# Machine Learning with Java? Deeplearning4j!

Enrique Llerena Dominguez
mimacom

Platinum Sponsor

GRAND PARADE

# $whoami

- Enrique Llerena Dominguez
- Senior Software Engineer / Tech Lead @ mimacom
- Experience developing software for the finance, retail, pharma, and automotive industry

- Professional Interests: Software Architecture, Cloud Computing, Artificial Intelligence

- Free Time: Spending time with my kids, watching football, learning german and developing nice things

Twitter @ellerenad
Github @ellerenad
ellerenad@hotmail.com
ienjoysoftware.dev

Scan the QR code!


MIMACOM

# Agenda

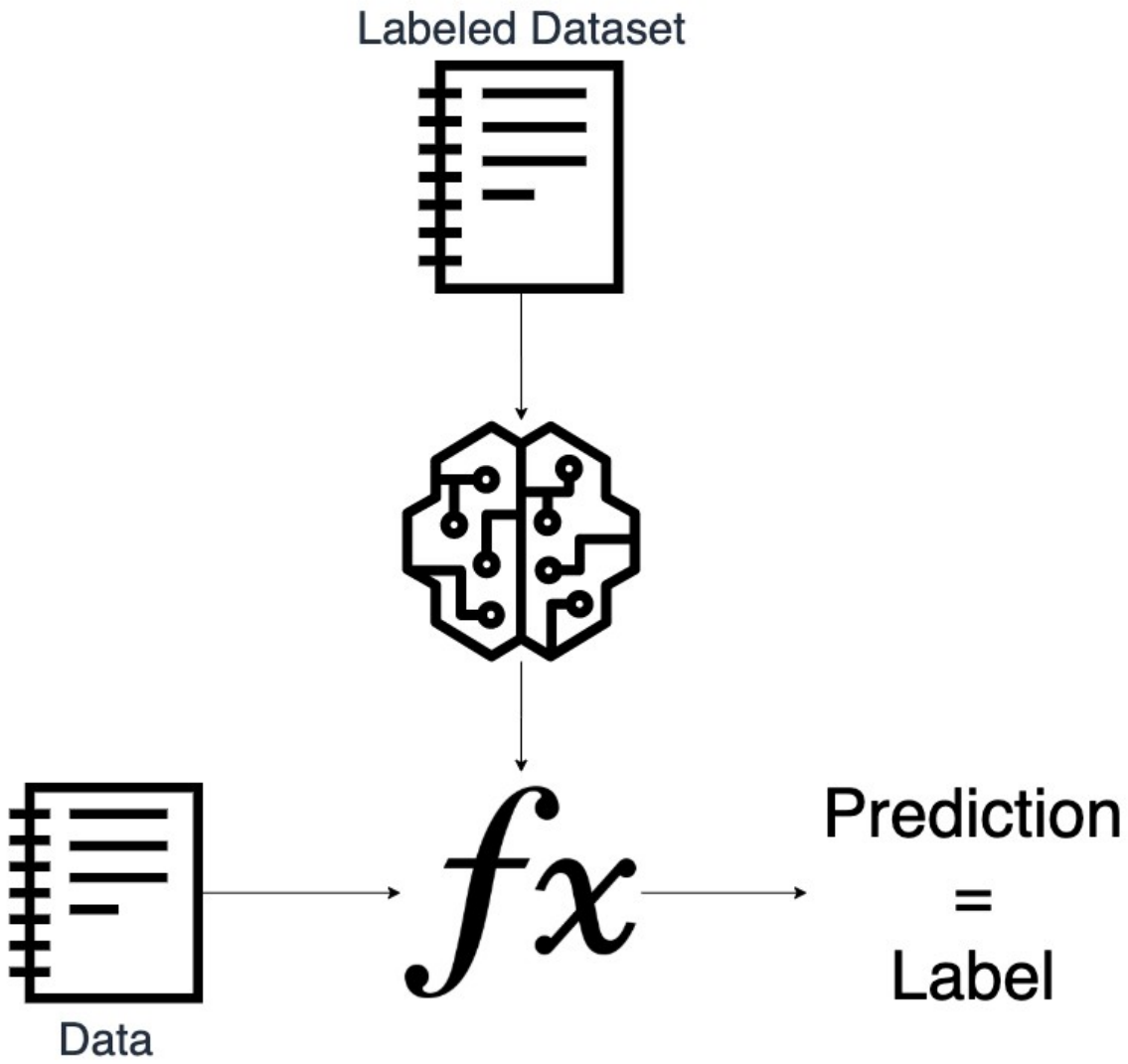
- Machine Learning 101 + deeplearning4j
- Deeplearning4j integrations
- Hands on examples:
  - Image classification – Approach and code
  - Sentiment analysis – Approach and code

# What is Machine Learning?

- Machine Learning is the study of computer algorithms that improve automatically through experience

Source: Mitchell, T. (1997), Machine Learning. http://www.cs.cmu.edu/~tom/mlbook.html

# What is Machine Learning?



# What is Deeplearning4j?

Open-source deep-learning library written for Java and Scala

# Story time!

Iris classification AKA the “Machine Learning Hello World”

Peter the biologist

Image source: Photo by Alex on Unsplash


Image source:
https://en.wikipedia.org/wiki/Gasp%C3%A9_Peninsula

Images source:
https://en.wikipedia.org/wiki/Iris_flower_data_set

Iris
Virginica

Iris
Setosa

Iris
Versicolor

Images source:
https://en.wikipedia.org/wiki/Iris_flower_data_set

Image source: Photo by Blaz Erzetic on Unsplash

Images source: Photo by Ben Mullins on Unsplash


Images source:Photo by Jarritos Mexican Soda on Unsplash


Image source: Photo by Alex on Unsplash


Teresa the software engineer, Peter’s daughter

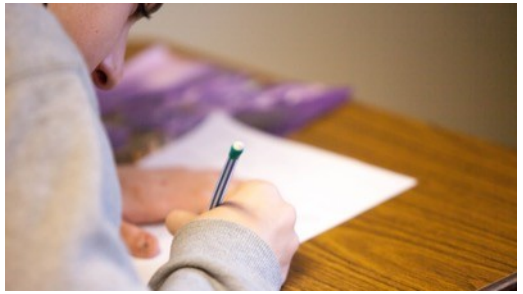
Photo by Jarritos Mexican Soda on Unsplash


Photo by Jarritos Mexican Soda on Unsplash


Photo by Jarritos Mexican Soda on Unsplash

Photo by NESA by Makers on Unsplash

# Resources



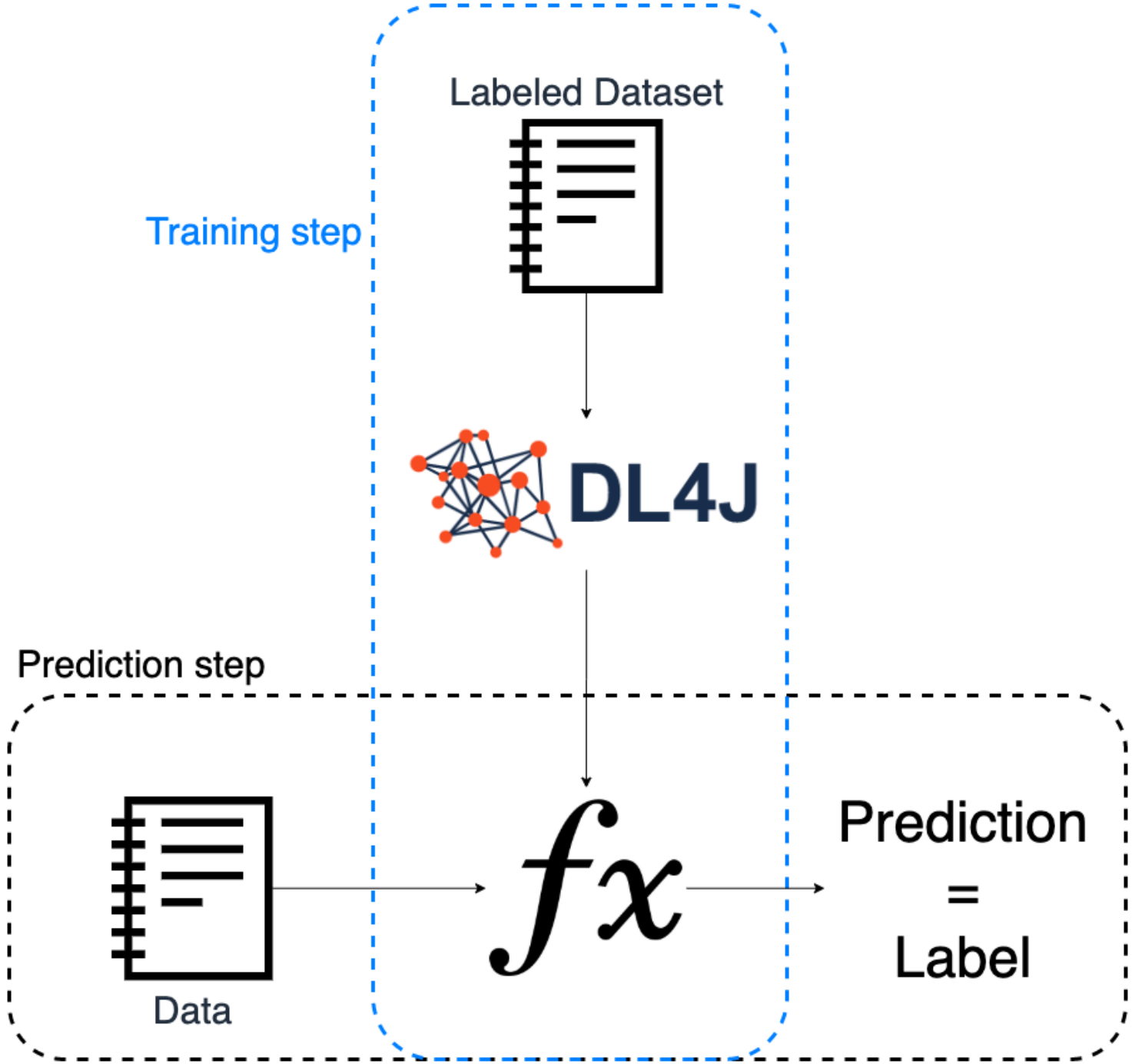
Labeled dataset



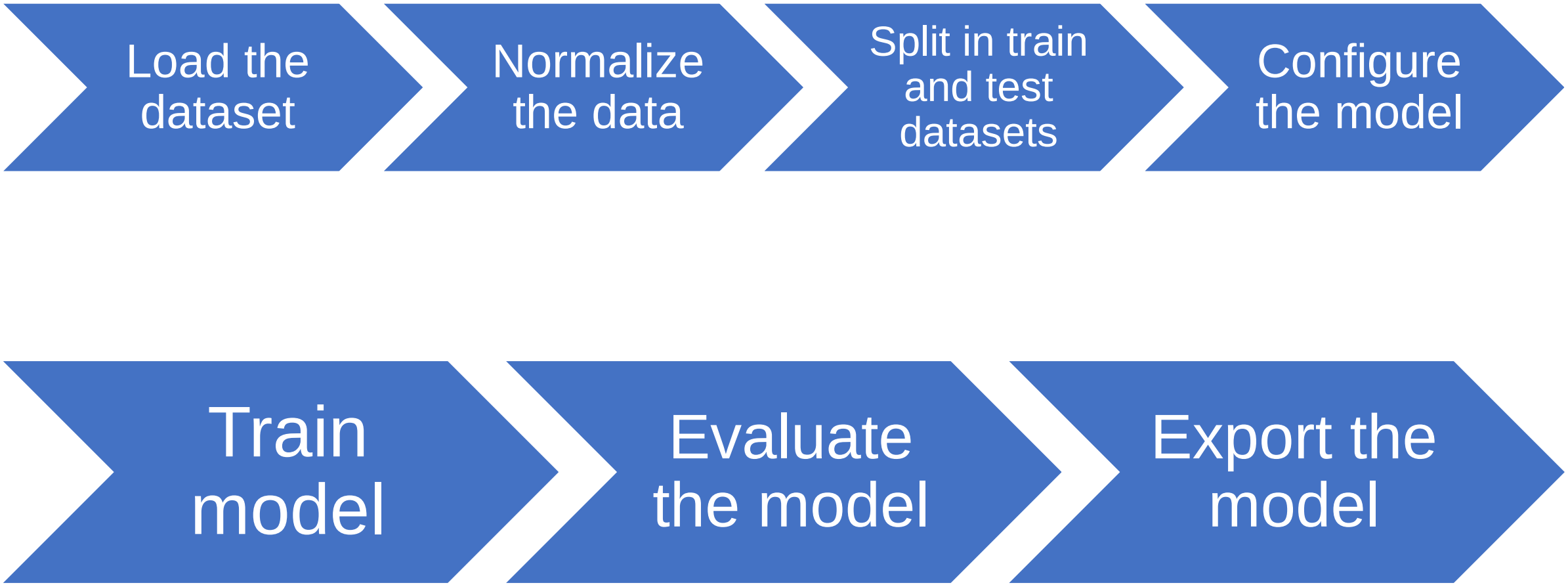
# Problem description

- Type: Classification
- Dataset:
  - Number of instances: 150
  - Number of attributes: 4
    - sepal length in cm
    - sepal width in cm
    - petal length in cm
    - petal width in cm
  - Number of classes: 3
    - Iris Setosa
    - Iris Versicolour
    - Iris Virginica

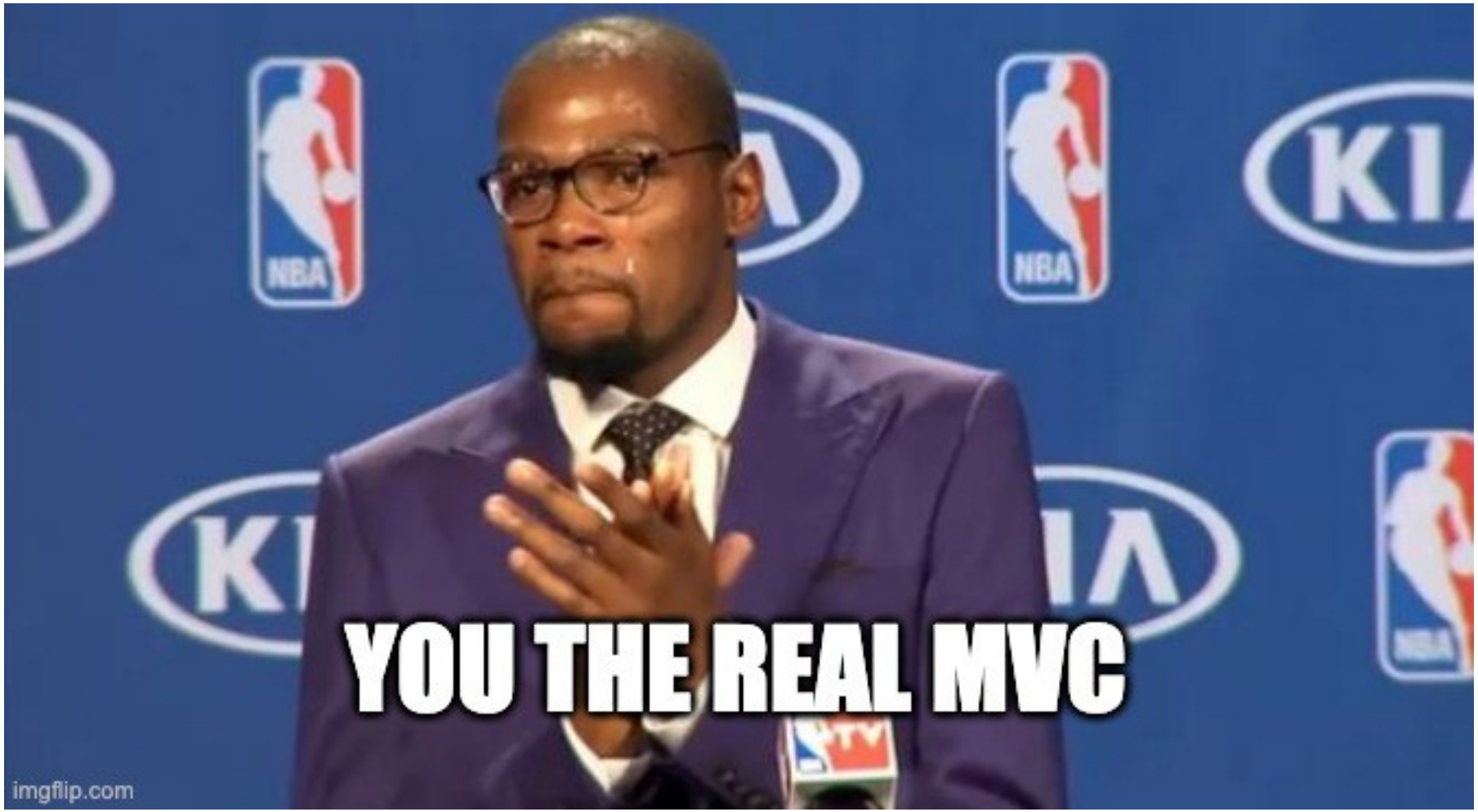
Source: University of California, Irvine , Iris Data Set. https://archive.ics.uci.edu/ml/datasets/iris


Labeled Dataset
Training step
DL4J
Prediction step
Data
fx
Prediction
=
Label

# Training step



# Iris Classifier Trainer Most Valuable Code




```
private static DataSet loadData(String path) throws IOException, InterruptedException {
        DataSet allData;
        try (RecordReader recordReader = new CSVRecordReader(0, ',')) {
            recordReader.initialize(new FileSplit(new File(path)));
            DataSetIterator iterator = new RecordReaderDataSetIterator(
                                        recordReader, TOTAL_LINES, LABEL_INDEX, CLASSES_COUNT);
            allData = iterator.next();
        }
        return allData;
    }
```

Load the dataset → Normalize the data → Split in train and test datasets → Configure the model → Train model → Evaluate the model → Export the model

```
private static DataSet loadData(String path) throws IOException, InterruptedException {
        DataSet allData;
        try (RecordReader recordReader = new CSVRecordReader(0, ',')) {
            recordReader.initialize(new FileSplit(new File(path)));
            DataSetIterator iterator = new RecordReaderDataSetIterator(
                                        recordReader, TOTAL_LINES, LABEL_INDEX, CLASSES_COUNT);
            allData = iterator.next();
        }
        return allData;
    }
```

Load the dataset → Normalize the data → Split in train and test datasets → Configure the model → Train model → Evaluate the model → Export the model

```
private static DataSet loadData(String path) throws IOException, InterruptedException {
        DataSet allData;
        try (RecordReader recordReader = new CSVRecordReader(0, ',')) {
            recordReader.initialize(new FileSplit(new File(path)));
            DataSetIterator iterator = new RecordReaderDataSetIterator(
                recordReader, TOTAL_LINES, LABEL_INDEX, CLASSES_COUNT);

            allData = iterator.next();
        }
        return allData;
    }
```


```
// Normalize the data
DataNormalization normalizer = new NormalizerStandardize();
normalizer.fit(allData); // Get stats about the data
normalizer.transform(allData); // Transform the data by applying the normalization
```


```
// Split in train and test datasets
SplitTestAndTrain testAndTrain = allData.splitTestAndTrain(TRAIN_TO_TEST_RATIO); // 65%
DataSet trainingData = testAndTrain.getTrain();
DataSet testData = testAndTrain.getTest();
```

```
private static MultiLayerConfiguration getMultiLayerConfiguration() {
    return new NeuralNetConfiguration.Builder()
            .seed(SEED)
            .activation(Activation.TANH)
            .weightInit(WeightInit.XAVIER)
            .updater(new Sgd(0.1))
            .l2(1e-4)
            .list()
            // The input layer must have FEATURES_COUNT = 4 nodes
            .layer(new DenseLayer.Builder().nIn(FEATURES_COUNT).nOut(3)
                    .build())
            .layer(new DenseLayer.Builder().nIn(3).nOut(3)
                    .build())
            .layer( new OutputLayer.Builder(LossFunctions.LossFunction.NEGATIVELOGLIKELIHOOD)
                    .activation(Activation.SOFTMAX)
                    .nIn(3)
                    // The output layer must have CLASSES_COUNT = 3 nodes
                    .nOut(CLASSES_COUNT).build())
            .build();
}
```

Load the dataset → Normalize the data → Split in train and test datasets → **Configure the model** → Train model → Evaluate the model → Export the model

```
private static MultiLayerConfiguration getMultiLayerConfiguration() {
    return new NeuralNetConfiguration.Builder()
            .seed(SEED)
            .activation(Activation.TANH)
            .weightInit(WeightInit.XAVIER)
            .updater(new Sgd(0.1))
            .l2(1e-4)
            .list()
            // The input layer must have FEATURES_COUNT = 4 nodes
            .layer(new DenseLayer.Builder().nIn(FEATURES_COUNT).nOut(3)
                    .build())
            .layer(new DenseLayer.Builder().nIn(3).nOut(3)
                    .build())
            .layer( new OutputLayer.Builder(LossFunctions.LossFunction.NEGATIVELOGLIKELIHOOD)
                    .activation(Activation.SOFTMAX)
                    .nIn(3)
                    // The output layer must have CLASSES_COUNT = 3 nodes
                    .nOut(CLASSES_COUNT).build())
            .build();
}
```

Load the dataset → Normalize the data → Split in train and test datasets → **Configure the model** → Train model → Evaluate the model → Export the model

```
private static MultiLayerConfiguration getMultiLayerConfiguration() {
    return new NeuralNetConfiguration.Builder()
            .seed(SEED)
            .activation(Activation.TANH)
            .weightInit(WeightInit.XAVIER)
            .updater(new Sgd(0.1))
            .l2(1e-4)
            .list()
            // The input layer must have FEATURES_COUNT = 4 nodes
            .layer(new DenseLayer.Builder().nIn(FEATURES_COUNT).nOut(3)
                    .build())
            .layer(new DenseLayer.Builder().nIn(3).nOut(3)
                    .build())
            .layer( new OutputLayer.Builder(LossFunctions.LossFunction.NEGATIVELOGLIKELIHOOD)
                    .activation(Activation.SOFTMAX)
                    .nIn(3)
                    // The output layer must have CLASSES_COUNT = 3 nodes
                    .nOut(CLASSES_COUNT).build())
            .build();
}
```

| sepal length | sepal width | petal length | petal width |
|---|---|---|---|
| 5,10 | 3,50 | 1,40 | 0,20 |

| Types |
|---|
| Iris Setosa |
| Iris Virginica |
| Iris Versicolor |

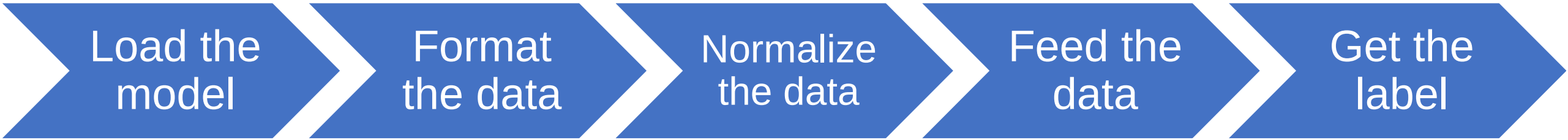
```
// Get configuration of the Neural Network
MultiLayerConfiguration configuration = getMultiLayerConfiguration();

// Train Neural Network
MultiLayerNetwork model = new MultiLayerNetwork(configuration);
model.init();
//Print score every 100 parameter updates
model.setListeners(new ScoreIterationListener(100));

// Do TRAIN_ITERATIONS = 1000 iterations to train the model
for(int x = 0; x < TRAIN_ITERATIONS; x++) {
    model.fit(trainingData);
}
```

```
private static Evaluation evaluate(MultiLayerNetwork model,
                                   DataSet testData) {
    INDArray output = model.output(testData.getFeatures());
    Evaluation eval = new Evaluation(CLASSES_COUNT); // 4 Classes
    eval.eval(testData.getLabels(), output);
    return eval;
}
```

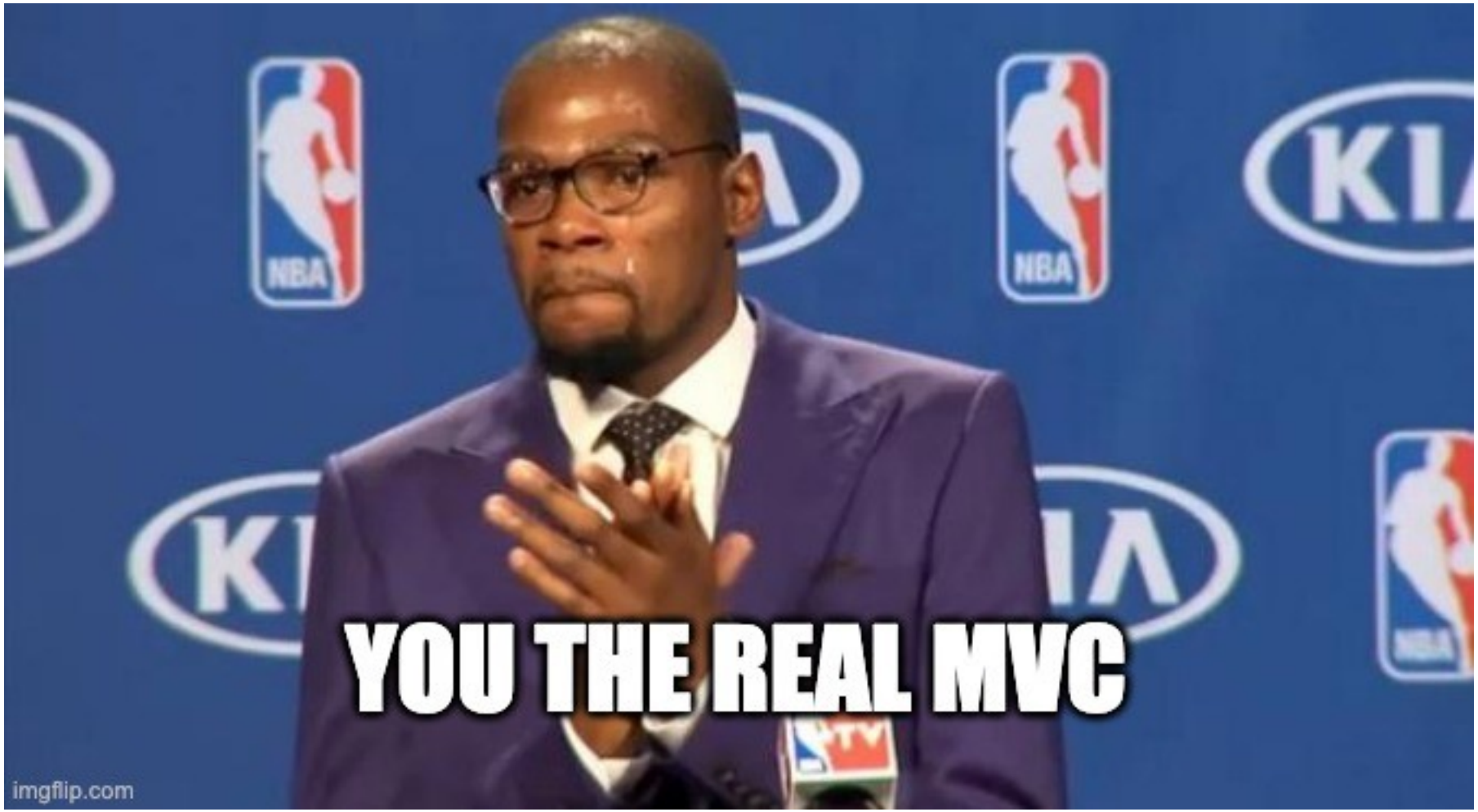
```
=======================Evaluation Metrics=======================
 # of classes:    3
 Accuracy:        0.9245
 Precision:       0.9206
 Recall:          0.9167
 F1 Score:        0.9163
Precision, recall & F1: macro-averaged (equally weighted avg. of 3 classes)


=========================Confusion Matrix=======================
  0  1  2
----------
 21  0  0 | 0 = 0
  0 15  1 | 1 = 1
  0  3 13 | 2 = 2

Confusion matrix format: Actual (rowClass) predicted as (columnClass) N times
================================================================
```


```
// Storing the model
File locationToSaveModel = new File(outputPath + STORED_MODEL_FILENAME);
model.save(locationToSaveModel, false);

// Storing the normalizer
File locationToSaveNormalizer = new File(outputPath + STORED_NORMALIZER_FILENAME);
NormalizerSerializer.getDefault().write(normalizer, locationToSaveNormalizer);
```

# Prediction step



# Iris Classifier Predictor Most Valuable Code



Load the model → Format the data → Normalize the data → Feed the data → Get the label

```
// Load the model
File locationToSaveModel = new File(basePath + STORED_MODEL_FILENAME);
MultiLayerNetwork restoredModel = MultiLayerNetwork.load(locationToSaveModel, false);

// Load normalizer
File locationToSaveNormalizer = new File(basePath + STORED_NORMALIZER_FILENAME);
DataNormalization restoredNormalizer = normalizerSerializer.getDefault()
                                                .restore(locationToSaveNormalizer);
```

Load the model → **Format the data** → Normalize the data → Feed the data → Get the label

```
private static INDArray getArray(Iris iris) {
    float[] input = new float[FIELDS_COUNT];
    input[INDEX_SEPAL_LENGTH] = iris.getSepalLength();
    input[INDEX_SEPAL_WIDTH] = iris.getSepalWidth();
    input[INDEX_PETAL_LENGTH] = iris.getPetalLength();
    input[INDEX_PETAL_WIDTH] = iris.getPetalWidth();

    DataBuffer dataBuffer = new FloatBuffer(input);
    NDArray ndArray = new NDArray(1, FIELDS_COUNT);
    ndArray.setData(dataBuffer);
    return ndArray;
}
```

Load the model → Format the data → Normalize the data → Feed the data → Get the label

```
// Normalize the data the same way it was normalized in the training phase
dataNormalizer.transform(indArray);

// Do the prediction
INDArray result = model.output(indArray, false);
```

Load the model → Format the data → Normalize the data → Feed the data → Get the label

```
private static int getIndexPredictedLabel(INDArray predictions) {
        int maxIndex = 0;
        // We should get max CLASSES_COUNT amount of predictions with probabilities.
        for (int i = 0; i < CLASSES_COUNT; i++) {
            if (predictions.getFloat(i) > predictions.getFloat(maxIndex)) {
                maxIndex = i;
            }
        }
        return maxIndex;
    }
```


```
List<String> LABELS = Arrays.asList("Iris Setosa",
                                    "Iris Versicolour",
                                    "Iris Virginica");
int predictedLabelIndex = getIndexPredictedLabel(result);
return LABELS.get(predictedLabelIndex);
```

Prediction step

| sepal length | sepal width | petal length | petal width |
|---|---|---|---|
| 5,10 | 3,50 | 1,40 | 0,20 |

**FlowerClassification API**

Iris Setosa


# Peter is happy!

# What did we learn?

- 2 main steps
  - Training
    - Input: Labeled dataset
    - Output: Trained model
  - Prediction
    - Input: Trained model + unlabeled data
    - Output: Prediction -> in this case, of type “classification”

- How to use Deeplearning4j to train a model, export it, load it and perform a prediction

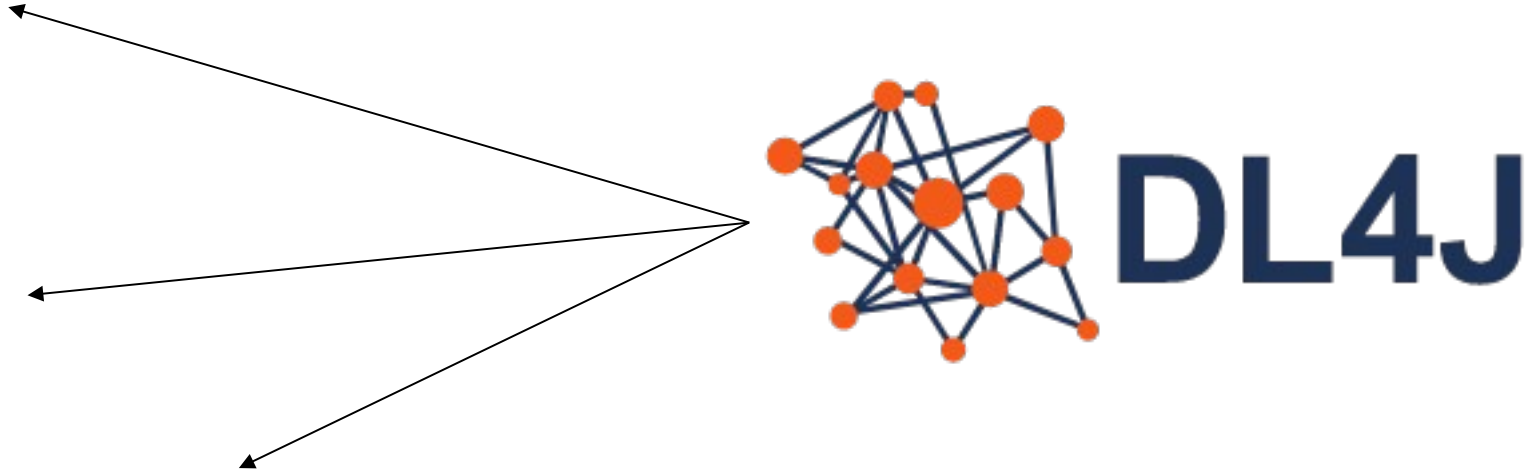
# Machine Learning Approaches

- Supervised
  - Classification
  - Regression
- Unsupervised
- Reinforcement learning

Source: Wikipedia, Machine Learning. https://en.wikipedia.org/wiki/Machine_learning , retrieved

# Machine Learning Approaches

- Supervised
  - Classification
  - Regression
- Unsupervised
- Reinforcement learning



# Machine Learning Algorithms

- Examples:
  - Support Vector Machines
  - Linear regression
  - Logistic regression
  - Naive Bayes
  - Linear discriminant analysis
  - Decision trees
  - Neural Networks (Multilayer perceptron)
  - ...


Source: Wikipedia, Machine Learning. https://en.wikipedia.org/wiki/Machine_learning

# Deeplearning4j integrations


Distributed training

# Deeplearning4j integrations

Parallel training

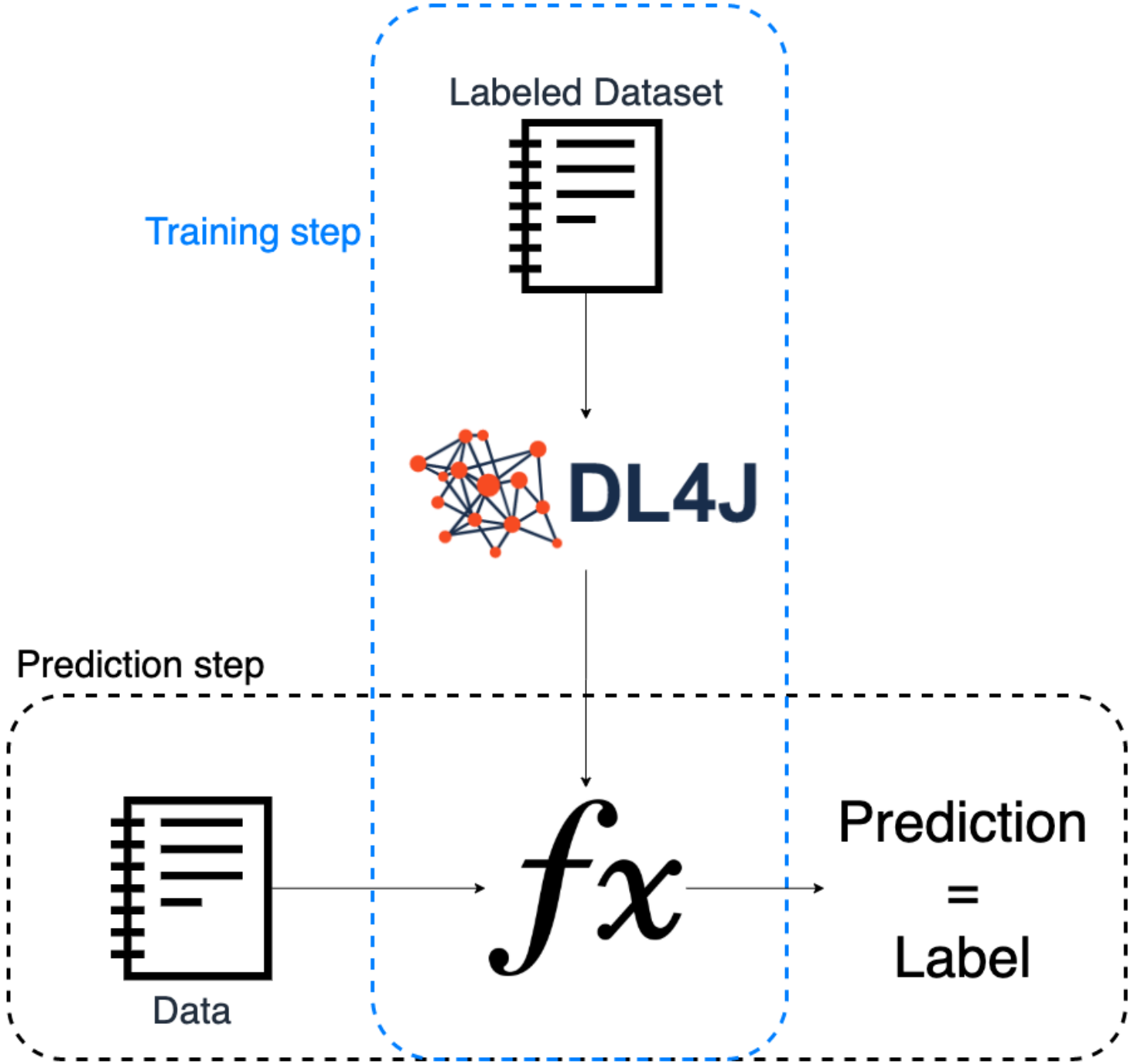

**Import models**

# Deeplearning4j integrations

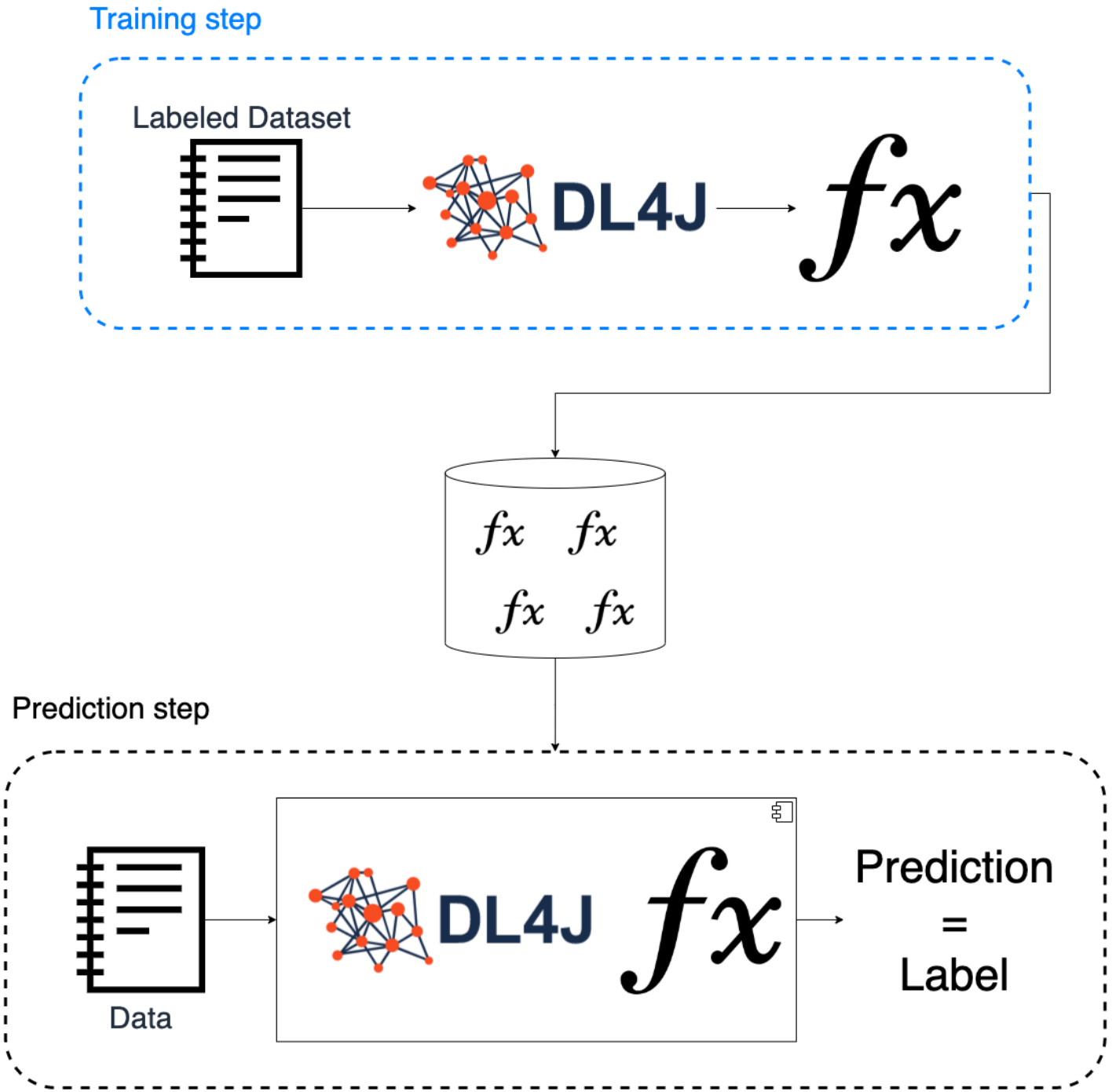

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML.

source: Keras description, Keras, https://en.wikipedia.org/wiki/Keras

# One step back

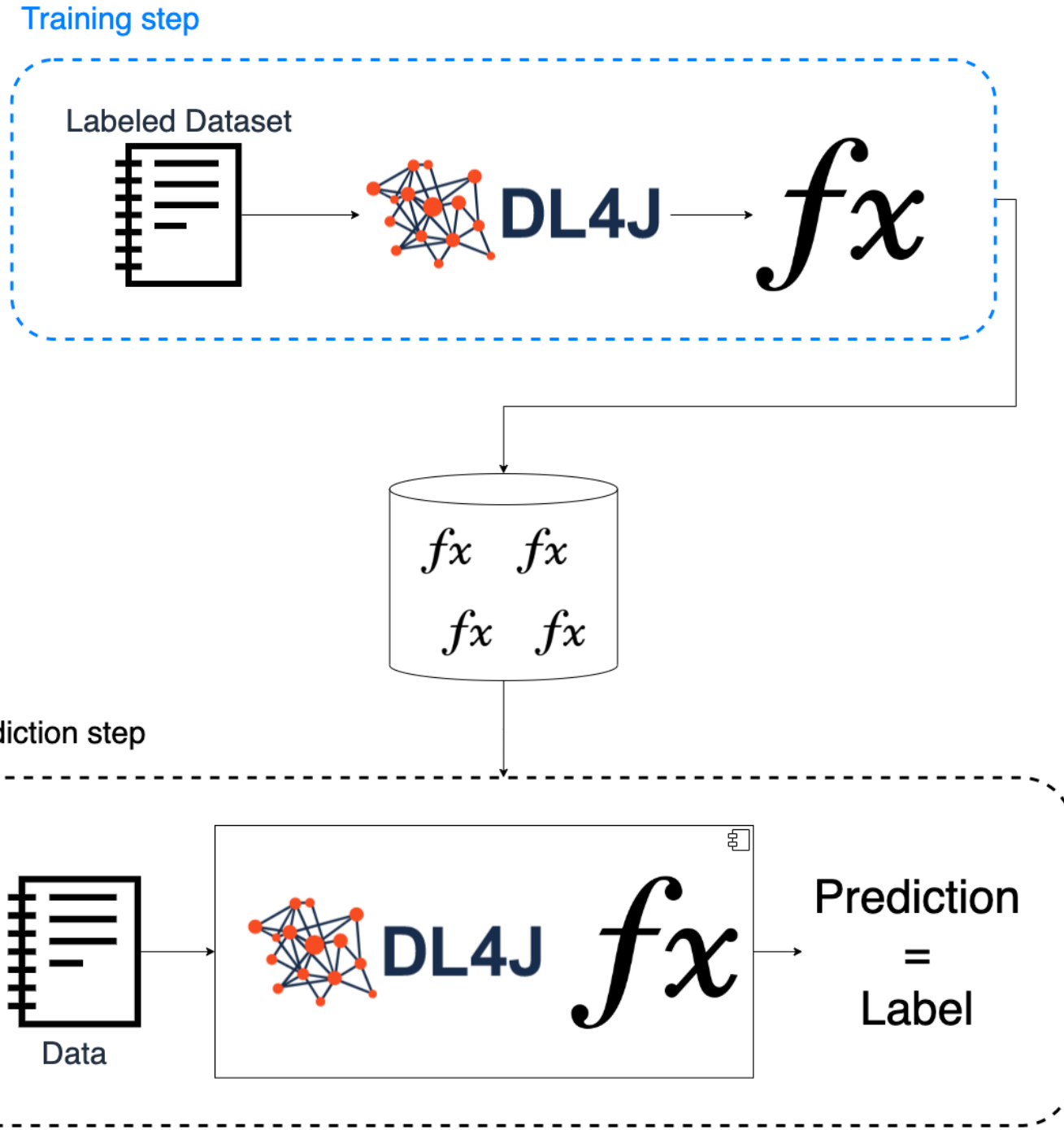


Decoupled in time!





* JVM Language

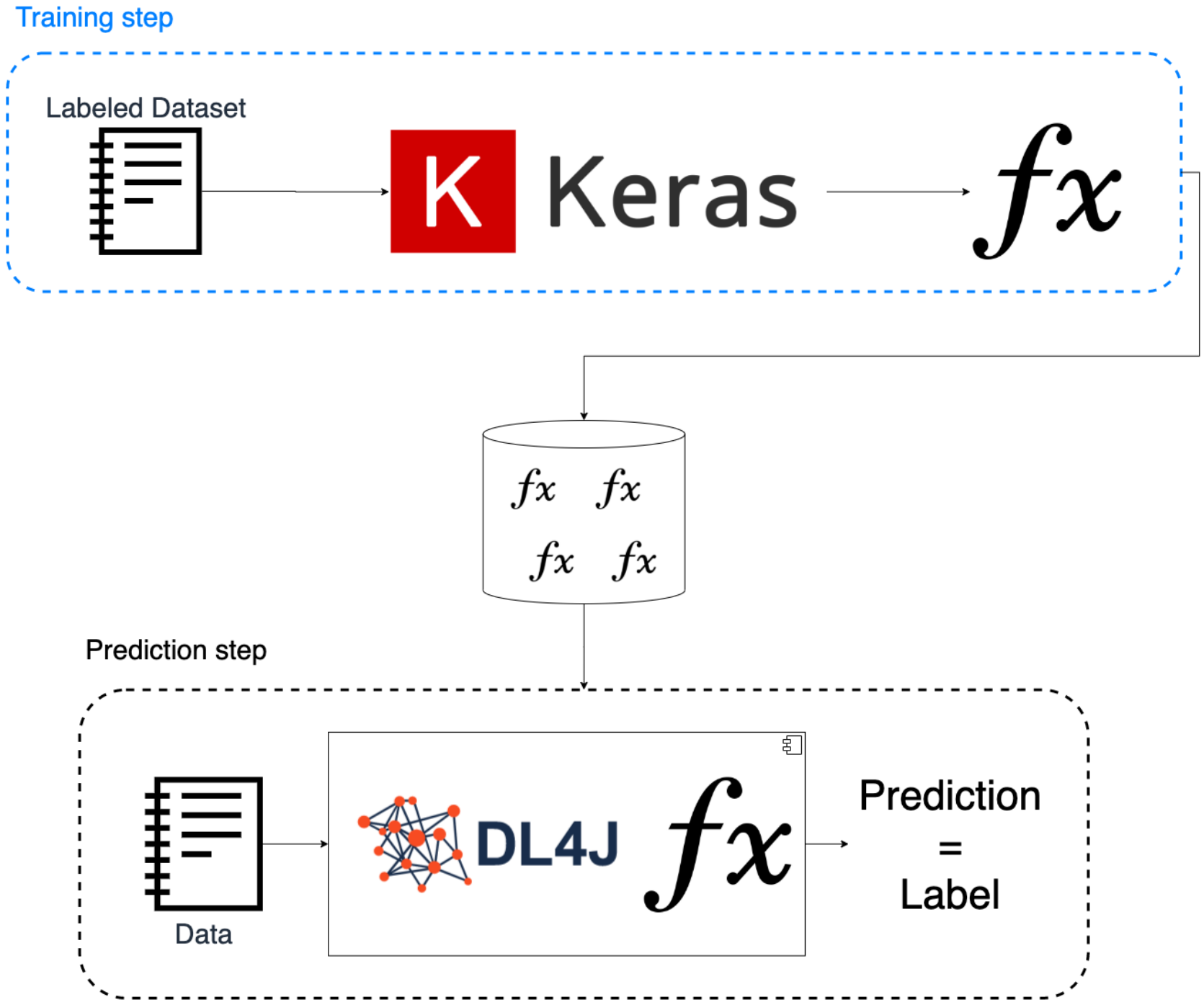


DL4J + Keras

=

Flexibility

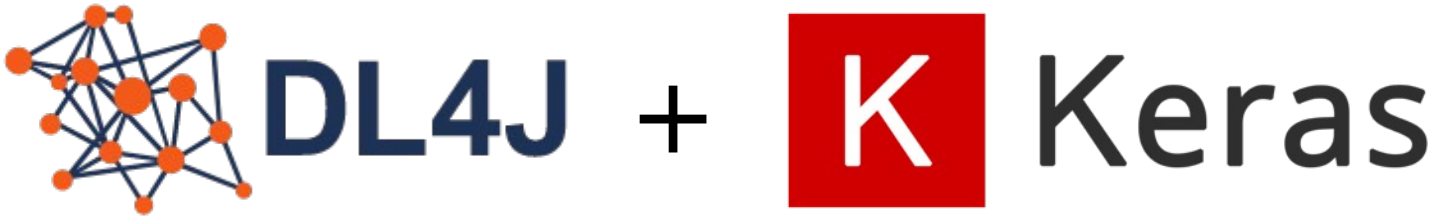

python


Training step
Labeled Dataset
Keras
Prediction step
Data
DL4J
Prediction
=
Label


Java

# Deeplearning4j integrations

- Enables the integration of people with different technical profiles
- Opens the door to more pretrained models




=

Flexibility

# Image Classification

- Approach:
  - Deeplearning4j’s Model Zoo
  - OpenCV

# Deeplearning4j's Model Zoo

master

deeplearning4j / deeplearning4j / deeplearning4j-zoo / src / main / java / org / deeplearning4j / zoo / **model** /

**AlexDBlack** Refactor packages to fix split package issues (#411) ...

| | |
|---|---|
| .. | |
| helper | Refactor packages to fix split package issues (#411) |
| AlexNet.java | Fixing issues from Sonar report (#391) |
| Darknet19.java | Eclipse Migration Initial Commit |
| FaceNetNN4Small2.java | Eclipse Migration Initial Commit |
| InceptionResNetV1.java | Eclipse Migration Initial Commit |
| LeNet.java | Eclipse Migration Initial Commit |
| NASNet.java | Refactor packages to fix split package issues (#411) |
| ResNet50.java | Eclipse Migration Initial Commit |
| SimpleCNN.java | Eclipse Migration Initial Commit |
| SqueezeNet.java | Eclipse Migration Initial Commit |
| TextGenerationLSTM.java | Eclipse Migration Initial Commit |
| TinyYOLO.java | Eclipse Migration Initial Commit |
| UNet.java | Various SameDiff fixes (#21) |
| VGG16.java | Eclipse Migration Initial Commit |
| VGG19.java | Eclipse Migration Initial Commit |
| Xception.java | Eclipse Migration Initial Commit |
| YOLO2.java | Eclipse Migration Initial Commit |


YOLO

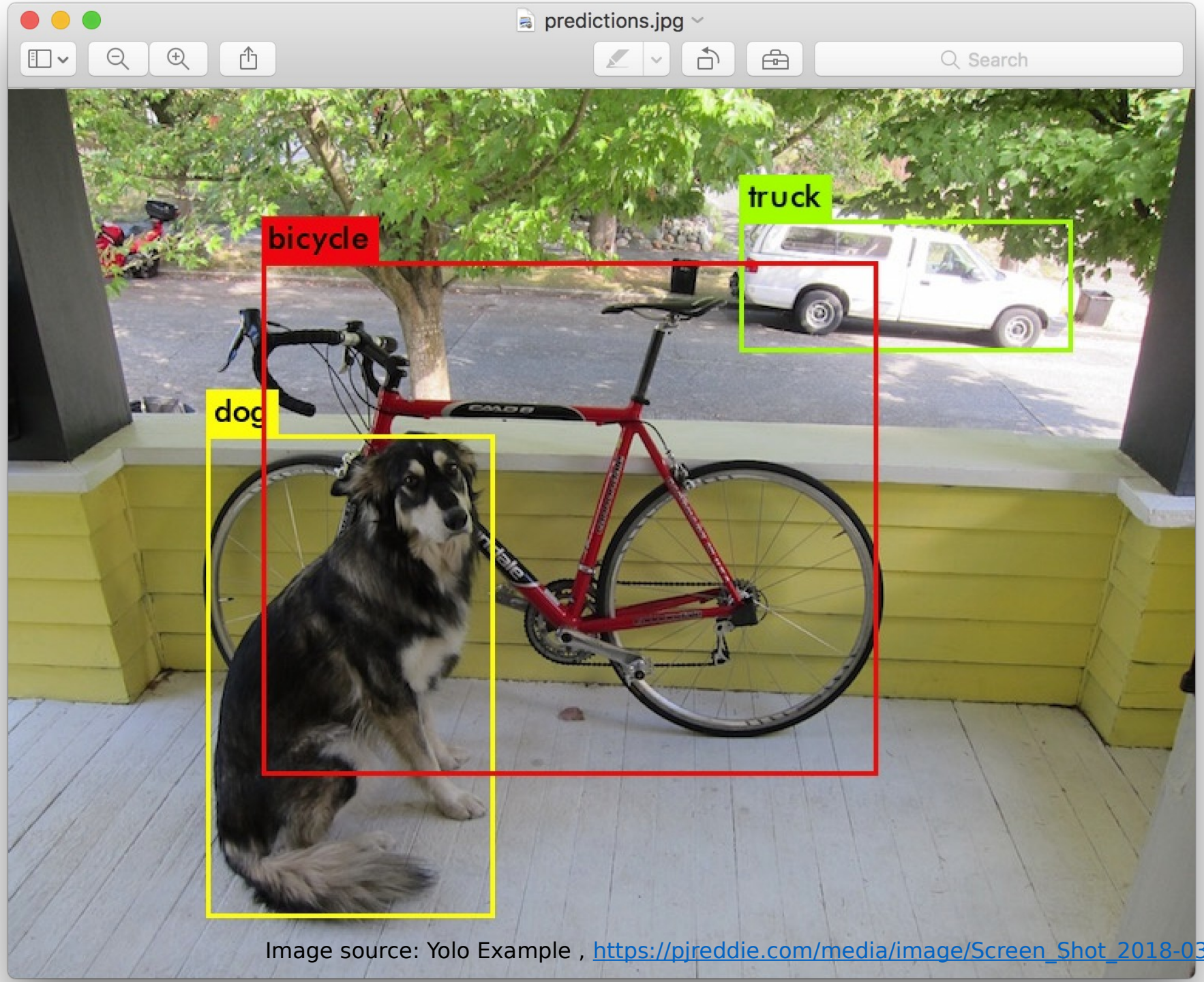
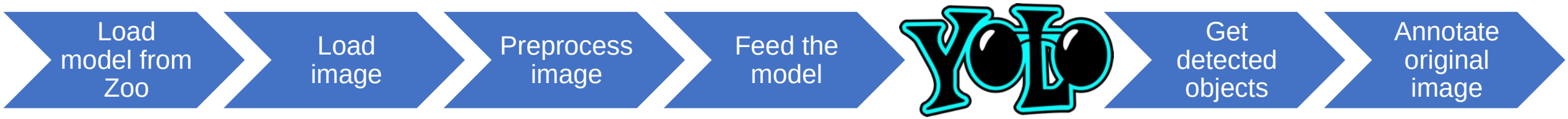



Image source: Yolo Example , https://pjreddie.com/media/image/Screen_Shot_2018-03-24_at_10.48.42_PM.png

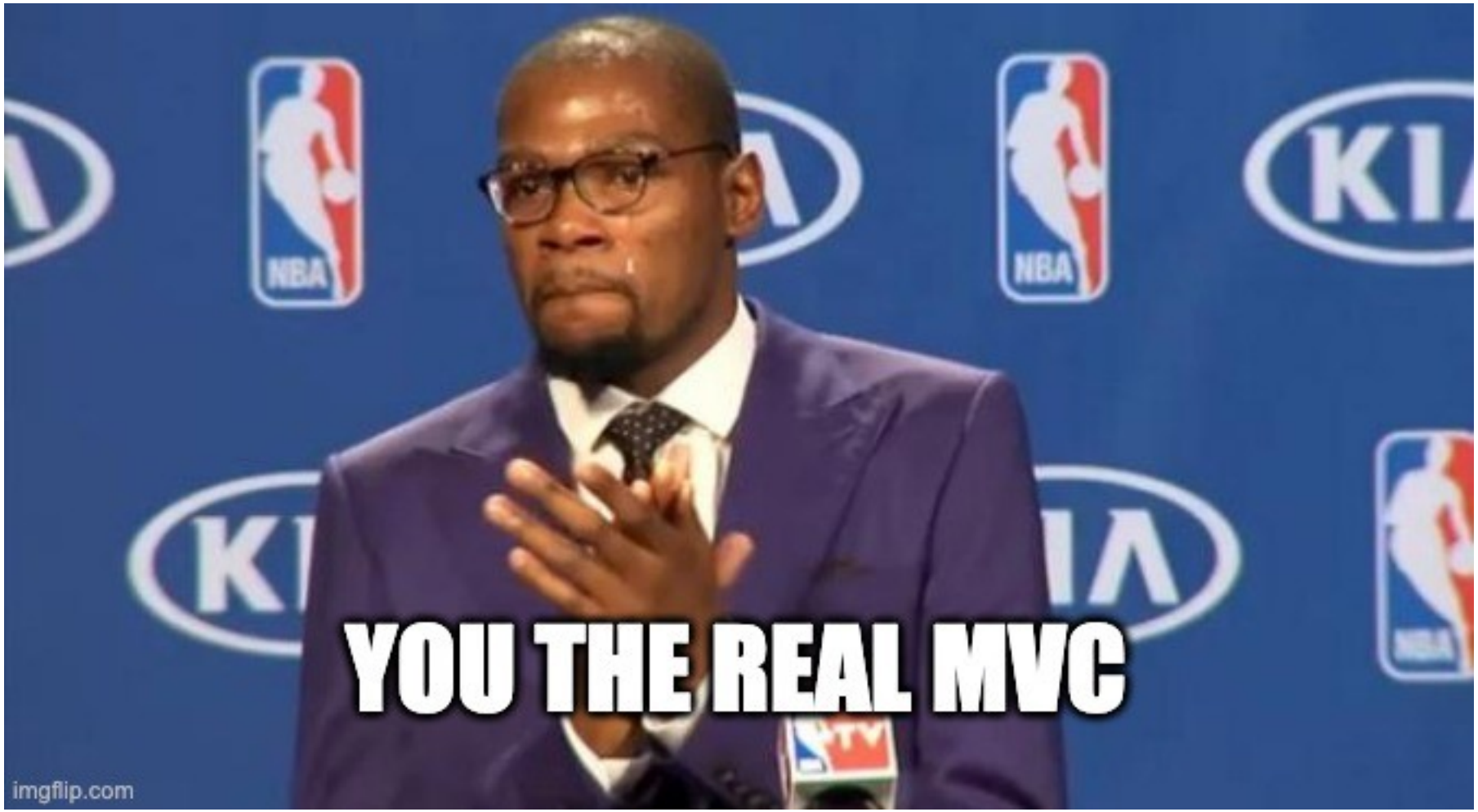
# YOLOv2 Available Labels

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| person | fire hydrant | elephant | skis | wine glass | broccoli | diningtable | toaster |
| bicycle | stop sign | bear | snowboard | cup | carrot | toilet | sink |
| car | parking meter | zebra | sports ball | fork | hot dog | tvmonitor | refrigerator |
| motorbike | bench | giraffe | kite | knife | pizza | laptop | book |
| aeroplane | bird | backpack | baseball bat | spoon | donut | mouse | clock |
| bus | cat | umbrella | baseball glove | bowl | cake | remote | vase |
| train | dog | handbag | skateboard | banana | chair | keyboard | scissors |
| truck | horse | tie | surfboard | apple | sofa | cell phone | teddy bear |
| boat | sheep | suitcase | tennis racket | sandwich | pottedplant | microwave | hair drier |
| traffic light | cow | frisbee | bottle | orange | bed | oven | toothbrush |

# Steps overview



# Yolo2ImageClassifier - Most Valuable Code



```java
public List<DetectedObject> classify(String inputImagePath, String outputImagePath) throws IOException {
        // Load the model from the zoo
        yolo2Model = YOLO2.builder().build();
        pretrainedComputationGraph = (ComputationGraph) yolo2Model.initPretrained();

        // Load the image from disk
        File fileOriginalImage = new File(inputImagePath);
        INDArray iNDArrayOriginalImage = imageLoader.asMatrix(fileOriginalImage);

        // Resize the image and change the format to match the required by YOLO2
        yoloImageLoader = new NativeImageLoader(YOLO2_WIDTH, YOLO2_HEIGHT, CHANNELS,
                                                new ColorConversionTransform(COLOR_BGR2RGB));
        Mat matResizedImage = yoloImageLoader.asMat(iNDArrayOriginalImage);

        // Scale the images, as in "normalize the pixels to be on the range from 0 to 1"
        ImagePreProcessingScaler scaler = new ImagePreProcessingScaler(0, 1);
        INDArray iNDArrayTransformedImage = yoloImageLoader.asMatrix(matResizedImage);
        scaler.transform(iNDArrayTransformedImage);

        // Perform the classification
        INDArray outputs = pretrainedComputationGraph.outputSingle(iNDArrayTransformedImage);
        List<DetectedObject> detectedObjects = YoloUtils.getPredictedObjects(
                        Nd4j.create(((YOLO2) yolo2Model).getPriorBoxes()),
                        outputs,
                        DETECTION_THRESHOLD,
                        NMS_THRESHOLD);

        // Annotate the original image
        Image originalImage = imageLoader.asImageMatrix(fileOriginalImage);
        int originalWidth = originalImage.getOrigW();
        int originalHeight = originalImage.getOrigH();
        annotate(originalWidth, originalHeight, matResizedImage, detectedObjects, outputImagePath);

        return detectedObjects;
    }
```

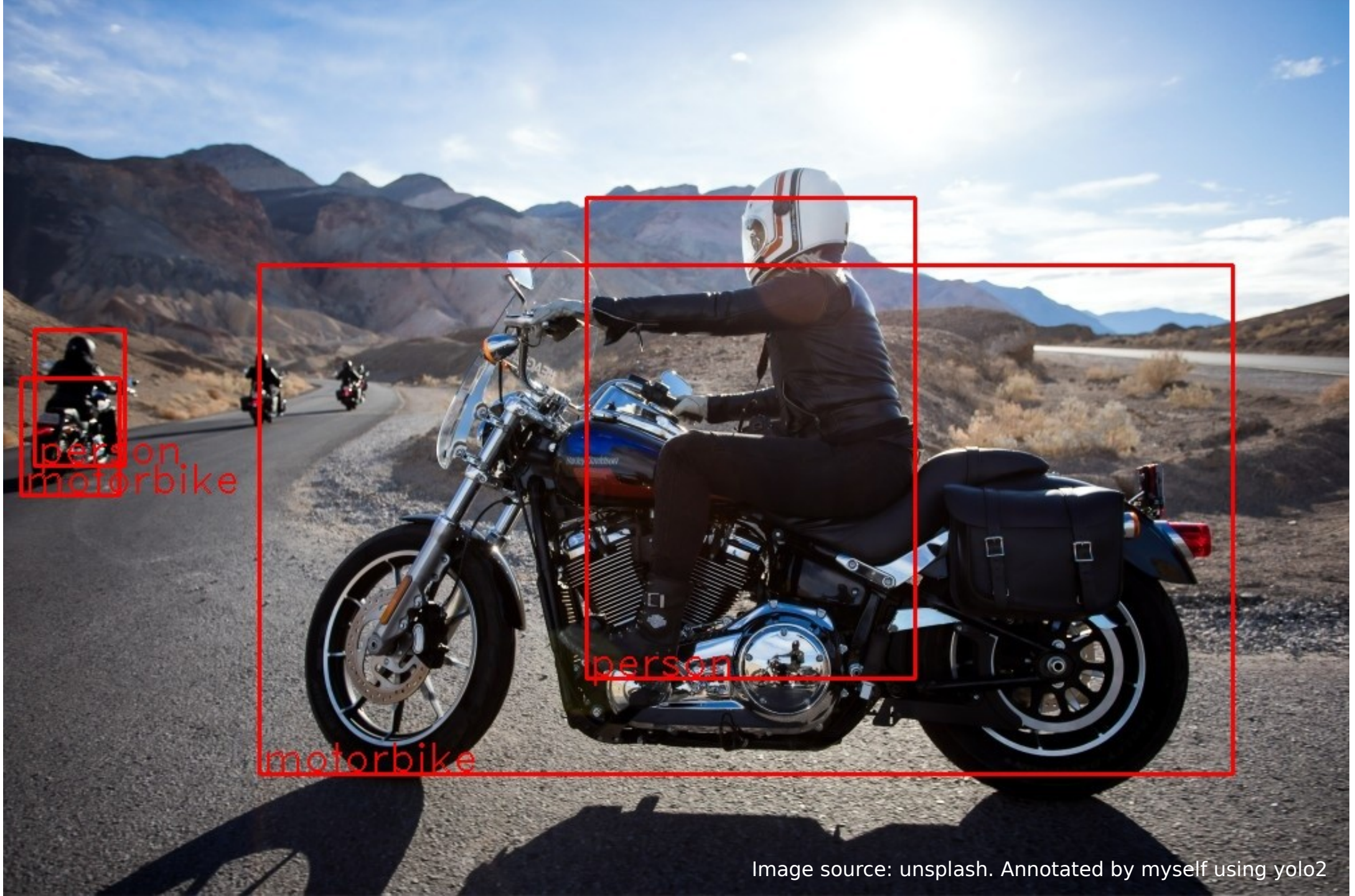



Image source: unsplash. Annotated by myself using yolo2

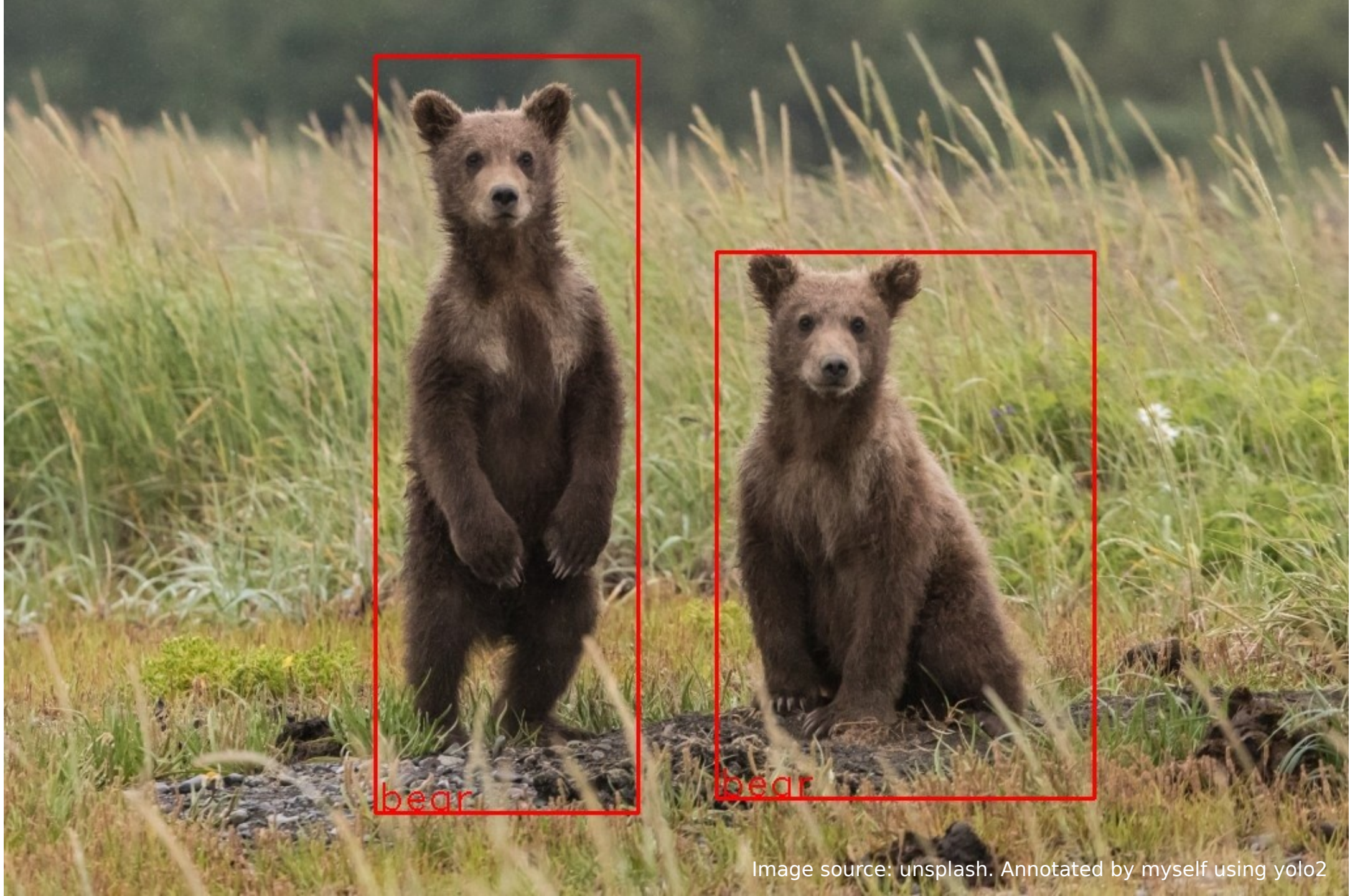



Image source: unsplash. Annotated by myself using yolo2

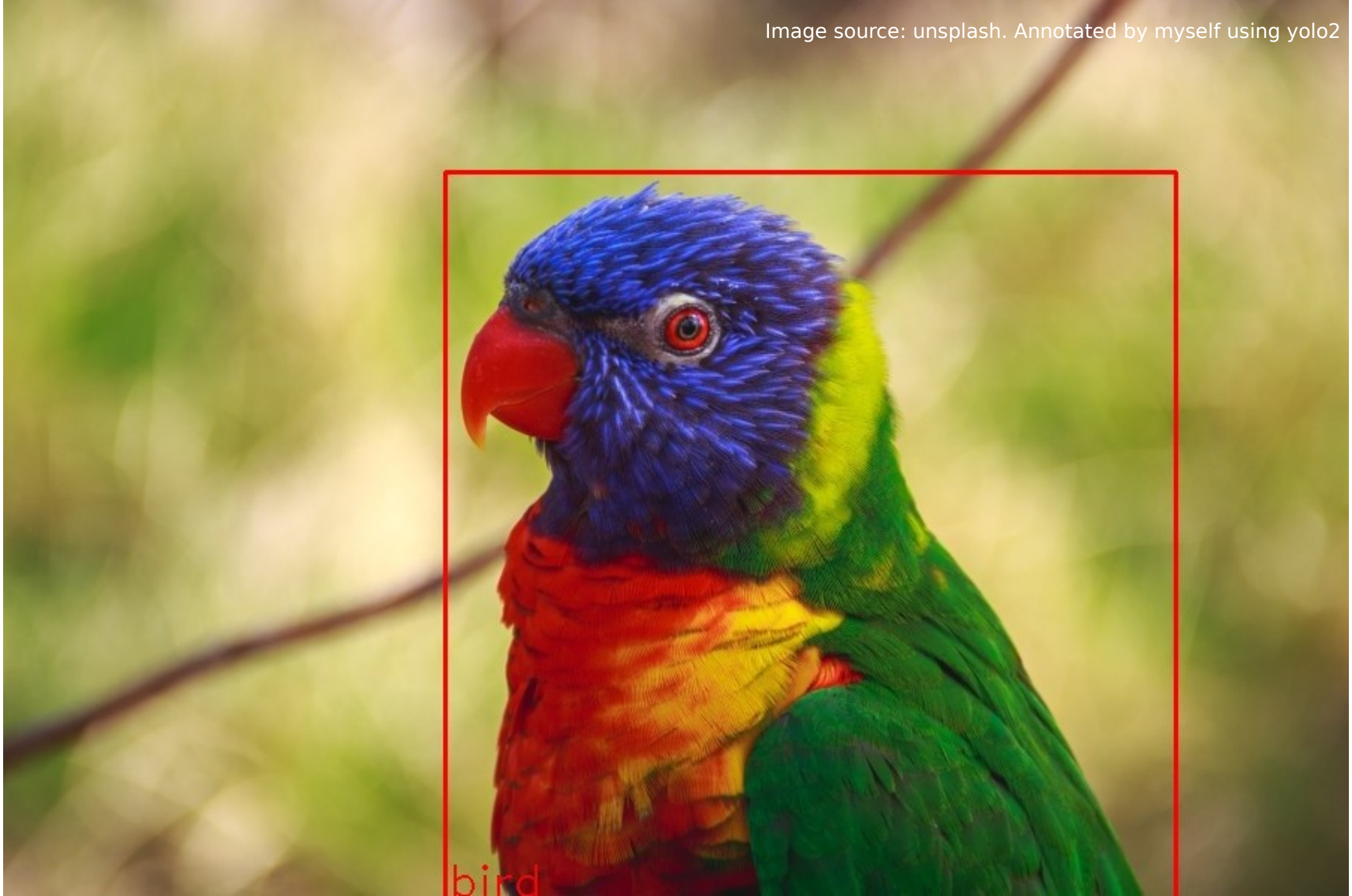


Image source: unsplash. Annotated by myself using yolo2
bird


Image source: unsplash. Annotated by myself using yolo2

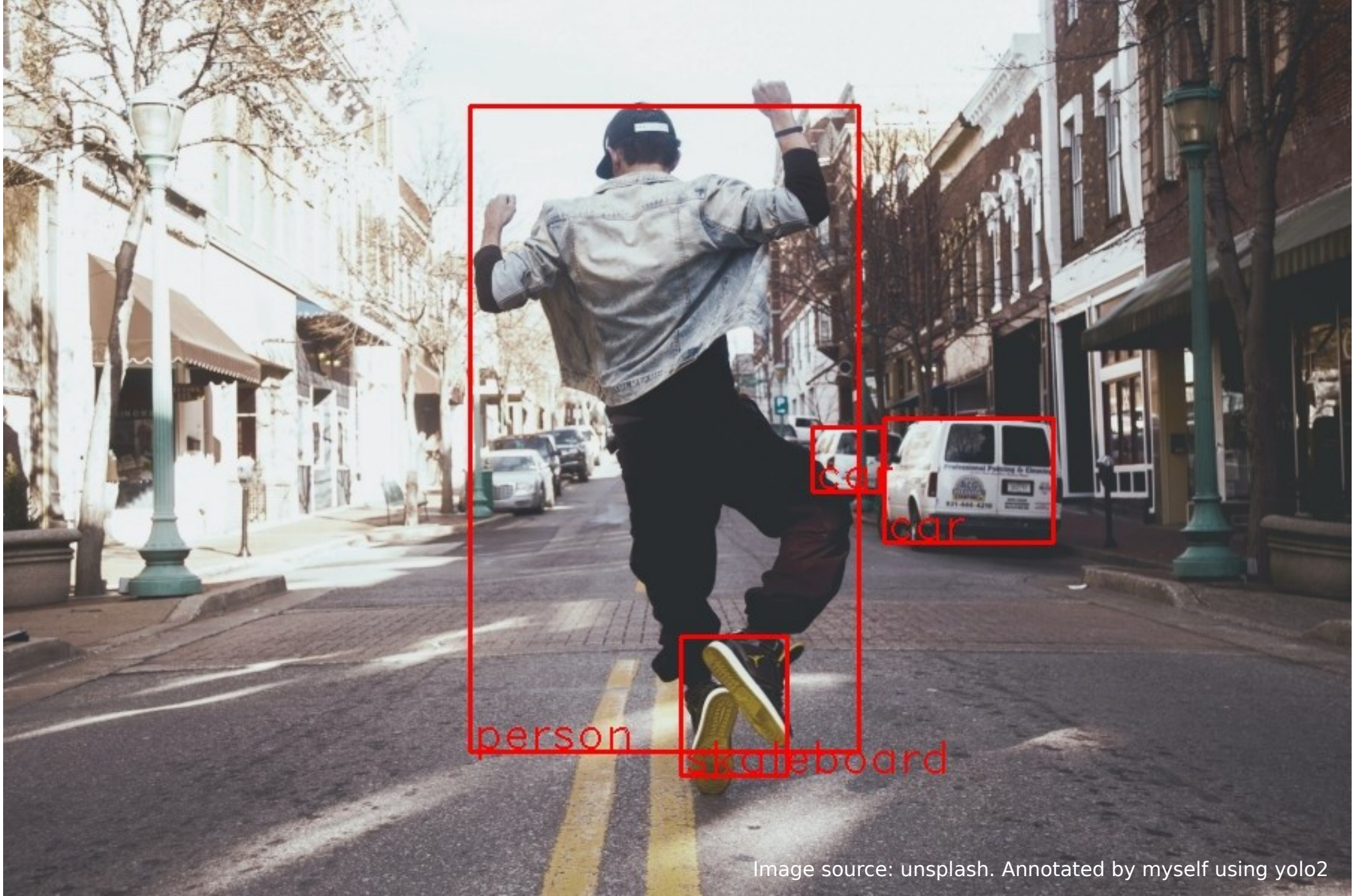



Image source: unsplash. Annotated by myself using yolo2


Image source: Deeplearning4j Animal Classification Dataset. Annotated by myself using yolo2


Image source: Deeplearning4j Animal Classification Dataset. Annotated by myself using yolo2

# Sentiment Analysis

- Approach:
  - Take an example and modify it
    - deeplearning4j.examples - ImdbReviewClassificationCNN.java
  - Get a new dataset from https://www.kaggle.com/kazanova/sentiment140

Dataset

781

# Sentiment140 dataset with 1.6 million tweets

Sentiment analysis with tweets

Μαριος Μιχαηλιδης KazAnova • updated 3 years ago (Version 2)



**Usability** 8.8

**License** Other (specified in description)

**Tags** internet, online communities, social networks, linguistics, languages

Description

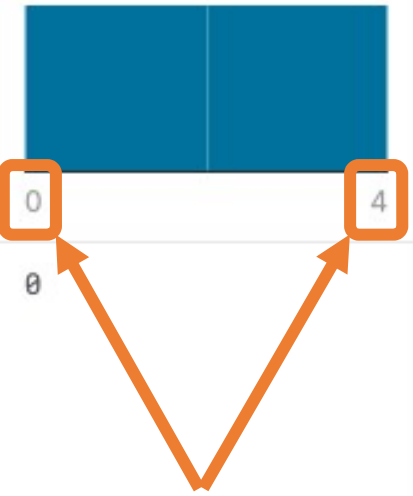
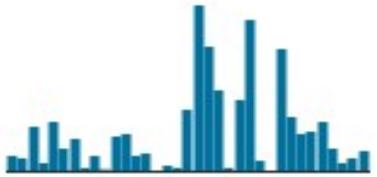
## Context

This is the sentiment140 dataset. It contains 1,600,000 tweets extracted using the twitter api . The tweets have been annotated (0 = negative, 4 = positive) and they can be used to detect sentiment .

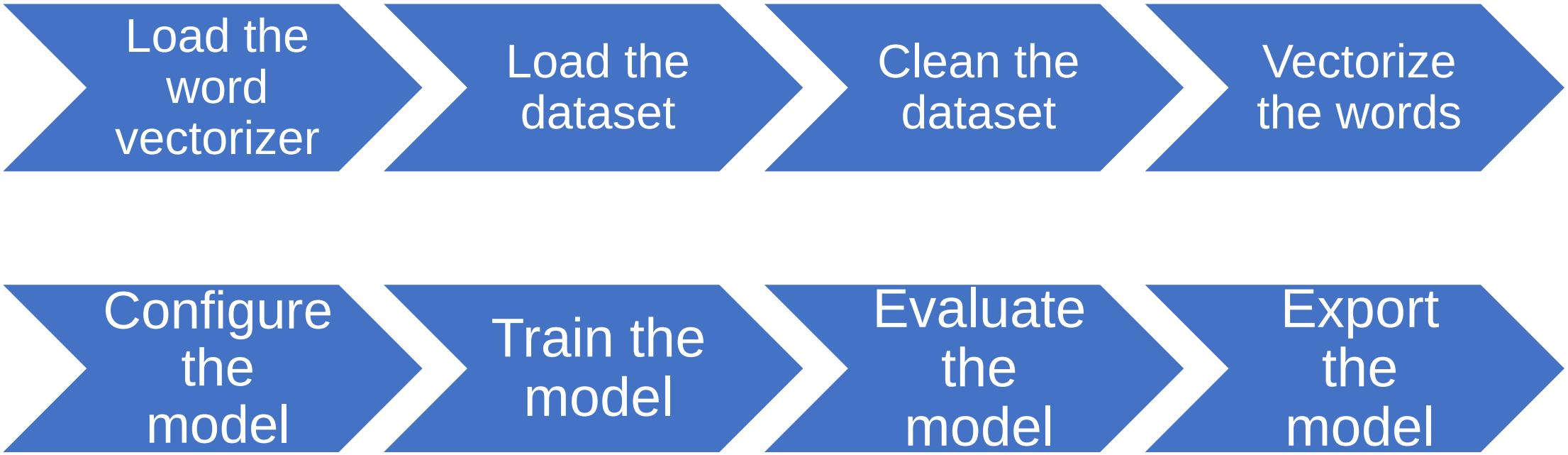
## Content

It contains the following 6 fields:

1. target: the polarity of the tweet (*0* = negative, *2* = neutral, *4* = positive)

| # 0 | # 1467810369 | A Mon Apr 06 22:1... | A NO_QUERY | A _TheSpecialOne_ | A @switchfoot htt... |
|---|---|---|---|---|---|
| target | id | date | flag | user | text |
|  0 4 |  1.47b 2.33b | 774362 unique values | 1 unique value | 659775 unique values | 1581465 unique values |
| 0 | 1467810672 | Mon Apr 06 22:19:49 PDT 2009 | NO_QUERY | scotthamilton | is upset that he can't update his Facebook by texting it... and might cry as a result School today ... |
| 0 | 1467810917 | Mon Apr 06 22:19:53 PDT 2009 | NO_QUERY | mattycus | @Kenichan I dived many times for the ball. Managed to save 50% The rest go out of bounds |
| 0 | 1467811184 | Mon Apr 06 22:19:57 PDT 2009 | NO_QUERY | ElleCTF | my whole body feels itchy and like its on fire |
| 0 | 1467811193 | Mon Apr 06 22:19:57 PDT 2009 | NO_QUERY | Karoli | @nationwideclass no, it's not behaving at all. i'm mad. why am i here? because I can't see you all o... |
| 0 | 1467811372 | Mon Apr 06 22:20:00 | NO_QUERY | joy_wolf | @Kwesidei not the |

# Overview


Load the word vectorizer
Load the dataset
Clean the dataset
Vectorize the words
Configure the model
Train the model
Evaluate the model
Export the model

# Vectorizing words

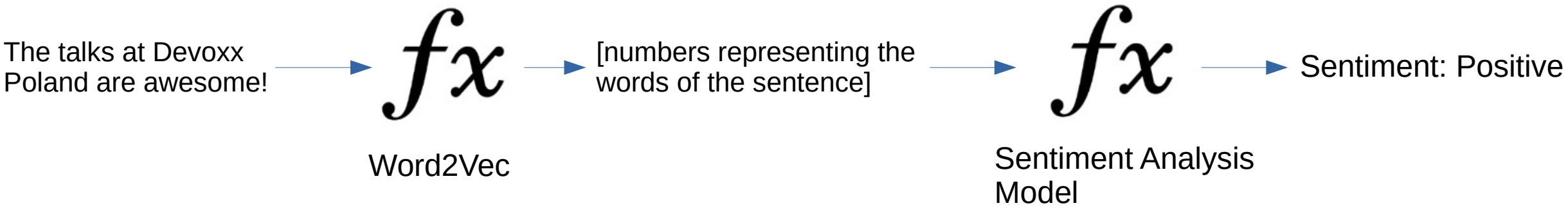
Goal:
Transform the words into vectors (numbers) that the model can understand

Word2Vec

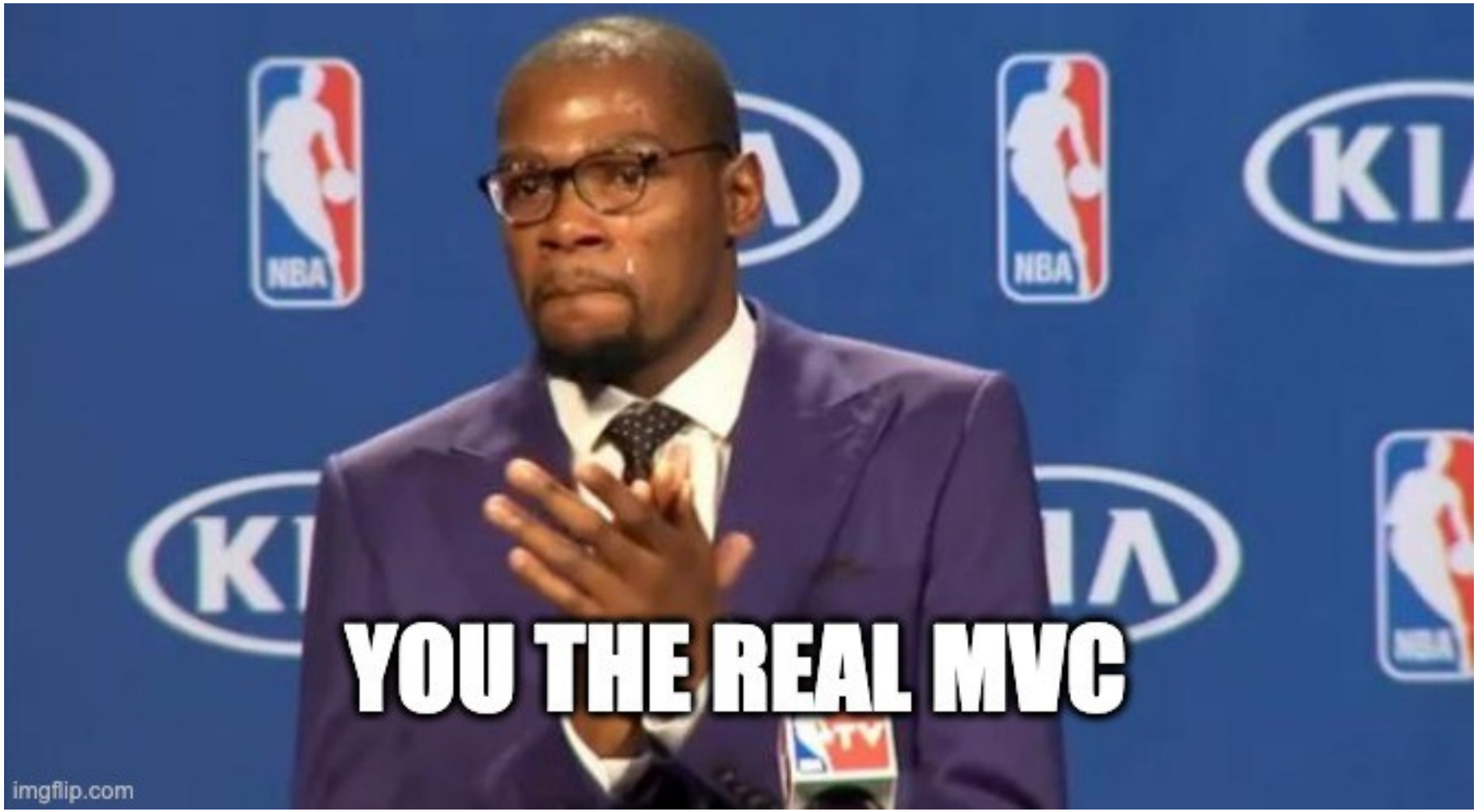
- Two-layer neural network
- Input: a corpus of text
- Output: a vector space
  - Each unique word in the corpus is assigned a corresponding vector in the space

Source: Wikipedia, Word2vec, https://en.wikipedia.org/wiki/Word2vec

# Vectorizing words



# Sentiment Analysis – Most Valuable Code



# Goal:
# Transform the words into vectors (numbers) that the model can understand

```
WordVectors wordVectors = WordVectorSerializer.loadStaticModel(new File(wordVectorsPath));
```

- This is loading a trained Word2Vec model from Google, can be downloaded from https://dl4jdata.blob.core.windows.net/resources/wordvectors/GoogleNews-vectors-negative300.bin.gz
- Example to train your own word2vec model at SimpleExampleWord2Vec.java at https://github.com/ellerenad/deeplearning4j-playground

```
LineIterator lineIterator = FileUtils.lineIterator(new File(path), "UTF-8");
int counter = 0;
String line = null;
while (lineIterator.hasNext()) {
    line = lineIterator.nextLine();
    try {
        String[] parts = line.split("\",\"");
        String label = getLabel(parts);
        String data = getData(parts);
        // Dataset splitting (test and training)
        if (counter % 2 == 0) {
            trainingLabelList.add(label);
            trainingDataList.add(data);
        } else {
            testLabelList.add(label);
            testDataList.add(data);
        }
    } catch(Exception ex){
        log.error("Exception: {}. Counter = {}. Line = {}", ex, counter, line);
    }
    counter++;
}
```

```
LineIterator lineIterator = FileUtils.lineIterator(new File(path), "UTF-8");
int counter = 0;
String line = null;
while (lineIterator.hasNext()) {
    line = lineIterator.nextLine();
    try {
        String[] parts = line.split("\",\"");
        String label = getLabel(parts);
        String data = getData(parts);
        // Dataset splitting (test and training)
        if (counter % 2 == 0) {
            trainingLabelList.add(label);
            trainingDataList.add(data);
        } else {
            testLabelList.add(label);
            testDataList.add(data);
        }
    } catch(Exception ex){
        log.error("Exception: {}. Counter = {}. Line = {}", ex, counter, line);
    }
    counter++;
}
```

```
LineIterator lineIterator = FileUtils.lineIterator(new File(path), "UTF-8");
int counter = 0;
String line = null;
while (lineIterator.hasNext()) {
    line = lineIterator.nextLine();
    try {
        String[] parts = line.split("\",\"");
        String label = getLabel(parts);
        String data = getData(parts);
        // Dataset splitting (test and training)
        if (counter % 2 == 0) {
            trainingLabelList.add(label);
            trainingDataList.add(data);
        } else {
            testLabelList.add(label);
            testDataList.add(data);
        }
    } catch(Exception ex){
        log.error("Exception: {}. Counter = {}. Line = {}", ex, counter, line);
    }
    counter++;
}
```

```java
LineIterator lineIterator = FileUtils.lineIterator(new File(path), "UTF-8");
int counter = 0;
String line = null;
while (lineIterator.hasNext()) {
    line = lineIterator.nextLine();
    try {
        String[] parts = line.split("\",\"");
        String label = getLabel(parts);
        String data = getData(parts);
        // Dataset splitting (test and training)
        if (counter % 2 == 0) {
            trainingLabelList.add(label);
            trainingDataList.add(data);
        } else {
            testLabelList.add(label);
            testDataList.add(data);
        }
    } catch(Exception ex){
        log.error("Exception: {}. Counter = {}. Line = {}", ex, counter, line);
    }
    counter++;
}
```

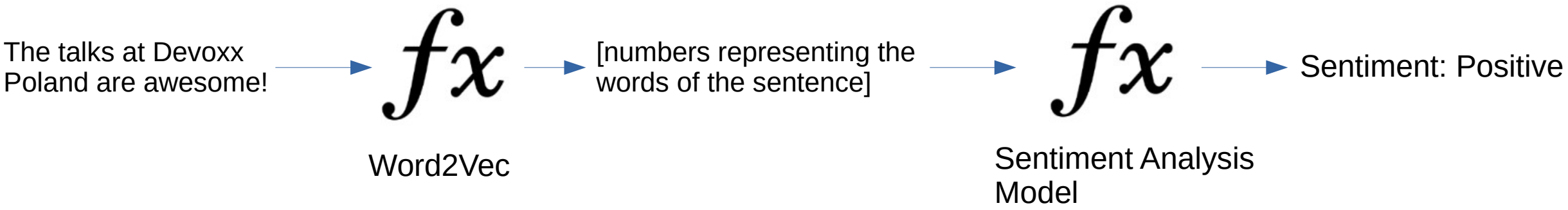
```
LineIterator lineIterator = FileUtils.lineIterator(new File(path), "UTF-8");
int counter = 0;
String line = null;
while (lineIterator.hasNext()) {
    line = lineIterator.nextLine();
    try {
        String[] parts = line.split("\",\"");
        String label = getLabel(parts);
        String data = getData(parts);
        // Dataset splitting (test and training)
        if (counter % 2 == 0) {
            trainingLabelList.add(label);
            trainingDataList.add(data);
        } else {
            testLabelList.add(label);
            testDataList.add(data);
        }
    } catch(Exception ex){
        log.error("Exception: {}. Counter = {}. Line = {}", ex, counter, line);
    }
    counter++;
}
```

```
DataWrapper dataWrapper = getCleanData(path);

// The sentence providers shuffle the data internally
LabeledSentenceProvider trainingSentenceProvider = new CollectionLabeledSentenceProvider(
                                                    dataWrapper.getTraining_data(),
                                                    dataWrapper.getTraining_labels(),
                                                    rng);


// The iterator vectorizes internally the words
DataSetIterator trainingDataSetIterator = new CnnSentenceDataSetIterator.Builder(Format.CNN2D)
            .sentenceProvider(trainingSentenceProvider)
            .wordVectors(wordVectors)
            .minibatchSize(minibatchSize)
            .maxSentenceLength(maxSentenceLength)
            .useNormalizedWordVectors(false)
            .build();
```

Configure the model

```
ComputationGraphConfiguration config = new NeuralNetConfiguration.Builder()
                .weightInit(WeightInit.RELU)
                .activation(Activation.LEAKYRELU)
                .updater(new Adam(0.01))
                .convolutionMode(ConvolutionMode.Same)
                .l2(0.0001)
                .graphBuilder()
                .addInputs("input")
                .addLayer("cnn3", new ConvolutionLayer.Builder()
                        .kernelSize(3, vectorSize)
                        .stride(1, vectorSize)
                        .nOut(cnnLayerFeatureMaps)
                        .build(), "input")
                .addLayer("cnn4", new ConvolutionLayer.Builder()
                        .kernelSize(4, vectorSize)
                        .stride(1, vectorSize)
                        .nOut(cnnLayerFeatureMaps)
                        .build(), "input")
                .addLayer("cnn5", new ConvolutionLayer.Builder()
                        .kernelSize(5, vectorSize)
                        .stride(1, vectorSize)
                        .nOut(cnnLayerFeatureMaps)
                        .build(), "input")
                .addVertex("merge", new MergeVertex(), "cnn3", "cnn4", "cnn5")
                .addLayer("globalPool", new GlobalPoolingLayer.Builder()
                        .poolingType(globalPoolingType)
                        .dropOut(0.5)
                        .build(), "merge")
                .addLayer("out", new OutputLayer.Builder()
                        .lossFunction(LossFunctions.LossFunction.MCXENT)
                        .activation(Activation.SOFTMAX)
                        .nOut(2)    //2 classes: positive or negative
                        .build(), "globalPool")
                .setOutputs("out")
                //Input has shape [minibatch, channels=1, length=1 to 256, 300]
                .setInputTypes(InputType.convolutional(truncateReviewsToLength, vectorSize, 1))
                .build();
```

# Vectorizing words



Configure the model

```
ComputationGraphConfiguration config = new NeuralNetConfiguration.Builder()
                .weightInit(WeightInit.RELU)
                .activation(Activation.LEAKYRELU)
                .updater(new Adam(0.01))
                .convolutionMode(ConvolutionMode.Same)
                .l2(0.0001)
                .graphBuilder()
                .addInputs("input")
                .addLayer("cnn3", new ConvolutionLayer.Builder()
                        .kernelSize(3, vectorSize)
                        .stride(1, vectorSize)
                        .nOut(cnnLayerFeatureMaps)
                        .build(), "input")
                .addLayer("cnn4", new ConvolutionLayer.Builder()
                        .kernelSize(4, vectorSize)
                        .stride(1, vectorSize)
                        .nOut(cnnLayerFeatureMaps)
                        .build(), "input")
                .addLayer("cnn5", new ConvolutionLayer.Builder()
                        .kernelSize(5, vectorSize)
                        .stride(1, vectorSize)
                        .nOut(cnnLayerFeatureMaps)
                        .build(), "input")
                .addVertex("merge", new MergeVertex(), "cnn3", "cnn4", "cnn5")
                .addLayer("globalPool", new GlobalPoolingLayer.Builder()
                        .poolingType(globalPoolingType)
                        .dropOut(0.5)
                        .build(), "merge")
                .addLayer("out", new OutputLayer.Builder()
                        .lossFunction(LossFunctions.LossFunction.MCXENT)
                        .activation(Activation.SOFTMAX)
                        .nOut(2)    //2 classes: positive or negative
                        .build(), "globalPool")
                .setOutputs("out")
                //Input has shape [minibatch, channels=1, length=1 to 256, 300]
                .setInputTypes(InputType.convolutional(truncateReviewsToLength, vectorSize, 1))
                .build();
```


```
DataSetIterator trainIter = dataSetIteratorWrapper.getTrain();
DataSetIterator testIter = dataSetIteratorWrapper.getTest();

// Listeners to print information during the training and the final evaluation
net.setListeners(new ScoreIterationListener(100),
                 new EvaluativeListener(testIter, 1, InvocationType.EPOCH_END));
// Train the neural network
net.fit(trainIter, nEpochs);

// Export the neural network
net.save(new File("/example/path/"));
```

# Sentiment analysis prediction example

“this code is driving me crazy! what was this guy thinking!? *Annotates the code* oh... it was me”

P(Negative) = 0.72

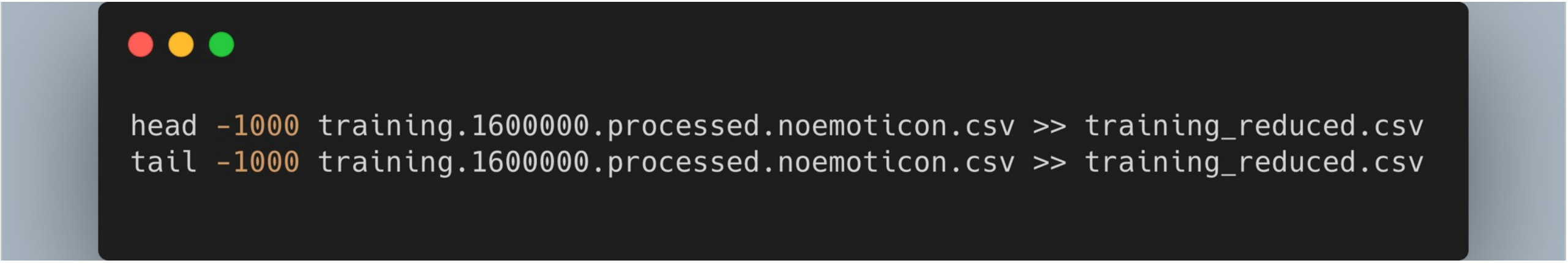
P(Positive) = 0.28

Label: Negative

# Problems during development

- Slow feedback
- Memory management
- Loading the dataset

# Sampling for faster feedback



# Comparing results

Samples: 800000

| | |
|---|---|
| Accuracy | 0.7472 |
| Precision | 0.7200 |
| Recall | 0.8065 |
| F1 Score | 0.7608 |
| | |
| Execution Time | 1581209 ms |
| | ~26.3 min |

40x

10x

Samples: 2000

| | |
|---|---|
| Accuracy | 0.6721 |
| Precision | 0.6382 |
| Recall | 0.7886 |
| F1 Score | 0.7055 |
| | |
| Execution Time | 165002 ms |
| | ~2.75 min |

# Comparing results

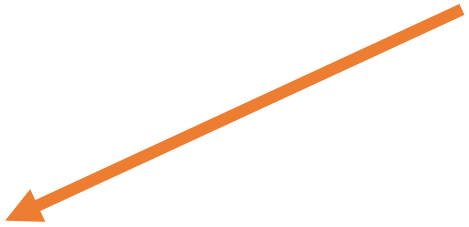
## Samples: 2000

| | |
|---|---|
| Accuracy | 0.6721 |
| Precision | 0.6382 |
| Recall | 0.7886 |
| F1 Score | 0.7055 |
| | |
| Execution Time | 165002 ms |
| | ~2.75 min |

Loading the word vectors:

154804 ms

~2.58 min



```
WordVectors wordVectors = WordVectorSerializer.loadStaticModel(new File(wordVectorsPath));
```

# Memory Management

- Tweak the JVM with the params:

```
-Xms1024m // Initial memory allocation pool
-Xmx10g // Maximum memory allocation pool
```

More details at: https://deeplearning4j.konduit.ai/config/config-memory

# Load the data set

On files bigger than 40,000 lines, the java.util.Scanner would read just up to the half of the file, messing with the training.

```
// Load dataset
LineIterator lineIterator = FileUtils.lineIterator(new File(path), "UTF-8");
```

# Thanks!

# Questions?

All the code can be found at

https://github.com/ellerenad/deeplearning4j-playground

Twitter @ellerenad

Github @ellerenad

Blog posts about this talk at ienjoysoftware.dev

Scan the QR code to get all the relevant links!