Homework 1

An Introduction to Neural Networks

11-785: Introduction to Deep Learning (Fall 2021)

OUT: **September 9, 2021, 12:00 AM**
Early Deadline/HW1P2 MCQ Deadline: **September 16, 2021, 11:59 PM**
DUE: **September 30, 2021, 11:59 PM**

Start Here

- **Collaboration policy:**
  - You are expected to comply with the University Policy on Academic Integrity and Plagiarism.
  - You are allowed to talk with / work with other students on homework assignments
  - You can share ideas but not code, you must submit your own code. All submitted code will be compared against all code submitted this semester and in previous semesters using MOSS.
- **Overview:**
  - **Part 2**: This section of the homework is an open ended competition hosted on Kaggle.com, a popular service for hosting predictive modeling and data analytics competitions. The competition page can be found here.
- **Quiz**
  - **Part 2**: You need to take a quiz before you start with HW1-Part 2. This quiz can be found on Canvas under **HW1P2: MCQ (Early deadline)**. It is **mandatory** to complete this quiz before the early deadline for HW1-Part 2.
- **Submission:**
  - **Part 2**: See this for details.

Part 2: Frame Level Classification of Speech

This part of the homework is a live competition on kaggle.

In this challenge you will take your knowledge of feedforward neural networks and apply it to a more useful task than recognizing handwritten digits: speech recognition. You are provided a dataset of audio recordings (utterances) and their phoneme state (subphoneme) labels. The data comes from articles published in the Wall Street Journal (WSJ) that are read aloud and labelled using the original text. If you have not encountered speech data before or have not heard of phonemes or spectrograms, we will clarify the problem further.

Data

- **train.npy**: (14542,)
- **train_labels.npy**: (14542,)
- **dev.npy**: (2683,)
- **dev_labels.npy**: (2683,)
- **test.npy**: (2600,)

1 Task

Your task is to generate predictions for the phonemes of the test set. You will be evaluated on the accuracy of the prediction of the phoneme state labels for each frame in the test set. Grade cut-offs are released after the early deadline. For detailed information, please look at the kaggle page.

If you have not encountered speech data before or have not heard of phonemes or spectrograms, we will clarify the problem further.

The training data comprises of:

- Speech recordings (raw mel spectrogram frames)
- Frame-level phoneme state labels

The test data comprises of:

- Speech recordings (raw mel spectrogram frames)
- Phoneme state labels are not given

Your job is to identify the phoneme state label for each frame in the test data set. It is important to note that utterances are of variable length.

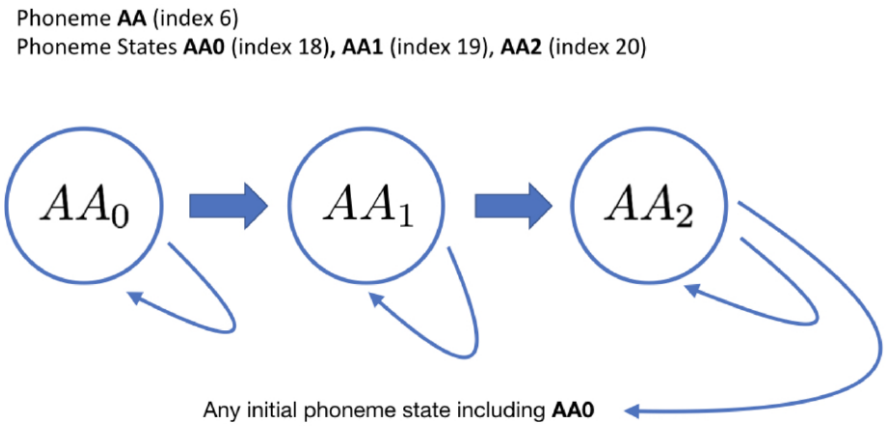
2 Phonemes and Phoneme States

As letters are the atomic elements of written language, phonemes are the atomic elements of speech. It is crucial for us to have a means to distinguish different sounds in speech that may or may not represent the same letter or combinations of letters in the written alphabet.

For this challenge, we will consider a total of 71 phonemes in this language.

A powerful technique in speech recognition is to model speech as a markov process with unobserved states. This model considers observed speech to be dependent on unobserved state transitions. We refer to these unobserved states as phoneme states or subphonemes. For each phoneme, there are 3 respective phoneme states. The transition graph of the phoneme states for a given phoneme is as follows:

Example: ["+BREATH+", "+COUGH+", "+NOISE+", "+SMACK+", "+UH+", "+UM+", "AA", "AE", "AH", "AO", "AW", "AY", "B", "CH", "D", "DH", "EH", "ER", "EY", "F", "G", "HH", "IH", "IY", "JH", "K", "L", "M", "N", "NG", "OW", "OY", "P", "R", "S", "SH", "SIL", "T", "TH", "UH", "UW", "V", "W", "Y", "Z", "ZH"]



Hidden Markov Models (HMMs) estimate the parameters of this unobserved markov process (transition and emission probabilities) that maximize the likelihood of the observed speech data. Your task is to instead take a model-free approach and classify mel spectrogram frames using a neural network that takes a frame (plus optional context) and outputs class probabilities for all 71 phoneme states. Performance on the task will be measured by classification accuracy on a held out set of labelled mel spectrogram frames. Training/dev labels are provided as integers [0-70].

3 Speech Representation

Raw speech signal (also known as the speech waveform) is stored simply as a sequence of numbers that represent the amplitude of the sound wave at each time step. This signal is typically composed of sound waves of several different frequencies overlaid on top of one another. For human speech, these frequencies represent the frequencies at which the vocal tract vibrates when we speak and produce sound. Since this signal is not very useful for speech recognition if used directly as a waveform, we convert it into a more useful representation called a **"melspectrogram"** in the feature extraction stage.

4 Feature Extraction

The variation with time of the frequencies present in a particular speech sample are very useful in determining the phoneme being spoken. In order to separate out all the individual frequencies present in the signal, we perform a variant of the Fourier Transform, called the **Short-Time Fourier Transform (STFT)** on small, overlapping segments (called frames, each of 25ms) of the waveform. A single vector is produced as the result of this transform. Since we use a stride of 10ms between each frame, we end up with 100 vectors per second of speech. Finally, we convert each vector into a 40-dimensional vector (refer the links in the optional readings section for exact details of how this is done). For an utterance T seconds long, this leaves us with a matrix of shape (100*T, 40) known as the **melspectrogram**. Note that in the dataset provided to you, we have already done all of this pre-processing and provided the final **(*, 40) shaped melspectrograms** to you. An illustration of this process is represented in the figure below.

The data provided in this assignment consists of these melspectrograms, and phoneme labels for each 40-dimensional vector in the melspectrogram. The task in this assignment is to predict the label of a particular 40-dimensional vector in an utterance.

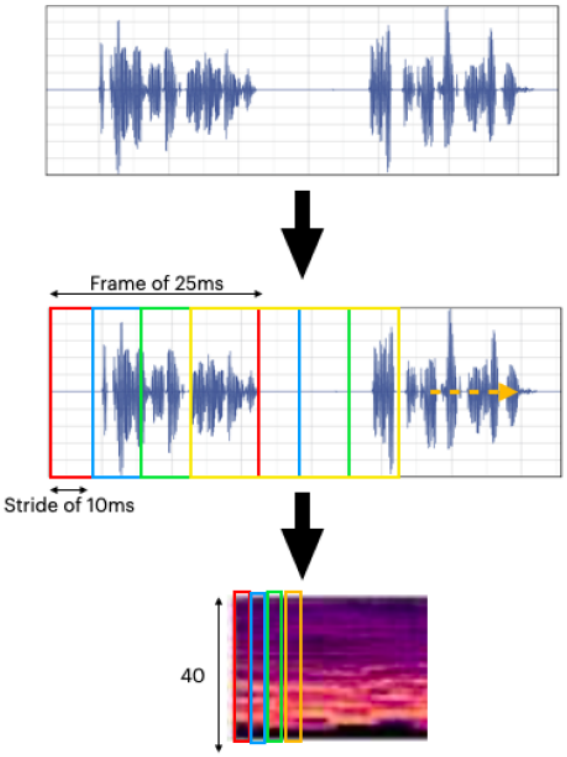


Figure 1: Feature extraction

4.1 Data Files

You can find the training and test data under the 'Data' tab on Kaggle.

A diagrammatic representation of the data is as follows:

Note: Since each vector represents only 25ms of speech it may not be sufficient to feed only a single vector into the network at a time. Instead, it may be useful to provide the network with some **"context"** of size K around each vector in terms of additional vectors from the speech input. Concretely, a context of size 5 would mean that we provide an input of size (11, 40) to the network - the size 11 may be explained as : the vector to predict the label for, 5 vectors preceding this vector, and 5 vectors following it. It is worth thinking about how you would handle providing context before one of the first K frames of an utterance or after one of the last K frames.

Hint: There are several ways to implement this, but you could try :

1. Concatenating all utterances and padding with K 0-valued vectors before and after the resulting matrix

   OR

2. Pad each utterance with K 0-valued vectors, at the extra cost of bookkeeping the beginning index of each utterance's first vector.

Note: When loading the data, you might want to add some context to every frame for better results. The way we add context is to add some form of zero padding to add some of the previous and next frames to the current frame. For example, if we consider a single frame of dimension (1000, 40) and we consider the context to be 5, then we want to implement zero padding before and after this frame such that the dimension

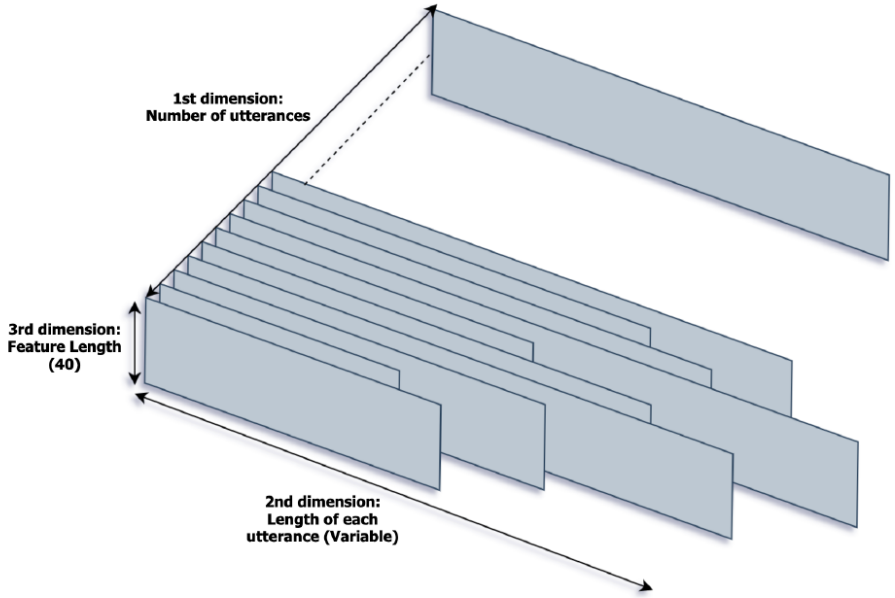


Figure 2: Data representation

becomes (1010, 40). Subsequently, the input layer will have (1 + 2* context_size)*40 nodes. Try to think about why this calculation would make sense with respect to the Dataloader and the speech data given.

Context is a hyperparameter and the recommended value of context to be set for this homework is between 5-30.

5 Evaluation Metric

The evaluation metric for this competition is frame-level accuracy. There are a total of 1910011 frames in the test set. You will be ranked by unweighted accuracy on those phoneme state labels.

6 Submission Format

Submission files should contain two columns: Id and Label.

Id: The 0-based index of the frame in the test set [0-1910011] (data type: int).

Label: The predicted label of the phoneme [0-70] (data type: int).

Id=0 is the first frame in the first utterance.

Id=1910011 is the last frame in the last utterance.

A sample submission file is available on the Data page on Kaggle. Please submit your prediction/submission files here. You will be allowed a maximum of 10 submissions every day.

You need to make atleast one submission (of prediction file csv) to Kaggle before the early deadline.

7 Toy Problem

Along with the main homework dataset, we also provide a toy dataset to help you prototype the framework, debug the algorithm and get familiar with how to download data and submit the results to kaggle. This is not a mandatory submission but we highly recommend you to start the homework on the toy dataset since it is much smaller than the main dataset and each data operation would be faster to save your life.

7.1 Data Description

The format and data type of the toy dataset are the same as the main dataset, and you can download it from this link on Kaggle.

Training: **train.py**: (1000,)
Validation: **dev.py**: (200,)
Testing: **test.py**:

7.2 Usage

- Download data from Kaggle: the data downloading is the same as the one for the main dataset. Try to implement the interface with Kaggle, data downloading and decompressing.
- Implement the dataset loader and network model. You can implement a simple one layer network for toy problem to assist testing other parts of codes and enlarge your network once training on the main dataset.
- Set up your training workflow and make sure it is runnable on toy dataset.
- Since the toy dataset is small and not representative, the trained model on toy dataset cannot accurately indicate the performance of your network.
- Write the results from the test data and submit it to Kaggle.

Notice that the toy problem is used for making sure your codes runnable, checking the type and shape of the data, learning to interface with the Kaggle. The accuracy of the model on the toy problem cannot indicate the performance of your network on the main dataset.

8 Optional Readings

Go through this link to understand how Mel-Spectrograms are generated