Speech Autoencoder using Fully-Connected Neural Networks

Andrew Seagraves

1/4/19

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 permitted between adjacent input chunks as a means for incorporating temporal structure into the autoencoder.
- Driver implemented for automated hyperparameter studies

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 in 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):

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

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

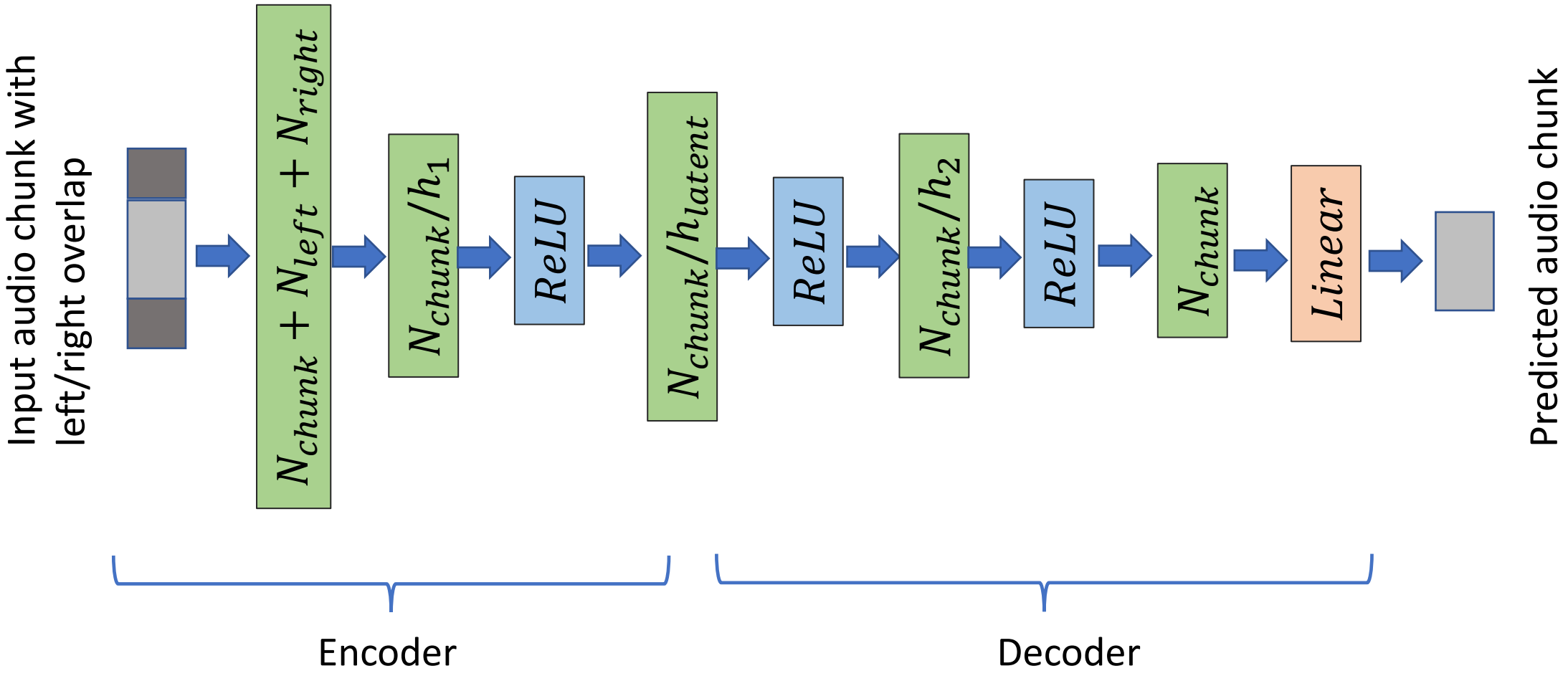
$\boldsymbol{X_{batch,in}} = \boldsymbol{X_{in}}[\boldsymbol{c_{0i}} : \boldsymbol{c_{1i}}]$ # batch feature matrix

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

Implementation aspects

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

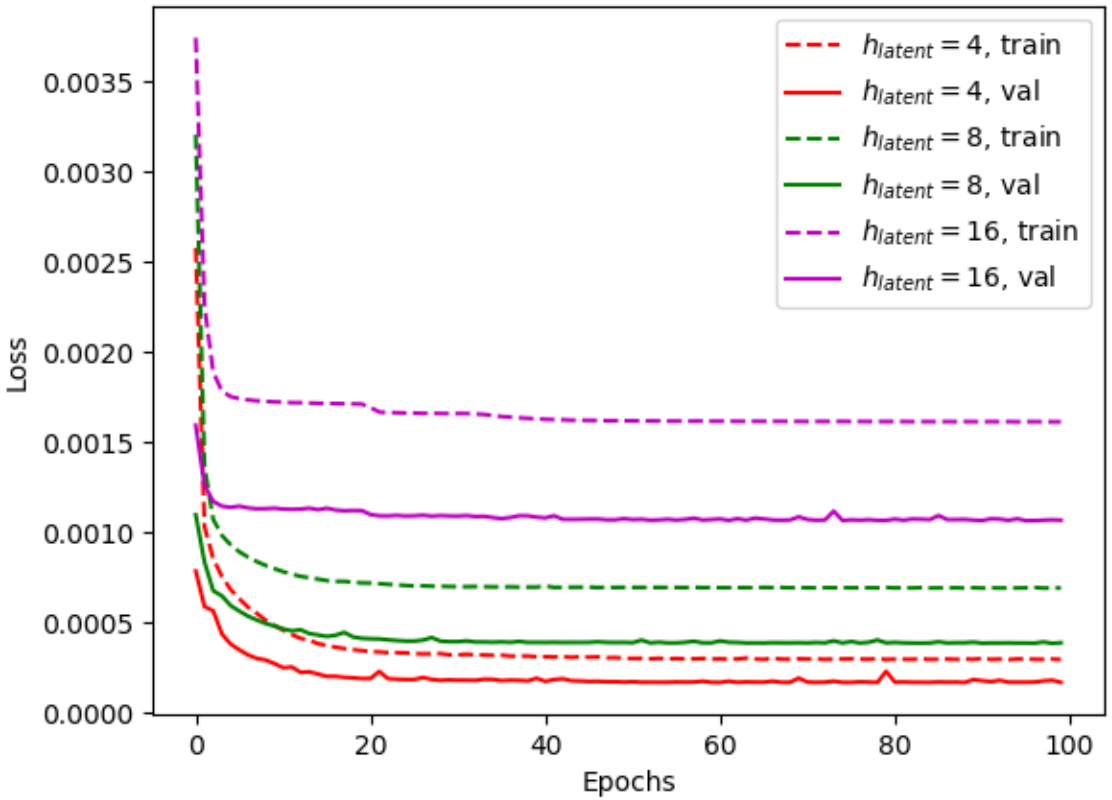
Example Network Architecture (5 layers)

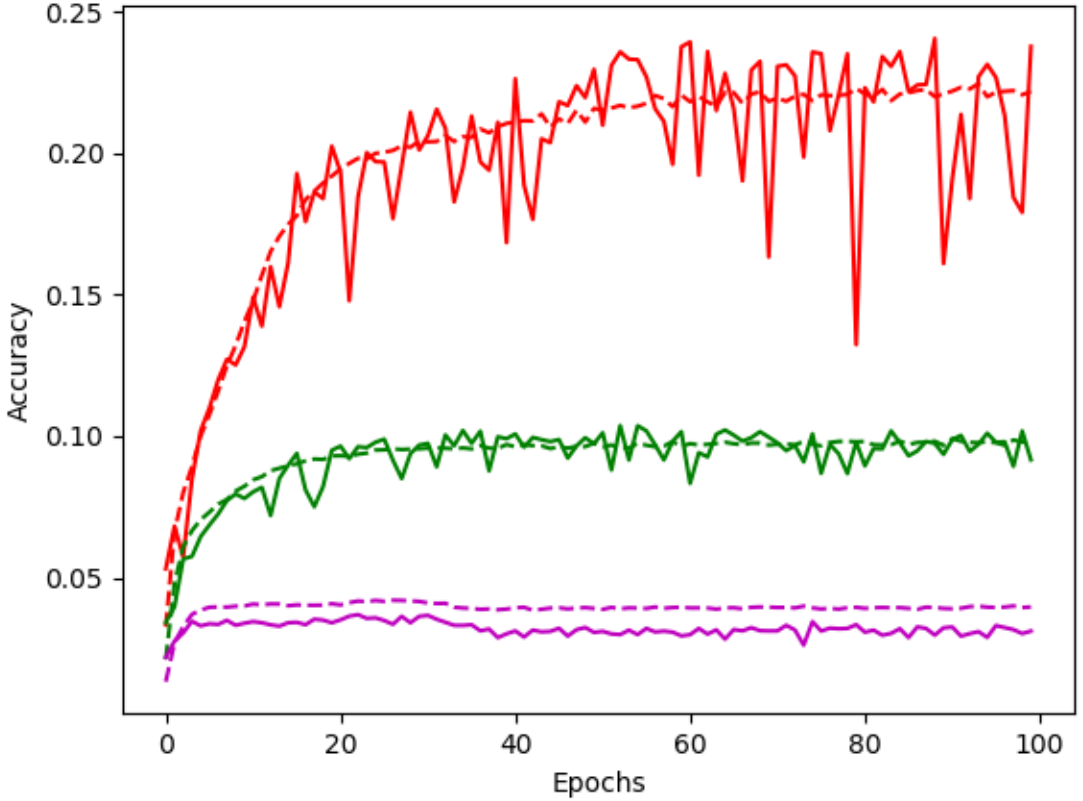


Training Experiments on Small Corpus

- 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
  - Hardware: NVIDIA GeForce GTX 770M (3GB, 960 cuda cores)
  - 2408 mini-batches per epoch, ~3min per epoch

3-layer network – training results

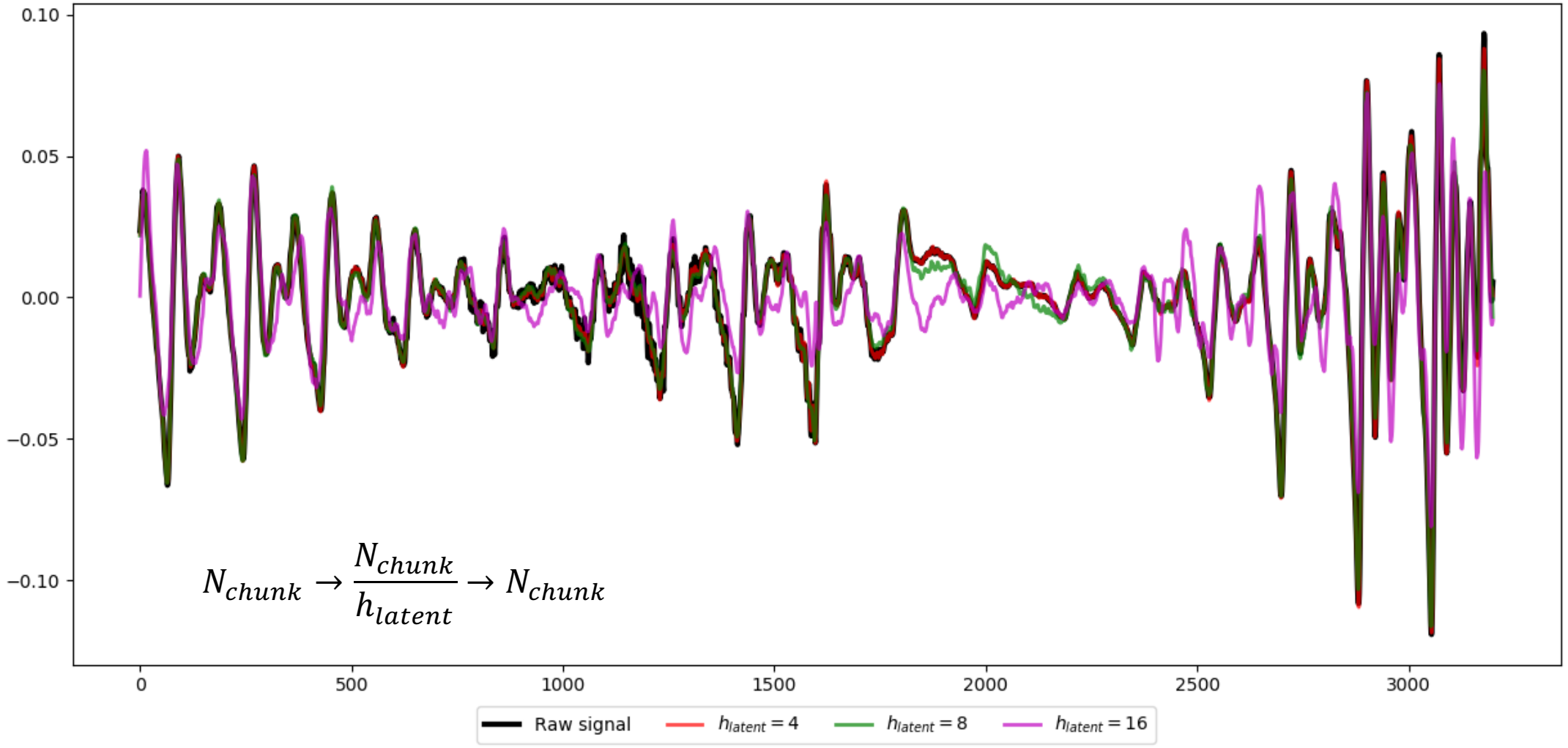




$$N_{chunk} \rightarrow \frac{N_{chunk}}{h_{latent}} \rightarrow N_{chunk}$$

3-layer network

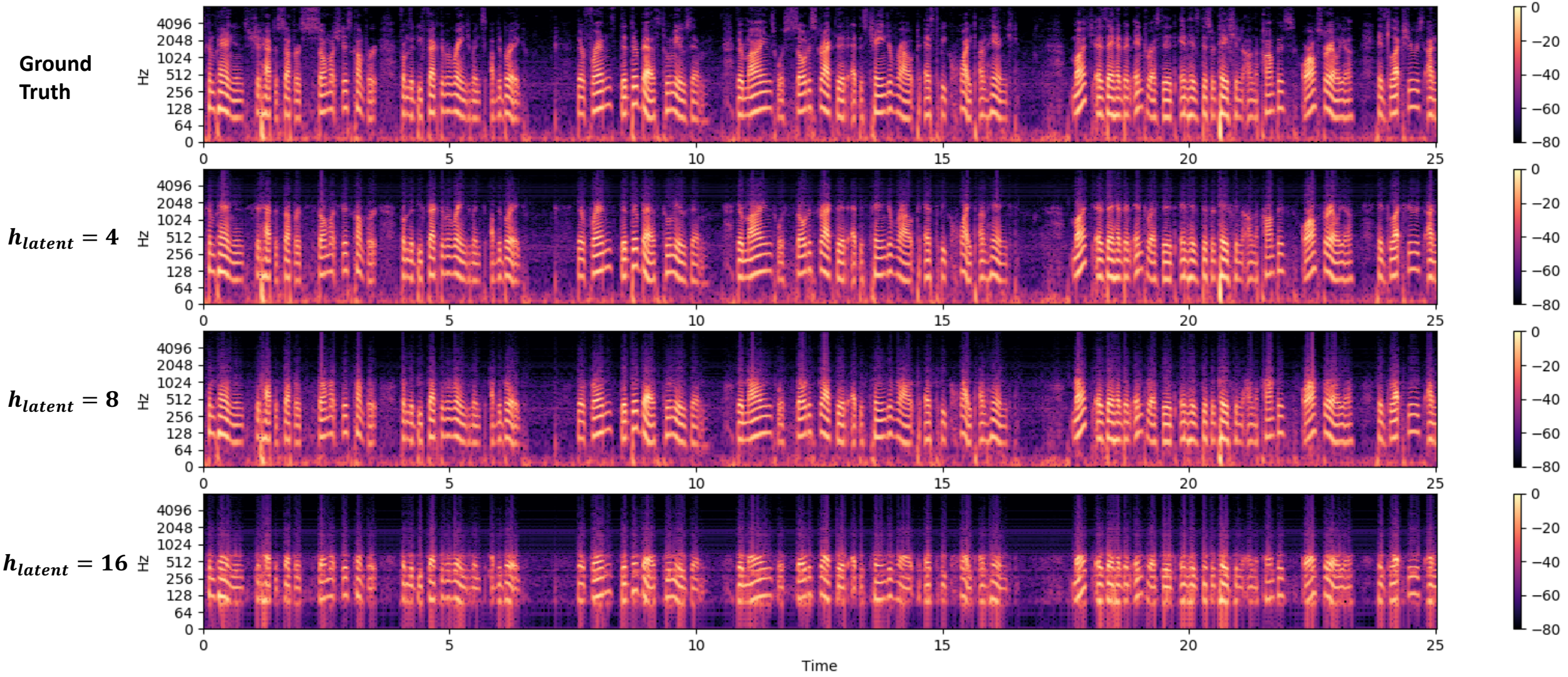
Speaker 2803
250ms audio sample (5 chunks)



Comparison of ground truth and reconstructed audio signals (model checkpoint with highest val_acc)

3-layer network

Speaker 2803
25s audio sample, log-spectrogram



3-layer network

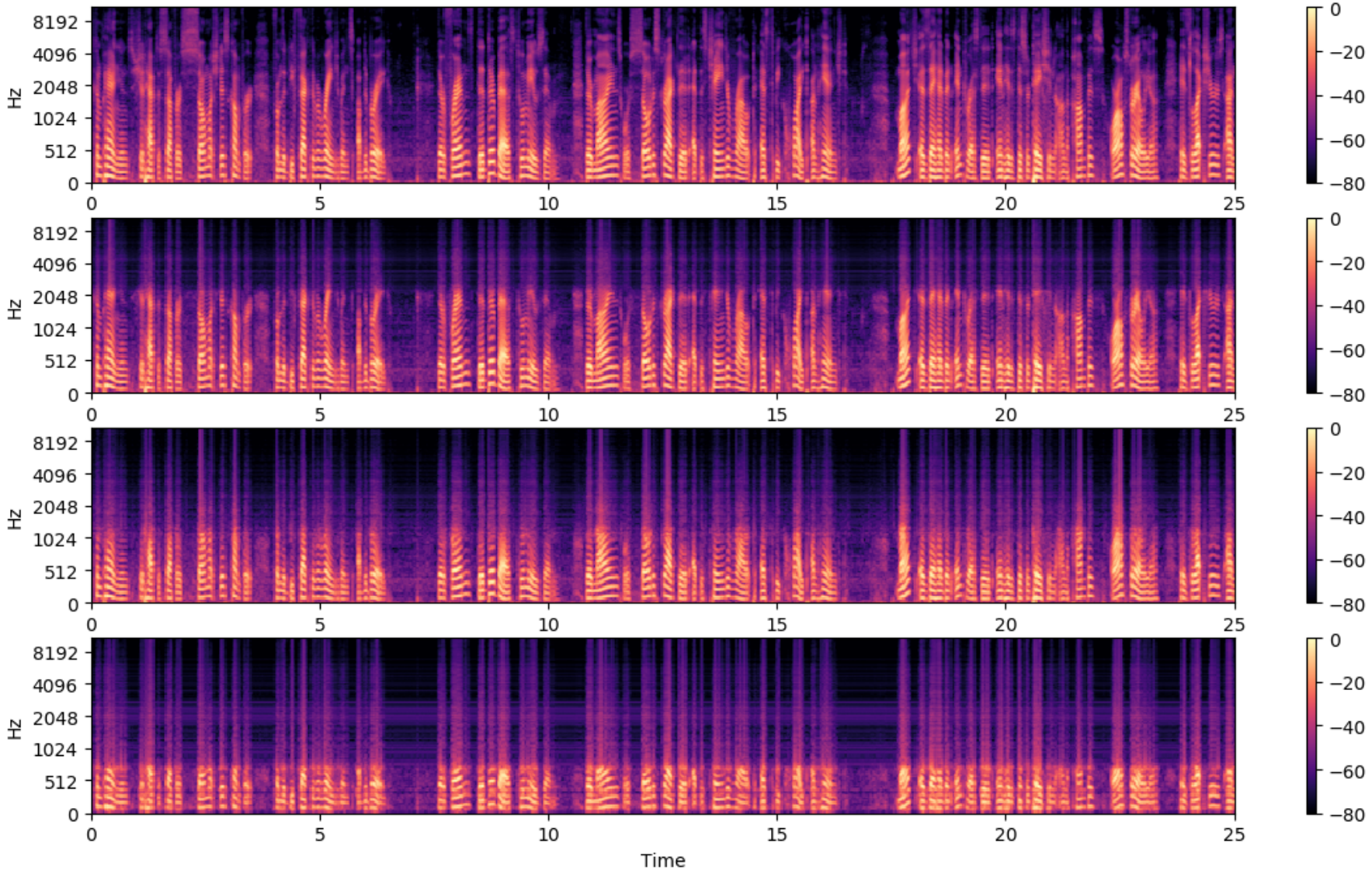
Speaker 2803
25s audio sample, mel-spectrogram

Ground Truth

$h_{latent} = 4$

$h_{latent} = 8$

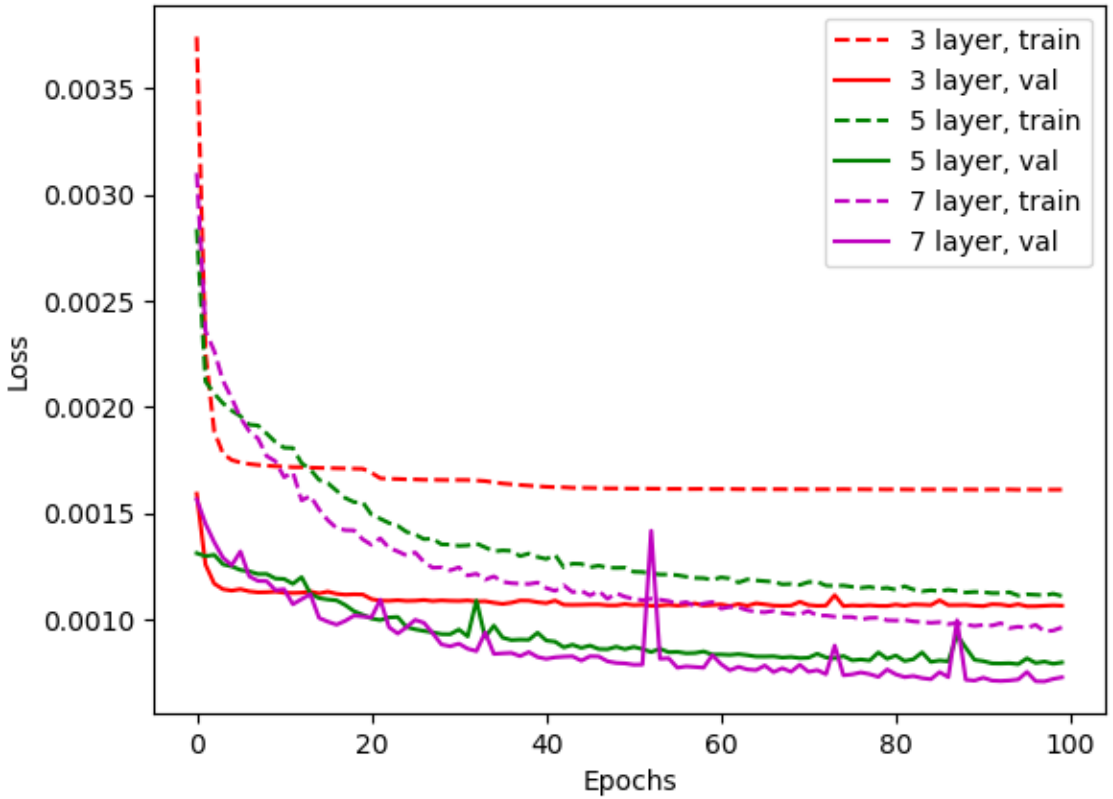
$h_{latent} = 16$

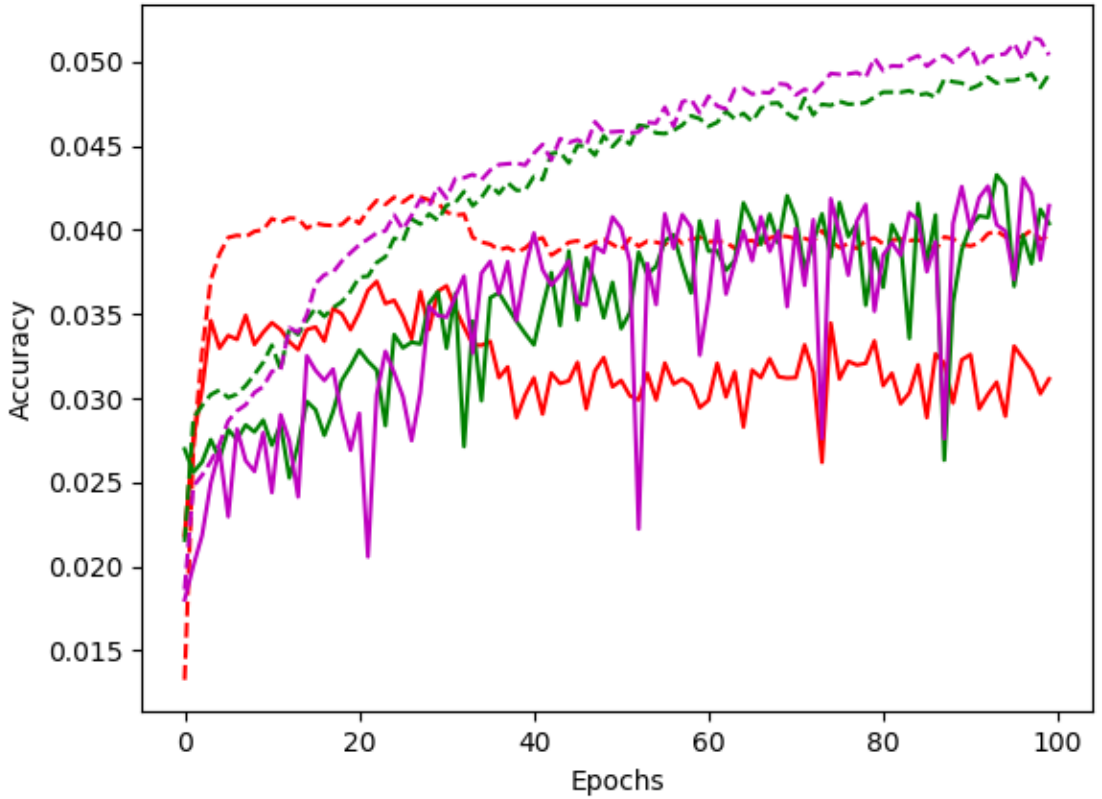


3-layer network - observations

- The reconstructed audio signals are in reasonable agreement with ground truth
- However, accuracy deteriorates quickly as the latent size decreases
- For $h_{latent} = 16$, there are significant discrepancies between reconstructed and ground truth signals
- Plotting the reconstructions in log- and mel-spectrograms after applying STFT provides a more insightful visualization.
- The reconstructed spectrograms show good qualitative agreement with ground truth.
- For all values of h_{latent}, the 3-layer network filters out high-frequency information above a certain threshold frequency. The threshold decreases with h_{latent}. This is seen most clearly in the mel-spectrogram.
- For $h_{latent} = 16$, the 3-layer network also filters out frequency information below ~100Hz. This is seen most clearly in the log-spectrogram.

Increasing network depth

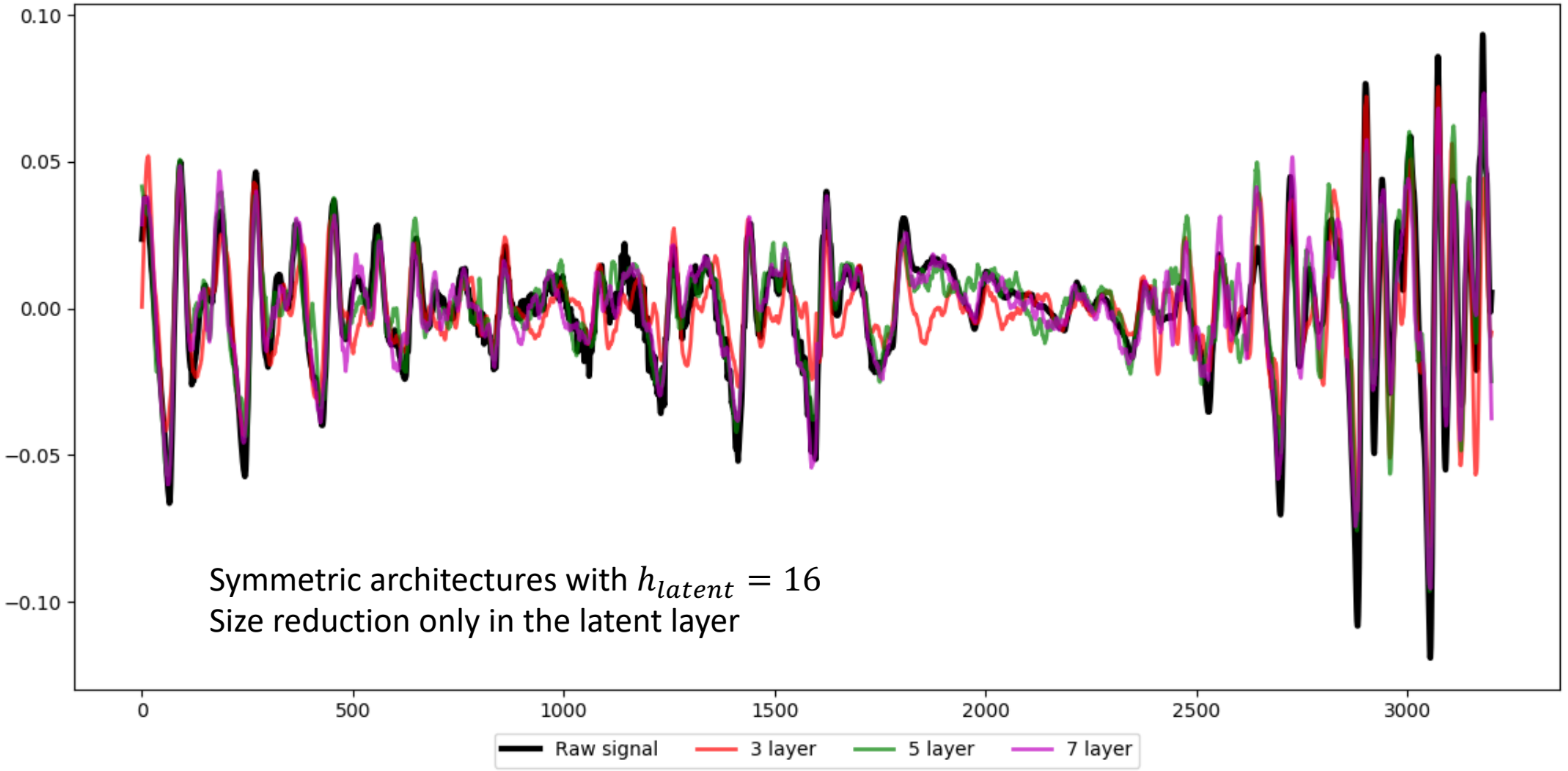




Symmetric architectures with increasing numbers of layers and $h_{latent} = 16$
Size reduction occurs only in the latent layer

Increasing network depth

Speaker 2803
200ms audio sample (4 chunks)

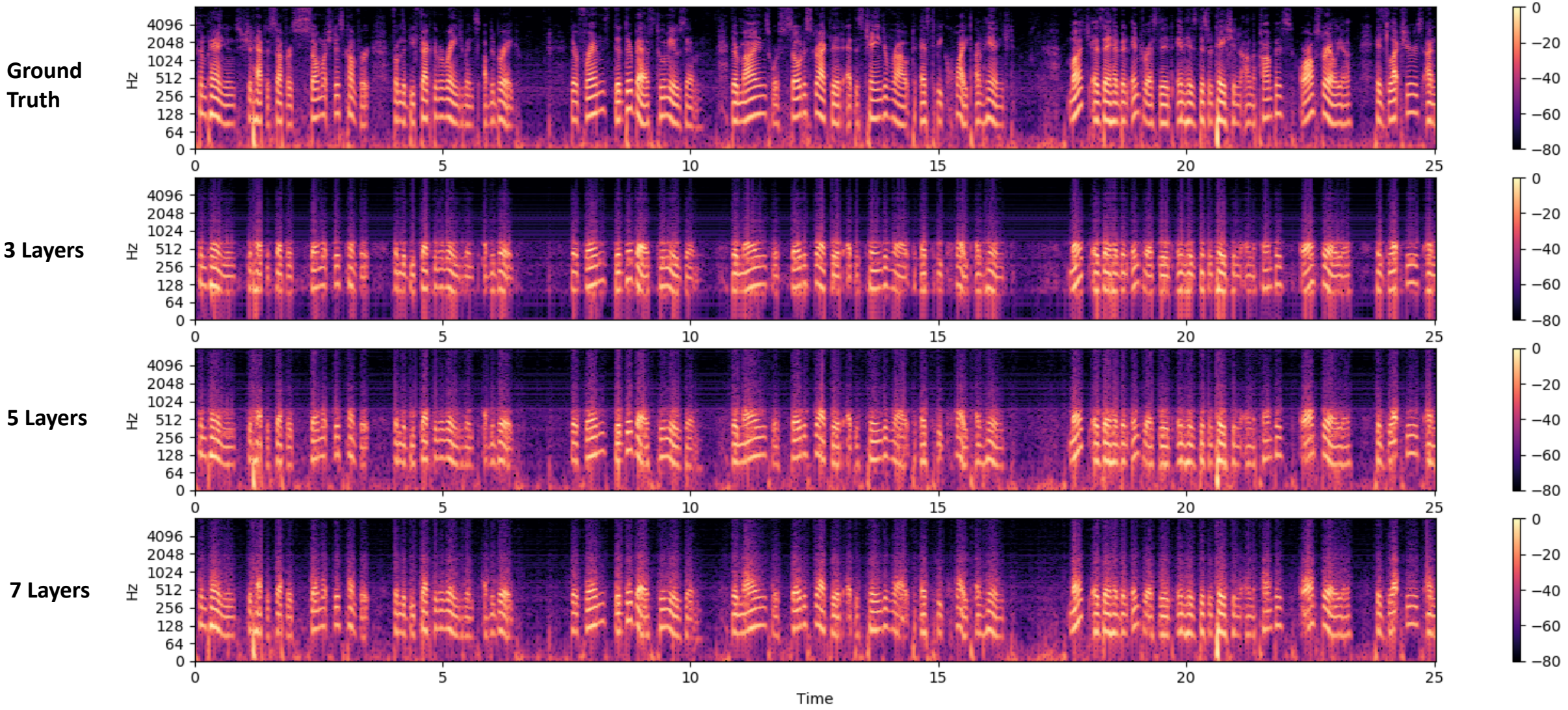


Symmetric architectures with $h_{latent} = 16$
Size reduction only in the latent layer

Comparison of ground truth and reconstructed audio signals (model checkpoint with highest val_acc)

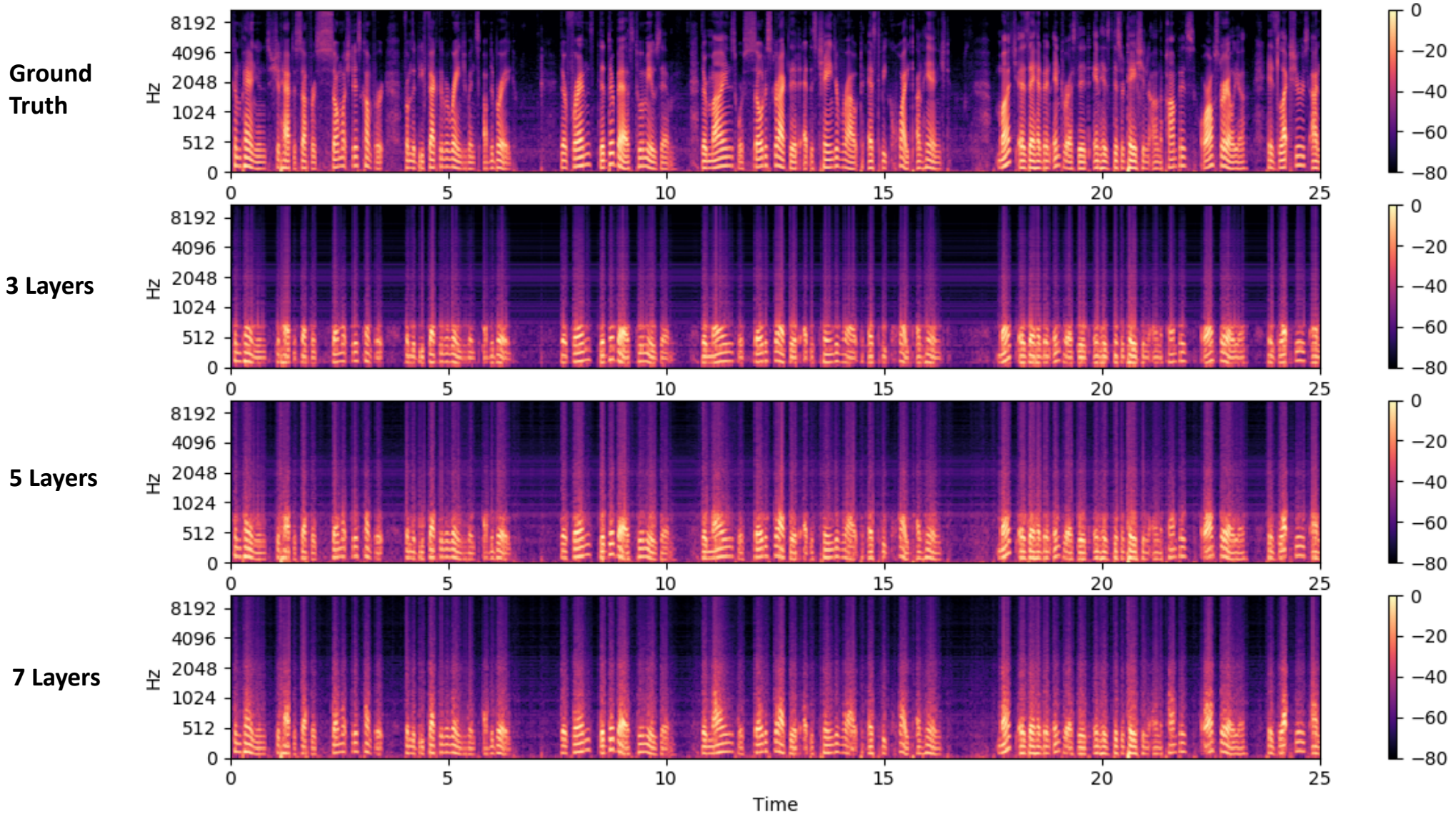
Increasing network depth

Speaker 2803
25s audio sample, log-spectrogram



Increasing network depth

Speaker 2803
25s audio sample, mel-spectrogram



Increasing network depth - observations

- In this experiment, we investigated symmetric networks with $h_{latent} = 16$ and increasing number of layers (3, 5, and 7).
- Increasing the network depth improves the agreement between reconstructed and ground truth signals
- The log-spectrogram shows that frequency information below 100Hz is captured in the reconstruction when the network depth is increased to 5 or 7 layers.
- In the mel-spectrogram, the differences between 3, 5, and 7 layers are more subtle. We can see that at 7 layers, the reconstruction captures structure in higher frequency bands.

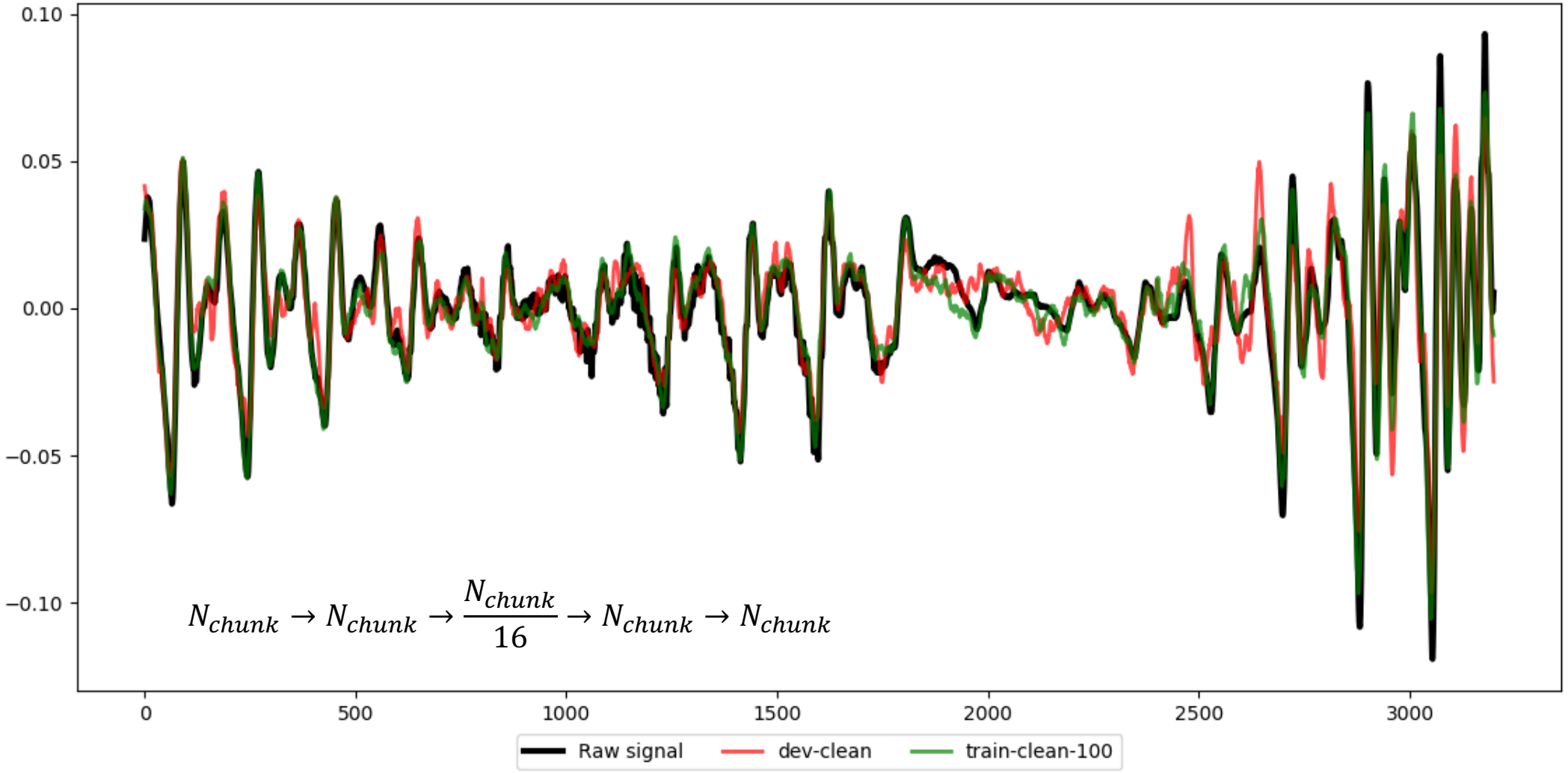
Training Experiment on Large Corpus

- 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
  - Hardware: NVIDIA GeForce GTX 770M
  - 45099 mini-batches per epoch
  - ~ 3.1hrs per epoch

Training on Large Corpus

Speaker 2803
200ms audio sample (4 chunks)



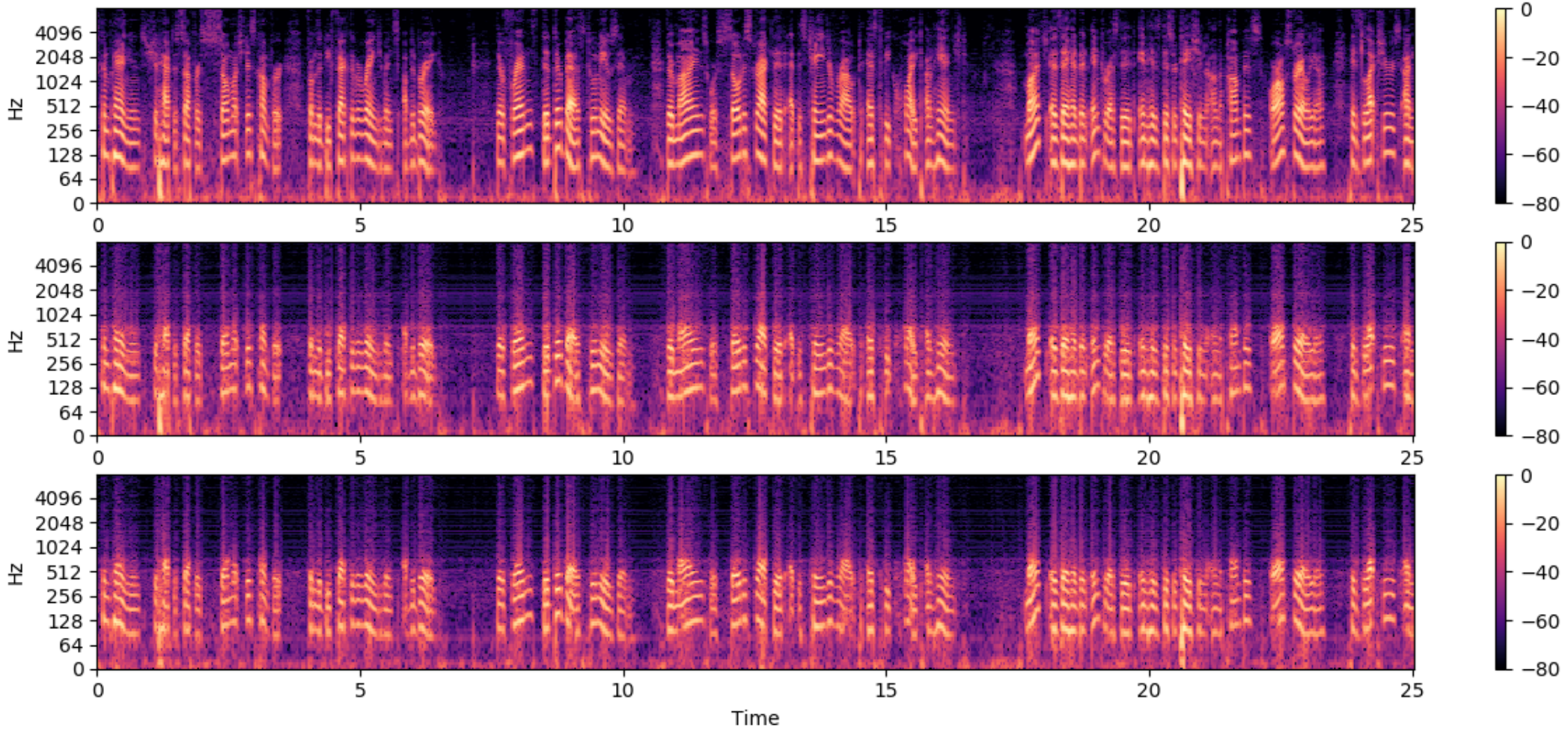
Training on Large Corpus

Speaker 2803
25s audio sample, log-spectrogram

Ground Truth

dev-clean 100 epochs

train-clean-100 10 epochs



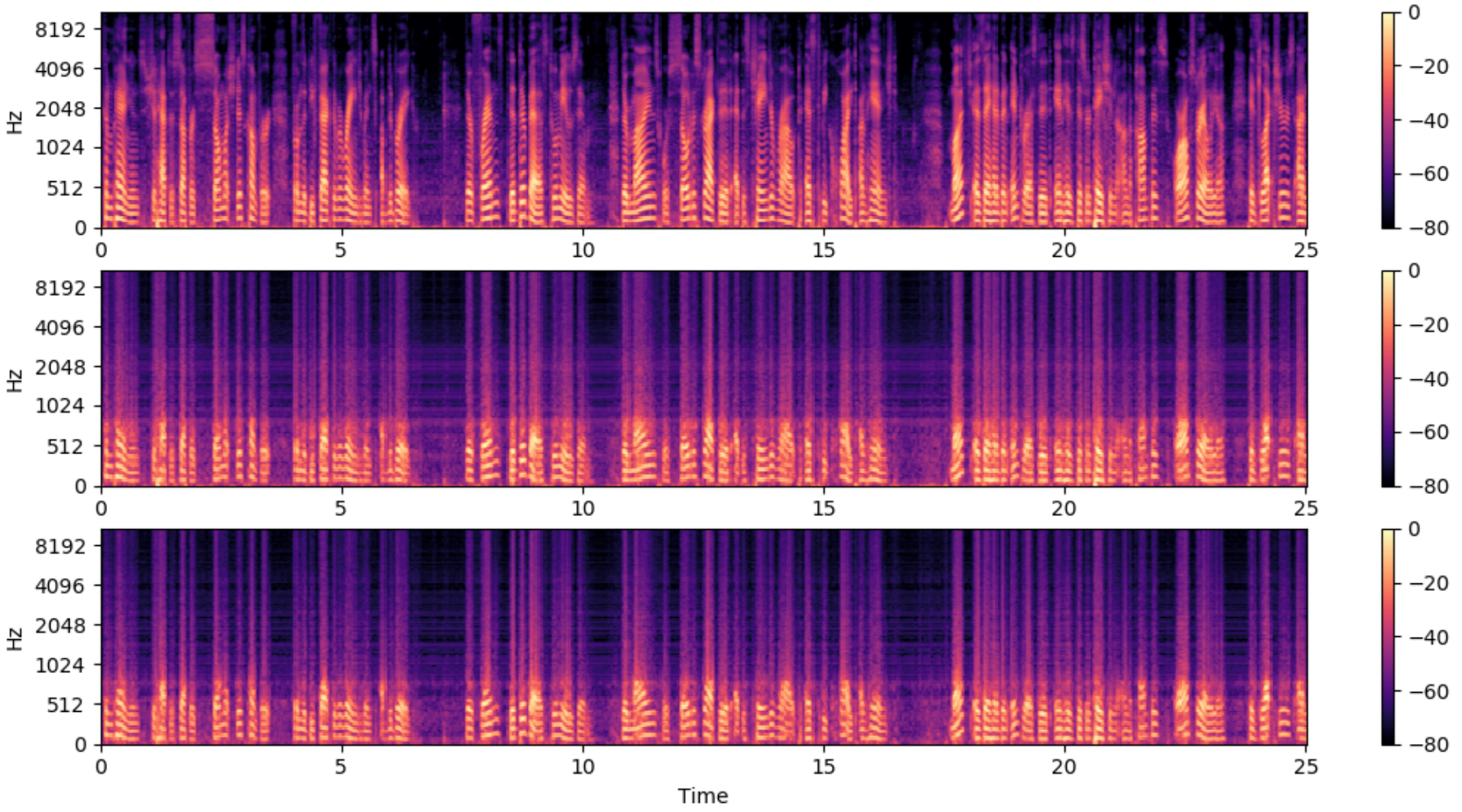
Training on Large Corpus

Speaker 2803
25s audio sample, mel-spectrogram

Ground Truth

dev-clean 100 epochs

train-clean-100 10 epochs



Training on Large Corpus - observations

- The networks trained on dev-clean and train-clean-100 produce almost identical reconstructions when viewed in the spectrograms.
- It should be noted that training had not yet converged on the large corpus after 10 epochs.
- It will be interesting to see if we can achieve high quality reconstructions with more severe latent size restrictions by employing deeper networks (7 or more layers) and training on the large corpus.