1

```
import numpy as np
import matplotlib.pyplot as plt
import soundfile as sf
from IPython.display import Audio
import librosa
import librosa.display

y, sr = sf.read("/content/speech.wav")
Audio(y, rate=sr)
num_samples = len(y)
duration = num_samples / sr
librosa.resample(y.astype(float), orig_sr=sr, target_sr=16000)
```

```
array([0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
       0.00000000e+00, 7.27595761e-12, 3.63797881e-12])
```

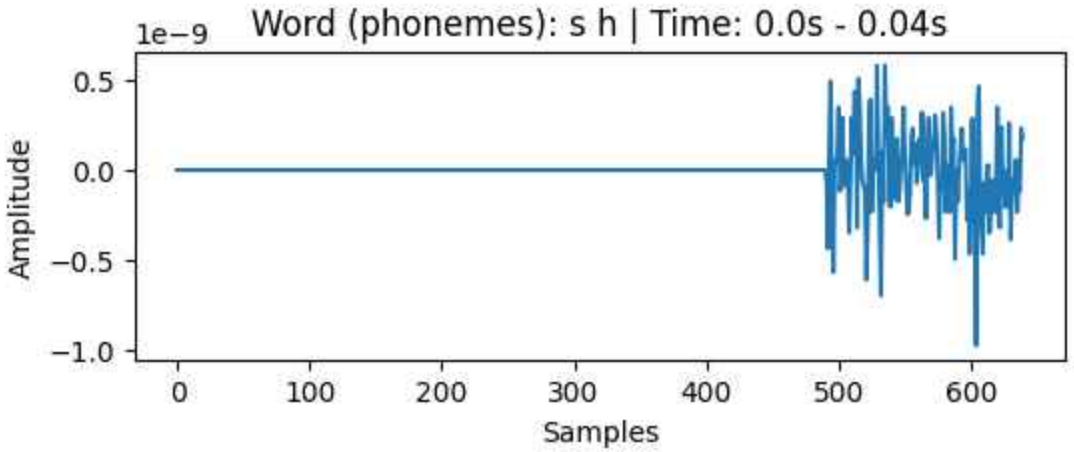
2

```
import torch
import librosa
import soundfile as sf
from transformers import Wav2Vec2Processor, Wav2Vec2ForCTC

processor = Wav2Vec2Processor.from_pretrained("Bluecast/wav2vec2-Phoneme")
model = Wav2Vec2ForCTC.from_pretrained("Bluecast/wav2vec2-Phoneme")

if sr != 16000:
    y = librosa.resample(y.astype(float), orig_sr=sr, target_sr=16000)
    sr = 16000

input_values = processor(y, sampling_rate=sr,
return_tensors="pt").input_values

with torch.no_grad():
  logits = model(input_values).logits

predicted_ids = torch.argmax(logits, dim=-1)
phoneme_sequence = processor.batch_decode(predicted_ids)
print("Recognized phoneme sequence:")
print(phoneme_sequence[0])
```

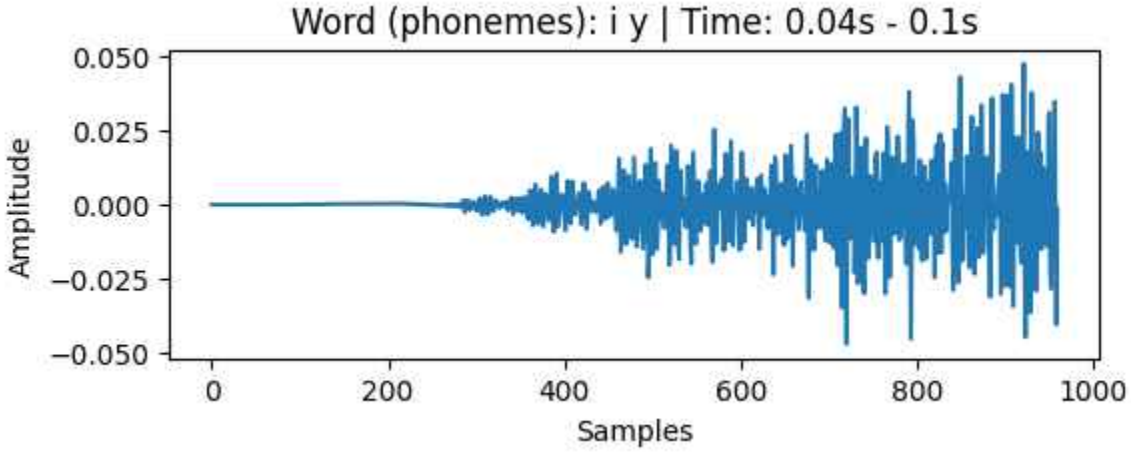
```
model.safetensors:   0%|          | 0.00/1.26G [00:00<?, ?B/s]
```

```
Loading weights:   0%|          | 0/424 [00:00<?, ?it/s]
```

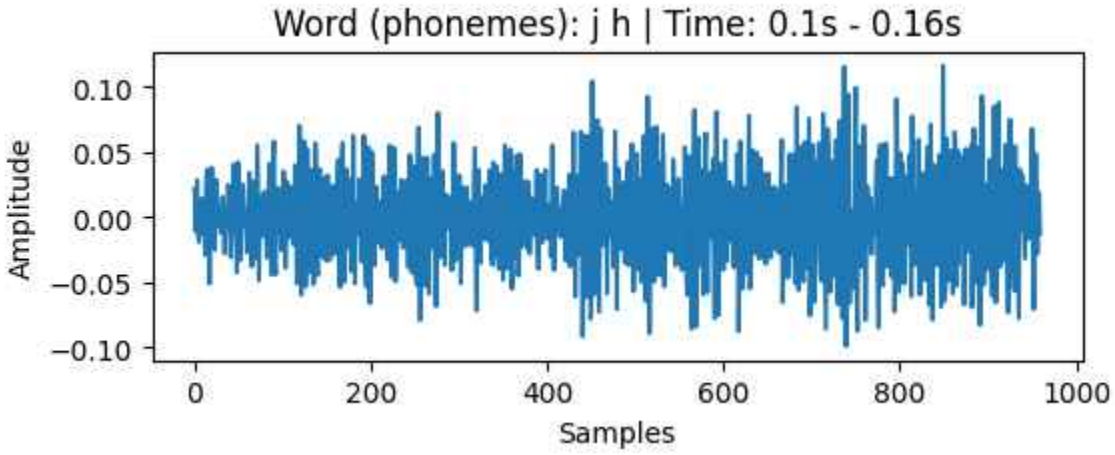
```
Recognized phoneme sequence:
sh iy jh ah jh d dh ah b l uw p eh n p uh t ih t aa n dh ah t ey b ah l ah n d

3

phoneme_sequence_str = processor.batch_decode(predicted_ids)[0]
words = phoneme_sequence_str.strip().split(" ")
frame_ids = torch.argmax(logits, dim=-1)[0].cpu().numpy()
frame_phonemes = processor.batch_decode([frame_ids])[0]
frame_duration = model.config.inputs_to_logits_ratio / sr

segments = []
current_word = []
start_time = 0.0

for i, phoneme in enumerate(frame_phonemes):
    if phoneme == " " or phoneme == "|":
        if current_word:
            end_time = i * frame_duration
            segments.append((" ".join(current_word), start_time,
end_time))
            current_word = []
            start_time = end_time
    else:
        current_word.append(phoneme)
if current_word:
    segments.append((" ".join(current_word), start_time,
len(frame_phonemes) * frame_duration))

for word, start, end in segments:
    start_sample = int(start * sr)
    end_sample = int(end * sr)
    plt.figure(figsize=(6, 2))
    plt.plot(y[start_sample:end_sample])
    plt.title(f"Word (phonemes): {word} | Time: {round(start,2)}s - {round(en
    plt.xlabel("Samples")
    plt.ylabel("Amplitude")
    plt.show()
```



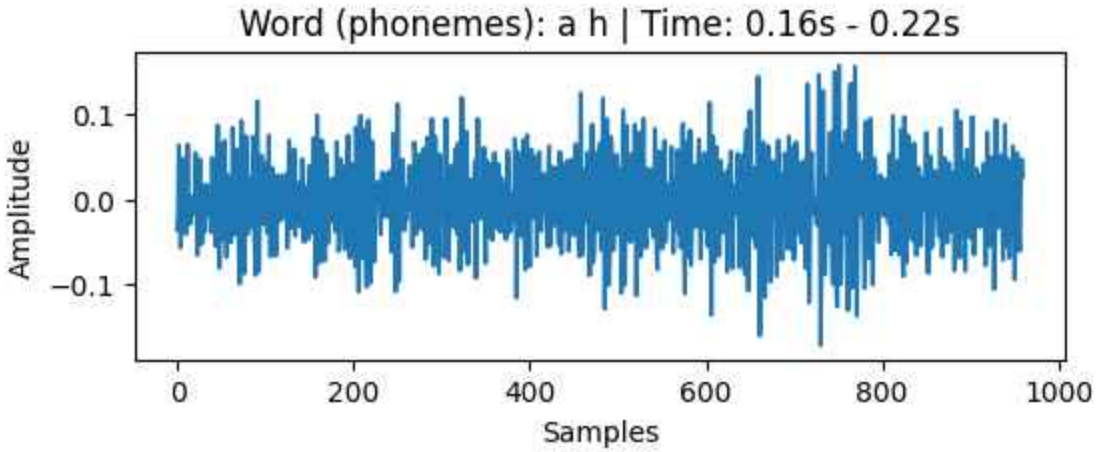
jpeg



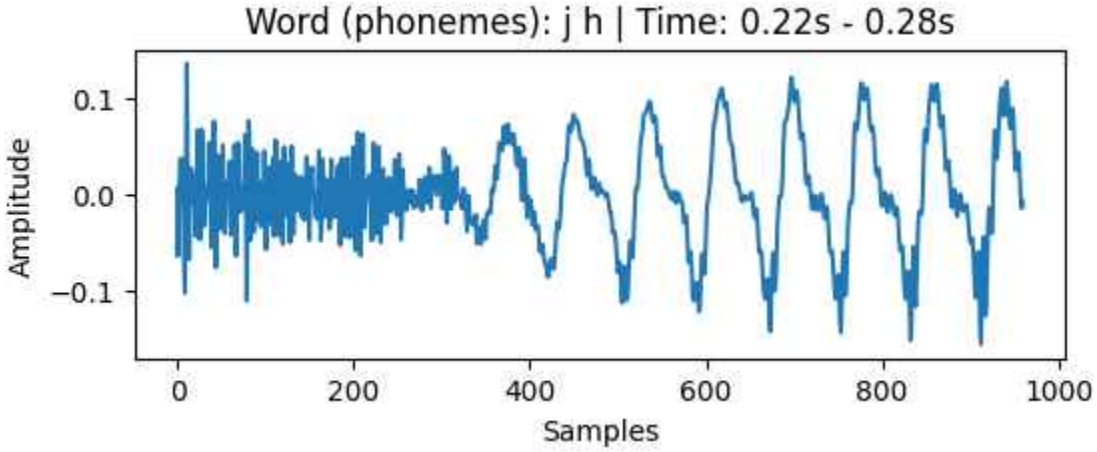
jpeg



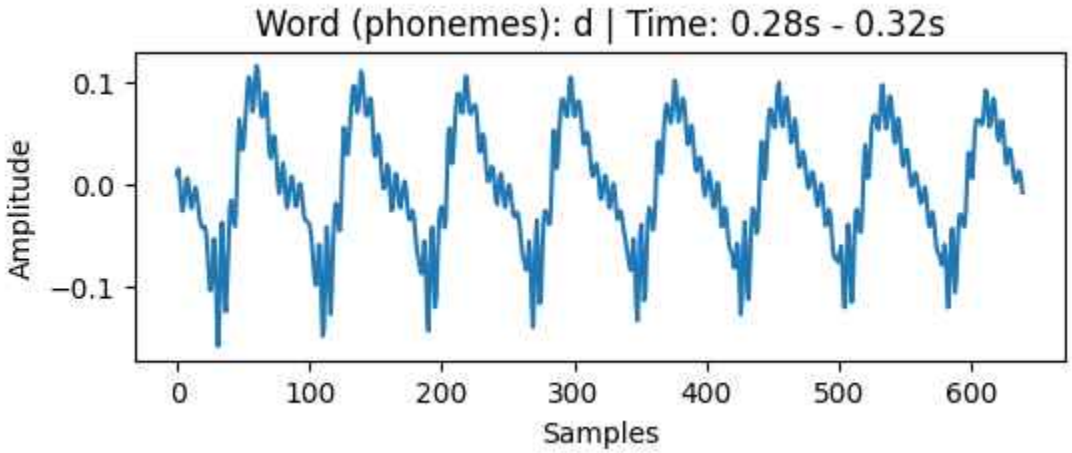
jpeg



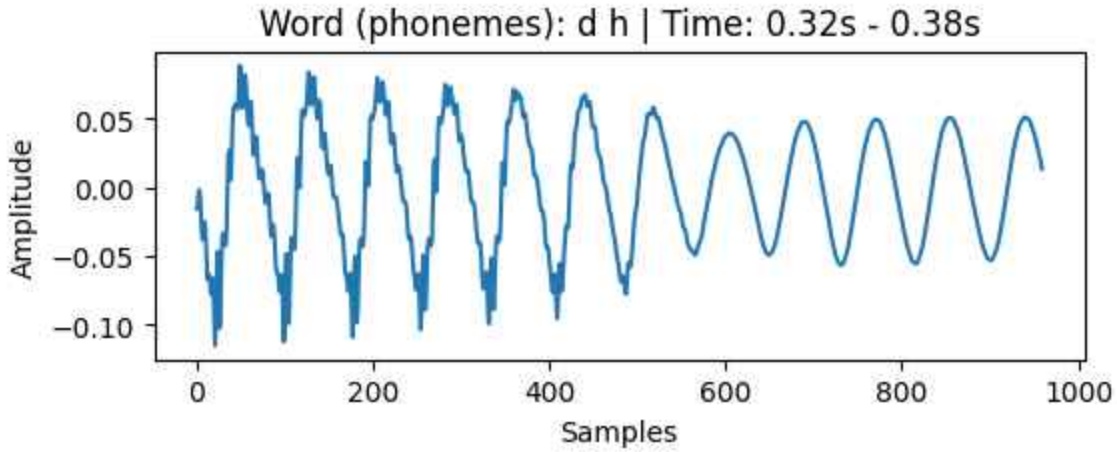
jpeg



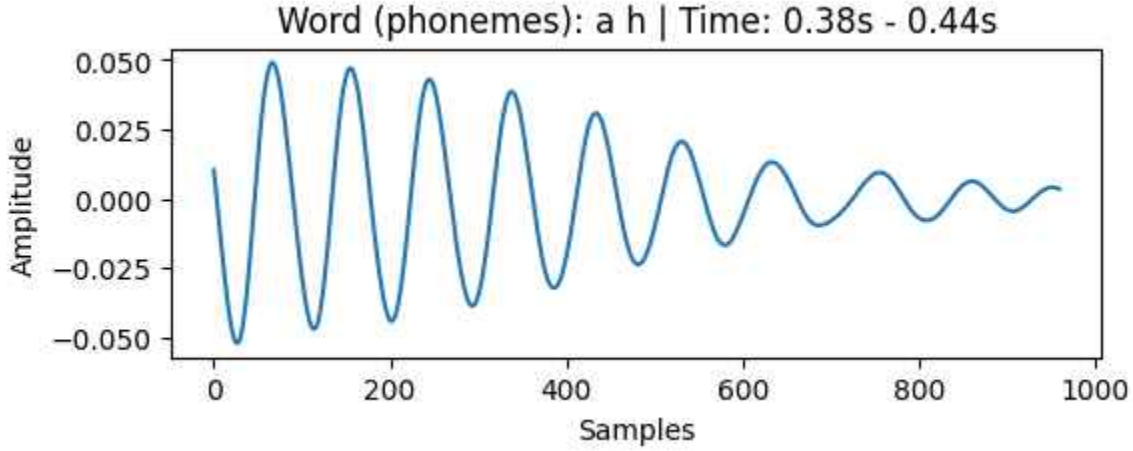
jpeg



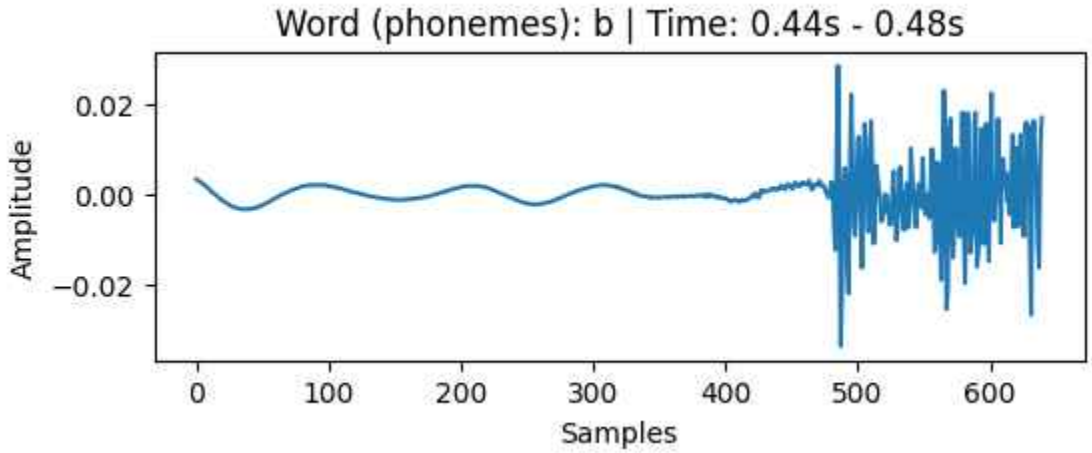
jpeg



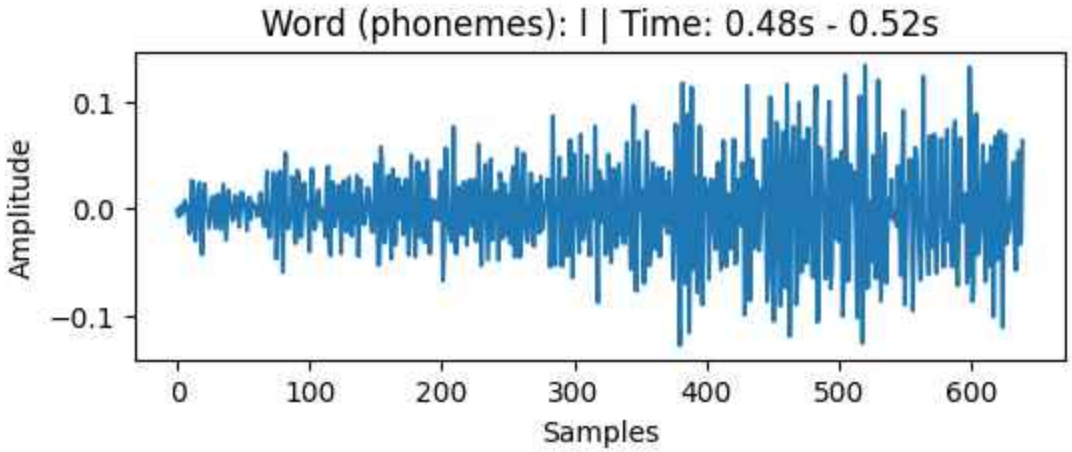
jpeg



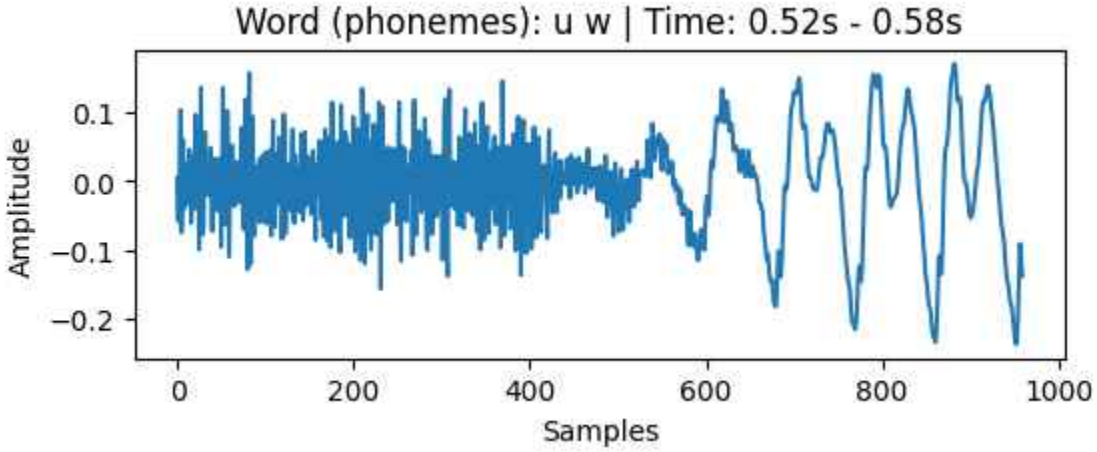
jpeg



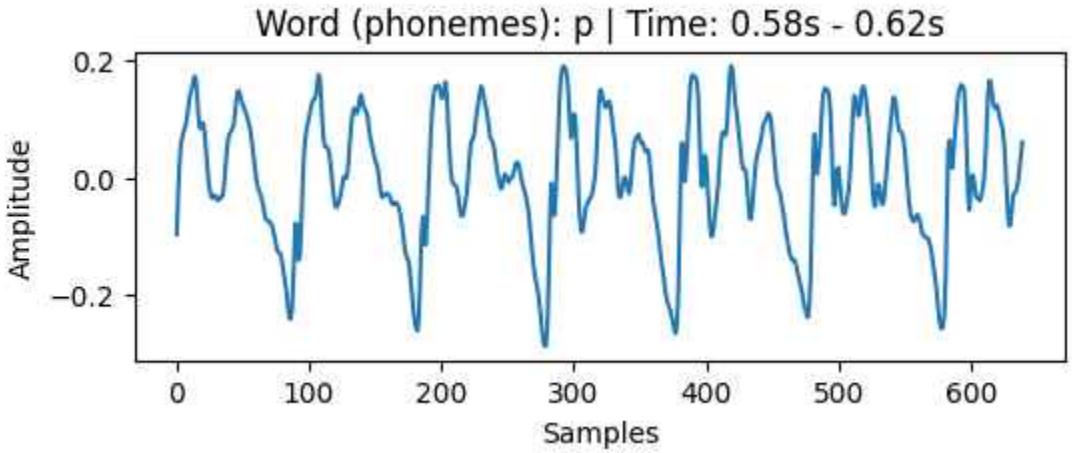
jpeg



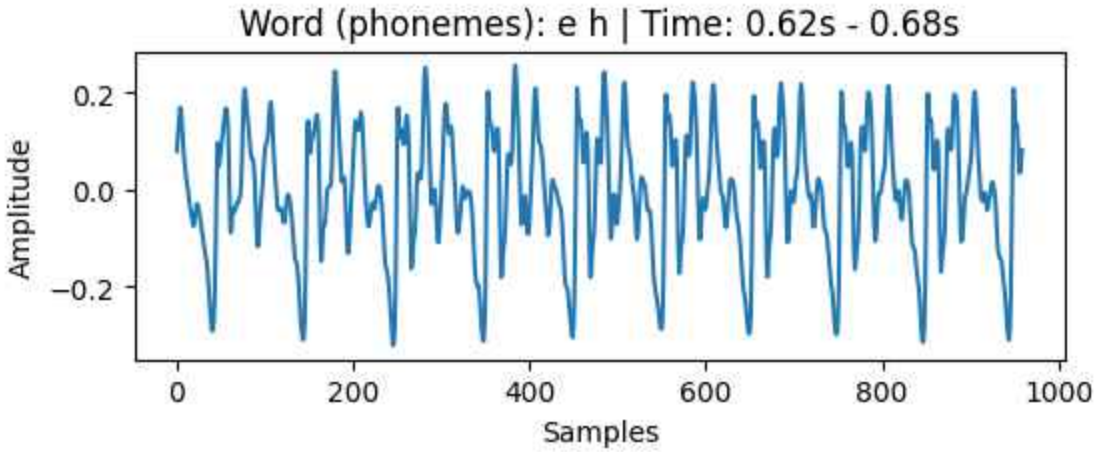
jpeg



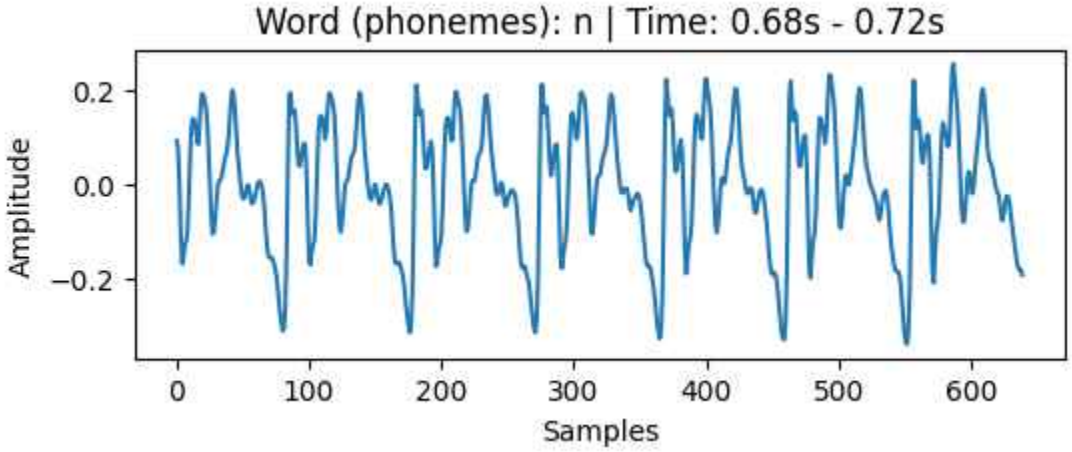
jpeg



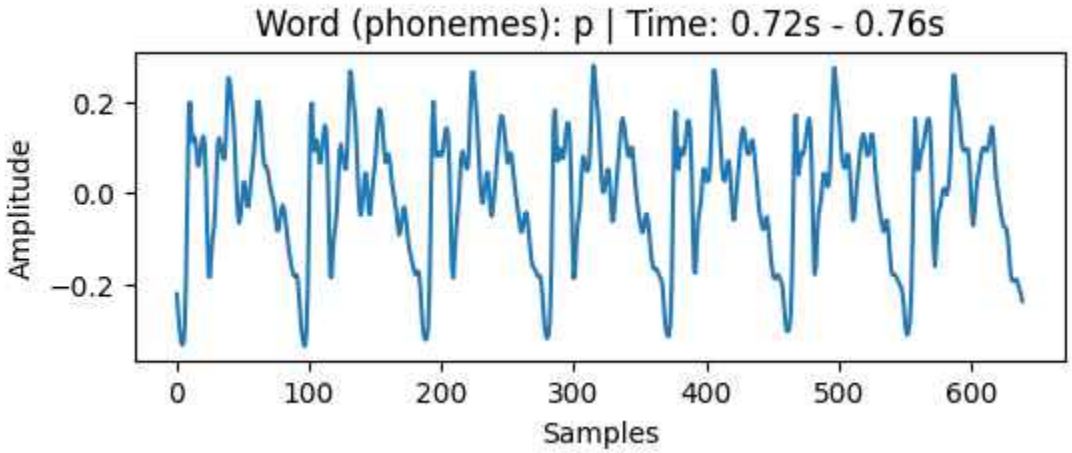
jpeg



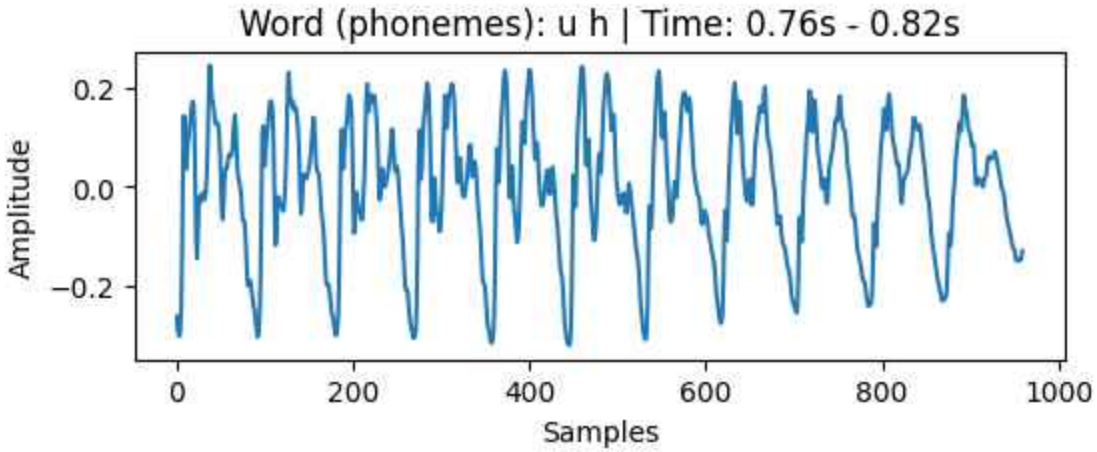
jpeg



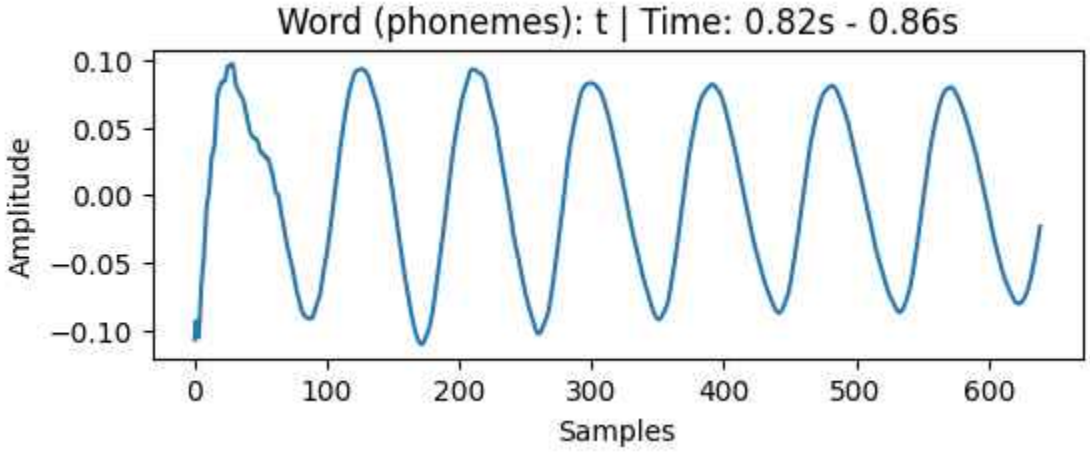
jpeg



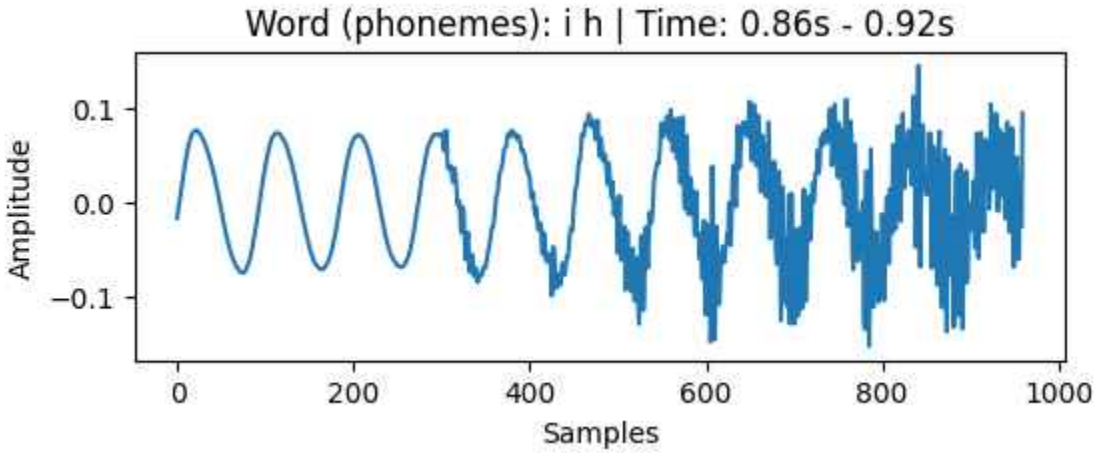
jpeg



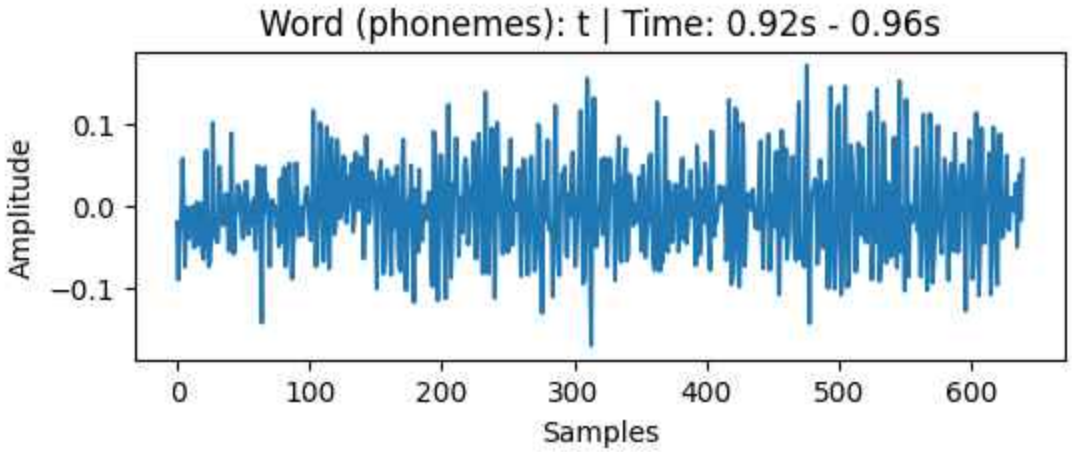
jpeg



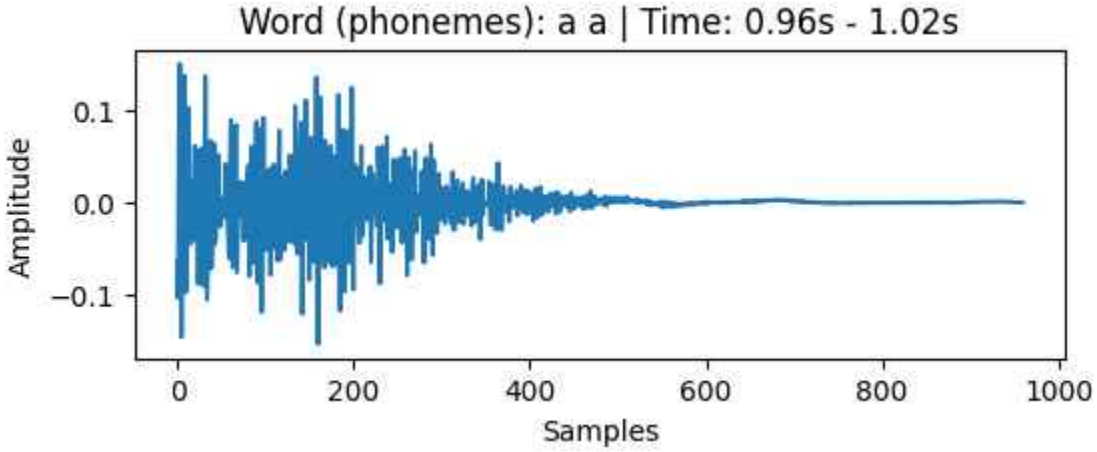
jpeg



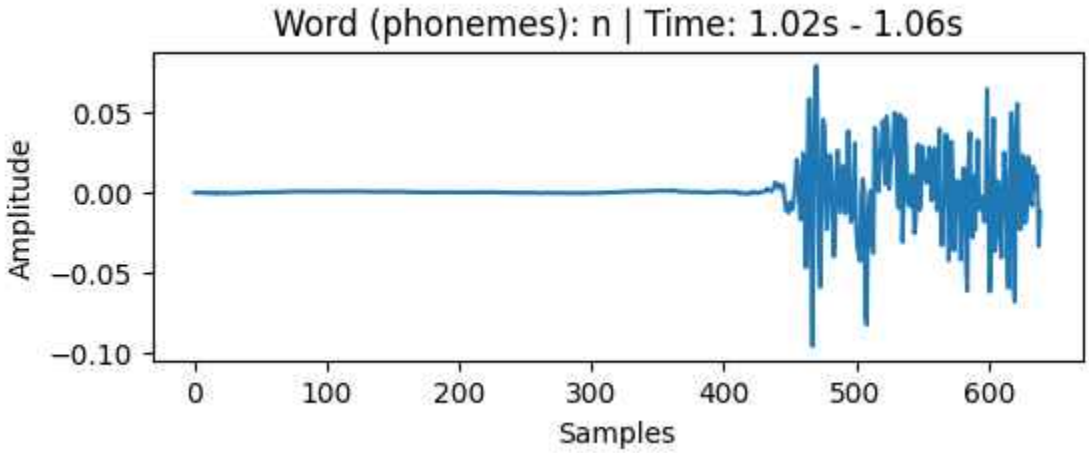
jpeg



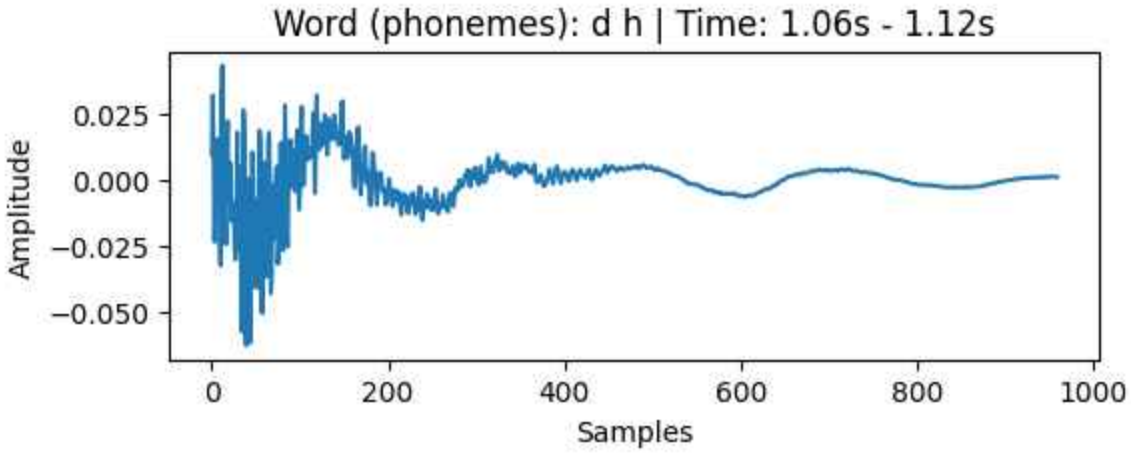
jpeg



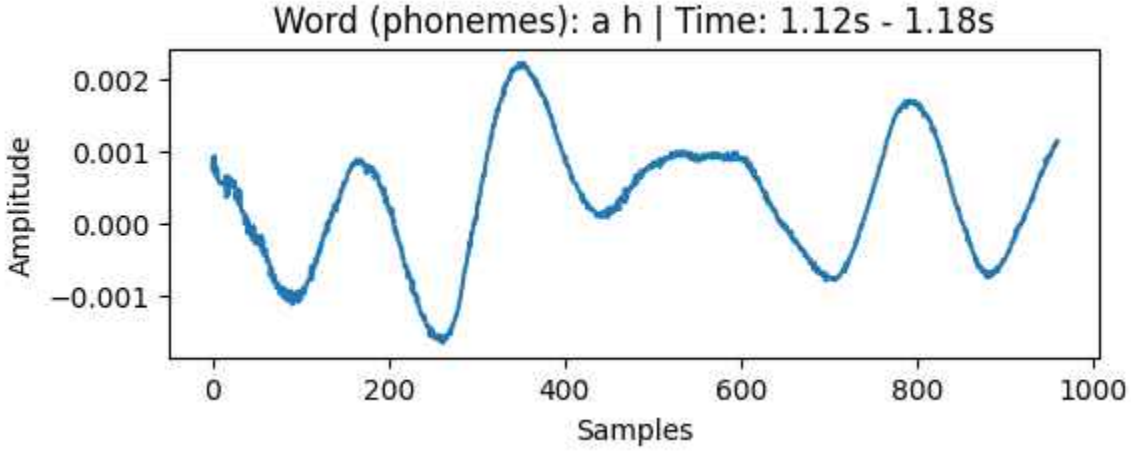
jpeg



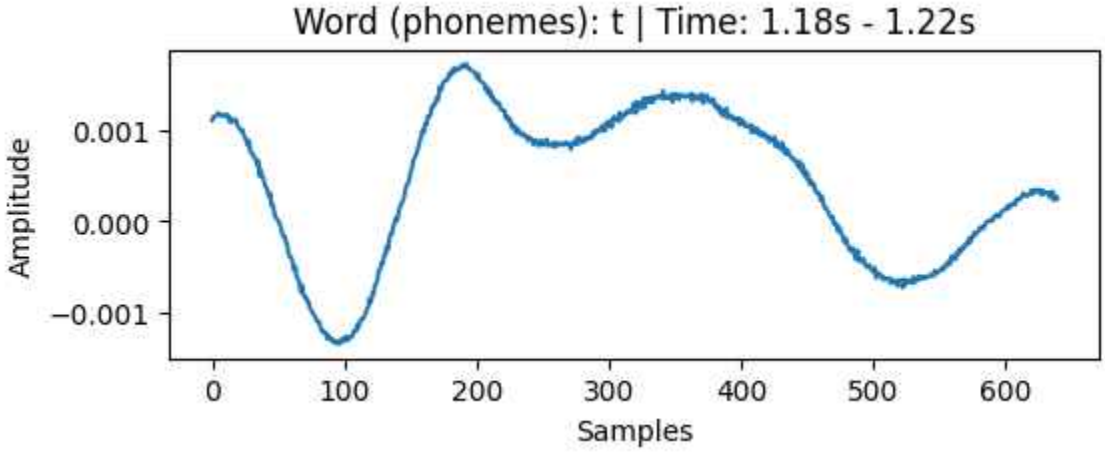
jpeg



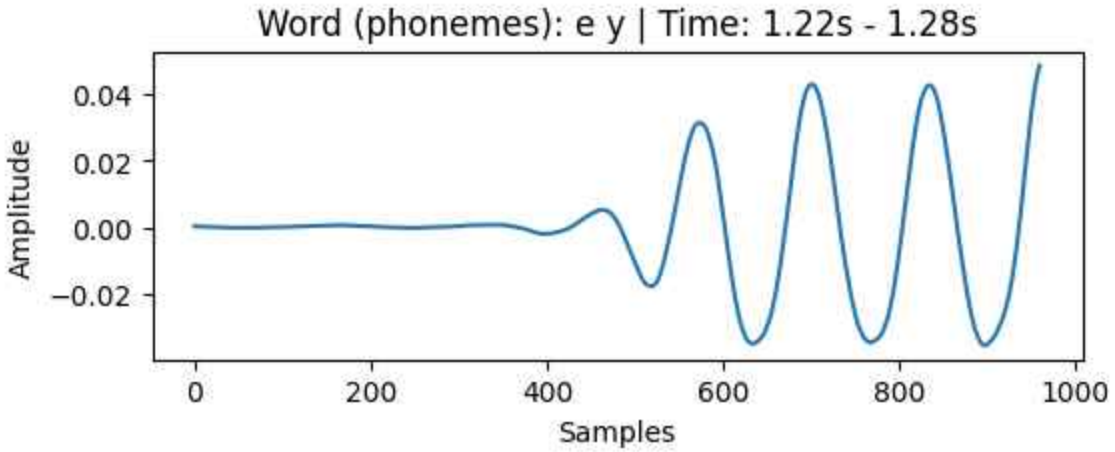
jpeg



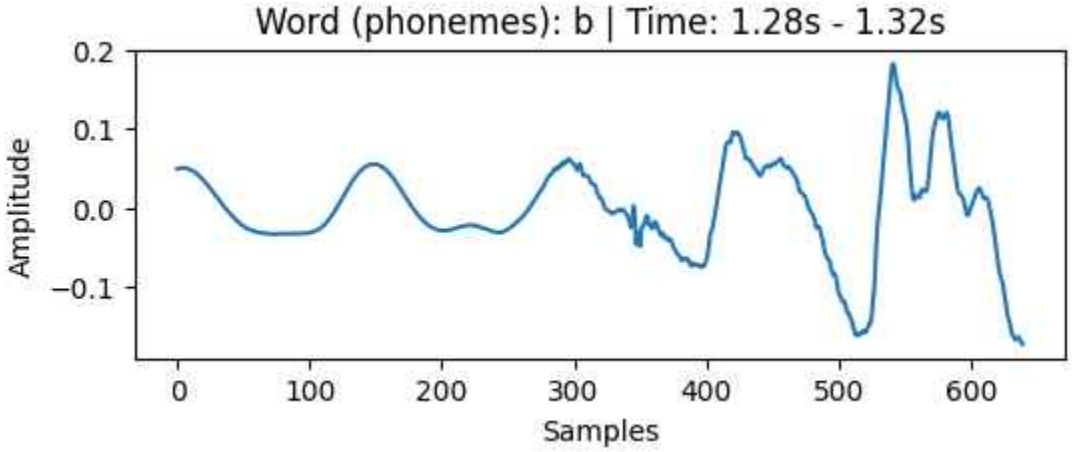
jpeg



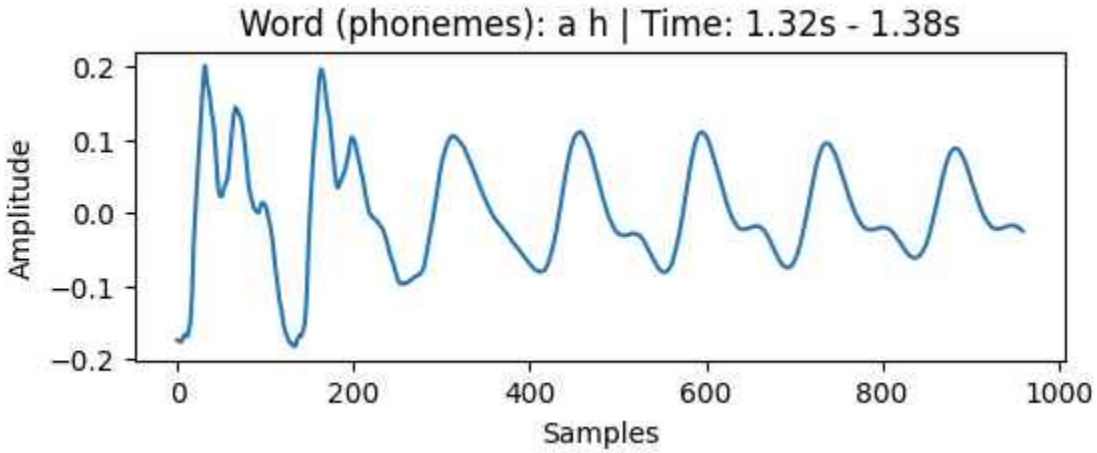
jpeg



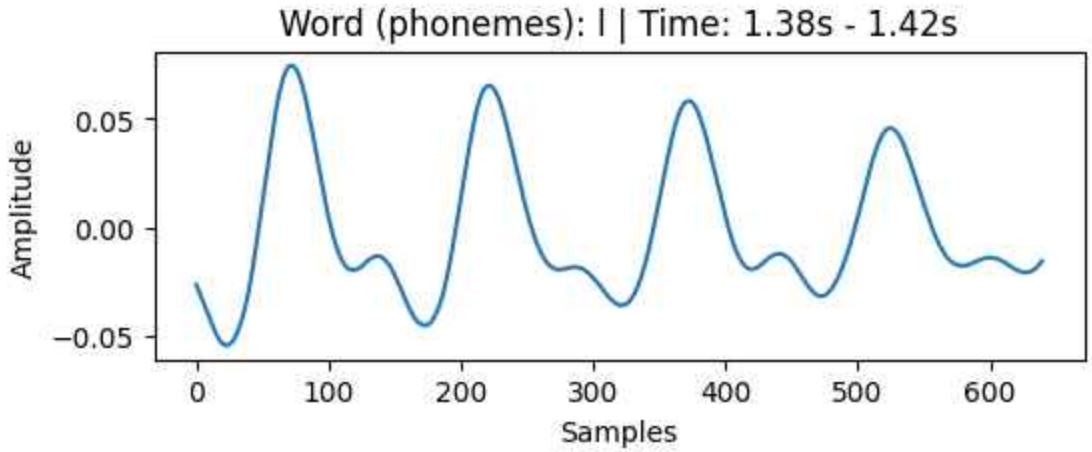
jpeg



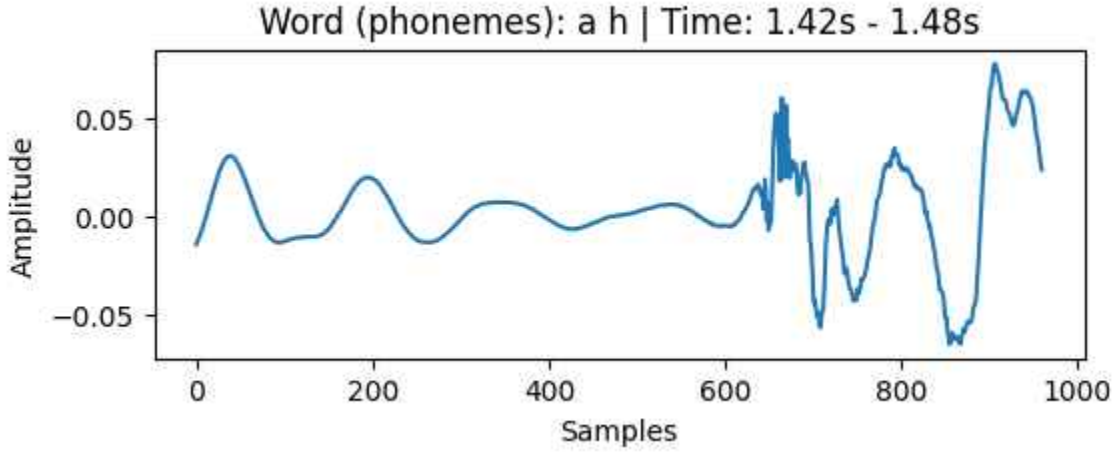
jpeg



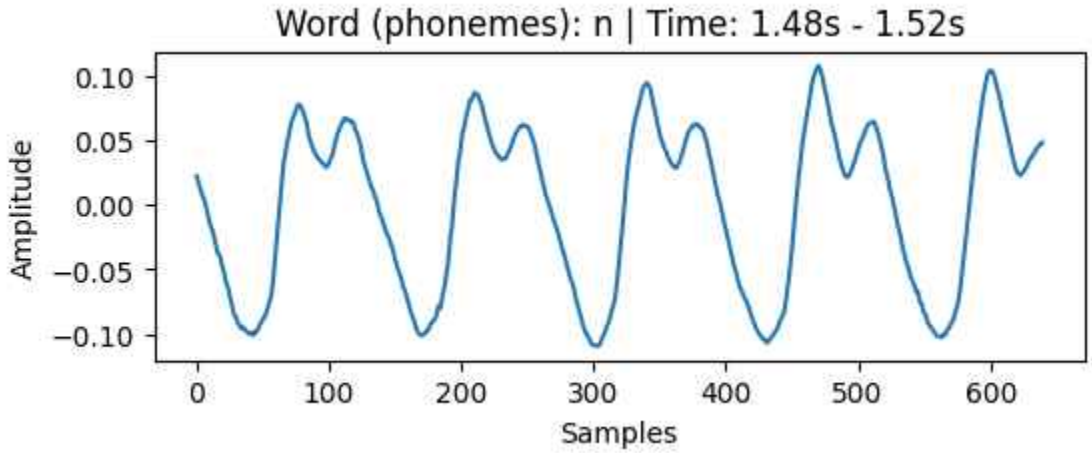
jpeg



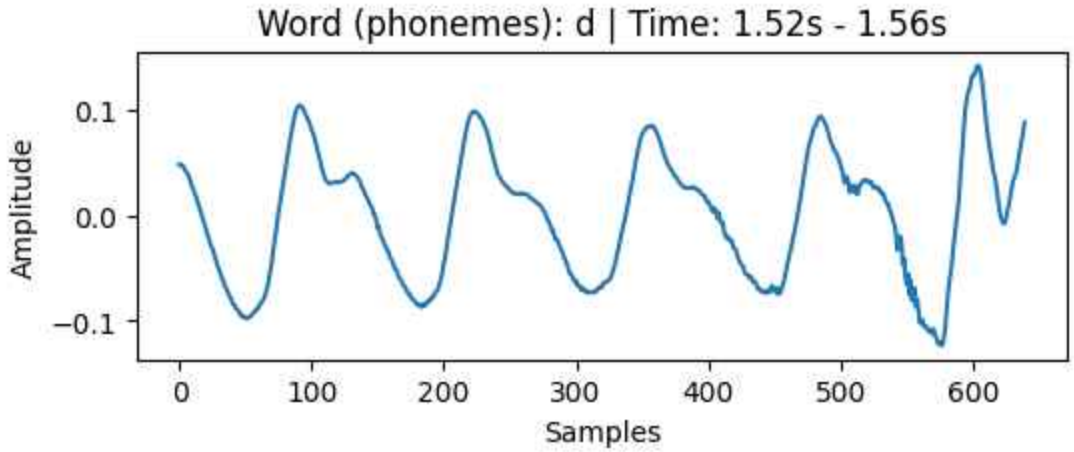
jpeg



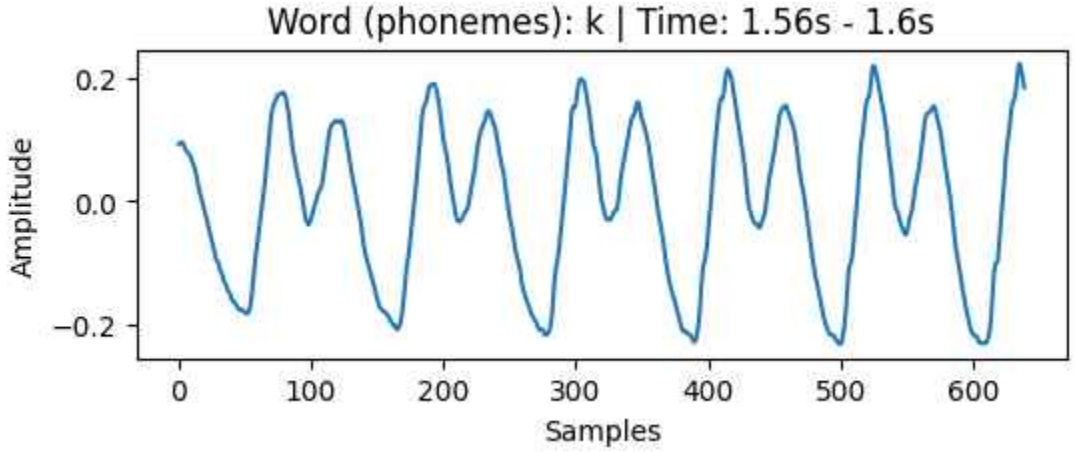
jpeg



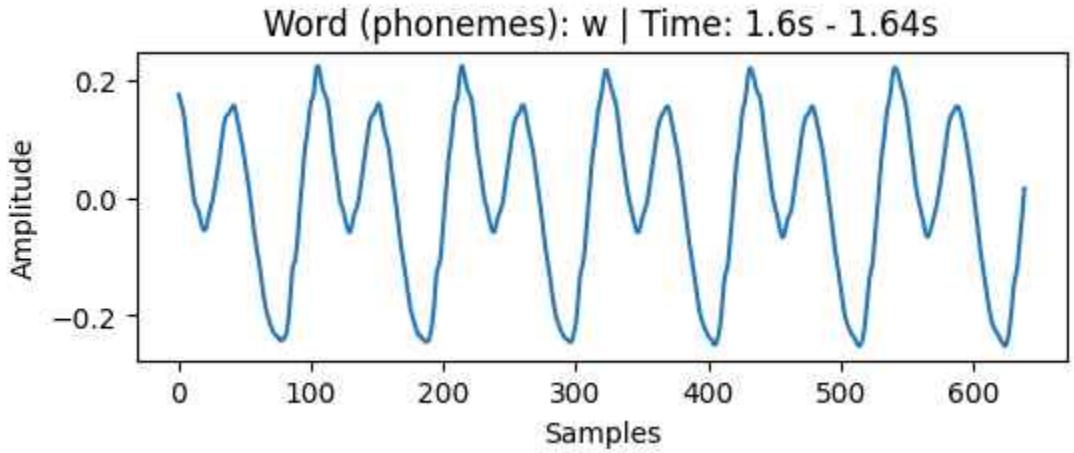
jpeg



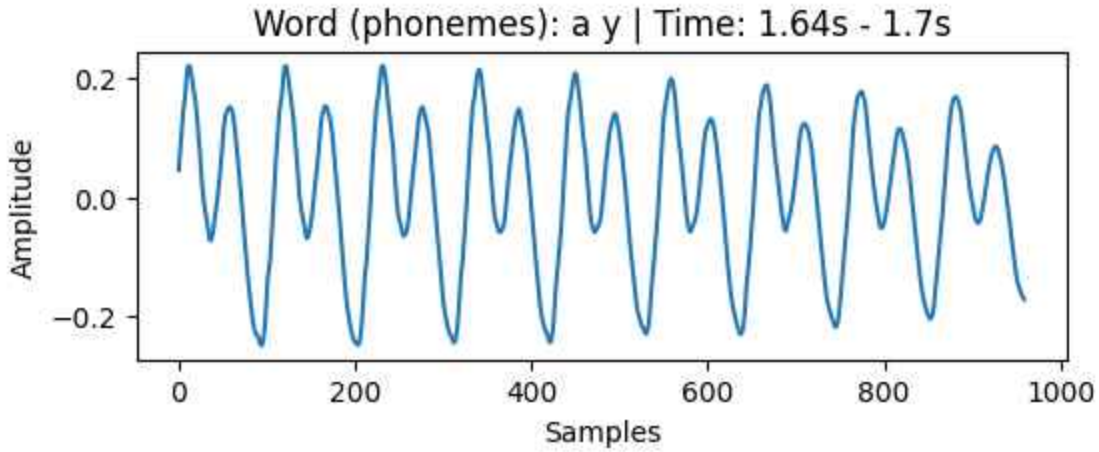
jpeg



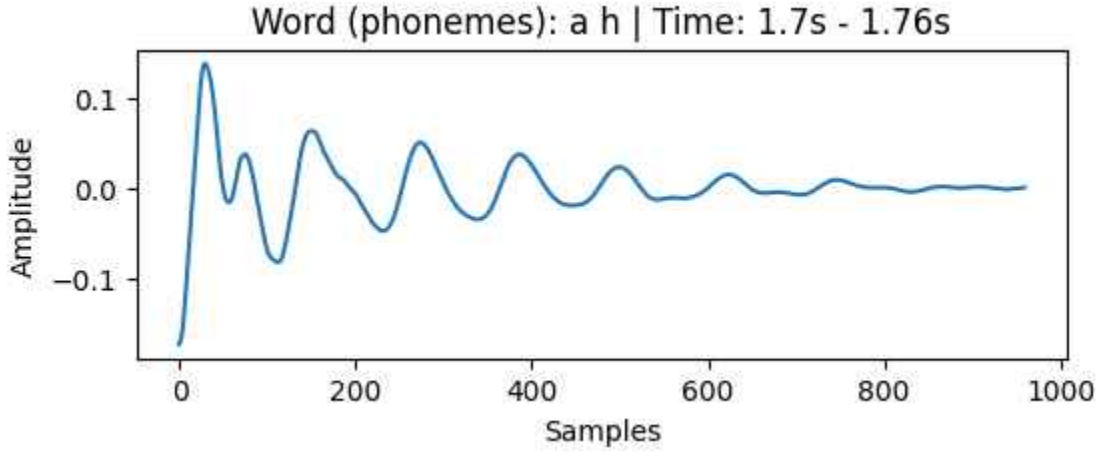
jpeg



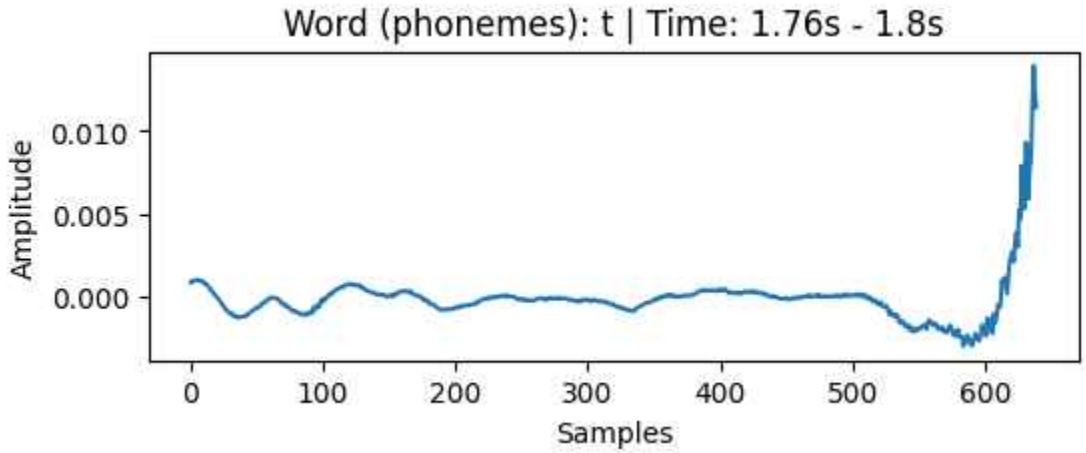
jpeg



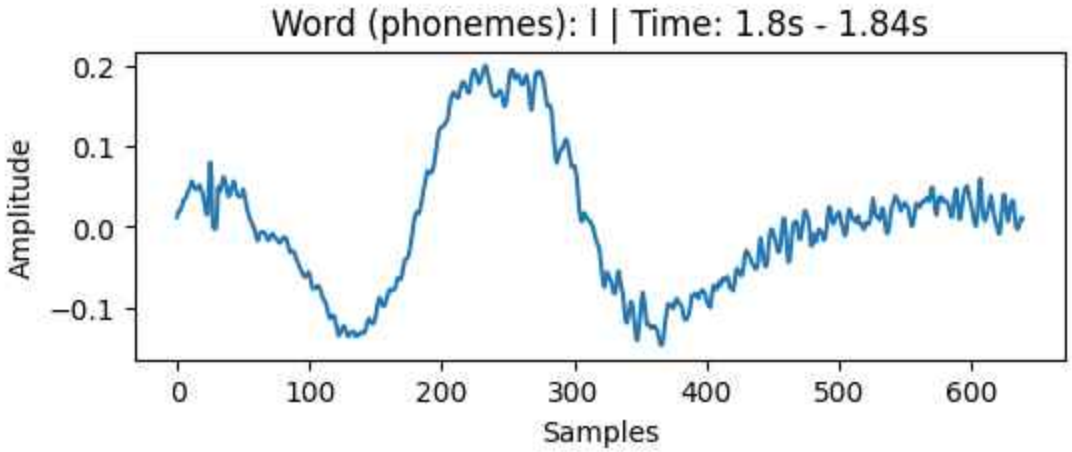
jpeg



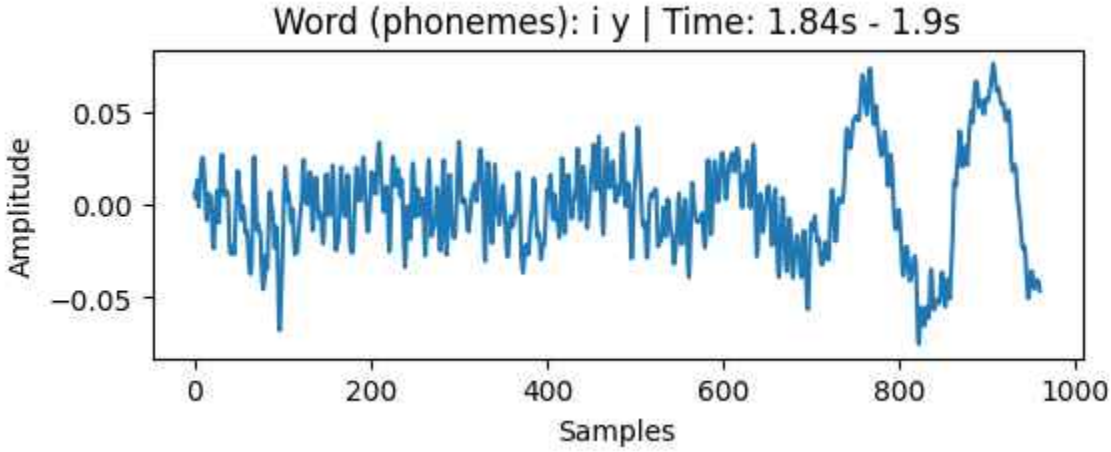
jpeg



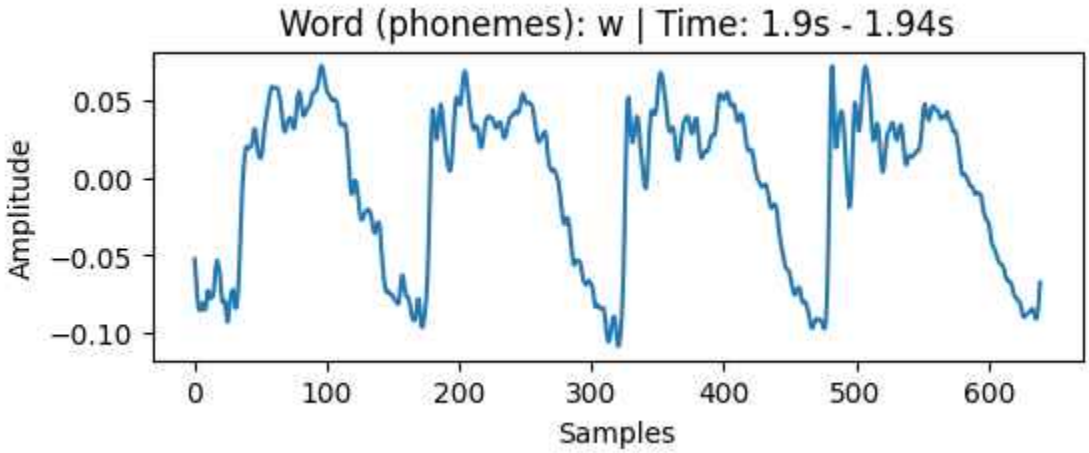
jpeg



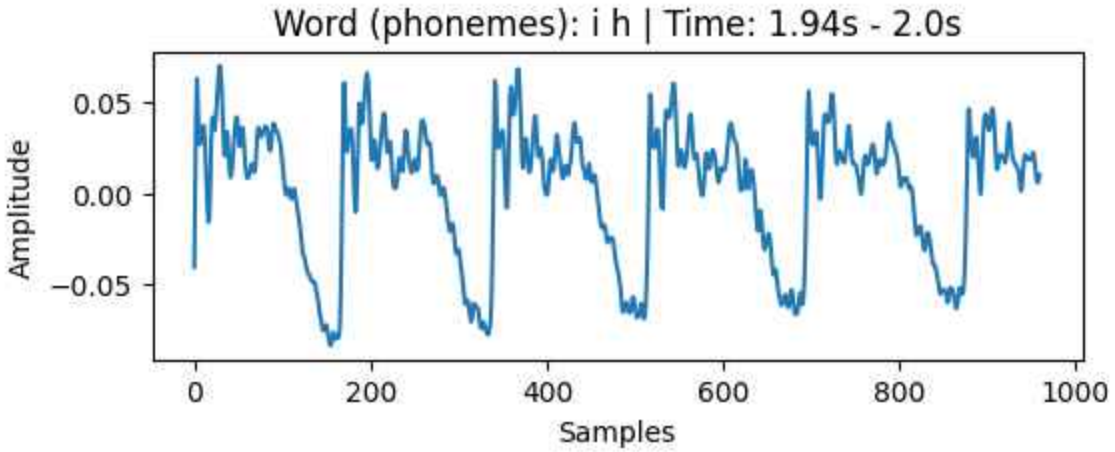
jpeg



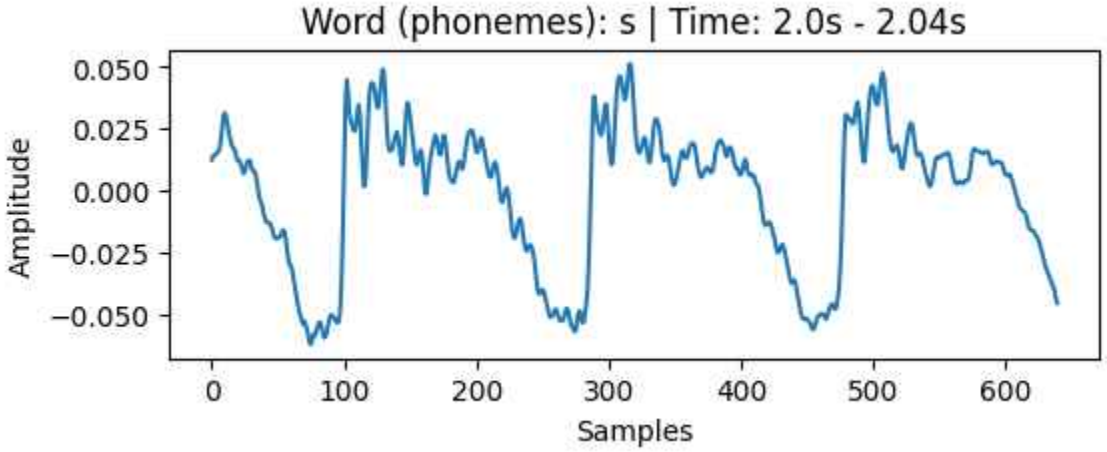
jpeg



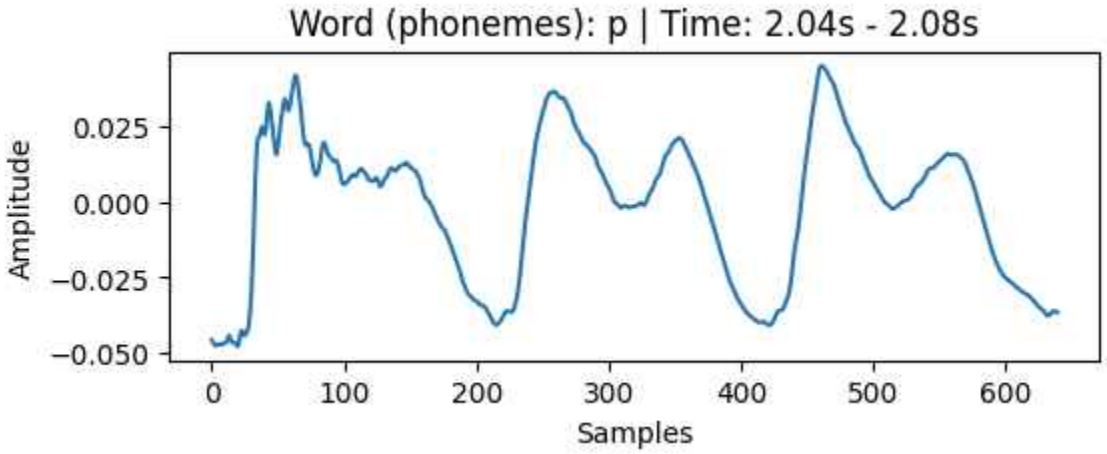
jpeg



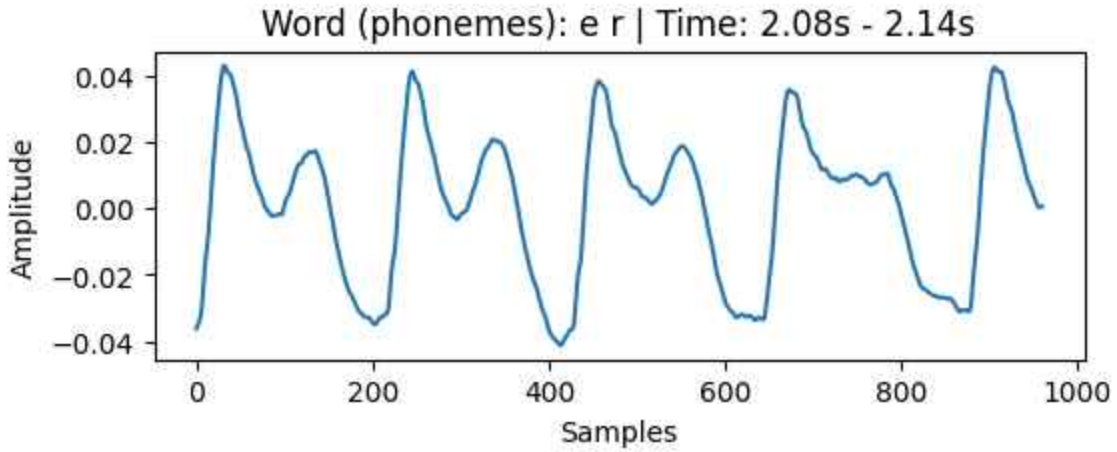
jpeg



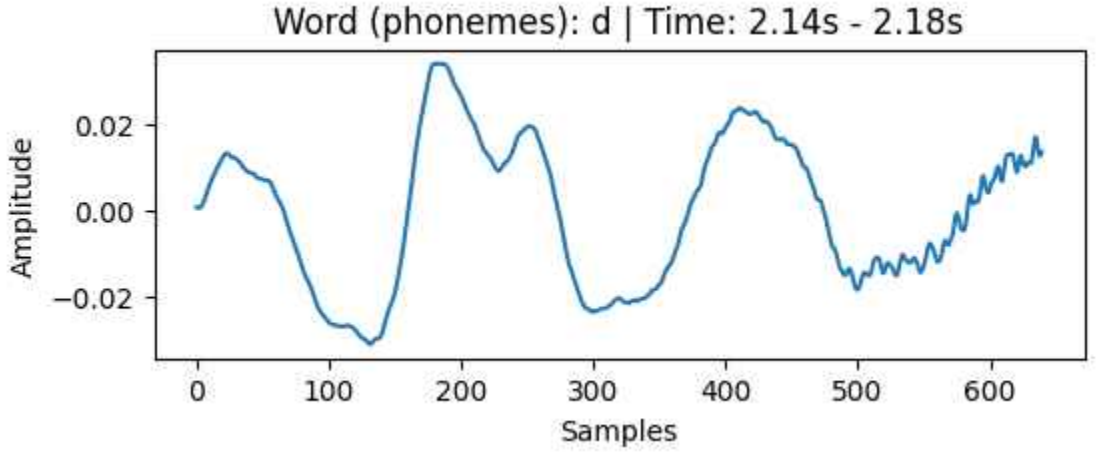
jpeg



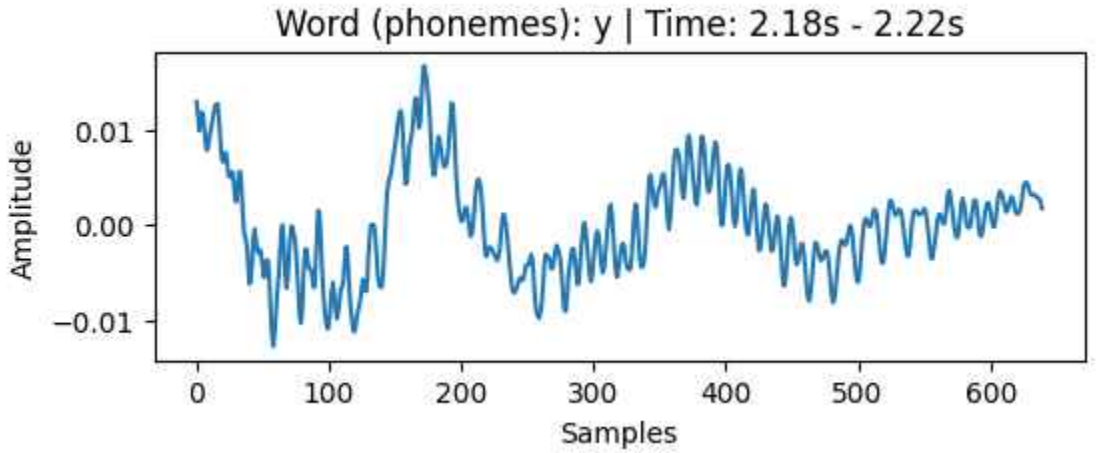
jpeg



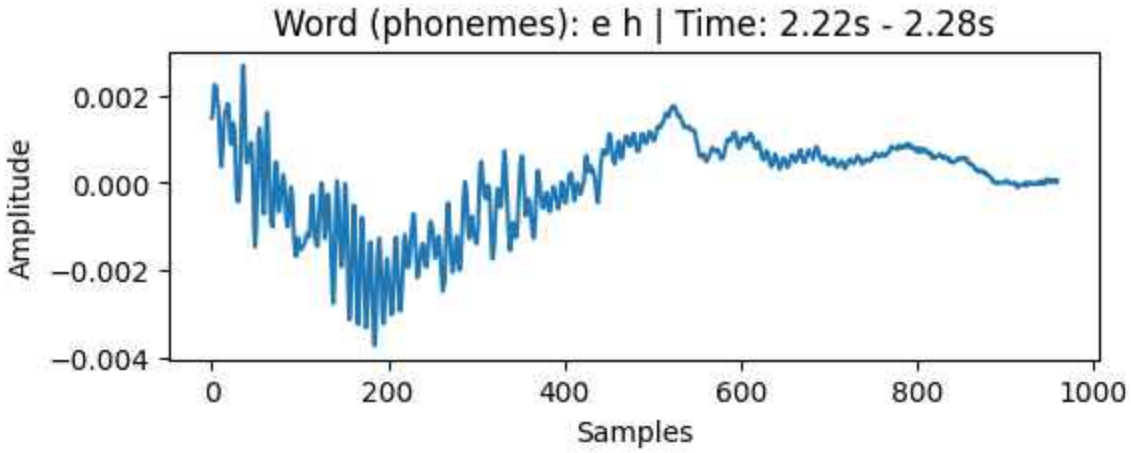
jpeg



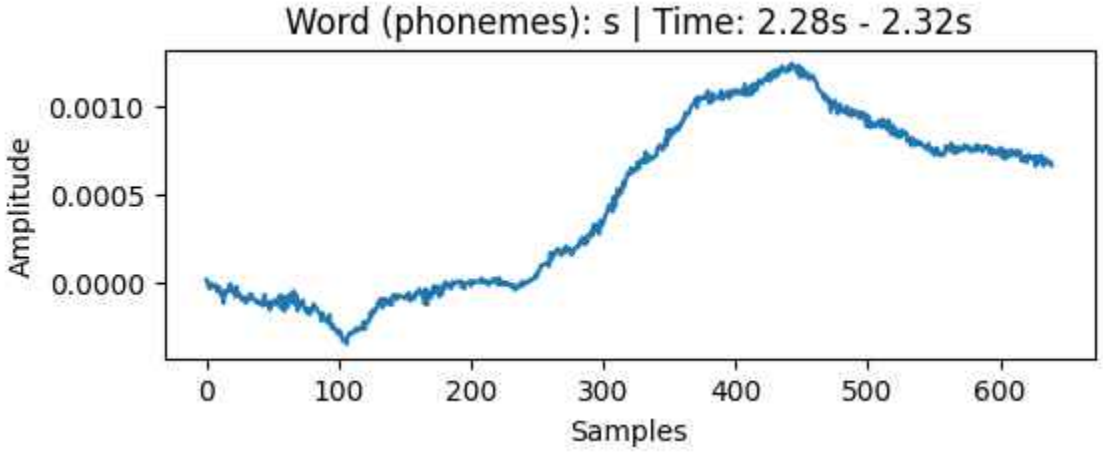
jpeg



jpeg



jpeg

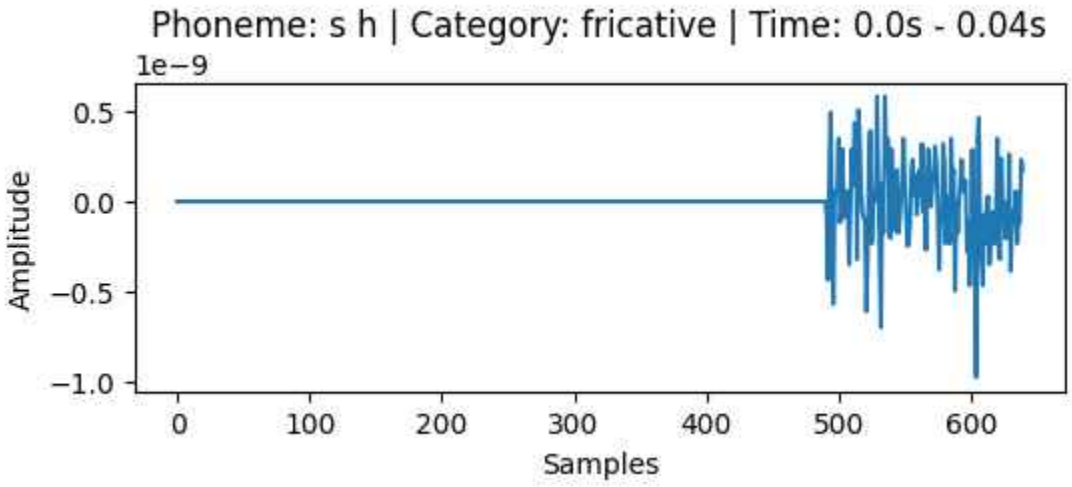


jpeg

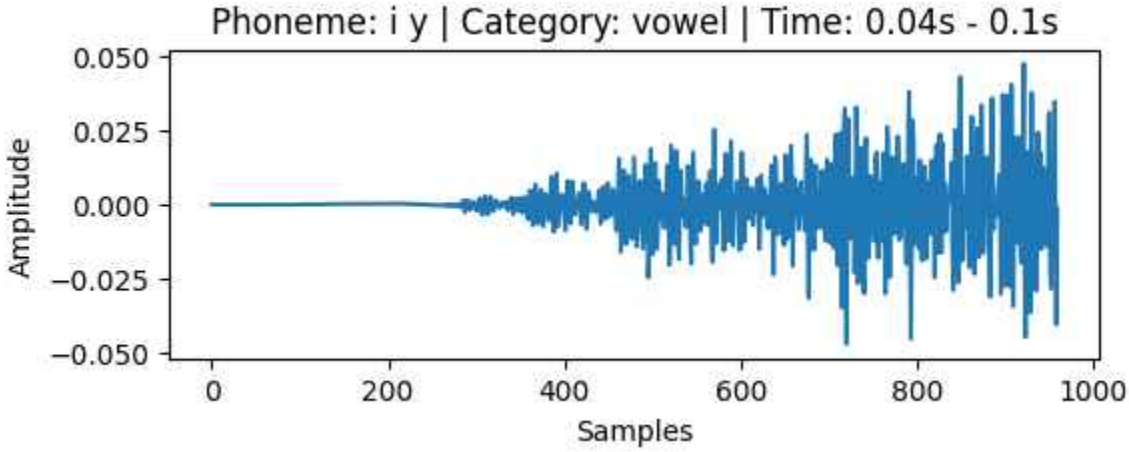
4

```
phoneme_categories = {
"vowel": ["a a", "a e", "a h", "a o", "e h", "e r", "i h", "i y", "u h", "u w
"plosive": ["p", "b", "t", "d", "k", "g"],
"fricative": ["f", "v", "t h", "d h", "s", "z", "s h", "z h", "h h"],
"affricate": ["c h", "j h"],
"semivowel": ["i", "r", "w", "y"],
"diphthong": ["a y", "e y", "o y", "a w", "o w"],
"whisper": ["s i l", "s p n"]
}
def get_category(phoneme):
  for category, phones in phoneme_categories.items():
    if phoneme in phones:
      return category
  return "unknown"
for phoneme, start, end in segments:
  category = get_category(phoneme)
  start_sample = int(start * sr)
```

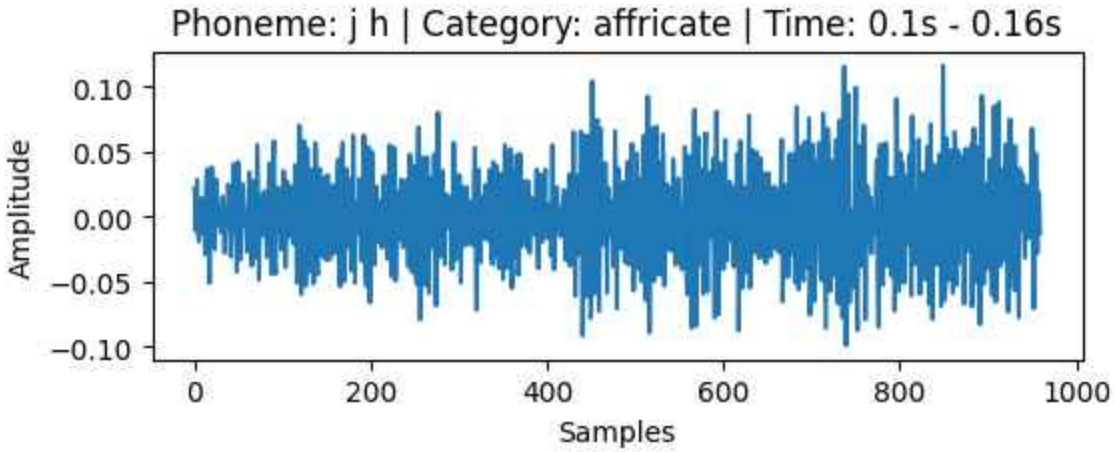
```
end_sample = int(end * sr)
plt.figure(figsize=(6, 2))
plt.plot(y[start_sample:end_sample])
plt.title(f"Phoneme: {phoneme} | Category: {category} | Time: {round(start,
plt.xlabel("Samples")
plt.ylabel("Amplitude")
plt.show()
```



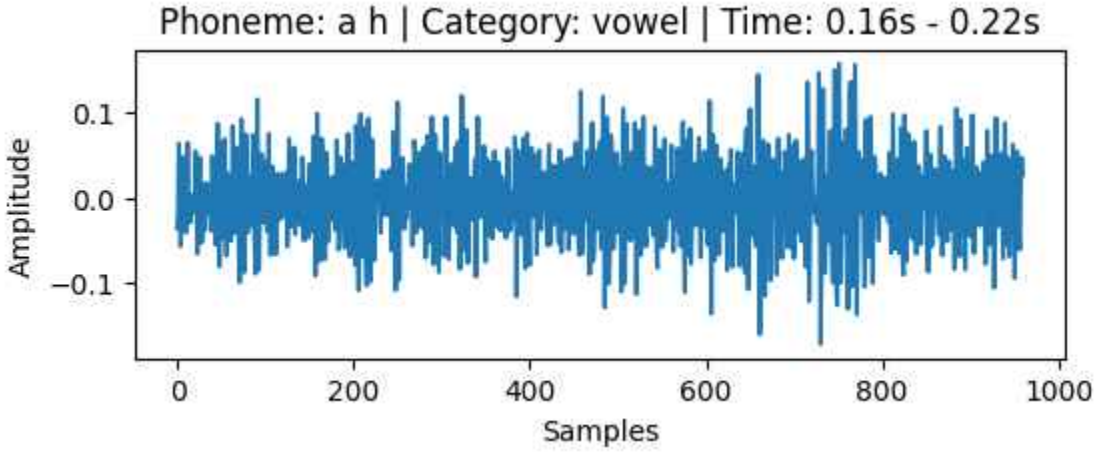
jpeg



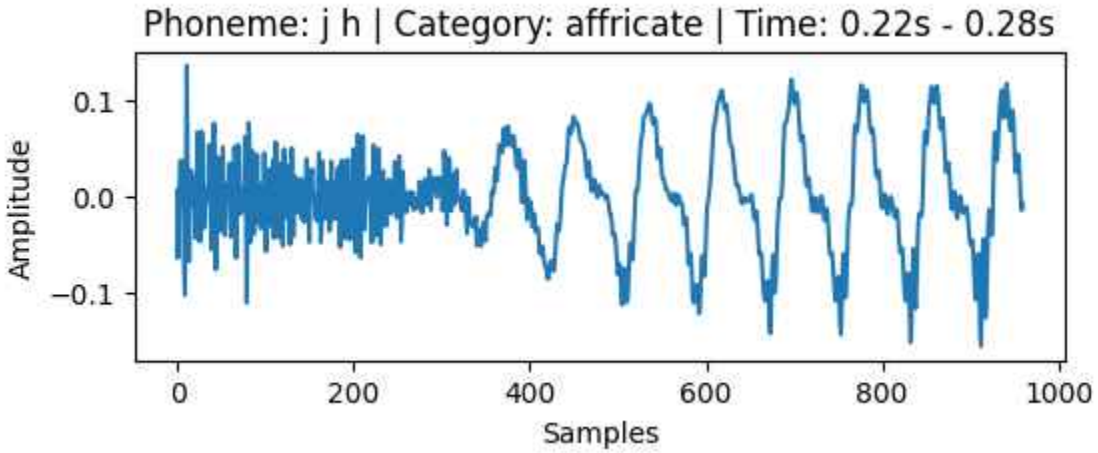
jpeg



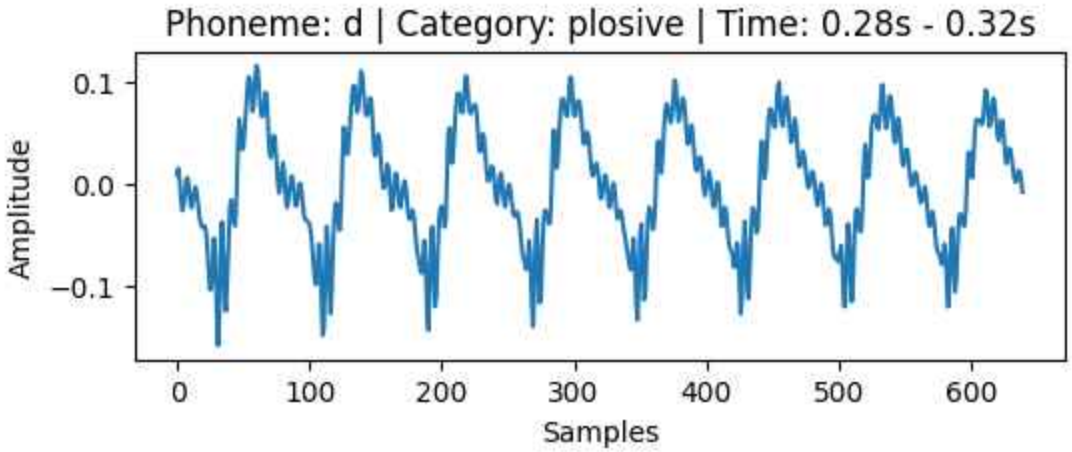
jpeg



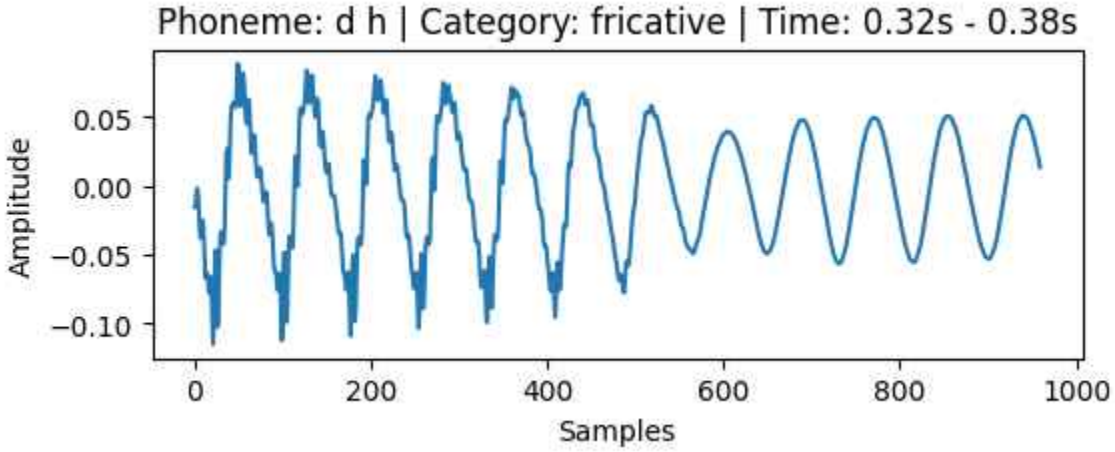
jpeg



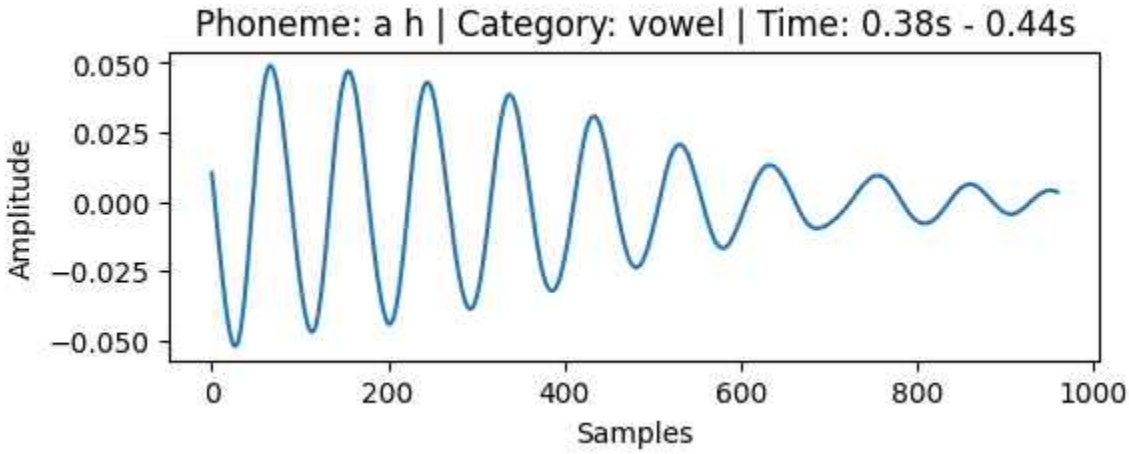
jpeg



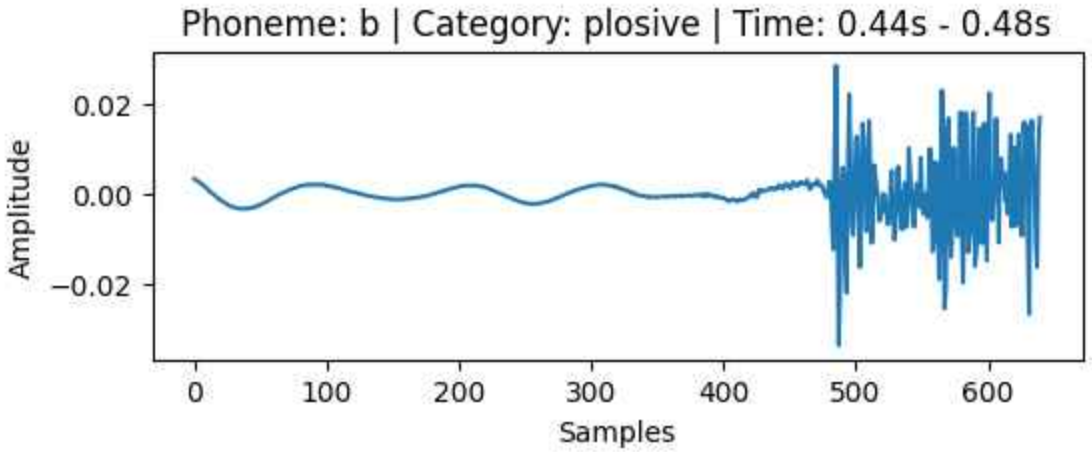
jpeg



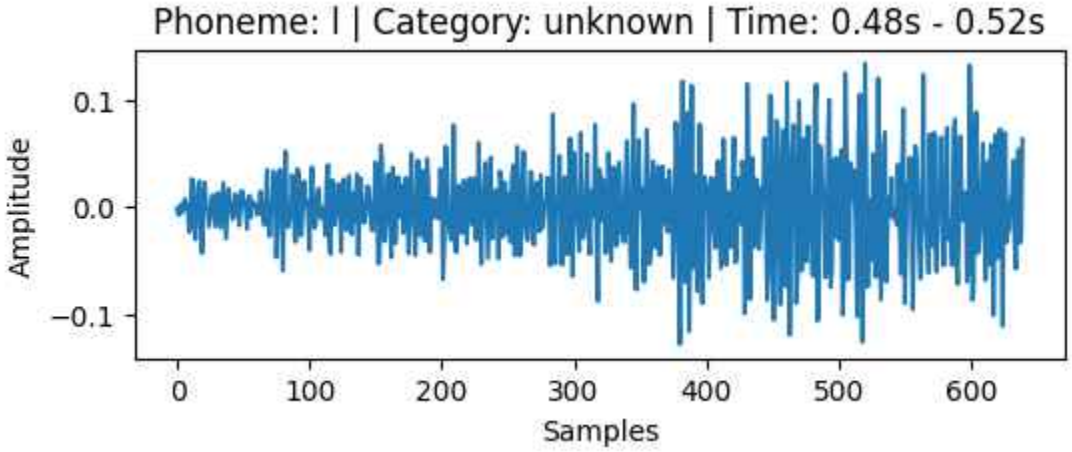
jpeg



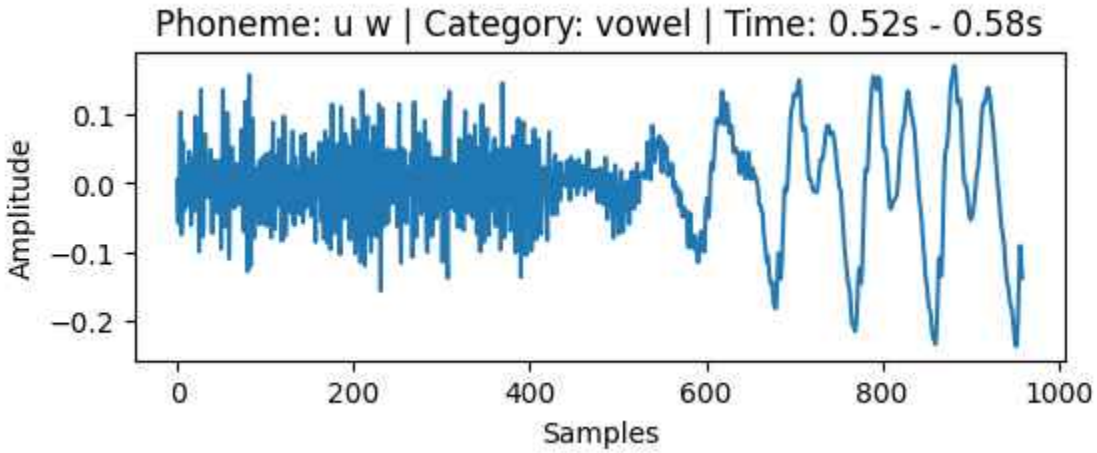
jpeg



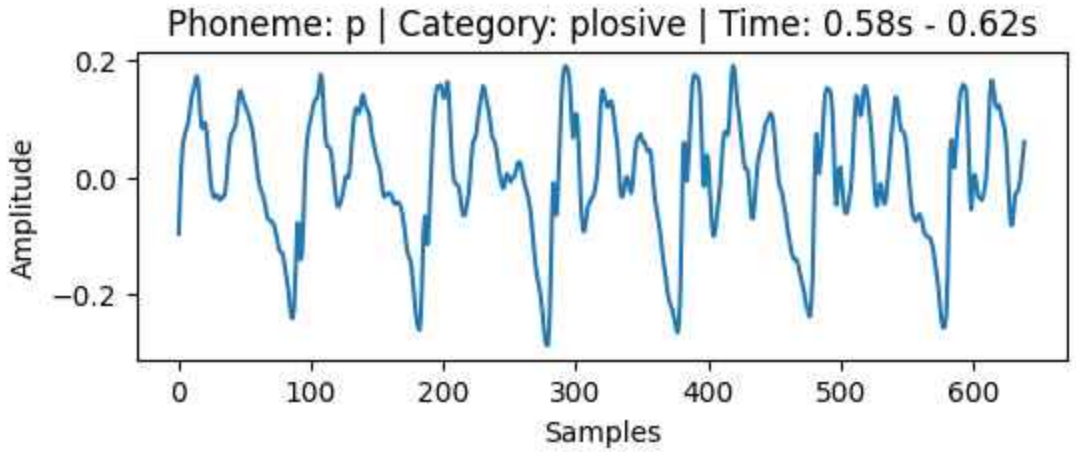
jpeg



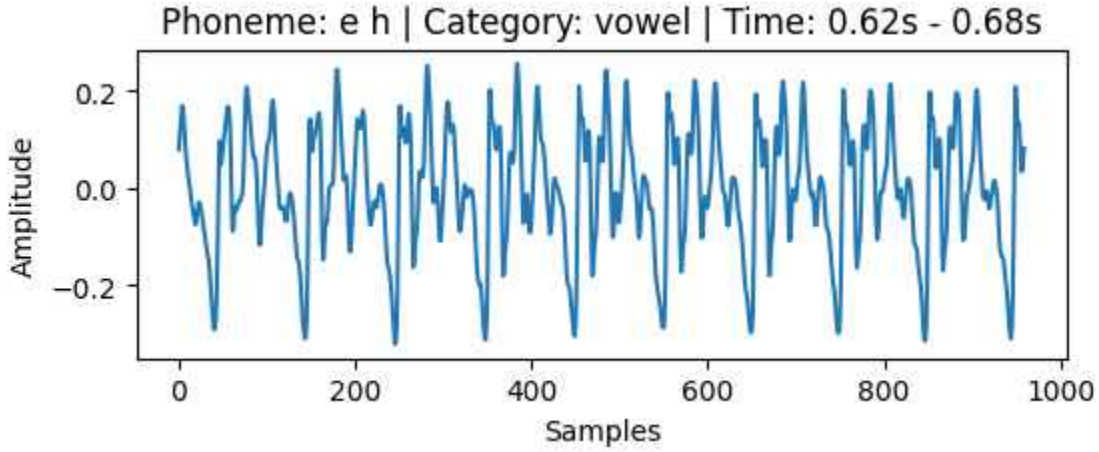
jpeg



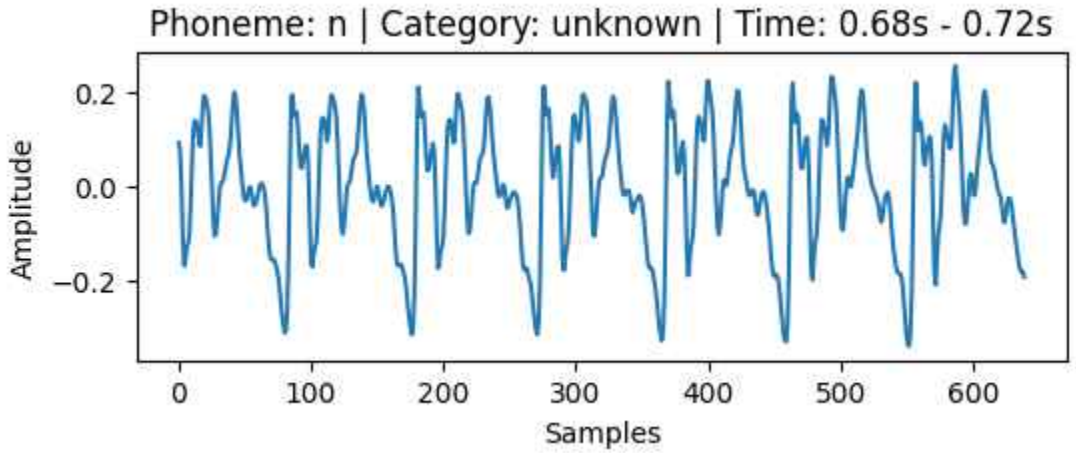
jpeg



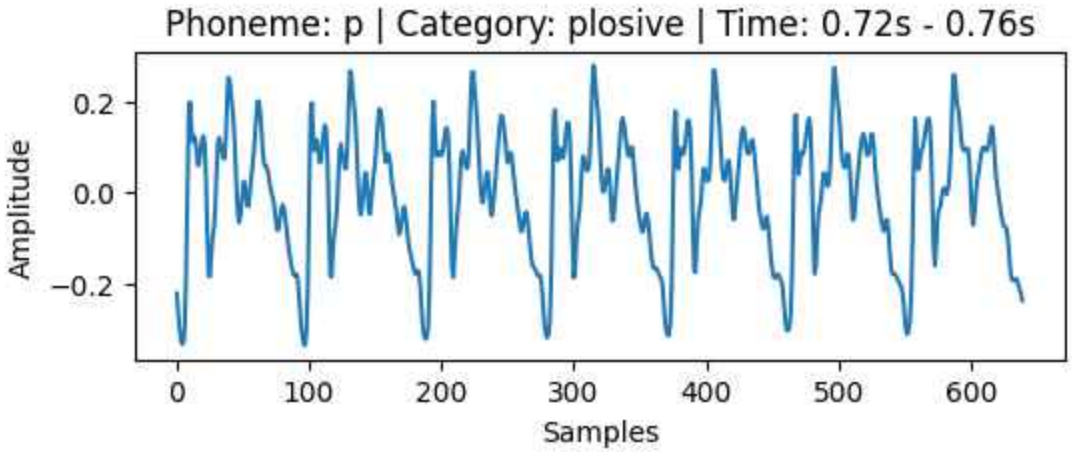
jpeg



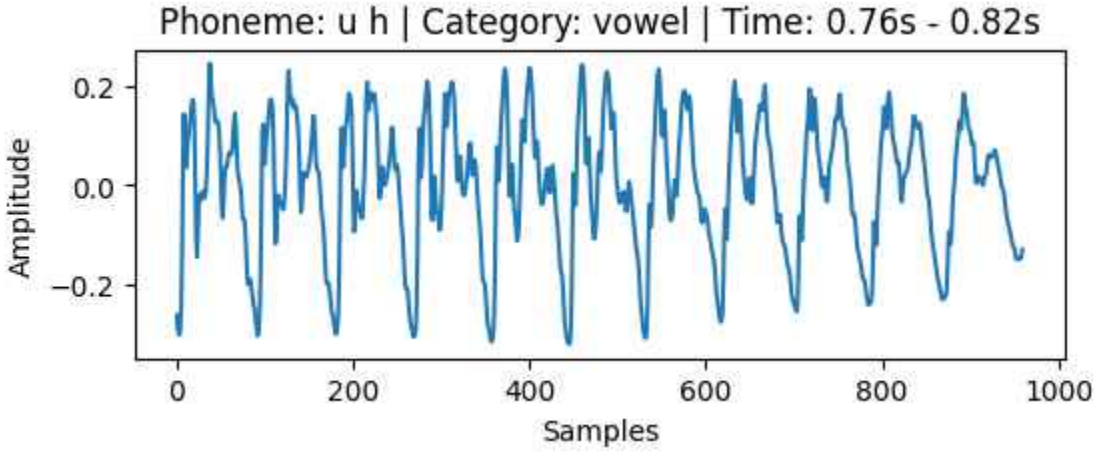
jpeg



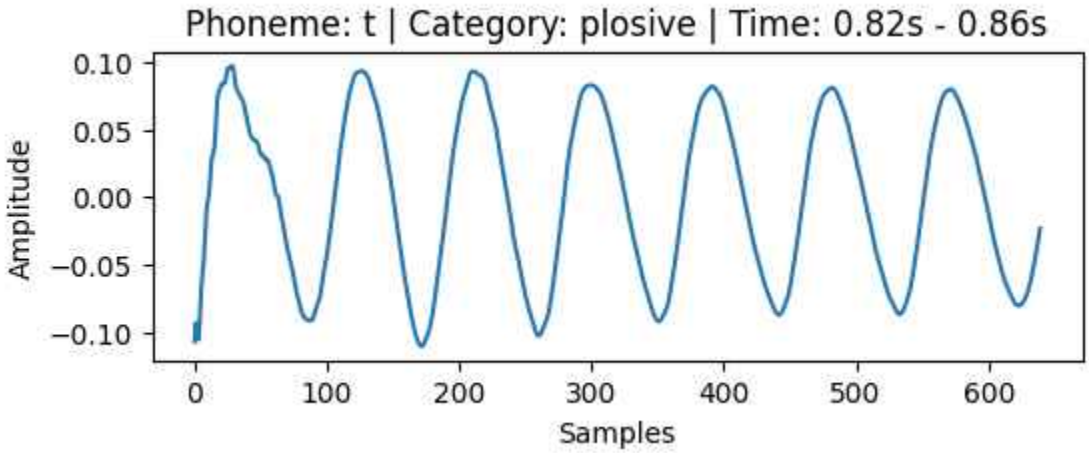
jpeg



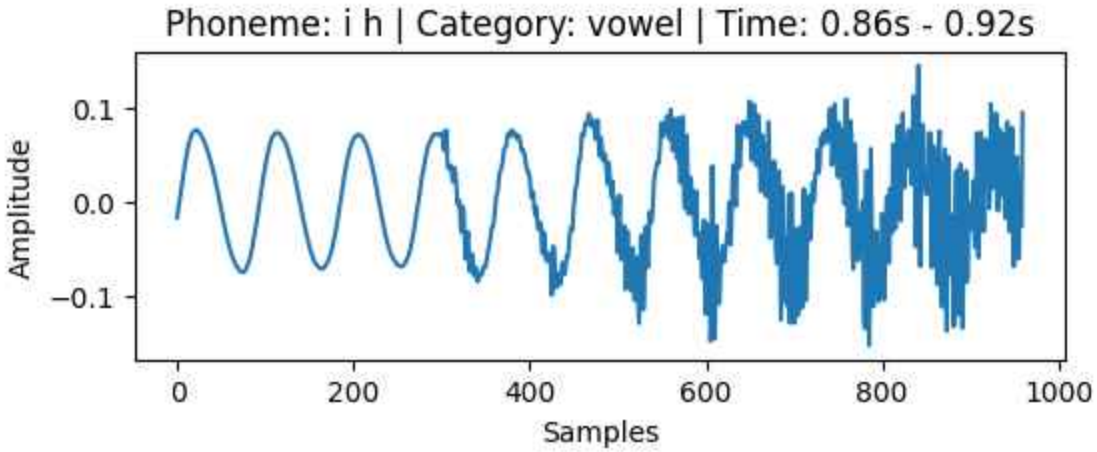
jpeg



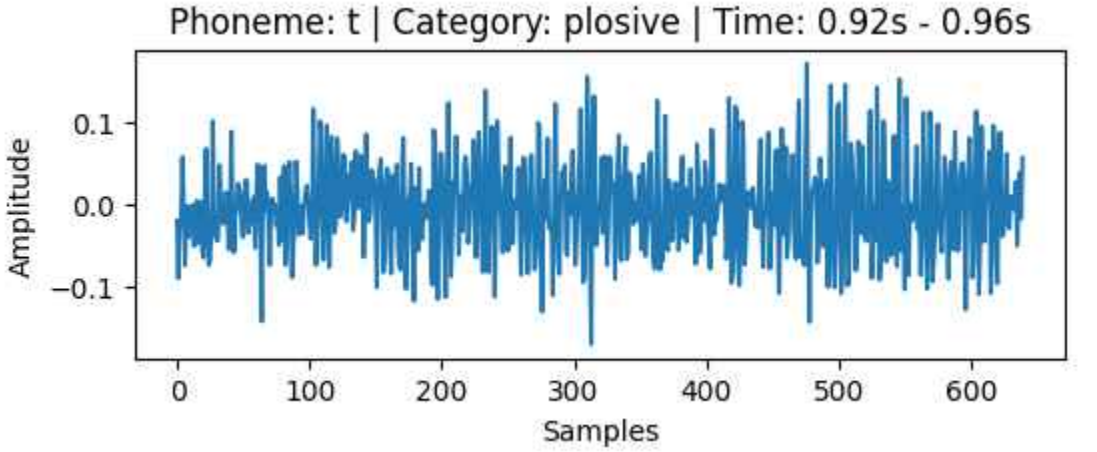
jpeg



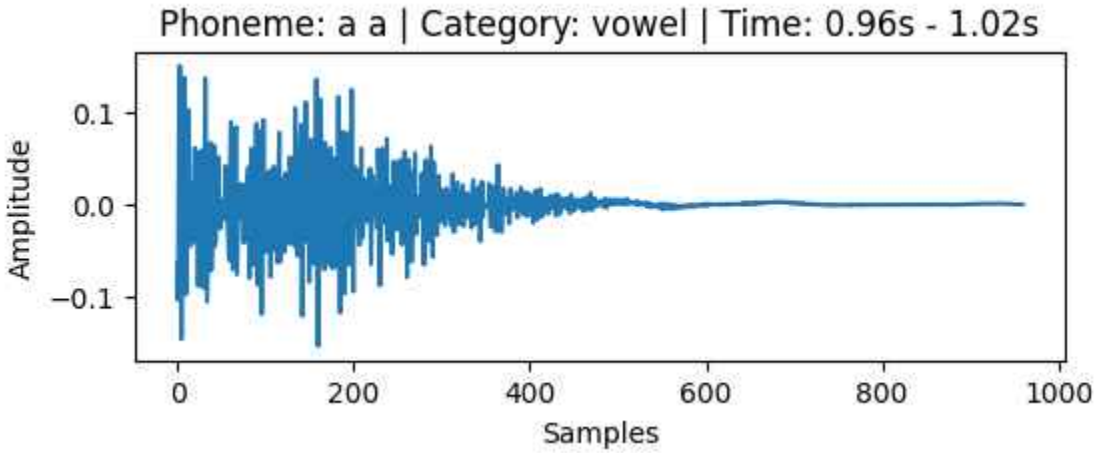
jpeg



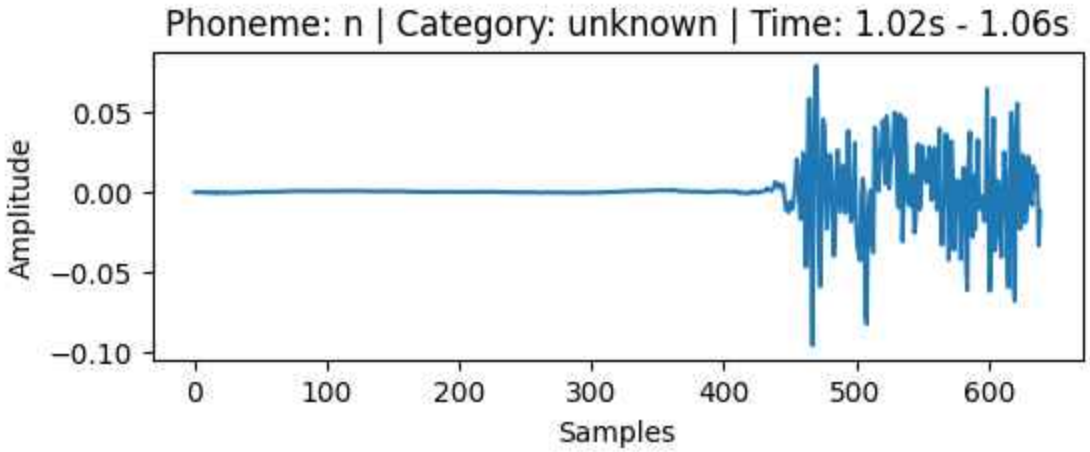
jpeg



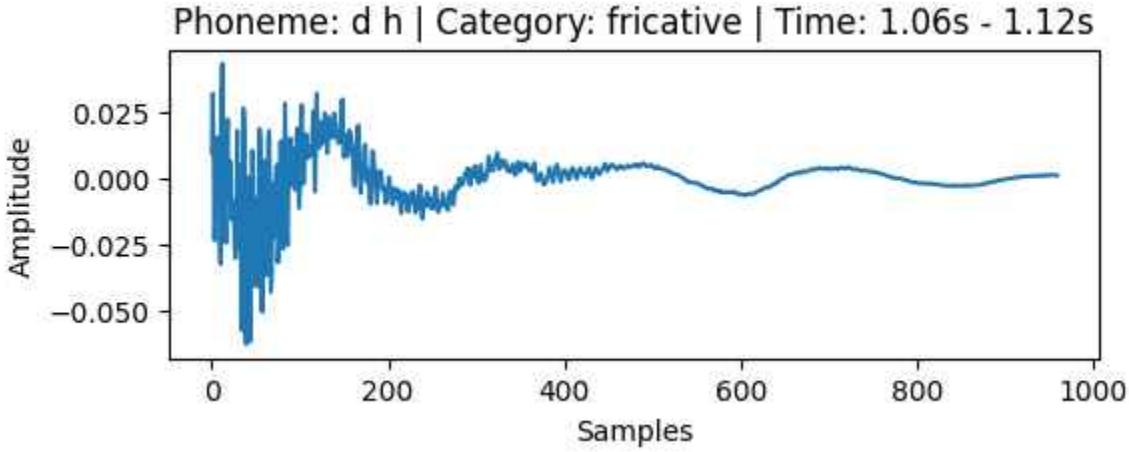
jpeg



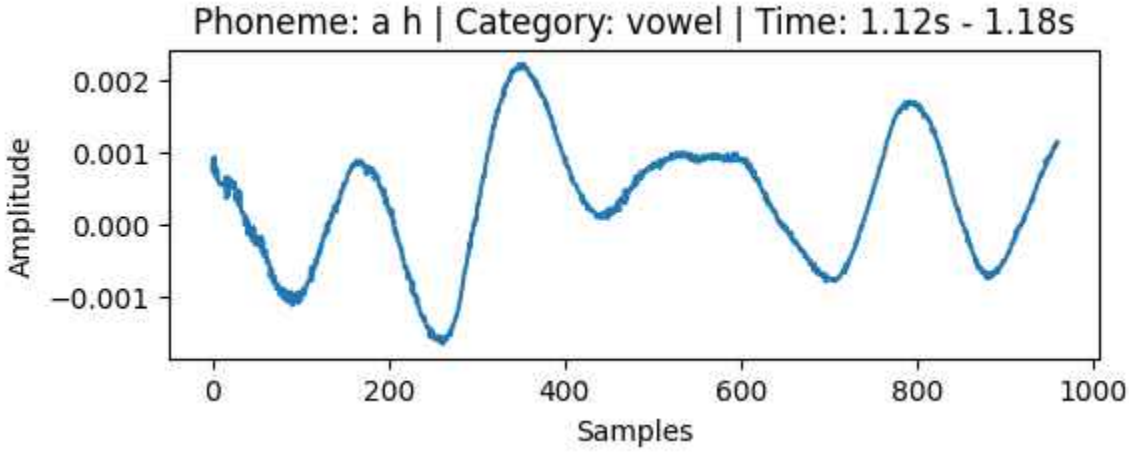
jpeg



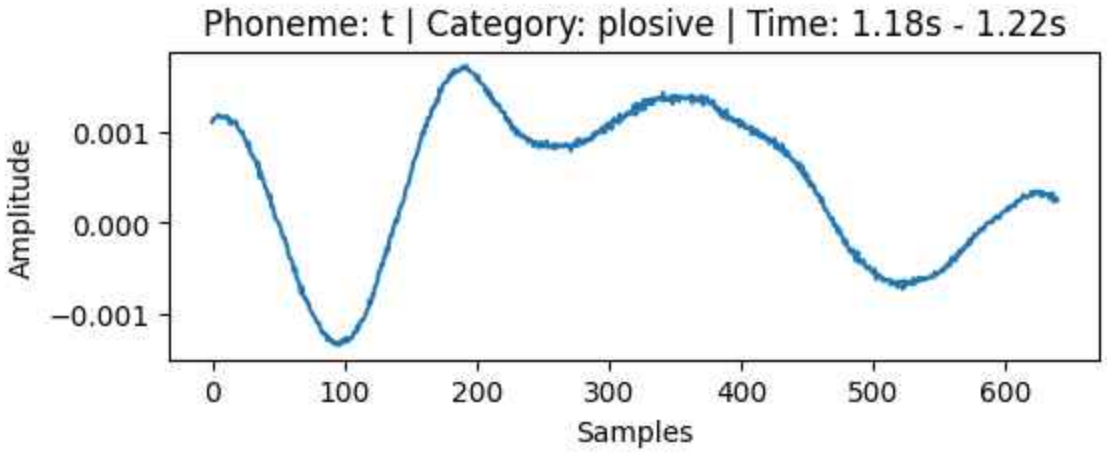
jpeg



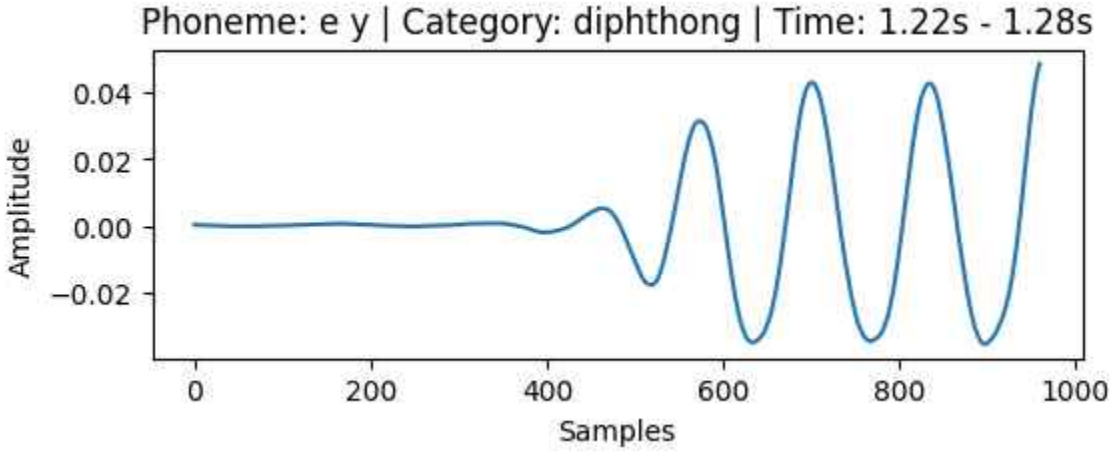
jpeg



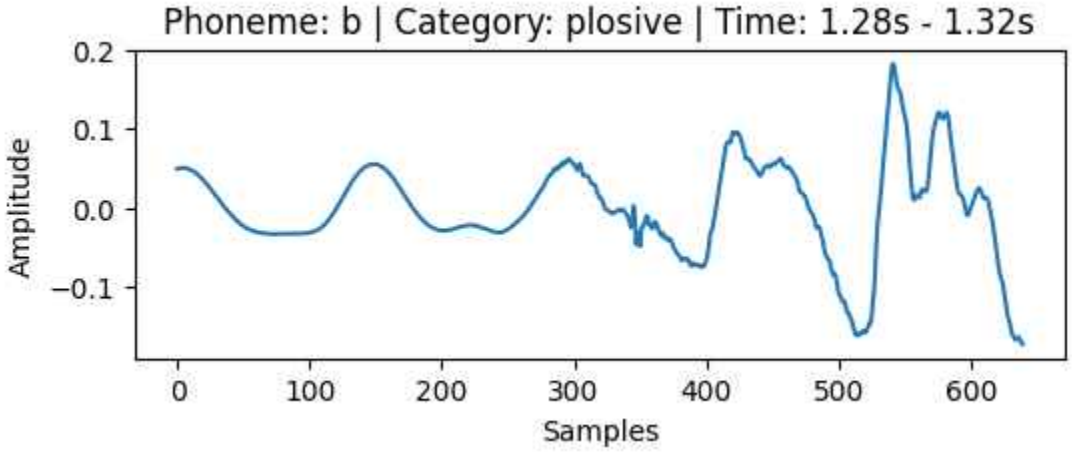
jpeg



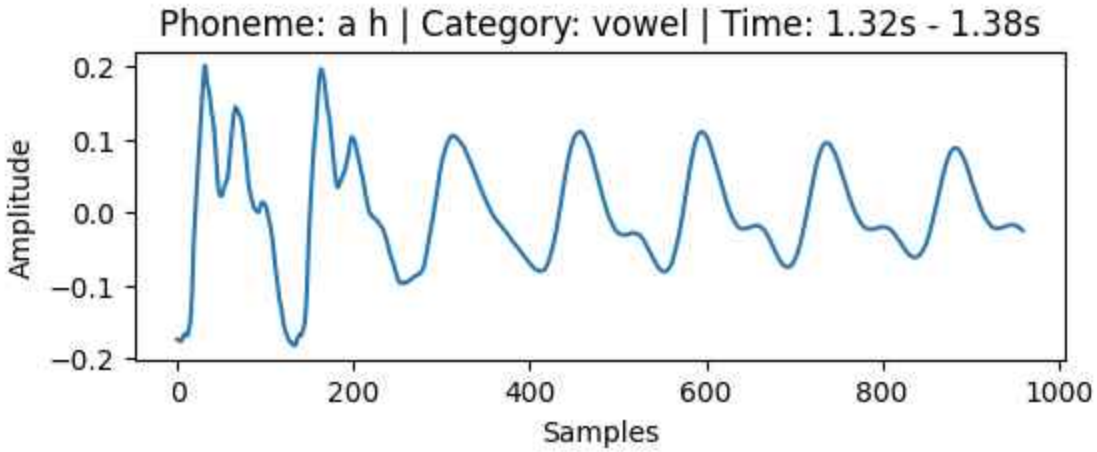
jpeg



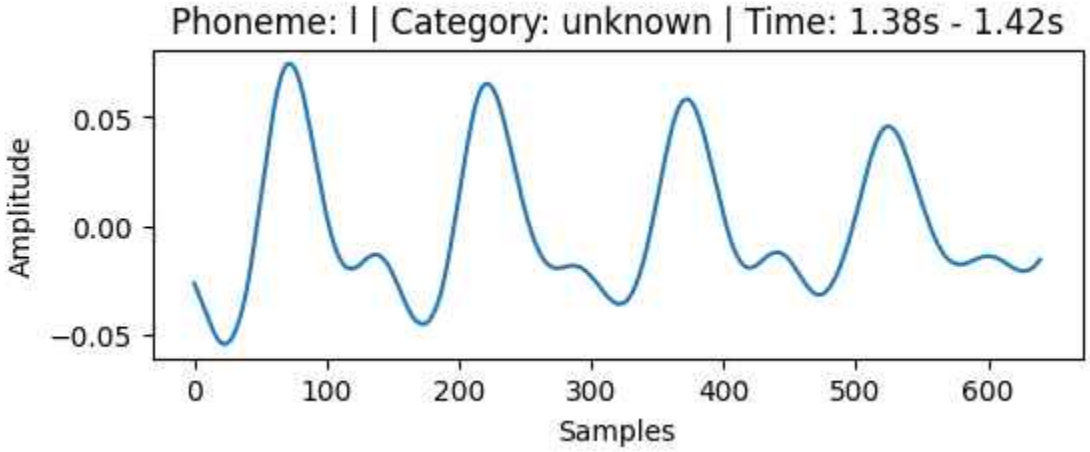
jpeg



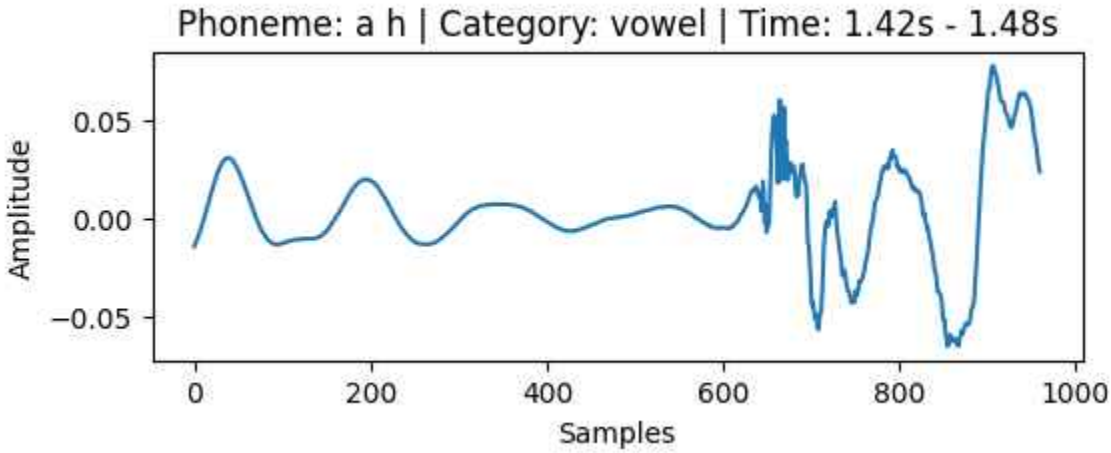
jpeg



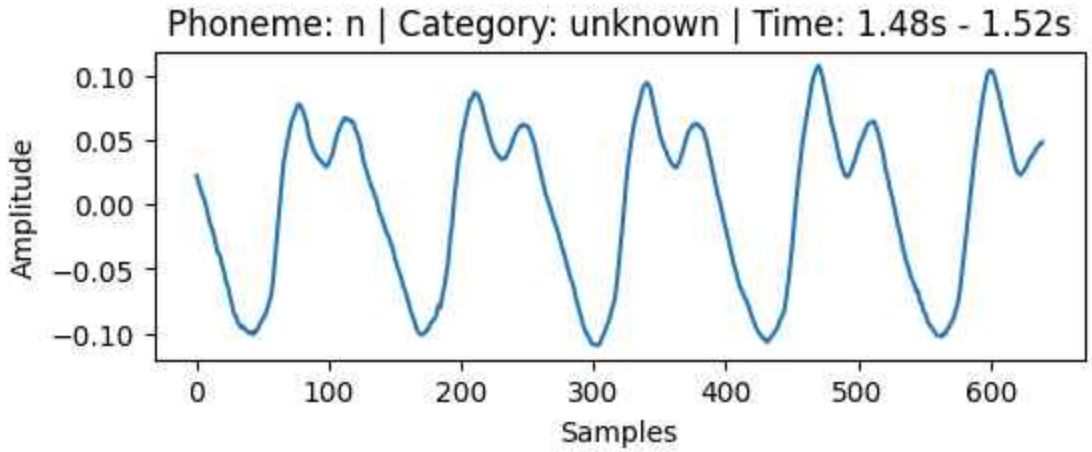
jpeg



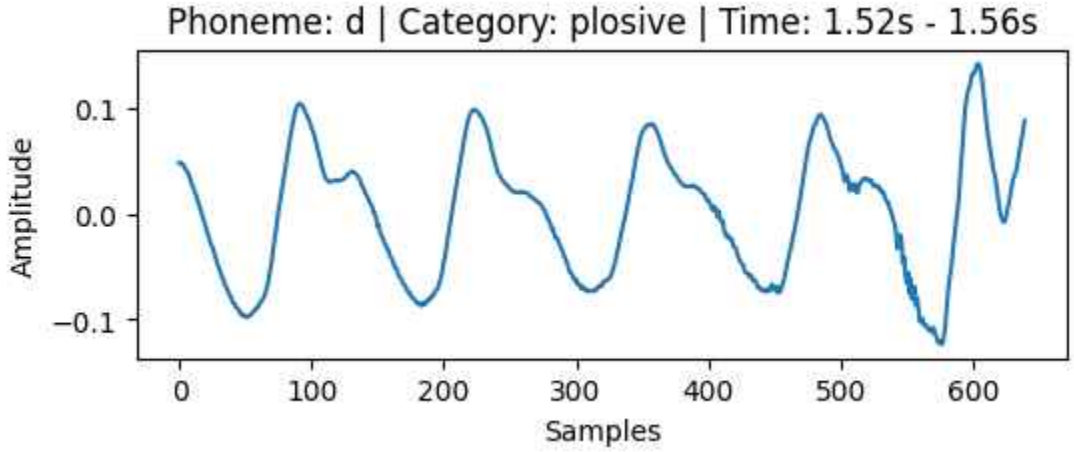
jpeg



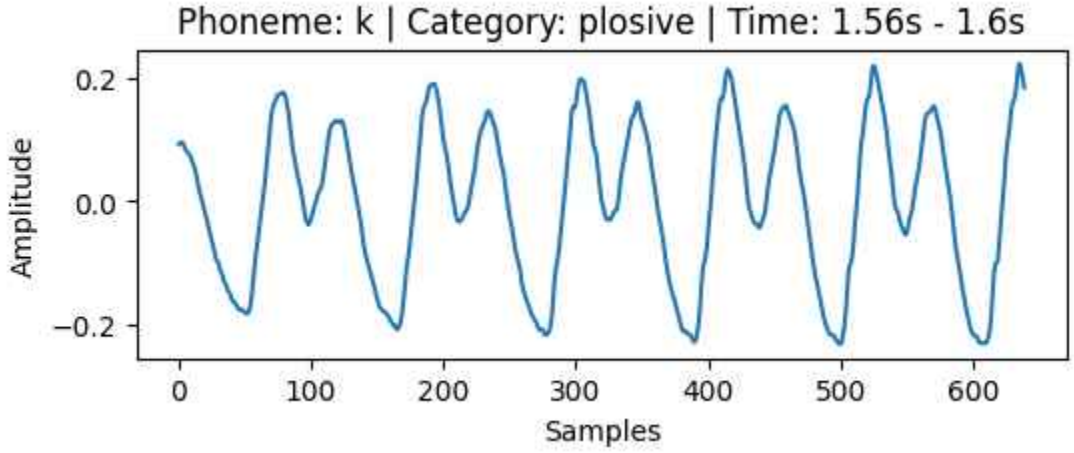
jpeg



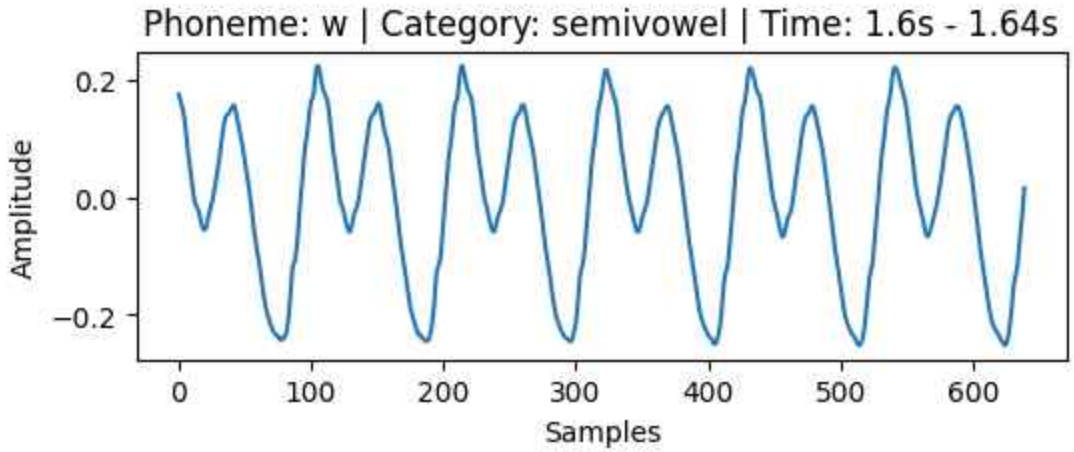
jpeg



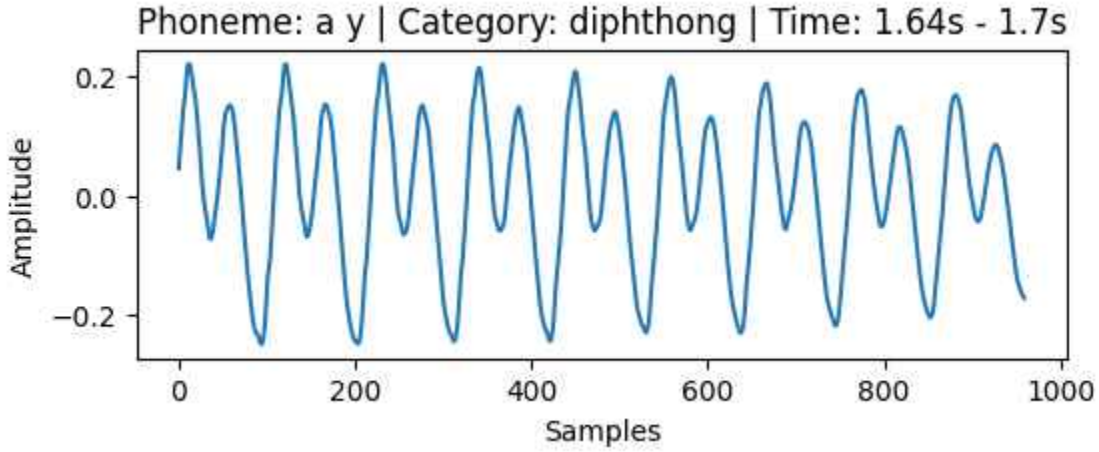
jpeg



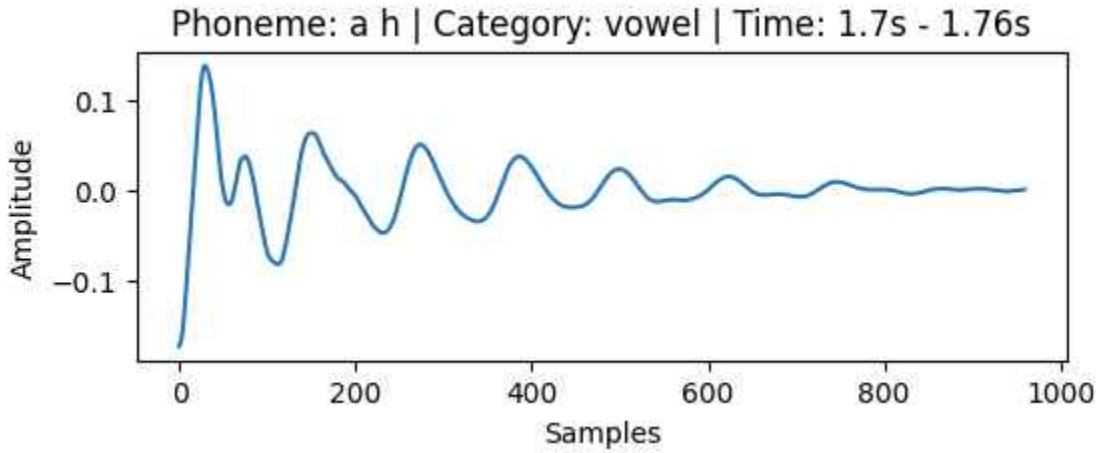
jpeg



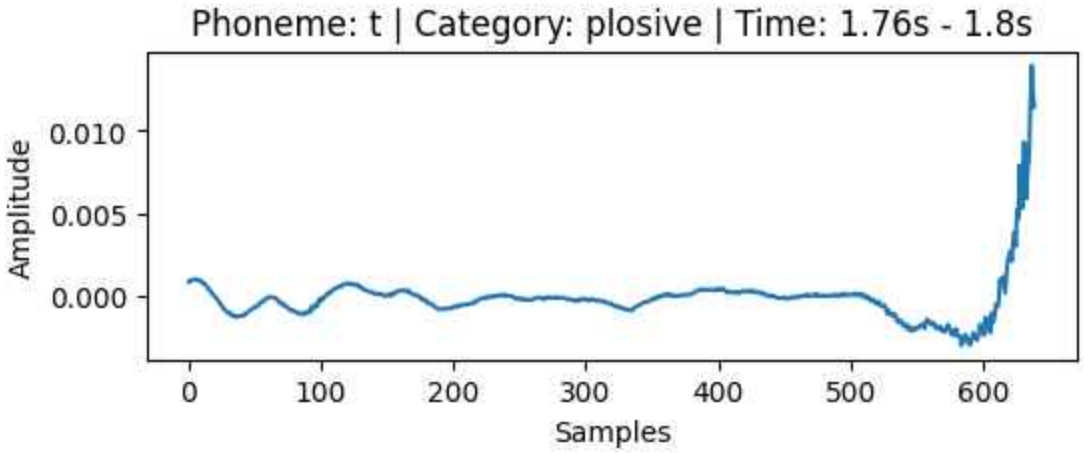
jpeg



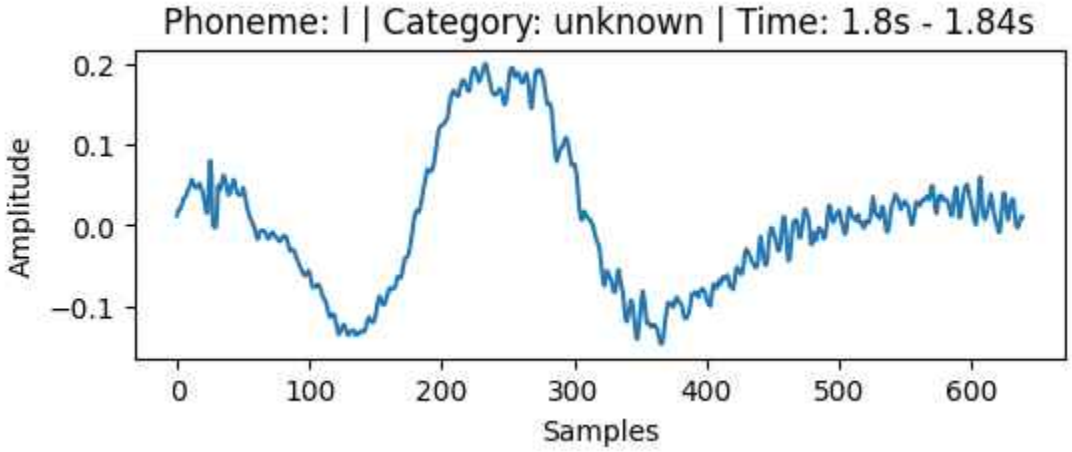
jpeg



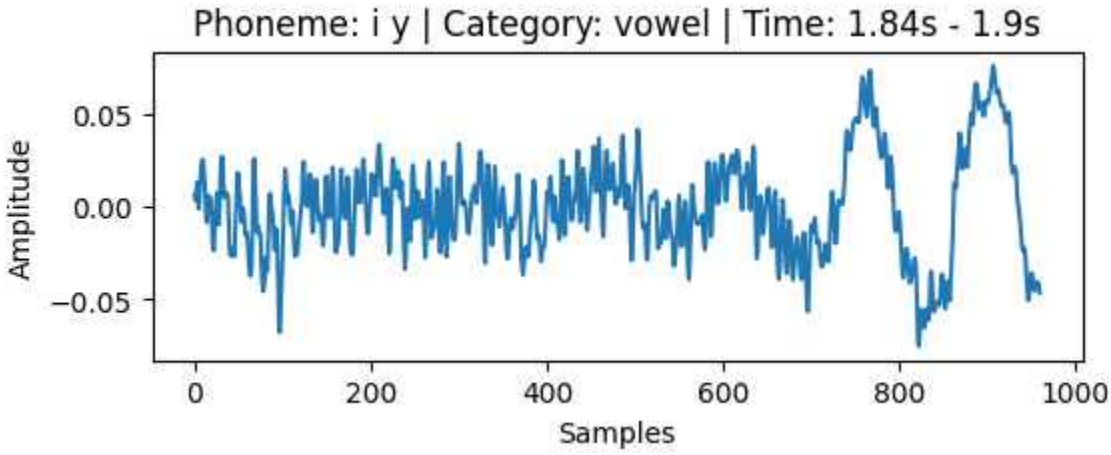
jpeg



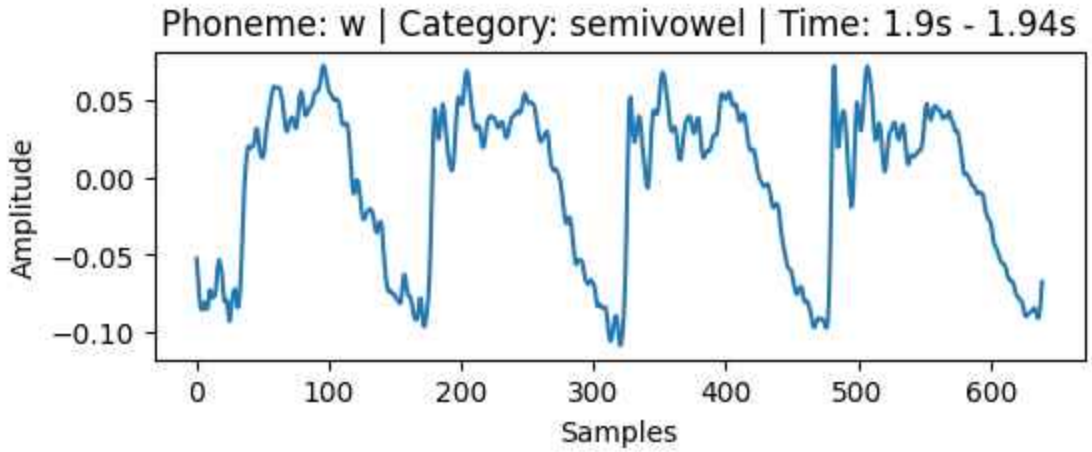
jpeg



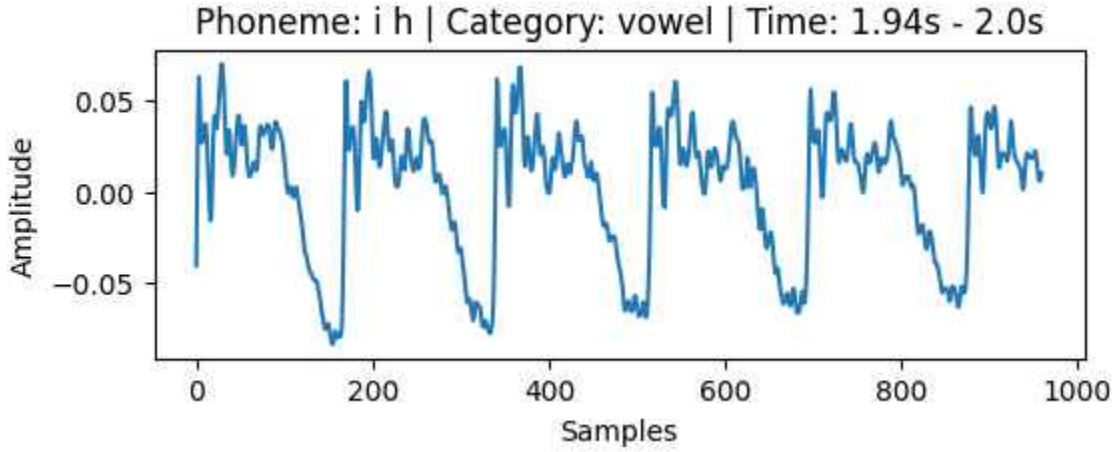
jpeg



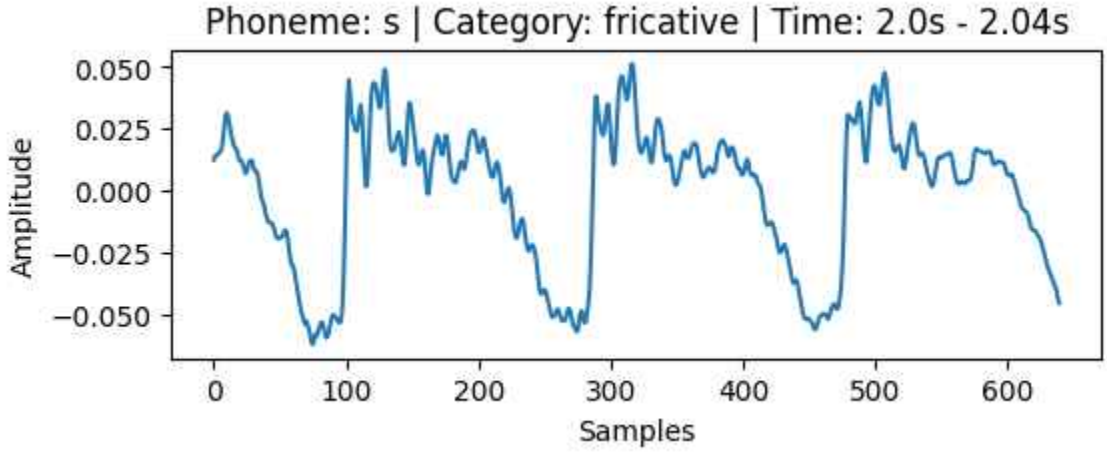
jpeg



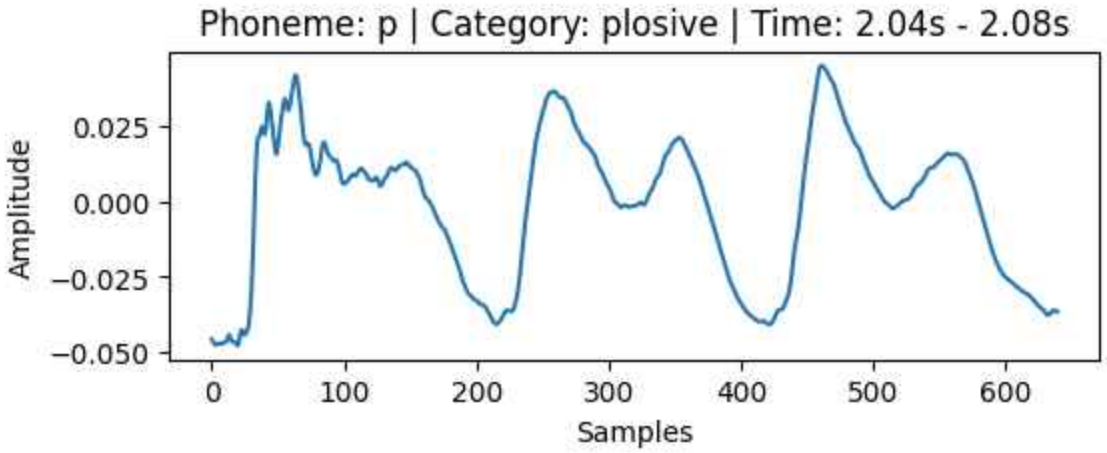
jpeg



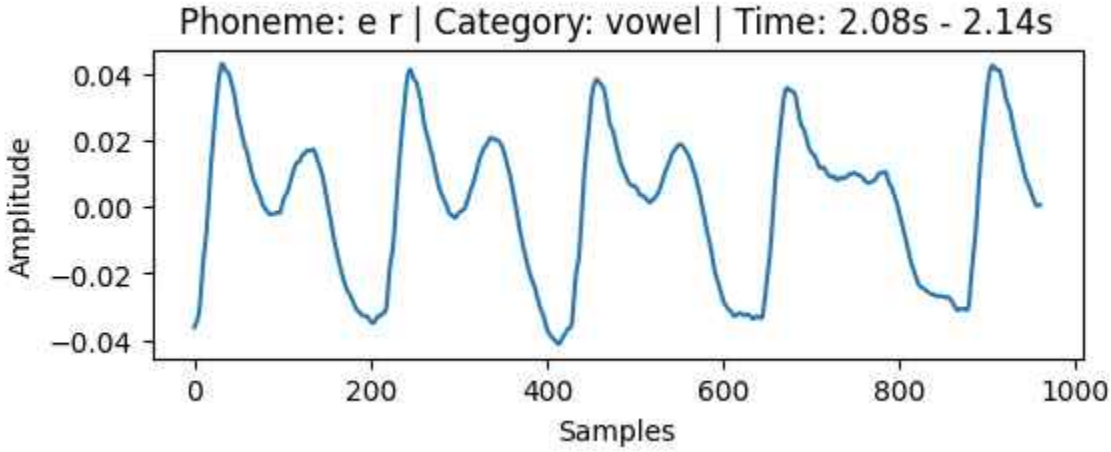
jpeg



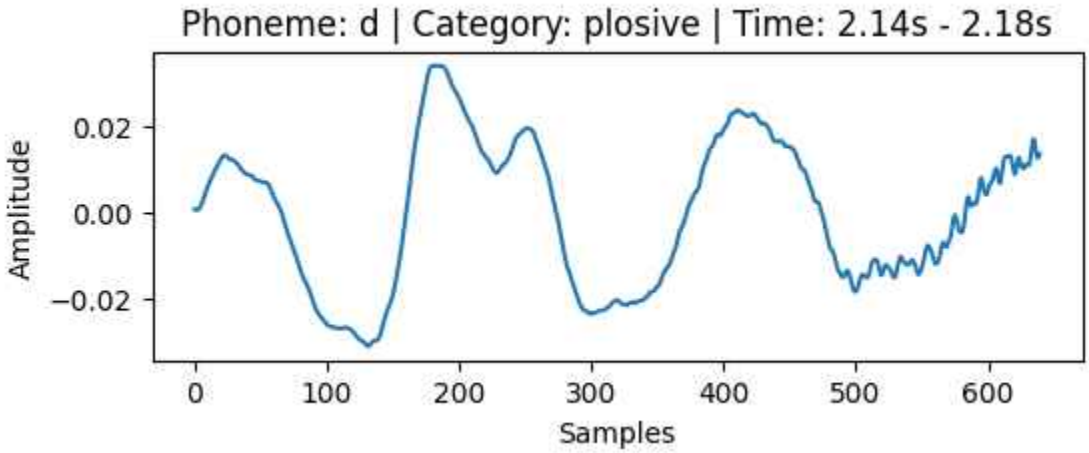
jpeg



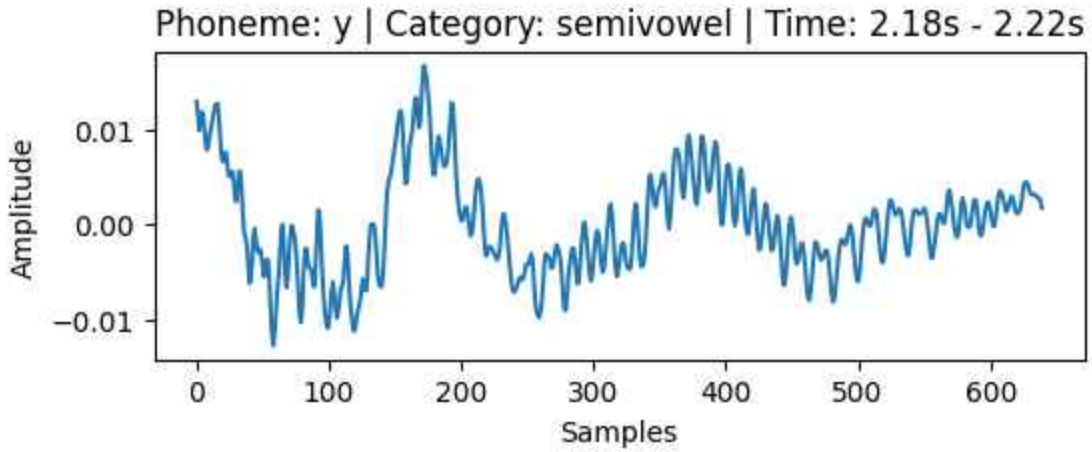
jpeg



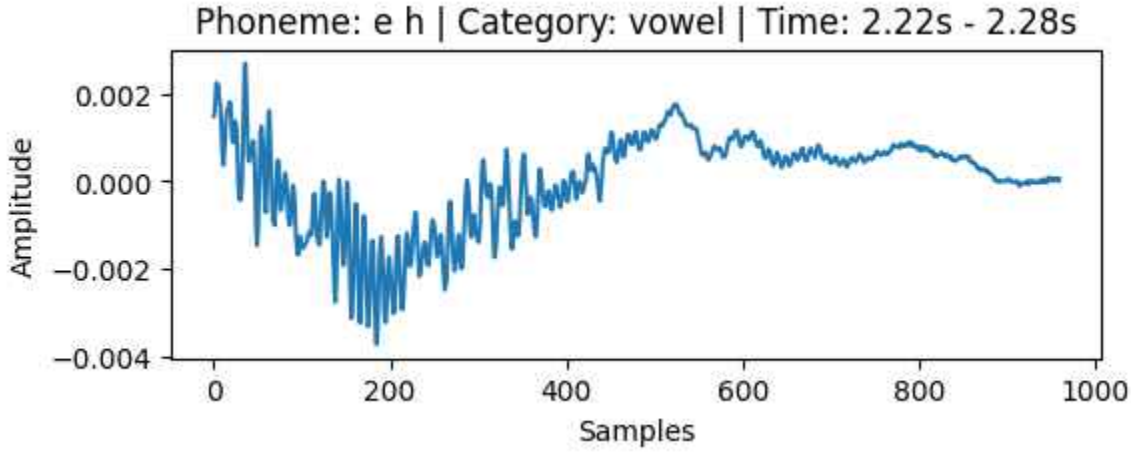
jpeg



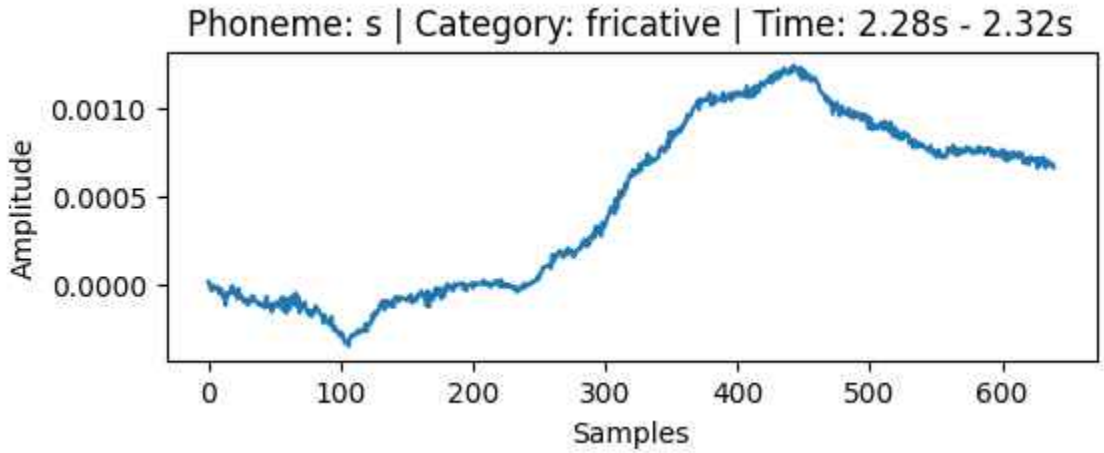
jpeg



jpeg



jpeg

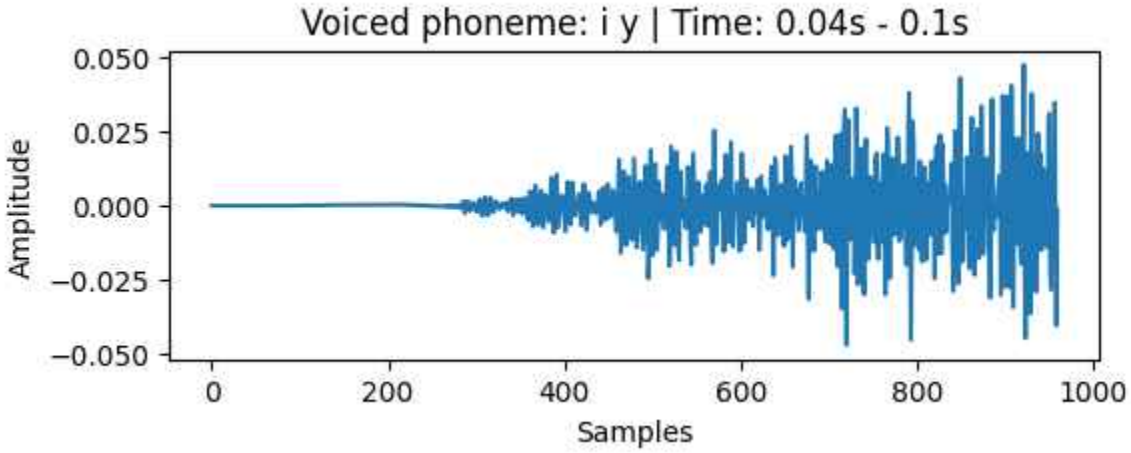


jpeg

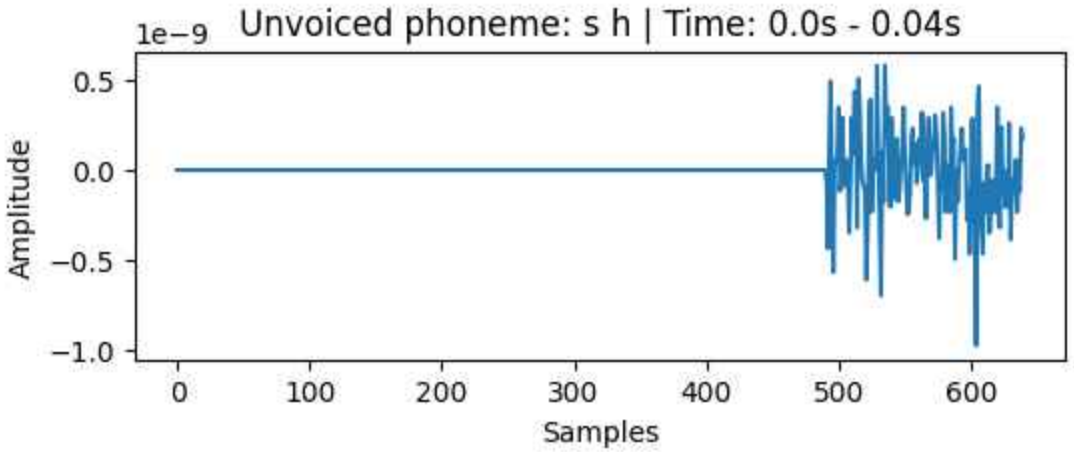
5

```
voiced_phonemes = ["a a","a e","a h","a o","e h","e r","i h","i y","u h","u w
unvoiced_phonemes = ["p","t","k","f","s","s h","t h","c h","h h"]
```

```
voiced_segment = None
unvoiced_segment = None

for phoneme, start, end in segments:
    if voiced_segment is None and phoneme in voiced_phonemes:
        voiced_segment = (phoneme, start, end)
    if unvoiced_segment is None and phoneme in unvoiced_phonemes:
        unvoiced_segment = (phoneme, start, end)
    if voiced_segment and unvoiced_segment:
        break

for label, segment in [("Voiced", voiced_segment), ("Unvoiced",
unvoiced_segment)]:
    if segment:
        phoneme, start, end = segment
        start_sample = int(start * sr)
        end_sample = int(end * sr)
        plt.figure(figsize=(6, 2))
        plt.plot(y[start_sample:end_sample])
        plt.title(f"{label} phoneme: {phoneme} | Time: {round(start,2)}s - {r
        plt.xlabel("Samples")
        plt.ylabel("Amplitude")
        plt.show()
```



jpeg



jpeg

objective 2

i.

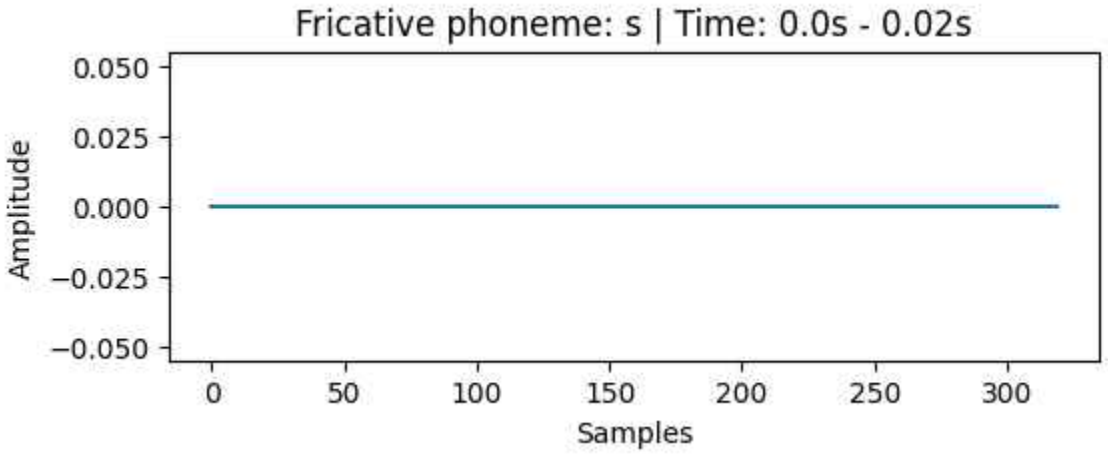
```
import soundfile as sf
import librosa
y, sr = sf.read("/content/speech (1).wav")
print("Original sample rate:", sr)
print("Original shape:", y.shape)
if y.ndim > 1:
  y = librosa.to_mono(y.T)
if sr != 16000:
  y = librosa.resample(y.astype(float), orig_sr=sr, target_sr=16000)
  sr = 16000
print("Resampled sample rate:", sr)
print("Number of samples:", len(y))
print("Duration (sec):", round(len(y)/sr, 3))

Original sample rate: 24000
Original shape: (36288,)
Resampled sample rate: 16000
Number of samples: 24192
Duration (sec): 1.512
```

ii.

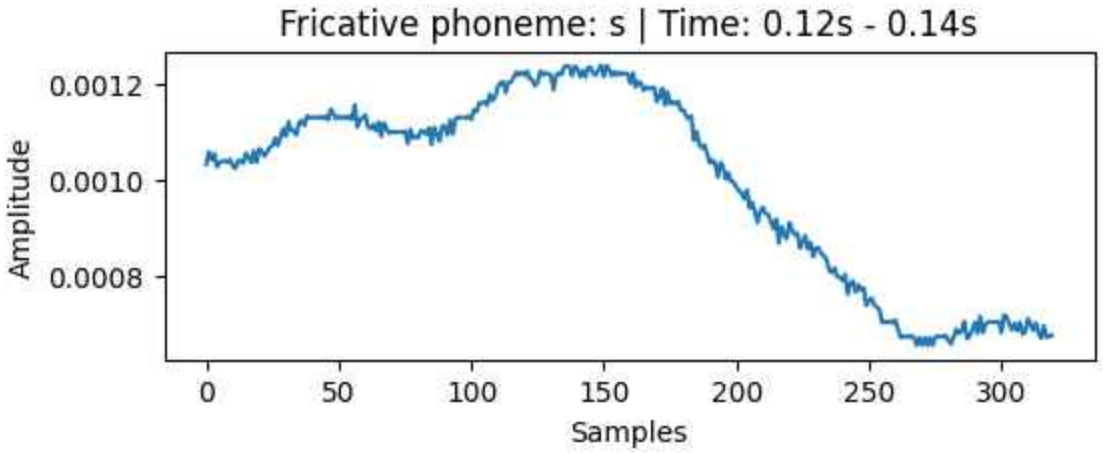
```
import soundfile as sf
import librosa
import torch
from transformers import Wav2Vec2Processor, Wav2Vec2ForCTC
import matplotlib.pyplot as plt

processor = Wav2Vec2Processor.from_pretrained("Bluecast/wav2vec2-Phoneme")
model = Wav2Vec2ForCTC.from_pretrained("Bluecast/wav2vec2-Phoneme")

input_values = processor(y, sampling_rate=sr,
```

```
return_tensors="pt").input_values
with torch.no_grad():
    logits = model(input_values).logits

predicted_ids = torch.argmax(logits, dim=-1)
phoneme_sequence = processor.batch_decode(predicted_ids)

print("Recognized phoneme sequence:")
print(phoneme_sequence[0])

frame_ids = torch.argmax(logits, dim=-1)[0].cpu().numpy()
frame_phonemes = processor.batch_decode([frame_ids])[0]

frame_duration = model.config.inputs_to_logits_ratio / sr
segments = []
current_phoneme = frame_phonemes[0]
start_time = 0.0

for i, phoneme in enumerate(frame_phonemes[1:], start=1):
    if phoneme != current_phoneme:
        end_time = i * frame_duration
        segments.append((current_phoneme, start_time, end_time))
        current_phoneme = phoneme
        start_time = end_time

segments.append((current_phoneme, start_time, len(frame_phonemes) *
frame_duration))

print("Number of phoneme segments:", len(segments))

Loading weights:   0%|          | 0/424 [00:00<?, ?it/s]


Recognized phoneme sequence:
sh iy s iy z y uw
Number of phoneme segments: 17

phoneme_categories = {
    "vowel": ["a a", "a e", "a h", "a o", "e h", "e r", "i h", "i y","u h", "
    "plosive": ["p", "b", "t", "d", "k", "g"],
    "fricative": ["f", "v", "t h", "d h", "s", "z", "s h", "z h", "h h"],
    "affricate": ["c h", "j h"],
    "semivowel": ["l", "r", "w", "y"],
    "diphthong": ["a y", "e y", "o y", "a w", "o w"],
    "whisper": ["s i l", "s p n"]
}

def get_category(phoneme):
    for category, phones in phoneme_categories.items():
```

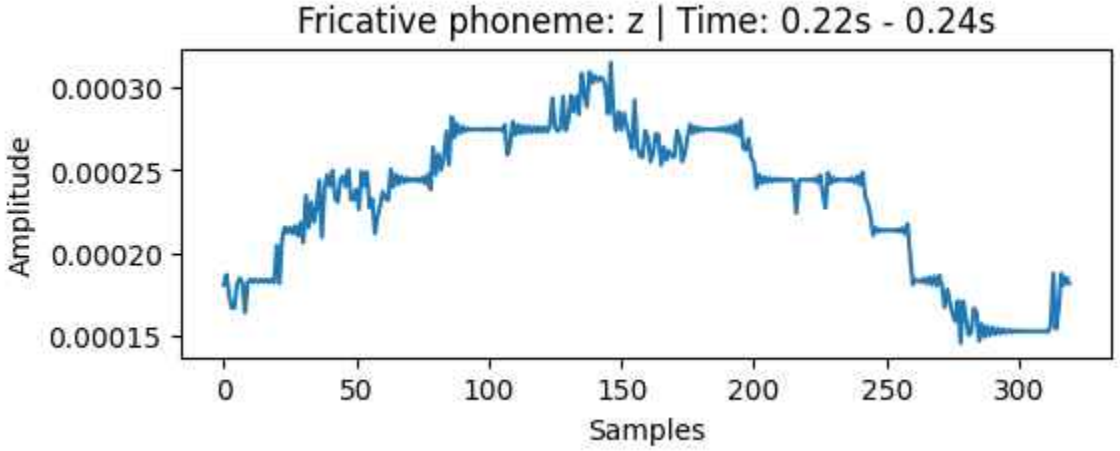
```
        if phoneme in phones:
            return category
    return "unknown"

fricative_segments = []
approximant_segments = []

for phoneme, start, end in segments:
    category = get_category(phoneme)
    if category == "fricative":
        fricative_segments.append((phoneme, start, end))
    elif category == "semivowel":
        approximant_segments.append((phoneme, start, end))

for phoneme, start, end in fricative_segments:
    start_sample = int(start * sr)
    end_sample = int(end * sr)
    plt.figure(figsize=(6, 2))
    plt.plot(y[start_sample:end_sample])
    plt.title(f"Fricative phoneme: {phoneme} | Time: {round(start,2)}s - {rou
    plt.xlabel("Samples")
    plt.ylabel("Amplitude")
    plt.show()

for phoneme, start, end in approximant_segments:
  start_sample = int(start * sr)
  end_sample = int(end * sr)
  plt.figure(figsize=(6, 2))
  plt.plot(y[start_sample:end_sample])
  plt.title(f"Approximant phoneme: {phoneme} | Time: {round(start,2)}s - {rou
  plt.xlabel("Samples")
  plt.ylabel("Amplitude")
  plt.show()
```



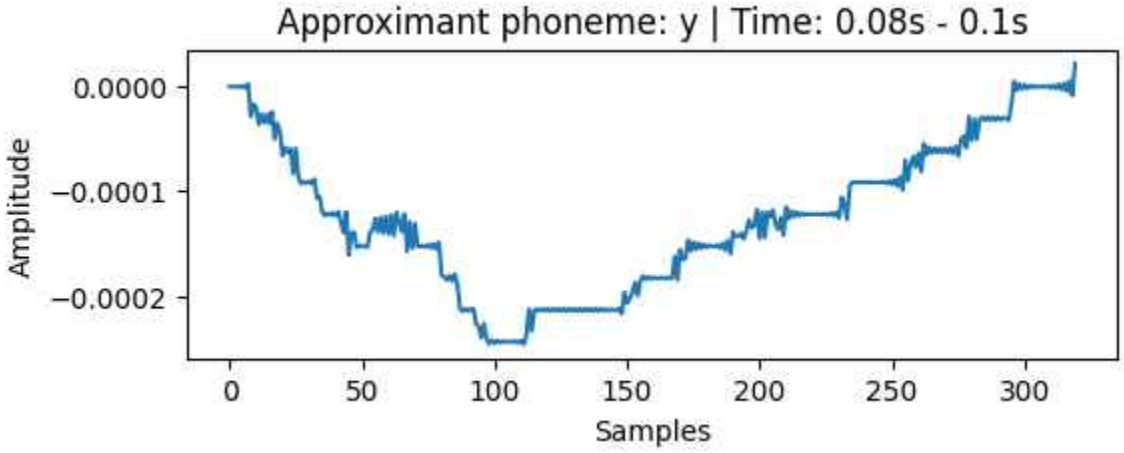
jpeg



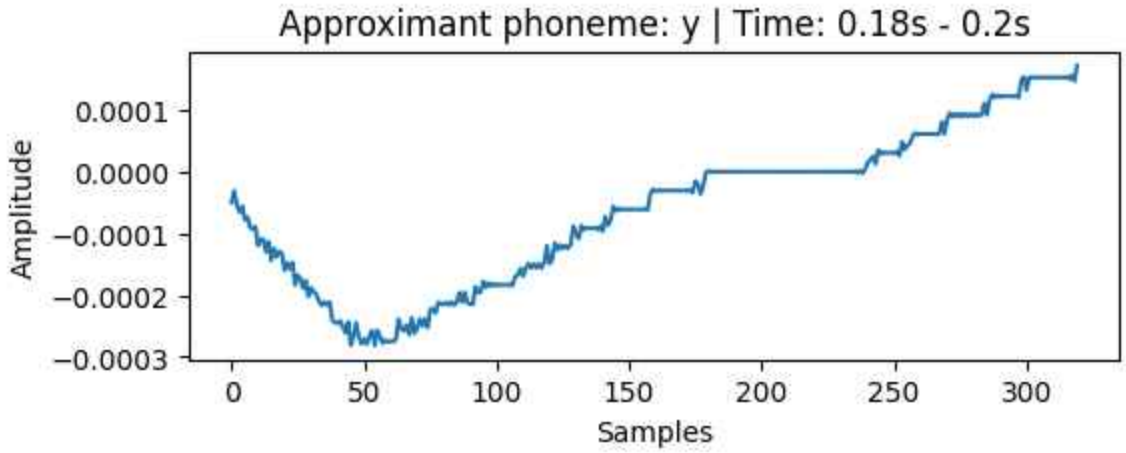
jpeg



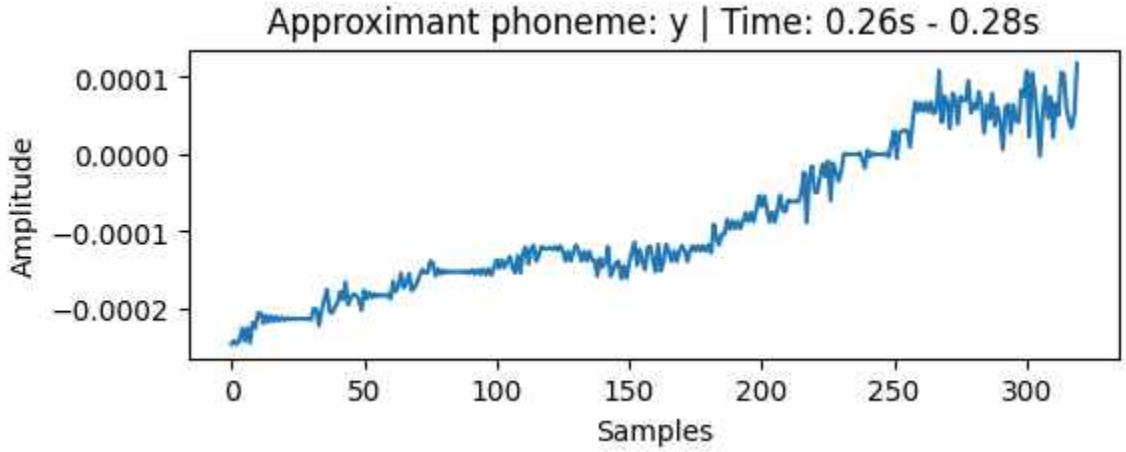
jpeg



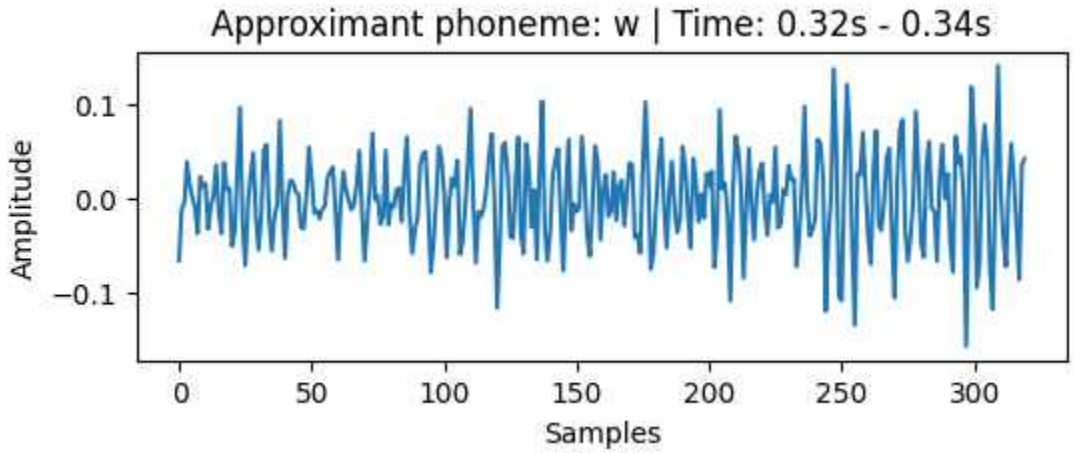
jpeg



jpeg



jpeg



jpeg

iii.

```
fricatives = ['s', 'z', 'sh']
approximants = ['y']

fricative_segments = []
```

```
approximant_segments = []

for phoneme, start, end in segments:
  if phoneme in fricatives:
    fricative_segments.append((phoneme, start, end))
  elif phoneme in approximants:
    approximant_segments.append((phoneme, start, end))
print("Fricative Segments:",fricative_segments)
print("Approximant Segments:",approximant_segments)

Fricative Segments: [('s', 0.0, 0.02), ('s', 0.12, 0.14), ('z', 0.22, 0.24)]
Approximant Segments: [('y', 0.08, 0.1), ('y', 0.18, 0.2), ('y', 0.26, 0.28)]
```

iv.

```
def extract_segment(signal, sr, start, end):
  return signal[int(start * sr): int(end * sr)]
fricative_signals = [(ph, extract_segment(y, sr, st, et)) for ph, st, et in f
approximant_signals = [(ph, extract_segment(y, sr, st, et)) for ph, st, et in

print("fractive segments",fricative_signals)
print("pproximant segments",approximant_signals)

fractive segments [('s', array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])), ('s', array
       0.00103518, 0.00103794, 0.00103863, 0.00103524, 0.00104182,
       0.00102777, 0.00102274, 0.00103919, 0.00103875, 0.00103375,
       0.00105458, 0.00104555, 0.00103481, 0.00105962, 0.00103567,
       0.00106406, 0.00105772, 0.00104856, 0.0010574 , 0.00106736,
       0.00107022, 0.0010859 , 0.00107059, 0.00108881, 0.00110596,
       0.00108984, 0.00112019, 0.00110185, 0.00110136, 0.00109252,
       0.00111855, 0.00112239, 0.00111247, 0.00113242, 0.00112816,
       0.00112907, 0.0011299 , 0.00112789, 0.00113087, 0.00112692,
```

0.00113216, 0.00112436, 0.00114649, 0.00113718, 0.00112597,
0.00113133, 0.00112721, 0.0011312 , 0.00112675, 0.00113217,
0.00112508, 0.00115582, 0.00110962, 0.00112424, 0.00112779,
0.00113425, 0.00110983, 0.00110538, 0.00111416, 0.00110129,
0.00111959, 0.00108957, 0.00111791, 0.00110566, 0.00109609,
0.00110016, 0.00109764, 0.00109913, 0.00109871, 0.00109782,
0.00110045, 0.00107502, 0.00110437, 0.00108607, 0.00108902,
0.0010872 , 0.001102 , 0.00109927, 0.00109439, 0.00110767,
0.00107297, 0.00110824, 0.00109333, 0.00107983, 0.00111103,
0.00111052, 0.00109249, 0.00112343, 0.00109467, 0.0011256 ,
0.00112988, 0.0011285 , 0.00113012, 0.00112768, 0.00113142,
0.00112511, 0.00114539, 0.00114135, 0.00115957, 0.0011586 ,
0.00116204, 0.00115489, 0.00117796, 0.00116468, 0.00117232,
0.00119316, 0.0012017 , 0.00120349, 0.00118124, 0.00120263,
0.00120337, 0.00121267, 0.00122414, 0.00121794, 0.00122363,
0.00121706, 0.00122597, 0.00120973, 0.0012078 , 0.00119547,
0.00120228, 0.00122564, 0.00121812, 0.00122206, 0.00122051,
0.00121722, 0.00118668, 0.00121651, 0.00122217, 0.00121843,
0.00123428, 0.00123727, 0.00123489, 0.00121637, 0.00123608,
0.00123139, 0.00121433, 0.00122674, 0.00121365, 0.00123282,
0.00123363, 0.00122084, 0.0012177 , 0.00122912, 0.00123783,
0.00121464, 0.00123744, 0.00122998, 0.0012161 , 0.0012242 ,
0.00121776, 0.00122321, 0.00121863, 0.00122227, 0.00121979,
0.00120032, 0.00122247, 0.00119293, 0.00120396, 0.00120054,
0.00118562, 0.00119279, 0.00118892, 0.00119012, 0.00119218,
0.00116208, 0.00117376, 0.00119017, 0.00115508, 0.00118301,
0.00115742, 0.00116159, 0.00115765, 0.00116284, 0.00114489,
0.0011454 , 0.00113203, 0.00112643, 0.00113239, 0.00108168,
0.00109917, 0.00107392, 0.00106545, 0.00107153, 0.0010531 ,
0.00103378, 0.00104039, 0.00103229, 0.00100434, 0.00103446,
0.00102583, 0.00100019, 0.00101369, 0.00099569, 0.00099384,
0.00098347, 0.00097617, 0.00097038, 0.00095874, 0.00097797,
0.0009398 , 0.00095166, 0.00093033, 0.00090984, 0.00093357,
0.00094131, 0.00092805, 0.00092647, 0.00091098, 0.00089694,
0.00091671, 0.00086672, 0.00090151, 0.00088602, 0.00087623,
0.00090917, 0.00089464, 0.00088427, 0.00088733, 0.00085615,
0.0008701 , 0.00088237, 0.00085443, 0.0008675 , 0.00083832,
0.00085634, 0.00085672, 0.0008455 , 0.00083979, 0.00083285,
0.00081012, 0.00080502, 0.00081481, 0.00079669, 0.00079614,
0.00078838, 0.00080096, 0.00075955, 0.00078305, 0.00078721,
0.00076483, 0.0007794 , 0.00076849, 0.00077344, 0.00073654,
0.00074607, 0.00075205, 0.00073781, 0.00073258, 0.00072778,
0.00070009, 0.00070316, 0.00070058, 0.00070363, 0.00069939,
0.00070621, 0.00068537, 0.00066964, 0.0006713 , 0.00067233,
0.0006699 , 0.00067327, 0.00066921, 0.00065342, 0.00066888,
0.00065359, 0.00066887, 0.00065344, 0.0006692 , 0.00065293,
0.00066991, 0.00067231, 0.00067122, 0.00067045, 0.00067425,
0.00066258, 0.00065628, 0.0006713 , 0.00068704, 0.00067799,

0.00068337, 0.0007028 , 0.00066712, 0.0006718 , 0.00068499,
0.00070153, 0.00067836, 0.00071367, 0.00067993, 0.00069748,
0.00070202, 0.00070249, 0.00070126, 0.0007023 , 0.00070215,
0.00067954, 0.00071654, 0.00071247, 0.00069649, 0.00068552,
0.00069885, 0.00068366, 0.00070136, 0.00067676, 0.00068178,
0.00070904, 0.00069703, 0.00068507, 0.00069989, 0.00067932,
0.00066772, 0.00069388, 0.0006705 , 0.0006708 , 0.00067425])), ('z', a
0.00018093, 0.00018462, 0.00018191, 0.00016378, 0.00018218,
0.00018398, 0.00018225, 0.00018395, 0.00018226, 0.00018396,
0.00018225, 0.00018397, 0.00018223, 0.000184 , 0.00018216,
0.00020455, 0.00018148, 0.0002084 , 0.00021557, 0.00021206,
0.00021529, 0.00021162, 0.00021616, 0.00021029, 0.00021824,
0.00020642, 0.00023467, 0.00021534, 0.0002306 , 0.00021851,
0.00022726, 0.00024378, 0.00020949, 0.00023928, 0.0002471 ,
0.00024033, 0.00024974, 0.00023292, 0.00023071, 0.00024433,
0.00024647, 0.00024013, 0.00025024, 0.00023215, 0.00023227,
0.00023839, 0.00022624, 0.00024915, 0.00024048, 0.00024878,
0.00022749, 0.00023267, 0.00021179, 0.00022179, 0.00022886,
0.00023702, 0.00023298, 0.00023179, 0.00025047, 0.00023975,
0.00024742, 0.00024162, 0.00024609, 0.00024262, 0.00024533,
0.00024317, 0.00024498, 0.00024334, 0.00024503, 0.00024303,
0.00024565, 0.00024197, 0.00024747, 0.00023831, 0.00026375,
0.00024731, 0.00025972, 0.00025023, 0.0002571 , 0.00027341,
0.00025324, 0.00028215, 0.00026984, 0.00027809, 0.00027216,
0.00027646, 0.00027341, 0.00027546, 0.00027423, 0.00027478,
0.00027479, 0.00027434, 0.00027513, 0.00027409, 0.00027527,
0.00027404, 0.00027521, 0.00027423, 0.00027486, 0.00027484,
0.00027374, 0.0002774 , 0.00025926, 0.00026486, 0.00027937,
0.00027124, 0.00027746, 0.00027222, 0.00027686, 0.00027263,
0.00027655, 0.00027286, 0.00027638, 0.000273 , 0.00027627,
0.00027308, 0.00027619, 0.00027316, 0.00027611, 0.00029362,
0.00027595, 0.0002735 , 0.00027558, 0.00029452, 0.00027408,
0.00027996, 0.00029522, 0.00028543, 0.00029457, 0.00028379,
0.00030848, 0.00029664, 0.00028846, 0.00030936, 0.00030271,
0.00030698, 0.00030378, 0.00030609, 0.00030531, 0.00030035,
0.00028415, 0.00031508, 0.00028539, 0.00027406, 0.00028533,
0.00027442, 0.0002845 , 0.00027658, 0.00027061, 0.00026311,
0.00029238, 0.0002643 , 0.00025839, 0.00025794, 0.00027394,
0.00026951, 0.00025526, 0.00026023, 0.00027203, 0.00027136,
0.00025306, 0.0002639 , 0.0002561 , 0.0002616 , 0.00025851,
0.00025794, 0.00027402, 0.00026941, 0.00025524, 0.00026066,
0.00027051, 0.00027773, 0.00027229, 0.00027649, 0.00027327,
0.00027567, 0.00027397, 0.00027504, 0.00027455, 0.00027451,
0.00027505, 0.00027403, 0.00027553, 0.00027354, 0.00027604,
0.00027299, 0.00027667, 0.00027223, 0.00027767, 0.00027076,
0.00028026, 0.00026331, 0.00026222, 0.00026913, 0.00025746,
0.00025463, 0.00023937, 0.00024725, 0.00024186, 0.00024592,
0.0002427 , 0.00024533, 0.00024313, 0.00024502, 0.00024337,

```
        0.00024482, 0.00024353, 0.00024469, 0.00024365, 0.00024457,
        0.00024377, 0.0002241 , 0.00024391, 0.00024427, 0.00024412,
        0.00024401, 0.00024446, 0.00024355, 0.00024512, 0.00024251,
        0.00024712, 0.00023602, 0.00022714, 0.0002484 , 0.00024182,
        0.00024559, 0.00024324, 0.00024462, 0.00024402, 0.00024393,
        0.00024468, 0.00024327, 0.00024538, 0.00024248, 0.00024635,
        0.00024112, 0.00024869, 0.00023415, 0.00022967, 0.00022206,
        0.00021056, 0.00021525, 0.00021267, 0.00021422, 0.00021323,
        0.00021392, 0.00021334, 0.00021396, 0.00021316, 0.00021428,
        0.00021267, 0.00021502, 0.00021148, 0.00021748, 0.00019761,
        0.00018076, 0.00018368, 0.00018337, 0.00018228, 0.00018439,
        0.00018142, 0.00018515, 0.00018071, 0.00018583, 0.00018004,
        0.00018652, 0.00017928, 0.00016711, 0.0001779 , 0.00016994,
        0.00016299, 0.00015871, 0.00017102, 0.00014535, 0.00017065,
        0.00016019, 0.00015036, 0.00015195, 0.00016634, 0.00016375,
        0.00014673, 0.00015695, 0.00014902, 0.00015561, 0.00014998,
        0.00015485, 0.00015063, 0.00015428, 0.00015113, 0.00015384,
        0.00015153, 0.00015348, 0.00015184, 0.0001532 , 0.00015208,
        0.000153  , 0.00015226, 0.00015285, 0.00015238, 0.00015275,
        0.00015246, 0.00015269, 0.00015251, 0.00015265, 0.00015253,
        0.00015272, 0.00015208, 0.00015605, 0.00018754, 0.00015398,
        0.00016637, 0.00018738, 0.00018053, 0.00018503, 0.00018152]))]
pproximant segments [('y', array([ 2.02358933e-07, -9.25938366e-08, -4.167668
        -4.54514520e-07,  8.17351975e-07, -1.44407386e-06,  2.80200038e-06,
        -2.84037524e-05, -1.65790843e-05, -1.97905320e-05, -3.60753766e-05,
        -2.61548557e-05, -3.45265289e-05, -2.64092814e-05, -3.52476491e-05,
        -2.38965295e-05, -5.01644390e-05, -3.47239402e-05, -4.31625376e-05,
        -6.58359131e-05, -5.82103530e-05, -6.30978830e-05, -5.92992437e-05,
        -8.30837525e-05, -5.88715193e-05, -8.68748466e-05, -9.28900845e-05,
        -9.07066133e-05, -9.23679472e-05, -9.05281995e-05, -9.31167888e-05,
        -8.84686160e-05, -1.06577696e-04, -1.05268657e-04, -1.20072189e-04,
        -1.23484264e-04, -1.20939760e-04, -1.23085090e-04, -1.21004967e-04,
        -1.23464415e-04, -1.19622884e-04, -1.29437627e-04, -1.39757409e-04,
        -1.19790653e-04, -1.61115255e-04, -1.40313132e-04, -1.39030468e-04,
        -1.53900066e-04, -1.52754830e-04, -1.52091758e-04, -1.52820052e-04,
        -1.53719957e-04, -1.39595300e-04, -1.36820396e-04, -1.26079089e-04,
        -1.39099240e-04, -1.24909522e-04, -1.40004093e-04, -1.24032391e-04,
        -1.40974938e-04, -1.22847705e-04, -1.42589066e-04, -1.20186349e-04,
        -1.29593958e-04, -1.38767646e-04, -1.21853169e-04, -1.57304705e-04,
        -1.29289838e-04, -1.54517867e-04, -1.31061694e-04, -1.53154120e-04,
        -1.52559020e-04, -1.52129054e-04, -1.53501082e-04, -1.51240311e-04,
        -1.54365713e-04, -1.50354230e-04, -1.55378017e-04, -1.48856081e-04,
        -1.80416711e-04, -1.82059652e-04, -1.85046476e-04, -1.80660689e-04,
        -1.86016448e-04, -1.79412396e-04, -1.90341656e-04, -2.13082865e-04,
        -2.14734173e-04, -2.12444924e-04, -2.14592728e-04, -2.13221298e-04,
        -2.12451123e-04, -2.26812277e-04, -2.29076017e-04, -2.40528156e-04,
        -2.26671822e-04, -2.41740810e-04, -2.46166426e-04, -2.42440205e-04,
        -2.45541334e-04, -2.43024173e-04, -2.44982832e-04, -2.43565155e-04,
```

-2.44454830e-04, -2.44083712e-04, -2.43941235e-04, -2.44600378e-04,
-2.43405229e-04, -2.45192845e-04, -2.42657727e-04, -2.46449490e-04,
-2.38188863e-04, -2.13028601e-04, -2.33785817e-04, -2.14154570e-04,
-2.12886560e-04, -2.14494343e-04, -2.12658852e-04, -2.14651795e-04,
-2.12550309e-04, -2.14724510e-04, -2.12504237e-04, -2.14750995e-04,
-2.12491548e-04, -2.14754808e-04, -2.12492465e-04, -2.14752523e-04,
-2.12493207e-04, -2.14755361e-04, -2.12485524e-04, -2.14768253e-04,
-2.12467887e-04, -2.14789325e-04, -2.12445768e-04, -2.14809159e-04,
-2.12432540e-04, -2.14810338e-04, -2.12450192e-04, -2.14765634e-04,
-2.12532032e-04, -2.14633750e-04, -2.12731029e-04, -2.14343600e-04,
-2.13149207e-04, -2.13734485e-04, -2.14072614e-04, -2.12200044e-04,
-2.17237975e-04, -1.96702342e-04, -2.06965648e-04, -2.03148535e-04,
-1.96772293e-04, -1.87494850e-04, -2.00166352e-04, -1.85664947e-04,
-1.81098745e-04, -1.84653181e-04, -1.81956770e-04, -1.83899392e-04,
-1.82628137e-04, -1.83303448e-04, -1.83146331e-04, -1.82872507e-04,
-1.83472468e-04, -1.82680029e-04, -1.83482654e-04, -1.82943826e-04,
-1.62394135e-04, -1.84783508e-04, -1.55813759e-04, -1.65490172e-04,
-1.64202633e-04, -1.46375838e-04, -1.57265720e-04, -1.48746651e-04,
-1.55820599e-04, -1.49877538e-04, -1.54803958e-04, -1.50865904e-04,
-1.53800138e-04, -1.51911008e-04, -1.52695735e-04, -1.53091270e-04,
-1.51419226e-04, -1.54494643e-04, -1.49835731e-04, -1.56361581e-04,
-1.47454062e-04, -1.59929885e-04, -1.38932723e-04, -1.43095414e-04,
-1.43294572e-04, -1.40352029e-04, -1.45555066e-04, -1.38156174e-04,
-1.27641368e-04, -1.35070441e-04, -1.32519344e-04, -1.17541815e-04,
-1.45225669e-04, -1.20019889e-04, -1.44182355e-04, -1.20213779e-04,
-1.24469749e-04, -1.18220400e-04, -1.31480512e-04, -1.36634888e-04,
-1.25222083e-04, -1.42264093e-04, -1.15336443e-04, -1.26469298e-04,
-1.18796015e-04, -1.24617334e-04, -1.20065932e-04, -1.23635080e-04,
-1.20878016e-04, -1.22938131e-04, -1.21489051e-04, -1.22396625e-04,
-1.21971476e-04, -1.21965248e-04, -1.22358906e-04, -1.21613237e-04,
-1.22686324e-04, -1.21294084e-04, -1.23024103e-04, -1.20889512e-04,
-1.23594858e-04, -1.19901088e-04, -1.25924009e-04, -1.05780462e-04,
-1.12804293e-04, -1.27020234e-04, -9.44739440e-05, -9.16161516e-05,
-9.11009265e-05, -9.20843740e-05, -9.10219969e-05, -9.20456368e-05,
-9.11228417e-05, -9.18944133e-05, -9.13248514e-05, -9.16380377e-05,
-9.16412682e-05, -9.12550022e-05, -9.20986640e-05, -9.07132635e-05,
-9.27388901e-05, -8.99525185e-05, -9.36562428e-05, -8.88128416e-05,
-9.51510156e-05, -8.66461196e-05, -9.90513945e-05, -7.02205580e-05,
-9.01128515e-05, -7.42843258e-05, -6.67222776e-05, -7.71225896e-05,
-6.35929755e-05, -8.13524239e-05, -5.44079230e-05, -6.52507297e-05,
-5.79106272e-05, -6.35521719e-05, -5.88711700e-05, -6.30095019e-05,
-5.91303105e-05, -6.29686983e-05, -5.89825795e-05, -6.32986776e-05,
-5.84493391e-05, -6.41037477e-05, -5.71875717e-05, -6.64204708e-05,
-5.01880422e-05, -4.74731205e-05, -5.83173824e-05, -2.85198330e-05,
-5.13283303e-05, -3.08256131e-05, -5.00726164e-05, -3.16358637e-05,
-2.91679753e-05, -3.20264371e-05, -2.89081945e-05, -3.21742846e-05,
-2.88649462e-05, -3.21116531e-05, -2.90460302e-05, -3.17801605e-05,
-2.96042417e-05, -3.07822484e-05, -3.18731763e-05, -1.71186985e-05,

```
 5.05911885e-06, -3.38949030e-06,  2.65886774e-06, -2.19262438e-06,
 1.83144584e-06, -1.51863787e-06,  1.22998608e-06, -9.51986294e-07,
 6.77886419e-07, -4.01807483e-07,  1.20839104e-07,  1.69151463e-07,
-4.71191015e-07,  7.89645128e-07, -1.12864655e-06,  1.49564585e-06,
-1.89978164e-06,  2.35578045e-06, -2.88803130e-06,  3.54199437e-06,
-4.40812437e-06,  5.71645796e-06, -8.30349745e-06,  2.21281662e-05])),
-5.59686450e-05, -7.76202651e-05, -7.35665672e-05, -9.02966131e-05,
-9.28817608e-05, -8.91872915e-05, -1.19253848e-04, -1.09140441e-04,
-1.09882181e-04, -1.29916065e-04, -1.13401416e-04, -1.43508048e-04,
-1.24254468e-04, -1.37767143e-04, -1.29122753e-04, -1.32290705e-04,
-1.59663963e-04, -1.47442333e-04, -1.57287432e-04, -1.47219049e-04,
-1.83014490e-04, -1.67664955e-04, -1.73267967e-04, -1.88700069e-04,
-1.76668982e-04, -2.02199357e-04, -1.88004284e-04, -1.95016386e-04,
-1.98810973e-04, -2.10084894e-04, -2.16382236e-04, -2.11191451e-04,
-2.16054672e-04, -2.10644299e-04, -2.40455207e-04, -2.44151073e-04,
-2.45268631e-04, -2.41530332e-04, -2.52017984e-04, -2.60951289e-04,
-2.43214512e-04, -2.81053071e-04, -2.66340387e-04, -2.44555005e-04,
-2.67164607e-04, -2.78849766e-04, -2.70808931e-04, -2.79157743e-04,
-2.65760405e-04, -2.57372099e-04, -2.81345507e-04, -2.56931060e-04,
-2.66664545e-04, -2.77735468e-04, -2.72876932e-04, -2.75743630e-04,
-2.74037622e-04, -2.75150378e-04, -2.71728553e-04, -2.38562701e-04,
-2.54880462e-04, -2.57487438e-04, -2.48351018e-04, -2.63728958e-04,
-2.36504304e-04, -2.61734938e-04, -2.53284379e-04, -2.38854846e-04,
-2.49337667e-04, -2.38067412e-04, -2.52453727e-04, -2.22441391e-04,
-2.21986091e-04, -2.28779681e-04, -2.11827937e-04, -2.13180581e-04,
-2.15067877e-04, -2.11610080e-04, -2.15977459e-04, -2.11076229e-04,
-2.16246524e-04, -2.11023929e-04, -1.95754517e-04, -2.11374208e-04,
-1.95171844e-04, -2.12264189e-04, -2.14129614e-04, -2.14750471e-04,
-1.86568985e-04, -1.96067049e-04, -1.94386797e-04, -1.77471375e-04,
-1.86996709e-04, -1.80216652e-04, -1.85269484e-04, -1.81522773e-04,
-1.84200951e-04, -1.82419404e-04, -1.83461307e-04, -1.82980279e-04,
-1.83154116e-04, -1.82819131e-04, -1.84619363e-04, -1.69920851e-04,
-1.67309685e-04, -1.56866328e-04, -1.69087041e-04, -1.56237889e-04,
-1.49056592e-04, -1.56022637e-04, -1.49247120e-04, -1.55838978e-04,
-1.49379193e-04, -1.56028691e-04, -1.46349426e-04, -1.20223427e-04,
-1.48815248e-04, -1.40971169e-04, -1.15872899e-04, -1.26309198e-04,
-1.18746684e-04, -1.24800150e-04, -1.19743694e-04, -1.24349666e-04,
-1.17218602e-04, -8.81332962e-05, -9.96808230e-05, -1.08592620e-04,
-8.73333775e-05, -9.37034711e-05, -9.04343833e-05, -9.19261438e-05,
-9.18381993e-05, -9.05907582e-05, -9.33194387e-05, -8.86264097e-05,
-9.68072272e-05, -7.29021485e-05, -8.66725532e-05, -7.92943538e-05,
-5.61089473e-05, -6.37407866e-05, -5.93325676e-05, -6.21463842e-05,
-6.03152585e-05, -6.14823439e-05, -6.07794209e-05, -6.11616560e-05,
-6.09847484e-05, -6.10606367e-05, -6.09771378e-05, -6.12056683e-05,
-6.06022950e-05, -6.21592117e-05, -3.60267804e-05, -2.85595379e-05,
-3.17483355e-05, -2.96092039e-05, -3.12285993e-05, -2.99490348e-05,
-3.09745301e-05, -3.01500404e-05, -3.08156450e-05, -3.02675180e-05,
-3.07455339e-05, -3.02771514e-05, -3.08201707e-05, -3.00732936e-05,
```

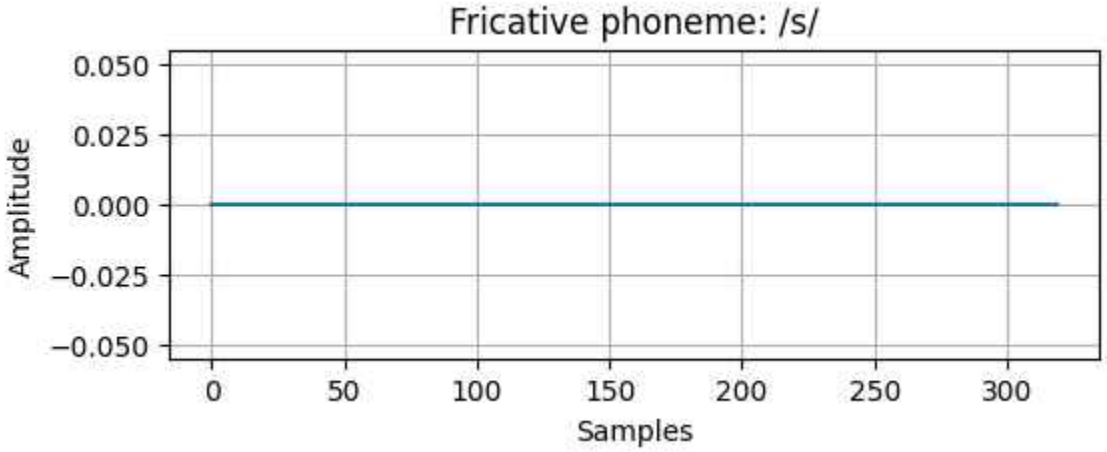
```
-3.12488701e-05, -2.91725009e-05, -3.35401564e-05, -1.50460255e-05,
-2.04633106e-05, -3.62127757e-05, -2.25729746e-05, -6.99714292e-07,
 1.03052298e-06, -1.05706567e-06,  1.01048499e-06, -9.40141035e-07,
 8.60803993e-07, -7.77872629e-07,  6.93878974e-07, -6.10336429e-07,
 5.28365490e-07, -4.48941137e-07,  3.72921932e-07, -3.00977263e-07,
 2.33965693e-07, -1.72381988e-07,  1.16822775e-07, -6.77246135e-08,
 2.52912287e-08,  1.02154445e-08, -3.88245098e-08,  6.06087269e-08,
-7.56554073e-08,  8.44884198e-08, -8.73405952e-08,  8.48376658e-08,
-7.75035005e-08,  6.60220394e-08, -5.11863618e-08,  3.36149242e-08,
-1.43918442e-08, -5.92262950e-09,  2.63898983e-08, -4.61732270e-08,
 6.45668479e-08, -8.07922333e-08,  9.40926839e-08, -1.03798811e-07,
 1.09328539e-07, -1.10128894e-07,  1.05734216e-07, -9.56206350e-08,
 7.95698725e-08, -5.71017154e-08,  2.80851964e-08,  7.98900146e-09,
-5.12518454e-08,  1.02227204e-07, -1.61075150e-07,  2.28727004e-07,
-3.05619324e-07,  3.93149094e-07, -4.92800609e-07,  6.06902177e-07,
-7.38727977e-07,  8.93458491e-07, -1.07854430e-06,  1.30652916e-06,
-1.59825868e-06,  1.99296483e-06, -2.57304782e-06,  3.55188968e-06,
 1.46086531e-05,  1.90232095e-05,  2.49281511e-05,  1.25150837e-05,
 3.51894487e-05,  2.81069879e-05,  3.18333186e-05,  2.99484964e-05,
 3.04549176e-05,  3.12178672e-05,  2.90534081e-05,  3.31051269e-05,
 2.56294734e-05,  4.87858488e-05,  3.57824028e-05,  4.24040554e-05,
 4.59596486e-05,  5.80311680e-05,  6.29746937e-05,  5.97644030e-05,
 6.18215126e-05,  6.06292306e-05,  6.11283904e-05,  6.12043659e-05,
 6.06463436e-05,  6.16008329e-05,  6.03440712e-05,  6.17754267e-05,
 8.07395700e-05,  6.11455907e-05,  8.46527473e-05,  9.50499889e-05,
 8.86453781e-05,  9.42604311e-05,  8.89376461e-05,  9.41082762e-05,
 8.90518495e-05,  9.39906458e-05,  8.91938107e-05,  9.38110752e-05,
 8.94220939e-05,  9.35214412e-05,  8.97925929e-05,  9.30308815e-05,
 1.10847875e-04,  9.17468642e-05,  1.15407965e-04,  1.25014456e-04,
 1.20023768e-04,  1.23619247e-04,  1.20900506e-04,  1.22906640e-04,
 1.21549972e-04,  1.22276659e-04,  1.22191792e-04,  1.21579542e-04,
 1.23044709e-04,  1.20217177e-04,  1.47962652e-04,  1.53538829e-04,
 1.32111192e-04,  1.52335400e-04,  1.53069952e-04,  1.51959321e-04,
 1.53305387e-04,  1.51828426e-04,  1.53347384e-04,  1.51868517e-04,
 1.53227767e-04,  1.52067849e-04,  1.52945897e-04,  1.52438093e-04,
 1.52476641e-04,  1.53025001e-04,  1.51738961e-04,  1.53975445e-04,
 1.50447973e-04,  1.55916903e-04,  1.46805076e-04,  1.72033557e-04])),
-2.25871452e-04, -2.42389317e-04, -2.24911812e-04, -2.44438881e-04,
-2.18132816e-04, -2.25327938e-04, -2.06015451e-04, -2.06348428e-04,
-2.19320136e-04, -2.08786281e-04, -2.17849651e-04, -2.09895486e-04,
-2.16907414e-04, -2.10750863e-04, -2.16102038e-04, -2.11523235e-04,
-2.15355656e-04, -2.12244850e-04, -2.14661588e-04, -2.12905361e-04,
-2.14044136e-04, -2.13465231e-04, -2.13565654e-04, -2.13818887e-04,
-2.13428939e-04, -2.13497551e-04, -2.15115928e-04, -2.00018112e-04,
-2.01318238e-04, -2.22037605e-04, -2.02323456e-04, -1.88176389e-04,
-1.75496069e-04, -2.04051321e-04, -2.06081429e-04, -1.97876478e-04,
-1.92755688e-04, -1.78262795e-04, -1.88295031e-04, -1.65894482e-04,
-1.95214365e-04, -1.86217687e-04, -1.81681971e-04, -1.82767399e-04,
```

-1.88800361e-04, -2.02408715e-04, -1.76638147e-04, -1.87552156e-04,
-1.79612922e-04, -1.85949379e-04, -1.80804549e-04, -1.84883116e-04,
-1.81888696e-04, -1.83661861e-04, -1.83410113e-04, -1.81504729e-04,
-1.87275378e-04, -1.65138117e-04, -1.78241389e-04, -1.69377003e-04,
-1.54463749e-04, -1.73298293e-04, -1.66554935e-04, -1.55135349e-04,
-1.76764181e-04, -1.73060558e-04, -1.65857899e-04, -1.57115457e-04,
-1.49604501e-04, -1.54070149e-04, -1.53513814e-04, -1.38533738e-04,
-1.41854602e-04, -1.57861345e-04, -1.48962587e-04, -1.55271293e-04,
-1.50587686e-04, -1.54029767e-04, -1.51633707e-04, -1.53098284e-04,
-1.52491484e-04, -1.52291643e-04, -1.53260131e-04, -1.51553337e-04,
-1.53972403e-04, -1.50865133e-04, -1.54636888e-04, -1.50225183e-04,
-1.55250629e-04, -1.49639760e-04, -1.55805290e-04, -1.49118103e-04,
-1.56292546e-04, -1.48665247e-04, -1.56713941e-04, -1.48266990e-04,
-1.36762625e-04, -1.47837418e-04, -1.37320545e-04, -1.46903680e-04,
-1.39555501e-04, -1.32898684e-04, -1.46713981e-04, -1.53852408e-04,
-1.31611130e-04, -1.53755405e-04, -1.27377542e-04, -1.20687284e-04,
-1.42467907e-04, -1.23334059e-04, -1.18491698e-04, -1.38756717e-04,
-1.30139917e-04, -1.19525554e-04, -1.22882586e-04, -1.22290803e-04,
-1.21079596e-04, -1.23693026e-04, -1.19907389e-04, -1.24701022e-04,
-1.19035671e-04, -1.25448321e-04, -1.38753167e-04, -1.25957784e-04,
-1.38361225e-04, -1.26233121e-04, -1.17855045e-04, -1.26285042e-04,
-1.38247240e-04, -1.26160536e-04, -1.38396048e-04, -1.26148589e-04,
-1.37554453e-04, -1.37674200e-04, -1.58019378e-04, -1.35049617e-04,
-1.46453429e-04, -1.40950535e-04, -1.20251469e-04, -1.58249124e-04,
-1.50297885e-04, -1.41532713e-04, -1.38751726e-04, -1.61302713e-04,
-1.44900085e-04, -1.60540236e-04, -1.41029261e-04, -1.27651787e-04,
-1.13775066e-04, -1.43917598e-04, -1.23480131e-04, -1.38557720e-04,
-1.49542087e-04, -1.21235185e-04, -1.19613745e-04, -1.37725699e-04,
-1.51966524e-04, -1.16757699e-04, -1.51151529e-04, -1.39445576e-04,
-1.16733529e-04, -1.26029525e-04, -1.38850795e-04, -1.25501916e-04,
-1.39026772e-04, -1.25446488e-04, -1.18700184e-04, -1.25430262e-04,
-1.39073149e-04, -1.25387742e-04, -1.18780634e-04, -1.25336912e-04,
-1.18809505e-04, -1.25363134e-04, -1.18671567e-04, -1.25718521e-04,
-1.17868221e-04, -1.27633932e-04, -9.07501599e-05, -1.08907741e-04,
-1.17604068e-04, -1.03472827e-04, -1.03777616e-04, -8.47273841e-05,
-9.70325564e-05, -8.66073970e-05, -9.62701597e-05, -8.69197829e-05,
-9.61715632e-05, -8.69199939e-05, -7.58496608e-05, -8.69454670e-05,
-7.56561276e-05, -8.76710837e-05, -7.16986542e-05, -5.37313608e-05,
-6.79120858e-05, -5.37971573e-05, -7.15542410e-05, -8.79280851e-05,
-7.51964326e-05, -8.79358995e-05, -7.16996728e-05, -5.29031895e-05,
-7.31831533e-05, -7.48023522e-05, -5.92929282e-05, -6.15408644e-05,
-6.07274560e-05, -6.13116717e-05, -6.14426244e-05, -4.97667934e-05,
-2.33801038e-05, -8.87067290e-05, -2.98719169e-05, -1.54730296e-05,
-4.87499929e-05, -4.95630375e-05, -2.98245432e-05, -1.29778637e-05,
-2.88208394e-05, -9.18513251e-06, -5.98661427e-05, -1.17198942e-05,
-2.25018521e-05, -3.49425318e-05, -2.33602113e-05, -2.02744559e-07,
 6.72545866e-07, -7.02639227e-07, 5.18324669e-07, -1.31287379e-07,
-5.63086360e-07, 1.99244823e-06, -7.24381243e-06, -1.77624170e-05,

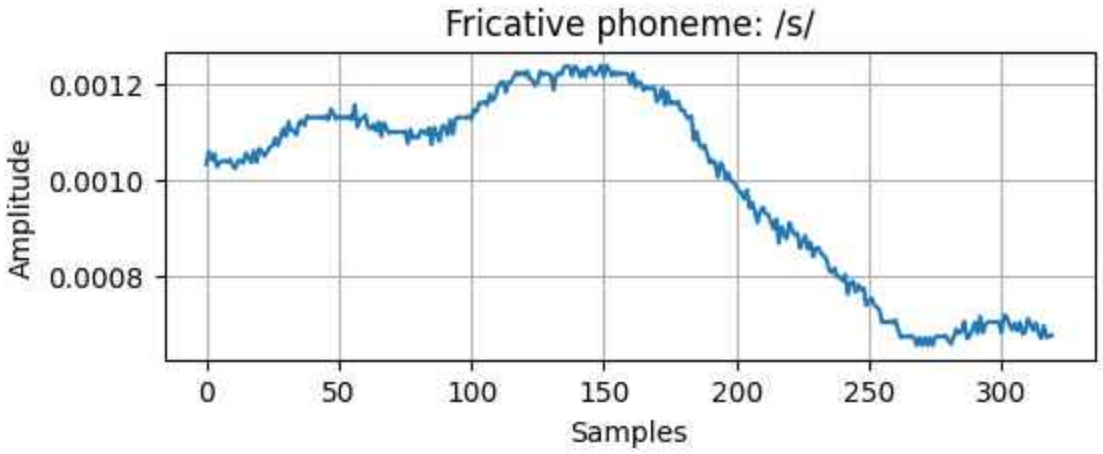
```
        4.89068771e-06, -2.81927350e-06,  1.81605719e-06, -1.13129499e-06,
        5.69969416e-07, -3.73693183e-08, -5.56465238e-07,  1.40722841e-06,
       -3.25799920e-06,  1.58175826e-05,  2.99948733e-05, -5.27361408e-06,
        2.78751832e-05,  3.02181579e-05,  3.11001204e-05,  3.01826512e-05,
        9.28132795e-06,  4.30197688e-05,  6.74624462e-05,  5.57634048e-05,
        6.61343802e-05,  5.58085740e-05,  6.65569678e-05,  5.50583936e-05,
        6.77506905e-05,  5.26487129e-05,  5.70113771e-05,  1.08642038e-04,
        4.13791277e-05,  7.54587818e-05,  6.73059840e-05,  3.35425138e-05,
        7.91312195e-05,  7.16489740e-05,  3.91863286e-05,  7.50748441e-05,
        7.02280086e-05,  7.00765522e-05,  9.55071300e-05,  5.35538420e-05,
        6.56533521e-05,  5.88311814e-05,  8.09172634e-05,  6.52661547e-05,
        2.75624916e-05,  6.41616061e-05,  4.59877774e-05,  7.88103789e-05,
        3.73078510e-05,  6.09364361e-05,  4.18387353e-05,  6.24731183e-06,
        6.06714748e-05,  6.45462424e-05,  2.78092921e-05,  6.42938539e-05,
        4.33940440e-05,  4.16999683e-05,  8.33980739e-05,  7.54343346e-05,
        1.07874977e-04,  2.22595409e-05,  1.05183106e-04,  8.17866530e-05,
        4.64478508e-05, -2.45845877e-06,  5.38646709e-05,  8.73673707e-05,
        4.75668348e-05,  7.41481781e-05,  2.18735076e-05,  6.62505627e-05,
        5.09228557e-05,  1.05835963e-04,  1.04693230e-04,  5.98717015e-05,
        4.41018492e-05,  3.43886204e-05,  5.23831695e-05,  1.17620453e-04]))]
```

v.

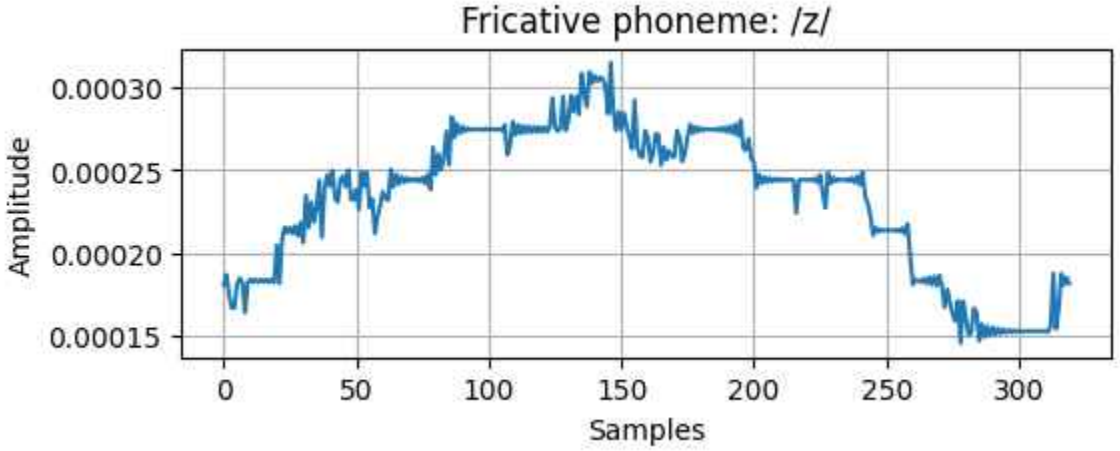
```
def plot_segments(segments, title):
  for ph, sig in segments:
    plt.figure(figsize=(6,2))
    plt.plot(sig)
    plt.title(f"{title} phoneme: /{ph}/")
    plt.xlabel("Samples")
    plt.ylabel("Amplitude")
    plt.grid(True)
    plt.show()
plot_segments(fricative_signals, "Fricative")
plot_segments(approximant_signals, "Approximant")
```



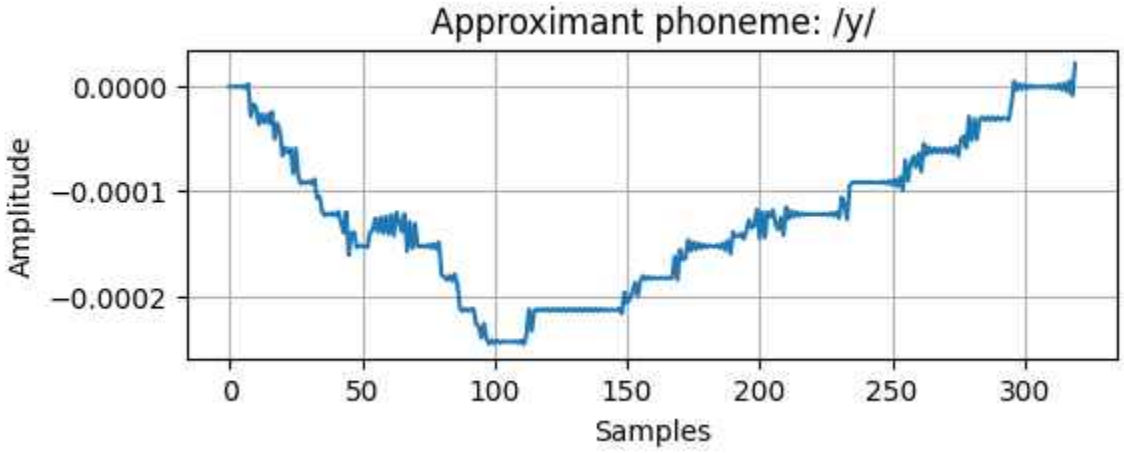
jpeg



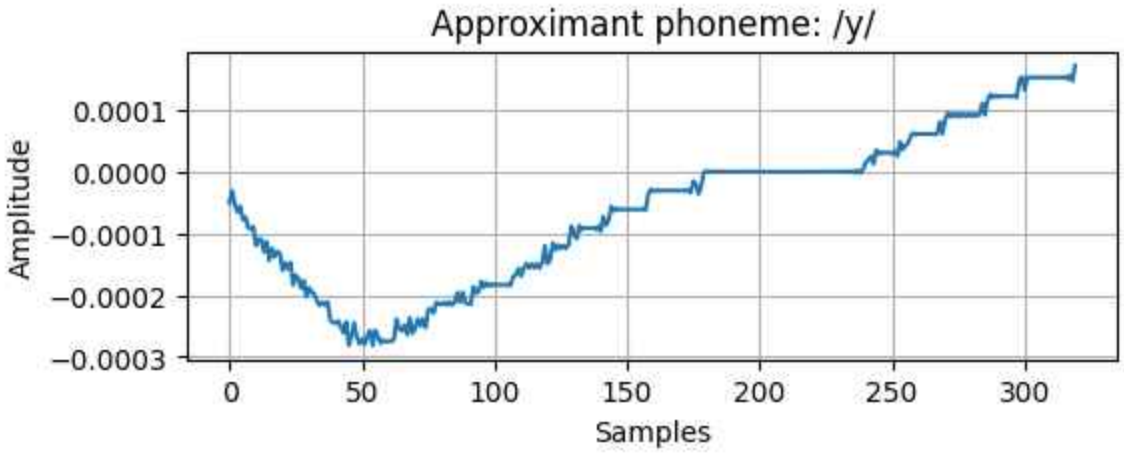
jpeg



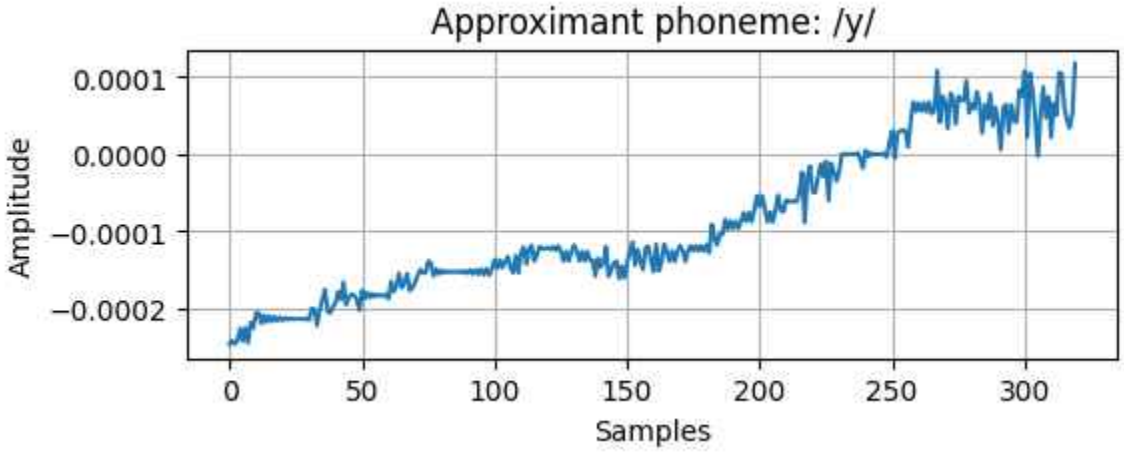
jpeg



jpeg


Approximant phoneme: /y/
Amplitude
0.0001
0.0000
−0.0001
−0.0002
−0.0003
0
50
100
150
200
250
300
Samples

jpeg


Approximant phoneme: /y/
Amplitude
0.0001
0.0000
−0.0001
−0.0002
0
50
100
150
200
250
300
Samples

jpeg