# HW1: Audio-based Multimedia Event Detection

11-775 Large-Scale Multimedia Analysis, Spring 2022
Due on: Wednesday February 16, 2022 4:30 PM

In this course you will learn to build a multimedia event detection (MED) pipeline step-by-step through the completion of three assignments. As a first step, in this homework you will learn how to process the audio from videos to build an audio-based MED pipeline to classify videos. In homework 2, you will learn to process the visual elements of the videos. In homework 3, you will learn how to fuse both modalities to achieve the task in hand. Please **START EARLY!** The tools required to build the modules on the pipeline take time to understand, and the feature extraction process is also time-consuming.

## 1 Task

Build an audio-based MED pipeline on AWS that extracts different audio features and evaluates different models to classify the provided dataset.

## 2 MED Pipeline

As we can see in the MED pipeline diagram depicted on Figure 1, the first step is to extract the audio features from the training data. For this homework, we will use the traditional MFCC features and the neural-based self-supervised SoundNet features. Once you have extracted both types of audio representations, they will require some post-processing to pack them into a final representation such that each video can be represented by a single feature vector. For this, we ask you to implement the bag-of-words representation through k-means clustering. Once you have the single-vector representations for each video and the provided labels from the training data, you will train two common classifiers: a Support Vector Machine (SVM) and a Multi-Layer Perceptron (MLP). Once these classification models have been trained, then we can move into the testing phase. For testing, we extract and pack the video representations of the testing data using the same parameters/models that were used in the training phase. Then we score the videos with the trained classifiers and evaluate the MED system.

In short, for this homework you need to:

- Learn to extract MFCC and SoundNet audio features from the provided videos.
- Learn to represent the videos as a bag-of-audio-words through a k-means approach.
- Train an SVM and an MLP as classification models
- Test the classification models
- Write the output from the classification models into an evaluation format

To help you get started, we provide an example framework. Please note that the provided code is just an example to help you understand the basic components of an MED system, and the original parameters may not perform well. You are NOT required to follow the provided pipeline. We highly encourage you to create your own pipeline and try other tools, audio features, single-feature representations of temporal data, classifiers, etc.. Take this as an opportunity to test any ideas that you might have.

In the end, we will require you to provide a zip file with your pipeline code, video feature representations and a README.md file with the instructions explaining how to run your pipeline.

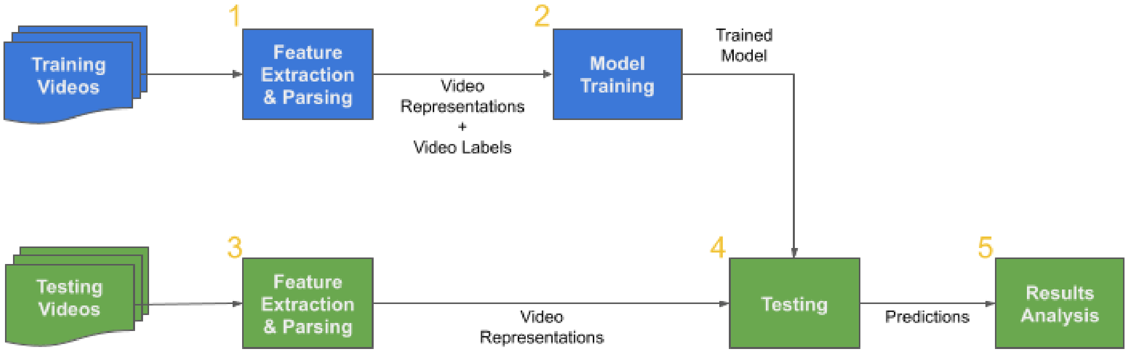

Figure 1: MED Pipeline Overview

## 3 Dataset

For this course, the TAs collected an audio-visual dataset that consists of 8249 videos from 15 different events or classes. In this dataset, each video portrays only one event. The data has been split into 7500 video for training/validation and 749 videos for testing. This dataset will be used on the three homework assignments. To give you an example of the events found in the videos we have: tickling, singing, motorcycling, and hitting baseball. In Figure 2 you can see a sample frame for each event we just mentioned.

The event-name ID mapping is provided in the cls_map.txt as well as the list of all the events found in the dataset. This file can be found within the data.zip file. The content of this mapping file describes the class number assigned to each label as shown in the example below:

```
Label,Category
blowing_out_candles,0
dribbling_basketball,1
flipping_pancake,2
...
```

Also within the dataset you can find the ground-truth labels for the training data in the file train_val.csv, which looks as follows:

```
Id,Category
NjUzNjkxMTk0MzI5NDYwNTA2NA==,0
LTcxNTE1MDY1Mjc5NzMxMTExNDI=,0
NTMyNTQxODcwODc5MjU2NDk4NA==,0
...
```

As we can see, the first three files belong to class 0 which is blowing_out_candles. The field Id refers to the file name of the video and the Category refers to the number ID of the class.

**NOTE:** Remember to shuffle and split the label file into your own training and validation set, e.g. 6,000 videos for training and 1,500 for validation in an 80/20 ratio. Splitting the data is important to fine-tune the hyper-parameters and conduct ablation studies. Remember to report any interesting findings in your report!

Finally, in test_for_student.csv you will find a list of 749 IDs corresponding to the test videos. Note that the class labels have been deliberately removed for evaluation purposes and the content of this file looks as follows:

```
Id,Category
LTExODM2Mzc0ODQyOTc1ODE4NDM=,
LTUwNDU3NzgyNjE2Mzk0OTU1NjQ=,
ODU3OTE0MDU5NzM5NDI2MDQ2,
...
```

You are expected to label every testing video using your MED pipeline into 15 event types using the two audio features described in the next section.

Figure 2: Sample frames from the training dataset. The corresponding classes from left to right are: tickling, singing, motorcycling, and hitting baseball.

## 4 Audio Features

To extract the audio features, first you need to extract the audio from the videos. For this you can use the FFMPEG tool which is the swiss knife of anyone working with multimedia. You can extract a single-channel audio from the mp4 video files into wav audio files with the following command:

```
ffmpeg -y -i LTExODM2Mzc0ODQyOTc1ODE4NDM=.mp4 -ac 1 -f wav \
    LTExODM2Mzc0ODQyOTc1ODE4NDM=.wav
```

In this line, the parameter -y tells ffmpeg to do not ask for any confirmation. The parameter -i indicates the input file (e.g. a video file from the dataset). The parameter -ac sets the number of channels for the output, while the parameter -f tells the output format. At the end of the command we just simply put the name of the output file. It is a good practice to keep the same file name and just change the file extension to avoid any future problems while keeping the pipeline well-organized.

### 4.1 MFCC Features

Mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of sound. MFC is based on a linear cosine transform of a log-power spectrum on a nonlinear mel-scale of frequency. Mel-frequency cepstral coefficients (MFCC) are coefficients that collectively build an MFC.

A well-known tool to extract MFCC is openSMILE. For each video, you can process it as follows:

```
SMILExtract -C config/MFCC12_0_D_A.conf \
            -I LTExODM2Mzc0ODQyOTc1ODE4NDM=.wav \
            -O LTExODM2Mzc0ODQyOTc1ODE4NDM=.mfcc.csv
```

We suggest you look into the .conf to get a better sense of the MFCC features. Once you have finished your first version of the pipeline you can revisit the configuration and tweak it to improve the MED pipeline's performance.

Alternatively, you can use the librosa package if you are building your pipeline using Python through the librosa.feature.melspectrogram function.

#### 4.1.1 Bag-of-Words Representation

A popular approach to represent a video is through bag-of-(audio)-words, as it reduces the dimensionality of the features, learns shared concepts (words), and results in a more compact representation. To achieve this, you need to train a K-Means clustering model to group the audio feature vectors to represent a video with a single vector using the cluster centroids.

K-means clustering can be memory and time-consuming. To speed up the clustering process, you can sub-sample the data and select only a small portion of the feature vectors, e.g. randomly select 20% of the audio features from each video. As with the rest of the homework, it is up to you whether to use this speed up strategy or not.

Please implement functions to train a K-Means model or any other clustering algorithm to represent videos. An easy way to train a K-Means model in Python is by using the sklearn package. As

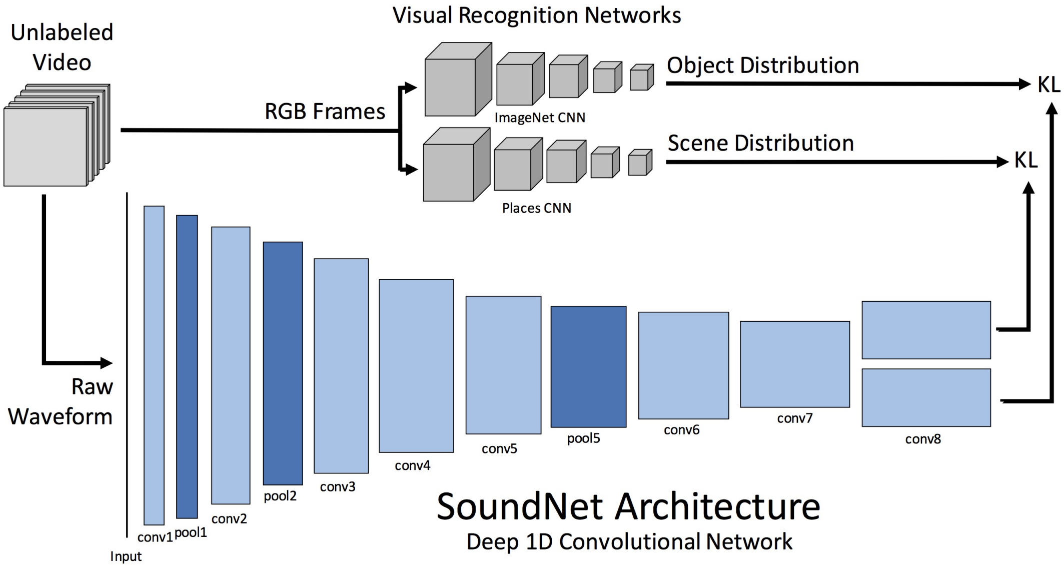

Figure 3: SoundNet Model

an initial approach you could set $k = 200$. Once the K-Means model has been trained, you can represent each video in a $k-$dimensional vector, where each dimension indicates the counts of feature representations (MFCC or SoundNet) to their nearest cluster center. In other words, the Bag-of-Words representation is a histogram of the nearest clusters for each audio feature in the sequence. For this homework, you may also try different distance metrics other than the default $L2$ distance. Try to find out which metric is more suitable for audio features. Please note that exception handling may be required for very short or very long videos.

### 4.2 SoundNet Features

SoundNet learns rich natural sound representations by capitalizing large amounts of unlabeled sound. This CNN based model was developed by the MIT CSAIL Lab. It yields a significant performance improvement on standard benchmarks for acoustic classification. This model is versatile as you can extract feature representations from different layers (e.g. `conv3`, `conv4`, `conv6`, and `conv7`) from a pre-trained model. We provide you an open-source PyTorch implementation along with an extractor script (`extract_feats.py`).

The extracted features are 4-dimensional arrays in (`N, C, S, 1`) format, where `N` is the batch size, `C` is the number of filters or dimensionality of the features, `S` is the sequence length of the feature representation, and the list dimension from the output array is set to 1. Since SoundNet outputs a sequence of feature vectors to represent the audio, at this point you might be asking yourself: how do you generate a single-vector representation for each video using SoundNet features? To answer this question, you can refer to the implementation in the original paper. For this homework, we expect you to try different SoundNet features from different layers and employ different types of feature aggregation strategies and report your results. As a first exercise try training an MLP using the `y_scns` features. What are your results?

An interesting experiment is to determine which convolutional layer brings the best results. Another experiment would consist in determining which features show the best performance for the task e.g. after the convolutional layer, the batch normalization, the ReLU or max-pool layers.

## 5 Model Training and Testing

Now that we have extracted the audio features, now we can train and test the classification models for your MED pipeline. In this homework you will train a Support Vector Machine (SVM) and a Multi-layer Perceptron. You don't have to reinvent the wheel and you can use the models already implemented on scikit-learn and libsvm. The main goal of this homework is for you to learn how to build a pipeline and integrate whatever tools are necessary. Nonetheless, feel free to write functions to fine-tune the hyper-parameters as well as the cost and kernels. In the case of the MLP you can play around with the number of layers and number of hidden units.

Evaluating the models on the validation set provides important information to determine the model's performance. A model that performs well in the validation set is likely to perform well in the testing set, under the assumption that data is i.i.d.. At this point, you need to define an evaluating criteria (e.g. top-1 accuracy) and select the best-performing model in the validation test and use it for testing.

For training and testing the models, you might want to follow these steps:

1. Train the classification models using the extracted MFCC and SoundNet features.
2. Validate the model using your previously defined validation set. Define a criteria, such as top-1 accuracy, and try different training configurations to find the best models
3. Use your best model to predict the events of the testing videos listed in `test_for_student.label`. Store the prediction results in a csv file preserving the order of the videos.
4. **Optionally**, you can submit it to Kaggle https://www.kaggle.com/c/11775-s22-hw1 and estimate how well your models performs on the testing set.
5. For Canvas, name your best submission as `best.csv` and specify your method in the report.

As a summary, we expect you to submit the results for each audio feature and your best model in three files: {`mfcc|soundnet|best`}`.csv`. As a reference, we provide you a baseline implementation under spring2022/hw1/. A successful submission should aim to get comparable results to the baseline submitted by the TAs on Kaggle. Submissions that significantly exceed the baseline will get extra points.

**NOTE:** You are not limited to use only MFCC and SoundNet features. For your *best* submission you can use a fusion of both features or even make use of any other features. You are also encouraged to try different feature aggregation methods and classification models.

## 6 What to Submit

**In Canvas:** Please compress your submission into a zip file named as **`andrewid_hw1.zip`** and submit it through the turn-in link in Canvas. To ease the evaluation of your submission, the zip file should contain the following:

1. **report.pdf**: Your PDF report with your pipeline design, findings, results, and analysis.
2. **code.zip**: A .zip file with your code only. Please be sure to add a `README.md` with the instructions on how to run your code or scripts.
3. **mfcc.csv**, **soundnet.csv**, and **best.csv**: The classification results for each testing video. The submission csv format is as follows:

   ```
   Id,Category
   LTExODM2Mzc0ODQyOTc1ODE4NDM=,7
   LTUwNDU3NzgyNjE2Mzk0OTU1NjQ=,1
   ODU3OTE0MDU5NzM5NDI2MDQ2,0
   ...
   ```

In the report, please describe the detailed steps and parameters in your MED pipeline for extracting both audio features: MFCC and Soundnet, as well as the feature aggregation methods and hyper-parameters of the classification models during training. Then, specify the size of your training and validation sets, as well as the validation metric you used for validation, e.g. top-1 accuracy, top-5 accuracy, average precision, etc. Please report your best model's performance on the validation set. For the report, we also ask you to add the confusion matrix, and make an analysis that describes which classes are harder to detect and with which other classes those are being confused. Then, please report the time your MED pipeline takes by running on CPU for feature extraction, classification, and total time it took during testing.

Also, by the end of the report please tell us the amount of credits left on your AWS account once you have finished the homework. They are all yours, we will not take them back ;D, we would like to get an idea if these are enough for this sort of experiments.

**In Kaggle**: Please submit your **best.csv** to the Kaggle leaderboard https://www.kaggle.com/c/11775-s22-hw1 and briefly explain you pipeline. You can submit up to 5 times/day. The final performance on the whole testing set will be revealed on February 16, 2022.

## 7 AWS

In this course you will use Amazon Web Services (AWS) for building a large-scale multimedia processing pipeline. We will provide you a $50 coupon per homework, such that you do not need to spend your own money for the homework. However, keep in mind that AWS charges you as soon as you start the instance until it is stopped, it doesn't matter if it is in an idle state or not. Please remember to **STOP** your instance when you are not using it. **Do not terminate it**, as it will erase all your progress.

We recommend that you start your development on a CPU instance to get familiar with the process and be confident that your pipeline works. Once you feel comfortable you can dive into neural models that may require a GPU instance. Hence, you can start with a t2.large CPU instance with a storage around 100 GB as its cost is around $0.09/hr. If you would like to use a GPU instance we recommend you to a g4dn.xlarge which costs around $0.90/hr.

If you have never used AWS before, we recommend you to follow this guide to setup your EC2 instance and your file system.