Distributed Deep Learning Inference

using Apache MXNet* and Apache Spark

Naveen Swamy

Amazon AI

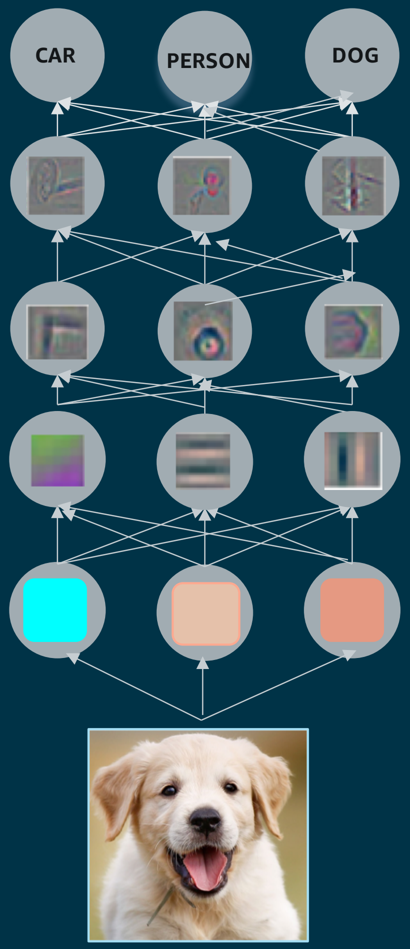
* APACHE INCUBATOR

Outline

- Review of Deep Learning
- Apache MXNet Framework
- Distributed Inference using MXNet and Spark

Deep Learning

- Originally inspired by our biological neural systems.
- A System that learns important features from experience.
- Layers of Neurons learning concepts.
- Deep learning != deep understanding



Output
(object identity)

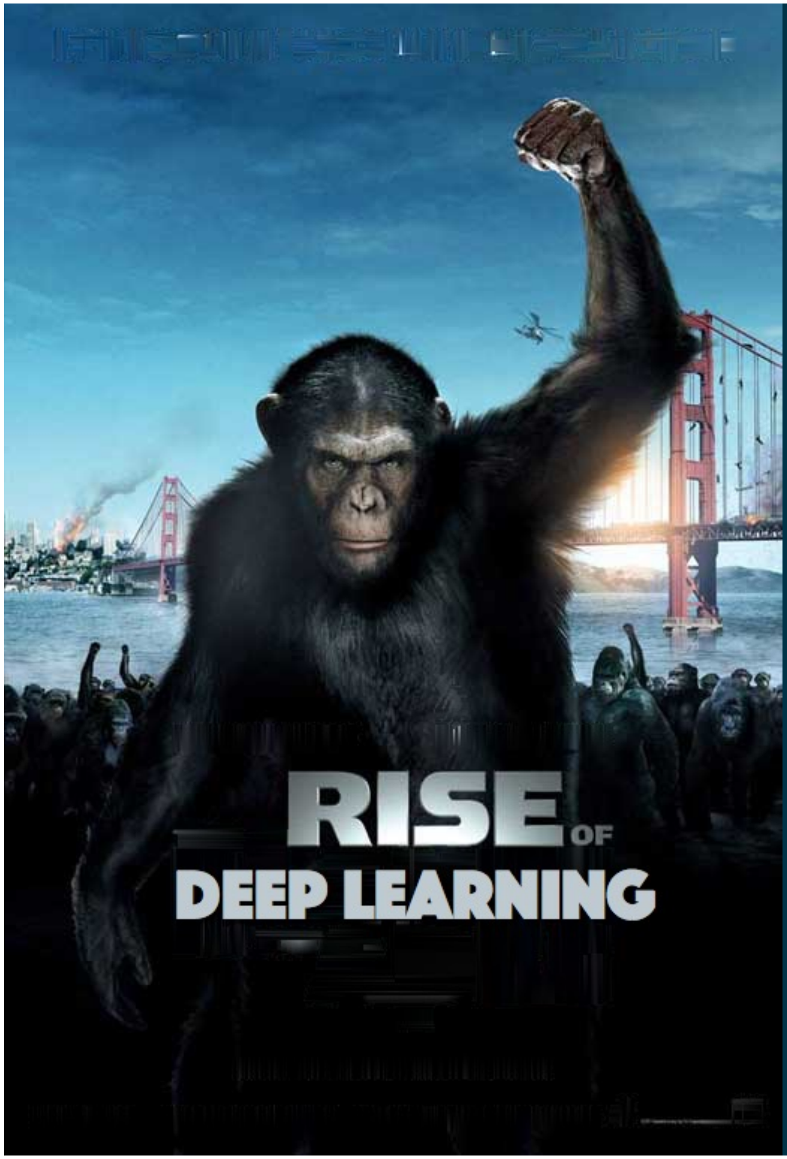
3rd hidden layer
(object parts)

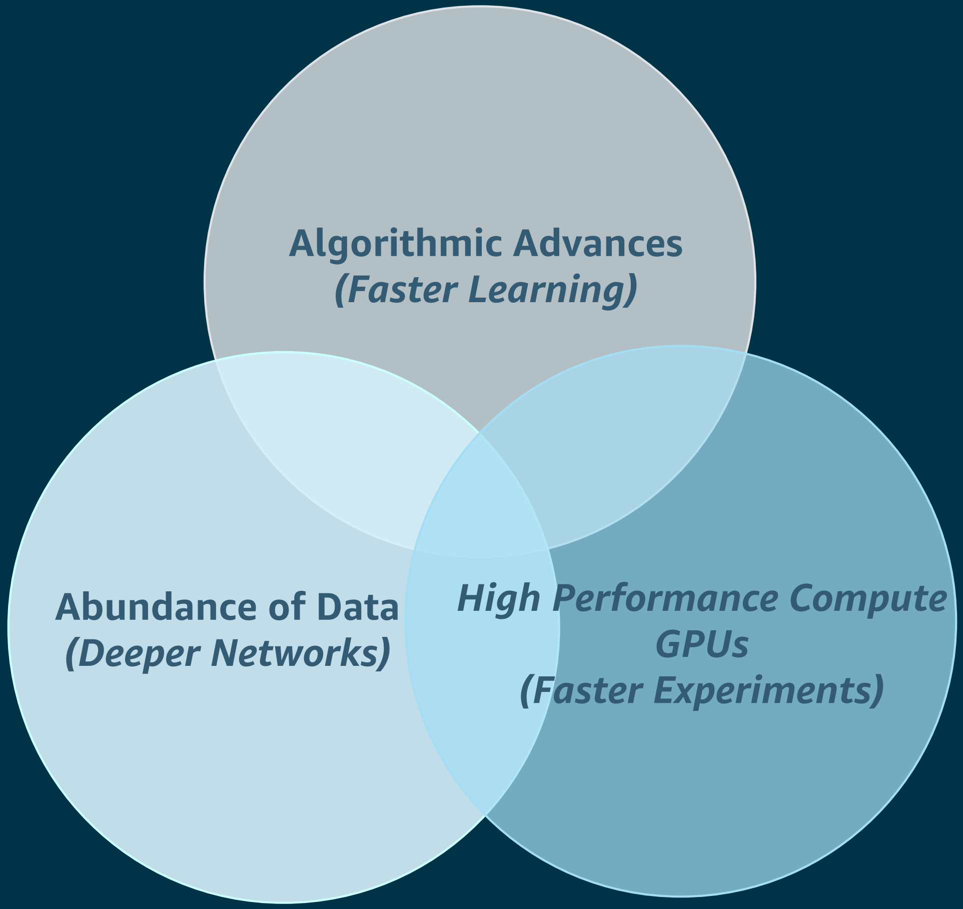
2nd hidden layer
(corners & contours)

1st hidden layer
(edges)

Input layer
(Raw pixels)

Credit: Ian Goodfellow etal., Deep Learning Book





Algorithmic Advances
(Faster Learning)

Abundance of Data
(Deeper Networks)

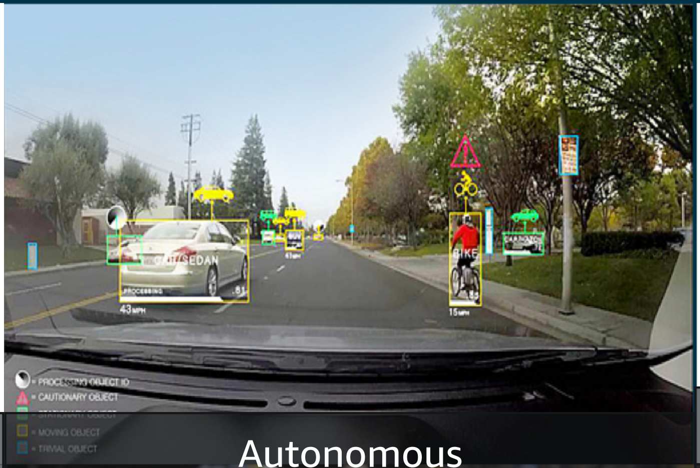
High Performance Compute
GPUs
(Faster Experiments)

Bigger and Better Models = Better AI Products

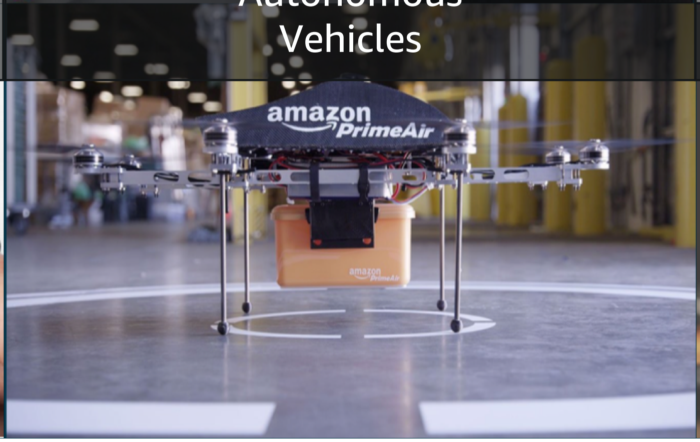
Why does Deep Learning matter?

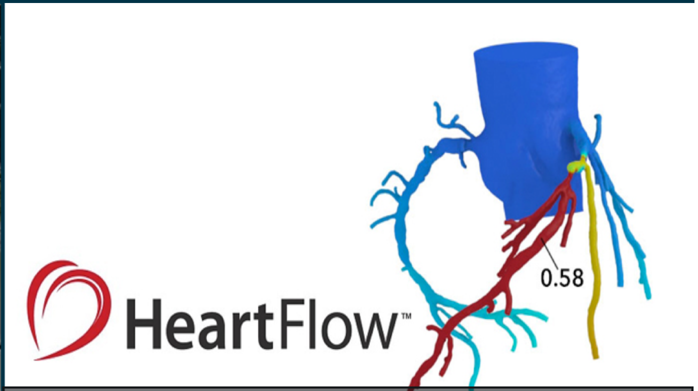
amazon alexa
"Alexa, who was President when Barack Obama was nine?"
"Alexa, how's my commute?"
"Alexa, what's the weather?"
"Alexa, did the 49ers win?"

Personal Assistants



Autonomous Vehicles

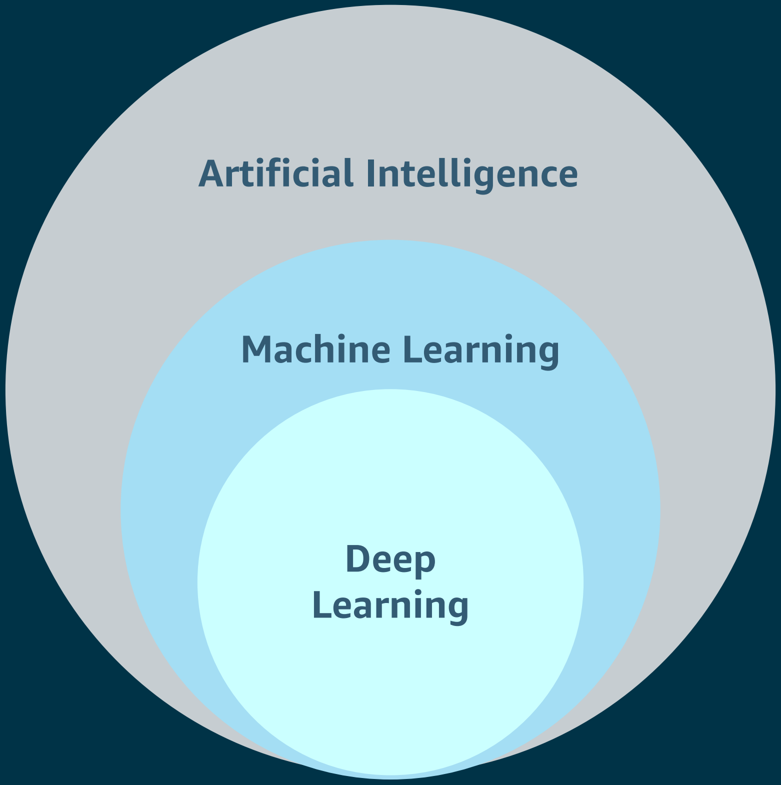




Health care

Solve Intelligence ???

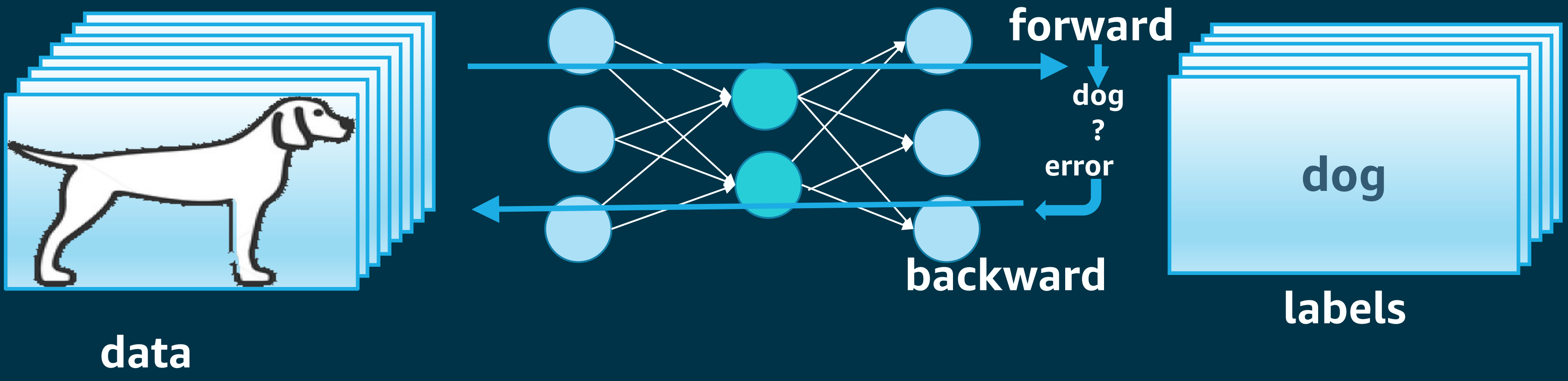
Deep Learning & AI, Limitations



DL Limitations:

- Requires lots of data and compute power.
- Cannot detect Inherent bias in data - Transparency.
- Uninterpretable Results.

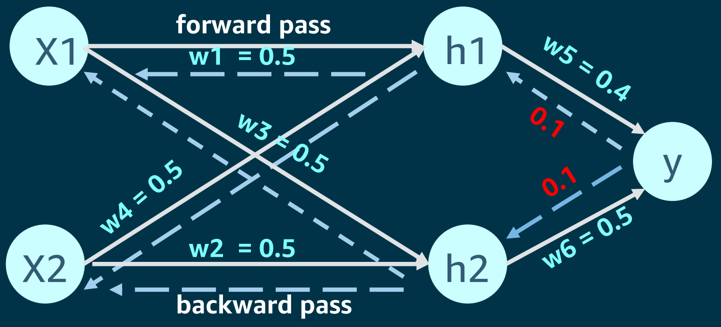
Deep Learning Training



- Pass data through the network – forward pass

- Define an objective – Loss function

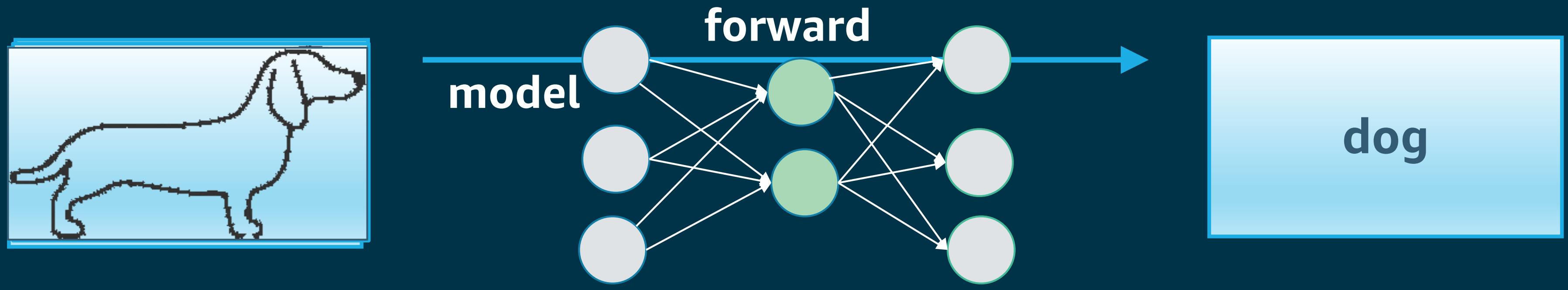
- Send the error back – backward pass



y = 1.0
y` = 0.9
loss = y – y`
l = 0.1

Model: Output of Training a neural network

Deep Learning Inference



model

forward

dog

- **Real time Inference:** Tasks that require immediate result.

- **Batch Inference:** Tasks where you need to run on a large data sets.
  - Pre-computations are necessary - Recommender Systems.
  - Backfilling with state-of-the art models.
  - Testing new models on historic data.

Types of Learning

- **Supervised Learning** – Uses labeled training data learning to associate input data to output.

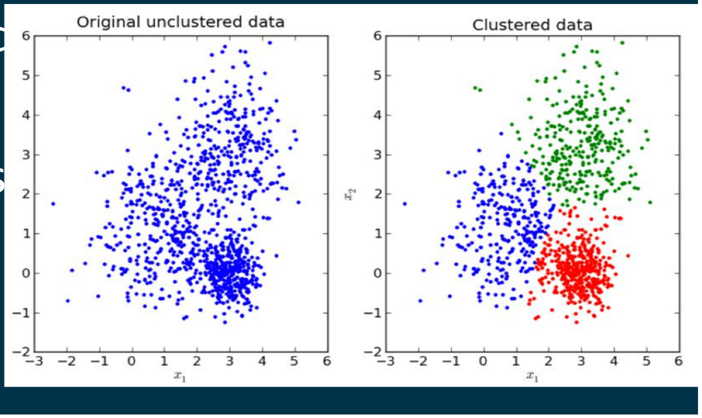
Example: Image classification, Speech Recognition, Machine translation

- **Unsupervised Learning -** Learns patterns from Unlabeled data.

Example: Clustering, Association discovery.

- **Active Learning** – Semi-supervised, human in the midd

- **Reinforcement Learning** – learn from environment, us feedback.



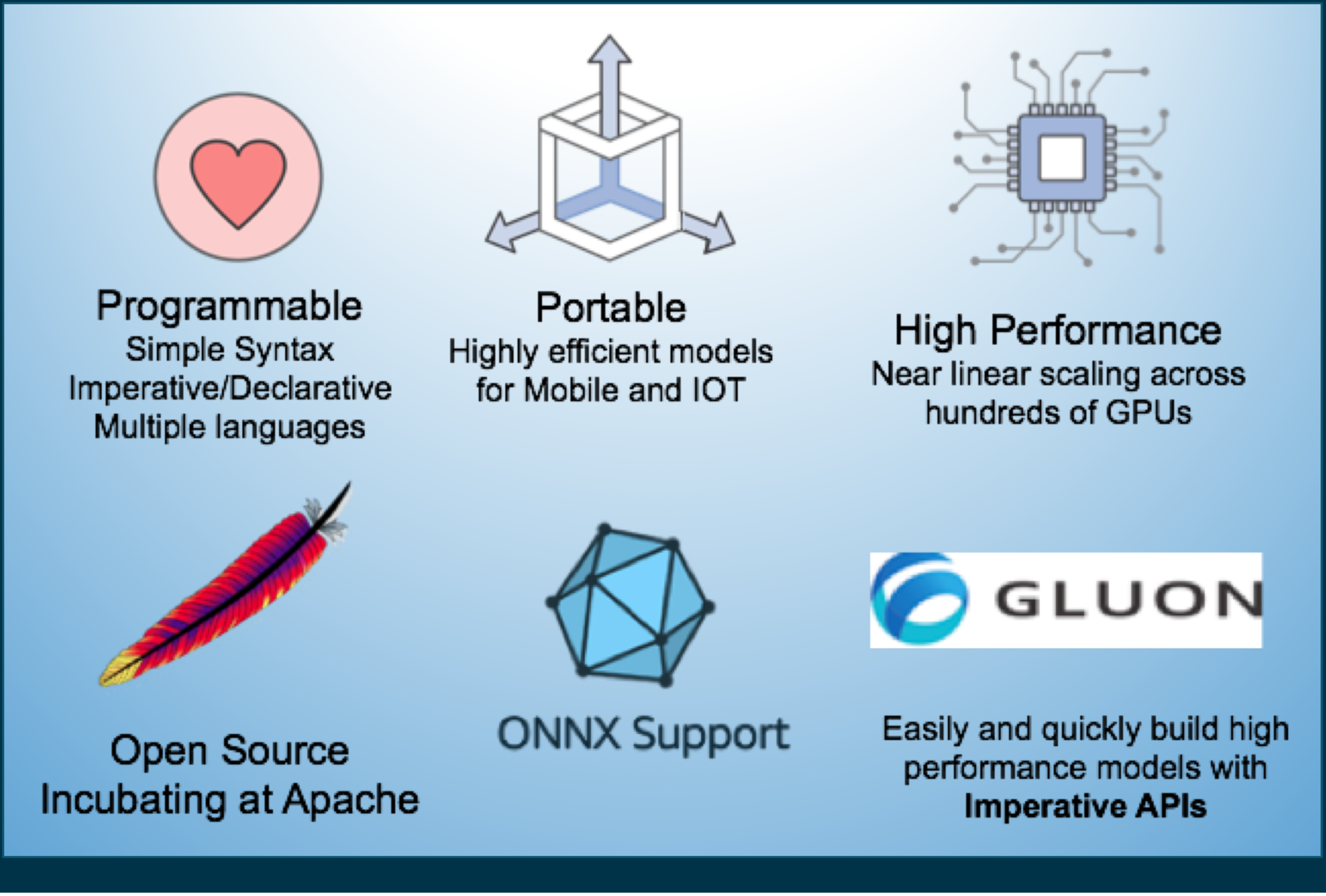
Outline

- Review of Deep Learning

- Apache MXNet Framework

- Distributed Inference using MXNet and Spark

Why MXNet



Programmable
Simple Syntax
Imperative/Declarative
Multiple languages

Portable
Highly efficient models
for Mobile and IOT

High Performance
Near linear scaling across
hundreds of GPUs

Open Source
Incubating at Apache

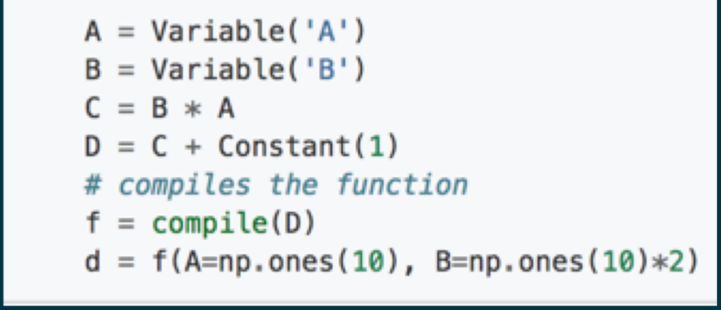
ONNX Support

GLUON

Easily and quickly build high
performance models with
Imperative APIs

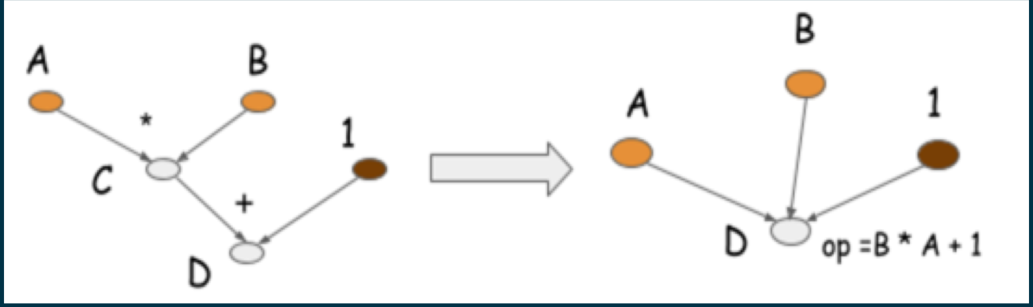
MXNet – NDArray & Symbol

- **NDArray–** Imperative Tensor Operations that work on both CPU and GPUs.
- **Symbol APIs** – similar to NDArray but adopts declarative programming for optimization.



```
A = Variable('A')
B = Variable('B')
C = B * A
D = C + Constant(1)
# compiles the function
f = compile(D)
d = f(A=np.ones(10), B=np.ones(10)*2)
```

Symbolic Program

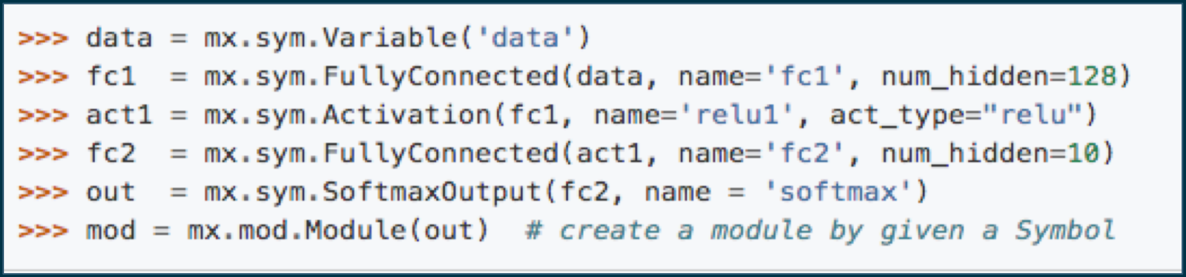


Computation Graph

MXNet - Module

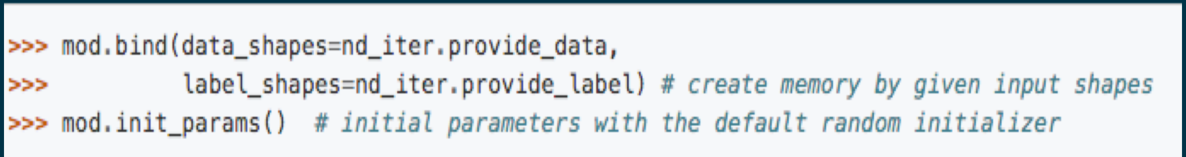
High level APIs to work with Symbol

1) Create Graph



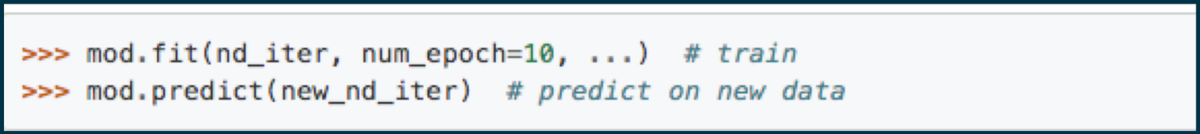
```
>>> data = mx.sym.Variable('data')
>>> fc1  = mx.sym.FullyConnected(data, name='fc1', num_hidden=128)
>>> act1 = mx.sym.Activation(fc1, name='relu1', act_type="relu")
>>> fc2  = mx.sym.FullyConnected(act1, name='fc2', num_hidden=10)
>>> out  = mx.sym.SoftmaxOutput(fc2, name = 'softmax')
>>> mod = mx.mod.Module(out)  # create a module by given a Symbol
```

2) Bind



```
>>> mod.bind(data_shapes=nd_iter.provide_data,
>>>          label_shapes=nd_iter.provide_label) # create memory by given input shapes
>>> mod.init_params()  # initial parameters with the default random initializer
```

3) Pass data



```
>>> mod.fit(nd_iter, num_epoch=10, ...)  # train
>>> mod.predict(new_nd_iter)  # predict on new data
```

Outline

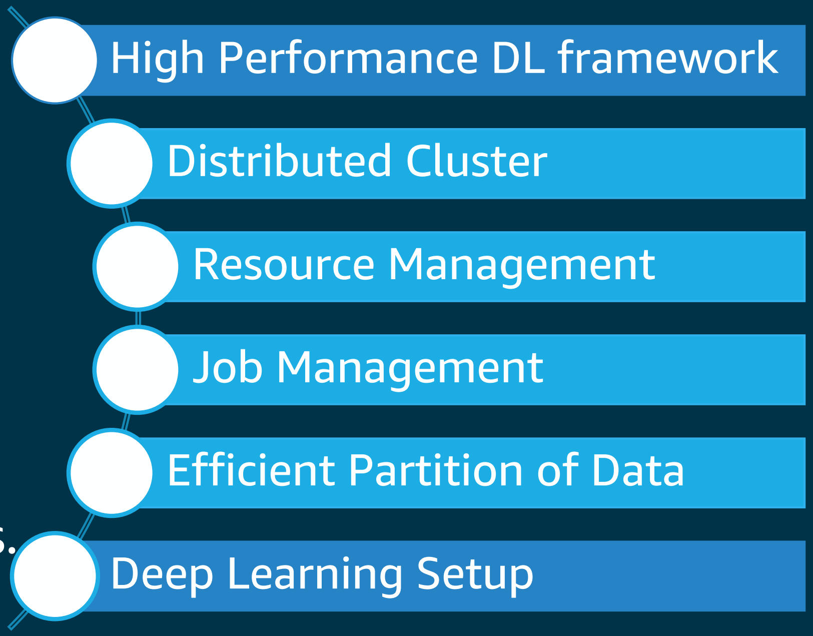
- Review of Deep Learning
- Apache MXNet Framework
- Distributed Inference using MXNet and Spark

Distributed Inference Challenges

- Similar to large scale data processing systems

Apache Spark:

- Multiple Cluster Managers
- Works well with MXNet.
- Integrates with Hadoop & big data tools.



High Performance DL framework

Distributed Cluster

Resource Management

Job Management

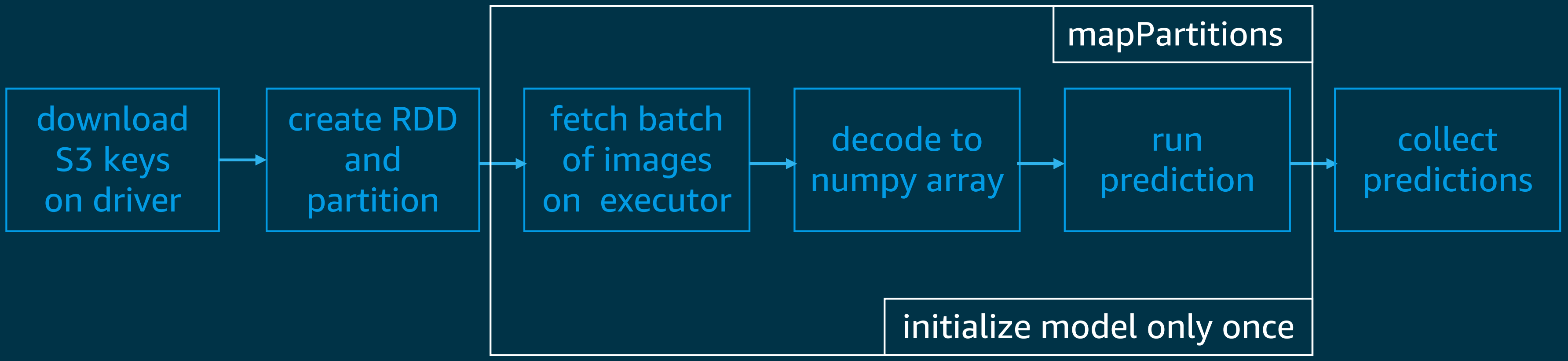
Efficient Partition of Data

Deep Learning Setup

MXNet + Spark for Inference.

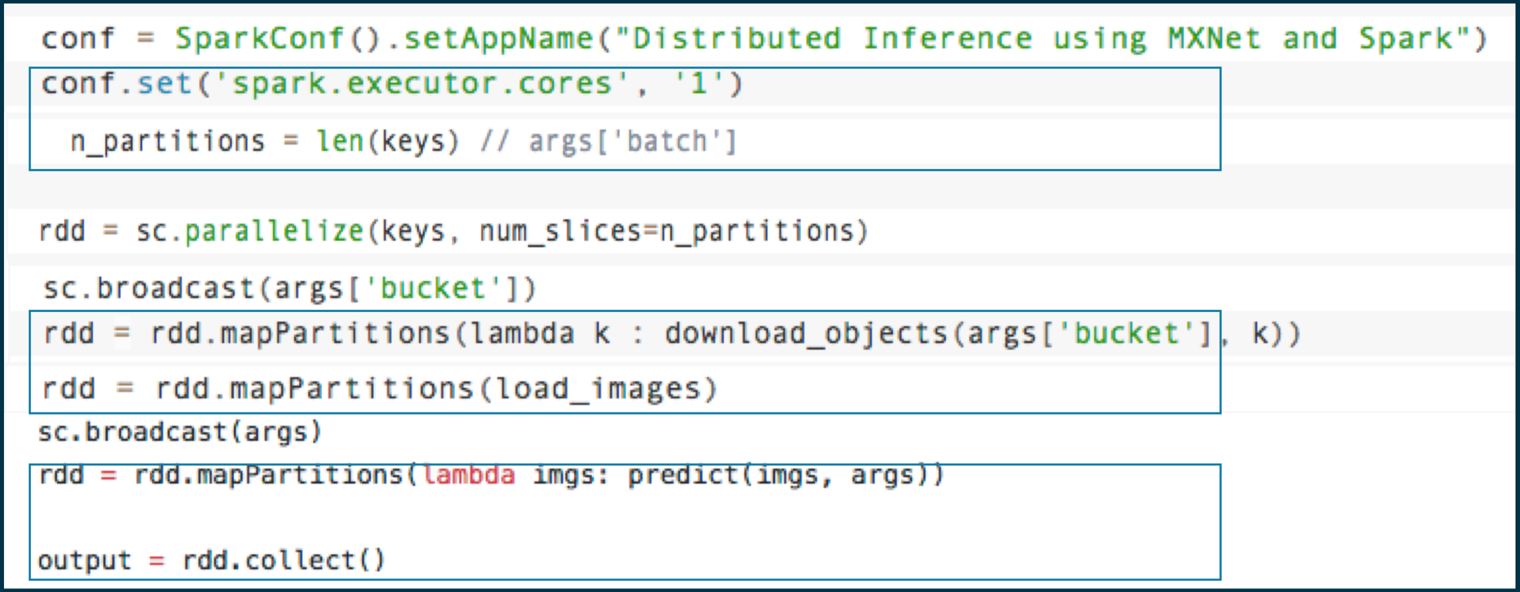
- ImageNet trained ResNet-18 classifier.
- For demo, CIFAR-10 test dataset with 10K Images.
- PySpark on Amazon EMR, MXNet is also available in Scala.
- Inference on CPUs, can be extended to use GPUs.

Distributed Inference Pipeline



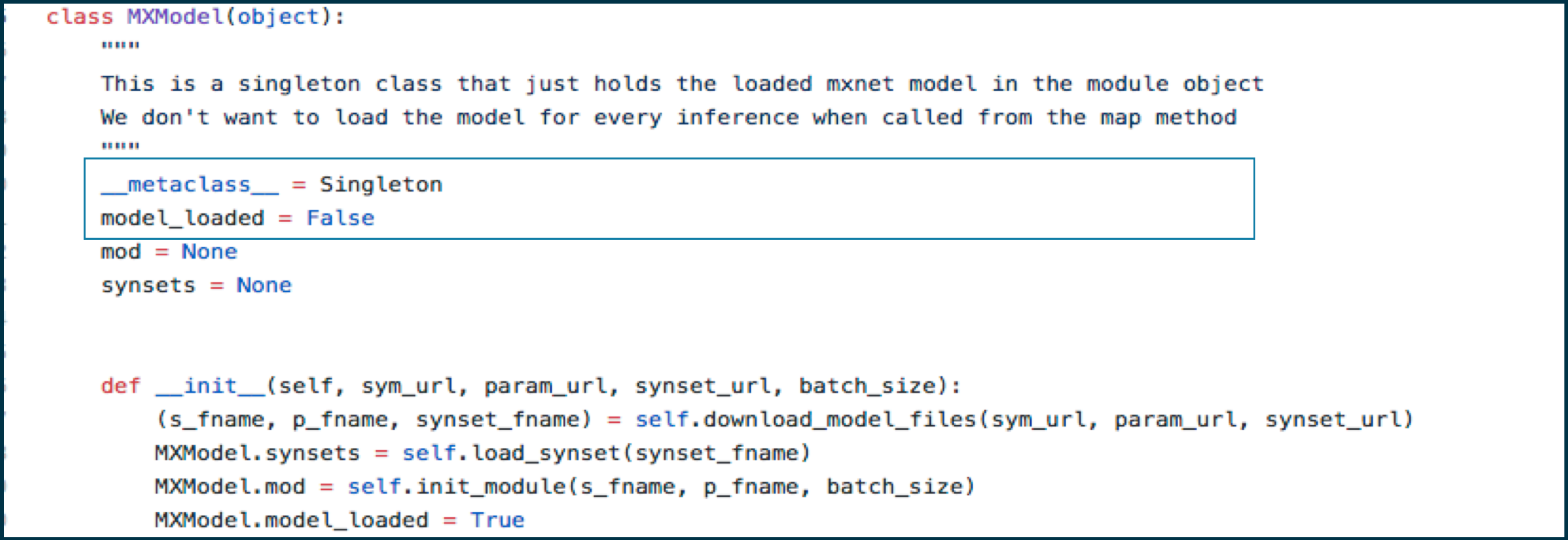
MXNet + Spark for Inference.

On the driver

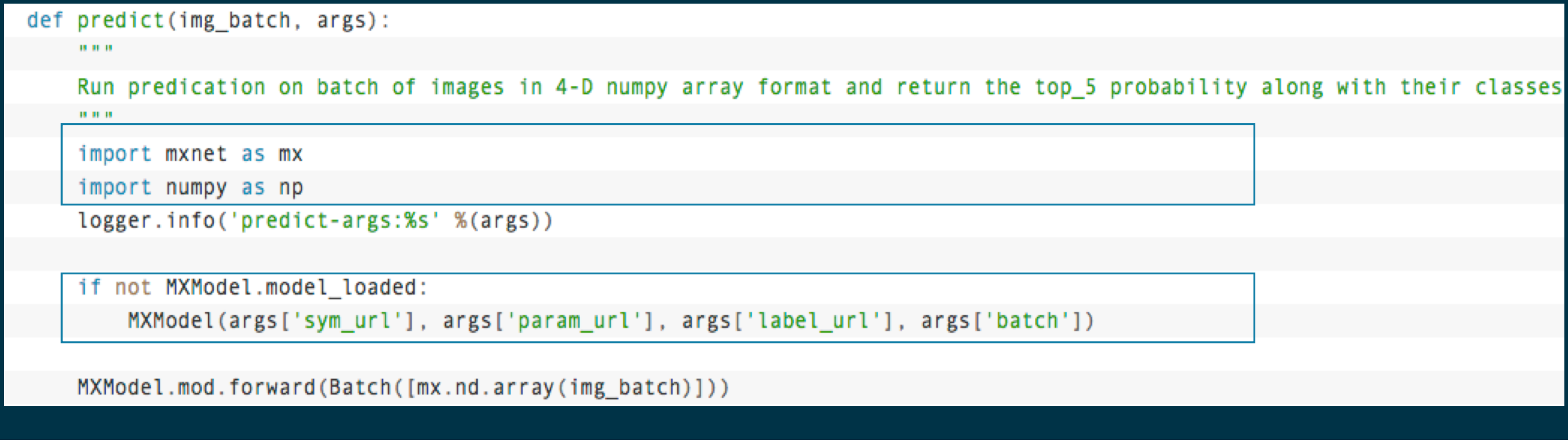


```
conf = SparkConf().setAppName("Distributed Inference using MXNet and Spark")
conf.set('spark.executor.cores', '1')
 n_partitions = len(keys) // args['batch']

rdd = sc.parallelize(keys, num_slices=n_partitions)
sc.broadcast(args['bucket'])
rdd = rdd.mapPartitions(lambda k : download_objects(args['bucket'], k))
rdd = rdd.mapPartitions(load_images)
sc.broadcast(args)
rdd = rdd.mapPartitions(lambda imgs: predict(imgs, args))

output = rdd.collect()
```

On the executor



```python
class MXModel(object):
    """
    This is a singleton class that just holds the loaded mxnet model in the module object
    We don't want to load the model for every inference when called from the map method
    """
    __metaclass__ = Singleton
    model_loaded = False
    mod = None
    synsets = None


    def __init__(self, sym_url, param_url, synset_url, batch_size):
        (s_fname, p_fname, synset_fname) = self.download_model_files(sym_url, param_url, synset_url)
        MXModel.synsets = self.load_synset(synset_fname)
        MXModel.mod = self.init_module(s_fname, p_fname, batch_size)
        MXModel.model_loaded = True
```



```python
def predict(img_batch, args):
    """
    Run predication on batch of images in 4-D numpy array format and return the top_5 probability along with their classes
    """
    import mxnet as mx
    import numpy as np
    logger.info('predict-args:%s' %(args))

    if not MXModel.model_loaded:
        MXModel(args['sym_url'], args['param_url'], args['label_url'], args['batch'])

    MXModel.mod.forward(Batch([mx.nd.array(img_batch)]))
```

Summary

- Overview of Deep Learning
    - How Deep Learning works and Why Deep Learning is a big deal.
    - Phases of Deep Learning
    - Types of Learning
- Apache MXNet – Efficient deep learning library
    - NDArray/Symbol/Module
- Apache MXNet and Spark for distributed Inference.

What’s Next ?

- Released simplified Scala Inference APIs (v1.2.0)
  - Available on Maven : org.apache.mxnet

- Working on Java APIs for Inference.

- Dataframe support is under consideration.

- MXNet community is fast evolving, join hands to democratize AI.

Resources/References

- https://github.com/apache/incubator-mxnet
- Blog- Distributed Inference using MXNet and Spark
- Distributed Inference code sample on GitHub
- Apache MXNet Gluon Tutorials
- Apache MXNet – Flexible and efficient deep learning.
- The Deep Learning Book
- MXNet – Using pre-trained models
- Amazon Elastic MapReduce

Thank You

nswamy@apache.org