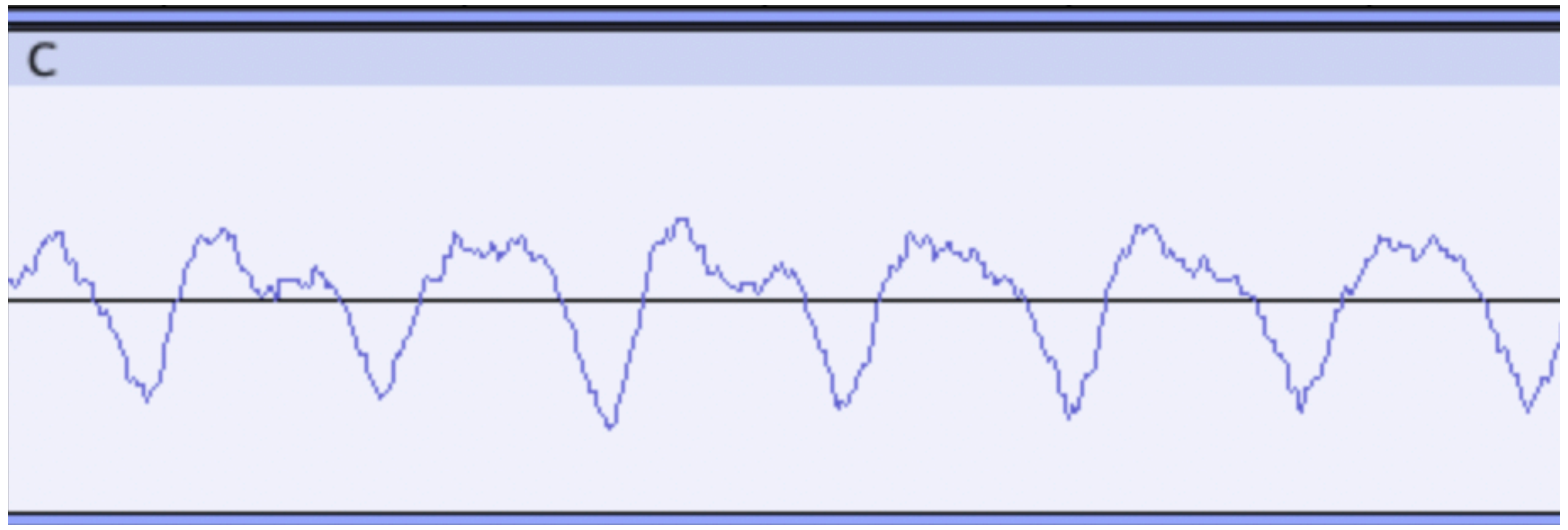
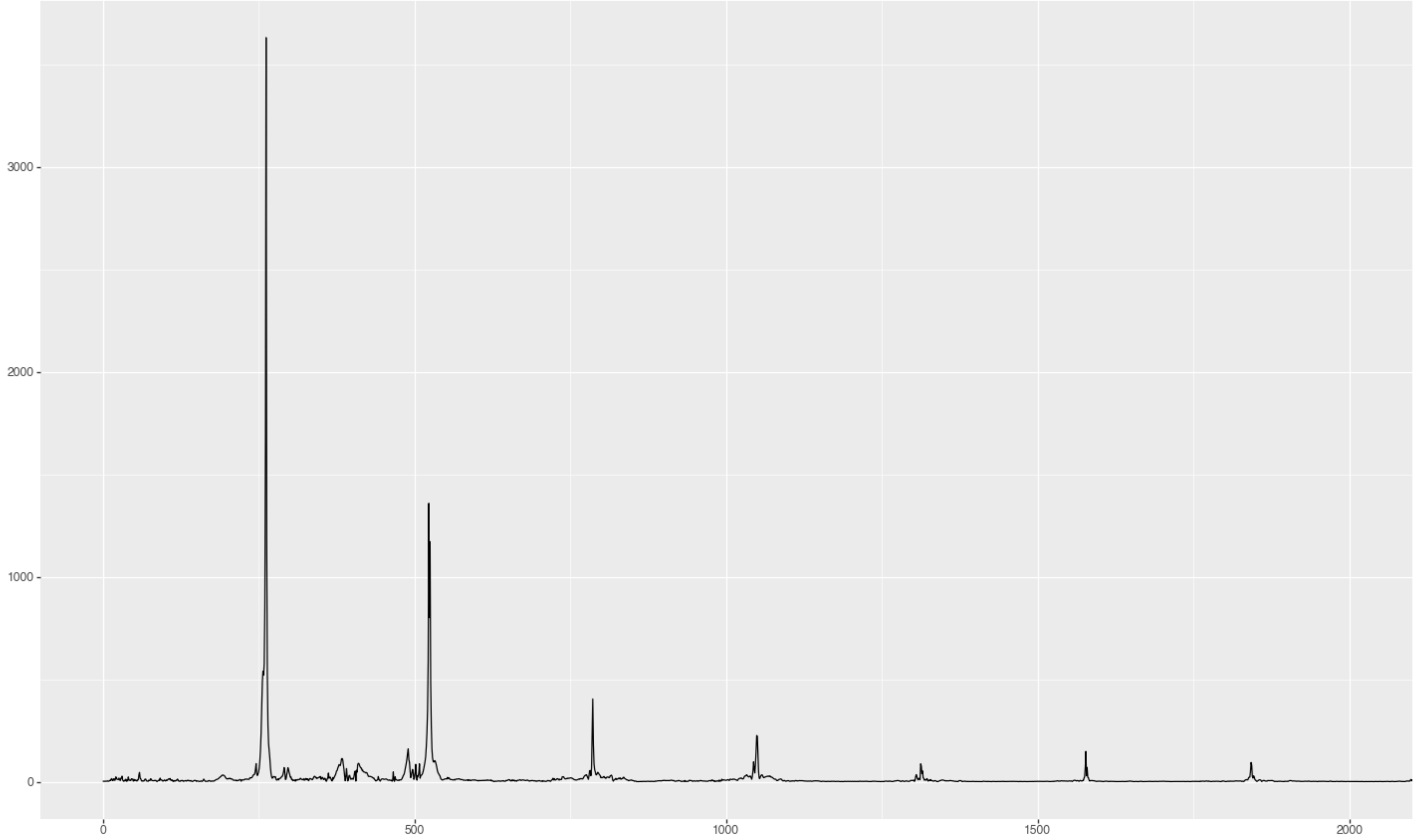
Neural Networks for Speech

Ling 282/482: Deep Learning for Computational Linguistics
C.M. Downey
Fall 2025

Last time

- Introduction to **acoustic data**
- Raw sound data known as the **waveform**
  - Just **amplitude across time** ("time domain" data)
- Apply the **Fourier Transform** to get the **spectrum** of component frequencies instead ("frequency domain")

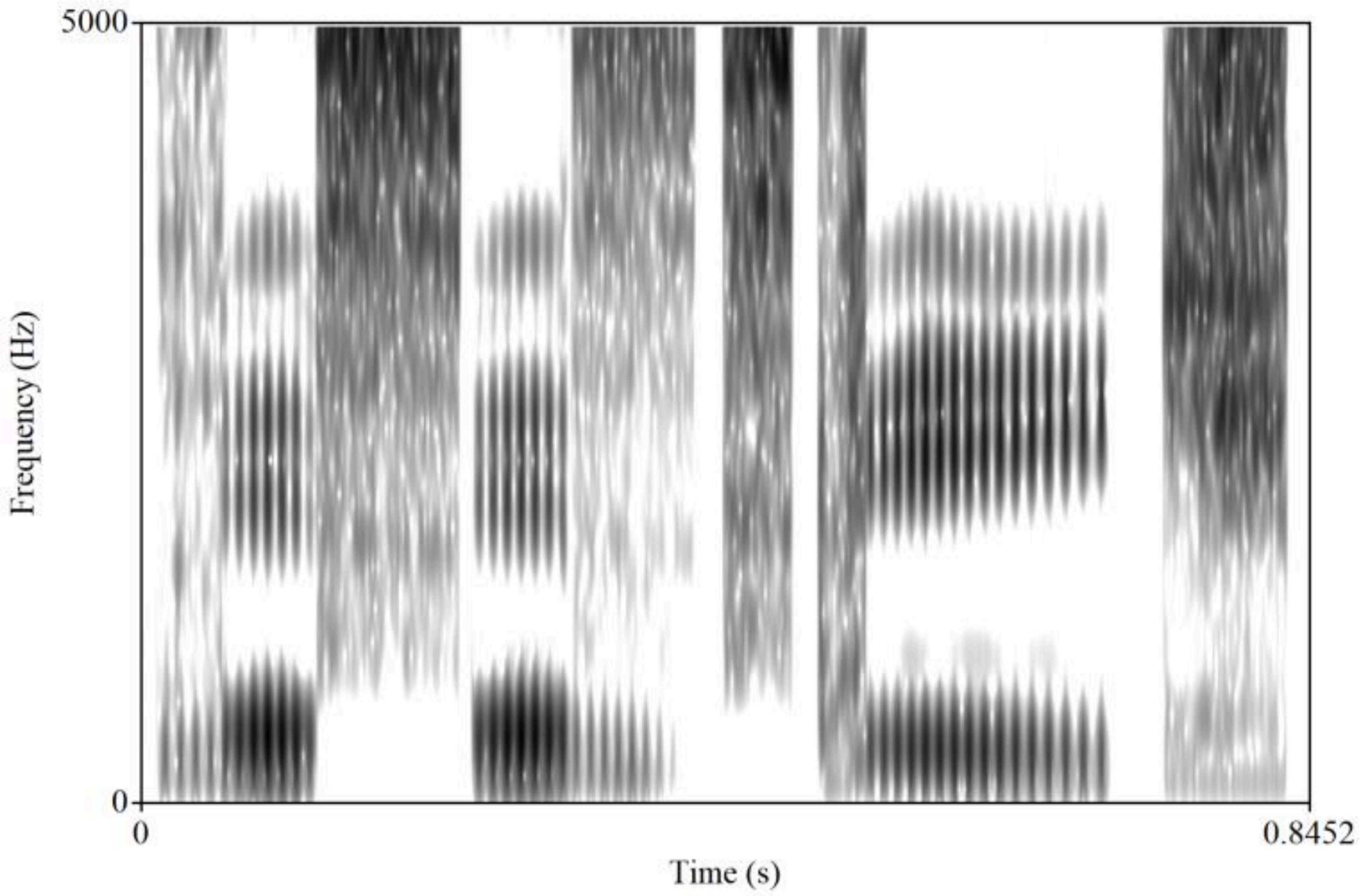


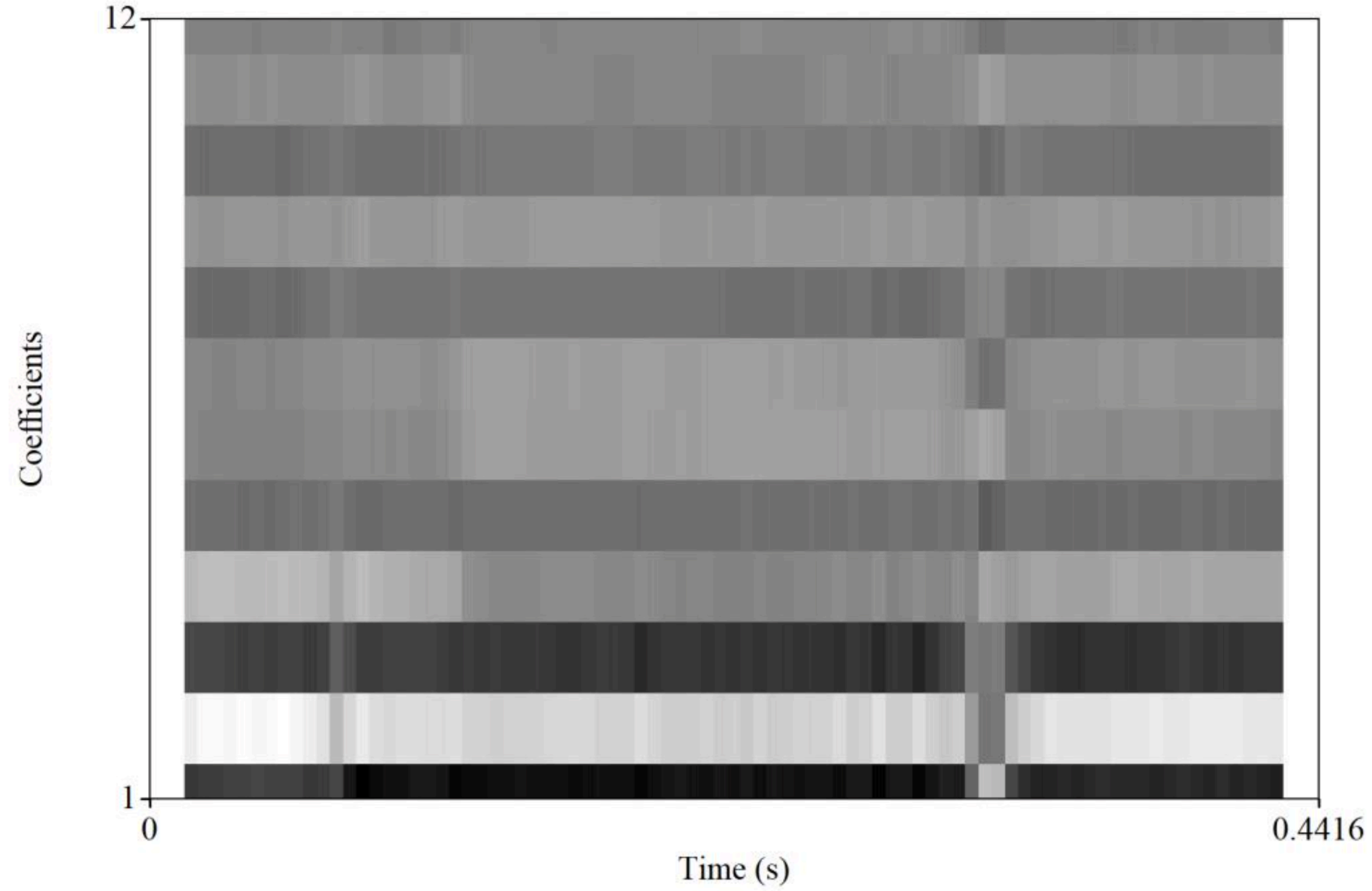


Additional signal transformations

MFCCs

- The full FT spectrum is often re-sampled into **Mel Frequency Cepstral Coefficients (MFCCs)**
  - This is fairly advanced signal processing
  - The main point is to **reduce the dimensionality** of the spectrum
  - Can be thought of as a **compression** of the full spectrum
- Traditionally used as **input to machine learning models**

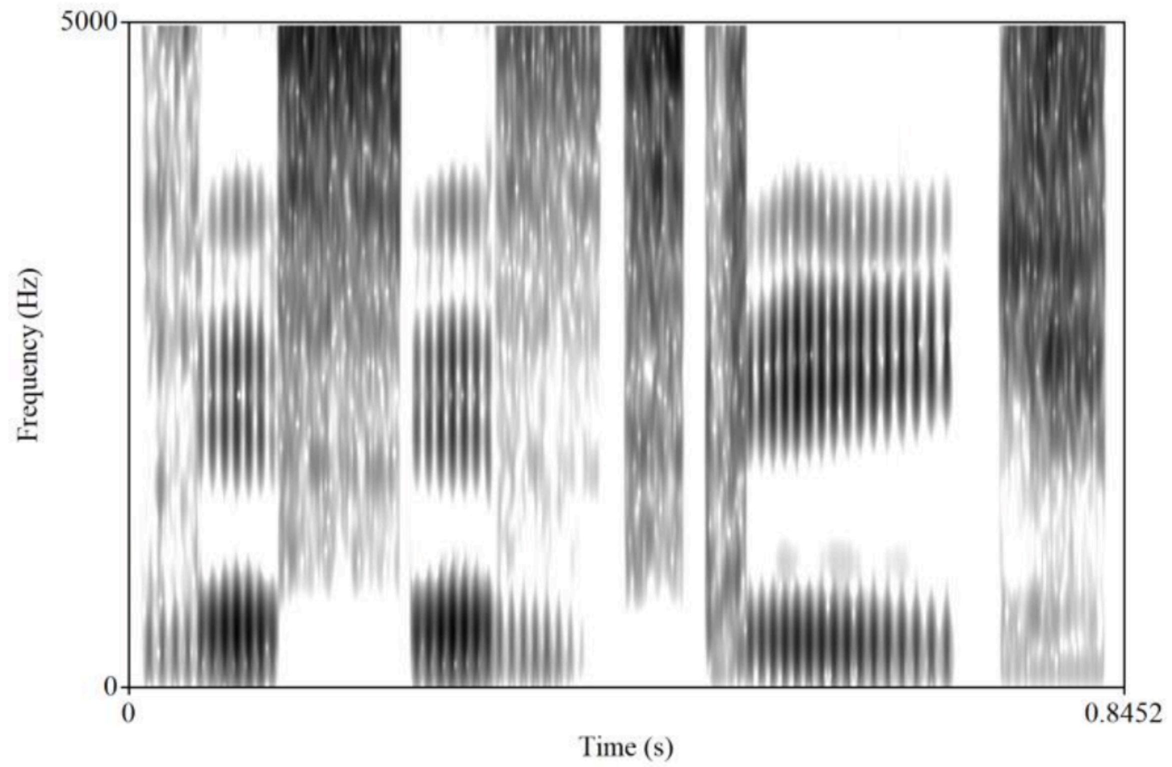


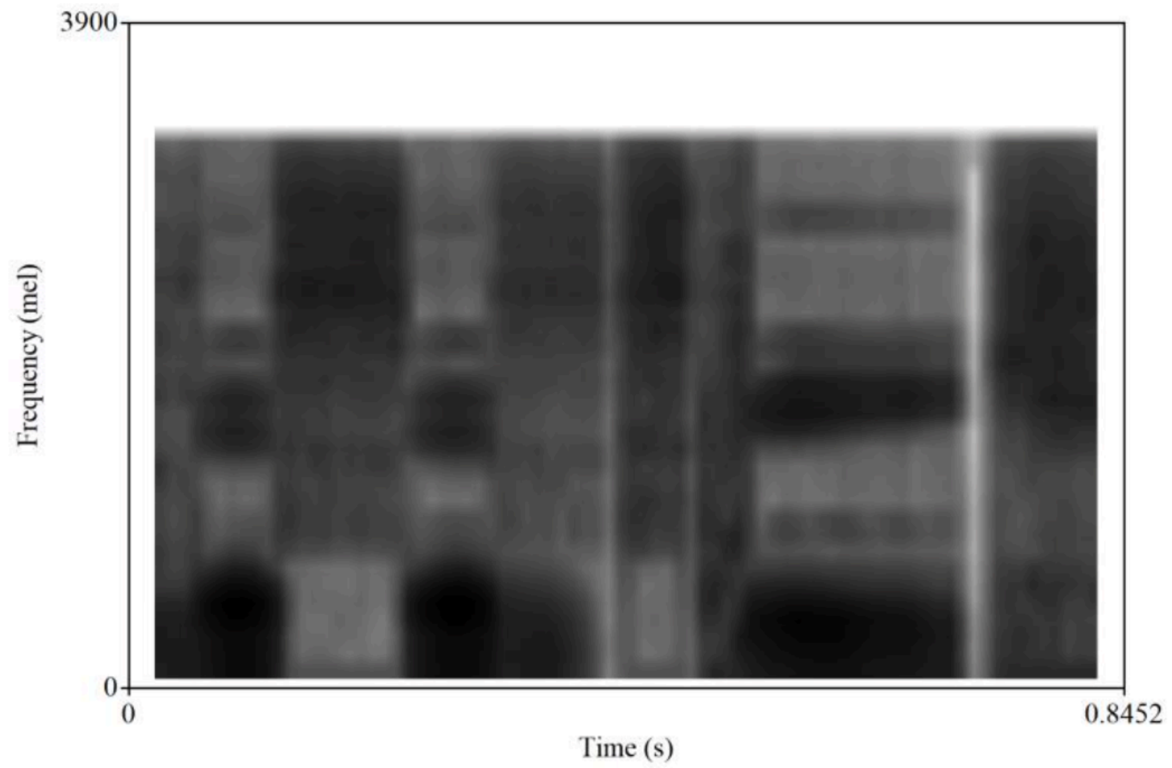


Log Mel Spectrogram

- Spectrogram may also be re-sampled according to the **(Log) Mel Scale**, but not reduced to MFCCs
  - The result called a **Log Mel Spectrogram**
  - Has **more information** than MFCCs
- Popular for **Neural Networks**, which don't need so much feature extraction in the input

Standard spectrogram





Mel spectrogram

Neural Modeling Overview

Speech Encoder-Decoder

- Standard task for speech processing is called **Automatic Speech Recognition (ASR)**
  - Essentially: **speech-to-text**
- Neural approaches have typically used an **Encoder-Decoder** model
  - Encoder learns how to **usefully represent speech signal**
  - Decoder outputs the **corresponding text sequence**

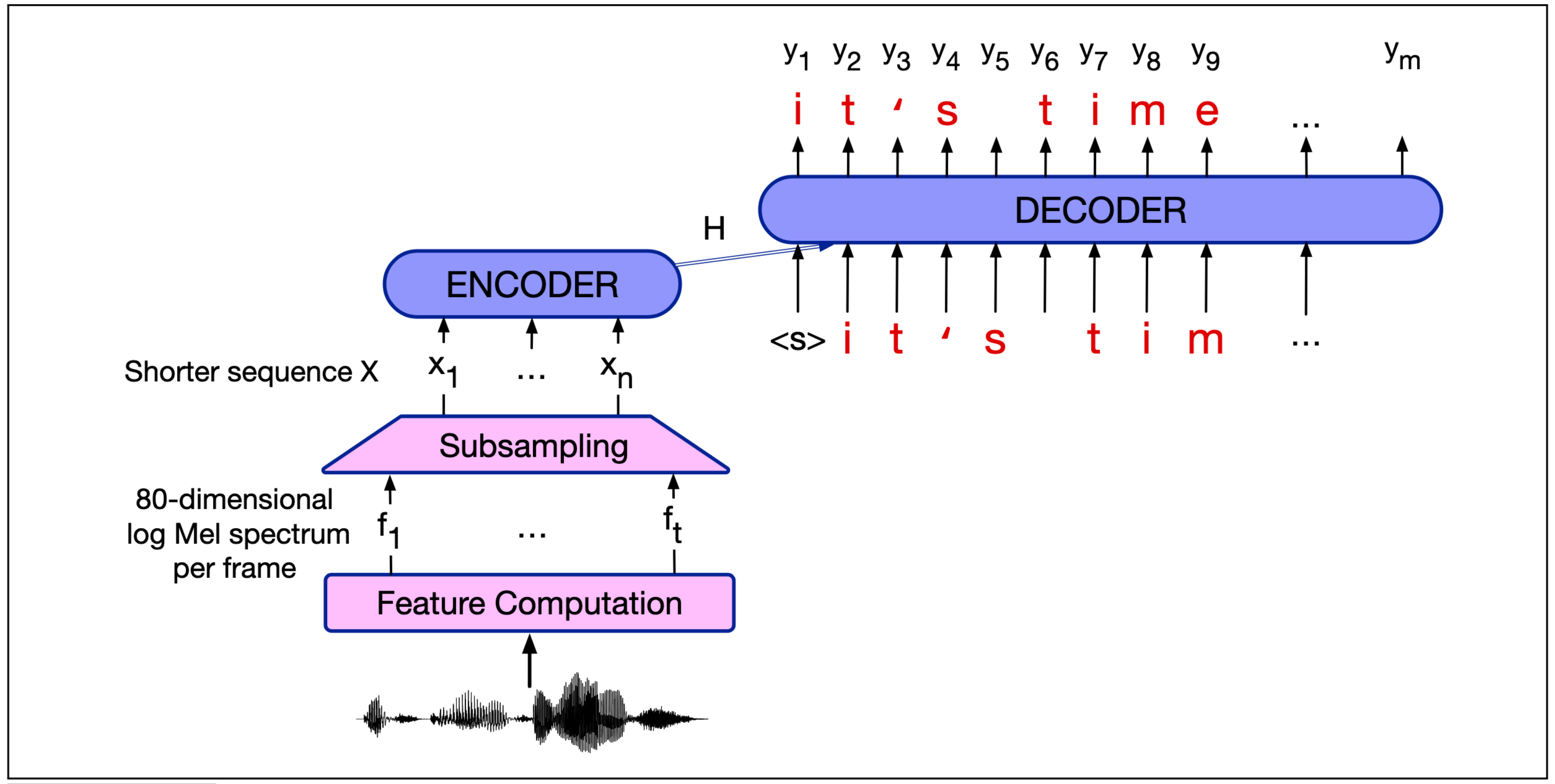


Figure 15.5 Schematic architecture for an encoder-decoder speech recognizer.

Speech Encoder-Decoder

- The main bodies of these models are often **familiar architectures** like the Transformer
- However, each has **additional components** to handle the more complex ASR task
- Almost all encoders use **Convolutional Neural Networks (CNNs)**

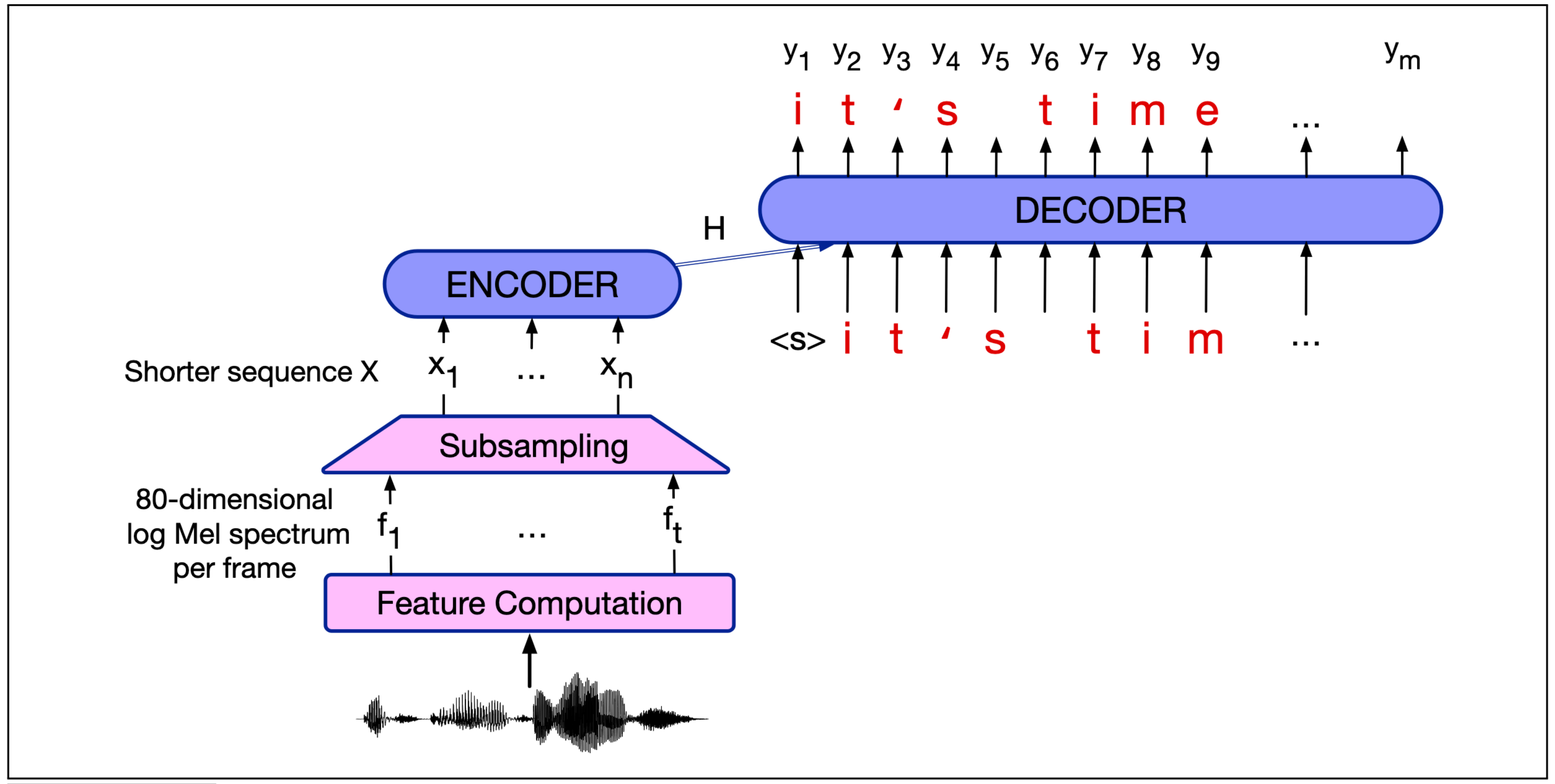


Figure 15.5 Schematic architecture for an encoder-decoder speech recognizer.

Convolutional Neural Networks

- Key intuition: a **"sliding" filter** called a **kernel**
  - Extracts information from **contiguous regions** of the input
  - In audio, filter is over a **span of time**
  - In images, an **area of the image**
- Essentially, take the **dot product** of the kernel with the input window
- Name comes from math concept of Convolution (which is slightly different)

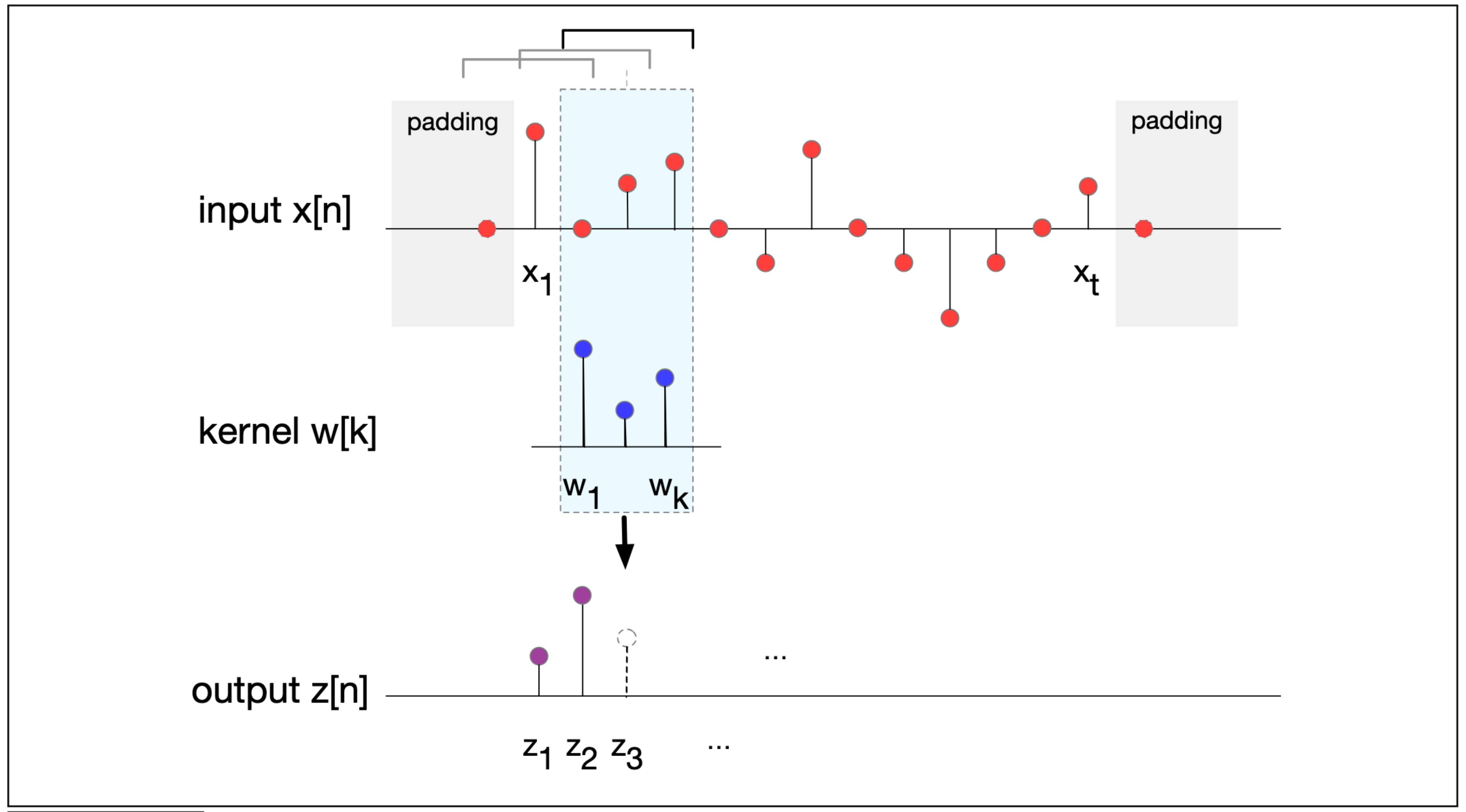


Figure 15.3 A schematic view of convolution with a kernel (filter) width of 3, and with a padding of 1, showing a zero value added at the start and end of the signal. The (already flipped) kernel is walked across the input, and the output at each frame $\mathbf{z}_i$ is the dot product of the kernel with the input in the window. The figure shows the computation of $\mathbf{z}_3$.

Speech CNN Layer

- The kernel contains the **learned weights**
- Kernel is **"slid" across time**
  - Has an option called **stride**, which controls how far to move in each step
  - Higher stride → output sequence is **shorter** than input
- Each embedding dimension usually called a **channel**

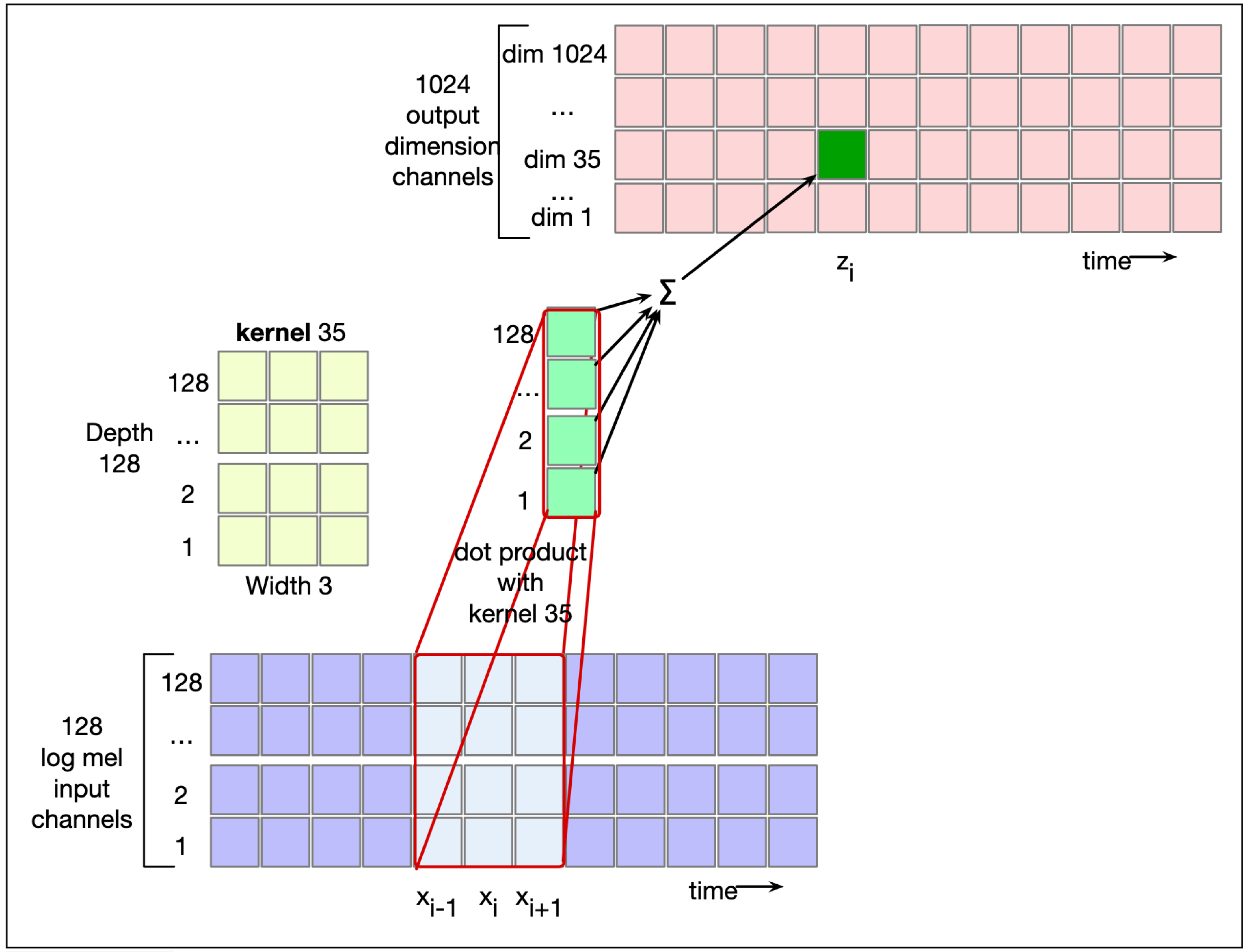


Figure 15.4 A schematic view of a convolutional net with 128 input channels and 1024 output channels. We see how at time point i one of the 1024 kernels ("kernel 35", each of depth 128 and width 3) is dot-product-ed with (each of) the 128 log mel spectrum input vectors, and then summed to produce a single value for one dimension of the output embedding at time i.

Speech CNN Layer

- For kernel length 3, 128 channels:
  - Kernel becomes size **128x3**
  - However, we still **treat this like a dot product:** sum is taken over whole 128x3 area
  - Another way to think about it: treat the input area and kernel as **single vectors**, and take the dot product
  - Gives the value for a **single output channel!**
  - **Each output channel** gets a **different kernel**

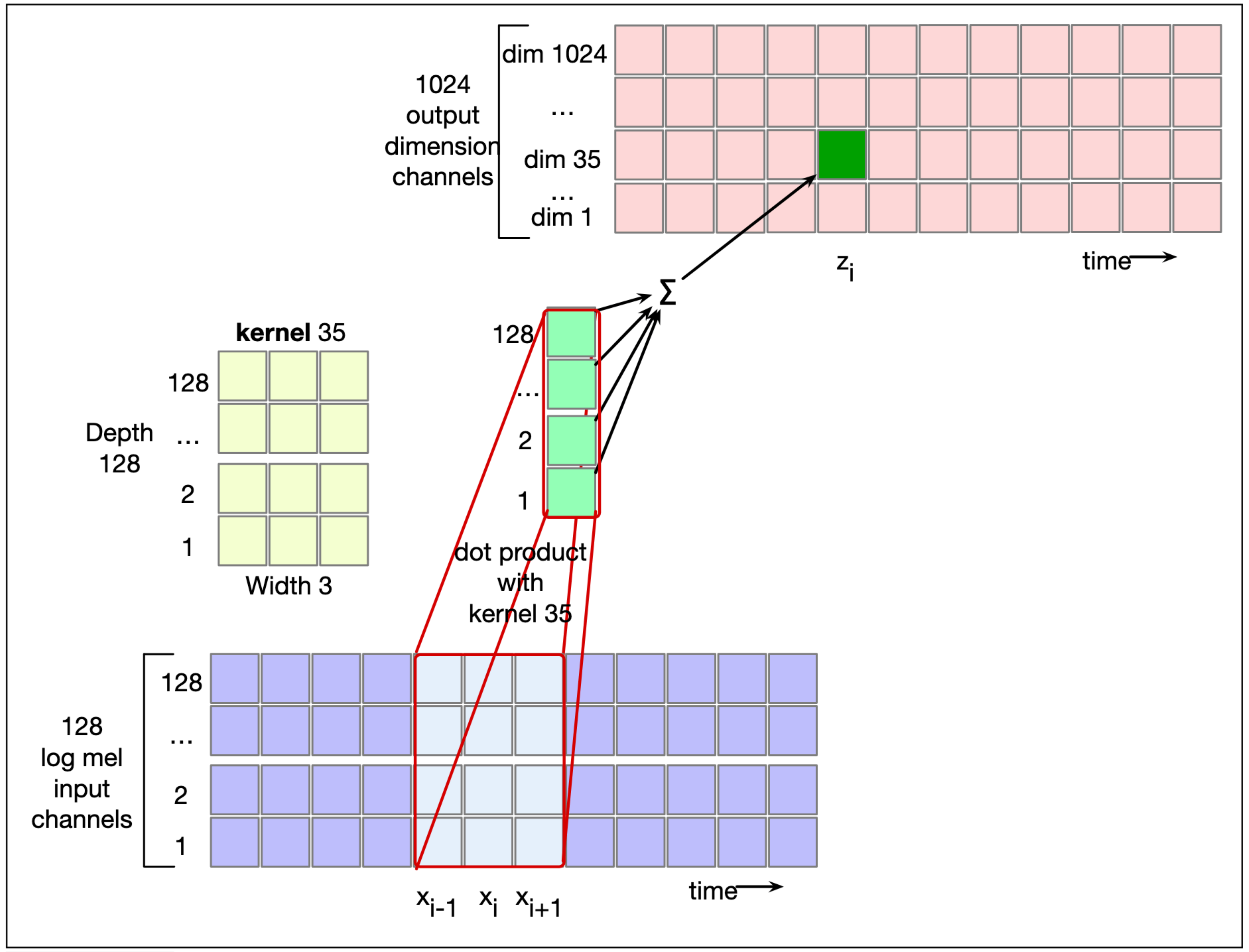


Figure 15.4 A schematic view of a convolutional net with 128 input channels and 1024 output channels. We see how at time point *i* one of the 1024 kernels ("kernel 35", each of depth 128 and width 3) is dot-product-ed with (each of) the 128 log mel spectrum input vectors, and then summed to produce a single value for one dimension of the output embedding at time *i*.

CTC

Why we need CTC

- In audio, each input timestep corresponds to e.g. a **25ms window**
    - Speech sounds might stretch across **multiple time-steps**
    - Text transcription **doesn't tell us** how to align characters to time!
- How do we figure out this alignment?

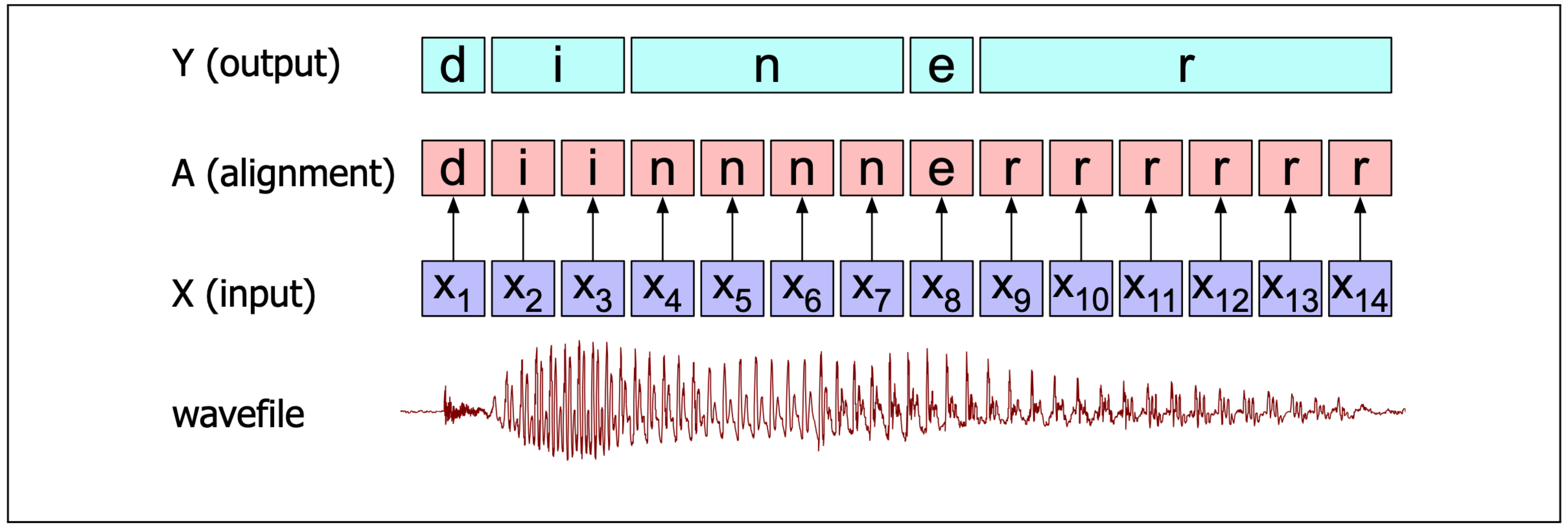


Figure 15.12 A naive algorithm for collapsing an alignment between input and letters.

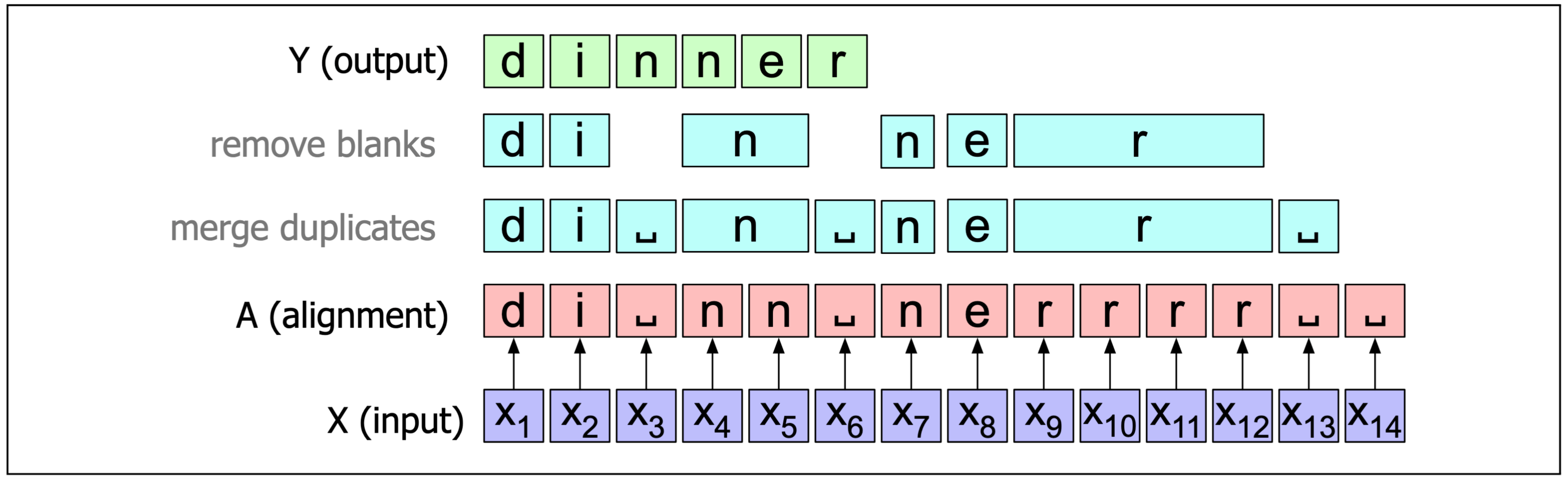


Figure 15.13 The CTC collapsing function B, showing the space blank character ␣; repeated (consecutive) characters in an alignment A are removed to form the output Y.

CTC

- CTC: **Connectionist Temporal Classification**
  - Essentially, a way to **align symbols (characters)** to output time-steps
- Uses **blank** symbol for two purposes:
  - To represent **silence**
  - To **break up** duplicate characters
- Otherwise **merges duplicates**

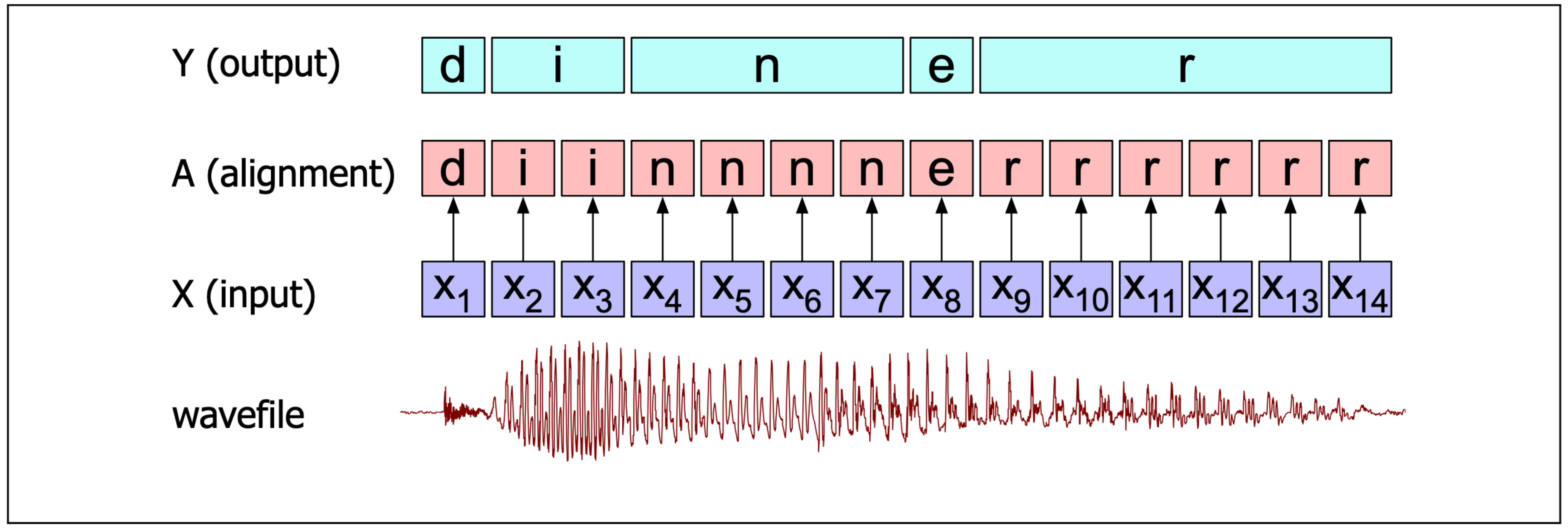


Figure 15.12 A naive algorithm for collapsing an alignment between input and letters.

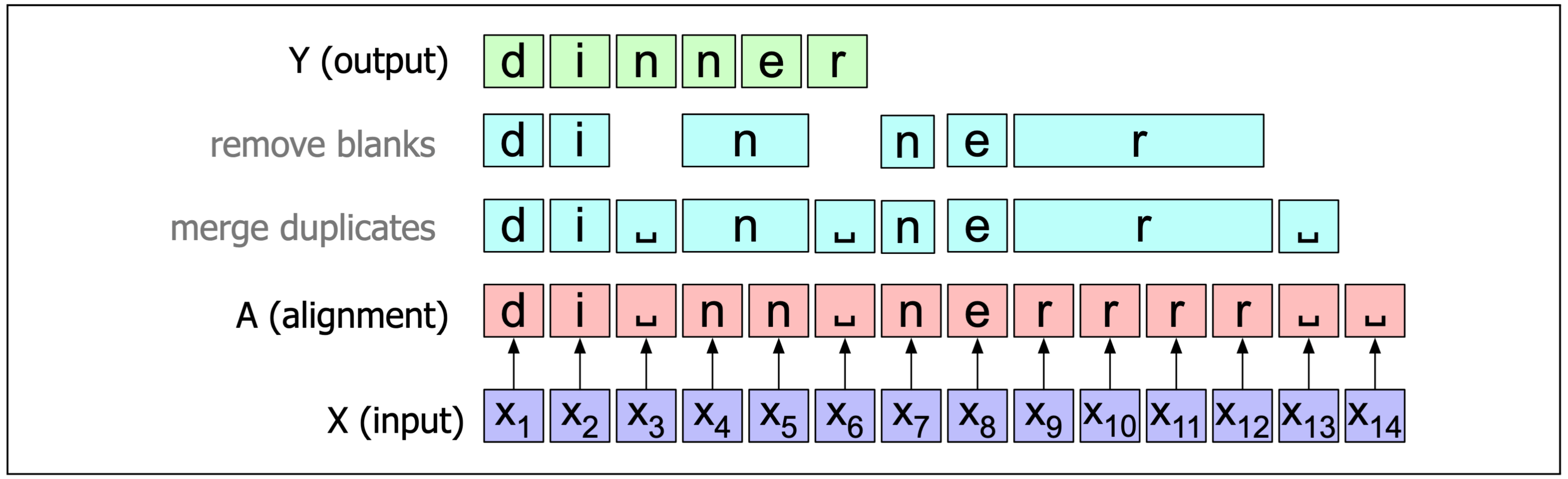


Figure 15.13 The CTC collapsing function B, showing the space blank character ␣; repeated (consecutive) characters in an alignment A are removed to form the output Y.

CTC

- Model still learns to output **one character per timestep**, including the special symbols
- **Multiple alignments** may be compatible with the same output
    - Because of this, calculating **loss** during training requires an approximation of summing over all alignments

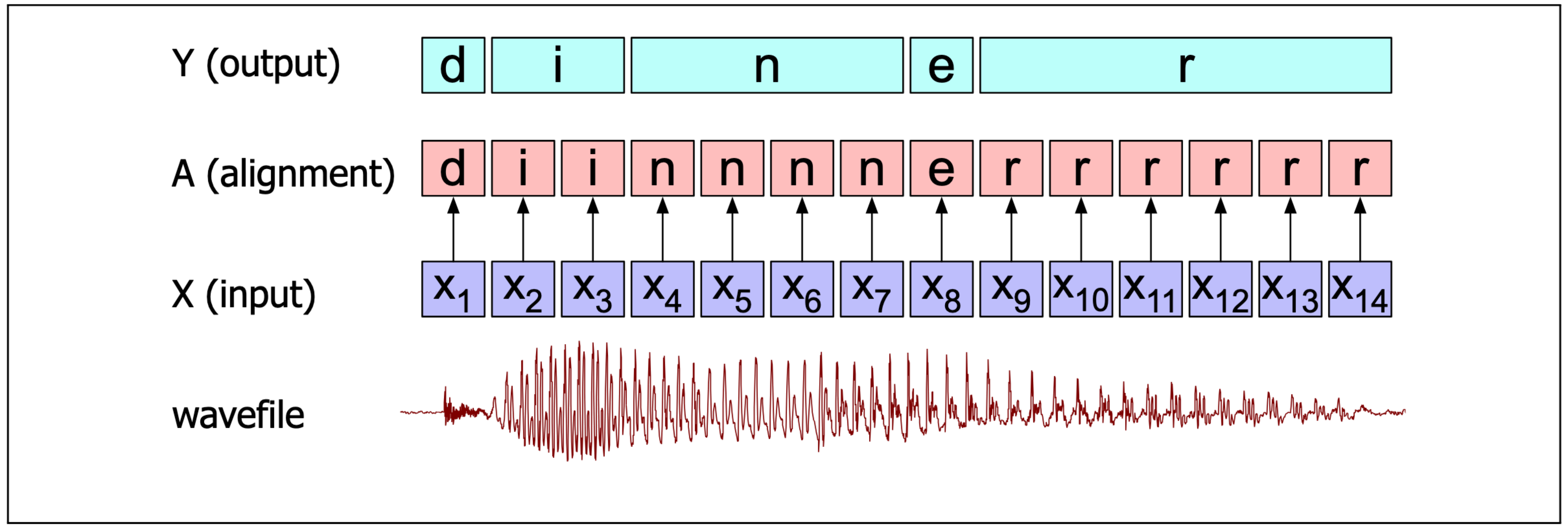


Figure 15.12 A naive algorithm for collapsing an alignment between input and letters.

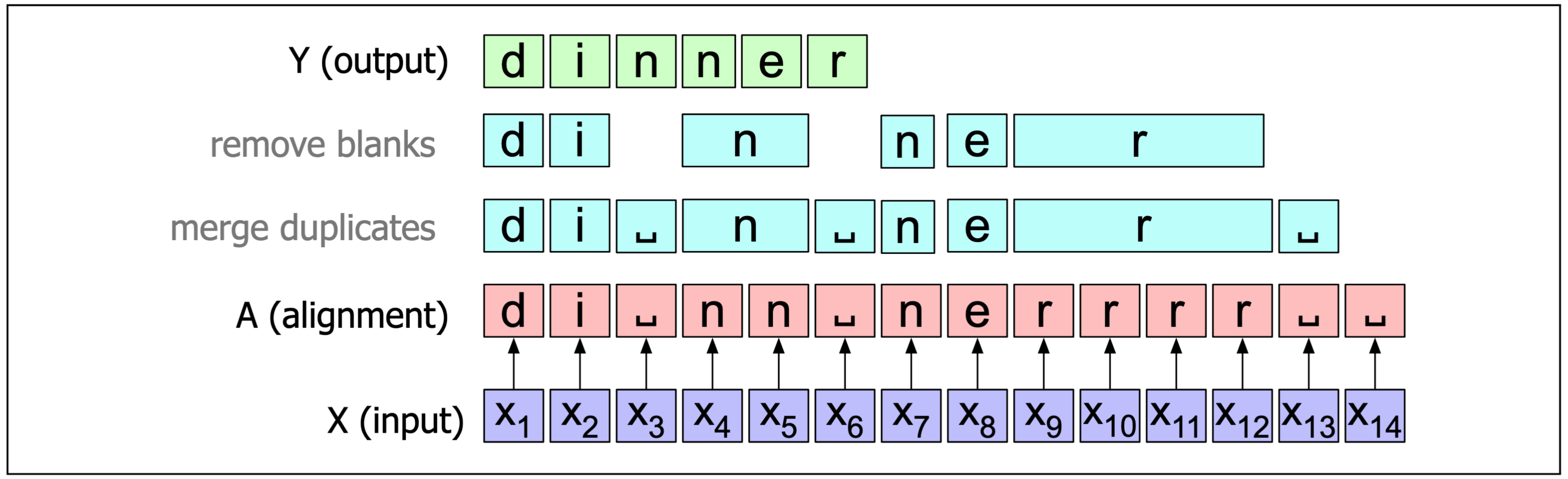


Figure 15.13 The CTC collapsing function B, showing the space blank character ␣; repeated (consecutive) characters in an alignment A are removed to form the output Y.

LM Integration

- ASR is a **difficult task**
  - Learns to output what seems **most likely given audio**, but might not be sensible language
- Solution: incorporate an **auxiliary language model**
  - LM trained **just on text**
  - Biases system towards **probable sequences of words**

ASR Model Overview

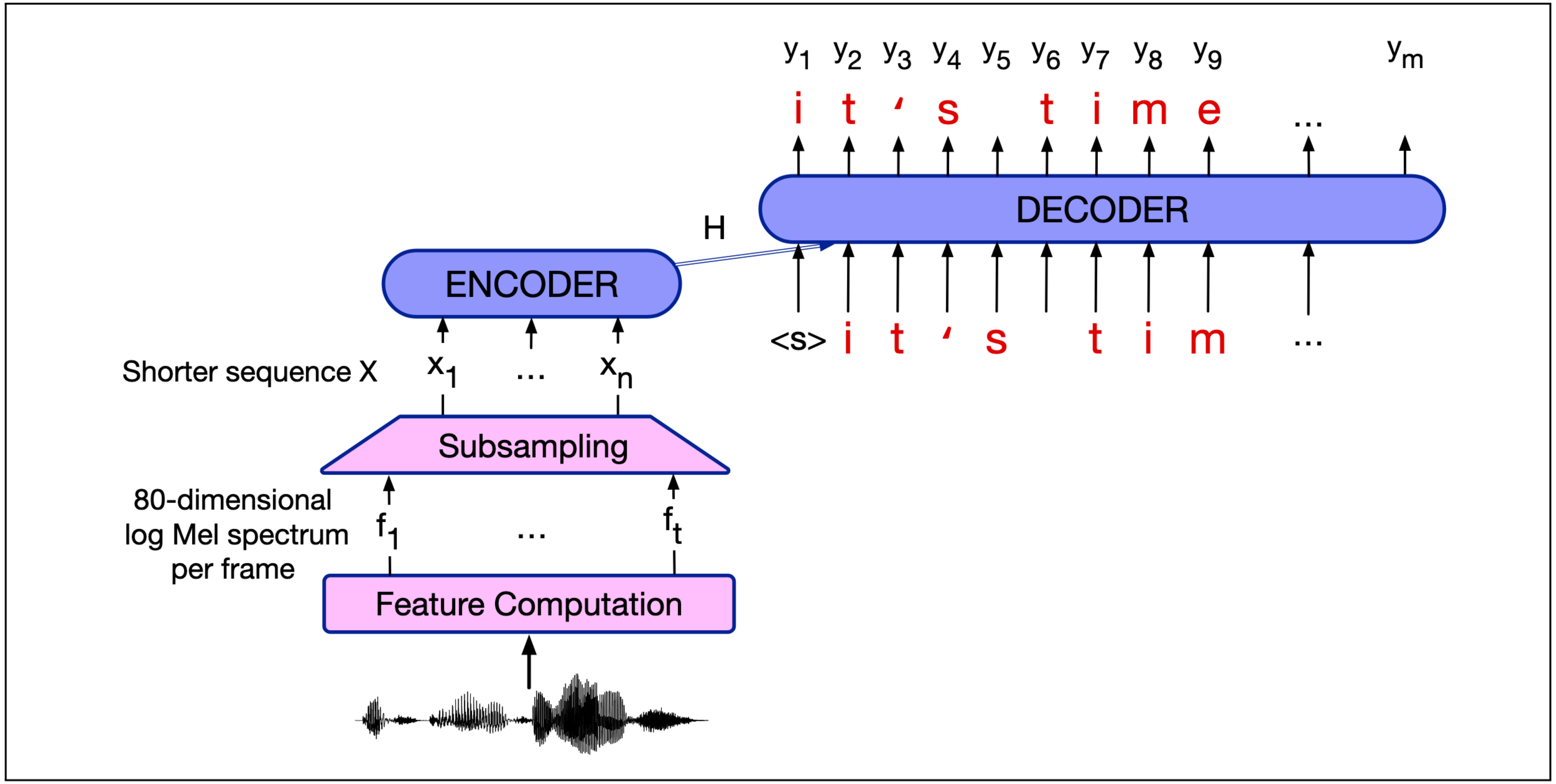
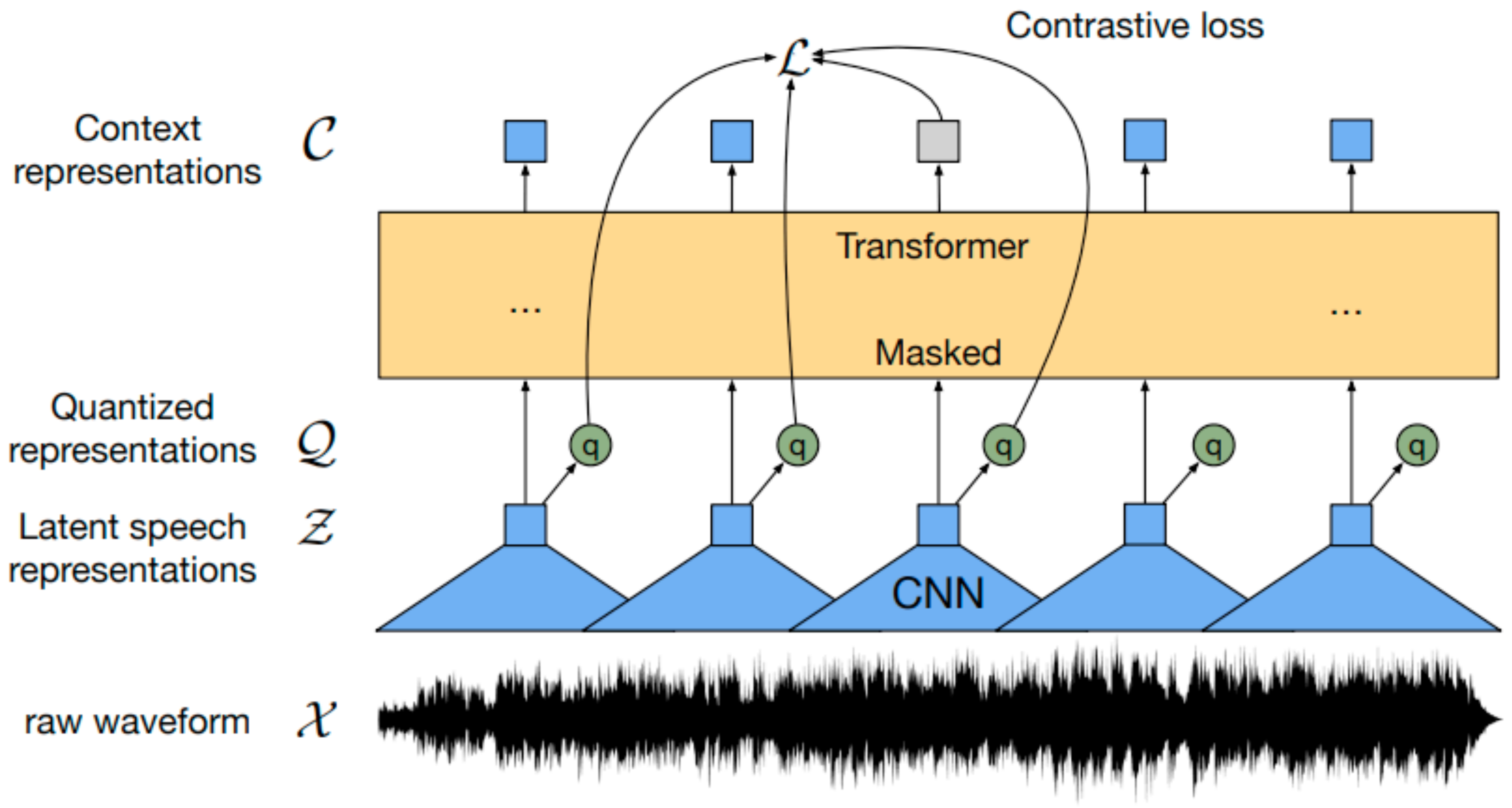


Figure 15.5 Schematic architecture for an encoder-decoder speech recognizer.

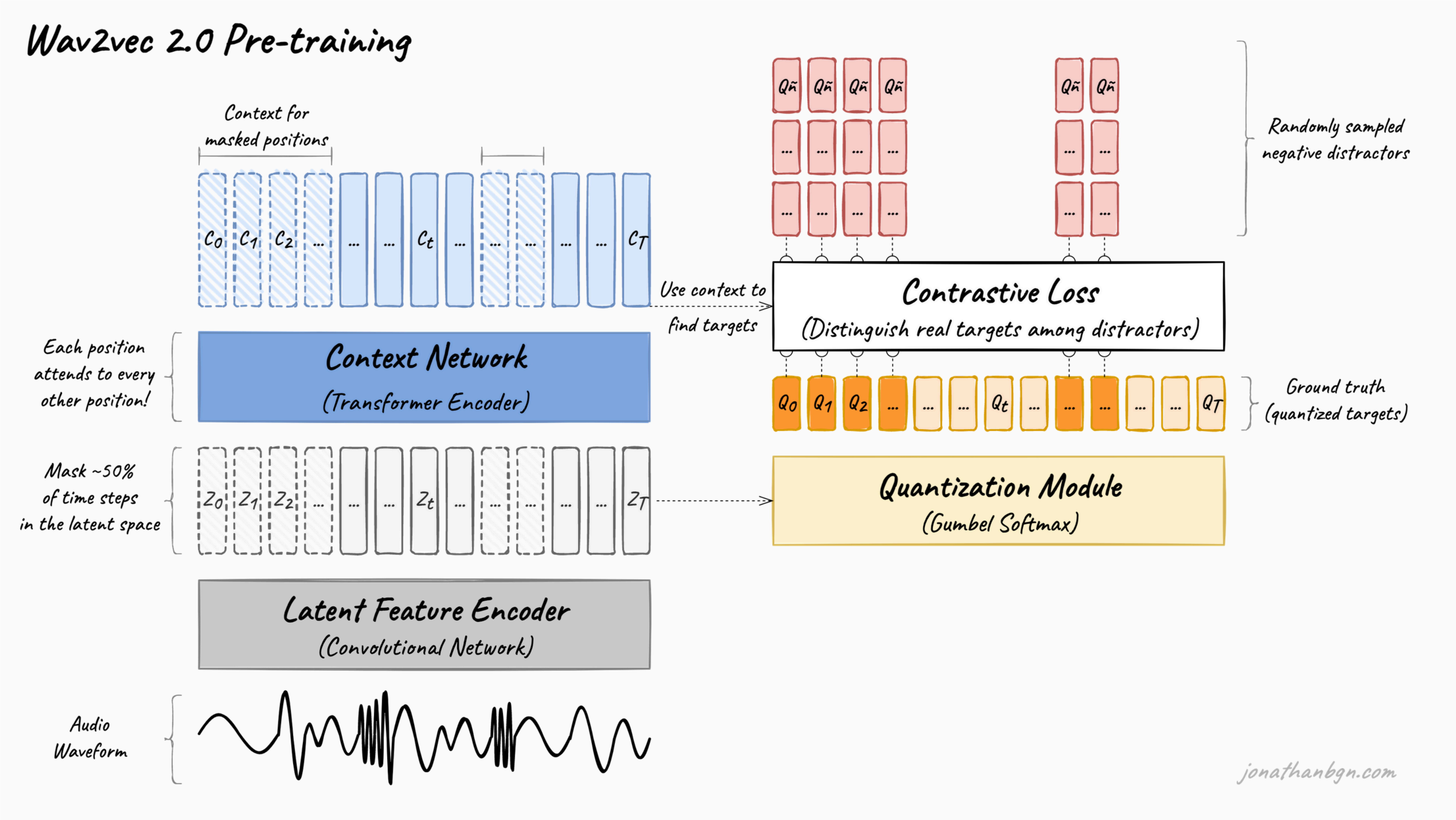
Self-supervised Models

Speech Self-Supervision

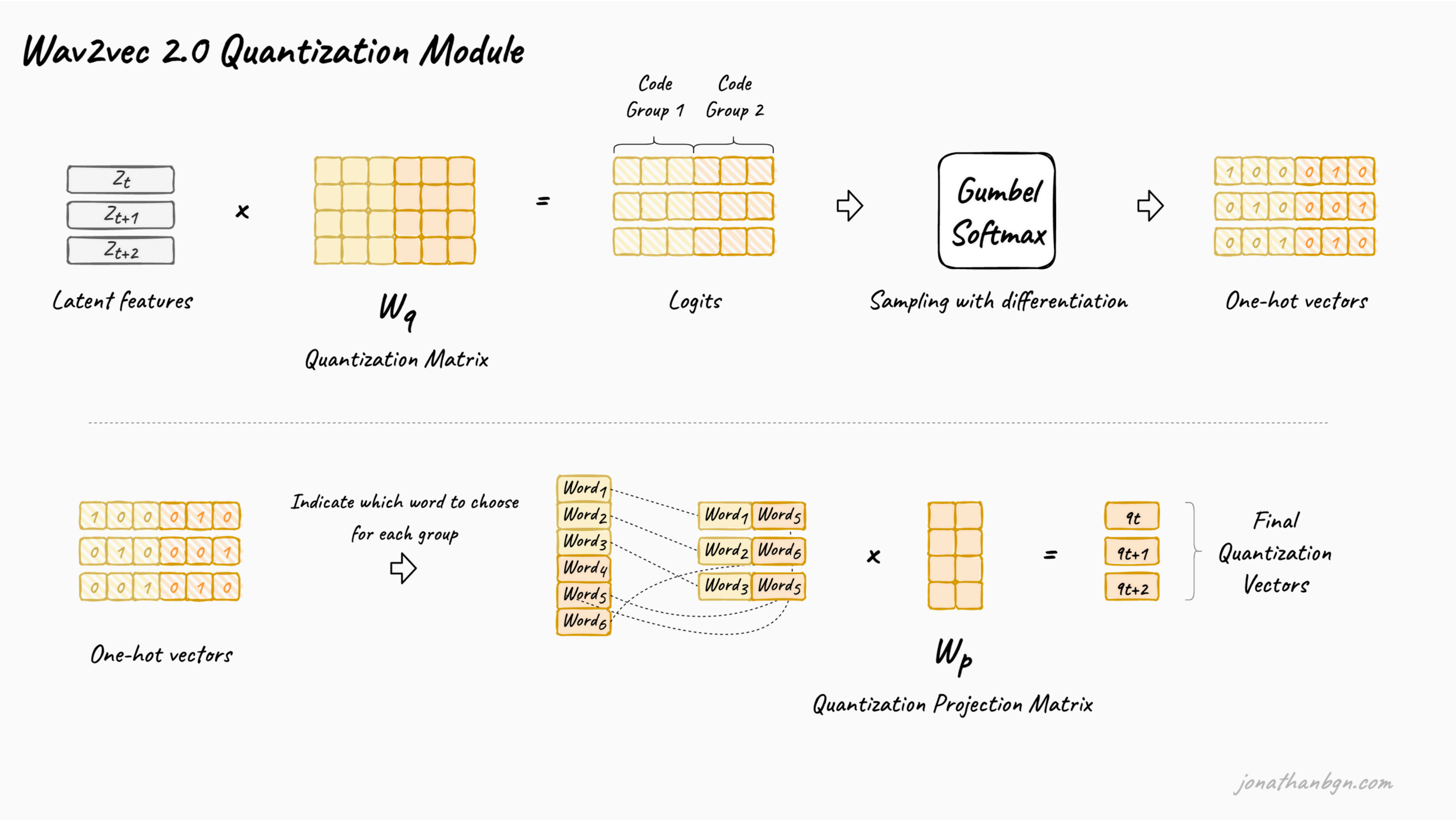
- How can we apply **pre-training** to speech models?
- Transcribed audio (needed for ASR training) is **rare?**
- How can we pre-train on **audio only?**
    - Models like **wav2vec** have attempted to do just this
- Visualizations on next slides from [this helpful blog post]



wav2vec Overview



wav2vec Quantization



wav2vec Contrastive Loss