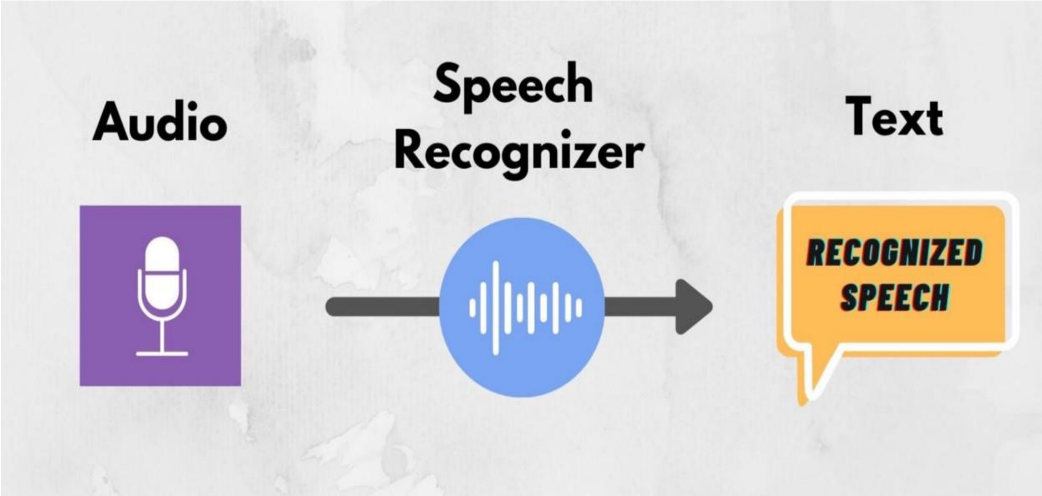
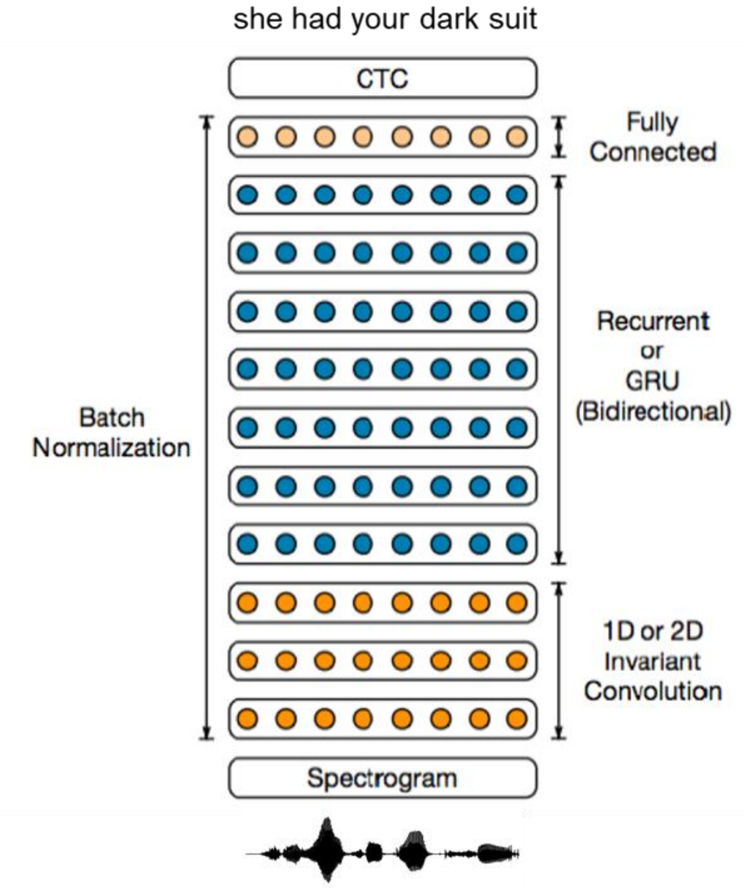
Wav2vec

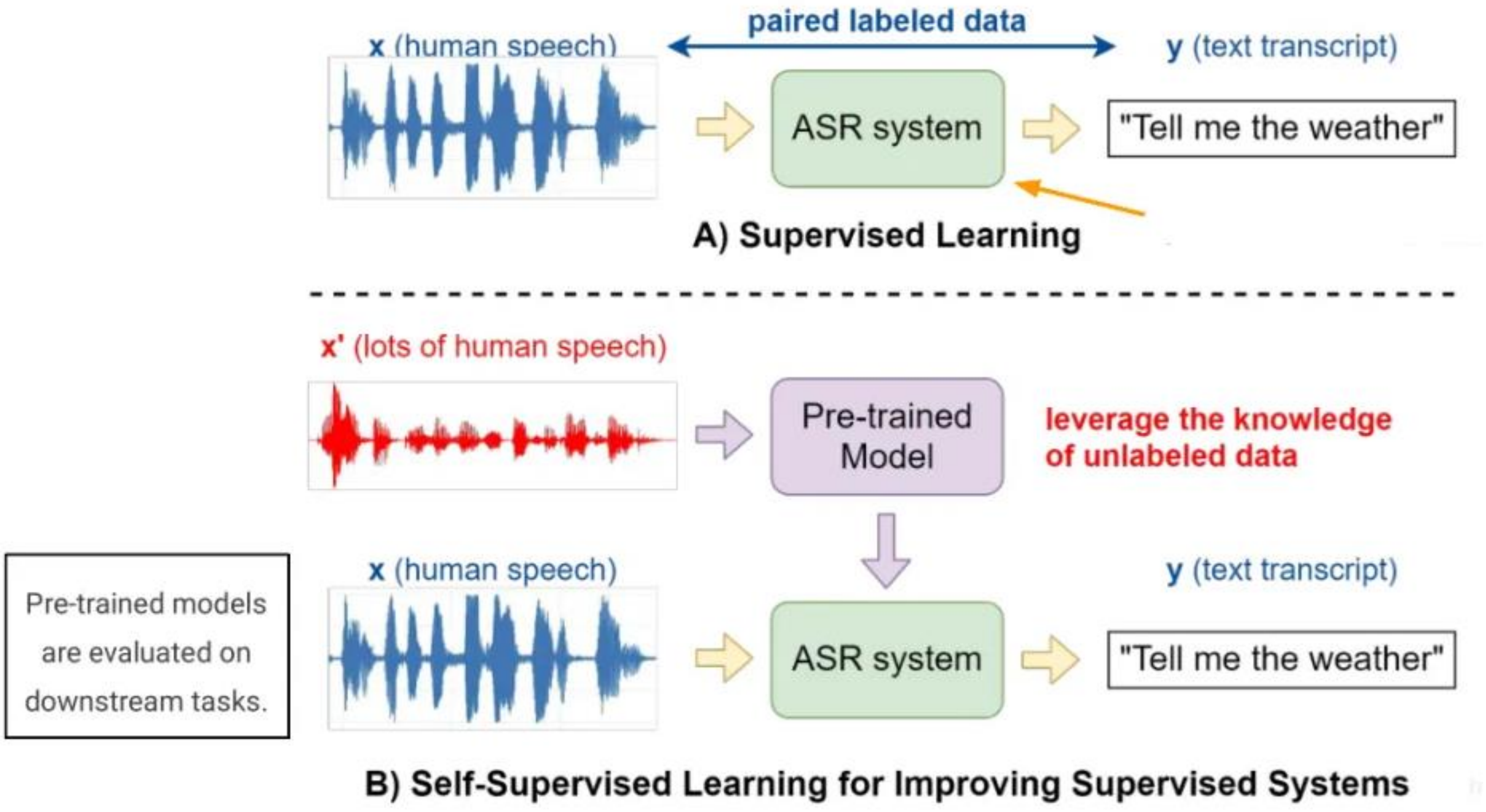
Automatic Speech Recognition (ASR)



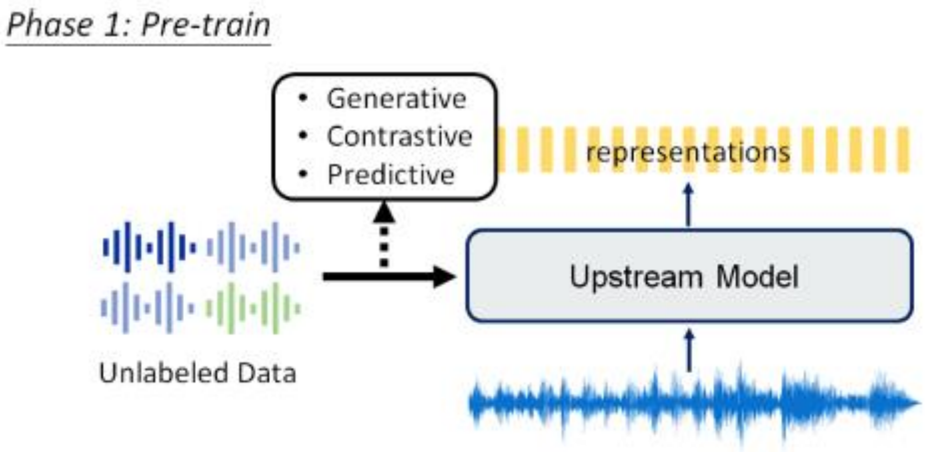


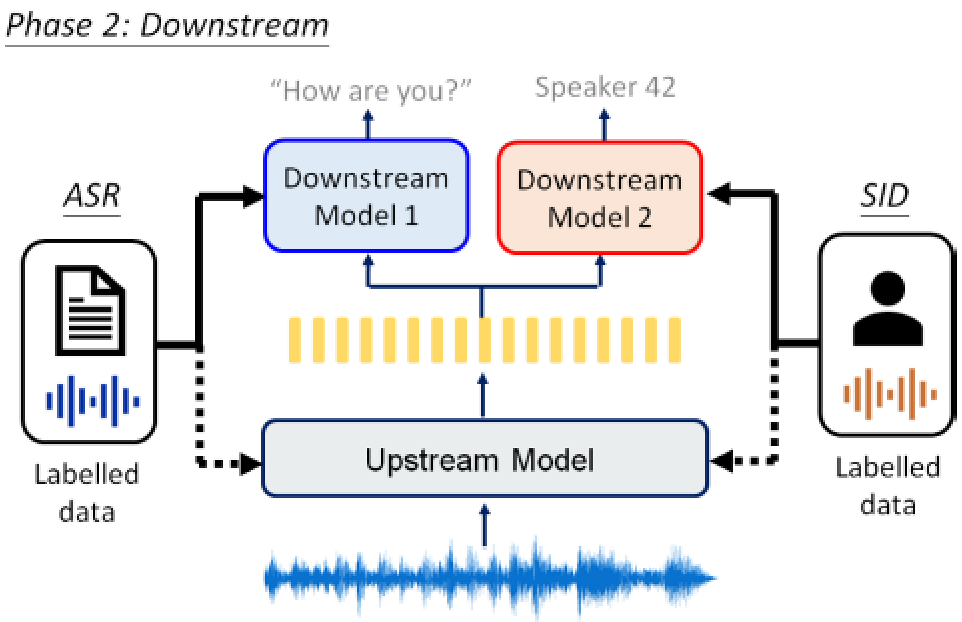
End-to-end ASR

Automatic Speech Recognition (ASR)



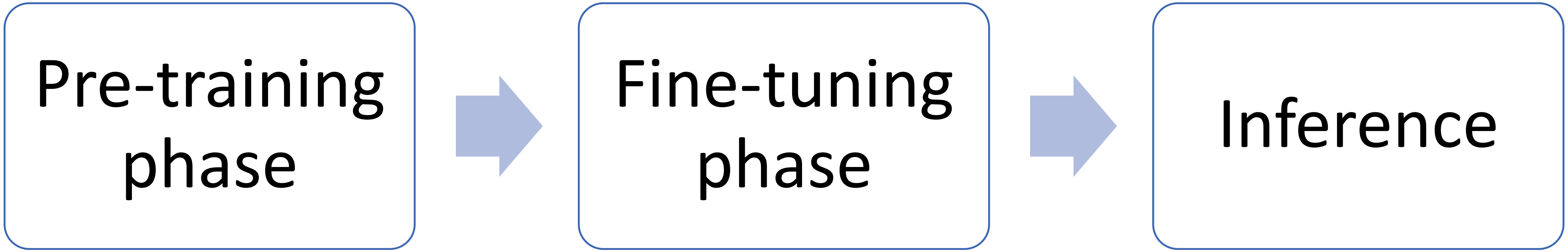
Foundation models in ASR



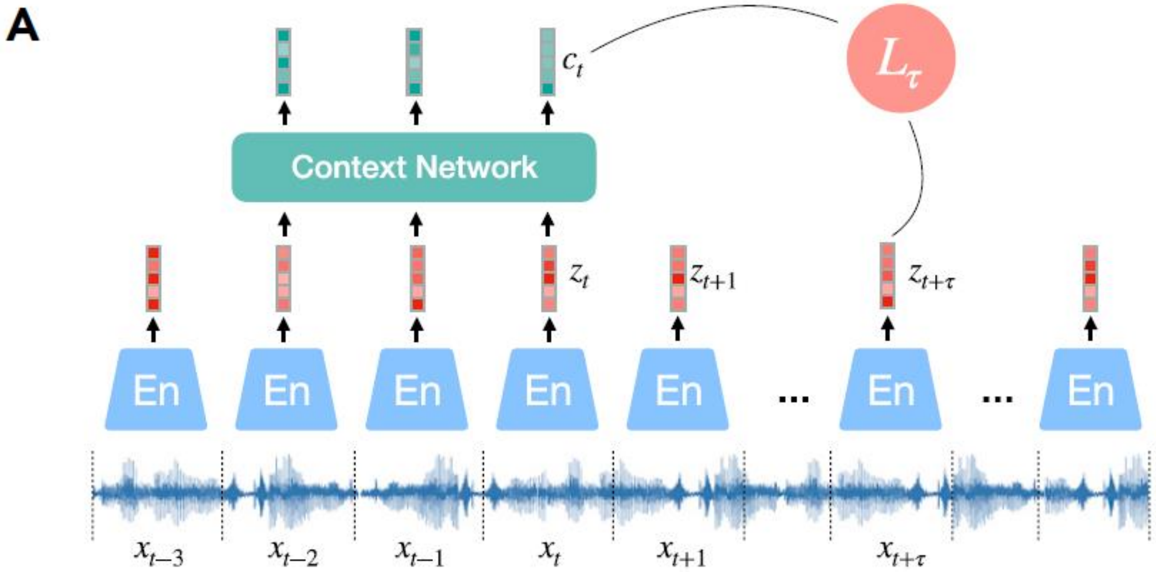


- In the first stage, we use SSL to pre-train a *representation model*, also called an *upstream model* or a *foundation model*.
- In the second stage, downstream tasks use either the learned representation from the frozen model, or fine-tune the entire pre-trained model in a supervised phase. Automatic speech recognition (ASR) and speaker identification (SID) are examples of downstream applications.

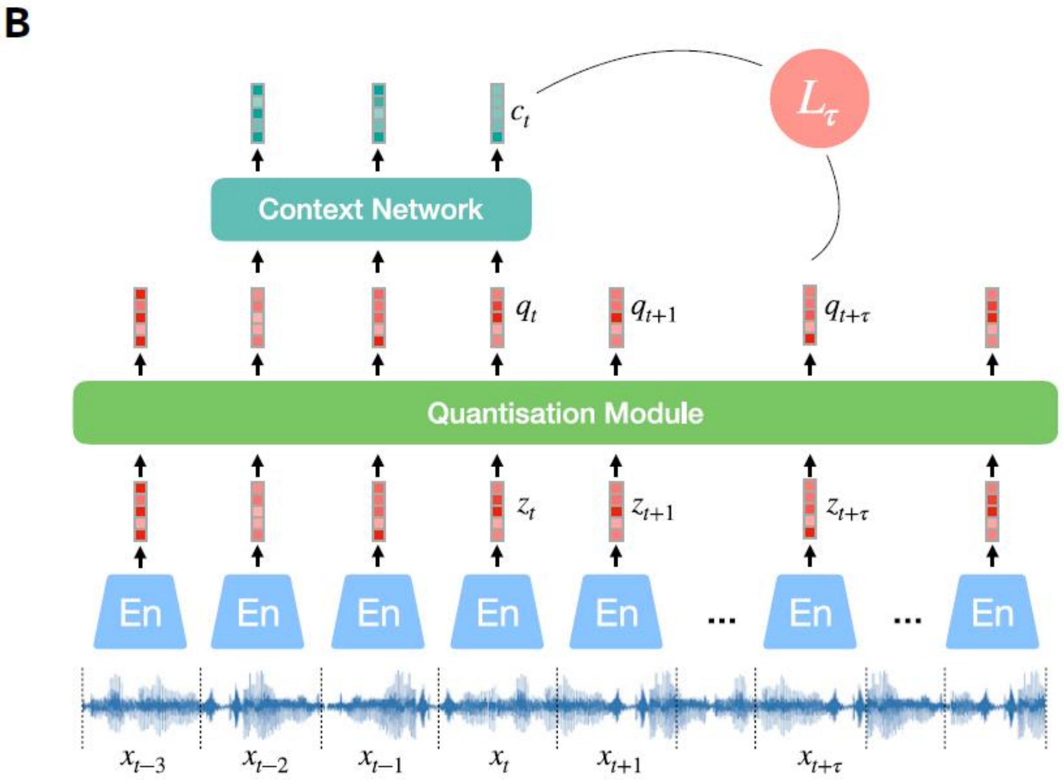
Foundation models in ASR



Foundation models in ASR



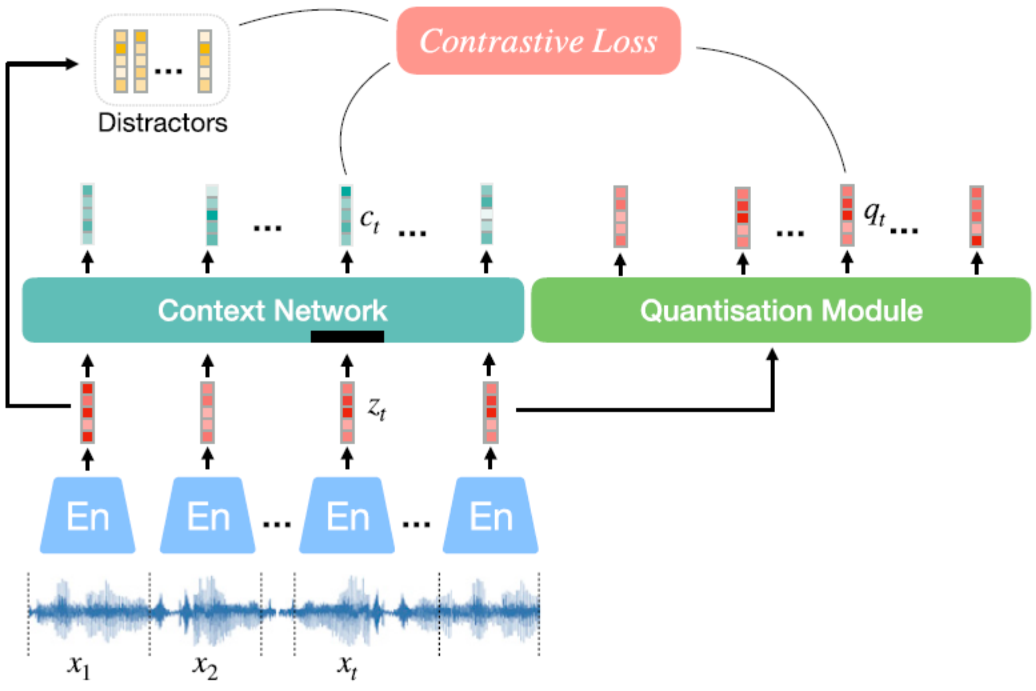
Wav2vec



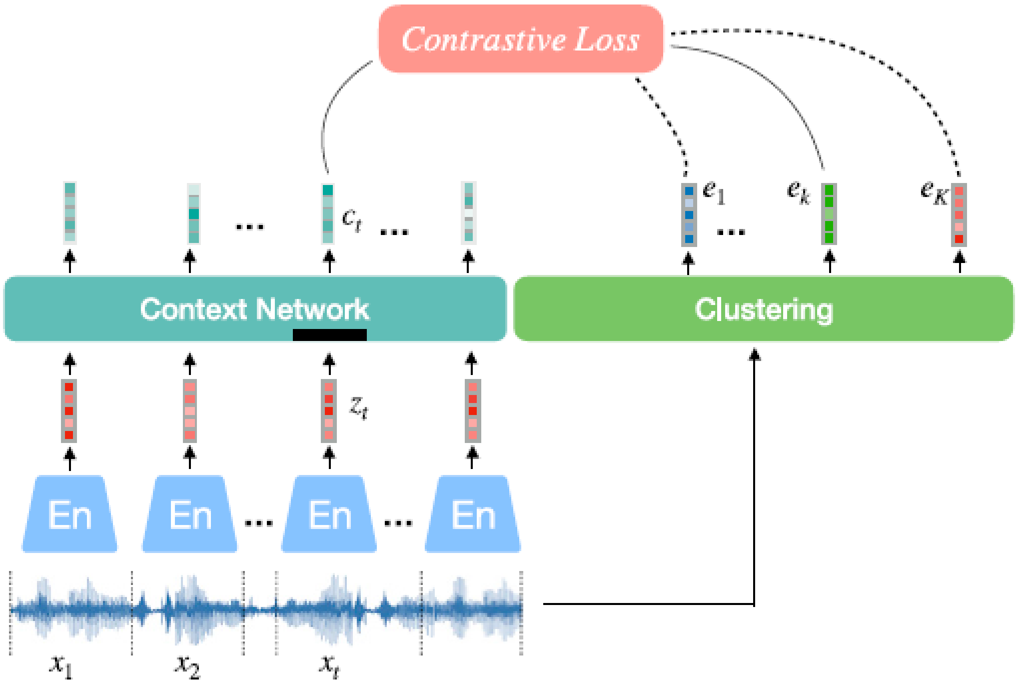
VQ-wav2vec

Foundation models in ASR

C

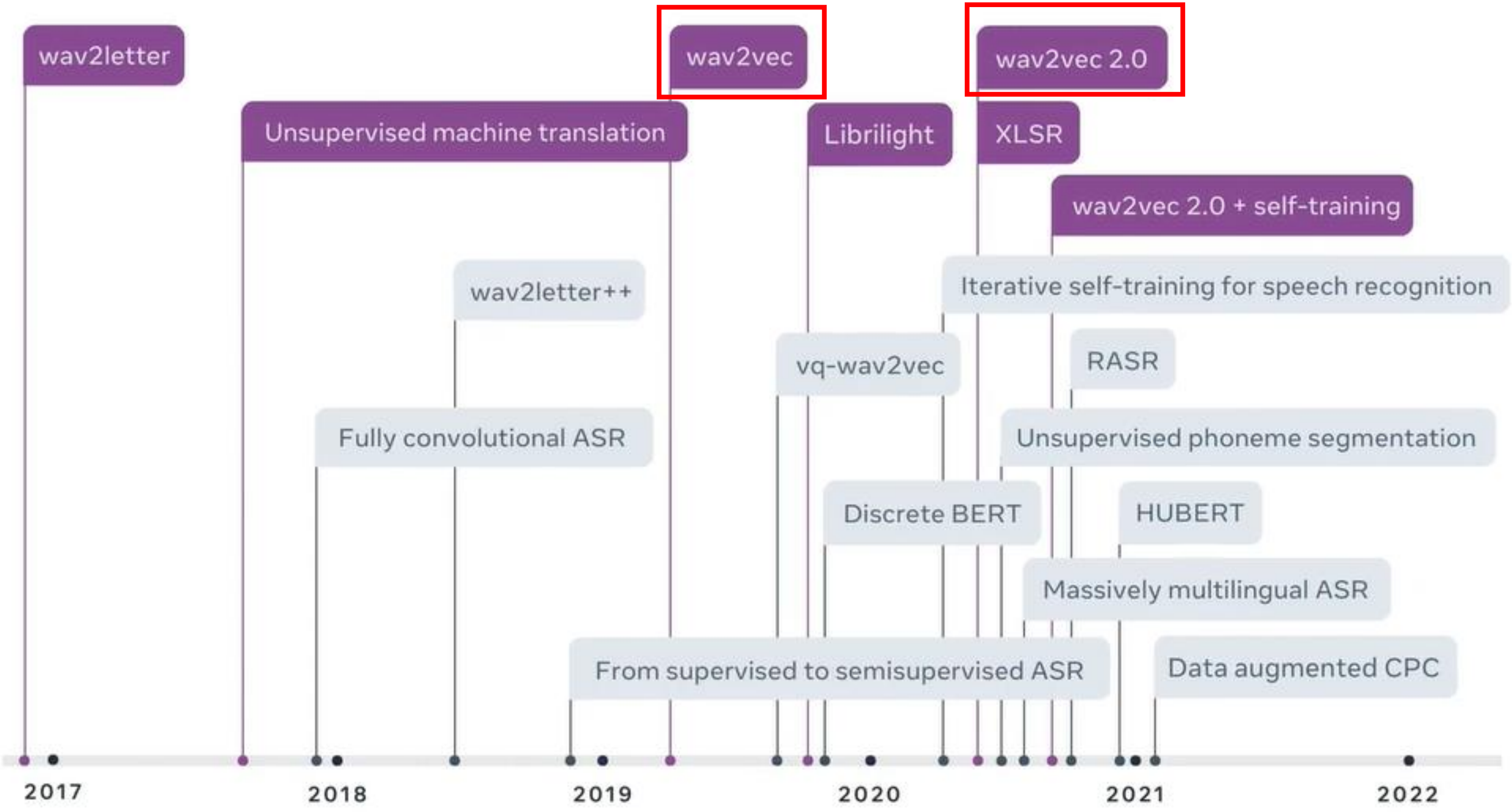


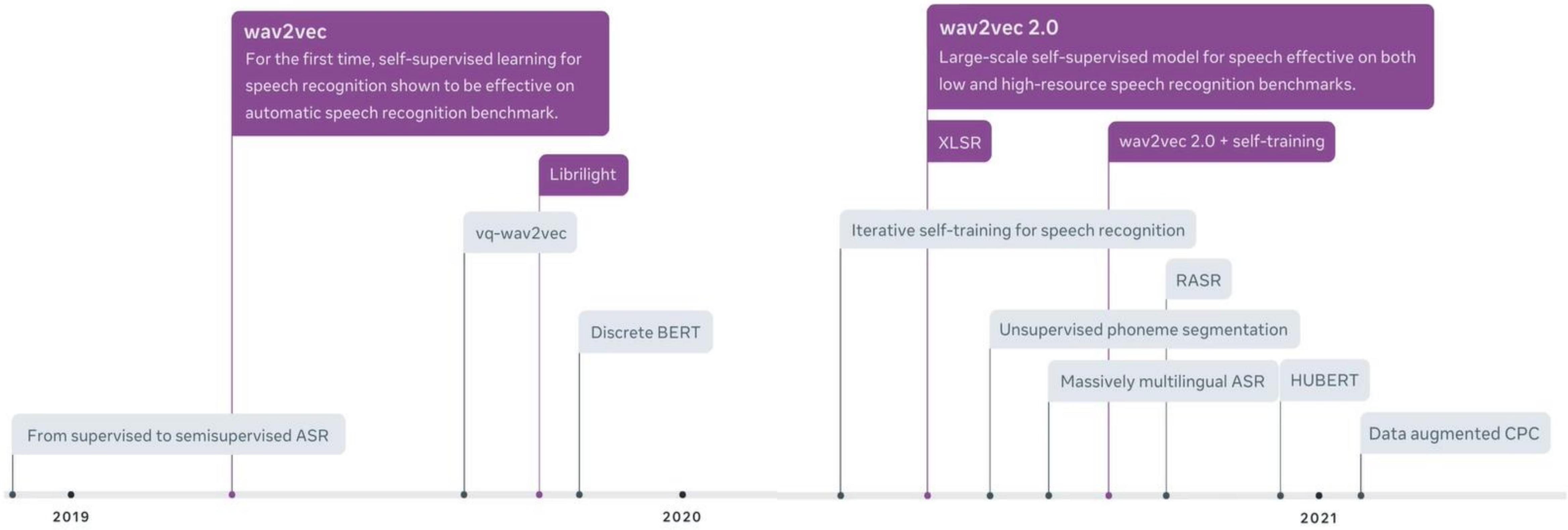
D



Wav2vec 2.0

HuBERT


wav2letter
wav2vec
wav2vec 2.0
Unsupervised machine translation
Librilight
XLSR
wav2vec 2.0 + self-training
wav2letter++
Iterative self-training for speech recognition
vq-wav2vec
RASR
Fully convolutional ASR
Unsupervised phoneme segmentation
Discrete BERT
HUBERT
Massively multilingual ASR
From supervised to semisupervised ASR
Data augmented CPC
2017
2018
2019
2020
2021
2022



wav2vec
For the first time, self-supervised learning for speech recognition shown to be effective on automatic speech recognition benchmark.
Librilight
vq-wav2vec
Discrete BERT
From supervised to semisupervised ASR
wav2vec 2.0
Large-scale self-supervised model for speech effective on both low and high-resource speech recognition benchmarks.
XLSR
wav2vec 2.0 + self-training
Iterative self-training for speech recognition
RASR
Unsupervised phoneme segmentation
Massively multilingual ASR
HUBERT
Data augmented CPC
2019
2020
2021

WAV2VEC: UNSUPERVISED PRE-TRAINING FOR SPEECH RECOGNITION

Steffen Schneider, Alexei Baevski, Ronan Collobert, Michael Auli
Facebook AI Research

ABSTRACT

We explore unsupervised pre-training for speech recognition by learning representations of raw audio. wav2vec is trained on large amounts of unlabeled audio data and the resulting representations are then used to improve acoustic model training. We pre-train a simple multi-layer convolutional neural network optimized via a noise contrastive binary classification task. Our experiments on WSJ reduce WER of a strong character-based log-mel filterbank baseline by up to 36 % when only a few hours of transcribed data is available. Our approach achieves 2.43 % WER on the nov92 test set. This outperforms Deep Speech 2, the best reported character-based system in the literature while using two orders of magnitude less labeled training data.[1]

Wav2vec

- Apply unsupervised pre-training to improve supervised speech recognition.
- Exploiting unlabeled audio data which is much easier to collect than labeled data.
- Wav2vec is trying to predict the future of an audio sequence.
    - First, pre-train a large network on unlabeled data to learn useful contextual representations of the text/audio sequence.
    - Second, use these pre-trained representations for a variety of tasks for which not enough data is available.
- This enables exploiting unlabeled audio data which is much easier to collect than labeled data.
- Wav2vec, is a convolutional neural network that takes raw audio as input and computes a general representation that can be input to a speech recognition system.
- The objective is a contrastive loss that requires distinguishing a true future audio sample from negatives.
- Different to previous work (van den Oord et al., 2018), we move beyond frame-wise phoneme classification and apply the learned representations to improve strong supervised ASR systems.
- Wav2vec relies on a fully convolutional architecture which can be easily parallelized over time on modern hardware compared to recurrent models used in previous work
- Wav2vec adjusts the CPC structure to a fully convolutional architecture, enabling easy parallelization over time on hardware. One CNN encodes the raw waveform into audio representations for each time step, and the other captures global context information into a context vector

Wav2vec

- At the core of wav2vec are two distinct networks:
  - the encoder network
  - the context network.
- Both are convolutional neural networks, albeit with different settings.
- Our model takes raw audio signal as input and then applies two networks.
- The encoder network embeds the audio signal in a latent space and
- the context network combines multiple time-steps of the encoder to obtain contextualized representations.
- **The encoder network** reduces the dimensionality of the speech data, by encoding 30 milliseconds of audio into a 512-dimensional feature vector $\mathbf{z}_t$ at each timestep t, every 10 ms.
- **The context network** takes as input the encoder output features, encoding 210 ms of raw audio into another 512-dimensional feature vector $\mathbf{c}_t$.
- The objective is to aggregate information over a longer timeframe to model higher-order information. This network outputs *contextual representations* c_t that are used to predict future audio samples.

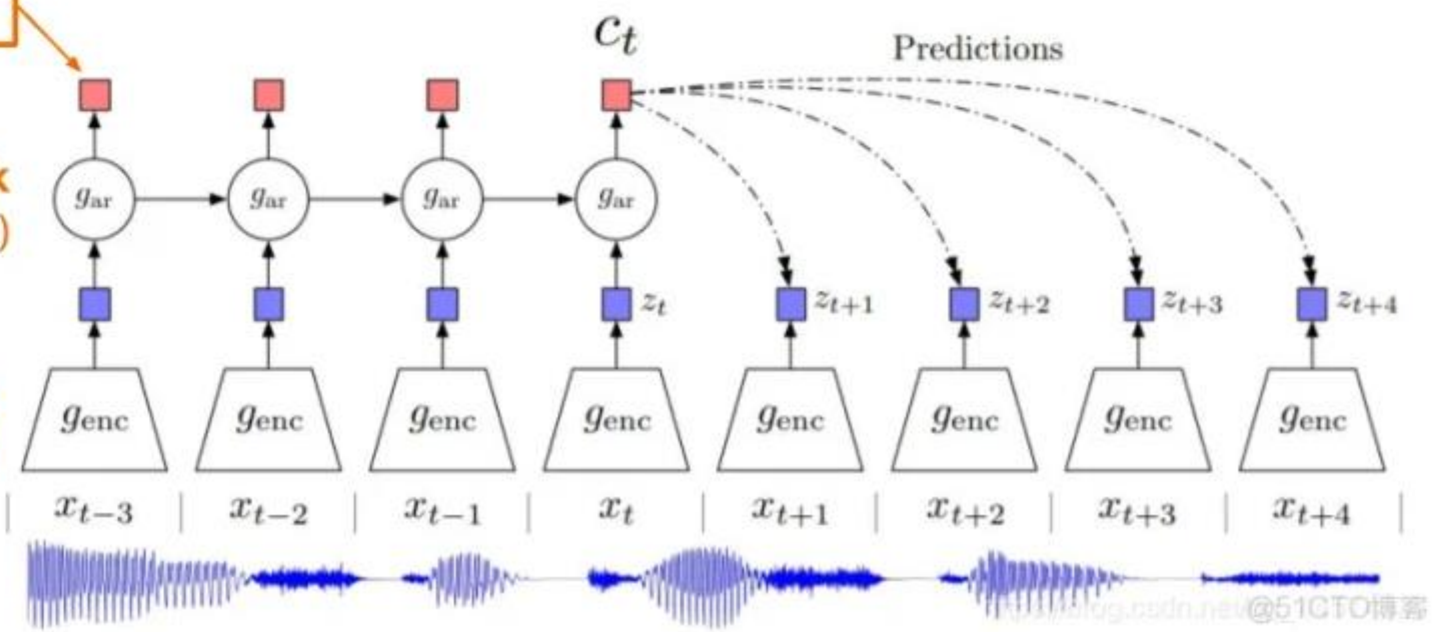
CPC

Intuition:
Pulls temporally nearby representations closer and pushes temporally distant ones further.

Learned representations

Context Network
1-layer Gated Recurrent Units (GRU)

Encoder Network
5-layer convolutional neural network (CNN)



Important exps: Phone / Speaker Classification

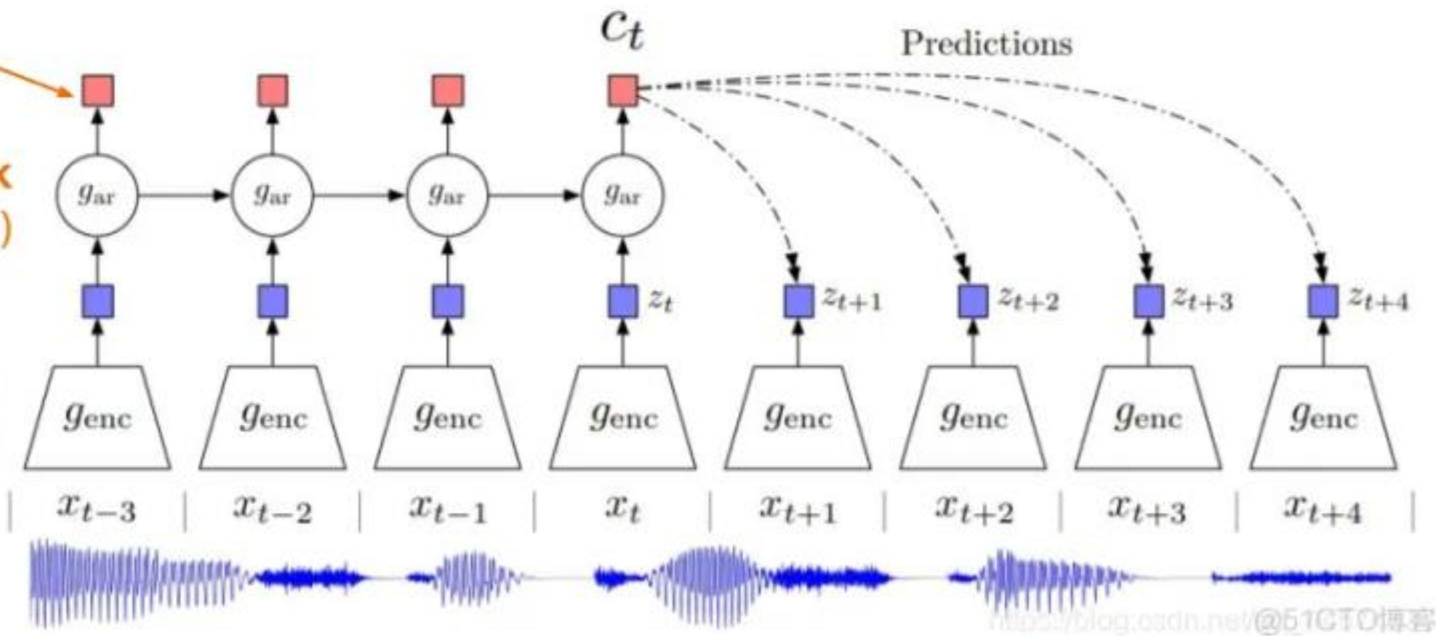
Wav2vec

Contribution:
self-supervised pre-training is shown to improve supervised **ASR**

Used as input for ASR models, replace acoustic features

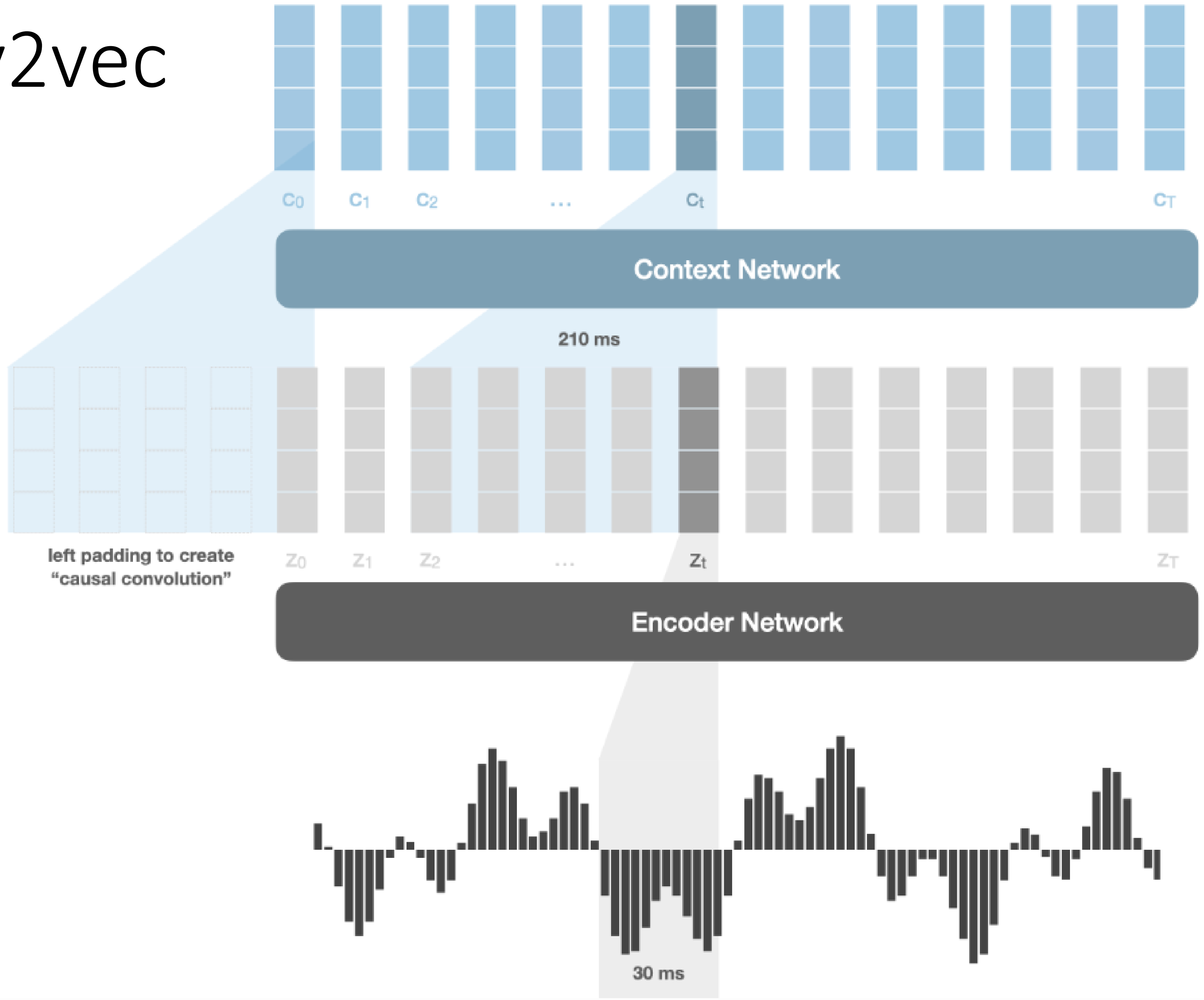
Context Network
9-layer convolutional neural network (CNN)

Encoder Network
5-layer convolutional neural network (CNN)



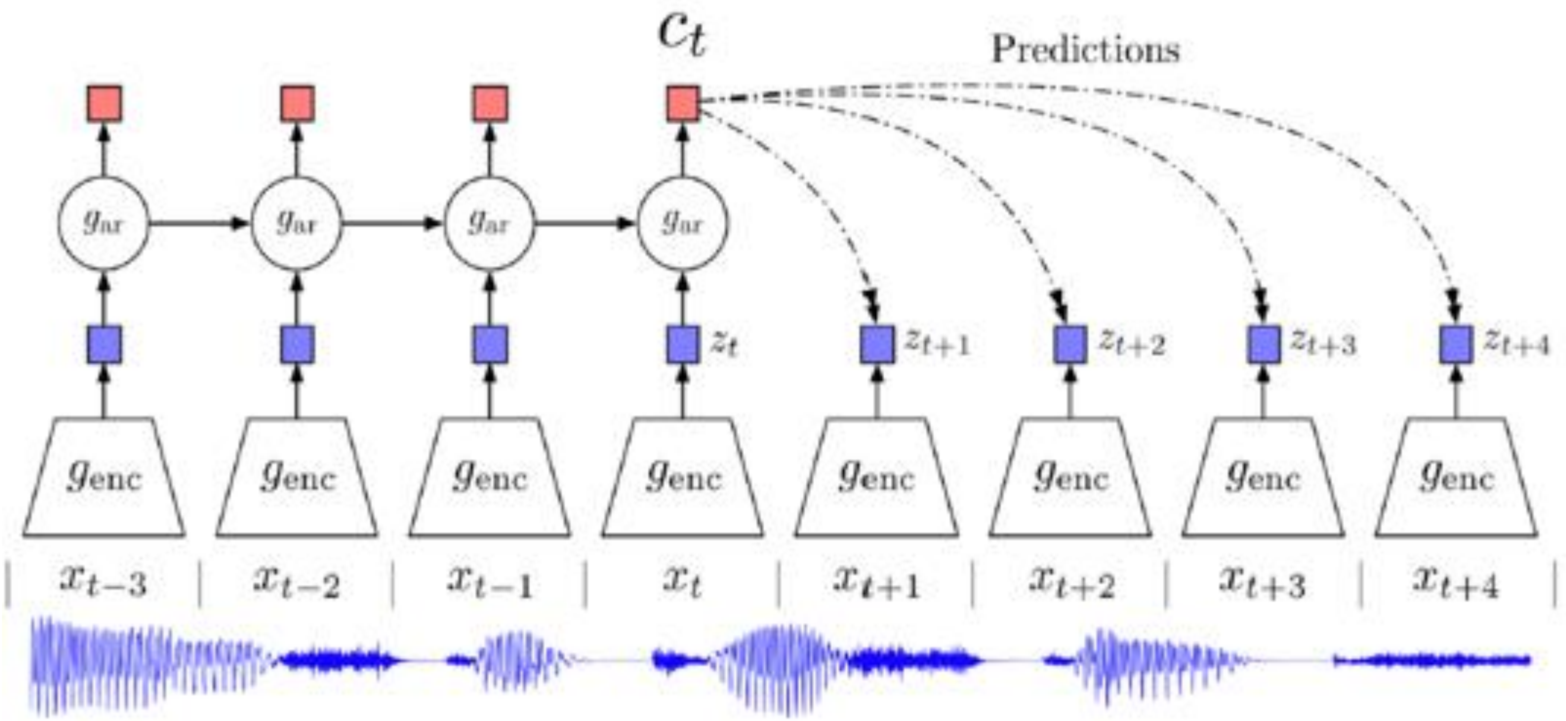
Important exps: ASR on WSJ / TIMIT

Wav2vec



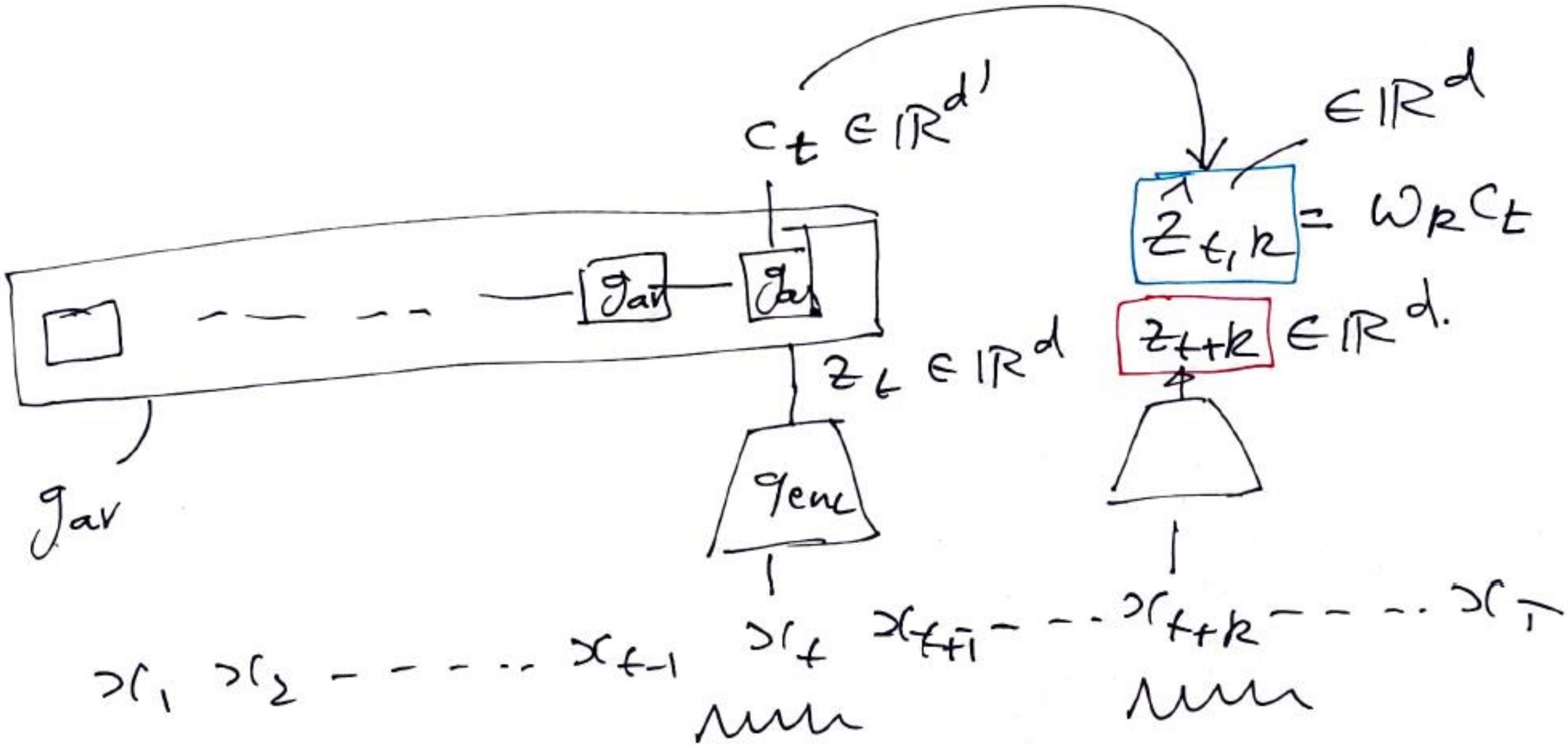
c_0 c_1 c_2 ... c_t c_T
Context Network
210 ms
left padding to create
“causal convolution”
z_0 z_1 z_2 ... z_t z_T
Encoder Network
30 ms

CPC



$$f_k(x_{t+k}, c_t) = \exp\left(z_{t+k}^T W_k c_t\right)$$

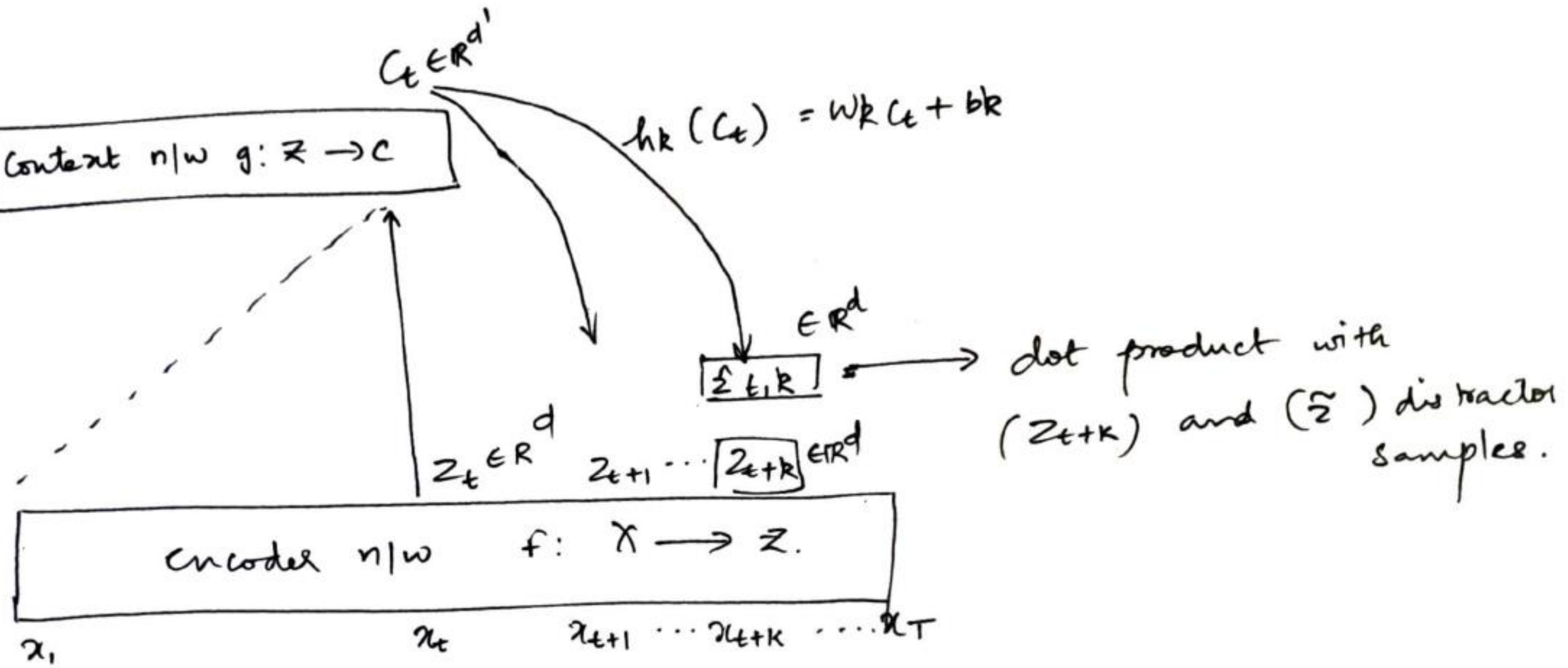
$$\mathcal{L}_\mathrm{N} = -\mathop{\mathbb{E}}_{X}\left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)}\right]$$



$$z_t = g_{enc}(x_t) \longrightarrow \text{10ms latent variable}$$

$$c_t = g_{ar}(z_{\leq t})$$

Notations → wav2vec



$$z_t = f_{enc}(x_t).$$

$c_t = g_{cont}(z_t \cdots z_{t-v})$ for receiptive field size 'v'.

$\boxed{z_{t+k}}$ → distinguish the sample z_{t+k} ; that is k steps in future from distractor samples $\tilde{z}$;

$\tilde{z}$ drawn from a proposal distribution p_n ;

where $p_n(z) = \frac{1}{T}$; where T is the sequence length of the audio.

In practice, sample <u>ten negative examples</u> by uniformly choosing distractors from each audio sequence.

distinguish z_{t+k} from $\tilde{z}$; by minimizing the contrastive loss for each step $k = 1, \ldots K$.

Prediction process is done 'K' times at each timestep 't'.

True sample : z_{t+k}.

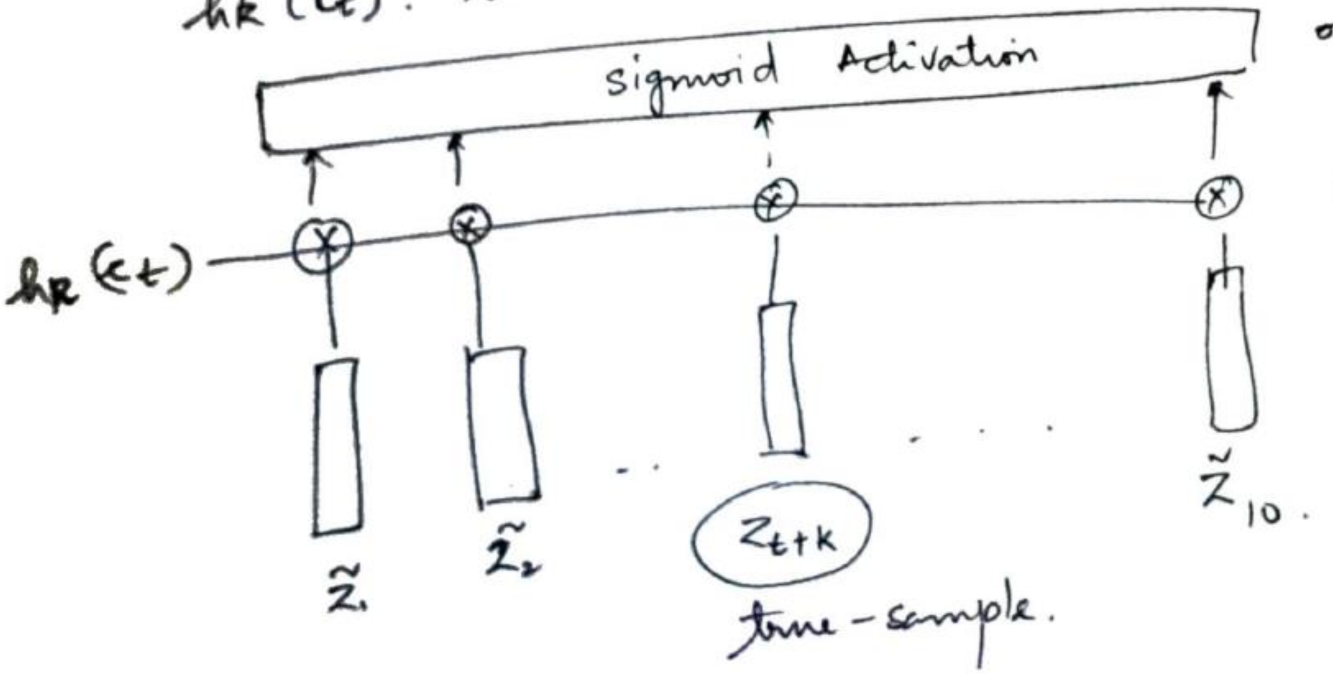
- distractor samples : $\tilde{z}_1, \tilde{z}_2 \dots \tilde{z}_{10}$.

for time-step t; context vector c_t.

undergoes step-specific affine transformation $h_k\,c(t)$

$h_k(c_t)$. similarity with $z_{t+\hat{k}}$ and $\tilde{z}$

#



$\sigma(x) = \dfrac{1}{1+\exp(-x)}$

dot-pdt.

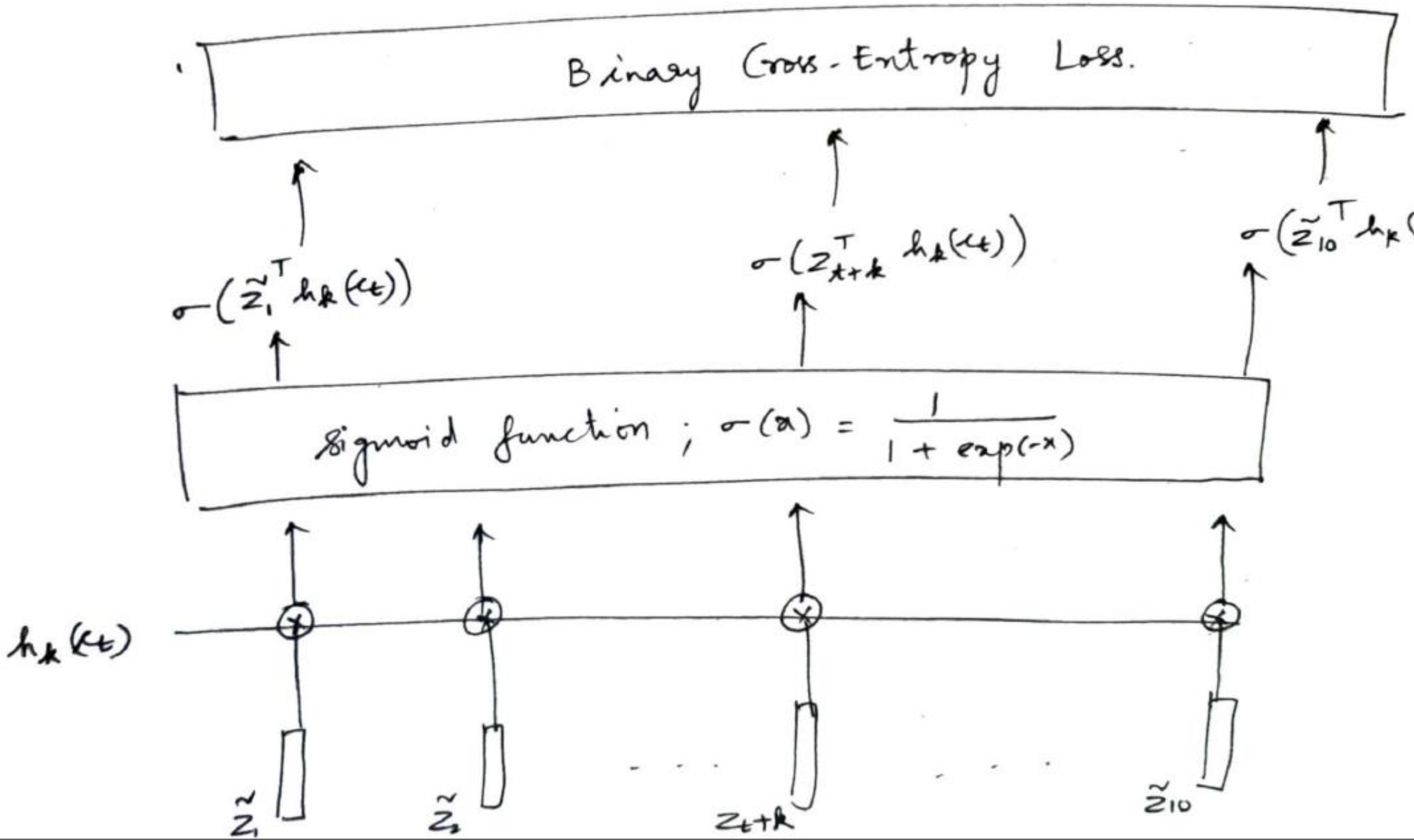
$\left[z_{t+k}^{T}\, h_k(c_t) \right]$

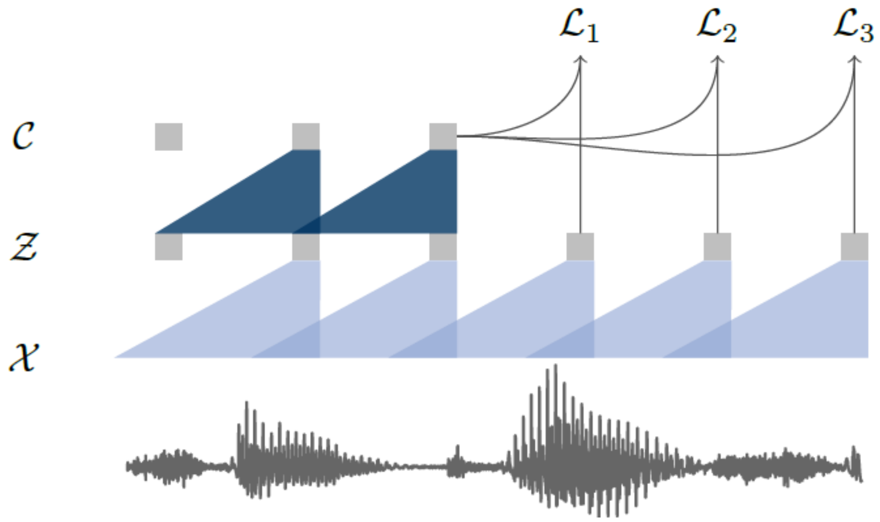
$\left[\tilde{z}^{T}\, h_k(c_t) \right]$

at time step 't'

$$L_k = -\left(\log \sigma\left(z_{t+k}^{T} h_k(c_t)\right) + \lambda \sum_{\tilde{z} \sim p_n}\left[\log \sigma\left(-\tilde{z}^{T} h_k(c_t)\right)\right]\right)$$

$$L = \sum \text{loss of positive pairs} + \sum \text{loss of negative pairs}$$





Total Loss

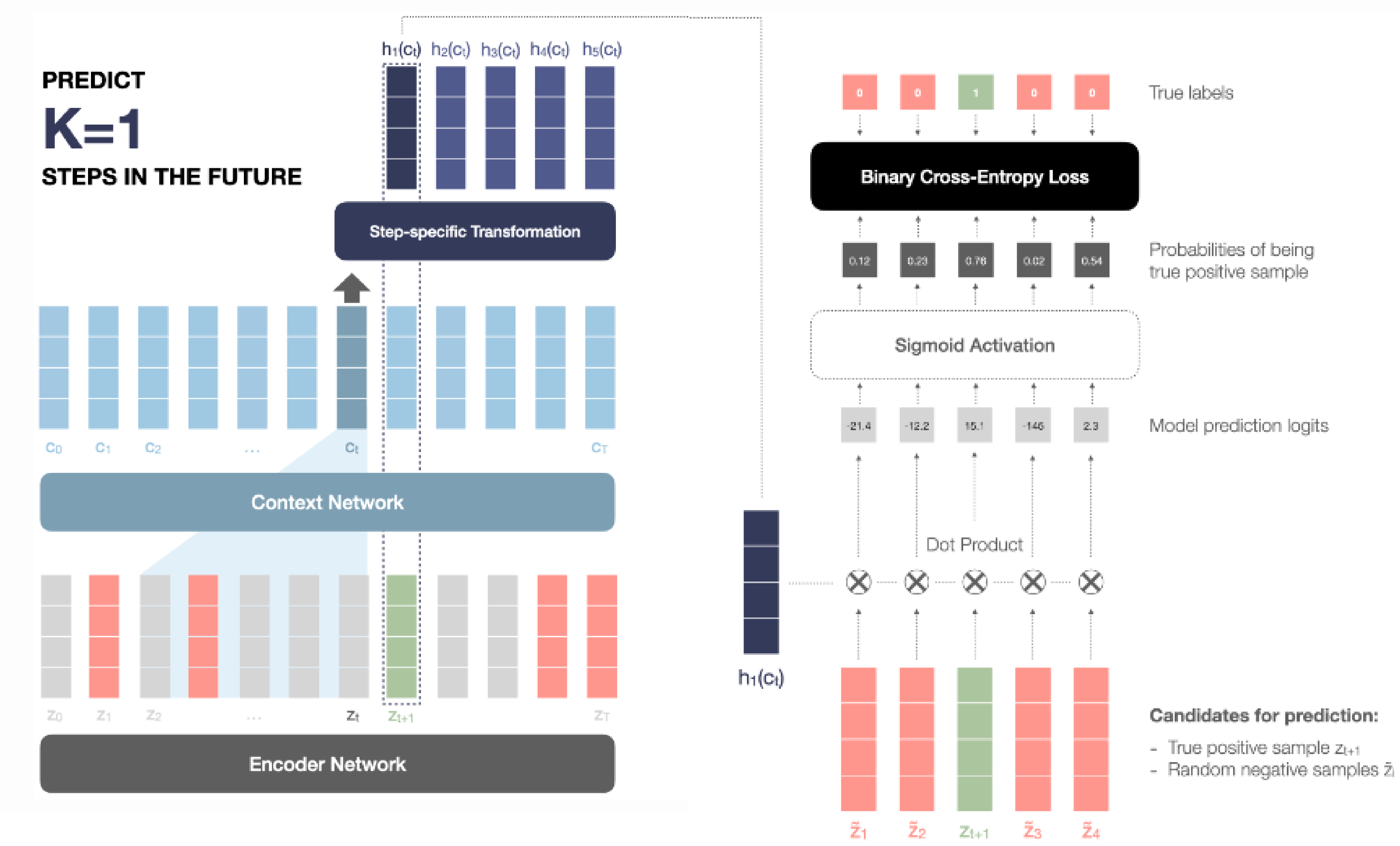
$$L_k = -\sum_{t=1}^{T-k} \left(\log \sigma \left(z_{t+k}^T h_k(c_t) \right) + \lambda \underset{\tilde{z} \sim p_n}{E} \left[\log \sigma \left(-\tilde{z}^T h_k(c_t) \right) \right] \right)$$

and for $k = 1, \ldots K$

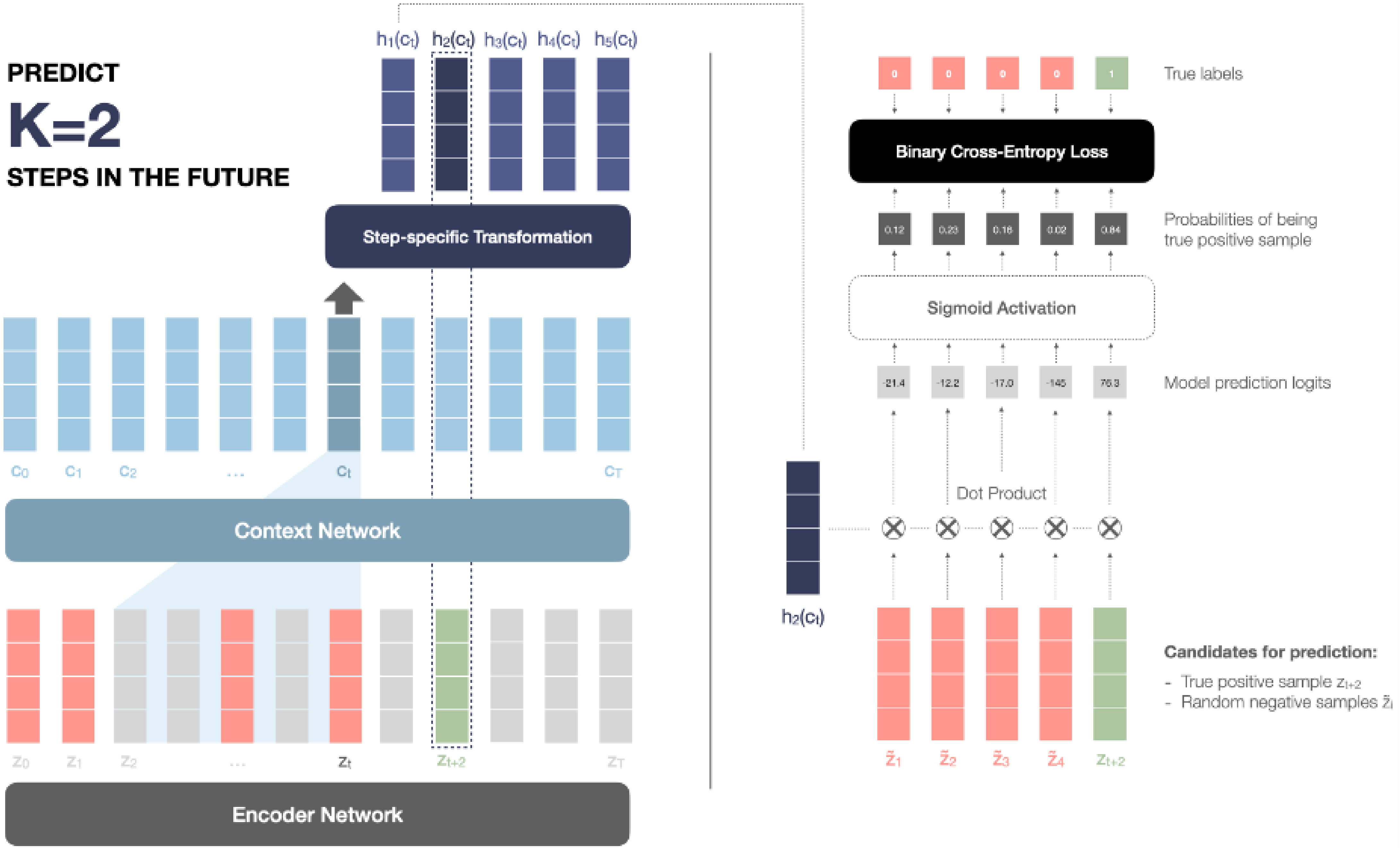
$$L = \sum_{k=1}^{K} L_k .$$

At a given timestep *t*, for each step $k = 1 \ldots 12$, we do the following:

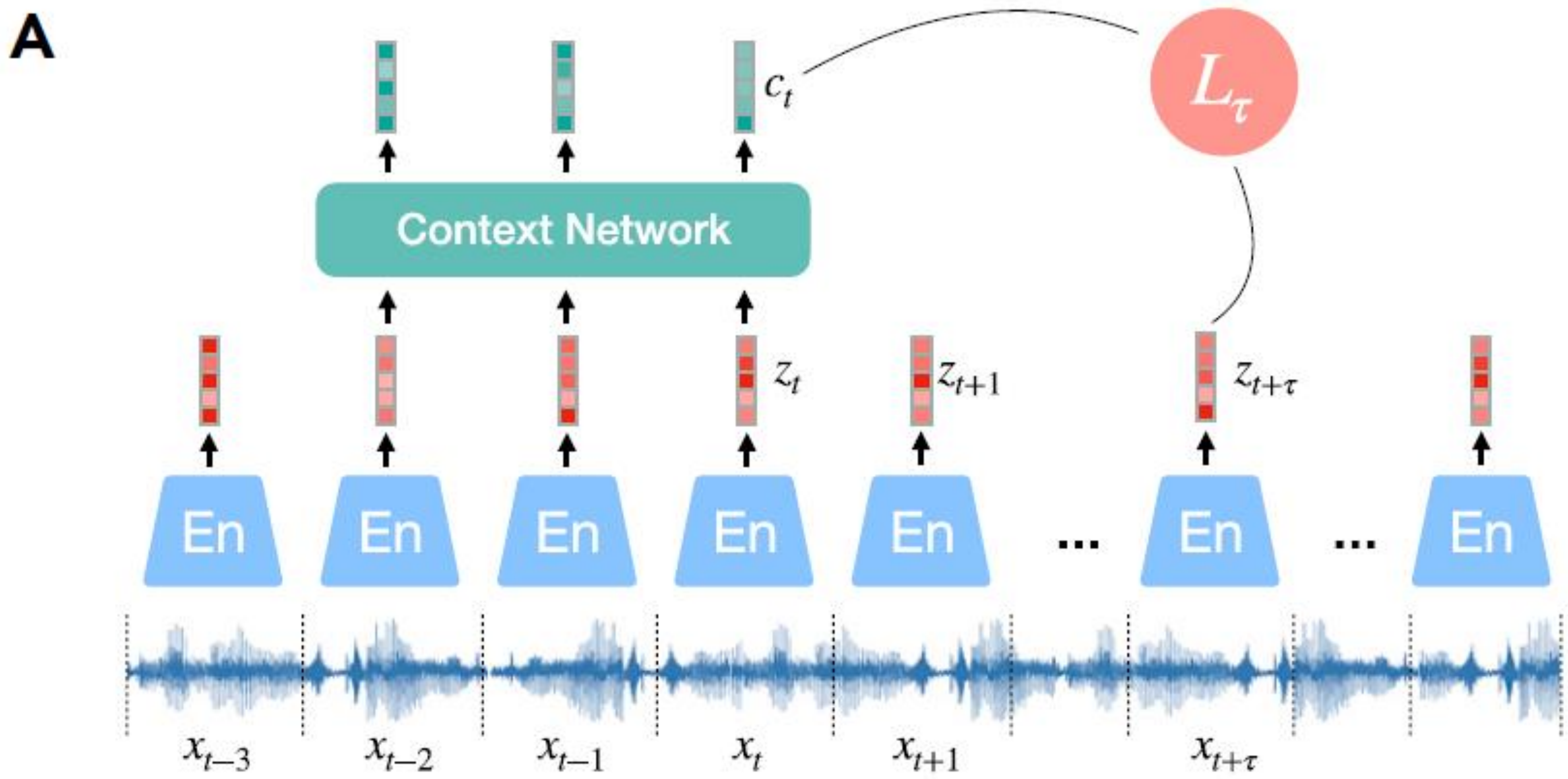
1. Extract the true audio sample $\mathbf{z}_{t+k}$ at future step *k*
2. Pick 10 random *negative* samples $\tilde{\mathbf{z}}_{1..10}$ from the *same audio sequence*
3. Compute a step *k*-specific transformation $\mathbf{h}_k(\mathbf{c}_t)$ of the context vector at time *t*
4. Compute the similarity (dot product) of the transformed context vector with all **z** candidates
5. Compute the final probabilities of positive/negative through a sigmoid activation
6. Compare with the ground truth and penalize the model for wrong predictions (binary cross-entropy loss)



PREDICT
K=1
STEPS IN THE FUTURE
$h_1(c_t)$ $h_2(c_t)$ $h_3(c_t)$ $h_4(c_t)$ $h_5(c_t)$
Step-specific Transformation
c_0 c_1 c_2 … c_t c_T
Context Network
z_0 z_1 z_2 … z_t z_{t+1} z_T
Encoder Network
0 0 1 0 0
True labels
Binary Cross-Entropy Loss
0.12 0.23 0.78 0.02 0.54
Probabilities of being true positive sample
Sigmoid Activation
-21.4 -12.2 15.1 -146 2.3
Model prediction logits
Dot Product
$h_1(c_t)$
Candidates for prediction:
- True positive sample z_{t+1}
- Random negative samples $\tilde{z}_i$
$\tilde{z}_1$ $\tilde{z}_2$ z_{t+1} $\tilde{z}_3$ $\tilde{z}_4$

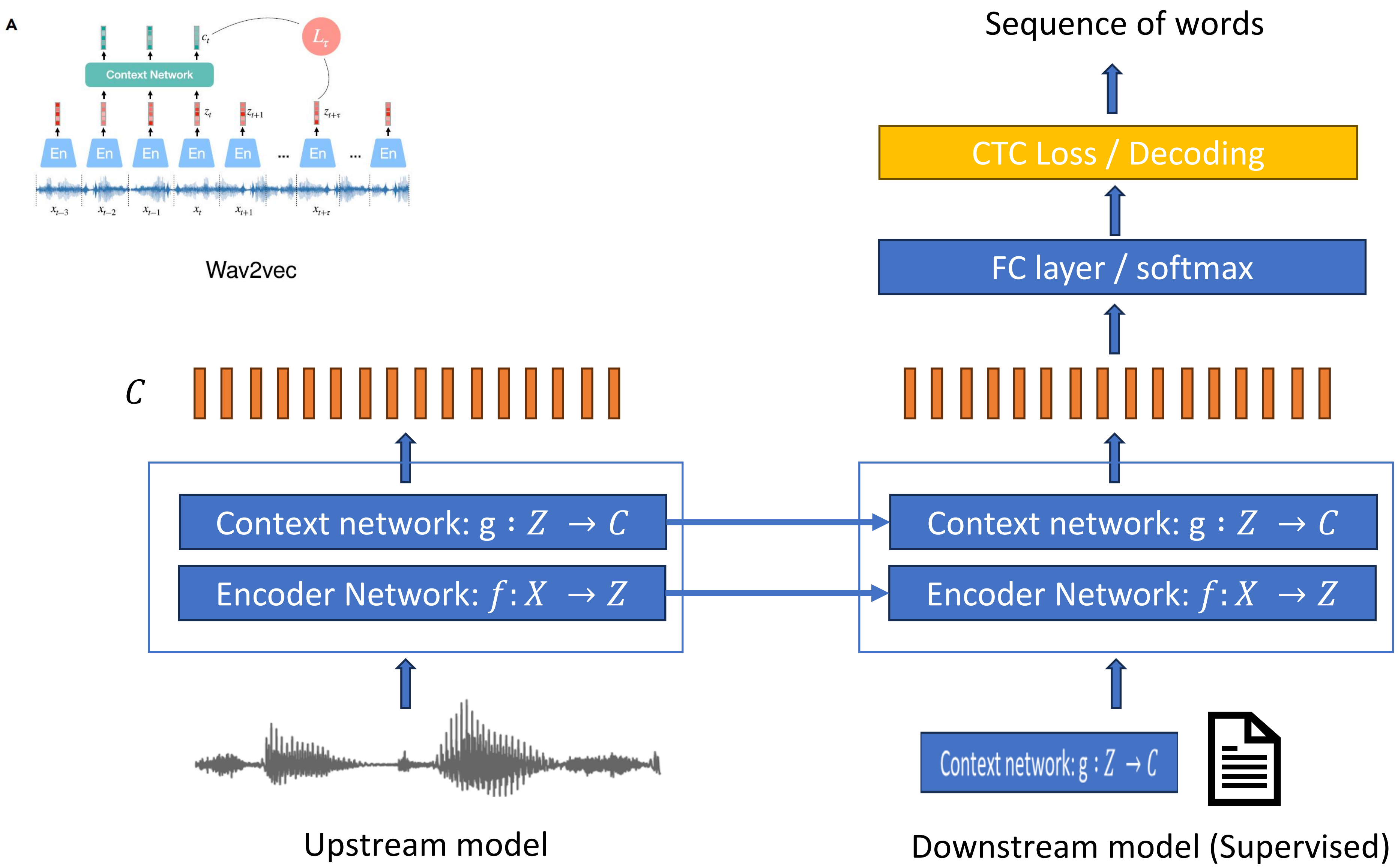


PREDICT
K=2
STEPS IN THE FUTURE
h1(ct) h2(ct) h3(ct) h4(ct) h5(ct)
Step-specific Transformation
c0 c1 c2 ... ct cT
Context Network
z0 z1 z2 ... zt zt+2 zT
Encoder Network
0 0 0 0 1
True labels
Binary Cross-Entropy Loss
0.12 0.23 0.16 0.02 0.84
Probabilities of being true positive sample
Sigmoid Activation
-21.4 -12.2 -17.0 -145 76.3
Model prediction logits
Dot Product
h2(ct)
Candidates for prediction:
- True positive sample zt+2
- Random negative samples z̃i
z̃1 z̃2 z̃3 z̃4 zt+2


A
Context Network
c_t
L_τ
z_t
z_{t+1}
$z_{t+\tau}$
En
...
x_{t-3}
x_{t-2}
x_{t-1}
x_t
x_{t+1}
$x_{t+\tau}$

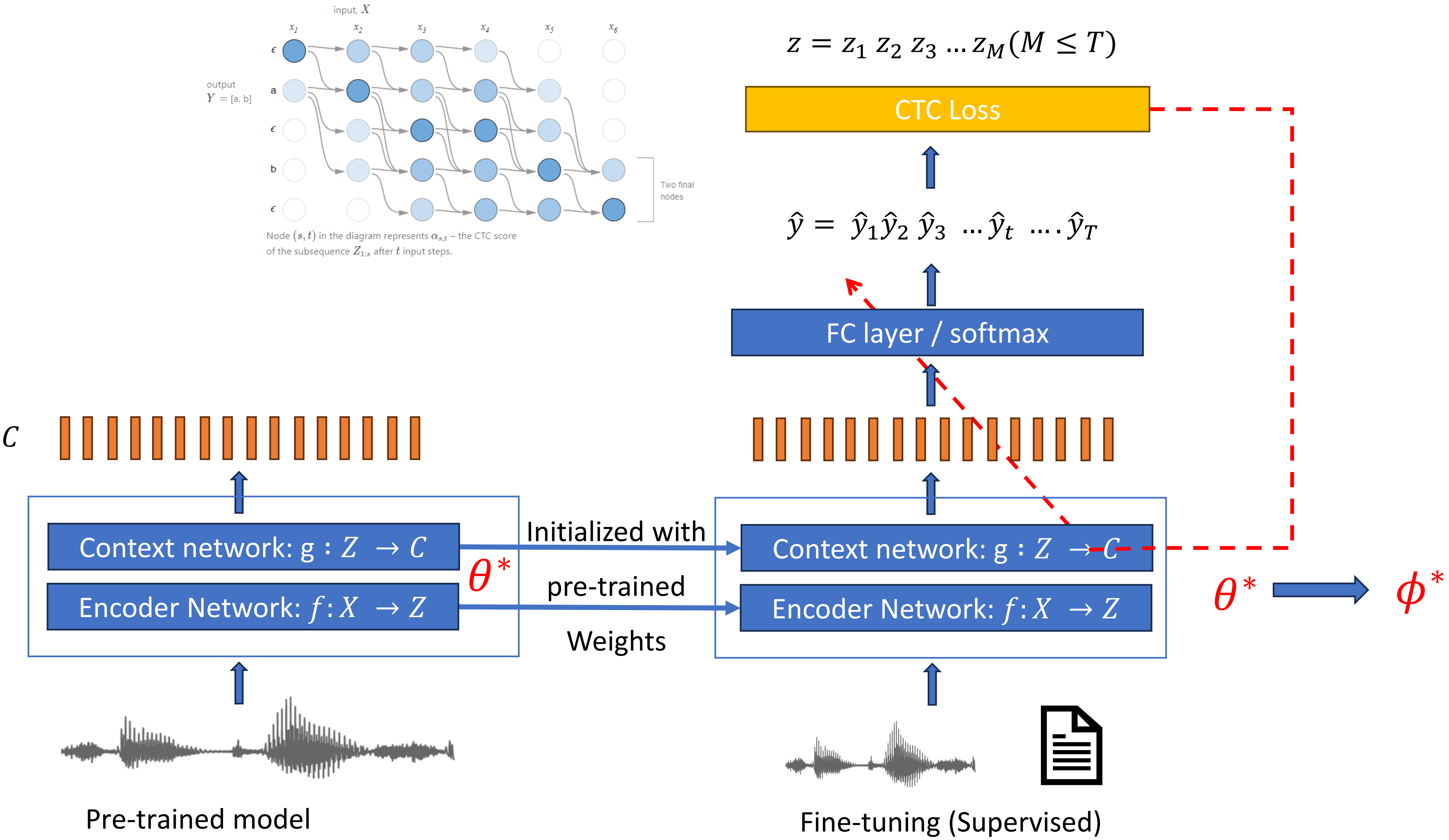
Wav2vec

After training, we input the representations $\mathbf{c}_i$ produced by the context network to the acoustic model instead of log-mel filterbank features.

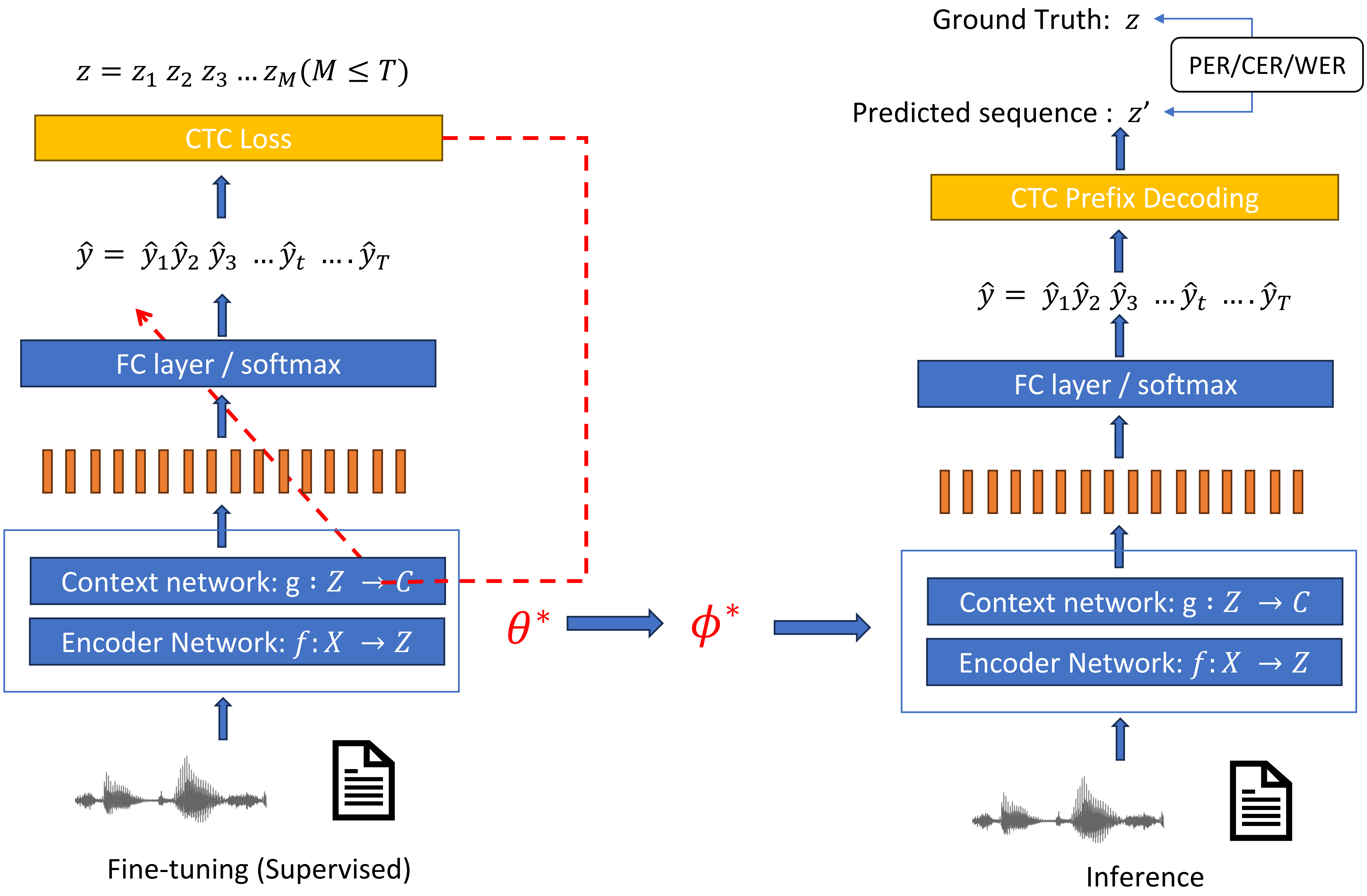


Foundation models in ASR

Pre-training phase → Fine-tuning phase → Inference



input, X
x_1 x_2 x_3 x_4 x_5 x_6
output
Y = [a, b]
ϵ
a
ϵ
b
ϵ
Two final nodes
Node (s, t) in the diagram represents $\alpha_{s,t}$ – the CTC score of the subsequence $Z_{1:s}$ after t input steps.
$z = z_1\, z_2\, z_3 \ldots z_M (M \le T)$
CTC Loss
$\hat{y} = \hat{y}_1 \hat{y}_2\, \hat{y}_3\, \ldots \hat{y}_t\, \ldots . \hat{y}_T$
FC layer / softmax
C
Context network: g : $Z \rightarrow C$
Encoder Network: $f : X \rightarrow Z$
θ^*
Initialized with
pre-trained
Weights
Context network: g : $Z \rightarrow C$
Encoder Network: $f : X \rightarrow Z$
$\theta^* \Rightarrow \phi^*$
Pre-trained model
Fine-tuning (Supervised)



Ground Truth: z
PER/CER/WER
Predicted sequence : z'
$z = z_1\, z_2\, z_3 \ldots z_M (M \leq T)$
CTC Loss
CTC Prefix Decoding
$\hat{y} = \hat{y}_1 \hat{y}_2\, \hat{y}_3\; \ldots \hat{y}_t\; \ldots . \hat{y}_T$
$\hat{y} = \hat{y}_1 \hat{y}_2\, \hat{y}_3\; \ldots \hat{y}_t\; \ldots . \hat{y}_T$
FC layer / softmax
FC layer / softmax
Context network: g : $Z \rightarrow C$
Encoder Network: $f: X \rightarrow Z$
θ^*
ϕ^*
Context network: g : $Z \rightarrow C$
Encoder Network: $f: X \rightarrow Z$
Fine-tuning (Supervised)
Inference

3 Experimental Setup

3.1 Data

We consider the following corpora: For phoneme recognition on TIMIT (Garofolo et al., 1993b) we use the standard train, dev and test split where the training data contains just over three hours of audio data. Wall Street Journal (WSJ; Garofolo et al. (1993a); Woodland et al. (1994)) comprises about 81 hours of transcribed audio data. We train on si284, validate on nov93dev and test on nov92. Librispeech (Panayotov et al., 2015) contains a total of 960 hours of clean and noisy speech for training. For pre-training, we use either the full 81 hours of the WSJ corpus, an 80 hour subset of clean Librispeech, the full 960 hour Librispeech training set or a combination of all of them.

To train the baseline acoustic model we compute 80 log-mel filterbank coefficients for a 25 ms sliding window with stride 10 ms. Final models are evaluated in terms of both word error rate (WER) and letter error rate (LER).

Results

Pre-training phase

Fine-tuning phase

Inference

Pre-training Dataset:
WSJ - 80hrs
Librispeech – 960 hrs

PER task:
TIMIT Speech Corpus

WER task:
WSJ corpus

PER Results

	dev	test
CNN + TD-filterbanks (Zeghidour et al., 2018a)	15.6	18.0
Li-GRU + MFCC (Ravanelli et al., 2018)	–	16.7 ± 0.26
Li-GRU + FBANK (Ravanelli et al., 2018)	–	15.8 ± 0.10
Li-GRU + fMLLR (Ravanelli et al., 2018)	–	14.9 ± 0.27
Baseline	16.9 ± 0.15	17.6 ± 0.11
wav2vec (Librispeech 80h)	15.5 ± 0.03	17.6 ± 0.12
wav2vec (Librispeech 960h)	13.6 ± 0.20	15.6 ± 0.23
wav2vec (Librispeech + WSJ)	$\mathbf{12.9} \pm \mathbf{0.18}$	$\mathbf{14.7} \pm \mathbf{0.42}$

Table 2: Results for phoneme recognition on TIMIT in terms of PER. All our models use the CNN-8L-PReLU-do0.7 architecture (Zeghidour et al., 2018a).

negatives	dev PER	train time (h)
1	16.3	6.1
2	15.8	6.3
5	15.9	8.2
10	15.5	10.5
20	15.7	15.3

Table 3: Effect of different number of negative samples during pre-training for TIMIT on the development set.

WER Results

			nov93dev		nov92	
			LER	WER	LER	WER
Deep Speech 2 (12K h labeled speech; Amodei et al., 2016)			-	4.42	-	3.1
Trainable frontend (Zeghidour et al., 2018a)			-	6.8	-	3.5
Lattice-free MMI (Hadian et al., 2018)			-	5.66†	-	2.8†
Supervised transfer-learning (Ghahremani et al., 2017)			-	4.99†	-	2.53†
4-GRAM LM (Heafield et al., 2013)						
Baseline	–	–	3.32	8.57	2.19	5.64
wav2vec	Librispeech	80 h	3.71	9.11	2.17	5.55
wav2vec	Librispeech	960 h	2.85	7.40	1.76	4.57
wav2vec	Libri + WSJ	1,041 h	2.91	7.59	1.67	4.61
wav2vec large	Librispeech	960 h	2.73	6.96	1.57	4.32
WORD CONVLM (Zeghidour et al., 2018b)						
Baseline	–	–	2.57	6.27	1.51	3.60
wav2vec	Librispeech	960 h	2.22	5.39	1.25	2.87
wav2vec large	Librispeech	960 h	2.13	5.16	1.02	2.53
CHAR CONVLM (Likhomanenko et al., 2019)						
Baseline	–	–	2.77	6.67	1.53	3.46
wav2vec	Librispeech	960 h	2.14	5.31	1.15	2.78
wav2vec large	Librispeech	960 h	2.11	5.10	0.99	2.43

Thank you

https://ai.meta.com/blog/wav2vec-state-of-the-art-speech-recognition-through-self-supervision/