INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

ZeroPrompt: Streaming Acoustic Encoders are Zero-Shot Masked LMs

Xingchen Song[1,2,3], *Di Wu*[2,3],
Binbin Zhang[2,3], *Zhendong Peng*[2,3], *Bo Dang*[3], *Fuping Pan*[3], *Zhiyong Wu*[1]

[1]Tsinghua Univ., Beijing, China [2]Horizon Inc., Beijing, China [3]WeNet Open Source Community

xingchen.song@horizon.ai

Abstract

In this paper, we present ZeroPrompt (Figure 1-(a)) and the corresponding Prompt-and-Refine strategy (Figure 3), two simple but effective **training-free** methods to decrease the Token Display Time (TDT) of streaming ASR models **without any accuracy loss**. The core idea of ZeroPrompt is to append zeroed content to each chunk during inference, which acts like a prompt to encourage the model to predict future tokens even before they were spoken. We argue that streaming acoustic encoders naturally have the modeling ability of Masked Language Models and our experiments demonstrate that ZeroPrompt is engineering cheap and can be applied to streaming acoustic encoders on any dataset without any accuracy loss. Specifically, compared with our baseline models, we achieve 350 $\sim$ 700ms reduction on First Token Display Time (TDT-F) and 100 $\sim$ 400ms reduction on Last Token Display Time (TDT-L), with theoretically and experimentally equal WER on both Aishell-1 and Librispeech datasets.

Index Terms: end-to-end speech recognition, streaming ASR

1. Introduction

In the past few years, end-to-end models, such as connectionist temporal classification (CTC) [1], RNN-Transducer (RNN-T) [2], and attention-based encoder-decoder (AED) [3] models, have achieved significant success on various ASR tasks. Recently, there has been a growing interest in developing end-to-end ASR models with streaming capability. Among them, chunk-based acoustic encoders [4, 5, 6] have gained popularity and have been adopted in many previous works. These methods utilize bi-directional recurrent networks [7] or fully-connected self-attention networks [8] within a chunk. In this work, we primarily focus on chunk-based methods due to their full-context utilization in a chunk.

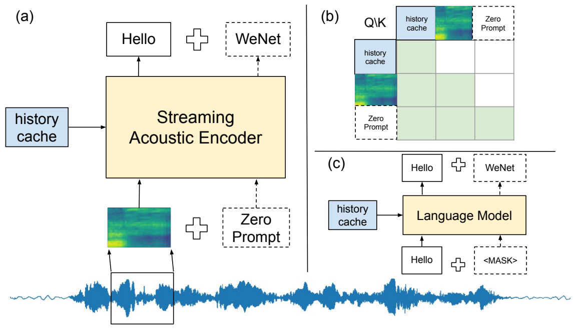

Figure 1: *(a) Illustration of ZeroPrompt. (b) To keep the prediction of the current chunk not affected by zeroed future frames, we use a chunk-level autoregressive attention mask. (c) A symmetrical perspective on Masked LM.*

In streaming scenarios such as real-time subtitles, ASR systems need to decode speech with low latency, producing words as soon as possible [9]. A straightforward way to reduce latency is directly decreasing chunk size (i.e., from 640ms to 320ms). However, there is often a trade-off between performance and latency and lower chunk size usually leads to higher WER. Another way to reduce latency is to apply regularization either on loss function [10, 11] or input spectrogram [12] to push forward the emission of tokens. While being successful in terms of reducing the token emission latency of streaming ASR models, the definition of token emission latency (i.e., The timestamp or frame index when the model predicts the token) underestimates the true user-perceived latency (such as Token Display Time) in chunk-based models, since they do not account for chunk cumulative time (a.k.a, the time to wait before the input signal forms a chunk). Here, we further provide an example to explain why token emission latency does not correlate well with our notion of user-perceived latency. In Figure 2, assume the second char of the recognition result happens at 1000ms and is pushed forward to 800ms after training with emission regularization, the model still needs to wait until 1200ms to form a valid chunk and hence start to decode and emit the second char.

To better measure the latency terms that accurately capture the user-perceived latency, we propose two metrics as illustrated in Figure 2: ***First Token Display Time*** **(TDT-F)** and ***Last Token Display Time*** **(TDT-L)** - the minimum chunk cumulative time required to output the first or last character. In real-time subtitle scenarios, those metrics can be used to evaluate the initial on-screen time of the first and last characters. For simplicity, we ignore the chunk computation time because it is usually much smaller than the chunk cumulative time, i.e., inference one chunk with 640ms chunk size usually takes only 50ms on a desktop CPU using single thread.

In this paper, we explore a training-free method, called ZeroPrompt, which appends zeroed content to each chunk to prompt the model to predict future tokens through its zero-shot ability of Masked LMs that has been implicitly learned during training. We argue that previous works mainly focus on the **decoder part** of encoder-decoder E2E ASR structure rather than the **encoder part** to estimate the internal LM because the encoder part is usually optimized with CTC loss and CTC is generally not considered capable of modeling context between output tokens due to conditional independence assumption [15]. However, CTC-optimized ASR encoders learn the training data distribution and are affected by the frequency of words in the training data. The CTC-optimized encoder therefore at least has the modeling ability of a unigram LM to do something like MaskPredict (see Figure 1-(a) and Figure 1-(c) for a clearer comparison between ZeroPrompt and MaskPredict [16]), and this paper aims to adopt this zero-shot ability to predict future tokens even before they were spoken and hence greatly reduce the TDT-F & TDT-L during inference. Besides, to ensure that the final decoding result is not affected, we propose to use a chunk-level autoregressive attention mask described in Figure 1-(b), coupled with a revision strategy called Prompt-and-Refine, to iteratively predict future tokens and refine them when the real future chunk arrives (see Figure 3 for a detailed example). Experimental results in Section 3 show that our methods

10.21437/Interspeech.2023-1497

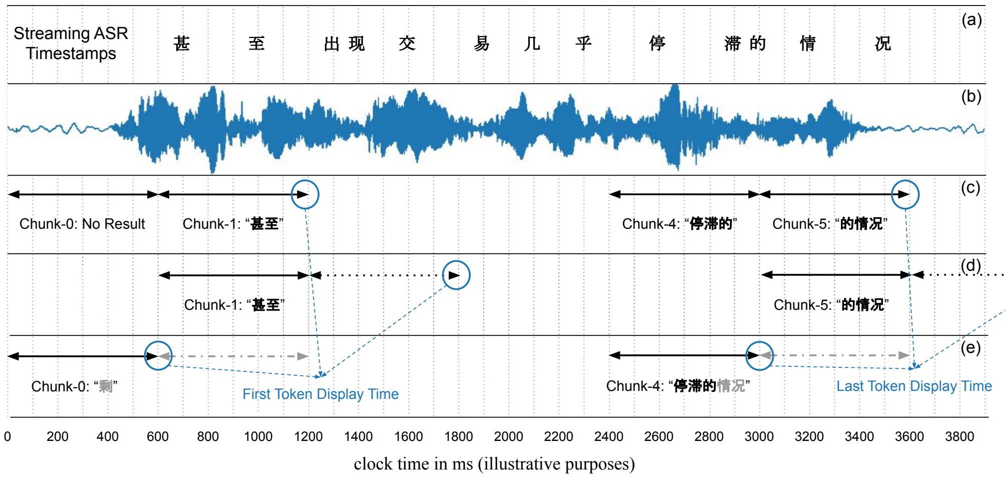

Figure 2: *Illustration of timeline and latency metrics of a streaming ASR system. From top to bottom: (a) Streaming ASR timestamps. (b) Waveforms. (c) Causal method, 600ms chunk size without right context. (d) LookAhead methods [13, 14], 600ms chunk size with 600ms* ***real*** *right context (dotted line in black, a.k.a. LookAhead chunk). (e) ZeroPrompt method, 600ms chunk size with 600ms zeroed context (dash-dotted line in grey, a.k.a ZeroPrompt chunk), the black tokens mean predictions from the current chunk while grey tokens mean predictions from ZeroPrompt chunk.*

have many advantages which can be summarized as:

- ZeroPrompt does not require any model re-training and it takes nearly zero engineering cost to plugin any chunk-based streaming decoding procedure.
- ZeroPrompt can not only decrease the TDT-F & TDT-L for **partial** recognition results but also keep the WER unaffected for **final** recoginition results. In other words, we achieve the theoretically and experimentally best trade-off between latency and WER.

2. Proposed Methods & Related Works

As shown in Figure 1-(a), during inference, we process the utterance chunk-by-chunk, and append a certain number of zeroed future frames (called ZeroPrompt chunk) to each chunk. The history cache, current chunk, and ZeroPrompt chunk are together fed to the acoustic encoder to produce the prediction for both the current chunk ("Hello") and the ZeroPrompt chunk ("WeNet"). Figure 1-(c) reveals that streaming acoustic encoders are zero-shot Masked Language Models (Masked LMs) and hence the ability of ZeroPrompt is something like MaskPredict used in standard Masked LMs.

This paper is related to LookAhead methods which use either **real** [13, 14, 17] or **fake** [18] future frames. In previous work [13, 14], using the real right context requires waiting for the arrival of future content, which results in additional latency (Figure 2-(d)). Another study [17] proposed a 2-pass strategy to process the current chunk first and revise it later once the future chunk is received, but its TDT-F & TDT-L are identical to our baseline causal method (Figure 2-(c)) when compared within equal chunk size.

To avoid waiting for future context, CUSIDE [18] proposed an extra simulation encoder that is jointly trained with the ASR model and optimized with a self-supervised loss called autoregressive predictive coding (APC) [19] to simulate a certain number of future frames for every chunk. While both CUSIDE and ZeroPrompt generate fake future information to avoid waiting time, they differ in how they utilize the generated futures. Specifically, ZeroPrompt directly concatenates the decoding results from the current chunk (black tokens in Figure 2-(e)) and ZeroPrompt chunk (grey tokens in Figure 2-(e)), whereas CUSIDE only uses the result from the current chunk (black tokens in Figure 2-(d)) as decoding output, and the simulated future is only used to enhance the recognition accuracy of the current chunk. Due to the different usage of the fake future content, the TDT-F & TDT-L of CUSIDE are still identical to our causal baseline under equal chunk size. Moreover, ZeroPrompt uses much simpler zero padding to get fake futures, so it does not require any extra parameters or model re-training compared to CUSIDE. Thanks to the internal ability of the Masked LM that is implicitly learned by the streaming encoder during training, ZeroPrompt can emit certain tokens even if the input is all zero.

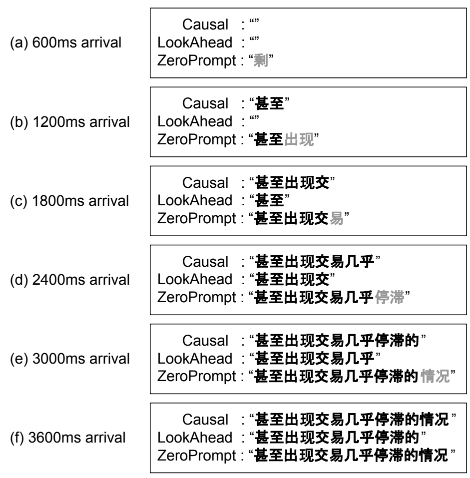

Figure 3: *Comparison of on-screen time among three methods. We can clearly see that ZeroPrompt significantly improved the user-perceived latency. By comparing the result of ZeroPrompt in (a) & (b), we observe that the mistake made by the first ZeroPrompt chunk ("剩") is quickly fixed after the arrival of the second chunk which contains the real infos of the first few characters ("甚至"), this is so called Prompt-and-Refine.*

We further provide a concrete example to compare ZeroPrompt with other methods in Figure 2. It should be noted that it's reasonable for the predictions from the first ZeroPrompt chunk to be inaccurate due to the lack of contextual information. However, this is not a significant issue since most of the errors are homophones of the correct counterparts, i.e., ("剩, sheng in English") vs. ("甚, shen in English") in this exam-

Table 1: *Comparison of different ZeroPrompt length across different chunk size on different dataset. From left to right: (a) length of ZeroPrompt. (b) First Token Display Time (TDT-F). (c) Last Token Display Time (TDT-L). (d) Prompts Error Rate for First chunk (PER-F). (e) Prompts Error Rate for Last chunk (PER-L). (f) Prompts Error Rate for All chunks (PER-A). (g) Word Error Rate (WER, 1st-pass Greedy Search / 2nd-pass Rescore). (h) Real Time Factor (RTF, 1st-pass Greedy Search / 2nd-pass Rescore, tested on Intel(R) Core(TM) i5-8400 CPU @ 2.80GHz using int8 quantization and single-thread). (i) Prompts Per Chunk (PPC). We note that the PER of Librispeech is significantly lower than that of Aishell-1. This is because we decode Librispeech using Byte Pair Encoding (BPE) but calculate the Prompts Error Rate using English characters. A BPE usually consists of several characters, and even if the BPE is incorrect, there may be correct letters, in other words, the denominator of PER increases while the numerator decreases.*

(a) ZeroPrompt	(b)TDT-F	(c) TDT-L	(d) PER-F (%)	(e) PER-L (%)	(f) PER-A (%)	(g) WER (%)	(h) RTF	(i) PPC
Aishell-1 (test), 104765 total characters, 7176 total sentences								
640ms chunk size with 59081 total chunks								
0ms	1279ms (∼)	4806ms (∼)	-	-	-	5.81 / 5.05	0.04351 / 0.05063	-
80ms	1272ms (↓7)	4762ms (↓44)	87 / 2191 = 3.9%	7 / 947 = 0.7%	442 / 12059 = 3.6%	5.81 / 5.05	0.04816 / 0.05495	0.20
160ms	1234ms (↓45)	4706ms (↓100)	266 / 4351 = 6.1%	19 / 1937 = 0.9%	1162 / 23450 = 4.9%	5.81 / 5.05	0.05009 / 0.05722	0.39
320ms	876ms (↓403)	4603ms (↓203)	1834 / 7457 = 24.5%	152 / 4211 = 3.6%	5183 / 46180 = 11.2%	5.81 / 5.05	0.05378 / 0.06282	0.78
640ms	646ms (↓633)	4472ms (↓334)	5816 / 10150 = 57.3%	867 / 7408 = 11.7%	20181 / 71091 = 28.3%	5.81 / 5.05	0.06447 / 0.07425	1.20
1280ms	646ms (↓633)	4432ms (↓374)	6563 / 10570 = 62.0%	1217 / 7712 = 15.7%	24179 / 74220 = 32.5%	5.81 / 5.05	0.08486 / 0.09876	1.26
320ms chunk size with 114559 total chunks								
0ms	1015ms (∼)	4575ms (∼)	-	-	-	6.13 / 5.27	0.06007 / 0.06748	-
80ms	965ms (↓50)	4551ms (↓24)	109 / 2406 = 4.5%	6 / 1770 = 0.3%	1263 / 25484 = 4.9%	6.13 / 5.27	0.06609 / 0.07526	0.22
160ms	939ms (↓76)	4524ms (↓51)	289 / 4697 = 6.1%	33 / 3575 = 0.9%	28823 / 48779 = 5.9%	6.13 / 5.27	0.07446 / 0.07884	0.43
320ms	762ms (↓253)	4443ms (↓132)	1795 / 7939 = 22%	224 / 7677 = 2.9%	9692 / 91571 = 10.5%	6.13 / 5.27	0.07974 / 0.08979	0.80
640ms	641ms (↓374)	4353ms (↓222)	5750 / 10065 = 57%	1595 / 11493 = 13.8%	36893 / 137868 = 26.7%	6.13 / 5.27	0.09645 / 0.11290	1.20
1280ms	621ms (↓394)	4290ms (↓285)	6509 / 10268 = 63.3%	2052 / 11767 = 17.4%	44869 / 144402 = 31.0%	6.13 / 5.27	0.13690 / 0.15990	1.26
160ms chunk size with 225482 total chunks								
0ms	971ms (∼)	4423ms (∼)	-	-	-	6.35 / 5.39	0.09616 / 0.10590	-
80ms	889ms (↓82)	4428ms (↑5)	231 / 3718 = 6.2%	3 / 835 = 0.3%	1655 / 51827 = 3.1%	6.35 / 5.39	0.10830 / 0.12070	0.23
160ms	826ms (↓145)	4446ms (↑23)	659 / 6552 = 10.0%	53 / 4294 = 1.2%	4995 / 99480 = 5.0%	6.35 / 5.39	0.11180 / 0.12530	0.44
320ms	700ms (↓271)	4388ms (↓35)	2150 / 7549 = 28.4%	276 / 7785 = 3.5%	18433 / 182513 = 10.0%	6.35 / 5.39	0.13040 / 0.14710	0.81
640ms	574ms (↓397)	4271ms (↓152)	5894 / 8527 = 69%	2104 / 12304 = 17.1%	73053 / 275551 = 26.5%	6.35 / 5.39	0.16700 / 0.19220	1.22
1280ms	549ms (↓422)	4234ms (↓189)	6761 / 8918 = 75.8%	2612 / 12544 = 20.8%	89647 / 289123 = 31.0%	6.35 / 5.39	0.24220 / 0.28250	1.28
Librispeech (test_clean), 283993 total characters, 2620 total sentences								
640ms chunk size with 31381 total chunks								
0ms	1136ms (∼)	7328ms (∼)	-	-	-	4.41 / 3.80	0.04826 / 0.05644	-
80ms	1038ms (↓98)	7280ms (↓48)	60 / 4501 = 1.3%	35 / 1607 = 2.1%	459 / 35507 = 1.2%	4.41 / 3.80	0.05184 / 0.06111	1.13
160ms	935ms (↓201)	7235ms (↓93)	146 / 8344 = 1.7%	49 / 3040 = 1.6%	1112 / 68465 = 1.6%	4.41 / 3.80	0.05543 / 0.06435	2.18
320ms	761ms (↓375)	7149ms (↓179)	812 / 13916 = 5.8%	230 / 5929 = 3.8%	5667 / 123304 = 4.5%	4.41 / 3.80	0.05951 / 0.06979	3.93
640ms	662ms (↓474)	7098ms (↓230)	2552 / 17570 = 14.5%	577 / 8002 = 7.2%	16743 / 159472 = 10.4%	4.41 / 3.80	0.07006 / 0.08295	5.08
1280ms	658ms (↓478)	7091ms (↓237)	2522 / 17696 = 14.2%	658 / 8372 = 7.8%	18085 / 162531 = 11.1%	4.41 / 3.80	0.09096 / 0.10720	5.18
320ms chunk size with 61432 total chunks								
0ms	928ms (∼)	7147ms (∼)	-	-	-	4.76 / 4.04	0.06996 / 0.08025	-
80ms	853ms (↓75)	7128ms (↓19)	67 / 5185 = 1.2%	71 / 2552 = 2.7%	1041 / 70996 = 1.4%	4.76 / 4.04	0.07476 / 0.08630	1.16
160ms	789ms (↓139)	7091ms (↓56)	157 / 9028 = 1.7%	69 / 5103 = 1.3%	2372 / 136656 = 1.7%	4.76 / 4.04	0.08155 / 0.09210	2.22
320ms	662ms (↓266)	7005ms (↓142)	839 / 13855 = 6.0%	363 / 10664 = 3.4%	10814 / 246692 = 4.3%	4.76 / 4.04	0.08963 / 0.10370	4.02
640ms	569ms (↓359)	6950ms (↓197)	2361 / 15297 = 15.0%	977 / 13863 = 7.0%	29997 / 317552 = 9.4%	4.76 / 4.04	0.10890 / 0.12630	5.17
1280ms	561ms (↓367)	6945ms (↓202)	2389 / 15241 = 15.6%	1135 / 14228 = 7.9%	32287 / 323612 = 9.9%	4.76 / 4.04	0.14990 / 0.17770	5.28
160ms chunk size with 121531 total chunks								
0ms	857ms (∼)	7043ms (∼)	-	-	-	5.10 / 4.30	0.11770 / 0.12970	-
80ms	786ms (↓71)	7063ms (↑20)	65 / 5395 = 1.2%	59 / 2345 = 2.5%	1462 / 140612 = 1.0%	5.10 / 4.30	0.12880 / 0.14350	1.16
160ms	704ms (↓153)	7048ms (↑5)	135 / 10685 = 1.2%	84 / 5830 = 1.4%	3833 / 271459 = 1.4%	5.10 / 4.30	0.13480 / 0.15050	2.23
320ms	579ms (↓278)	6959ms (↓84)	833 / 11942 = 6.9%	470 / 11533 = 4.0%	16573 / 493650 = 3.3%	5.10 / 4.30	0.14760 / 0.17060	4.06
640ms	505ms (↓352)	6909ms (↓134)	2246 / 12438 = 18%	1274 / 14642 = 8.7%	44768 / 638700 = 7.0%	5.10 / 4.30	0.19030 / 0.22190	5.26
1280ms	502ms (↓355)	6903ms (↓140)	2381 / 12612 = 18.8%	1438 / 15262 = 9.4%	48181 / 649938 = 7.4%	5.10 / 4.30	0.26960 / 0.31480	5.35
Librispeech (test_other), 274213 total characters, 2939 total sentences								
640ms chunk size with 31120 total chunks								
0ms	1209ms (∼)	6428ms (∼)	-	-	-	11.48 / 10.40	0.04826 / 0.05644	-
80ms	1130ms (↓79)	6407ms (↓21)	126 / 5013 = 2.5%	47 / 1708 = 2.7%	800 / 33840 = 2.3%	11.48 / 10.40	0.05184 / 0.06111	1.09
160ms	1032ms (↓177)	6362ms (↓66)	366 / 9226 = 3.9%	110 / 3243 = 3.3%	2342 / 66182 = 3.5%	11.48 / 10.40	0.05543 / 0.06435	2.13
320ms	821ms (↓388)	6252ms (↓176)	1545 / 15336 = 10.0%	371 / 6970 = 5.3%	9839 / 121799 = 8.0%	11.48 / 10.40	0.05951 / 0.06979	3.91
640ms	668ms (↓541)	6208ms (↓220)	3446 / 18363 = 18.7%	904 / 9195 = 9.8%	23456 / 158130 = 14.8%	11.48 / 10.40	0.07006 / 0.08295	5.08
1280ms	665ms (↓544)	6202ms (↓226)	3598 / 18616 = 19.3%	1033 / 9549 = 10.8%	24264 / 160299 = 15.1%	11.48 / 10.40	0.09096 / 0.10720	5.15
320ms chunk size with 60793 total chunks								
0ms	978ms (∼)	6215ms (∼)	-	-	-	12.19 / 11.06	0.06996 / 0.08025	-
80ms	898ms (↓80)	6235ms (↑20)	150 / 5671 = 2.6%	79 / 2773 = 2.8%	1606 / 67888 = 2.3%	12.19 / 11.06	0.07476 / 0.08630	1.12
160ms	840ms (↓138)	6194ms (↓21)	378 / 10017 = 3.7%	130 / 5570 = 2.3%	4445 / 131505 = 3.3%	12.19 / 11.06	0.08155 / 0.09210	2.16
320ms	716ms (↓262)	6108ms (↓107)	1526 / 15766 = 9.6%	595 / 12212 = 4.8%	17909 / 241994 = 7.4%	12.19 / 11.06	0.08963 / 0.10370	3.98
640ms	613ms (↓365)	6052ms (↓163)	3339 / 17106 = 19.5%	1578 / 15498 = 10.1%	41238 / 312230 = 13.2%	12.19 / 11.06	0.10890 / 0.12630	5.13
1280ms	611ms (↓367)	6051ms (↓164)	3517 / 17336 = 20.2%	1699 / 15953 = 10.6%	42756 / 316521 = 13.5%	12.19 / 11.06	0.14990 / 0.17770	5.21
160ms chunk size with 120144 total chunks								
0ms	909ms (∼)	6095ms (∼)	-	-	-	13.14 / 11.85	0.11770 / 0.12970	-
80ms	835ms (↓74)	6124ms (↑29)	114 / 6222 = 1.8%	70 / 3211 = 2.1%	2192 / 134209 = 1.6%	13.14 / 11.85	0.12880 / 0.14350	1.11
160ms	758ms (↓151)	6108ms (↑13)	363 / 12015 = 3.0%	126 / 7313 = 1.7%	6349 / 262655 = 2.4%	13.14 / 11.85	0.13480 / 0.15050	2.19
320ms	629ms (↓280)	6045ms (↓50)	1467 / 13593 = 10.7%	931 / 14206 = 6.5%	26746 / 487285 = 5.4%	13.14 / 11.85	0.14760 / 0.17060	4.06
640ms	552ms (↓357)	5990ms (↓105)	3169 / 14189 = 22.3%	2239 / 17967 = 12.4%	60588 / 628283 = 9.6%	13.14 / 11.85	0.19030 / 0.22190	5.23
1280ms	548ms (↓361)	5982ms (↓113)	3319 / 14385 = 23.0%	2435 / 18567 = 13.1%	62239 / 636094 = 9.7%	13.14 / 11.85	0.26960 / 0.31480	5.29

ple. Additionally, these errors will be quickly corrected by the Prompt-and-Refine strategy, as demonstrated in Figure 3.

3. Experiments

To demonstrate the effectiveness of our proposed ZeroPrompt, we carry out our experiments on the open-source Chinese Mandarin speech corpus Aishell-1 [20] and English speech corpus Librispeech [21]. ZeroPrompt is a training-free method and can be directly applied to a well-trained chunk-based ASR model. To ensure the reproducibility of experiments, we used checkpoints downloaded from the official WeNet [22] website for all of our baseline models and keep the exact same settings as in open-sourced Aishell-1 and Librispeech recipes.

3.1. Metrics

Besides **Token Display Time** (TDT, TDT-F for First token and TDT-L for Last token), **Word Error Rate** (WER) and **Real Time Factor** (RTF), we propose several additional metrics to better analyze the effectiveness of ZeroPrompt. Specifically, we introduce two new metrics that are designed for ZeroPrompt:

- **Prompts Error Rate** (PER, PER-F for First chunk, PER-L for Last chunk and PER-A for All chunks): PER is calculated by dividing Prompt Errors (PE) by the Number of Prompts (NP). NP represents the number of future characters decoded from the ZeroPrompt chunk, while PE denotes the number of errors that occur among those characters.
- **Prompts Per Chunk** (PPC): The PPC is obtained by dividing the total number of prompts by the total number of chunks. This metric provides insight into the average number of future characters prompted per chunk.

3.2. Main Results

We present the main results of ZeroPrompt in Table 1, from which 5 conclusions can be deduced:

- A larger ZeroPrompt length generally results in lower Token Display Time (TDT) for all languages and chunk sizes. However, when the length exceeds a certain threshold (i.e., greater than 640ms), there is a latency ceiling imposed by both the chunk size (TDT cannot be smaller than chunk size due to the required data collecting time) and the leading silence (the ASR model cannot prompt tokens if both the current chunk and the ZeroPrompt chunk contain only silences or zeros).
- A larger ZeroPrompt length also results in a higher PER, but this is not a significant problem because they can be rapidly corrected using our Prompt-and-Refine strategy, which is described in Section 2 and illustrated in Figure 3.
- The closer a chunk is to the end of a sentence, the more accurate the prompts are. It is clear that PER-L is much better than PER-F, which is reasonable because the first tokens often lack context information while the last tokens have richer context.
- Thanks to the autoregressive attention mask (Figure 1-(b)) and the Prompt-and-Refine strategy (Figure 3), the WER for the final result remained unchanged. However, we observed a slight increase in RTF due to the increased input length. It's worth noting that, if compared within a similar RTF, *[640ms chunk size & 640ms ZeroPrompt, Aishell-1, RTF 0.06447/0.07425]* significantly outperforms *[320ms chunk size & 0ms ZeroPrompt, Aishell-1, RTF 0.06007/0.06748]* in TDT-F (646ms v.s. 1015ms), TDT-L (4472ms v.s. 4575ms) and WER (5.81/5.05 v.s. 6.13/5.27). This is mainly because the 640ms chunk size provides more context information than the 320ms chunk size, and the 640ms ZeroPrompt greatly reduces latency compared to the 0ms baseline. Moreover, to offer users greater flexibility in balancing latency (TDT & PPC) and RTF, we further discuss a solution in Section 3.3.
- It appears that PPC only correlates with ZeroPrompt length, as different chunk sizes result in similar PPC values.

Overall, based on the results from Aishell-1, we can conclude that ZeroPrompt provides the best trade-off between latency (TDT & PPC) and WER, both theoretically and experimentally. It achieves a reduction of 350 $\sim$ 700ms in TDT-F and 100 $\sim$ 400ms in TDT-L, while keeping WER unchanged. This conclusion is further supported by the results from Librispeech, which demonstrate that ZeroPrompt generalizes well to any dataset without requiring any extra effort.

3.3. Solution to balance latency-RTF Trade-off

As described in Section 3.2, although the latency-WER tradeoff has been solved, there is also a trade-off between latency (TDT & PPC) and RTF. In this section, we present a solution, called **Intermediate ZeroPrompt**, to better balance latency and RTF. Specifically, we feed the ZeroPrompt chunk starting from different encoder layers to achieve different computation costs. From Table 2, it can be observed that one can simply change the start layer to meet the desired latency and RTF requirements.

Table 2: *Results of Intermediate ZeroPrompt [640ms chunk size & 640ms ZeroPrompt, Aishell-1]. 0 means we feed ZeroPrompt chunk to the first encoder layer and this is the default ZeroPrompt method used in Table 1. -1 means baseline without ZeroPrompt.*

StartLayer	TDT-F	TDT-L	PPC	RTF
0	646ms	4472ms	1.20	0.06447 / 0.07425
4	649ms	4477ms	1.20	0.05779 / 0.06906
6	778ms	4562ms	0.97	0.05444 / 0.06596
8	1099ms	4652ms	0.70	0.05186 / 0.06273
11	1149ms	4815ms	0.34	0.04734 / 0.05858
-1	1279ms	4806ms	-	0.04351 / 0.05063

3.4. Error Analysis

Lastly, we provide error analysis on *[640ms chunk size & 1280ms ZeroPrompt, Aishell-1]* as this configuration achieves the worst PER and the best PPC. We find that errors can be categried into two types:

- **Homophonic tokens**, typically occur at the beginning of prompts. This is reasonable because the current chunk may only contain partial pronunciations of the character, and ZeroPrompt forces the model to emit a complete character based on these partial pronunciations thus leading to homophone errors.
- **Semantically continuous but phonetically mismatched tokens**, typically occur at the end of a **very long** prompt. The trailing part of ZeroPrompt chunk contains no partial pronunciation, therefore the prediction of trailing prompts solely depends on the history context without any acoustic hints, like a Masked LM, this further validate our conjecture that streaming ascoutic encoders are zero-shot Masked LMs.

4. Acknowledgement

This work is supported by National Natural Science Foundation of China (62076144) and Shenzhen Science and Technology Program (WDZC20220816140515001).

5. References

[1] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, ser. ACM International Conference Proceeding Series, W. W. Cohen and A. W. Moore, Eds., vol. 148. ACM, 2006, pp. 369–376. [Online]. Available: https://doi.org/10.1145/1143844.1143891

[2] A. Graves, "Sequence transduction with recurrent neural networks," *CoRR*, vol. abs/1211.3711, 2012. [Online]. Available: http://arxiv.org/abs/1211.3711

[3] L. Dong, S. Xu, and B. Xu, "Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018*. IEEE, 2018, pp. 5884–5888. [Online]. Available: https://doi.org/10.1109/ICASSP.2018.8462506

[4] J. Yu, W. Han, A. Gulati, C. Chiu, B. Li, T. N. Sainath, Y. Wu, and R. Pang, "Dual-mode ASR: unify and improve streaming ASR with full-context modeling," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. [Online]. Available: https://openreview.net/forum?id=Pz_dcqfcKW8

[5] L. Dong, F. Wang, and B. Xu, "Self-attention aligner: A latency-control end-to-end model for ASR using self-attention network and chunk-hopping," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*. IEEE, 2019, pp. 5656–5660. [Online]. Available: https://doi.org/10.1109/ICASSP.2019.8682954

[6] B. Zhang, D. Wu, Z. Yao, X. Wang, F. Yu, C. Yang, L. Guo, Y. Hu, L. Xie, and X. Lei, "Unified streaming and non-streaming two-pass end-to-end model for speech recognition," *CoRR*, vol. abs/2012.05481, 2020. [Online]. Available: https://arxiv.org/abs/2012.05481

[7] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 5998–6008. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[9] Y. Shangguan, R. Prabhavalkar, H. Su, J. Mahadeokar, Y. Shi, J. Zhou, C. Wu, D. Le, O. Kalinli, C. Fuegen, and M. L. Seltzer, "Dissecting user-perceived latency of on-device E2E speech recognition," in *Interspeech 2021, 22nd Annual Conference of the International Speech Communication Association, Brno, Czechia, 30 August - 3 September 2021*, H. Hermansky, H. Cernocký, L. Burget, L. Lamel, O. Scharenborg, and P. Motlícek, Eds. ISCA, 2021, pp. 4553–4557. [Online]. Available: https://doi.org/10.21437/Interspeech.2021-1887

[10] J. Yu, C. Chiu, B. Li, S. Chang, T. N. Sainath, Y. He, A. Narayanan, W. Han, A. Gulati, Y. Wu, and R. Pang, "Fastemit: Low-latency streaming ASR with sequence-level emission regularization," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021, Toronto, ON, Canada, June 6-11, 2021*. IEEE, 2021, pp. 6004–6008. [Online]. Available: https://doi.org/10.1109/ICASSP39728.2021.9413803

[11] Z. Tian, H. Xiang, M. Li, F. Lin, K. Ding, and G. Wan, "Peak-first CTC: reducing the peak latency of CTC models by applying peak-first regularization," *CoRR*, vol. abs/2211.03284, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2211.03284

[12] X. Song, D. Wu, Z. Wu, B. Zhang, Y. Zhang, Z. Peng, W. Li, F. Pan, and C. Zhu, "Trimtail: Low-latency streaming ASR with simple but effective spectrogram-level length penalty," *CoRR*, vol. abs/2211.00522, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2211.00522

[13] D. Povey, H. Hadian, P. Ghahremani, K. Li, and S. Khudanpur, "A time-restricted self-attention layer for ASR," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018*. IEEE, 2018, pp. 5874–5878. [Online]. Available: https://doi.org/10.1109/ICASSP.2018.8462497

[14] C. Wu, Y. Wang, Y. Shi, C.-F. Yeh, and F. Zhang, "Streaming Transformer-Based Acoustic Models Using Self-Attention with Augmented Memory," in *Proc. Interspeech 2020*, 2020, pp. 2132–2136.

[15] K. Deng and P. C. Woodland, "Adaptable end-to-end ASR models using replaceable internal lms and residual softmax," *CoRR*, vol. abs/2302.08579, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2302.08579

[16] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, 2019, pp. 4171–4186. [Online]. Available: https://doi.org/10.18653/v1/n19-1423

[17] Z. Li, H. Miao, K. Deng, G. Cheng, S. Tian, T. Li, and Y. Yan, "Improving streaming end-to-end ASR on transformer-based causal models with encoder states revision strategies," in *Interspeech 2022, 23rd Annual Conference of the International Speech Communication Association, Incheon, Korea, 18-22 September 2022*, H. Ko and J. H. L. Hansen, Eds. ISCA, 2022, pp. 1671–1675. [Online]. Available: https://doi.org/10.21437/Interspeech.2022-707

[18] K. An, H. Zheng, Z. Ou, H. Xiang, K. Ding, and G. Wan, "CUSIDE: chunking, simulating future context and decoding for streaming ASR," in *Interspeech 2022, 23rd Annual Conference of the International Speech Communication Association, Incheon, Korea, 18-22 September 2022*, H. Ko and J. H. L. Hansen, Eds. ISCA, 2022, pp. 2103–2107. [Online]. Available: https://doi.org/10.21437/Interspeech.2022-11214

[19] Y. Chung and J. R. Glass, "Generative pre-training for speech with autoregressive predictive coding," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*. IEEE, 2020, pp. 3497–3501. [Online]. Available: https://doi.org/10.1109/ICASSP40776.2020.9054438

[20] H. Bu, J. Du, X. Na, B. Wu, and H. Zheng, "AISHELL-1: an open-source mandarin speech corpus and a speech recognition baseline," in *20th Conference of the Oriental Chapter of the International Coordinating Committee on Speech Databases and Speech I/O Systems and Assessment, O-COCOSDA 2017, Seoul, South Korea, November 1-3, 2017*. IEEE, 2017, pp. 1–5. [Online]. Available: https://doi.org/10.1109/ICSDA.2017.8384449

[21] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015*. IEEE, 2015, pp. 5206–5210. [Online]. Available: https://doi.org/10.1109/ICASSP.2015.7178964

[22] B. Zhang, D. Wu, Z. Peng, X. Song, Z. Yao, H. Lv, L. Xie, C. Yang, F. Pan, and J. Niu, "Wenet 2.0: More productive end-to-end speech recognition toolkit," in *Interspeech 2022, 23rd Annual Conference of the International Speech Communication Association, Incheon, Korea, 18-22 September 2022*, H. Ko and J. H. L. Hansen, Eds. ISCA, 2022, pp. 1661–1665. [Online]. Available: https://doi.org/10.21437/Interspeech.2022-483