**DEEP LEARNING PROJECT PROPOSAL**

* **Research objective / context and motivation**

The main idea of the project will be about sound classification but especially sound coming from urban areas. What if we can detect gun shots sounds (or any alarming sound) and differentiate them from common sounds in a split of a second. This application could help the city administration/police to notice dangerous events faster than ever.

* **Methodology**

The general idea is to transform sound data into images through feature engineer technique such as “Mel” spectrogram[[1]](#footnote-1) (i.e. scale base on pitch) or chromagram (i.e. scale base on pitch categories). This manipulation will allow us to apply a Convolutional Neural Networks (CNNs) and use each feature engineer as a channel to end up with a feature map.

* **Dataset**
  + The dataset we will use is an Urban Sound Classification dataset. This dataset contains 8 000 samples with 10 classes. It will allow us to perform our training for our model with a dataset that will contain data related to our application and meet a threshold for accuracy allowing for implementation:  
    <https://urbansounddataset.weebly.com/>
  + A secondary dataset we can use later to test our classifier better is AudioSet which contains 2 084 320 samples with 632 classes which would be to computationally heavy as an initial dataset. We can use the same 10 classes as the urban sound classification dataset to compare models later in the project. <https://research.google.com/audioset/?fbclid=IwAR3If9WF29_QwarlvzjwylQVYxxTKNhCAcpA0vanD_hhOe0e8XVfVcyFMYs>
* **Data preprocessing and manipulation**
  + When it comes to the data preprocessing, there are many different articles on how to transform sound files directly into a spectrogram which will be the initial step for us[[2]](#footnote-2)[[3]](#footnote-3)[[4]](#footnote-4). There are also multiple libraries that exist to convert a *.WAV* file to a spectrogram representation, such as Python\_speech\_features and libROSA libraries, which we plan to use[[5]](#footnote-5).
  + There is a possibility for us to use data augmentation to make sure our dataset is bigger. Not only will make it easier to train with more example, but depending on how we decide to do this it could also account for distortion in the sound received through the means we have, thus making the model more robust.
    - The following paper[[6]](#footnote-6) presents how we can use methods for data augmentation both on the audio file directly and on the spectrograms created after. We plan on using some of those methods to explore the potential improvements to the model’s accuracy.
  + To find specific methods for data augmentation on audio files we can use similar methods found in this paper. We want to use methods like, time stretching, pitch shifting and more importantly background noise from the MUDA library. Moreover, we can also create multiple sets of data so that we can figure out what the optimal data augmentation type or hybrid is[[7]](#footnote-7).
* **Estimation method and models**

There are different ways to tackle this classification models

* + The estimation can be done with the help of pre-trained CNN model such as the VGG-16. This will leverage some of the feature extraction or feature engineering by unsupervised learning and then train a fully connected network through supervised learning.
  + In this paper[[8]](#footnote-8), they propose an architecture that adds attention in the estimation based on CNN. This design is called a Convolutional Block Attention Module (CBAM) which applies attention in two sections: in the channel attention map and the spatial attention map. The idea is to learn what and where to emphasize as well as what to suppress or to refine in order to identify effectively a feature.
  + When it comes to a good architecture for models there is this paper[[9]](#footnote-9) that gives an overview acoustic event detection. The type of data is slightly different, but the architecture could still help us. For their state-of-the-art they use Bag of Audio Words (BoAW) with a Deep Neural Neural (DNN) and Hidden Markov Models (HMM). They did a good amount data augmentation that increase drastically their accuracy which could be useful to consider for our application. Finally, they also proposed a new CNN architecture that could be part of our project.
  + There is also this paper that goes through the most common and love one such as ResNet, AlexNet, VGG, etc [[10]](#footnote-10). It went through all these common CNN architectures and compare them for audio classification. This could particularly useful to learn how to use the AudioSet dataset from google.
  + Last but not least, there is also a possibility to add two type of DNN such as the Recurrent Neural Network (RNN) with CNN to create a CRNN[[11]](#footnote-11). This type of architecture is relevant when we are doing temporal classification of sounds events. This could be presented as a future improvement in order to put the sound classifier in production.
* **Hyperparameters tuning**
  + Hyperparameters tuning will take place at different steps of the development process. We will try to apply state-of-art hyperparameter settings found in the literature for the different model architectures we will use. However, we are aware that some hyperparameters increasing the model complexity or the number of training samples could be limited by our computation power.
  + We will also take into account hyperparameters taking place during the sound preprocessing. We discovered in the literature review that some methods during the data preprocessing such as the short time Fournier transform requires hyperparameters tuning in order to carefully chose the time window for the decomposition of the sound signal in a redundant time frame[[12]](#footnote-12).
* **Performance measure**
  + In a similar way to which one would do sentiment analysis we can look at accuracy in a similar way. Additionally, to compare architecture a simple classification error could be provided by a confusion matrix of 10x10 which would allow us to understand what the problematic classes are.

1. <http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/> [↑](#footnote-ref-1)
2. <https://fairyonice.github.io/implement-the-spectrogram-from-scratch-in-python.html> [↑](#footnote-ref-2)
3. <https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html> [↑](#footnote-ref-3)
4. <http://datagenetics.com/blog/november32012/index.html> [↑](#footnote-ref-4)
5. <https://stackoverflow.com/questions/44787437/how-to-convert-a-wav-file-to-a-spectrogram-in-python3> [↑](#footnote-ref-5)
6. <https://arxiv.org/ftp/arxiv/papers/1912/1912.05472.pdf> [↑](#footnote-ref-6)
7. <https://arxiv.org/pdf/1608.04363.pdf> [↑](#footnote-ref-7)
8. <https://arxiv.org/ftp/arxiv/papers/1901/1901.06032.pdf> [↑](#footnote-ref-8)
9. <https://arxiv.org/pdf/1604.07160.pdf> [↑](#footnote-ref-9)
10. <https://arxiv.org/pdf/1609.09430.pdf> [↑](#footnote-ref-10)
11. <https://tutcris.tut.fi/portal/files/13594874/1702.06286.pdf> [↑](#footnote-ref-11)
12. <https://ccrma.stanford.edu/~jos/sasp/Short_Time_Fourier_Transform.html> [↑](#footnote-ref-12)