The state of the art (1 of 2)

- Class slides

What you should already know

- Neural networks perform regression from input to output by **learning intermediate representations**

- Neural networks are made from basic building blocks: **layers**

- There are different types of layer, e.g.,

  - fully-connected (FC)

  - convolutional

  - LSTM, ... etc

- Case study from last week: Tacotron 2

NATURAL TTS SYNTHESIS BY CONDITIONING WAVENET ON MEL SPECTROGRAM PREDICTIONS

Jonathan Shen[1], Ruoming Pang[1], Ron J. Weiss[1], Mike Schuster[1], Navdeep Jaitly[1], Zongheng Yang[2],*
Zhifeng Chen[1], Yu Zhang[1], Yuxuan Wang[1], RJ Skerry-Ryan[1], Rif A. Saurous[1], Yannis Agiomyrgiannakis[1],
and Yonghui Wu[1]

[1]Google, Inc., [2]University of California, Berkeley,
{jonathanasdf,rpang,yonghui}@google.com

ABSTRACT

This paper describes Tacotron 2, a neural network architecture for speech synthesis directly from text. The system is composed of a recurrent sequence-to-sequence feature prediction network that maps character embeddings to mel-scale spectrograms, followed by a modified WaveNet model acting as a vocoder to synthesize time-domain waveforms from those spectrograms. Our model achieves a mean opinion score (MOS) of 4.53 comparable to a MOS of 4.58 for professionally recorded speech. To validate our design choices, we present ablation studies of key components of our system and evaluate the impact of using mel spectrograms as the conditioning input to WaveNet instead of linguistic, duration, and F_0 features. We further show that using this compact acoustic intermediate representation allows for a significant reduction in the size of the WaveNet architecture.

Index Terms— Tacotron 2, WaveNet, text-to-speech

1. INTRODUCTION

Generating natural speech from text (text-to-speech synthesis, TTS) remains a challenging task despite decades of investigation [1]. Over time, different techniques have dominated the field. Concatenative synthesis with unit selection, the process of stitching small units of pre-recorded waveforms together [2, 3] was the state-of-the-art for many years. Statistical parametric speech synthesis [4, 5, 6, 7], which directly generates smooth trajectories of speech features to be synthesized by a vocoder, followed, solving many of the issues that concatenative synthesis had with boundary artifacts. However, the audio produced by these systems often sounds muffled and unnatural compared to human speech.

WaveNet [8], a generative model of time domain waveforms, produces audio quality that begins to rival that of real human speech and is already used in some complete TTS systems [9, 10, 11]. The inputs to WaveNet (linguistic features, predicted log fundamental frequency (F_0), and phoneme durations), however, require significant domain expertise to produce, involving elaborate text-analysis systems as well as a robust lexicon (pronunciation guide).

Tacotron [12], a sequence-to-sequence architecture [13] for producing magnitude spectrograms from a sequence of characters, simplifies the traditional speech synthesis pipeline by replacing the production of these linguistic and acoustic features with a single neural network trained from data alone. To vocode the resulting magnitude spectrograms, Tacotron uses the Griffin-Lim algorithm [14] for phase estimation, followed by an inverse short-time Fourier transform. As

the authors note, this was simply a placeholder for future neural vocoder approaches, as Griffin-Lim produces characteristic artifacts and lower audio quality than approaches like WaveNet.

In this paper, we describe a unified, entirely neural approach to speech synthesis that combines the best of the previous approaches: a sequence-to-sequence Tacotron-style model [12] that generates mel spectrograms, followed by a modified WaveNet vocoder [10, 15]. Trained directly on normalized character sequences and corresponding speech waveforms, our model learns to synthesize natural sounding speech that is difficult to distinguish from real human speech.

Deep Voice 3 [11] describes a similar approach. However, unlike our system, its naturalness has not been shown to rival that of human speech. Char2Wav [16] describes yet another similar approach to end-to-end TTS using a neural vocoder. However, they use different intermediate representations (traditional vocoder features) and their model architecture differs significantly.

2. MODEL ARCHITECTURE

Our proposed system consists of two components, shown in Figure 1: (1) a recurrent sequence-to-sequence feature prediction network with attention which predicts a sequence of mel spectrogram frames from an input character sequence, and (2) a modified version of WaveNet which generates time-domain waveform samples conditioned on the predicted mel spectrogram frames.

2.1. Intermediate Feature Representation

In this work we choose a low-level acoustic representation: mel-frequency spectrograms, to bridge the two components. Using a representation that is easily computed from time-domain waveforms allows us to train the two components separately. This representation is also smoother than waveform samples and is easier to train using a squared error loss because it is invariant to phase within each frame.

A mel-frequency spectrogram is related to the linear-frequency spectrogram, i.e., the short-time Fourier transform (STFT) magnitude. It is obtained by applying a nonlinear transform to the frequency axis of the STFT, inspired by measured responses from the human auditory system, and summarizes the frequency content with fewer dimensions. Using such an auditory frequency scale has the effect of emphasizing details in lower frequencies, which are critical to speech intelligibility, while de-emphasizing high frequency details, which are dominated by fricatives and other noise bursts and generally do not need to be modeled with high fidelity. Because of these properties, features derived from the mel scale have been used as an underlying representation for speech recognition for many decades [17].

---

*Work done while at Google.

FASTPITCH: PARALLEL TEXT-TO-SPEECH WITH PITCH PREDICTION

Adrian Łańcucki

NVIDIA Corporation

ABSTRACT

We present FastPitch, a fully-parallel text-to-speech model based on FastSpeech, conditioned on fundamental frequency contours. The model predicts pitch contours during inference. By altering these predictions, the generated speech can be more expressive, better match the semantic of the utterance, and in the end more engaging to the listener. Uniformly increasing or decreasing pitch with FastPitch generates speech that resembles the voluntary modulation of voice. Conditioning on frequency contours improves the overall quality of synthesized speech, making it comparable to state-of-the-art. It does not introduce an overhead, and FastPitch retains the favorable, fully-parallel Transformer architecture, with over $900\times$ real-time factor for mel-spectrogram synthesis of a typical utterance.

Index Terms— text-to-speech, speech synthesis, fundamental frequency

1. INTRODUCTION

Recent advances in neural text-to-speech (TTS) enabled real-time synthesis of naturally sounding, human-like speech. Parallel models are able to synthesize mel-spectrograms orders of magnitude faster than autoregressive ones, either by relying on external alignments [1], or aligning themselves [2]. TTS models can be conditioned on qualities of speech such as linguistic features and fundamental frequency [3]. The latter has been repeatedly shown to improve the quality of neural, but also concatenative models [4, 5]. Conditioning on F_0 is a common approach to adding singing capabilities [6]. or adapting to other speakers [5].

In this paper we propose FastPitch, a feed-forward model based on FastSpeech that improves the quality of synthesized speech. By conditioning on fundamental frequency estimated for every input symbol, which we refer to simply as a pitch contour, it matches the state-of-the-art autoregressive TTS models. We show that explicit modeling of such pitch contours addresses the quality shortcomings of the plain feed-forward Transformer architecture. These most likely arise from collapsing different pronunciations of the same phonetic units in the absence of enough linguistic information in the textual input alone. Conditioning on fundamental frequency also improves convergence, and eliminates the

need for knowledge distillation of mel-spectrogram targets used in FastSpeech. We would like to note that a concurrently developed FastSpeech 2 [7] describes a similar approach.

Combined with WaveGlow [8], FastPitch is able to synthesize mel-spectrograms over $60\times$ faster than real-time, without resorting to kernel-level optimizations [9]. Because the model learns to predict and use pitch in a low resolution of one value for every input symbol, it makes it easy to adjust pitch interactively, enabling practical applications in pitch editing. Constant offsetting of F_0 with FastPitch produces naturally sounding low- and high-pitched variations of voice that preserve the perceived speaker identity. We conclude that the model learns to mimic the action of vocal chords, which happens during the voluntary modulation of voice.

2. MODEL DESCRIPTION

The architecture of FastPitch is shown in Figure 1. It is based on FastSpeech and composed mainly of two feed-forward Transformer (FFTr) stacks [1]. The first one operates in the resolution of input tokens, the second one in the resolution of the output frames. Let $\boldsymbol{x} = (x_1, \ldots, x_n)$ be the sequence of input lexical units, and $\boldsymbol{y} = (y_1, \ldots, y_t)$ be the sequence of target mel-scale spectrogram frames. The first FFTr stack produces the hidden representation $\boldsymbol{h} = \text{FFTr}(\boldsymbol{x})$. The hidden representation $\boldsymbol{h}$ is used to make predictions about the duration and average pitch of every character with a 1-D CNN
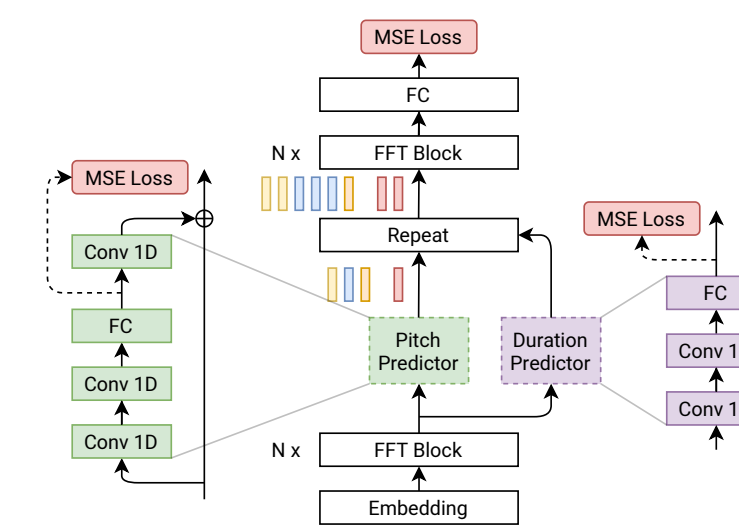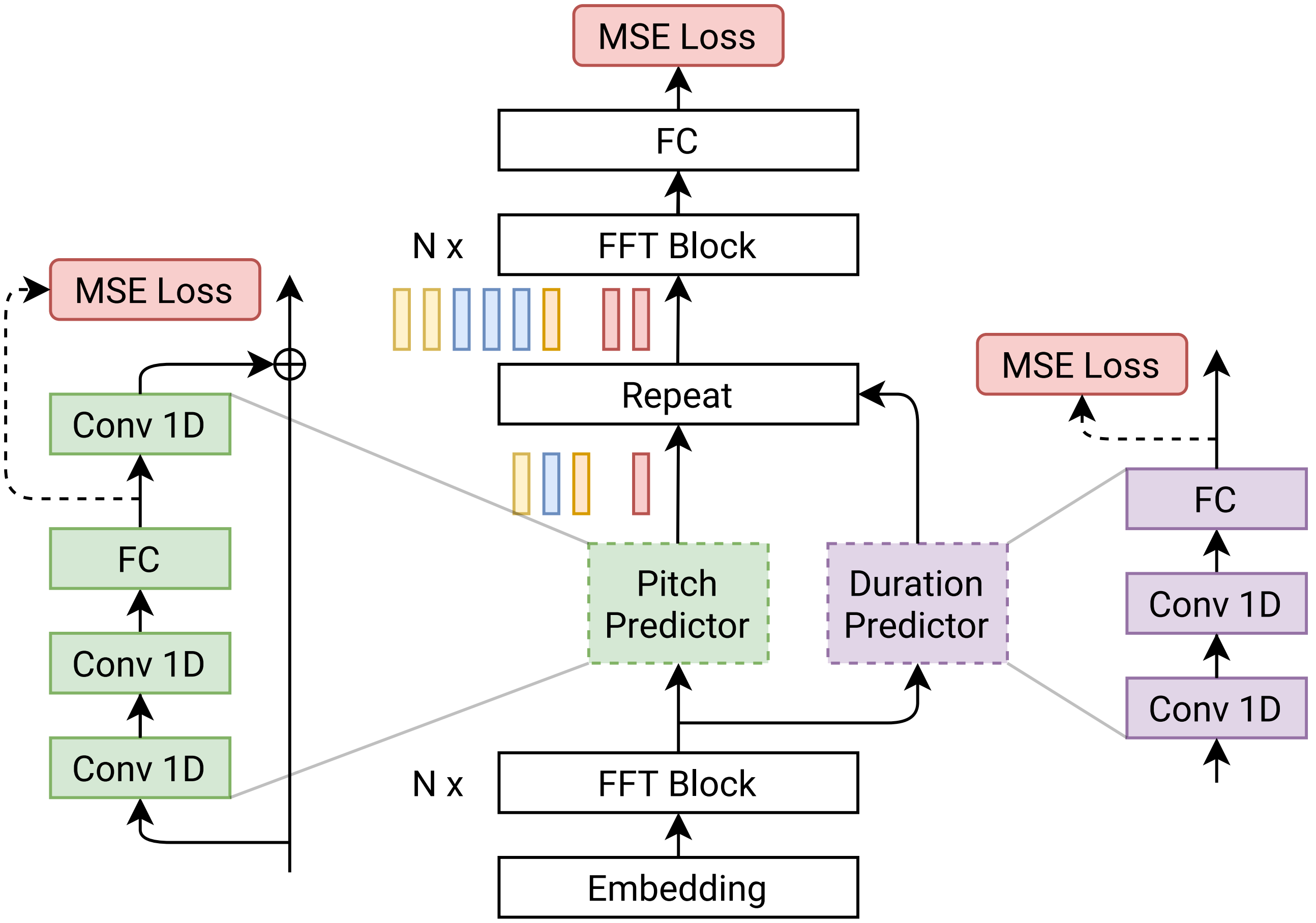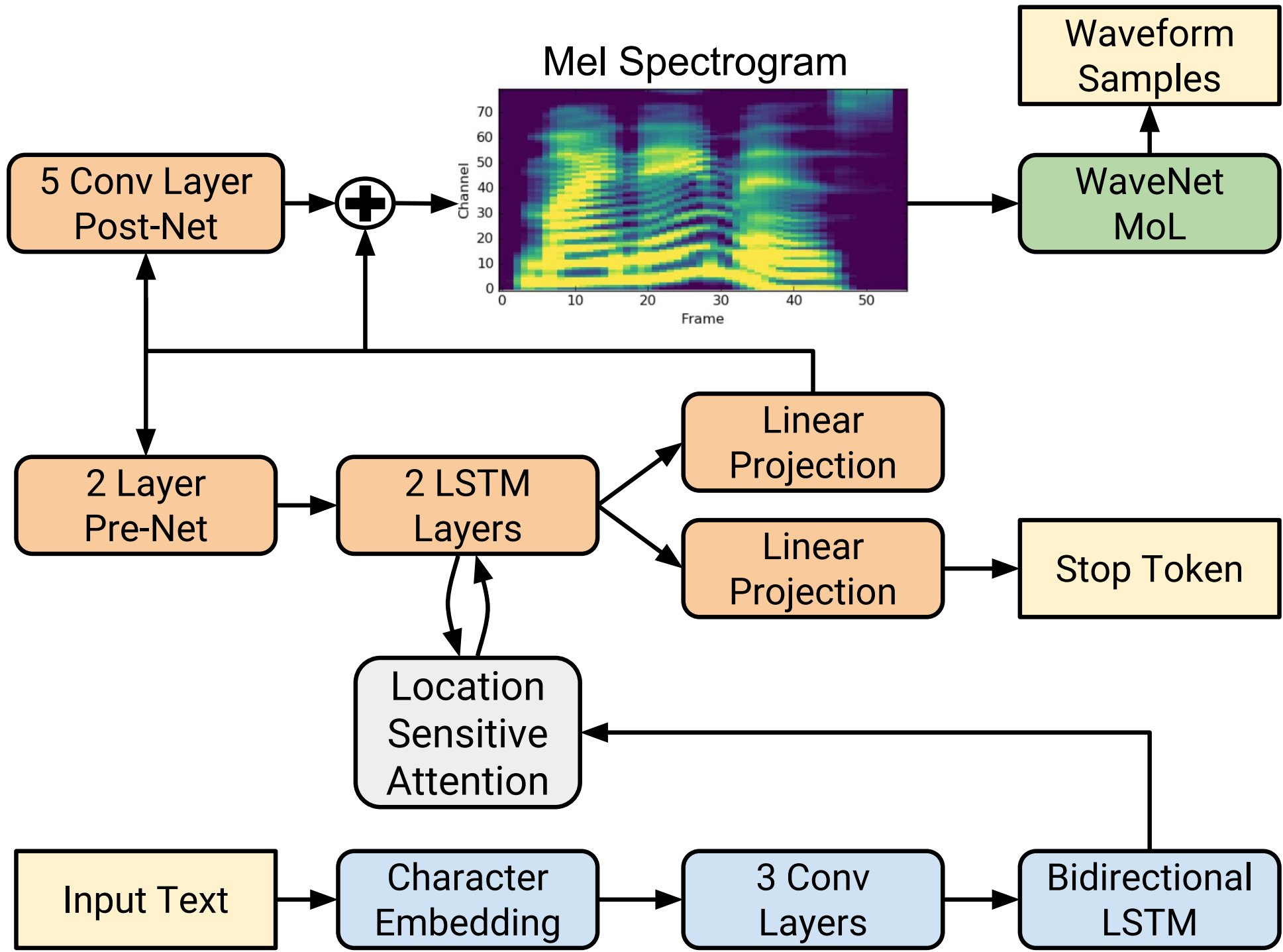

Fig. 1. **Architecture of FastPitch** follows FastSpeech [1]. A single pitch value is predicted for every temporal location.

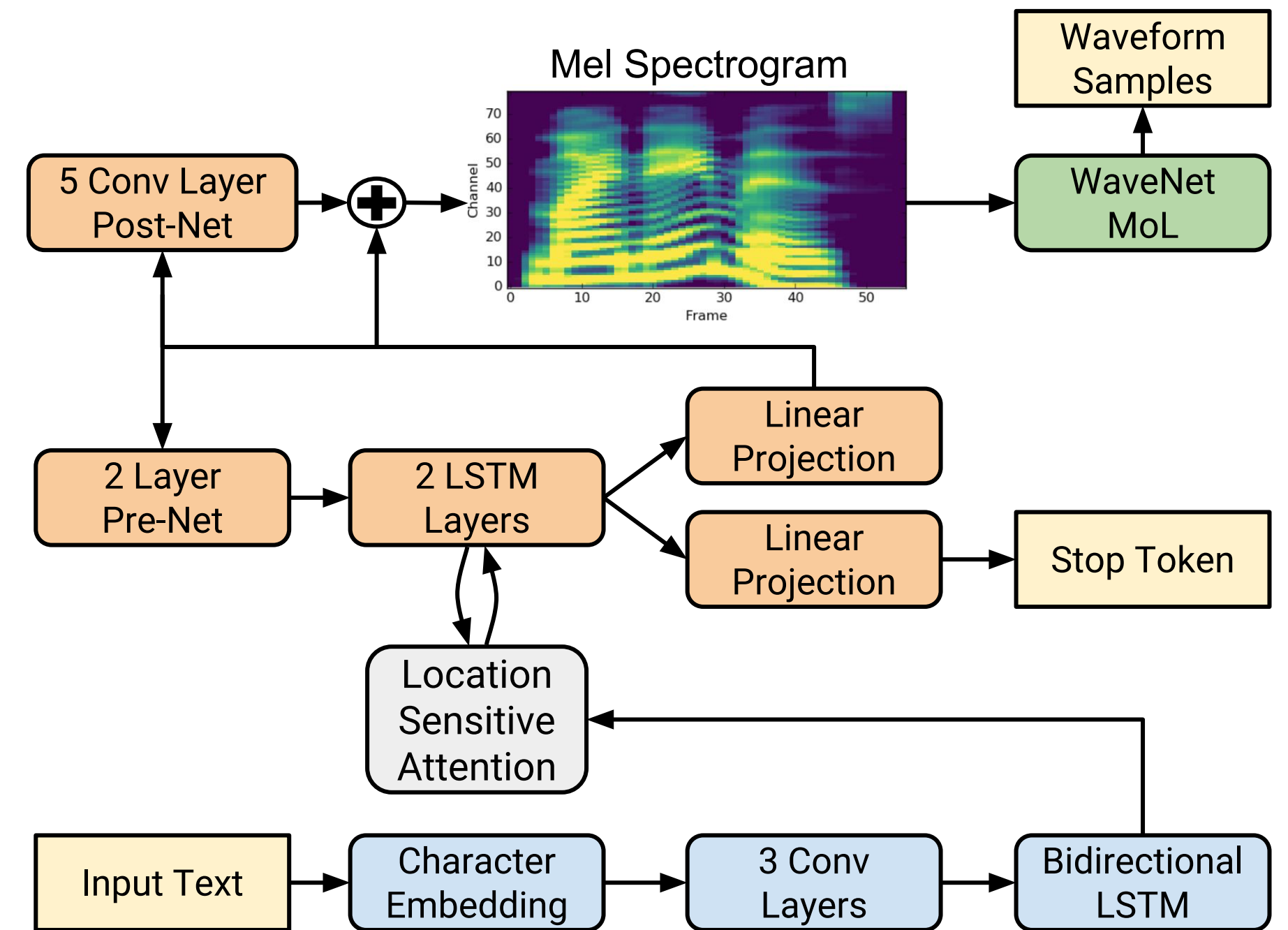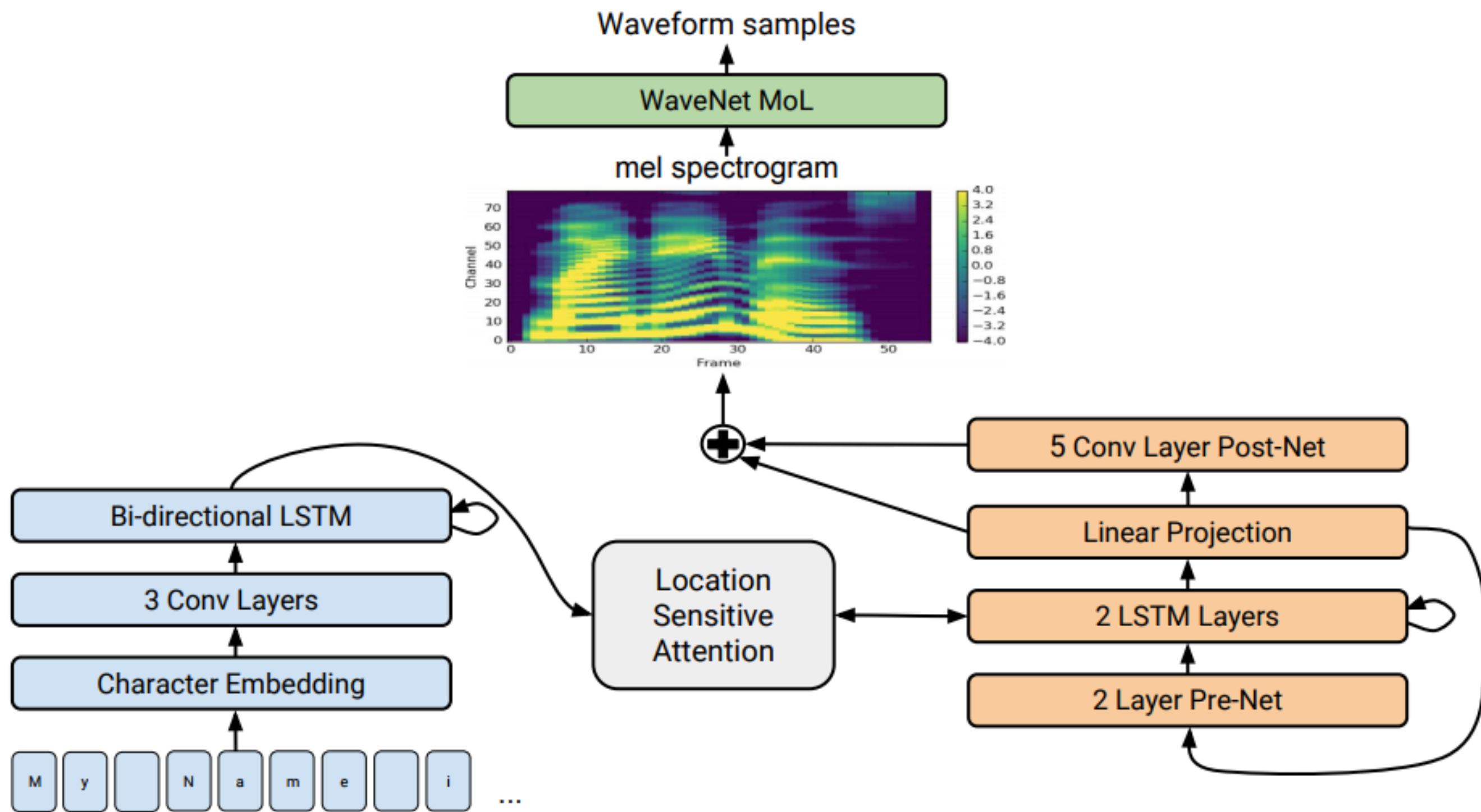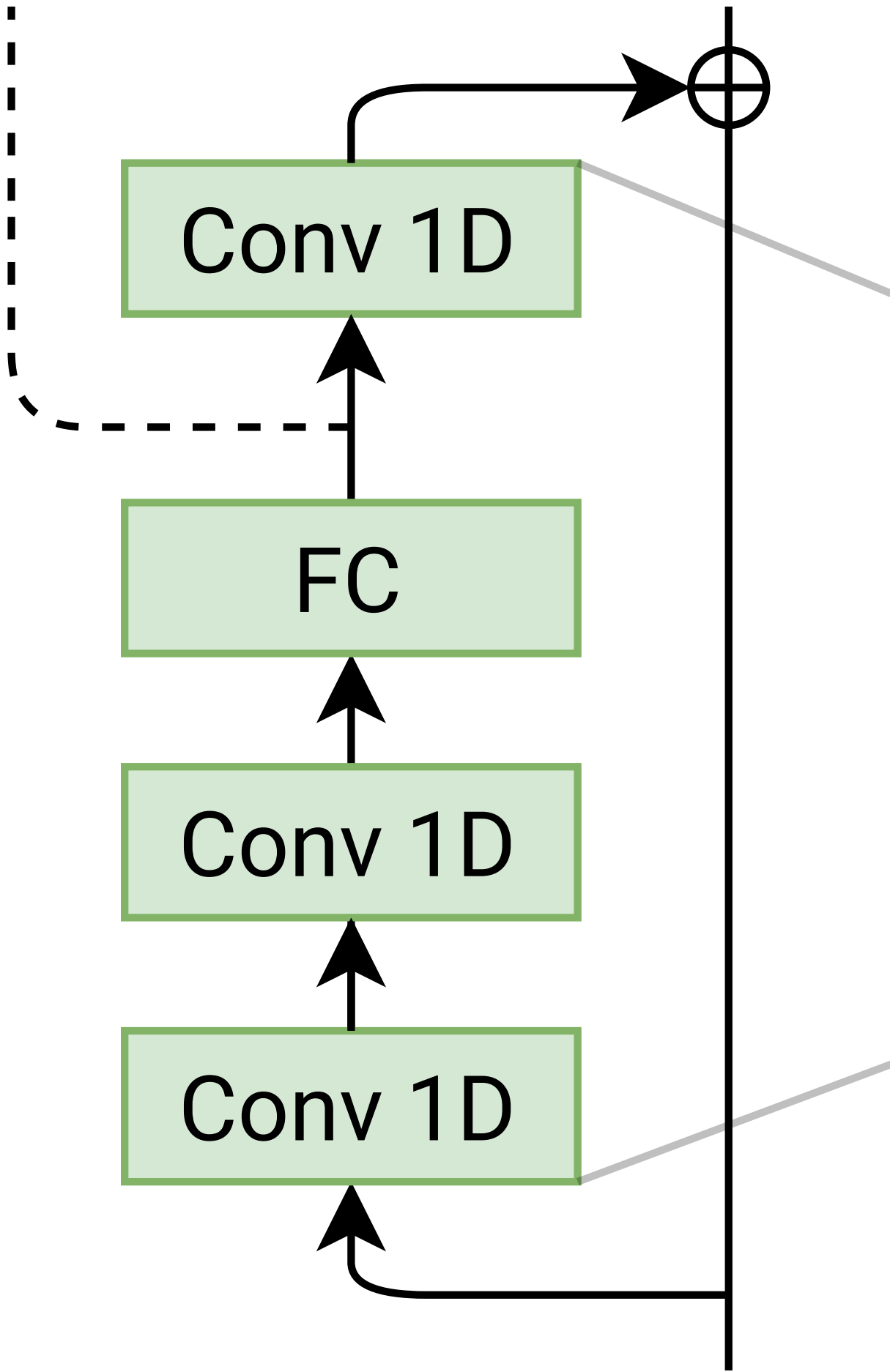# Architecture diagrams
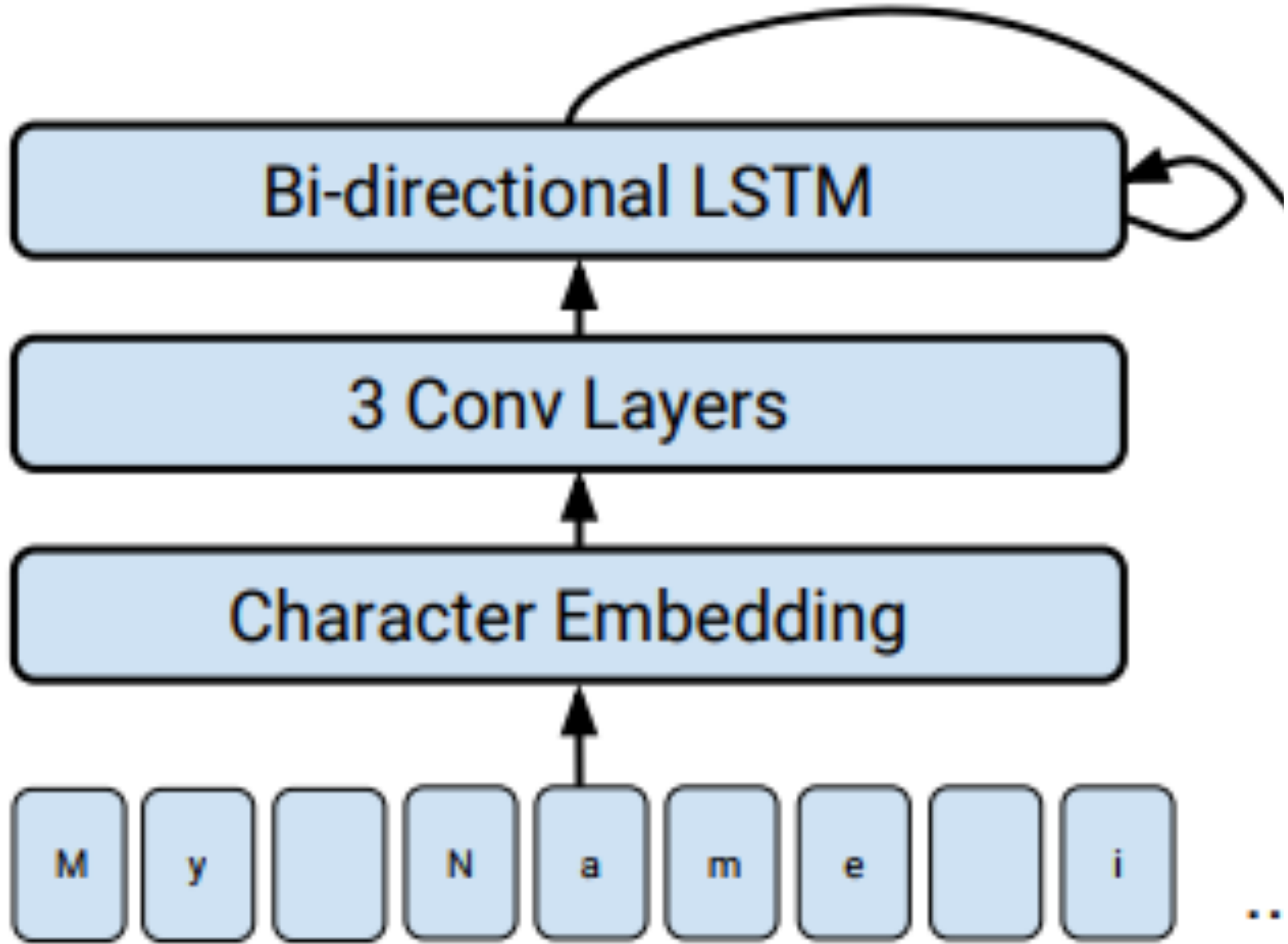
Understanding architecture diagrams

Fig. 1. Block diagram of the Tacotron 2 system architecture.

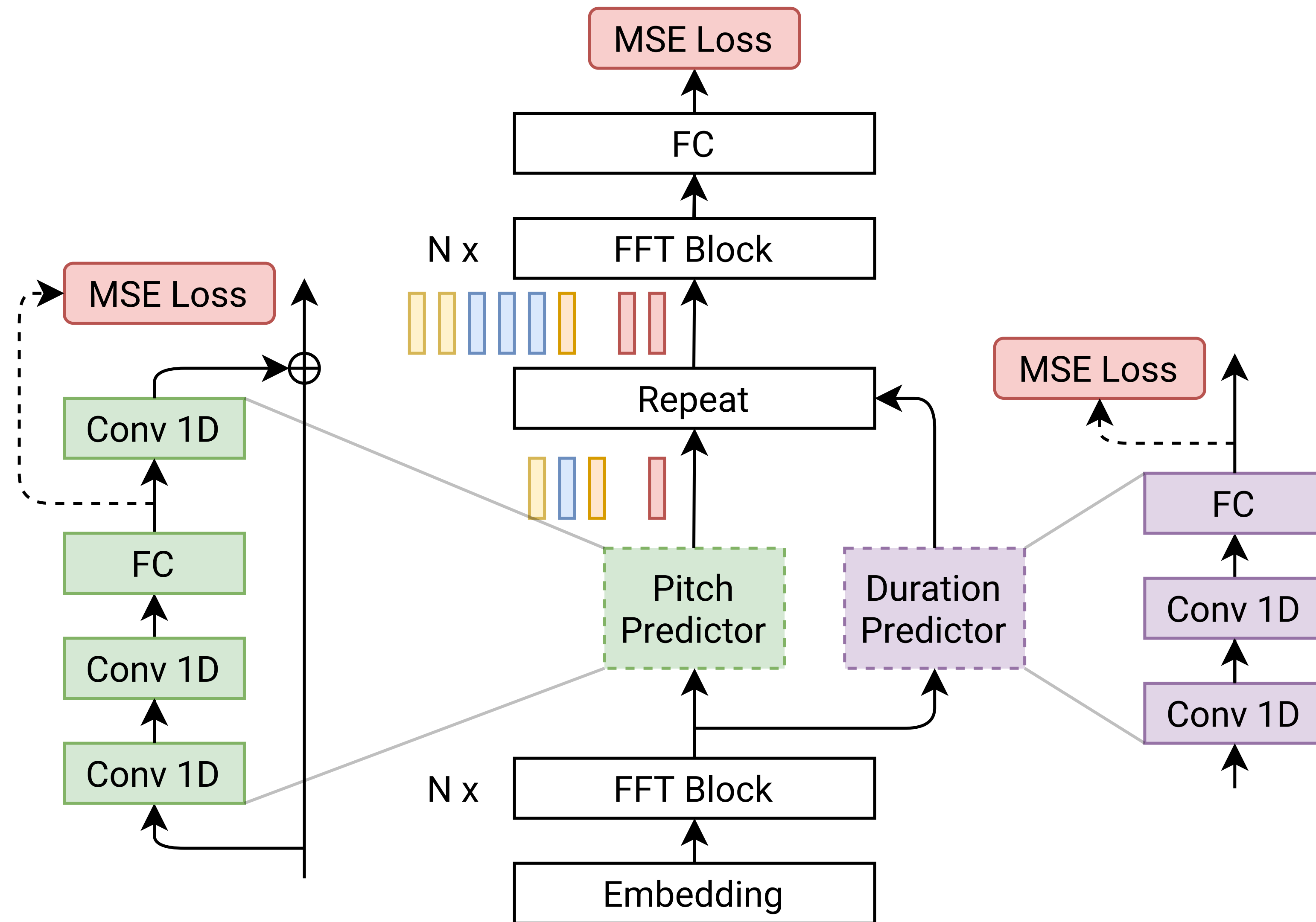# Stacking layers to create more powerful networks

Orientation

- Case study (group exercise)
  - FastPitch

- Q&A
  - how the state-of-the-art material relates to the assignment
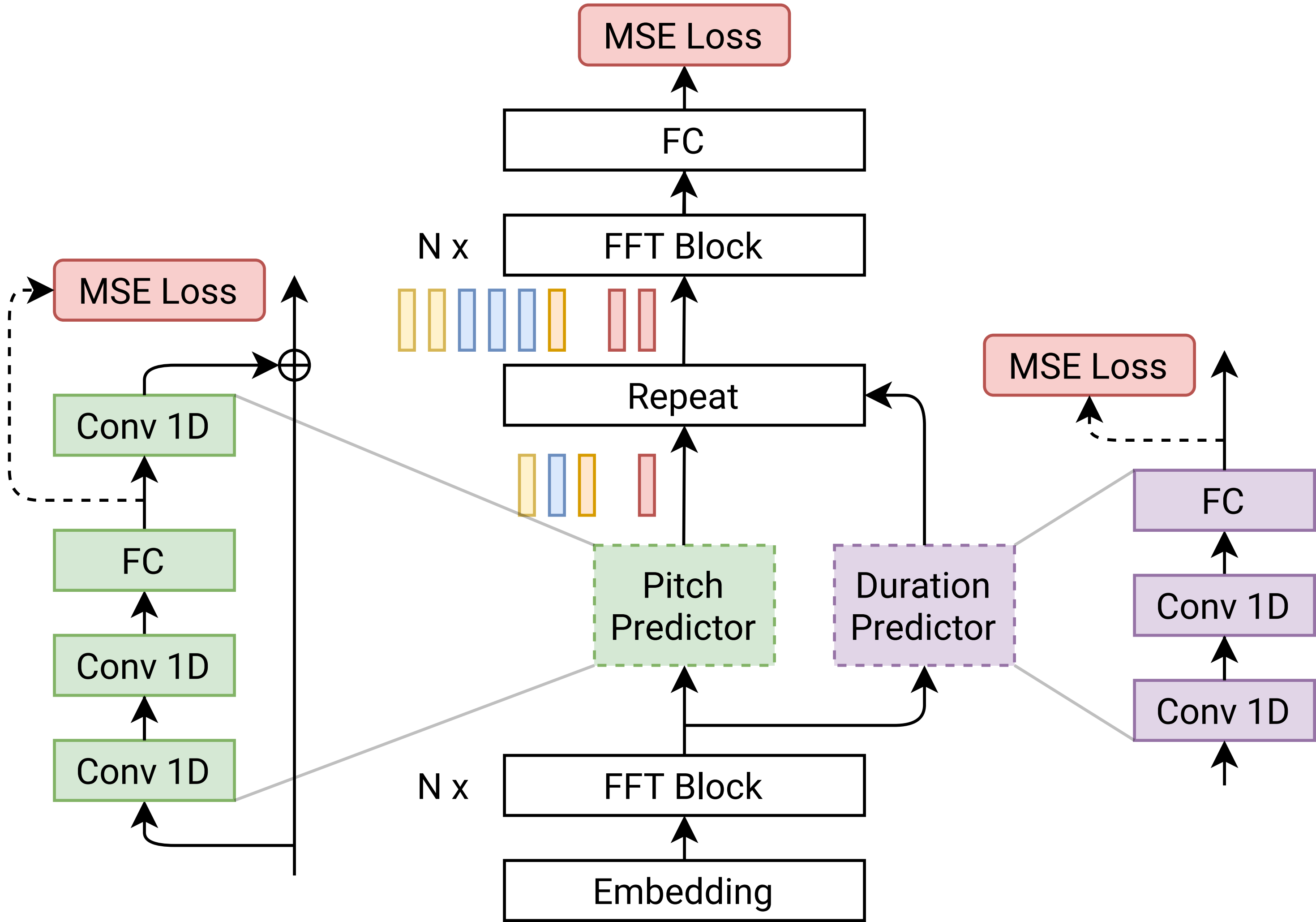
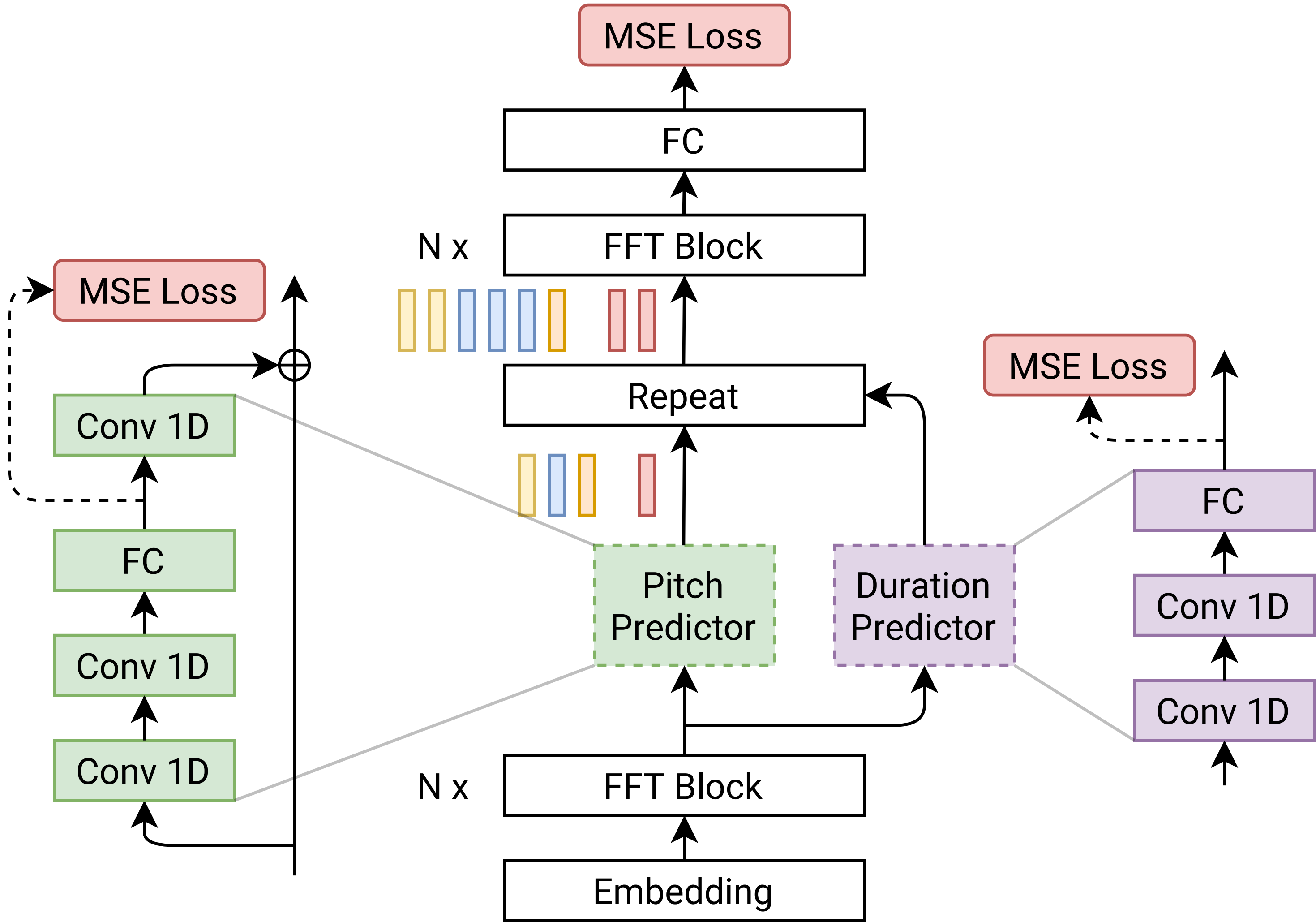- SoundStream
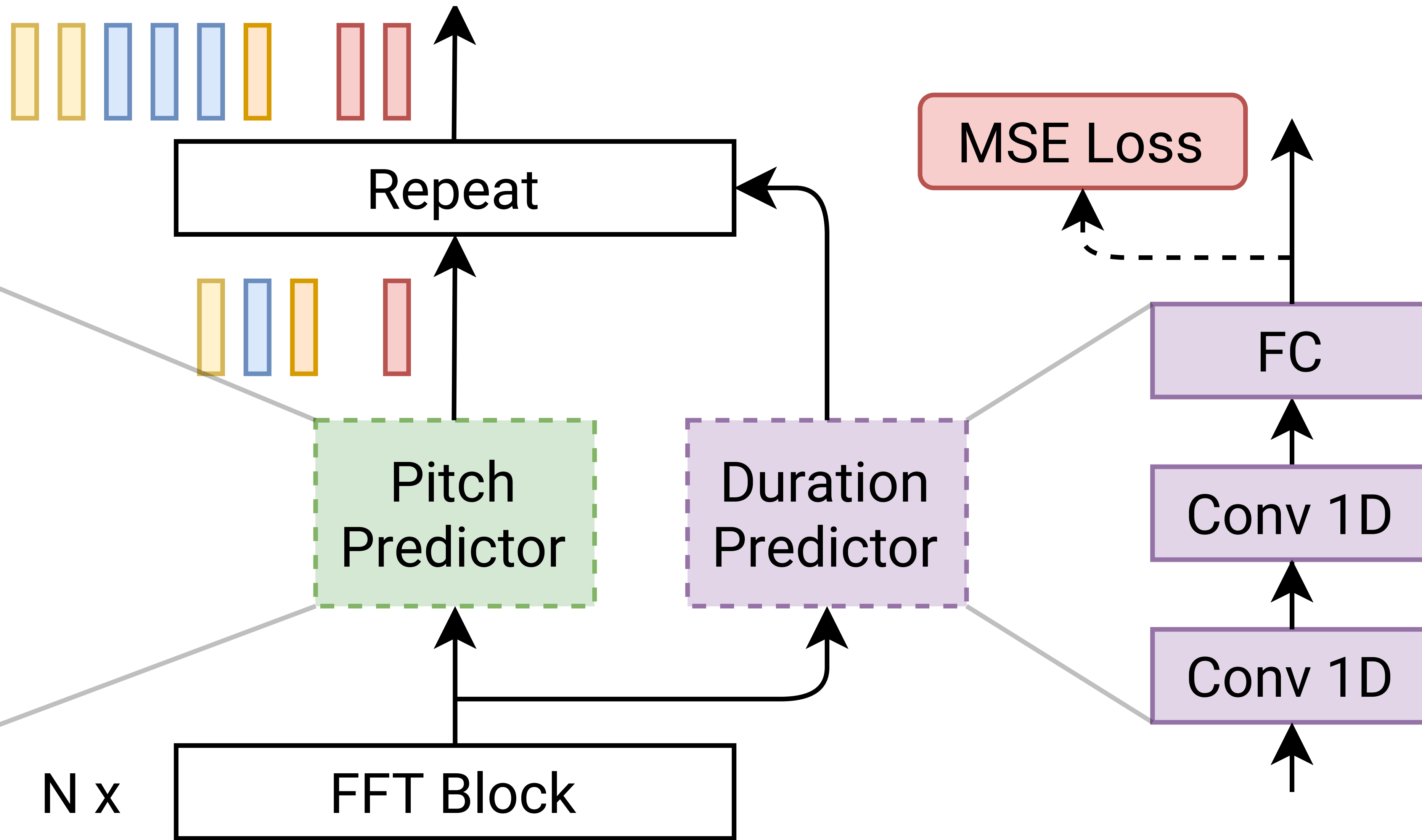 - learning to encode speech

Case study

FastPitch

Input and output - add them to the diagram

Duration - describe how it is predicted and used during inference

"Pitch" (F0!) - describe how it is **predicted** during inference
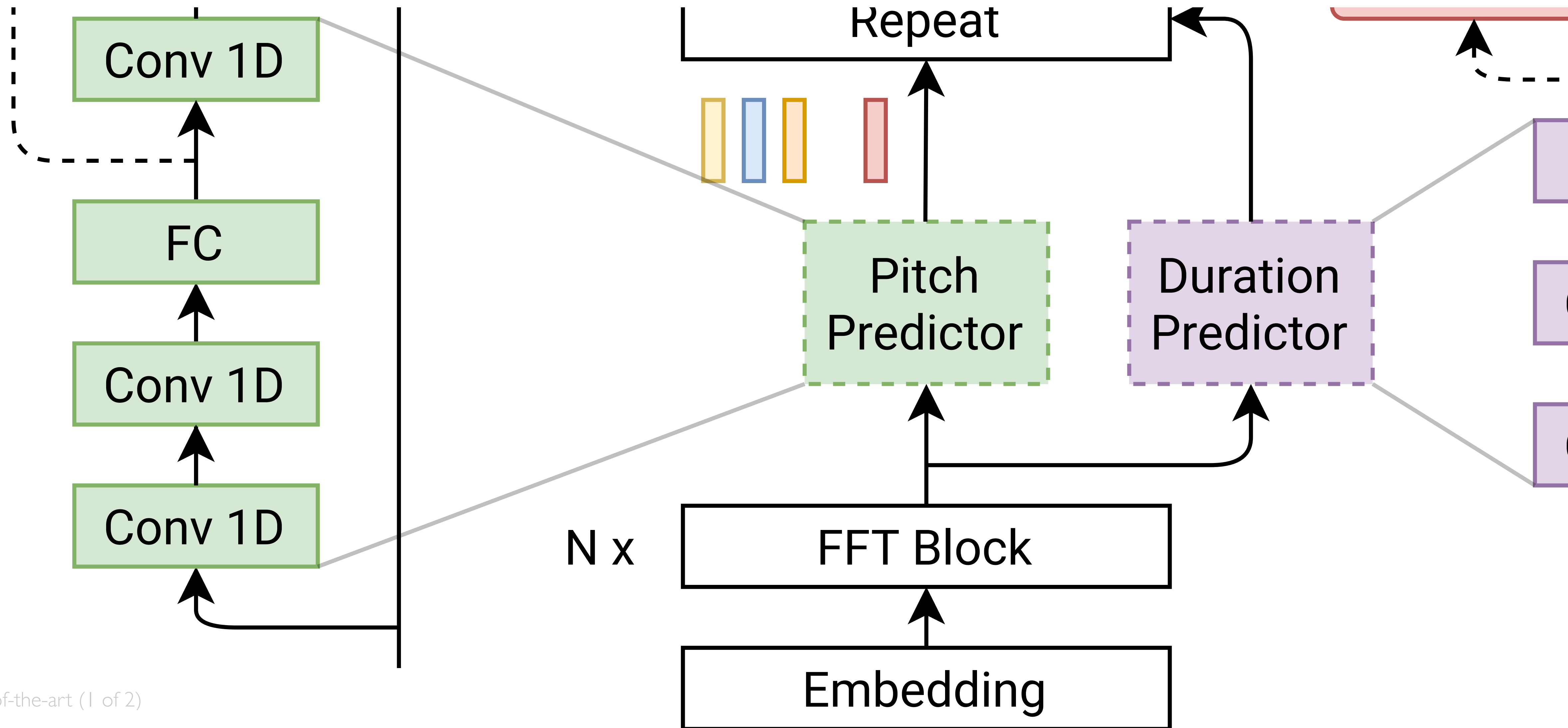
"Pitch" (F0!) - describe how it is **used** during inference

N x FFT Block

MSE Loss

Repeat

MSE Loss

Conv 1D

FC

Conv 1D

Pitch Predictor

Duration Predictor

Con

Con
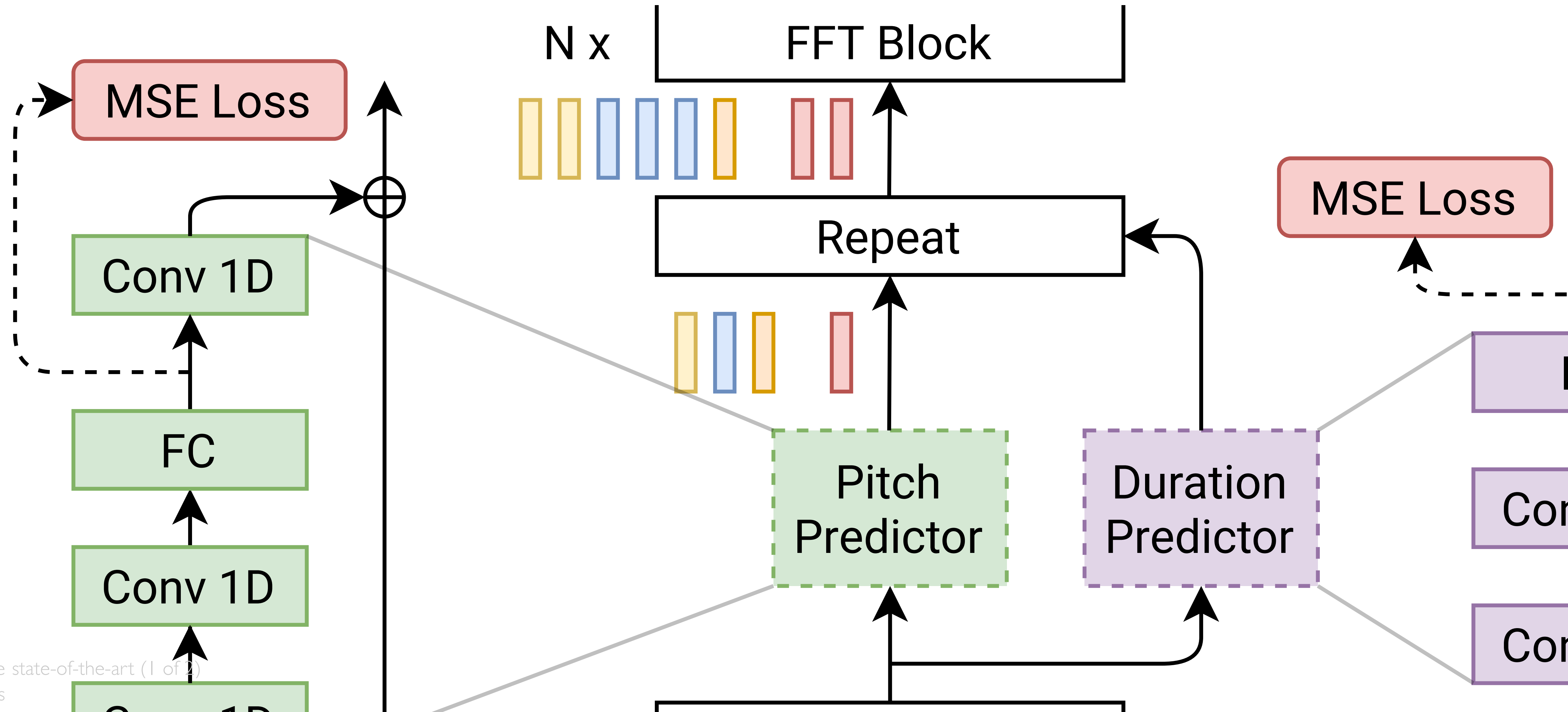
Representations - describe them all; which ones are learned?
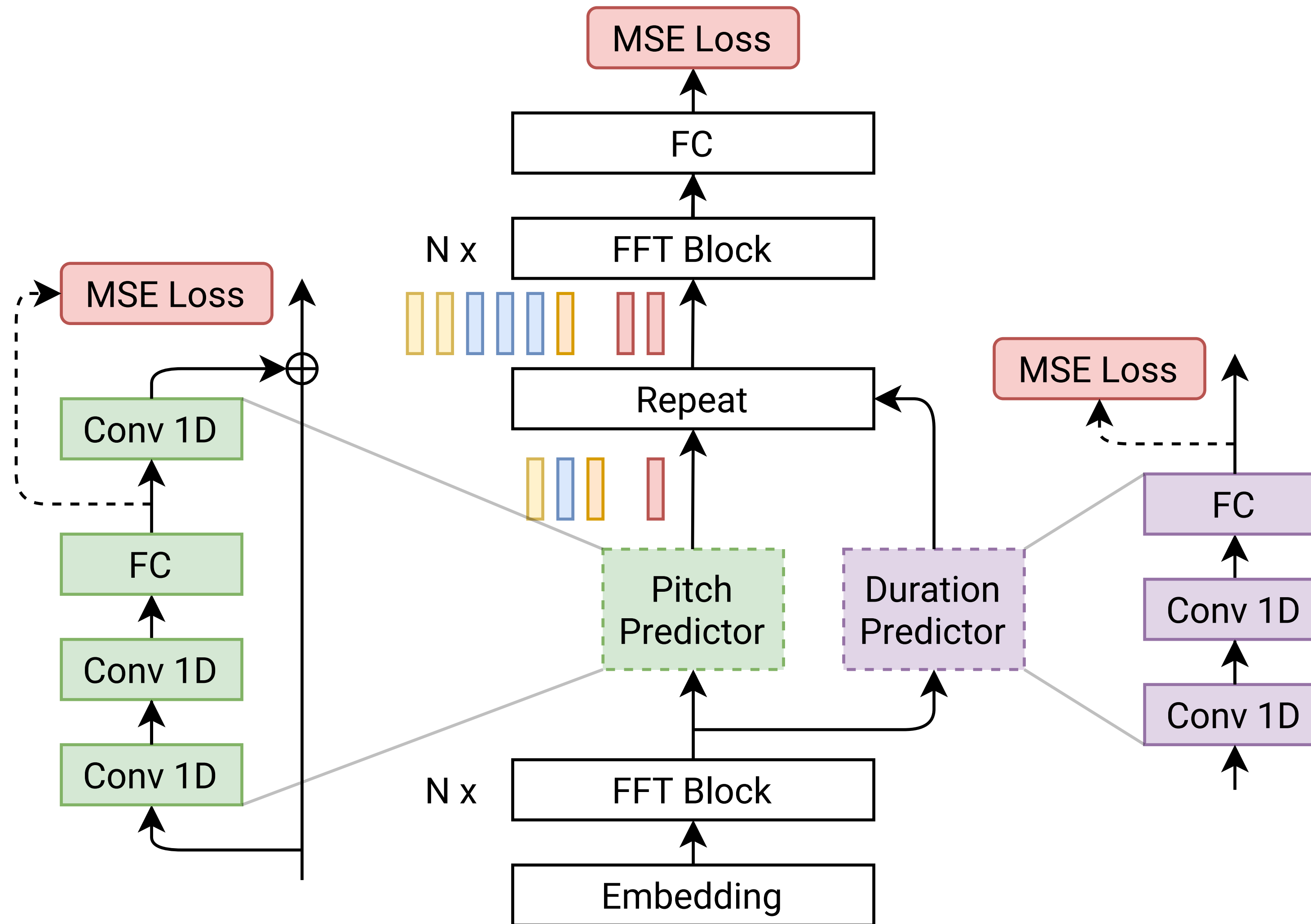
Combining representations - describe every point where this occurs
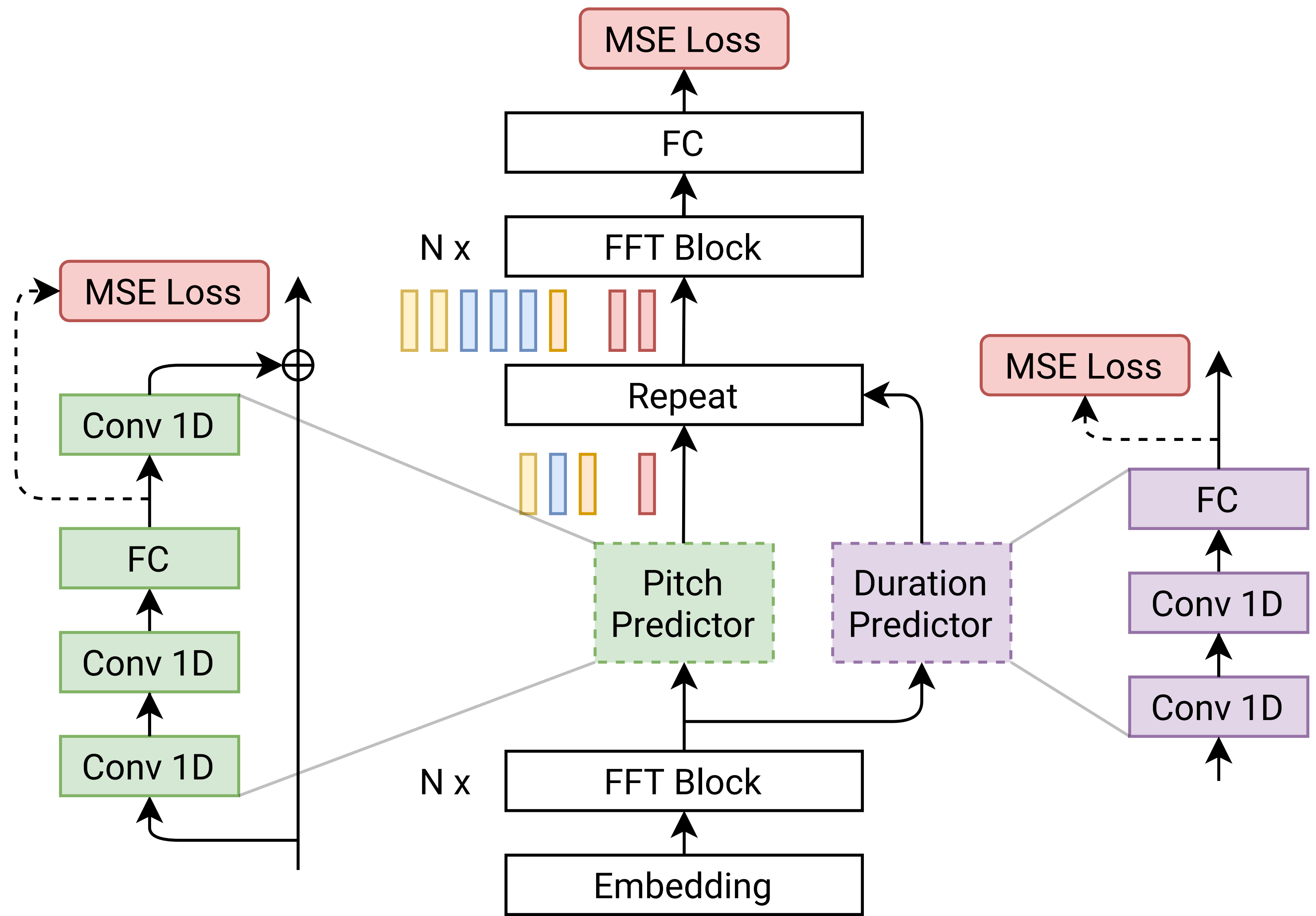
Training vs. **inference** - which one is this diagram describing ?
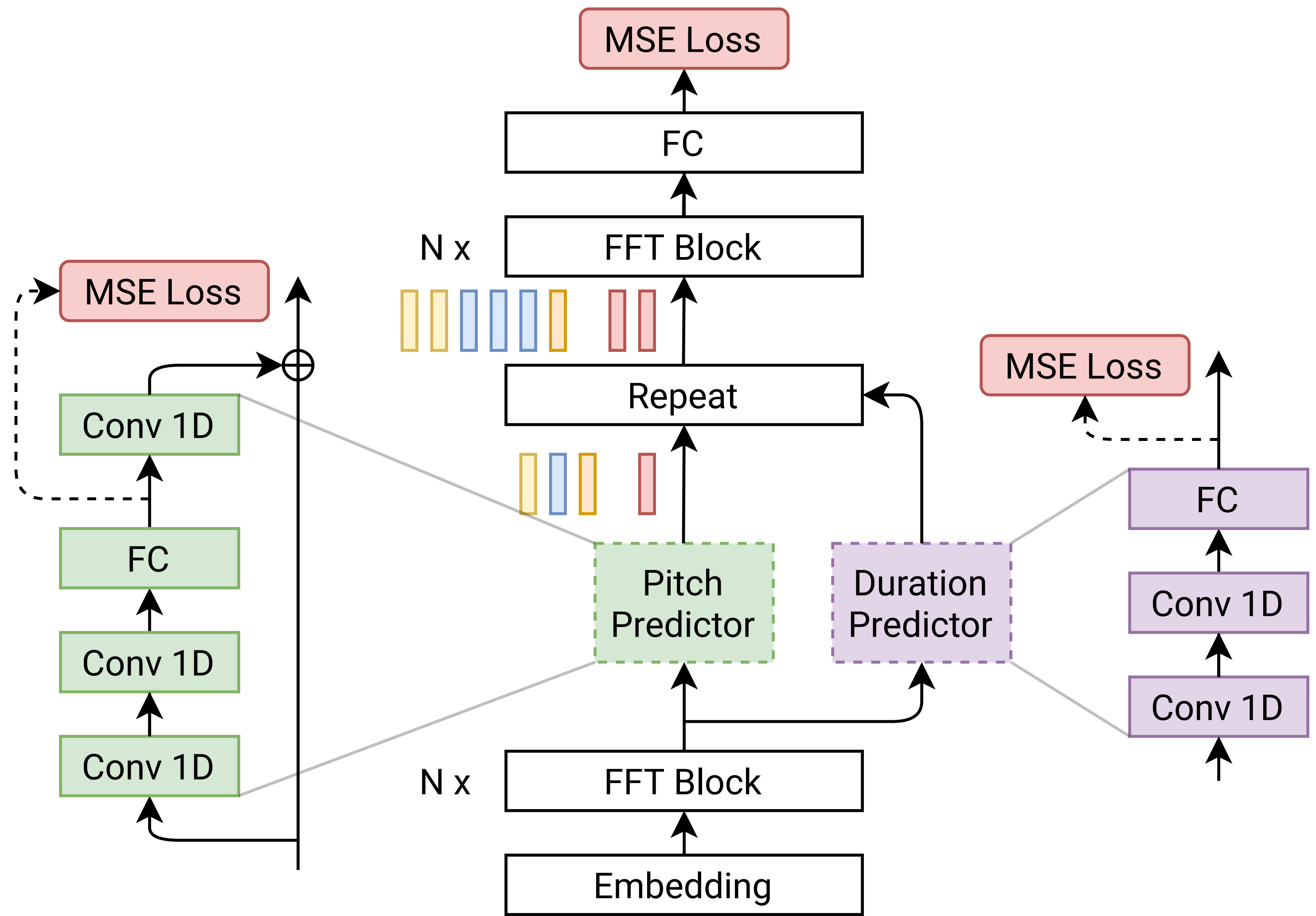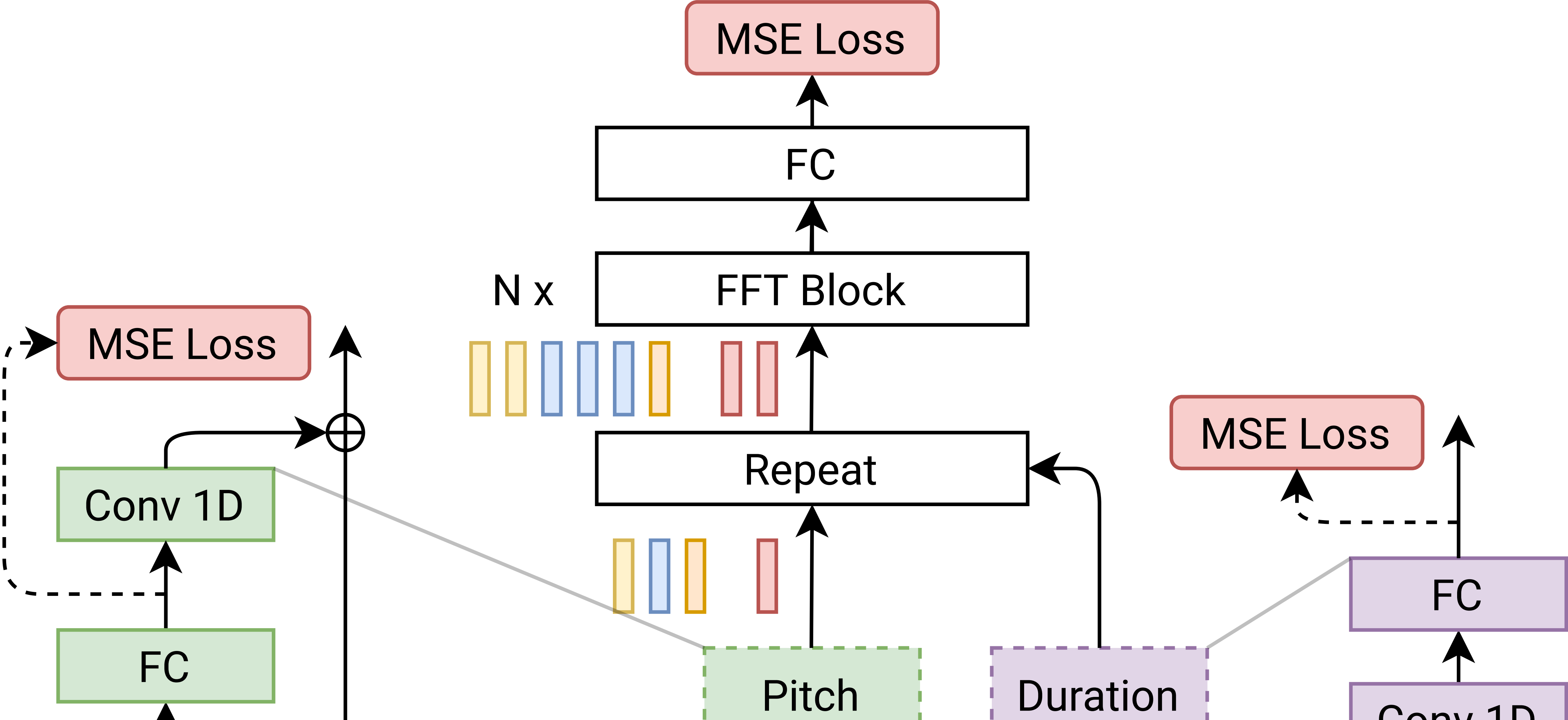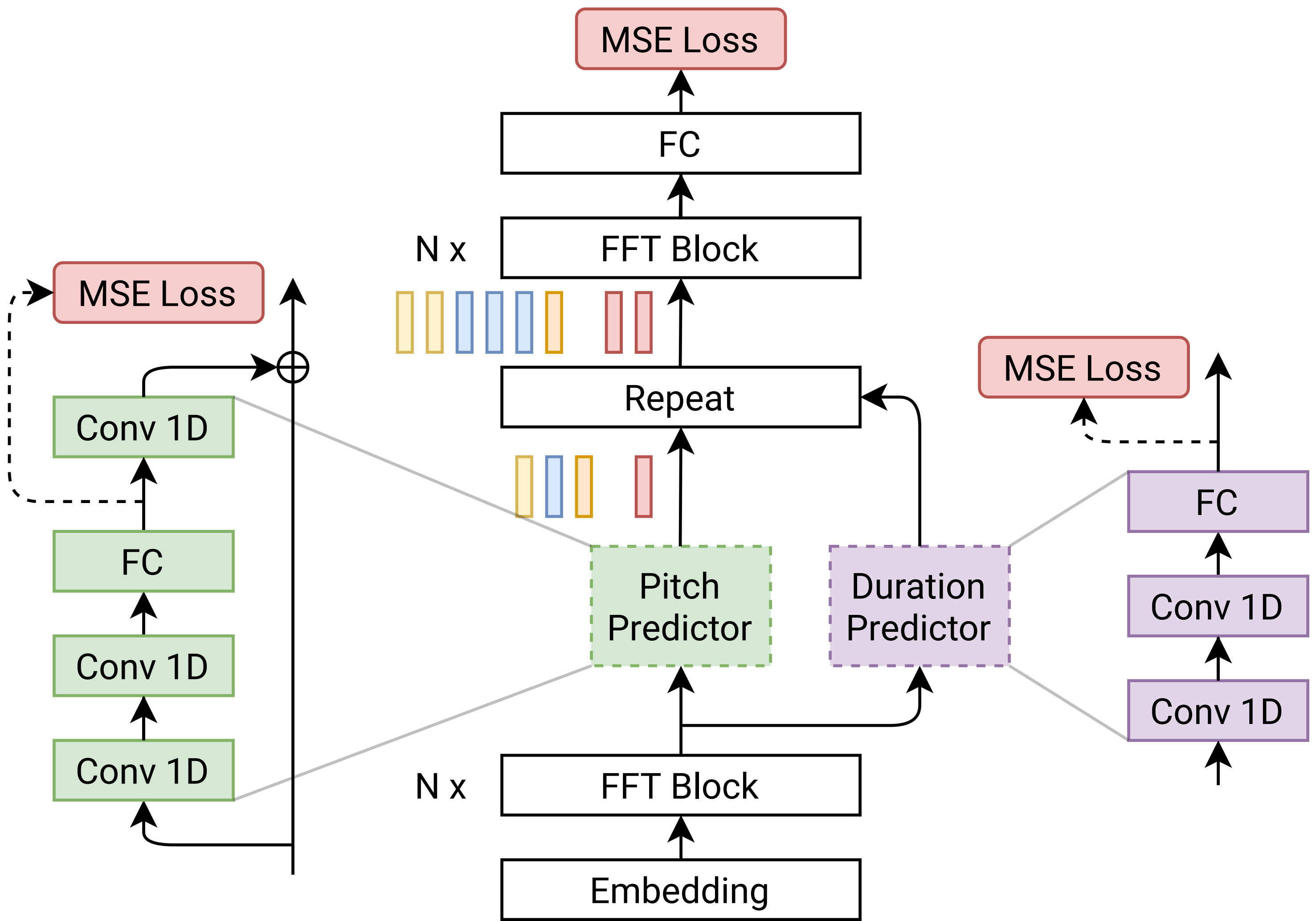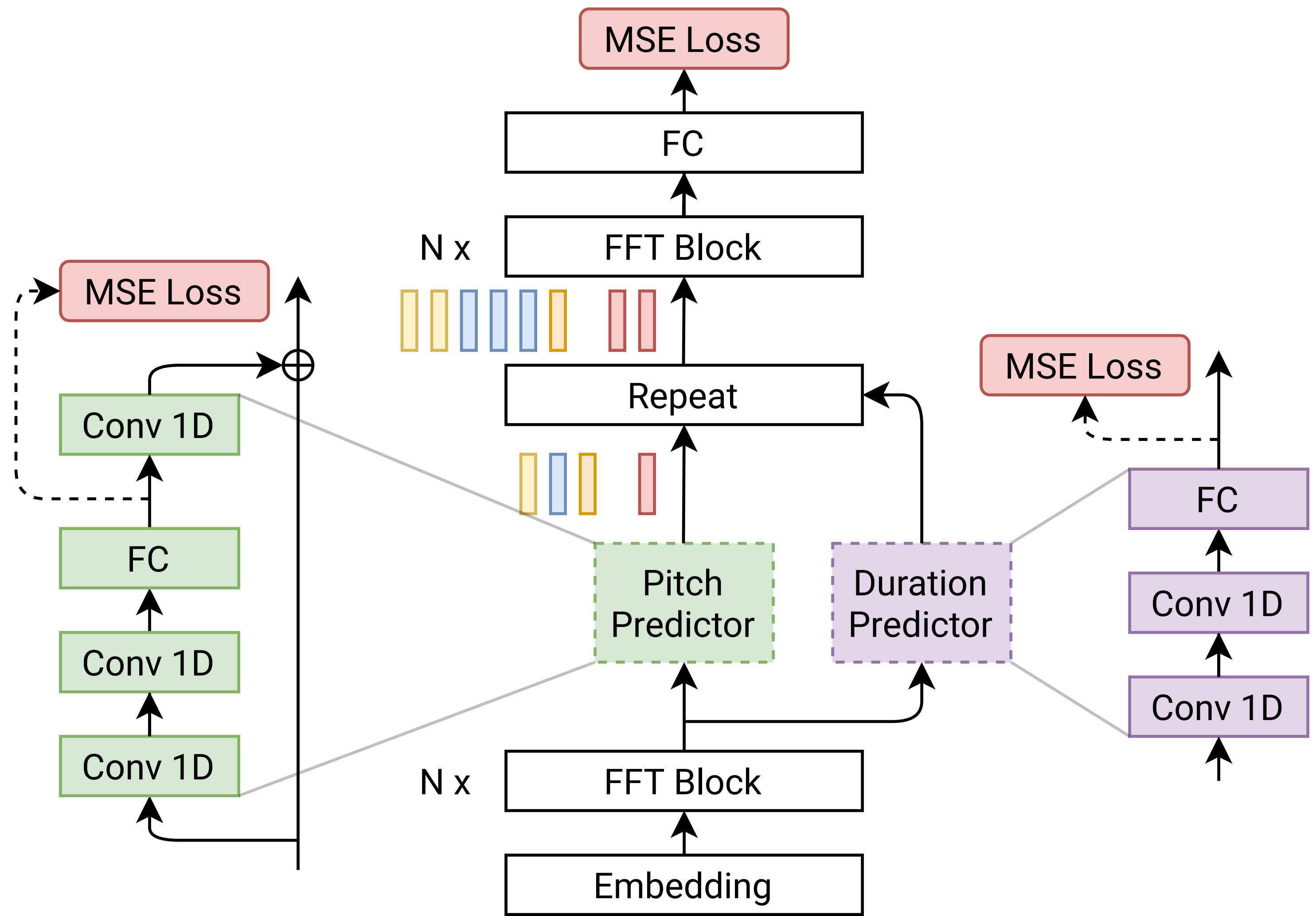
Terminology & jargon - add more to this list, then we will define them all

- architectures
  - fully parallel
  - recurrent
  - autoregressive
  - positional encoding
  - causal
- training methods
  - optimiser (e.g., Adam)
  - teacher forcing (in autoregressive model)

- input symbol, lexical unit, character, grapheme
- audio (*never* "audios" please!!)
- ground truth
- ablation study
- real-time factor (RTF)

Orientation

- Case study (group exercise)
  - FastPitch

- Q&A
  - how the state-of-the-art material relates to the assignment

- SoundStream
  - learning to encode speech

How the state-of-the-art material relates to the assignment

Tips on interpreting the marking sheet

Speech Synthesis assignment marking scheme 2023-24

Category		Points available
Understanding (theory)	Title, abstract	5
	Explaining unit selection	5
20 points	Theoretical connections to current methods	10
Critical thinking (putting theory into practice)	Data: script, dictionary, recording, alignment	5
	Signal processing: pitchmarking, F0, etc	5
20 points	Practical implications for current methods	10
Evaluation	Experimental design	10
	Execution of a basic listening test	5
20 points	Conclusions	5
Scientific writing	Conform with the journal style guide *and* anonymous submission, correct filename, exam number, state wordcount, page numbers	5
20 points	Clarity, coherence, structure, presentation, figures & captions, bibliography	15
Additional (for a higher mark)	*Any/all of these and/or going beyond the basic expectations in other ways:* • better script design (manual or automatic) • recording additional data • a more sophisticated listening test • forms of evaluation other than a listening test • using your knowledge of phonetics • …and so on	20
20 points		
TOTAL		**100**

- **Incorporate** these throughout the paper
- Example 1:
  - Unit selection performs implicit **regression** from linguistic features to acoustic properties
  - How do various current methods do that?
- Example 2:
  - In unit selection, several choices are available for **waveform representation**
  - Are these the same or different in current methods?
- etc.

Tips on interpreting the marking sheet

Speech Synthesis assignment marking scheme 2023-24

Category		Points available
Understanding (theory)	Title, abstract	5
	Explaining unit selection	5
20 points	Theoretical connections to current methods	10
Critical thinking (putting theory into practice)	Data: script, dictionary, recording, alignment	5
	Signal processing: pitchmarking, F0, etc	5
20 points	Practical implications for current methods	10
Evaluation	Experimental design	10
	Execution of a basic listening test	5
20 points	Conclusions	5
Scientific writing	Conform with the journal style guide *and* anonymous submission, correct filename, exam number, state wordcount, page numbers	5
20 points	Clarity, coherence, structure, presentation, figures & captions, bibliography	15
Additional (for a higher mark) **20 points**	*Any/all of these and/or going beyond the basic expectations in other ways:* • better script design (manual or automatic) • recording additional data • a more sophisticated listening test • forms of evaluation other than a listening test • using your knowledge of phonetics • …and so on	20
TOTAL		**100**

- Link everything to **current methods**. Do not do experiments with current methods, but use the literature to back up your claims.

- Example:

  - You will have discovered how sensitive (or not) unit selection is to many **design choices**, such as database contents, pitchmark accuracy, …

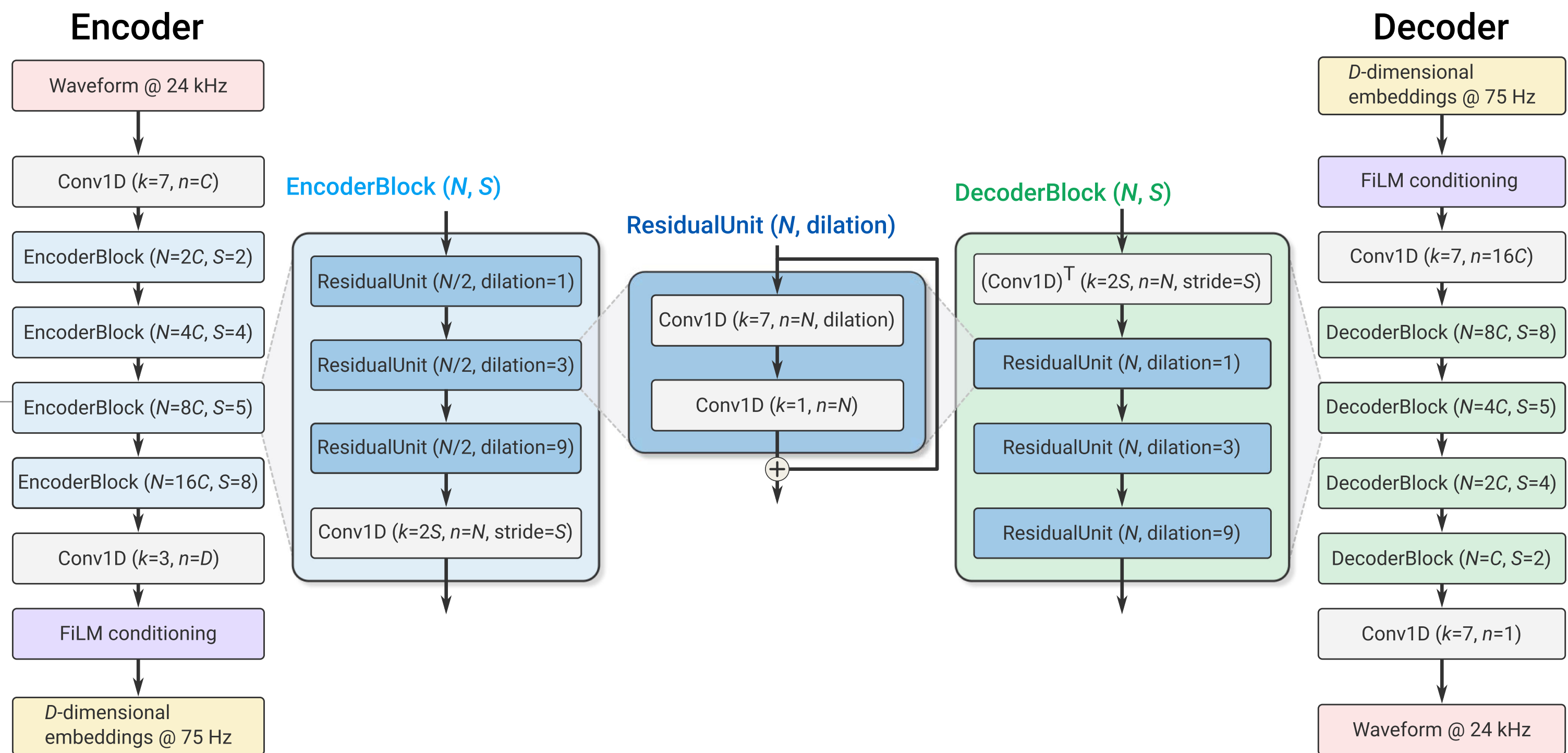 - Would current methods be more or less sensitive to each choice?

Orientation

- Case study (group exercise)
  - FastPitch

- Q&A
  - how the state-of-the-art material relates to the assignment

- SoundStream
  - learning to encode speech

Case study

SoundStream

Encoder

| Waveform @ 24 kHz |
| Conv1D (k=7, n=C) |
| EncoderBlock (N=2C, S=2) |
| EncoderBlock (N=4C, S=4) |
| EncoderBlock (N=8C, S=5) |
| EncoderBlock (N=16C, S=8) |
| Conv1D (k=3, n=D) |
| FiLM conditioning |
| D-dimensional embeddings @ 75 Hz |

EncoderBlock (N, S)

| ResidualUnit (N/2, dilation=1) |
| ResidualUnit (N/2, dilation=3) |
| ResidualUnit (N/2, dilation=9) |
| Conv1D (k=2S, n=N, stride=S) |

ResidualUnit (N, dilation)

| Conv1D (k=7, n=N, dilation) |
| Conv1D (k=1, n=N) |

$\oplus$

DecoderBlock (N, S)

| (Conv1D)^T (k=2S, n=N, stride=S) |
| ResidualUnit (N, dilation=1) |
| ResidualUnit (N, dilation=3) |
| ResidualUnit (N, dilation=9) |

Decoder

| D-dimensional embeddings @ 75 Hz |
| FiLM conditioning |
| Conv1D (k=7, n=16C) |
| DecoderBlock (N=8C, S=8) |
| DecoderBlock (N=4C, S=5) |
| DecoderBlock (N=2C, S=4) |
| DecoderBlock (N=C, S=2) |
| Conv1D (k=7, n=1) |
| Waveform @ 24 kHz |

Encoder

Waveform @ 24 kHz

Conv1D (k=7, n=C)

EncoderBlock (N=2C, S=2)

EncoderBlock (N=4C, S=4)

EncoderBlock (N=8C, S=5)

EncoderBlock (N=16C, S=8)

Conv1D (k=3, n=D)

FiLM conditioning

D-dimensional embeddings @ 75 Hz

EncoderBlock (N, S)

ResidualUnit (N/2, dilation=1)

ResidualUnit (N/2, dilation=3)

ResidualUnit (N/2, dilation=9)

Conv1D (k=2S, n=N, stride=S)

ResidualUnit (N, dilation)

Conv1D (k=7, n=N, dilation)

Conv1D (k=1, n=N)

$\oplus$

DecoderBlock (N, S)

(Conv1D)^T (k=2S, n=N, stride=S)

ResidualUnit (N, dilation=1)

ResidualUnit (N, dilation=3)

ResidualUnit (N, dilation=9)

Decoder

D-dimensional embeddings @ 75 Hz

FiLM conditioning

Conv1D (k=7, n=16C)

DecoderBlock (N=8C, S=8)

DecoderBlock (N=4C, S=5)

DecoderBlock (N=2C, S=4)

DecoderBlock (N=C, S=2)

Conv1D (k=7, n=1)

Waveform @ 24 kHz

Module state-of-the-art (1 of 2)
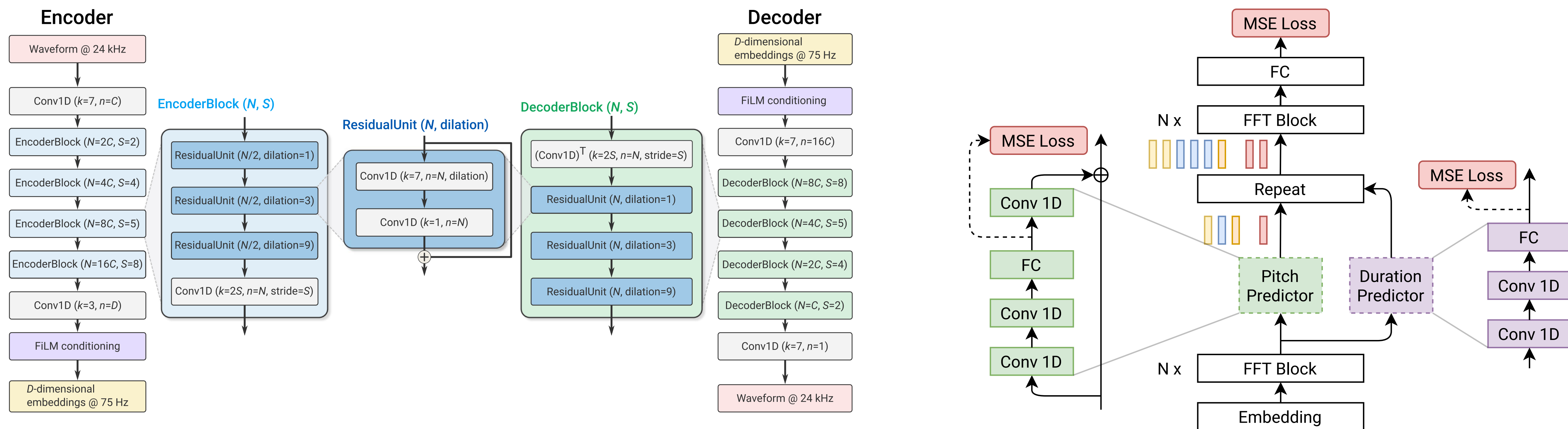Class

Warning: up and down have no meaning in architecture diagrams!

What is an audio codec ?

- Encoder operates at the transmitter end

  - reduces the number of bits-per-second required to represent the audio signal

  - outputs an **encoding** of the signal

- Decoder operates at the receiver end

  - **reconstructs** the audio signal from the encoding

  - reconstructed waveform will not be identical to the original (the codec is "lossy")
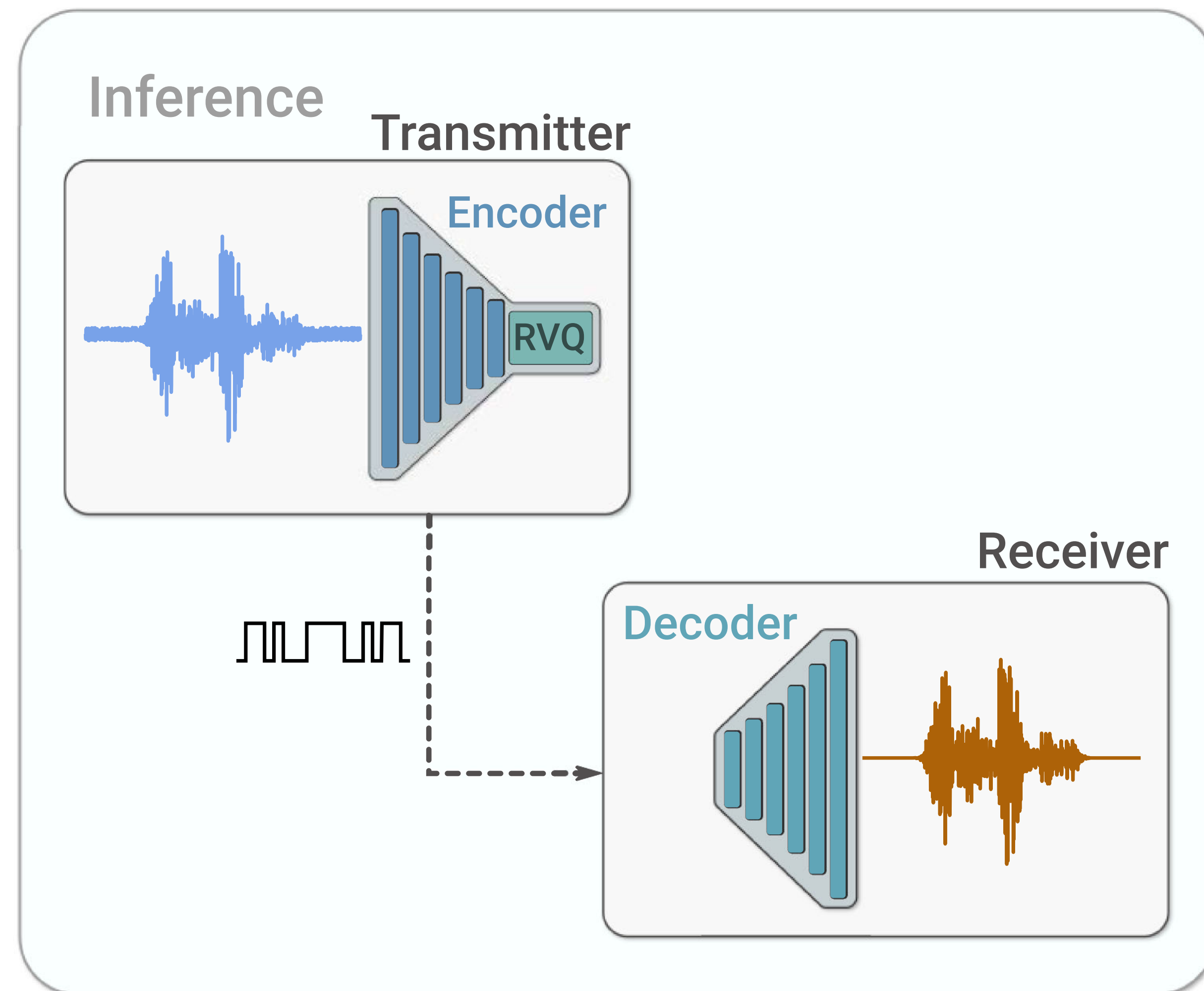
Key ideas in SoundStream (and similar audio encoder-decoder models)

- Reduce sequence length

  - input is a waveform, at a typical sample rate of 24 kHz (samples per second)

  - encoded representation is at 75 Hz (frames per second)

  - progressively reduce from 24 kHz to 75 Hz

- Quantise the learned representation

  - so it becomes a sequence of codes, not vectors

- Residual quantisation

  - to avoid the need for a very large set of codes

Key ideas in SoundStream (and similar audio encoder-decoder models)
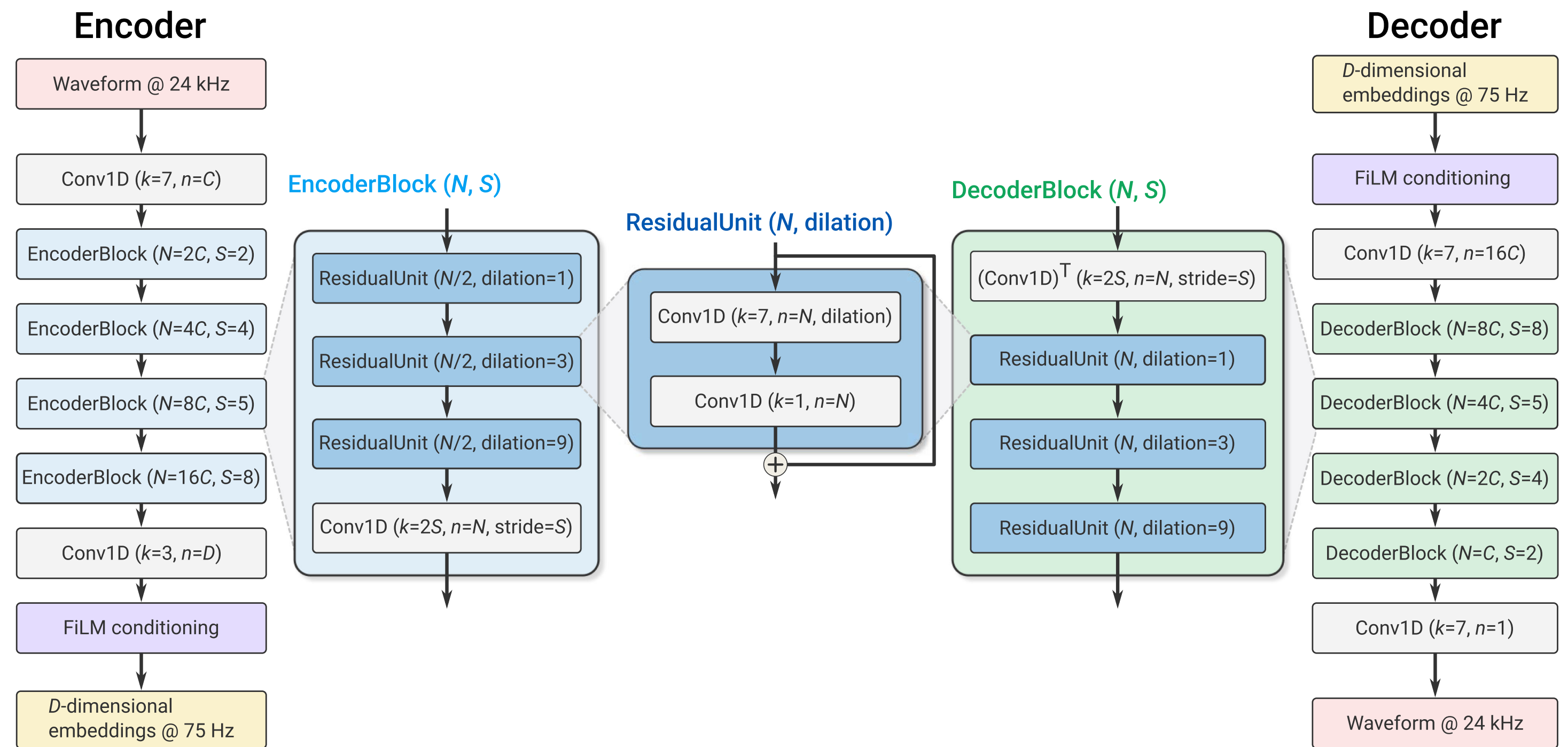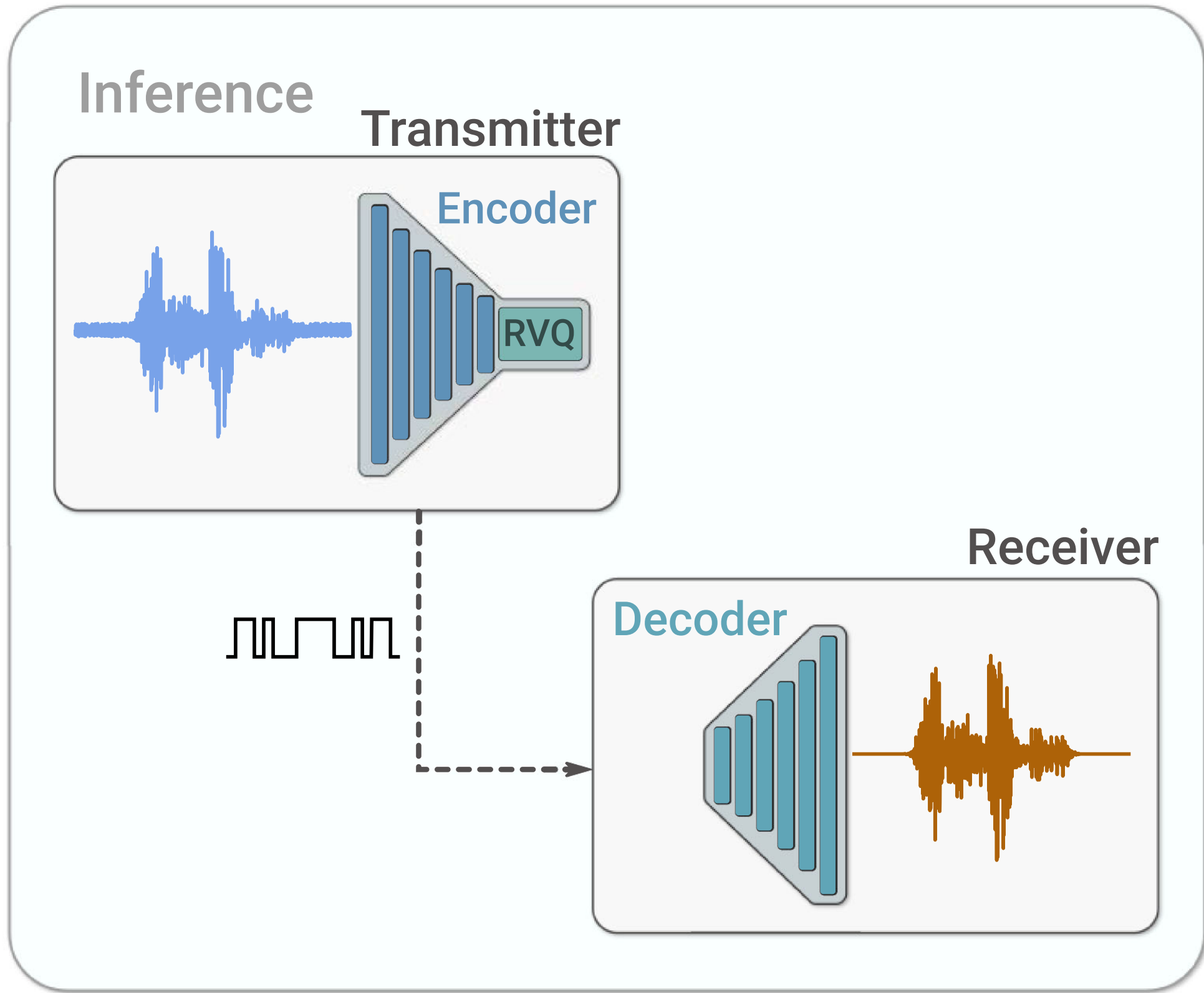
- **Reduce sequence length**
  - input is a waveform, at a typical sample rate of 24 kHz (samples per second)
  - encoded representation is at 75 Hz (frames per second)
  - progressively reduce from 24 kHz to 75 Hz
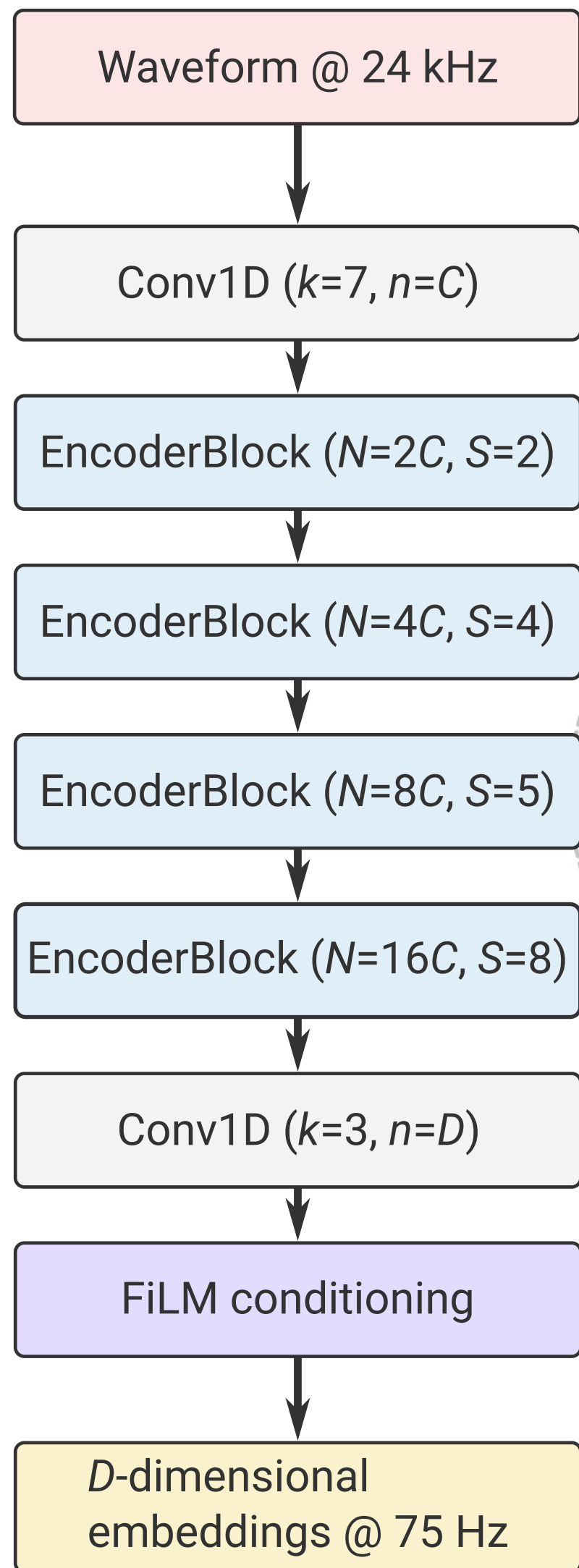
- Quantise the learned representation
  - so it becomes a sequence of codes, not vectors
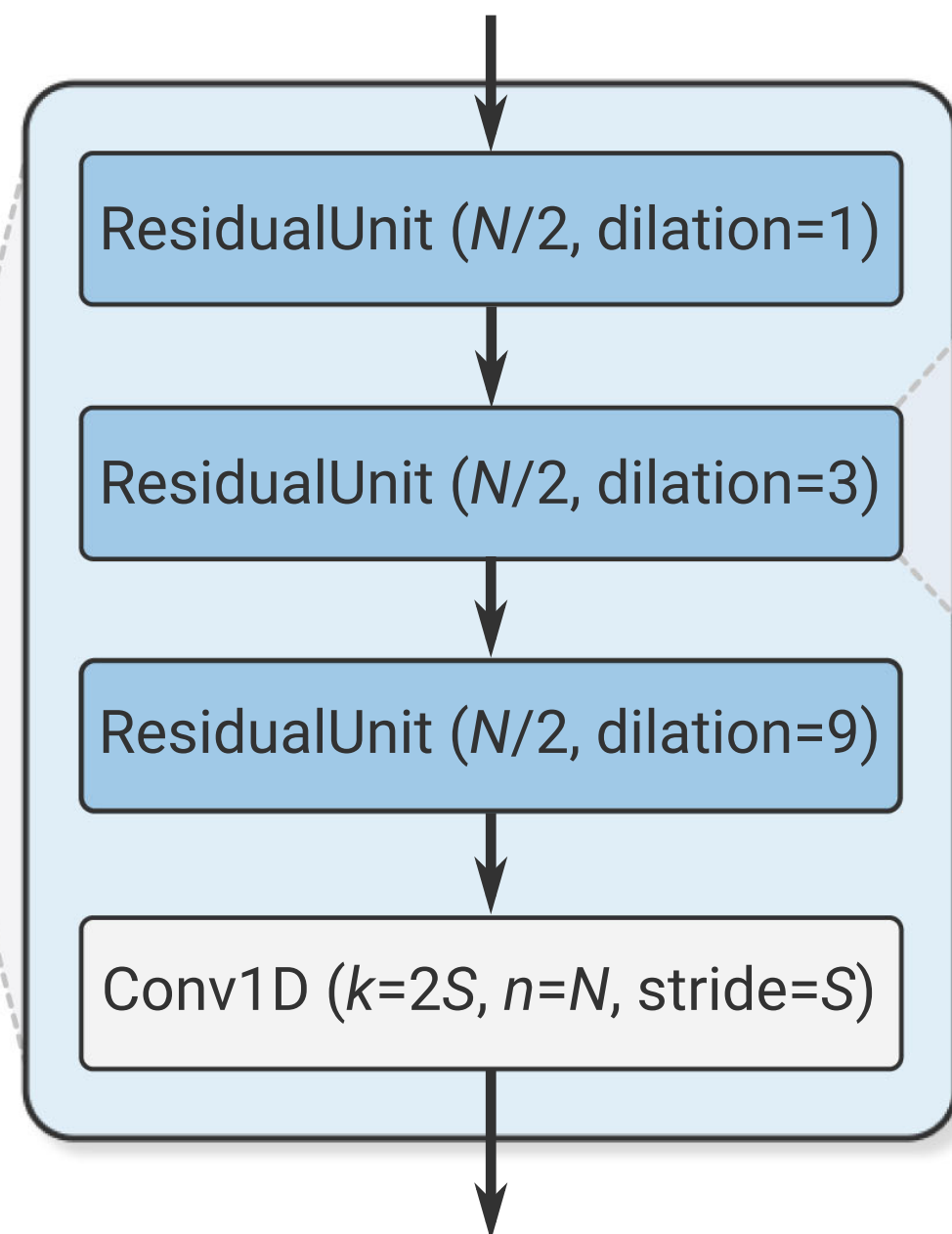
- Residual quantisation
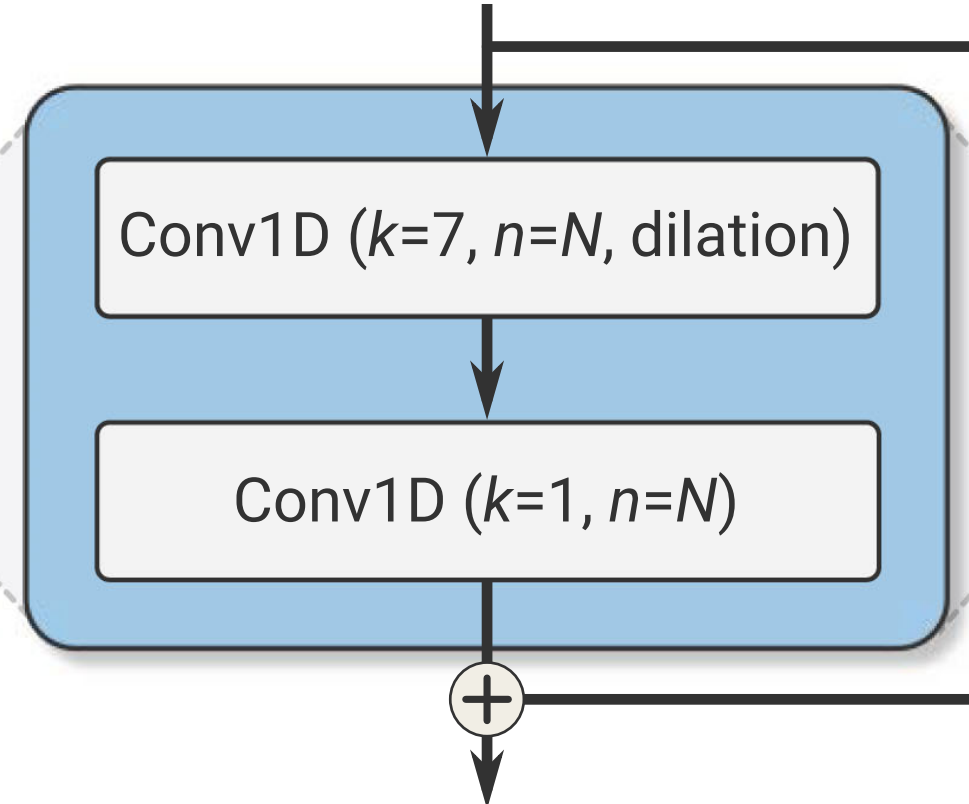 - to avoid the need for a very large set of codes

Encoder

Waveform @ 24 kHz

Conv1D (k=7, n=C)

EncoderBlock (N=$2C$, S=2)

EncoderBlock (N=$4C$, S=4)

EncoderBlock (N=$8C$, S=5)

EncoderBlock (N=$16C$, S=8)

Conv1D (k=3, n=D)

FiLM conditioning

D-dimensional embeddings @ 75 Hz

EncoderBlock (N, S)

ResidualUnit ($N/2$, dilation=1)

ResidualUnit ($N/2$, dilation=3)

ResidualUnit ($N/2$, dilation=9)

Conv1D (k=$2S$, n=N, stride=S)

ResidualUnit (N, dilation)

Conv1D (k=7, n=N, dilation)

Conv1D (k=1, n=N)

$\oplus$

DecoderBlock (N, S)

(Conv1D)^T (k=$2S$, n=N, stride=S)

ResidualUnit (N, dilation=1)

ResidualUnit (N, dilation=3)

ResidualUnit (N, dilation=9)

Decoder

D-dimensional embeddings @ 75 Hz

FiLM conditioning

Conv1D (k=7, n=$16C$)

DecoderBlock (N=$8C$, S=8)

DecoderBlock (N=$4C$, S=5)

DecoderBlock (N=$2C$, S=4)

DecoderBlock (N=C, S=2)

Conv1D (k=7, n=1)

Waveform @ 24 kHz

Module state-of-the-art (1 of 2)
Class

Encoder

Waveform @ 24 kHz

Conv1D (k=7, n=C)

EncoderBlock (N=2C, S=2)

EncoderBlock (N=4C, S=4)

EncoderBlock (N=8C, S=5)

EncoderBlock (N=16C, S=8)

Conv1D (k=3, n=D)

FiLM conditioning

D-dimensional embeddings @ 75 Hz

EncoderBlock (N, S)

ResidualUnit (N/2, dilation=1)

ResidualUnit (N/2, dilation=3)

ResidualUnit (N/2, dilation=9)

Conv1D (k=2S, n=N, stride=S)

ResidualUnit (N, dilation)

Conv1D (k=7, n=N, dilation)

Conv1D (k=1, n=N)

$+$

DecoderBlock (N, S)

(Conv1D)^T (k=2S, n=N, stride=S)

ResidualUnit (N, dilation=1)

ResidualUnit (N, dilation=3)

ResidualUnit (N, dilation=9)

Decoder

D-dimensional embeddings @ 75 Hz

FiLM conditioning

Conv1D (k=7, n=16C)

DecoderBlock (N=8C, S=8)

DecoderBlock (N=4C, S=5)

DecoderBlock (N=2C, S=4)

DecoderBlock (N=C, S=2)
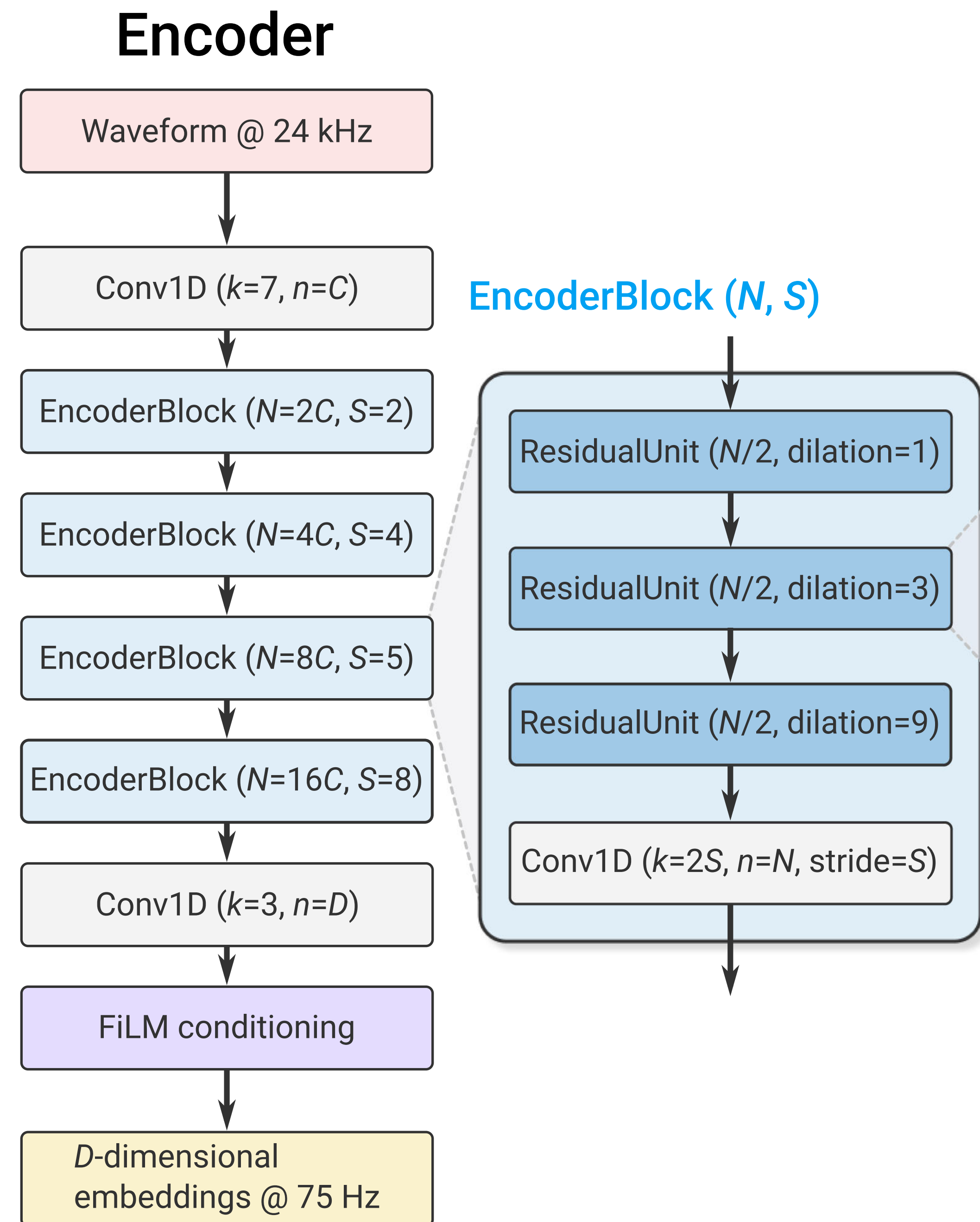
Conv1D (k=7, n=1)
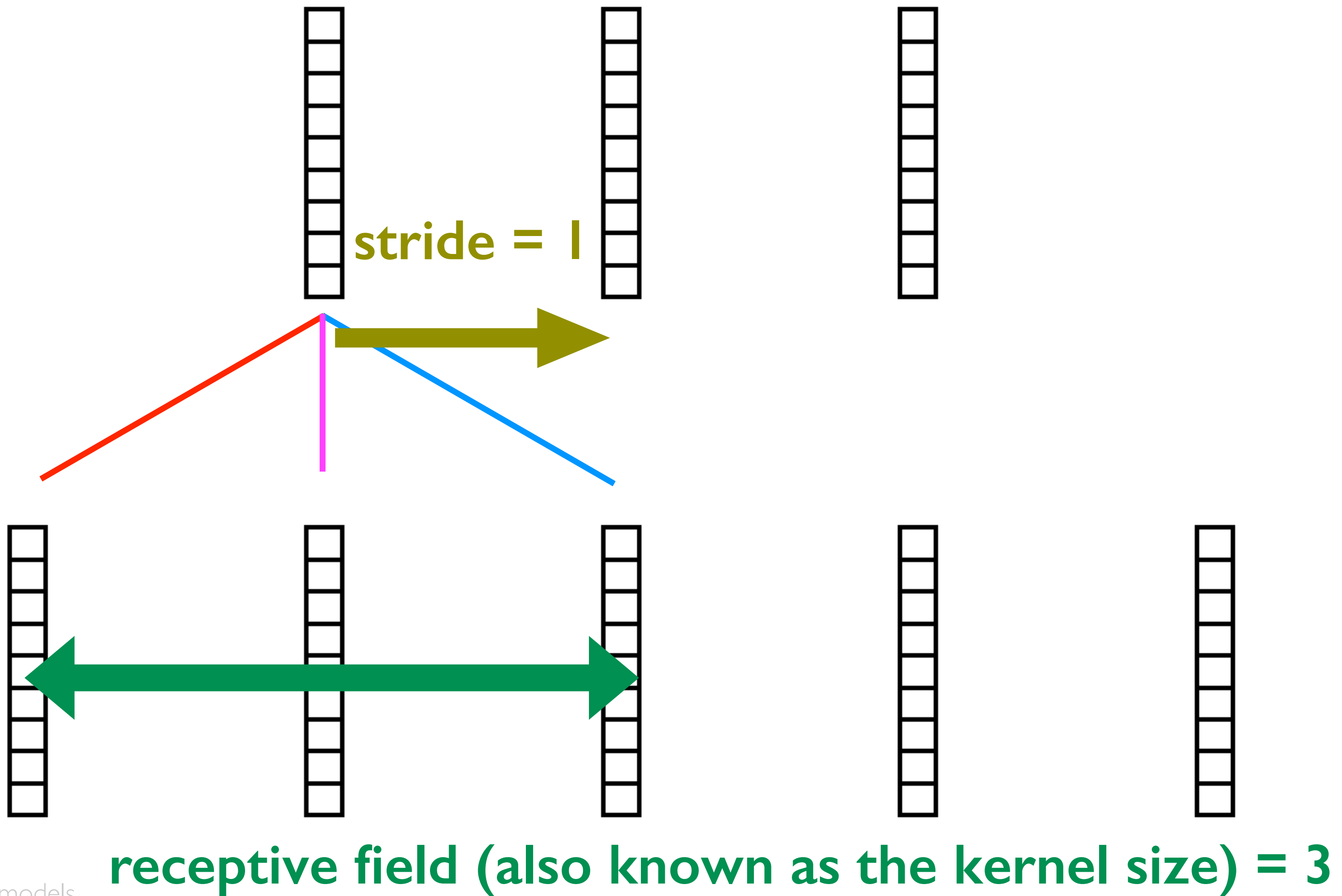
Waveform @ 24 kHz

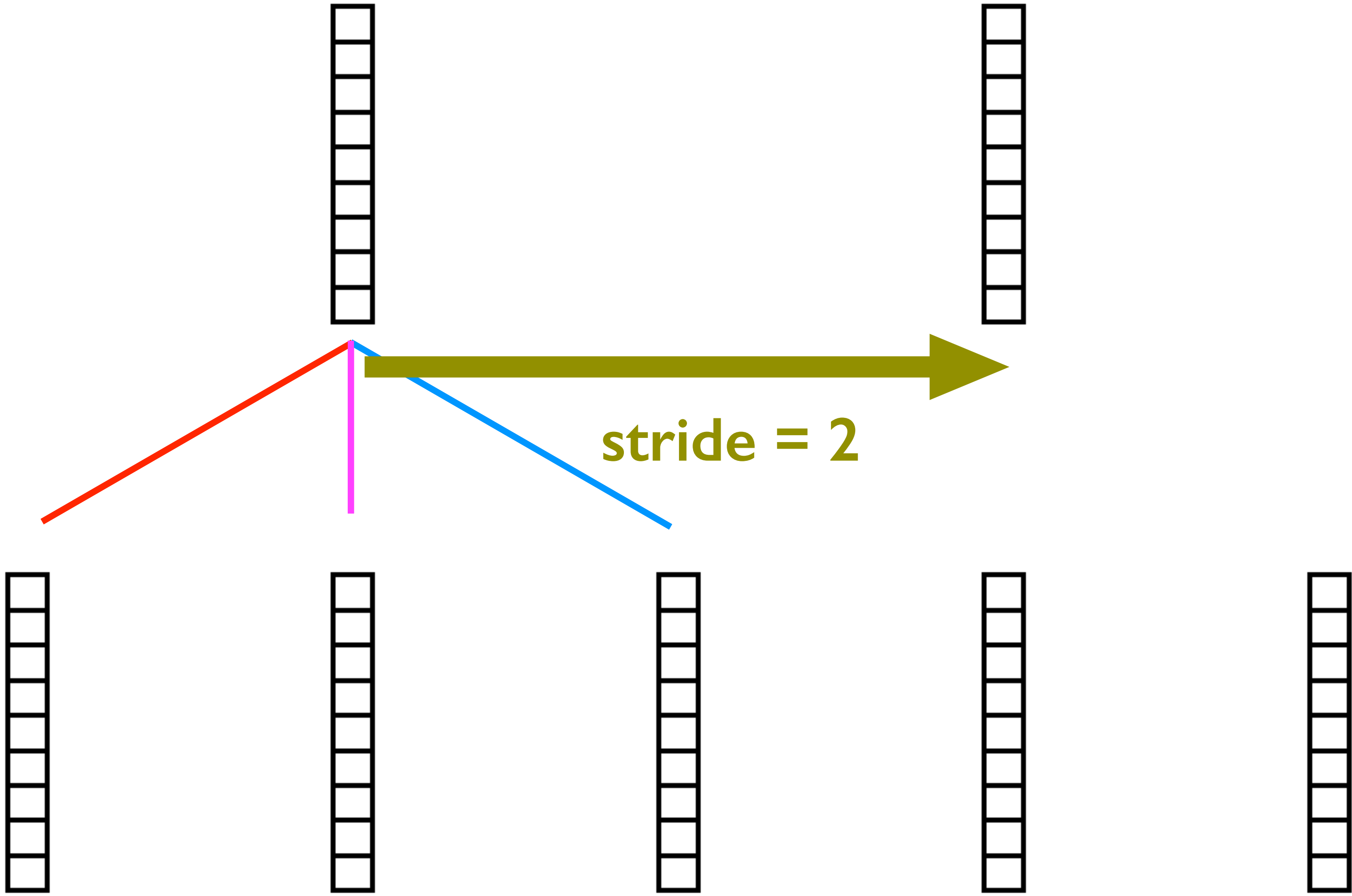# Reduce sequence length

- Fully convolutional encoder

- EncoderBlock

  - convolutional layers with

    - **stride**

    - **dilation**

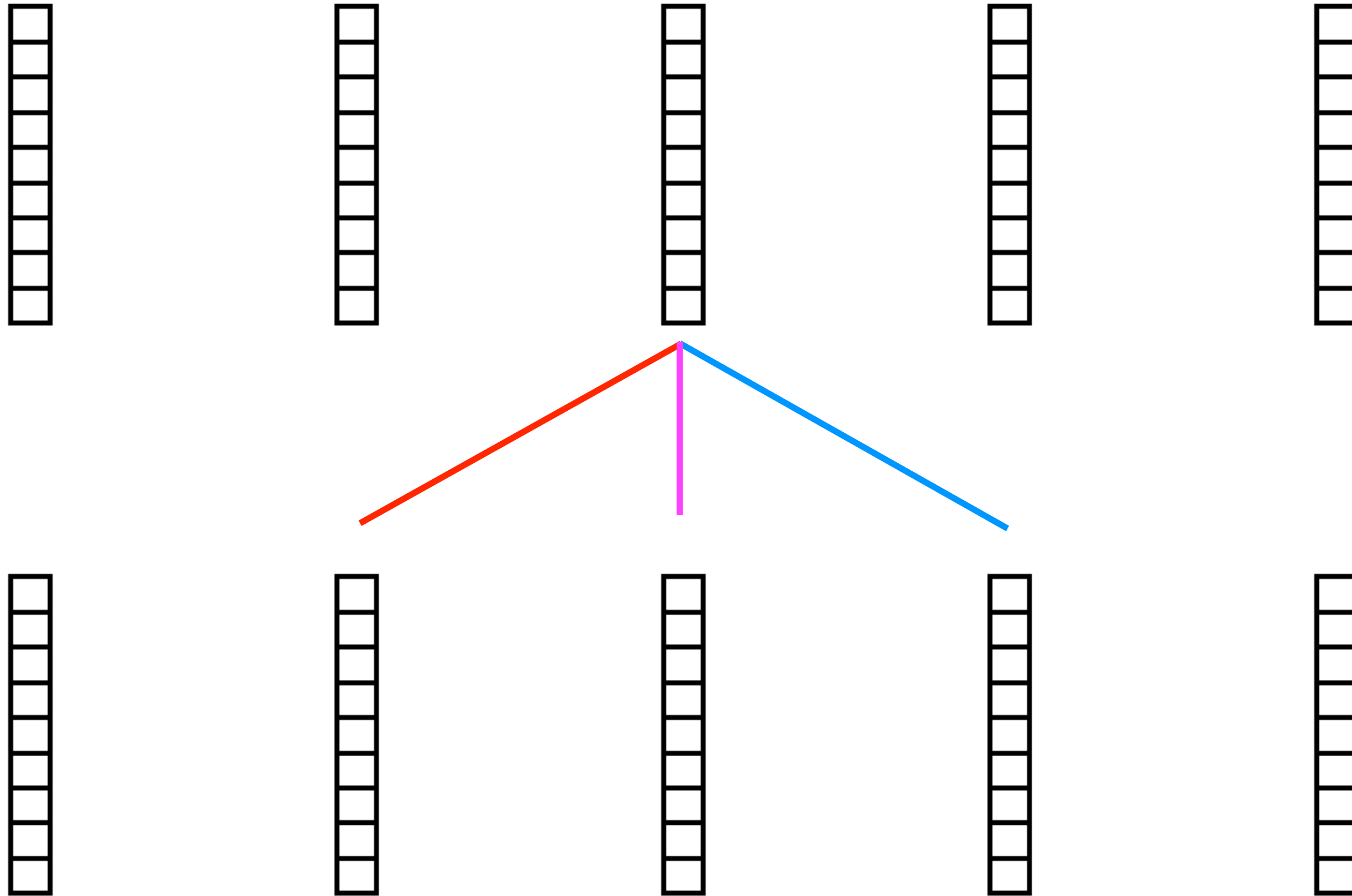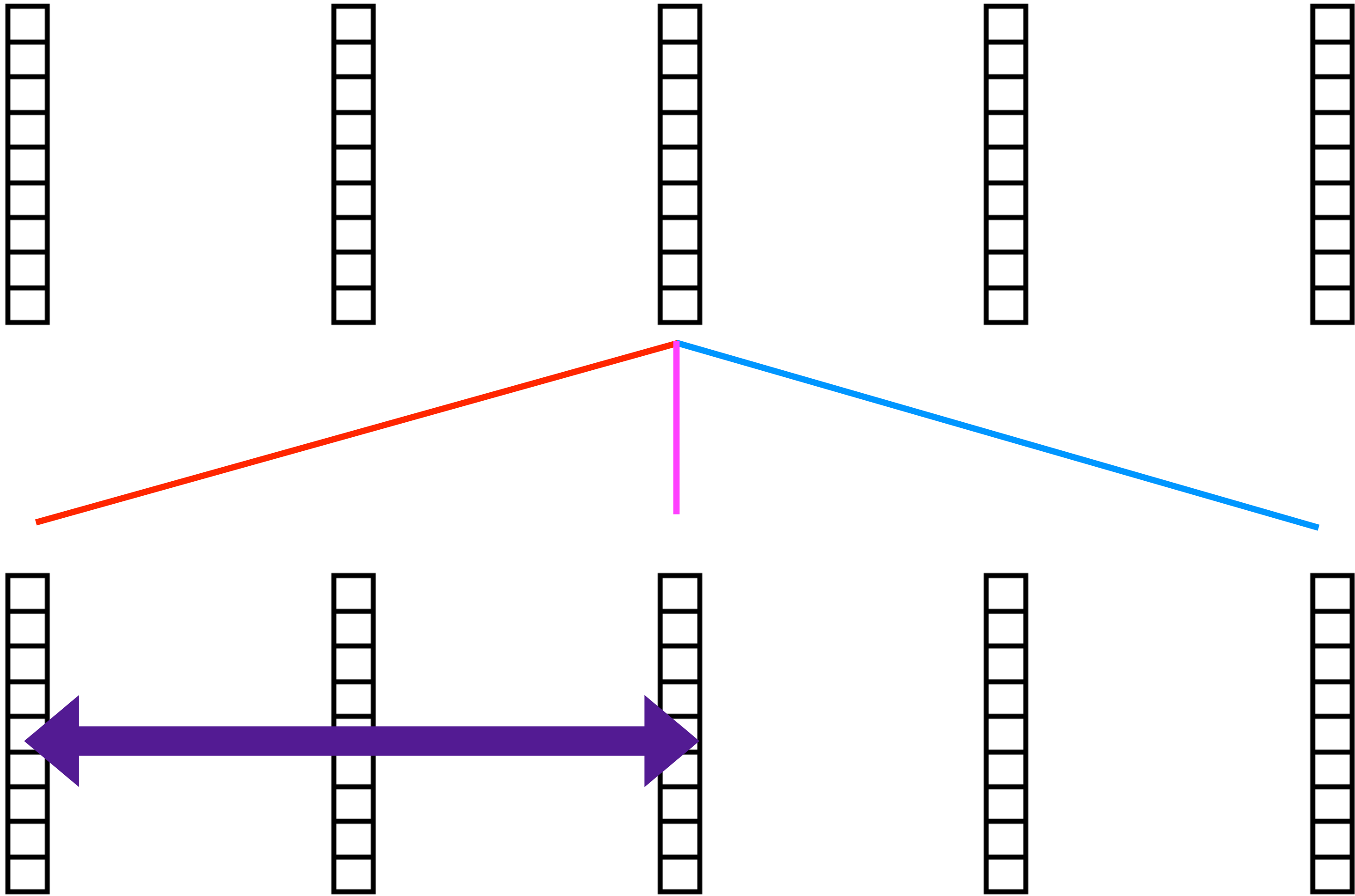- then stack as many EncoderBlocks as necessary to get from 24 000 Hz to 75 Hz

Encoder

Recap: convolution

stride = 1

receptive field (also known as the kernel size) = 3

Increasing the stride = downsampling

stride = 2

Dilation

# Dilation = wider receptive field with fewer parameters

dilation = 2

Key ideas in SoundStream (and similar audio encoder-decoder models)

- Reduce sequence length

  - input is a waveform, at a typical sample rate of 24 kHz (samples per second)

  - encoded representation is at 75 Hz (frames per second)

  - progressively reduce from 24 kHz to 75 Hz

- **Quantise the learned representation**

  - so it becomes a sequence of codes, not vectors

- Residual quantisation

  - to avoid the need for a very large set of codes

Model learns a representation that is a sequence of
256-dimensional **vectors**, at a rate of 75 frames per second
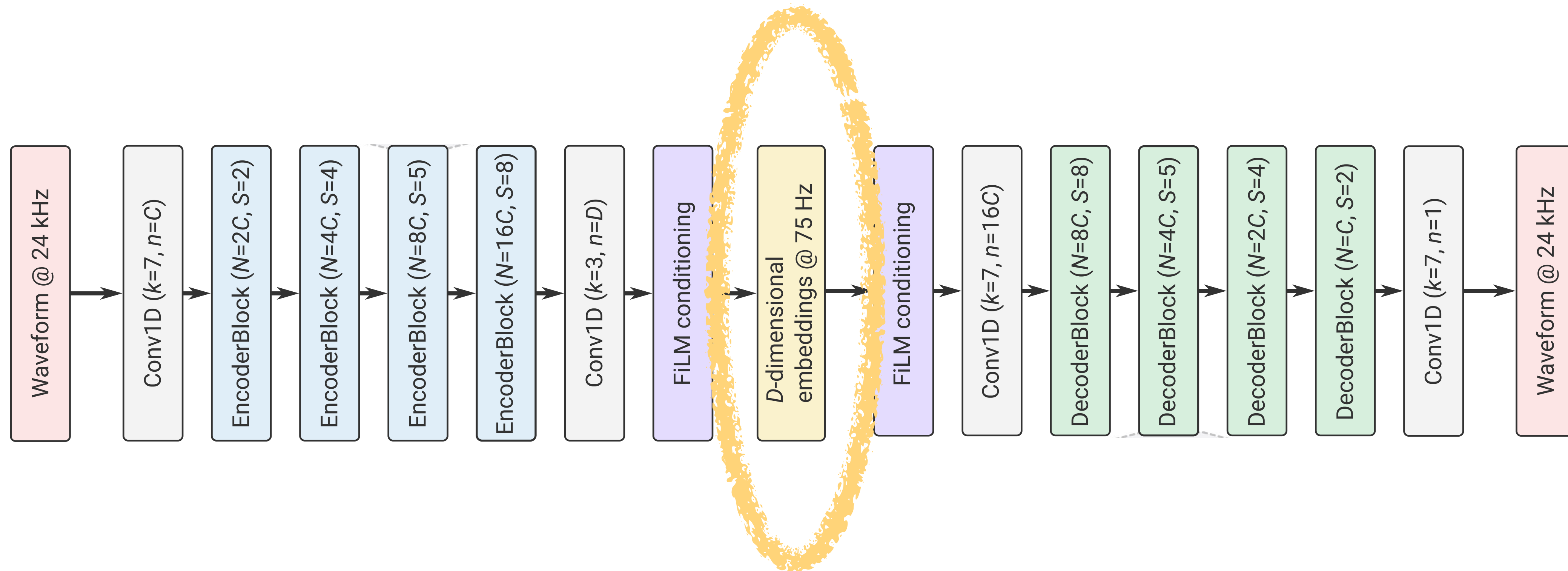
To further compress, each vector will be converted to a **code** using quantisation

Quantisation

- Convert a continuous value to a discrete one

- Why?

  - continuous values can take an "infinite" number of values (in a computer, this isn't quite true, but they do require a lot of bits to store, e.g., 64)

  - discrete values from a relatively small closed set require only a few bits to store

e.g., using a codebook size of 1024 requires only 10 bits to store/transmit each code

Vector quantisation

- Same as before but now for vectors (!)

- *Could* quantise each dimension independently, but that would not take advantage of correlations

- So, quantise the whole vector at once

- The result is a **codebook**
  - a mapping from codes to vectors
  - can use this to *encode* a new vector into a code
  - or to *decode* a code back to a vector

note: a code is also known as a "codeword"

Key ideas in SoundStream (and similar audio encoder-decoder models)

- <u>Reduce sequence length</u>

  - input is a waveform, at a typical sample rate of 24 kHz (samples per second)

  - encoded representation is at 75 Hz (frames per second)

  - progressively reduce from 24 kHz to 75 Hz

- <u>Quantise the learned representation</u>

  - so it becomes a sequence of codes, not vectors

- **Residual quantisation**

  - to avoid the need for a very large set of codes

Residual vector quantisation

- First perform "coarse" quantisation
  - "coarse" = small codebook

- Now measure the **error** that was introduced by quantisation

- Then represent that error with another codebook

- Repeat as many times as you wish, to reduce final error to desired level

What next?

- Just one more class left...

- Large speech language models
- Tasks beyond Text-To-Speech
- Current & future trends