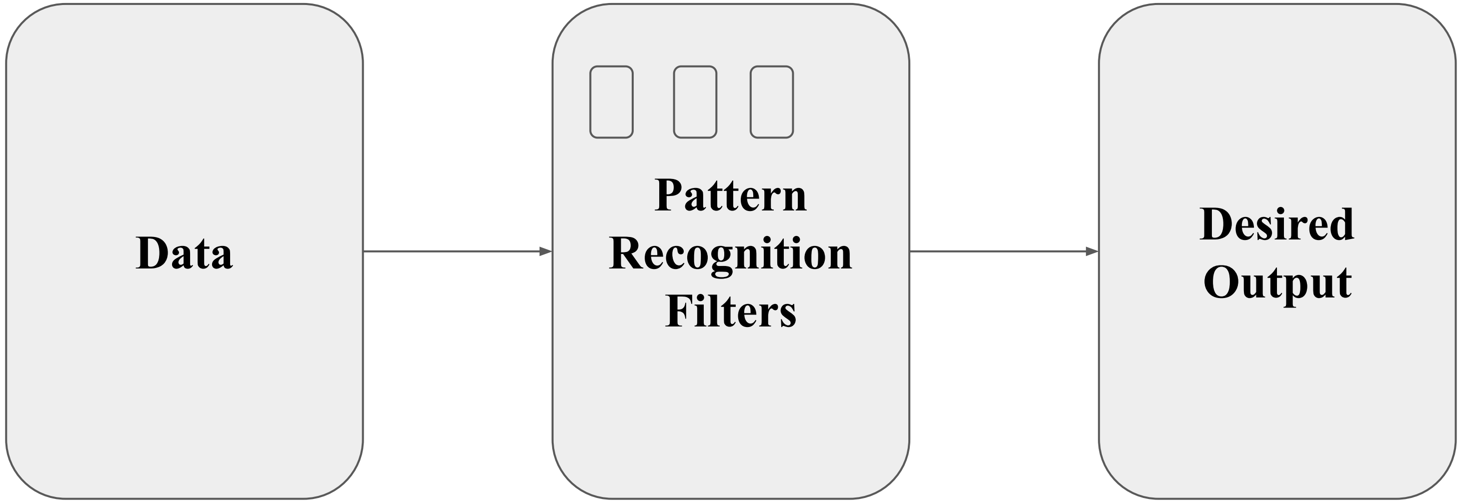
Day 3
GNITS AI Workshop

Continued ...

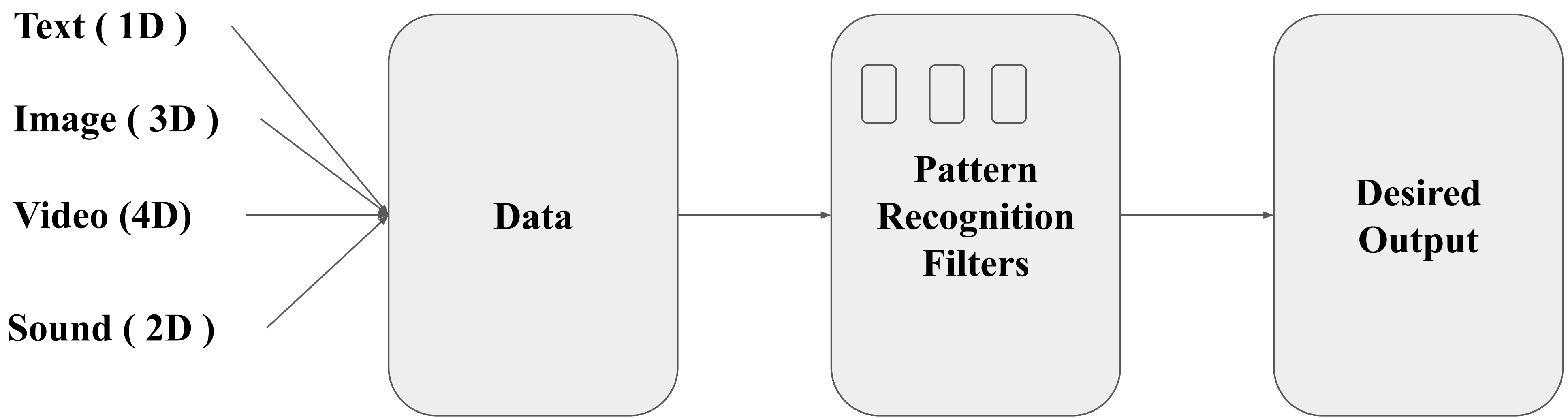
Video

- **https://www.youtube.com/watch?v=Q3oItpVa9fs**

Pattern Recognition



Pattern Recognition



Text (1D)

Image (3D)

Video (4D)

Sound (2D)

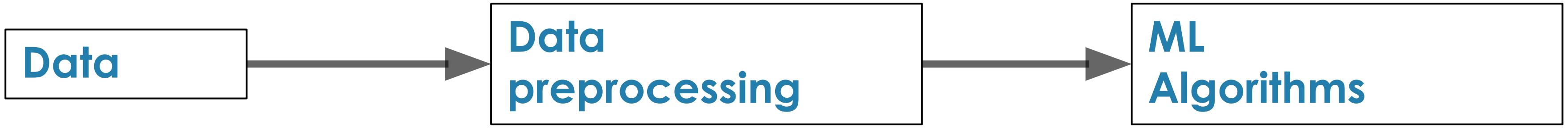
Data

Pattern Recognition Filters

Desired Output

Data

Types	Image	Text	Signal
dimensions	3D – width, height, depth	2D – length of text, sentences	2D – times series signal
Data preprocessing	Can be given directly or transformed if required	Word embedding are to be added to text	Noise reduction or transformation or can be given directly
Network input W * H * D * F	W – width of image H – height of image D – depth of image F – Frames, if image F = 1 if video F = frames	W – length of word embedding H – always 1 D – always 1 F – always 1	W – time of signal H – amplitude of signal D – always 1 F – always 1

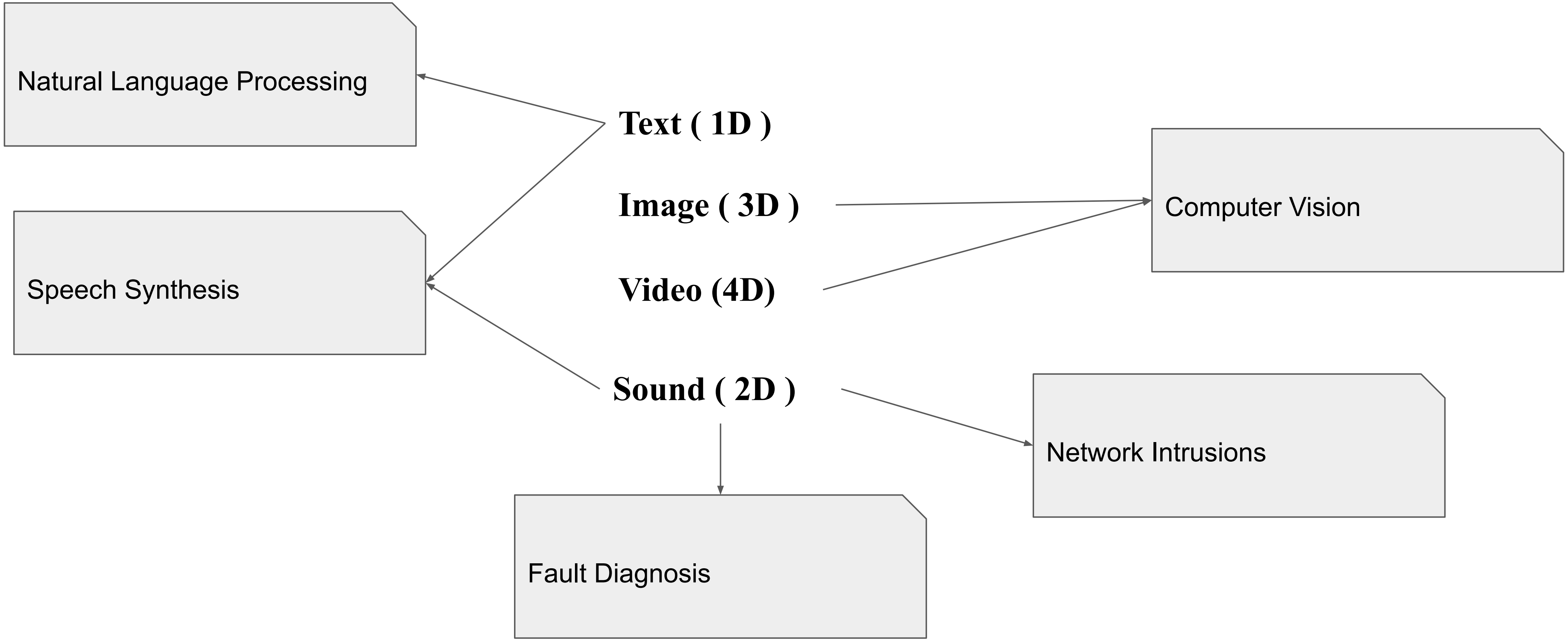


Data

Data preprocessing

ML Algorithms

Fields of Study



Natural Language Processing

Text (1D)

Image (3D)

Computer Vision

Speech Synthesis

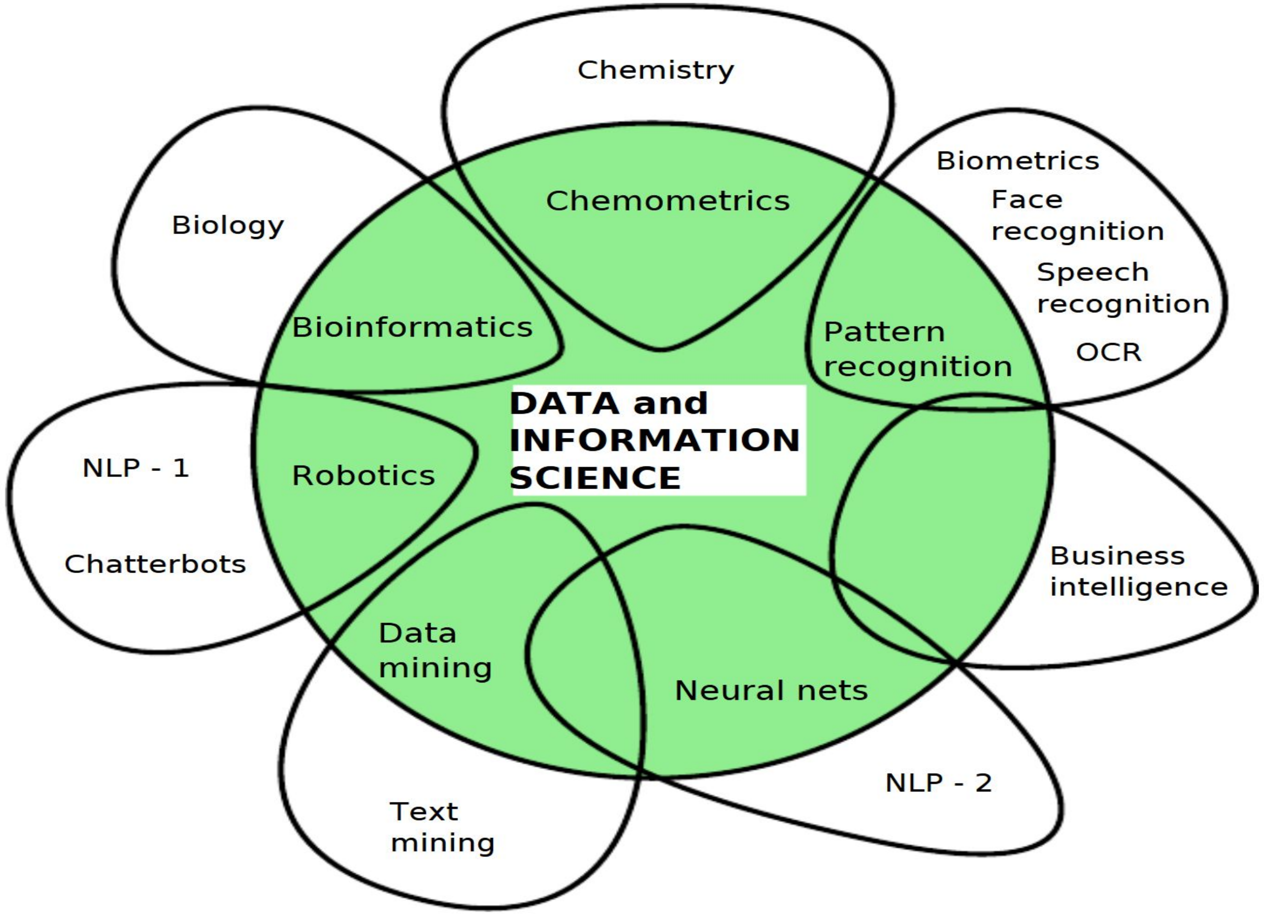
Video (4D)

Sound (2D)

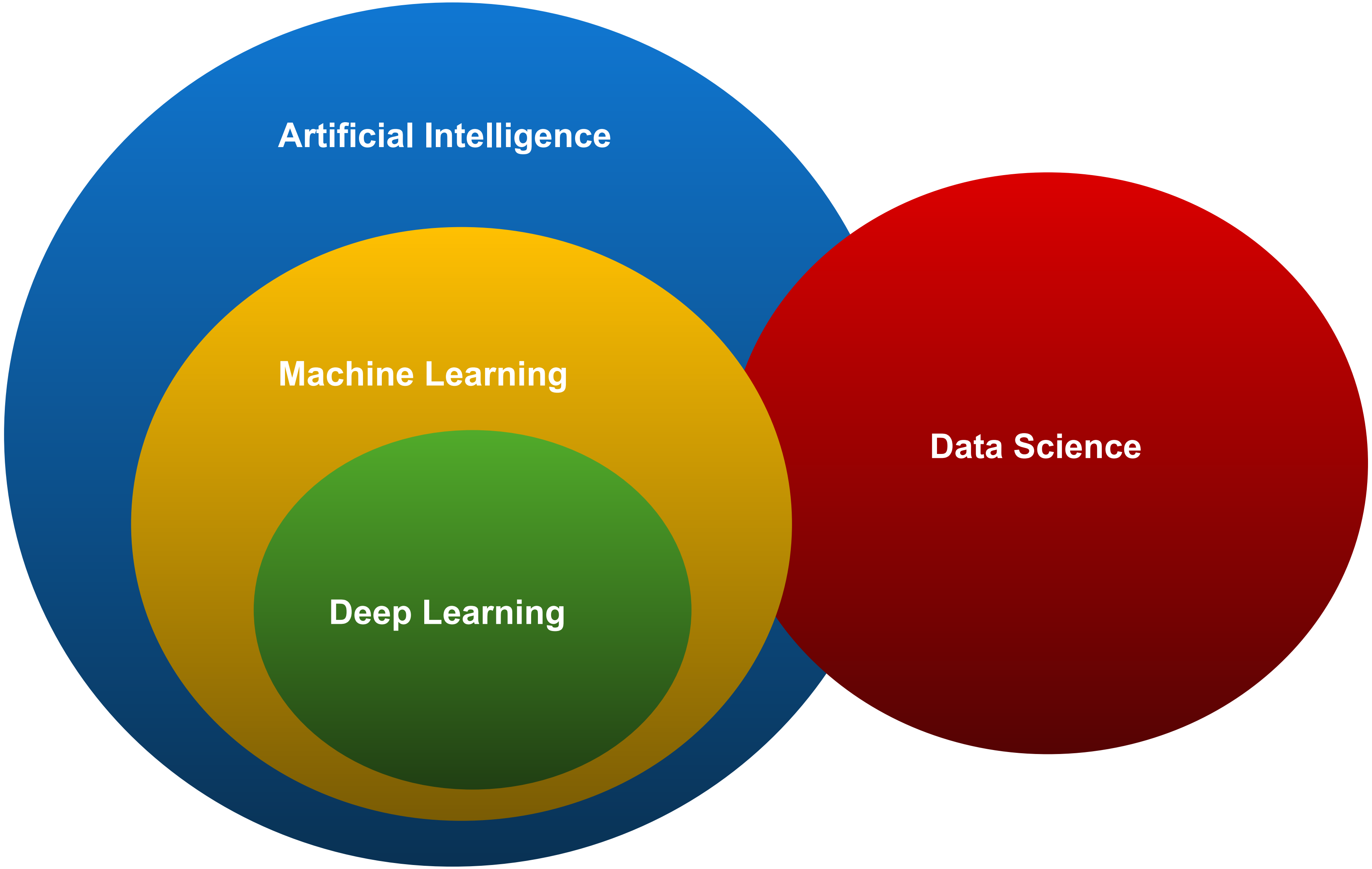
Network Intrusions

Fault Diagnosis

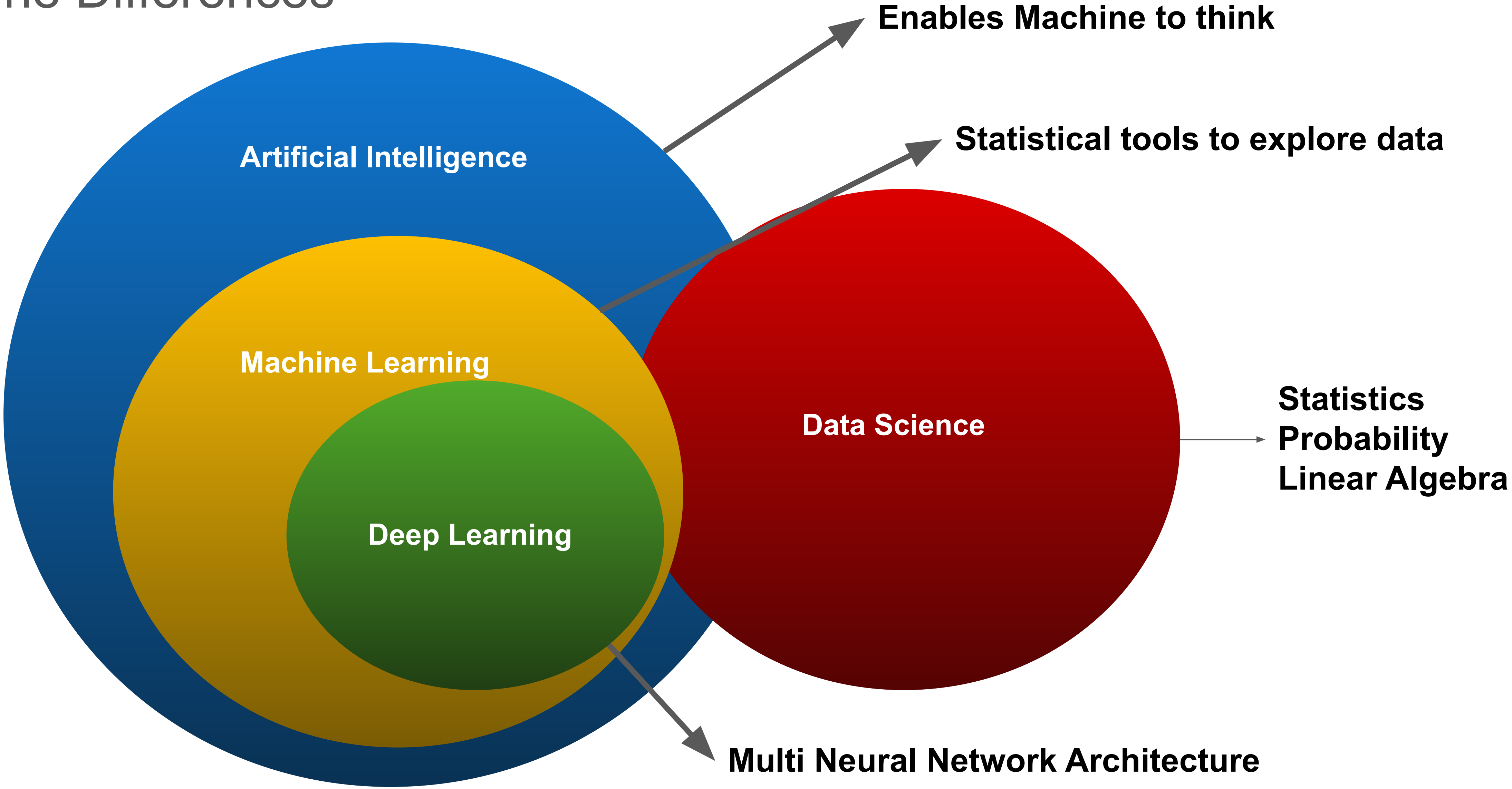
Applications



The Differences



The Differences



Enables Machine to think

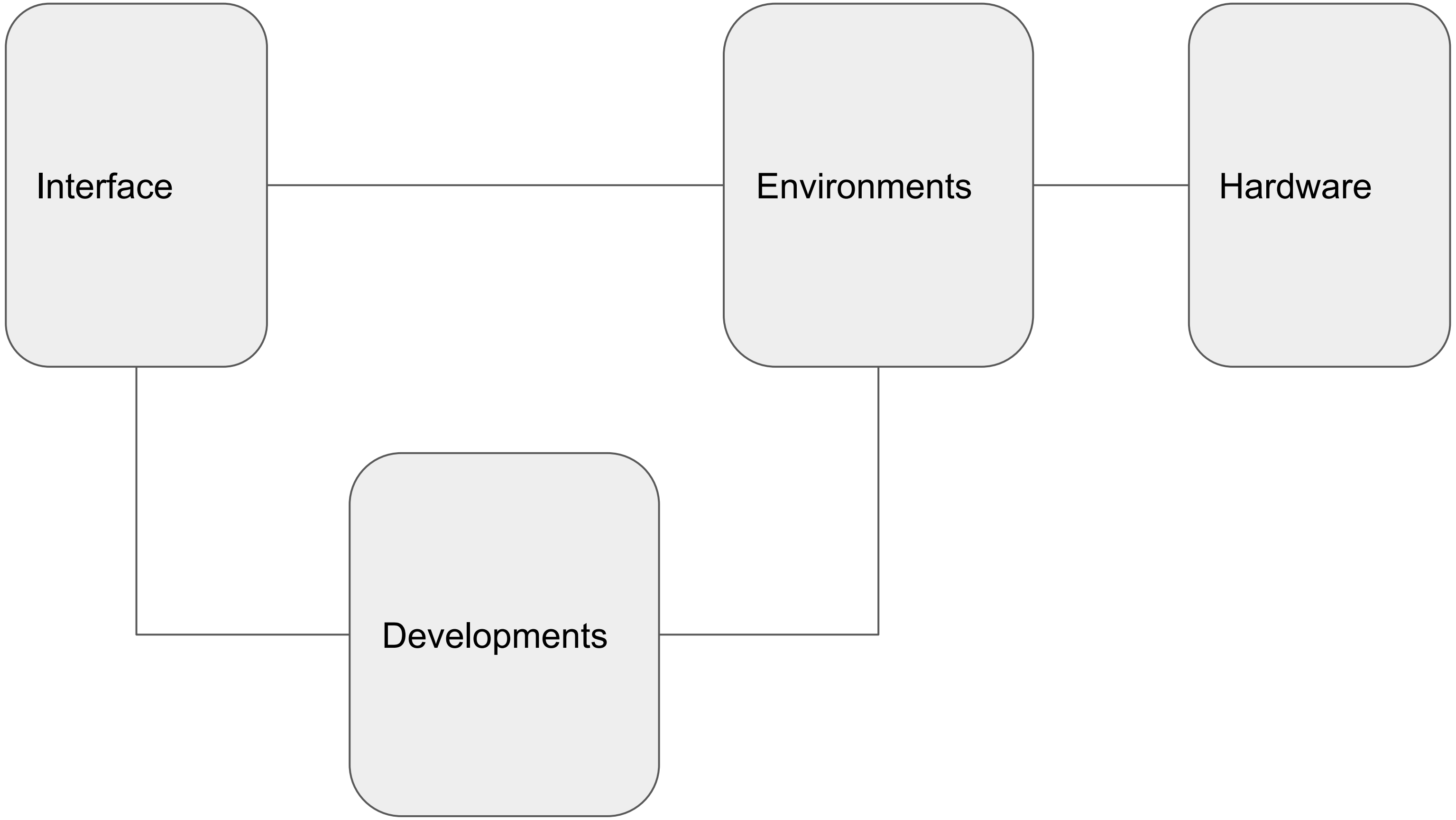
Statistical tools to explore data

Statistics
Probability
Linear Algebra

Multi Neural Network Architecture

Video

Applications



AI Hardware and the Battle for More Computational Power

More computational power and cost-efficiency

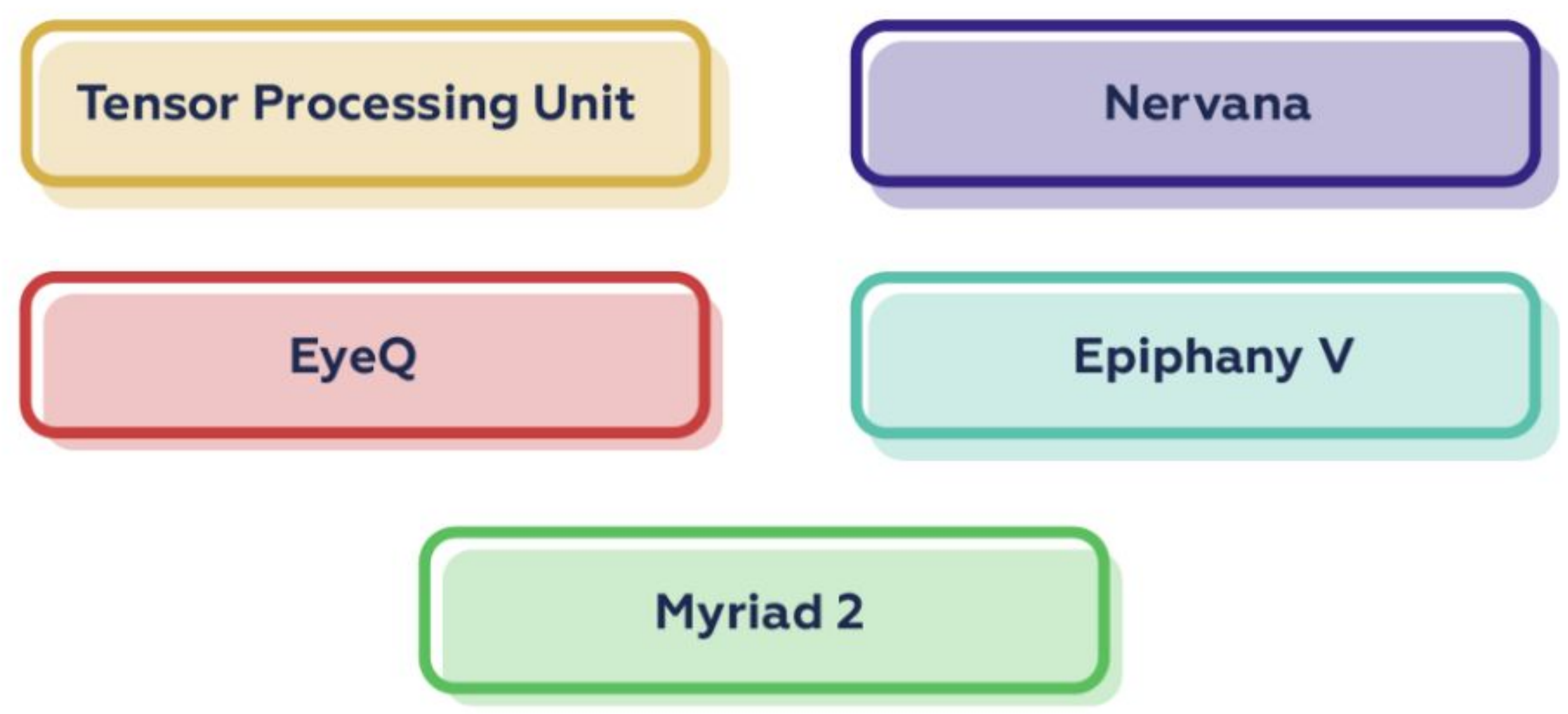
Cloud and Edge computing

Faster insights

New materials

New architectures

AI Hardware and the Battle for More Computational Power



Tensor Processing Unit

Nervana

EyeQ

Epiphany V

Myriad 2

Top 5 AI Hardware Solutions

BENTOML

Model Serving Made Easy
From trained ML models to production-grade prediction services with just few lines of code

Unified Format for deployment

Unified model packing format enabling both online and offline serving on any platform.

High Performance model Serving

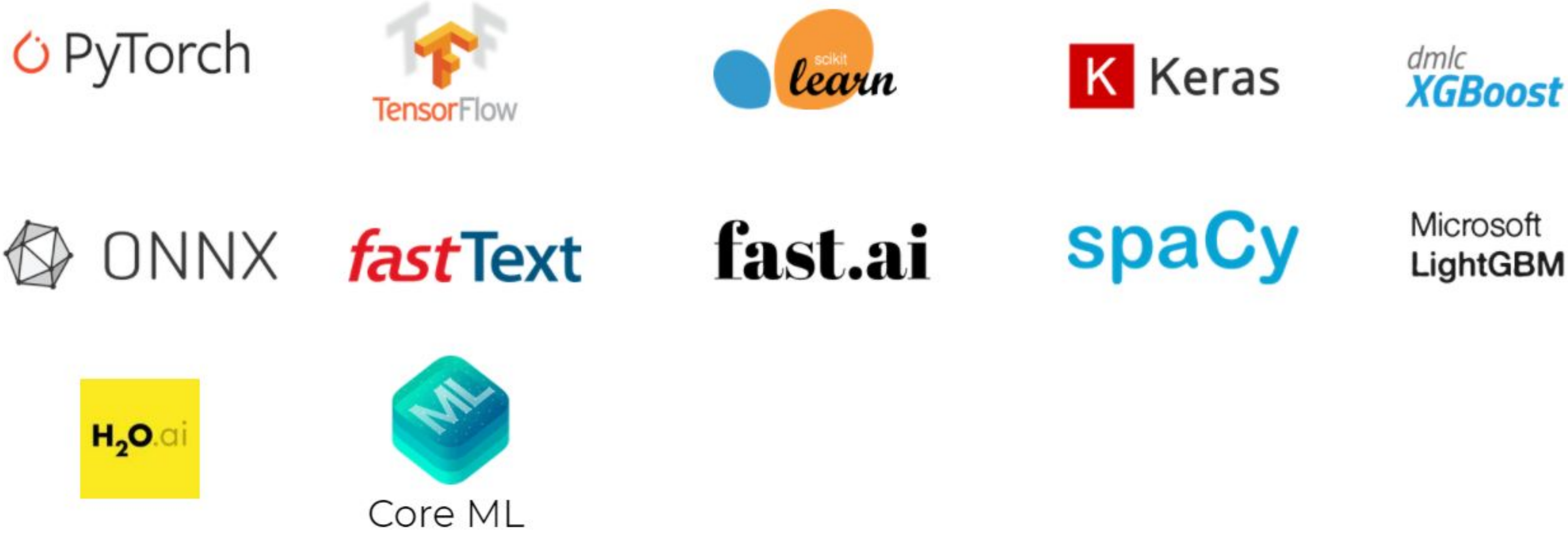
100x the throughput of your regular flask based model server, advanced micro-batching mechanism.

Devops best practices baked in

Deliver high quality prediction services that speaks the Devops language perfectly with common infrastructure.

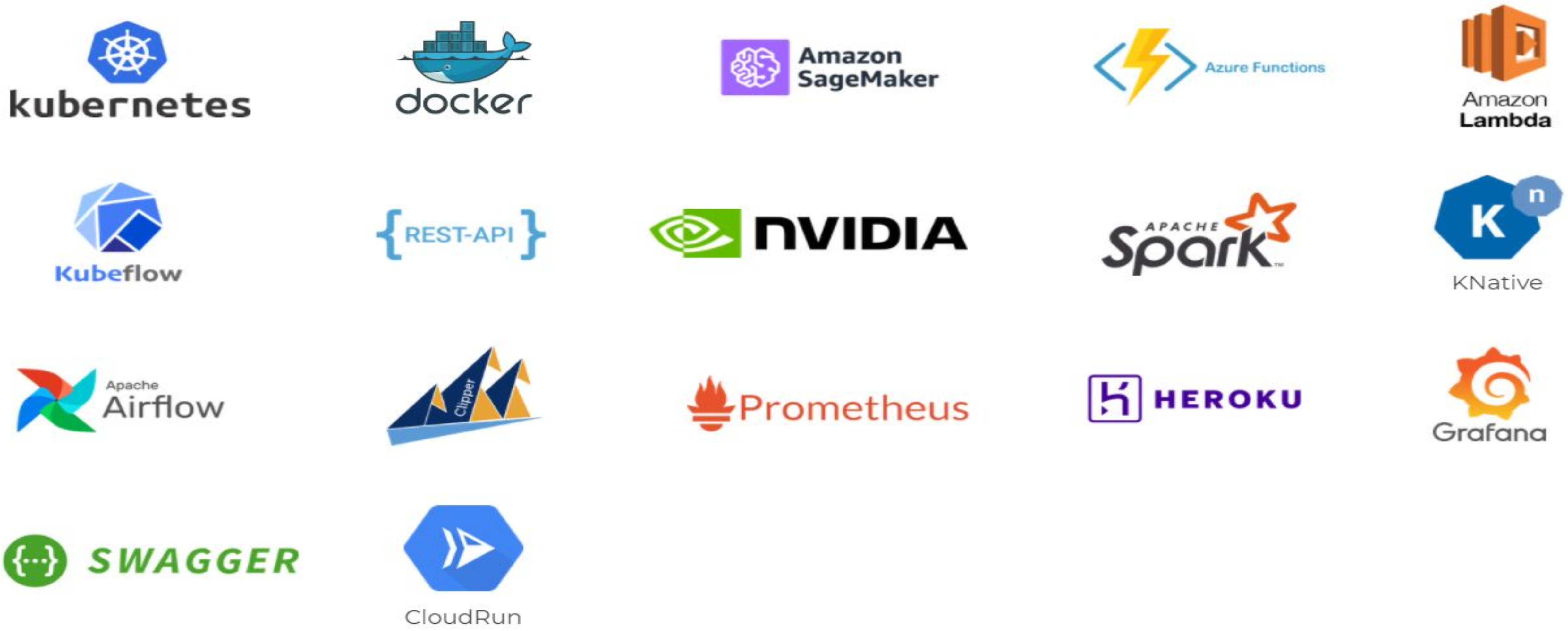
BENTOML

Support all major ML frameworks



BENTOML

Built to work with Devops & Infrastructure tools



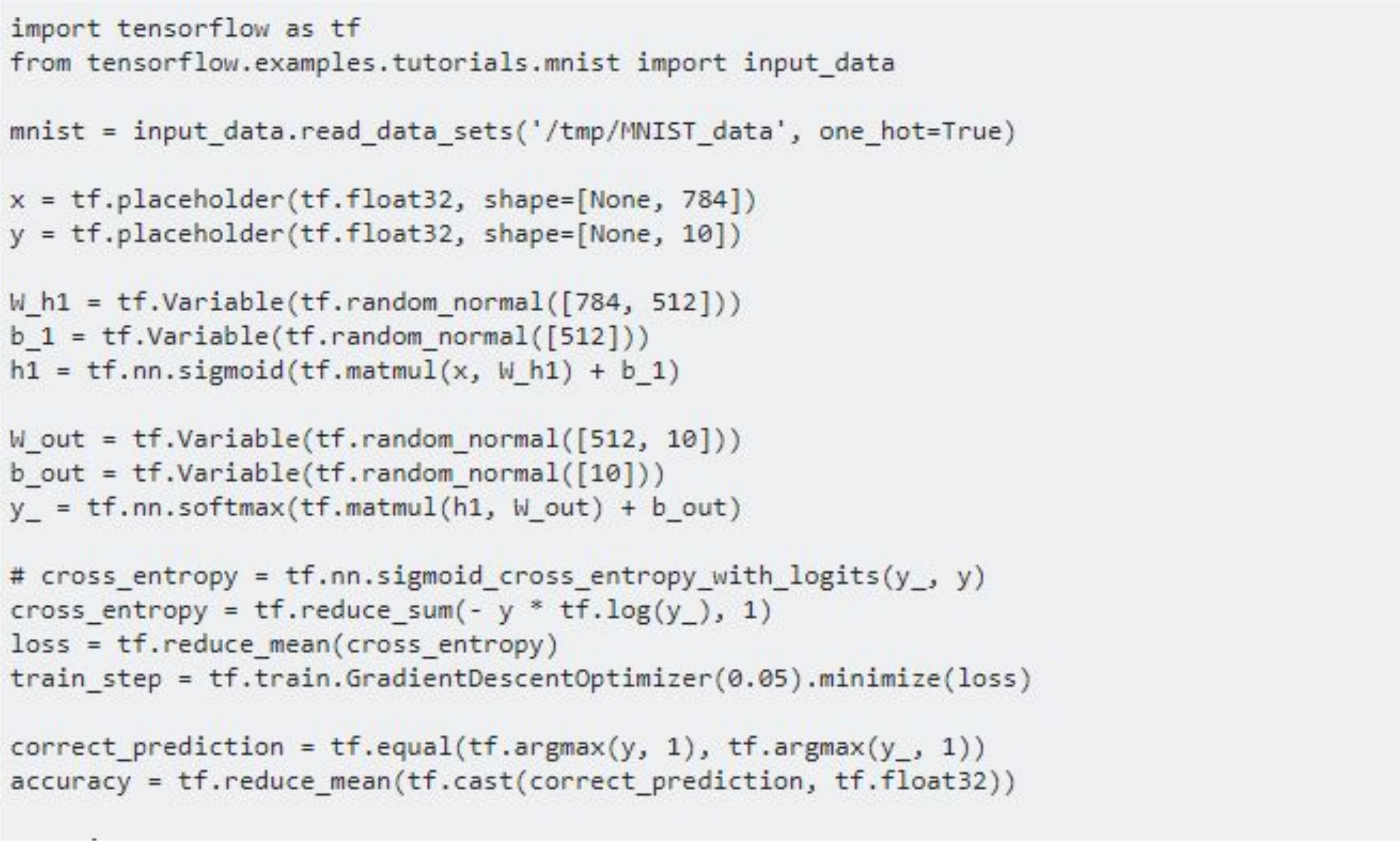
WHAT IS BEING USED IN INDUSTRY FOR CREATING DEEP NETWORKS

Tensorflow for Artificial Neural Network:

[https://github.com/tensorflow/tensorflow]

- ✔ High Level Neural Network API using Python.
- ✔ Open source library for numerical computation using data flow graphs.
- ✔ Deploy framework across multiple GPUs and CPUs.
- ✔ Supports all types of Neural Networks. More usage for Generative models and RNN.
- ✔ Multidimensional data arrays(tensors) are used for communications between nodes.

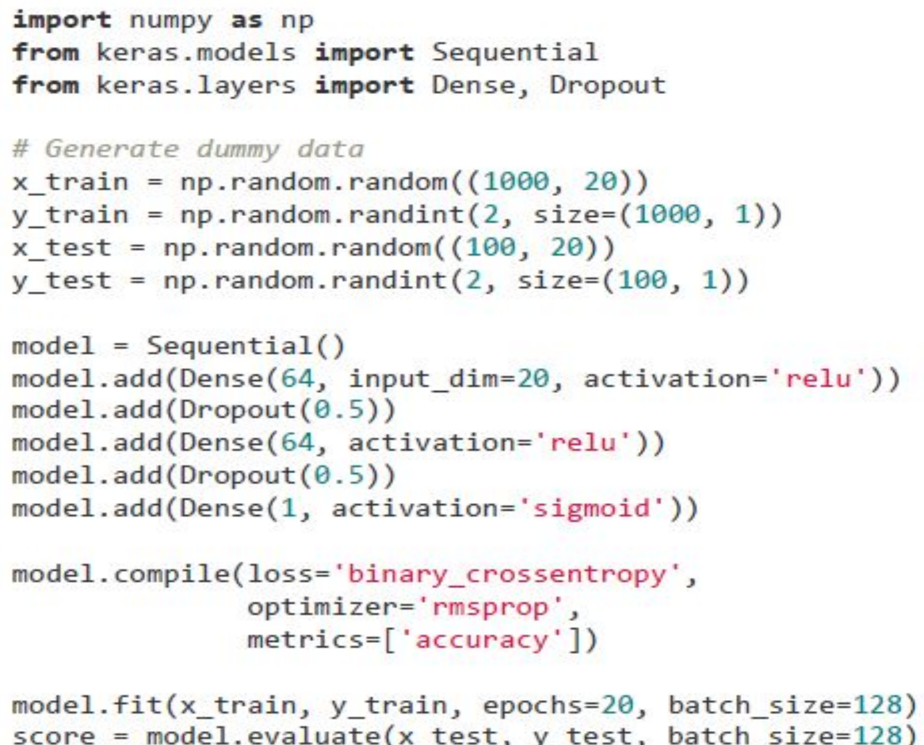
WHAT IS BEING USED IN INDUSTRY FOR CREATING DEEP NETWORKS



```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets('/tmp/MNIST_data', one_hot=True)

x = tf.placeholder(tf.float32, shape=[None, 784])
y = tf.placeholder(tf.float32, shape=[None, 10])

W_h1 = tf.Variable(tf.random_normal([784, 512]))
b_1 = tf.Variable(tf.random_normal([512]))
h1 = tf.nn.sigmoid(tf.matmul(x, W_h1) + b_1)

W_out = tf.Variable(tf.random_normal([512, 10]))
b_out = tf.Variable(tf.random_normal([10]))
y_ = tf.nn.softmax(tf.matmul(h1, W_out) + b_out)

# cross_entropy = tf.nn.sigmoid_cross_entropy_with_logits(y_, y)
cross_entropy = tf.reduce_sum(- y * tf.log(y_), 1)
loss = tf.reduce_mean(cross_entropy)
train_step = tf.train.GradientDescentOptimizer(0.05).minimize(loss)

correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

Keras : [https://github.com/fchollet/keras]

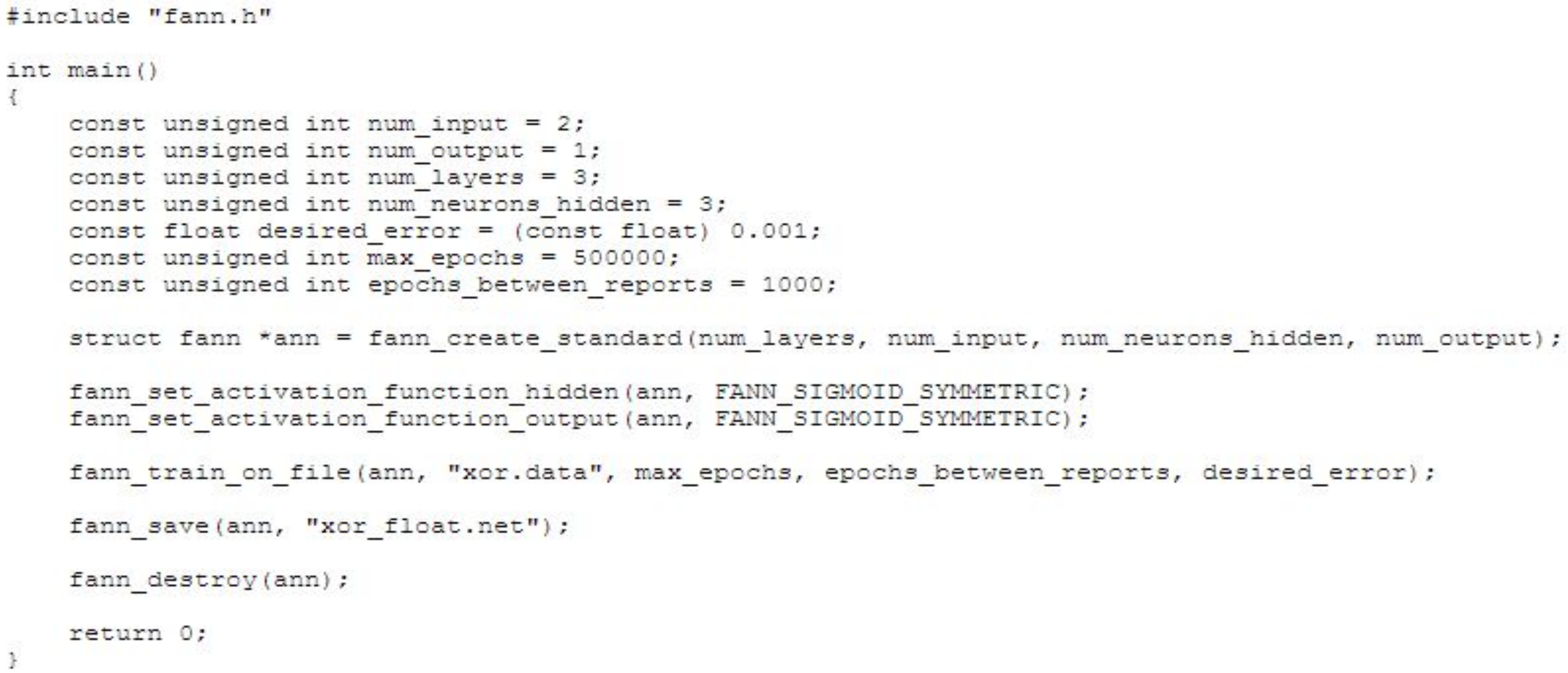
- ✔ High level neural network API in Python. Can be built on Tensor flow library.
- ✔ Support for Fully Connected and Sparsely Connected.
- ✔ Supports CNN and RNN.
- ✔ Default use Tensor flow manipulation library.
- ✔ Enable fast experimentation with easy and fast prototyping.



```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Dropout

# Generate dummy data
x_train = np.random.random((1000, 20))
y_train = np.random.randint(2, size=(1000, 1))
x_test = np.random.random((100, 20))
y_test = np.random.randint(2, size=(100, 1))

model = Sequential()
model.add(Dense(64, input_dim=20, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))

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

model.fit(x_train, y_train, epochs=20, batch_size=128)
score = model.evaluate(x_test, y_test, batch_size=128)
```

FANN: Fast Artificial Neural Network : **[https://github.com/libfann/fann]**

- ✔ Multilayer Artificial Neural Network Library in C.
- ✔ Support for Fully Connected and Sparsely Connected.
- ✔ Easy to use, save and load entire ANNs.
- ✔ Several different activation functions implemented.
- ✔ Framework for easy handling of training data sets.



```
#include "fann.h"

int main()
{
    const unsigned int num_input = 2;
    const unsigned int num_output = 1;
    const unsigned int num_layers = 3;
    const unsigned int num_neurons_hidden = 3;
    const float desired_error = (const float) 0.001;
    const unsigned int max_epochs = 500000;
    const unsigned int epochs_between_reports = 1000;

    struct fann *ann = fann_create_standard(num_layers, num_input, num_neurons_hidden, num_output);

    fann_set_activation_function_hidden(ann, FANN_SIGMOID_SYMMETRIC);
    fann_set_activation_function_output(ann, FANN_SIGMOID_SYMMETRIC);

    fann_train_on_file(ann, "xor.data", max_epochs, epochs_between_reports, desired_error);

    fann_save(ann, "xor_float.net");

    fann_destroy(ann);

    return 0;
}
```

OTHER LIBRARIES

using C++

Caffe : **[http://caffe.berkeleyvision.org/tutorial/]**

-> Supports CNN, RNN, LSTM and fully connected NN designs, GPU enabled

PaddlePaddle : **[https://github.com/PaddlePaddle/Paddle]**

-> highly efficient RNN, but can also support CNN and complicated DNN

Torch: **[http://torch.ch/]**

-> Neural networks and Energy based models

using java

Deeplearning4j : **[https://github.com/deeplearning4j/deeplearning4j]**

using python

Theanets: **[https://github.com/lmjohns3/theanets]**

Lasagane: **[https://github.com/Lasagne/Lasagne]**

Questions Time