```
#include <iostream>
#include "NNFactorization.h"
#include <chrono>

int main(int argc, const char * argv[]) {
    // insert code here...

    using namespace TopicModels;

    int numTrials{1};
    int numForwardProp{100};
    int numBackProp{120};
    float lambda{.1};//0 = no supervision,  any bigger zero = semi -supervision
    float percentUnlabled{99};
    RVectorXf perCorrNoBack(numTrials);
    RVectorXf perCorr(numTrials);

    for (int n : boost::irange(0, numTrials)){

        const int numTopics{30};
        int inputNumLayers{3};  //  Change this from 1 to 27.
        int inputPoolSize{75};  // 0 means no pooling, 1 means 1*2+1 = 3 paragraphs, 2 = 2*2+1
 = pool over 5 paragraphs
        int inputPoolSkip{1};  // Cannot change this for now!!!!!!!!!

        RMatrixXf spectrograms;
        RMatrixXf numInClassf;
        RMatrixXi numInClass;
        RMatrixXi numInClassPool;
        RVectorXi classBoundaries;
        loadMatrix(inputPath + "textSpectrogram/dataAudioT1k.txt",  3252, 2049, spectrograms);

        loadMatrix(inputPath + "textSpectrogram/numClassPoolAudio1k.txt", 1, 12, numInClassf);

        numInClassPool = numInClassf.cast<int>();
        loadMatrix(inputPath + "textSpectrogram/numClassAudio1k.txt", 1, 3, numInClassf);
        numInClass = numInClassf.cast<int>();

        assert(numInClassPool.sum() == numInClass.sum());
        CreateLabelMatrix lm(numInClass);
        createClassBoundaries(numInClassPool, classBoundaries);

        RMatrixXf knownLabels;
        addMissingLabelsRandomBoundaries(classBoundaries, lm.getClassMatrix(), knownLabels, pe
rcentUnlabled);
        SemiSupNNMF ssnnmf(spectrograms, lm.getClassMatrix(), knownLabels, numTopics, lambda);
  //Comment this to run here for one layer=======
        ssnnmf.run(numForwardProp);
                                                                                //comment here
as well for one layer====
        RMatrixXf cm;
        std::cout << "Classification SSNMF = " << classify(lm.getClassMatrix(), ssnnmf.getLabe
lRecon()) << std::endl;
        confusionMatrix(lm.getClassMatrix(), ssnnmf.getLabelRecon(), cm);
        std::cout << "Confusion Matrix SSNMF = " << std::endl << cm << std::endl;
        ssnnmf.saveAll();
        DeepNNMF nnmf(spectrograms, lm.getClassMatrix(), classBoundaries, inputNumLayers, inpu
tPoolSize, inputPoolSkip, numTopics, lambda);
        //        nnmf.initialize();
        nnmf.addMissingLabelsRandomBoundaries(classBoundaries, percentUnlabled);
        nnmf.forwardProp(numForwardProp);  // Forward prop is really just initialization.
```

```
        perCorrNoBack[n] = confusionMatrix(lm.getClassMatrix(), nnmf.getLabelRecon(), cm);
        std::cout << "Confusion Matrix No Back: " << std::endl;
        std::cout << cm << std::endl;
        std::cout << "Percent Correct No Back= " << perCorrNoBack[n] << std::endl;
        nnmf.saveAll("last_noBack_", inputNumLayers - 1);
        nnmf.saveAll("first_noBack_", 0);
        saveMatrix(machineLearningPath + "_labelRecon_noBack", nnmf.getLabelRecon());
        nnmf.backProp(numBackProp);  // Back prop includes neural net ideas.
        //
        nnmf.saveAll("last_", inputNumLayers - 1);
        nnmf.saveAll("second_", 1);//REMOVE NEXT TIME === JUST FOR SAVING SECOND LAYER
        nnmf.saveAll("first_", 0);
        saveMatrix(machineLearningPath + "_labelRecon", nnmf.getLabelRecon());
        nnmf.saveBackRecon();



        perCorr[n] = confusionMatrix(lm.getClassMatrix(), nnmf.getLabelRecon(), cm);
        std::cout << "Percent Correct = " << perCorr[n] << std::endl;
        std::cout << "Confusion Matrix: " << std::endl;
        std::cout << cm << std::endl;
        //        std::cout << "Trial Number = " << n << std::endl << std::endl;
    }

    std::cout << perCorr << std::endl << std::endl;
    std::cout << "Average Correct No Back= " << perCorrNoBack.sum()/static_cast<float>(numTria
ls) << std::endl;
    std::cout << "Average Correct = " << perCorr.sum()/static_cast<float>(numTrials) << std::e
ndl;
    std::cout << "lambda = " << lambda << std::endl;

    return 0;
}
```