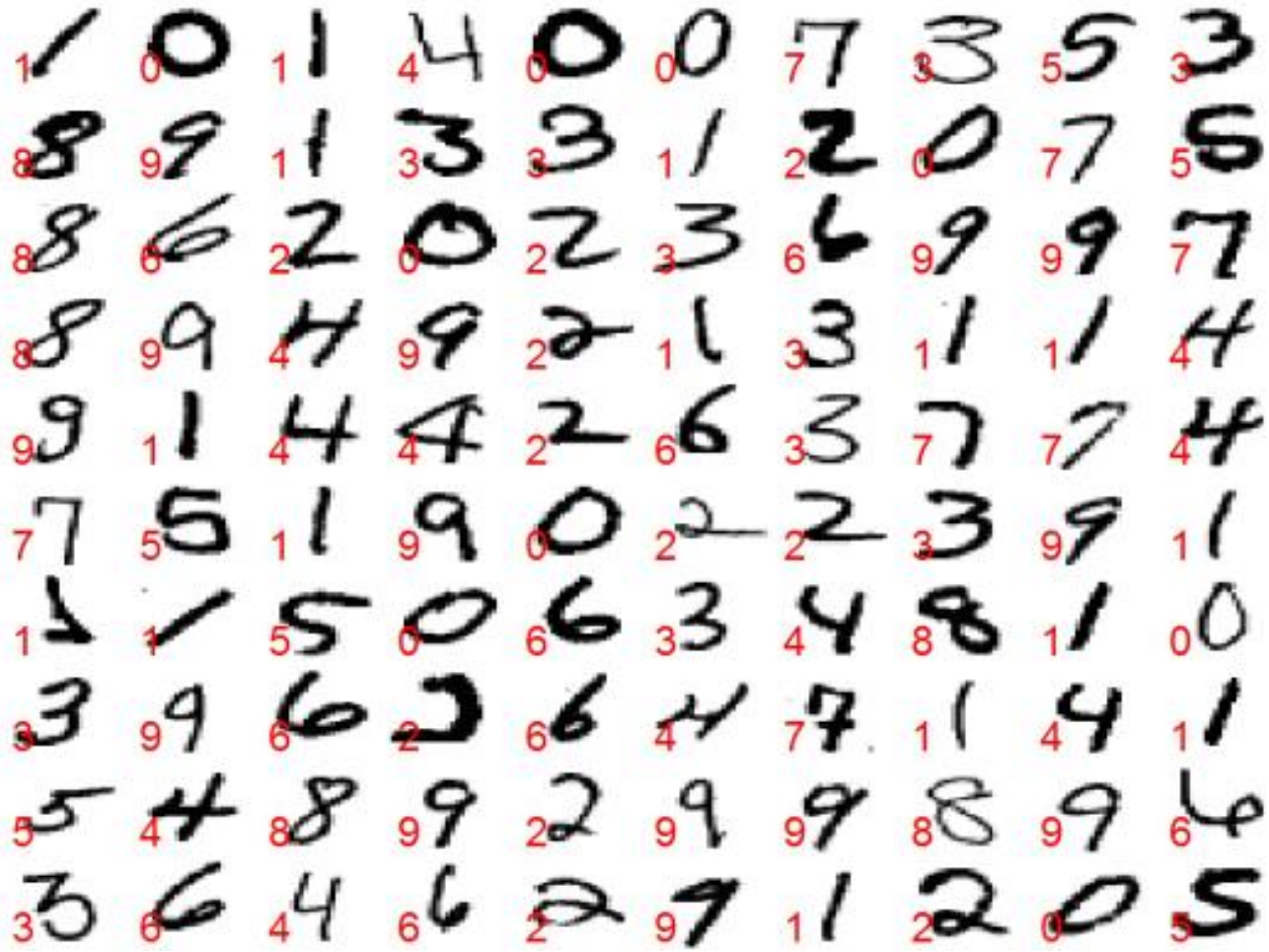
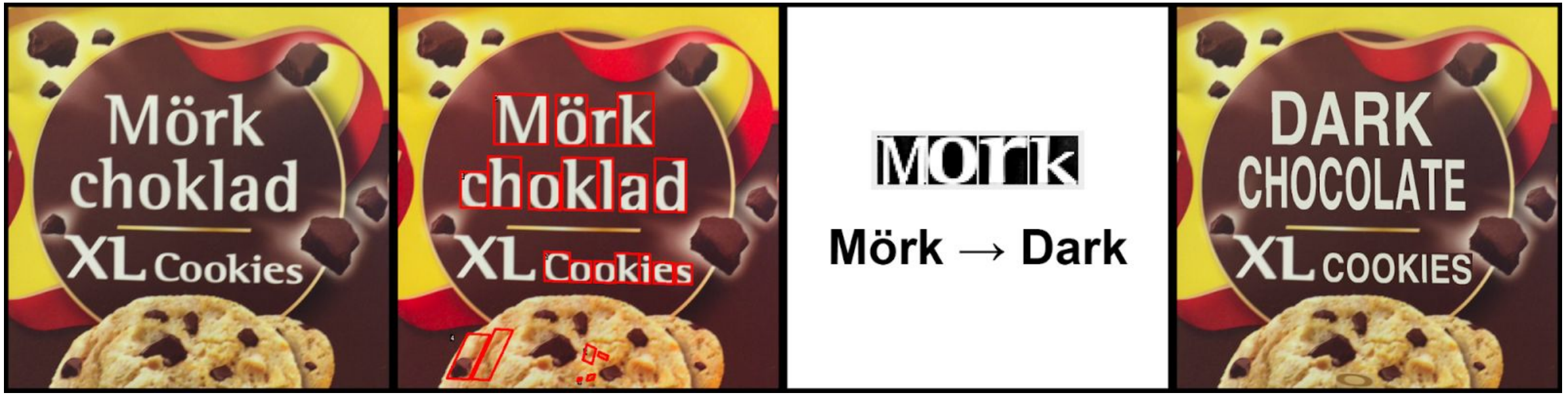
Deep Learning with Apache Spark

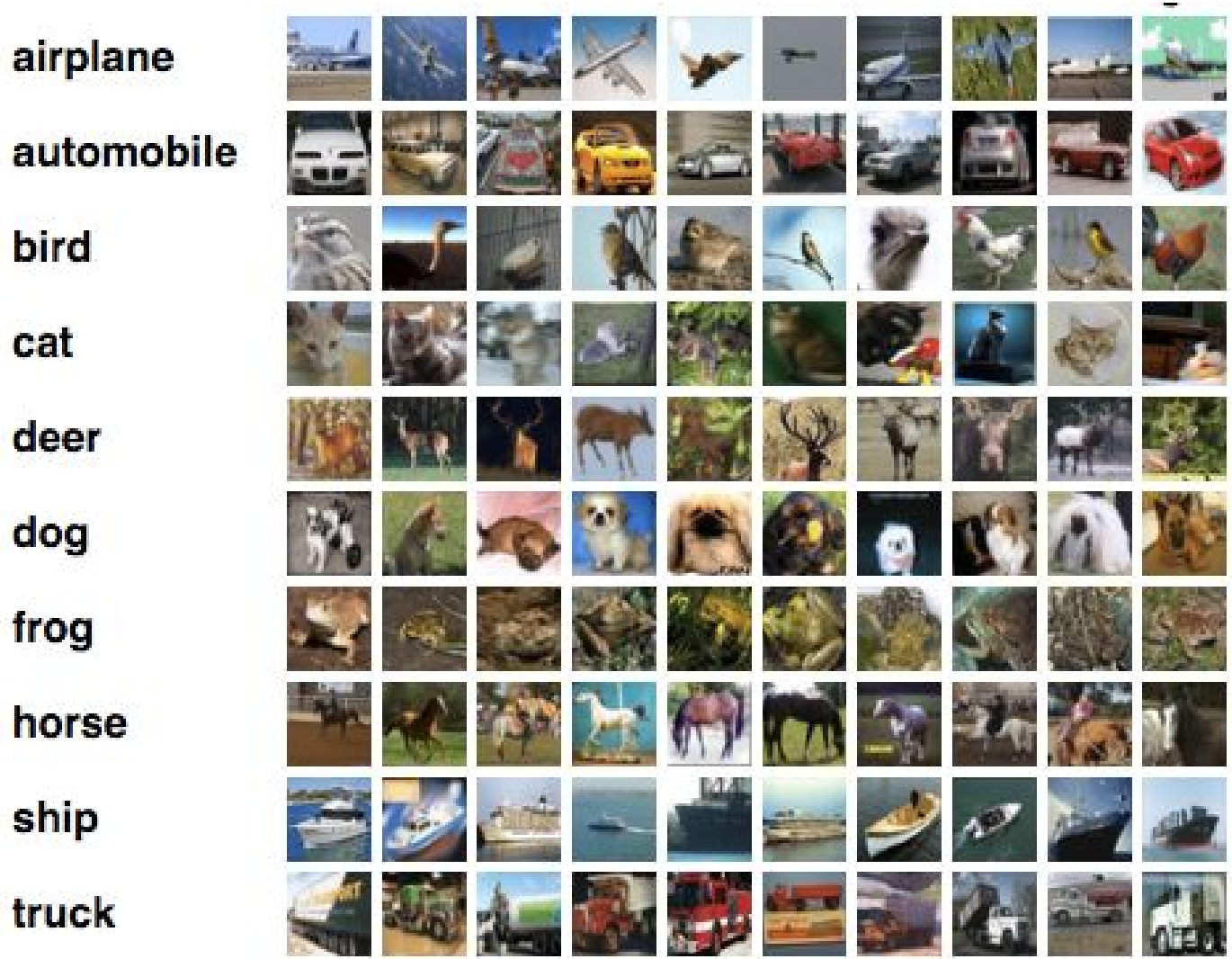
An Introduction

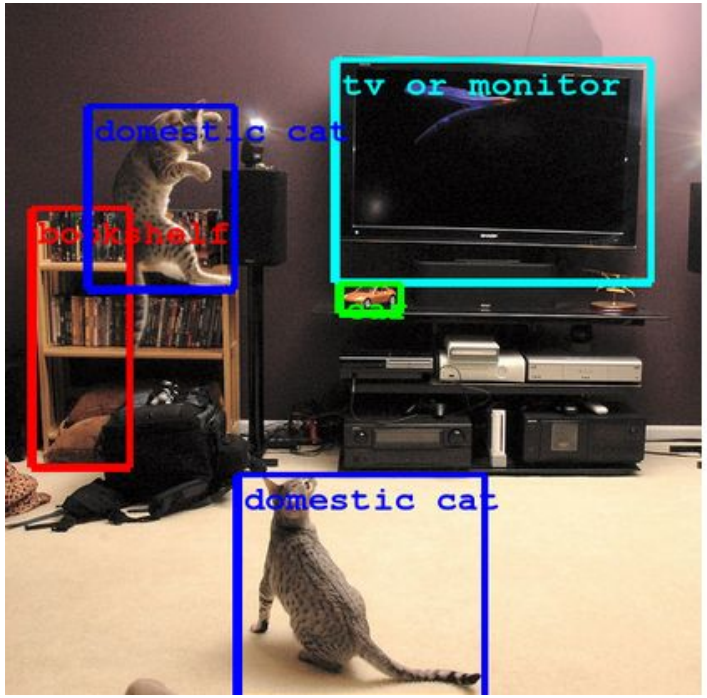
(credits: XKCD)

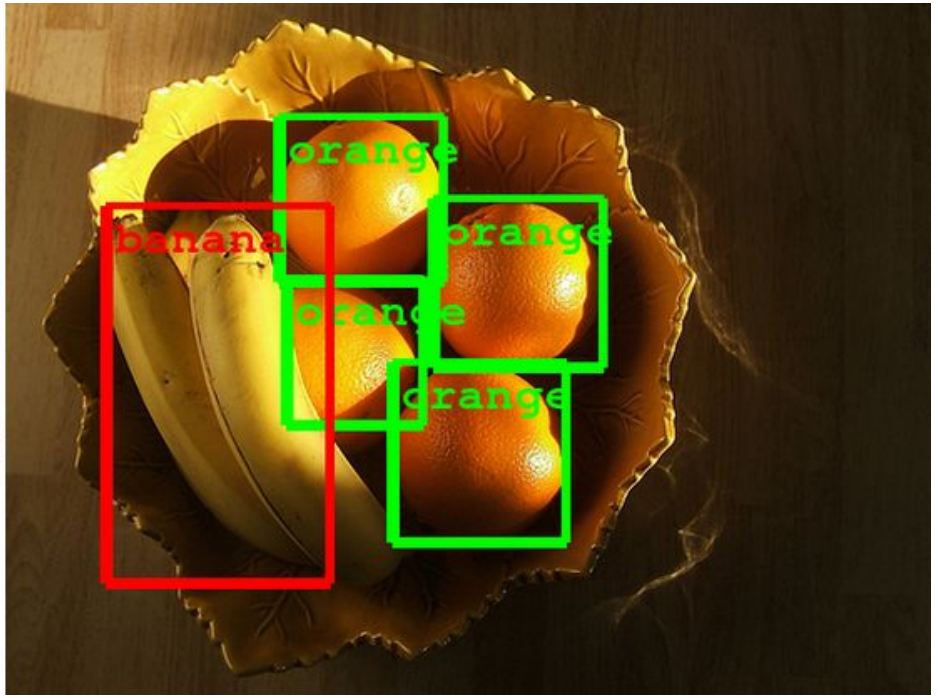


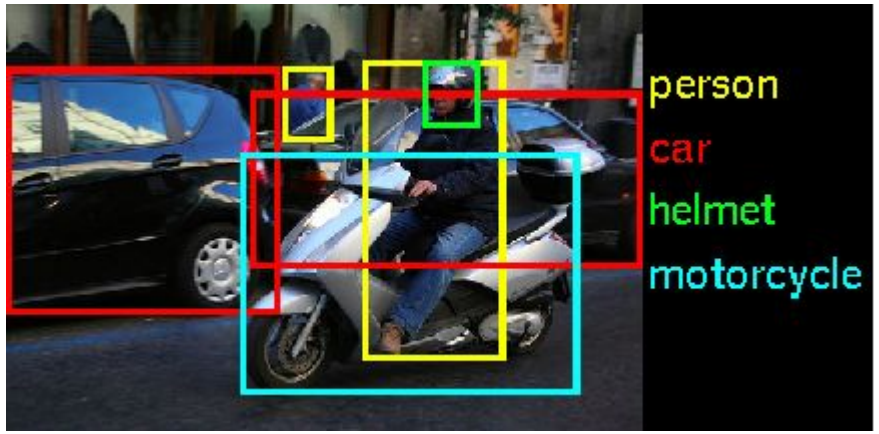


(credits: Google Research)


airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck







(credits: Google Research)


Google


Hi. I'm Cortana.
Ask me a question!


Phone
Safari
What can I help you with?

Machine Learning

Machine learning is a method of data analysis that automates analytical model building. Using algorithms that iteratively learn from data, machine learning allows computers to find hidden insights without being explicitly programmed where to look.

Deep Learning

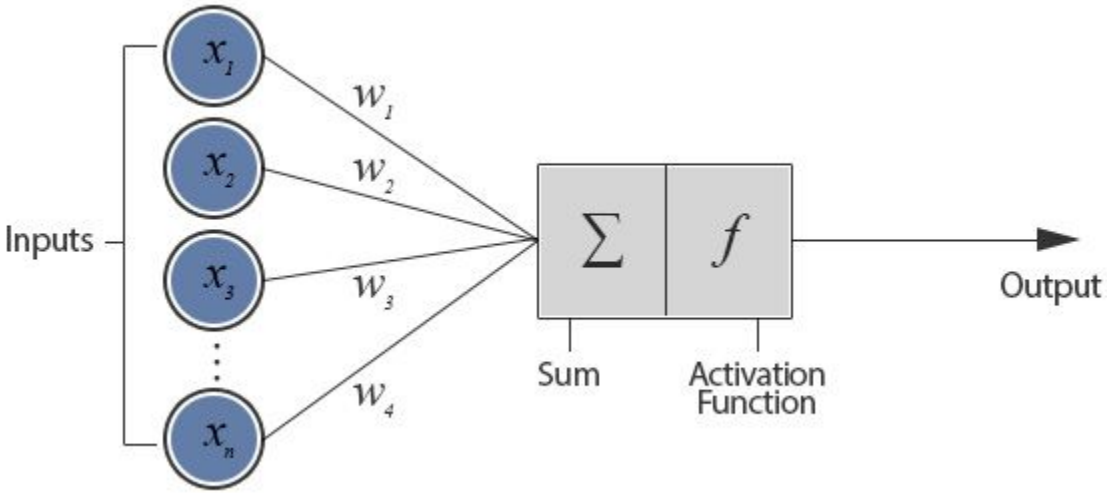
- A family of Machine Learning algorithms
- It has been observed that they perform better than standard ML algorithms (regression, SVM, …) for problems such as image recognition, audio recognition, NLP

Agenda

- Introduction
- A little bit of theory
- Distributed Neural Network with Apache Spark
- Example & Code

Perceptrons

- Invented in 1957 at the Cornell Aeronautical Laboratory by Frank Rosenblatt
- An algorithm for supervised learning of binary classifiers
- Takes *n* binary input and produces a single binary output
- For each input x_i there is a weight w_i that determines how relevant the input x_i is to the output
- *b* is the bias and defines the activation threshold

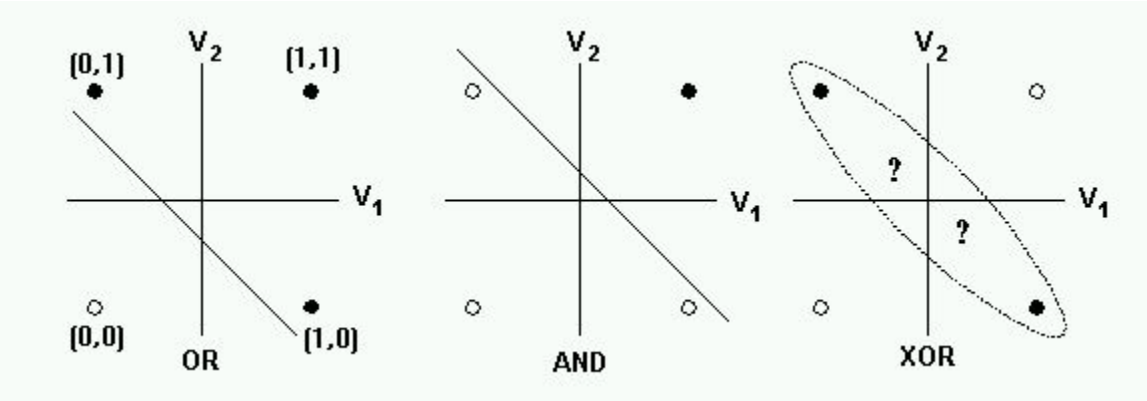


$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

(credits for image: The Project Spot)

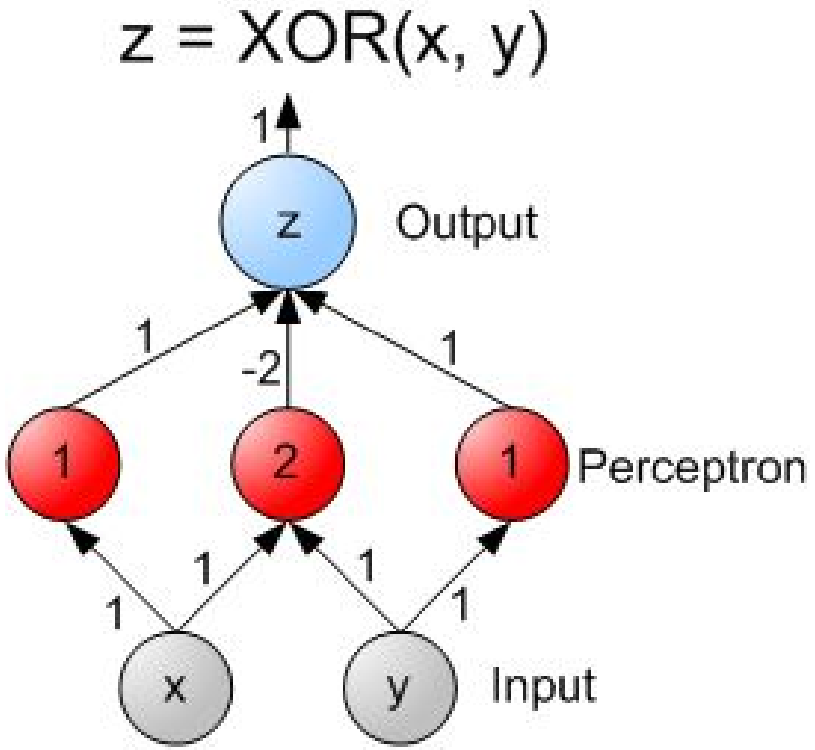
Perceptrons

- Problem: a single perceptron can learn only linearly separable problems!
- XOR is not linearly separable



Towards Neural Networks: Perceptron layers

Solution: introduce more layers of perceptrons!



This 2-layer perceptron network actually solves XOR!

Supervised learning

- Input: a training dataset with (manually) labeled data
- Output: a model that can predict new examples based on the existing observations
- **Cost function**: a function that represents the error, i.e. how much the prediction of your NN approximates the expected output of the classification
- Many supervised learning algorithms are trained by minimizing the cost function

Training Neural Networks

We want the neural network that minimizes the cost (or error) function

$$C(w,b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$$

where $a = a(w,b,x)$ is the output of the NN when x is the ouput.

The challenge of training NNs is finding an algorithm for minimizing the cost function that is computationally efficient.

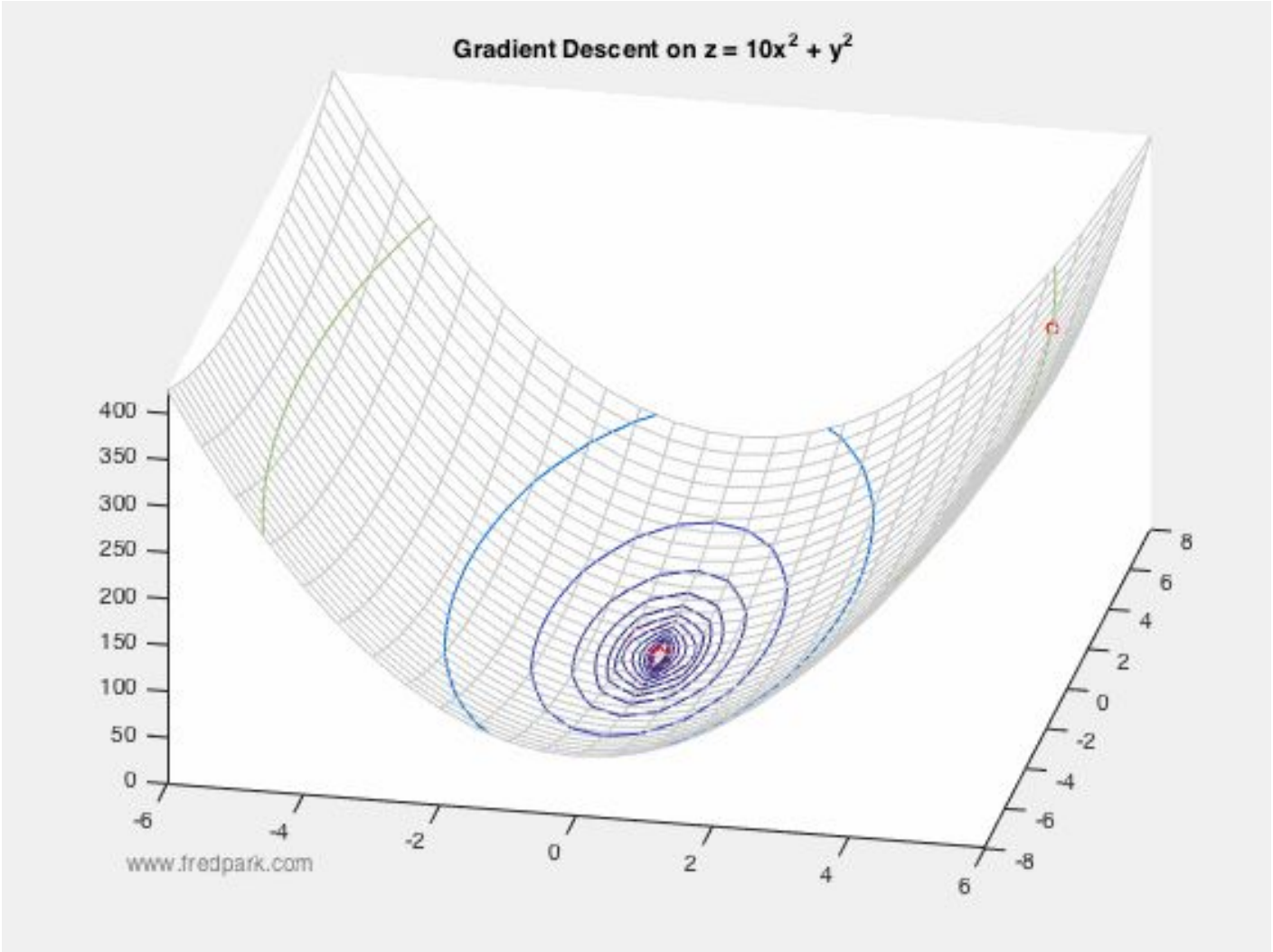
Training Neural Networks: Gradient Descent

An iterative, optimization algorithm for finding the minimum of a function *C(w)*

- Initialize *w*
- Calculate the gradient $\nabla_w C$
- Update: $w \rightarrow w - \eta \nabla_w C(w)$
- Stop when minimum is reached

η is the learning rate of the gradient descent

Training Neural Networks: Gradient Descent



Training Neural Networks: Gradient Descent

- Problem: we have to calculate the gradients for the whole dataset for just one update step!
- This is unfeasible for high dimensionality problems, because there would be too many partial derivatives to compute
- Complex Neural Networks can have billion of parameters!

Training Neural Networks: SGD

We introduce an improved version: the mini-batch Stochastic Gradient Descent.

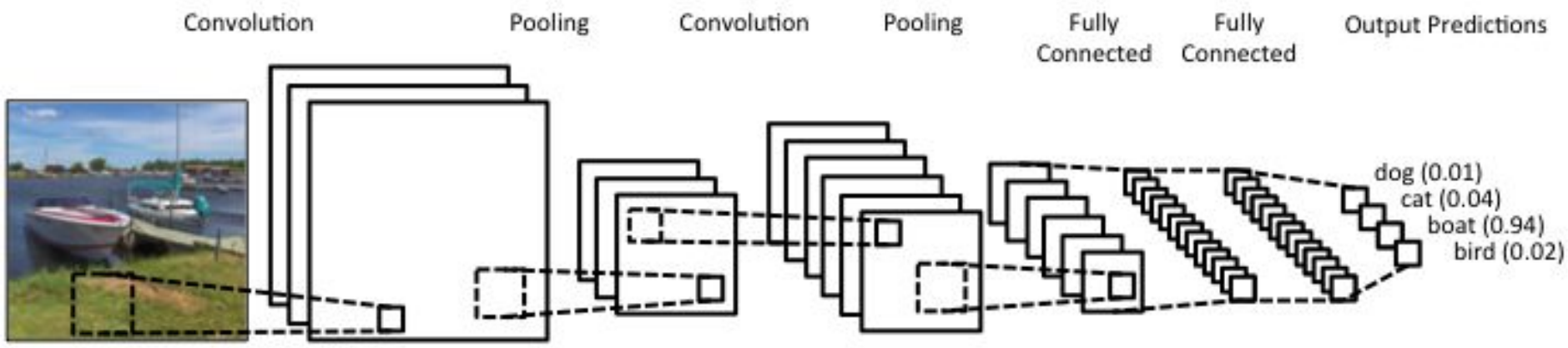
- Initialize x
- For each epoch repeat the following steps:
- Randomly shuffle the training dataset
- For each sample of size n of the training dataset, calculate the gradient w.r.t. the mini batch and perform the update

Training Neural Networks: SGD

- The mini-batch SGD is the algorithm of choice for training most NNs, because it is significantly faster than the vanilla gradient descent.
- It is also more memory efficient! (does not require the whole gradient vector to be in memory)

Convolutional neural networks

- You can combine several layers to obtain complex neural networks
- Convolution layers work on windows of the image instead of pixels and help to detect features
- Pooling layers offer translation invariance



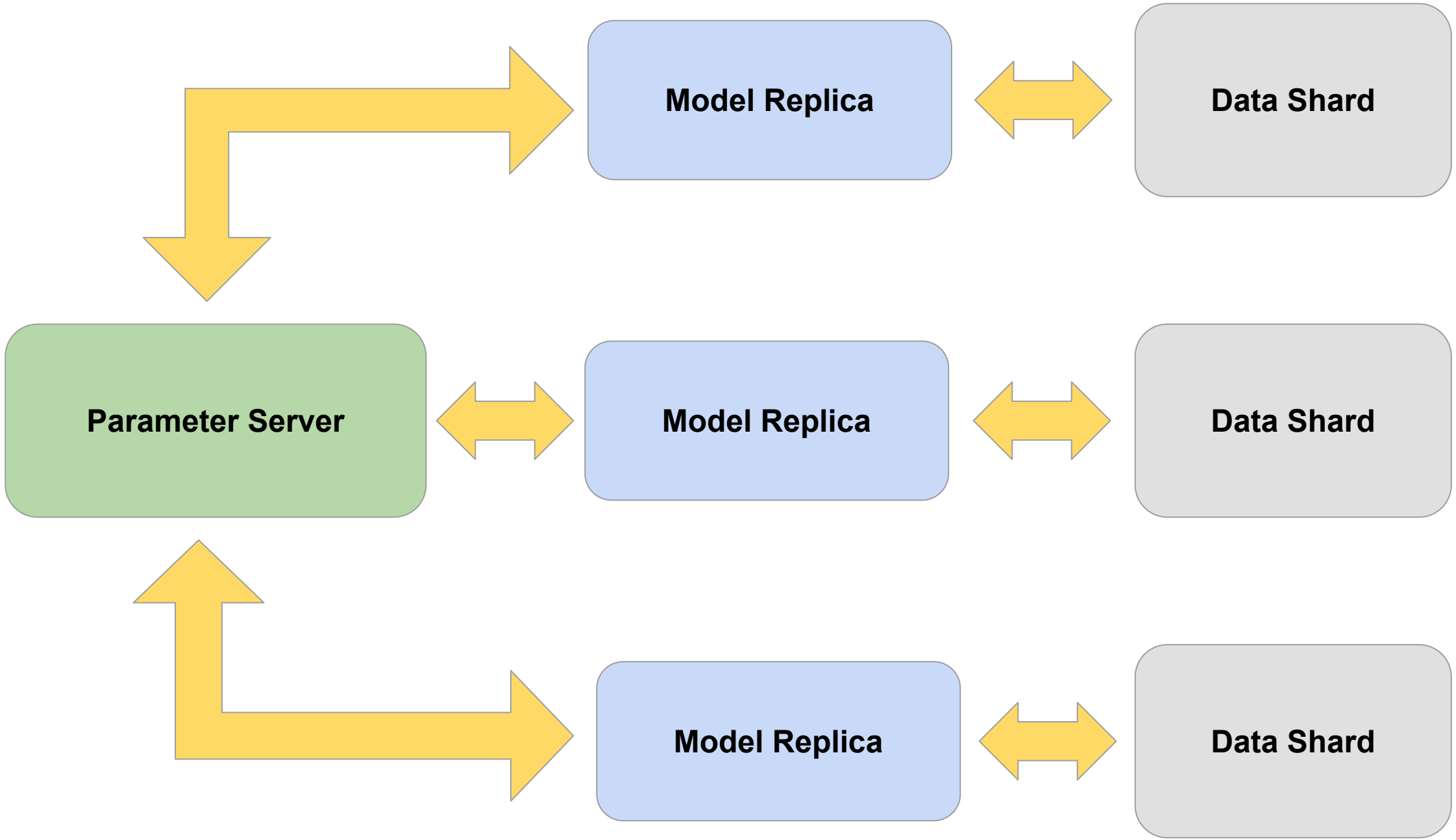
(credits: Clarifai)

Issues with single-node training of NNs

- Increasing the number of training examples, model parameters, or both, can drastically improve model accuracy
- Yet the training speed-up is small when the model does not fit in GPU memory
- Hence, the size of the data or parameters is tipically reduced
- Wouldn’t it be wonderful to use large-scale clusters of machines to distribute training in deep neural networks?

Distributed Neural Networks

- In 2012, Google published *Large Scale Distributed Deep Networks*
- In that paper, they described the main concepts behind their Deep Learning infrastructure:
  - Downpour SGD
  - Model Replicas on Distributed (Partitioned) Data


Model Replica
Data Shard
Parameter Server
Model Replica
Data Shard
Model Replica
Data Shard

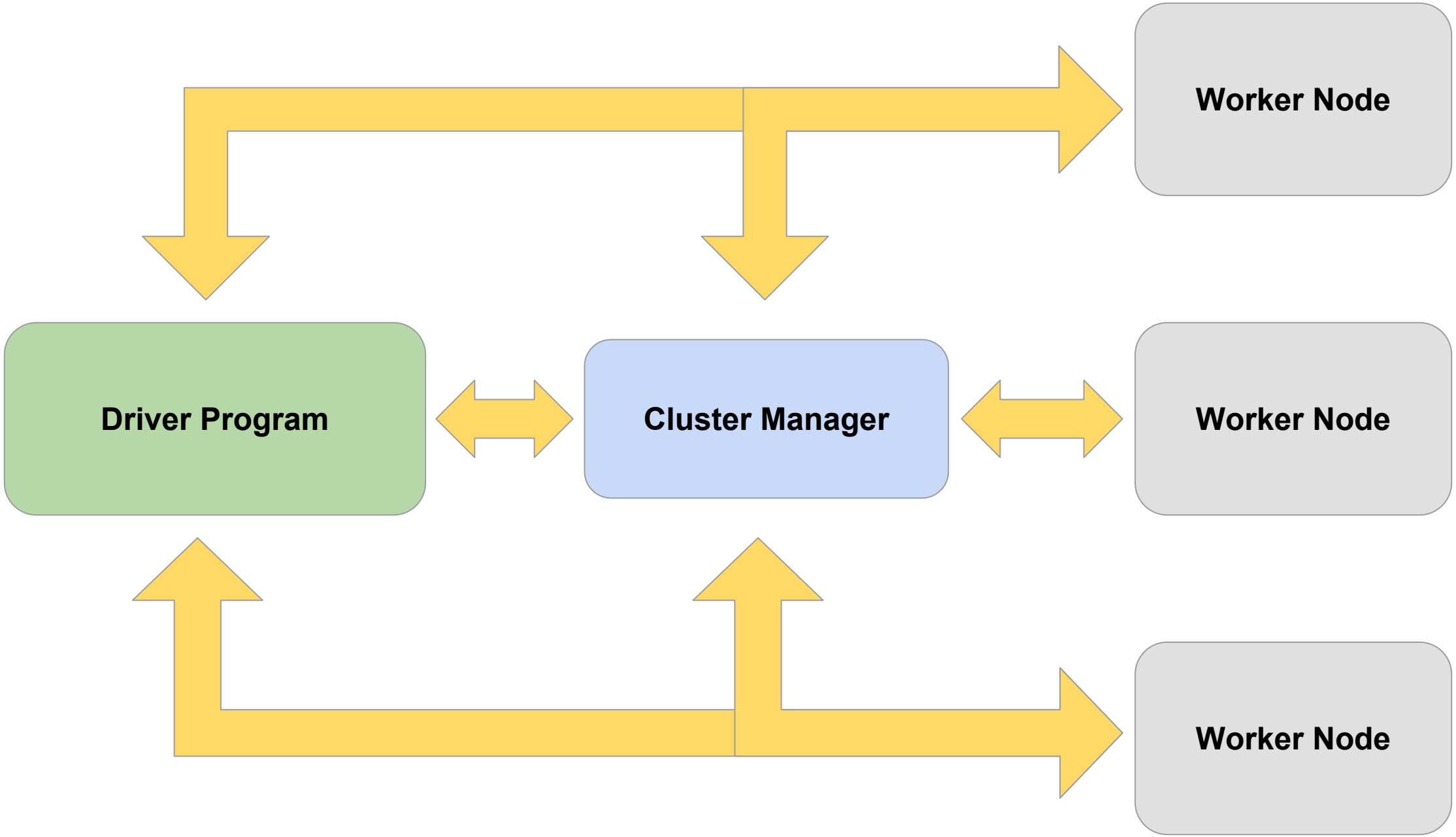
How can we easily distribute computation?

What is Apache Spark?

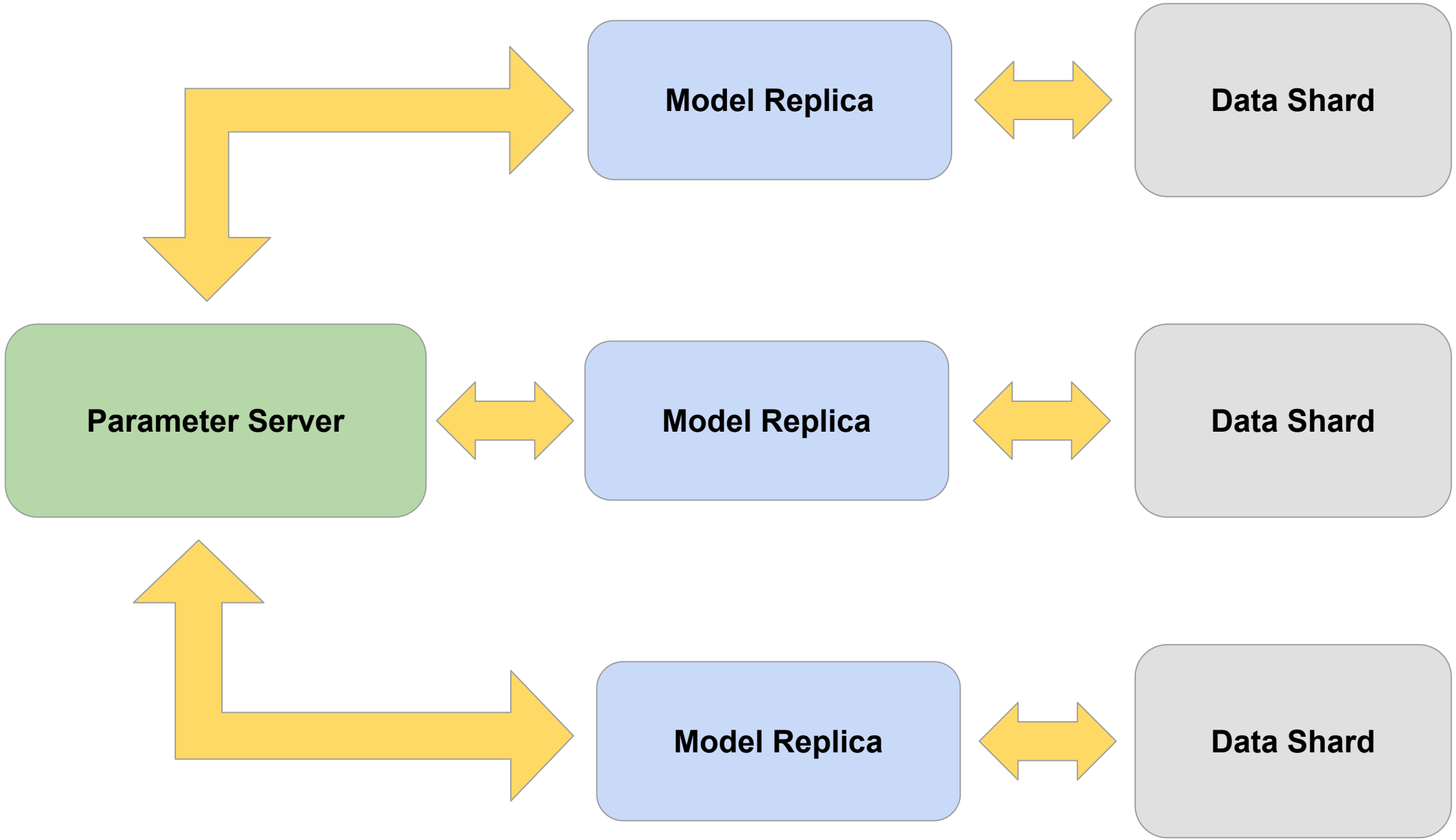
A fast and general engine for large-scale distributed data processing, with built-in modules for streaming, sql, machine learning and graph processing

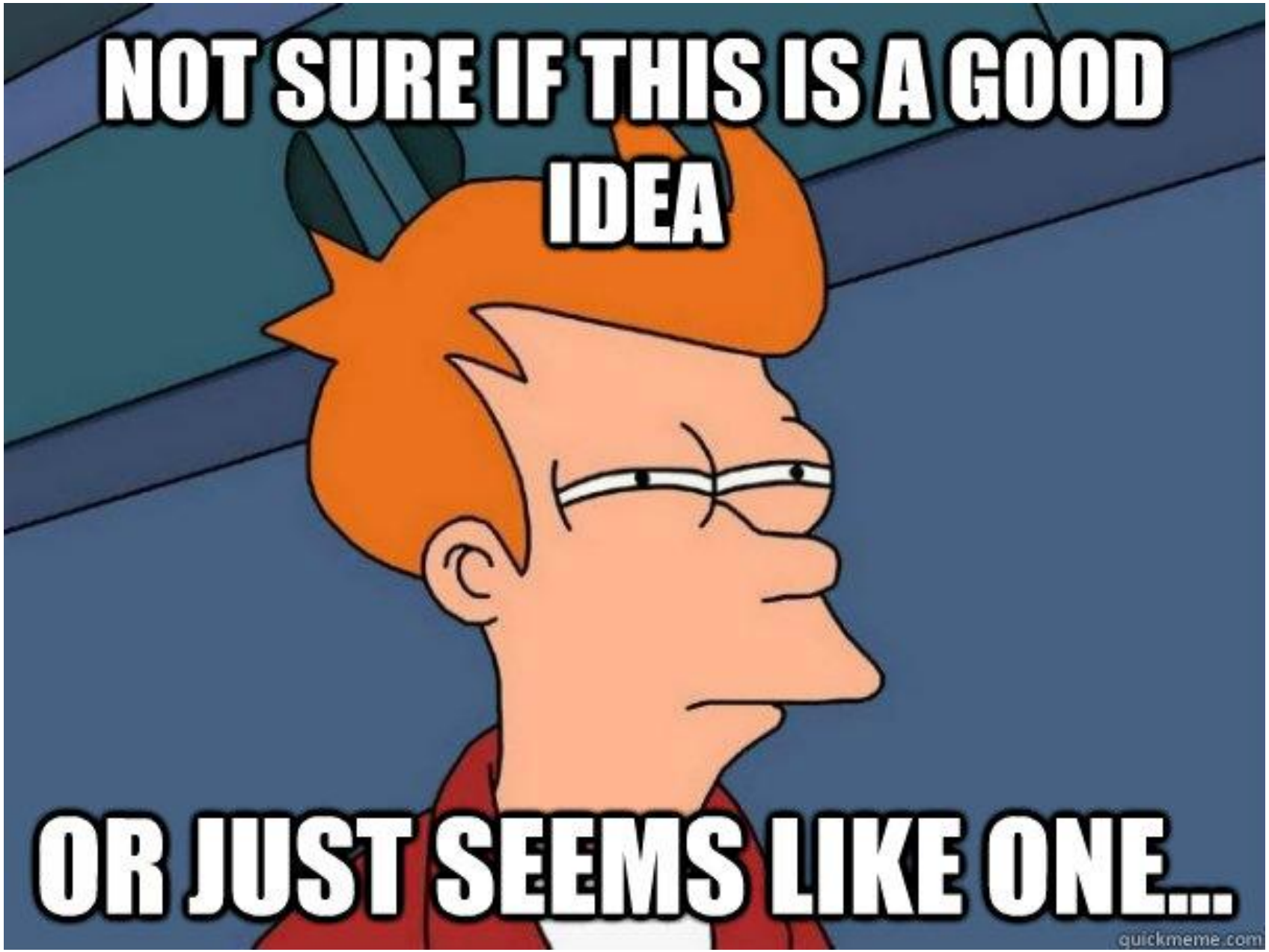
Advantages of using Spark for Deep Learning

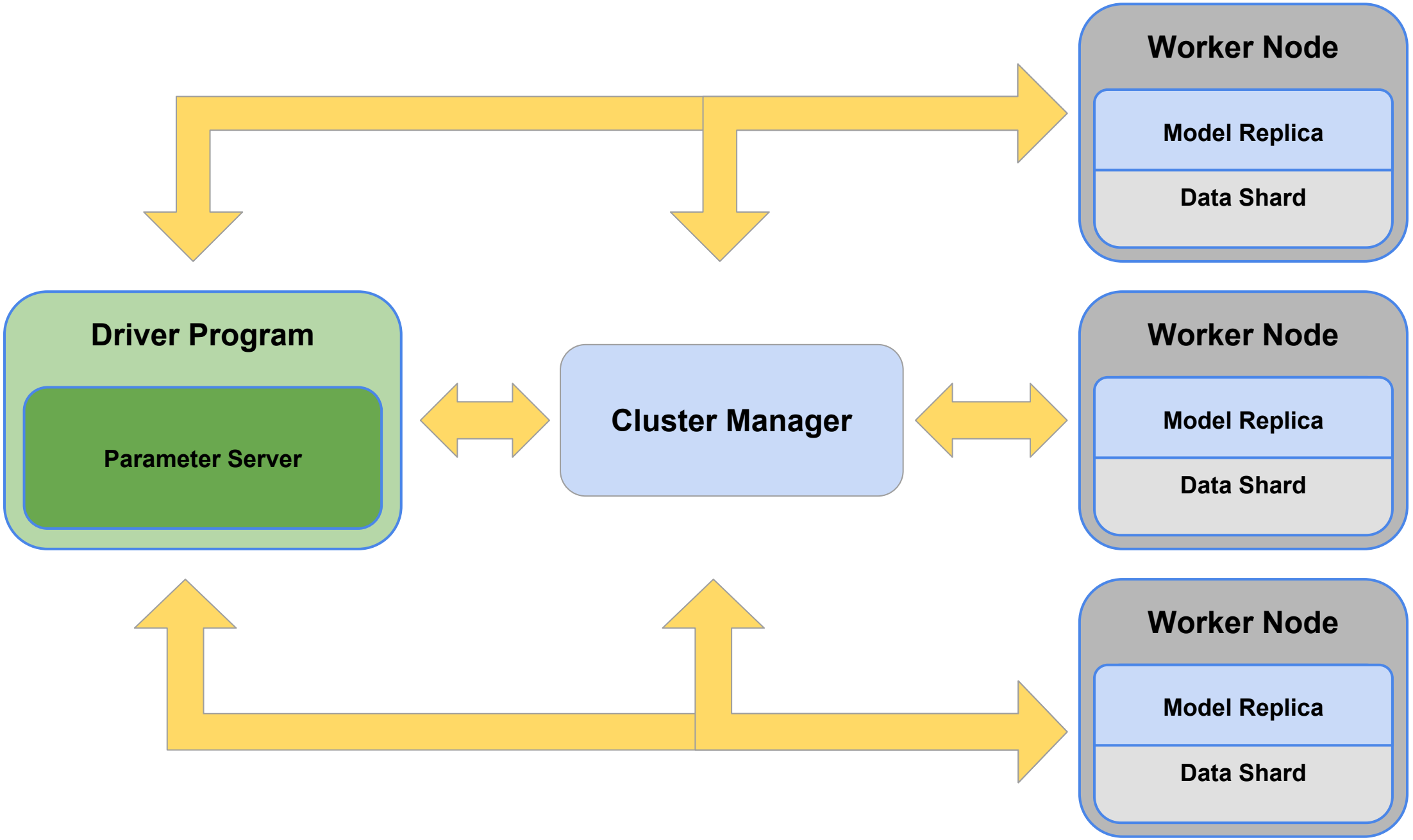
- Distributed computing out of the box
- Take advantage of the Hadoop ecosystem (Hive, HDFS, ...)
- Take advantage of the Spark ecosystem (Spark Streaming, Spark SQL, ...)


Worker Node
Driver Program
Cluster Manager
Worker Node
Worker Node


NOT SURE IF DÉJÀ VU
OR IF THERE'S A GLITCH IN THE MATRIX
memegenerator.net


Model Replica
Data Shard
Parameter Server
Model Replica
Data Shard
Model Replica
Data Shard


NOT SURE IF THIS IS A GOOD IDEA
OR JUST SEEMS LIKE ONE...
quickmeme.com

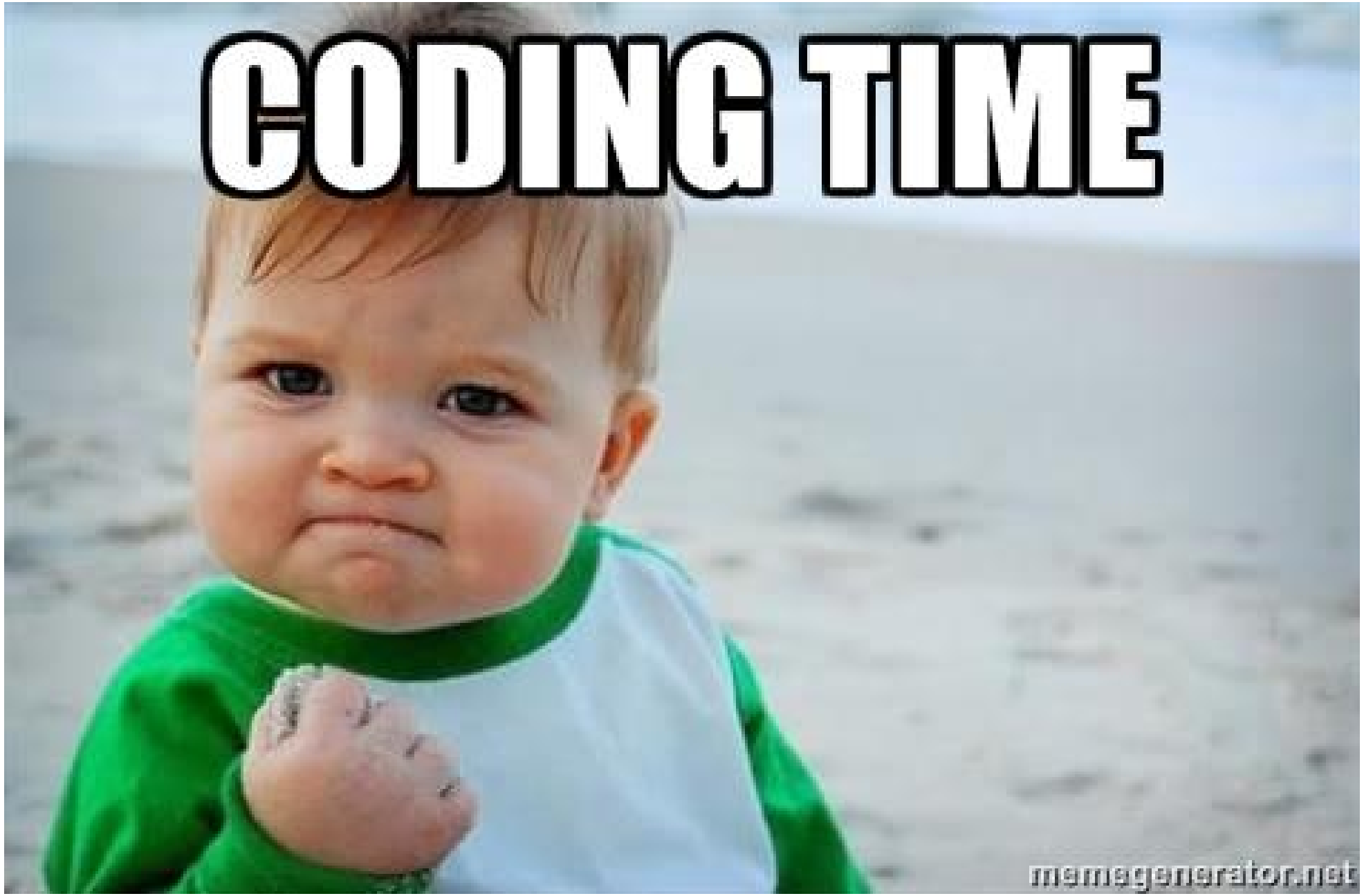

Worker Node
Model Replica
Data Shard
Driver Program
Parameter Server
Cluster Manager
Worker Node
Model Replica
Data Shard
Worker Node
Model Replica
Data Shard


WHAT IF I TOLD YOU
THAT SOMEONE ALREADY DID IT?
imgflip.com

Deeplearning4j

Works as a job within Spark:

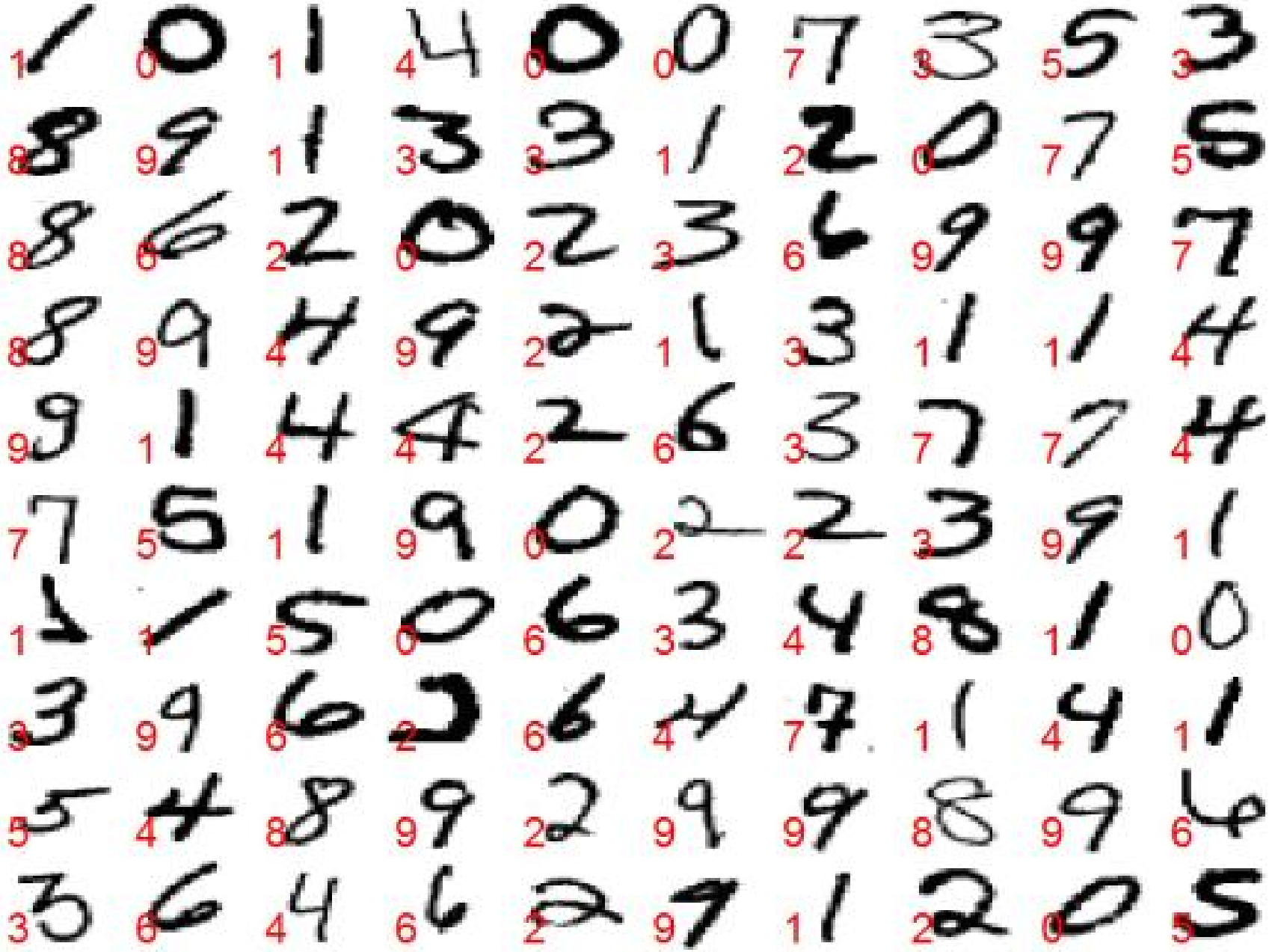
1. Shards of the input dataset are distributed over all cores
2. Workers process data synchronously in parallel
3. A model is trained on each shard of the input dataset
4. Workers send the transformed parameters of their models back to the master
5. The master averages the parameters
6. The parameters are used to update the model on each worker’s core
7. When the error ceases to shrink, the Spark job ends


CODING TIME
memegenerator.net

A practical example: handwriting recognition (MNIST)

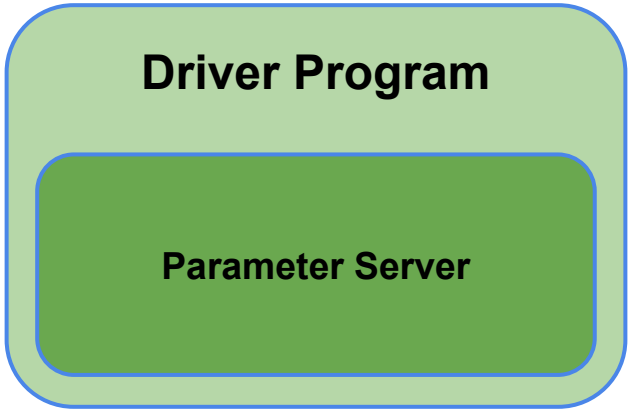
What is MNIST:

- A dataset of handwritten digits
- A subset of a larger set available from NIST (National Institute of Standards and Technology)
- The digits have been size-normalized and centered in a fixed-size image
- Has a training set of 60,000 examples,
- Has a test set of 10,000 examples


1 0 1 4 0 0 7 3 5 3
8 9 1 3 3 1 2 0 7 5
8 6 2 0 2 3 6 9 9 7
8 9 4 9 2 1 3 1 1 4
9 1 4 4 2 6 3 7 7 4
7 5 1 9 0 2 2 3 9 1
1 1 5 0 6 3 4 8 1 0
3 9 6 2 6 4 7 1 4 1
5 4 8 9 2 9 9 8 9 6
3 6 4 6 2 9 1 2 0 5

Create the Spark Context

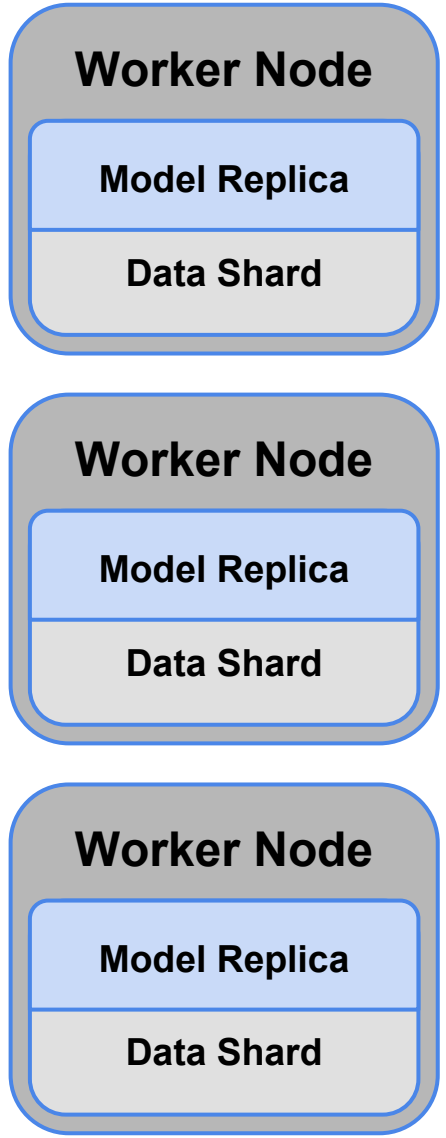
```
val sc = new SparkContext(new SparkConf()
                    .setMaster(masterNode)
                    .setAppName("Spark-Deep-Learning")
                    .set(SparkDl4jMultiLayer.AVERAGE_EACH_ITERATION, “true”))
```



Load MNIST dataset into Apache Spark

```
// split data
val (trainingSet, testSet) = new MnistDataSetIterator(1, numSamples,true)
                                                    .splitDatasetAt(numForTraining)


// distribute training data over the cluster
val sparkTrainingData = sc.parallelize(trainingSet).cache()
```

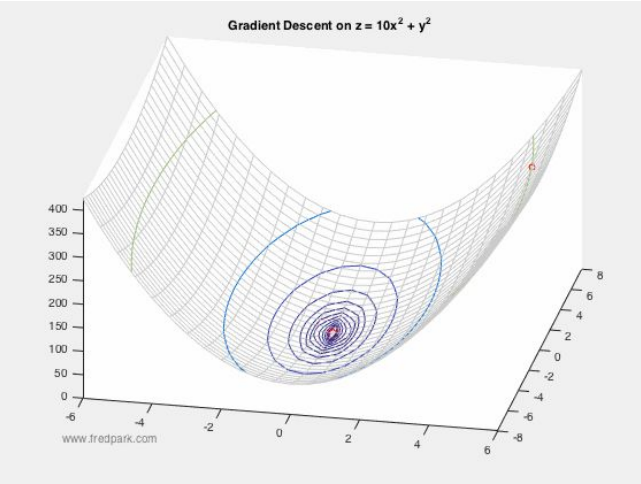


Prepare the Neural Network

```
// prepare the neural network
val neuralNetwork = prepareNeuralNetwork(height, width, numChannels)


// create Spark multi layer network
val sparkNeuralNetwork = new SparkDl4jMultiLayer(sc, neuralNetwork)
```

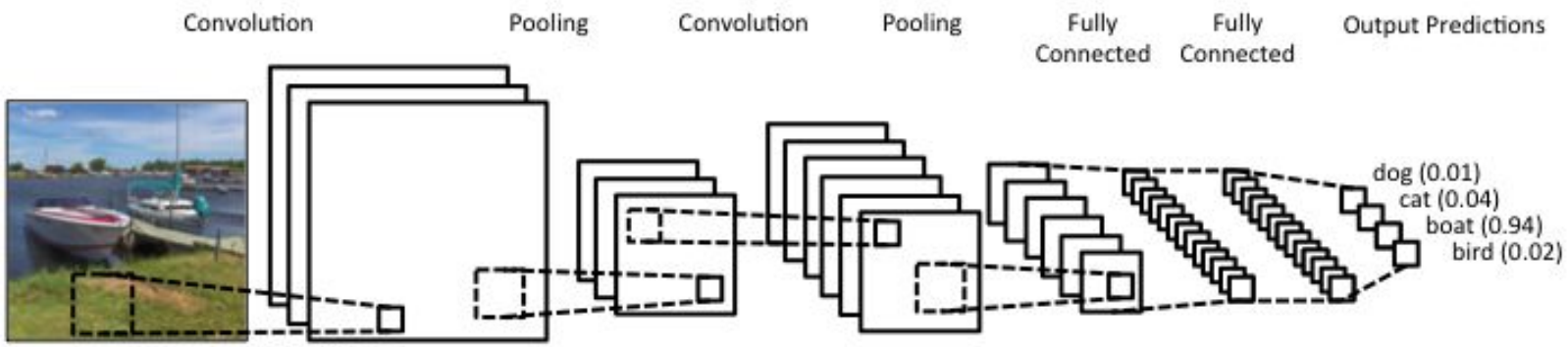
The Neural Network MultiLayer




```
new NeuralNetConfiguration.Builder()
        .seed(seed) // Random number generator seed.
        .iterations(iterations) // Number of optimization iterations.
        .regularization(true) // Whether to use regularization
        .l2(0.0005) // L2 regularization coefficient.
        .learningRate(0.1) // Learning rate.
        .optimizationAlgo(
            OptimizationAlgorithm.STOCHASTIC_GRADIENT_DESCENT
        ).updater(Updater.ADAGRAD) // Gradient updater.
        .list(6) // Number of layers (not including the input layer).
```

Each layer will be similar to this one

```
new ConvolutionLayer.Builder(5, 5)
        .nIn(numChannels)          // Number of input neurons (channels).
        .stride(1, 1)
        .nOut(20)                  // Number of output neurons.
        .weightInit(WeightInit.XAVIER) // Weight initialization scheme.
        .activation("relu")        // Activation function: rectified linear.
        .build()
```



Train the Convolutional Neural Network

```
(0 to epochs).foreach { i =>
    val network = sparkNeuralNetwork
                        .fitDataSet(sparkTrainingData, numCores * batchSize)
 }
```

$$C(w,b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$$

What can we do now? Evaluate our DNN

```
network.evaluateOn(testSet).stats
```

```
Examples labeled as 0 classified by model as 0: 978 times
...
Examples labeled as 4 classified by model as 9: 27 times
...
Examples labeled as 7 classified by model as 9: 26 times
...
Examples labeled as 9 classified by model as 9: 884 times


====================Scores====================
 Accuracy:  0.9565
 Precision: 0.9563
 Recall:    0.9563
 F1 Score:  0.9563

Produced with 100 epochs.
```

http://bit.ly/1Yq8Tzu