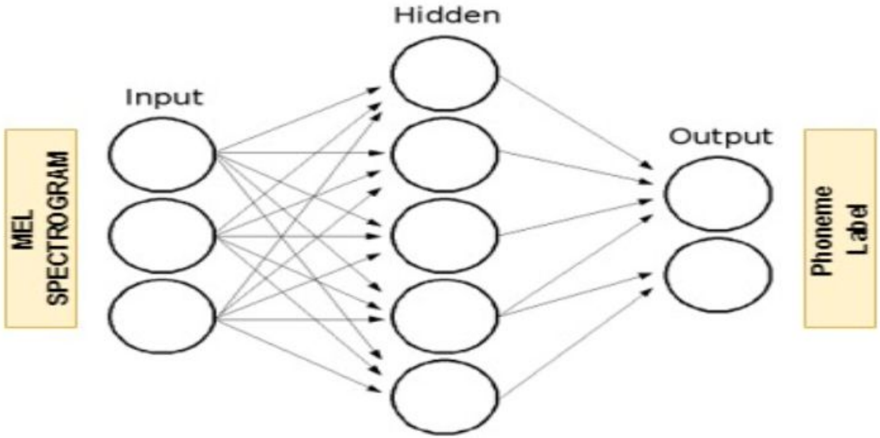
-Bootcamp-
How to get Started with HW1P2

Sarthak Bisht, Yooni Choi

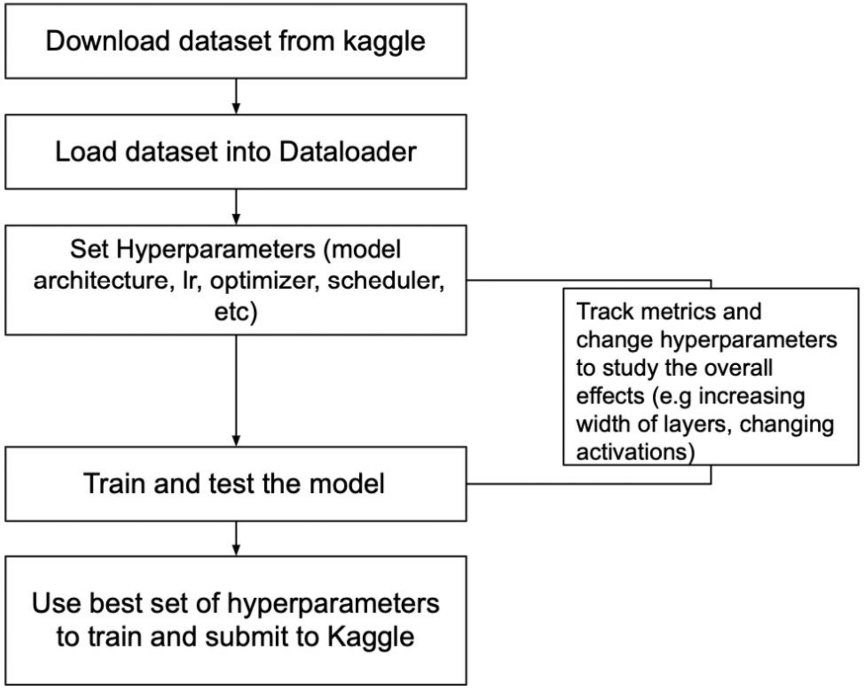
Carnegie Mellon University

Overview



Dataset of Audio Recordings → Predict Phoneme labels

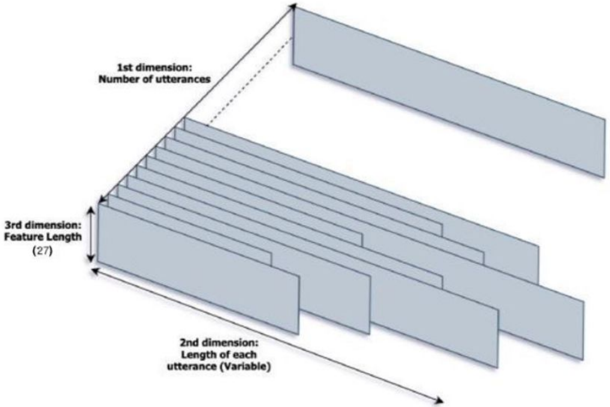
Workflow



Carnegie Mellon University

Data

Raw Speech Signal (Speech waveform) → Short Time Fourier Transform → Melspectogram



Data

```
PHONEMES = [
            '[SIL]',    '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',     'T',      'TH',     'UH',     'UW',
            'V',      'W',      'Y',      'Z',      'ZH',     '[SOS]',  '[EOS]']
```

Dataset

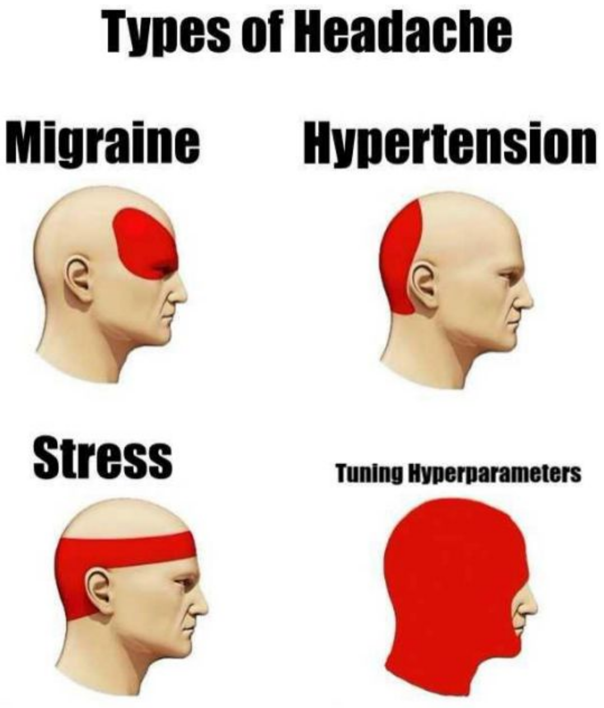
```python
class AudioDataset(torch.utils.data.Dataset):

    def __init__(self, root, phonemes = PHONEMES, context=0, partition= "train-clean-100"): # Feel free to add more arguments

        self.context    = context
        self.phonemes   = phonemes

        # TODO: MFCC directory - use partition to acces train/dev directories from kaggle data using root
        self.mfcc_dir       = NotImplemented
        # TODO: Transcripts directory - use partition to acces train/dev directories from kaggle data using root
        self.transcript_dir = NotImplemented

        # TODO: List files in sefl.mfcc_dir using os.listdir in sorted order
        mfcc_names          = NotImplemented
        # TODO: List files in self.transcript_dir using os.listdir in sorted order
        transcript_names    = NotImplemented

        # Making sure that we have the same no. of mfcc and transcripts
        assert len(mfcc_names) == len(transcript_names)

        self.mfccs, self.transcripts = [], []

        # TODO: Iterate through mfccs and transcripts
        for i in range(len(mfcc_names)):
        #   Load a single mfcc
            mfcc        = NotImplemented
        #   Do Cepstral Normalization of mfcc (explained in writeup)
        #   Load the corresponding transcript
            transcript  = NotImplemented # Remove [SOS] and [EOS] from the transcript
            # (Is there an efficient way to do this without traversing through the transcript?)
            # Note that SOS will always be in the starting and EOS at end, as the name suggests.
        #   Append each mfcc to self.mfcc, transcript to self.transcript
            self.mfccs.append(mfcc)
            self.transcripts.append(transcript)
```

Dataset

```
# NOTE:
# Each mfcc is of shape T1 x 27, T2 x 27, ...
# Each transcript is of shape (T1+2) x 27, (T2+2) x 27 before removing [SOS] and [EOS]

# TODO: Concatenate all mfccs in self.mfccs such that
# the final shape is T x 27 (Where T = T1 + T2 + ...)
self.mfccs          = NotImplemented

# TODO: Concatenate all transcripts in self.transcripts such that
# the final shape is (T,) meaning, each time step has one phoneme output
self.transcripts    = NotImplemented
# Hint: Use numpy to concatenate

# Length of the dataset is now the length of concatenated mfccs/transcripts
self.length = len(self.mfccs)

# Take some time to think about what we have done.
# self.mfcc is an array of the format (Frames x Features).
# Our goal is to recognize phonemes of each frame
# From hw0, you will be knowing what context is.
# We can introduce context by padding zeros on top and bottom of self.mfcc
self.mfccs = NotImplemented # TODO

# The available phonemes in the transcript are of string data type
# But the neural network cannot predict strings as such.
# Hence, we map these phonemes to integers

# TODO: Map the phonemes to their corresponding list indexes in self.phonemes
self.transcripts = NotImplemented
# Now, if an element in self.transcript is 0, it means that it is 'SIL' (as per the above example)
```

Carnegie Mellon University

Running Ablations

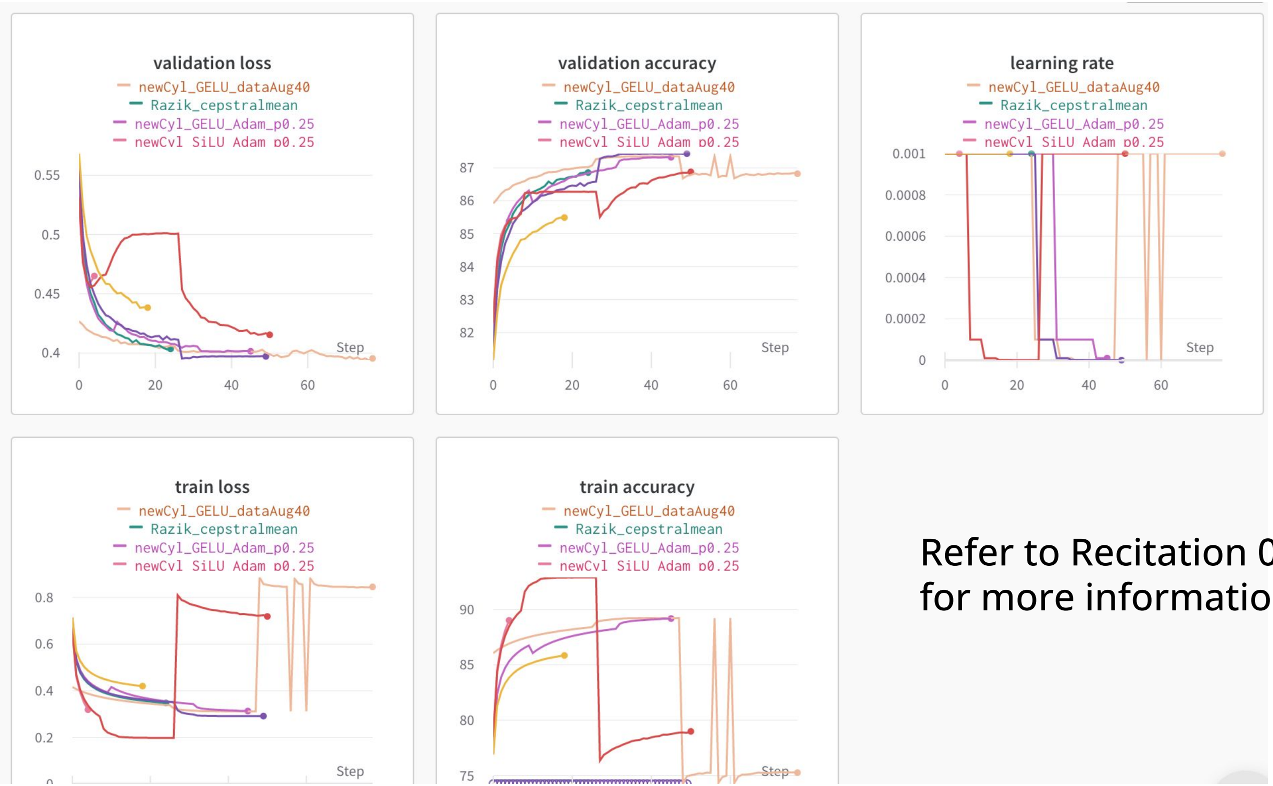


Running Ablations

Epoch	5	5	5	5	5	5
ctx	0	4	8	16	8	16
layers	2	2	2	2	4	4
activations	relu	relu	relu	relu	splus	splus
architecture	Pyramid (max(1024, 10*D) --> 128)	Pyramid (max(1024, 10*D) --> 128)	Pyramid (max(1024, 10*D) --> 128)	Pyramid (max(1024, 10*D) --> 128)	inverted pyramid (D -->2048)	inverted pyramid (max(2D, 128) --> 4D) D--> -->2048
batchsize	256	256	256	256	512	512
dropout	none	none	none	none	0.25 every layer	0.25 every layer
BN	none	none	none	none	every layer preactivation	every layer preactivation
optimizer	ADAM	ADAM	ADAM	ADAM	ADAM	ADAM
scheduler	steplr	steplr	steplr	steplr	reduce on plateau	reduce on plateau
weight init	gaussian	gaussian	gaussian	gaussian	xavier	xavier
Regularization	none	none	none	none	none	none
Initial LR	0.001	0.001	0.001	0.001	0.001	0.001

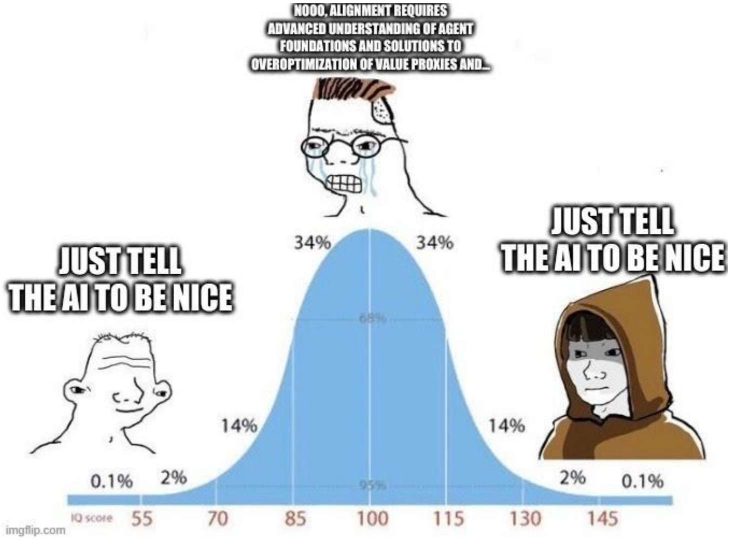
Carnegie Mellon University

Running Ablations



Refer to Recitation 0H
for more information!

Running Ablations



- Progressively build on your experiments
- Incorporate some domain knowledge
- Start with several simple architectures

High Cutoff Architecture

https://www.youtube.com/watch?v=dQw4w9WgXcQ
wink