```
target    = [0,     0,     0,     0,     0,     0,     0,     1,     0,     0]
predicted = [0.01,  0.00,  0.02,  0.01,  0.03,  0.01,  0.00,  0.87,  0.02,  0.03]
outputError = [0.01,  0.00,  0.02,  0.01,  0.03,  0.01,  0.00,  -0.13,  0.02,  0.03]
```

predicted: by softmax, sums to 1
output error: predicted – target

```
        private double[] softmax(double[] inputs) {
            double[] output = new double[inputs.length];
            double max = inputs[0];
            // Find max for numerical stability
            for (int i = 1; i < inputs.length; i++) {
                if (inputs[i] > max) max = inputs[i];
            }
            // Compute exp and sum
            // Use max to avoid NaN cases
            double sum = 0;
            for (int i = 0; i < inputs.length; i++) {
                output[i] = Math.exp(inputs[i] - max);
                sum += output[i];
            }
            // Normalize
            // Notice that max
            for (int i = 0; i < inputs.length; i++) {
                output[i] /= sum;
            }
            return output;
        }

        int predict(double[] x) {
            double[] output = forward(x)[1];
            int best = 0;
            for (int i = 1; i < output.length; i++) {
                if (output[i] > output[best]) best = i;
            }
            return best;
        }

    static double evaluateAccuracy(NeuralNetwork net, double[][] images, int[] labels) {
        int correct = 0;
        for (int i = 0; i < images.length; i++) {
            if (net.predict(images[i]) == labels[i]) correct++;
        }
        return (double) correct / images.length;
    }

    static double crossEntropyLoss(double[] predicted, double[] target) {
        double loss = 0.0;
        double epsilon = 1e-12; // avoid log(0)
        for (int i = 0; i < predicted.length; i++) {
            loss -= target[i] * Math.log(predicted[i] + epsilon);
        }
        return loss;
    }
```

totalLoss += crossEntropyLoss(predicted, target);

avgLoss = totalLoss / trainImages.length;