Lightweight and Efficient End-to-End Speech Recognition Using Low-Rank Transformer

ICASSP 2020


Genta Indra Winata


Samuel Cahyawijaya

Zhaojiang Lin

Zihan Liu


Pascale Fung

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

CAIRE
Center for Artificial Intelligence Research

Outline

- Background
- Preliminaries
- Low-Rank Transformer
- Experiment Setup
- Results and Analysis
- Conclusion

Background

Issues on Speech Recognition

- Speech recognition requires **large memory capacity**
- Large capacity is proportional to **high computational power and time** in training and inference, especially RNNs
- It is **ideal** to have **ASR run on low-end devices**, such as smartphone

Research Questions

- Can *smaller models* perform **better** than *larger models*?
- How to compress model **without any performance loss**? And **speedup training and inference** to save the computation cost?

Preliminaries

Low-Rank Matrix Factorization

Model Compression

End-to-End Speech Recognition

Low-Rank Matrix Factorization

A large matrix can be decomposed into two smaller matrices, where the rank of the matrices is smaller than the dimension of the original matrix.

$$\underset{m \times n}{\mathbf{W}} = \underset{m \times r}{\mathbf{U}} \quad \underset{r \times n}{\mathbf{V}}$$

Computation advantages:

- Produce compact and dense matrices
- Reducing flops from $m \times n \rightarrow (m+n)r$
- Compressing the model size $m \times n \rightarrow (m+n)r$

Non-negative Matrix Factorization

NMF algorithms aim at finding a rank r approximation of the form.

$$\underset{m\times n}{\mathbf{W}} = \underset{m\times r}{\mathbf{U}} \quad \underset{r\times n}{\mathbf{V}},$$

$$\underset{U,V}{\text{minimize}} \quad ||\mathbf{W} - \mathbf{UV}||_F^2.$$

where W and U are non-negative matrices of dimensions m x r and r x n, respectively.

Preliminaries

Low-Rank Matrix Factorization

Model Compression

End-to-End Speech Recognition

Model Compression

In-Training

- Reduce the training time and memory cost
- The model is trained to learn compact representations

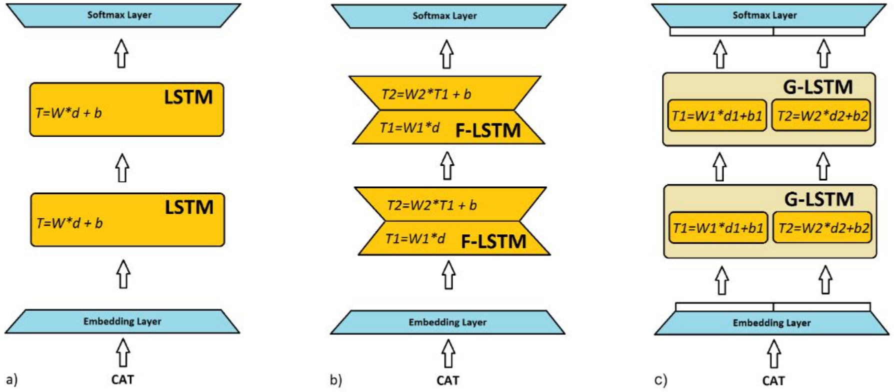
Post-Training

- Large model training may have bottlenecks in time and speed
- Useful for pre-trained models
- An approximation of the original model

In-Training Factorized LSTM (Kuchaiev and Ginsburg, 2017)

The model accelerates the training of LSTM. Apply matrix factorization by design.

The model improves the speed of training and inference with a small performance loss.

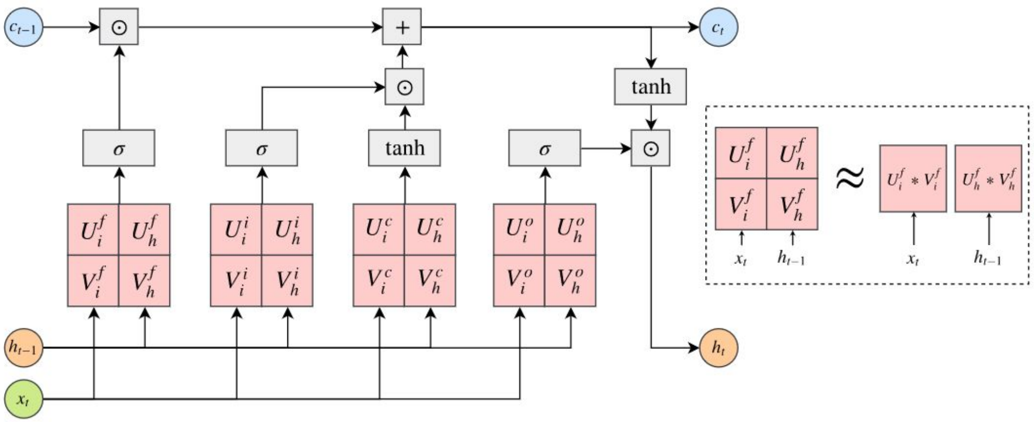


Kuchaiev, Oleksii and Ginsburg, Boris, Factorization tricks for lstm networks, ICLR Workshop, 2017.

Post-Training Factorized LSTM (Winata, et al. 2019)

A comprehensive comparison of post-training methods on LSTM on language model and downstream NLP tasks.

Low-Rank Matrix Factorization generally achieves better than pruning.



Winata, G.I., Madotto, A., Shin, J., Barezi, E.J. and Fung, P., 2019. On the effectiveness of low-rank matrix factorization for lstm model compression. *PACLIC, Hakodate, Japan*

Preliminaries

Low-Rank Matrix Factorization

Model Compression

End-to-End Speech Recognition

End-to-End Speech Recognition

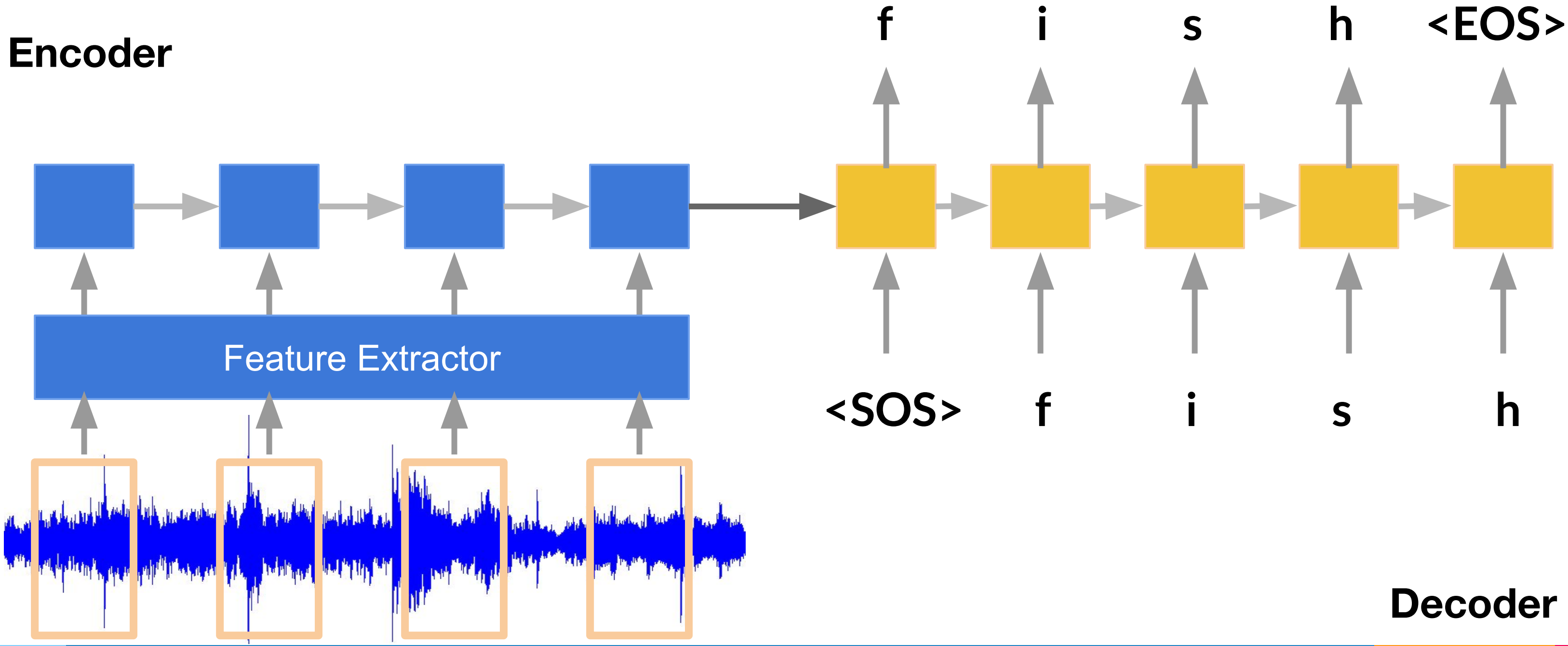
There are three main end-to-end sequence-to-sequence ASR architectures:

- RNN-based models with attention (Chan, et al 2016)
- Transformer-based model, a fully-attentional feed-forward architecture (Dong, et al 2018)
- Hybrid Attention-CTC (Kim, et al 2016; Hori, et al 2017)

- Chan, W., Jaitly, N., Le, Q. and Vinyals, O., 2016, March. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 4960-4964). IEEE.
- Dong, L., Xu, S. and Xu, B., 2018, April. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 5884-5888). IEEE.

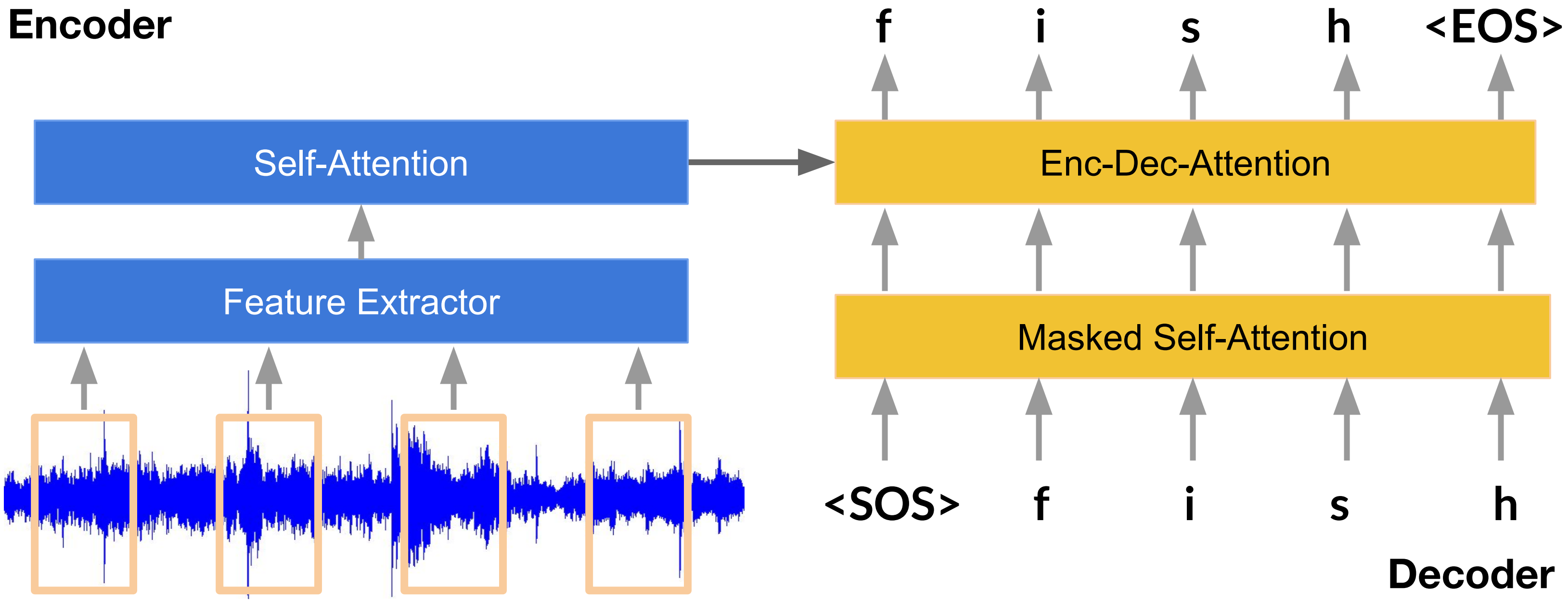
RNN with Attention Model (Chan, et al 2016)

The encoder processes the audio input and the decoder generates the transcription.



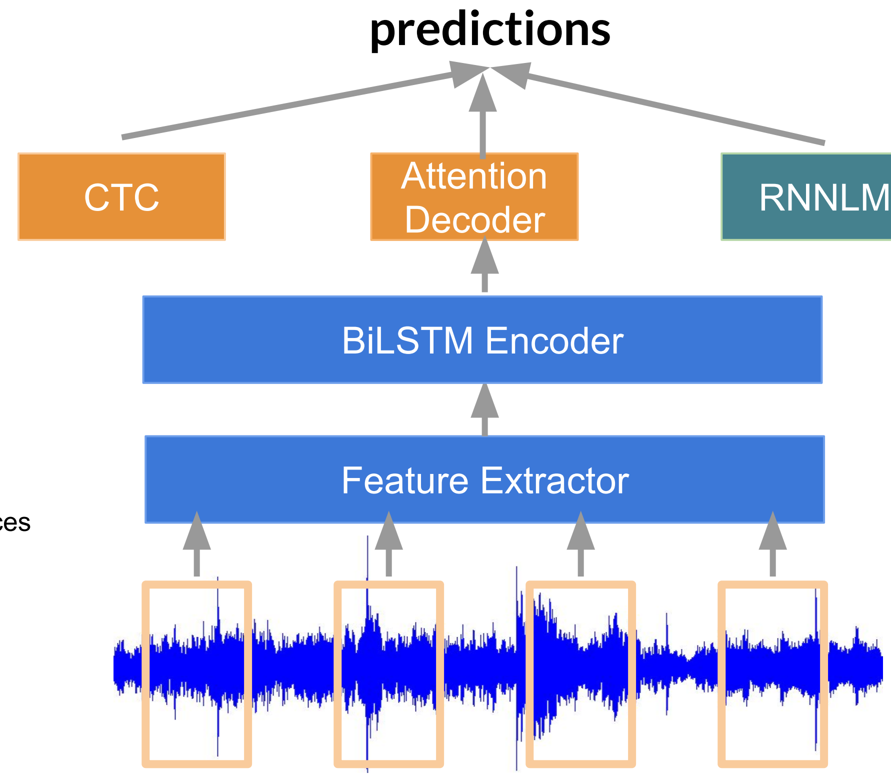
Transformer Model (Dong, et al 2018)

Remove the recurrence and apply attention to speed up the training and inference



Joint CTC-Attention Model (Kim, et al 2016; Hori, et al 2017)

Joint train with multiple objectives.



- Hori, T., Watanabe, S., Zhang, Y. and Chan, W., 2017. Advances in Joint CTC-Attention based End-to-End Speech Recognition with a Deep CNN Encoder and RNN-LM.
- Kim, S., Hori, T. and Watanabe, S., 2017, March. Joint CTC-attention based end-to-end speech recognition using multi-task learning. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)* (pp. 4835-4839). IEEE.

Low-Rank Transformer

A lightweight and efficient transformer

Low Rank Transformer (LRT)

- A factorized transformer-based model architecture
- Replacement large high-rank matrices with low-rank matrices to eliminate the computational bottlenecks.

Objective

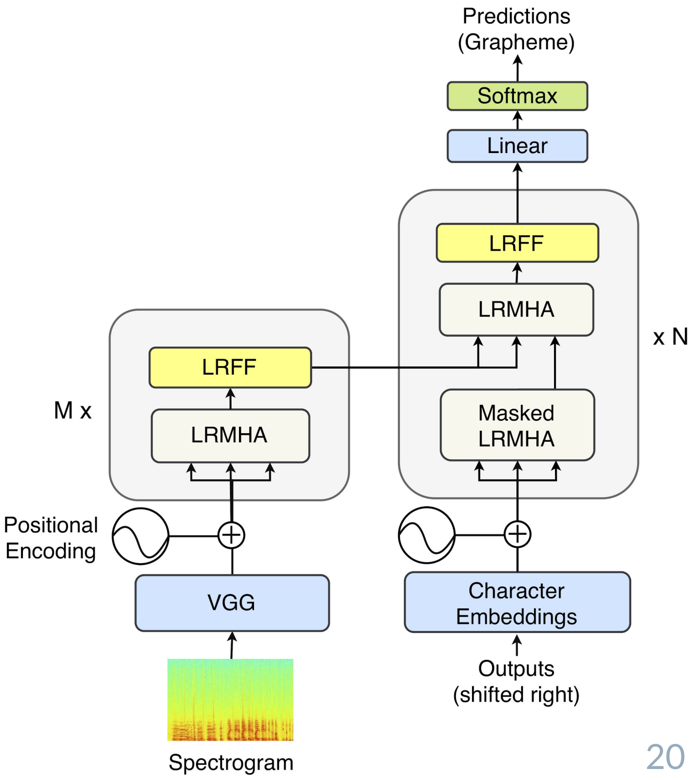
Predict graphemes given audio inputs

Model Architecture

Input Encoder: VGG Encoder

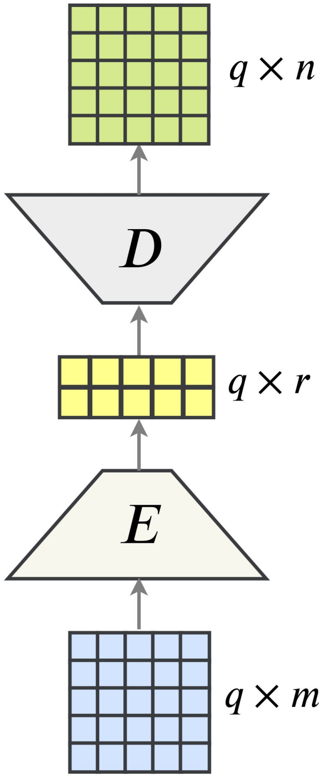
Components:

- Low-Rank Multi-Head Attention (LRMHA)
- Low-Rank Feed Forward Network (LRFF)



Linear Encoder-Decoder (LED) Unit

Each m x n matrix is approximated by the multiplication of the linear encoder unit and a linear decoder unit.



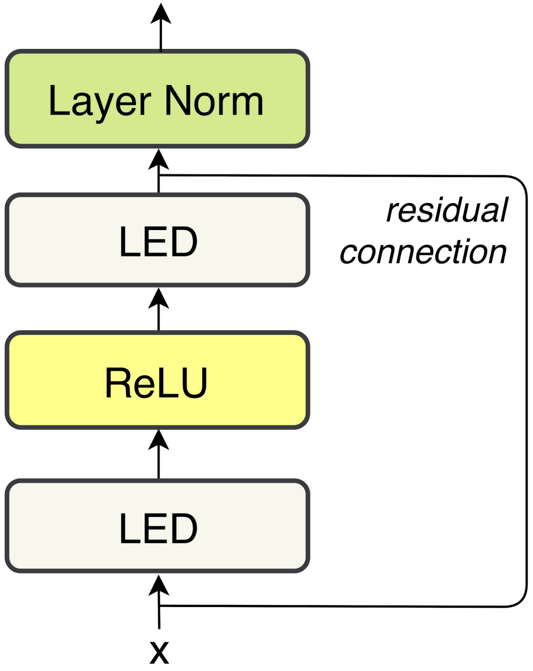
If $r << \{m, n\}$:

- **Less parameters** compared to linear layer
- **Better generalization** due to the bottleneck layer
- **Faster training** with less flops

Low Rank Feed Forward (LRFF)

- Two LED units
- Residual connection
- Layer normalization

$$g(x) = \text{LayerNorm}(\max(0, xE_1D_1)E_2D_2 + x),$$

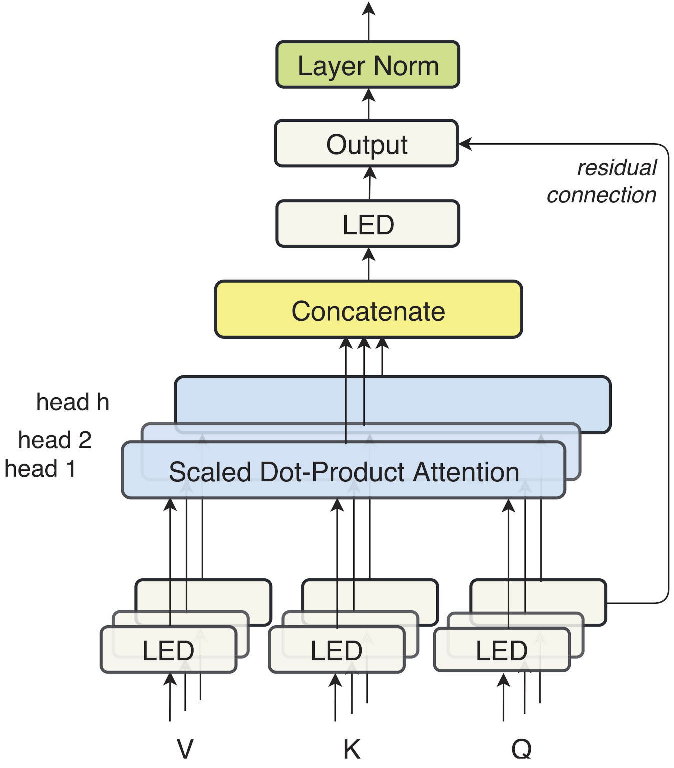


Low Rank Multi Head Attention (LRMHA)

- Utilize LED units
  - **Faster** Q, K, V projection
  - Attention **regularization**

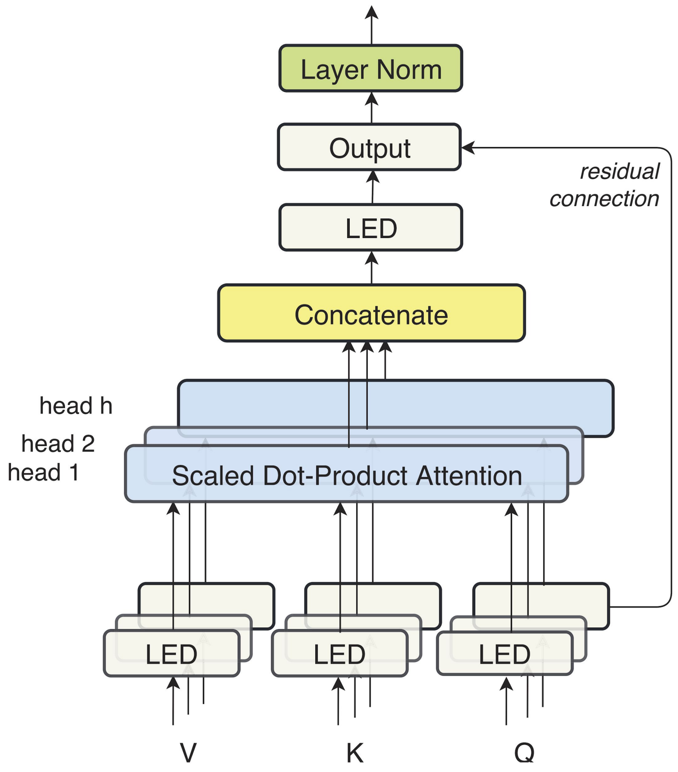
$$\text{Attention}(Q, K, V) = \text{Softmax}(\frac{QK^T}{\sqrt{d_k}}V),$$

$$hd_i = \text{Attention}(QE_i^Q D_i^Q, KE_i^K D_i^K, VE_i^V D_i^V),$$
$$f(Q, K, V) = \text{Concat}(h_1, \cdots, h_H)E^O D^O + Q,$$



Low Rank Multi Head Attention (LRMHA)

- Utilize LED units
  - **Faster** Q, K, V projection
  - Attention **regularization**
- Residual connection
  - To avoid gradient issues
- Layer Normalization



Experimental Setup

Datasets

AiShell-1

- A multi-accent Mandarin Chinese speech dataset.
- Consists of 150 hours, 10 hours, and 5 hours of training, validation, and testing, respectively.

HKUST

- A conversational telephone Chinese speech recognition dataset.
- Consists of 152 hours, 4.2 hours, and 5 hours of training, validation, and testing, respectively.

Baseline

- **Transformer-based model**

- **3 different model size**
  - Small
  - Medium
  - Large

- **Different model size per dataset** due to different embedding size

Dataset	Model Name	# Param
AiShell-1	Transformer (Small)	≈7.8M
	Transformer (Medium)	≈11.5M
	Transformer (Large)	≈22M
HKUST	Transformer (Small)	≈8.7M
	Transformer (Medium)	≈12.7M
	Transformer (Large)	≈25.1M

Training Phase

We train all characters in the corpus, including <PAD>,<SOS>, and <EOS>. The model consists of 2 encoder layers and 4 decoder layers.

The uncompressed Transformer (Large) has a dim_{inner} of 2048, dim_{model} of 512, and dim_{emb} of 512. We select the same parameters as the LRT model with r= 100, r= 75 and r= 50.

Inference Phase

We generate the predictions using a beam-search decoding, we take

α= 1, γ= 0.1, and a beam size of 8.

$$P(Y) = \alpha P_{trans}(Y|X) + \gamma\sqrt{wc(Y)},$$

We evaluate our model using a single GeForce GTX 1080Ti GPU and

three Intel Xeon E5-2620 v4 CPU cores. wc(Y) is the word count to

avoid generating very short/long sentences

Results and Analysis

Results on AiShell-1 Dataset

LRT model **outperforms all baseline models** with the same number of parameters.

LRT achieves better performance with around **60% compression rate** compared to baseline **Transformer (Large)** model

Model	Params	CER
Hybrid approach		
HMM-DNN [12]	-	8.5%
End-to-end approach		
Attention Model [13]	-	23.2%
+ RNNLM [13]	-	22.0%
CTC [14]	≈11.7M	19.43%
Framewise-RNN [14]	≈17.1M	19.38%
ACS + RNNLM [13]	≈14.6M	18.7%
Transformer (large)	25.1M	13.49%
Transformer (medium)	12.7M	14.47%
Transformer (small)	8.7M	15.66%
LRT ($r = 100$)	12.7M	**13.09%**
LRT ($r = 75$)	10.7M	13.23%
LRT ($r = 50$)	8.7M	13.60%

Results on HKUST Dataset

LRT model **outperforms all baseline models** with the same number of parameters.

LRT achieves better performance with around **60% compression rate** compared to baseline **Transformer (Large)** model

Model	**Params**	**CER**
Hybrid approach		
DNN-hybrid [12]	-	35.9%
LSTM-hybrid (with perturb.) [12]	-	33.5%
TDNN-hybrid, lattice-free MMI (with perturb.) [12]	-	28.2%
End-to-end approach		
Attention Model [12]	-	37.8%
CTC + LM [15]	≈12.7M	34.8%
MTL + joint dec. (one-pass) [12]	≈9.6M	33.9%
+ RNNLM (joint train) [12]	≈16.1M	32.1%
Transformer (large)	22M	29.21%
Transformer (medium)	11.5M	29.73%
Transformer (small)	7.8M	31.30%
LRT ($r = 100$)	11.5M	**28.95%**
LRT ($r = 75$)	9.7M	29.08%
LRT ($r = 50$)	7.8M	30.74%

Memory and Time Efficiency

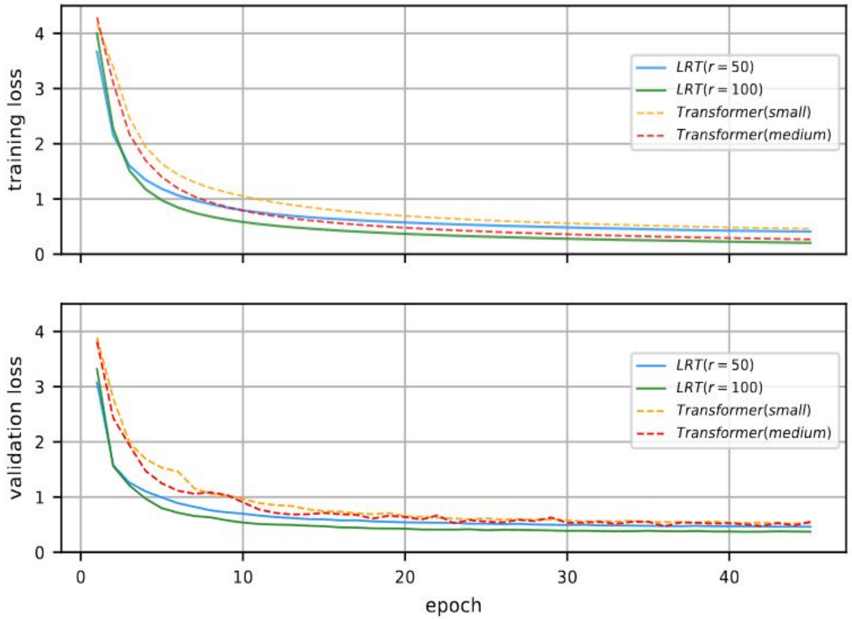
Our LRT models gain inference time speed-up by up to 1.35x in the GPU and 1.23x in the CPU, compared to the uncompressed Transformer (large) baseline model.

dataset	r	ΔCER	compress.	speed-up		$\lvert\bar{X}\rvert$
				GPU	CPU only	
AiShell-1	base	0	0	1	1	23.08
	100	0.40%	49.40%	1.17x	1.15x	23.15
	75	0.26%	57.37%	1.23x	1.16x	23.17
	50	-1.10%	65.34%	1.30x	1.23x	23.19
HKUST	base	0	0	1	1	22.43
	100	0.26%	47.72%	1.21x	1.14x	22.32
	75	0.13%	55.90%	1.26x	1.15x	22.15
	50	-1.53%	64.54%	1.35x	1.22x	22.49

LRT Training Convergence

LRT model is more stable to train and convergences faster in just around **15 epochs**.

LRT model achieves lower training & validation loss compared to the baseline model with the same number of parameters



Conclusion

Let’s answer our questions

- Can *smaller models* perform **better** than *larger models*?

Yes, it is! With the better approach, smaller models can not only performs better but also faster than larger models!

Let’s answer our questions

- Can *smaller models* perform **better** than *larger models*?

Yes, it is! With the better approach, smaller models can not only performs better but also faster than larger models!

- How to compress model **without any performance loss**? And **speedup training and inference** to save the computation cost?

In-training compression, LRT!

Conclusion

- LRT is a **memory-efficient** with **faster-computational** neural architecture that eliminate the memory and time bottlenecks.
- LRT can **generalize better** on test set while also **reducing** the **parameters by 50%.**
- LRT is **faster to converge** compared to normal transformer model.

Thank you

All questions are welcome