Prestudy Plan

Helmer Nylén
helmern@kth.se

2020–04–21

This is an informal plan for the prestudy.

What's the point?

- Solving a simplified version of the bigger task by imposing a set of constraints on the problem. In the prestudy we assume that the noise is additive and independent of the speech.
- Laying a foundation which can be expanded upon for the full task.
- Familiarizing myself with terminology and tools needed in the full task.
- Testing out different methods for classifying the added noise (see below).
- Selecting one of the methods for use in the full task.

What needs to be done?

- **Find noise data.** I have some 10 samples of electrical hum and 15 of AC noise. Noisex-92 can also be used for comparison. I am a bit concerned that I don't have enough hum/AC noise to properly train a network, but we'll see.
- **Find speech data.** TIMIT should work for the prestudy. If another corpus is found it should be easy to switch.
- **Combine the two into noisy recordings.** I was planning to do this based on the Audio Degradation Toolbox [1], since it can be used to create other degradations as well. Someone wrote a Python implementation and I wrote a script using that, but as it is apparently not equivalent to the Matlab original I may resort to using Matlab for dataset preparation.
- **Select features.** MFCCs are widespread in speech recognition and most literature I've read use them for noise/audio scene classification as well [2, 3, 4, 5, 6]. Kaldi can extract these easily, which is convenient. Other features which could be considered are LSFs or LPCs [7, 2, 8], Wavelet transforms [9] (may be used in [10]?), perceptual features, or lower-level features such as spectral flux/centroid/rolloff and others [2, 11]. A more modern approach seems to be to simply perform some transform (FFT, constant-Q transform) and input the entire spectrum into a Convolutional Neural Network [12, 10], but it is not necessarily better.

  While selecting appropriate features can probably be as important as designing the classifier itself, the plan is to stick to MFCCs for the prestudy. However, I'll make sure the code is structured enough that other features, if needed, can be added or replaced later. For example MFCC+Δ is often used or tested against only MFCC.
- **Use Voice Activity Detection?** In the prestudy we assume that the noise in the recording does not depend on the speech. It may then improve performance to filter out the parts of the signal which are dominated by speech sounds. WebRTC is an open-source project for browsers to support, for example, voice communication, and there is a Python wrapper

for its VAD. Using this we can supply only the feature vectors of unvoiced frames to the classifier, which may then yield a better result.

There are not many natural pauses in the recordings in TIMIT, but we can introduce pauses by padding with silence at the beginning and end. The pauses that do exist are annotated, so we can easily find and stretch these.

- **Select classifier algorithms.**
  1. **Hidden Markov Model using Gaussian Mixture Models (GMM-HMM).** These are pretty standard and used in several of the reports I've read.
  2. **HMM using a Generative Model (Gen-HMM).** Proposed in [13]. Similar to the above in that both use HMMs for time-dependence, but we swap out the GMM for a neural network-based model instead.
  3. **Long short-term memory (LSTM).** A type of recurrent neural network (RNN), which are designed to work well on sequential data, with a certain type of unit. A HMM depends only on the previous state but this network can remember relevant information over a longer sequence.
  4. **Convolutional Neural Network (CNN).** While these are often used successfully with images there are also successful applications to sound classification [6][1] (found some other reports but haven't read through them yet). I think it could be interesting to try such a network as they have been around a while and proved successful once computing power caught up.
  5. **Support Vector Machine (SVM).** A straightforward classification technique which aims to find a border between classes of data. Different kernels can be used to allow non-linear borders and a Radial Basis Function kernel is the one I'm considering. Used in [5] as one of three classifiers.

  I also considered a vanilla RNN, an RNN with Gated Recurrent Units, and a Multi-Layer Perceptron, but I felt the other types of RNN are maybe too similar to LSTM to be interesting? It may be worth it to swap the CNN or SVM for the MLP, though.

- **Implement classifier algorithms.** The MFCC feature extraction is more or less done at the moment, so the next step is to implement these. I plan on doing that in the order they are listed.
  1. I'm fairly familiar with GMM-HMMs and there seems to be existing implementations in Python (e.g. in scikit-learn). The code for Gen-HMMs also includes GMM-HMMs for comparison, so I might just use that.
  2. I got the Gen-HMM code up and running but it seems to be tailored towards phoneme recognition, so it may require a bit of tinkering to get it to classify noise instead. Should be doable but might be more of a headache than I'm expecting.
  3. Pytorch has LSTM layers built-in, and a tutorial for beginners, so I should be set. I have also worked with vanilla RNNs before in the Deep Learning course.
  4. Convolutional layers are also included in Pytorch. I don't have much intuition to go on when deciding on the network design so I figured I could use the parameters from an existing report such as [12] (though they use other features, for a different task, so I may need other sources). As far as I've understood a CNN needs a constant-size input, so my idea is to give it the MFCCs of say 1 s of audio and classify that. You can then classify the entire recording based on the mode of assigned classes for the 1 s frames. (In [12] they instead pad/trim recordings to get the right size.)

[1] Interestingly, they were able to use networks which had been pre-trained on image data to improve their performance on audio data.

5. These are also in scikit-learn. In [5] they do not use the MFCCs directly but rather gather statistics via GMMs which they then pass on to the SVM. This would not be time-dependent per se, but since we are really interested in what noise is present in the entire recording an SVM may still work for noise which is more or less stationary. Another approach would be the 1s frames as above in the CNN case.

In 1 and 2 you would train an HMM for each noise class and select the noise type which has the highest likelihood of having generated the sequence we are interested in, or *clean* if all likelihoods are under a certain threshold (which will be determined empirically). Though the classifiers in 3-5 can be designed to select one of multiple classes, I think it makes more sense to use a set of binary classifiers here too as in the real-world case there may be multiple disturbances at the same time.

- **Train classifiers.** Since we are not using crazy amounts of data it can probably be done on my home computer. The GPU supports CUDA although it is a bit old, and I can leave it running overnight if needed.
- **Measure classifier performance.** One obvious measure is the precision and recall for the different classes of noise. However, it may also be relevant to consider both training time and classification time as well, and maybe ease of use (which would be rather subjective, but things like how sensitive they are to having e.g. just the right hyperparameters).

References

[1] Matthias Mauch, Sebastian Ewert, et al. "The audio degradation toolbox and its application to robustness evaluation". In: *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR 2013)*. 2013.

[2] Dongge Li et al. "Classification of general audio data for content-based retrieval". In: *Pattern Recognition Letters* 22.5 (2001), pp. 533–544. ISSN: 0167-8655.

[3] Ling Ma, Ben Milner, and Dan Smith. "Acoustic Environment Classification". In: *ACM Trans. Speech Lang. Process.* 3.2 (July 2006), pp. 1–22. ISSN: 1550-4875. DOI: 10.1145/1149290.1149292. URL: https://doi.org/10.1145/1149290.1149292.

[4] Panu Maijala et al. "Environmental noise monitoring using source classification in sensors". eng. In: *Applied Acoustics* 129 (2018), pp. 258–267. ISSN: 0003-682X.

[5] Constantine Kotropoulos and Stamatios Samaras. "Mobile phone identification using recorded speech signals". In: *2014 19th International Conference on Digital Signal Processing*. IEEE. 2014, pp. 586–591.

[6] Md. Shamim Hussain and Mohammad Ariful Haque. "SwishNet: A Fast Convolutional Neural Network for Speech, Music and Noise Classification and Segmentation". In: (2018). arXiv: 1812.00149.

[7] Christophe Couvreur et al. "Automatic Classification of Environmental Noise Events by Hidden Markov Models". eng. In: *Applied Acoustics* 54.3 (1998), pp. 187–206. ISSN: 0003-682X.

[8] K El-Maleh, A Samouelian, and P Kabal. "Frame level noise classification in mobile environments". eng. In: *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258)*. Vol. 1. IEEE, 1999, 237–240 vol.1. ISBN: 0780350413.

[9] Caio Cesar Enside de Abreu, Marco Aparecido Queiroz Duarte, and Francisco Villarreal. "An immunological approach based on the negative selection algorithm for real noise classification in speech signals". eng. In: *AEUE - International Journal of Electronics and Communications* 72 (2017), pp. 125–133. ISSN: 1434-8411.

[10] S. Qi et al. "Audio recording device identification based on deep learning". In: *2016 IEEE International Conference on Signal and Image Processing (ICSIP)*. 2016, pp. 426–431.

[11] ITU-T. *Single-ended method for objective speech quality assessment in narrow-band telephony applications*. Tech. rep. Series P: Telephone Transmission Quality, Telephone Installations, Local Line Networks, 2004.

[12] A. R. Avila et al. “Non-intrusive Speech Quality Assessment Using Neural Networks”. In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 631–635.

[13] Dong Liu et al. “Powering Hidden Markov Model by Neural Network based Generative Models”. In: (2019).

(Some of the references are slightly broken and I will of course fix that for the report.)