Introduction to Machine Learning
academic year 2025/2026
dr hab. inż. Agnieszka Jastrzębska

IML project description

The story

The goal of the project is to prepare a machine learning module that can be hypothetically used in an automated, voice-based intercom device. Imagine that you are working in a team of several programmers and the access to your floor is restricted with doors. There is an intercom that can be used to open the door. You are implementing a machine learning module that will recognize if a given person has the permission to open the door or not. To simplify this problem, we interpret this as a binary recognition problem. Class 1 is made of allowed persons. Class 0 is made of not allowed persons. The medium used to distinguish between the two classes is voice. Thus, the project to be delivered in in fact a voice recognition project. To shift the focus to convolutional neural networks, we assume that the voice recording must be turned into spectrograms in the pre-processing stage. Thus, the project to be delivered is in fact a voice recognition project but implementing it will require techniques that can also be applied to different kinds of image processing.

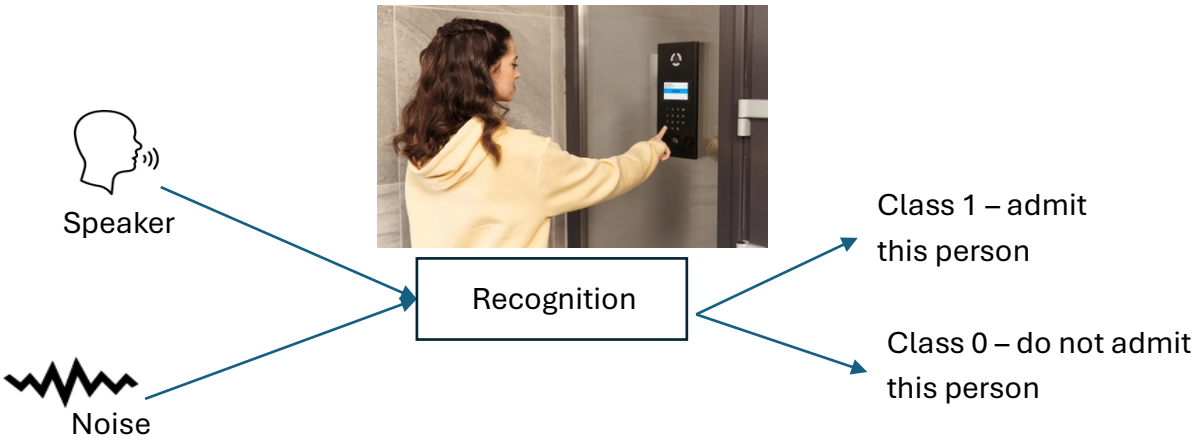
Project components - deliverables

The delivered project will be composed of three key components:

1. Code that documents the development of the best-working voice recognition module. These codes should cover the procedures of data loading, preprocessing, training, evaluation, and comparison of multiple models. Plots and output values that document the findings should be stored either as images in the project directory, or as output of cells, if Jupyter Notebook is used.
2. A simple Jupyter-notebook-based program that will be used to interact with prepared codes. This program uses only one, best recognition model.
3. Report discussing your work and the results. Should be no more than 10 pages long.

All should be implemented in Python. Code must be documented with docstrings, Jupyter markdown cells and/or comments. It can be a composition of Jupyter notebooks and python codes or just one type of those. The code must be reproducible. The programs will be running on laptops with microphones. If somebody does not have a microphone in their laptop, please contact me and we will arrange a USB microphone in such a case.

A simple diagram below shows the idea behind this project.



The obvious issue hindering the recognition capability is background noise. Its influence must be studied in the report and appropriate codes must be prepared to inspect this issue.

There are two essential metrics to be measured separately on train, validation, and test sets:

- False Acceptance Ratio – the total number of incorrectly "accepted" instances from Class 0 divided by the number of samples from Class 0.
- False Rejection Ratio – the total number of incorrectly "rejected" instances from Class 1 divided by the number of samples from Class 1.

Both are equally important and should not be far away from each other.

Additional measures such as measures based on SNR or WER are allowed as well.

The data

Its your task to collect suitable data for this project. You ought to have two classes in the data: class 0 (not allowed) and class 1 (allowed). Note that the recordings can be as short as 3 seconds (even 1s-long can be used) and varied. The dataset cannot be trivial (like DAPS). The recommended path is to create the dataset on your own.

Use case scenarios

Please see below essential use cases.

As a final user:

1. I am initializing the listening procedure. The program listens to me with the use of a laptop microphone and detects the class. Detected class and, optionally,

confidence level, is displayed on the screen. The Jupyter-notebook program is used in this scenario.

As a programmer:

1. I am running all the actions that start from loading of voice recordings and end with data converted to spectrograms. The use of spectrograms is obligatory.
2. I am running all the actions that lead to data cleaning. The step ends with normalized data without duplicates. Other data manipulations are allowed as well, for instance cutting of tracks.
3. I am running all the actions that lead to the successful training of all considered Convolutional Neural Networks. Networks are serialized to the hard drive and can be loaded from there. The use of CNNs is obligatory. The networks must be trained from scratch on spectrograms generated from the audio data. The training process must be well-documented. To guarantee a high grade, several techniques/architectural components must be used, and their impact on the training/validation performance tested. Some examples:
    a. data augmentation, including quality enhancements: removing silent passages from the training data, choosing the length of a single excerpt, input normalization
    b. changing the number and size of the layers
    c. different optimizers (SGD, Adam), learning rate schedule, weight decay
    d. batch normalization (before vs after activations)
    e. skip connections
    f. dropout
    g. Monte Carlo dropout for uncertainty measurement
    h. activation functions (ReLU, sigmoid) tested against different weight initializations (Xavier, He, uniform)
    i. a comparison to pre-trained models (transfer learning, fine-tuning)
    j. any nonconservative technique will be a plus.
4. Data exploration should not be limited to the first part of the project, but follow and complement the model development. Models, once trained, can be used to explore the dataset again and pinpoint samples of interest. Some example methods of data exploration:
    a. manual comparison of speakers and built-in augmentation types
    b. calculating statistics based on spectrograms
    c. most and least difficult samples, samples that are learned early. samples that different architectures struggle with
    d. samples that are similar to other samples from the model's perspective (e.g. they produce similar activations in the last layer)

5. I am running the actions that allow to compare multiple considered CNNs. The comparison is text-based, plot-based, and tables-based. Metrics should be explained and clearly presented.
6. I am running the actions that evaluate quality of a single neural network with the use of multiple test scenarios. Test scenarios must cover the influence of noise and reverberation on the recognition quality. Methods for generating noise and reverberation can be very crude.
7. If I manage to overcome a significant roadblock during training, I can document the findings and the suspected root cause for some extra points, even if the root cause analysis is off the mark. Example: my training process was experiencing huge spikes of training loss at the beginning of each epoch. I investigated the data loader and realized that the last few batches consisted only of samples from a single class. I think this caused the model to drift towards this class, which impaired it when the balanced data returned in the next epoch. After shuffling the data properly, the effect subsided.

Aspects to cover in the code, and then in the report:

- data augmentation, including quality enhancements: removing silent passages from the training data,
- choosing the length of a single excerpt, input normalization,
- changing the number and size of the layers,
- different optimizers (SGD, Adam), learning rate schedule, weight decay,
- batch normalization (before vs after activations),
- skip connections,
- dropout,
- Monte Carlo dropout for uncertainty measurement,
- activation functions (ReLU, sigmoid) tested against different weight,
- initializations (Xavier, He, uniform),
- a comparison to pre-trained models (transfer learning, fine-tuning).

Your project needs to cover any eight (8) of the above-listed points to be considered for 5.0 grade, any five (5) to be considered for 4.0 grade, and any three (3) to be considered for 3.0 grade.

Deliverables in time

Milestone #1 – reading the data, initial exploratory data analysis, creating spectrograms for at least a part of the data (the raw recordings, preferably), a first, very simple network trained to at least 0.6 macro-averaged f1-score on the training data, part of the dataset chosen for the test set, no report

Points to get: 0 to 3 (Reading the data and creating spectrograms: 0 or 1, simple exploratory data analysis: 0 or 1, creating a simple network and making it train: 0 or 1). The network for this milestone can be an unchanged model copied from a tutorial, like this one: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html (if you use pytorch), as long as it trains

Milestone #2 – in-depth data exploration, cleaning and pre-processing with the aim of making the model better. Different variants of models created and evaluated, no report, no interactive notebook for the end user yet

Points to get: 0 to 2 (0 or 1 or 2 points for progress. If the experiments and data exploration are enough for the whole project, you will be informed and granted corresponding points from Milestone #3 in advance).

Milestone #3 – complete solution, report

Points to get: 0 to 5 (improvements in comparison to the value delivered for Milestone #2 – 0 or 1 or 2 or 3 points, report 0 or 1 or 2 points)

The content of the code can be revised at all stages (you can go back to stage 1 and change it later).