Speech autoencoder using deep fully-connected networks

Andrew Seagraves

1/3/19

Implementation Summary

- Vanilla autoencoder built using Keras/TensorFlow
- Custom pipeline for pre-processing and on-the-fly batch generation
- Specialized utility functions for training on LibriSpeech corpuses
- Mini-batches generated by traversing the raw audio signal for each speaker and breaking it into chunks of a specified size which become the feature vectors.
- Overlap allowed on input chunks as a means for incorporating temporal structure into the autoencoder. The corresponding non-overlapping output chunks are used as labels.

Pre-processing Workflow

build_speech_dict.py

- Generates a “speech_dict” data structure from the corpus
  - Mirrors the internal structure of the LibriSpeech corpus. Useful for traversing the speakers, chapters, and utterances.

- Concatenates each speaker’s utterances into a master 1D numpy array for the speaker. Writes array to disk.
  - The sequence of the utterances is preserved in the array.
  - Speaker array used for on-the-fly batch generation

Batch Generation Workflow

batch_mapping() function from utilities.py

- Generates a batch_map[] data structure at the start of training
- batch_map[] is a member of DataGenerator class

Batch generation during training (pseudo code):

$\forall \mathbf{i} \in [\mathbf{0}, \dots, \boldsymbol{N_{batch}} - \mathbf{1}]$

$[\boldsymbol{speaker_i}, \boldsymbol{c_{0i}}, \boldsymbol{c_{1i}}] =$ **batch_map[i]**

$\boldsymbol{X_{in}} =$ **chunkify_speaker_data(**$\boldsymbol{speaker_i}$**, **chunk_sizes, with_win=True)** (chunked feature matrix with l/r overlap)

$\boldsymbol{X_{out}} =$ **chunkify_speaker_data(**$\boldsymbol{speaker_i}$**, **chunk_sizes, with_win=False)** (chunked feature matrix, no overlap)

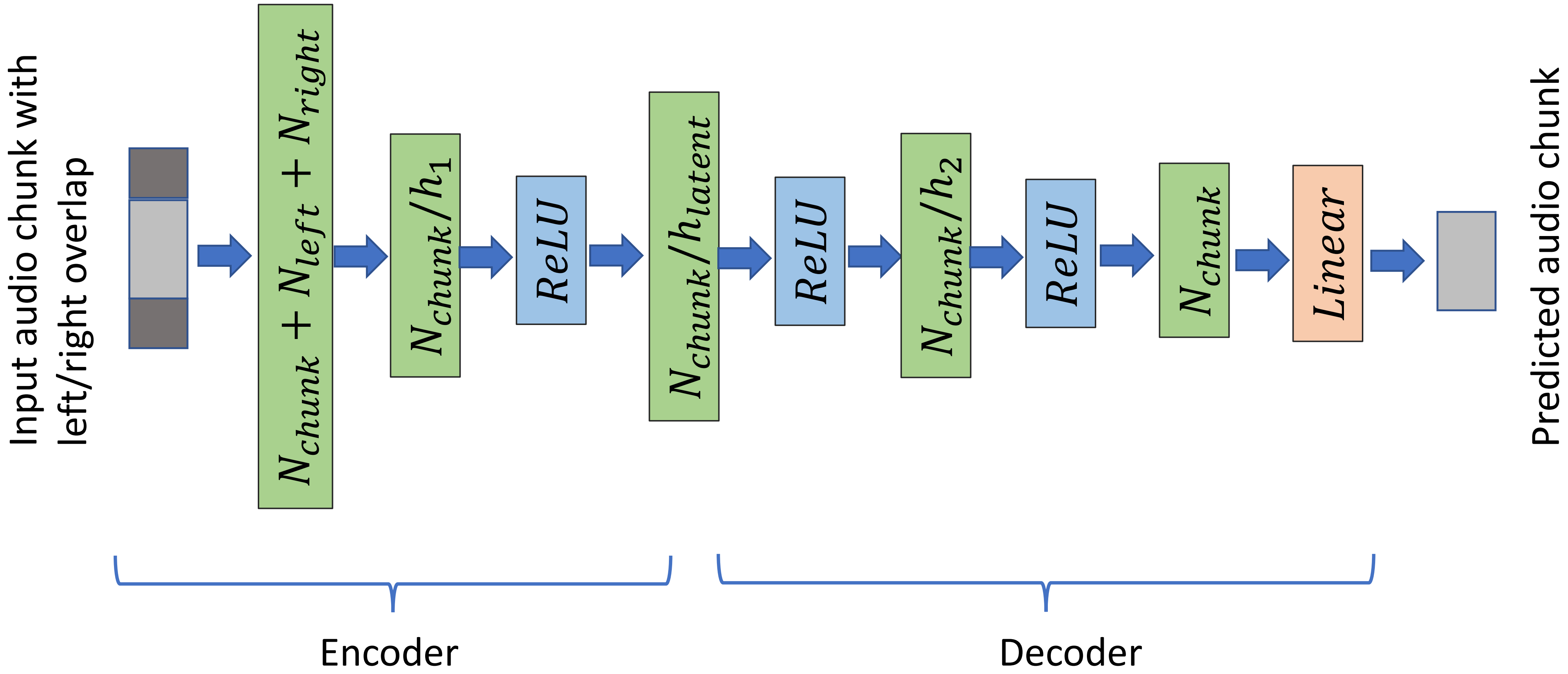
$\boldsymbol{X_{batch,in}} = \boldsymbol{X_{in}}[\boldsymbol{c_{0i}}: \boldsymbol{c_{1i}}]$

$\boldsymbol{X_{batch,out}} = \boldsymbol{X_{out}}[\boldsymbol{c_{0i}}: \boldsymbol{c_{1i}}]$

Implementation aspects

- Breaking the speaker data array into a chunked matrix is impractically slow using for loops
- A fast implementation is achieved using numpy
    - $\boldsymbol{as\_strided}$ with l/r overlap and $\boldsymbol{reshape}$ with no overlap

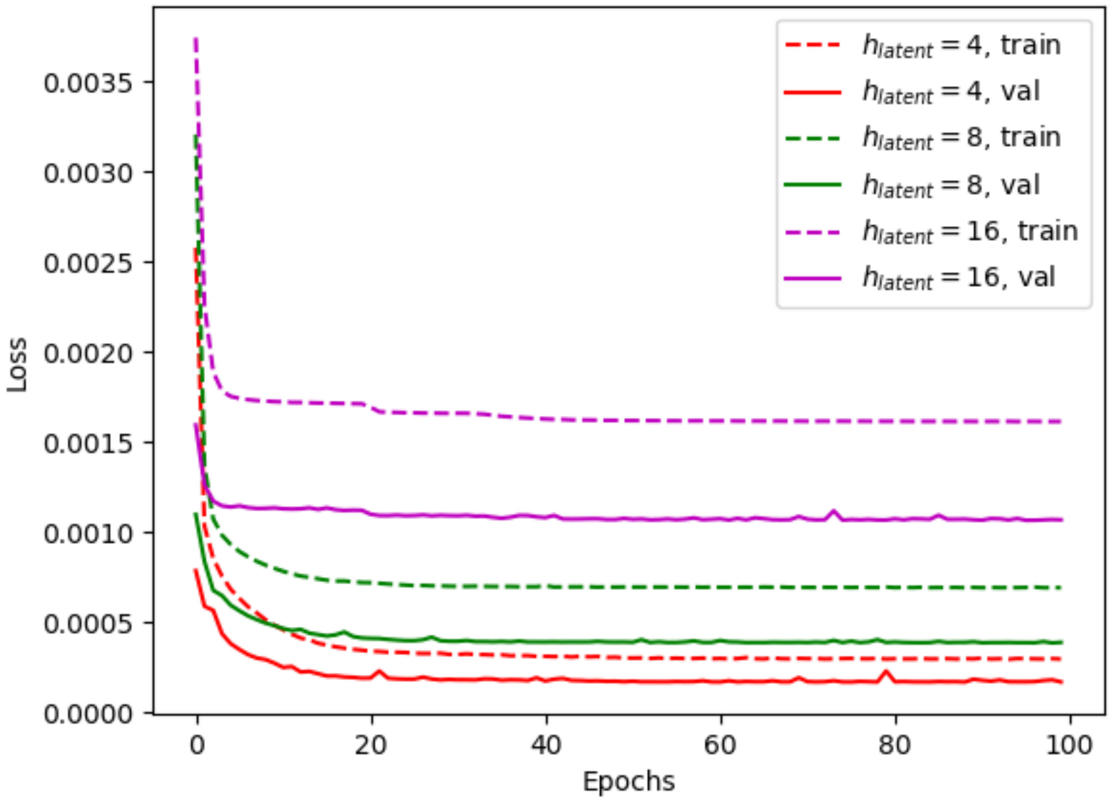
Example Network Architecture

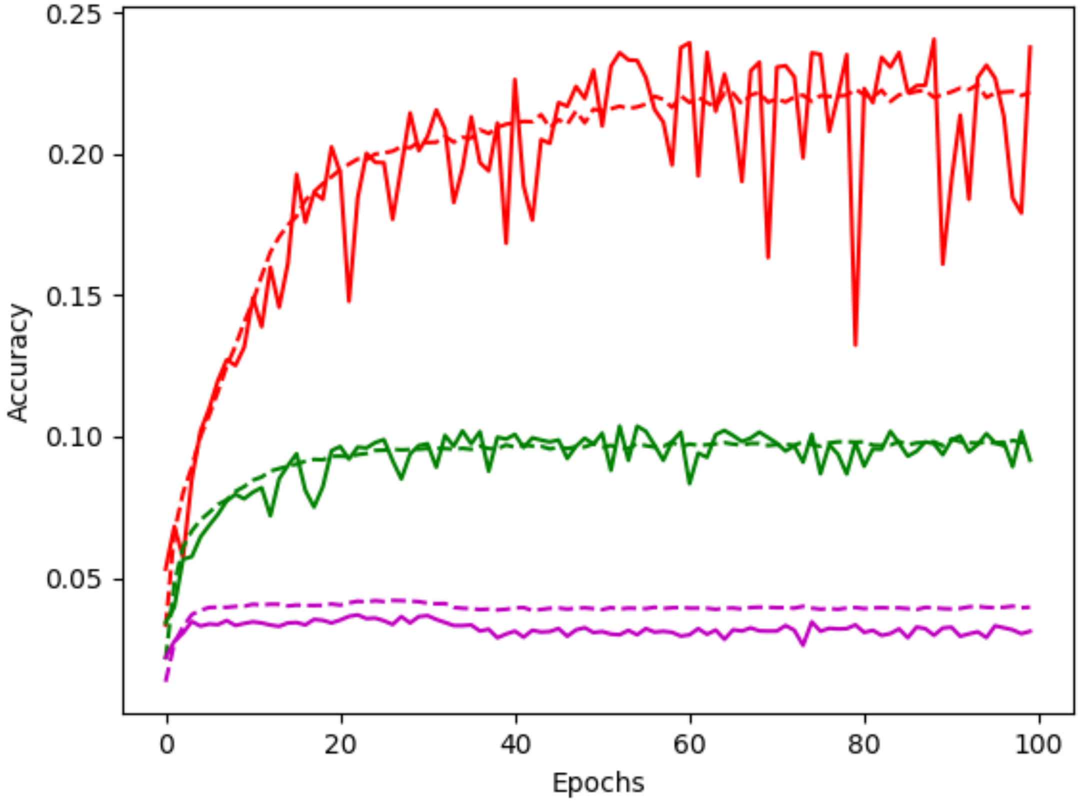


Training Experiments on Small Corpus

- dev-clean dataset (40 speakers, 5.1 hrs total, 294M samples @16kHz)
- Tried chunk size, batch size from Chorowski, et al. (2019)
  - $N_{chunk} = 5120\ (\Delta t_{chunk} = 320ms,\ 16\text{kHz}),$ batch size = 64
  - This chunk size is too large for the vanilla autoencoder and training fails with a non-decreasing loss function
- Reducing chunk size and increasing batch size proved successful
  - $N_{chunk} = 800\ (\Delta t_{chunk} = 50ms,\ 16\text{kHz}),$ batch size = 128
- Training parameters:
  - 100 epochs, learning rate = 1e-4, Adam optimizer, MSE loss
  - 2408 mini-batches per epoch, ~3min per epoch

3-layer networks

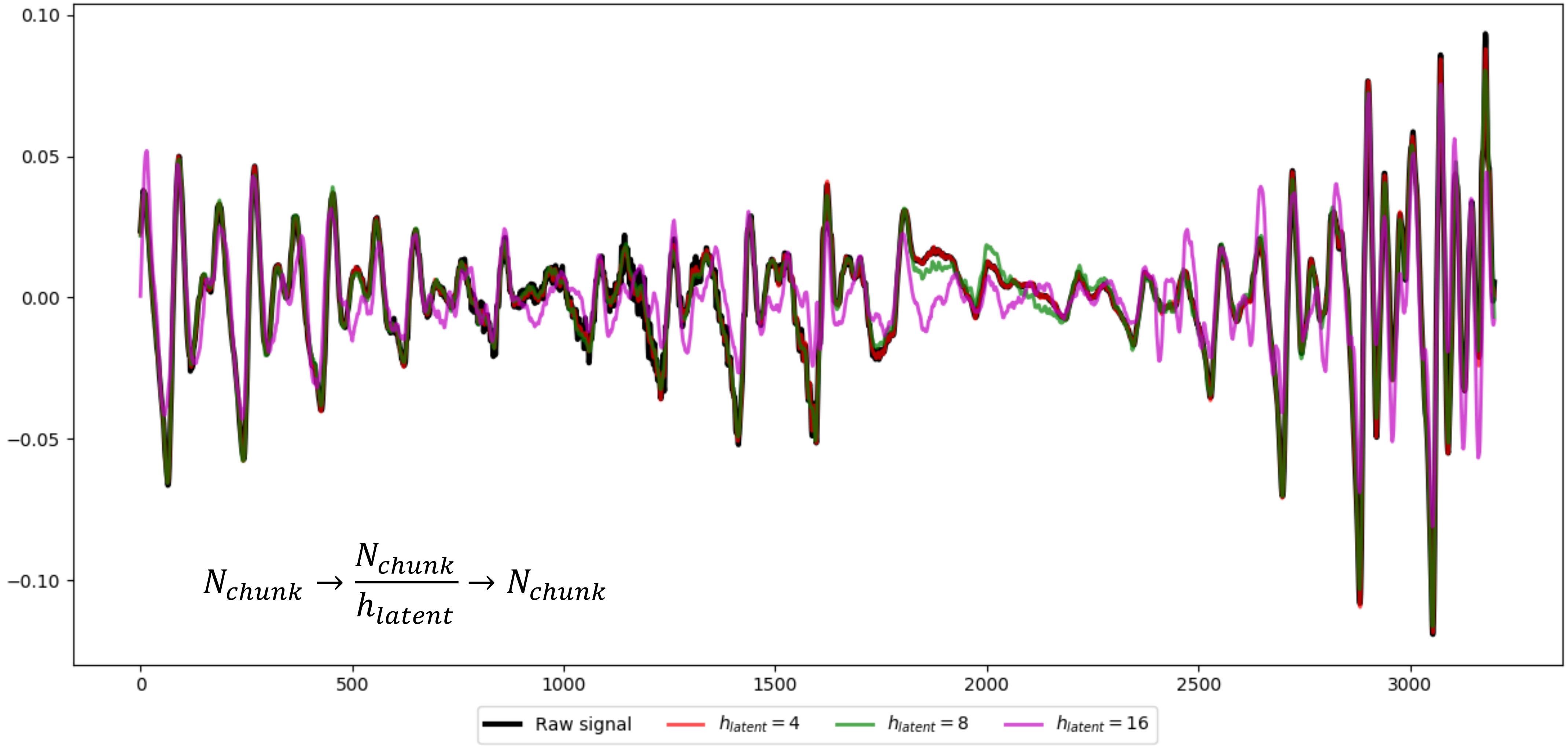




$$N_{chunk} \rightarrow \frac{N_{chunk}}{h_{latent}} \rightarrow N_{chunk}$$

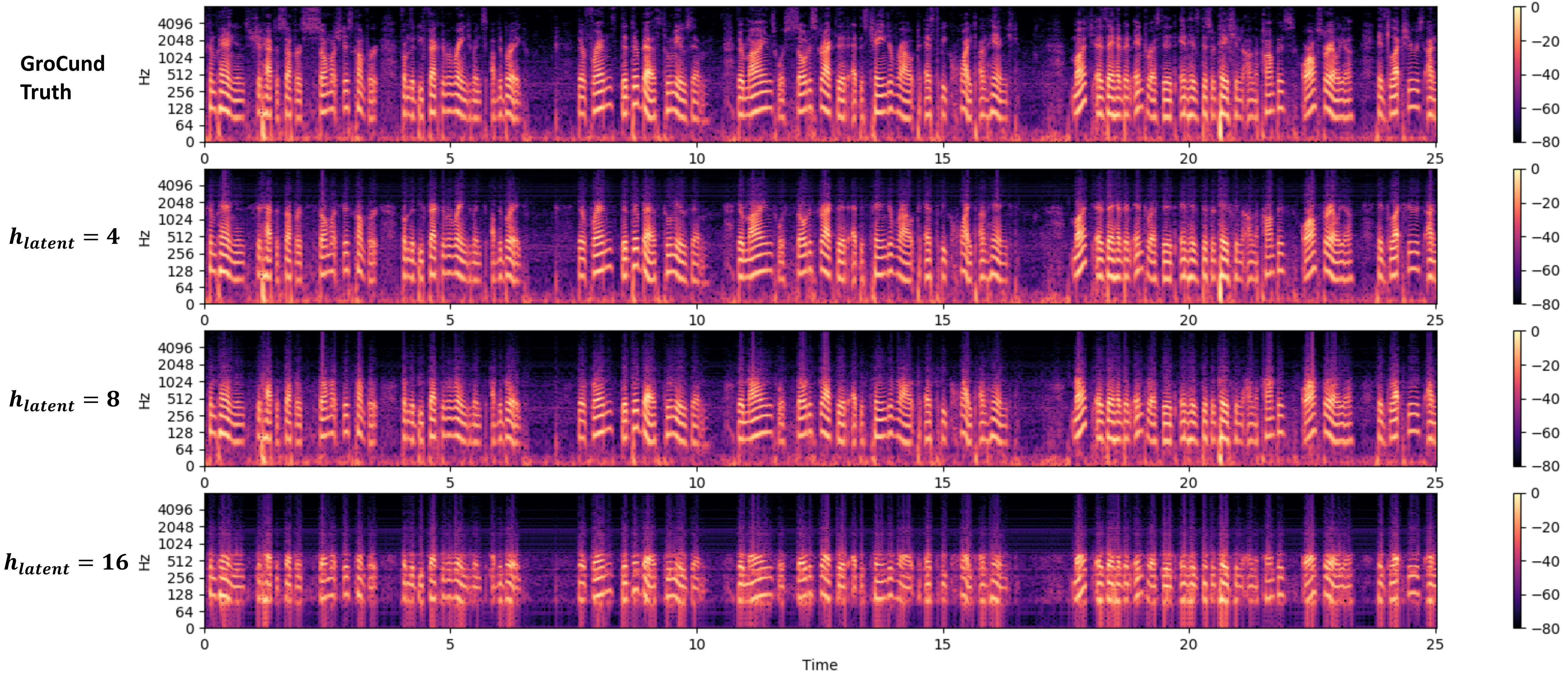
3-layer network

Speaker 2803
250ms audio sample (5 chunks)

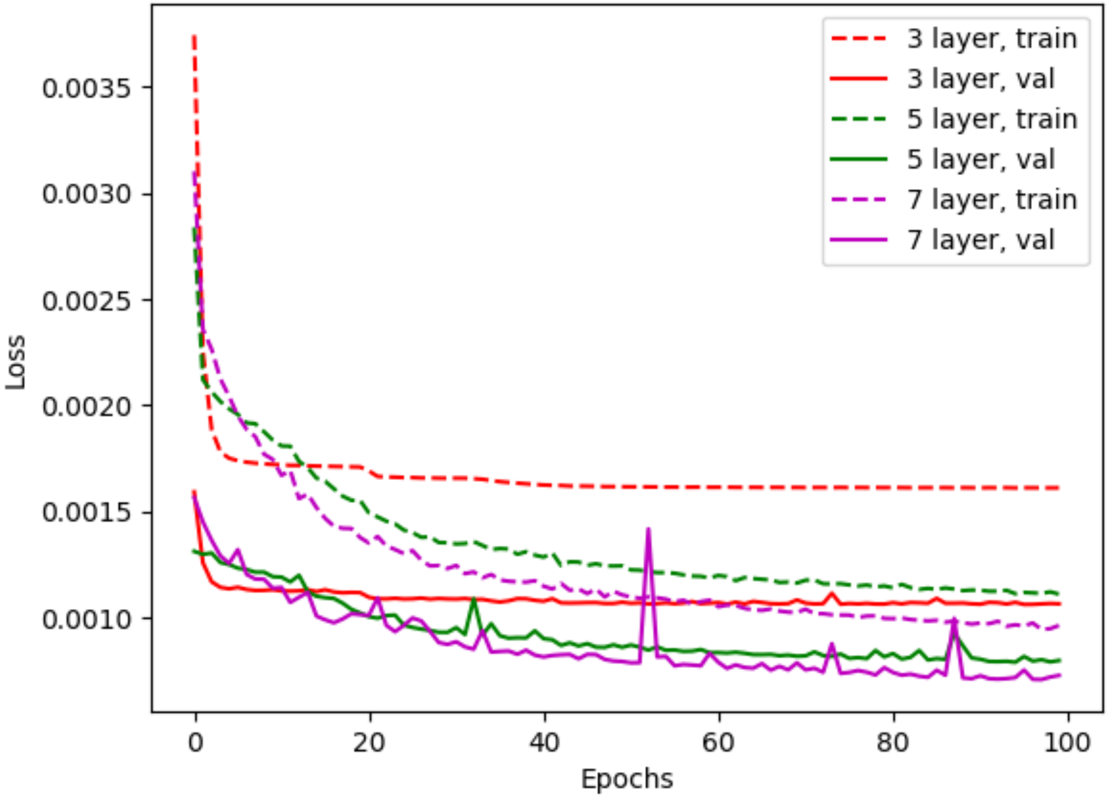


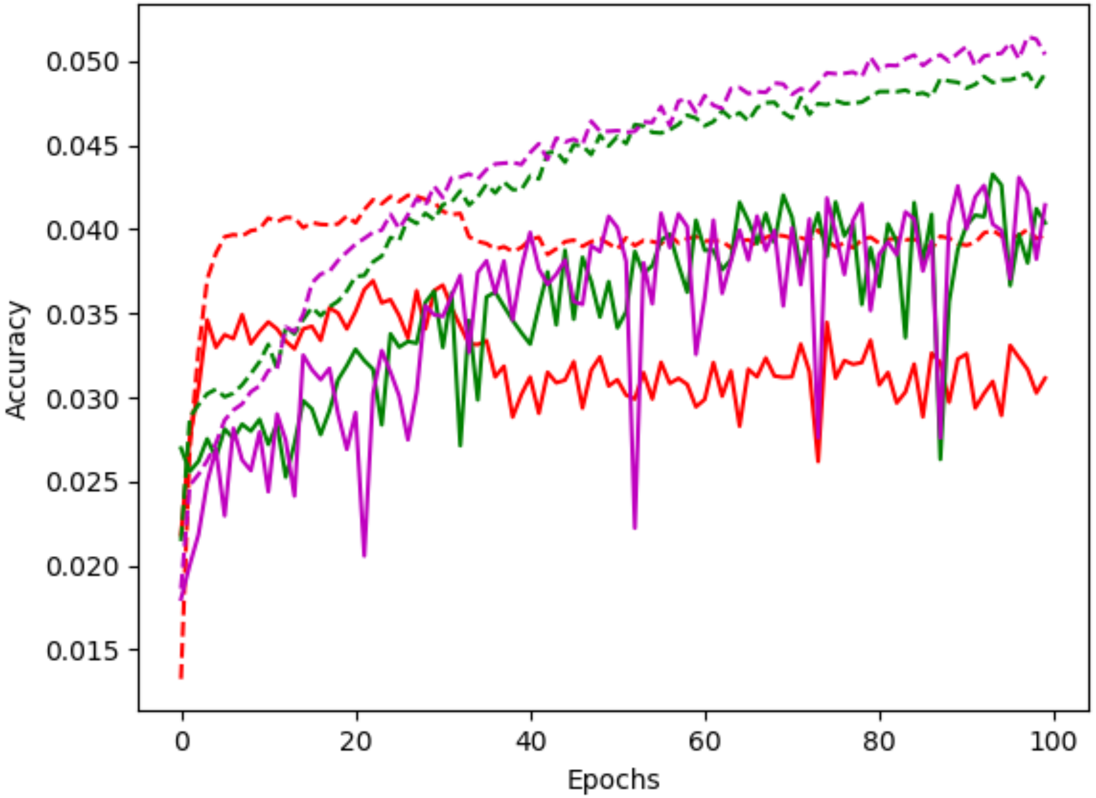
3-layer networks

Speaker 2803
25s audio sample, log-spectrogram



Increasing network depth

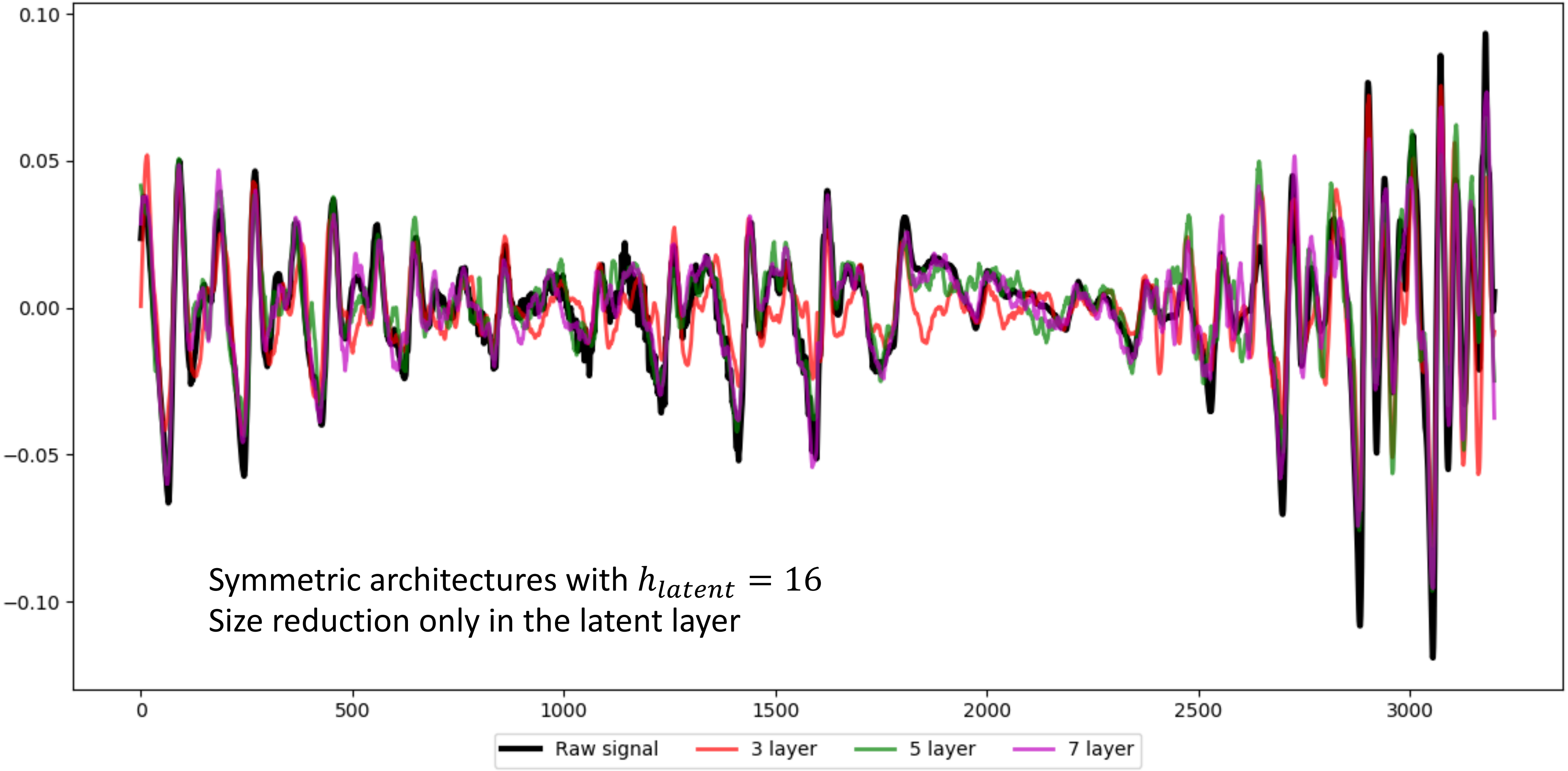




Symmetric architectures with $h_{latent} = 16$
Size reduction only in the latent layer

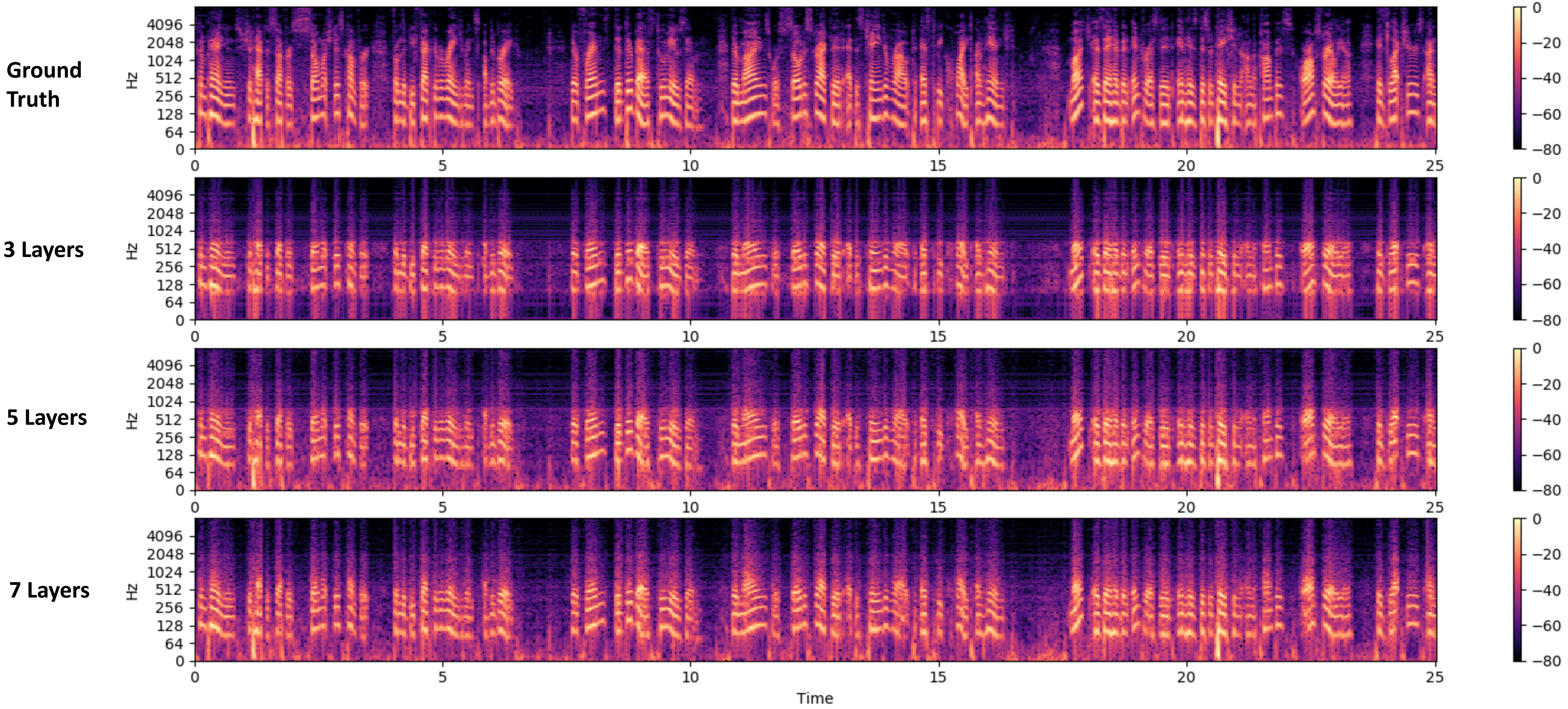
Increasing network depth

Speaker 2803
200ms audio sample (4 chunks)



Increasing network depth

Speaker 2803
25s audio sample, log-spectrogram



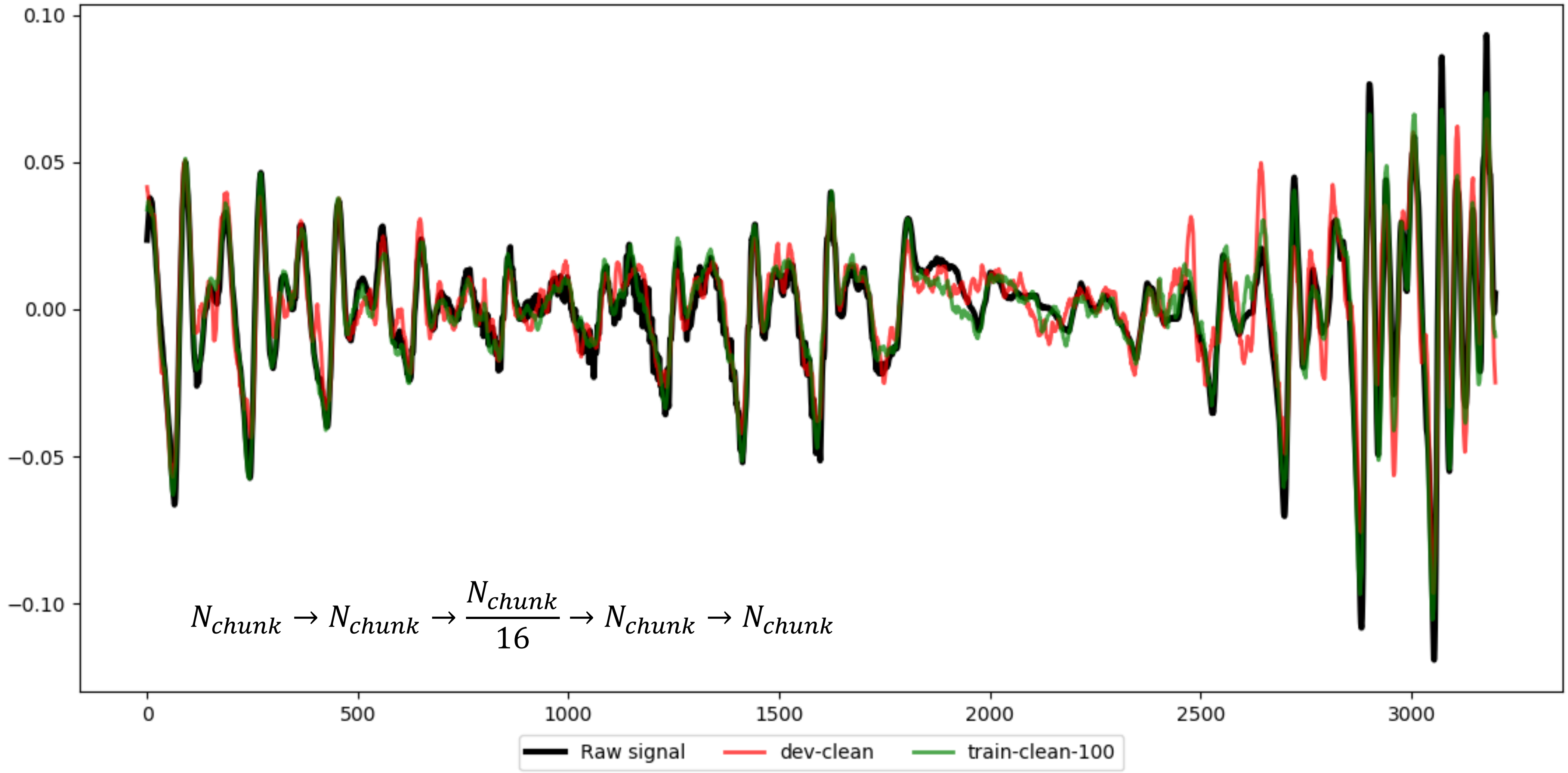
Training Experiment on Large Corpus

- train-clean-100 dataset (251 speakers, 100.6 hrs, 5.8B samples @16kHz)

- Experiment 1:
  - Symmetric architecture, 5 layers, $h_{latent} = 16$
  - $N_{chunk} = 800\ (\Delta t_{chunk} = 50ms,\ 16\text{kHz})$, batch size = 128
  - 10 epochs, learning rate = 1e-4, Adam optimizer, MSE loss
  - 45099 mini-batches per epoch
  - ~ 3.1hrs per epoch

Training on Large Corpus

Speaker 2803
200ms audio sample (4 chunks)



Training on Large Corpus

Speaker 2803
25s audio sample, log-spectrogram