Lab – Deep Learning

Data Mining, Spring 2018

Carol (berm@itu.dk) | Daniel (dafr@itu.dk) | Mathias (jams@itu.dk)

IT UNIVERSITY OF COPENHAGEN

Today's Lab: Deep Learning Framework

Deep Learning Framework

- In today’s lab you will try your hand at the deep learning framework, DL4J
  - https://deeplearning4j.org/documentation

Deep Learning for Java

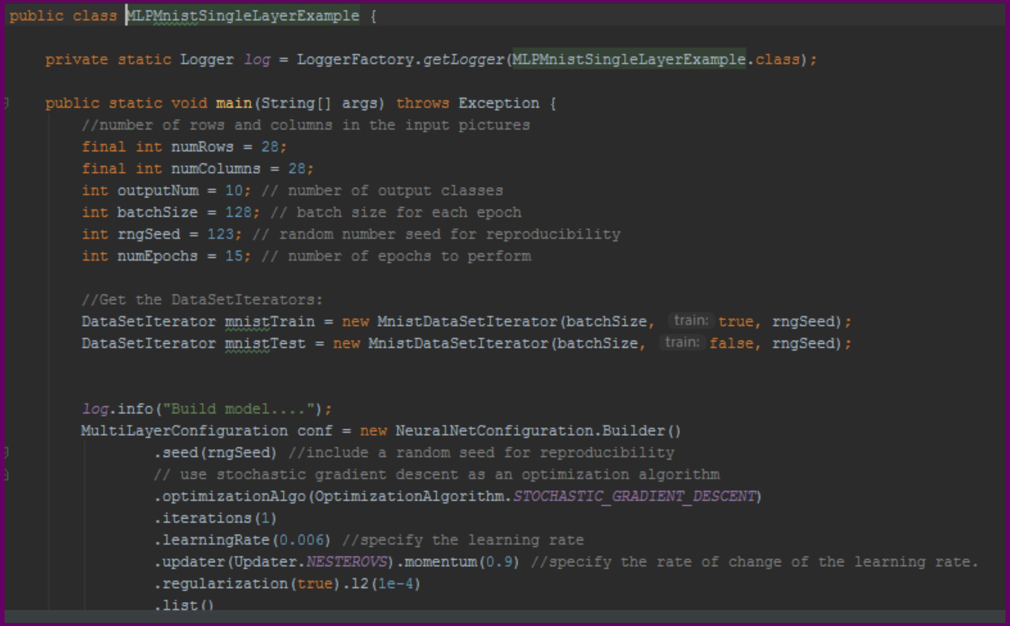
Open-Source, Distributed, Deep Learning Library for the JVM

Two part exercise

- Purpose: get experience with deep learning
- Part 1: Experiment with an existing CNN topology and get familiar with Deeplearning4j.
  - A very high accuracy should be achieved
  - Should be quite straight forward
- Part 2: See if you can create a well performing CNN topology yourself – hard

Part 1: DL4J - Code

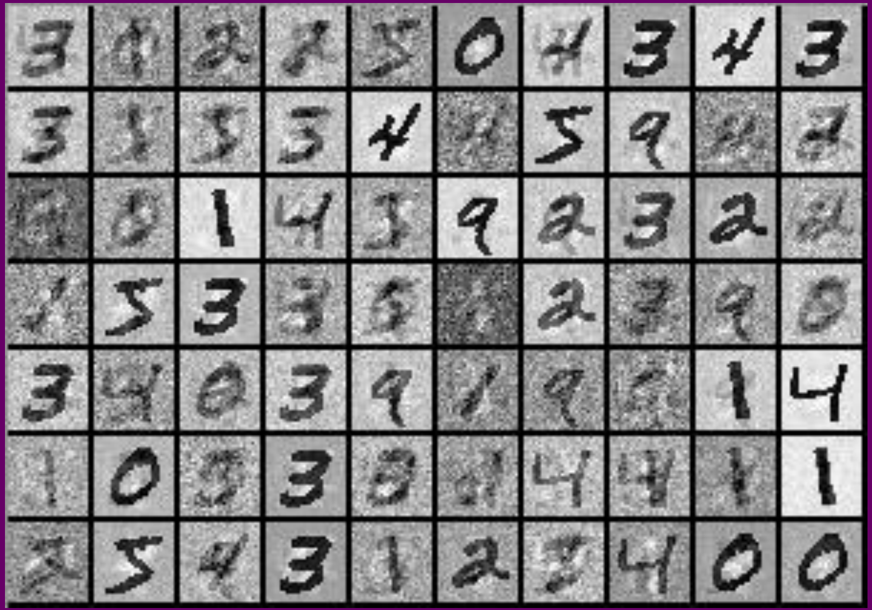
- All the code is already provided, with some default parameters and variables
- How to import and run the code can be found at https://deeplearning4j.org/mnist-for-beginners
  - Also contains documentation and information regarding the provided code
- All dependencies taken care of (using Maven)



```java
public class MLPMnistSingleLayerExample {

    private static Logger log = LoggerFactory.getLogger(MLPMnistSingleLayerExample.class);

    public static void main(String[] args) throws Exception {
        //number of rows and columns in the input pictures
        final int numRows = 28;
        final int numColumns = 28;
        int outputNum = 10; // number of output classes
        int batchSize = 128; // batch size for each epoch
        int rngSeed = 123; // random number seed for reproducibility
        int numEpochs = 15; // number of epochs to perform

        //Get the DataSetIterators:
        DataSetIterator mnistTrain = new MnistDataSetIterator(batchSize,  train: true, rngSeed);
        DataSetIterator mnistTest = new MnistDataSetIterator(batchSize,  train: false, rngSeed);


        log.info("Build model....");
        MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
                .seed(rngSeed) //include a random seed for reproducibility
                // use stochastic gradient descent as an optimization algorithm
                .optimizationAlgo(OptimizationAlgorithm.STOCHASTIC_GRADIENT_DESCENT)
                .iterations(1)
                .learningRate(0.006) //specify the learning rate
                .updater(Updater.NESTEROVS).momentum(0.9) //specify the rate of change of the learning rate.
                .regularization(true).l2(1e-4)
                .list()
```

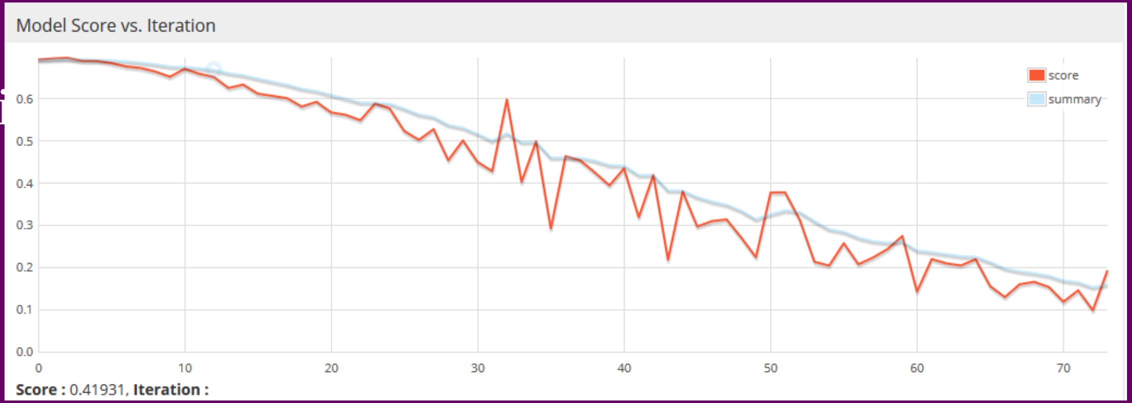
Part 1: Data

- MNIST dataset
  - Giant dataset containing images of handwritten integers (grayscale)
  - |TrainingSet| = 60,000
  - |TestSet| = 10,000
  - Input size: 28x28x1
  - Output size: 10



Part 1: Plan of Attack

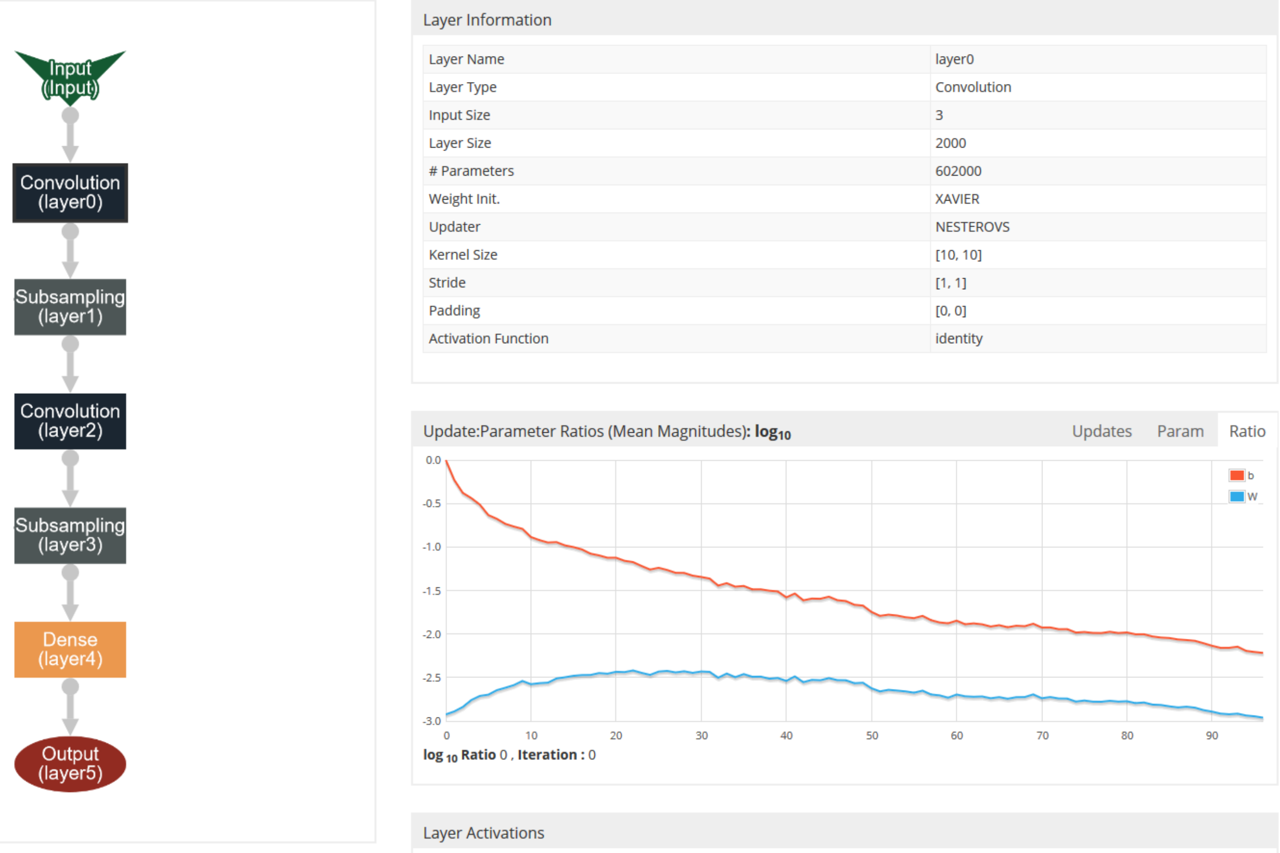
- Read the tutorial found on previous slide to understand the code
- Download the .zip
- Import the project in your IDE
- Run the code
- Fiddle around with parameters and values
- Get lowest score (in lowest iterati
- Visualize data


Model Score vs. Iteration

Score : 0.41931, **Iteration :**

Visualize the learning

- UI server can be accessed at: http://localhost:9000
- Visualize learning and model score
- Somewhat visualize the CNN being trained and the layers



Layer Information

Layer Name	layer0
Layer Type	Convolution
Input Size	3
Layer Size	2000
# Parameters	602000
Weight Init.	XAVIER
Updater	NESTEROVS
Kernel Size	[10, 10]
Stride	[1, 1]
Padding	[0, 0]
Activation Function	identity

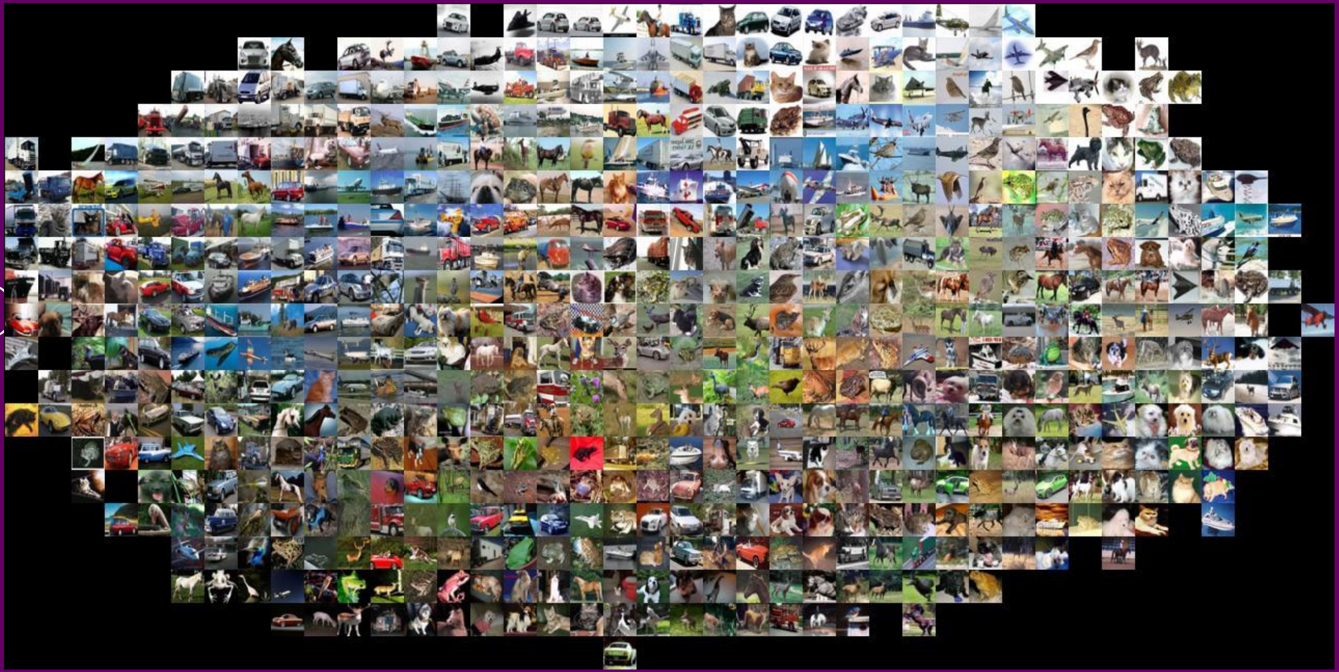
Update:Parameter Ratios (Mean Magnitudes): $\log_{10}$

$\log_{10}$ Ratio 0 , **Iteration :** 0

Layer Activations

Part 2: Design your own CNN

- Cifar10 dataset
- 60,000 images
- 10 distinct object categories
- Colors - 3 channels (RGB) – a whole new dimension



Part 2: Dataset and Preprocessing

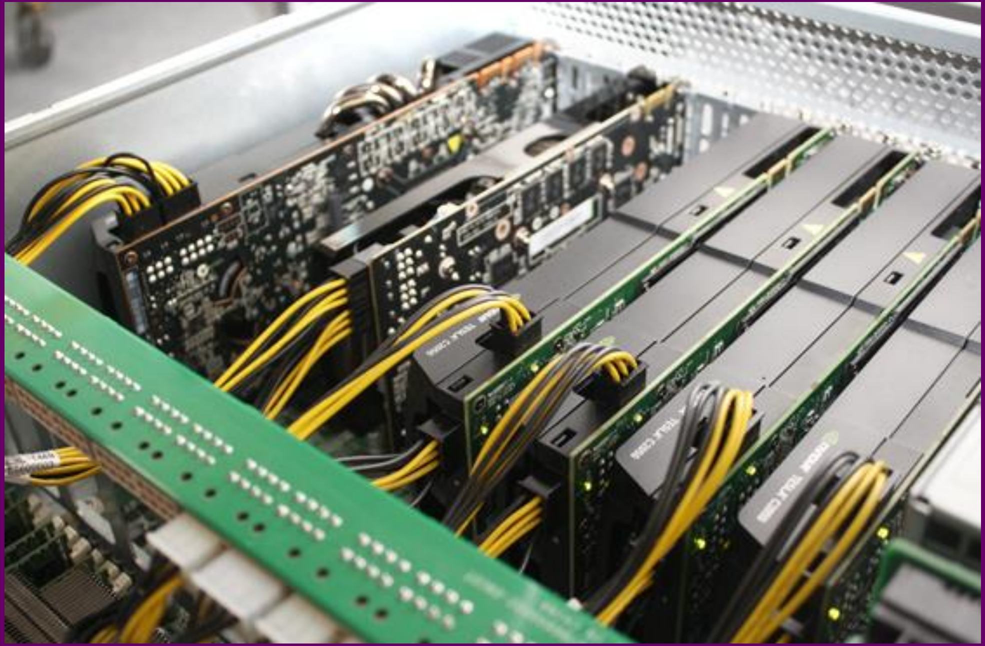
- Already preprocessed.
  - Preprocessing has a big influence on the result!
- Dataset split into test (10,000 images) and training (50,000 images)
- [3x32x32]
- All categories have around 5,000 examples – important!
- Normalize image data [0..1] – done when reading the images
- The dataset is provided

Part 2: Plan of attack

- Fill in the missing parts in Cifar10Example.java
- Design the CNN – get started here: https://deeplearning4j.org/convolutionalnets.html#dl4j-code-example
- Experiment with feature maps.
- Compare your accuracy with others https://www.kaggle.com/c/cifar-10/leaderboard

Performance

- The code provided does not utilize the GPU
- Can take a while to train – work on group projects meanwhile



Deep Learning Framework

- Optionally you can use either Keras or Tensorflow, which uses python
  - https://keras.io/
  - https://www.tensorflow.org/

Optional: Keras / Tensorflow

- Keras is simple, Tensorflow more complicated
  - Keras uses Tensorflow, meaning installation of Tensorflow is required to use Keras
  - Check https://www.tensorflow.org/install/ on how to install Tensorflow
- No code provided, but for Keras, check https://github.com/keras-team/keras/tree/master/examples for examples

Thanks for listening!