ECLIPSE 2021 CON

Machine Learning with Java? Deeplearning4j!

Enrique Llerena Dominguez

$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

@ellerenad

@ellerenad

ellerenad@hotmail.com

https://ienjoysoftware.dev


MIMACOM

Agenda

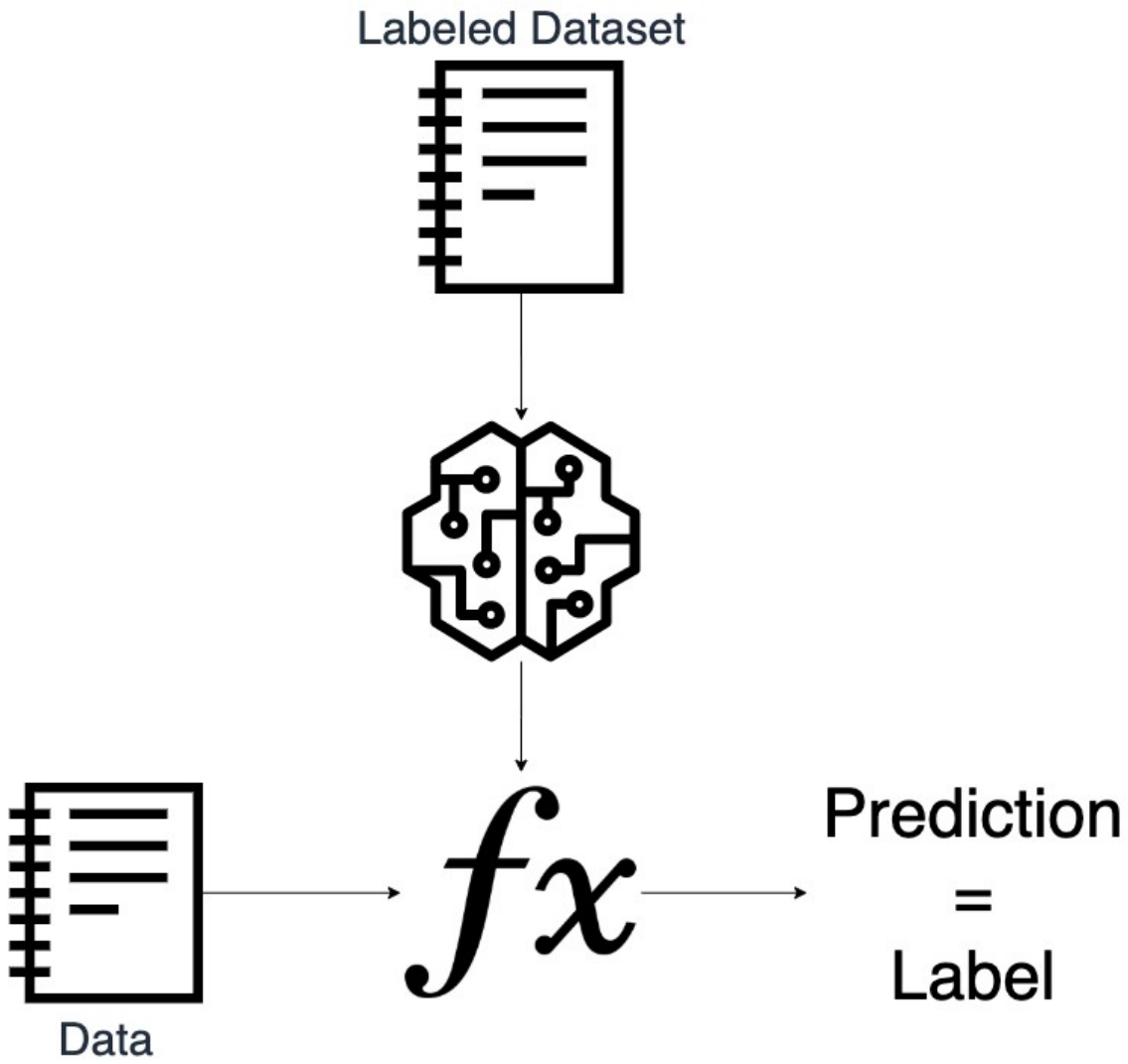
- Machine Learning 101 + deeplearning4j
- Deeplearning4j integrations
- Hands on examples:
  - Image classification – Approach and code

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

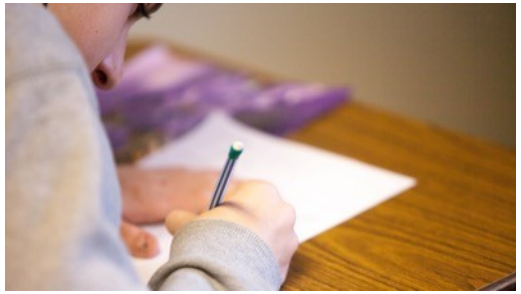
Photo by Jarritos Mexican Soda on Unsplash

Photo by Jarritos Mexican Soda on Unsplash

Photo by Jarritos Mexican Soda on Unsplash

Photo by NESA by Makers on Unsplash

Resources

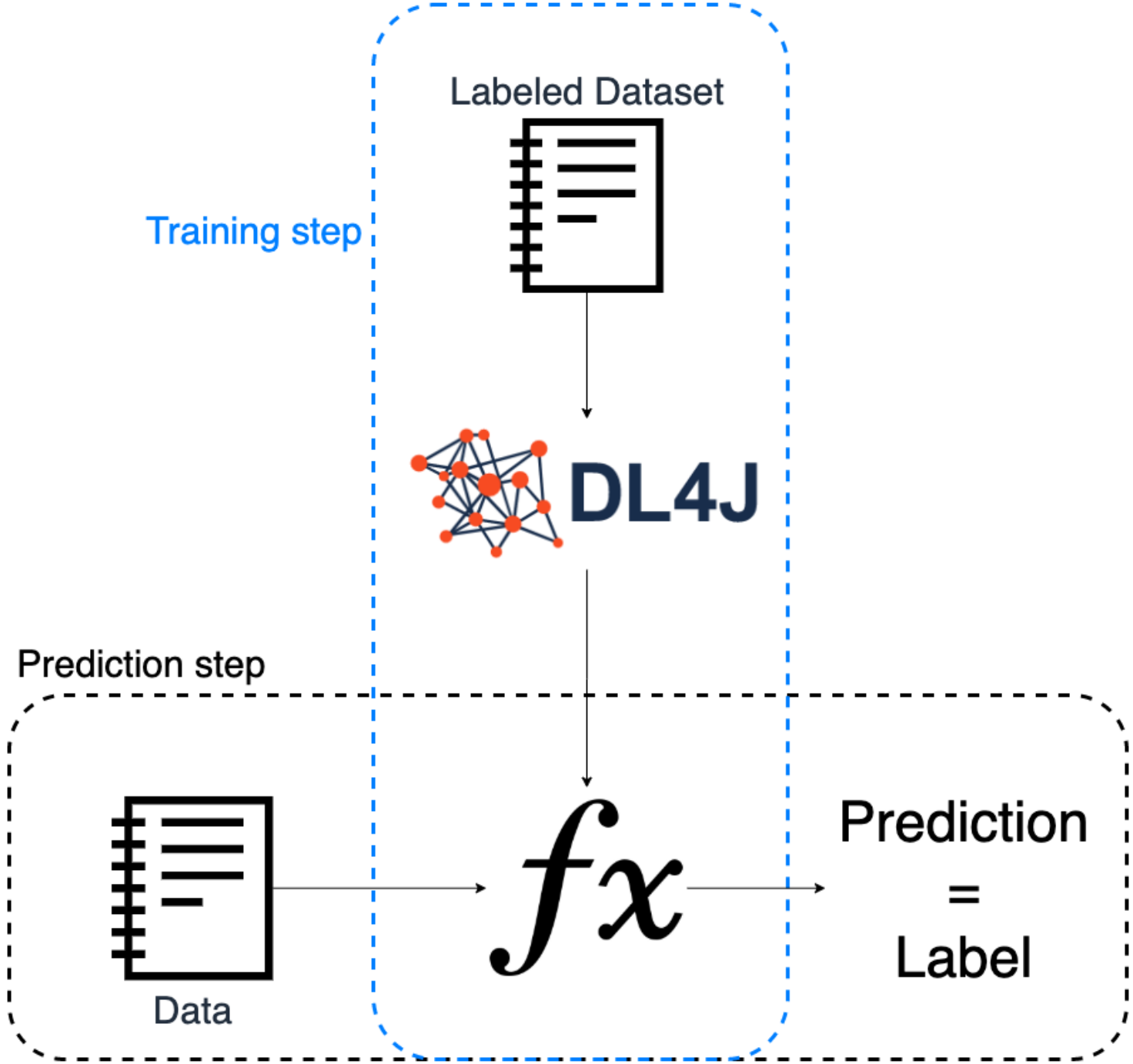


Labeled dataset

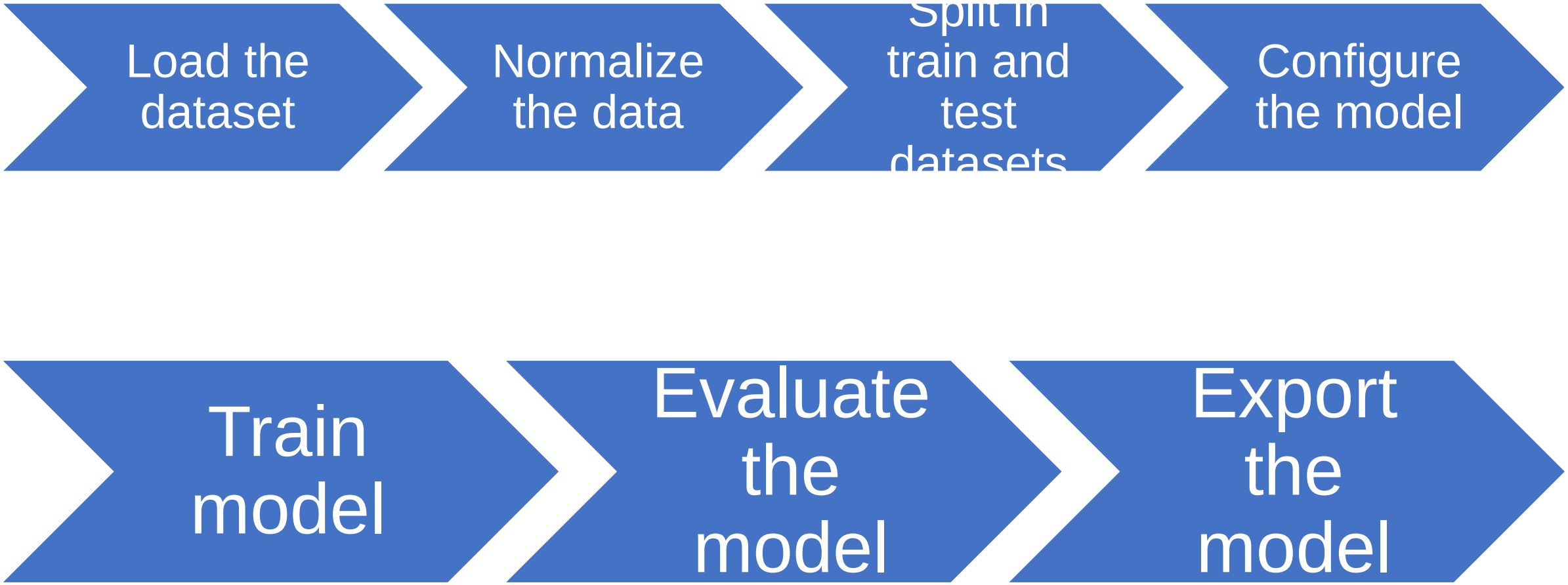
Problem description

- 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

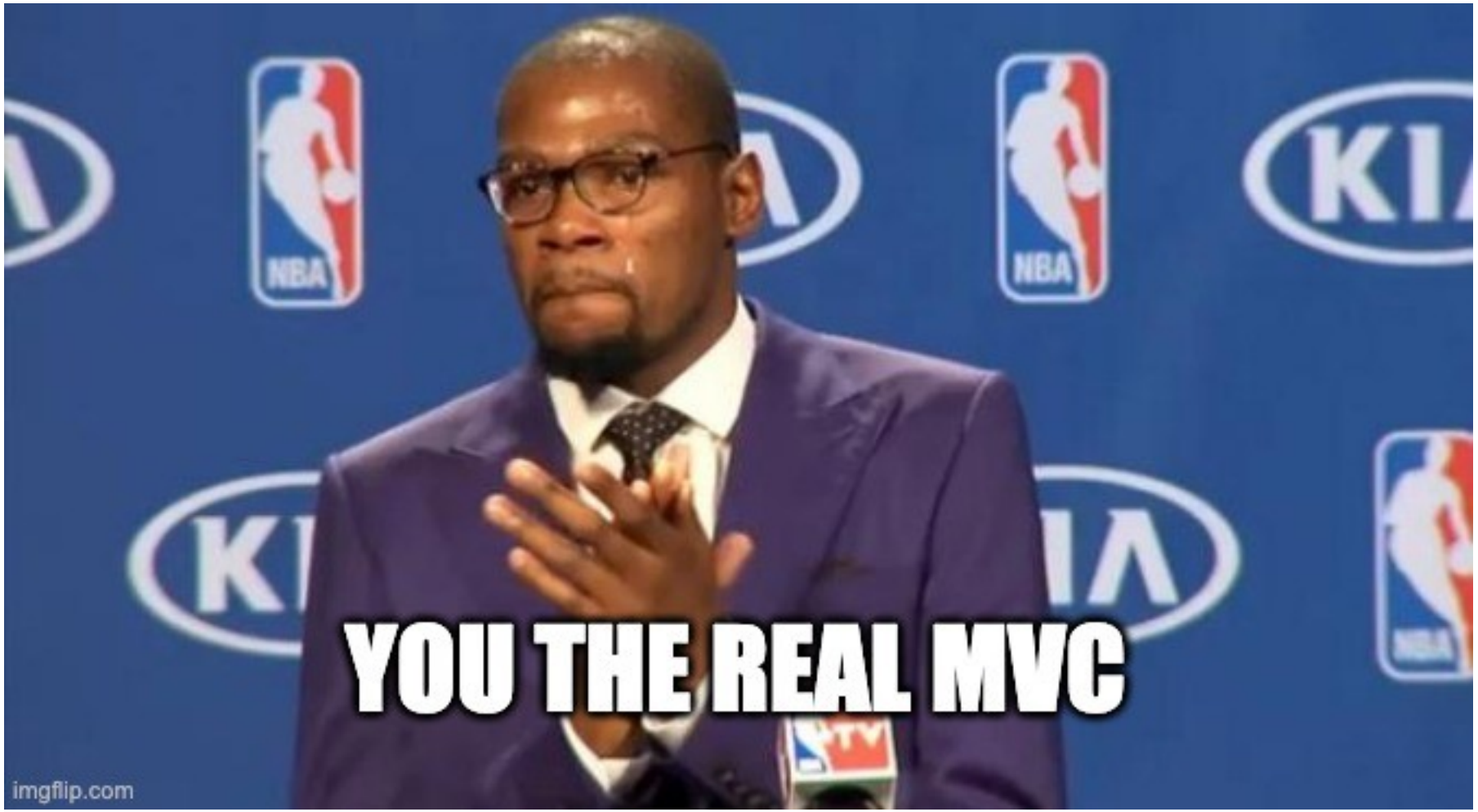
Source: University of California, Irvine , Iris Data Set. https://archive.ics.uci.edu/ml/datasets/iris


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();
```

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();
}
```

```
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);
}
```

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 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;
}
```

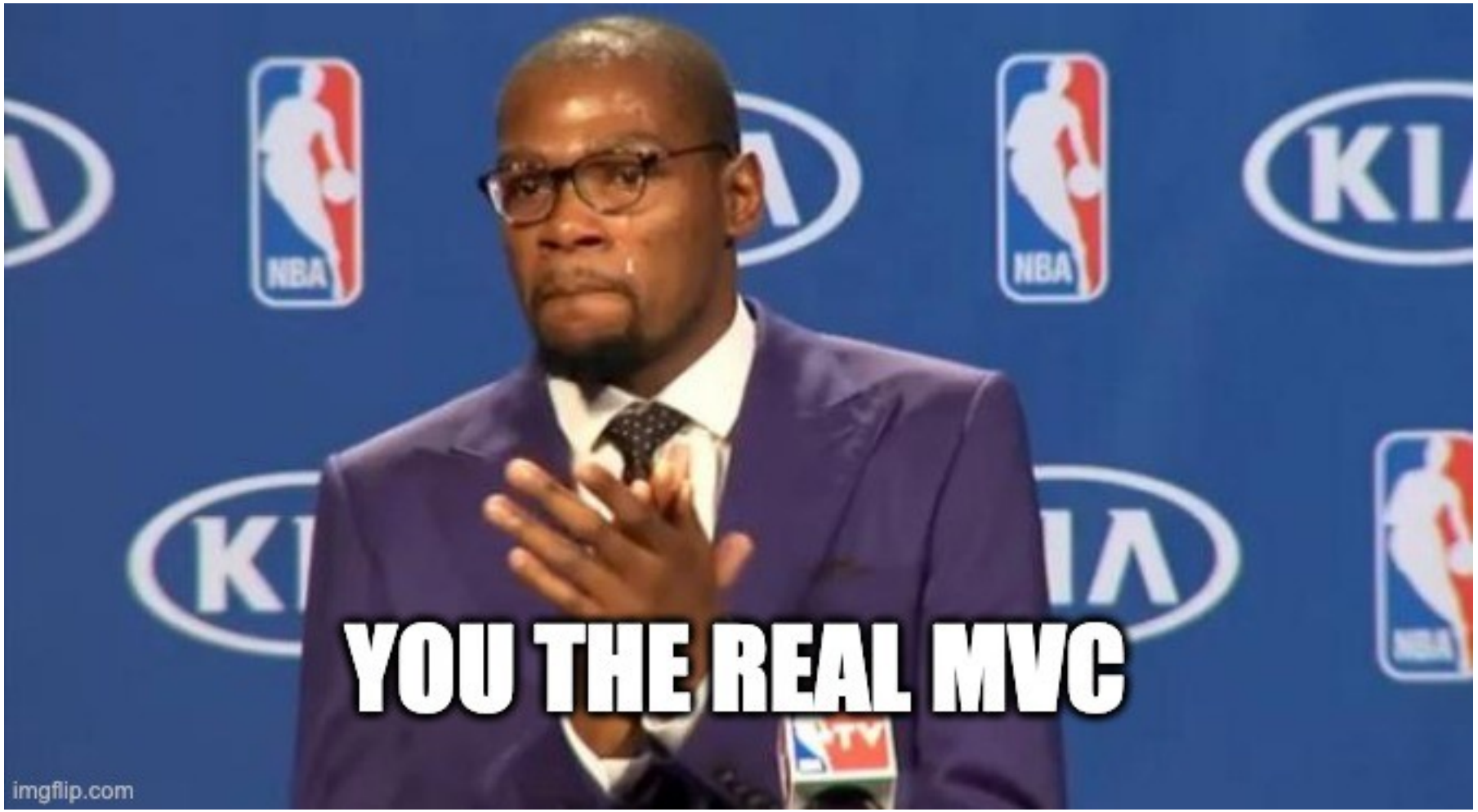
Load the dataset → Normalize the data → Split in train and test datasets → Configure the model → Train model → **Evaluate the model** → Export the model

```
=======================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;
}
```


```
// 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);
```


```java
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

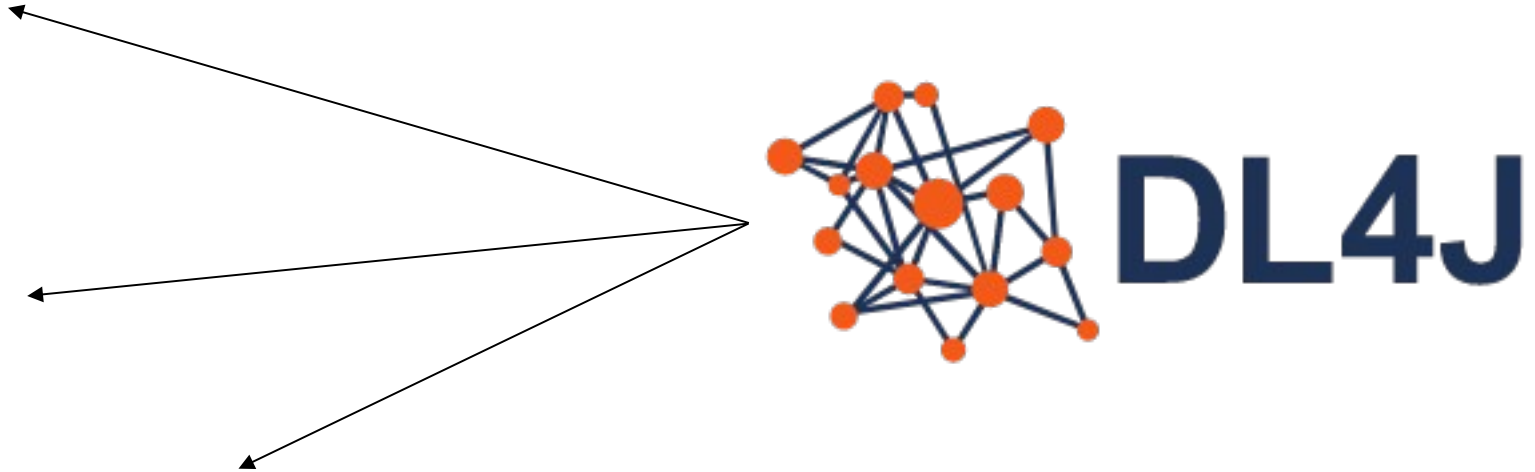
Machine Learning Approaches

- Supervised
  - Classification
  - Regression
- Unsupervised
- Reinforcement learning

Source: Wikipedia, Machine Learning. https://en.wikipedia.org/wiki/Machine_learning , retrieved 08.2020

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

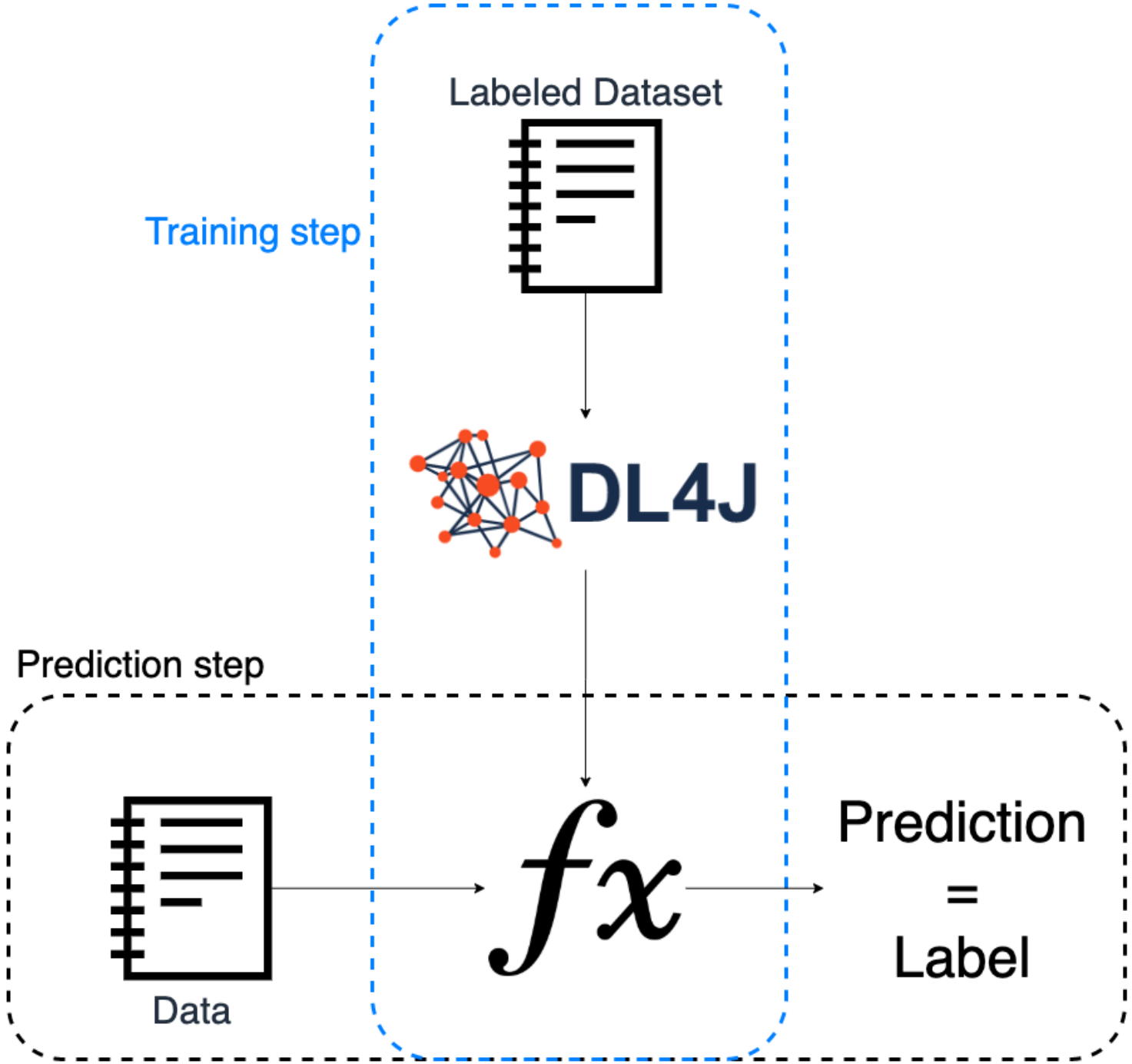
Import models

Deeplearning4j integrations

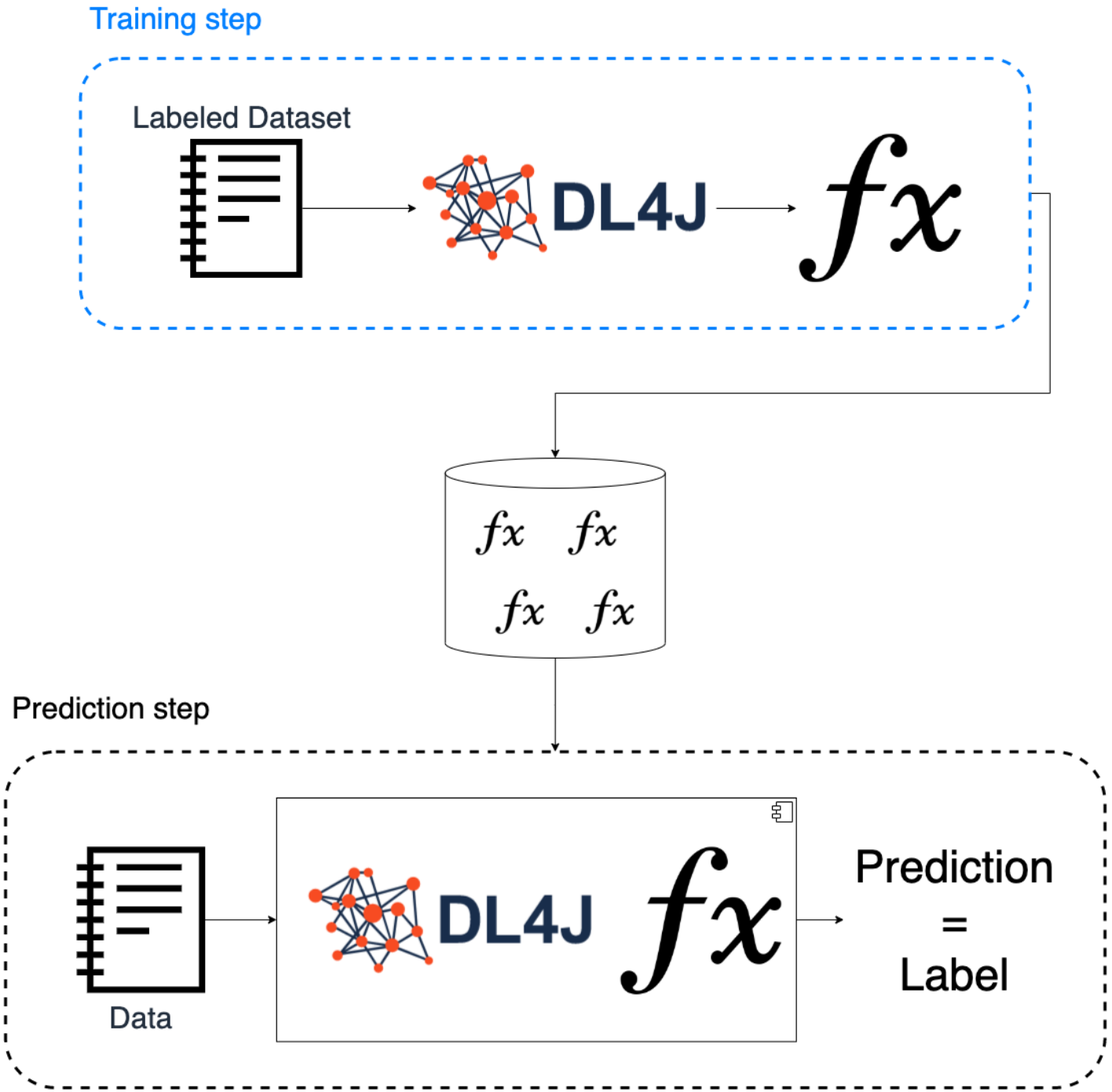
Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML.

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

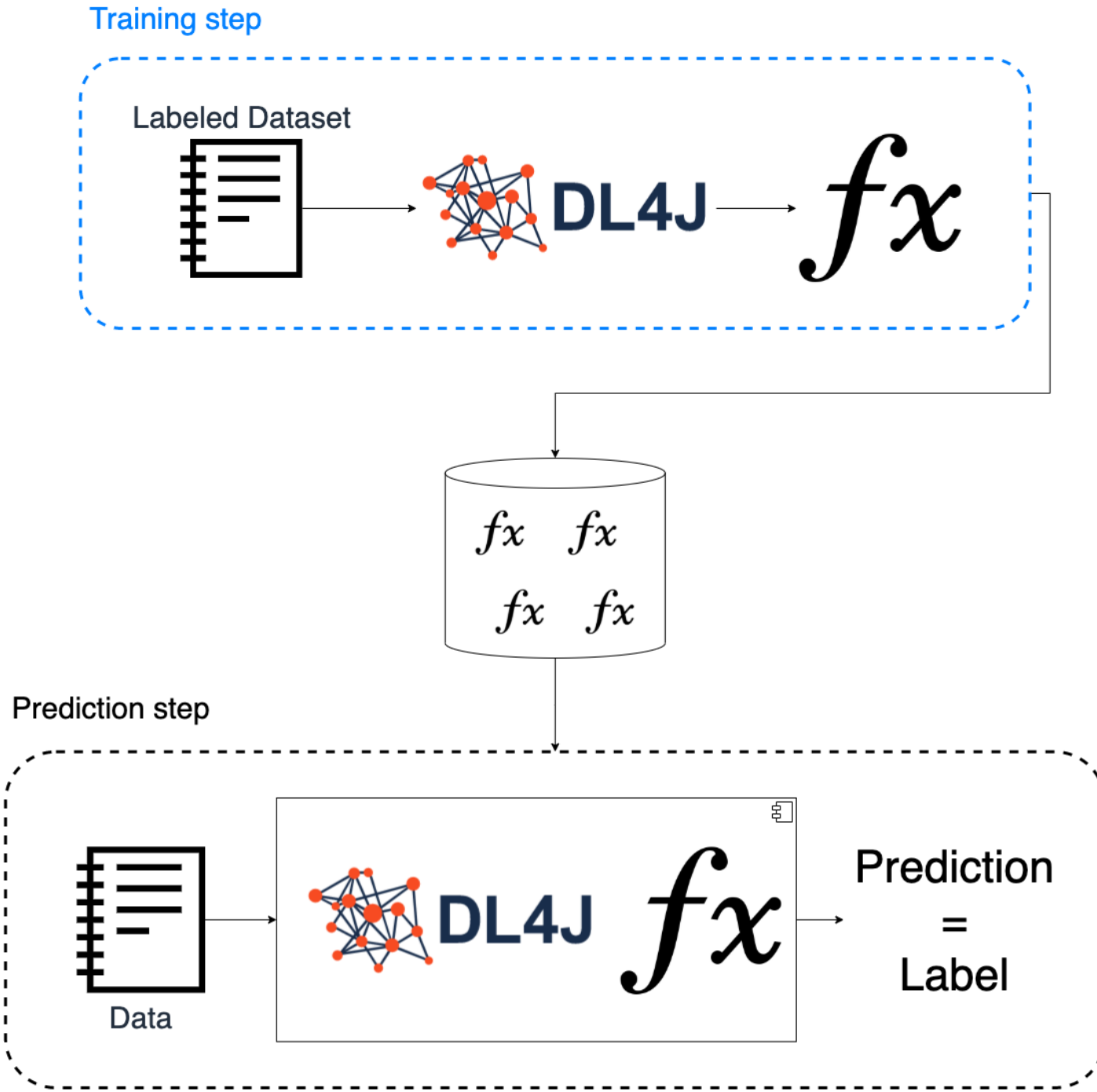
One step back



Decoupled in time!



* JVM Language

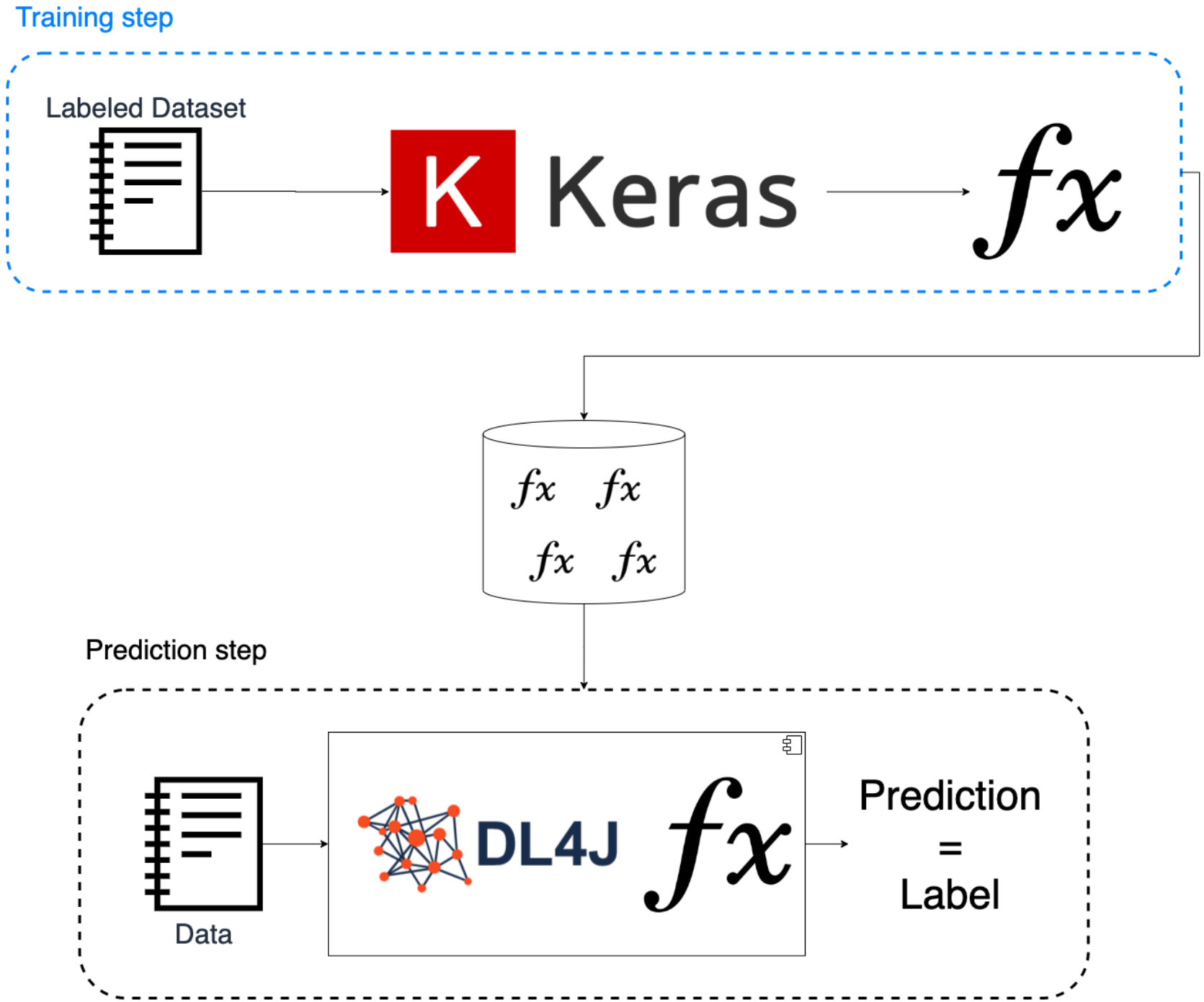


DL4J + Keras

=

Flexibility

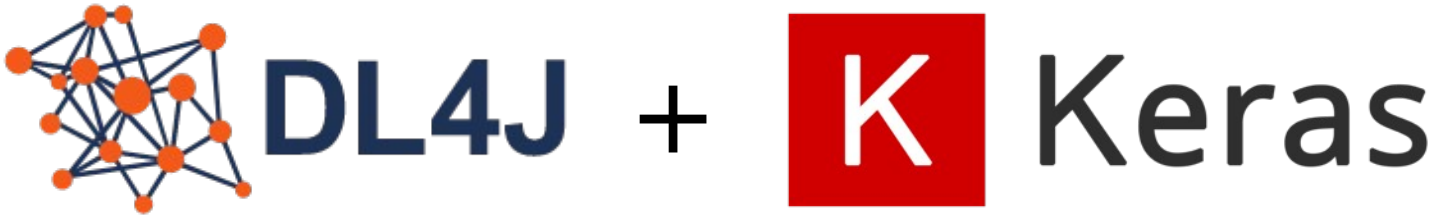

python


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



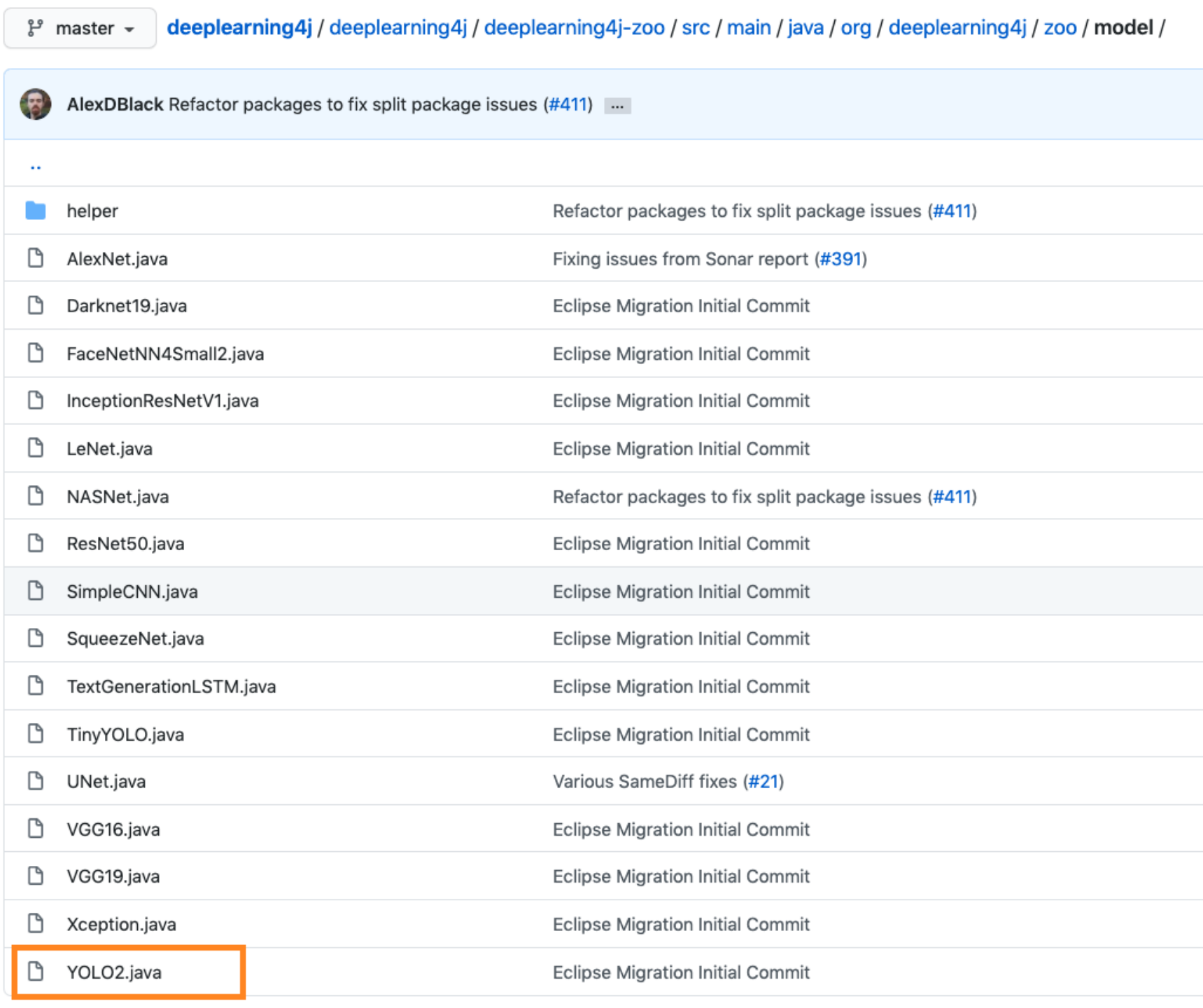
=

Flexibility

Image Classification

- Approach:
  - Deeplearning4j’s Model Zoo
  - OpenCV

Deeplearning4j's Model Zoo




YOLO

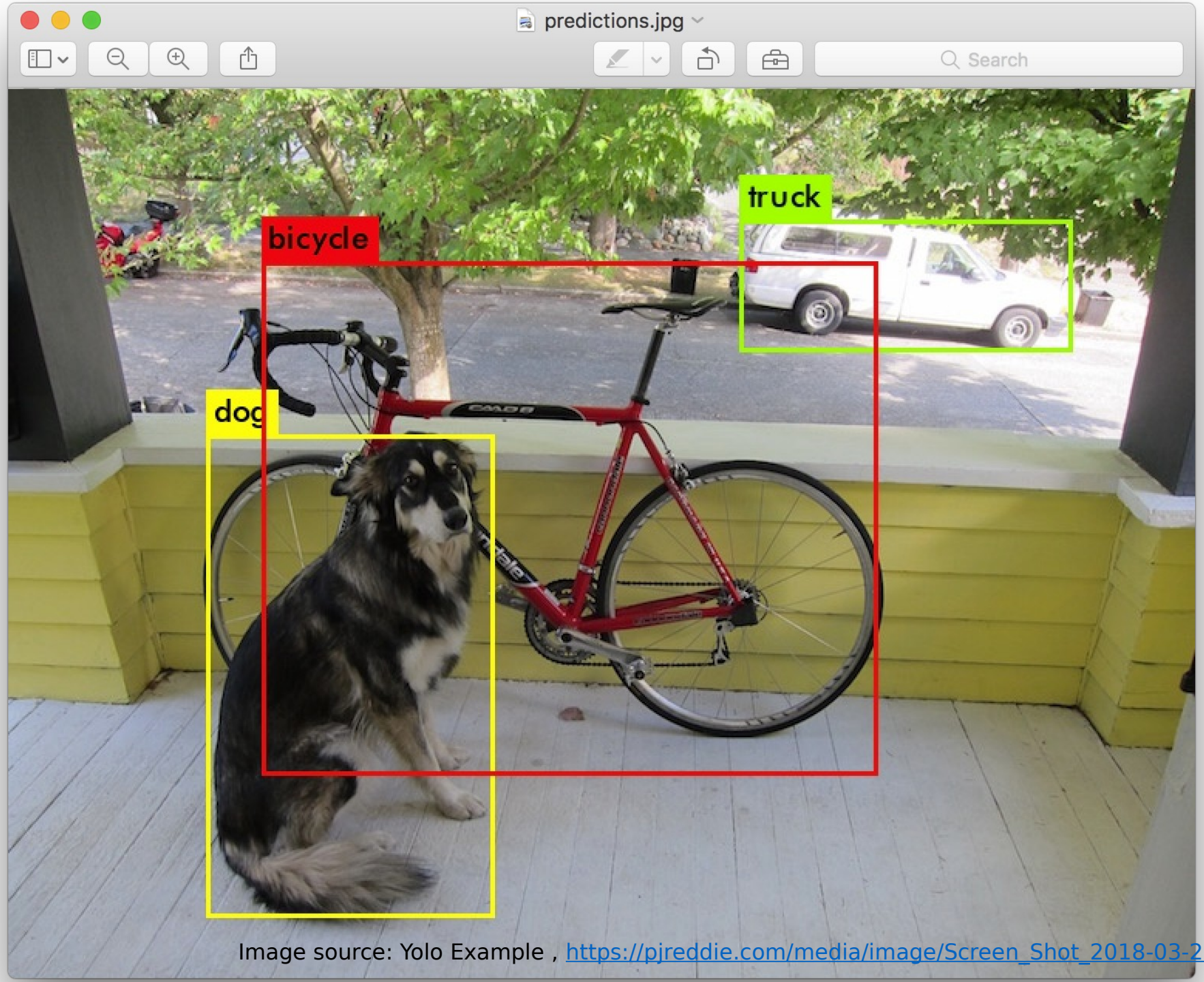
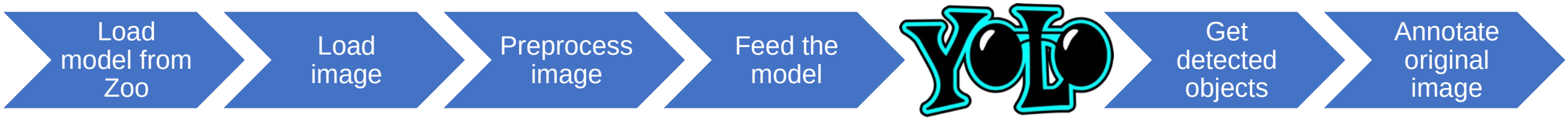


Image source: Yolo Example , https://pjreddie.com/media/image/Screen_Shot_2018-03-24_at_10.48.42_PM.png

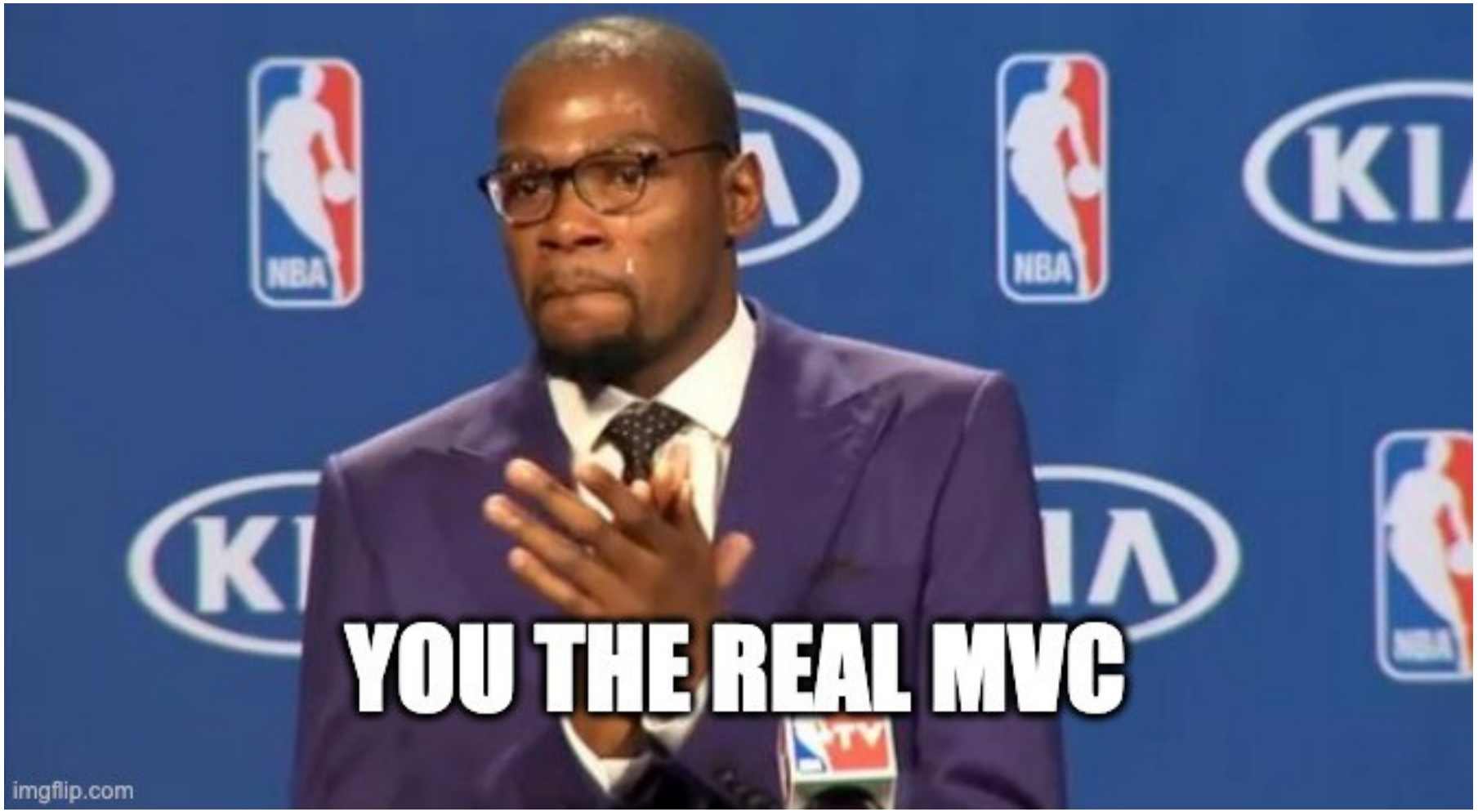
YOLOv2 Available Labels

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;
    }
```

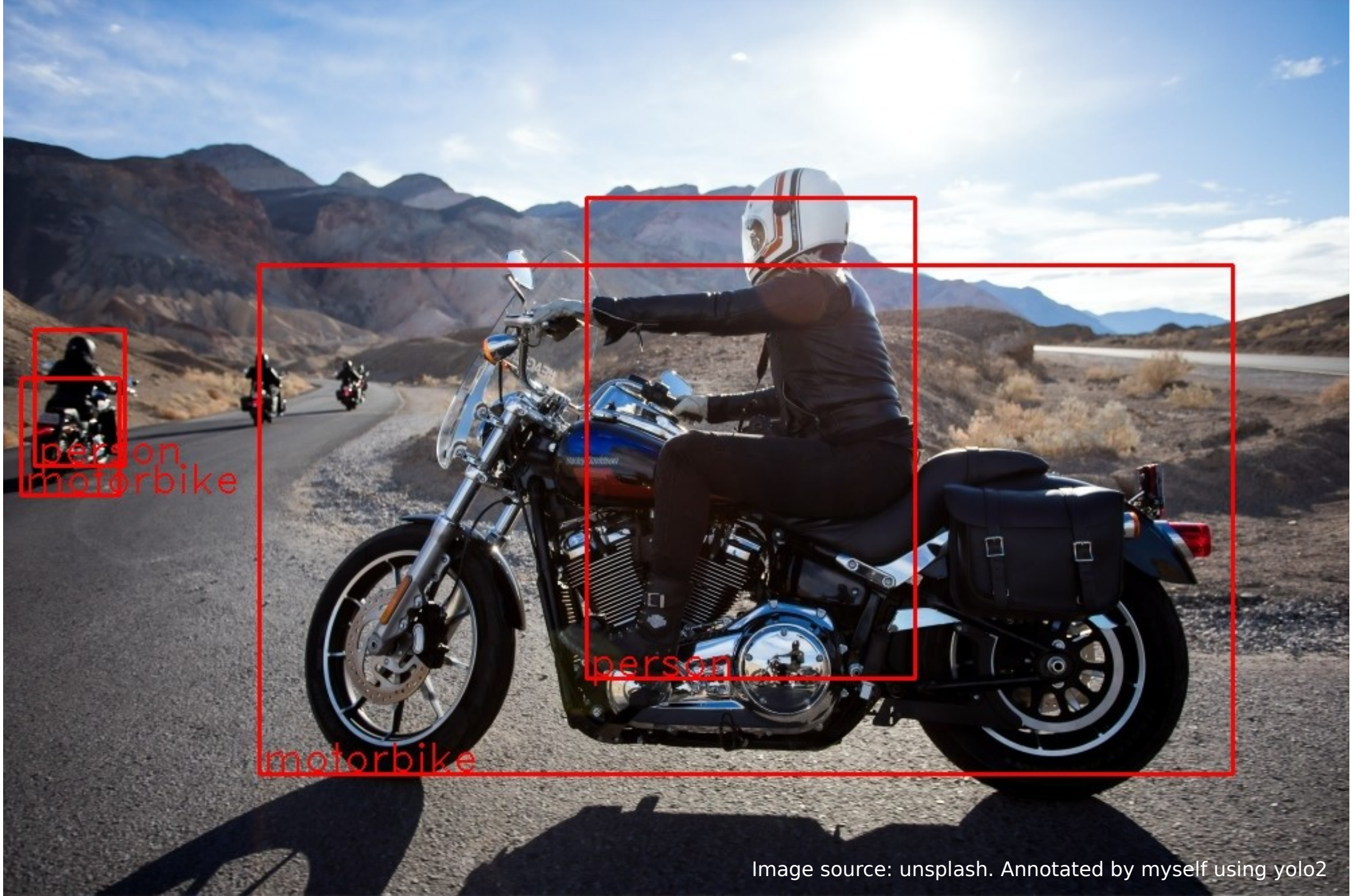


Image source: unsplash. Annotated by myself using yolo2

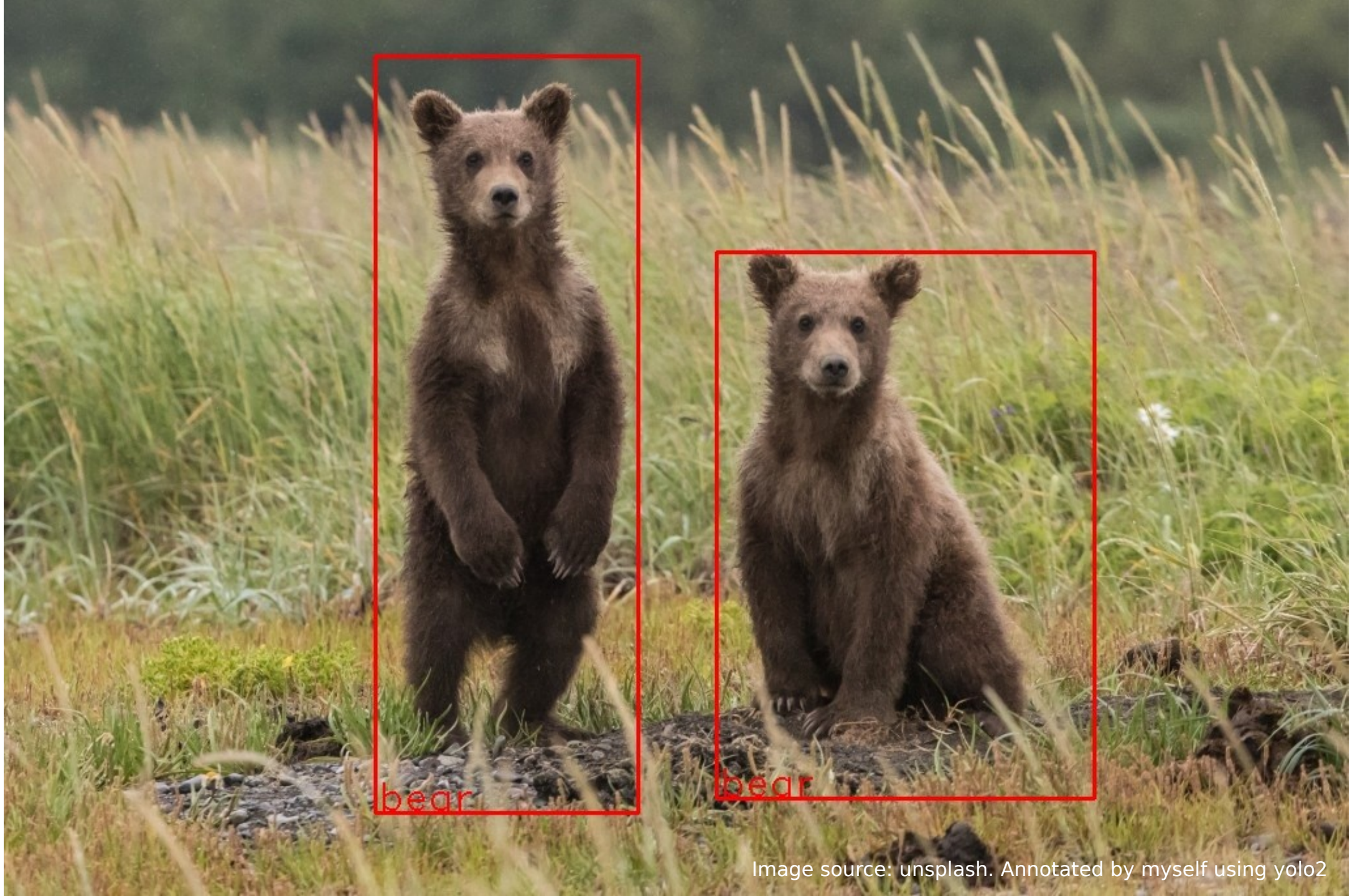


Image source: unsplash. Annotated by myself using yolo2

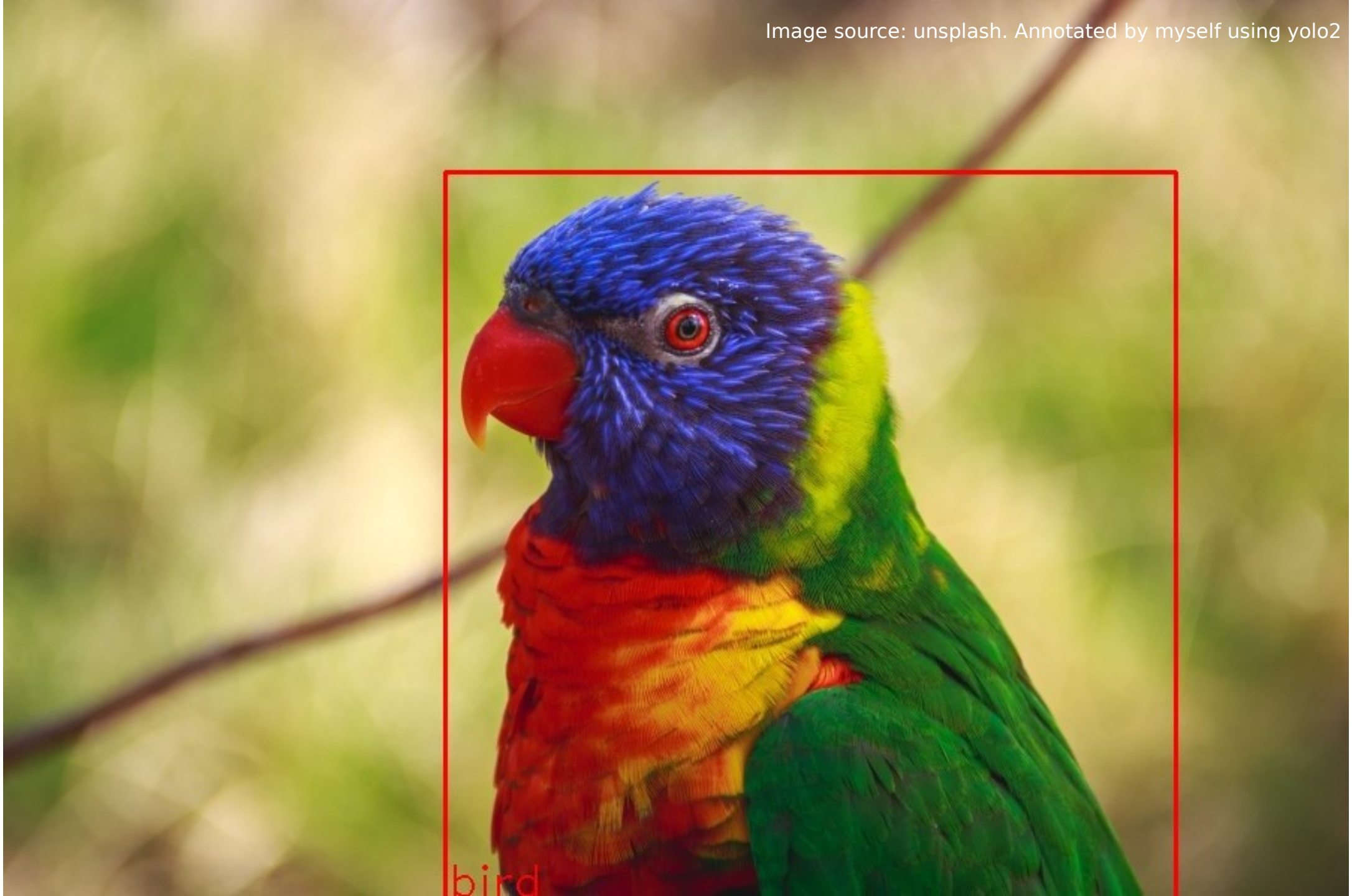


Image source: unsplash. Annotated by myself using yolo2
bird

Image source: unsplash. Annotated by myself using yolo2

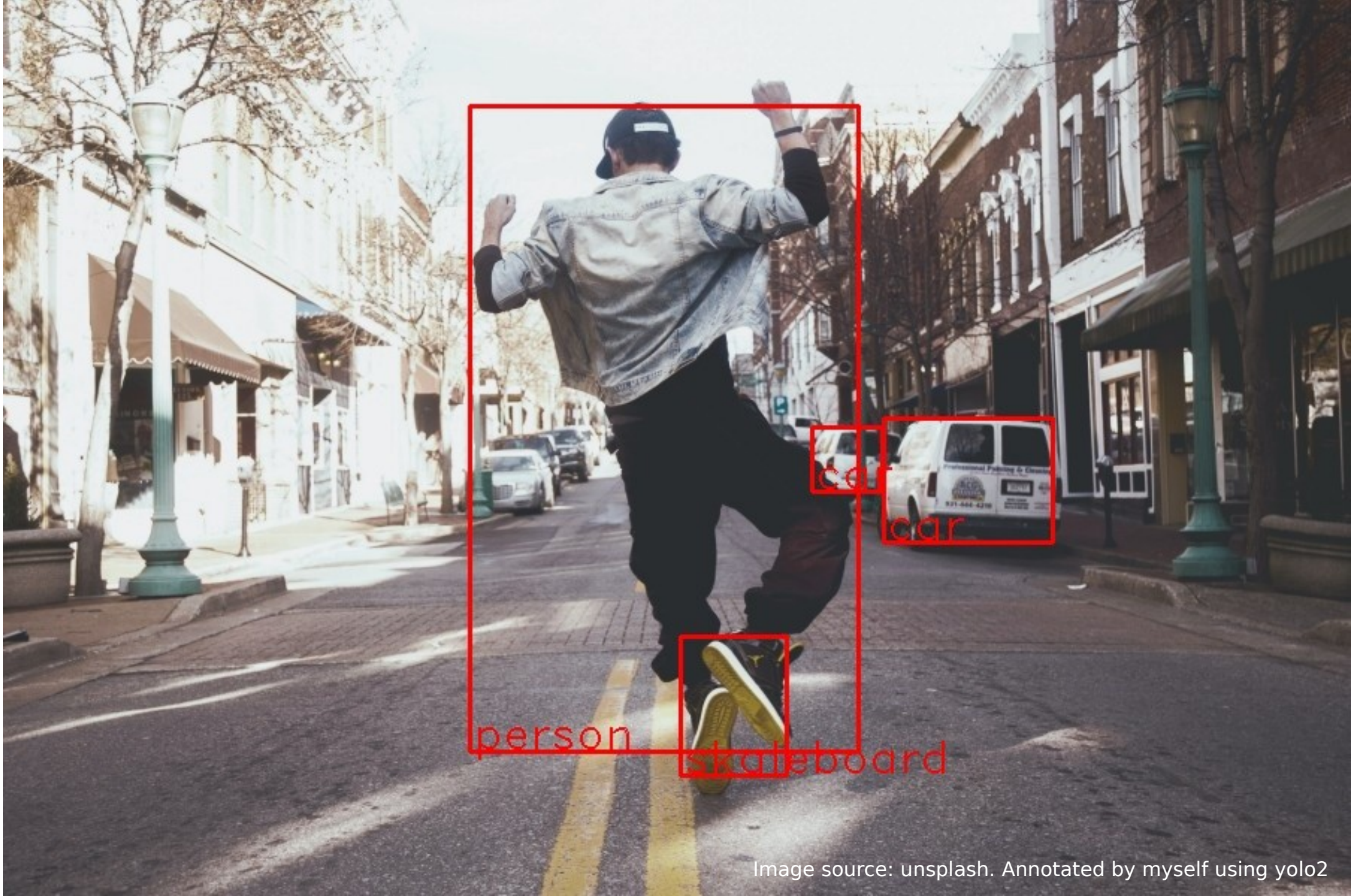


Image source: unsplash. Annotated by myself using yolo2

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

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

Test a lot!

Agenda

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

Scan the QR code to get all the relevant links!

Scan the QR code to get all the relevant links!

How do you want to use Deeplearning4j?

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!