This is the code for the Java implementation of the MLP from Neuroph Studio, I will first try with supervised learning, then I will try unsupervised learning, if the results will be promising, I will tweak them and only then implement in Android (doing it before wouldn’t make sense as the load on the android emulator would be too big). The training set will consist of 40-50 images (half melanoma half non-melanoma) and then will be checked against 20 images, % will be calculated based on correct answers (1-melanoma , 0- nonmelanoma)

package org.neuroph.samples;

import java.util.Arrays;  
import org.neuroph.core.NeuralNetwork;  
import org.neuroph.nnet.MultiLayerPerceptron;  
import org.neuroph.core.data.DataSet;  
import org.neuroph.core.data.DataSetRow;  
import org.neuroph.util.TransferFunctionType;

public class MelanomaMultiLayerPerceptronSample {

public static void main(String[] args) {

// create training set (A,B,C,D are the attributes of each melanoma photo so 4 inputs, only 1 output where 0 is negative and 1 is positive for melanoma)  
DataSet trainingSet = new DataSet(4, 1);  
trainingSet.addRow(new DataSetRow(new double[]{A1, B1, C1, D1}, new double[]{0}));  
trainingSet.addRow(new DataSetRow(new double[]{A2, B2, C2, D2}, new double[]{1}));  
trainingSet.addRow(new DataSetRow(new double[]{A3, B3, C3, D3}, new double[]{1}));  
trainingSet.addRow(new DataSetRow(new double[]{A4, B4, C4, D4}, new double[]{0}));

// create multi layer perceptron  
MultiLayerPerceptron myMlPerceptron = new MultiLayerPerceptron(TransferFunctionType.TANH, 2, 3, 1);  
// learn the training set  
myMlPerceptron.learn(trainingSet);

// test perceptron  
System.out.println("Testing trained neural network");  
testNeuralNetwork(myMlPerceptron, trainingSet);

// save trained neural network  
myMlPerceptron.save("myMlPerceptron.nnet");

// load saved neural network  
NeuralNetwork loadedMlPerceptron = NeuralNetwork.createFromFile("myMlPerceptron.nnet");

// test loaded neural network  
System.out.println("Testing loaded neural network");  
testNeuralNetwork(loadedMlPerceptron, trainingSet);

}

public static void testNeuralNetwork(NeuralNetwork nnet, DataSet testSet) {

for(DataSetRow dataRow : testSet.getRows()) {

nnet.setInput(dataRow.getInput());  
nnet.calculate();  
double[ ] networkOutput = nnet.getOutput();  
System.out.print("Input: " + Arrays.toString(dataRow.getInput()) );  
System.out.println(" Output: " + Arrays.toString(networkOutput) );

}

}

}