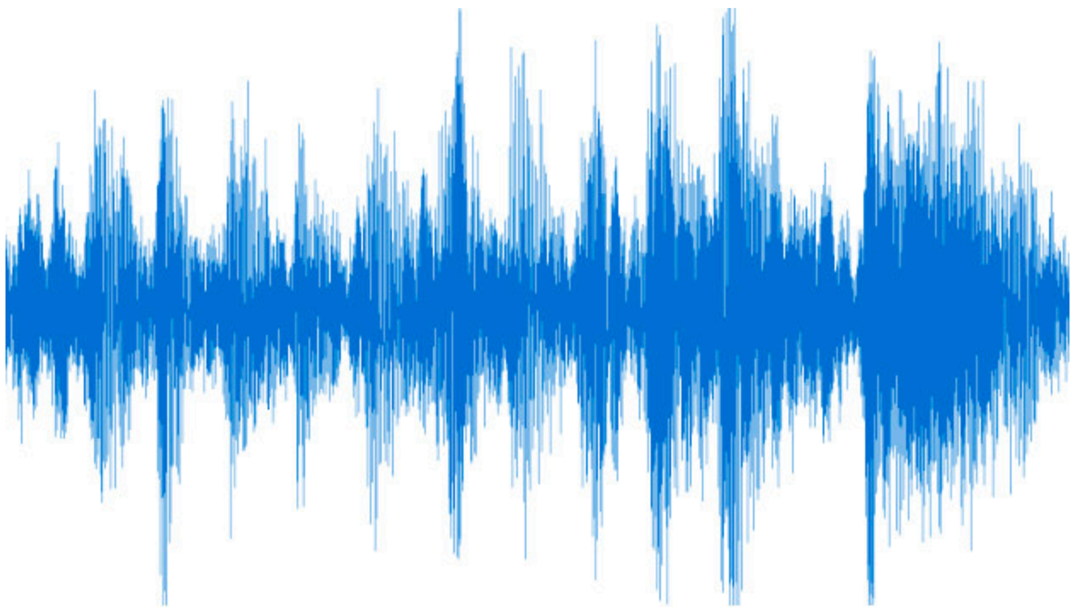
Transformer ASR with Contextual Block Processing

Emiru Tsunoo, Yosuke Kashiwagi, Toshiyuki Kamakura, Shinji Watanabe

Presented by: Desh Raj

Motivation

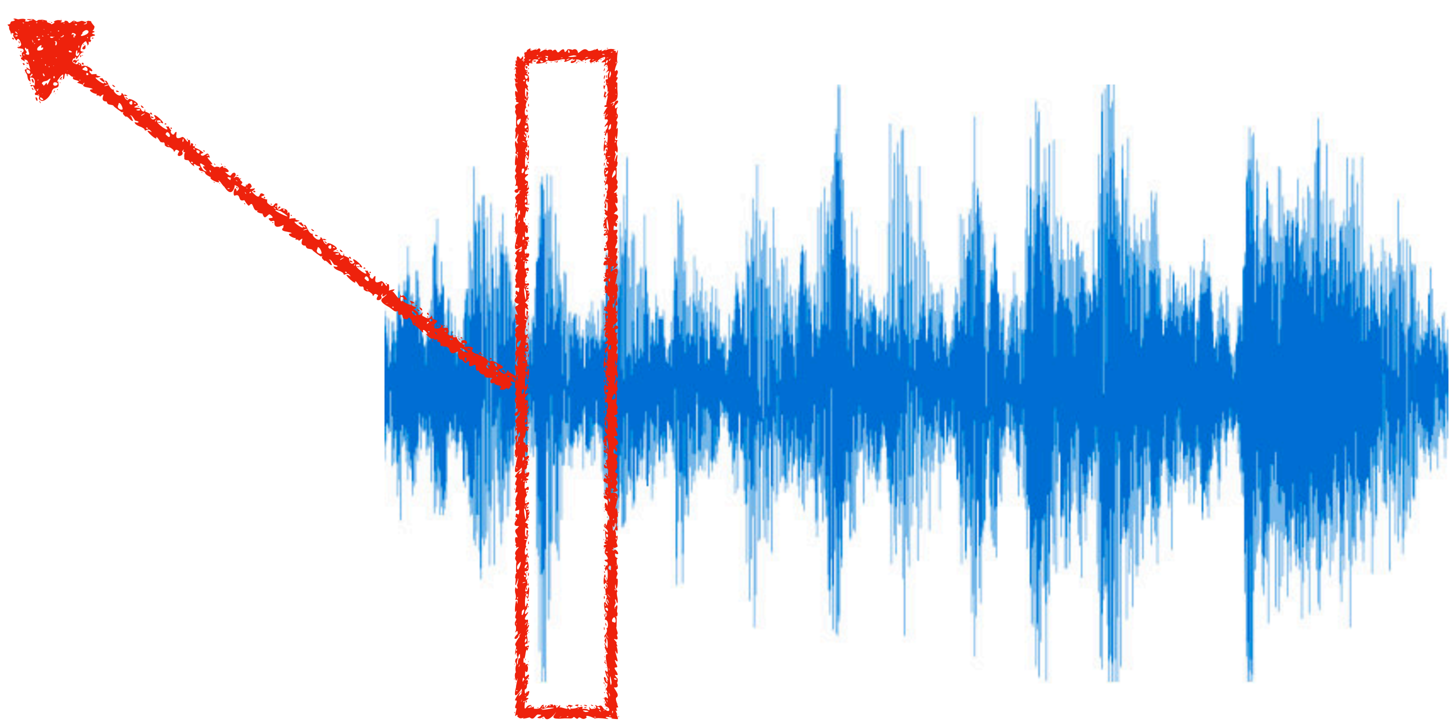
- Local and global characteristics in speech



Motivation

- Local and global characteristics in speech

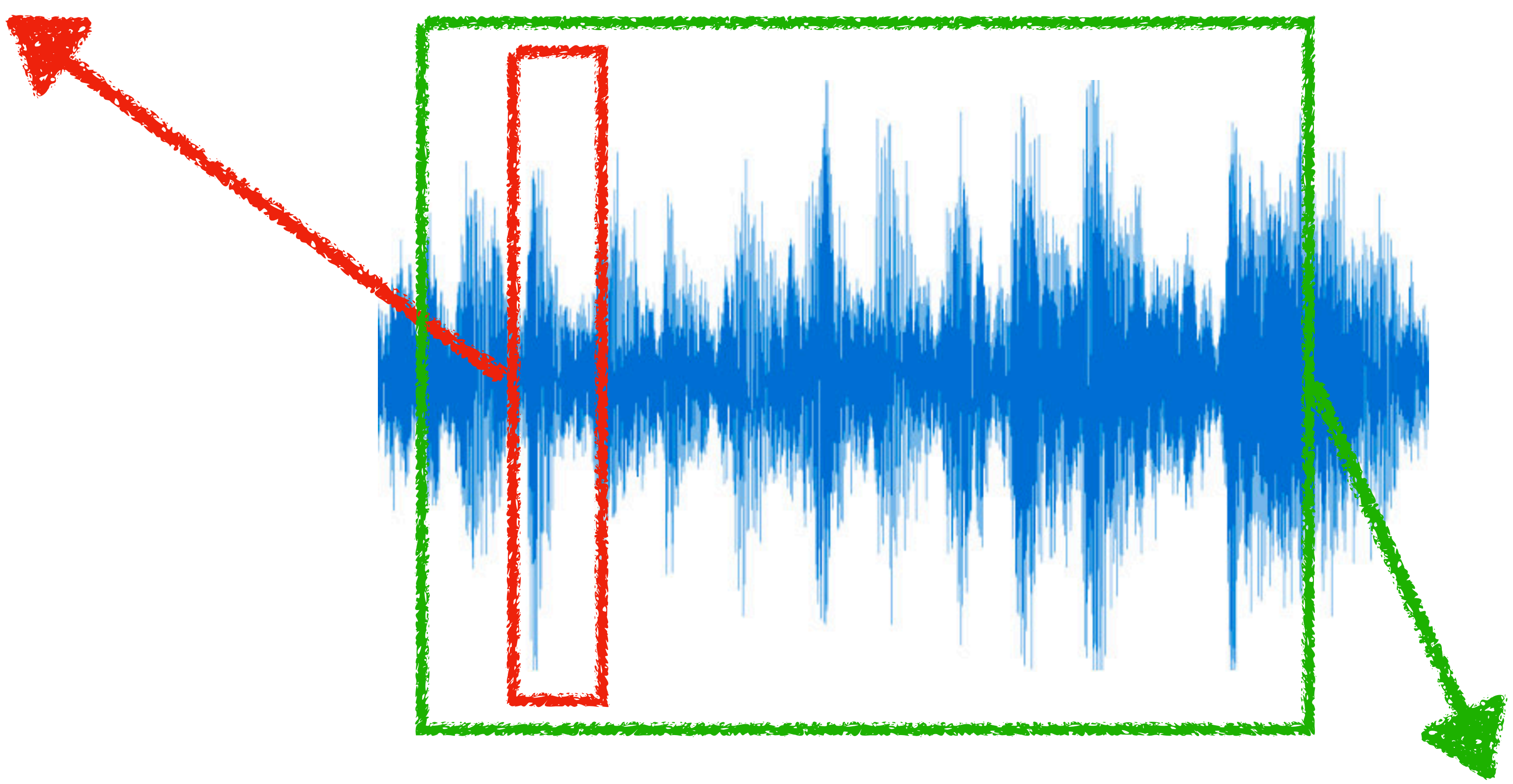
Phonetic events occur at temporally local level



Motivation

- Local and global characteristics in speech

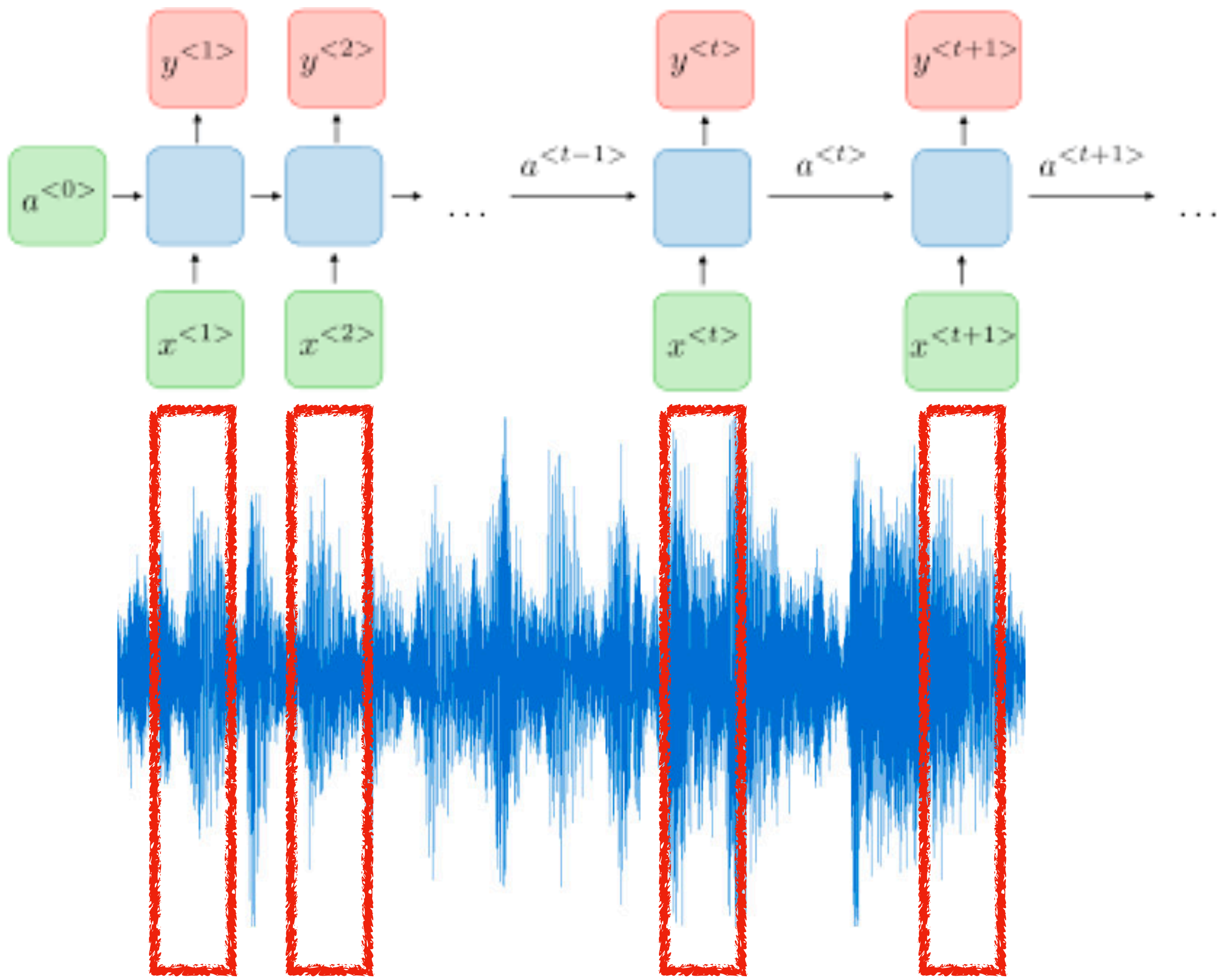
Phonetic events occur at temporally local level



Speaker, channel, and linguistic context exist globally.

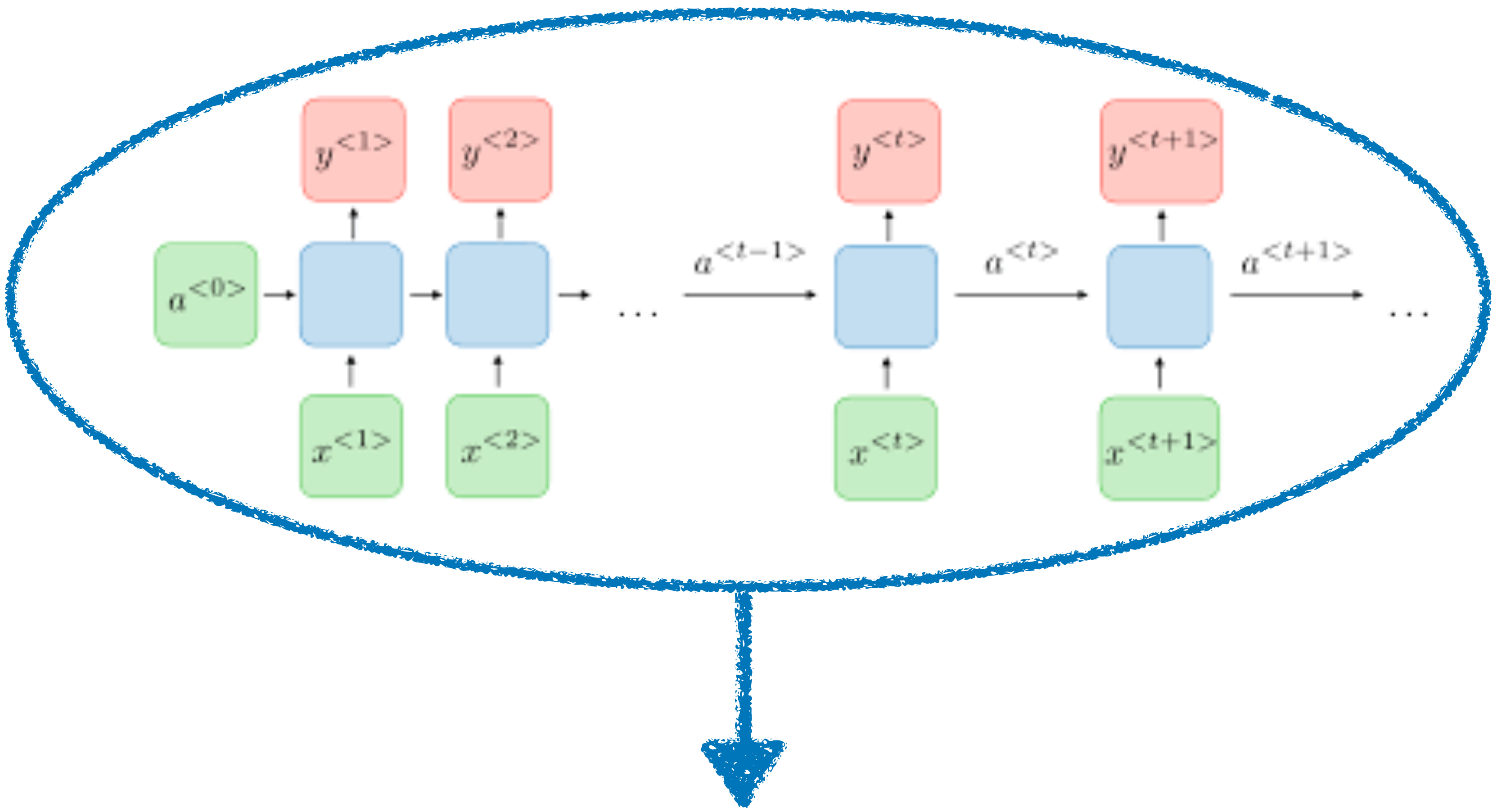
Motivation

- RNNs can exploit both local and global information.



Motivation

- RNNs can exploit both local and global information.



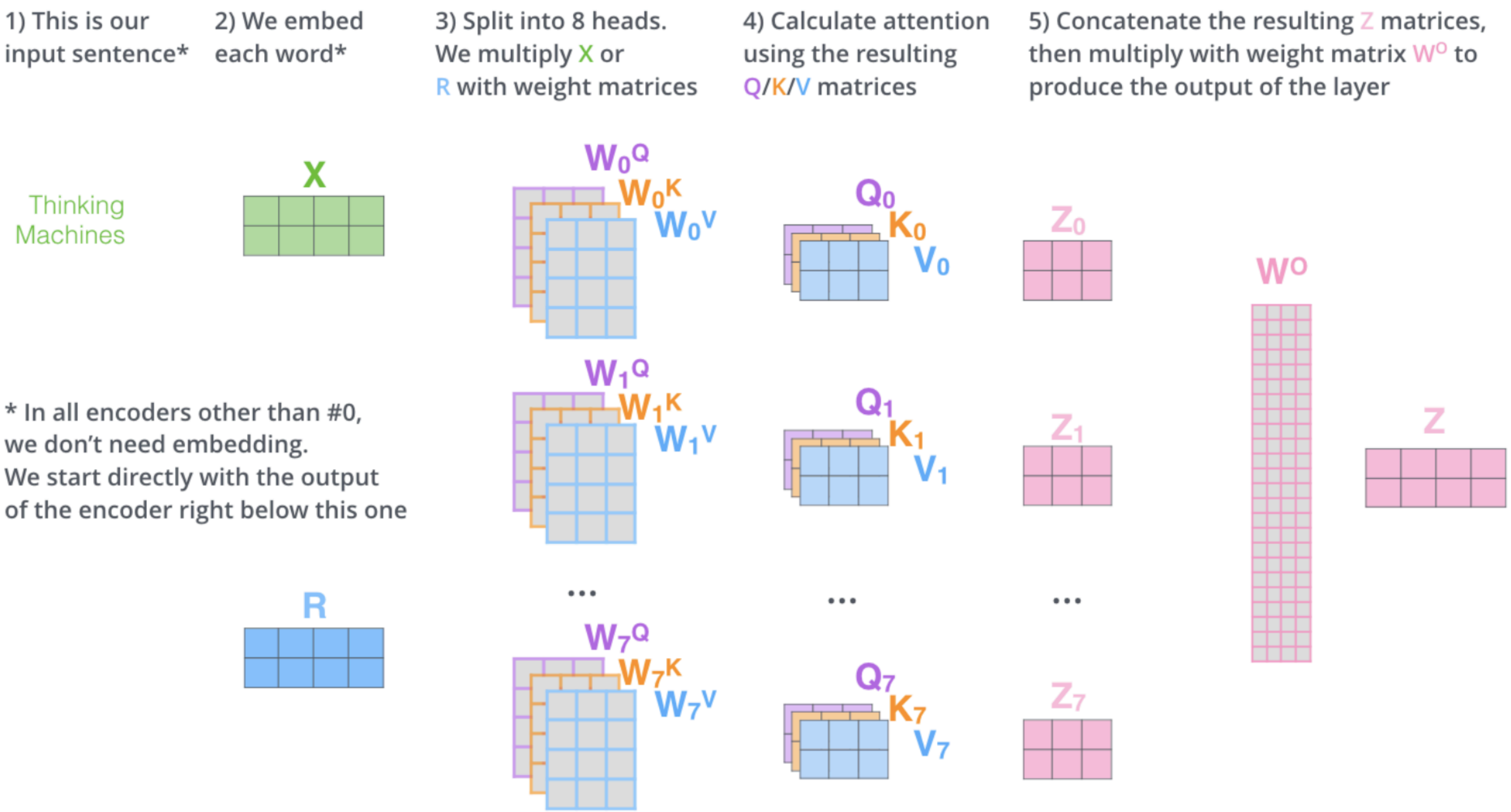
But this computation is sequential!

Motivation

- Is there a way to exploit both global and local features but using batch computation?

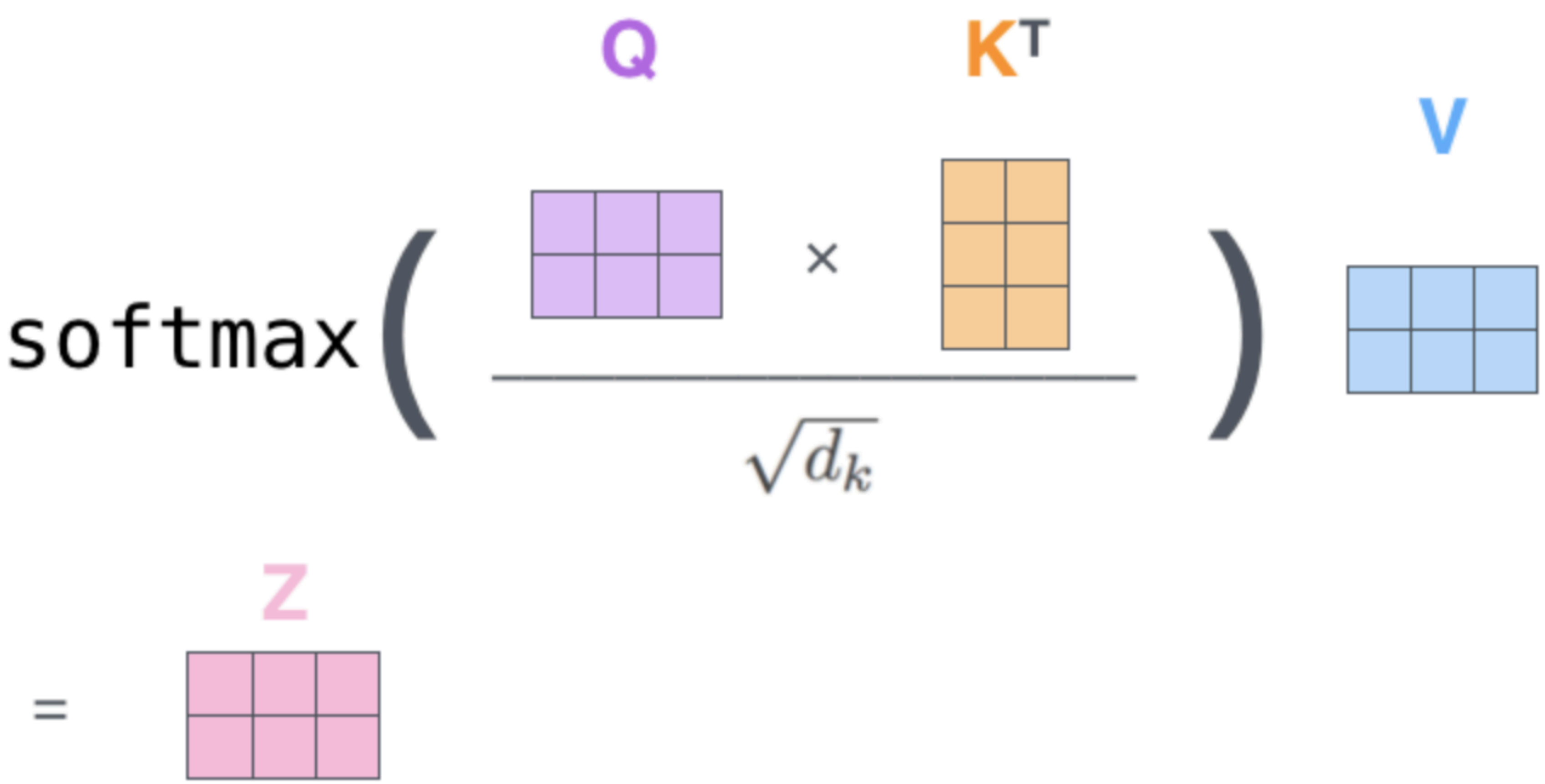
- YES! Use Transformers :)

Transformer



Source: http://jalammar.github.io/illustrated-transformer/

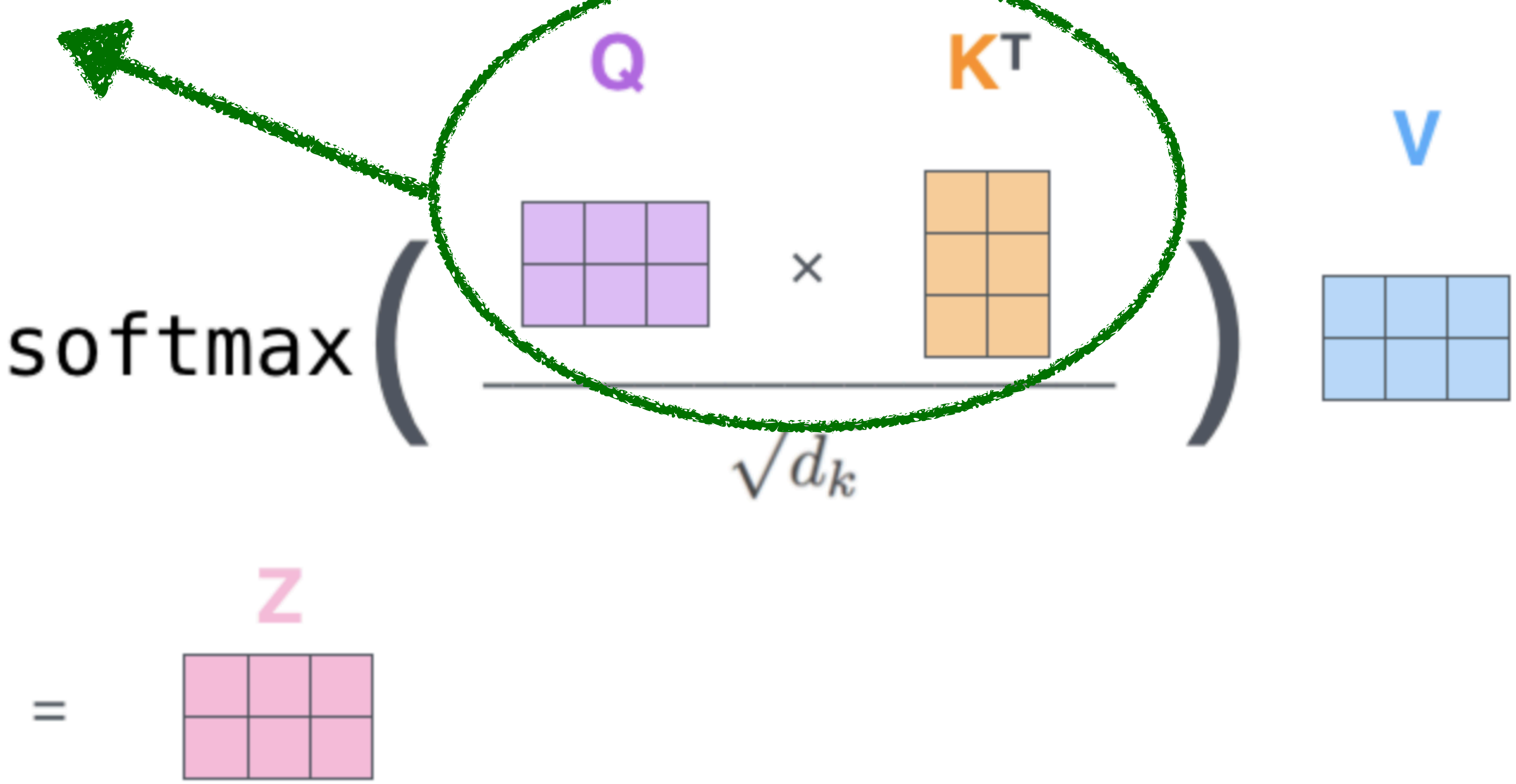
Transformer



The self-attention calculation in matrix form

Transformer

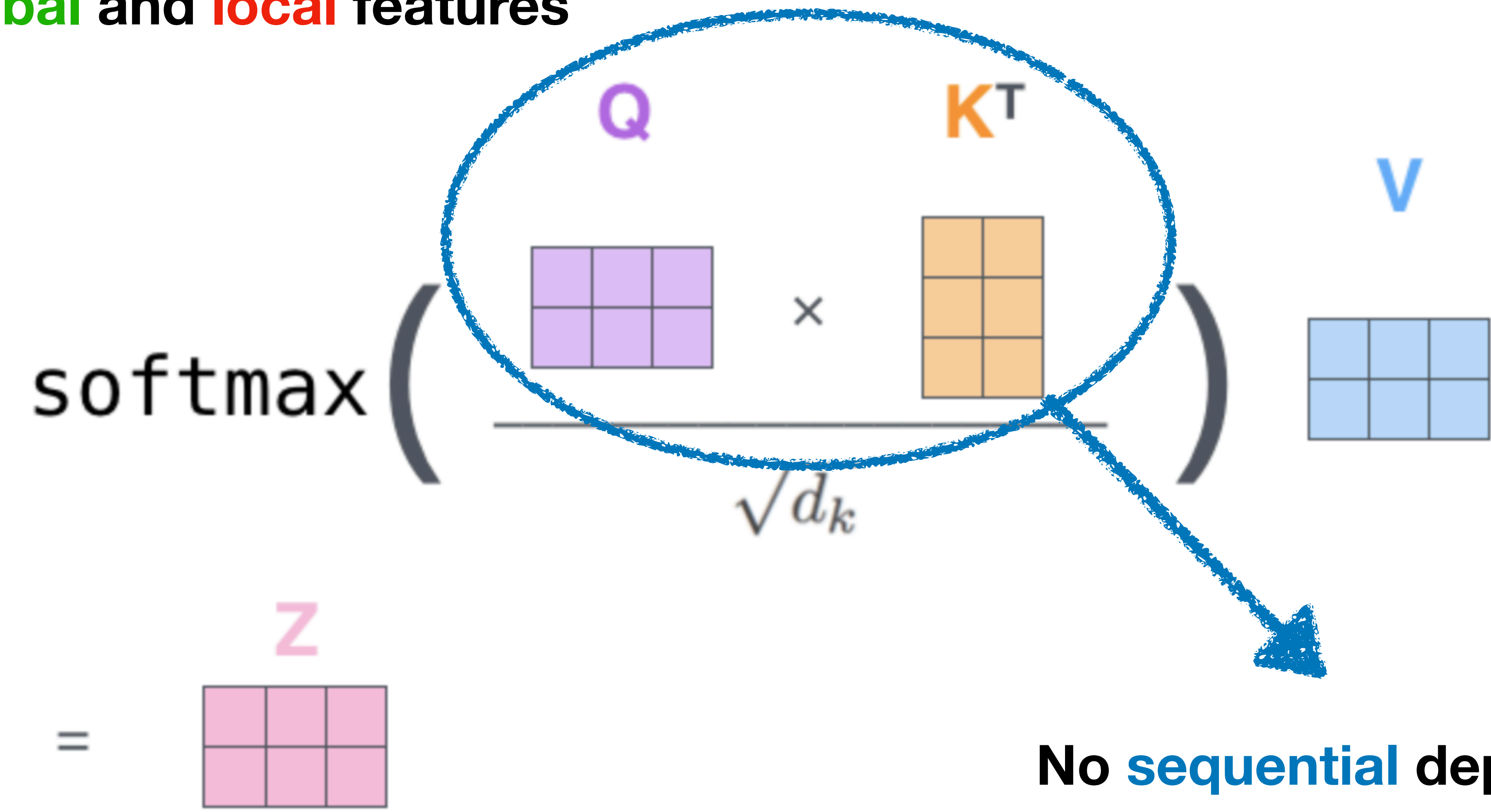
Computed for all pairs of input frames -> global and local features



The self-attention calculation in matrix form

Transformer

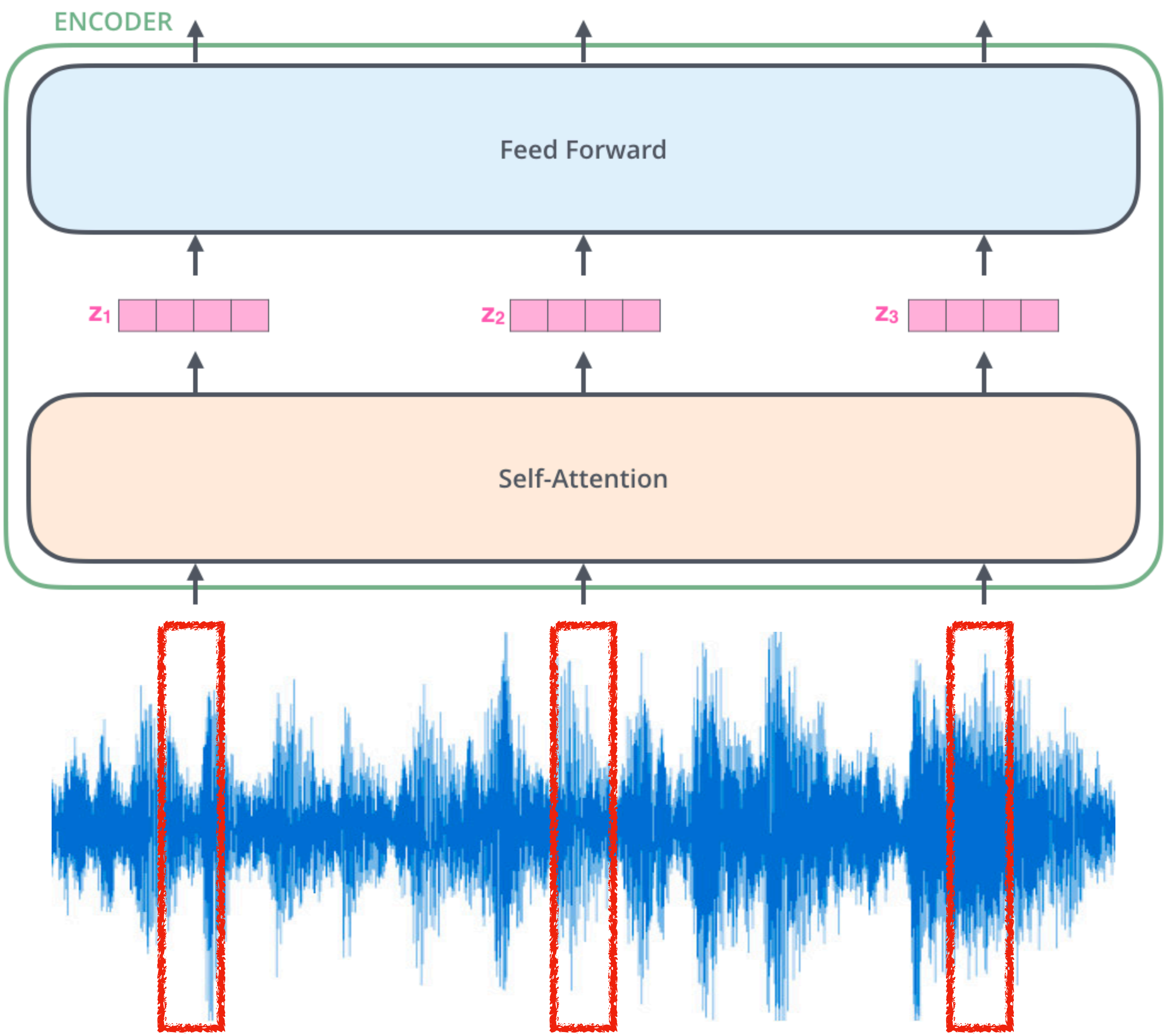
Computed for all pairs of input frames -> global and local features



No sequential dependence

The self-attention calculation in matrix form

Transformer



Transformer

- So everything is perfect, we can all go home now :)
- …but not quite!

Transformer

- So everything is perfect, we can all go home now :)
- …but not quite!

DQ: Can you spot the problems in using Transformers directly for ASR?

Transformer: Limitations

1. For ASR, input sequence can contain thousands of frames! Model has **$O(n^2)$ complexity**.

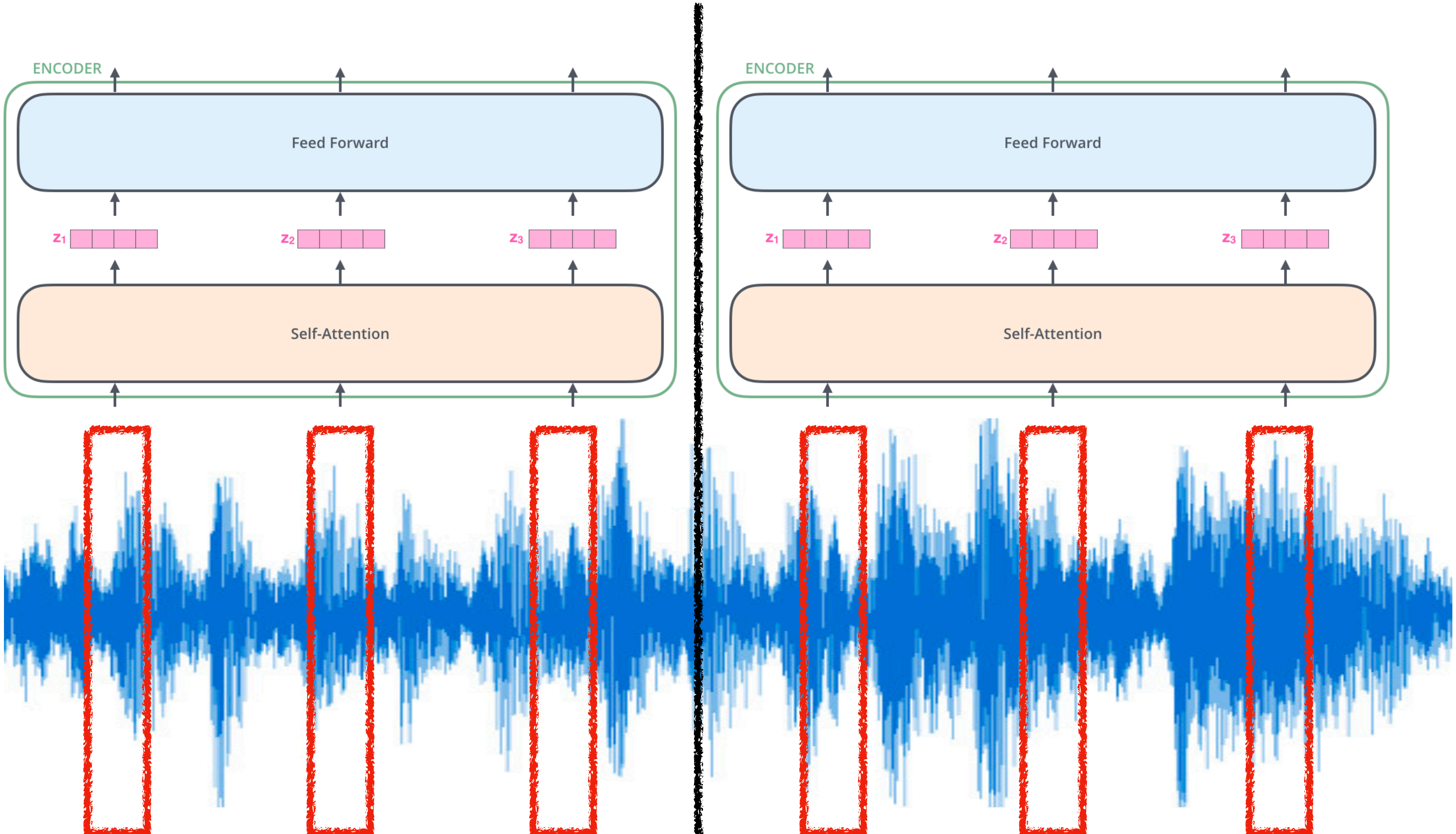
2. Encoder needs to process the whole utterance before decoding can start -> **online ASR** cannot work!

Early efforts

- **Downsampling** before self-attention layer -> reduces sequence size
- **Local masking** -> can compute self-attention for a chunk of frames

Sperber et al. ***Self-attentional acoustic models****. Interspeech 2018.*

Early efforts

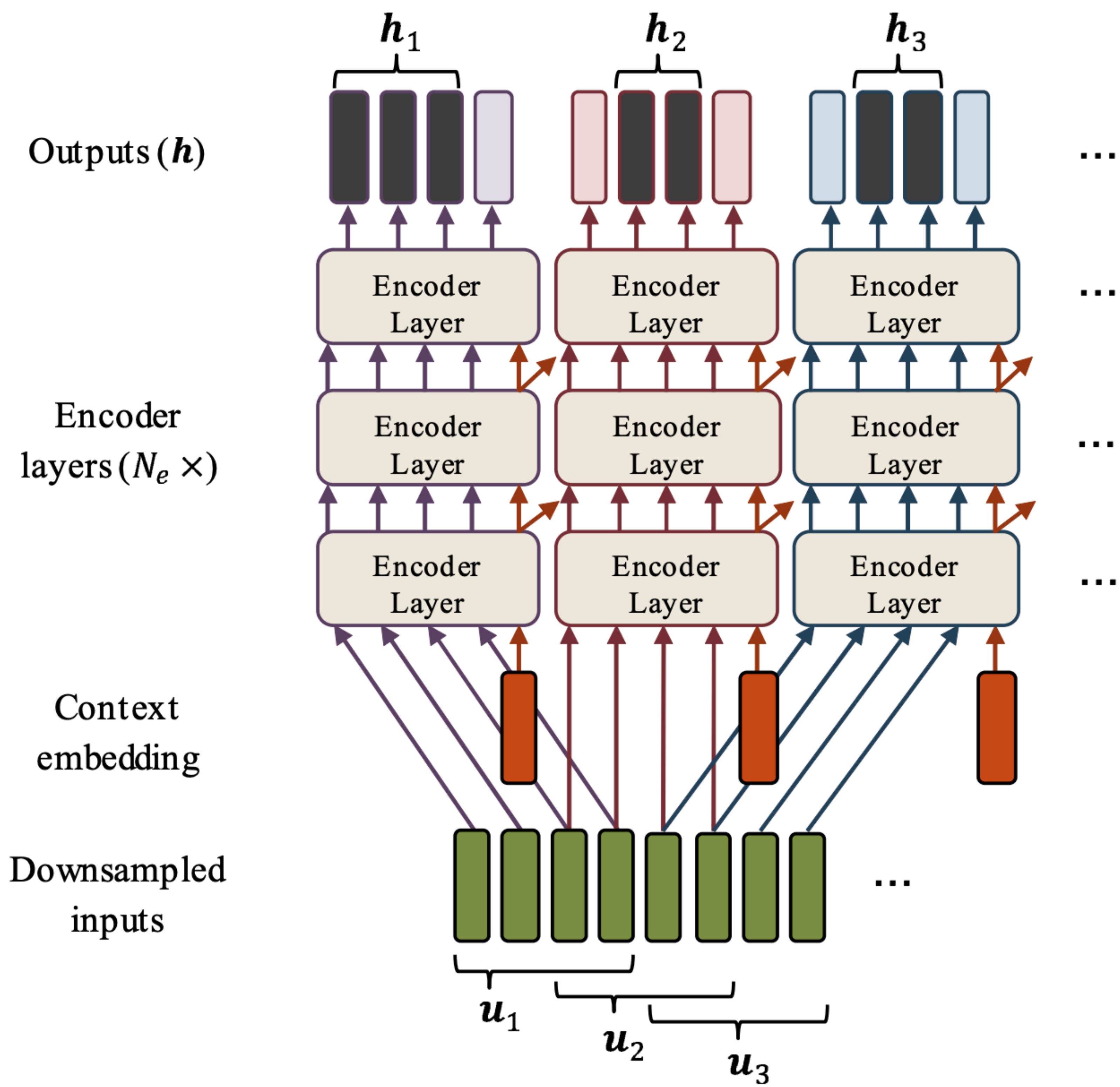


Early efforts

DQ: Can you spot a problem with using such naive block processing?

- We don't have global features anymore. This degrades performance!

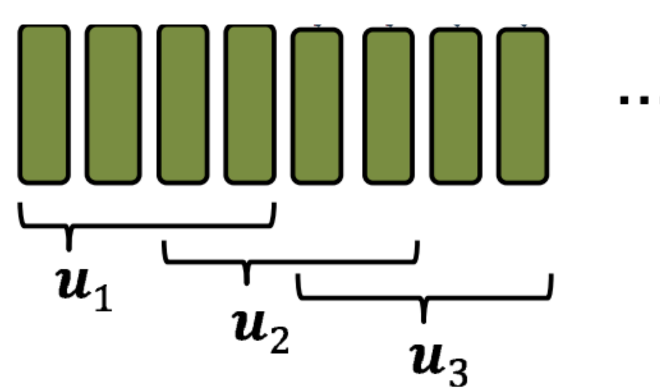
Contextual Block Processing



Contextual Block Processing

2 convolutional layers with stride 2 -> downsampled to T/4 frames

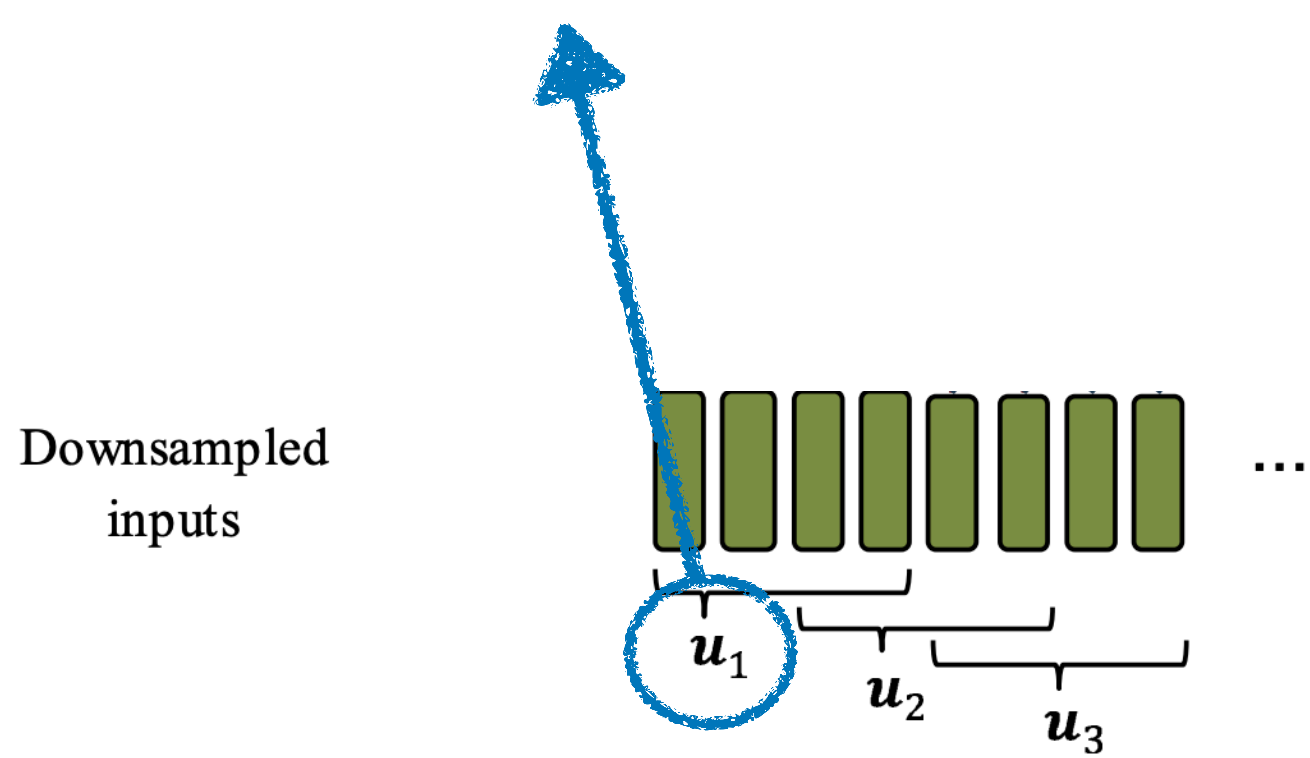
Downsampled inputs



Contextual Block Processing

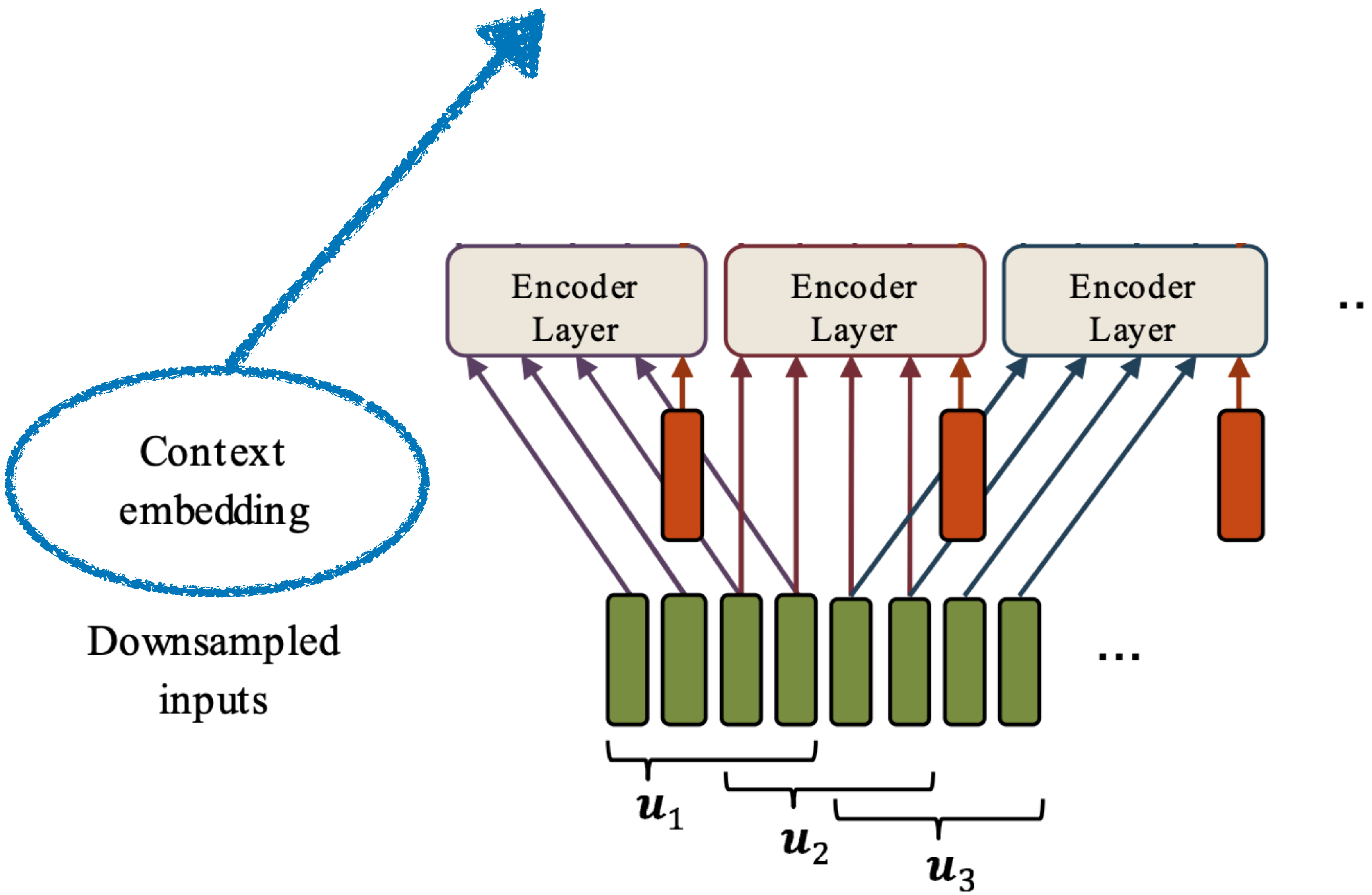
- In experiments, $L_{block} = 2\ L_{hop}$
- Higher L_{block} means more context but lower parallelization

$$\mathbf{u}_b = (u_{(b-1)\cdot L_{\text{hop}}+1}, \ldots, u_{(b-1)\cdot L_{\text{hop}}+L_{\text{block}}})$$



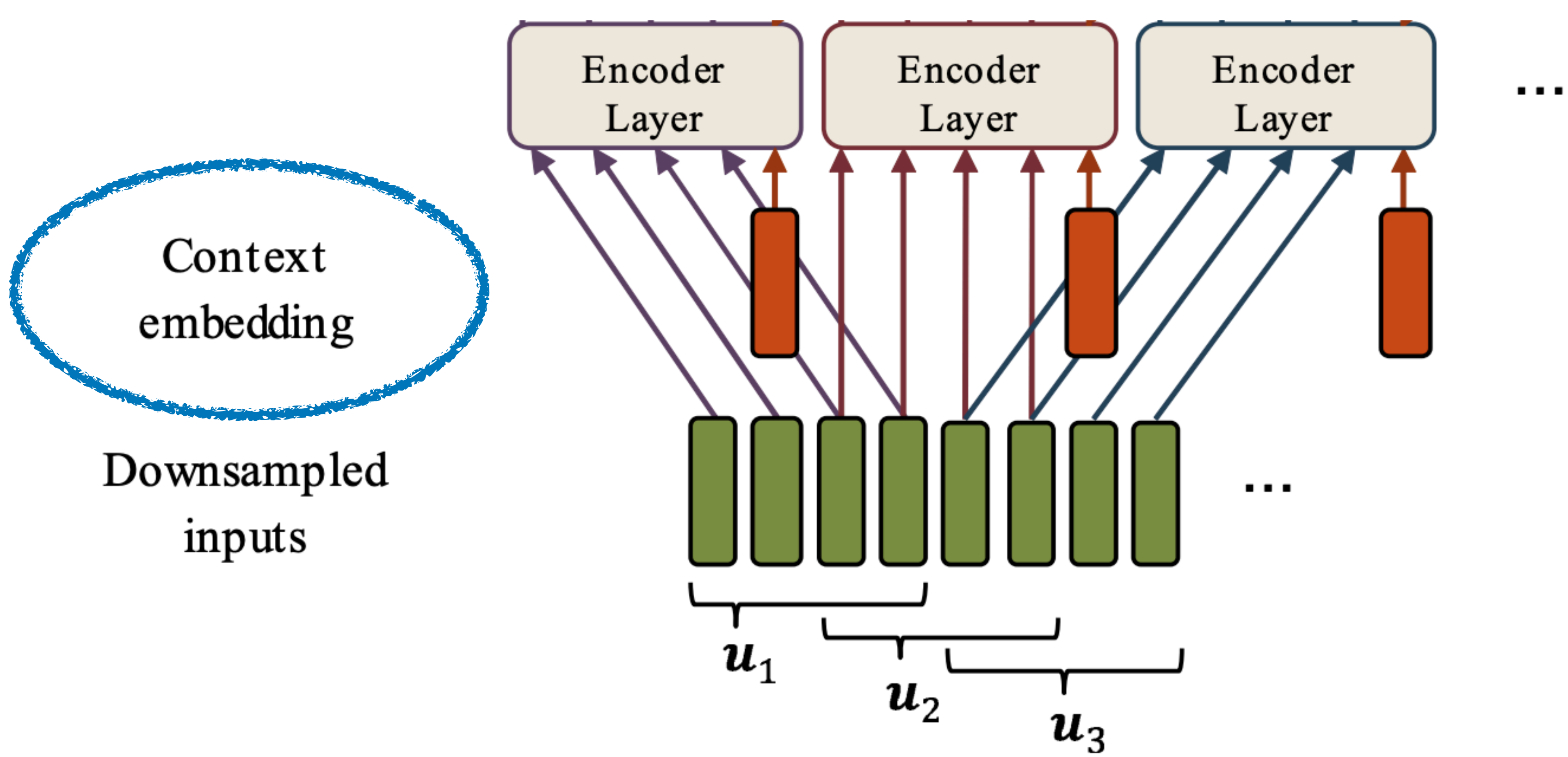
Contextual Block Processing

Additional vector inputs to the encoder

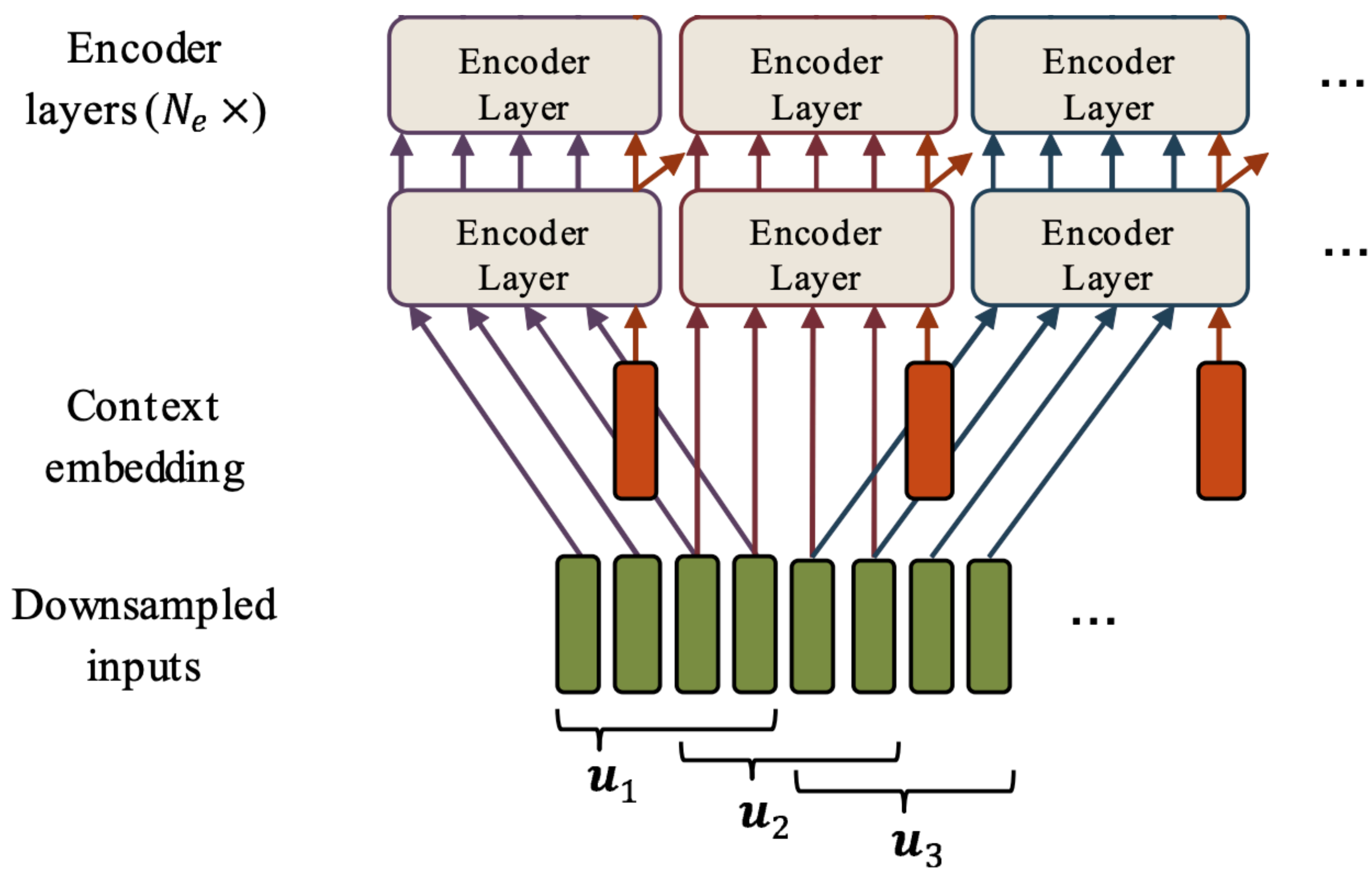


Contextual Block Processing

- **How to initialize?**
    1. Positional encoding
    2. Average input
    3. Maximum input

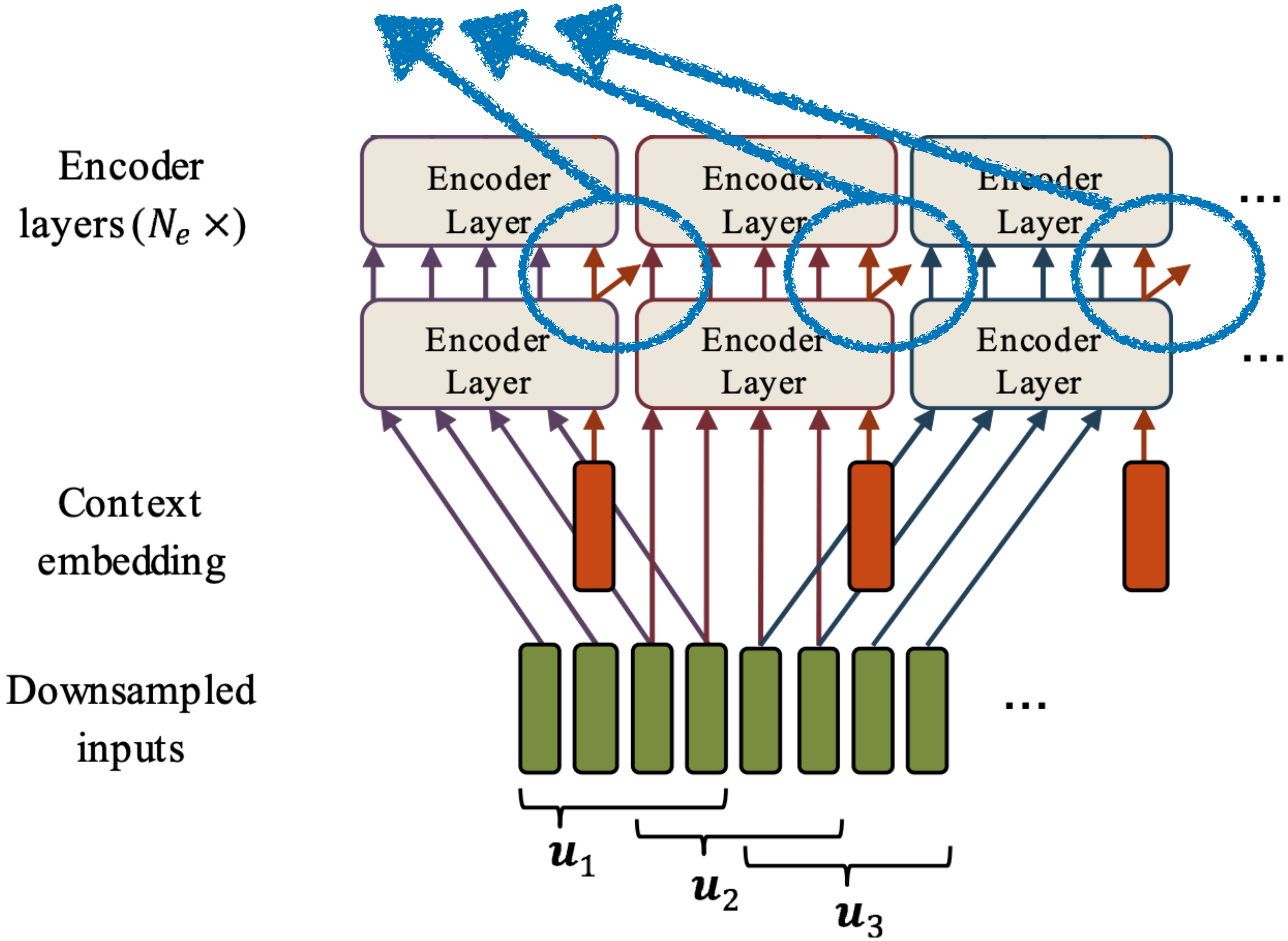


Contextual Block Processing



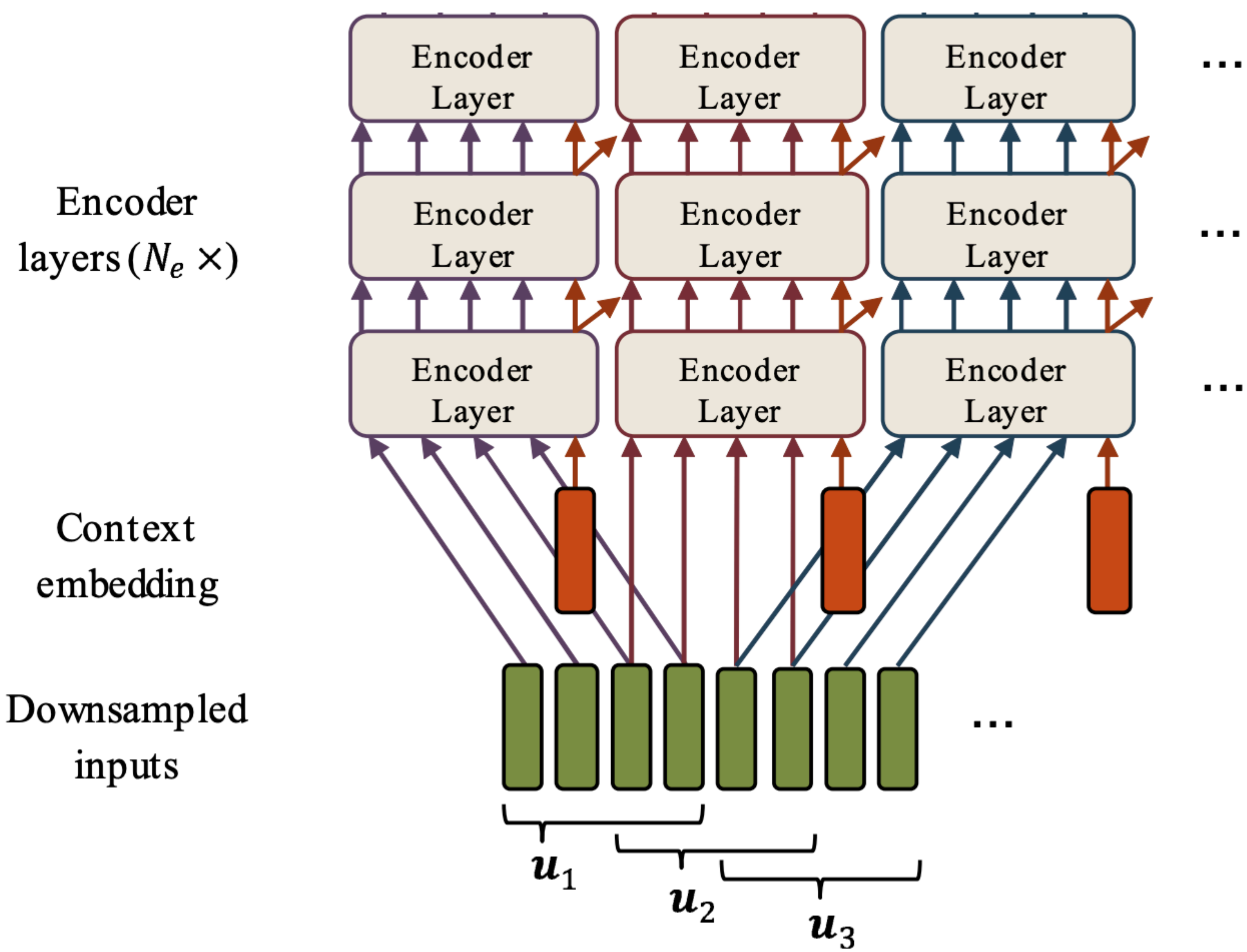
Contextual Block Processing

In each layer, previous block passes a context vector to the next block -> Context Inheritance

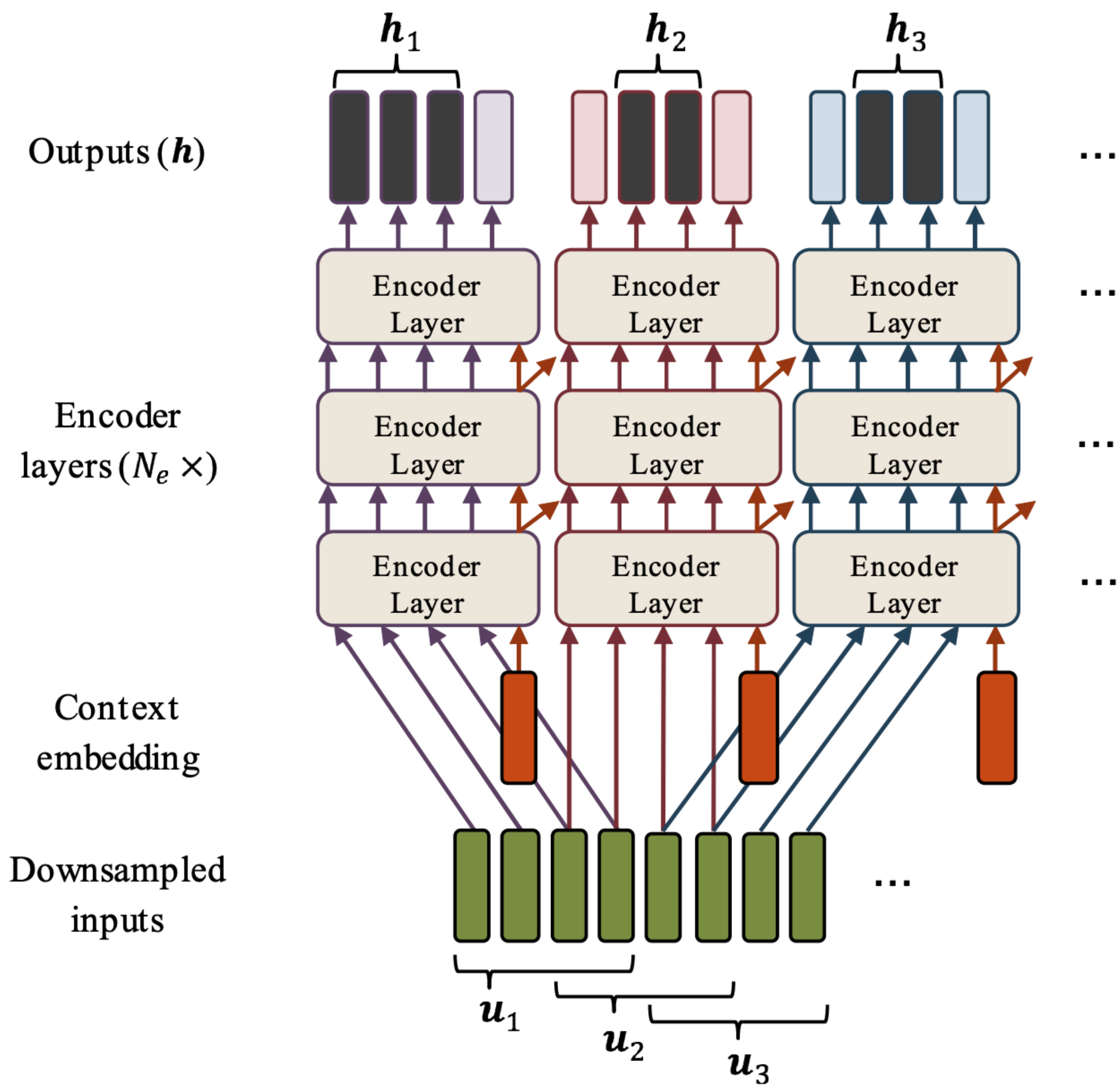


Contextual Block Processing

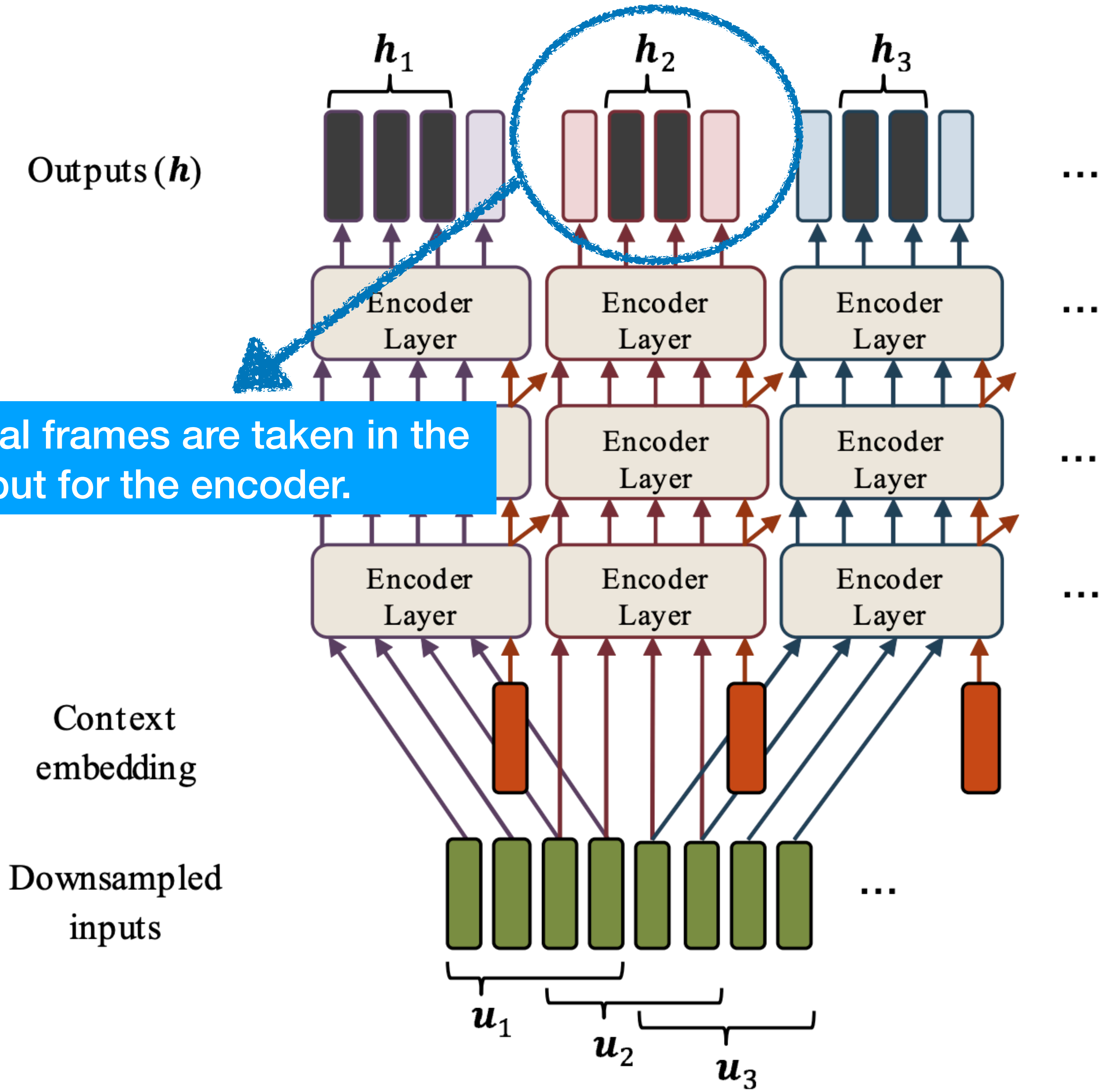
This framework enables a deeper layer to hold longer context information.



Contextual Block Processing



Contextual Block Processing



Only central frames are taken in the output for the encoder.

Contextual Block Processing

- Only encoder processes in blocks. **Decoder is still sequential**, because it is difficult for the decoder to do such block processing.

DQ: Why do you think it is difficult for decoder to process in blocks?

Contextual Block Processing

- Only encoder processes in blocks. **Decoder is still sequential**, because it is difficult for the decoder to do such block processing.

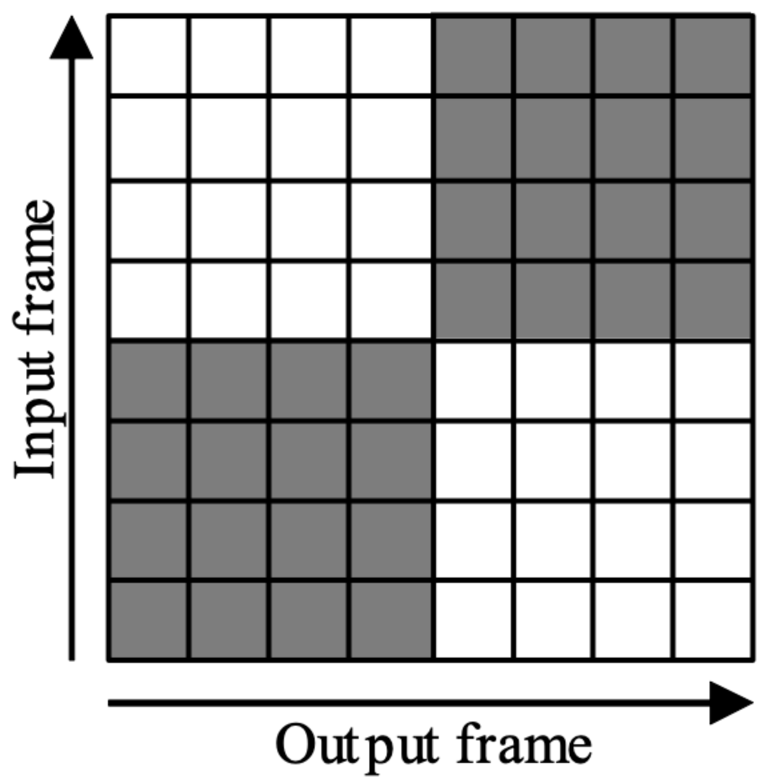
DQ: Why do you think it is difficult for decoder to process in blocks?

- Estimating the optimal alignment between encoder output and decoder is difficult!

Implementation

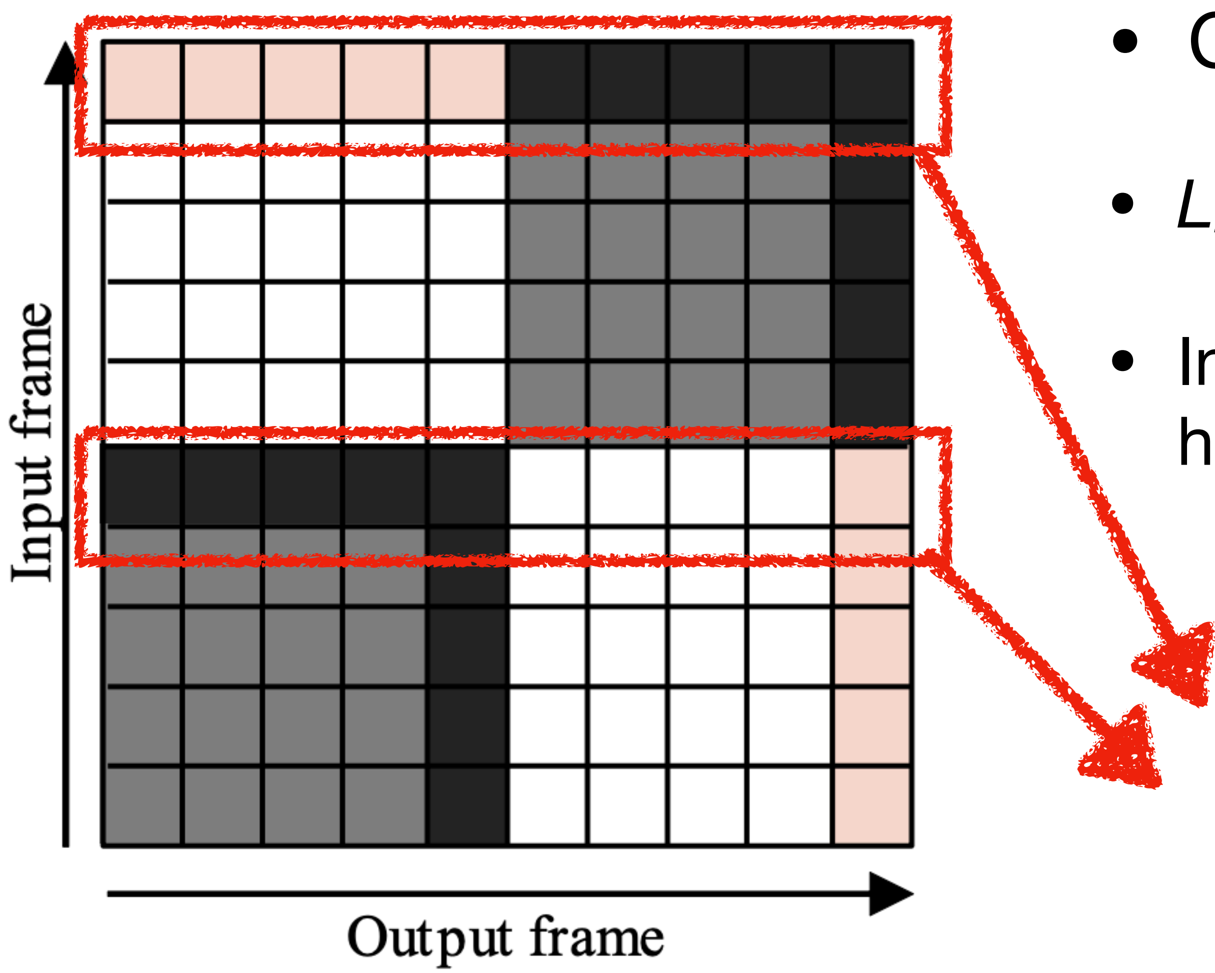
- Similar to implementation in earlier work on block processing.
- Whole utterance is given to each encoder.
- Encoder has mask which is applied on utterance to select frames which it has to process.

Implementation



- Naive block processing.
- $L_{block} = L_{hop} = 4$
- Dark regions pass values

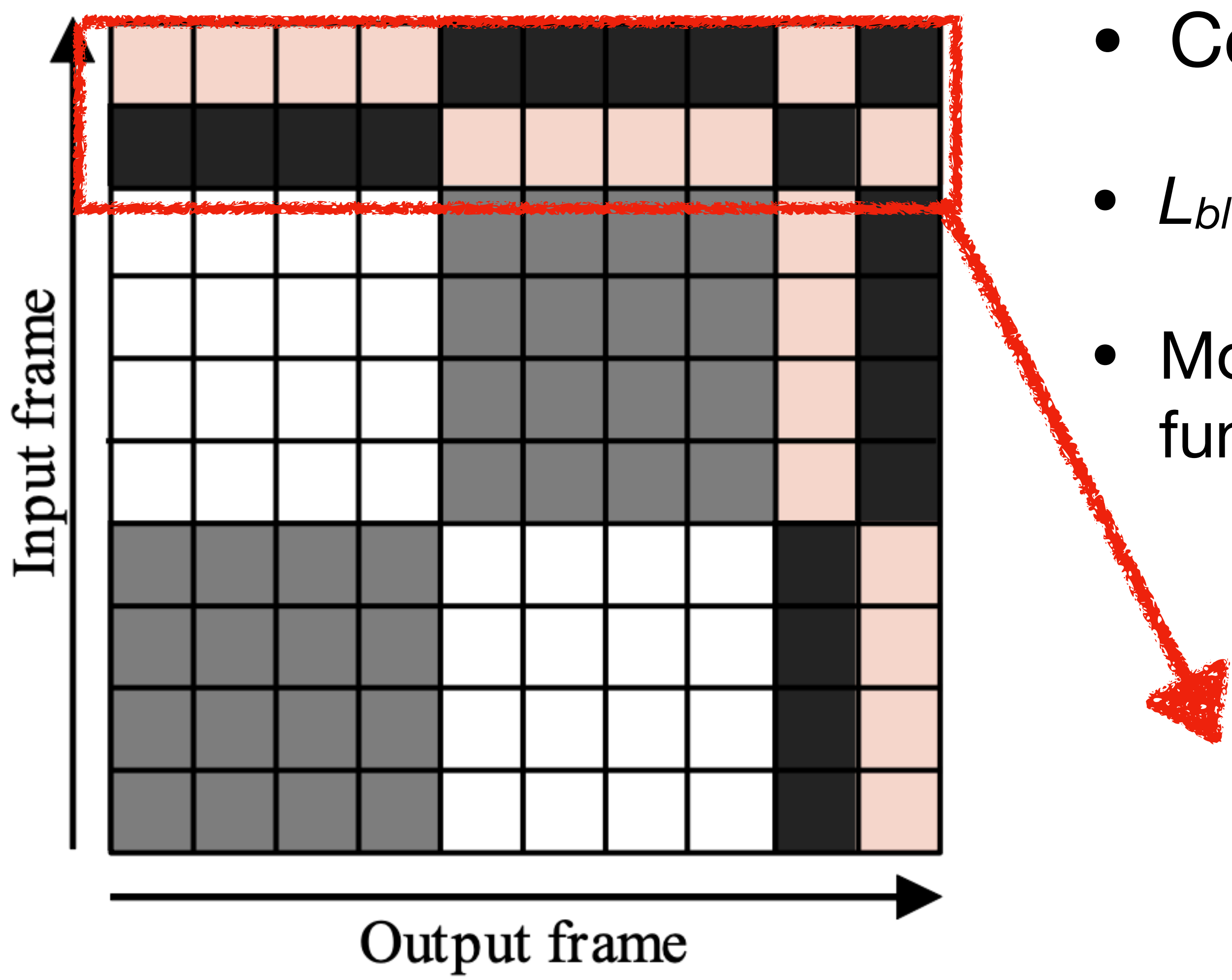
Implementation



- Contextual block processing
- L_{block} = L_{hop} = 4
- Inserting vector after each block is hard

Context vector

Implementation



- Contextual block processing
- $L_{block} = L_{hop} = 4$
- Modified mask with same functionality

Context vectors

Experiments

- **Datasets**: WSJ (English), Librispeech (English), VoxForge (Italian), AISHELL (Mandarin)

- 80-dim **Fbank features** extracted using 25 ms windows with a hop size of 10 ms

- Trained using multitask learning (attention + CTC) using Espnet

WER Results

Table 1. Word error rates (WERs) in the WSJ evaluation task with $L_{\text{block}} = 16$ and $L_{\text{hop}} = 8$.

	eval92
Batch encoding	
biLSTM [13]	6.7
Transformer	**5.0**

Hybrid HMM-DNN (Kaldi) gets 4.36% WER with a trigram LM and 2.36% with a big dictionary and 4-gram LM rescoring.

We expect online encoders to do worse than batch encoders.

WER Results

Table 1. Word error rates (WERs) in the WSJ evaluation task with $L_{\text{block}} = 16$ and $L_{\text{hop}} = 8$.

	eval92
Batch encoding	
biLSTM [13]	6.7
Transformer	**5.0**
Online encoding	
LSTM	8.4

Unidirectional LSTM -> naturally online model

WER Results

Table 1. Word error rates (WERs) in the WSJ evaluation task with $L_{\mathrm{block}} = 16$ and $L_{\mathrm{hop}} = 8$.

	eval92
Batch encoding	
biLSTM [13]	6.7
Transformer	**5.0**
Online encoding	
LSTM	8.4
Block Transformer	7.5

Naive block processing, without context vectors

WER Results

Table 1. Word error rates (WERs) in the WSJ evaluation task with $L_{\text{block}} = 16$ and $L_{\text{hop}} = 8$.

	eval92
Batch encoding	
biLSTM [13]	6.7
Transformer	**5.0**
Online encoding	
LSTM	8.4
Block Transformer	7.5
Contextual Block Transformer	
—*PE*	6.0
—*Avg. input*	6.3
—*Max input*	10.9

WER Results

Table 1. Word error rates (WERs) in the WSJ evaluation task with $L_{\text{block}} = 16$ and $L_{\text{hop}} = 8$.

	eval92
Batch encoding	
biLSTM [13]	6.7
Transformer	**5.0**
Online encoding	
LSTM	8.4
Block Transformer	7.5
Contextual Block Transformer	
—*PE*	6.0
—*Avg. input*	6.3
—*Max input*	10.9
—*PE + Avg. input*	**5.7**
—*PE + Max input*	7.9

Only this initialization used for further experiments

WER Results

	LibriSpeech (WER) clean	LibriSpeech (WER) other	VoxForge (CER)	AISHELL (CER)
Batch encoding				
biLSTM [13]	**4.2**	13.1	10.5	9.2
Transformer	4.5	**11.2**	**9.3**	**6.4**
Online encoding				
LSTM	5.3	16.1	14.6	11.8
Block	4.8	13.2	11.5	7.8
Contextual Block				
—*PE + Avg. input*	**4.6**	**13.1**	**10.3**	**7.6**

WER Results

	LibriSpeech (WER)		VoxForge (CER)	AISHELL (CER)
	clean	other		
Batch encoding				
biLSTM [13]	**4.2**	13.1	10.5	9.2
Transformer	4.5	**11.2**	**9.3**	**6.4**
Online encoding				
LSTM	5.3	16.1	14.6	11.8
Block	4.8	13.2	11.5	7.8
Contextual Block				
—*PE + Avg. input*	**4.6**	**13.1**	**10.3**	**7.6**

Contextual inheritance does not seem to help a lot.

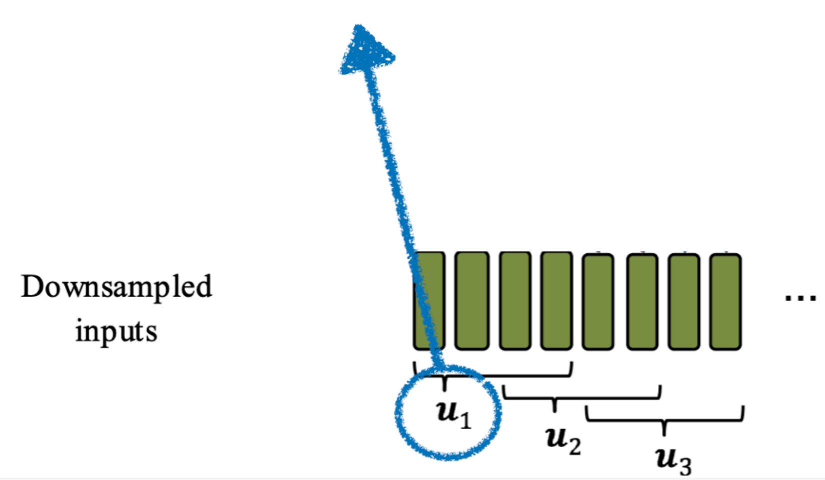
Effect of block size

We saw earlier!

Contextual Block Processing

- In experiments, $L_{block} = 2\ L_{hop}$
- Higher L_{block} means more context but lower parallelization

$$\mathbf{u}_b = (u_{(b-1)\cdot L_{\text{hop}}+1}, \ldots, u_{(b-1)\cdot L_{\text{hop}}+L_{\text{block}}})$$



Effect of block size

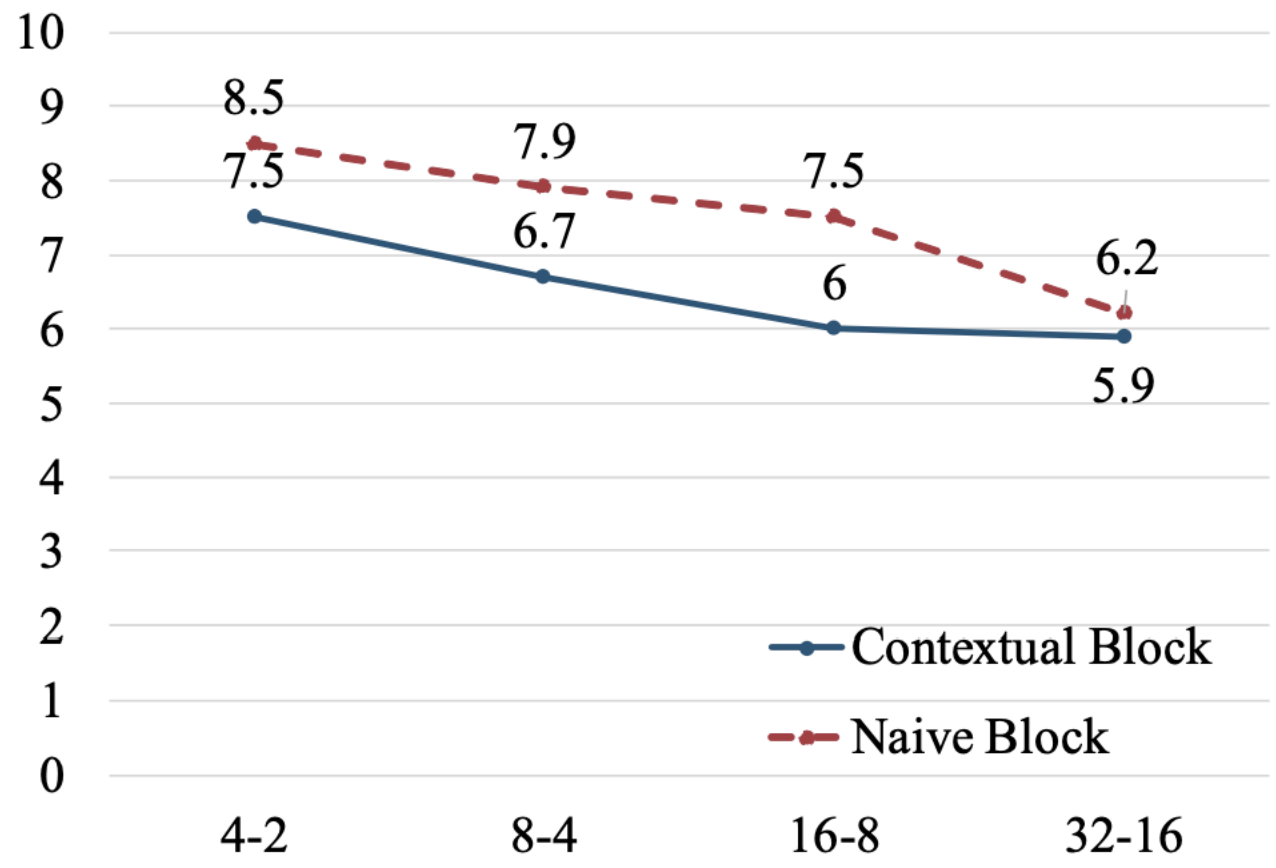


Fig. 4. The WERs in the WSJ evaluation task for various block sizes ($L_{\text{block}} - L_{\text{hop}}$).

Effect of block size

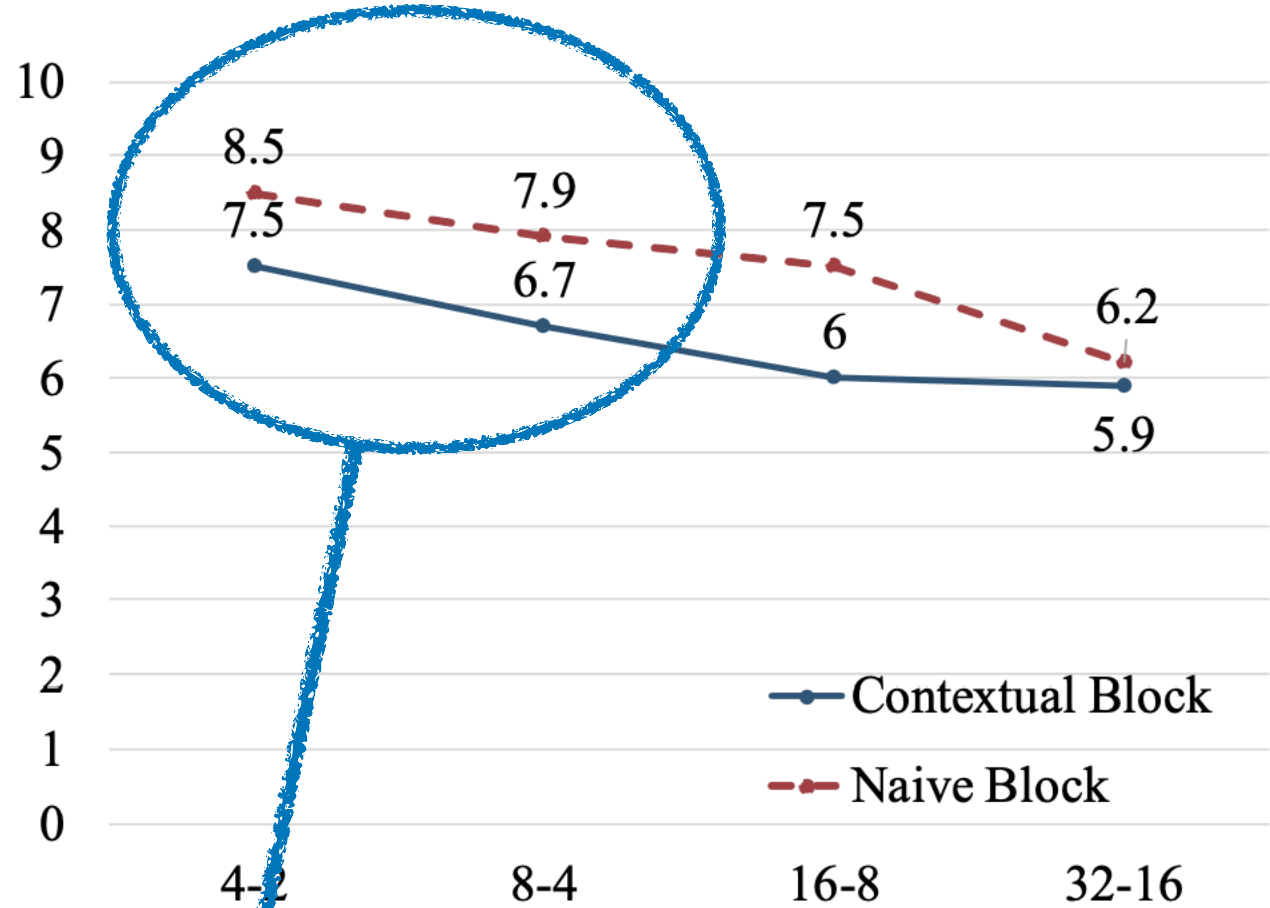


Fig. 4. The WERs in the WSJ evaluation task for various block sizes ($L_{\text{block}} - L_{\text{hop}}$).

For smaller block sizes, context vectors are VERY useful

Effect of block size

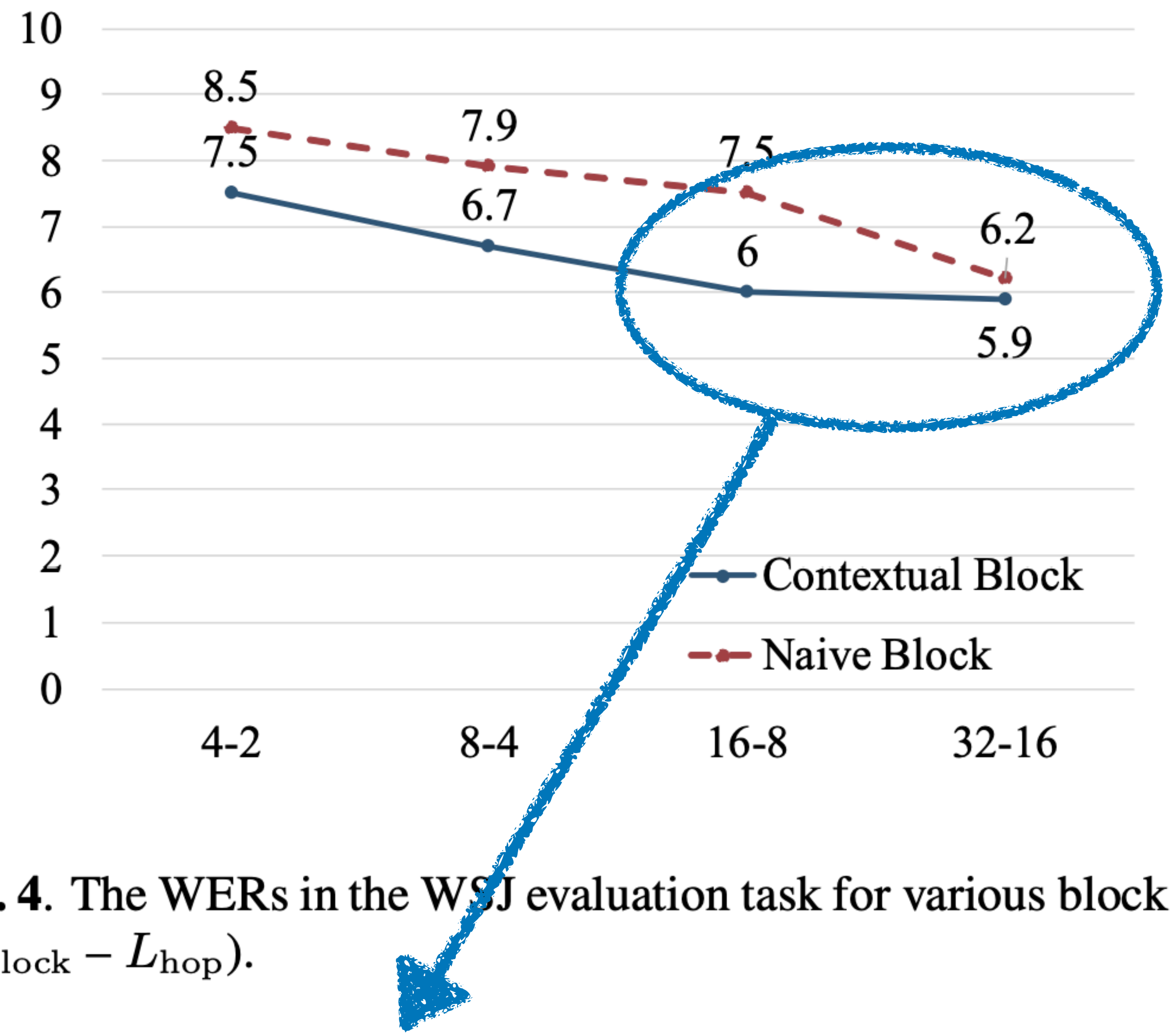


Fig. 4. The WERs in the WSJ evaluation task for various block sizes ($L_{\mathrm{block}} - L_{\mathrm{hop}}$).

Block size 16 is sufficient for contextual block processing

Effect of block size

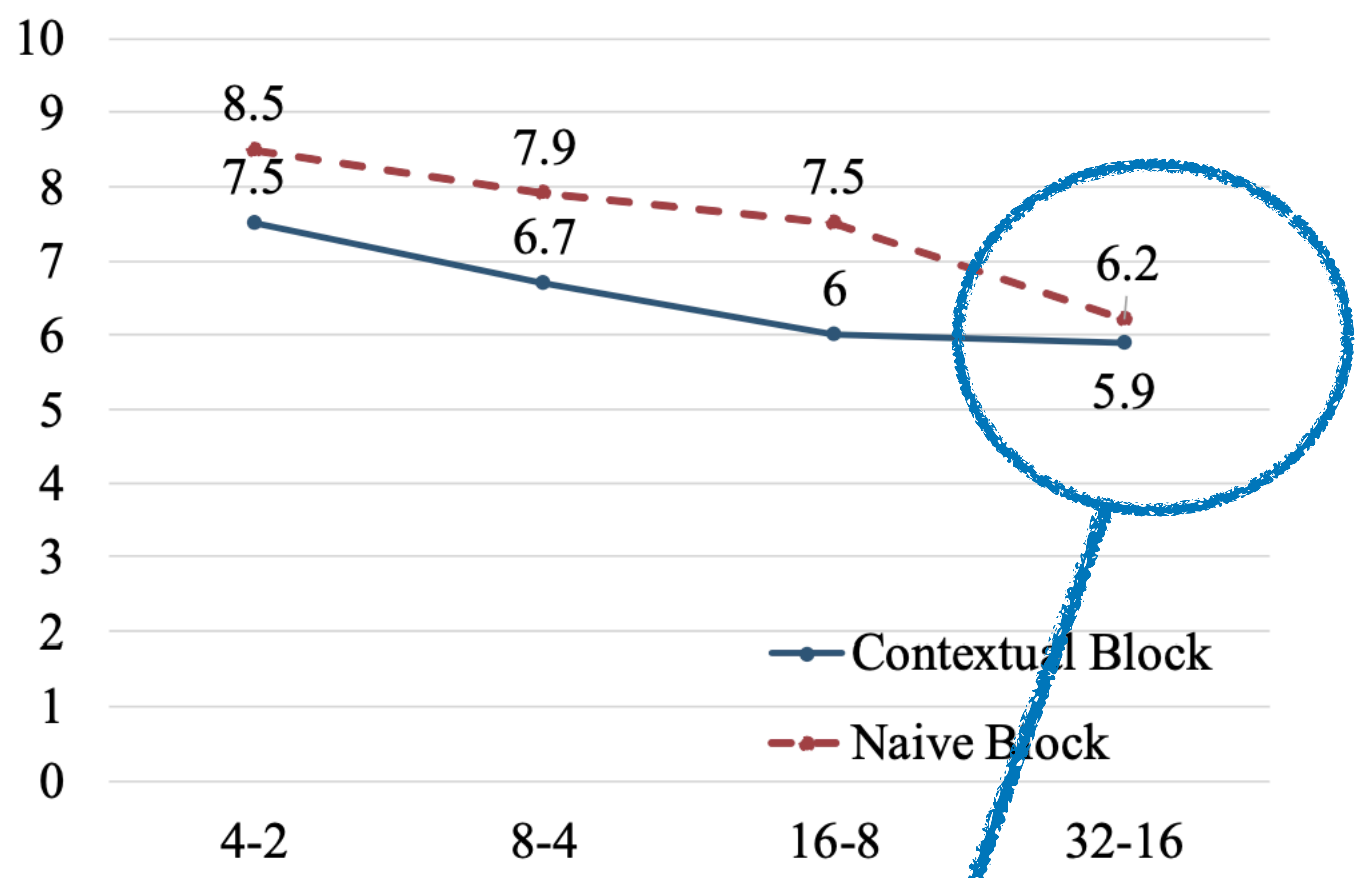


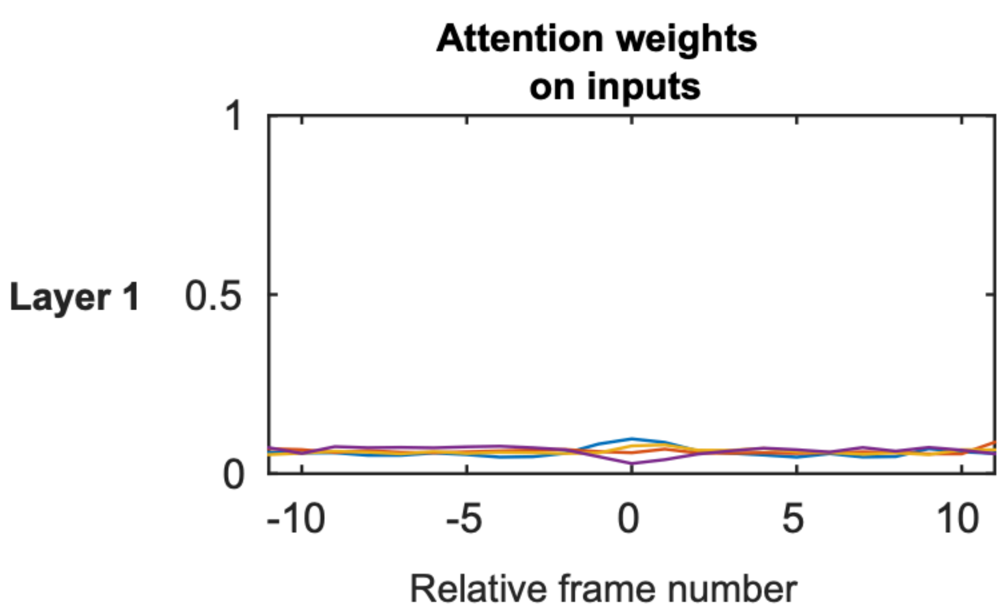
Fig. 4. The WERs in the WSJ evaluation task for various block sizes ($L_{\text{block}} - L_{\text{hop}}$).

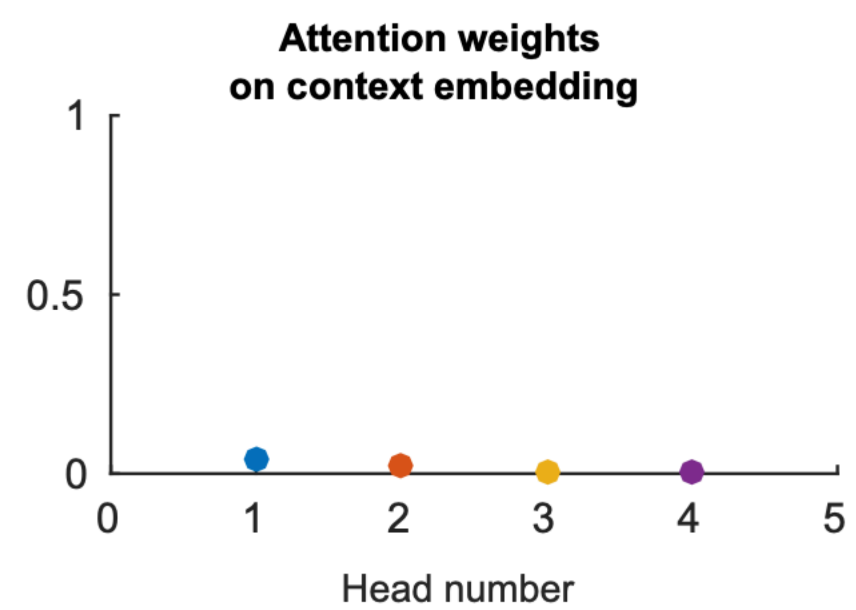
Block size of 32 is sufficient to acquire certain context information for naive block processing

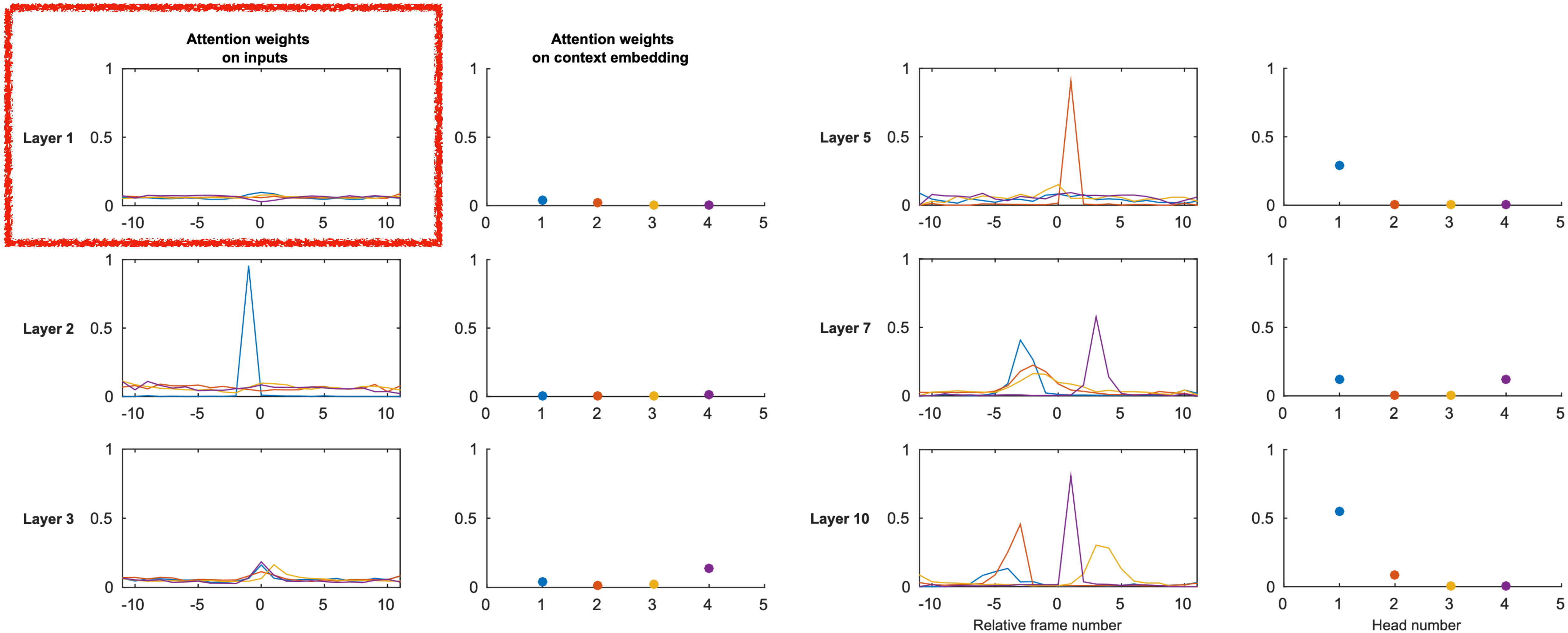
Analysis of attention weights

Analysis performed on a randomly sampled evaluation utterance

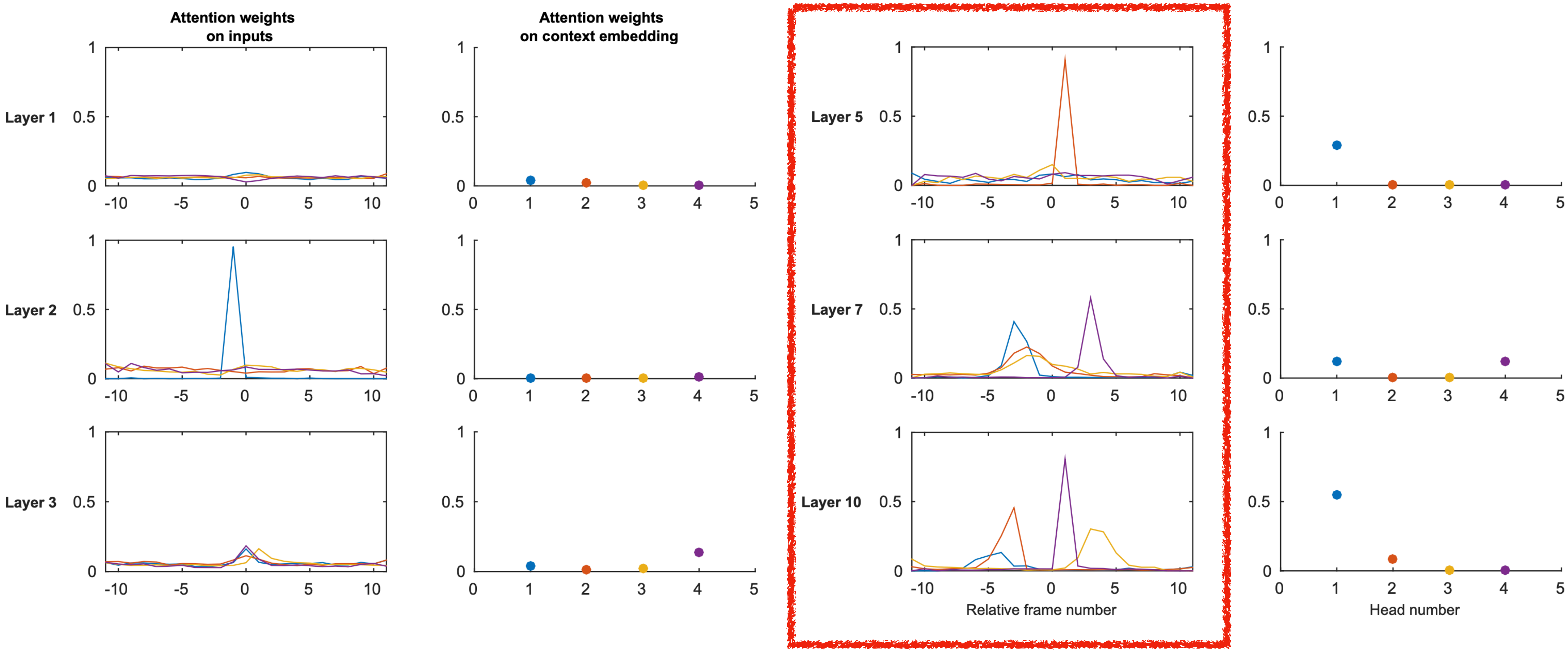
Each color corresponds to an attention head



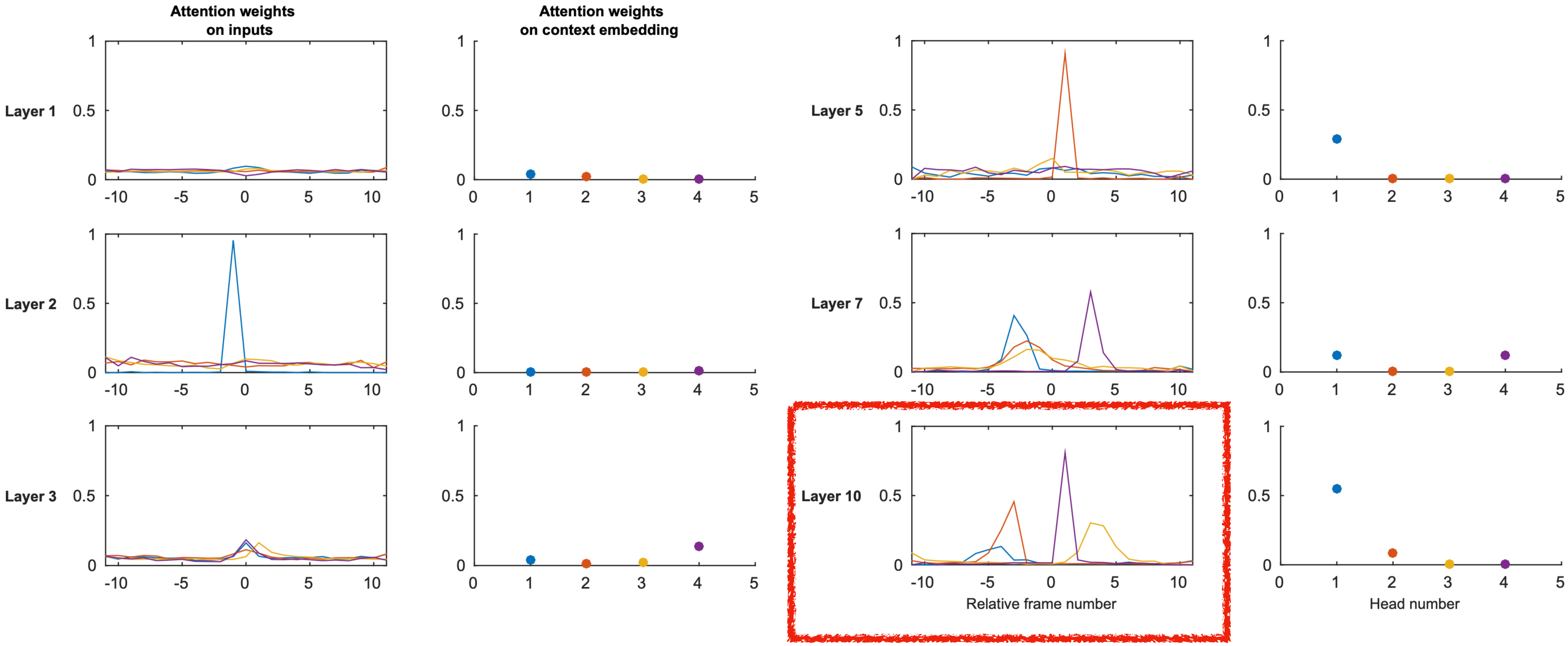




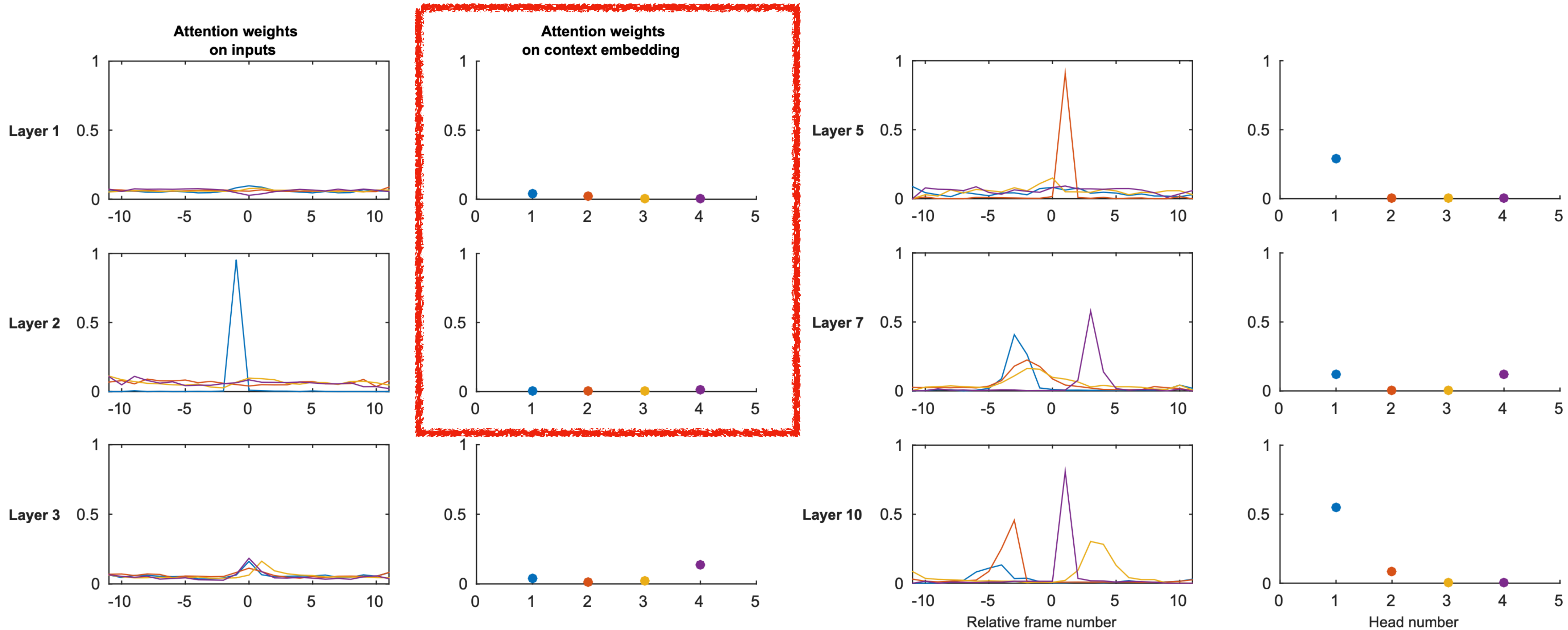
At the bottom, attention tends to the input sequence evenly



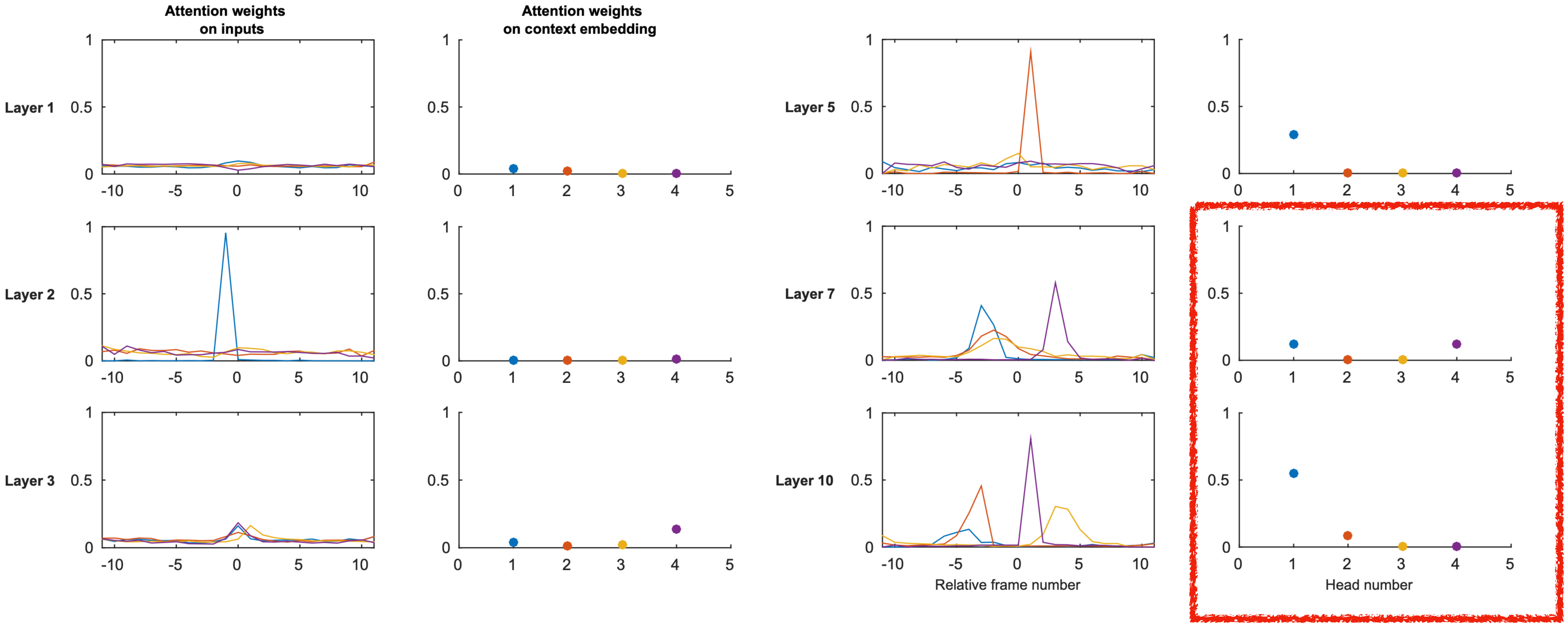
In deeper layers, attention weights start to develop peaks



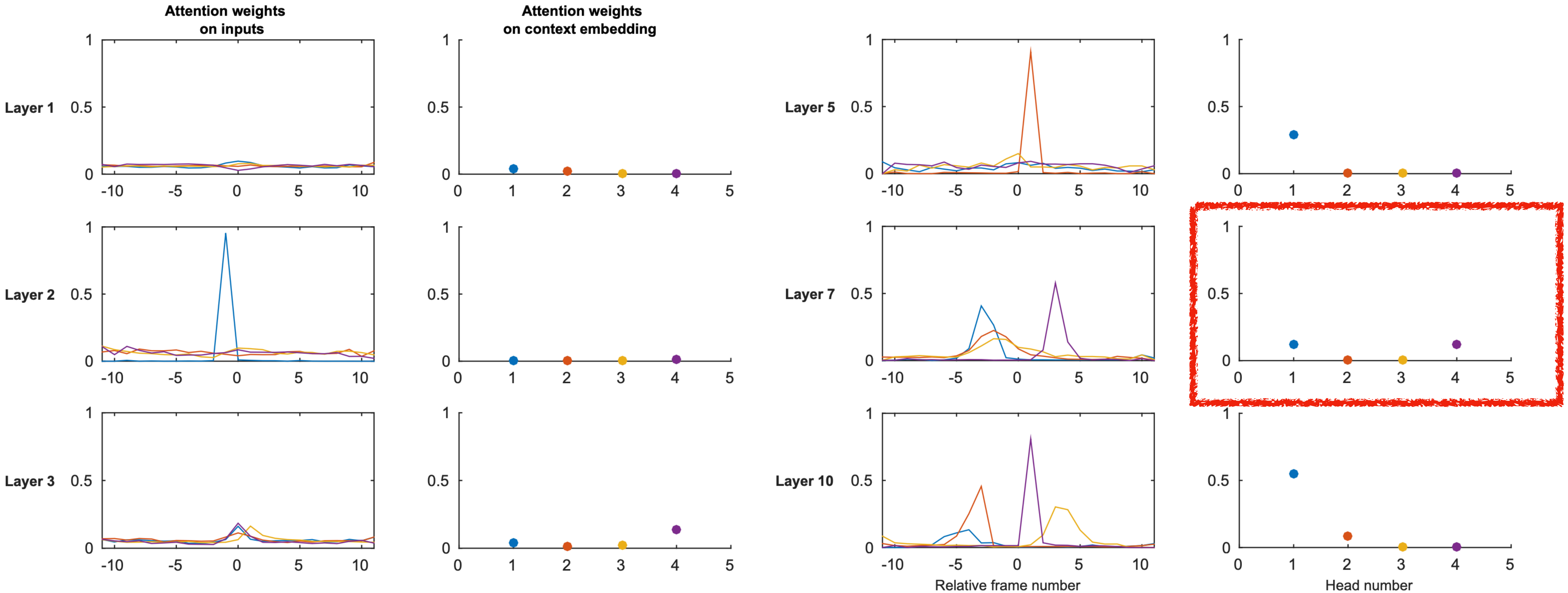
Different heads attend to different parts of the input



Context vector is not very useful in lower layers



Deeper layers seem to rely on context information more



But not all attention heads use the context information

Key takeaways

- Transformers are good, but not for long input sequences.
- Process in blocks, but pass context with a vector.
- Faster + online ✓

“We can’t transform, but we’re not helpless.”

–Optimus Prime (Transformers)