Contextual Speech Recognition using Seq2Seq Models

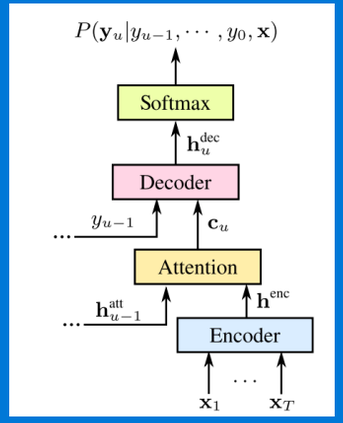
Intern Project Summary

Speaker: **Zhehuai (Tom) Chen**

Mentor: **Mahaveer Jain**

Joint work with **Mahaveer Jain, Yongqiang Wang, Duc Le,**

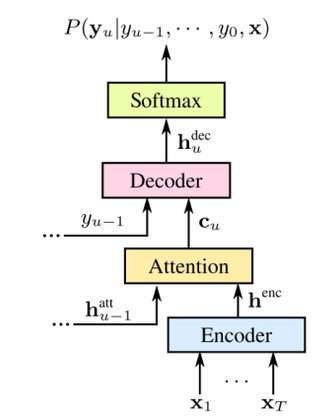
Michael L. Seltzer, Christian Fuegen



SJTU SPEECH LAB
上海交通大學智能語音實驗室

Intern Projects

- **PySpeech**
  - E2E Speech Processing
  - Distributed Training
  - Contextual Speech Recognition (Assistant)
    - Shallow Fusion Framework (paper 1)
    - Deep Context Framework (paper 2)



Hey portal, call **@name**

- **Ninja**
  - RNNLM Lattice Rescoring
  - Lattice Processing

Intern Projects

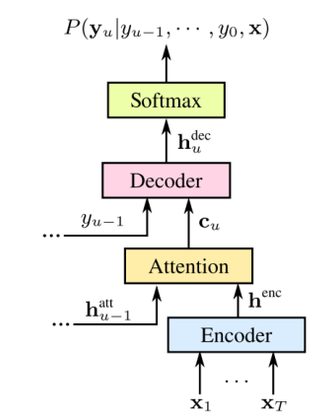
RNNLM	Lattice	Rescore
kaldi	**kaldi**	lattice prune (kaldi)
caffe2	ninja	**arcbeam (ninja)**

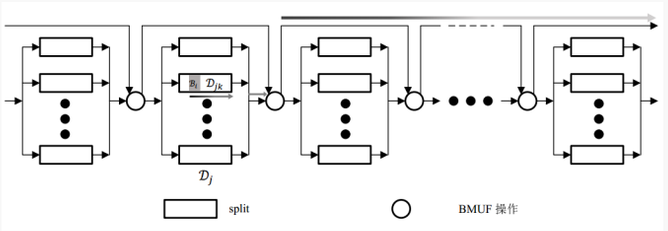
1. Support multiple lattice/algorithm/toolkit
2. Consistent improvement

- **Ninja**
- RNNLM Lattice Rescoring
- Lattice Processing

Outline

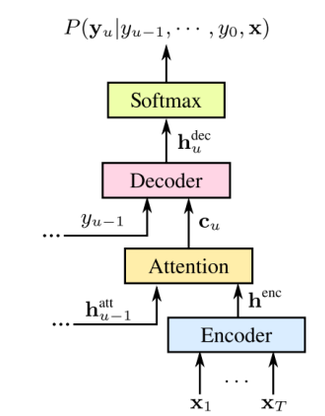
- **PySpeech**
  - E2E Speech Processing
  - Distributed Training
  - Contextual Speech Recognition (Assistant)
    - Shallow Fusion Framework (paper 1)
    - Deep Context Framework (paper 2)

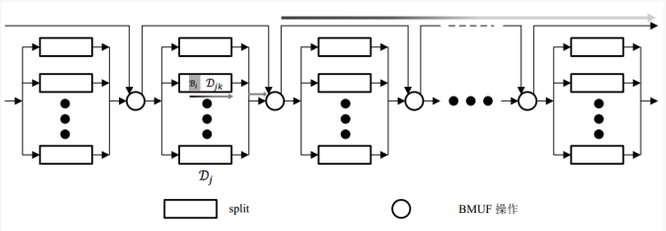




Outline

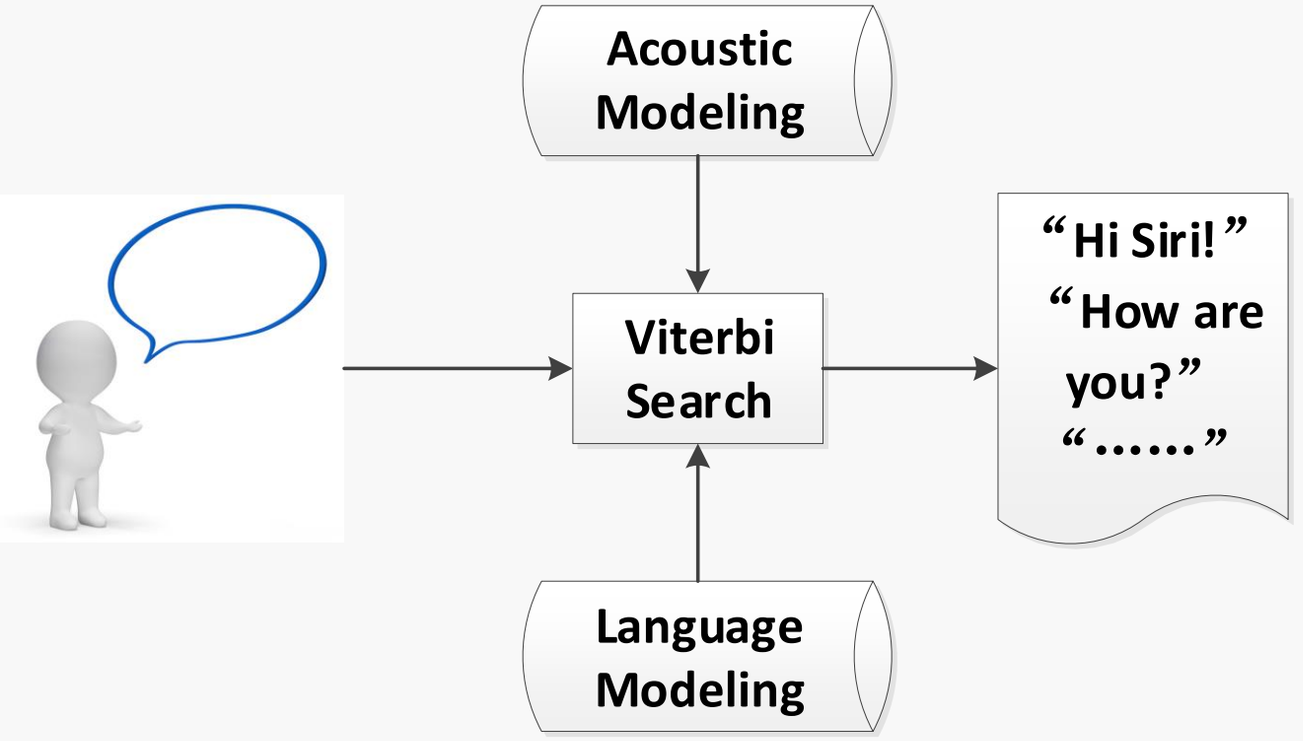
- **PySpeech**
  - E2E Speech Processing
  - Distributed Training
  - Contextual Speech Recognition (Assistant)
    - Shallow Fusion Framework (paper 1)
    - Deep Context Framework (paper 2)

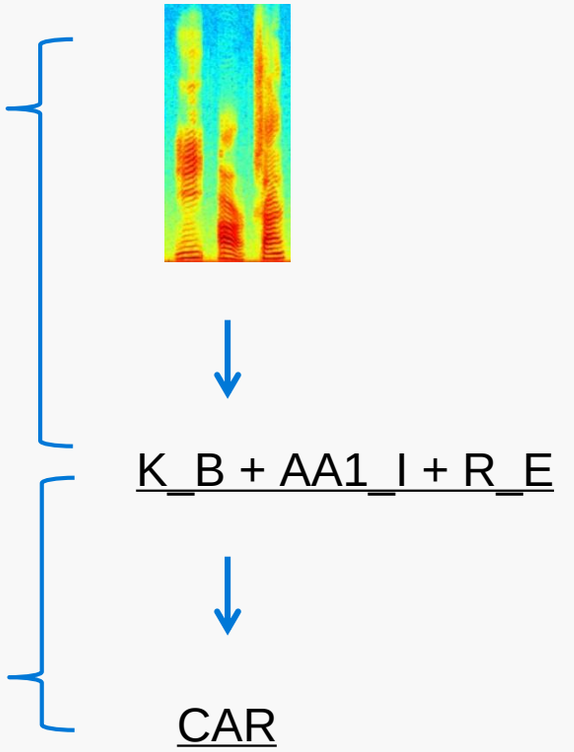




E2E Speech Processing

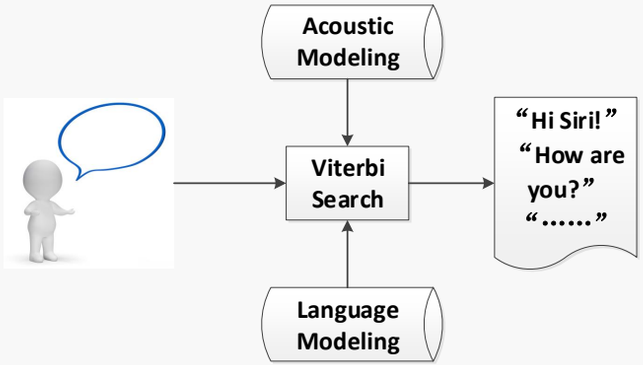
- Automatic Speech Recognition



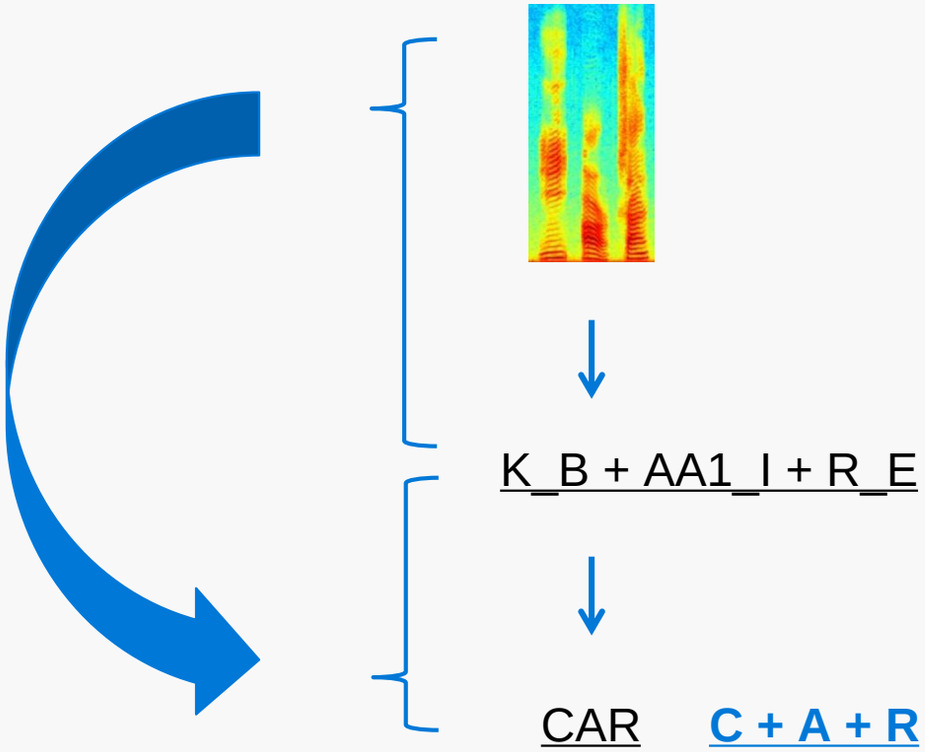


E2E Speech Processing

- ASR → End-to-end Speech Recognition

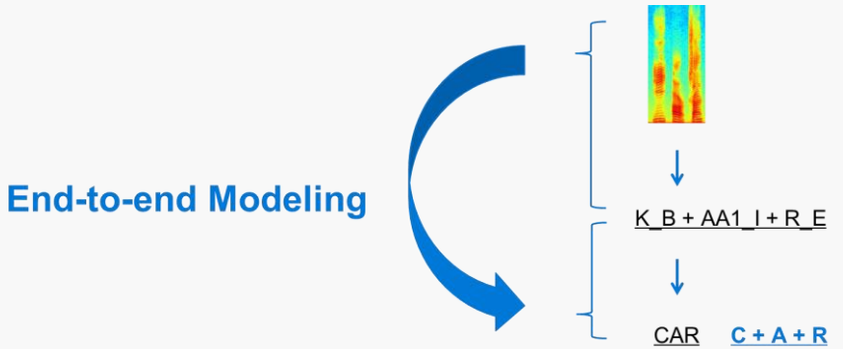


End-to-end Modeling



E2E Speech Processing

- Motivation
  - Only need transcription & audios
  - Removal of seed models, alignment & LM, etc.



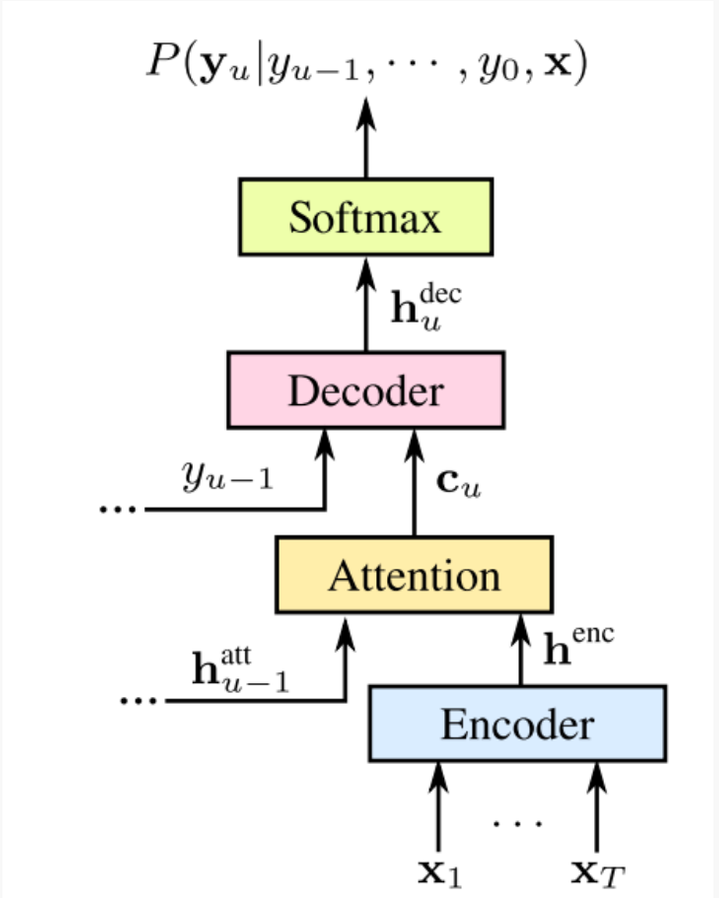
E2E Speech Processing

- Attention based Encoder-decoder

$$P(\mathbf{l}|\mathbf{x}) = \prod_i P(l_i|\mathbf{x}, \mathbf{l}_{1:i-1}) \quad (1)$$

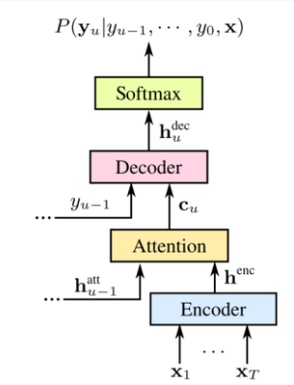
$$\mathbf{h} = \text{Encoder}(\mathbf{x}) \quad (2)$$

$$P(l_i|\mathbf{x}, \mathbf{l}_{1:i-1}) = \text{AttentionDecoder}(\mathbf{h}, \boldsymbol{s}_{i-1}) \quad (3)$$



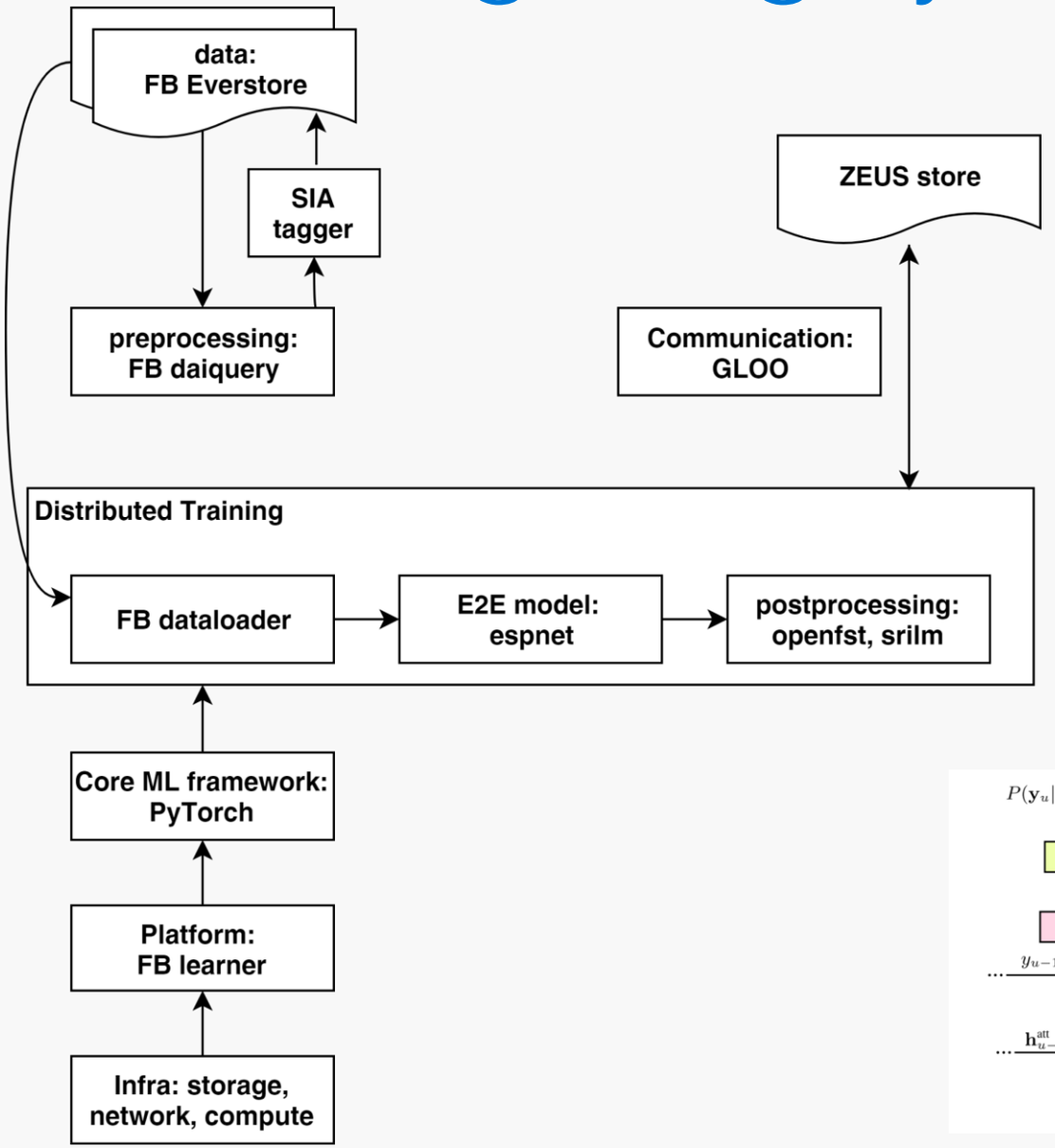
PySpeech: E2E Speech Processing using Pytorch

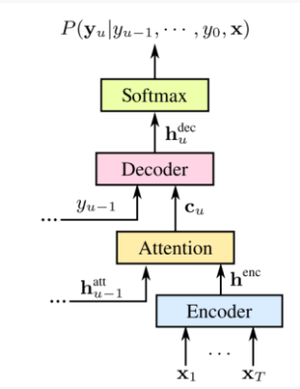
- Motivation
  - Research friendly, especially E2E ☆
  - Efficiency
    - Leverage FB infra
    - Deployment ready



PySpeech: E2E Speech Processing using Pytorch

- PySpeech Design
  - FB learner flow, everstore, ZEUS
  - Train: FB data loader, Espnet
  - Infer: beam search, openfst
  - Preprocess: SIA, daiquery

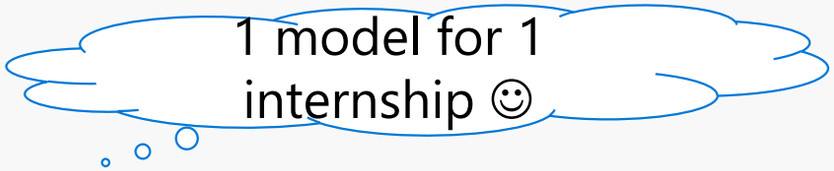




PySpeech: E2E Speech Processing using Pytorch

- Current word error rate (WER)

Data	PySpeech	NN-HMM
Aloha	4.9	5.6
Video	20.8	15+

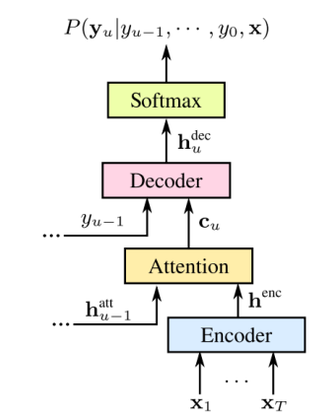


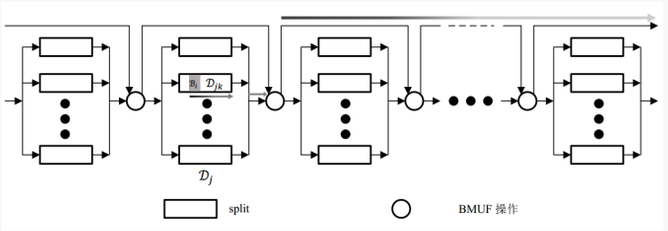
- 1 GPU Speed: 10 days for 1 epoch of 8000 hours data

procedure	Forward	Backward	Data loader	Others
Time taken (%)	31%	65%	2%	2%

Outline

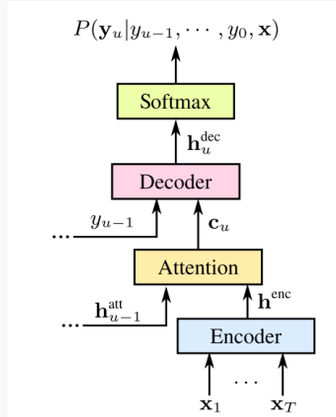
- **PySpeech**
  - E2E Speech Processing
  - Distributed Training
  - Contextual Speech Recognition (Assistant)
    - Shallow Fusion Framework (paper 1)
    - Deep Context Framework (paper 2)

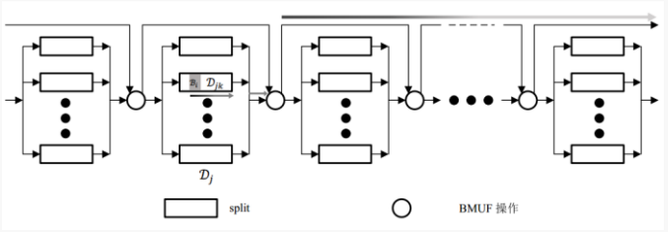




Distributed Training

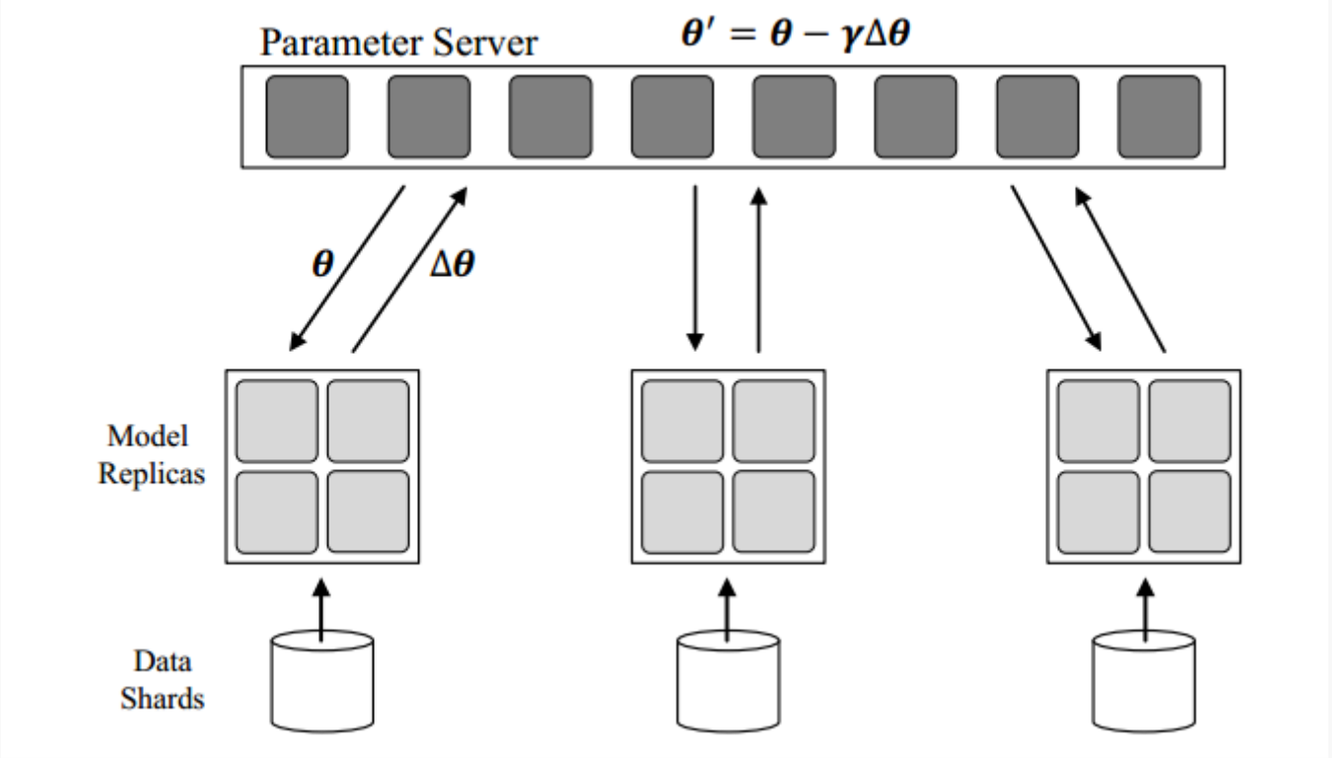
- Introduction
  - Sequence-to-sequence model
  - SSGD in Pytorch
- Block Momentum
  - Algorithm
  - BMUF in Pytorch
- Experiments





Synchronous SGD (SSGD or DPM)

- Scatter the data to each worker
- Forward and backward propagation (each worker)
- Gather the gradients
  - Synchronization
- Update all parameters

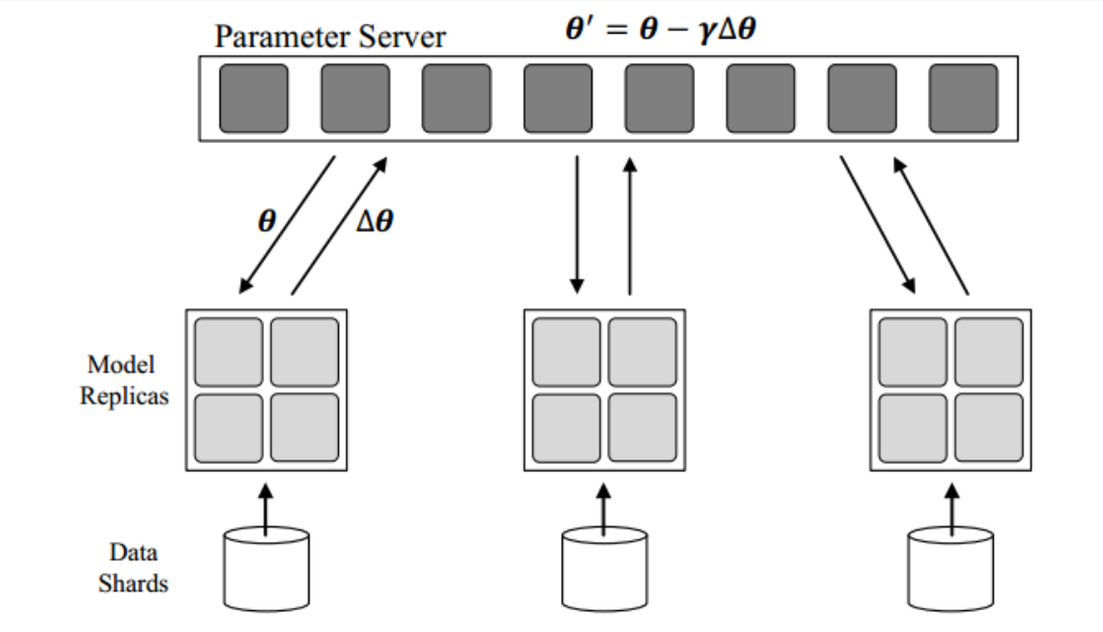


SSGD in Pytorch

- Speed problem

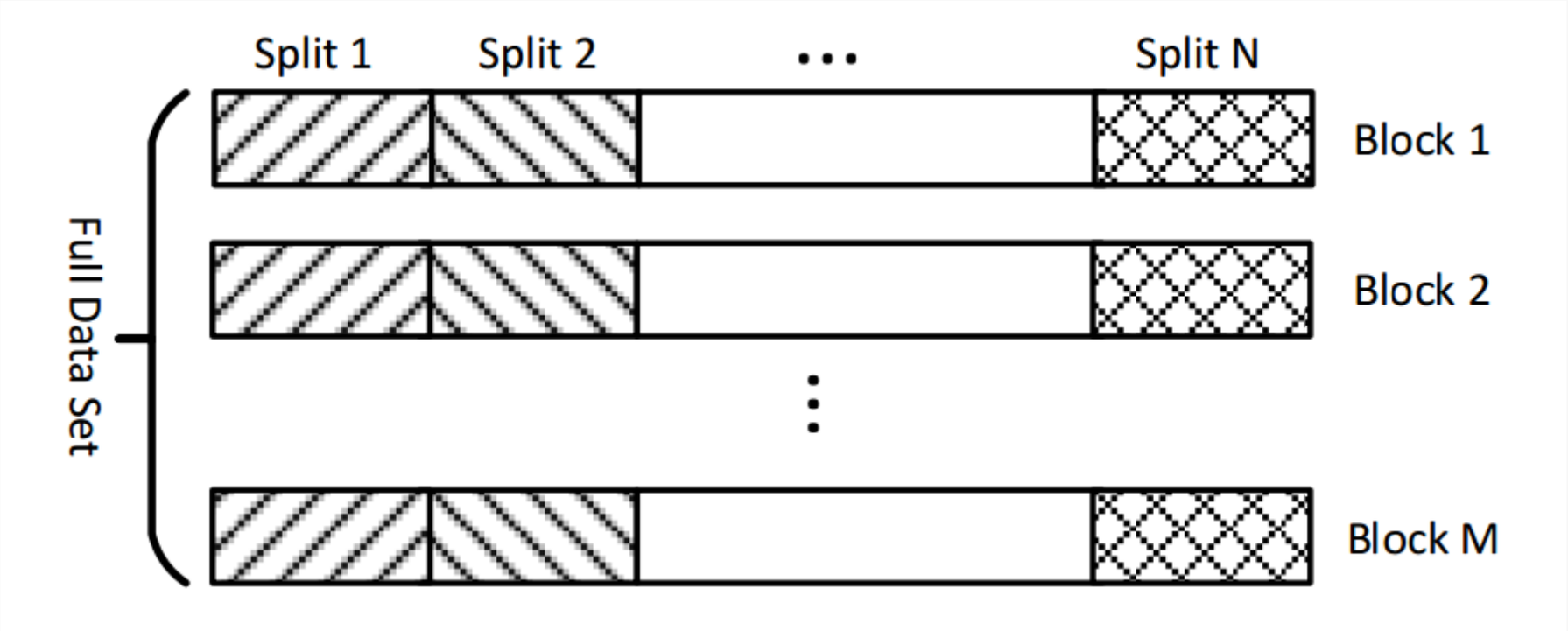
- Performance problem

	A	B	C	D
1	method	ngpu	acc	WER
2	SGD	1	0.887	34.1
3	SSGD	8	0.871	37.8



Block momentum method [1]

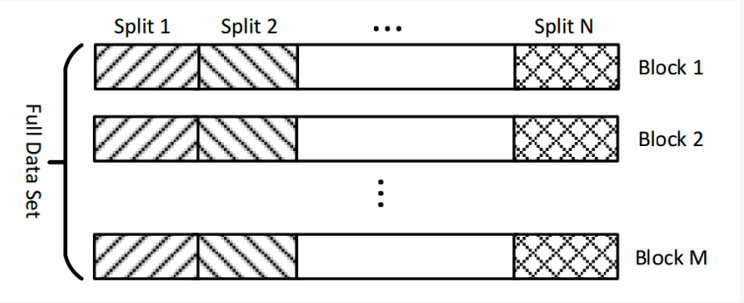
- Require less communication overheads
  - Incremental block training

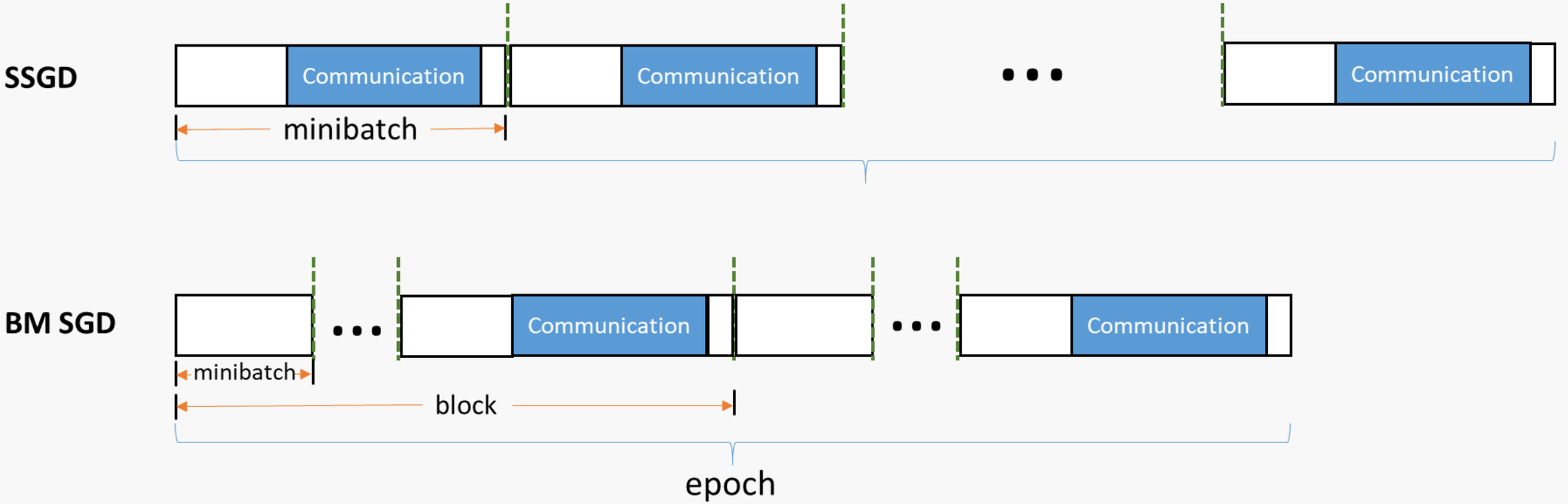


[1] Chen K, Huo Q. SCALABLE TRAINING OF DEEP LEARNING MACHINES BY INCREMENTAL BLOCK TRAINING[J]. 2016.

Block momentum method

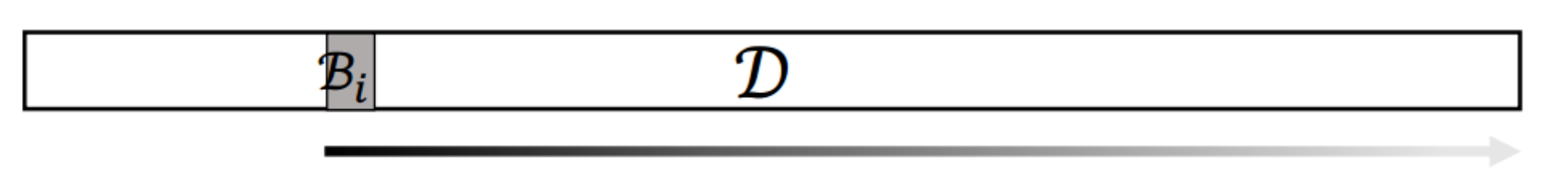
- Require less communication overheads
  - Incremental block training



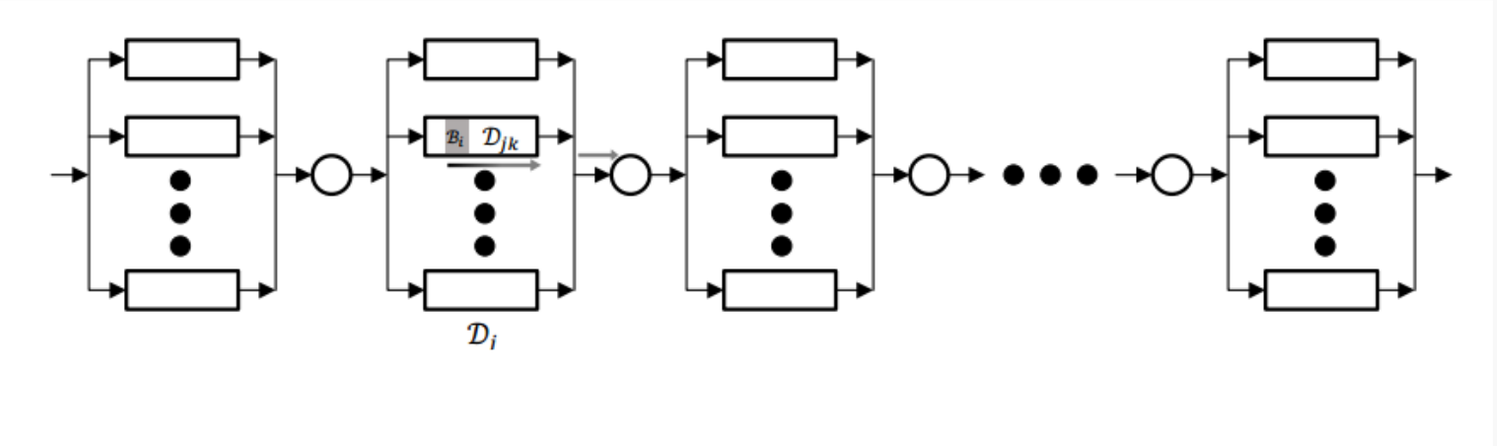


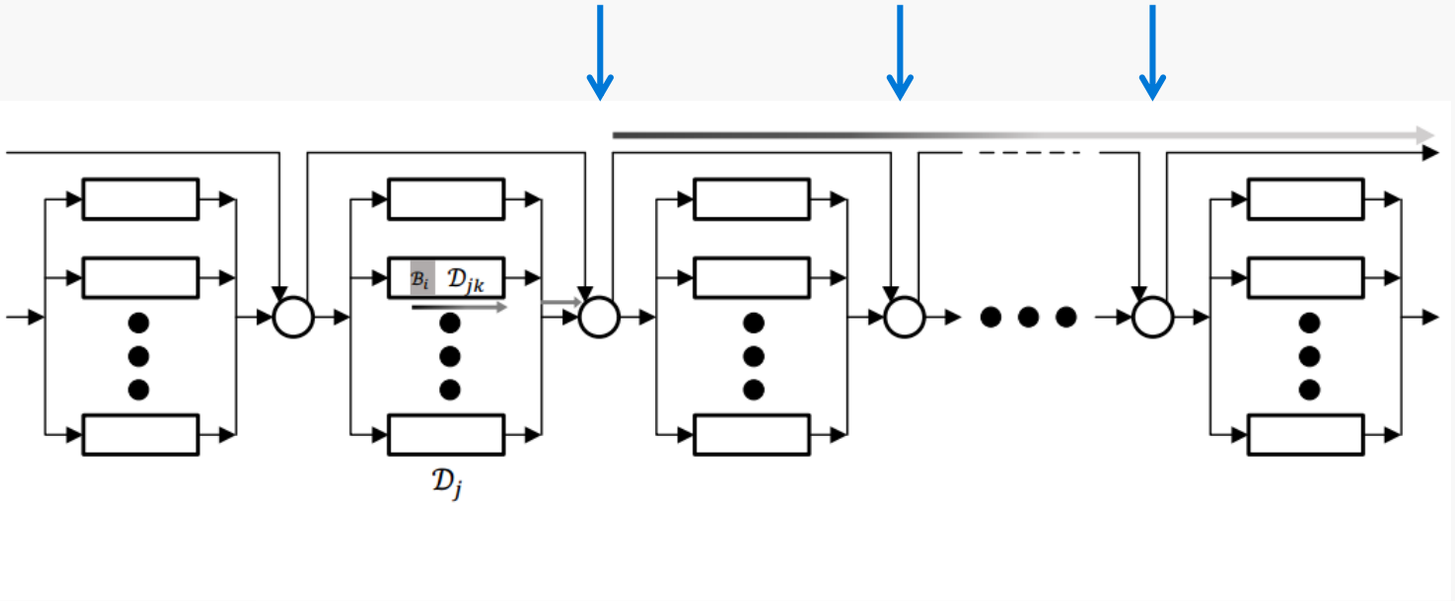
Block momentum method

- Require less communication overheads
  - Incremental block training
- Solve the model accuracy dropping
  - Use momentum



Block momentum method

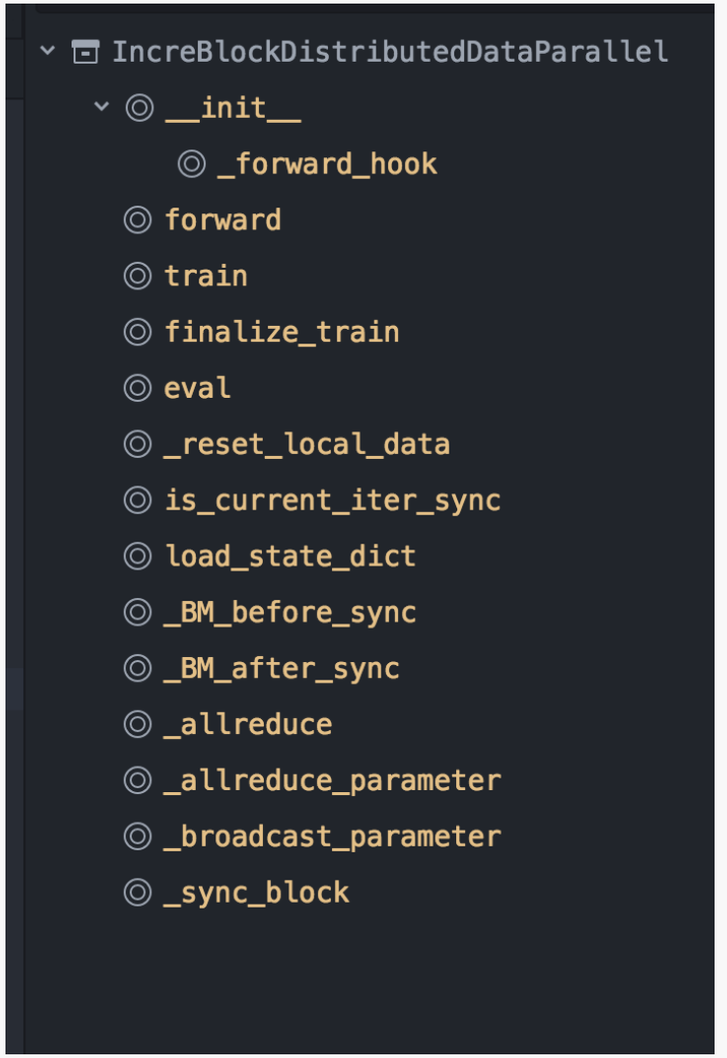




BMUF implementation

- One thread/flow for each GPU

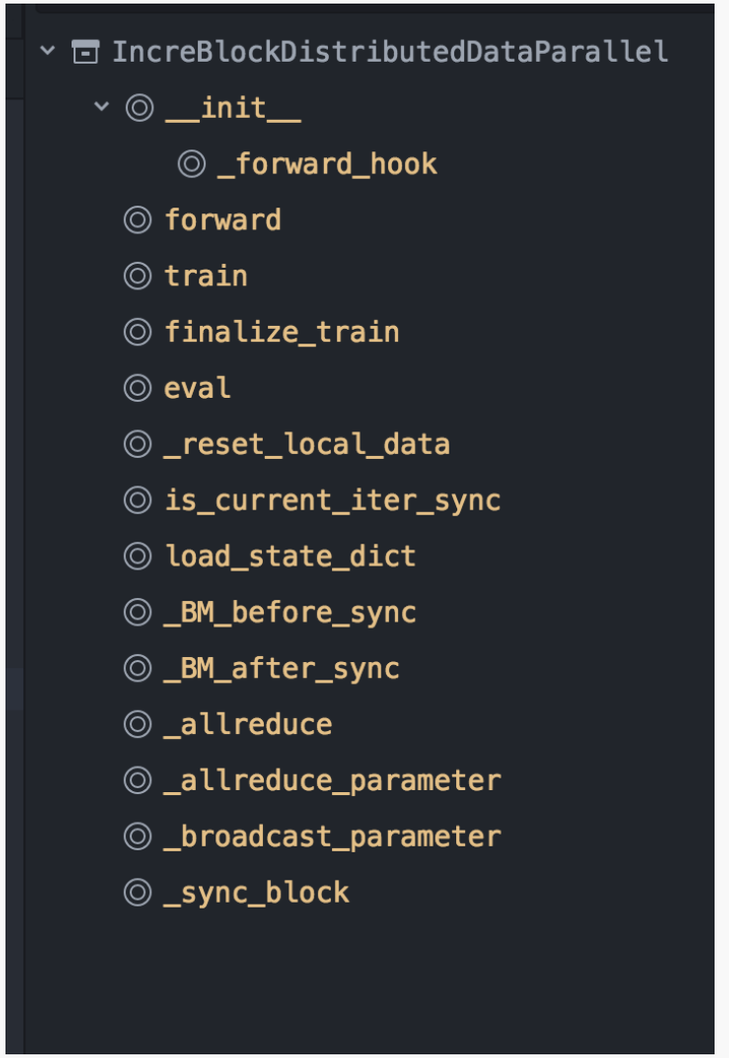
```
    # ASSUMING ALL NODES HAS SAME NUMBER OF GPUS
    for local_rank in range(num_gpus_per_node):
        mp = multiprocessing.get_context("spawn")
        single_gpu_worker = mp.Process(
            target=train,
            args=(train_args, flow_runid_str, node_rank,
            daemon=False,
        )
        single_gpu_workers.append(single_gpu_worker)
        single_gpu_worker.start()
    for single_gpu_worker in single_gpu_workers:
        single_gpu_worker.join()
```



BMUF implementation

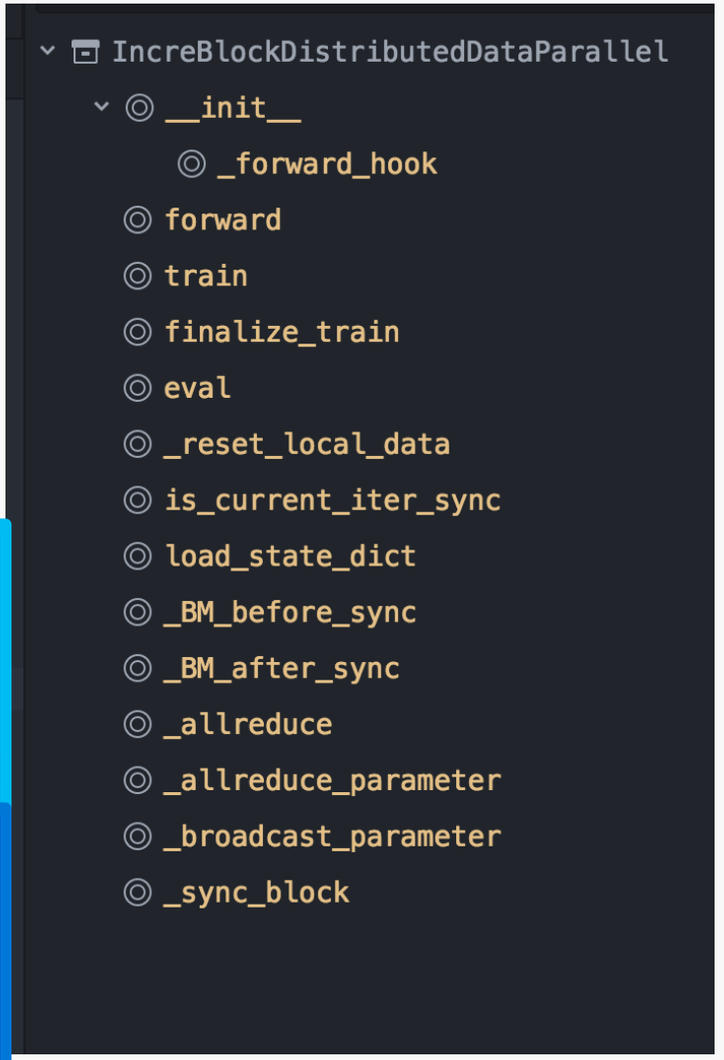
- One thread/flow for each GPU
- Each thread creates a class object

```
model = dist(
    e2e,
    optimizer,
    flow_runid_str,
    rank,
    world_size,
    block_sync_freq=args.block_sync_freq,
    sync_print_schedule=args.sync_print_schedule,
    BM=args.BM,
    reset_optimizer=args.reset_optimizer,
    fs=args.outdir + "/sync_file" if args.filesystem_sync else "",
)
```



BMUF implementation

a. Divide whole dataset into several blocks
b. Do warmup
c. For each GPU:
    i. train multiple minibatches on assigned data.
    ii. update the local model from each minibatch
d. Get the overall gradient for each GPU
e. Compute global gradient using momentum
f. Update model & sync with all GPUs.
g. Repeat c, d, e, f



Experiments

- Sequence-to-sequence model in Speech Recognition
- 3 corpus from 400 hours to 10000 hours
- Accuracy: word error rate (WER)
- Speed: frames per second (fr/sec)

Experiments

video - 10M (10000 hrs)

•setup: 1400*2+1000*2 batch:10

•nums. of gpus (epochs 5)

- better results and better speedups, e.g. in 8 gpus case
- we see almost linear speed up(32 gpus: 27X; 64 gpus: 49X) without WER being impacted largely (18.5 to 19.1 to 19.5)

	A	B	C	D	E
1	method	ngpu	WER	SPEED (fr/sec)	speedups
2	SSGD	8	22	12903	6.7
3	BMUF	8	18.8	15385	8
4		16	18.5	34783	18.1
5		32	19.1	53333	27.8
6		64	19.5	94118	49

Experiments

Synchronization overheads

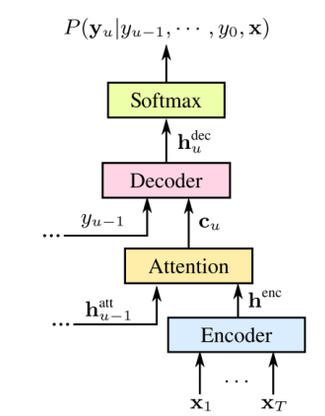
	A	B	C	D	E
1	method	ngpu	speedups	sync. time (communication + waiting)	sync time / total time
2	BMUF	8	6.9	19	0.03519
3		16	13.2	26	0.04815
4		32	26.7	33	0.06111
5		64	51.3	56	0.1037

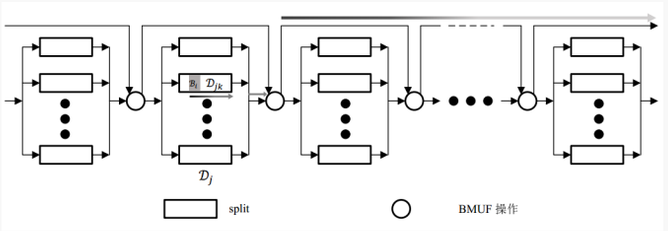
Conclusion

- Propose a block momentum based solution of distributed training for sequence-to-sequence (S2S) model in Pytorch
- Compare block momentum with SGD, Synchronous SGD and model average for S2S distributed training for videos and aloha data sets.
- Show that it is possible to get a linear speedup when increasing the number of GPUs, without WER deterioration
- In the future, we will work with Pytorch folks to put it in open source.

Outline

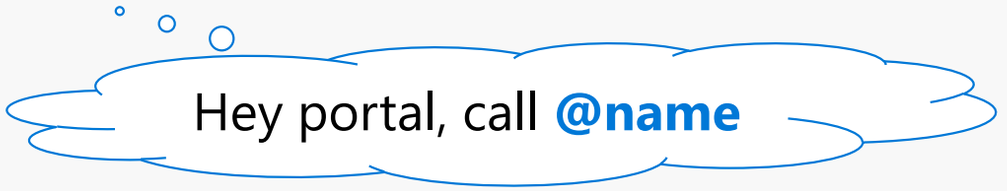
- **PySpeech**
  - E2E Speech Processing
  - Distributed Training
  - Contextual Speech Recognition (Assistant)
    - Shallow Fusion Framework (paper 1)
    - Deep Context Framework (paper 2)





Motivation

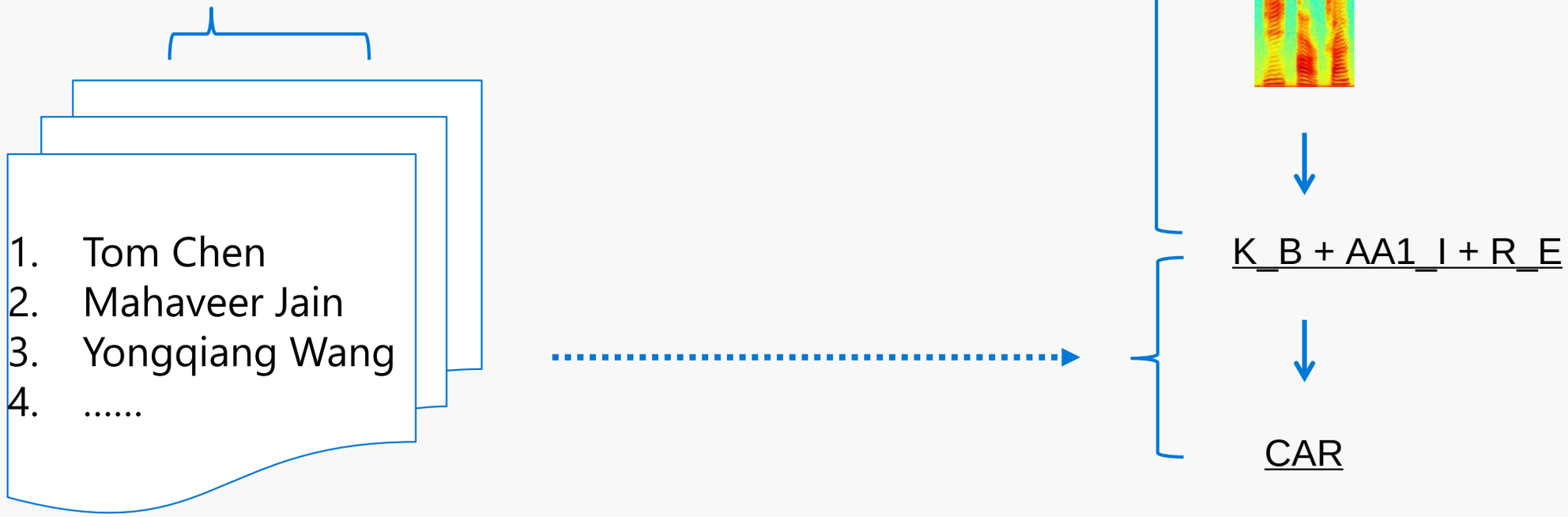
- End-to-end (E2E) ASR
  - Only need transcription & audios
  - Removal of **seed models**, **alignment** & **LM**, etc.
- Contextual (CTX) E2E ASR
  - Removal of **lexicon** & **G2P**, etc.
- Research
  - Within domain, E2E ASR works well
  - How to make it **adaptive**?



Introduction

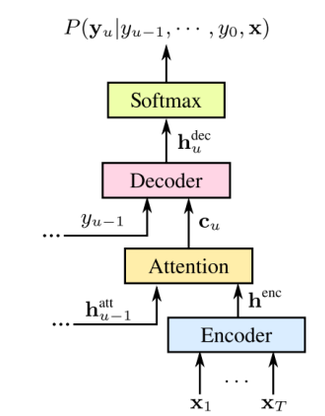
- Contextual (CTX) Speech Recognition

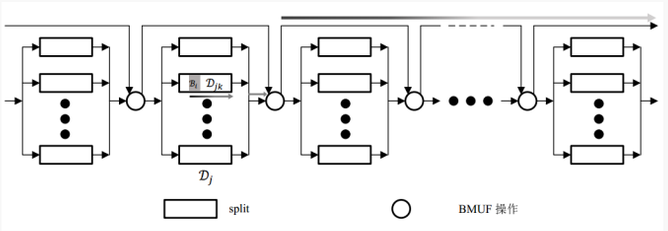
  - Hey portal call **@name** please



Outline

- **PySpeech**
  - E2E Speech Processing
  - Distributed Training
  - Contextual Speech Recognition (Assistant)
    - Shallow Fusion Framework (paper 1)
    - Deep Context Framework (paper 2)



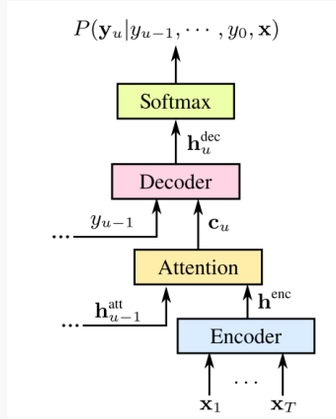


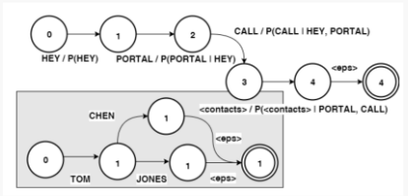
Contextual E2E ASR: inference

- Introduction
  - From ASR to End-to-end (E2E) ASR
  - Contextual (CTX) ASR

- Contextual E2E ASR
  - Framework
  - Search space
  - Decoding

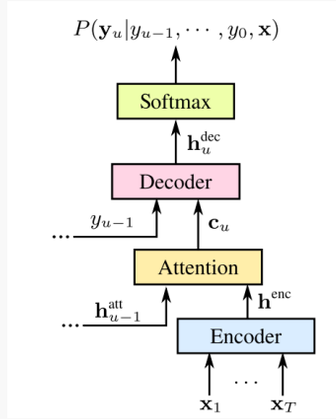
- Experiments

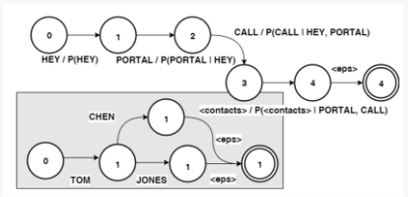




Outline

- Introduction
  - From ASR to End-to-end (E2E) ASR
  - Contextual (CTX) ASR
- Contextual E2E ASR
  - Framework
  - Search space
  - Decoding
- Experiments

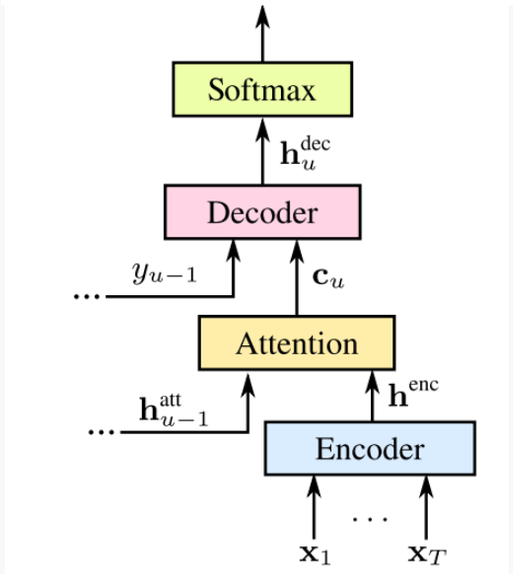




Framework

- Contextual (CTX) Speech Recognition

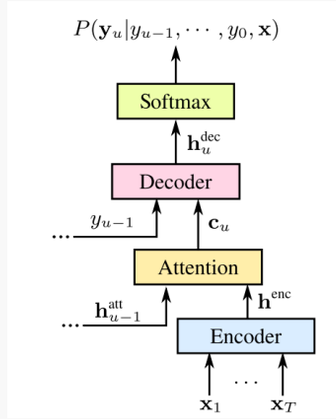
$$\mathbf{l}^* = \arg\max_{\mathbf{l}} \log P(\mathbf{l}|\mathbf{x}) + \lambda \log P_C(\mathbf{l})$$

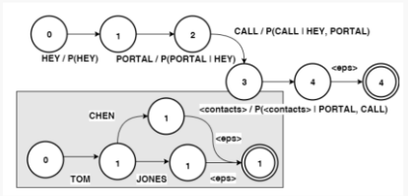


1. Tom Chen
2. Mahaveer Jain
3. Yongqiang Wang
4. .....

Outline

- Introduction
  - From ASR to End-to-end (E2E) ASR
  - Contextual (CTX) ASR
- **Contextual E2E ASR**
  - Framework
  - Search space
  - Decoding
- Experiments

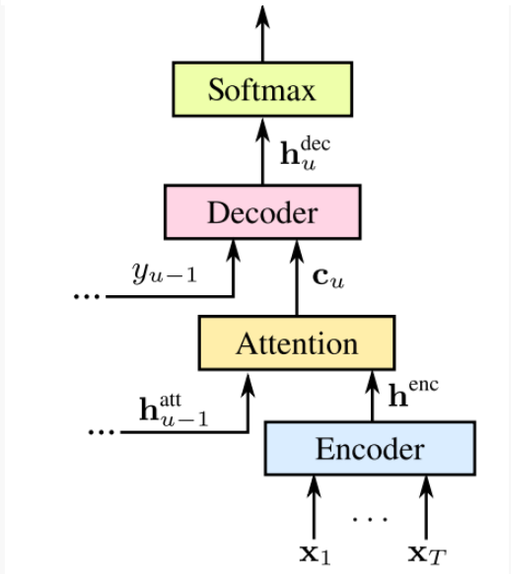


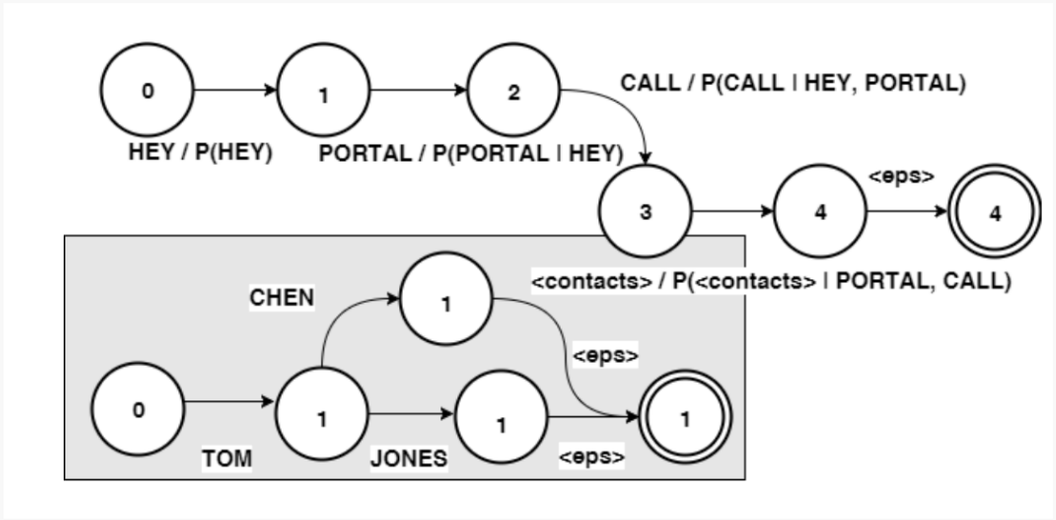


Search Space

- Class-based LM & WFST

$$\mathbf{l}^* = \arg\max_{\mathbf{l}} \log P(\mathbf{l}|\mathbf{x}) + \lambda \log P_C(\mathbf{l})$$



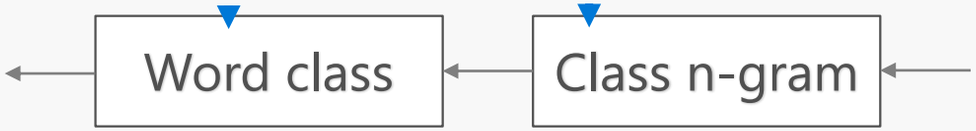


Search Space

- Class-based LM & WFST

$$P(w_1,\ldots,w_N) \approx \prod_{i=n}^{N} P(w_i \mid w_{i-n+1})$$

$$= P(w_i \mid c(w_i))P(c(w_i) \mid c(w_{i-1}),\ldots,c(w_{i-n+1}))$$



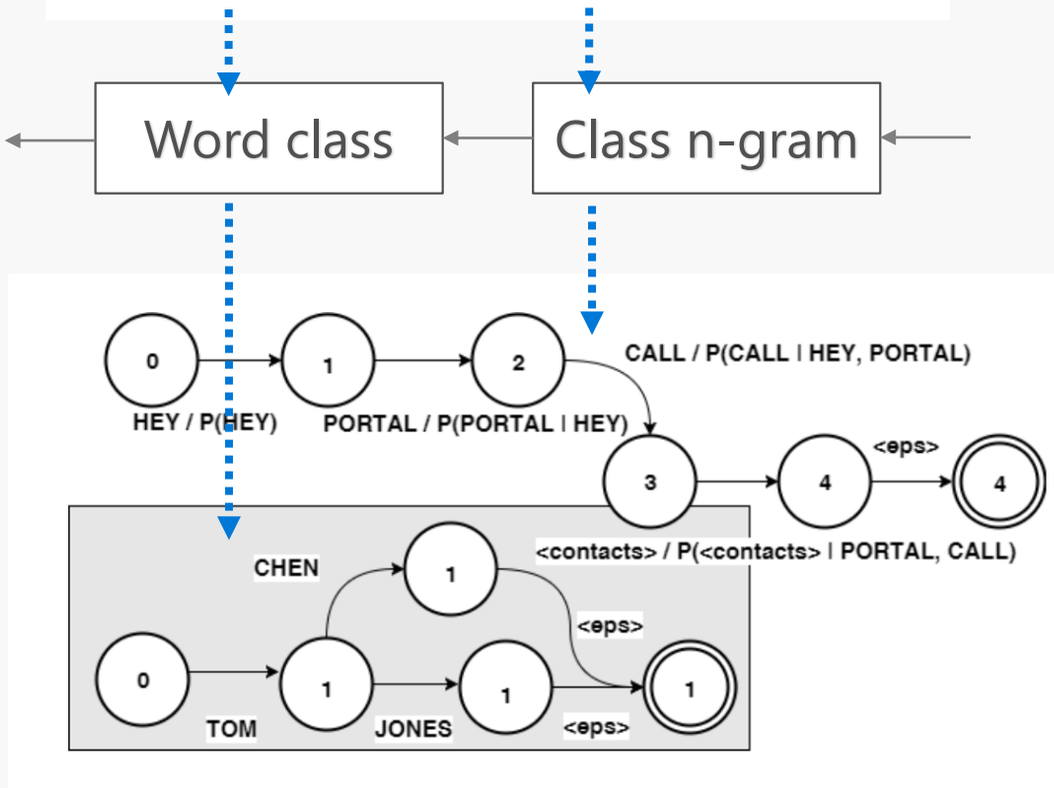
Hey portal, call **@name**

Search Space

- Class-based LM & **WFST**

$$P(w_1,\ldots,w_N) \approx \prod_{i=n}^{N} P(w_i \mid w_{i-n+1})$$

$$= P(w_i \mid c(w_i))P(c(w_i) \mid c(w_{i-1}),\ldots,c(w_{i-n+1}))$$



Hey portal, call **@name**

Search Space

- Class-based LM & WFST

Compile CLM to G.fst

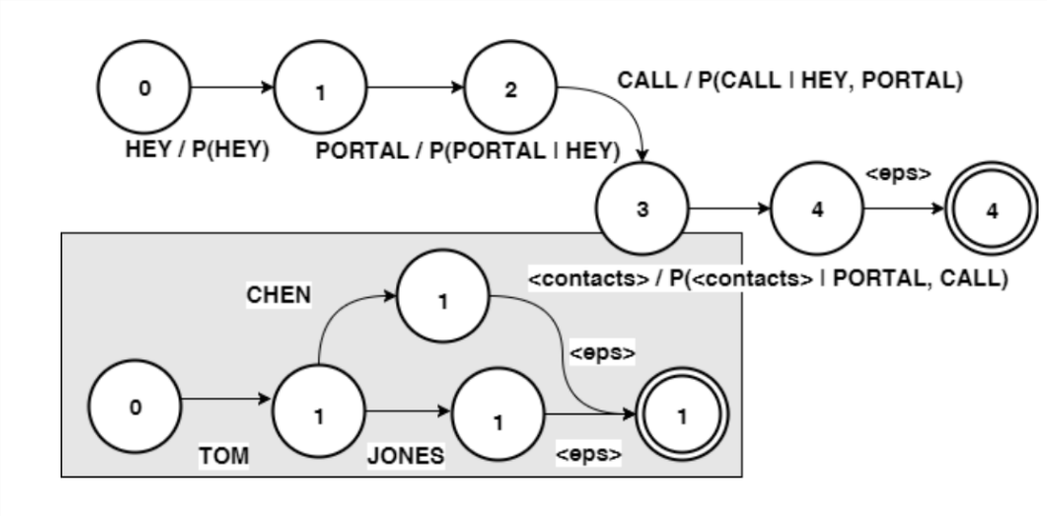
Transform to grapheme level LG.fst

Optimize LG.fst

On-the-fly generate word class WFST O.fst

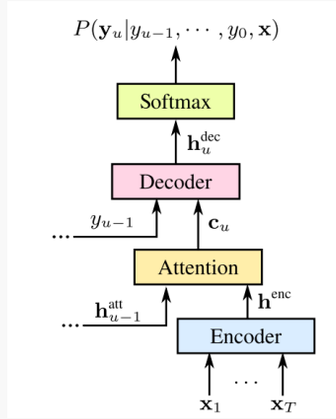
Optimize O.fst

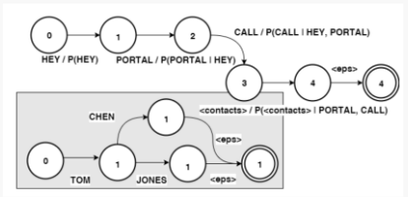
Insert O.fst into LG.fst



Outline

- Introduction
  - From ASR to End-to-end (E2E) ASR
  - Contextual (CTX) ASR
- Contextual E2E ASR
  - Framework
  - Search space
  - Decoding
- Experiments

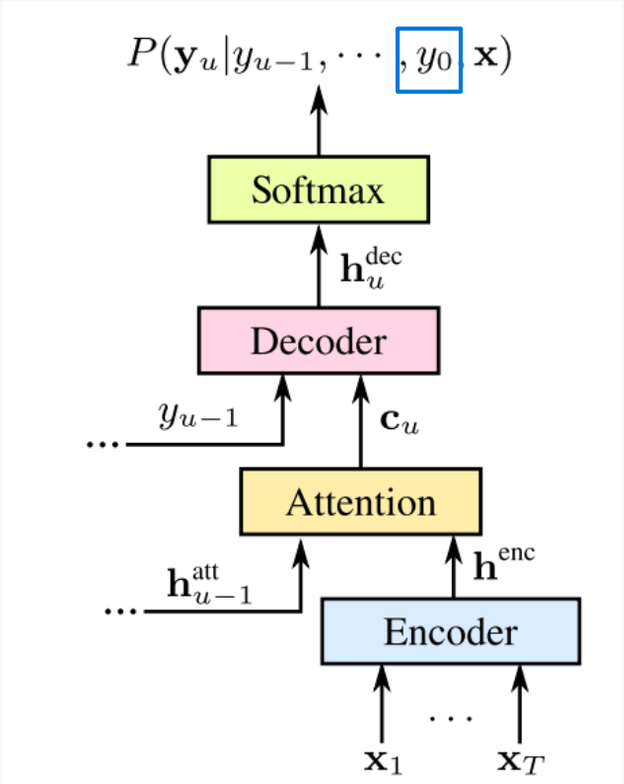




Decoding

- Problem:
  - E2E model and N-gram LM have different history length

$$P(w_1,\ldots,w_N) \approx \prod_{i=n}^{N} P(w_i \mid w_{i-n+1})$$

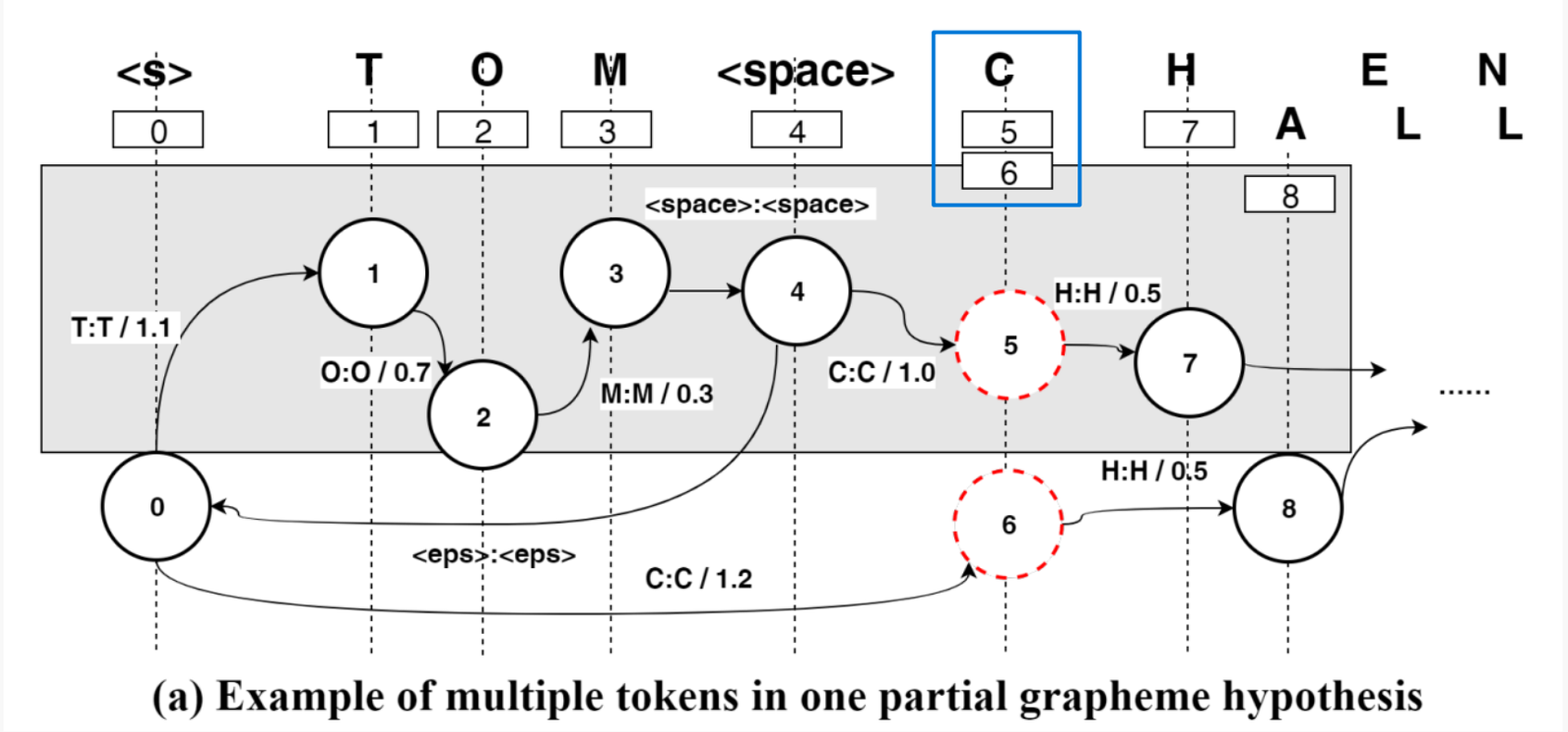


Decoding

- Problem:
  - E2E model and N-gram LM have different history length
  - Un-deterministic WFST from N-gram LM
  - e.g.
    T O M <space> C H E N
    T O M <space> C A L L

Decoding

- Problem:
  - E2E model and N-gram LM have different history length
  - Un-deterministic WFST from N-gram LM
  - e.g.
    T O M <space> C H E N
    T O M <space> C A L L

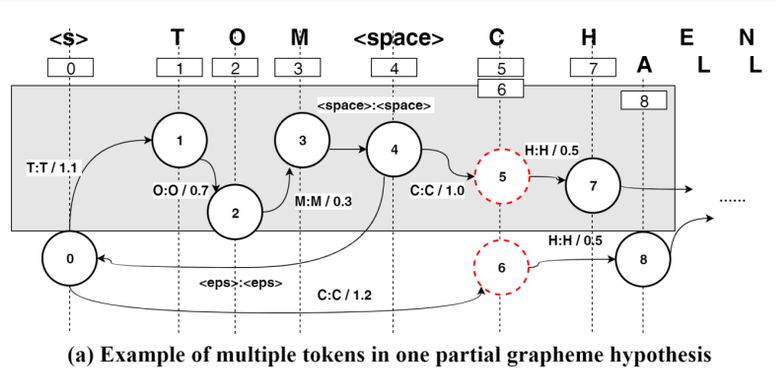


(a) Example of multiple tokens in one partial grapheme hypothesis

Un-deterministic

Decoding

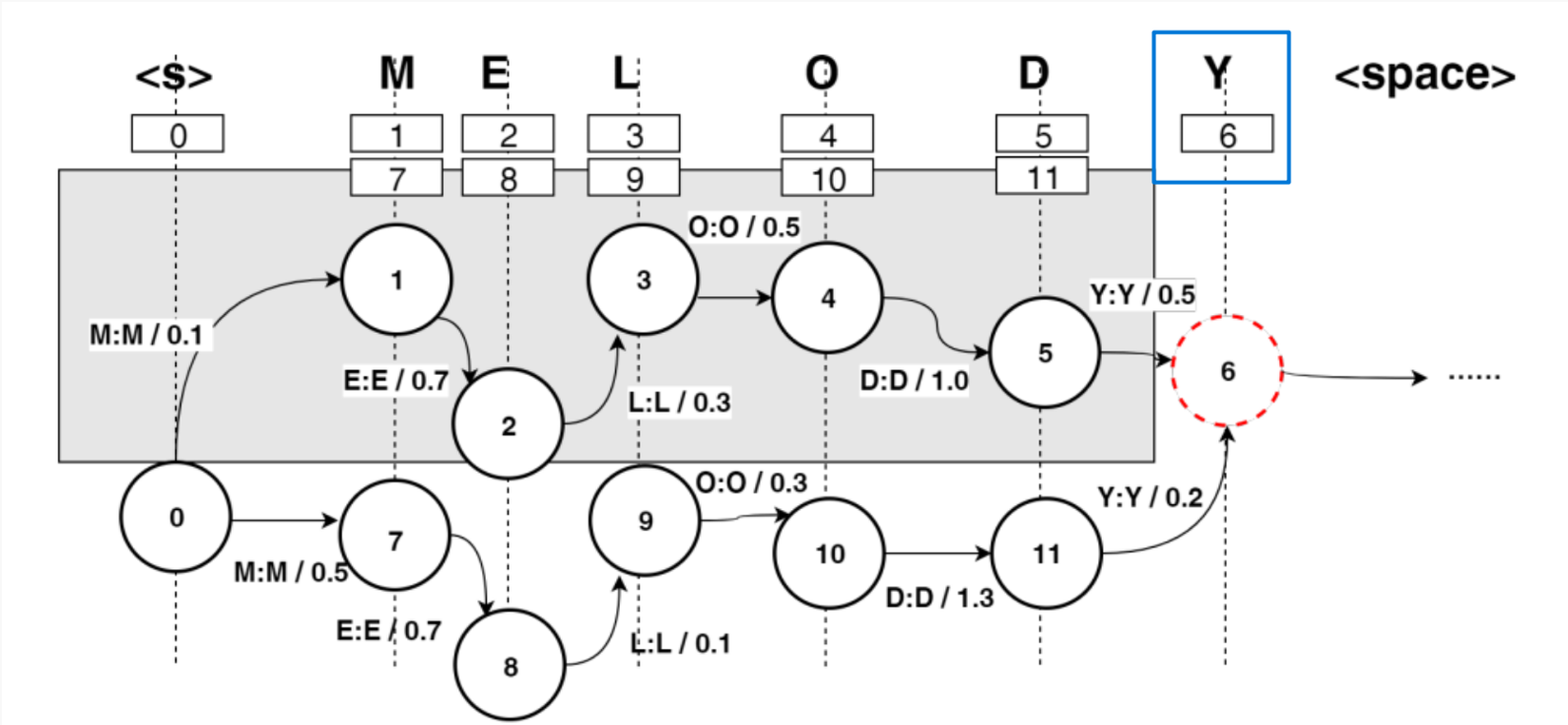
- Problem:
  - E2E model and N-gram LM have different history length
  - Un-deterministic WFST from N-gram LM
- Solution:
  - For each hyp. keep multiple tokens: **Token Passing Algorithm**



(a) Example of multiple tokens in one partial grapheme hypothesis

Decoding

- Solution:
  - For each hyp. , keep multiple tokens: Token Passing Algorithm
  - For each hyp. , remain best in each WFST state: **Token recombination**

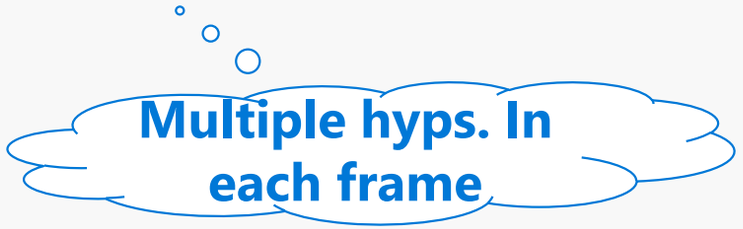


(b) Example of token recombination in one partial grapheme hypothesis

Decoding

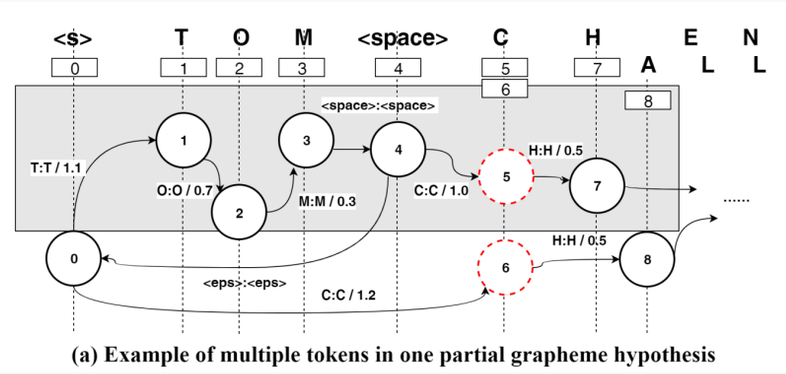
- Solution:
  - For each hyp. , keep multiple tokens: Token Passing Algorithm
  - For each hyp. , remain best in each WFST state: **Token recombination**

    - In CTC / HMM-DNN: hashMap(**frame**, state)
    - In S2S: hashMap(**hyps.**, state)

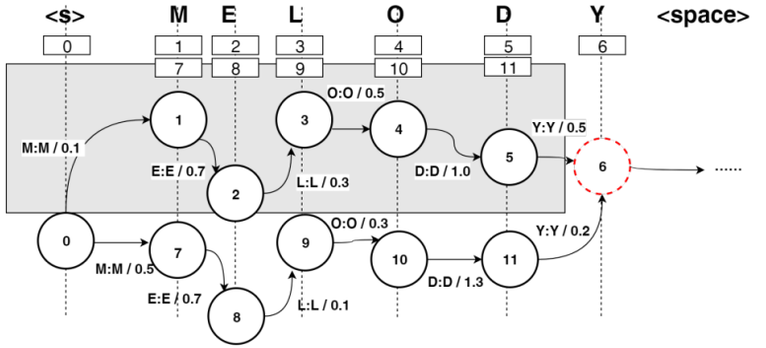


Decoding

- Problem:
  - E2E model and N-gram LM have different history length
  - Un-deterministic WFST from N-gram LM
- Solution:
  - For each hyp. , keep multiple tokens: **Token Passing Algorithm**
  - For each hyp. , remain best in each WFST state: **Token recombination**



(a) Example of multiple tokens in one partial grapheme hypothesis



(b) Example of token recombination in one partial grapheme hypothesis

Decoding

Algorithm 1: Token Passing Algorithm for E2E Model

Input: $\mathbf{h}$, defined in Equation (2)
Initialization: $\mathcal{H} = \{(\boldsymbol{s}_0, \texttt{"<bos>"}, \texttt{FST.start}, 0)\}$;
while EndDetection($\mathcal{H}$) [19] [2] **do**
  $\mathcal{H}' \leftarrow \{\}$;
  for $(\boldsymbol{s}_k, \mathbf{l}_k, t_k, q_k) \in \mathcal{H}$ **do**
    for *each grapheme* l **do**
      $\mathcal{H}_l = \{\}$;
      • extend decoder network by grapheme l
        $\mathbf{l}'_k \leftarrow \mathbf{l}_k + l$; $q'_k \leftarrow q_k + p(l|\boldsymbol{s}_k, \mathbf{h})$;
        $\boldsymbol{s}'_k \leftarrow$ UpdateDecoderState$(\boldsymbol{s}_k, l)$;
    end
  end
  $\mathcal{H} \leftarrow$ SelectTopN$(\mathcal{H}', B)$
end
return best path in $\mathcal{H}$;

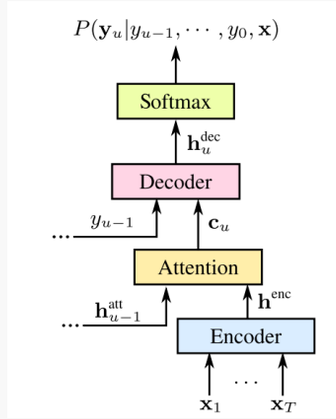
Standard beam search

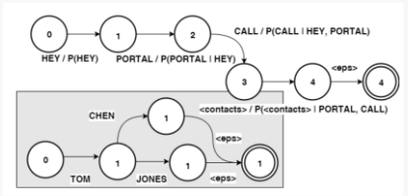
Multiple tokens in each hyps.

Token recombination: hashMap(hyps., state)

Outline

- Introduction
  - From ASR to End-to-end (E2E) ASR
  - Contextual (CTX) ASR
- Contextual E2E ASR
  - Framework
  - Search space
  - Decoding
- Experiments





Experiments

- Target:
  - Improve both general ASR and CTX ASR
- Train
  - 10 million utts. Aloha data
  - Fine-tune with 2 million “HEY PORTAL” data
  - 2*1400 BLSTM enc. + 2*700 LSTM dec.
  - CTC + S2S criteria
- Test
  - 3-gram CLM
  - CTX ASR, e.g. Hey portal call **Tom Chen**

Experiments

Table 1. *WER of End-to-end ASR in General and Contextual ASR.*

system	General	Contextual
E2E	5.9	35.1
+ n-gram LM	**5.6**	31.4
+ Class LM	5.7	**13.5**

- Significant improvement in CTX ASR

Experiments

Table 1. *WER of End-to-end ASR in General and Contextual ASR.*

system	General	Contextual
E2E	5.9	35.1
+ n-gram LM	**5.6**	31.4
+ Class LM	5.7	**13.5**

- Significant improvement in CTX ASR
- Not hurt general ASR

Experiments

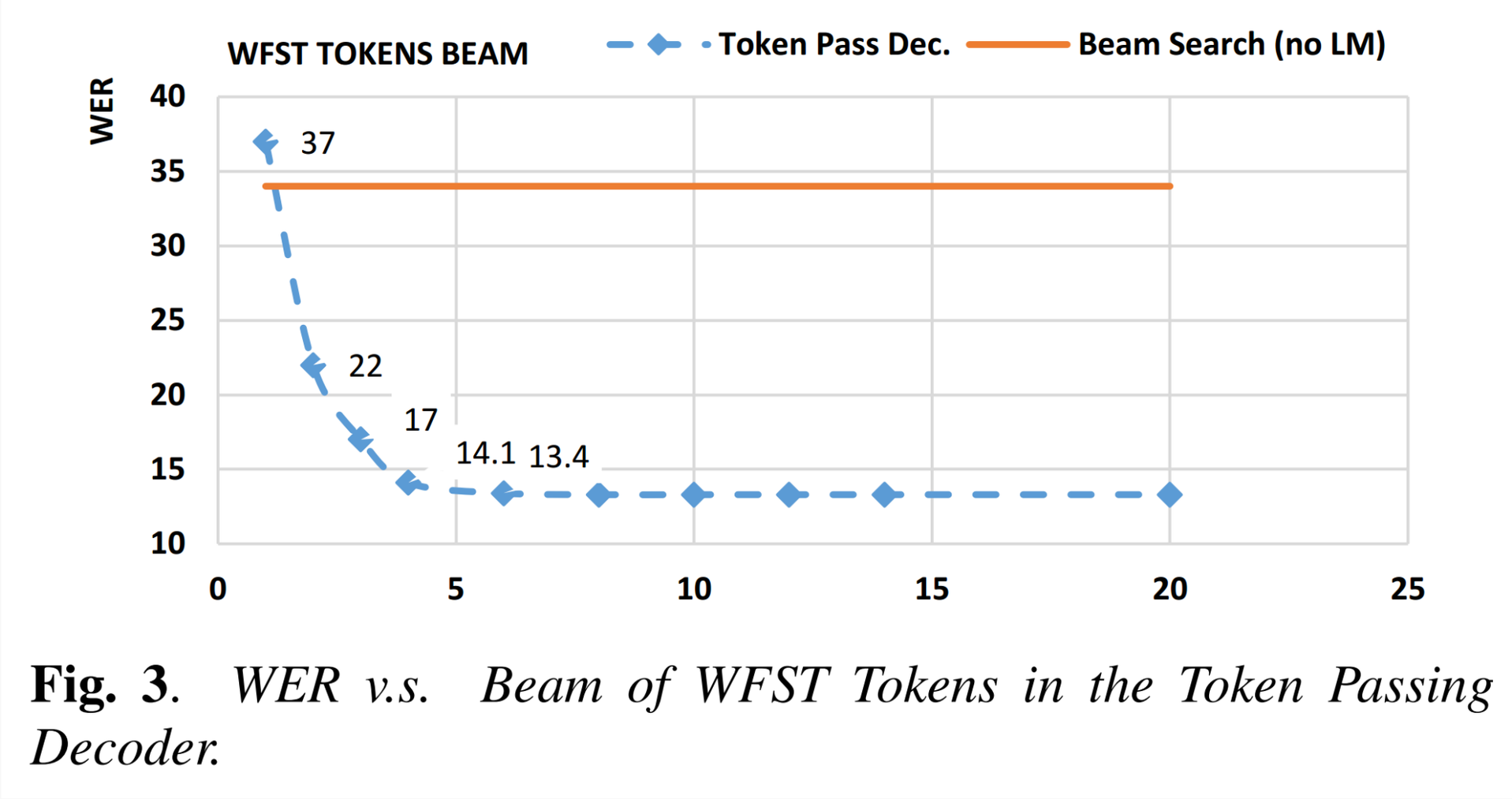


Fig. 3. *WER v.s. Beam of WFST Tokens in the Token Passing Decoder.*

- Num. of tokens in each hypothesis

Experiments

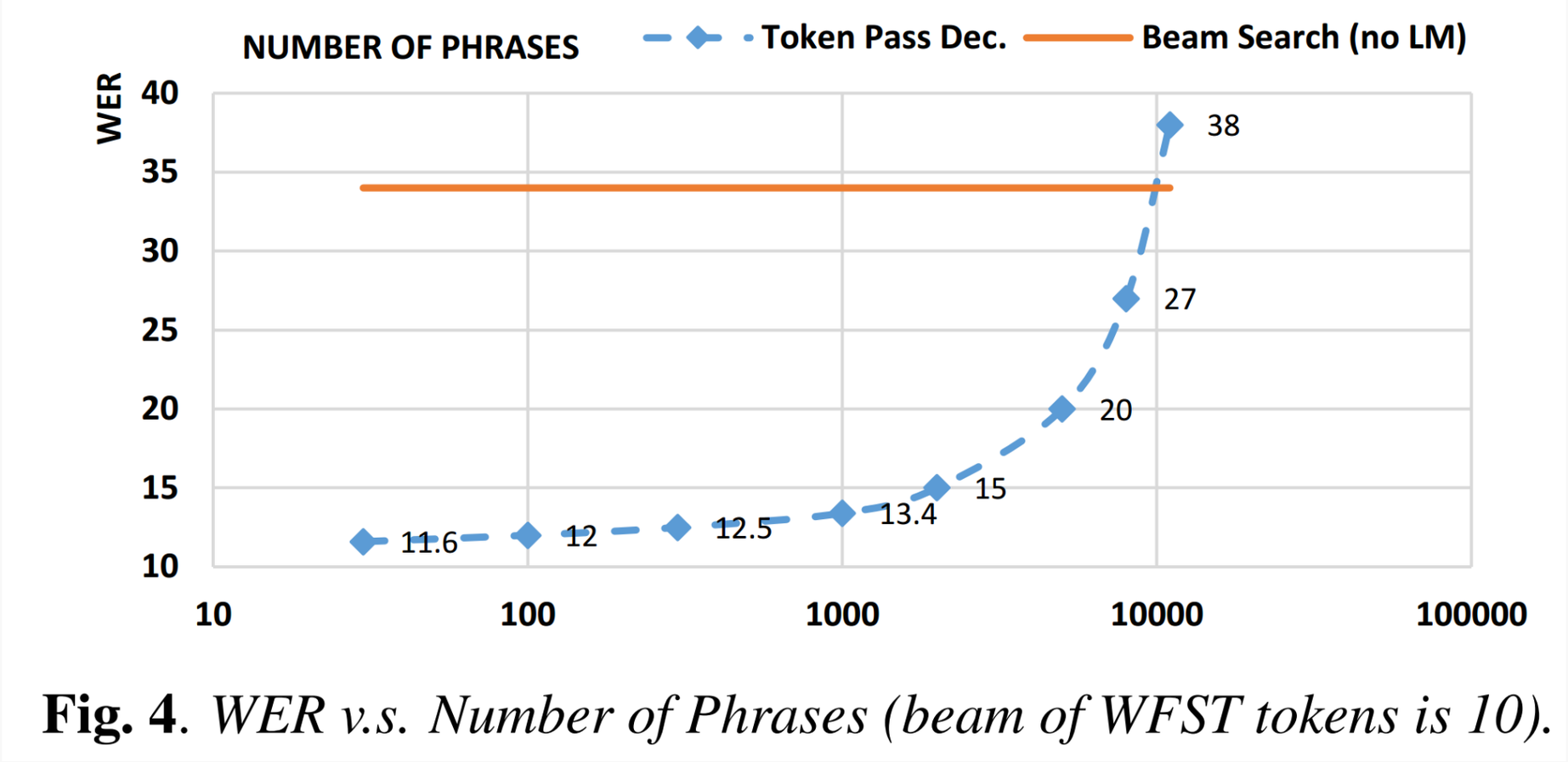


Fig. 4. *WER v.s. Number of Phrases (beam of WFST tokens is 10).*

- 5K phrases

Experiments

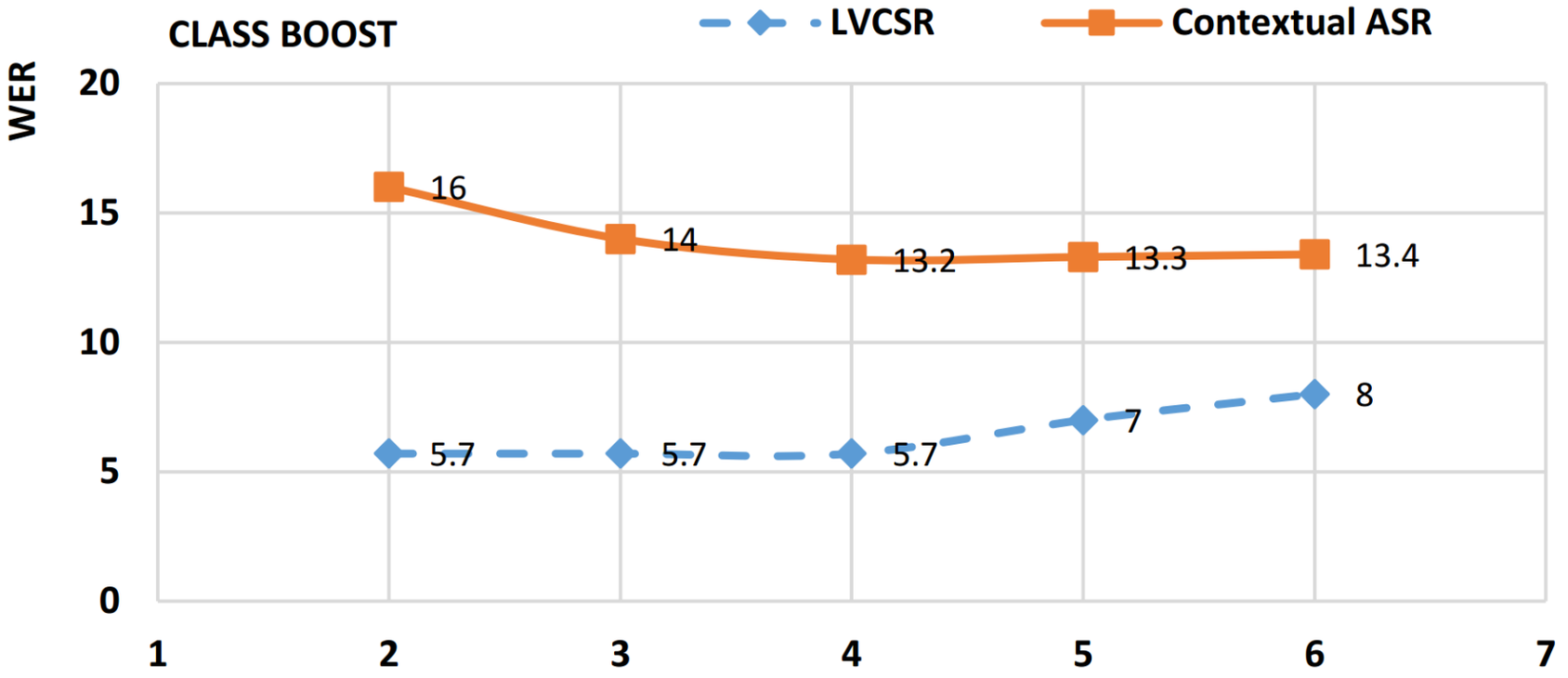
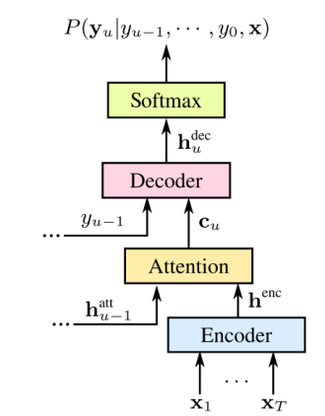


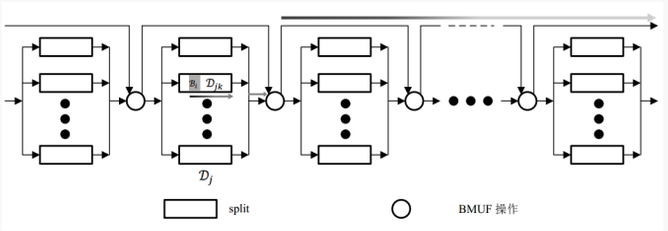
Fig. 5. *class boosting*, defined in Section 3.1 and its Effects in General and Contextual ASR. Similar trend in *keyword boosting*.

- Tradeoff between general and contextual ASR

Outline

- **PySpeech**
  - E2E Speech Processing
  - Distributed Training
  - Contextual Speech Recognition (Assistant)
    - Shallow Fusion Framework (paper 1)
    - Deep Context Framework (paper 2)

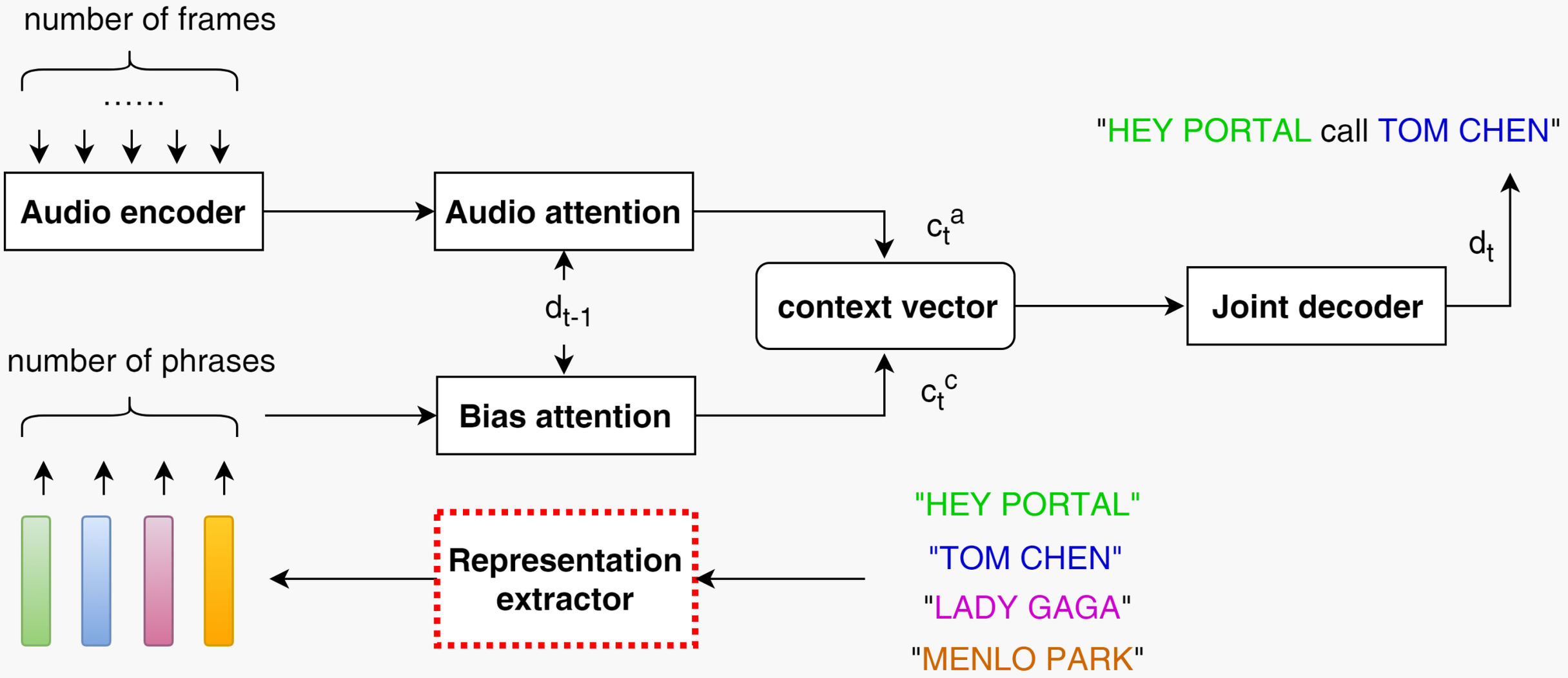




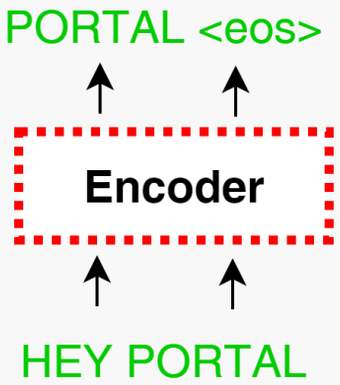
Contextual E2E ASR: training

- Motivation
  - **convey contexts to E2E model**
- We propose
  - Better convey contexts: improve the contextual **representation**
  - Generalize unseen contexts: learn **grapheme-to-phoneme** (G2P)
- Results
  - Better results from both ideas
  - Better scaling up

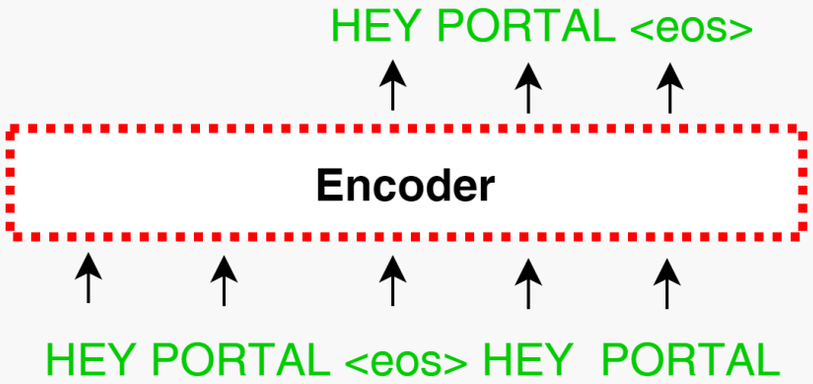
Deep Context Framework



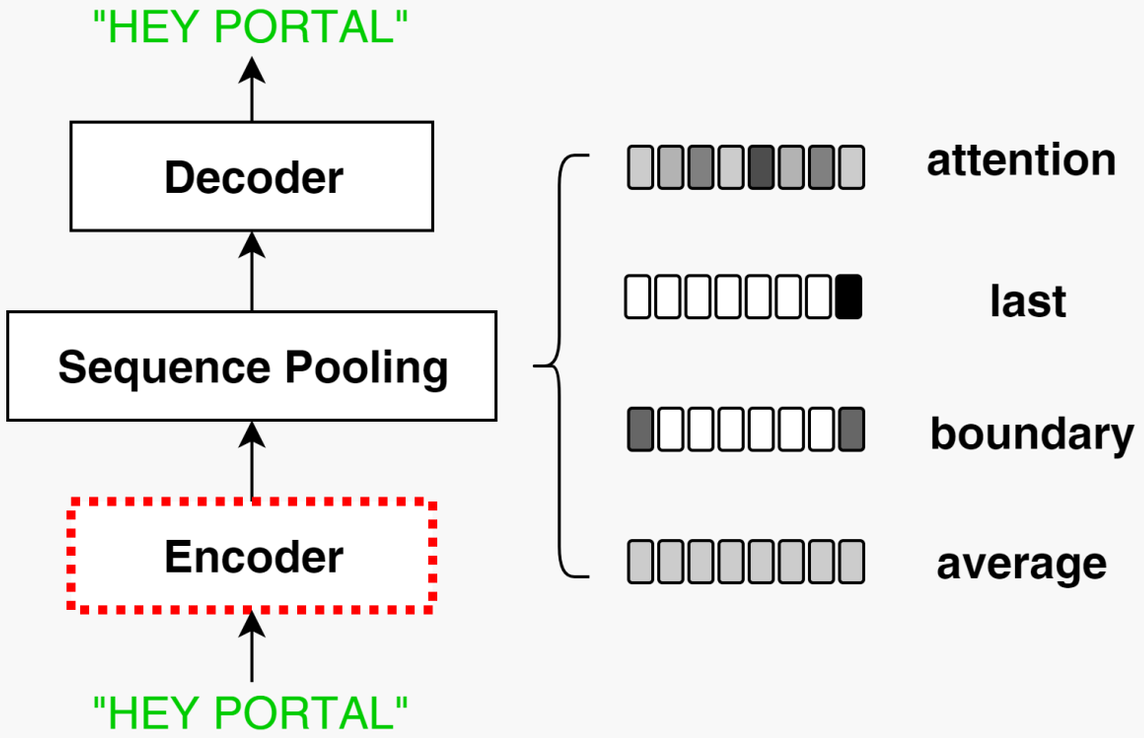
Better Representations



a) NNLM



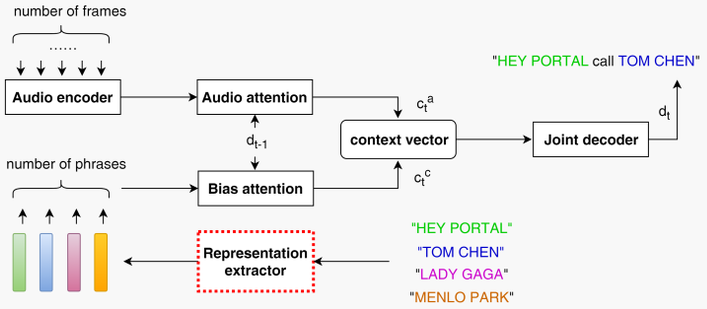
b) sequence-to-sequence

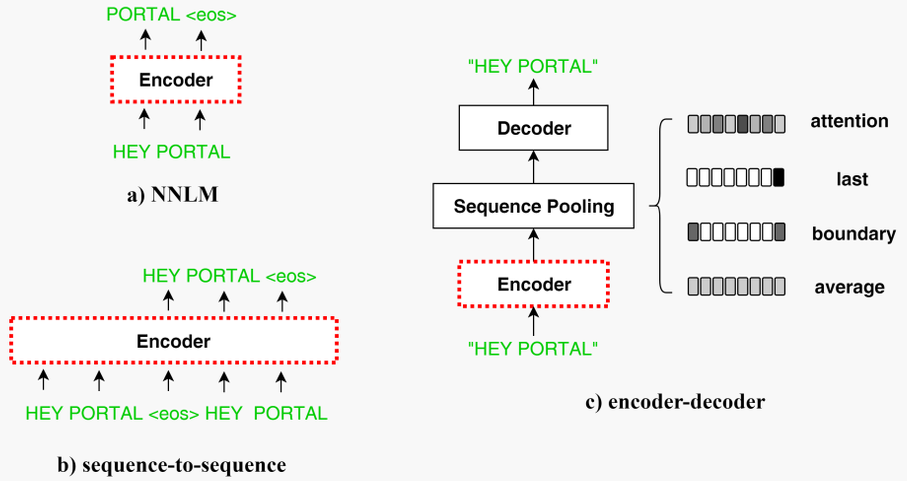


c) encoder-decoder

Better Representations

- Fixed encoder-decoder
  - Consistent sequence pooling function: **last, boundary, average**
- Attention encoder-decoder
  - **Attention** for training
  - **Boundary** for inference





Integrate Phoneme

- Motivation
  - Unseen pronunciation and OOV, e.g. **TOM** v.s. **TOMMY**
  - Similar word pieces might have distinct pronunciation, e.g. **Kathi** v.s. **Katie**

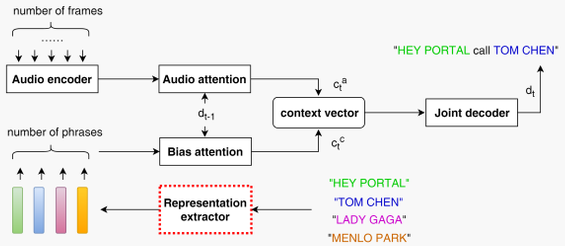
```
REF:  hey portal call KATHI CLENDENIN

HYP:  hey portal call KATIE CLANTONE
Eval:                 S     S
HYP:  hey portal call KATIE CLENTONY
Eval:                 S     S
```

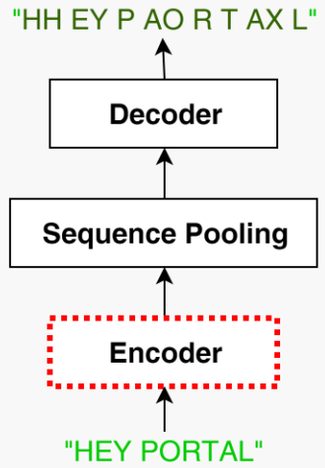
Standard E2E

CLAS

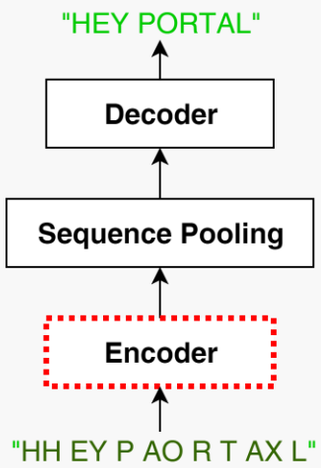
Integrate Phoneme

- 5 variants
  - b, c need lexicons in inference

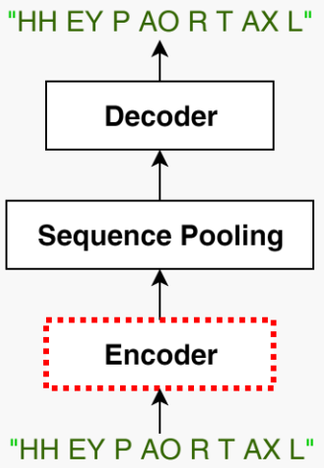




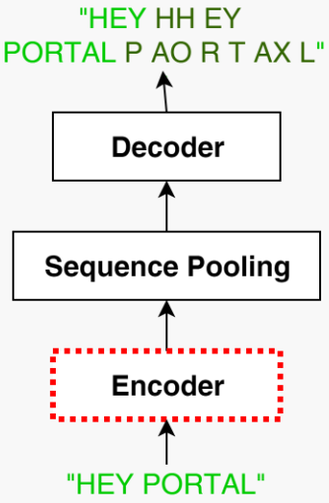
a) graphemes to phonemes



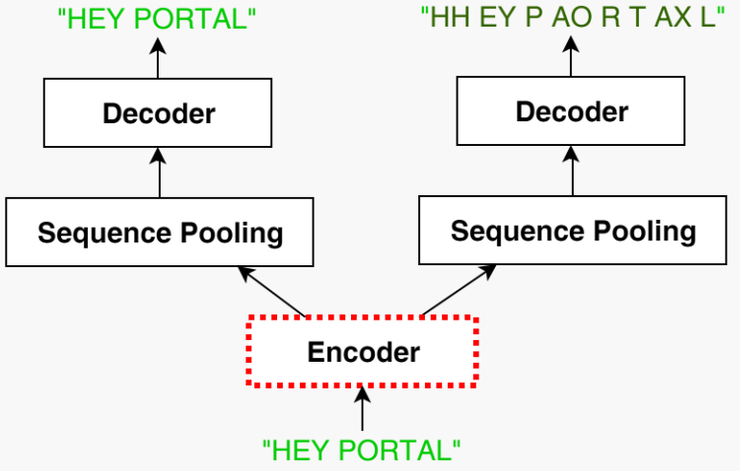
b) phonemes to graphemes



c) phonemes to phonemes



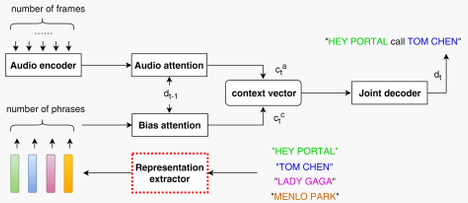
d) graphemes to concatenated graphemes and phonemes



e) graphemes to multi-task graphemes and phonemes

Experiments

- Single domain (default):
  - 100 hours train + 5 hours testset (calling domain)
- Multiple domains:
  - 100 hours train + 5 hours testset
  - 5000 hours train
- Unseen domain:
  - 10 hours testset (Web app)
- Representation:
  - entities of 5000 hours
- Background entities
  - 40 in default



Experiments

- Better representation

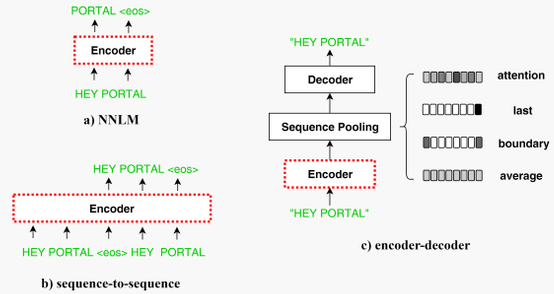


Table 1: *Comparisons of Representation Extractors.*

Extractor	Representation from hidden vectors	WER
n/a	n/a	28.7
NNLM	Last	28.7
S2S	Last	20.2
Att. Enc.-Dec.	Average	19.5
Fixed Enc.-Dec.	Average	10.3
	First and Last	6.7

Experiments

- Representation training

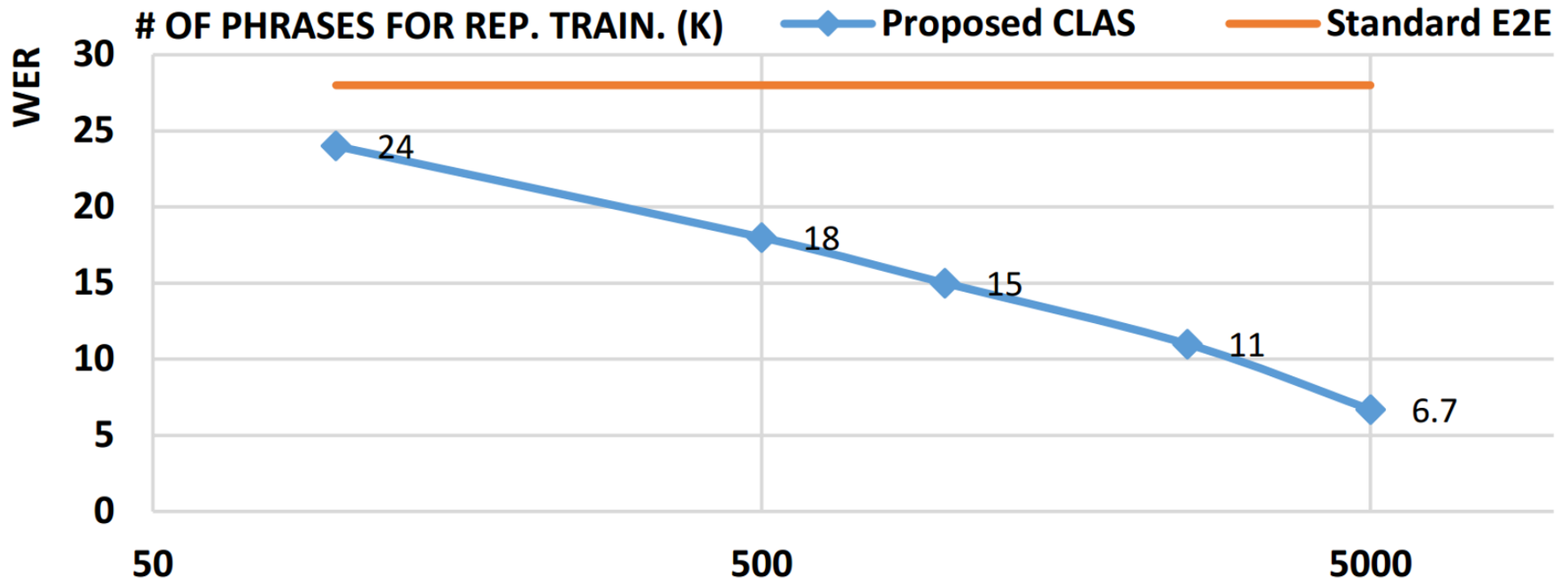


Figure 4: *# of Cross-domain entities for Representation Training v.s. WER.*

Experiments

- Integrate phoneme

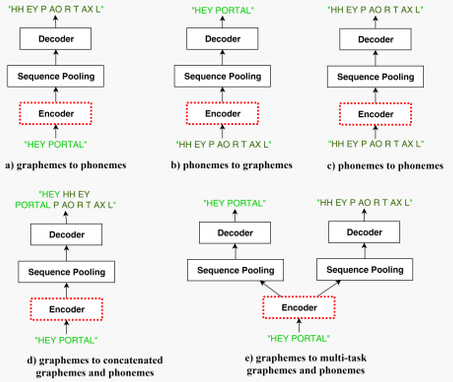


Table 2: *Methods to Integrate Phonemes*

Input Seq.	Output Seq.	WER
n/a	n/a	28.7
Grapheme	Grapheme	6.7
Phoneme	Phoneme	14.3
Phoneme	Grapheme	10.0
Grapheme	Phoneme	4.2
Grapheme	Grapheme $\oplus$ Phoneme	4.7
Grapheme	Grapheme $\otimes$ Phoneme	3.7

Experiments

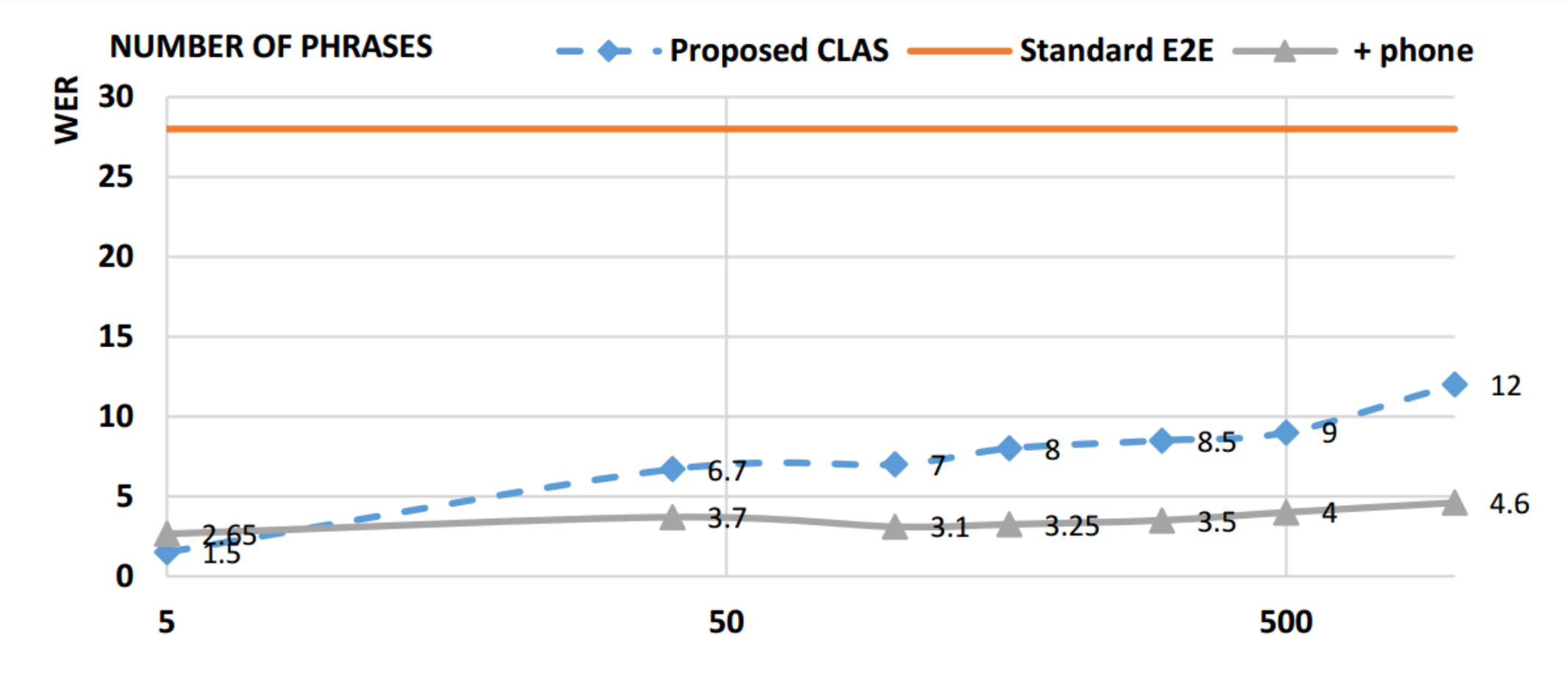
- Domain transfer

Table 3: *Domain Transfer Using a Representation Extractor and External Language Model.*

system	E2E	Improved CLAS	+ Phoneme	+ WFSTLM
WER	38.2	24.5	23.0	11.3

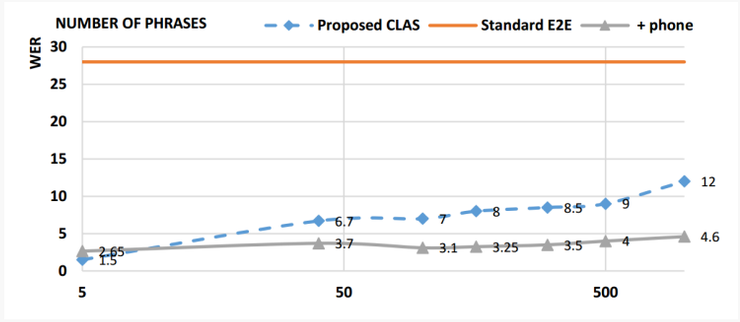
Experiments

- More entities



Experiments

- More entities
  - Why better scaling up: similar graphemes distinct pronunciations




```
REF:  hey portal call ESTHER FUJII

HYP:  hey portal call ESTER  FURGE     0. Standard E2E
Eval:                 S      S
HYP:  hey portal call ESTER  fujii     1. Proposed CLAS
Eval:                 S
HYP:  hey portal call esther fujii     2.    + Phoneme
Eval:

REF:  hey portal call KATHI CLENDENIN

HYP:  hey portal call KATIE CLANTONE   0. Standard E2E
Eval:                 S     S
HYP:  hey portal call KATIE CLENTONY   1. Proposed CLAS
Eval:                 S     S
HYP:  hey portal call kathi clendenin  2.    + Phoneme
Eval:
```

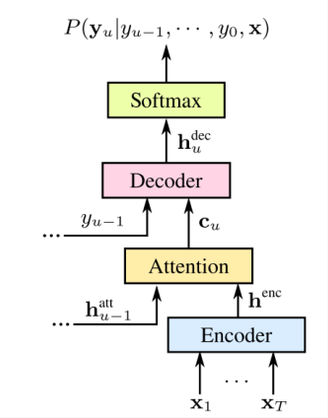
Experiments

- More data more domains

Table 4: *Scaling Up to Larger Data and Multiple Domains.*

Domains	Single	Multiple	
Data Size (hours)	100	100	5000
Standard E2E	28.7	34.1	22.6
CLAS	20.2	28.0	18.5
+ Improve Rep.	6.7	25.1	13.6
+ Phoneme	3.7	23.2	12.5

Intern Summary



- **PySpeech**
  - E2E Speech Processing
  - Distributed Training
  - Contextual Speech Recognition (Assistant)
    - Shallow Fusion Framework (paper 1)
    - Deep Context Framework (paper 2)

- **Ninja**
  - RNNLM Lattice Rescoring
  - Lattice Processing

Future topics

- Joint ASR and SLU modeling
  - e.g. REF: temps for the weekend not today
  - REC: turn off the <time> weekend </time> not <time> today </time>
  - 10% WER increase for now
- Acoustics mismatch
  - Multiple encoders and each for one environment
- Tune PySpeech baseline
  - Discriminative training, schedule sampling, etc.

- Thanks for all your helps!

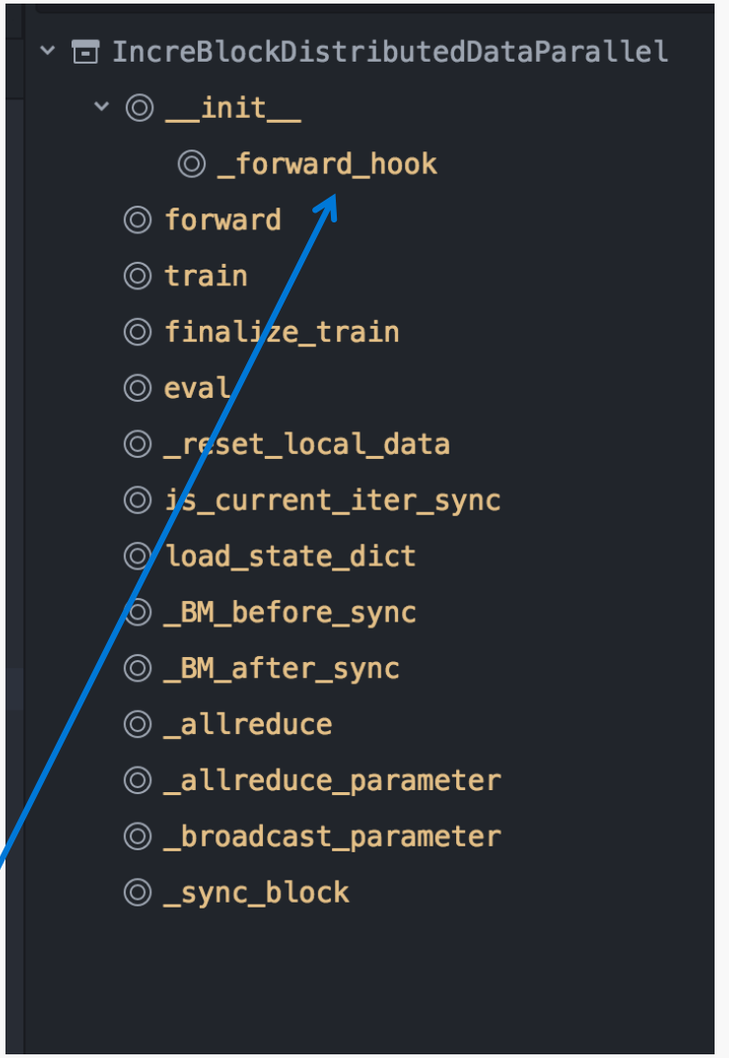
Appendix

BMUF implementation

a. divide whole dataset into several blocks
b. Do warmup
c. For each GPU:
    i. train multiple iterations on assigned data.
    ii. update the local model in each iteration

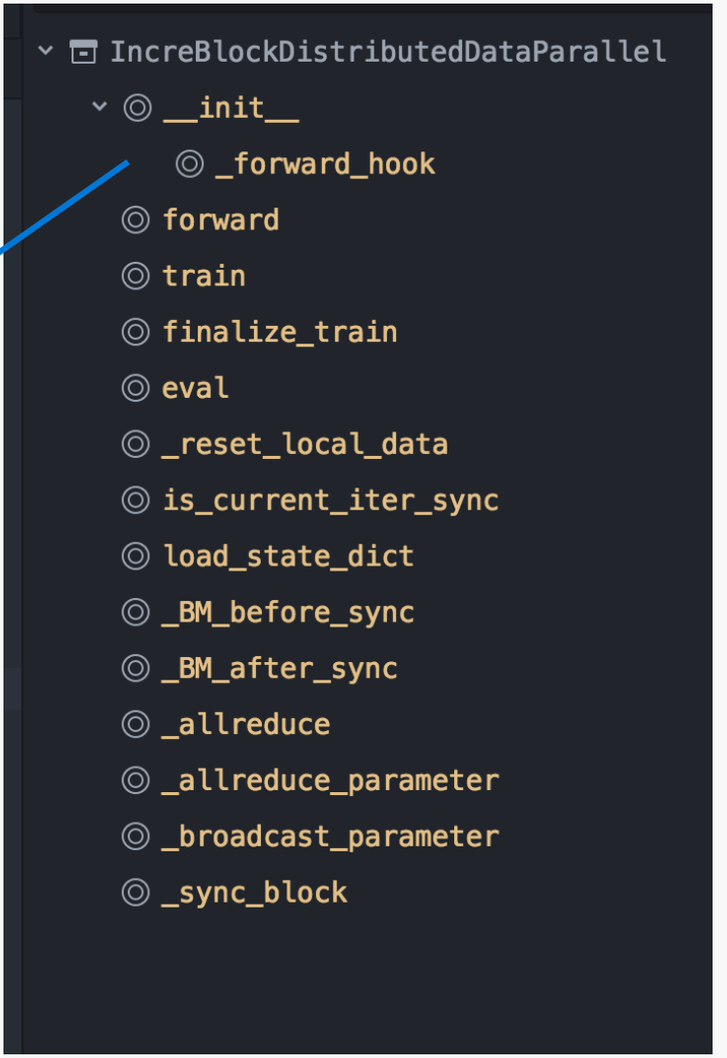
d. Get the overall gradient for each GPU
e. Compute global gradient using momentum
f. Update model & sync with all GPUs.

g. repeat c, d, e, f



BMUF implementation

```
def _forward_hook(module, input):
    if not self.training:
        return

    self.batch_id += 1
    if self.batch_id % self.block_sync_freq == 0:
        self._sync_block()
        # reset optimizer
        if self.reset_optimizer:
            pass  #...

self.register_forward_pre_hook(_forward_hook)
```

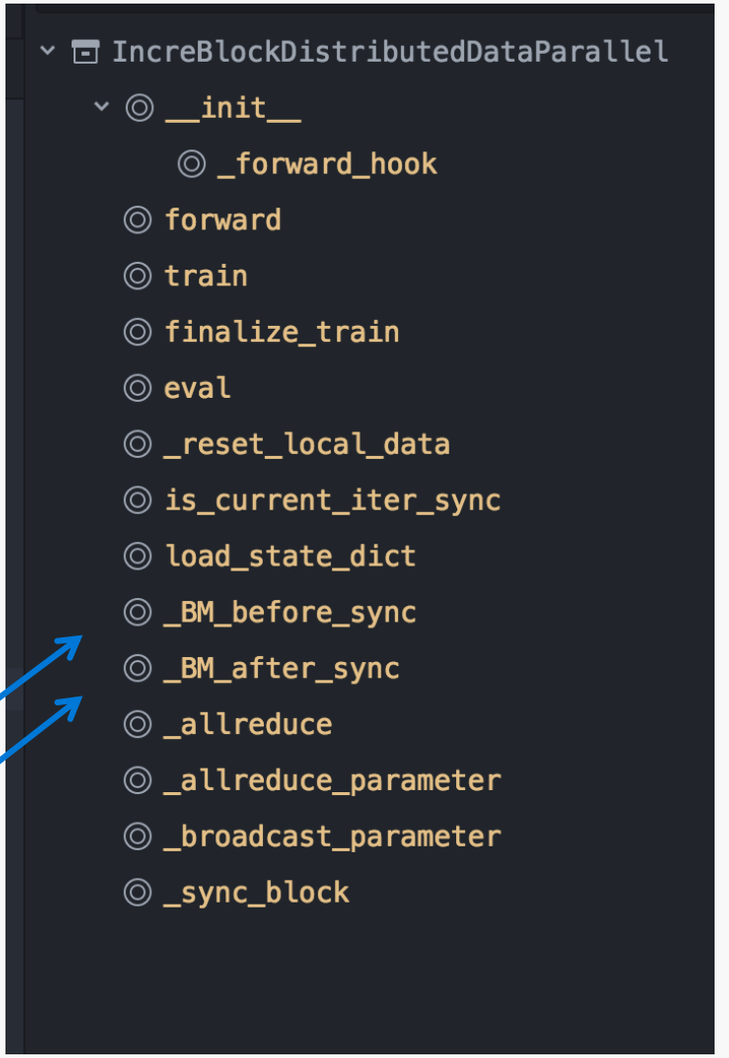


BMUF implementation

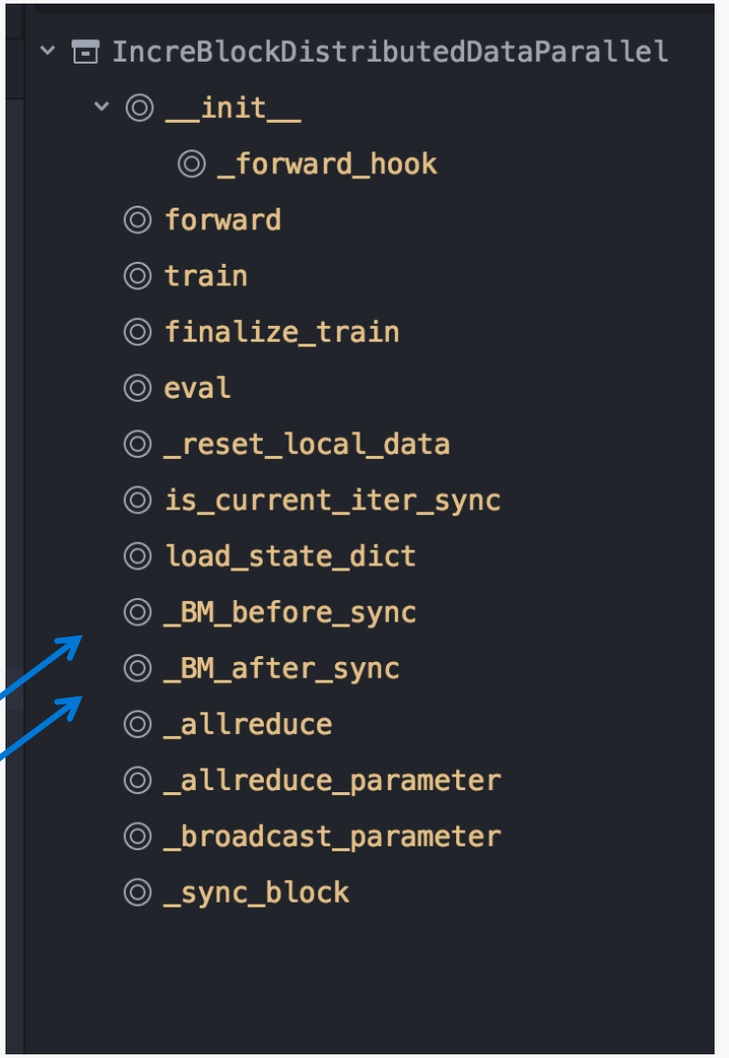
a. divide whole dataset into several blocks
b. Do warmup
c. For each GPU:
    i. train multiple iterations on assigned data.
    ii. update the local model in each iteration

d. Get the overall gradient for each GPU
e. Compute global gradient using momentum
f. Update model & sync with all GPUs.

g. repeat c, d, e, f



BMUF implementation

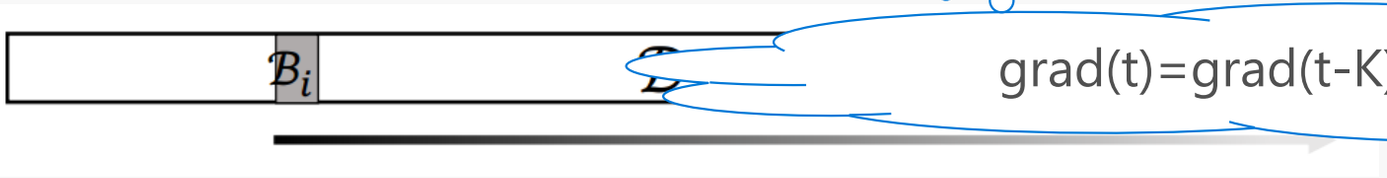


grad(t)=para(t)-para(t-**K**)

d. Get the overall gradient for each GPU

e. Compute global gradient using momentum

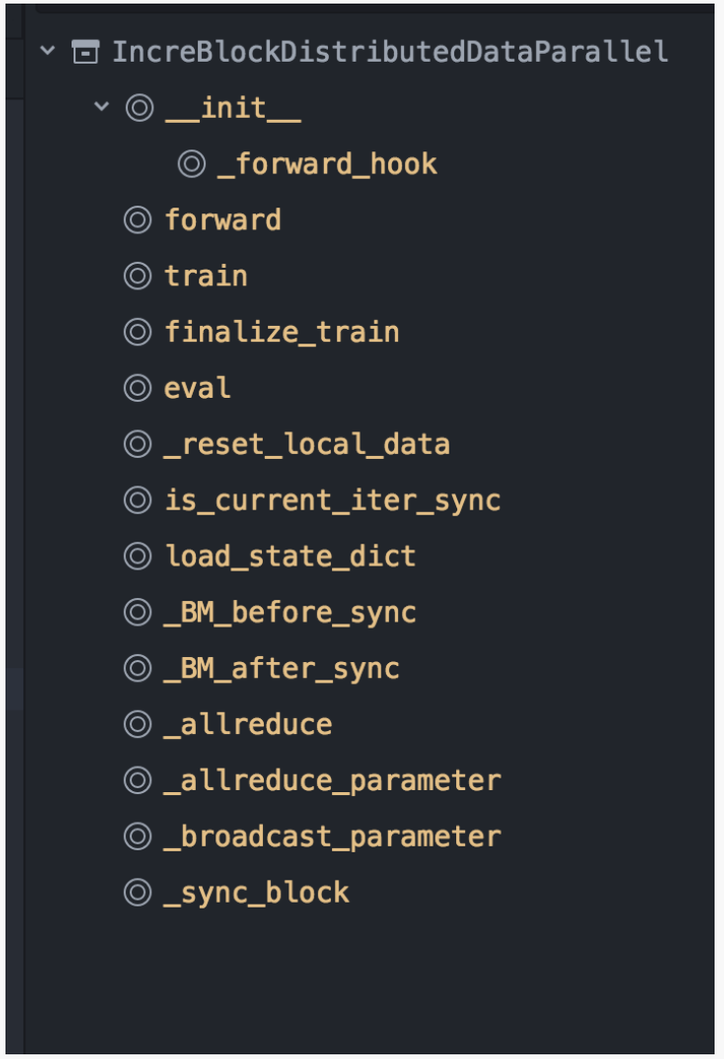
f. Update model & sync with all GPUs.



grad(t)=grad(t-K)*0.9+grad(t)*0.1

BMUF implementation

```python
def _allreduce_parameter(self):
    for param in self.module.parameters():
        sync_para = param.data if self.BM == 0 else self.param_grad_map[param]
        # do it here to reduce nan
        sync_para /= self.process_group.size()
        self._allreduce(sync_para, dist.ReduceOp.SUM)
```



d.

e. Compute **global gradient** using momentum

f.

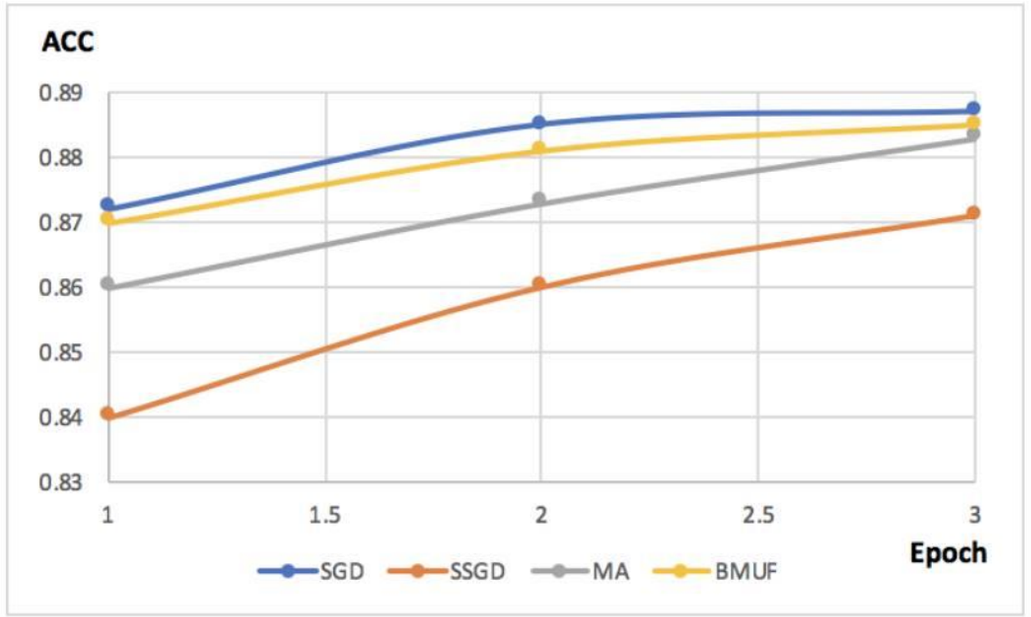
g.

Experiments

aloha - 1M (400 hrs, more comparisons)

- setup: 2048*2+500*2 batch:20
- final convergence result of SGD v.s. SSGD v.s. MA v.s. BMUF
  - BMUF is the best among them

	A	B	C	D
1	method	ngpu	acc	WER
2	SGD	1	0.887	34.1
3	SSGD	8	0.871	37.8
4	MA	16	0.883	35.4
5	BMUF	16	0.885	33.8



Experiments

aloha - 10M (8000 hrs)

- setup: 1400*2+700*2 batch:20
- num. of gpus
  - better results and better speedups

	A	B	C	D	E
1	method	ngpu	WER	SPEED (fr/sec)	speedups
2	SSGD	8	5.8	29630	6.2
3	BMUF	8	5.2	33103	6.9
4		16	5.1	63158	13.2
5		32	5.2	128000	26.7
6		64	5.2	246154	51.3

Experiments

- warmup
  - 10000-20000 iterations of warmup should be enough (batch=20)

	A	B
1	warmup (hour)	WER
2	96	35.6
3	192	33.8
4	288	33.1
5	384	33.3

- sync. frequency
  - if we use higher sync. frequency, the WER will be slightly better

	A	B
1	sync freq. (# of batches)	WER
2	20	33.9
3	90	34.9
4	160	35.1
5	230	36
6	300	36.5

Experiments

- momentum tuning (ngpu=16)
  - The best momentum can be (1-1/ngpu) according to https://docs.microsoft.com/en-us/cognitive-toolkit/multiple-gpus-and-machines#6-block-momentum-sgd

	A	B
1	momentum	WER
2	0.93	34.6
3	0.94	34.8
4	0.95	33.8
5	0.96	34
6	0.97	34.4

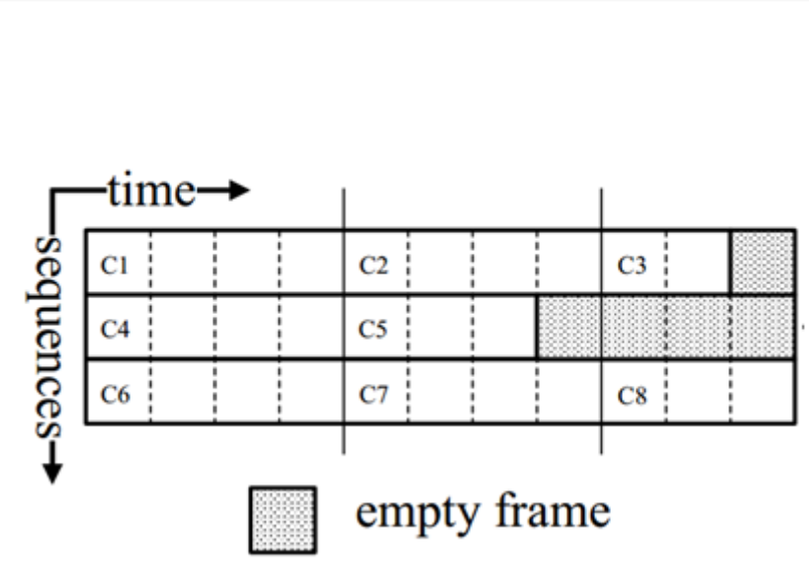
- local optimizer and the corresponding learning rate
  - we use adadelta for the local optimizer
  - slightly increasing the learning rate is better; reset the local optimizer in each gpu after the BM synchronization is better

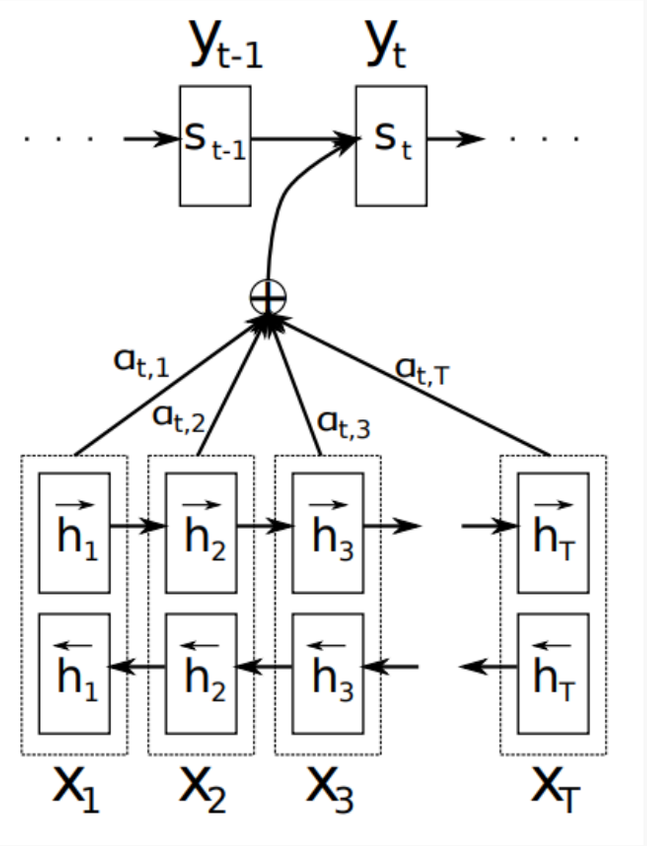
	A	B
1	learning rate	WER
2	1	33.8
3	1.5	33.6
4	2	33.2
5	reset optimizer	WER
6	TRUE	33.8
7	FALSE	35.3

Sequence-to-sequence model

- Widely used in machine translation and speech recognition

- Sequence batching + data parallel





Sequence-to-sequence model

- Widely used in machine translation and speech recognition