Homework 1

An Introduction to Neural Networks

11-785: Introduction to Deep Learning (Spring 2022)

OUT: **January 23, 2022**
Early Deadline/HW1P2 MCQ Deadline: **January 31, 2022, 11:59 PM**
DUE: **February 17, 2022, 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 and 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.
  - You are allowed to help your friends debug
  - You are allowed to look at your friends code
  - You are allowed to copy math equations from any source that are not in code form
  - You are not allowed to type code for your friend
  - You are not allowed to look at your friends code while typing your solution
  - You are not allowed to copy and paste solutions off the internet
  - You are not allowed to import pre-built or pre-trained models
  - Meeting regularly with your study group to work together is highly encouraged. You may discuss ideas and help debug each other's code. You can even see from each other's solution what is effective, and what is ineffective. You can even ”divide and conquer” to explore different strategies together before piecing together the most effective strategies. However, the actual code used to obtain the final submission must be entirely your own.
- **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.
  - **Part 2 Multiple Choice Questions**: 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 the the competition page for details.

Homework objective

After this homework, you would ideally have learned:

- To solve a medium-scaled classification problem using an MLP
  - How to set up the MLP
  - How to handle the data
  - How to train the model
  - How to optimize the model
- To explore architectures and hyperparameters for the optimal solution
  - To identify and tabulate all the various design/architecture choices, parameters and hyperparameters that affect your solution
  - To devise strategies to search through this space of options to find the best solution.
- The process of staging the exploration
  - To initially set up a simple solution that is easily implemented and optimized
  - To stage your data (e.g. by initially working on a subsample of the training data) to efficiently search through the space of solutions.
  - To track losses and performance on validation data to ensure the code is working properly and the model is being trained properly
  - To subset promising configurations/settings and then evaluate those on the larger (complete) dataset
- To engineer the solution using your tools
  - To use objects from the PyTorch framework to build an MLP.
  - To deal with issues of data loading, memory usage, arithmetic precision etc. to maximize the time efficiency of your training and inference.

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**: (28539,)
- **train_labels.npy**: (28539,)
- **dev_mfcc.npy**: (2703,)
- **dev_labels.npy**: (2703,)
- **test.npy**: (2620,)

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.

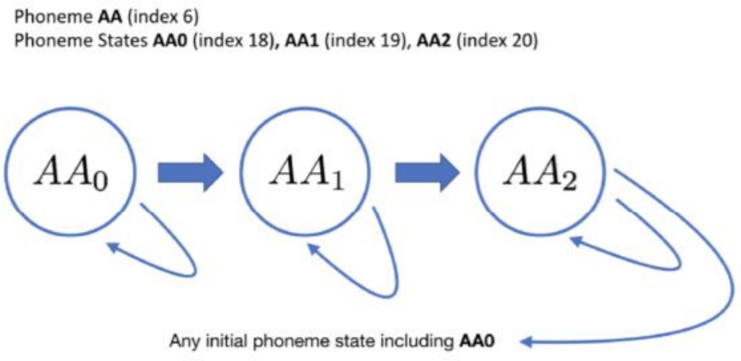
2 Phonemes and Phoneme States

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 40 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”]


Phoneme AA (index 6)
Phoneme States AA0 (index 18), AA1 (index 19), AA2 (index 20)
AA_0
AA_1
AA_2
Any initial phoneme state including AA0

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 40 phoneme states. Performance on the task will be measured by classification accuracy on a held out set of labeled mel spectrogram frames. Training/dev labels are provided as integers [0-39].

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 13-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, 13) 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 **(*, 13) 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 13-dimensional vector in the melspectrogram. The task in this assignment is to predict the label of a particular 13-dimensional vector in an utterance.

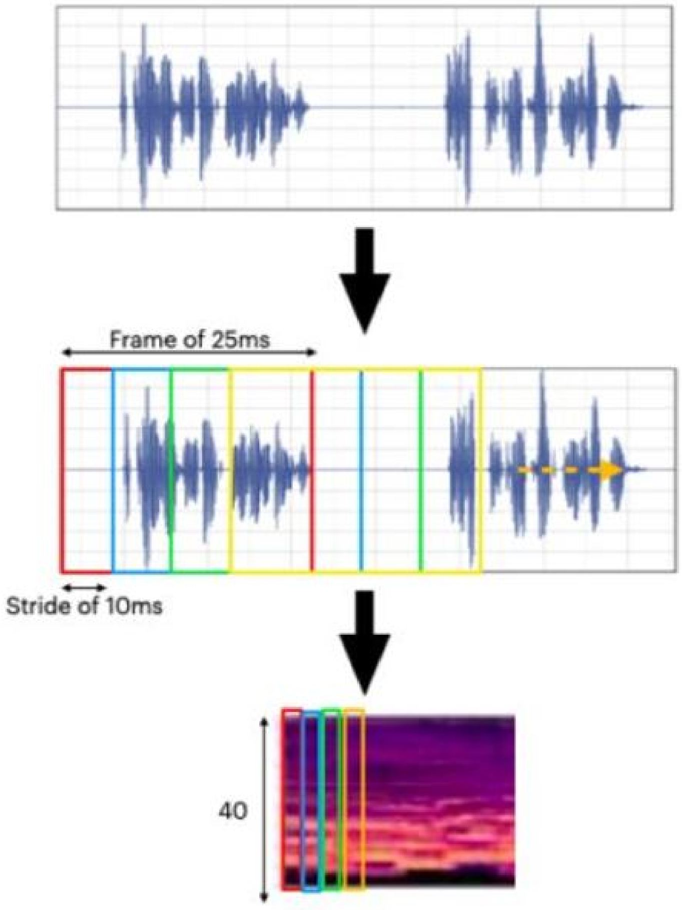


Figure 1: Feature extraction

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, 13) 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, 13) and we consider the context to be 5, then we want to implement zero padding before and after this frame such that the dimension

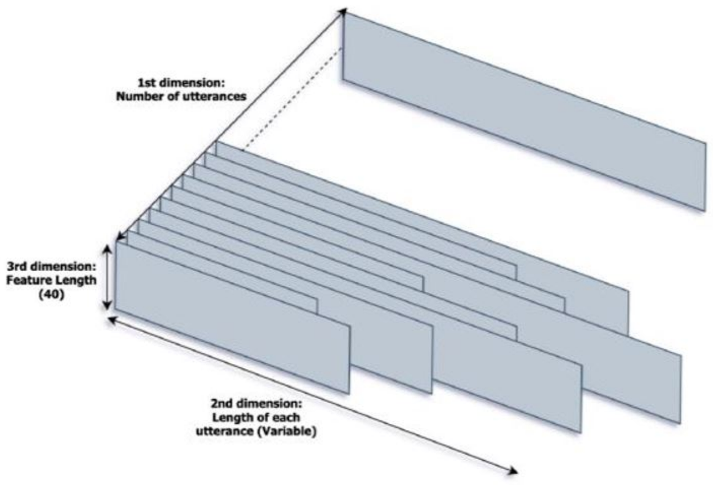

1st dimension: Number of utterances
3rd dimension: Feature Length (40)
2nd dimension: Length of each utterance (Variable)

Figure 2: Data representation

becomes (1010, 13). Subsequently, the input layer will have (1 + 2* context size)*13 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 0-50.

5 Evaluation

The evaluation metric for this competition is frame-level accuracy. There are a total of 1943253 frames in the test set. You will be ranked by unweighted accuracy on those phoneme state labels.

This homework is worth 100 points. The distribution of the points is as follows:

3 points: Early MCQs (due January 31 2022)

7 points: Preliminary Submission (due January 31 2022)

90 points: High Cutoff

70 points: Medium Cutoff

50 points: Low Cutoff

30 points: Very Low Cutoff

6 Submission Format

Submission files should contain two columns: Id and Label.

Id: The 0-based index of the frame in the test set [0-1943252] (data type: int).

Label: The predicted label of the phoneme [0-39] (data type: int).

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

Id=1943252 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 a basic model) 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 and Description

We have provided four comma-separated values (.csv) files which have a column of the names of the sampled numpy files(xxxx.npy).Using the CSV files, you should be able to sample the training set which would be approximately 10-25% the size of the original dataset, depending on the csv file.

The code to do the sampling is already provided in the starter notebook.

You should be able to achieve approximately the following results on running the starter notebook on the **'train_filenames_subset_8192_v2.csv'**.

Epoch 1:

Training Accuracy: Approx. 45.8%

Validation Accuracy: Approx. 44.51%

Training Loss: Approx. 1.89

Epoch 2:

Training Accuracy: Approx. 46.54%

Validation Accuracy: Approx. 45.23%

Training Loss: Approx. 1.82

Epoch 3:

Training Accuracy: Approx. 46.85%

Validation Accuracy: Approx. 45.51%

Training Loss: Approx. 1.80

Your loss curve on the training dataset after the first epoch should look like the figure below. Submitting the results of this model on the test set will be enough for you to satisfy the early submission cutoff, which is worth 7 points.Your model should not take more than 3 minutes to run 1 epoch on this dataset.

Though not mandatory, we recommend starting with 'train_filenames_subset_0008_v2.csv' to debug your code, and then move on to higher sized datasets. This will ultimately prevent you from wasting your valuable time on unseen mistakes.

Training: **train_filenames_subset_8192_v2.csv**: (8192,)

Training: **train_filenames_subset_2048_v2.csv**: (2048,)

Training: **train_filenames_subset_0512_v2.csv**: (512,)

Training: **train_filenames_subset_0008_v2.csv**: (8,)

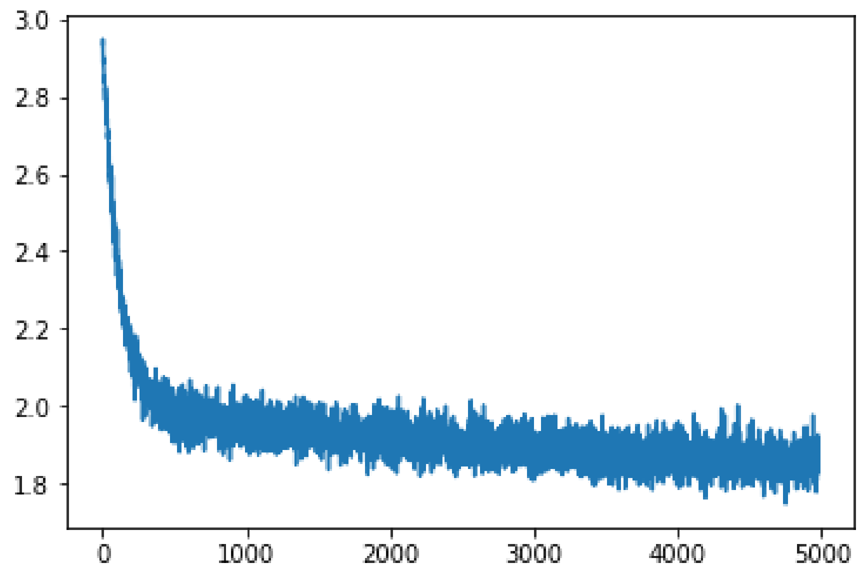


Figure 3: Loss curve per iteration

7.2 Mandatory Preliminary Submission

There is a mandatory preliminary submission and an associated MCQ that, together, are worth 10% of the points for the homework. The deadline for this preliminary submission is posted on the course website and piazza. This submission is intended to get you started quickly on the homework. We provide starter code in the form of a notebook that should, hopefully, make it relatively straightforward to make this submission.

For the preliminary submission, you must download the data, train a very simple preliminary model with it using the sampled (10%) dataset, using the (code in the) notebook provided. You must then process both the validation and evaluation data with the obtainded model, and submit the evaluation results to Kaggle. You must also answer the preliminary-submission MCQ on canvas, which queries you about the homework, the values of specific data instances, and the loss obtained while training the preliminary model. The MCQ is worth 3 points. The actual preliminary submission is worth 7 points. You must get an accuracy of at least 42% on the preliminary submission to get the 7 points. If you do not make the cutoff you get 0 points for the submission.

You will find attached with this writeup, a Python notebook to get you started with the homework. It is not mandatory to use this notebook, and you are free to write your own code. With the starter code and the sampled dataset, you will be able to reach the early submission cutoff. **Note:** You need to submit your results that crosses the preliminary cutoff before the early deadline. Not doing so will cost you 7 points. The preliminary cutoff is set much lower than the final cutoff, and running the starter notebook should be enough to reach this cutoff.

7.3 Usage

- Download data from Kaggle: the data in the toy problem is the same as the one for the main dataset, although a very small portion of the main dataset. Try to implement the interface with Kaggle, data downloading and decompressing.
- Implement the data 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 running 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.

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 Hyperparameter Tuning

Below are a few variations in hyperparameters that might help you reach the low cutoff.

Hyperparameters	**Values**
Number of Layers	2-8
Activations	ReLU, LeakyReLU, softplus, tanh, sigmoid
Batch Size	64, 128, 256, 512, 1024, 2048
Architecture	Cylinder, Pyramid, Inverse-Pyramid, Diamond
Dropout	0-0.5, Dropout in alternate layers
LR Scheduler	Fixed, StepLR, ReduceLROnPlateau, Exponential, CosineAnnealing
Weight Initialization	Gaussian, Xavier, Kaiming(Normal and Uniform), Random, Uniform
Context	0-50
Batch-Norm	Before or After Activation, Every layer or Alternate Layer or No Layer
Optimizer	Vanilla SGD, Nesterov's momentum, RMSProp, Adam
Regularization	Weight Decay
LR	0.001, you can experiment with this
Context	0-50
Normalization	You can try Cepstral Normalization

Table 1: Hyperparameter Tuning

Alongwith these, R-Drop: Regularized Dropout might also lead to better performance.

9 Optional Reading

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