DEEPLEARNING4J

DL4J COMPONENTS

- Open Source DeepLearning Library
- CPU & GPU support
- Hadoop-Yarn & Spark integration
- Futre additions

DL4J COMPONENTS

- Open Source Deep Learning Library

What is DeepLearning and which part of it is covered by DL4j? (Section 1)

- CPU & GPU support
- Hadoop-Yarn & Spark integration
- Futre additions

DL4J COMPONENTS

- Open Source Deep Learning Library

What is DeepLearning and which part of it is covered by DL4j? (Section 1)

- CPU & GPU support
- Hadoop-Yarn & Spark integration

What components are interfaced? (Section 2)

- Futre additions

DL4J COMPONENTS

- Open Source Deep Learning Library

What is DeepLearning and which part of it is covered by DL4j? (Section 1)

- CPU & GPU support
- Hadoop-Yarn & Spark integration

What components are interfaced? (Section 2)

How are the alogrithms distributed? (Section 3)

- Futre additions

DL4J COMPONENTS

- Open Source Deep Learning Library

What is DeepLearning and which part of it is covered by DL4j? (Section 1)

- CPU & GPU support
- ~~Hadoop-Yarn (MR)~~ & Spark integration

What components are interfaced? (Section 2)

How are the alogrithms distributed? (Section 3)

- Futre additions

DL4J COMPONENTS

- Open Source Deep Learning Library

What is DeepLearning and which part of it is covered by DL4j? (Section 1)

- CPU & GPU support
- ~~Hadoop-Yarn (MR)~~ & Spark integration

What components are interfaced? (Section 2)

How are the alogrithms distributed? (Section 3)

- Futre additions

Section 4 / The End.

DL4J COMPONENTS

- Open Source Deep Learning Library

What is DeepLearning and which part of it is covered by DL4j? (Section 1)

- CPU & GPU support
- ~~Hadoop-Yarn (MR)~~ & Spark integration

What components are interfaced? (Section 2)

How are the alogrithms distributed? (Section 3)

- Futre additions

Section 4 / The End.

Lets first look at what actually Deep Learning is

Generally: Deep Learning might be only an alternative term for greater sized artificial neural networks?

Bigger Datasets are also required (scaling/performance issues?)

Deep Learning in Deeplearning4J

DL4J offers the following Deep Neural Networks:

- Restricted Boltzmann Machines
- Convolutional Nets (Images)
- Recurrent nets/LSTMs
- Recursive Autoencoders
- Deep Belief Networks
- Recursive Neural Tensor (~Vector/Array) Networks
- Stacked Denoising Autoencoders

Data frame (/set) readers included for: MNIST, Labeled Faces in the Wild (LFW), IRIS

And DL4J also supports pipelining / stacking of layers.

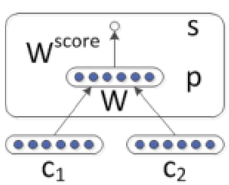
DEEP ARTIFICIAL NEURAL NETWORKS (DNNS)

Supervised Learning

Approximate pre-defined lables by output of network (e.g. classify faces as 1 and everything else as 0); can use output from unsupervised learning (e.g. DBNs) for weight initialization

Recursive Neural Networks

Appy the same layer multiple times



$$
\begin{aligned}
s &= W^{score}p \qquad (9)\\
p &= f(W[c_1;c_2]+b)
\end{aligned}
$$

Convolutional Neural Networks (CNNs)

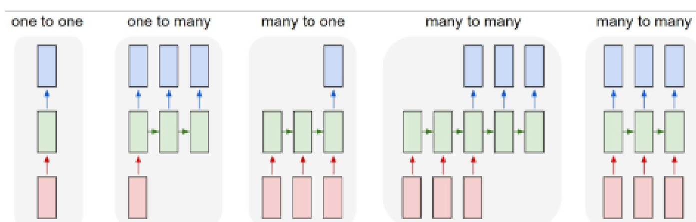
Tries to copy natural visual system (see next slides)

Recurrent Neural Networks (RNNs)

Those networks have backwards connections between their layers

Long Short Term Memory Networks (LSTMNs)

Special Case of RNNs, recently popular



Unsupervised Learning

Extract some features from the input without any predefined targets (e.g. dimensions with high variance or spare feature representations, such as lines/shapes/borders)

Restricted Boltzmann Machine

Deep Boltzmann Machine

Deep Belief Networks (DBNs)

Aims at a good stochastic reconstruction of the input

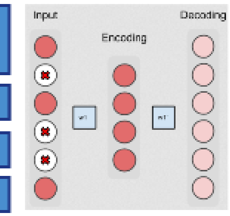
Convolutional Deep Belief Neworks (CDBNs)

Autoencoder

Stacked Autoencoder

Denoising Autoencoder

Sparse Autoencoder



$$\mathcal{L}_{ae} = ||W\sigma(W^T X) - X||^2$$

Spare Coding Models

$$\mathcal{L}_{sc} = \underbrace{||WH - X||_2^2}_{\text{reconstruction term}} + \underbrace{\lambda ||H||_1}_{\text{sparsity term}}$$

Semi-Supervised Learning

Combine both.

Reeinforcement Learning

Define a reward function and try to gain as much reward as possible as a result of the chosen actions (could be considered as somewhat supervised);

also here some work is currently in progress DNN implementing this type of learning (some of those above can be adapted to do so)

More Info & Sources:

http://stats.stackexchange.com/questions/118199/what-are-the-differences-between-sparse-coding-and-autoencoder

ttp://stats.stackexchange.com/questions/114385/what-is-the-difference-between-convolutional-neural-networks-restricted-boltzma

https://www.youtube.com/watch?v=oRxgRHJztPI

https://www.youtube.com/watch?v=lekCh_i32iE

MOTIVATION: RECURRENT NEURAL NETWORKS

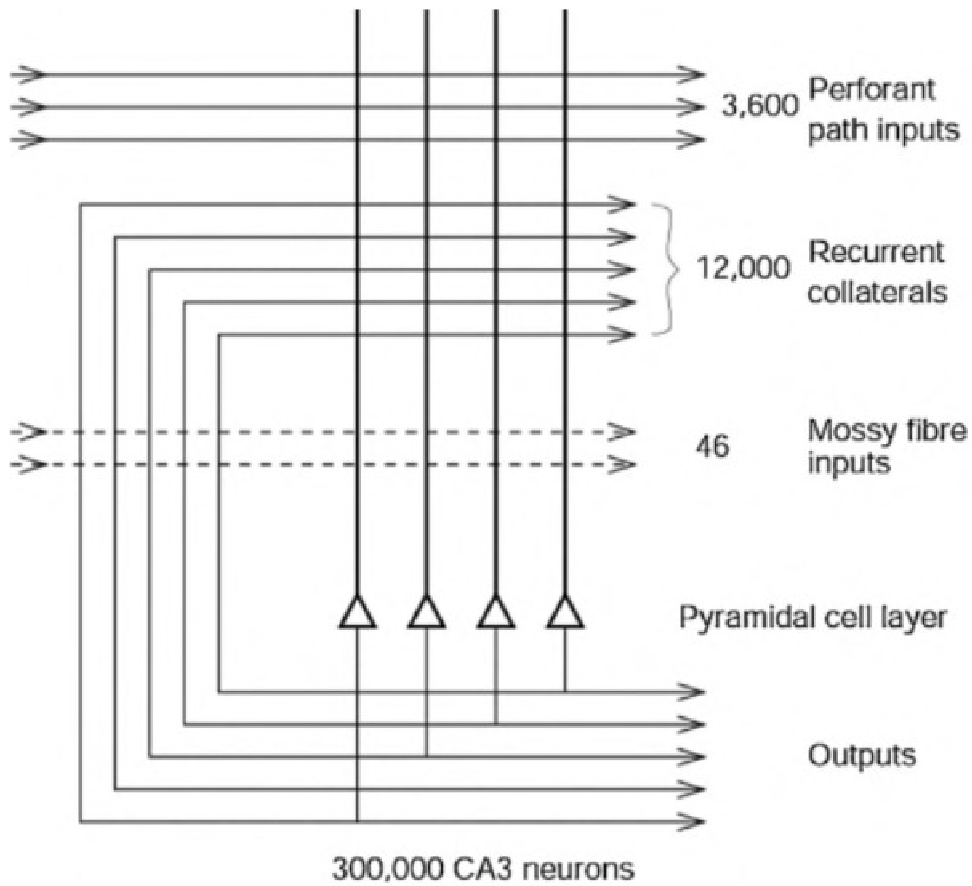


Fig. 3. The numbers of connections from three different sources onto each CA3 cell from three different sources in the rat.
After Rolls and Treves (1998), Treves and Rolls (1992).

- Recurrent connections in Hippocampus enable sequential memory and different types of time depended learning

USES FOR RECURSIVE/RECURRENT NEURAL NETWORKS

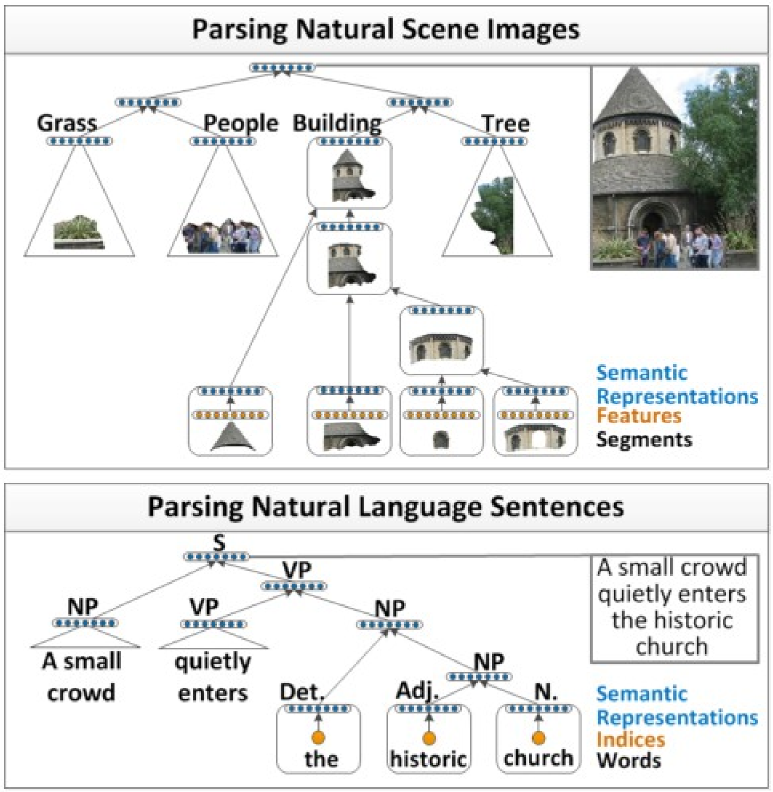


Figure 1. Illustration of our recursive neural network architecture which parses images and natural language sentences. Segment features and word indices (orange) are first mapped into semantic feature space (blue) and then recursively merged by the same neural network until they represent the entire image or sentence. Both mappings and mergings are learned.

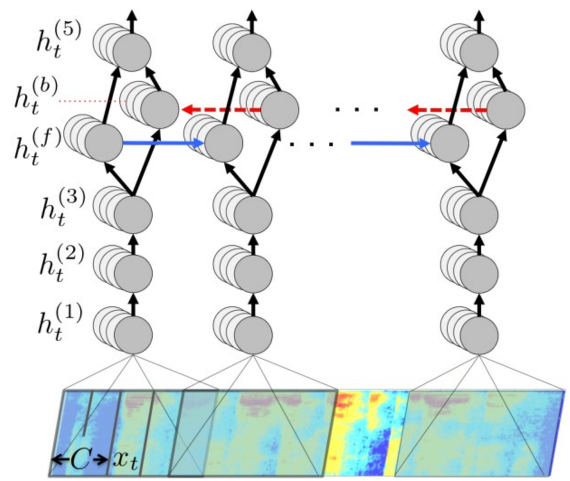
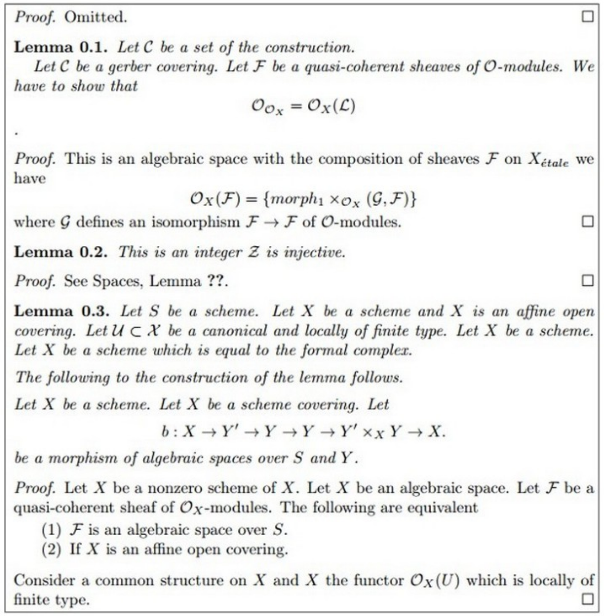


Figure 1: Structure of our RNN model and notation.



Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{étale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

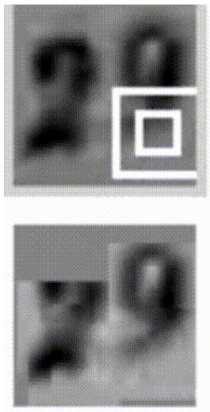
$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

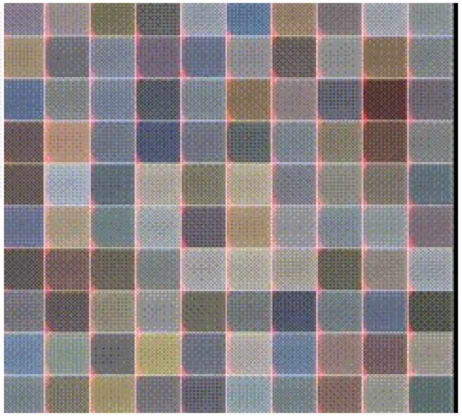
be a morphism of algebraic spaces over S and Y.

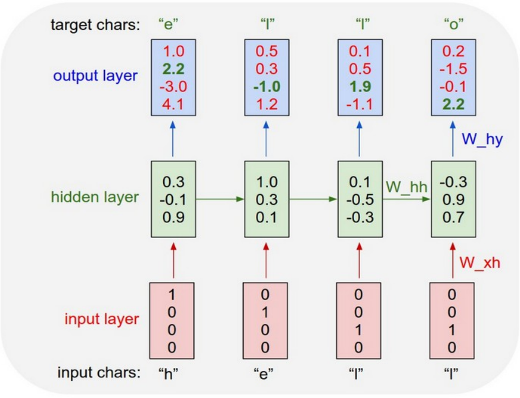
Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □







Left: Socher et al. (2011) - Parsing Natural Scenes and Natural Languagewith Recursive Neural Networks (Google)

Top Middle: Hannun (2014) - Deep Speech: Scaling up end-to-endspeech recognition (Baidu)

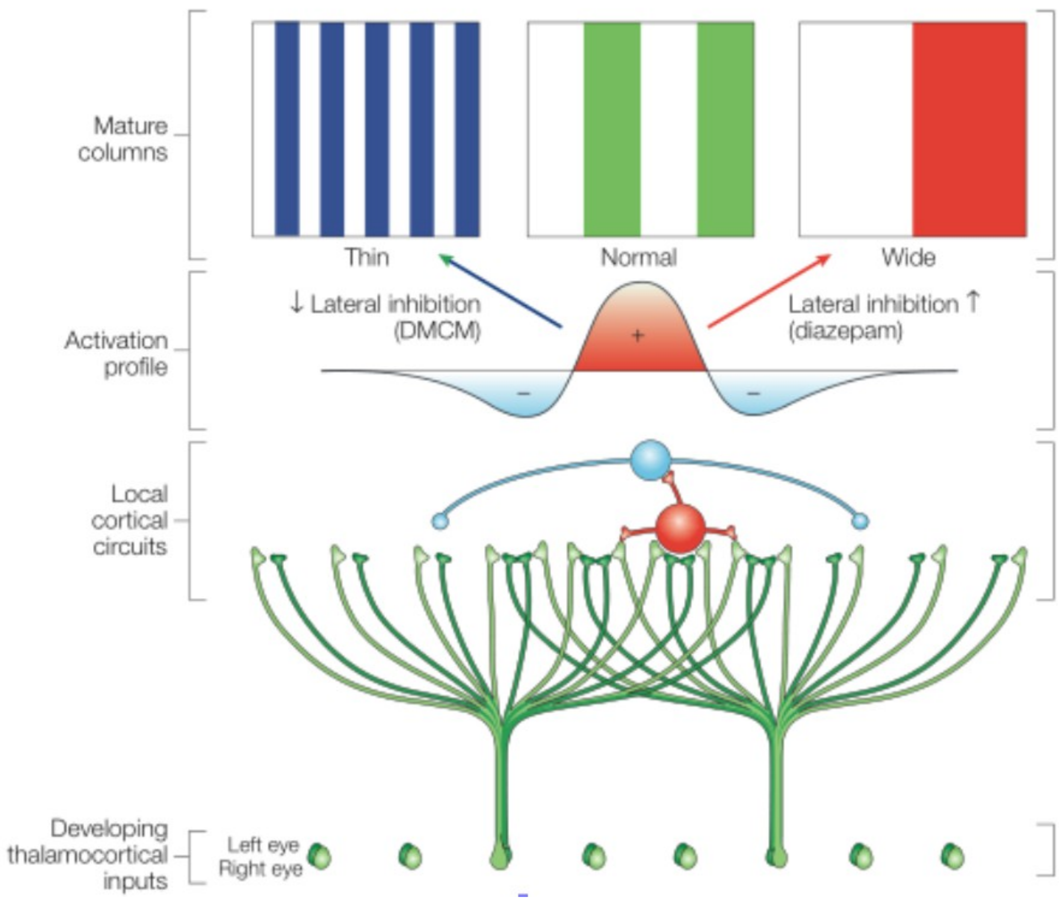
Rest: Karpathy (2015) - The Unreasonable Effectiveness of Recurrent Neural Networks [link]

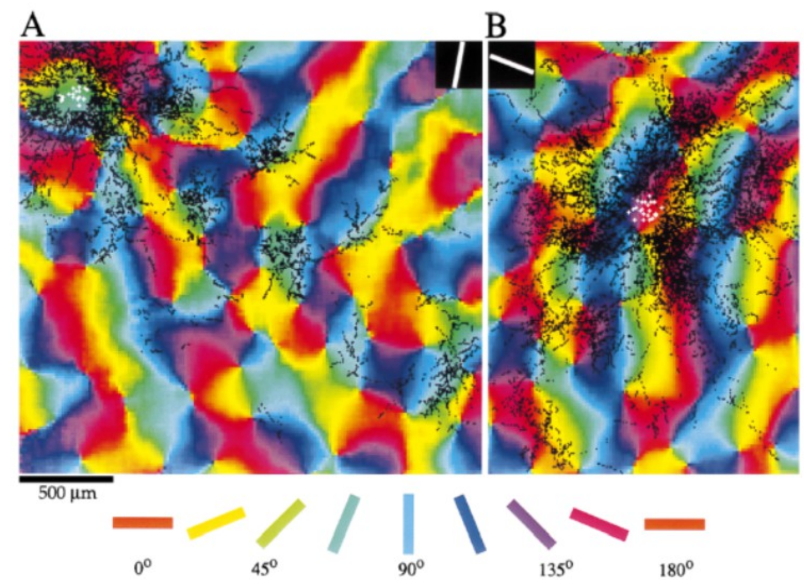
Further Readings: Gillick (2016) - Multilingual Language Processing From Bytes; Sak (2015) - Fast and Accurate Recurrent Neural Network Acoustic Models for Speech Recognition; Marek et al (2015) - Large-Scale Language Classification (200 Languages, Twitter)

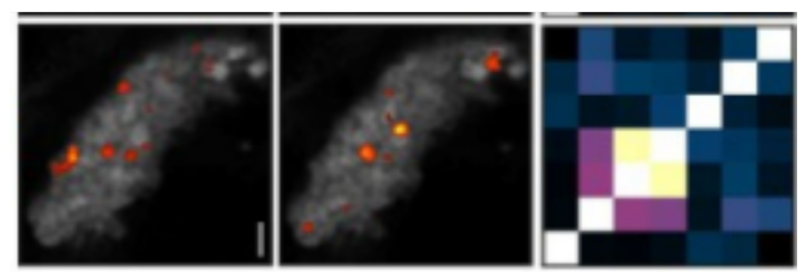
EXAMPLE: RECURRENT NEURAL NETWORKS IN DL4J

```
//Get a DataSetIterator that handles vectorization of text into something we can use to train
// our GravesLSTM network.
CharacterIterator iter = getShakespeareIterator(miniBatchSize,exampleLength);
int nOut = iter.totalOutcomes();

//Set up network configuration:
MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
    .optimizationAlgo(OptimizationAlgorithm.STOCHASTIC_GRADIENT_DESCENT).iterations(1)
    .learningRate(0.1)
    .rmsDecay(0.95)
    .seed(12345)
    .regularization(true)
    .l2(0.001)
    .weightInit(WeightInit.XAVIER)
    .updater(Updater.RMSPROP)
    .list()
    .layer(0, new GravesLSTM.Builder().nIn(iter.inputColumns()).nOut(lstmLayerSize)
            .activation("tanh").build())
    .layer(1, new GravesLSTM.Builder().nIn(lstmLayerSize).nOut(lstmLayerSize)
            .activation("tanh").build())
    .layer(2, new RnnOutputLayer.Builder(LossFunction.MCXENT).activation("softmax")        //MCXENT + softmax for classif
            .nIn(lstmLayerSize).nOut(nOut).build())
    .backpropType(BackpropType.TruncatedBPTT).tBPTTForwardLength(tbpttLength).tBPTTBackwardLength(tbpttLength)
    .pretrain(false).backprop(true)
    .build();

MultiLayerNetwork net = new MultiLayerNetwork(conf);
net.init();
net.setListeners(new ScoreIterationListener(1));

//Do training, and then generate and print samples from network
int miniBatchNumber = 0;
for( int i=0; i<numEpochs; i++ ){
    while(iter.hasNext()){
        DataSet ds = iter.next();
        net.fit(ds);
        if(++miniBatchNumber % generateSamplesEveryNMinibatches == 0){
            System.out.println("--------------------");
            System.out.println("Completed " + miniBatchNumber + " minibatches of size " + miniBatchSize + "x" + exampleLen
            System.out.println("Sampling characters from network given initialization \"" + (generationInitialization == n
            String[] samples = sampleCharactersFromNetwork(generationInitialization,net,iter,rng,nCharactersToSample,nSamp
            for( int j=0; j<samples.length; j++ ){
                System.out.println("----- Sample " + j + " -----");
                System.out.println(samples[j]);
                System.out.println();
            }
        }
    }
```

MOTIVATION: SPARSITY & EXTRACTION OF STIMULY STATISTICS





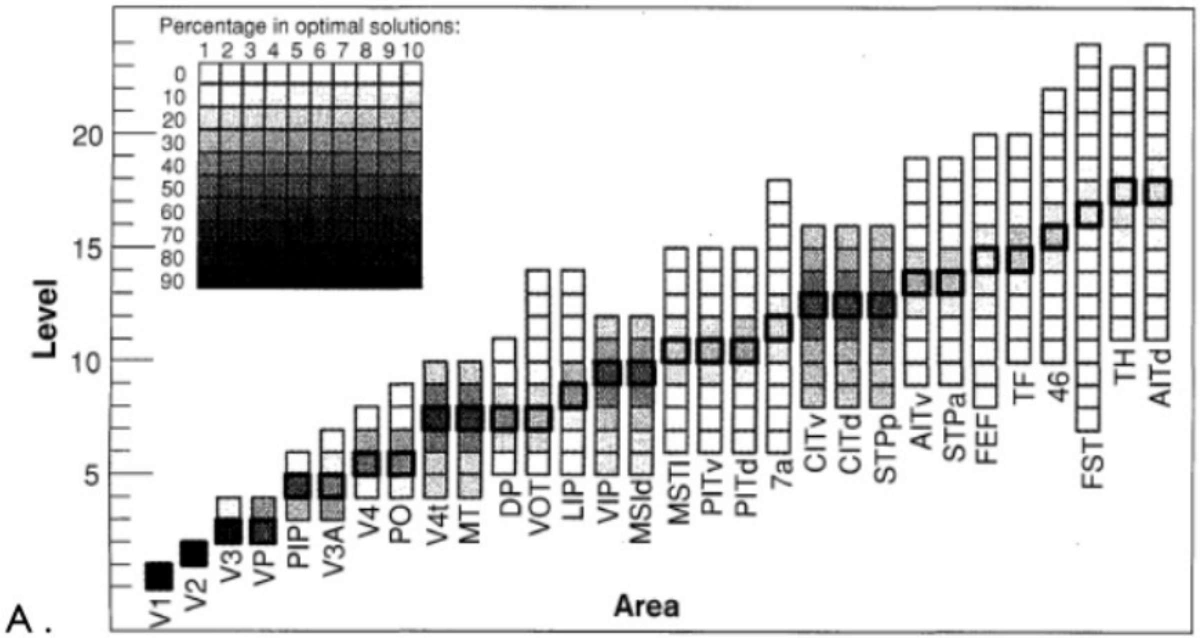


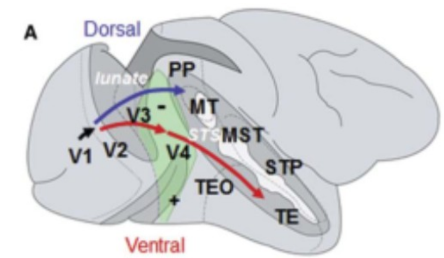
- High information content (direction of high variance extracted by e.g. Denoising AE /)
- Low overlap of feature maps, so more feat can be extracted

EXAMPLE: UNSUPERVISED LEARNING IN DL4J

```
log.info("Build model....");
MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
   .seed(seed)
   .gradientNormalization(GradientNormalization.ClipElementWiseAbsoluteValue)
   .gradientNormalizationThreshold(1.0)
   .iterations(iterations)
   .momentum(0.5)
   .momentumAfter(Collections.singletonMap(3, 0.9))
   .optimizationAlgo(OptimizationAlgorithm.CONJUGATE_GRADIENT)
   .list(4)
   .layer(0, new AutoEncoder.Builder().nIn(numRows * numColumns).nOut(500)
           .weightInit(WeightInit.XAVIER).lossFunction(LossFunction.RMSE_XENT)
           .corruptionLevel(0.3)
           .build())
        .layer(1, new AutoEncoder.Builder().nIn(500).nOut(250)
                .weightInit(WeightInit.XAVIER).lossFunction(LossFunction.RMSE_XENT)
                .corruptionLevel(0.3)

                .build())
        .layer(2, new AutoEncoder.Builder().nIn(250).nOut(200)
                .weightInit(WeightInit.XAVIER).lossFunction(LossFunction.RMSE_XENT)
                .corruptionLevel(0.3)
                .build())
        .layer(3, new OutputLayer.Builder(LossFunction.NEGATIVELOGLIKELIHOOD).activation("softmax")
                .nIn(200).nOut(outputNum).build())
   .pretrain(true).backprop(false)
        .build();

MultiLayerNetwork model = new MultiLayerNetwork(conf);
model.init();
model.setListeners(Collections.singletonList((IterationListener) new ScoreIterationListener(listenerFreq)));

log.info("Train model....");
```

MOTIVATION: CONVLUTIONAL NEURAL NETWORKS

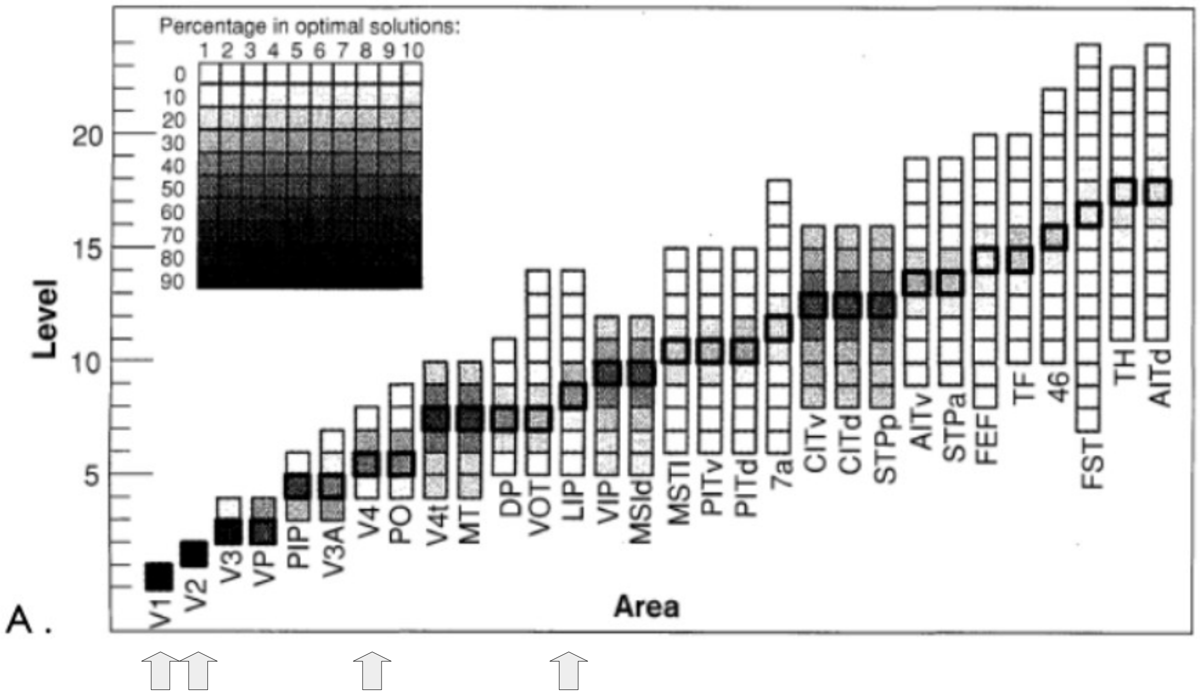


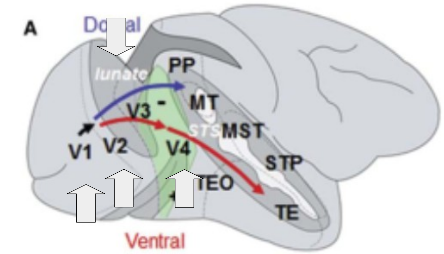


Top: Scannel (1993) - the connectional organization of the cat visual cortex
Bottom Right: Fellemann & Van Essen (1991) - Distributed Hierarchical Processing in the Primate Cerebral Cortex

The (human) visual sytem is hierarchically organized (each area thereof consists of multiple neuron layers [I-VI] with different types of connectivity.

MOTIVATION: CONVLUTIONAL NEURAL NETWORKS

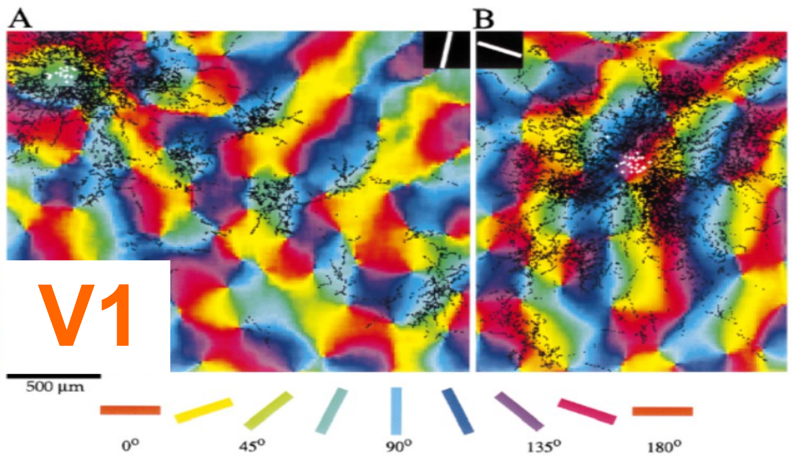




Top: Scannel (1993) - the connectional organization of the cat visual cortex
Bottom Right: Fellemann & Van Essen (1991) - Distributed Hierarchical Processing in the Primate Cerebral Cortex

The (human) visual sytem is hierarchically organized (each area thereof consists of multiple neuron layers [I-VI] with different types of connectivity.

MOTIVATION: CONVLUTIONAL NEURAL NETWORKS



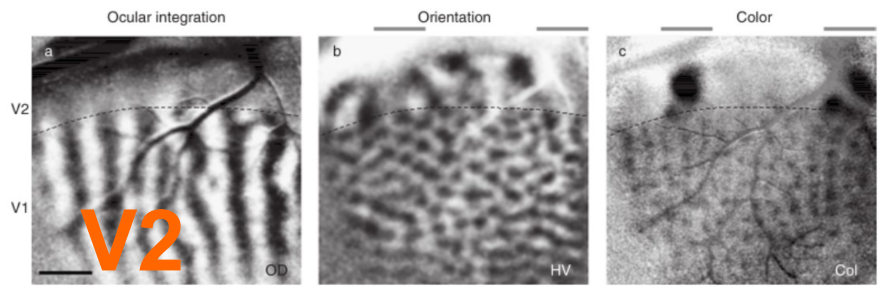


Figure 4 Feature maps (ocularity, orientation, and color) within V1 and V2 of macaque monkey. (a) Ocular dominance (OD) map (left eye minus right eye). V1–V2 border (indicated by dotted line in (a)–(c)). (b) Orientation map (horizontal minus vertical; HV). V2 orientation domains are as much as several times larger than those of V1. (c) Color (Col) map (red–green luminance grating minus luminance grating) reveals V2 thin stripes and V1 blobs. Note that locations of color activations in V2 (dark bars) are complementary to locations of oriented domains in V2 (compare (b) and (c)). Scale bar = 1 mm. From Lu HD and Roe AW (2007) Functional organization of color domains in V1 and V2 of macaque monkey revealed by optical imaging. *Cerebral Cortex* (doi: 10.1093/cercor/bhm081).

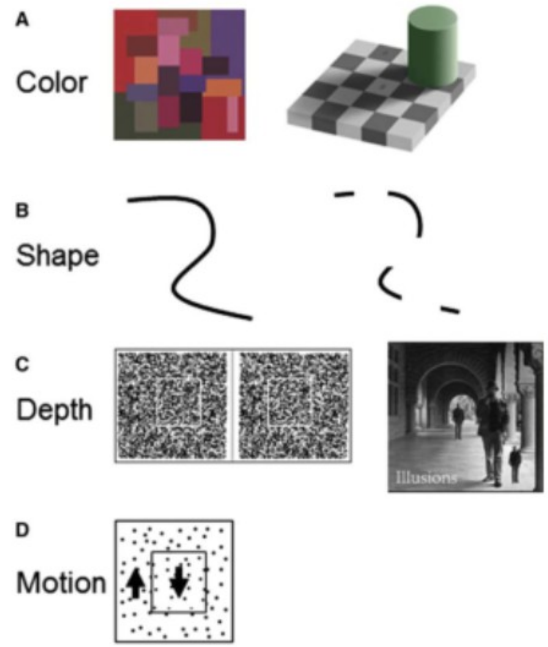


Figure 5. Examples of Transformations in V4
(A) Color: color constancy (left) and lightness constancy (right
(B) Shape: curvature, sparse coding of curvature.
(C) Depth: binocular correspondence, size constancy.
(D) Motion: motion contrast-defined shape.

V4

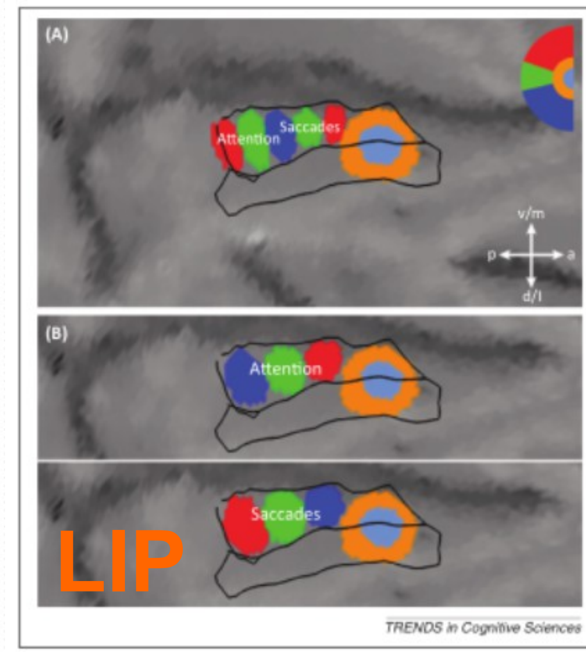


Figure 3. Possibilities for lateral intraparietal area (LIP) topographic organization. **(A)** Separate maps for attention and saccades. **(B)** Overlapping attention and saccade maps.

Hierarchical Processing for Extraction of different features (= Feature Maps) in the Visual System

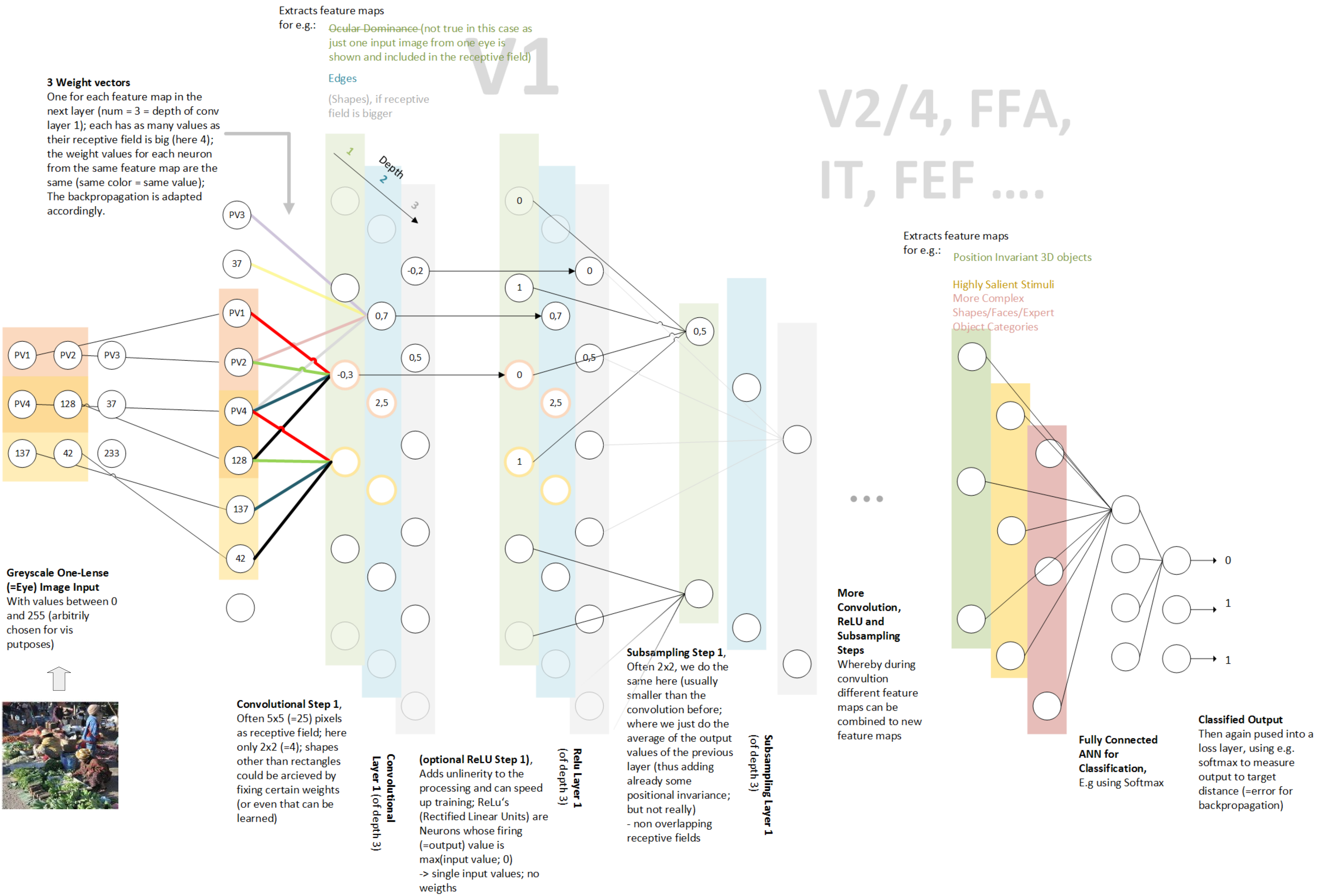
See next slide for how CNN do this ...

TL: Betsch & König et al. (2004) - The world from a cat's perspective – statistics of natural videos

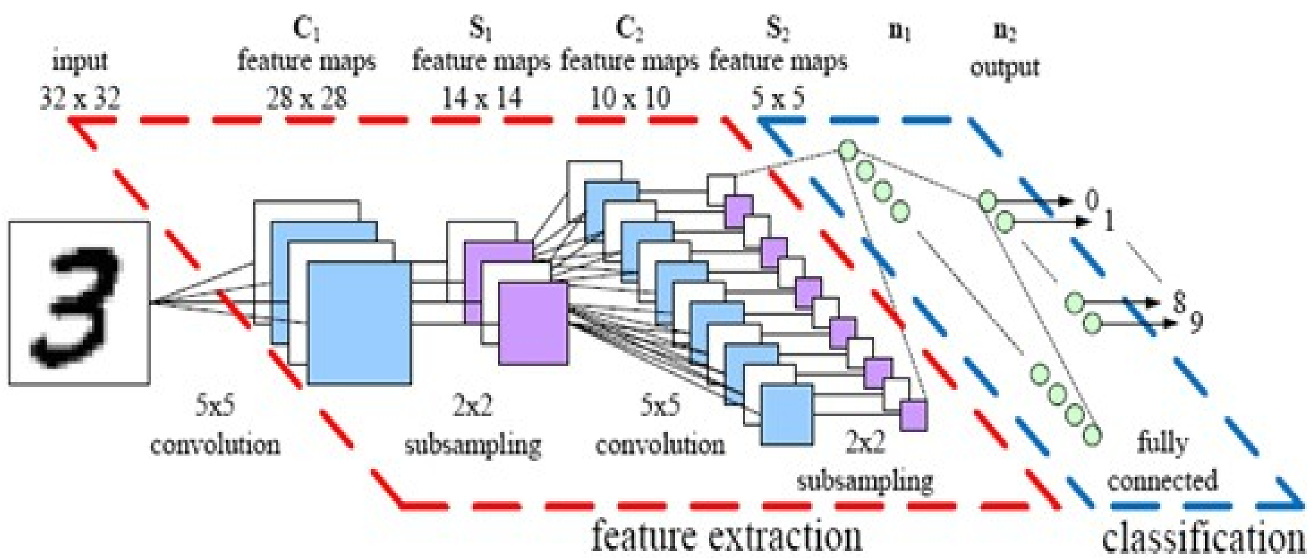
BL: Roe et al. (2009) - Visual System: Functional Architecture of Area V2

C: Roe et al. (2012) - Toward a Unified Theory of Visual Area V4

R: Patel (2014) - Topographic organization in the brain: searching for general principles


Extracts feature maps
for e.g.: ~~Ocular Dominance~~ (not true in this case as just one input image from one eye is shown and included in the receptive field)
Edges
(Shapes), if receptive field is bigger
V1
V2/4, FFA, IT, FEF ....
3 Weight vectors
One for each feature map in the next layer (num = 3 = depth of conv layer 1); each has as many values as their receptive field is big (here 4); the weight values for each neuron from the same feature map are the same (same color = same value); The backpropagation is adapted accordingly.
Depth
1
2
3
PV3
37
PV1
PV2
PV4
128
137
42
PV1 PV2 PV3
PV4 128 37
137 42 233
-0,2
0,7
0,5
-0,3
2,5
0
1
0,7
0
2,5
1
0
0,5
0,5
Extracts feature maps
for e.g.: Position Invariant 3D objects
Highly Salient Stimuli
More Complex Shapes/Faces/Expert Object Categories
0
1
1
Greyscale One-Lense (=Eye) Image Input
With values between 0 and 255 (arbitrily chosen for vis putposes)
Convolutional Step 1,
Often 5x5 (=25) pixels as receptive field; here only 2x2 (=4); shapes other than rectangles could be arcieved by fixing certain weights (or even that can be learned)
Convolutional Layer 1 (of depth 3)
(optional ReLU Step 1),
Adds unlinerity to the processing and can speed up training; ReLu's (Rectified Linear Units) are Neurons whose firing (=output) value is max(input value; 0)
-> single input values; no weigths
Relu Layer 1 (of depth 3)
Subsampling Step 1,
Often 2x2, we do the same here (usually smaller than the convolution before; where we just do the average of the output values of the previous layer (thus adding already some positional invariance; but not really)
- non overlapping receptive fields
Subsampling Layer 1 (of depth 3)
More Convolution, ReLU and Subsampling Steps
Whereby during convultion different feature maps can be combined to new feature maps
Fully Connected ANN for Classification,
E.g using Softmax
Classified Output
Then again pused into a loss layer, using e.g. softmax to measure output to target distance (=error for backpropagation)

REAL USES FOR CONVLUTIONAL NEURAL NETWORKS



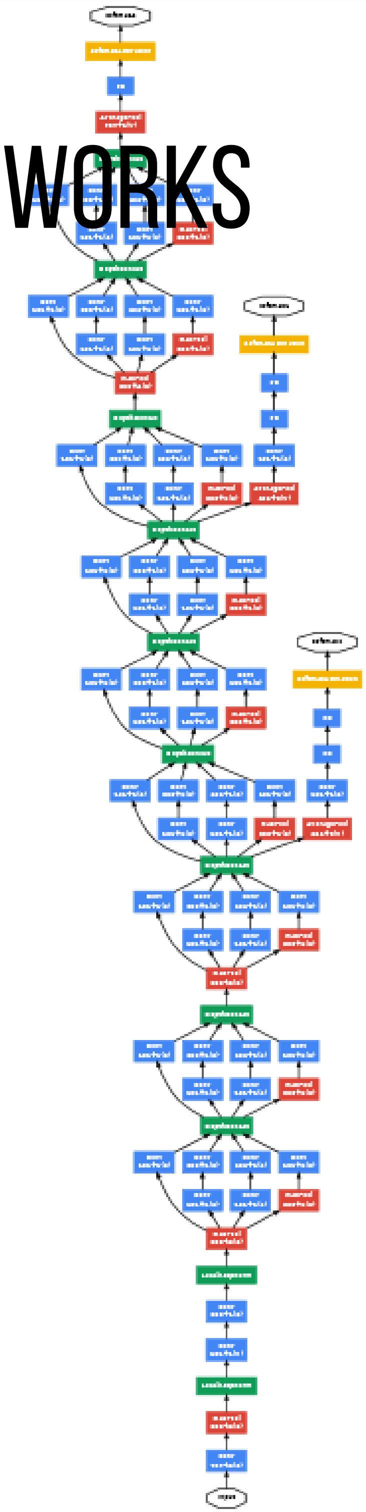


Figure 3: GoogLeNet network with all the bells and whistles

Left: Socher et al. (2011) - Parsing Natural Scenes and Natural Languagewith Recursive Neural Networks (Google)

Right: Szegedy (2014) - Going Deeper with Convolutions (GoogLeNet/Inception for Image Classification & Detection)

EXAMPLE: CONVLUTIONAL NEURAL NETWORKS IN DL4J

```java
log.info("Build model....");
MultiLayerConfiguration.Builder builder = new NeuralNetConfiguration.Builder()
        .seed(seed)
        .iterations(iterations)
        .regularization(true).l2(0.0005)
        .learningRate(0.1)
        .optimizationAlgo(OptimizationAlgorithm.STOCHASTIC_GRADIENT_DESCENT)
        .updater(Updater.ADAGRAD)
        .list(6)
        .layer(0, new ConvolutionLayer.Builder(5, 5)
                .nIn(nChannels)
                .stride(1, 1)
                .nOut(20)
                .weightInit(WeightInit.XAVIER)
                .activation("relu")
                .build())
        .layer(1, new SubsamplingLayer.Builder(SubsamplingLayer.PoolingType.MAX, new int[]{2, 2})
                .build())
        .layer(2, new ConvolutionLayer.Builder(5, 5)
                .nIn(20)
                .nOut(50)
                .stride(2,2)
                .weightInit(WeightInit.XAVIER)
                .activation("relu")
                .build())
        .layer(3, new SubsamplingLayer.Builder(SubsamplingLayer.PoolingType.MAX, new int[]{2, 2})
                .build())
        .layer(4, new DenseLayer.Builder().activation("relu")
                .weightInit(WeightInit.XAVIER)
                .nOut(200).build())
        .layer(5, new OutputLayer.Builder(LossFunctions.LossFunction.NEGATIVELOGLIKELIHOOD)
                .nOut(outputNum)
                .weightInit(WeightInit.XAVIER)
                .activation("softmax")
                .build())
        .backprop(true).pretrain(false);
new ConvolutionLayerSetup(builder,28,28,1);
```

COMBINATION OF DIFFEREN DNNS

Can be done by stacking of different layer types

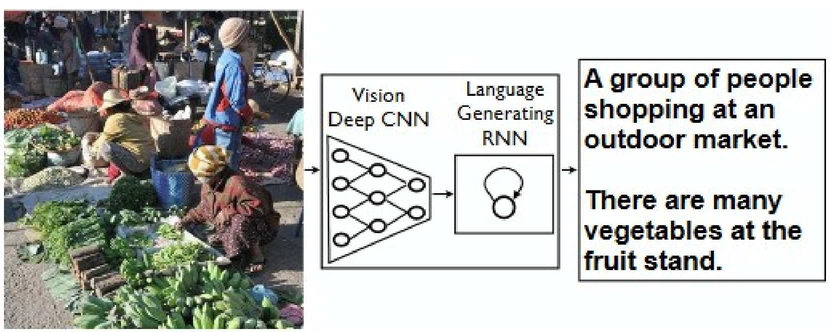


Figure 1. NIC, our model, is based end-to-end on a neural network consisting of a vision CNN followed by a language generating RNN. It generates complete sentences in natural language from an input image, as shown on the example above.

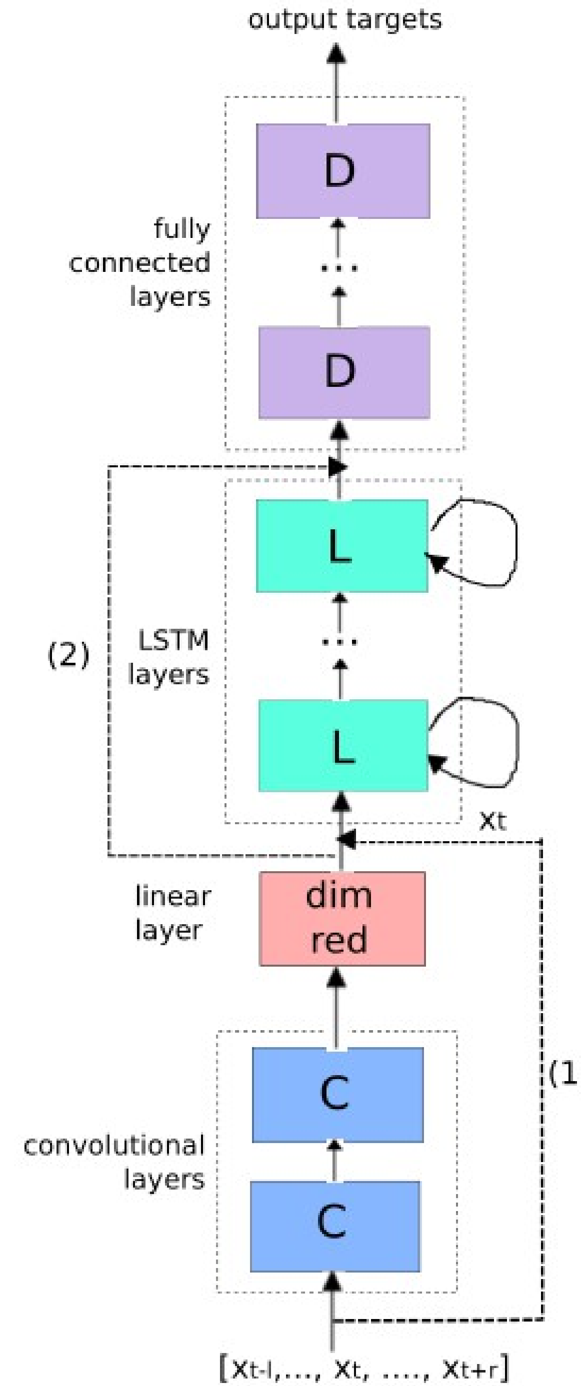
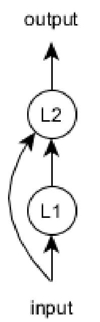


Fig. 1. CLDNN Architecture

Top: Vinyals (2015) - Show and Tell: A Neural Image Caption Generator

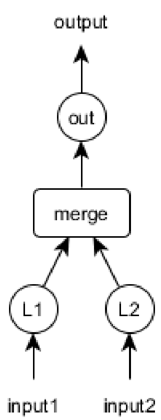
Right: Sak (2015) – Convolutional, Long Short-Term Memory, Fully Connected Deep Neural Networks

EXAMPLE: CHAINING NEURAL NETWORKS IN DL4J



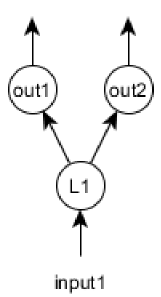
Recurrent Network with Skip Connections

```
ComputationGraphConfiguration conf = new NeuralNetConfiguration.Builder()
        .learningRate(0.01)
        .graphBuilder()
        .addInputs("input") //can use any label for this
        .addLayer("L1", new GravesLSTM.Builder().nIn(5).nOut(5).build(), "input")
        .addLayer("L2",new RnnOutputLayer.Builder().nIn(5+5).nOut(5).build(), "input", "L1")
        .setOutputs("L2")       //We need to specify the network outputs and their order
        .build();

ComputationGraph net = new ComputationGraph(conf);
net.init();
```



Multiple Inputs and Merge Vertex

```
ComputationGraphConfiguration conf = new NeuralNetConfiguration.Builder()
        .learningRate(0.01)
        .graphBuilder()
        .addInputs("input1", "input2")
        .addLayer("L1", new DenseLayer.Builder().nIn(3).nOut(4).build(), "input1")
        .addLayer("L2", new DenseLayer.Builder().nIn(3).nOut(4).build(), "input2")
        .addVertex("merge", new MergeVertex(), "L1", "L2")
        .addLayer("out", new OutputLayer.Builder().nIn(4+4).nOut(3).build(), "merge")
        .setOutputs("out")
        .build();
```



Multi Task Learning

```
ComputationGraphConfiguration conf = new NeuralNetConfiguration.Builder()
        .learningRate(0.01)
        .graphBuilder()
        .addInputs("input")
        .addLayer("L1", new DenseLayer.Builder().nIn(3).nOut(4).build(), "input")
        .addLayer("out1", new OutputLayer.Builder()
                .lossFunction(LossFunctions.LossFunction.NEGATIVELOGLIKELIHOOD)
                .nIn(4).nOut(3).build(), "L1")
        .addLayer("out2", new OutputLayer.Builder()
                .lossFunction(LossFunctions.LossFunction.MSE)
                .nIn(4).nOut(2).build(), "L1")
        .setOutputs("out1","out2")
        .build();
```

See: http://deeplearning4j.org/compgraph

DL4J COMPONENTS

- Open Source Deep Learning Library

What is DeepLearning and which part of it is covered by DL4j? (Section 1)

- CPU & GPU support
- Hadoop-Yarn (MR) & Spark integration

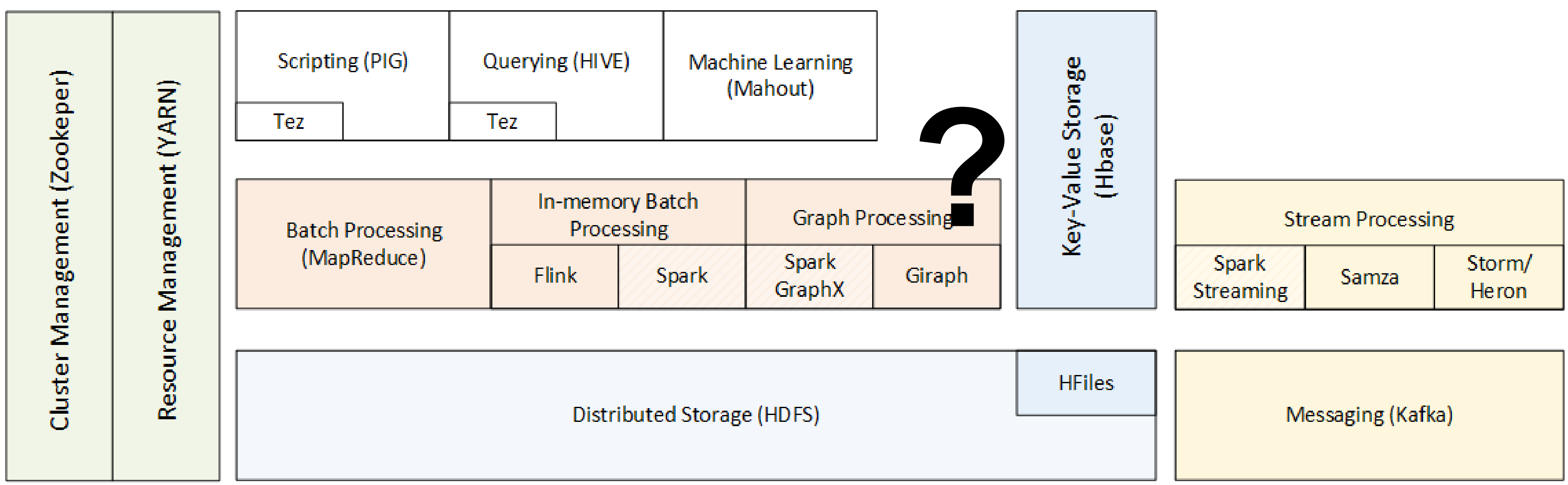
What components are interfaced? (Section 2)

How are the alogrithms distributed? (Section 3)

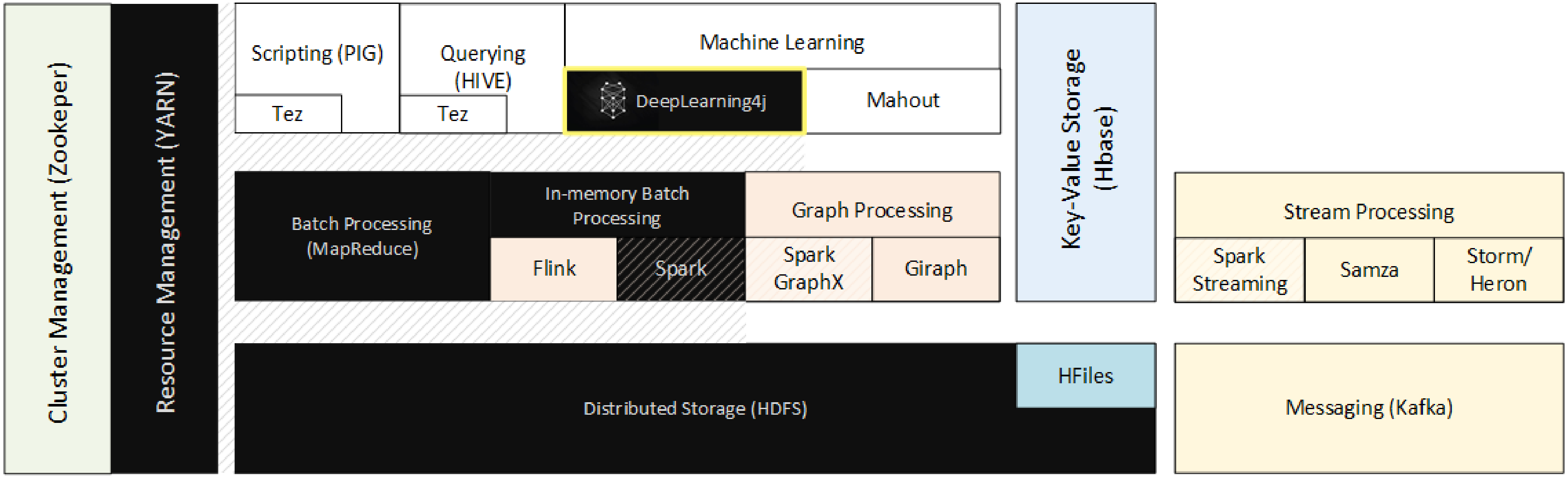
- Futre additions

Section 4 / The End.

(HADOOP) AND SPARK INTEGRATION



(HADOOP) AND SPARK INTEGRATION



DL4J COMPONENTS

- Open Source Deep Learning Library

What is DeepLearning and which part of it is covered by DL4j? (Section 1)

- CPU & GPU support
- Hadoop-Yarn (MR) & Spark integration

What components are interfaced? (Section 2)

How are the alogrithms distributed? (Section 3)

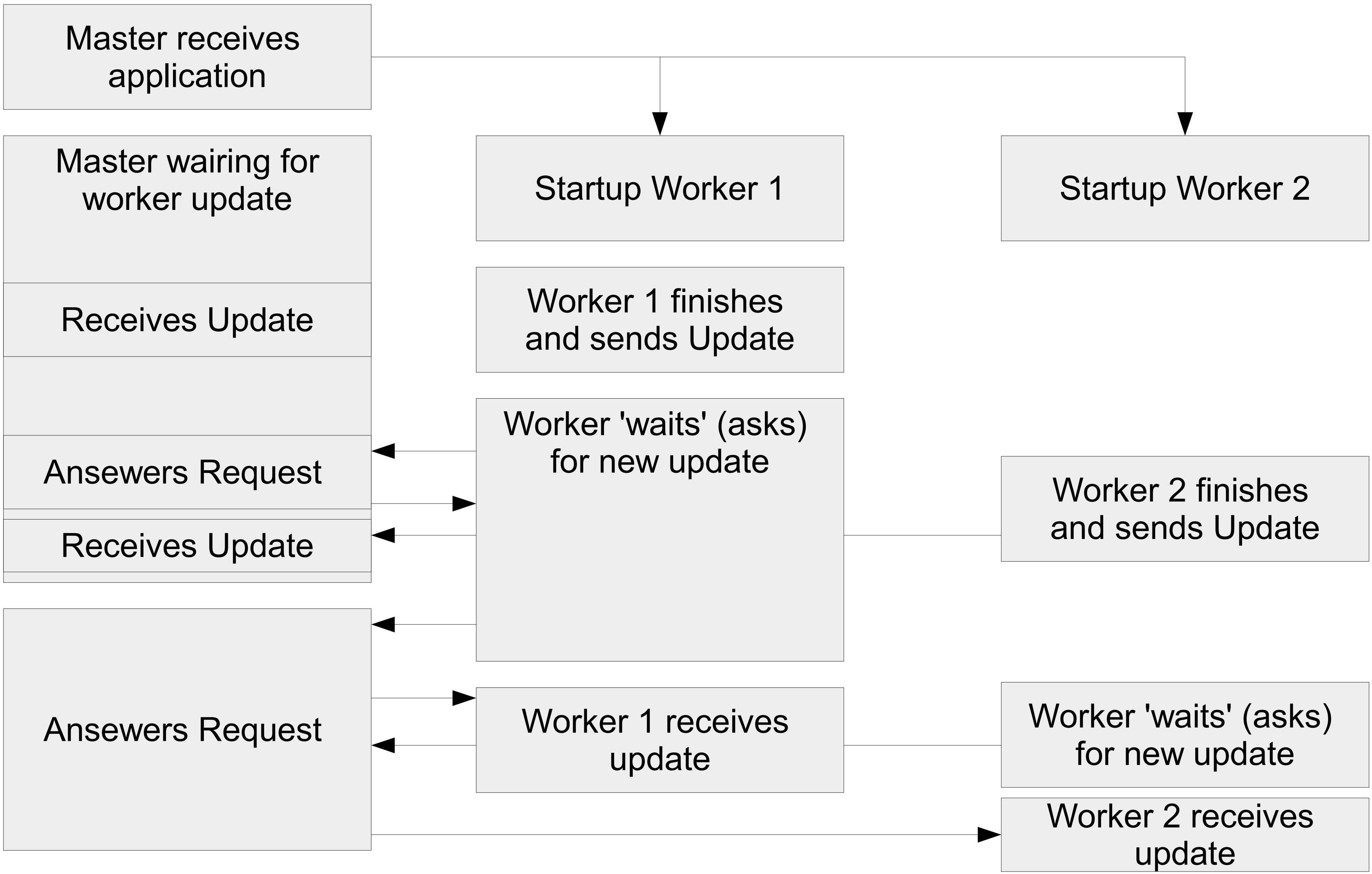
- Futre additions

Section 4 / The End.

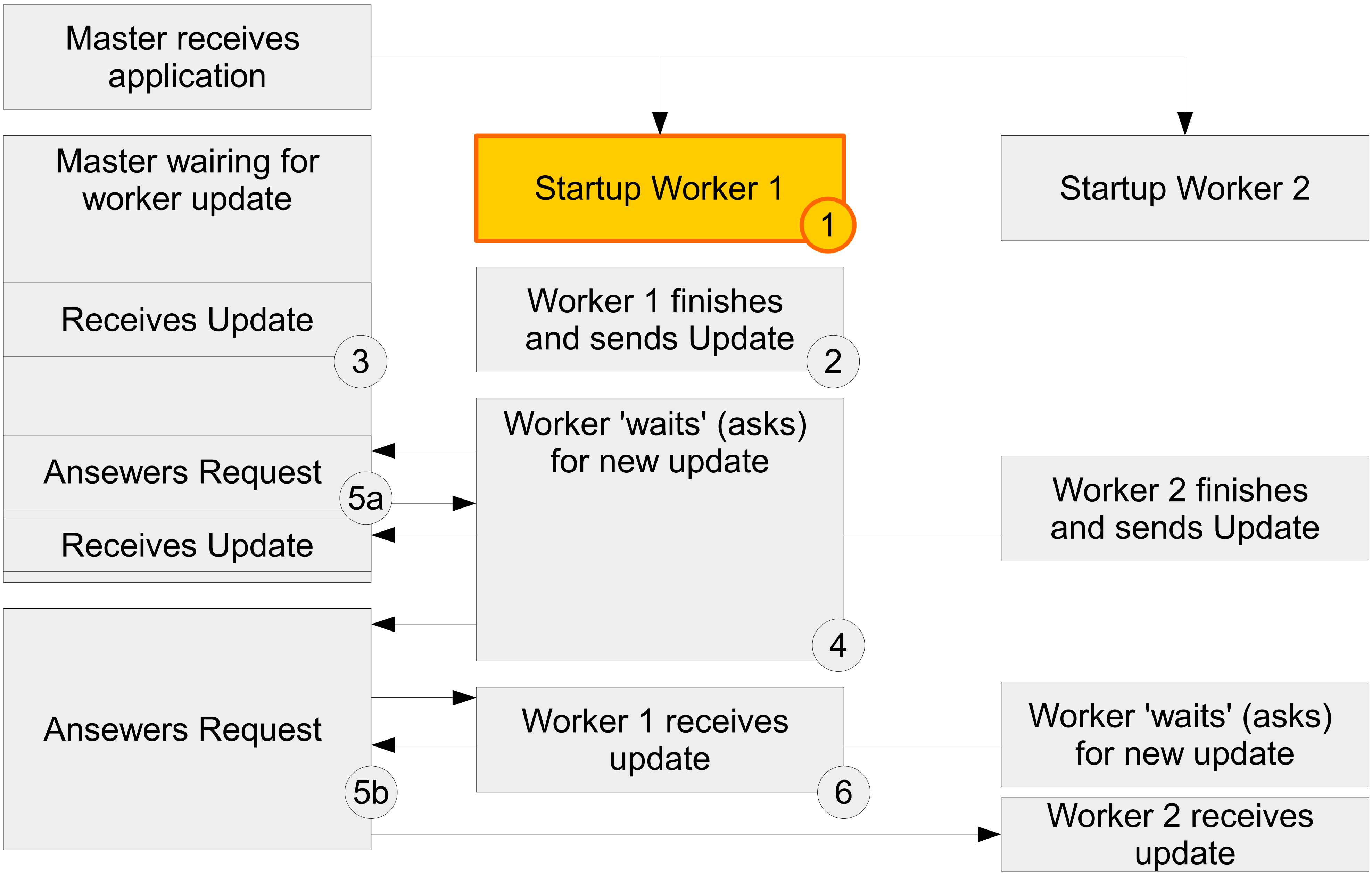
BUT HOW IS THIS CODE DISTRIBUTED?

- So far it is not – only the datasets and interations are.
- Lets take a look at the source code to proof this.

DISTRIBUTION ON MAPREDUCE (NOW DECRAPITATED)



DISTRIBUTION ON MAPREDUCE (NOW DECRAPITATED)



SOURCE CODE (MR/YARN)

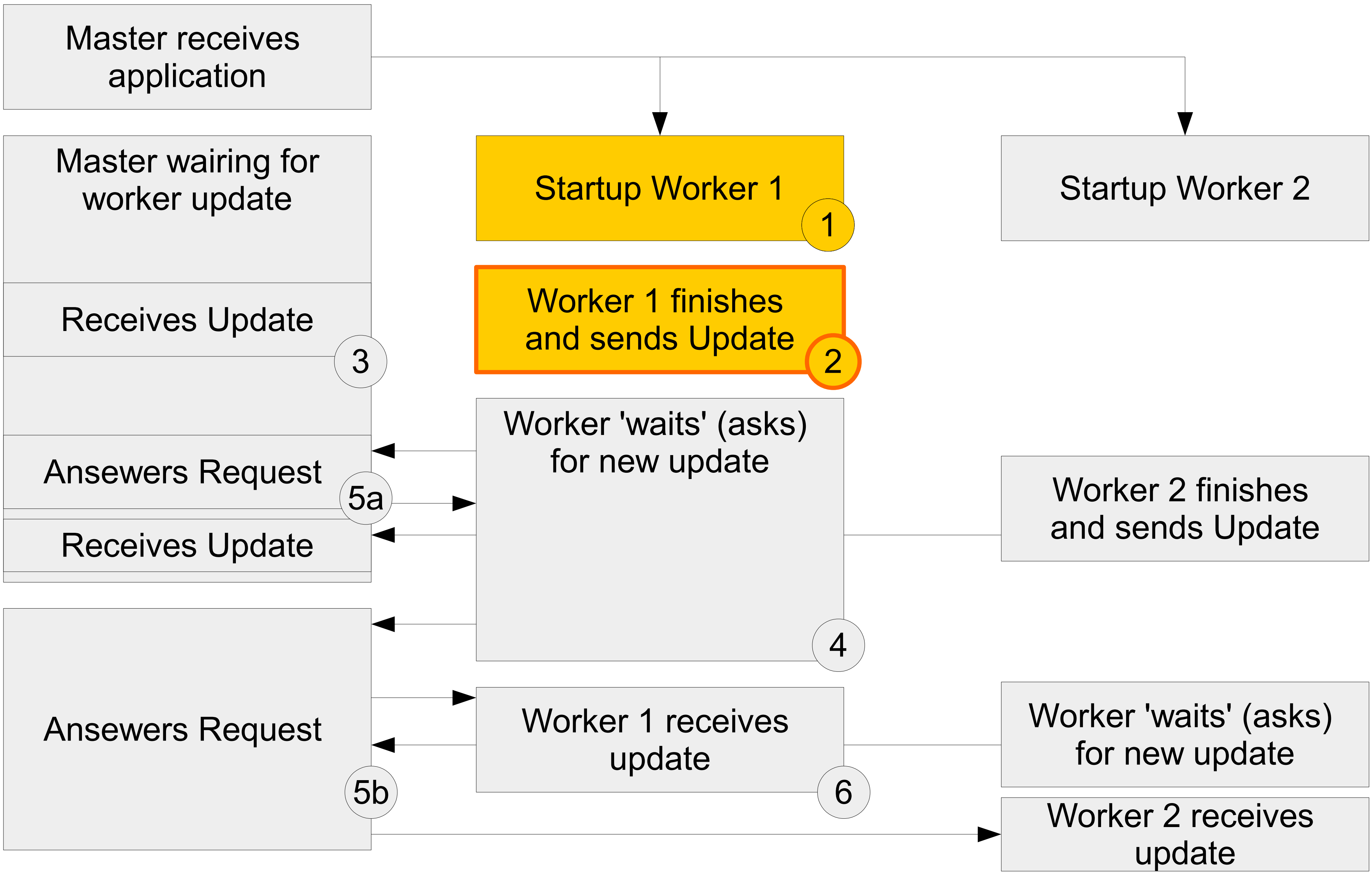
Worker main function after it has been first started:

```
// Do some work
currentState = WorkerState.STARTED;
//LinkedList<T> records = new LinkedList<T>();

int countTotal = 0;
int countCurrent = 0;
int currentIteration = 0;
int lastUpdate = 0;

computable.setRecordReader(recordParser);

for (currentIteration = 0; currentIteration < workerConf.getIterations(); currentIteration++) {
  //while (doIterations) {
  LOG.debug("Beginning iteration " + (currentIteration +1) + "/" + workerConf.getIterations());

  synchronized (currentState) {
    currentState = WorkerState.RUNNING;
  }
```

```
long mWorkerStart = System.currentTimeMillis();
T workerUpdate = computable.compute();

mWorkerExecutions++;
mWorkerTime += (System.currentTimeMillis() - mWorkerStart);
```

The worker initializeses and computes something during the first iteration with the given model on its data, the result is some kind of update.

DISTRIBUTION ON MAPREDUCE (NOW DECRAPITATED)



SOURCE CODE (MR/YARN)

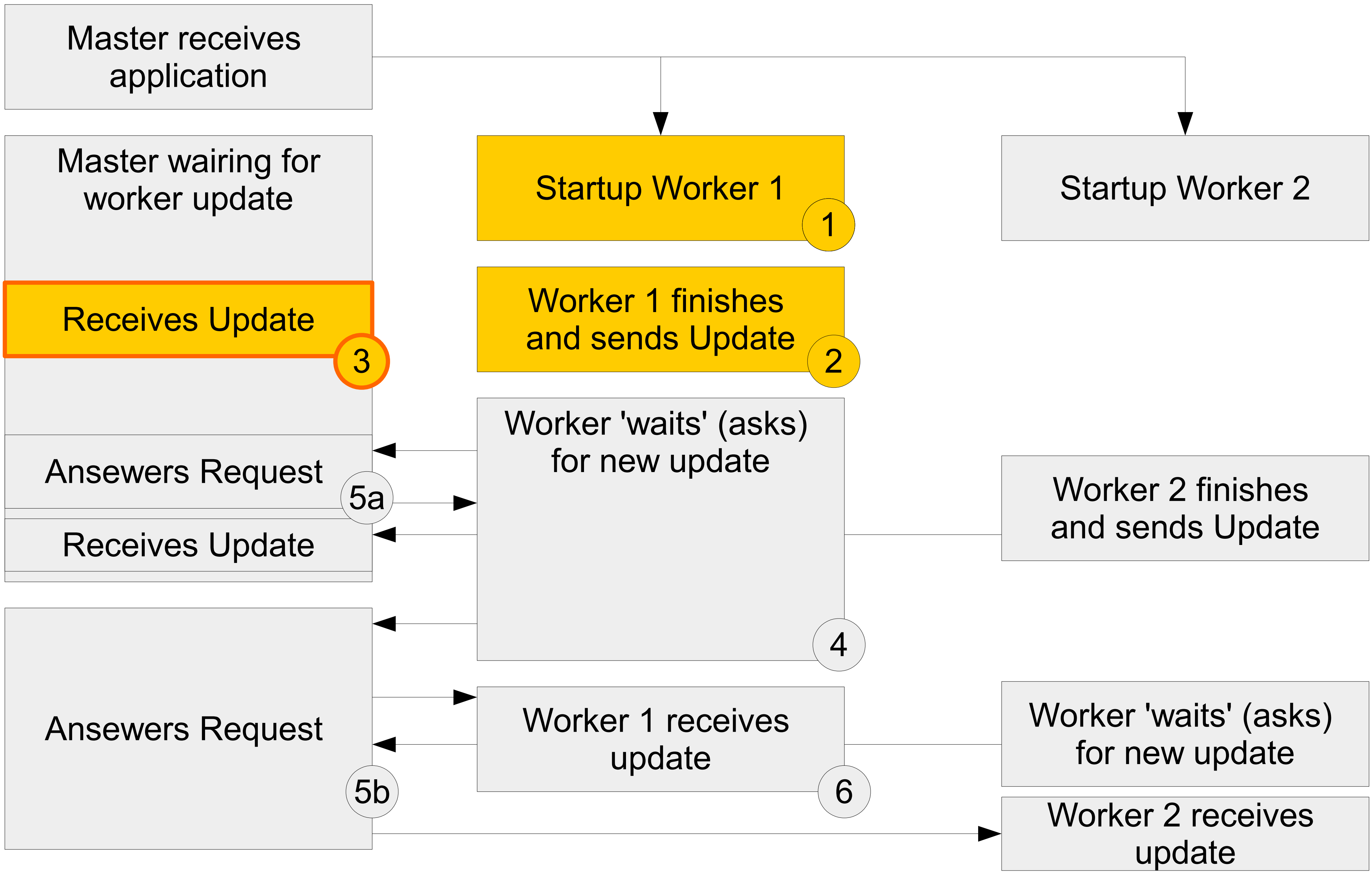
Worker main function after it has been first started:

```
try {
  synchronized (currentState) {
    ByteBuffer bytes = workerUpdate.toBytes();
    bytes.rewind();

    LOG.info("Sending an update to master");
    currentState = WorkerState.UPDATE;
    if (!masterService.update(workerId, bytes))
      LOG.warn("The master rejected our update");

    mUpdates++;
  }
} catch (AvroRemoteException ex) {
  LOG.error("Unable to send update message to master", ex);
  return -1;
}

// Wait on master for an update
int nextUpdate;
```

We continue here two slides later

The worker the worker then sends the update back to the master.

DISTRIBUTION ON MAPREDUCE (NOW DECRAPITATED)



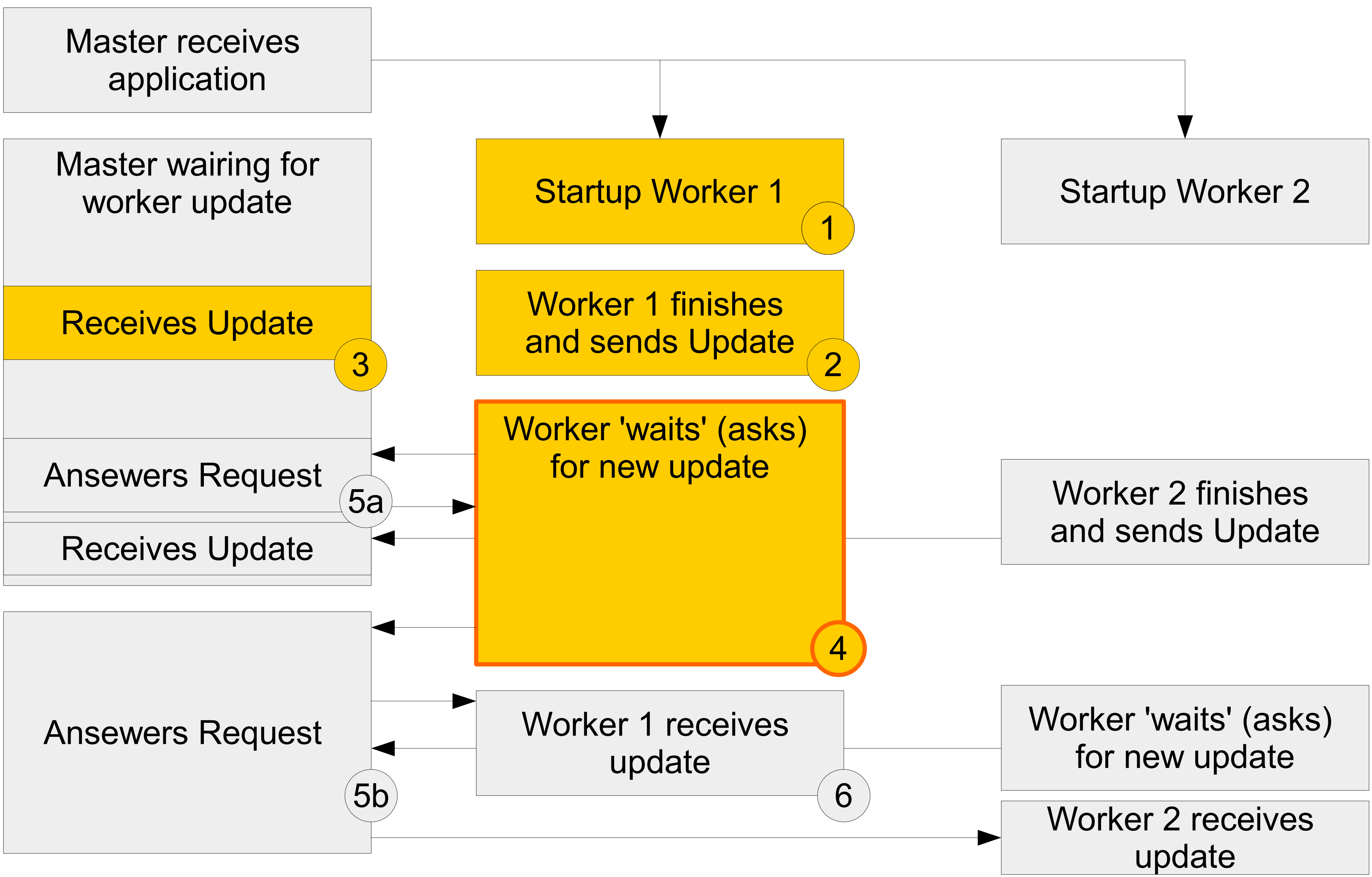
SOURCE CODE (MR/YARN)

```
synchronized (workersState) {
  workersUpdate.put(workerId, update);
  workersState.put(workerId, WorkerState.UPDATE);

  // Our latch should have the number of currently active workers
  if (workersUpdate.size() == expectedUpdates.get()) {
    LOG.info("Received updates from all workers, spawning local compute thread");

    // Fire off thread to compute update
    // TODO: need to fix this to something more reusable
    Thread updateThread = new Thread(new Runnable() {

      @Override
      public void run() {
        long startTime, endTime;

        startTime = System.currentTimeMillis();
        T result = computable.compute(workersUpdate.values(),
                masterUpdates.values());
        endTime = System.currentTimeMillis();

        LOG.info("Computed local update in " + (endTime - startTime) + "ms");
        expectedUpdates.set(workersCompleted.getCount());
```

Just fullfilled if all workers
Are done with the current iteration

Once the master receives an update from the worker and checks if all of the workers have finished the iterated (that is sent updates). If so, it starts integrating the immidiate results and updates the internal update count.

DISTRIBUTION ON MAPREDUCE (NOW DECRAPITATED)



SOURCE CODE (MR/YARN)

Worker after having sent an update (within its main function)

```
// Wait on master for an update
int nextUpdate;

try {
  LOG.info("Completed a batch, waiting on an update from master");
  nextUpdate = waitOnMasterUpdate(lastUpdate);

} catch (InterruptedException ex) {
  LOG.warn("Interrupted while waiting on master", ex);
  return -1;
} catch (AvroRemoteException ex) {
  LOG.error("Got an error while waiting on updates from master", ex);
  return -1;
}

// Time to get an update
try {
  ByteBuffer b = masterService.fetch(workerId, nextUpdate);
  b.rewind();
  T masterUpdate = updateable.newInstance();
  masterUpdate.fromBytes(b);
  computable.update(masterUpdate);
```

Loop with Timer inside

Gets data for next iteration

The worker goes over to wait for a next update ...

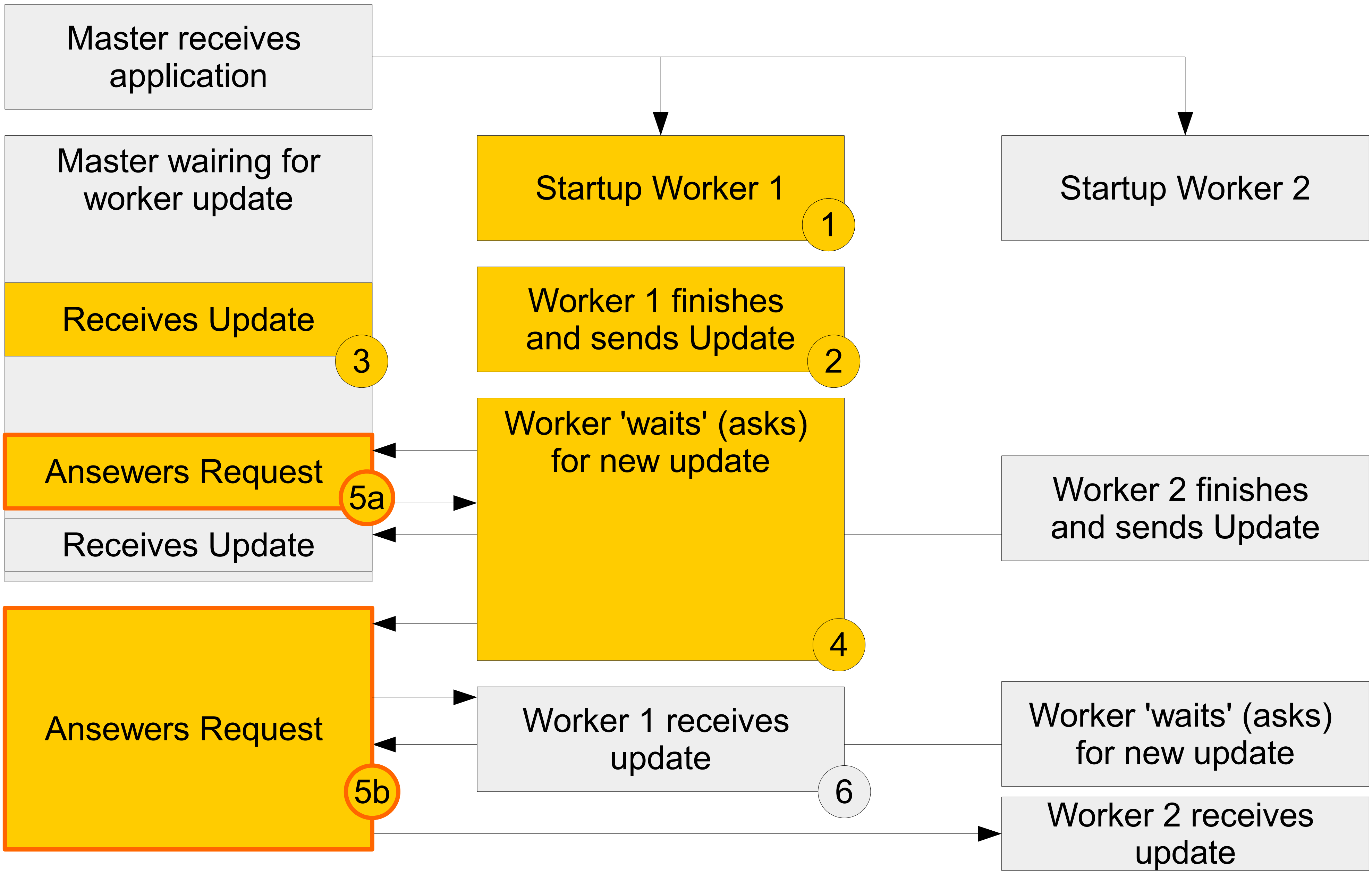
SOURCE CODE (MR/YARN)

Worker after having sent an update (waitonmasterupdate)

```
private int waitOnMasterUpdate(int lastUpdate) throws InterruptedException,
        AvroRemoteException {
  int nextUpdate = 0;
  long waitStarted = System.currentTimeMillis();
  long waitingFor = 0;

  while ((nextUpdate = masterService
          .waiting(workerId, lastUpdate, waitingFor)) < 0) {

    synchronized (currentState) {
      currentState = WorkerState.WAITING;
    }

    Thread.sleep(updateSleepTime);
    waitingFor = System.currentTimeMillis() - waitStarted;

    LOG.info("Waiting on update from master with lastID " + lastUpdate + " for " + waitingFor + "
    
    mWaits++;
  }

  mWaitTime += waitingFor;

  return nextUpdate;
}
```

... and it continutes to send periodical wait requests to the master service until it receivs a positive number as next update number.

DISTRIBUTION ON MAPREDUCE (NOW DECRAPITATED)



SOURCE CODE (MR/YARN)

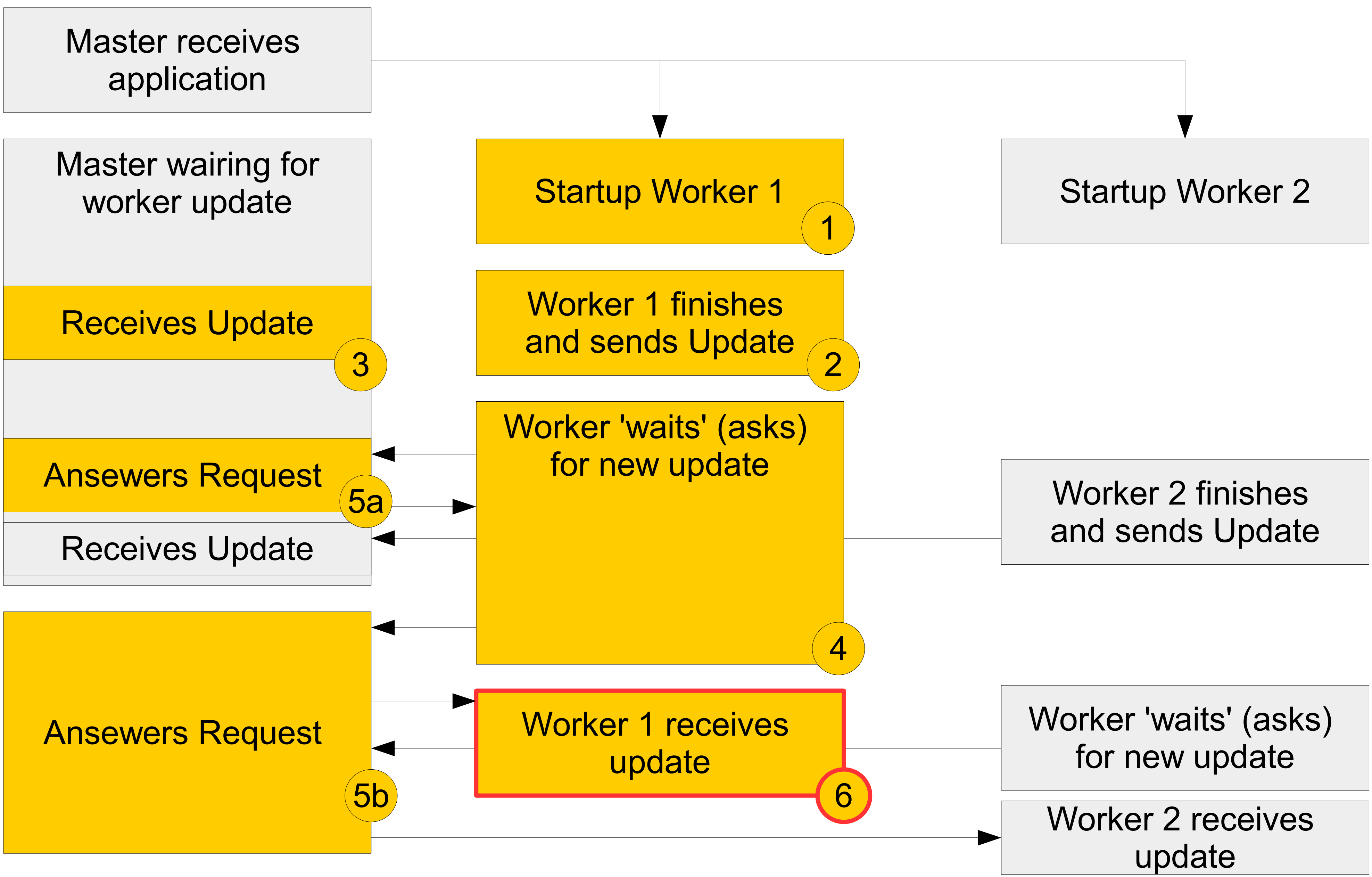
MasterService.waiting function (on master, triggered remotely by worker)

```
@Override
public int waiting(WorkerId workerId, int lastUpdate, long waiting)
        throws AvroRemoteException {

  synchronized (workersState) {
    workersState.put(workerId, WorkerState.WAITING);

    LOG.info("Got waiting message" + ", workerId="
            + Utils.getWorkerId(workerId) + ", workerState="
            + workersState.get(workerId) + ", lastUpdate=" + lastUpdate
            + ", currentUpdateId=" + currentUpdateId
            + ", waitingFor=" + waiting);
  }

  if (MasterState.UPDATING == masterState && lastUpdate == currentUpdateId)
    return -1;

  return currentUpdateId;
}
```

Only interesting if there
Is a new update

The master service registeres the worker as waiting and gives back the current update Id.

DISTRIBUTION ON MAPREDUCE (NOW DECRAPITATED)



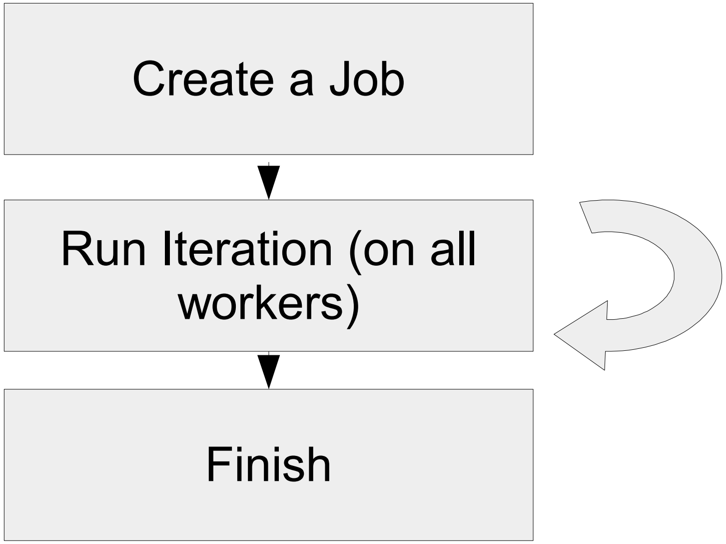
SOURCE CODE (MR/YARN)

```java
@Override
public ByteBuffer fetch(WorkerId workerId, int updateId)
        throws AvroRemoteException {

  LOG.info("Received a fetch request"
          + ", workerId=" + Utils.getWorkerId(workerId)
          + ", requestedUpdateId=" + updateId);

  synchronized (workersState) {
    workersState.put(workerId, WorkerState.RUNNING);
  }

  ByteBuffer bytes = masterUpdates.get(updateId).toBytes();
  bytes.rewind();

  return bytes;
}
```

Goes back to worker main function

If there is a new update, it fetches some kind of data (most probably related to the model parameters such as weights) from the master server for the current update.

DISTRIBUTION ON SPARK



Similiar to Mapreduce/Yarn, only that do not have to configure the worker separately.

SOURCE CODE (FOR SPARK)

Within a spark job for training a dataset on a multilayer (neural) network. - on **master**

```
public MultiLayerNetwork fitDataSet(JavaRDD<DataSet> rdd) {
    int iterations = conf.getConf(0).getNumIterations();
    log.info("Running distributed training:  (averaging each iteration = " + averageEachIteration + "
            iterations + "), (num partions = " + rdd.partitions().size() + ")");
    if(!averageEachIteration) {
        //Do multiple iterations and average once at the end
        runIteration(rdd);
    } else {
        //Temporarily set numIterations = 1. Control numIterations externall here so we can average b
        for(NeuralNetConfiguration conf : this.conf.getConfs()) {
            conf.setNumIterations(1);
        }

        //Run learning, and average at each iteration
        for(int i = 0; i < iterations; i++) {
            runIteration(rdd);
        }

        //Reset number of iterations in config
        if(iterations > 1 ){
            for(NeuralNetConfiguration conf : this.conf.getConfs()) {
                conf.setNumIterations(iterations);
            }
        }
    }
```

The master service registeres the worker as waiting and gives back the current update Id.

Source Code (Spark Integration)

Run an iteration ... (1)

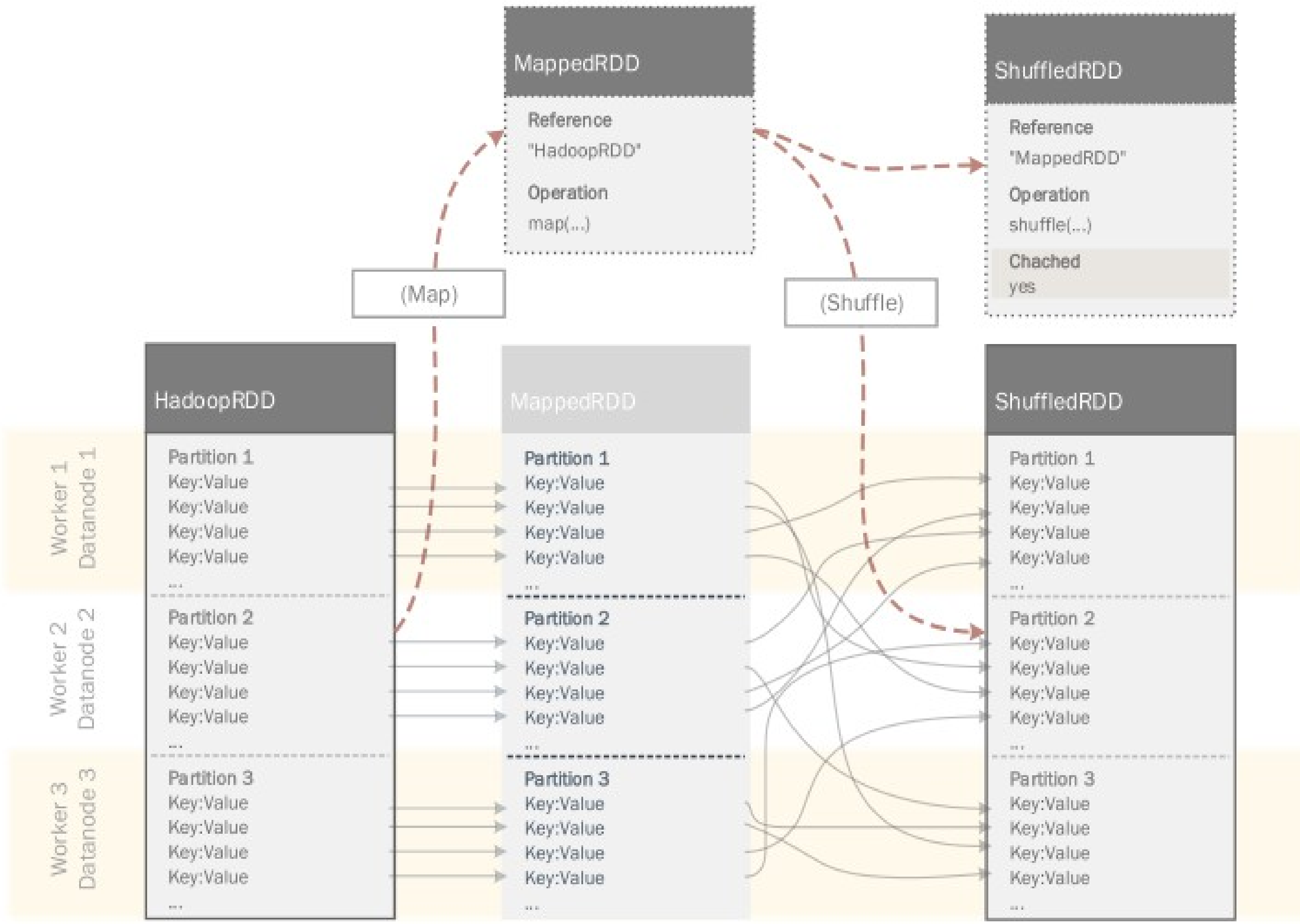
```
protected void runIteration(JavaRDD<DataSet> rdd) {
    int maxRep = 0;
    long maxSm = 0;
    int paramsLength = network.numParams(false);

    log.info("Broadcasting initial parameters of length " + paramsLength);

    INDArray valToBroadcast = network.params(false);
    this.params = sc.broadcast(valToBroadcast);
    Updater updater = network.getUpdater();
    if(updater == null) {
        network.setUpdater(UpdaterCreator.getUpdater(network));
        log.warn("Unable to propagate null updater");
        updater = network.getUpdater();
    }
    this.updater = sc.broadcast(updater);

    boolean accumGrad = sc.getConf().getBoolean(ACCUM_GRADIENT, false);
    if(accumGrad) {
        //Learning via averaging gradients
        JavaRDD<Tuple3<Gradient,Updater, ScoreReport>> results = rdd.mapPartitions(new GradientAccumF

        JavaRDD<Gradient> resultsGradient = results.map(new GradientFromTupleFunction());
        log.info("Ra                  e... averaging results now.");

        GradientAdder a = new GradientAdder(paramsLength);
        resultsGradient.foreach(a);
```

Seems like we are spreading some params and an 'updater' to the workers.

RDD?

RDD's?

We know them already! - Quick lookback:



Left: The initial outlook after reading data from the HDFS. The data from each DataNode will be loaded into local Memory partitions, each Block corresponds to one partition. Middle: After applying some function (one to one mapping), the outcome is not computed but only the nature of the function and the reference to the source RDD is saved (upper row). Right: A function requiring a shuffle is executed, that means the data is differently distributed among the worker nodes. Locality is lost and if we want to work with HadoopRDD and ShuffledRDD in combination, both need to be kept in Memory. Usually after a shuffle, the current RDD is cached, that is evaluated and stored in memory for future use. The key value pairs depicted here could be any serializable object.

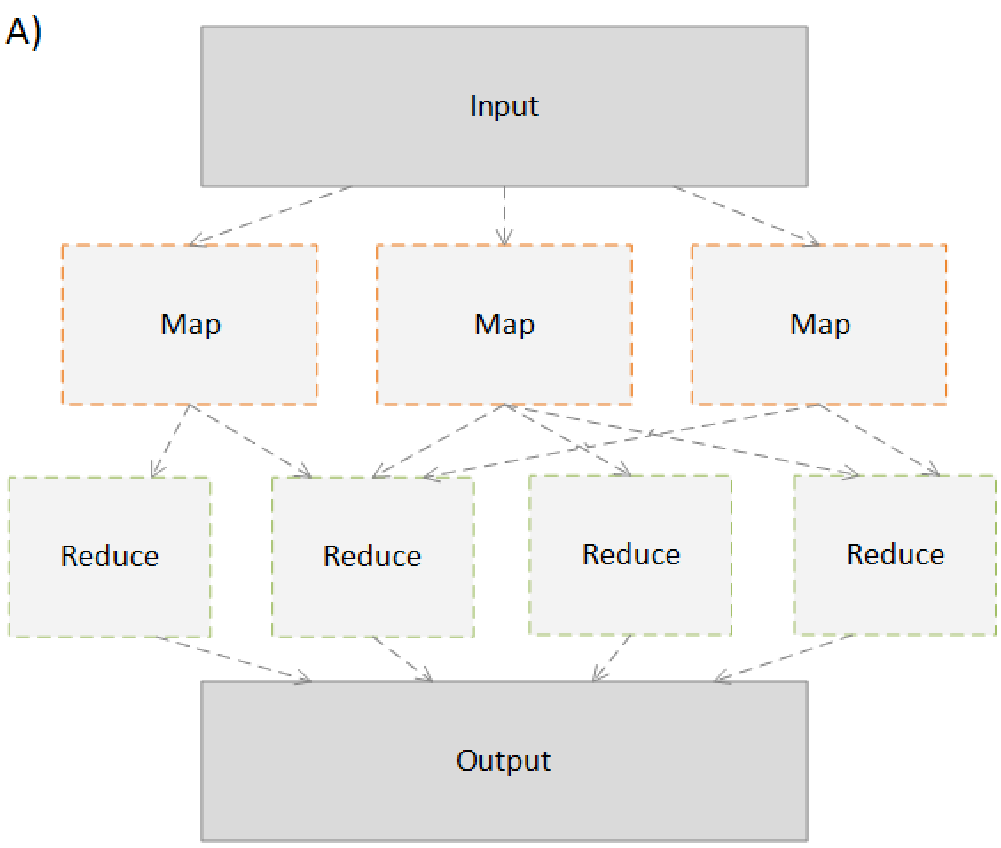
Source Code (Spark Integration)

Run an iteration ... (2)

```
GradientAdder a = new GradientAdder(paramsLength);
resultsGradient.foreach(a);
INDArray accumulatedGradient = a.getAccumulator().value();
boolean divideGrad = sc.getConf().getBoolean(DIVIDE_ACCUM_GRADIENT,false);
if(divideGrad) {
    maxRep = results.partitions().size();
    accumulatedGradient.divi(maxRep);
}
log.info("Accumulated parameters");
log.info("Summed gradients.");
network.setParameters(network.params(false).addi(accumulatedGradient));
log.info("Set parameters");
JavaDoubleRDD scores = results.mapToDouble(new ScoreMappingG());
lastScore = scores.mean();
if (!initDone) {
    JavaDoubleRDD sm = results.mapToDouble(new SMappingG());
    maxSm = sm.mean().longValue();
}

log.info("Processing updaters");
JavaRDD<Updater> resultsUpdater = results.map(new UpdaterFromGradientTupleFunction());

UpdaterAggregator aggregator = resultsUpdater.aggregate(
        resultsUpdater.first().getAggregator(false),
```

... and do even more parallel stuff. In the end integrate the results (from all workers) – that is, we also have to wait here until each worker has finished.

Iterative Reduce



Iterative Reduce (2)

- Does this even make sense if we need to wait always for the completion of all worker nodes?

Stan Kladko, Co-Founder of GalacticExchange, a VC-backed Deep Learning startup

2.6k Views

No.

Spark is significantly inefficient for deep learning and is not so easy to learn.

Implementing Deep Learning algorithms with Spark is awkward. So you get neither simplicity nor performance.

The best way to do Deep Learning is to use a GPU enabled library such as Theano.

Then if you want to build a cluster, use simple old-fashioned tools such as Torque/OpenMPI.

Written 18 Jan • View Upvotes

DL4J COMPONENTS

- Open Source Deep Learning Library

What is DeepLearning and which part of it is covered by DL4j? (Section 1)

- CPU & GPU support
- Hadoop-Yarn (MR) & Spark integration

What components are interfaced? (Section 2)

How are the alogrithms distributed? (Section 3)

- Futre additions

Section 4 / The End.

OUTLOOK

- Google DistBelief + Apache Horn

Deeplearning4j Roadmap

These priorities have been set by what the Skymind has seen demand for among clients and open-source community members. Contributors are welcome to add features whose priority they deem to be higher.

High priority:

- CUDA rewrite for ND4J (under way)
- CPU optimizations (C++ backend)
- Hyperparameter optimization (underway, basics done: Arbiter)
- Parameter server
- Sparse support for ND4J
- Performance tests for network training vs. other platforms (and where necessary: optimizations)
- Performance tests for Spark vs. local (ditto)
- Building examples at scale

Medium priority:

- OpenCL for ND4J
- CTC RNN (for speech etc.)

GOOGLE DIST BELIEF

Dean (2012) - Large Scale Distributed Deep Networks

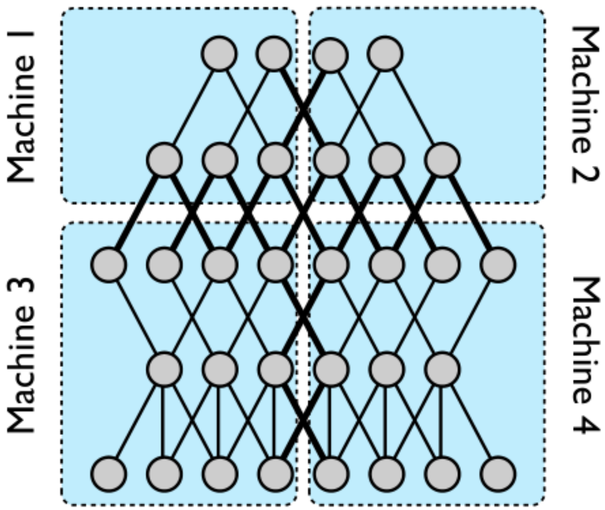


Figure 1: An example of model parallelism in DistBelief. A five layer deep neural network with local connectivity is shown here, partitioned across four machines (blue rectangles). Only those nodes with edges that cross partition boundaries (thick lines) will need to have their state transmitted between machines. Even in cases where a node has multiple edges crossing a partition boundary, its state is only sent to the machine on the other side of that boundary once. Within each partition, computation for individual nodes will the parallelized across all available CPU cores.

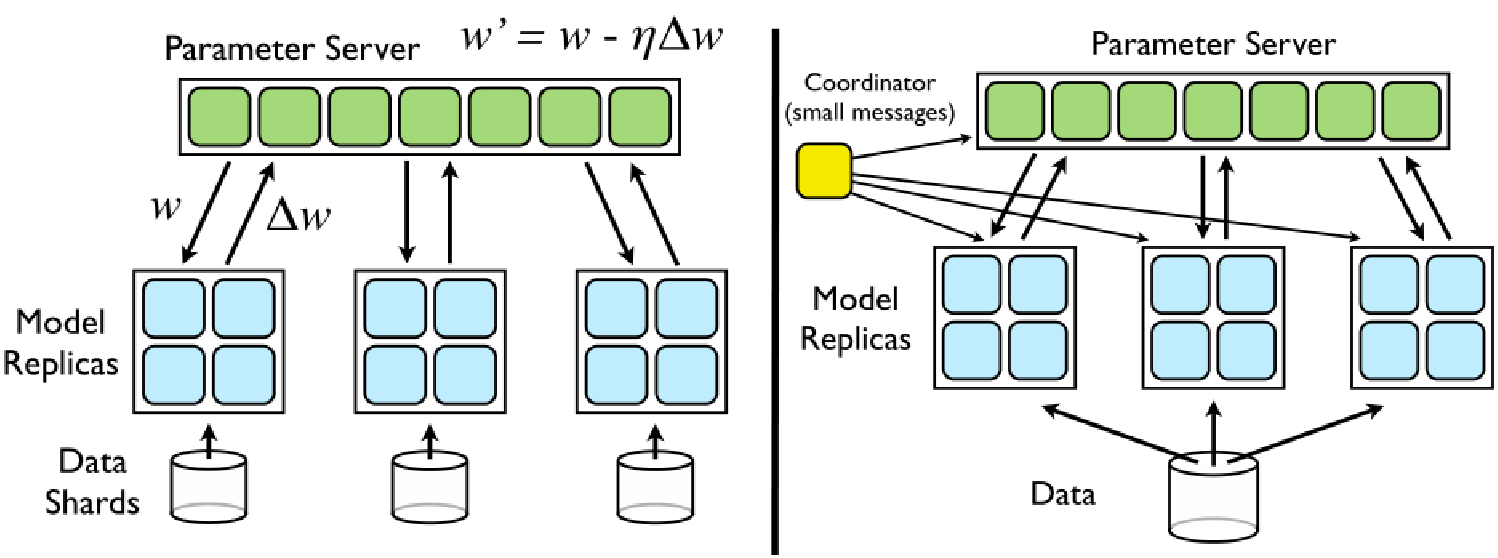


Figure 2: Left: Downpour SGD. Model replicas asynchronously fetch parameters w and push gradients ∆w to the parameter server. Right: Sandblaster L-BFGS. A single ‘coordinator’ sends small messages to replicas and the parameter server to orchestrate batch optimization.

2012, DistBelief by Jeff Dean (Google)

2013, Caffe by Yangqing jia (UC Berkeley)

2014, Deeplearning4J by Adam Gibson

2014, DeepDist by Dirk Neumann (Facebook)

...

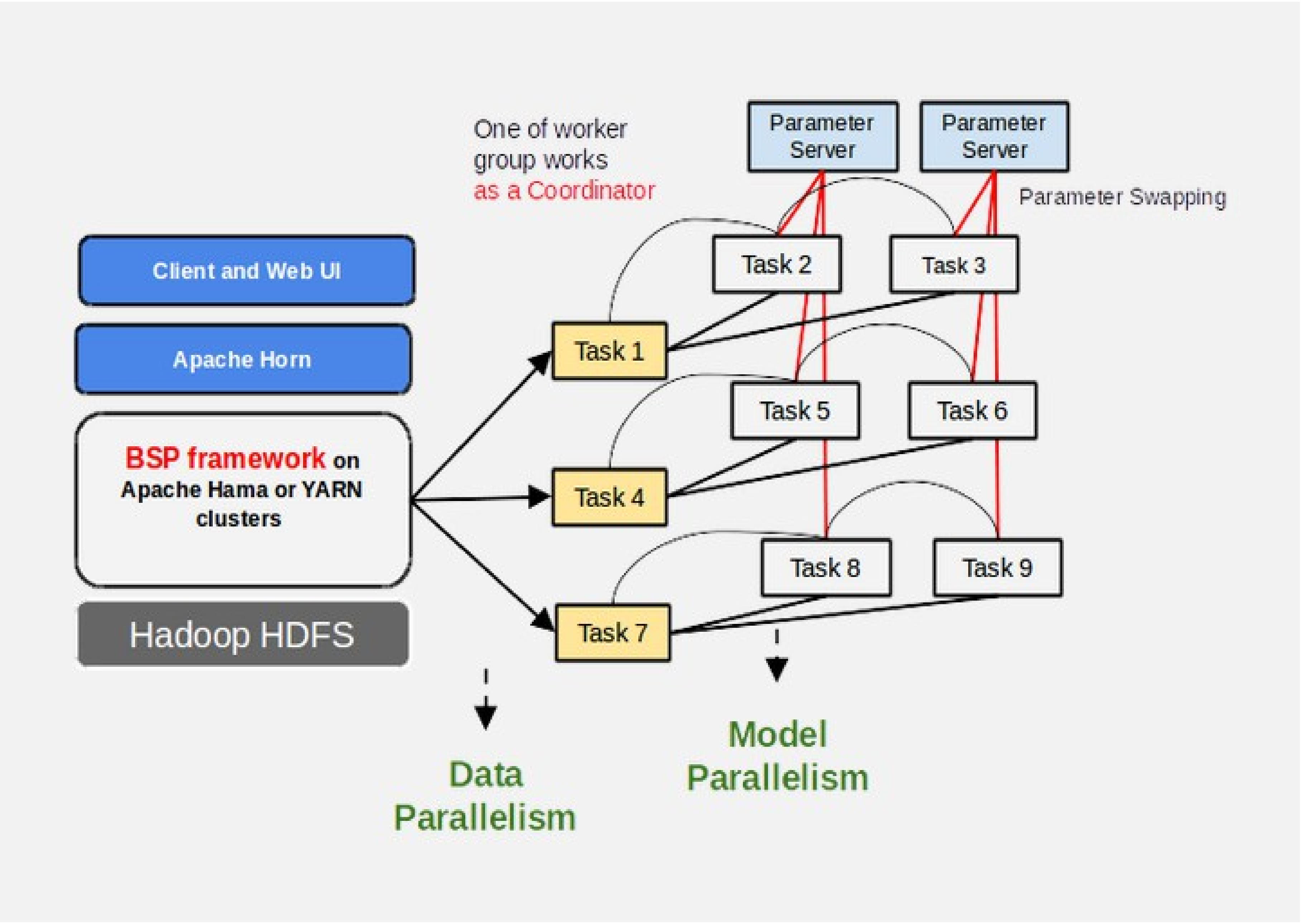
Taken from a Presentation by 윤진석

윤진석

Apache Horn

a Large-scale Deep Learning

Taken from a Presentation by 윤진석


One of worker group works as a Coordinator
Parameter Server
Parameter Server
Parameter Swapping
Client and Web UI
Apache Horn
BSP framework on Apache Hama or YARN clusters
Hadoop HDFS
Task 1
Task 2
Task 3
Task 4
Task 5
Task 6
Task 7
Task 8
Task 9
Data Parallelism
Model Parallelism