End-to-End Streaming ASR for Edge devices

Lei Xie

Audio, Speech and Language Processing Group (ASLP@NPU),
School of Computer Science, Northwestern Polytechnical University, Xi'an, China

Outline

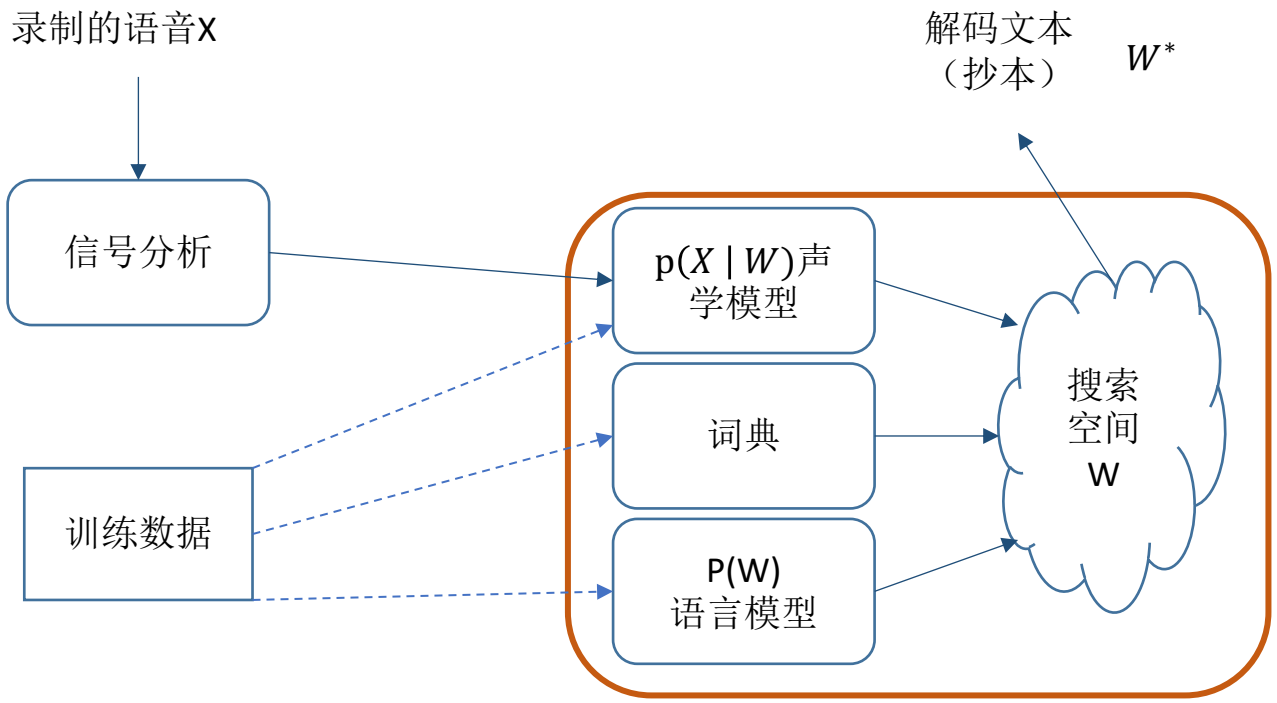
- End-to-end ASR
- RNN-T
- Cascade RNN-T
- U2
- WeNet

Outline

- **End-to-end ASR**
- RNN-T
- Cascade RNN-T
- U2
- WeNet

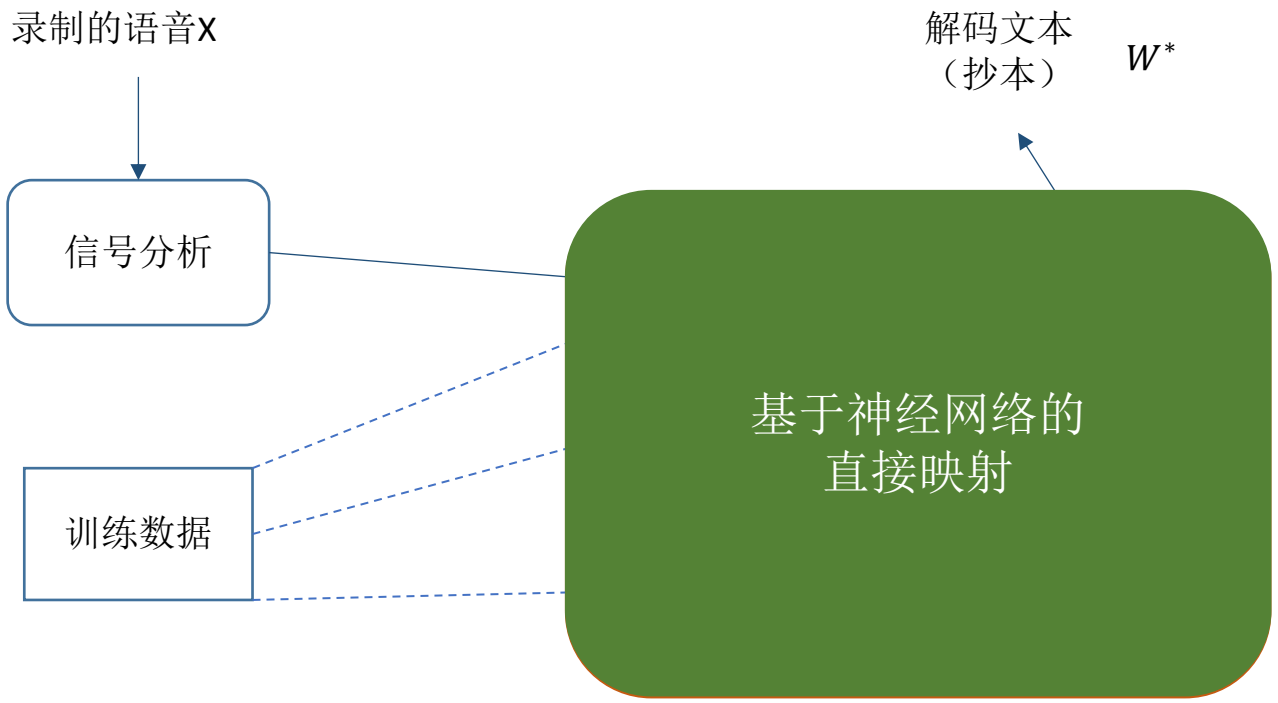
Traditional ASR

- Acoustic Model + Lexicon + Language Model → Decoding Graph
- Many building blocks with complicated training pipeline
  - Each optimized with own target → not directly leading to final lower error rate
- Pronunciation lexicon with phonetic/linguistic knowledge
- Forced alignment
- Painful engineering



E2E ASR

- One neural network for direct acoustic feature to grapheme mapping
- Simplified framework and training pipeline
- Training target directly associated with error rate
- Pronunciation lexicon can be discarded
- Forced alignment no longer required



- Two typical E2E framework
  - Attention-based Encoder Decoder (AED)
  - Transducer (T)

E2E ASR: Trends

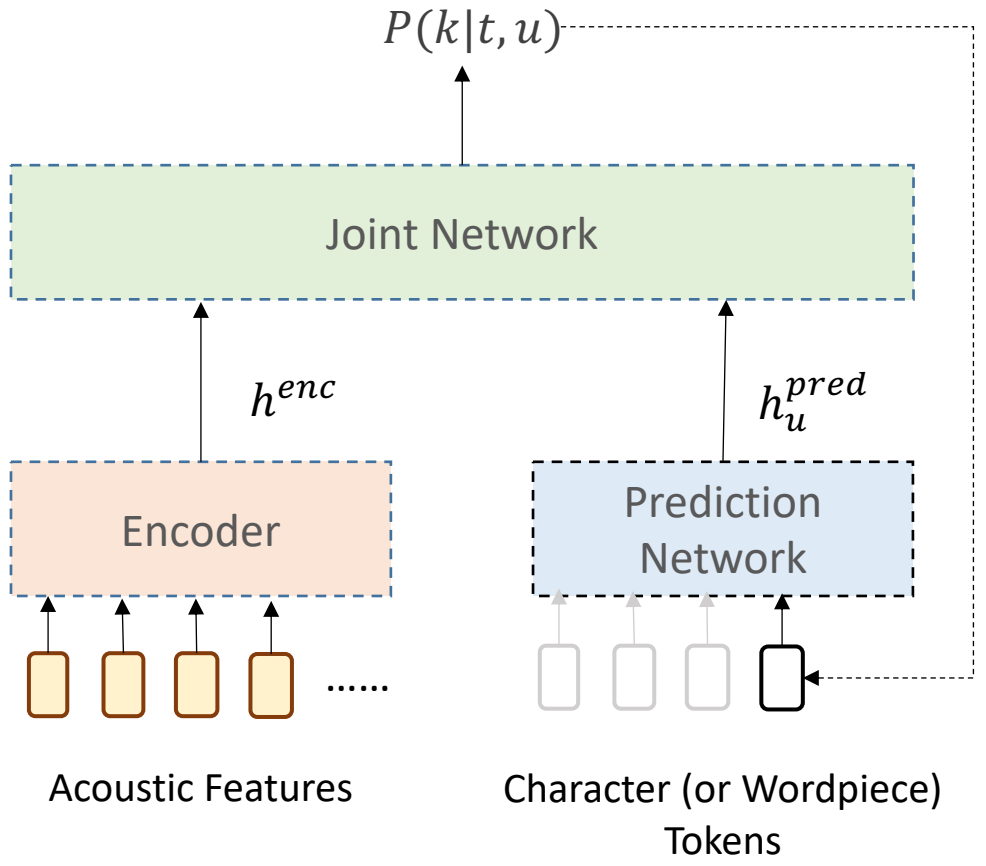
- Effective language modeling
- Training tricks
- Streaming
- Multi-lingual
- Low resource deployment, e.g., on edge/IoT devices
- Pre-training, unsupervised and weak-supervised learning
- End-to-end ideas
  - Front-end + back-end
  - Speech translation
  - ASR+NLU
  - ASR+ other tasks
  - …

Outline

- End-to-end ASR
- **RNN-T**
- Cascade RNN-T
- U2
- WeNet

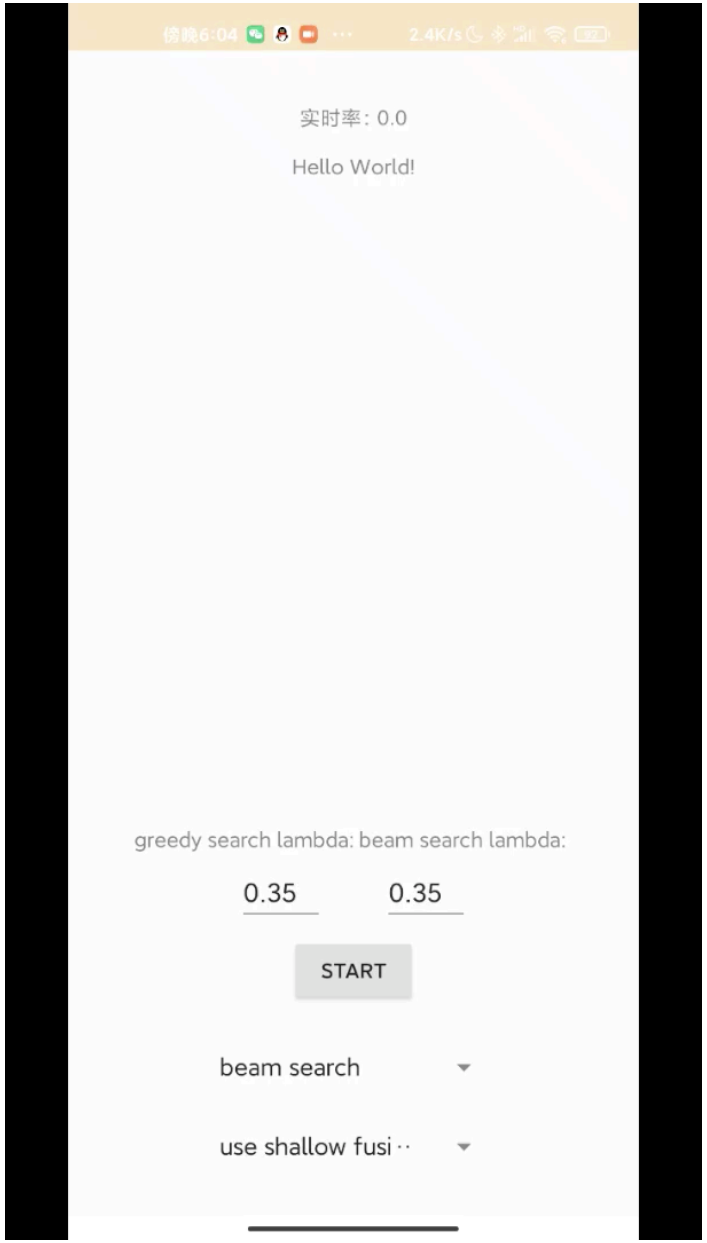
RNN Transducer

- Superior performance in streaming on-device speech recognition with high-accuracy and low-latency
- Variants: Transformer Transducer (T-T), conformer transducer (C-T), …
  - **Encoder network** converts the speech features into a high-dimensional representation h^{enc}
  - **Prediction network** generates a higher-dimensional representation h_u^{pred} of last label y_{u-1}
  - **Joint network** uses the h^{enc} and h_u^{pred} to predict the probability $P(k|t,u)$ of the next label
- Character and Wordpiece are common modeling units



Edge Deployment

- Quantization and deploy on phone using TF Lite



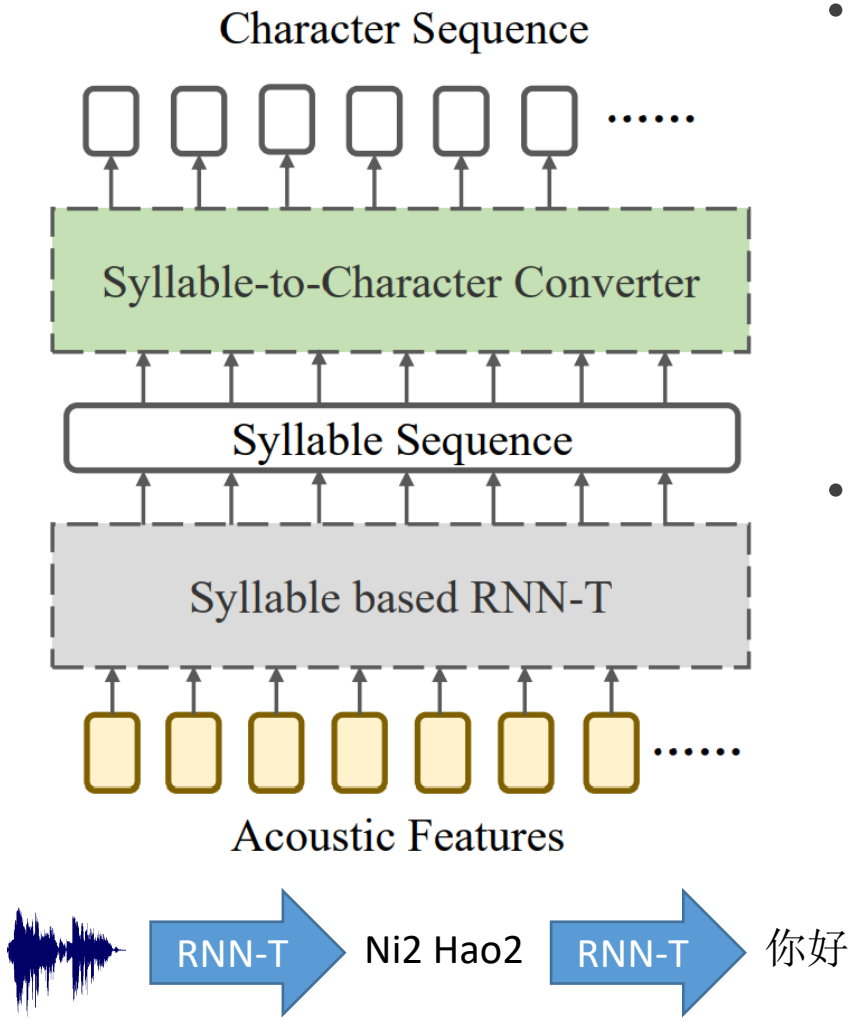
Outline

- End-to-end ASR
- RNN-T
- **Cascade RNN-T**
- U2
- WeNet

Introduction

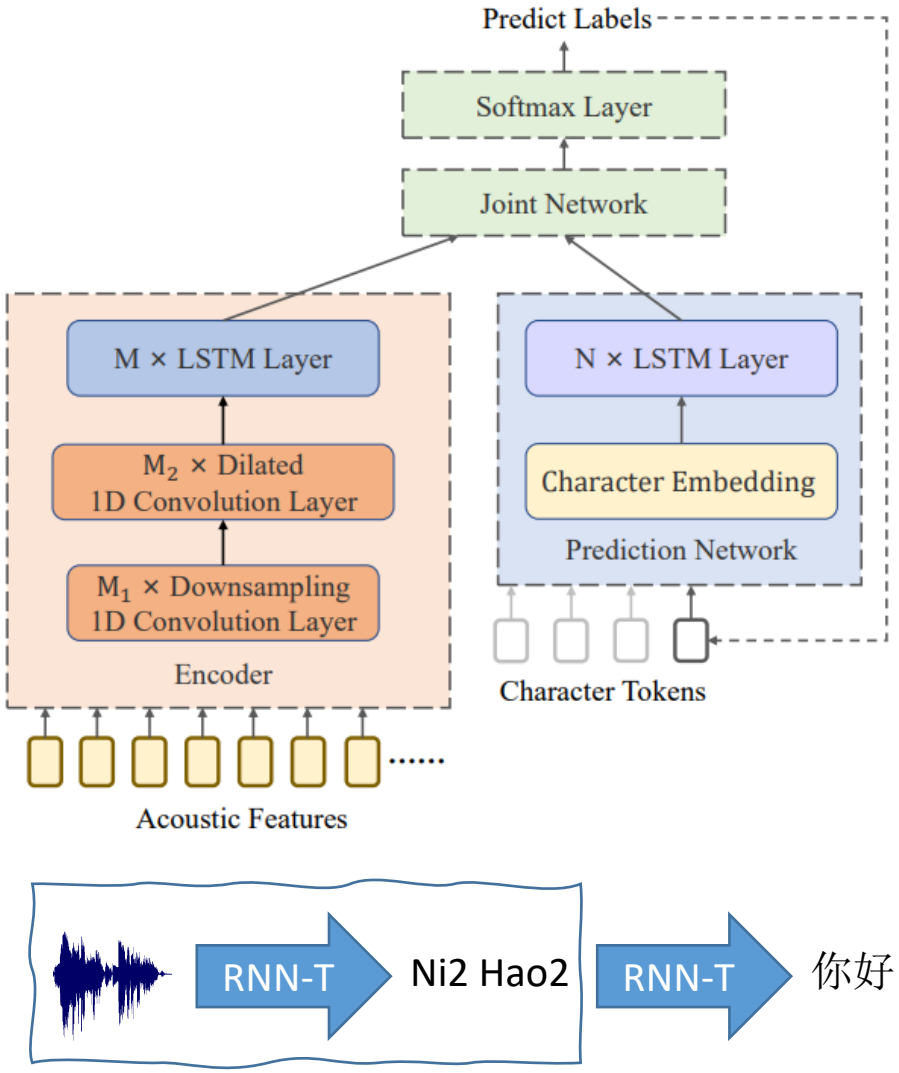
- Typical RNN-T
  - A prediction network to enhance language information
  - But language modeling ability is limited because it still needs **paired speech-text data to train**
- Plenty of effort on improving the performance by introducing additional language information
  - Language model fusion strategy, e.g., shallow, deep and component fusion
  - Data augmentation with text-to-speech utterances
  - 2-pass decoding, e.g., RNN-T+LAS
- Our approach: a novel **cascade RNN-T** approach for streaming on-device Mandarin speech recognition
  - Chinese is a character language and each character comes as a tonal syllable

Framework



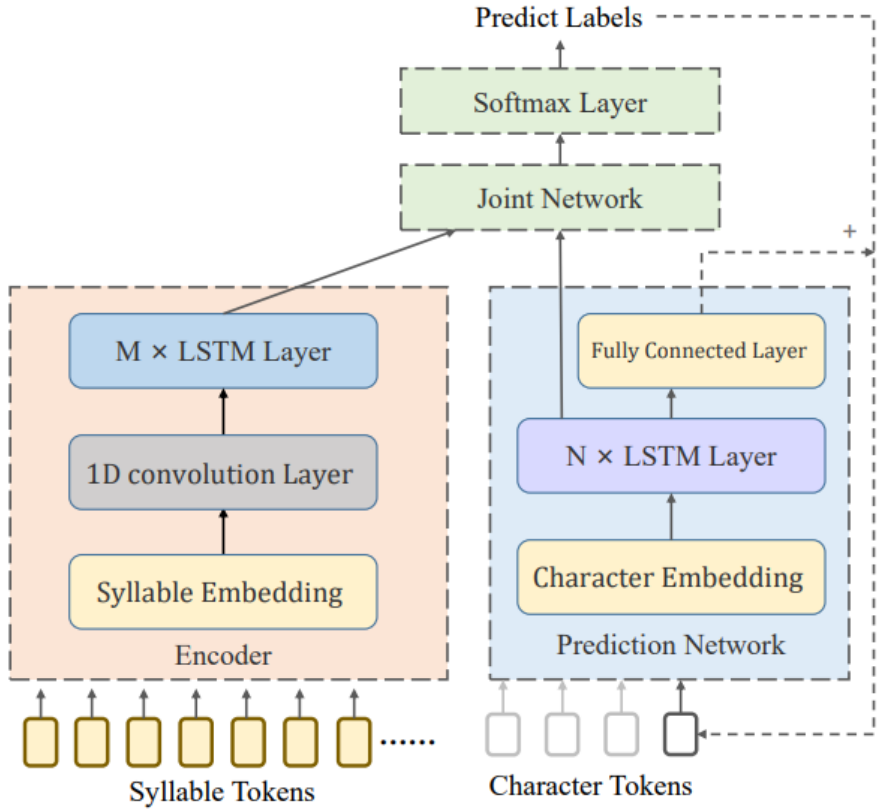
- We cascade two RNN transducers to strengthen the language modeling ability
  - **1st RNN-T**: transforms acoustic input into a syllable sequence
  - **2nd RNN-T**: converts the syllable sequence into the final character sequence
- Advantages:
  - A **rich text repository** can be easily used to strengthen the language modeling ability
  - The **OOV issue** does not exist by the introduction of RNN-T based syllable-to-character (S2C) converter
  - **Streaming ability** has been maintained as the use of the transducer framework

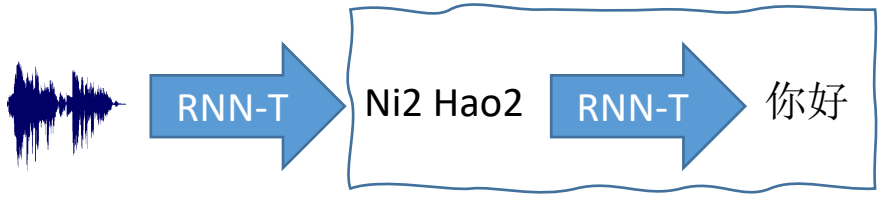
Syllable RNN-T



- Follow the general RNN-T structure
- We add M_1 layers of 1-D convolution with a stride size greater than 1 for **down-sampling**
  - Effectively reduce the resource consumption during the training process and decoding stage
- The M_2 layers of dilated 1-D convolution followed can **capture local context information** more effectively at a small cost of a short latency

S2C Converter





- **Modeling Context**: Add convolution layer before the encoder to get information of current token

$$\mathbf{h}^{conv} = \text{Convolution}(\mathbf{y}^s)$$

$$\mathbf{h}^{enc} = \text{Encoder}(\mathbf{h}^{conv}).$$

- **Self Shallow Fusion**: Add a fully-connected layer after the prediction network to enhance the language modeling capacity

$$\hat{P}(\mathbf{y}|\mathbf{y}^s) = \log P(\mathbf{y}|\mathbf{y}^s) + \lambda \log \text{FC}(\mathbf{h}^{pred}).$$

$$\hat{L} = L_{\text{RNN-T}} + L_{ce}$$

- **Text Augmentation**: Randomly replace input syllable sequence like SpecAugment
- **Syllable Correction**: First decode the training set using the syllable-level RNN-T to syllable sequences; then we use these syllable sequences with their corresponding correct character sequences to build additional correction text and mix these data with normal text data

Experiment Setup

- Training dataset:
  - Public 1000-hour AISHELL-2 corpus
  - Internal 7500-hour corpus
  - 2GB Chinese text data crawled from Internet
- Testing dataset:
  - TA1: AISHELL-1 test set
  - TA2: AISHELL-2 test set
  - VI: Voice input test set with 3.4 hours speech (3,063 sentences)
  - VA: Voice assistant test set with 3.9 hours speech (5,000 speech commands)
- Model structure
  - 70-dimensional FBANK (25ms window and 10ms shift)
  - 5139 characters and 1733 tonal syllables
  - 1st RNN-T:
    - Encoder: 6 convolution and 5 LSTM layers
    - Prediction: embedding + 2 LSTM layers
  - 2nd S2C RNN-T:
    - Encoder: syllable embedding + 2 LSTM layers
    - Prediction: syllable embedding + 1 LSTM layers

Exp on AISHELL-2

- The performance of the cascade RNN-T (E0) is worse than character-level RNN-T
- Syllable error rate on TA2 for the syllable-level RNN-T in the cascade approach (E0) is 12.8% which is lower than the CER of character RNN-T (B0) on the same test set

Table 1. Comparison of character-level RNN-T and Cascade RNN-T on test sets in CER.

Exp ID	Model	CER (%)		
		TA1	TA2	VI
B0	RNN-T Character	6.71	16.27	21.02
B1	+ Beam search	6.55	15.79	20.44
B2	+ Shallow fusion	**6.11**	**15.50**	**20.15**
E0	Cascade RNN-T	17.4	22.08	24.52

Exp on Tricks

- We improve the S2C RNN-T: syllable correction (E4) can bring further performance gain and we manage to surpass the character-based RNN-T on all test sets
- All tricks can effectively reduce the substitution errors, confirming that our proposed cascade RNN-T can greatly improve the language modeling ability

Table 2. Comparison of the results of various tricks to improve the performance of cascade RNN-T on test sets in CER.

Exp ID	Model	CER (%)		
		TA1	TA2	VI
B2	RNN-T Character + Beam search + Shallow fusion	6.11	15.50	20.15
E0	Cascade RNN-T	17.4	22.08	24.52
E1	+ Convolution layer	6.66	15.59	18.27
E2	+ Text augmentation	6.4	15.24	17.96
E3	+ Self shallow fusion	5.89	14.93	17.78
E4	+ Syllable correction	**5.72**	**14.85**	**17.60**

Table 3. Comparison of deletion, insertion and substitute errors of each model on the TA2 test set.

Exp ID	Model	CER (%) (D/I/S)
		TA2
B2	RNN-T Character + Beam search + Shallow fusion	15.50 (0.71/0.26/14.53)
E0	Cascade RNN-T	22.08(0.84/0.21/21.03)
E1	+ Convolution layer	15.59(0.76/0.17/14.66)
E2	+ Text augmentation	15.24(0.75/0.17/14.32)
E3	+ Self shallow fusion	14.93(0.75/0.17/14.01)
E4	+ Syllable correction	14.85(0.75/0.16/13.94)

Exp on 7500H

- Comparing E9 with B5, our proposed cascade RNN-T can better enhance the language modeling ability of RNN-T than the character RNN-T with shallow fusion
- From A0 and A1, we can observe that RNN-T based S2C converter has obvious advantages over LSTM which is also a streaming structure
- Our model has significantly better performance on named entities

Table 4. Comparison of character-level RNN-T and Cascade RNN-T on test sets in CER, training with 7500-hour corpus.

Exp ID	Model	CER (%)			
		TA1	TA2	VI	VA
B3	RNN-T Character	5.15	10.57	10.21	33.63
B4	+ Beam search	4.99	10.08	9.69	32.88
B5	+ Shallow fusion	4.85	9.96	9.5	32.71
E5	Cascade RNN-T	14.56	18.51	18.16	38.09
E6	+ Convolution layer	6.32	11.33	10.62	31.76
E7	+ Text augmentation	5.53	10.42	9.59	29.53
E8	+ Self shallow fusion	4.62	9.33	8.82	28.21
E9	+ Syllable correction	**4.57**	**9.16**	**8.65**	**28.07**
A0	RNN-T Syllable + LSTM S2C	11.48	15.02	14.71	36.33
A1	RNN-T Syllable + BLSTM S2C	4.98	9.59	9.05	32.1

Para. vs Acc. vs Latency

Table 5. Comparison of parameters, latency and performance for different models.

Exp ID	Model	Param.(M)/Latency	CER (%)			
			TA1	TA2	VI	VA
B5	RNN-T Character + Beam search + Shallow fusion	93M/280ms	4.85	9.96	9.5	32.71
E5	Cascade RNN-T	88.5M/280ms	14.56	18.51	18.16	38.09
E6	+ Convolution layer	92.5M/300ms	6.32	11.33	10.62	31.76
E7	+ Text augmentation	92.5M/300ms	5.53	10.42	9.59	29.53
E8	+ Self shallow fusion	95.5M/300ms	4.62	9.33	8.82	28.21
E9	+ Syllable correction	95.5M/300ms	**4.57**	**9.16**	**8.65**	**28.07**

- Comparing E4 with B2, we can see that the proposed cascade RNN-T achieves superior performance over the character-based RNN with similar levels of model parameters and recognition latency.

Xiong Wang, Zhuoyuan Yao, Xian Shi, Lei Xie, Cascade RNN-Transducer: Syllable Based Streaming On-device Mandarin Speech Recognition with a Syllable-to-Character Converter, IEEE SLT2021, https://arxiv.org/abs/2011.08469

Outline

- End-to-end ASR
- RNN-T
- Cascade RNN-T
- **U2**
- WeNet

Motivation

- **U2: Unified streaming and non-streaming E2E model**
  - Reduce develop/train/deploy efforts
  - Joint CTC/AED architecture
- Training
  - Joint training
  - Converge faster
- How to do streaming?
  - Shared encoder only sees **limited right context**
  - CTC decoder as **first pass** partial result
  - Attention decoder as the **second pass** final result

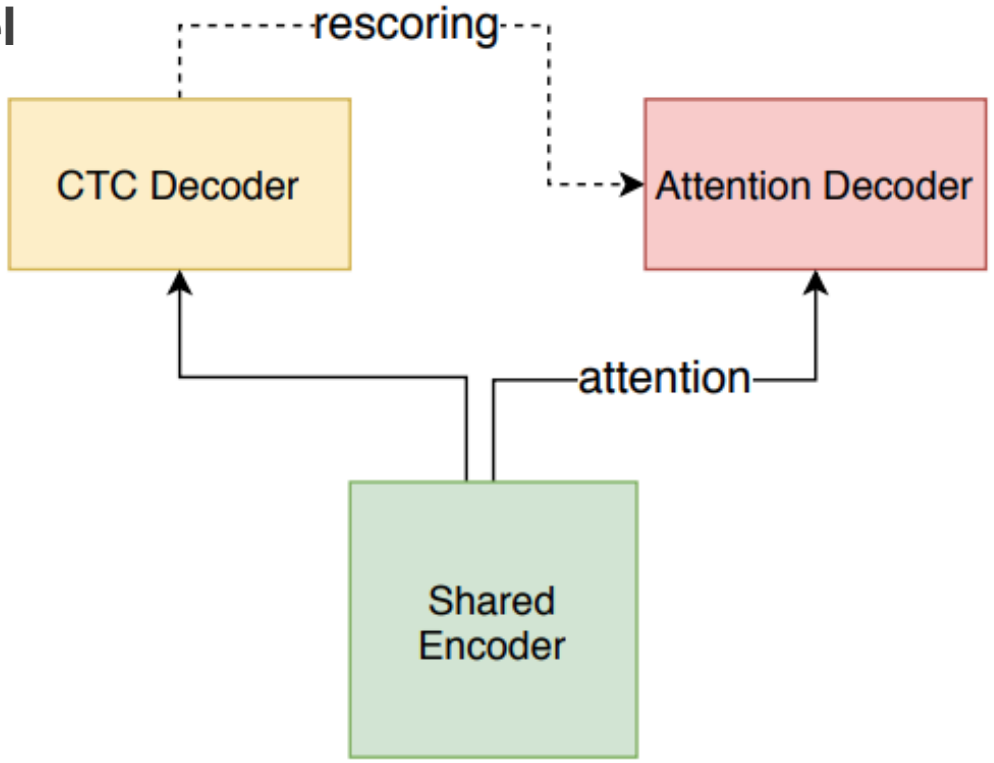


Figure 1: *Two pass CTC and AED joint architecture*

How to unify?

- **Dynamic chunk training**
  - Chunk attention gives better accuracy
  - Dynamic chunk attention
    - Full chunk: non-streaming case
    - Limited chunk: streaming case
    - Vary chunk from 1~full in training dynamically
  - The model **learns both streaming and non-streaming with arbitrary chunk size**
- **Decoding**
  - Large chunk shows better accuracy with more latency
  - Easy to balance accuracy and latency by tuning chunk size in decoding

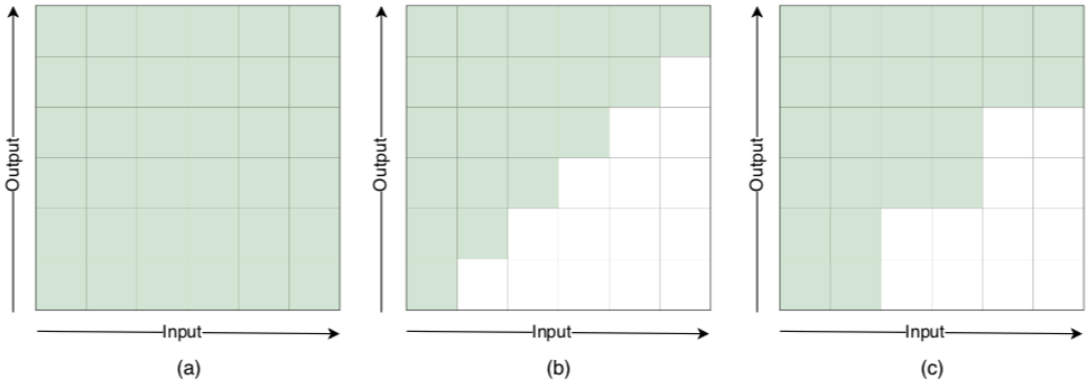


Figure 2: *Full attention, Left attention, Chunk Attention*

Experiments

- Dataset: AISHELL-1
- Baseline: static chunk trained model, static chunk inference
- Decoding methods: CTC only, Attention only, first CTC and second Attention
- Conclusions:
  - **Attention rescoring achieves the best accuracy**
  - **One unified model shows comparable performance to static chunk trained 5 (full/16/8/4/1) models**

Table 2: *Dynamic vs static chunk training*

training method	decoding mode	decoding chunk size				
		full	16	8	4	1
static chunk training, static chunk inference	attention decoder	5.35	5.95	5.99	6.15	6.36
	ctc prefix beam search	5.18	6.30	6.50	6.69	6.73
	attention rescoring	**4.86**	5.55	5.78	6.06	6.02
dynamic chunk training, static chunk inference	attention decoder	5.27	5.51	5.67	5.72	**5.88**
	ctc prefix beam search	5.49	6.08	6.41	6.64	7.58
	attention rescoring	4.90	**5.33**	**5.52**	**5.71**	6.23

Table 3: *Comparison to other streaming solutions*

model	params(M)	latency(ms)	CER
Sync-Transformer[24]	/	400	8.91
SCAMA[25]	43	600	7.39
MMA[14]	/	640	6.60
U2	47	640+Δ	**5.33**

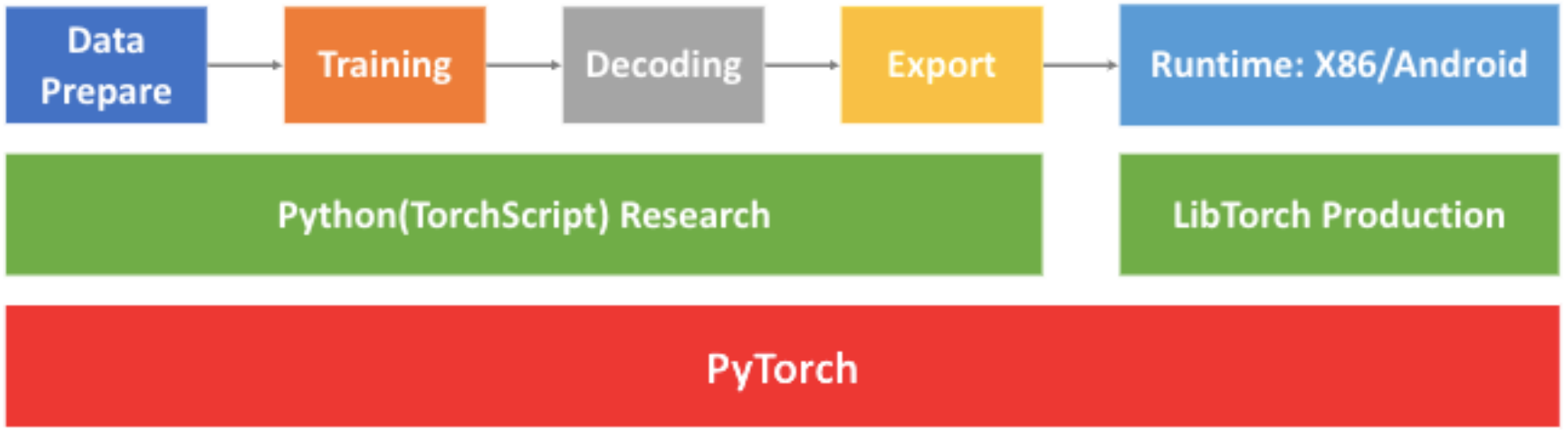
Outline

- End-to-end ASR
- RNN-T
- Cascade RNN-T
- U2
- **WeNet**

Motivation

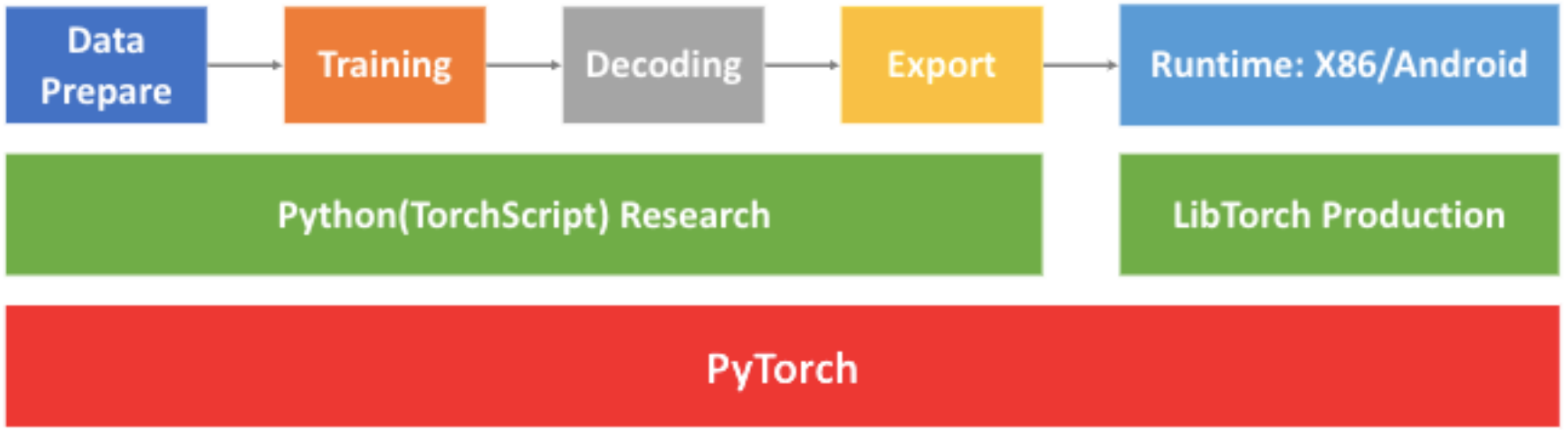
- **WeNet**: Production First and Production Ready End-to-End Speech Recognition Toolkit
  - Close the gap between research and production E2E speech recognition models
  - Reduce the effort of productionizing E2E models
- Streaming → **U2**
  - Many applications require streaming and low latency speech recognition
- Unified → **U2**
  - Reduce the develop/train/deploy efforts
- Production → **WeNet Product-oriented System Design**
  - Converting research model to production model
  - E2E inference workflow (such as beam search) in runtime
  - Deploy the model to different platforms (cloud/on-device)
- **U2 is adopted as the basic model architecture in WeNet.**

System Design



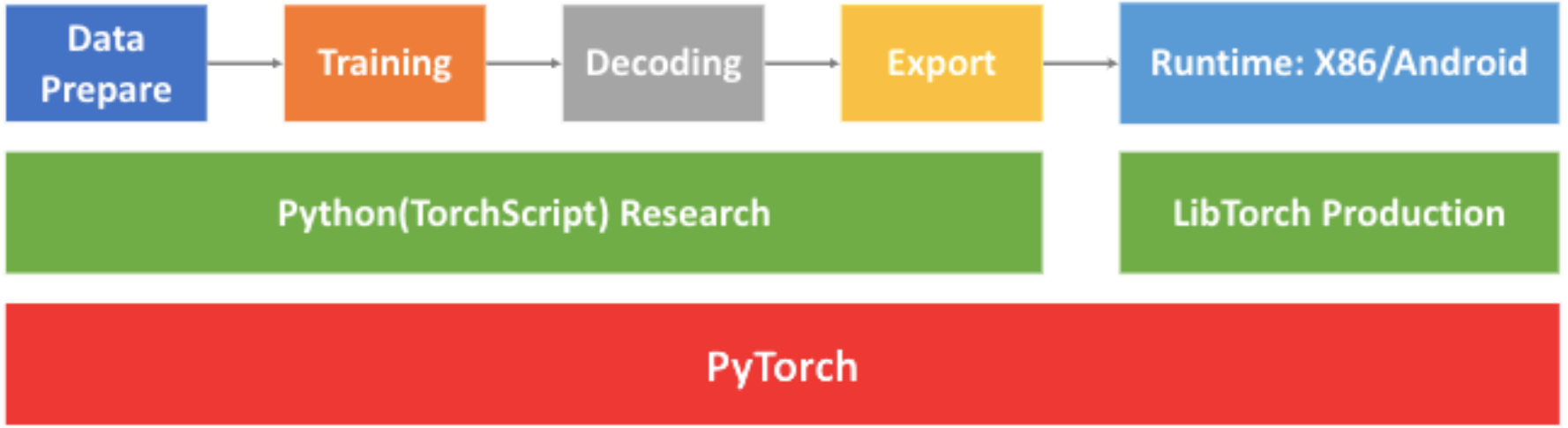
- **First Stack: WeNet is fully based on PyTorch and it's ecosystem**
  - TorchScript → developing model
  - Torchaudio → on-the-fly feature extraction
  - DistributedDataParallel → distributed training
  - Torch JIT (Just In Time) → model exportation
  - PyTorch quantization → model quantization
  - LibTorch → production runtime

System Design



- **Second Stack**
  - Python (TorchScript) Research: Ensure model could be correctly exported
  - LibTorch Production: Support various hardwares and platforms like CPU, GPU (CUDA) Linux, Android and iOS.

System Design



- **Third Stack: research → production pipeline**
  - Research pipeline
  - Runtime: X86 and Android runtime are prebuilt in WeNet; many product metrics like accuracy/latency/RTF are benchmarked.

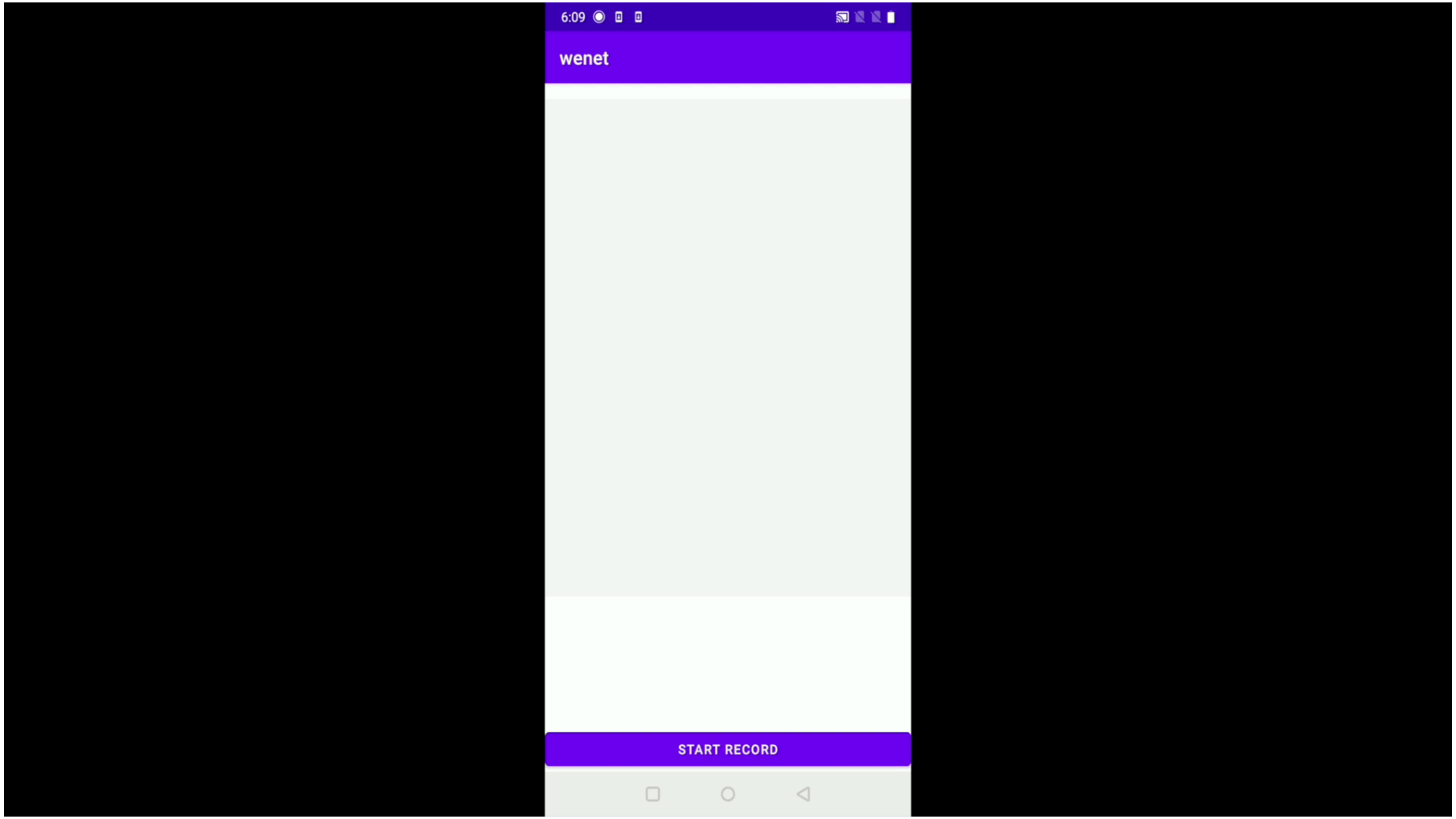
Benchmark

- **Model:** 32M parameter U2 (Transformer) model on Aishell-1
- **Platforms**
  - X86: Intel (R) Xeon (R) CPU E5-2620 v4 @ 2.10GHz, 16G memory
  - Android: Mi10, Qualcomm Snapdragon 865 4 cores, 8 GB memory
  - All tested with **single thread**

Table 3: *RTF benchmark*

model/decoding_chunk	full	16	8	4
server (x86) float	0.079	0.095	0.128	0.186
server (x86) int8	0.072	0.081	0.098	0.134
on-device (Android) float	0.164	0.251	0.350	0.505
on-device (Android) int8	0.082	0.114	0.130	0.201

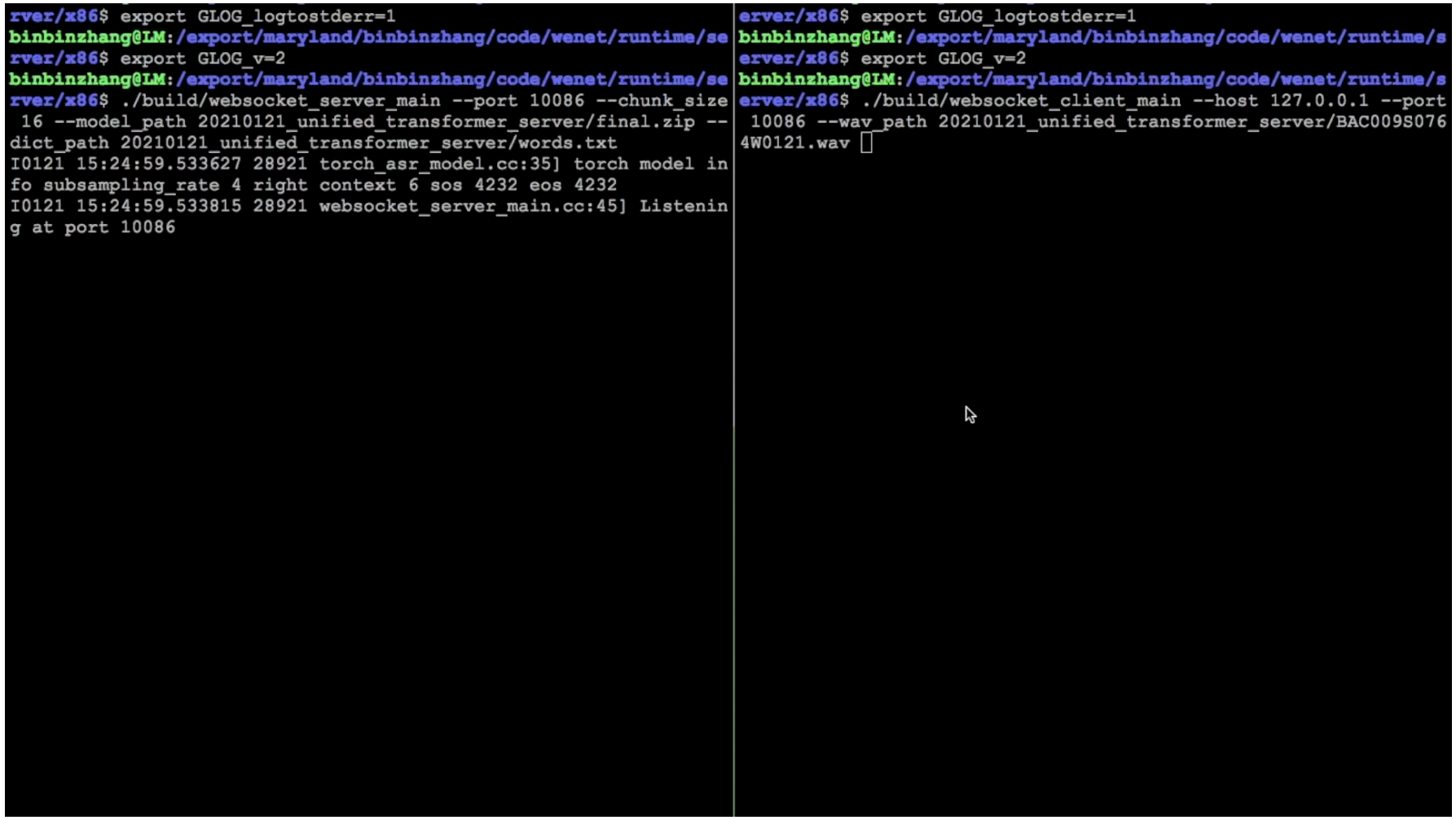
Runtime Demo



On-Device Streaming E2E

https://github.com/mobvoi/wenet/tree/main/runtime/device/android/wenet

Runtime Demo



Cloud Streaming E2E (Websocket Server/Client Interaction)

https://github.com/mobvoi/wenet/tree/main/runtime/device/android/wenet

WeNet Open Source

https://github.com/mobvoi/wenet

THANK YOU

Follow us thru Wechat