Homework 1

CMU 11-775

Piazza

Code

GCP DOC

OUT: Wed, Sep 13th, 2023

DUE: Friday, Oct 7th, 2023, 11:59pm EST

Written by: Yijun Qian

START HERE: Instructions

- **Collaboration policy:** Collaboration on solving the homework is allowed after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., "Jane explained to me what is asked in Question 2.1"). Second, you should write your codes on your own. *independently*: close the webpage, and send collaborators out of the room, so the code comes from you only. See the Academic Integrity Section of the course syllabus in the Resources section of Piazza.
- **Late Submission Policy:** See the Late Submission Section of the course syllabus in the Resources section of Piazza.
- **Submitting your work:**
  - **Canvas:** Please compress your submission into a zip file named as **andrewid_hw1.zip** and submit it through the turin-in link in Canvas. The contents of your zip file should be organized as the following:
    * 1. report.pdf: Your PDF report with your answers to the questions in this handout.
    * 2. code.zip: A zip file with your code only. Remember to remove all the audio folders and feature folders you generated during the experiments. Please be sure to add a README.md with the instructions on how to run your code to reproduce the results you reported and gained on the leaderboard.
    * 3. mfcc-50.mlp.csv,mfcc-50.svm.csv: The classification results for Task 2.
    * 4. snf.mlp.csv: The classification results for Task 3.
    * 5. task4.csv: The classification results for Task 4.

– **Kaggle:** You can submit results to Kaggle to validate if your implementations are correct. You can submit up to 10 times/day. The final performance on the private test set will be revealed one day after DDL. For the output of Task 2, you are expected to gain over 30% Acc. For Task 3, you are expected to gain over 50% Acc. And for Task 4, you are expected to gain over 65% Acc.

1 Introduction:

The goal of this assignment is to build an audio-based MED pipeline that extracts different audio features with SVM and MLP classifiers. This homework contains 4 tasks.

- In Task 1, you will just follow the instructions to extract bag-of-(audio)-word features.
- In Task 2, you will build SVM and MLP classifiers then train and test them.
- In Task 3, you will extract SoundNet[1] features then use the new features to train and test your MLP classifier.
- In Task 4, you will explore methods to improve your model's performance on your own.

If you are unfamiliar with pytorch or GCP, please **Start Early**. It may take longer time than you expected. Template code with instructions is provided here. You are NOT required to follow the template, just make sure you can pass requirements for each task and **your results can be reproduced given your submitted code**. You may also need to use GCP servers for this assignment. If it's your first time to use GCP, please follow this doc. Here are several docs which may help you to get your GCP node ready.

- connect to GCP with keys
- install python
- install conda
- install nvidia driver(Optional)

2 Task 1: Extract MFCC features and represent the videos as a bag-of-(audio)-words through a k-means approach. (20%)

In this task, you will learn how to extract the classic bag-of-words features (bof). Mel-frequency cepstrum (MFC) represents the short-term power spectrum of sound. MFC is based on a linear cosine transform of a log-power spectrum on a nonlinear melscale of frequency. Mel-frequency cepstral coefficients (MFCC) are coefficients that collectively build an MFC. We suggest you look into the **.conf** to understand the MFCC features better. 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 to build your pipeline using Python through the **librosa.feature.melspectrogram**

function. 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 then follow the instructions 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 using the sklearn package. 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 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.

3 Task 2: SVM & MLP Classifier (30%)

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 (MLP). You don't have to reinvent the wheel, and you can use the models already implemented on sklearn 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. 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.. In this assignment, we didn't provide you an isolated validation set, it's left for you to design how to split the dataset and improve the performance of your model. At this point, we suggest you follow our official evaluating criterion (top-1 accuracy) and select the best-performing model in the validation test and use it for testing.

Questions:

- With the default settings, which classifier performs better? Why?
- Which default parameter do you want to tune? Why? How do you change it and what's the difference?

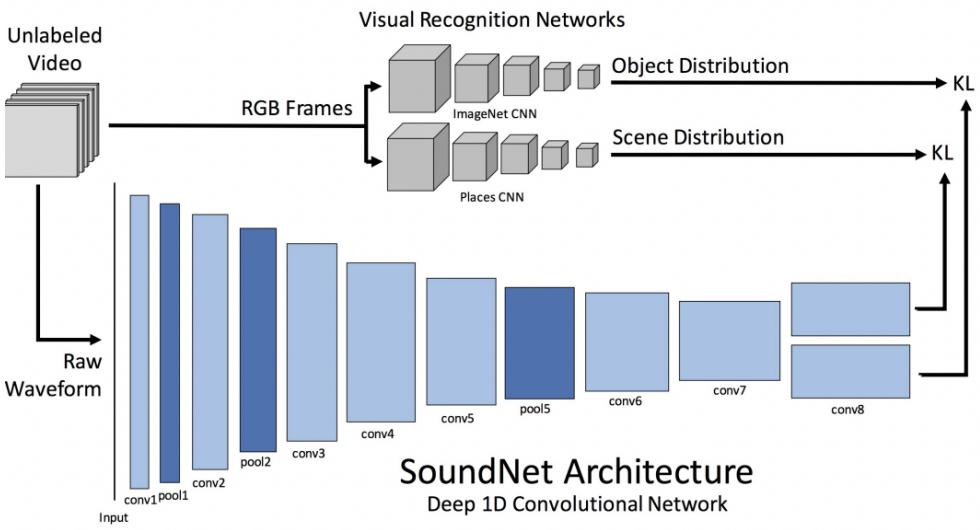

Figure 1: Architecture of SoundNet.

4 Task 3: More powerful SoundNet features (30%)

SoundNet[1] 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. 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 dimension 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[1] 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.

Questions:

- How do you generate a single-vector representation for each video using SoundNet features?
- You extract features from which layer? Why you choose this layer? What's the performance on the public test set?

5 Task 4: Improve your model (20%)

Now it comes to the most interesting part of this assignment. You need to improve the model on your own. Here are several hints but you are strongly encouraged to come up with a method which is not listed here.

- Split trainval.csv into train.csv and val.csv to validate your model variants. This is

important since the leaderboard limits the number of times you can submit, which means you cannot test all your experiments on the official test set.

- Try different number of K-Means clusters
- Try fuse features extracted from different layers of SoundNet
- Try finetune the feature extraction backbone instead of freezing all layers
- Try different classifiers (different SVM kernels, different MLP hidden sizes, etc.). Please refer to sklearn documentation.
- Try different fusion or model aggregation methods. For example, you can simply average two models' predictions (late fusion).

You are expected to achieve 65%+ performance on the final private test set. To achieve this goal, you may need to combine multiple methods, but it won't be that hard. **Top 10 results on Kaggle and surpassed the baseline submitted by the TA will gain extra 15% bonus**.

Questions:

- Introduce how do you improve the model. What's the performance? Why it works? If you have extra time and GPU resource, what's your next steps.
- What's the bottleneck of the current system?

References

[1] Aytar, Y., Vondrick, C., Torralba, A.: Soundnet: Learning sound representations from unlabeled video. Advances in neural information processing systems **29** (2016)