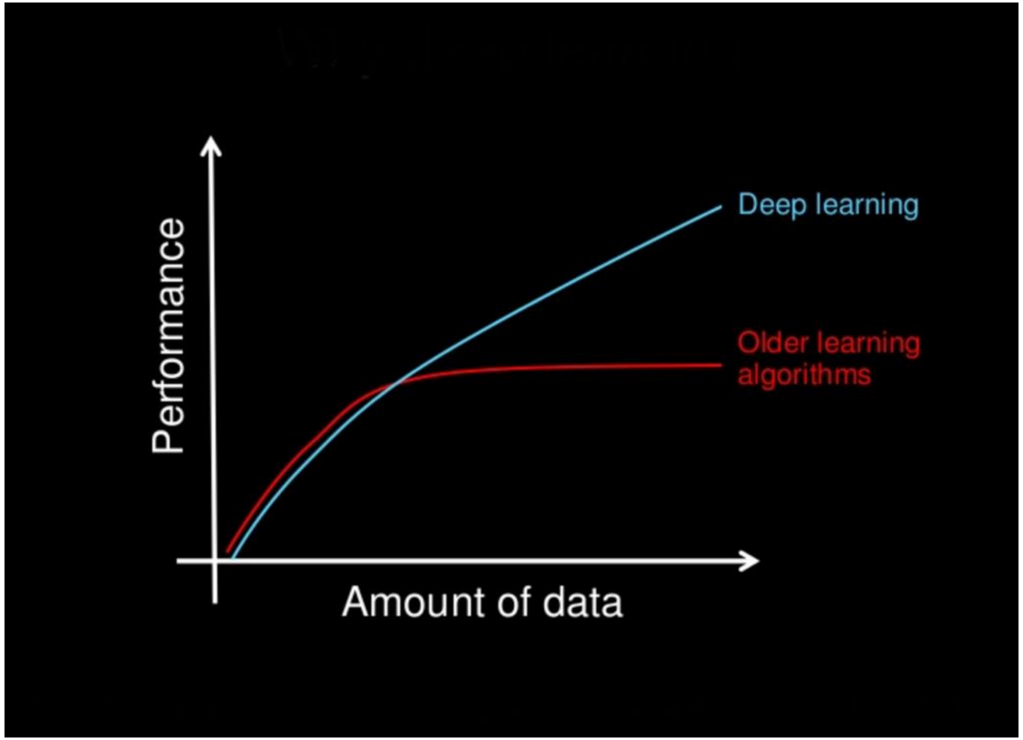
What can Deep Learning do?

- Natural Language Processing
- Speech Processing
- Computer Vision
- And more and more



Comparison of Machine learning & Deep learning

1. Data Dependency



https://machinelearningmastery.com/what-is-deep-learning/

2. Feature Engineering

- Feature engineering is a process of putting domain knowledge into the creation of feature extractors to reduce the complexity of the data and make patterns more visible to learning algorithms to work
- In Machine learning, most of the applied features need to be identified by an expert and then hand-coded as per the domain and data type
- Deep learning algorithms try to learn high-level features from data. This is a very distinctive part of Deep Learning and a major step ahead of traditional Machine Learning

hierarchical feature extraction of Deep Learning

How can you realize and differentiate circle and square?

Hierarchical feature extraction of Deep Learning

The first thing our eyes do is

1. Check whether there are 4 lines associated with a figure or not
2. If we find 4 lines, we further check, if they are
    a. connected
    b. closed
    c. perpendicular
    d. they are equal as well

hierarchical feature extraction of Deep Learning

- So, we took a complex task (identifying a square) and broke it in simple less abstract tasks
- Deep Learning essentially does this at a large scale

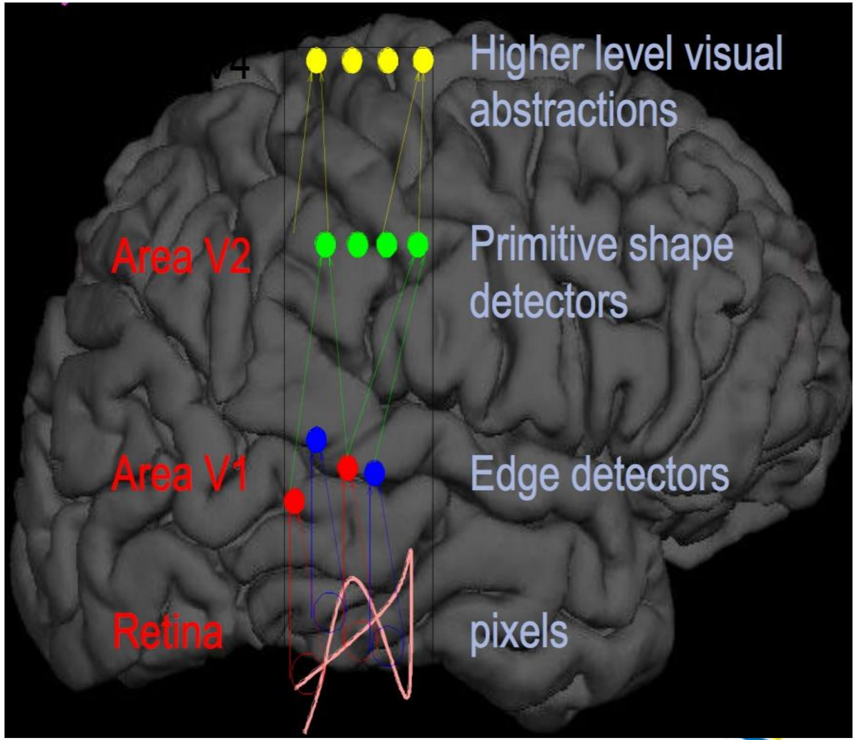
3. Execution time

- Usually, a deep learning algorithm takes a long time to train. This is because there are so many parameters in a deep learning algorithm that training them takes longer than usual.

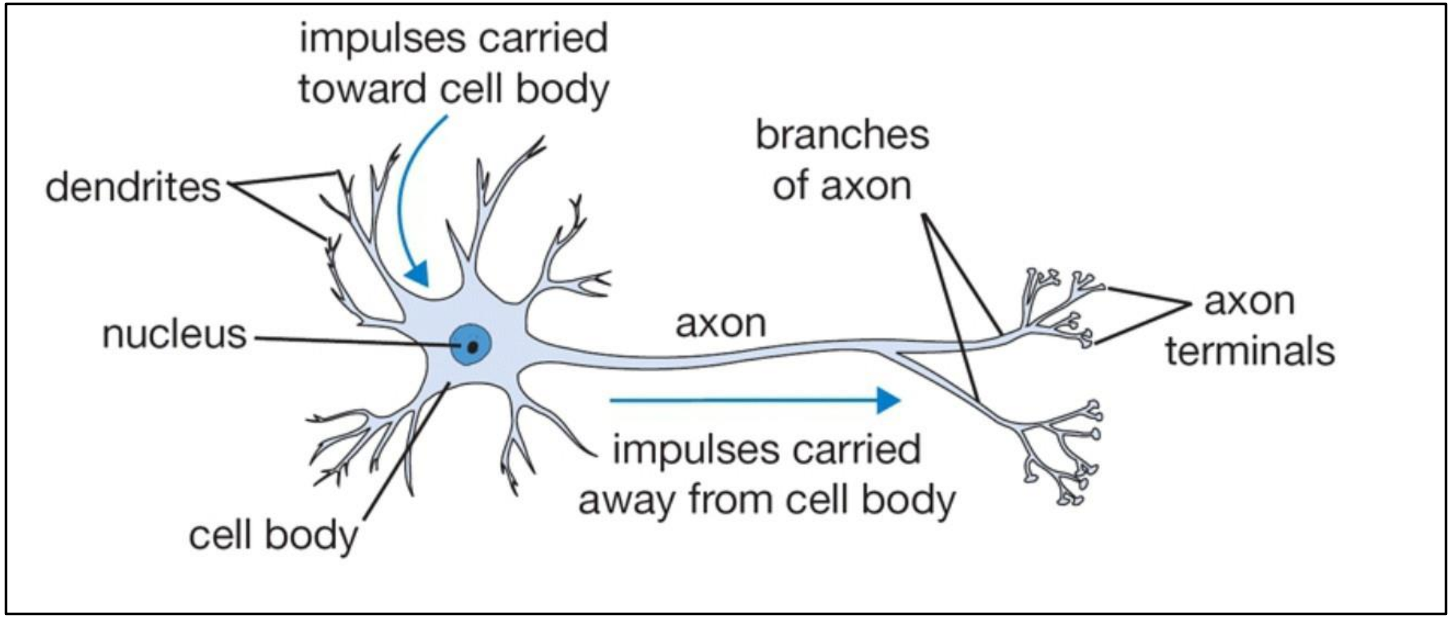
How does Deep Learning work?

"Brain" -like: Feature Hierarchies

- A deep learning model is designed to continually analyze data with a logic structure similar to how a human would draw conclusions



Biological Neuron


impulses carried toward cell body
dendrites
nucleus
cell body
branches of axon
axon
axon terminals
impulses carried away from cell body

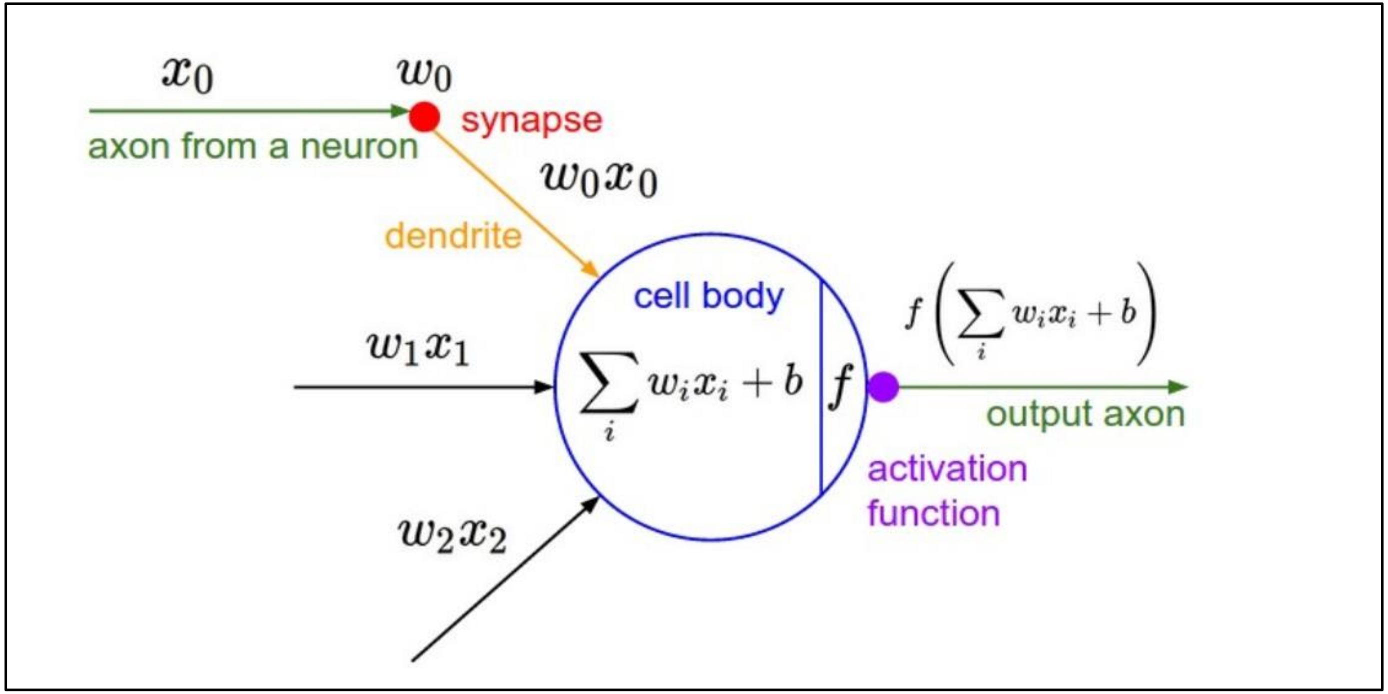
1. Multi-Layer Perceptrons

- is a class of feedforward artificial neural network
- An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer

2. Neurons

- The building block for neural networks are artificial neurons.
- These are simple computational units that have weighted input signals and produce an output signal using an activation function

Artificial Neurons



Neuron Weights

- You may be familiar with linear regression, in which case the weights on the inputs are very much like the coefficients used in a regression equation.
- Weights are often initialized to small random values, such as values in the range 0 to 0.3

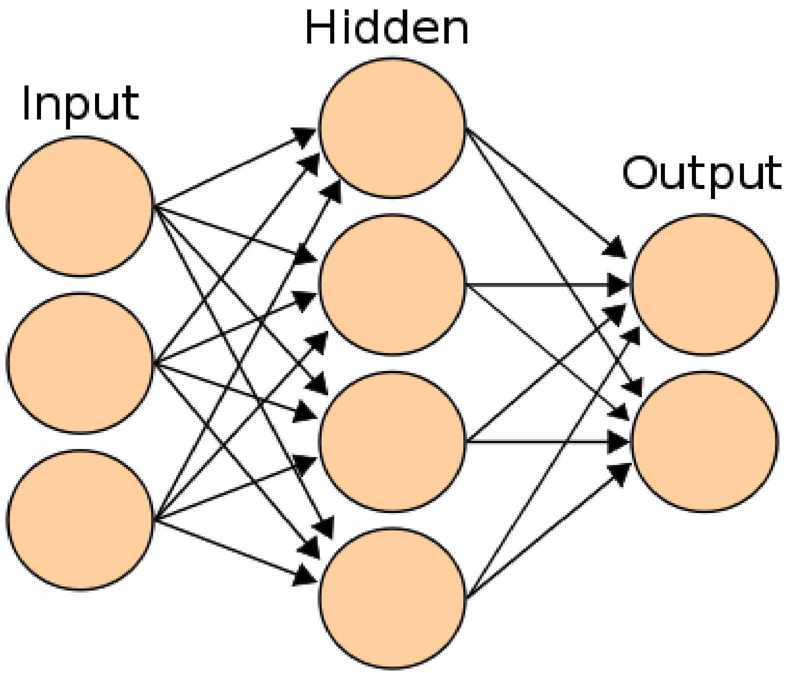
Activation

- The weighted inputs are summed and passed through an activation function, sometimes called a transfer function

3. Networks of Neurons

- Neurons are arranged into networks of neurons
- A row of neurons is called a layer and one network can have multiple layers.

Neural Network Architecture



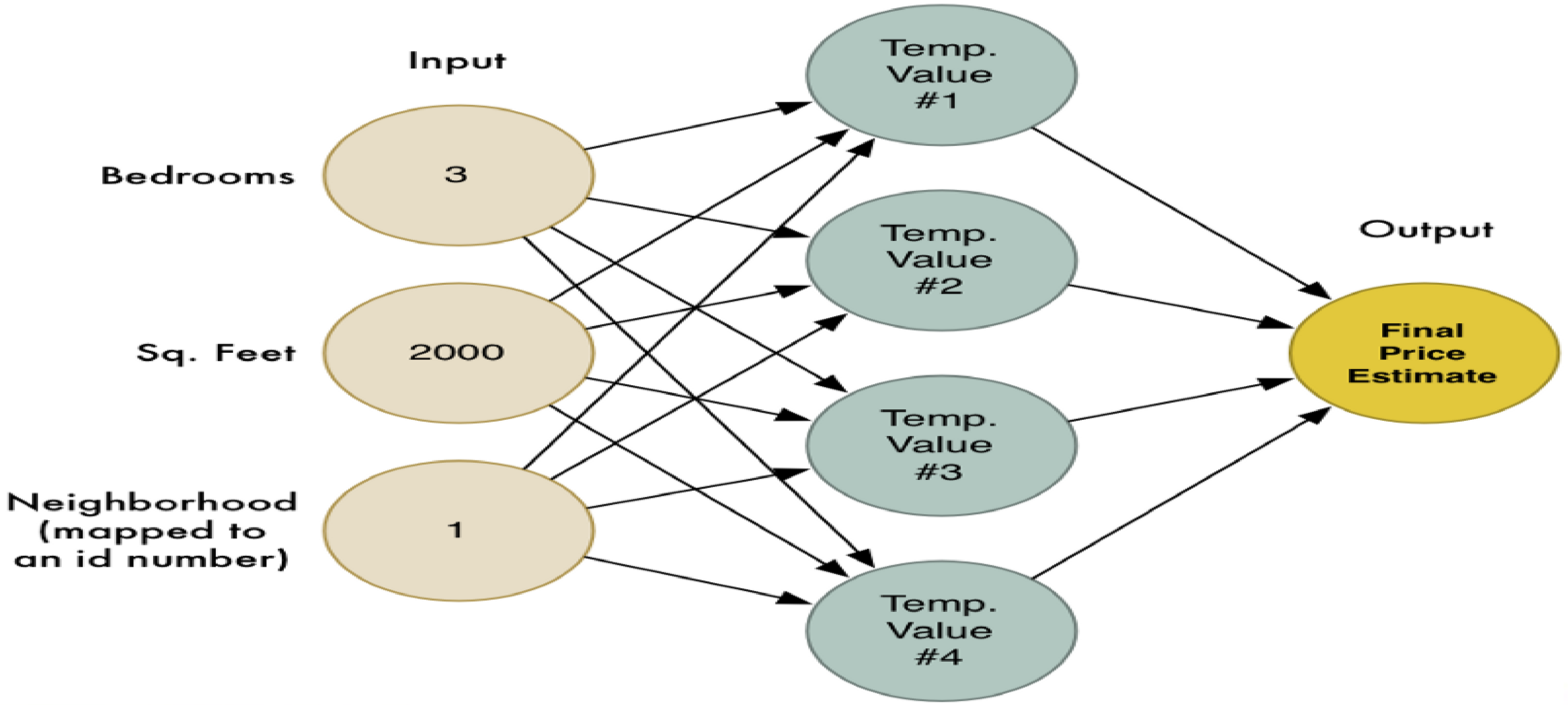
4. Training Networks

- Once configured, the neural network needs to be trained on your dataset.

Neural Network Architecture

- **Input Nodes (input layer):** No computation is done here within this layer, they just pass the information to the next layer
- **Hidden nodes (hidden layer):** In Hidden layers is where intermediate processing or computation is done, they perform computations and then transfer the weights (signals or information) from the input layer to the following layer
- **Output Nodes (output layer):** Here we finally use an activation function that maps to the desired output format
- **Connections and weights:** The *network* consists of connections, each connection transferring the output of a neuron to the input of another neuron

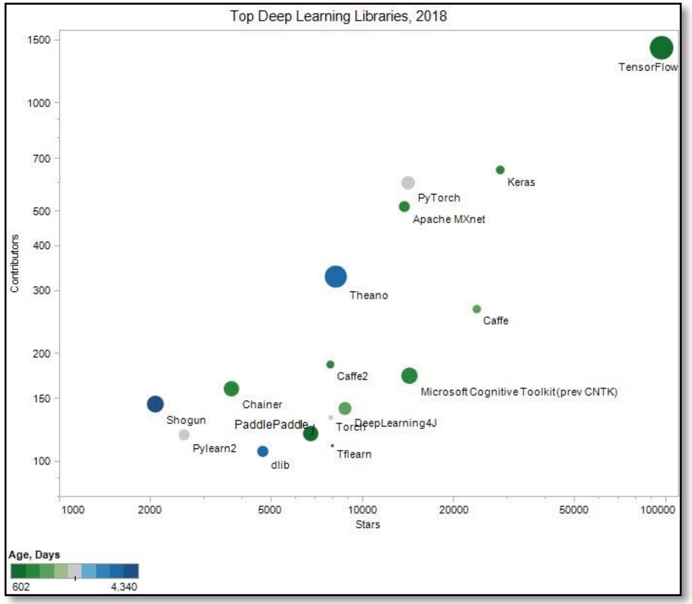
House price prediction neural network



Deep Learning Libraries and Platforms

Deep Learning Libraries

- TensorFlow
- **Keras**
- Caffe
- Microsoft Cognitive Toolkit
- PyTorch
- Apache MXnet
- DeepLearning4J
- Theano
- TFLearn
- Torch
- Caffe2
- PaddlePaddle
- DLib
- Chainer
- Neon
- Lasagne



https://www.kdnuggets.com/2018/04/top-16-open-source-deep-learning-libraries.html

Keras is multi-backend, multi-platform

- Develop in Python, R
  - On Unix, Windows, OSX
- Run the same code with…
  - TensorFlow
  - CNTK
  - Theano
  - MXNet
  - PlaidML
  - ??
- CPU, NVIDIA GPU, AMD GPU, TPU...

What's TensorFlow™?

- Open source software library for numerical computation using data flow graphs?!

- Originally developed by Google Brain Team to conduct machine learning and deep neural networks research

- General enough to be applicable in a wide variety of other domains

- TensorFlow provides an extensive suite of functions and classes that allow users to build various models from scratch

What’s Keras

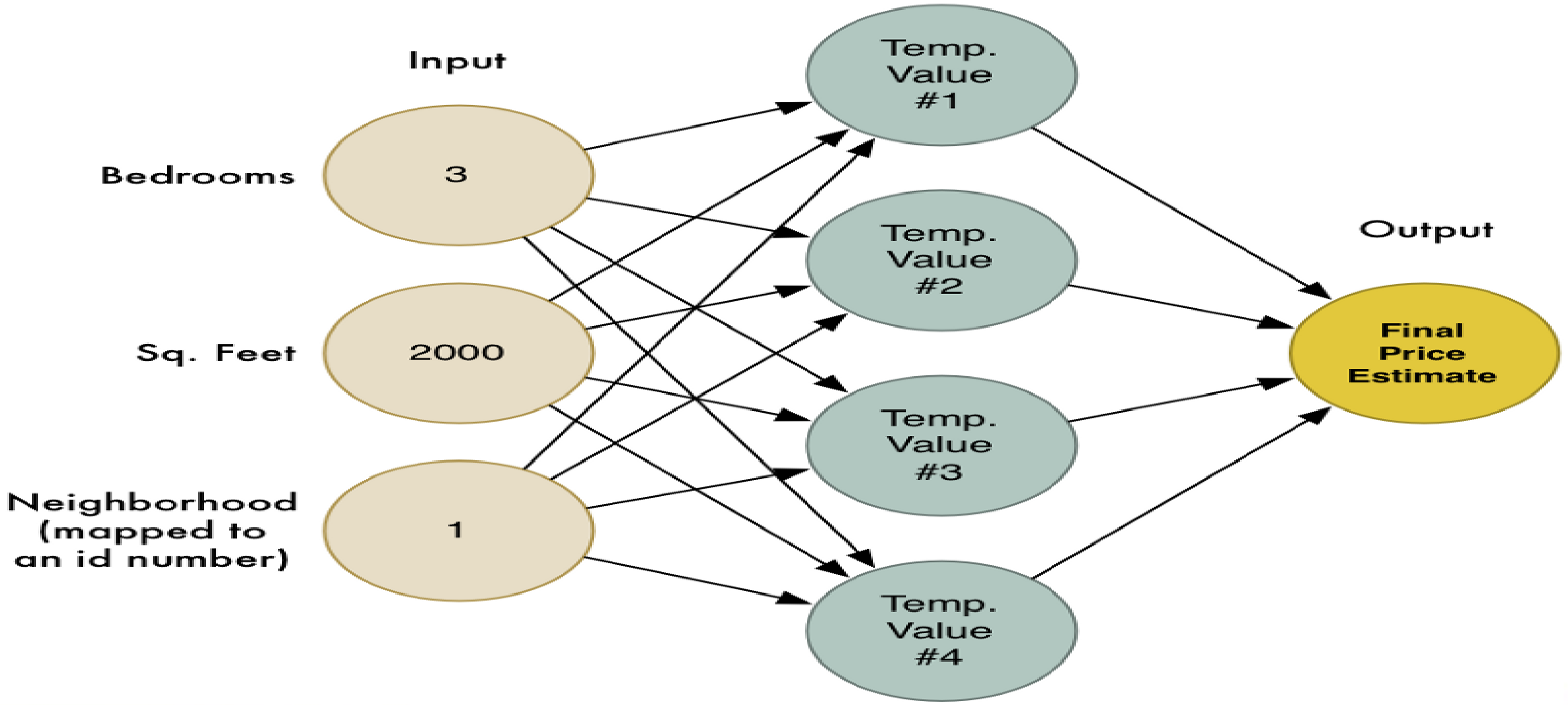
- Keras is a high-level library
- Models created by Keras can be executed on a backend
  - Tensorflow (default)
  - Theano
  - CNTK (Beta)
- It runs smoothly on both CPU and GPU
- Keras supports almost all the models of a neural network – fully connected, convolutional, pooling, recurrent, embedding, etc.
- Keras, being modular in nature
- facilitate experimentations

Getting started with programming

How to use Keras: An Introduction

- Three API styles:
  - Sequential Model
    - Dead simple
    - Only for single-input, single-output, sequential layer stack
    - (does not allow cases where you need to create models that shares layers)
    - Good for 70% of the use cases
  - Functional API
    - Like playing with lego bricks
    - A lot more flexibility
    - Multi input, Multi output, arbitrary graph topologies
    - Like residual networks
  - Model subclassing
    - Maximum flexibility

House price prediction neural network



Sequential Model

- The sequential model is a linear stack of layers
- Create a sequential model by passing a list of layer instances to the constructor
- layers are added piecewise via the sequential object
- input arrays of shape (*, 3)
- output arrays of shape (*, 4)

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(4, input_dim=3))
model.add(Dense(1))
```

Specifying the input shape

- The model needs to know what input shape it should expect
- the first layer in a Sequential Model needs to receive information about its input shape
- There are several possible ways to do this:

Ways of specifying input_shape

- Pass an input_shape argument to the first layer.
- This is a shape tuple(a tuple of integers or None entries)

```
model = Sequential()
model.add(Dense(32,
input_shape=(784,)))
```

- Some 2D layers, such as Dense, support the specification of their input shape via the argument input_dim

```
model = Sequential()
model.add(Dense(32,
input_dim=784))
```

Functional model

- you must create and define a standalone "Input" layer that specifies the shape of input data
- "shape" argument is a tuple that indicates the dimensionality of the input data
- The layers in the model are connected pairwise
- This is done by specifying where the input comes from when defining each new layer
- Note how the " input_layer " layer connects to the "Dense" layer
- Create the model: As with the Sequential API, the model is the thing that you can summarize, fit, evaluate, make predictions

```
from keras.models import Model
from keras.layers import Input
from keras.layers import Dense

input_layer = Input(shape=(3,))
hidden_layer = Dense(4)(input_layer)
out_layer = Dense(1)(input_layer)
model = Model(inputs= input_layer,
outputs=out_layer)
```

Common attribute of sequential and functional model

- Model.layer: flattened list of the layers comprising the model
- Model.input: the list of input tensors of the model.
- Model.output: the list of output tensors of the model.
- Model.summary(): prints a summary representation of your model
- Model.get_config(): returns a dictionary containing the configuration of the model
- Model.get_weights(): returns a list of all weight tensors in the model, as Numpy arrays
- Model.set_weights(): sets the values of the weights of the model from a list of Numpy arrays

Compilation

- Before training a model, we need to configure the learning process via the Compilation method
- It receives three arguments:
  - An optimizer
  - A loss function
  - A list of metrics

```
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

Training

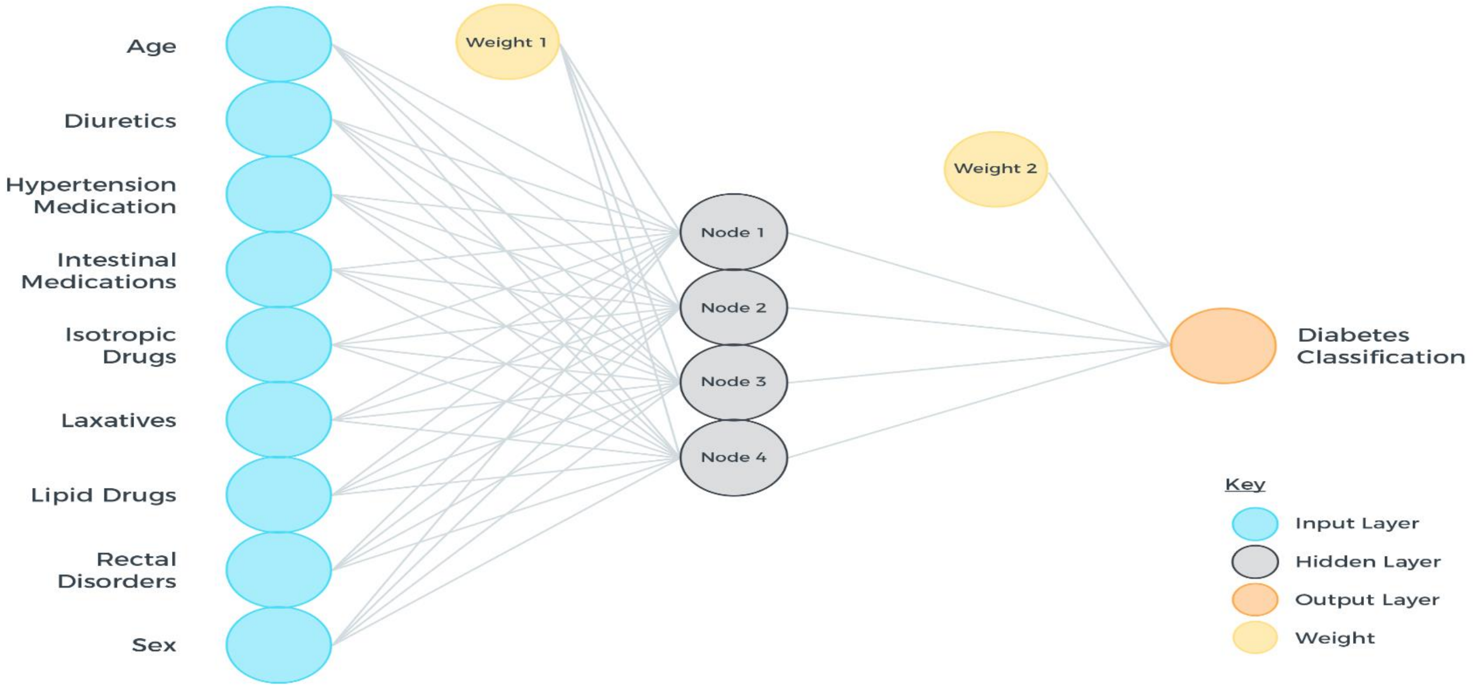
- Trained on Numpy arrays of input data and labels
  - Data: Numpy array of training data
  - Labels: Numpy array of target (label) data
- For training a model, you will typically use the Fit() function

```
import numpy as np
data = np.random.random((1000, 100))
labels = np.random.randint(2,
size=(1000, 1))

# Train the model, iterating on the data
in batches of 32 samples
model.fit(data, labels, epochs=10,
batch_size=32)
```

Use case: predict Diabetes

Diabetes Neural Network



Use case: predict diabetes

```
dataset = pd.read_csv("diabetes.csv", header=None).values
# print(dataset)
import numpy as np
X_train, X_test, Y_train, Y_test = train_test_split(dataset[:,0:8], dataset[:,8],
                                        test_size=0.25, random_state=87)

my_first_nn = Sequential() # create model
my_first_nn.add(Dense(20, input_dim=8, activation='relu')) # hidden layer
my_first_nn.add(Dense(1, activation='sigmoid')) # output layer

my_first_nn.compile(loss='binary_crossentropy', optimizer='adam')
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100, verbose=0,
                              initial_epoch=0)
print(my_first_nn.summary())
print(my_first_nn.evaluate(X_test, Y_test, verbose=0))
```

Result

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 20)                180
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 21
=================================================================
Total params: 201
Trainable params: 201
Non-trainable params: 0
```

References

- https://www.youtube.com/watch?v=yX8KuPZCAMo
- https://www.linkedin.com/pulse/list-useful-links-videos-slides-articles-deep-farshid-pirahansiah?articleId=6274891035365732352#comments-6274891035365732352&trk=prof-post

Books

- TensorFlow for Machine Intelligence (TFFMI)
- Hands-On Machine Learning with Scikit-Learn and TensorFlow. Chapter 9: Up and running with TensorFlow
- Fundamentals of Deep Learning. Chapter 3: Implementing Neural Networks in TensorFlow (FODL)
- Deep Learning with Keras: Implementing deep learning models and neural