Concept to Code: Semi-Supervised End-To-End Approaches For Speech Recognition

Omprakash Sonie
omprakash.s@flipkart.com

Venkateshan Kannan
ventangled@gmail.com

Goal:

- Provide foundation and relevant self & semi-supervised learning methods for ASR
- Provide learning from executing ‘Supervised to Semi-Supervised’ Case Study

Agenda: Part-I

- Foundation:
    - Speech, Dataset
    - RNNs, CNNs and Transformer comparison
    - End to end Convolution based model
    - Sequence to Sequence based acoustic model
    - Transformer based acoustic model
- Relevant approaches
    - Self training for end-to-end ASR
    - Semi-supervised with Word Selection
- Case Study
    - Supervised to Semi-Supervised
    - Practical tips

Case Study

END-TO-END ASR: FROM SUPERVISED TO SEMI-SUPERVISED LEARNING WITH MODERN ARCHITECTURES

A PREPRINT

Gabriel Synnaeve*
Facebook, NYC
gab@fb.com

Qiantong Xu*
Facebook, Menlo Park
qiantong@fb.com

Jacob Kahn*
Facebook, Menlo Park
jacobkahn@fb.com

Edouard Grave*
Facebook, Paris
egrave@fb.com

Tatiana Likhomanenko
Facebook, Menlo Park
antares@fb.com

Vineel Pratap
Facebook, Menlo Park
vineelkpratap@fb.com

Anuroop Sriram
Facebook, Menlo Park
anuroops@fb.com

Vitaliy Liptchinsky
Facebook, Menlo Park
vitaliy888@fb.com

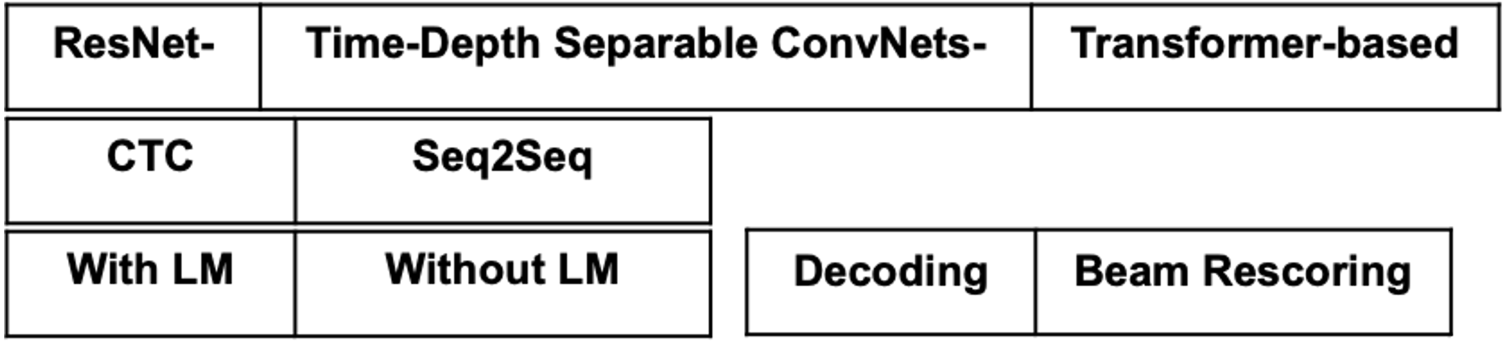
Ronan Collobert*
Facebook, Menlo Park
locronan@fb.com

[10] Facebook https://arxiv.org/pdf/1911.08460.pdf

End-to-End ASR: From Supervised to Semi Supervised Learning with Modern Architecture

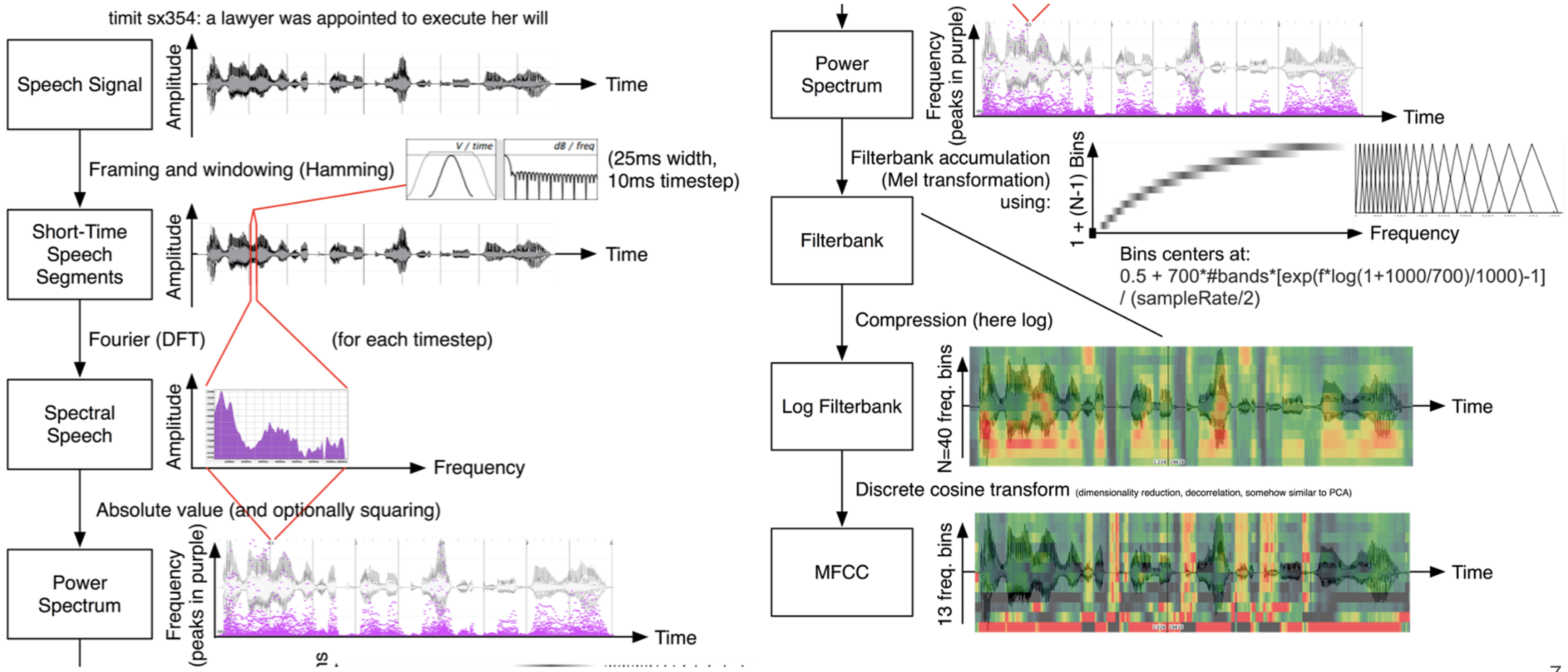
In this paper consider

- **ResNet-**, **Time-Depth Separable ConvNets-**, and **Transformer-based** acoustic models,
- Trained with **CTC** or **Seq2Seq** criterions.
- Perform experiments on the LIBRISPEECH dataset 960hrs **test-other**
- **with** and **without LM decoding**, optionally with **beam rescoring**.



Quick recap - Foundation

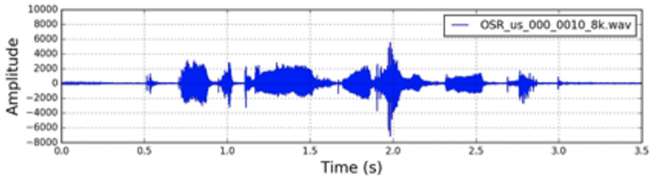
Speech Signal Processing

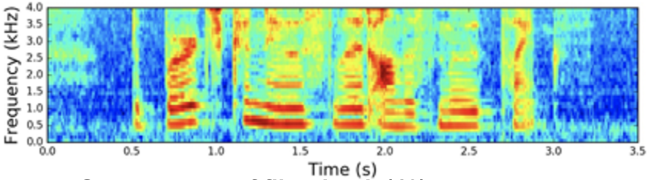


ASR Experiments - Feature Engineering

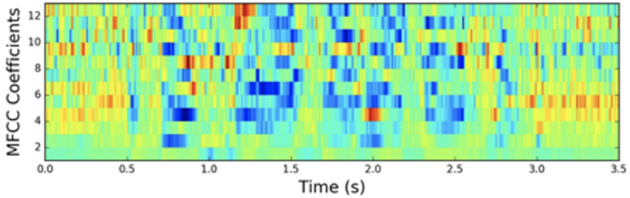
Spectrogram: Used with DNN, CNN

MFCC: GMM-HMM models





Spectrogram of filter bank (40)



39 MFCC: Mel-Frequency Cepstral Coefficients
13 Mel cepstral coefficients + 13 first derivatives + 13 second derivatives

Audio signal:
20/25ms (10ms stride)
30ms, 50ms, 32ms, 100ms (5ms shift)

Filters:
40, 80, 120 Filter bank
Gammatone filters

Feature Extraction:
Conv-1D, MaxPooling
VGGNet

Features from Raw waveform: SincNet

Augmentation:
Vary the speed, Time, Frequency,
Add various types of noises and audio

1

LIBRISPEECH: AN ASR CORPUS BASED ON PUBLIC DOMAIN AUDIO BOOKS

Vassil Panayotov, Guoguo Chen*, Daniel Povey*, Sanjeev Khudanpur*

*Center for Language and Speech Processing & Human Language Technology Center of Excellence
The Johns Hopkins University, Baltimore, MD 21218, USA

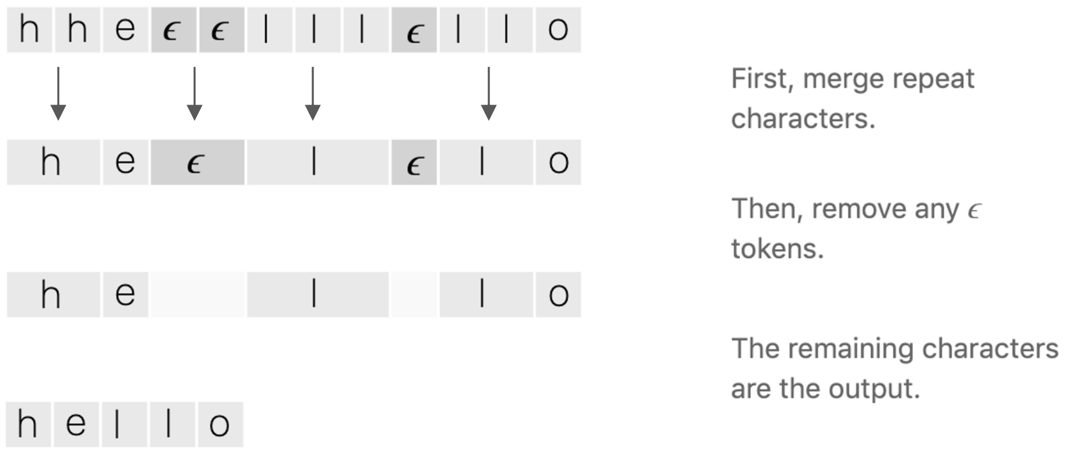
[1] LibriSpeech- An ASR Corpus Based On Public Domain Audio Books

Dataset - Librispeech

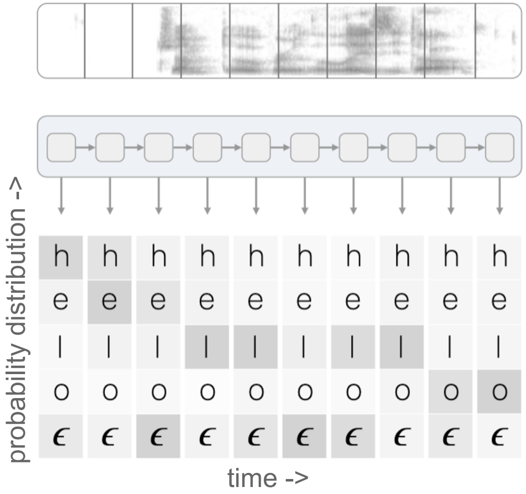
File	Size *.tar.gz	Audio files	Hours	Speakers	Wav2letter lst/
dev-clean	337.9 MB	**2,703**	5.4	**41**	2703
dev-other	314.3 MB	2,864	5.3	34	2864
test-clean	346.6 MB	**2,620**	5.4	**41**	2620
test-other	328.7 MB	2,939	5.1	34	2939
train-clean-100	6,387 MB	**28,539**	100.6	**252**	28,539
train-clean-360	23,049 MB	104,014	363.6	922	104,014
train-other-500	30,593 MB	148,688	496.7	1167	148,688
All 7 above					292,367
train-all-960	**~60 GB**	281,241	**960**		281,241

CTC Loss Function

- How do you distinguish between double ‘L’ or ‘L’ which was spoken for longer duration?
- CTC introduces blank token between symbols and in the beginning and end of the utterance



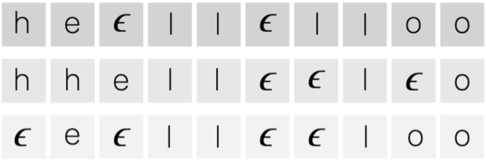
CTC Loss Function



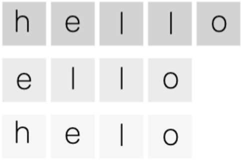
Convert input sequence of audio to spectrogram

Feed it to RNN/Transformer/MLP

The network give probability pt(a | X) distribution over all symbols for each input step
- From start to end



Compute the probability of different sequences/alignment



Marginalise over alignments to get distribution over outputs

2

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

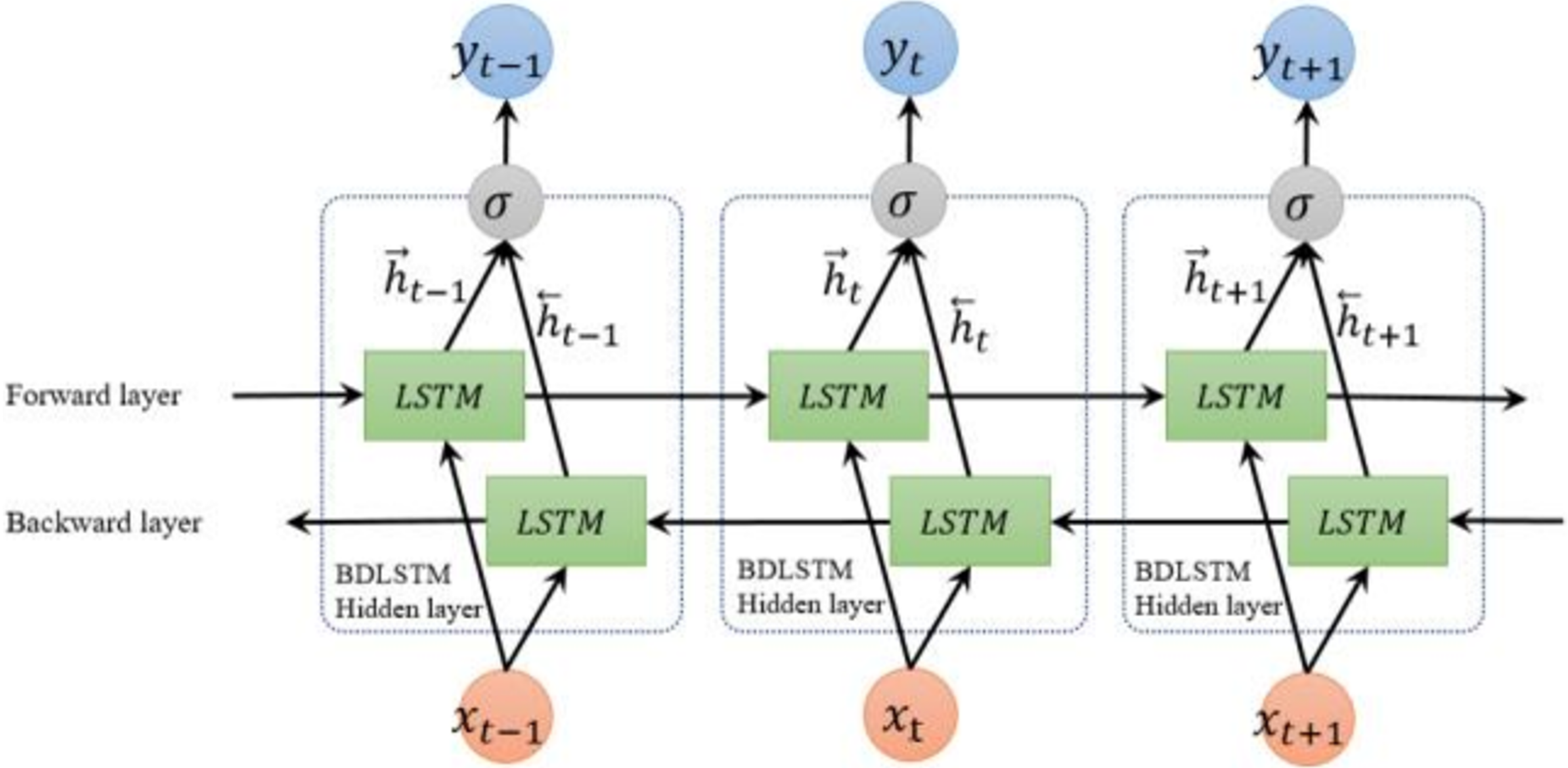
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

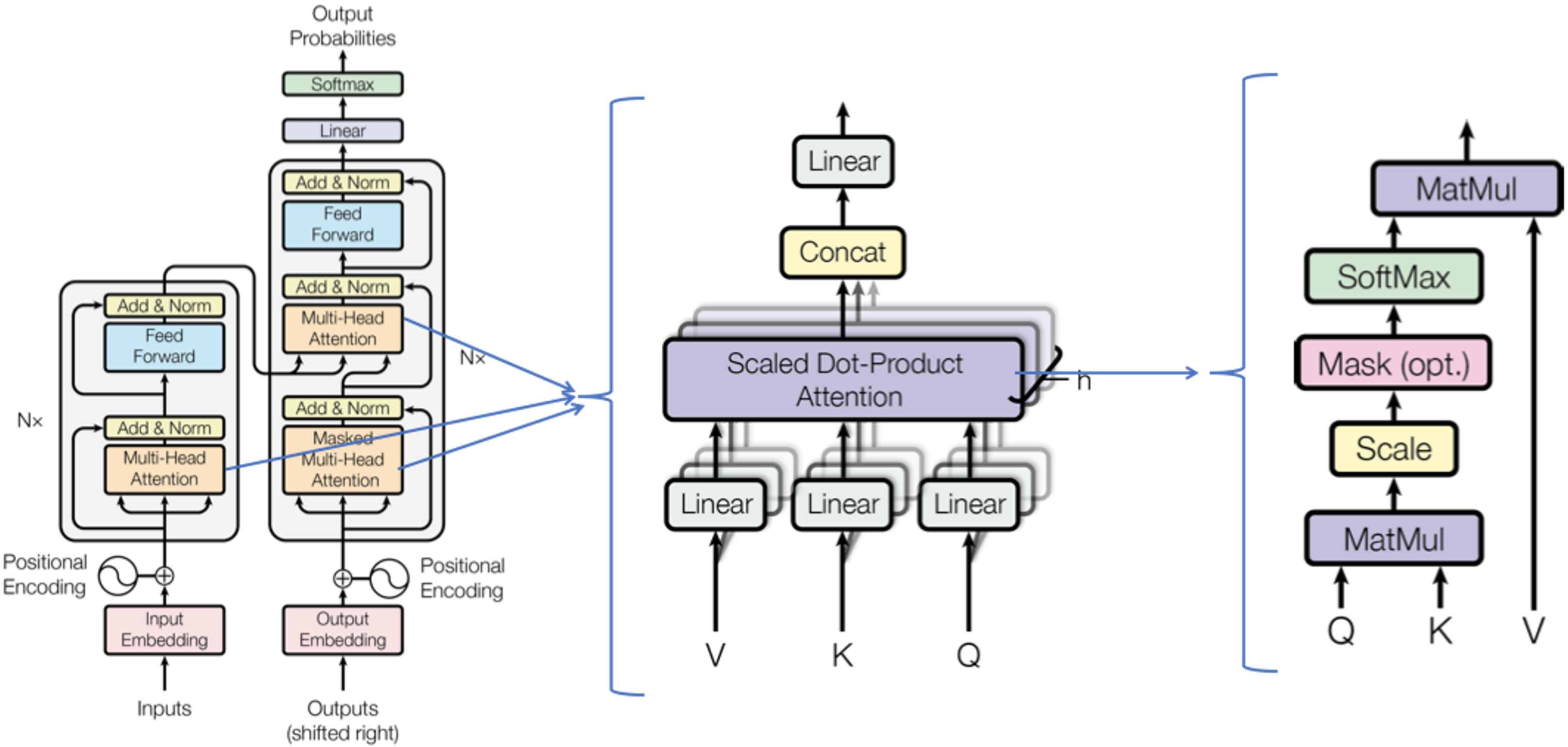
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

[2] Ashish Vaswani et al. Attention is all you need. Adv. NIPS, 2017. https://arxiv.org/pdf/1706.03762.pdf

RNN-Bi-LSTM (Recap)



Transformer (Recap)



3

A COMPARATIVE STUDY ON TRANSFORMER VS RNN IN SPEECH APPLICATIONS

Shigeki Karita[1],
(Alphabetical Order) *Nanxin Chen*[3], *Tomoki Hayashi*[5,6], *Takaaki Hori*[7], *Hirofumi Inaguma*[8], *Ziyan Jiang*[3], *Masao Someki*[5], *Nelson Enrique Yalta Soplin*[2], *Ryuichi Yamamoto*[4], *Xiaofei Wang*[3], *Shinji Watanabe*[3], *Takenori Yoshimura*[5,6], *Wangyou Zhang*[9]
[1]NTT Communication Science Laboratories, [2]Waseda University, [3]Johns Hopkins University,
[4]LINE Corporation, [5]Nagoya University, [6]Human Dataware Lab. Co., Ltd.,
[7]Mitsubishi Electric Research Laboratories, [8]Kyoto University, [9]Shanghai Jiao Tong University

[3] A Comparative Study on Transformers vx RNN in Speech Applications IEEE Automatic Speech Recognition and Understanding Workshop 2019.
https://arxiv.org/pdf/1909.06317.pdf

Transformer vs RNN

Transformers require more complex configurations

- optimizer
- network structure
- data augmentation

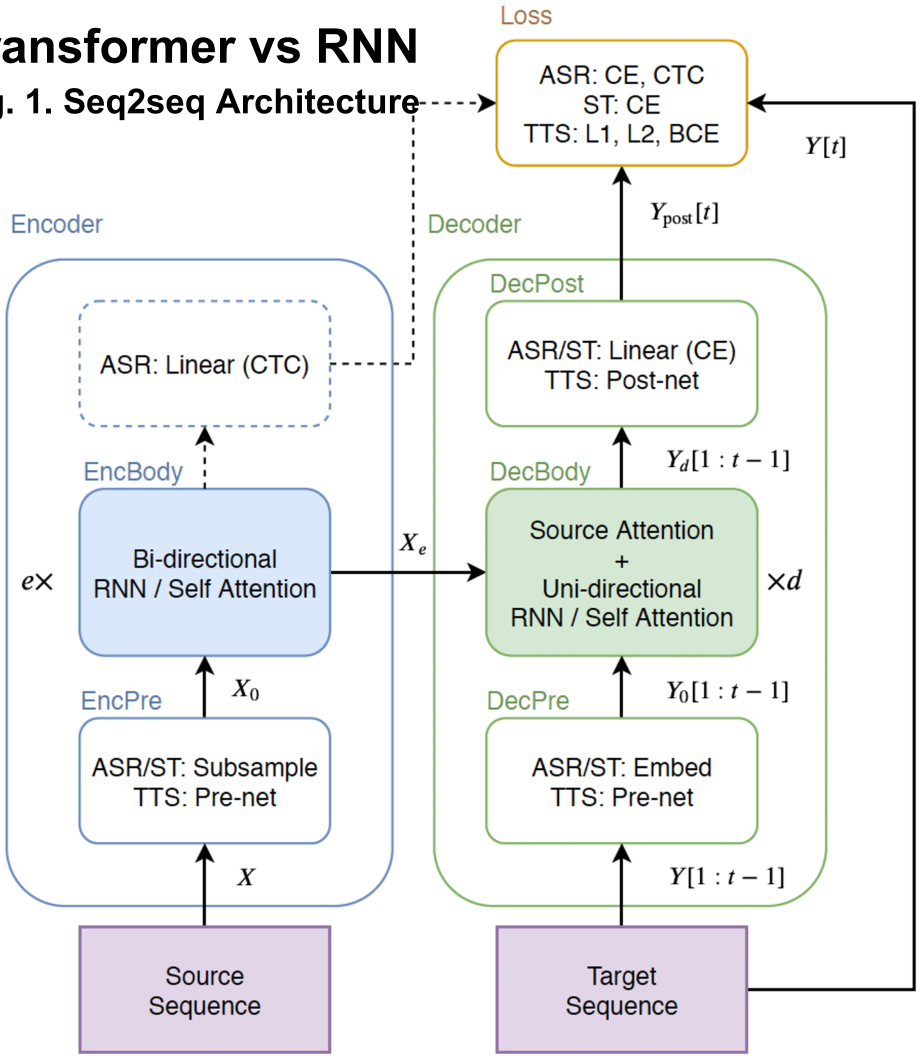
than the conventional RNN based models.

Provide practical guides for tuning Transformer in speech tasks to achieve state-of-the-art results

[3] A Comparative Study on Transformers vx RNN in Speech Applications IEEE ASR and Understanding Workshop 2019.

Transformer vs RNN

Fig. 1. Seq2seq Architecture



Encoder:

$$X_0 = \mathrm{EncPre}(X),$$
$$X_e = \mathrm{EncBody}(X_0),$$

e - no. of layers in EncoderBody

Decoder:

$$Y_0[1:t-1] = \mathrm{DecPre}(Y[1:t-1]),$$
$$Y_d[t] = \mathrm{DecBody}(X_e, Y_0[1:t-1]),$$
$$Y_{\mathrm{post}}[1:t] = \mathrm{DecPost}(Y_d[1:t]),$$

d - no. of layers in DecoderBody

t - target frame index

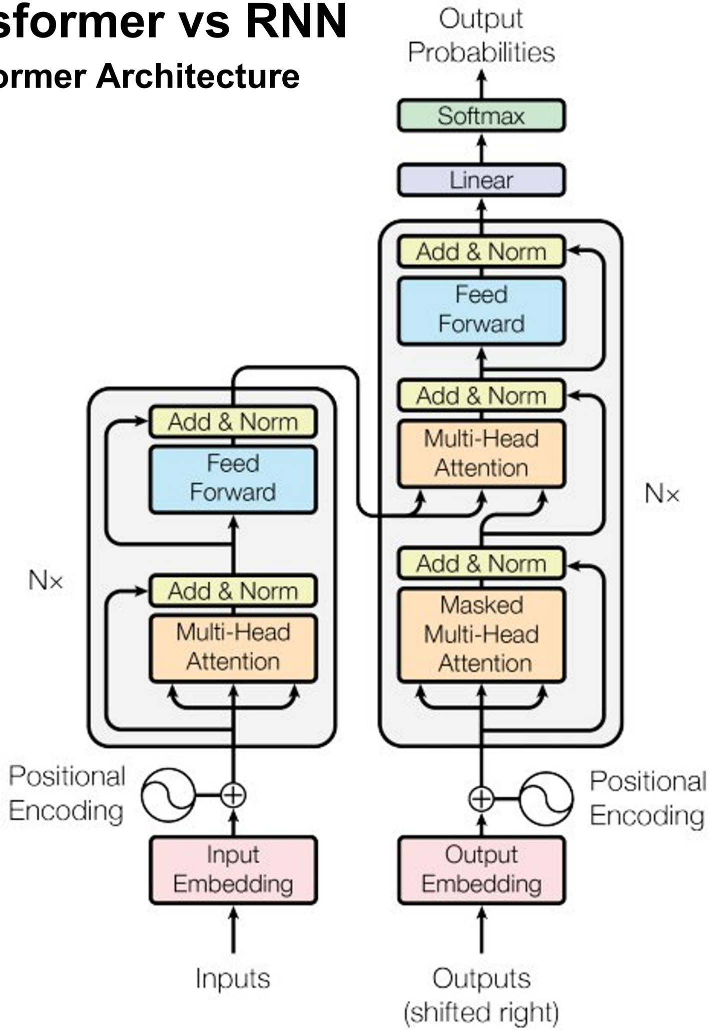
Loss:

$$L = \mathrm{Loss}(Y_{\mathrm{post}}, Y)$$

[3] A Comparative Study on Transformers vx RNN in Speech Applications IEEE ASR and Understanding Workshop 2019.

Transformer vs RNN

Transformer Architecture



Self-attention Encoder:

$$X_i' = X_i + \mathrm{MHA}_i(X_i, X_i, X_i),$$

$$X_{i+1} = X_i' + \mathrm{FF}_i(X_i'),$$

$$\mathrm{FF}(X[t]) = \mathrm{ReLU}(X[t]W_1^{\mathrm{ff}} + b_1^{\mathrm{ff}})W_2^{\mathrm{ff}} + b_2^{\mathrm{ff}},$$

Self-attention Decoder:

$$Y_j[t]' = Y_j[t] + \mathrm{MHA}_j^{\mathrm{self}}(Y_j[t], Y_j[1:t], Y_j[1:t]),$$

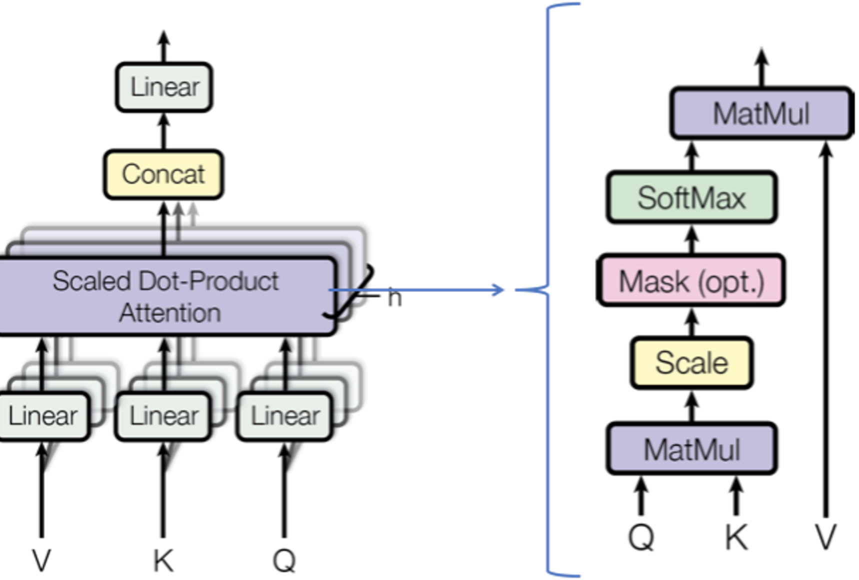
$$Y_j'' = Y_j + \mathrm{MHA}_j^{\mathrm{src}}(Y_j', X_e, X_e),$$

$$Y_{j+1} = Y_j'' + \mathrm{FF}_j(Y_j''),$$

[3] A Comparative Study on Transformers vx RNN in Speech Applications IEEE ASR and Understanding Workshop 2019.

Transformer vs RNN

Transformer Architecture



Multi-Head Attention:

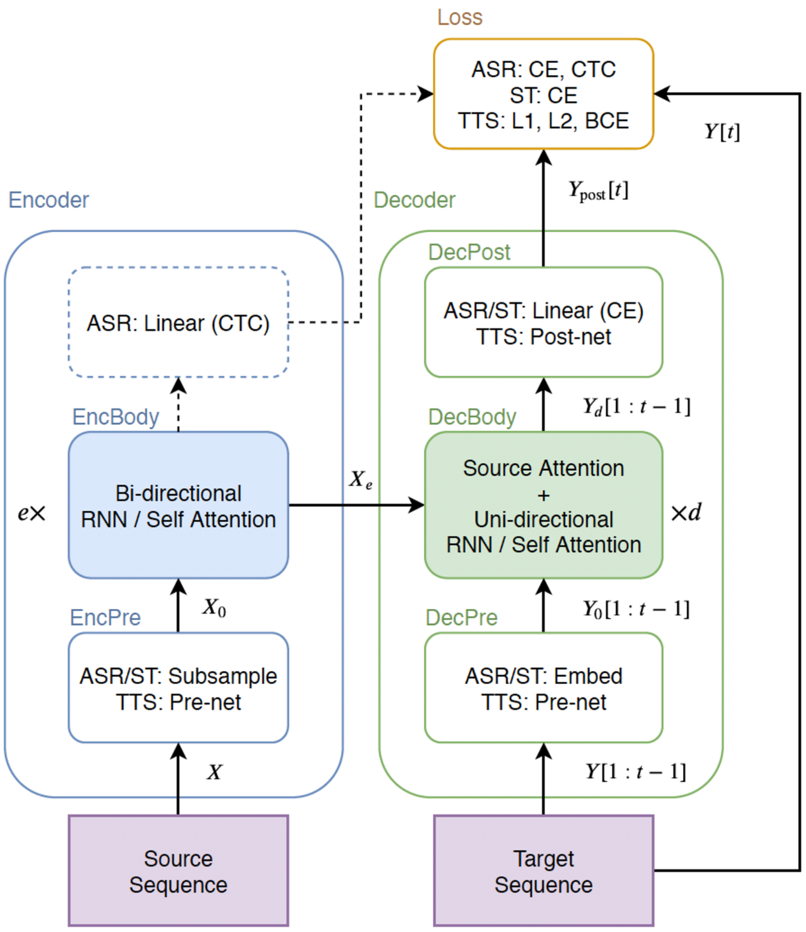
$$\mathrm{MHA}(Q,K,V) = [H_1, H_2, \dots, H_{d^{\mathrm{head}}}]W^{\mathrm{head}},$$

$$H_h = \mathrm{att}(QW_h^{\mathrm{q}}, KW_h^{\mathrm{k}}, VW_h^{\mathrm{v}}),$$

$$\mathrm{att}(X^{\mathrm{q}}, X^{\mathrm{k}}, X^{\mathrm{v}}) = \mathrm{softmax}\left(\frac{X^{\mathrm{q}}X^{\mathrm{k}\top}}{\sqrt{d^{\mathrm{att}}}}\right)X^{\mathrm{v}},$$

[3] A Comparative Study on Transformers vx RNN in Speech Applications IEEE ASR and Understanding Workshop 2019.

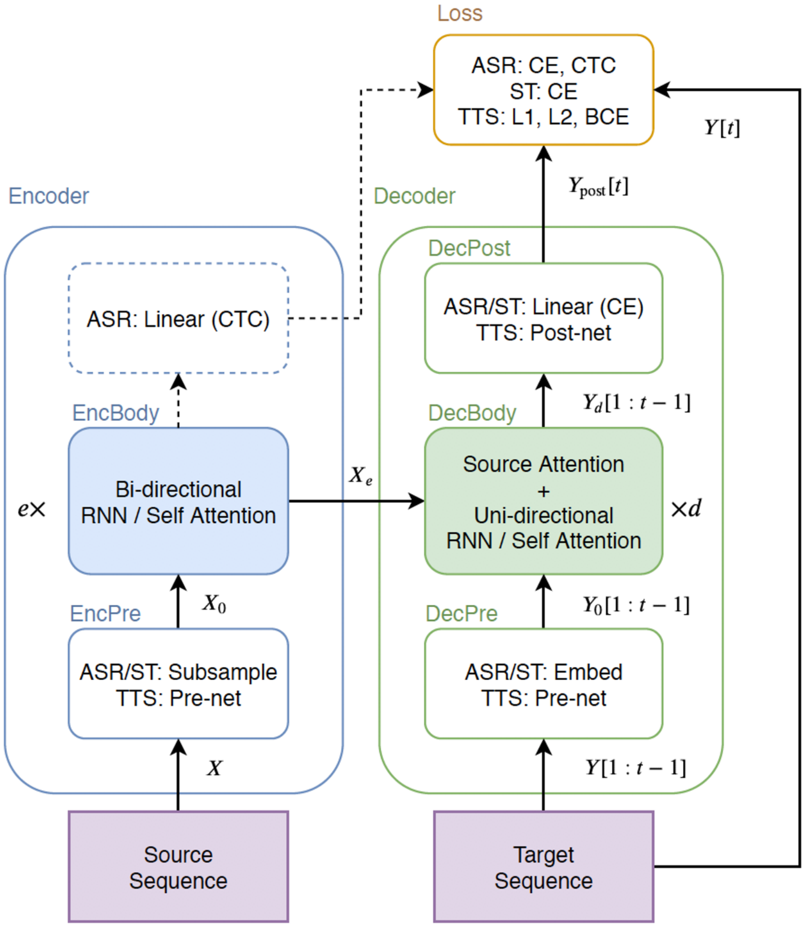
Transformer vs RNN



ASR Encoder Architecture

- X - a sequence of 83-dim log-mel filterbank frames with pitch features
- EncPre() transforms the source sequence X into a subsampled sequence X0 using
  - **two-layer CNN with 256 channels, stride size 2 and kernel size 3 or**
  - **VGG like max pooling**
- EncBody() transforms X0 into a sequence of encoded features Xe for the
- CTC and decoder networks.

Transformer vs RNN



ASR Decoder Architecture

- Receives the encoded sequence Xe and the prefix of a target sequence Y [1 : t - 1] of token IDs: characters or SentencePiece
- DecPre() in Eq. (3) embeds the tokens into learnable vectors.
- DecBody() and single-linear layer DecPost() predicts the posterior distribution of the next token prediction Ypost[t] given Xe and Y [1 : t - 1].

Transformer vs RNN

ASR Training

- During ASR training, both the decoder and the CTC module predict the frame-wise posterior distribution of Y given corresponding source X: ps2s(Y jX) and pctc(Y jX), respectively.
- Use the weighted sum of those negative log likelihood values:

$$L^{\text{ASR}} = -\alpha \log p_{\text{s2s}}(Y|X) - (1-\alpha) \log p_{\text{ctc}}(Y|X),$$

α is a hyperparameter.

ASR Decoding

- Decoder predicts the next token given the speech feature X and the previous predicted tokens using beam search, which combines the scores of S2S, CTC and the RNN language model (LM):

$$\hat{Y} = \operatorname*{argmax}_{Y \in \mathcal{Y}^*} \{\lambda \log p_{\text{s2s}}(Y|X_e) + (1-\lambda) \log p_{\text{ctc}}(Y|X_e) + \gamma \log p_{\text{lm}}(Y)\},$$

γ, λ are hyperparameters

$\mathcal{Y}^*$ is a set of hypotheses of the target sequence

Transformer vs RNN - Results

Table 3. Comparison of the Librispeech ASR benchmark

	dev_clean	dev_other	test_clean	test_other
RWTH (E2E) [44]	2.9	8.8	3.1	9.8
RWTH (HMM) [45]	2.3	**5.2**	2.7	**5.7**
Google SpecAug. [26]	N/A	N/A	**2.5**	5.8
ESPnet Transformer (ours)	**2.2**	5.6	2.6	**5.7**

Results are comparable to best performance.

Transformer vs RNN - Results

Transformer significantly outperformed RNN in 9 languages.

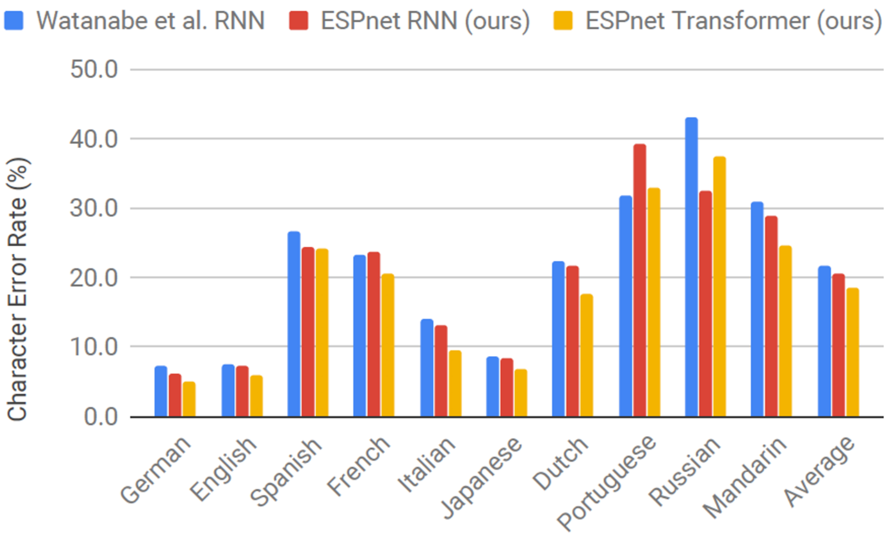


Fig. 3. Comparison of multilingual end-to-end ASR with the RNN in Watanabe et al. [46], ESPnet RNN, and ESPnet Transformer.

Practical Tips

- When Transformer suffers from under-fitting,
  - increase the minibatch size which results in faster training time and better accuracy
- Dropout is essential for Transformer to avoid over-fitting
- Data augmentation methods greatly improved both Transformer and RNN.
  - SpecAugment: A simple data augmentation method for automatic speech recognition. [Park D. et.al.]
  - Audio augmentation for speech recognition [T. Ko et. al.]
- The best decoding hyperparameters are for RNN are generally the best for Transformer.

- Transformer's decoding is much slower than Kaldi's system because the self-attention requires O(n2) in a naive implementation
  - To directly compare the performance with DNN-HMM based ASR systems, a faster decoding algorithm for Transformer was developed.
- The accumulating gradient strategy can be adopted to emulate the large minibatch if multiple GPUs are unavailable.

4

Jasper: An End-to-End Convolutional Neural Acoustic Model

Jason Li[1], *Vitaly Lavrukhin*[1], *Boris Ginsburg*[1], *Ryan Leary*[1], *Oleksii Kuchaiev*[1], *Jonathan M. Cohen*[1], *Huyen Nguyen*[1], *Ravi Teja Gadde*[2]

[1]NVIDIA, Santa Clara, USA
[2]New York University, New York, USA

[4] Jason Li et. al. Jasper: An end-to-end convolutional neural acoustic model. In Interspeech, 2019. https://arxiv.org/pdf/1904.03288.pdf

Jasper: An end-to-end convolutional neural acoustic model.

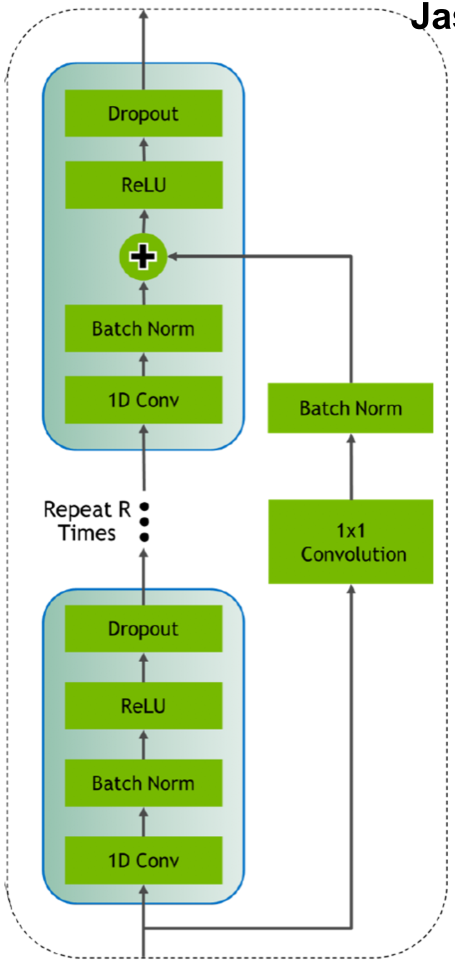


Figure 1: JasperBxR model

Computationally efficient end-to-end convolutional neural network acoustic model
Sub-block architecture is designed to facilitate fast GPU inference

A Typical Block

Uses only (example block)

- 1D convolutions
- batch normalization
- ReLU
- Dropout
- Residual connections

NovoGrad, a variant of the Adam optimizer with smaller memory footprint

B - number of blocks,

R - number of sub-blocks.

Jasper: An end-to-end convolutional neural acoustic model - *Complete Model*

Post-Processing:

- Three blocks

Pre-Processing:

- One block

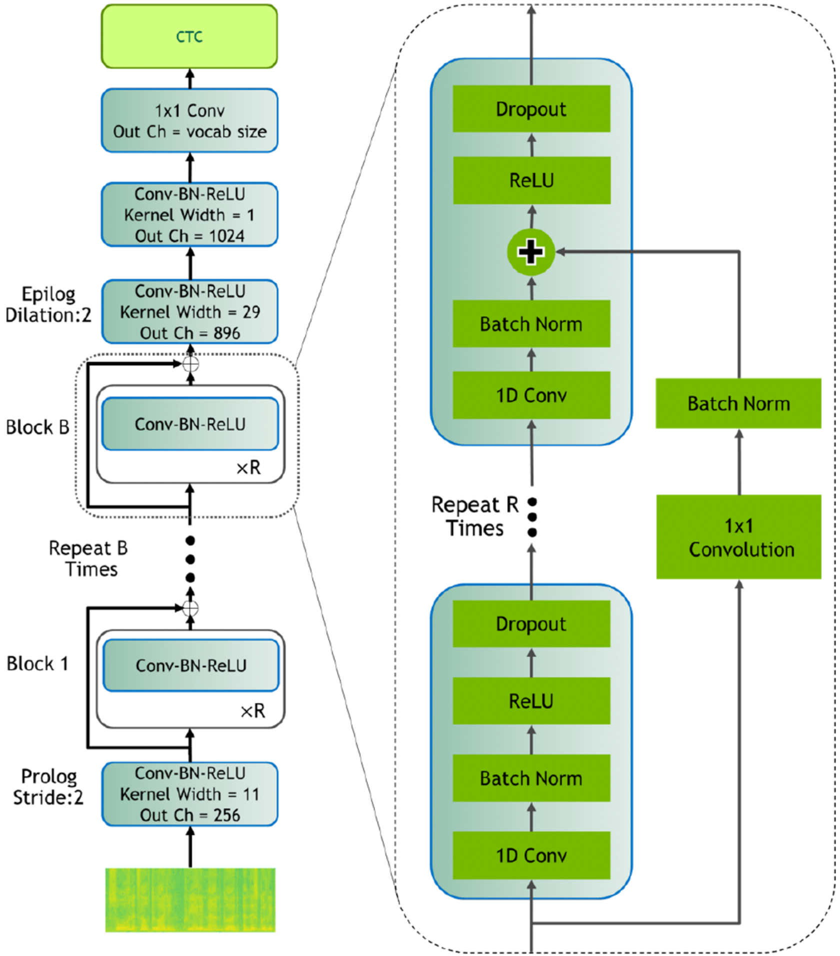


Figure 1: JasperBxR mode:

Jasper: An end-to-end convolutional neural acoustic model -
Model with Dense Residual

Output of a convolution block is **added** to the inputs of all the following blocks (not concatenated as in ResNet

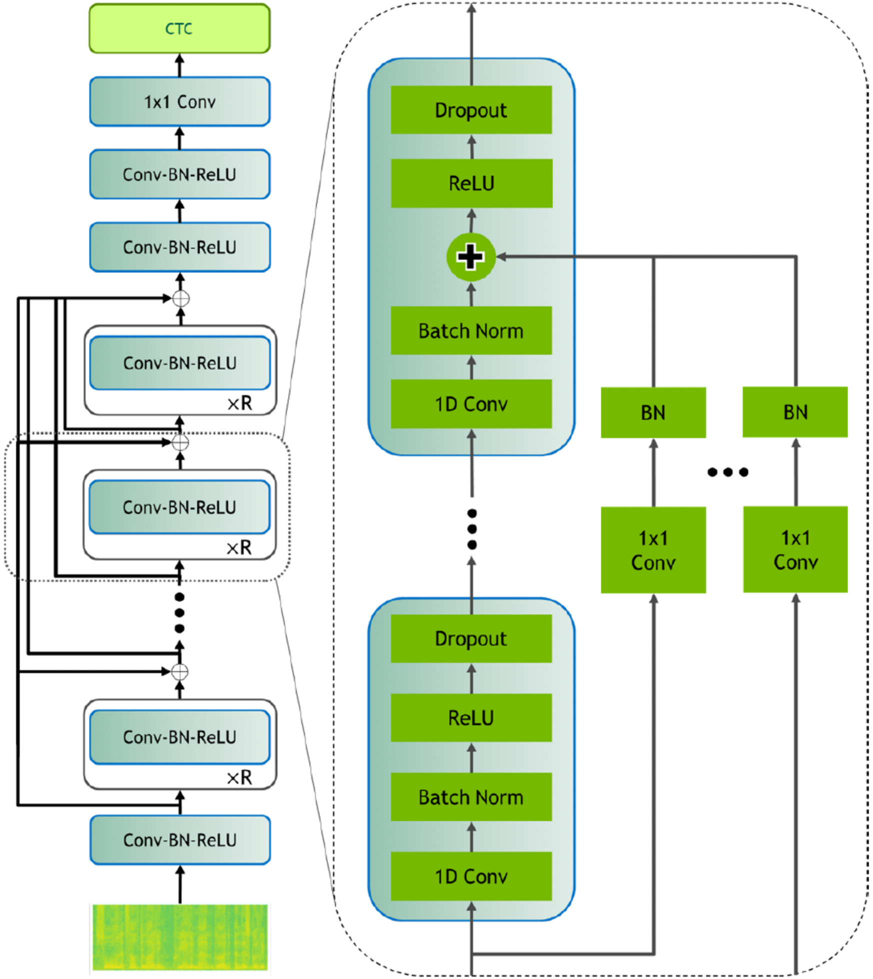
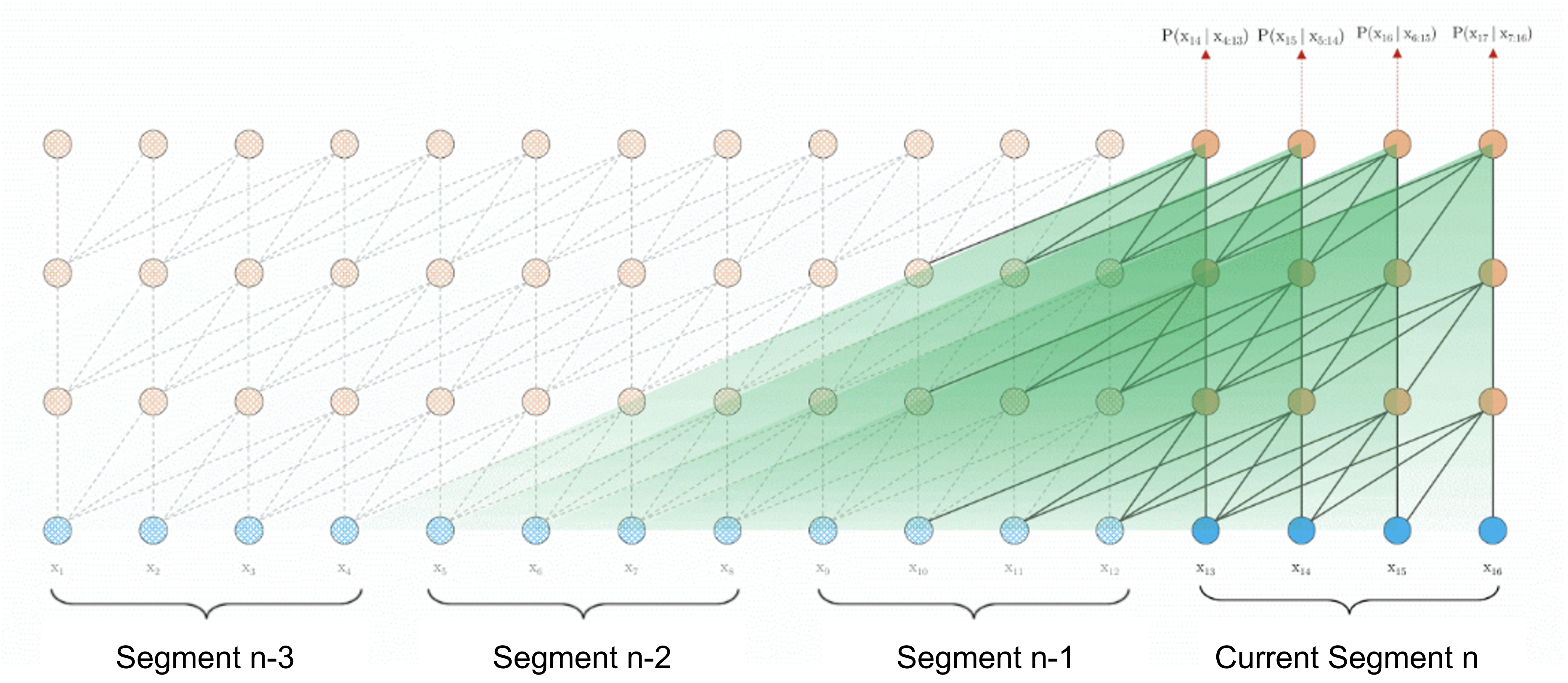


Figure 2: Jasper Dense Residual

Transformer-XL - Increase context over multiple segments



Jasper: An end-to-end convolutional neural acoustic model

Table 5: *LibriSpeech, WER (%)*

Model	E2E	LM	dev-clean	dev-other	test-clean	test-other
CAPIO (single) [23]	N	RNN	3.02	8.28	3.56	8.58
pFSMN-Chain [25]	N	RNN	2.56	7.47	2.97	**7.5**
DeepSpeech2 [26]	Y	5-gram	-	-	5.33	13.25
Deep bLSTM w/ attention [21]	Y	LSTM	3.54	11.52	3.82	12.76
wav2letter++ [27]	Y	ConvLM	3.16	10.05	3.44	11.24
LAS + SpecAugment [4] [28]	Y	RNN	-	-	*2.5*	*5.8*
Jasper DR 10x5	Y	-	3.64	11.89	3.86	11.95
Jasper DR 10x5	Y	6-gram	2.89	9.53	3.34	9.62
Jasper DR 10x5	Y	Transformer-XL	2.68	8.62	**2.95**	8.79
Jasper DR 10x5 + Time/Freq Masks [4]	Y	Transformer-XL	2.62	7.61	2.84	7.84

4: with time and frequency masks similar to SpecAugment [Park et. al.]

5

Sequence-to-Sequence Speech Recognition with Time-Depth Separable Convolutions

Awni Hannun, Ann Lee, Qiantong Xu, Ronan Collobert

Facebook AI Research

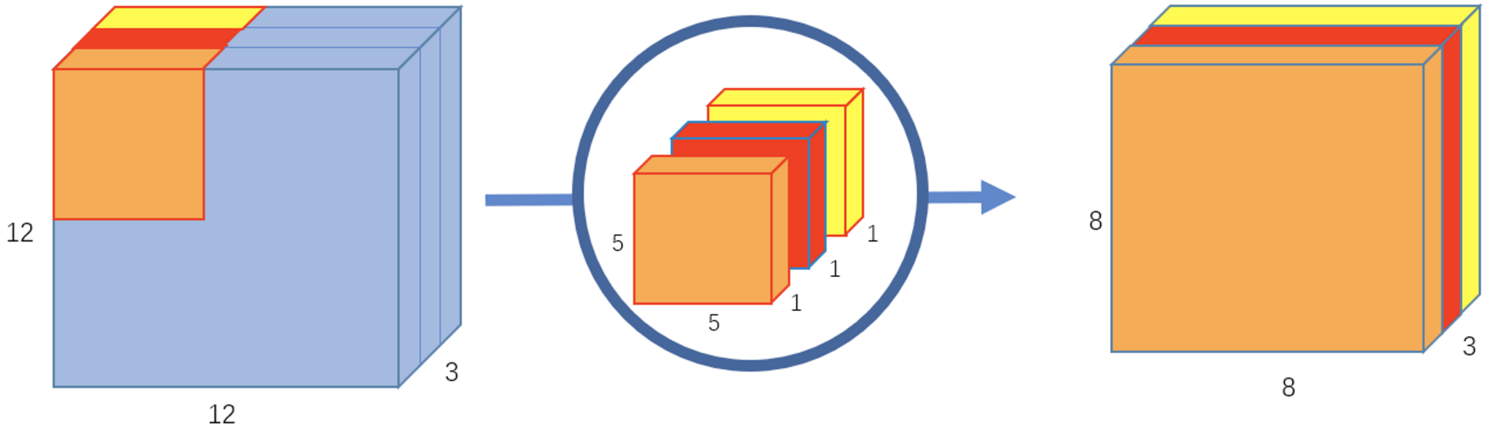
[5] Awni Hannun et. al. Sequence-to-sequence speech recognition with time-depth separable convolutions. Interspeech 2019. https://arxiv.org/pdf/1904.02619.pdf

Seq2seq speech recognition with time-depth separable convolutions

- Fully convolutional seq2seq encoder architecture
  - With a simple and efficient decoder
  - An order of magnitude more efficient than a strong RNN baseline
- Time-depth separable convolution block
  - Dramatically reduces the number of parameters in the model while keeping the receptive field large.
- Efficient beam search inference procedure to integrate a language model.
- Improves by more than 22% relative WER over the best previously reported seq2seq results on the **noisy** LibriSpeech test set.

Depthwise convolution:

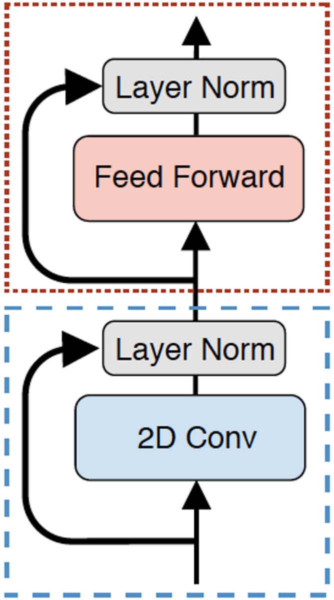
Example: use 3 kernels to transform a 12x12x3 image to a 8x8x3 image



Each 5x5x1 kernel iterates 1 channel of image (1 channel - not all channels) getting the scalar product of every 25 pixel group, giving out 8x8x1 image

Fig: Refer

Figure 1: The TDS convolution model architecture.

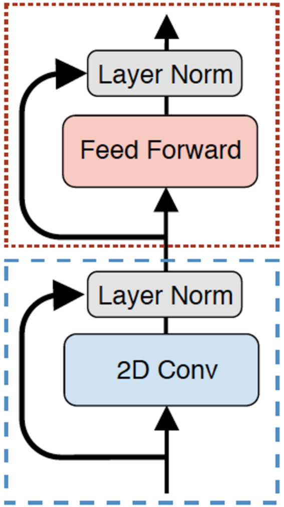


(a)

- Partially decouples the aggregation over time
- Increases the receptive field of the model with a negligible increase in the number of parameters
- generalizes much better than other deep convolutional architectures
- Block structure can be implemented efficiently using a standard 2D convolution.

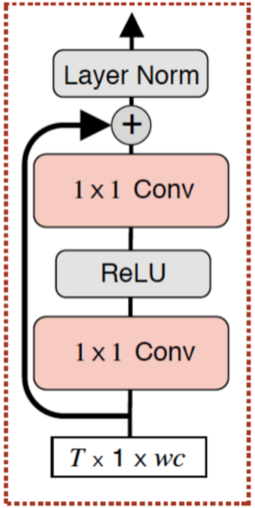
Figure 1: The TDS convolution model architecture.

(a) TDS convolution layer



(a)

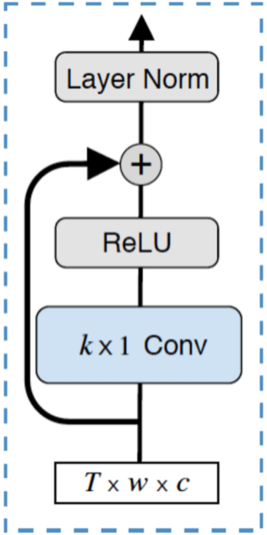
- Output of the convolution as T x 1 x wc
- Apply a fully-connected layer, which is a sequence of two 1x1 convolutions (i.e. linear layers) with a ReLU non-linearity in between.
- We add residual connections and layer normalization after the convolution and the fully connected layer.
- The layer normalization is over all dimensions for a given example including time.



(c) a fully connected block.

Input of shape T x w x c and produces an
Output of shape T x w x c
Follow convolution with ReLU

- T is the number of time-steps,
- w is the input width and
- c is the number of input (and output) channels



(b) a 2D convolution over time followed by

Experimental Setup:

Toolkit wav2letter++
Dataset: Full 960-hour LibriSpeech corpus
Best encoder has two 10-channel, three 14-channel and six 18-channel TDS blocks.
Three 1D convolutions to sub-sample over time, one as the first layer and
one in between each group of TDS blocks.
Kernel sizes are all 21 x 1.
A final linear layer produces the 1024-dimensional encoder output.

The decoder is a one-layer GRU with 512 hidden units.

Input features are 80-dimensional mel-scale filter banks computed every 10-ms with a 25-ms window.
We use 10k word pieces computed from the SentencePiece toolkit as the output token set.

All models are trained on 8 V100 GPUs with a batch size of 16 per GPU.
Synchronous SGD with a learning rate of 0.05, decayed by a factor of 0.5 every 40 epochs.
Clip the gradient norm to 15.
The model is pretrained for three epochs with the soft window and = 4.
Use 20% dropout, 5% label smoothing, 1% random sampling and 1% word piece sampling.

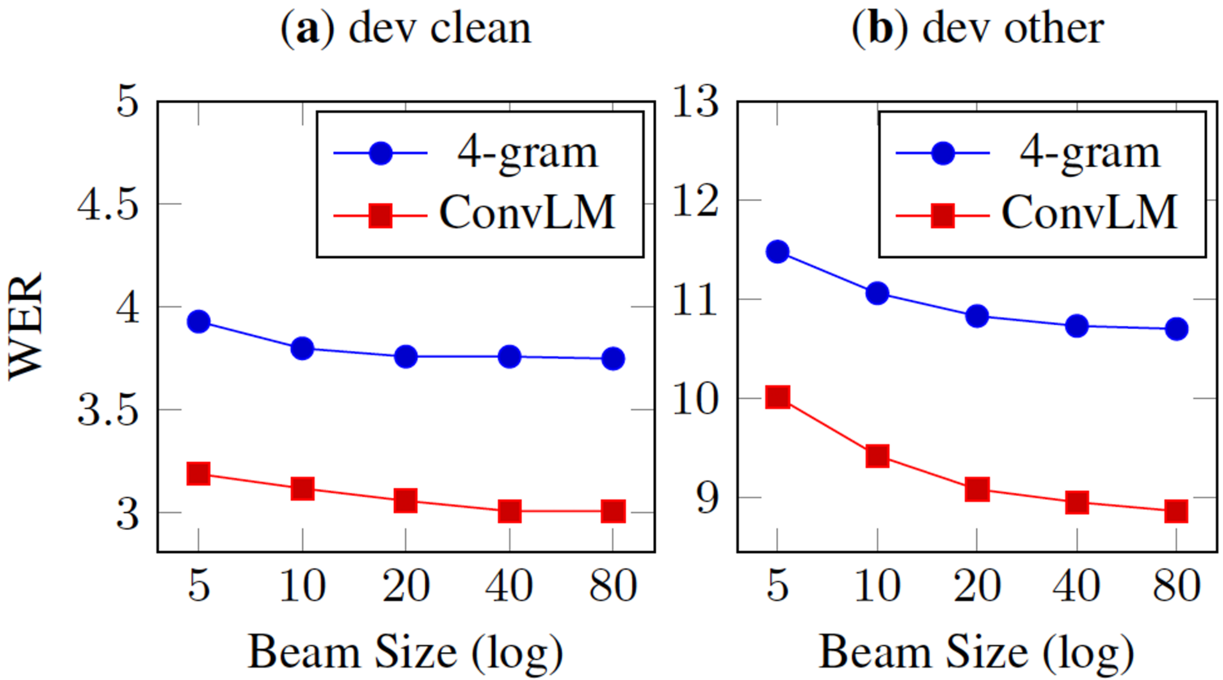
Seq2seq speech recognition with time-depth separable convolutions

Table 1: A comparison of the TDS conv model to other models on the Librispeech Dev and Test sets.

Model	Dev WER		Test WER	
	clean	other	clean	other
hybrid, speaker adapted				
CAPIO (single) [33] + RNN	3.12	8.28	3.51	8.58
CAPIO (ensemble) [33] + RNN	2.68	7.56	3.19	7.64
CNN ASG [31] + ConvLM	3.16	10.05	3.44	11.24
RNN S2S [23]	4.87	14.37	4.87	15.39
RNN S2S [23] + 4-gram	4.79	14.31	4.82	15.30
RNN S2S [23] + LSTM	3.54	11.52	3.82	12.76
TDS conv	5.04	14.45	5.36	15.64
TDS conv + 4-gram	3.75	10.70	4.21	11.87
TDS conv + ConvLM	3.01	8.86	3.28	9.84

Seq2seq speech recognition with time-depth separable convolutions

Figure 3: The WER as a function of beam size for both the 4-gram and the convLM.



6

TRANSFORMER-BASED ACOUSTIC MODELING FOR HYBRID SPEECH RECOGNITION

Yongqiang Wang[1], *Abdelrahman Mohamed*[1], *Duc Le*[1], *Chunxi Liu*[1], *Alex Xiao*[1],
Jay Mahadeokar [1,⋆], *Hongzhao Huang* [1,⋆], *Andros Tjandra* [2,⋆†], *Xiaohui Zhang* [1,⋆], *Frank Zhang* [1,⋆],
Christian Fuegen[1,⋆], *Geoffrey Zweig*[1,⋆], *Michael L. Seltzer*[1,⋆]

[1]Facebook AI, USA [2]Nara Institute of Science and Technology, Japan

[6] Yongqiang Wang, et. al. Transformer-based acoustic modeling for hybrid speech recognition, 2019. https://arxiv.org/pdf/1910.09799.pdf

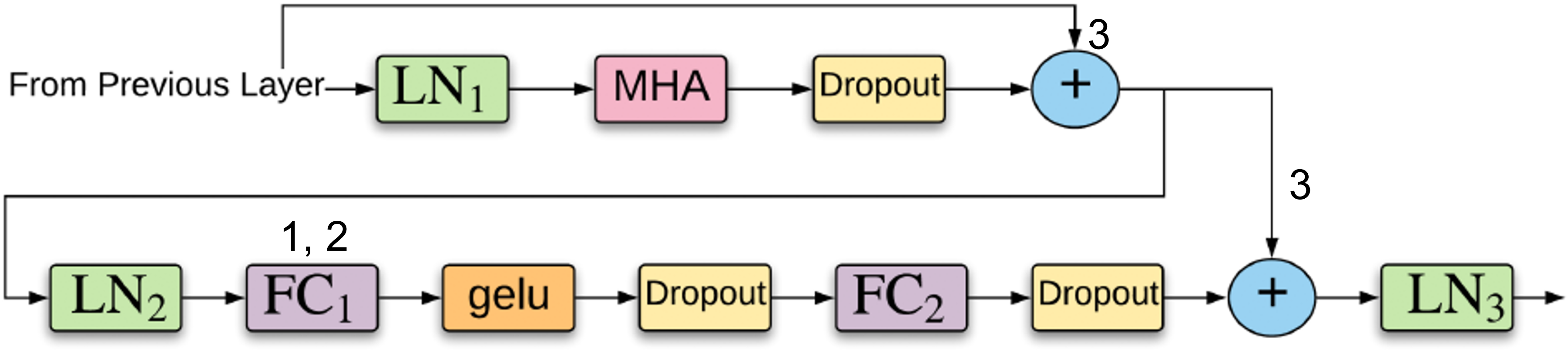
Architecture

For streaming applications - use limited right context in transformer models

Transformers:
12-layer transformer architecture with di = 768
Per head dimension is always 64
FFN dimension is always set to 4 x di.
This model has about 90M parameters.

BLSTMs:
a 5-layer BLSTM with 800 units per layer per direction (about 94M parameters)
a 6-layer BLSTM with 1000 units (about 163M parameters)

Figure 1: One transformer layer



1. Fully-connected feed-forward network (FFN), which is composed by two linear transformations and a nonlinear activation function in between.
2. The FFN network is applied to each position in the sequence separately and identically.

1. To allow stacking many transformer layer, residual connections are added to the MHA and FFN sublayers.

- Dropouts are also applied after MHA and linear transformation as a form of regularization.
- Layer Normalization is applied before MHA and FFN
- Third layer normalization (LN3) is necessary to prevent bypassing the transformer layer entirely.
- “gelu” non-linearity in the FFN network.

FAIRSEQ: A Fast, Extensible Toolkit for Sequence Modeling

Myle Ott[△*] **Sergey Edunov**[△*] **Alexei Baevski**[△] **Angela Fan**[△] **Sam Gross**[△]
Nathan Ng[△] **David Grangier**[▽†] **Michael Auli**[△]
△ Facebook AI Research
▽ Google Brain

https://github.com/pytorch/fairseq

FAIRSEQ

Open-source sequence modeling toolkit

Train custom models for

- translation,
- summarization,
- language modeling, and
- other text generation tasks.

Fast inference for non-recurrent models

The toolkit is based on PyTorch and supports distributed training across multiple GPUs and machines.

Support fast mixed-precision training and inference on modern GPUs.

Applications:

- Machine Translation
- Language Modeling
- Abstractive Document Summarization
- Story Generation
- Error Correction
- Multilingual Sentence Embeddings
- Dialogue

https://github.com/pytorch/fairseq

Experiment Setups

Toolkit: PyTorch based fairseq

80-dimensional log Mel-filter bank features are extracted with a 10ms frame shift.
A reduced 20ms frame rate is achieved either by stacking-and-striding 2 consecutive frames or by a stride-2 pooling in the convolution layer if it is used.
This not only reduces the computation but also slightly improves the recognition accuracy.

Speed perturbation and SpecAugment (LD policy without time warping) are used.
Focus on cross-entropy (CE) trained models and only selectively perform sMBR training on top of the best CE setup.

Use context- and position-dependent graphemes (i.e., chenones) in all experiments.
Bootstrap HMM-GMM system using the standard Kaldi Librispeech recipe.
Use 1-state HMM topology with fixed self-loop and forward transition probability (both 0.5).

Transformer Training Tricks:

Due to the quadratically growing computation cost with respect to the input sequence length

- segment the training utterances into segments that are not longer than 10 seconds 2.
- Though this creates a mismatch between training and testing, preliminary results show that training on shorter segments
    - not only increases the training throughput but
    - also helps the final WERs.

Transformers are more prone to over-fitting, thus require some regularization.

- SpecAugment is effective: without it, WER starts to increase after only 3 epochs,
- while WER continues to improve during training with SpecAugment.

BiLSTM and Transformer comparison:

Table 2: Architecture comparison on the Librispeech benchmark

Model Arch	#Params (M)	test-clean	test-other
BLSTM (800,5)	79	3.11	7.44
Trf-FS (768,12)	91	3.04	6.64
vggBLSTM (800,5)	95	2.99	6.95
vggTrf. (768,12)	93	2.87	6.46
vggBLSTM (1000,6)	163	2.86	6.63
vggTrf. (768, 20)	149	2.77	6.10

Positional Encoding - Convolution:

Use two VGG blocks beneath transformer layers:

Each VGG block contains 2 consecutive convolution layers with a 3-by-3 kernel followed by a ReLu and pooling layer; 32 channels are used in the convolution layer of the first VGG block and increase to 64 for the second block.

Maxpooling is performed at a 2-by-2 grid, with stride 2 in the first block and 1 in the second block.

For an input sequence of 80-dim feature vector at a 10ms rate, this VGG network produces a 2560-dim feature vector sequence at a 20ms rate.

Note that the perception field of each feature vector output by the VGG network consists of 80ms left-context and 80ms right context, the same right context length as Frame Stacking.

A linear projection is used to project the feature vector to the dimension accepted by transformers, 768.

Results - LibriSpeech:

Arch.	System	LM	test-clean	test-other
LAS	Park et al. [10]	NNLM + 4g	2.5	5.8
	Karita et al. [30]	NNLM	2.6	5.7
Hybrid	RWTH [38]	4g	3.8	8.8
		+NNLM	2.3	5.0
	Han et al. [41]	4g	2.9	8.3
		+NNLM	**2.2**	5.8
	Le et al. [24]	4g	3.2	7.6
	Ours	4g	**2.60**	**5.59**
		+NNLM	2.26	**4.85**

Table 4: Comparison with previous best results on Librispeech.

Results - Right Context (for streaming):

Inference: Force every layer to attend to a fixed limited right context during inference. This creates a large mismatch between training and inference, the resultant systems can still yield reasonable WERs if the number of right context frames is large enough.

RC	test-clean	test-other
∞	2.87	6.45
50	3.01	7.12
20	3.29	8.10
10	3.65	9.01

Table 5: Forcing transformer models to use limited right context (RC) per layer during inference. Given a 12-layer transformer, an RC of 10 frames translates to 2.48 seconds of total lookahead.

7

SELF-TRAINING FOR END-TO-END SPEECH RECOGNITION

Jacob Kahn, Ann Lee, Awni Hannun

Facebook AI Research

[7] Kahn Jacob et. al..Self-training for end-to-end speech recognition. ICASSP 2020. https://arxiv.org/pdf/1909.09116.pdf

Self Training for End-to-End Speech Recognition

Training with pseudo-labels substantially improves the accuracy of a baseline model.

Approach is to use strong baseline acoustic and language model to generate

- The pseudo-labels,
- Filtering mechanisms tailored to common errors from sequence-to-sequence models, and
- A novel ensemble approach to increase pseudo-label diversity
- Ensemble of four models and label filtering

Approach

Generate pseudo labels:

Train a strong baseline acoustic model on a small paired data set

Decoding:

Perform stable decoding with a language model (LM) trained on a large-scale text corpus to generate pseudo-labels.

Pseudo label filtering:

Evaluate one heuristic and one confidence-based method for pseudo-label filtering tailored to the mistakes often encountered with sequence-to-sequence models.

Ensemble:

An ensemble combines multiple models during training to improve label diversity and keep the model from being overly confident to noisy pseudo-labels.

Approach

Effectiveness of self-training on LibriSpeech, to study the trade-off between

the **amount of unpaired audio data**
the **quality of the pseudo-labels**, and
the **model performance**.

Clean speech setting:

- as the **label quality is high**, the model performance depends heavily on the **amount of data**.

Noisy speech setting:

- a **proper filtering mechanism** is essential for **removing noisy pseudo-labels**.
- In addition, using an **ensemble** of models can be complementary to filtering.

WER recovery rate (WRR):

Demonstrates how much gap between the baseline and the oracle that we can bridge with pseudo-labels.

$$\frac{\text{baseline WER} - \text{semi-supervised WER}}{\text{baseline WER} - \text{oracle WER}}$$

Approach - Filtering:

The pseudo-labelled data set ~D contains noisy transcriptions.

Achieve the **right balance** between the **size** of ~D and the **noise** in the pseudo labels.

Filtering techniques on the sentence level with heuristic techniques.

Sequence-to-sequence models easily fail at inference in two ways:

Looping

- Remove pseudo-labels which **contain an n-gram repeated more than c times**.

Early stopping

- Deal with early stopping by only keeping hypotheses with an **EOS probability above a threshold**.
- **Filter examples** where the **beam search terminates without finding any complete hypotheses.**

Length Normalised log likelihood:

For each pseudo-label, compute the length-normalized log likelihood from the sequence-to-sequence model as the confidence score

$$\text{ConfidenceScore}(\bar{Y}_i) = \frac{\log P_{\text{AM}}(\bar{Y}_i \mid X_i)}{|\bar{Y}_i|}$$

where $|\bar{Y}_i|$ is the number of tokens in the utterance

Approach - Ensemble:

Combine the **model scores during inference to generate a single pseudo-labelled set with higher quality**. As number of models increase, the decoding process becomes heavyweight.

Sample ensemble.

Given M bootstrapped acoustic models, we generate a pseudo-labelled data set, ~Dm, for each model in parallel.

Combine all M sets of pseudo-labels with uniform weights and optimize the following objective during training

$$\sum_{(X,Y)\in\mathcal{D}} \log P(Y \mid X) + \frac{1}{M}\sum_{m=1}^{M} \sum_{(X,\bar{Y})\in\bar{\mathcal{D}}_m} \log P(\bar{Y} \mid X)$$

First train M models on D using different randomly initialized weights.

Generate ~Dm with hyper-parameters tuned with each model, respectively.

During training, uniformly sample a pseudo-label from one of the M models as the target in every epoch.

Experiments using wav2letter++ framework:

Data - LibriSpeech

train-clean-100” set

containing **100 hours of clean speech as the paired data set**.

Clean speech setting,

360 hours of clean speech in the “train-clean-360” set as the unpaired audio set, and

Noisy speech setting,

500 hours of noisy speech in the “train-other-500” set.

Self-training - LM:

Remove all books related to the acoustic training data from the LM training data

Sentence segmentation using the NLTK toolkit

Normalize the text by lower-casing

Remove punctuation except for the apostrophe, and **replacing hyphens with spaces**.

Do not replace non-standard words with a canonical verbalized form.

Resulting LMs achieve comparable perplexity to LMs trained on the standard corpus on the dev sets.

Experiments using wav2letter++ framework:

Setting

Encoder consists of **nine TDS blocks** in **groups of three**, each with 10, 14 and 16 channels and a kernel width of 21.

Use the SentencePiece toolkit to compute 5,000 word pieces from the transcripts in “train-clean-100” as the target tokens.

Training process: **teacher-forcing** with 20% dropout, 1% random sampling, 10% label smoothing and 1% word piece sampling for regularization.

A single GPU with a batch size of **16** when training baselines, and

- 8 GPUs when training with pseudo-labels.

SGD without momentum for 200 epochs with a learning rate of 0.05, decayed by 0.5 every 40 epochs when using one GPU or 80 epochs for 8 GPUs.

Train a word piece convolutional LM (ConvLM)

All beam search hyper-parameters are **tuned on the dev sets** before generating the **pseudo-labels**.

When training models with the combined paired and pseudo-labelled data sets, start from **random initialization** instead of two-stage fine-tuning.

Results - Importance of Filtering

Label quality is defined as the WER of the **filtered pseudo-labels** as compared to the ground truth.

Heuristic filtering, i.e. "**no EOS + n-gram**" filters, with c=2 and n=4 and then add **confidence-based filtering** on top of the filtered data set.

Filtering improves the **pseudo-label quality** the threshold on the confidence score is adjusted

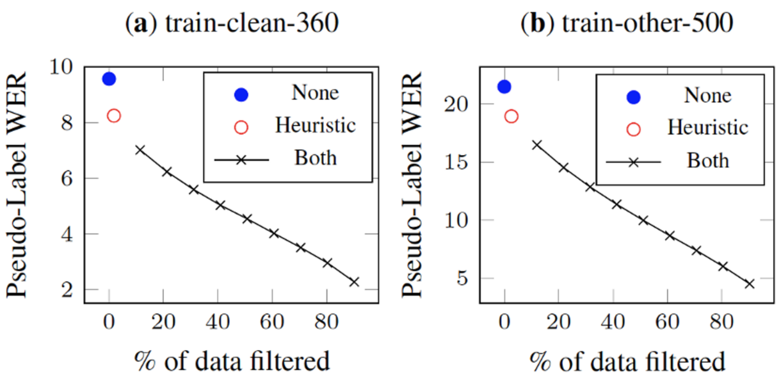
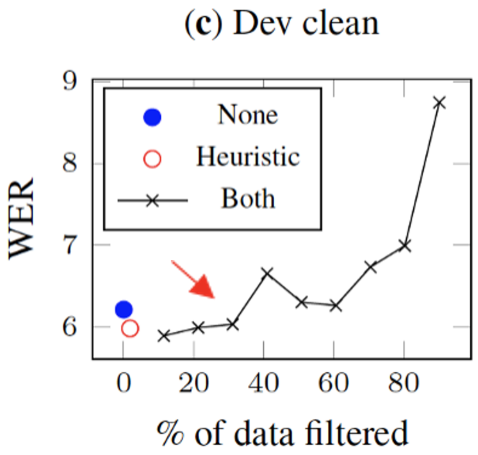
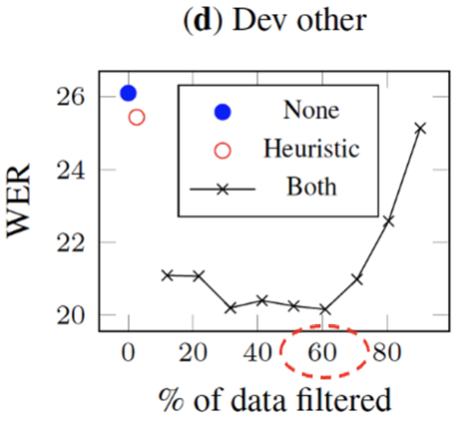


Fig. 1a, b: Results of different filtering functions and the corresponding pseudo-label quality

Results - Importance of Filtering

In the **clean setting**, the heuristic filter removes **1.8% of the data**, and further removal of the worst 10% of the pseudo-labels based on confidence scores results in a 5.2% relative improvement in WER on the dev clean set compared with a baseline without filtering.
More aggressive filtering improves the label quality but results in **worse** model performance.





In the noisy setting, **removing the worst 10% of the pseudo-labels results in a significant reduction in WER**, and the best performance comes from **filtering 60% of the labels** with a WER 22.7% relative lower on the dev other set compared with no filtering.
Filtering more data leads to the same degradation in model performance as in the clean setting.

Results - Model Ensemble

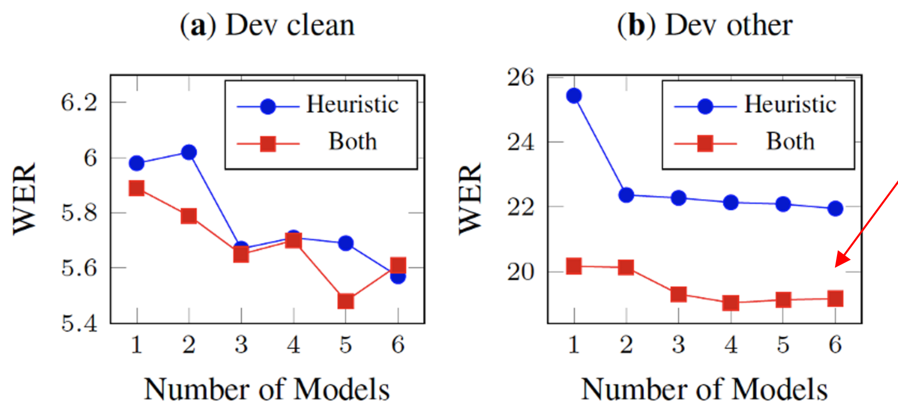


Fig. 2. WER with respect to number of models in ensemble under the clean ((**a**)) or noisy ((**b**)) setting. Results are with LM beam search decoding and averaged across three runs. (*Both:* heuristic and confidence-based filters)

Combining multiple models improves the performance, especially for the noisy setting, where we obtain a **13.7% relative** improvement with six models and heuristic filtering.

Since the **sample ensemble uses different transcripts** for the **same utterance at training time**, this keeps the model from being overly confident in a noisy pseudo-label.

In the noisy setting, model ensembles with both filterings improve WER by 27.0% relative compared with a single model without any filtering (Figure 1(d)).

Results - Comparison with Literature

Table 1. Best results from single runs tuned on the dev sets.

Method	No LM				With LM			
	Dev WER		Test WER (WRR)		Dev WER		Test WER (WRR)	
	clean	other	clean	other	clean	other	clean	other
Baseline Paired 100hr	14.00	37.02	14.85	39.95	7.78	28.15	8.06	30.44
Paired 100hr + Unpaired 360hr clean speech								
Oracle	7.20	25.32	7.99	26.59	3.98	17.00	4.23	17.36
Single Pseudo	9.61	29.72	10.27 (66.8%)	30.50 (70.7%)	5.84	21.86	6.46 (41.8%)	22.90 (57.6%)
Ensemble (5 models)	9.00	27.74	**9.62** (76.2%)	29.53 (78.0%)	5.41	20.31	**5.79** (59.3%)	21.63 (67.4%)
Paired 100hr + Unpaired 500hr noisy speech								
Oracle	6.90	17.55	7.09	18.36	3.74	10.49	3.83	11.28
Single Pseudo	10.90	28.37	11.48 (43.4%)	29.73 (47.3%)	6.38	19.98	6.56 (35.5%)	22.09 (43.6%)
Ensemble (4 models)	10.41	27.00	10.50 (56.1%)	**29.25** (49.6%)	6.01	18.95	6.20 (44.0%)	**20.11** (53.9%)

Summarizes best results, as well as the supervised baseline and the oracle models trained with ground-truth transcriptions.
Present results from both AM only greedy decoding and LM beam search decoding to demonstrate the full potential of self-training.
WER recovery rate (WRR) demonstrates **how much gap between the baseline and the oracle** that **can bridge with pseudo-labels**. WRR is defined as

Results - Comparison other Semi-Supervised Approach

Method	Text (# words)	No LM Test clean WER (WRR)	With LM Test clean WER (WRR)
Cycle TTE [9]	4.8M	21.5 (27.6%)	19.5 (30.6%*)
ASR+TTS [10]	3.6M	17.5 (38.0%)	16.6 (-)
this work	842.5M	**9.62** (**76.2%**)	**5.79** (**59.3%**)

Table 2. A comparison with previous work using 100hr paired data and 360hr unpaired audio. WRR is computed with the baseline and oracle WER from the original work if available. (*: The oracle WER is without LM decoding, so the WRR is an upper bound estimation.)

The conventional pseudo-labelling approach together with filtering and ensemble produces a WER at least **65.1%** relatively lower than the previously best results.

The gain comes from

- the strong baseline model with TDS-based encoders to generate the **pseudo-labels**, and
- a much larger unpaired text corpus, which we believe is easy to obtain in a real-world setting.

8

Semi-supervised DNN training with word selection for ASR

Karel Veselý, Lukáš Burget, Jan "Honza" Černocký

Brno University of Technology, Speech@FIT and IT4I Center of Excellence, Czech Republic

iveselyk@fit.vutbr.cz

[8] Karel Vesely, et. al.. Semi-supervised DNN training with word selection for ASR. In Interspeech, 2017.

Semi-supervised DNN training with word selection for ASR

- **Manually transcribed data**, is **slow** and **costly**. For some rare languages it might be even difficult to find native annotators. Can save a lot of time and other resources, if only **a part of the data is transcribed manually** and a **larger part is transcribed automatically by decoding**.

- The decoding is done with a ‘**seed’ ASR system** trained with the manually transcribed data, while typically we also generate some confidences. **Automatic transcripts are not perfect**, but still, they can be used to improve the **performance** of the **acoustic model** by the **semi-supervised training** (i.e. training with the mixed data: manually transcribed and automatically transcribed).

- This is ‘**self-learning**’, as the ASR system is **re-trained with its own outputs**. The **confidences** express the certainty of the decoded labels, and we will use them to **filter** or **assign weights** to the training data.

- ‘**Data selection**’ strategies on the level of a) **sentences**, b) **words** or c) **frames**.

Semi-supervised DNN training with word selection for ASR

Word level: Per-word confidence (MBR statistics)

- Use the ‘MBR confidence’, which is calculated as the statistics from the Minimum Bayes Risk (MBR) decoding. The quantity gamma(q; s) is the probability with which the word-symbol s is present at position q in the output word-sequence.
- Simply take the words from the bestpath in lattice and calculate their gammas as their confidences.
- This MBR confidence is the default word confidence implemented in Kaldi.

Sentence level: Per-sentence confidence (average word-confidence)

- The per-sentence confidence c*sent* is typically calculated as the average of the word confidences.
- It is good to think about it as an estimate of the word accuracy in a sentence.
- For self-training experiments we use ‘MBR confidences’.

Frame: Per-frame confidence (lattice-posterior)

- The frame-level confidence c*frame-i* is extracted from the lattice posteriors gamma (i; s), which express the probability of being in state s at time i.
- For each frame i, the confidence is c*frame-i* is taken from the best-path in lattice. The posteriors are computed using the forward-backward algorithm on the lattice.

The Seed System:

4599 dimensional softmax output

6 Hidden Layers
(2048 sigmoidal units)
40 dimensional fMLLR features are spliced by +/- 5 frames
Renormalized to have zero mean and unit variance
RBM pre-training to initialize the 6 hidden layers
Frame CE training – mini batch SGD
Re-train by 4 epochs of sMBR training

440 dimensional input

fMLLR features output

Auxiliary GMM system
Splicing +/- 4 frames of the *13-dimensional PLPs* (includes C0) extended by *3 kaldi-pitch features*
All features are cepstral mean variance normalized.
Spliced features are **projected to 40 dimensions** with a **global LDA+MLLT [19]** linear transform and **per-speaker fMLLR [20] linear transform**
4599 cross-word triphone tied states and **5.6 Gaussians per state**

Experimental Setup:

3. Experimental setup

Dataset -Vietnamese dataset.

The **training data** consist of **conversational telephone speech** and a small part of **prompted speech.**

The **development** set consists of **conversational speech only**.

- various telephone channels: landlines, different kinds of cellphones, or phones embedded in vehicles.

Consider the Limited Language Pack (LimitedLP) scenario, in which

11 hours of data are **transcribed**, and

74 hours are '**untranscribed**' (but we have the transcripts available for the analysis).

The results are shown on the development set composed of 9.8 hours of data.

The Vietnamese phone set consists of **29 phonemes**, which are marked with six **different tones**.

Used a **trigram language model** with **Kneser-Ney smoothing** built on the training transcripts from the 11 hours, the model has 12k 3-grams and 47k 2-grams.

Data selection

DNN is trained by ‘frame CE’ training with the mixed data: **manually transcribed** and **automatically transcribed** (decoded by the seed system).

In the first set of experiments, we investigate into the question of the granularity of the confidences. We want to know, what is the ideal size of the ‘data selection unit’.

Sentence selection

The **most common approach** in the literature is the selection of **whole sentences** - good to **leave out 30-50% of sentences**, which brings a 0.3% WER improvement compared to adding all the sentences.

Table 1: *Sentence selection (average MBR word-confidence)*

Added sentences	0%	30%	**50%**	**70%**	90%	100%
WER%	60.9	60.1	**59.8**	**59.8**	60.0	60.1

leave out 30-50% of sentences, better than including all sentences 60.1

Data selection

Word selection

Select the top N% words. The word-selection leads to **0.7% better results** than the sentence-selection. The optimal amount of added words roughly corresponds to the word-accuracy of the seed system.

The WER of the seed system 59.6 is better than the training with 0% added words 60.9. This is because the seed system is trained with ‘**sMBR**’, while the other results are with ‘**frame CE**’ training.

Table 2: *Word selection (per-word MBR confidence)*

Added words	0%	30%	**40%**	50%	100%	**Seed**
WER%	60.9	59.2	**59.1**	59.2	60.1	**59.6**

0.7% better results than sentence

added words roughly corresponds to the word-accuracy of the seed system, which is 100-59:6 = 40:4

Data selection

Frame selection

The **smallest possible unit** for data-selection is the ‘frame’, the frames are produced with **10ms** steps.
Select the frames according to the ‘**lattice-posterior’ confidence**.

Table 3: *Frame selection (lattice-posterior confidence)*

Added frames	0%	50%	**60%**	**70%**	80%	100%
WER%	60.9	59.3	**59.1**	**59.1**	59.3	60.1

The **best frame-selection result is on-par with the best word selection system** in table 2.
It is more **convenient to do the word-selection by word-confidences**, as the word confidences are **represented more compactly** than the frame confidences.

Data Weighting

Add all the untranscribed data, while the **confidences are used as weights** in the SGD training.
The **weights are used to scale the gradients from the individual frames**.

Table 5: *Weighted sentences (average MBR word-confidnce)*

Scale α	1.0	2.0	2.5	3.0	**3.5**	4.0
WER%	59.8	59.6	59.6	59.5	**59.3**	59.5

Table 1: *Sentence selection (average MBR word-confidence)*

Added sentences	0%	30%	**50%**	**70%**	90%	100%
WER%	60.9	60.1	**59.8**	**59.8**	60.0	60.1

‘weighted sentences’ from table 5 are better than ‘selected sentences’ in table 1 (WER 59.8 -> 59.3).
Even better results are achieved with the per-frame or the per-word weights (WER 59.3 -> 58.9 -> 58.8).

Data Weighting

Table 6: *Weighted words (per-word MBR confidence)*

Scale α	1.0	2.0	4.0	8.0	10.0	**12.0**	14.0
WER%	59.5	59.2	59.1	58.9	59.0	**58.8**	59.0

The best result 58.8 was obtained with the per-word confidences

Table 2: *Word selection (per-word MBR confidence)*

Added words	0%	30%	**40%**	50%	100%	**Seed**
WER%	60.9	59.2	**59.1**	59.2	60.1	**59.6**

If we compare the results of the 'selected words' in table 2 with the 'weighted words' in table 6, the improvement is 59.1 -> 58.8.

Re-tuning the systems

It is beneficial to 're-tune' the self trained 'initial model'.

Approach is to *keep the **output layer 'as-is'*** and *continue **training*** with the **11 hours of the manually transcribed data** and a *smaller initial learning rate* (0.001 instead of original 0.008).

Table 4: *'Data selection', re-tuning the initial models. Re-tuning is done with 11 hours of manually transcribed data, the initial model is built with mixed transcribed+untranscribed data.*

WER%	Initial model	Re-tuned + frame CE	Re-tuned + sMBR
Sentence selection	59.8	58.7	57.5
Word selection	**59.1**	58.4	**57.1**
Frame selection	**59.1**	**58.3**	**57.1**
No confidence	60.1	58.7	57.6

Table 8: *'Data weighting', re-tuning the initial models. Re-tuning is done with 11 hours of manually transcribed data, the initial model is built with mixed transcribed+untranscribed data.*

WER%	Initial model	Re-tuned + frame CE	Re-tuned + sMBR
Sentence weighting	59.3	58.3	57.2
Word weighting	**58.8**	58.2	**56.9**
Frame weighting	58.9	**58.1**	57.0
No confidence	60.1	58.7	57.6

Conclusion:

The overall WER improvements from the semi-supervised training become clear after re-tuning the 'simple word selection' models for all the three databases (table 12).

A '**simple word-selection**' setup **without hyperparameter tuning:**

- Choose the amount (%) of the selected words with highest confidence according to the word accuracy on the development set.

Table 12: *Final performance of the semi-supervised training based on 'simple word-selection'. The initial model is trained with the mixed transcribed+untranscribed data. The re-tuning is done with a smaller set of manually transcribed data.*

[WER%]	Vietnamese	Bengali	SWBD
Seed system (sMBR)	59.6	62.9	26.9
Initial model	59.1	62.3	24.4
+ re-tuned (frame CE)	58.4	61.6	24.2
+ re-tuned (sMBR)	57.1	60.6	23.7
Δ WER%	2.5	2.3	3.2

9

End-to-end ASR: from Supervised to Semi-Supervised Learning with Modern Architectures

A Preprint

Gabriel Synnaeve*
Facebook, NYC
gab@fb.com

Qiantong Xu*
Facebook, Menlo Park
qiantong@fb.com

Jacob Kahn*
Facebook, Menlo Park
jacobkahn@fb.com

Edouard Grave*
Facebook, Paris
egrave@fb.com

Tatiana Likhomanenko
Facebook, Menlo Park
antares@fb.com

Vineel Pratap
Facebook, Menlo Park
vineelkpratap@fb.com

Anuroop Sriram
Facebook, Menlo Park
anuroops@fb.com

Vitaliy Liptchinsky
Facebook, Menlo Park
vitaliy888@fb.com

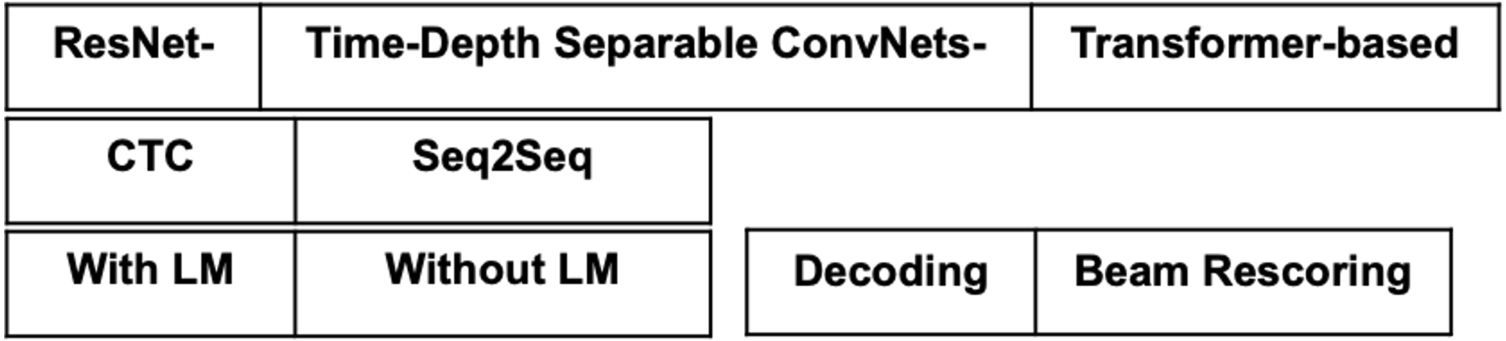
Ronan Collobert*
Facebook, Menlo Park
locronan@fb.com

[9] End-to-End ASR: From Supervised to Semi Supervised Learning with Modern Architecture
https://arxiv.org/pdf/1911.08460.pdf

End-to-End ASR: From Supervised to Semi Supervised Learning with Modern Architecture

In this paper consider

- **ResNet-**, **Time-Depth Separable ConvNets-**, and **Transformer-based** acoustic models,
- Trained with **CTC** or **Seq2Seq** criterions.
- Perform experiments on the LibriSpeech dataset 960hrs **test-other**
- **with** and **without LM decoding**, optionally with **beam rescoring**.



Acoustic Models

Three families of acoustic models (AMs).

1. ResNet Acoustic Models
2. Time-Depth Separable Convolution Acoustic Models
3. Transformers-based Acoustic Models

- All AMs are token-based, outputting 10k word pieces.
- All AMs take 80-channel log-melfilterbanks as input, with STFTs computed on 25ms Hamming windows strided by 10ms, except for TDS models that that are using a 30ms window.

Large number of Models to choose from

Fully convolutional [29]
Conv. Transformers [18]
TDS Convs. [17]
Decoding
LAS [16]
Decoding
biLSTM + attn. [5]
+ Transformer decoding
HMM/biLSTM [5]
+ Transformer rescoring
Transformers [6]
Conv. Transformers [33]
Conv. Transformers [34]
+ Transformer rescoring

CTC
ResNet (306M)
Decoding
Decoding
ResNet LIBRIVOX
Decoding
Decoding
TDS (200M)
Decoding
Decoding
TDS LIBRIVOX
Decoding
Decoding
Conv. Transformers (322M)
Decoding
+ Rescoring
Decoding
+ Rescoring
Conv. Transformers LIBRIVOX
Decoding
+ Rescoring
Decoding
+ Rescoring

Seq2Seq
TDS (190M)
Decoding
Decoding
Conv. Transformers (266M)
Decoding
+ Rescoring
Decoding
+ Rescoring
Conv. Transformers LIBRIVOX
Decoding
+ Rescoring
Decoding
+ Rescoring

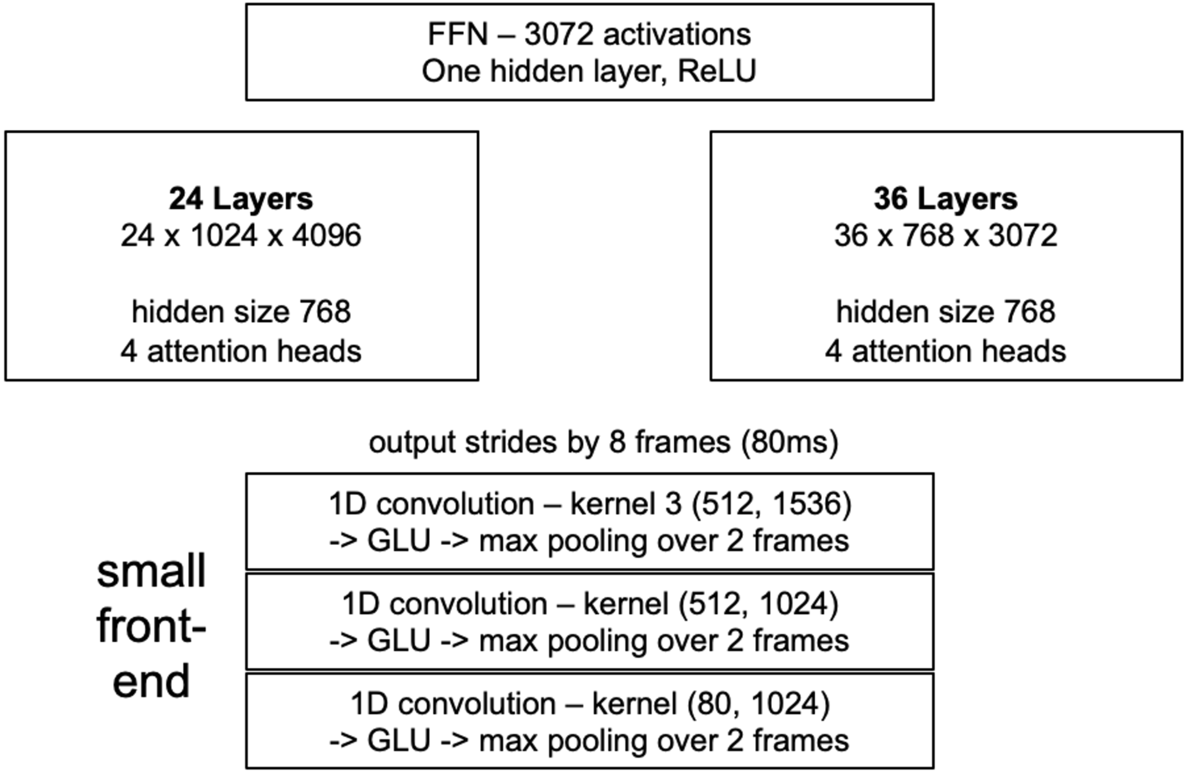
Phase-I: Decided to choose CTC and Transformer based Architecture

AM		LM		Dev		Test	
type	lexicon	type	lexicon	clean	other	clean	other
CTC							
ResNet (306M)	10k WP	-	-	3.96	10.12	4.00	10.02
Decoding	10k WP	4gram	word	3.14	8.50	3.57	8.64
Decoding	10k WP	GCNN	word	2.61	7.12	3.04	7.52
ResNet LIBRIVOX	10k WP	-	-	3.11	7.72	3.23	8.19
Decoding	10k WP	4gram	word	2.80	6.93	3.12	7.42
Decoding	10k WP	GCNN	word	2.44	5.97	2.81	6.42
TDS (200M)	10k WP	-	-	4.34	11.04	4.52	11.16
Decoding	10k WP	4gram	word	3.45	9.31	3.93	9.61
Decoding	10k WP	GCNN	word	2.94	7.71	3.42	8.17
TDS LIBRIVOX	10k WP	-	-	3.14	7.86	3.30	8.46
Decoding	10k WP	4gram	word	2.77	7.01	3.22	7.57
Decoding	10k WP	GCNN	word	2.52	6.23	2.88	6.74
Conv. Transformers (322M)	10k WP	-	-	2.98	7.36	3.18	7.49
Decoding	10k WP	4gram	word	2.51	6.21	2.92	6.65
+ Rescoring	10k WP	GCNN + Transf.	word	2.20	5.05	2.51	5.56
Decoding	10k WP	GCNN	word	2.2	5.32	2.55	5.91
+ Rescoring	10k WP	GCNN + Transf.	word	2.18	4.97	2.49	5.53
Conv. Transformers LIBRIVOX	10k WP	-	-	2.66	6.44	3.03	6.83
Decoding	10k WP	4gram	word	2.59	6.06	2.91	6.47
+ Rescoring	10k WP	GCNN + Transf.	word	2.21	4.82	2.61	5.33
Decoding	10k WP	GCNN	word	2.37	5.34	2.76	5.91
+ Rescoring	10k WP	GCNN + Transf.	word	2.16	4.67	2.63	5.27

4-gram and GCNN LM

Phase-I: Decided to choose CTC and Transformer based Architecture

For **CTC-trained models**, the output of the encoder HLe is followed by a linear layer to the output classes.



Phase II: Decide to choose Seq2Seq, CTC and Transformer based Architecture

For **Seq2Seq models**, we have an additional **decoder**

Decoder:

Stack of 6 transformers

Encoding dimensions 256

4 attention heads

Dropout on the self-attention

Layer drop - dropping entire layers at the FFN level.

Best Effort

For **CTC-trained models**, the output of the encoder HLe is followed by a linear layer to the output classes.

FFN – 3072 activations
One hidden layer, ReLU

24 Layers
24 x 1024 x 4096

hidden size 768
4 attention heads

36 Layers
36 x 768 x 3072

hidden size 768
4 attention heads

output strides by 8 frames (80ms)

small front-end

1D convolution – kernel 3 (512, 1536)
-> GLU -> max pooling over 2 frames

1D convolution – kernel (512, 1024)
-> GLU -> max pooling over 2 frames

1D convolution – kernel (80, 1024)
-> GLU -> max pooling over 2 frames

Experiment Details for Transformer based architecture:

Used wav2letter++1 toolkit

Dataset**:** LIBRISPEECH, and the standard text data for LM training.

All hyperparameters including model architecture are **cross-validated** on **dev-clean and dev-other**.

Used **Adagrad** to train **Transformers**.

Do linear warm-up of the learning rate over 32k to 64k updates. Start with a learning rate of 0.03, and halve it every 40 epochs after the first 150.

Batchsize per GPU to **8** for **Transformers**.

Transformers are trained on average for 3 days on 32 or 64 GPUs for biggest models (Transformers).

Used **SpecAugment**.

All the LMs are **trained** on the standard LIBRISPEECH LM corpus using toolkit

- **KenLM** [31] for **n-gram LM** and
- **fairseq** [32] for **GCNN** and **Transformer** LM.

Used the **GCNN-14B** as **ConvLM**, while the

Transformer LM is the same as the one trained on Google Billion Words

ASR Experiments - Transformer Architecture

Parameters:

- No. of layers:
  - 16, 20, 24, 32, 36, 48
- Embedding dimension:
  - 512, 768, 1024, 1536, 2048
- No. of heads:
  - 4, 8
- If overfits:
  - More training data or increase dropout

Subword:

- Phonemes 73, Subword 5k/10k

Language Model:

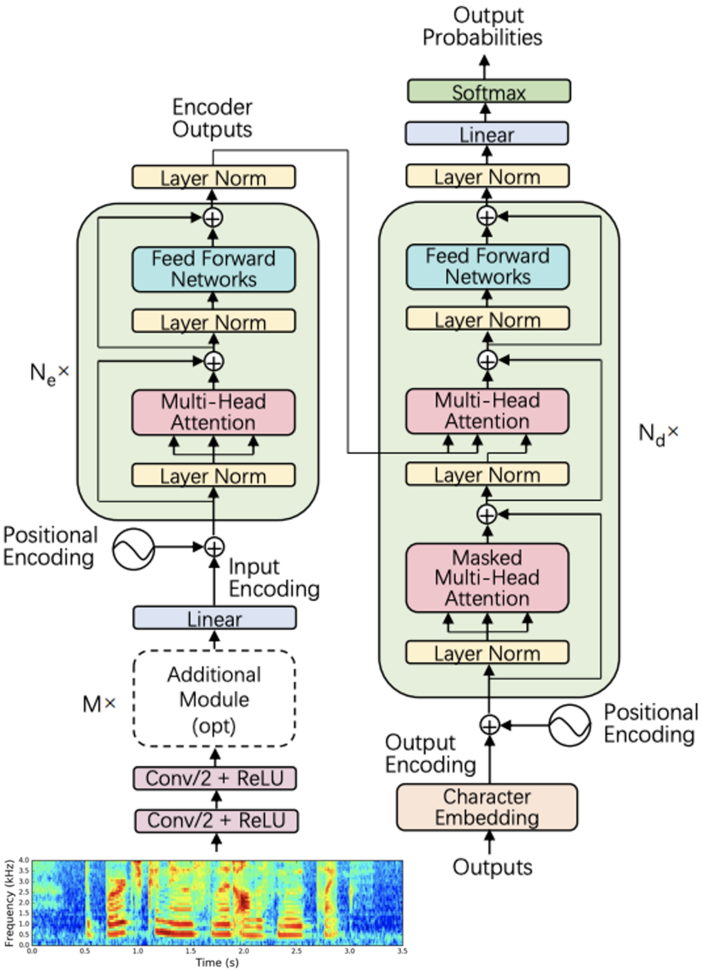
- 3-gram, 4-gram
- Gated CNN based

Inference:

- Beam size

Loss function:

- CTC, ASG or other loss



Dataset - Librispeech

File	Size *.tar.gz	Audio files	Hours	Speakers	Wav2letter lst/
dev-clean	337.9 MB	**2,703**	5.4	**41**	2703
dev-other	314.3 MB	2,864	5.3	34	2864
test-clean	346.6 MB	**2,620**	5.4	**41**	2620
test-other	328.7 MB	2,939	5.1	34	2939
train-clean-100	6,387 MB	**28,539**	100.6	**252**	28,539
train-clean-360	23,049 MB	104,014	363.6	922	104,014
train-other-500	30,593 MB	148,688	496.7	1167	148,688
All 7 above					292,367
train-all-960	~60 GB	281,241	960		281,241

ASR Experiments - Comparison

Models (Paper & 100 iterations numbers)	test-clean WER	test-other WER	Epochs	Paper no.s better than 100iter model: test-clean-WER	Paper no.s better than 100iter model: test-other-WER	Paper no.s better than 100iter model: test-clean-WER %
CTC Transformer (ngram) - paper	**2.92**	**6.65**	320	-1.49	-4.74	-50.93%
CTC Transformer (ngram) - experiment	**4.41**	**11.39**	100			
CTC Transformer (gcnn) - paper	**2.55**	**5.91**	320	-1.20	-4.01	-46.95%
CTC Transformer (gcnn) - experiment	**3.75**	**9.92**	100			

Configuration (am_transformer_ctc.arch)

```
V -1 1 NFEAT 0
WN 3 C NFEAT 1024 3 1 -1      GLU 2
                DO 0.2        M 1 1 2 1
WN 3 C 512 1024 3 1 -1        GLU 2
                DO 0.2        M 1 1 2 1
WN 3 C 512 2048 3 1 -1        GLU 2
                DO 0.2        M 1 1 2 1
RO 2 0 3 1
TR 1024 4096 4 460 0.2 0.2
TR 1024 4096 4 460 0.2 0.2
TR 1024 4096 4 460 0.2 0.2
TR 1024 4096 4 460 0.2 0.2
TR 1024 4096 4 460 0.2 0.2
TR 1024 4096 4 460 0.2 0.2
TR 1024 4096 4 460 0.2 0.2
TR 1024 4096 4 460 0.2 0.2
TR 1024 4096 4 460 0.2 0.2
TR 1024 4096 4 460 0.2 0.2
TR 1024 4096 4 460 0.2 0.2
TR 1024 4096 4 460 0.2 0.2
TR 1024 4096 4 460 0.2 0.2
TR 1024 4096 4 460 0.2 0.2
TR 1024 4096 4 460 0.2 0.2
TR 1024 4096 4 460 0.2 0.2
TR 1024 4096 4 460 0.2 0.2
TR 1024 4096 4 460 0.2 0.2
TR 1024 4096 4 460 0.2 0.2
TR 1024 4096 4 460 0.2 0.2
TR 1024 4096 4 460 0.2 0.2
TR 1024 4096 4 460 0.2 0.2
TR 1024 4096 4 460 0.2 0.2
TR 1024 4096 4 460 0.2 0.2
DO 0.2
L 1024 NLABEL
```

train-w2l-960.cfg

```
# Training config for Mini Librispeech
# Replace `[...]` with appropriate paths
--datadir=/w2l-libri/
--rundir=/w2l-libri/run-960hrs-21may
--archdir=/w2l-libri/wav2letter/tutorials/1-librispeech_clean/
--train=lists/train-all-960.lst
--valid=lists/dev-clean.lst
--input=flac
--arch=network.arch
--tokens=/w2l-libri/am/tokens.txt
--lexicon=/w2l-libri/am/lexicon.txt
--criterion=ctc
--lr=0.1
--maxgradnorm=1.0
--replabel=1
--surround=|
--onorm=target
--sqnorm=true
--mfsc=true
--filterbanks=40
--nthread=4
--batchsize=4
--runname=librispeech_clean_trainlogs
```

Prepare for wav2letter:

```
nohup python wav2letter/tutorials/1-librispeech_clean/prepare_data-360-500.py -
-src $W2LDIR/LibriSpeech/ --dst $W2LDIR &

/home/om/w2l_e2e/lists# ls -l | wc -l
  596205 May 20 20:13 dev-clean.lst             2,703
  591480 May 21 04:15 dev-other.lst                   2,864
  582262 May 20 20:13 test-clean.lst                  2,620
  613304 May 21 04:15 test-other.lst                  2,939
 8864514 May 20 20:13 train-clean-100.lst       28,539
32262983 May 21 04:23 train-clean-360.lst             104,014
44132546 May 21 04:37 train-other-500.lst             148,688


ls -lt audio
           4096 May 22 16:37 LibriSpeech
31918356480 Oct  3  2017 train-other-500.tar
23974318080 Oct  3  2017 train-clean-360.tar
 6641244160 Oct  3  2017 train-clean-100.tar
  355502080 Oct  3  2017 test-other.tar
  370585600 Oct  3  2017 test-clean.tar
  340326400 Oct  3  2017 dev-other.tar
  362526720 Oct  3  2017 dev-clean.tar
```

Prepare for wav2letter:

```
ls -l text
          291159 May 22 16:46 dev-clean.txt
          268815 May 22 16:46 dev-other.txt
4287216164 Oct  3  2017 librispeech-lm-norm.txt
4287216163 May 22 16:46 librispeech-lm-norm.txt.lower.shuffle
          284150 May 22 16:46 test-clean.txt
          275697 May 22 16:46 test-other.txt
          5298357 May 22 16:46 train-clean-100.txt
  19225281 May 22 16:46 train-clean-360.txt
  25539815 May 22 16:46 train-other-500.txt

ls -l am
19537672 May 22 17:01 librispeech-train+dev-unigram-10000-nbest10.lexicon
  419269 May 22 16:47 librispeech-train-all-unigram-10000.model
   82982 May 22 16:47 librispeech-train-all-unigram-10000.tokens
  190417 May 22 16:47 librispeech-train-all-unigram-10000.vocab
19427962 May 22 17:01 librispeech-train-unigram-10000-nbest10.lexicon
50063453 May 22 16:46 train.txt

ls -l decoder
-rw-r--r-- 1 root root 4395628122 May 22 16:48 4-gram.arpa
-rw-r--r-- 1 root root 4395628122 May 22 16:48 4-gram.arpa.lower
-rw-r--r-- 1 root root   42816162 May 22 17:01 decoder-unigram-10000-nbest10.lexicon
```

Train (on single GPU):

```
nohup /root/wav2letter/build/Train train --flagsfile
/root/wav2letter/recipes/models/sota/2019/librispeech/train_am_transformer_ctc.cfg --
minloglevel=0 --logtostderr=1 &
```

```
I0522 17:21:43.144173 64727 Train.cpp:573] Epoch 1 started!
I0522 20:50:28.656316 64727 Train.cpp:345]
epoch:     1 | nupdates:         35146 | lr: 0.400000 | lrcriterion: 0.400000 | runtime:
03:27:50 | bch(ms): 354.82 | smp(ms): 0.65 | fwd(ms): 120.91 | crit-fwd(ms): 8.84 |
bwd(ms): 189.45 | optim(ms): 43.45 | loss:   40.65332 | train-TER: 91.86 | train-WER:
95.17 | dev-clean-loss:   20.00567 | dev-clean-TER: 58.12 | dev-clean-WER: 72.39 | dev-
other-loss:   21.03868 | dev-other-TER: 64.24 | dev-other-WER: 79.34 | avg-isz: 1229 |
avg-tsz: 040 | max-tsz: 101 | hrs:  960.40 | thrpt(sec/sec): 277.25
```

Train (on 4-GPUs):

```
nohup mpirun -n 4 --allow-run-as-root  /root/wav2letter/build/Train train --flagsfile
/root/wav2letter/recipes/models/sota/2019/librispeech/train_am_transformer_ctc.cfg --enable_distributed
true --minloglevel=0 --logtostderr=1 &
```

```
/home/omp/sota# ps -ef | grep train
mpirun -n 4 --allow-run-as-root /root/wav2letter/build/Train train --flagsfile
/root/wav2letter/recipes/models/sota/2019/librispeech/train_am_transformer_ctc.cfg --enable_distributed true --
minloglevel=0 --logtostderr=1

/root/wav2letter/build/Train train --flagsfile
/root/wav2letter/recipes/models/sota/2019/librispeech/train_am_transformer_ctc.cfg --enable_distributed true --
minloglevel=0 --logtostderr=1

/root/wav2letter/build/Train train --flagsfile
/root/wav2letter/recipes/models/sota/2019/librispeech/train_am_transformer_ctc.cfg --enable_distributed true --
minloglevel=0 --logtostderr=1

/root/wav2letter/build/Train train --flagsfile
/root/wav2letter/recipes/models/sota/2019/librispeech/train_am_transformer_ctc.cfg --enable_distributed true --
minloglevel=0 --logtostderr=1

/root/wav2letter/build/Train train --flagsfile
/root/wav2letter/recipes/models/sota/2019/librispeech/train_am_transformer_ctc.cfg --enable_distributed true --
minloglevel=0 --logtostderr=1
```

Configuration:

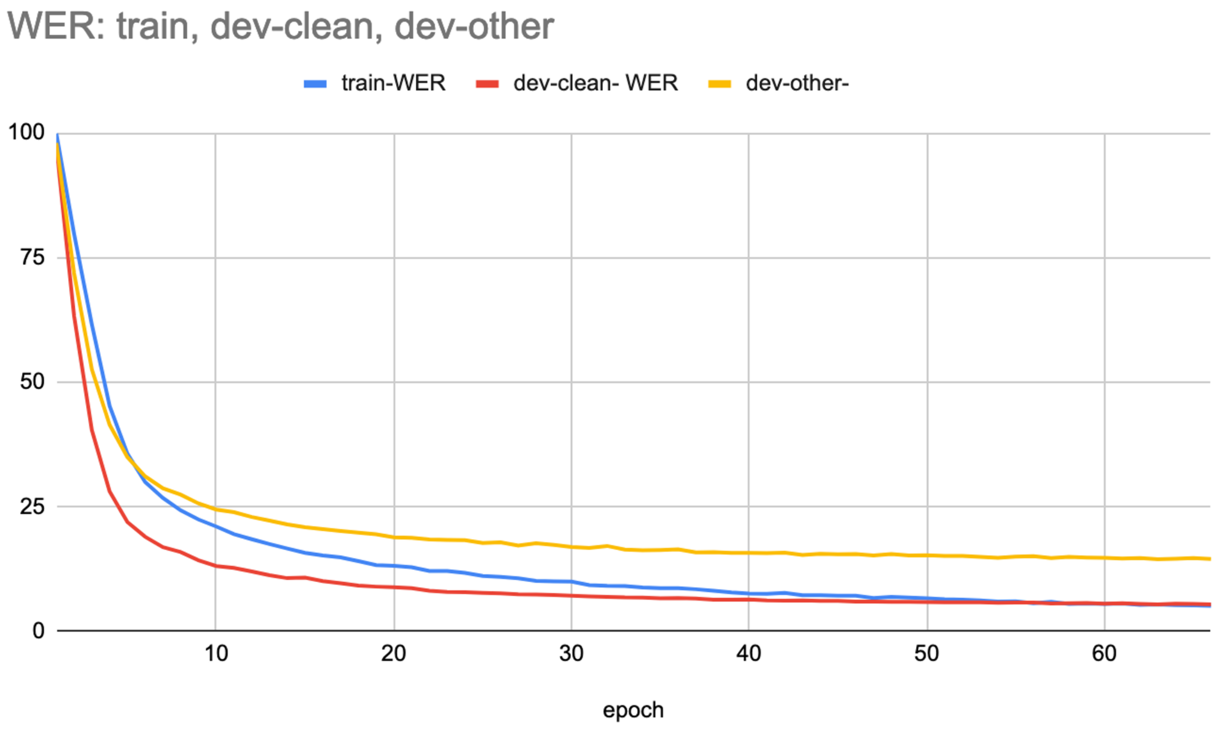
```
--runname=am_transformer_ctc_librispeech-23may945am
--rundir=/home/om/sota
--archdir=/root/wav2letter/recipes/models/sota/2019
--arch=am_arch/am_transformer_ctc.arch
--tokensdir=/home/om/sota/am
--tokens=librispeech-train-all-unigram-10000.tokens
--lexicon=/home/om/sota/am/librispeech-train+dev-unigram-10000-nbest10.lexicon
--train=/home/om/sota/lists/train-clean-100.lst,/home/om/sota/lists/train-clean-
360.lst,/home/om/sota/lists/train-other-500.lst
--valid=dev-clean:/home/om/sota/lists/dev-clean.lst,dev-other:/home/om/sota/lists/dev-other.lst
--criterion=ctc
--mfsc
--usewordpiece=true
--wordseparator=_
--labelsmooth=0.05
--dataorder=output_spiral
--inputbinsize=25
--softwstd=4
--memstepsize=5000000
--pcttraineval=1
--pctteacherforcing=99
--sampletarget=0.01
--netoptim=adadelta
--critoptim=adadelta
--lr=0.4
--lrcrit=0.4
--linseg=0
--momentum=0.0
--maxgradnorm=1.0
--onorm=target
--sqnorm
--nthread=6
--batchsize=8
--filterbanks=80
--minisz=200
--mintsz=2
--minloglevel=0
--logtostderr
--enable_distributed=true
```

Train - after 100 epochs:

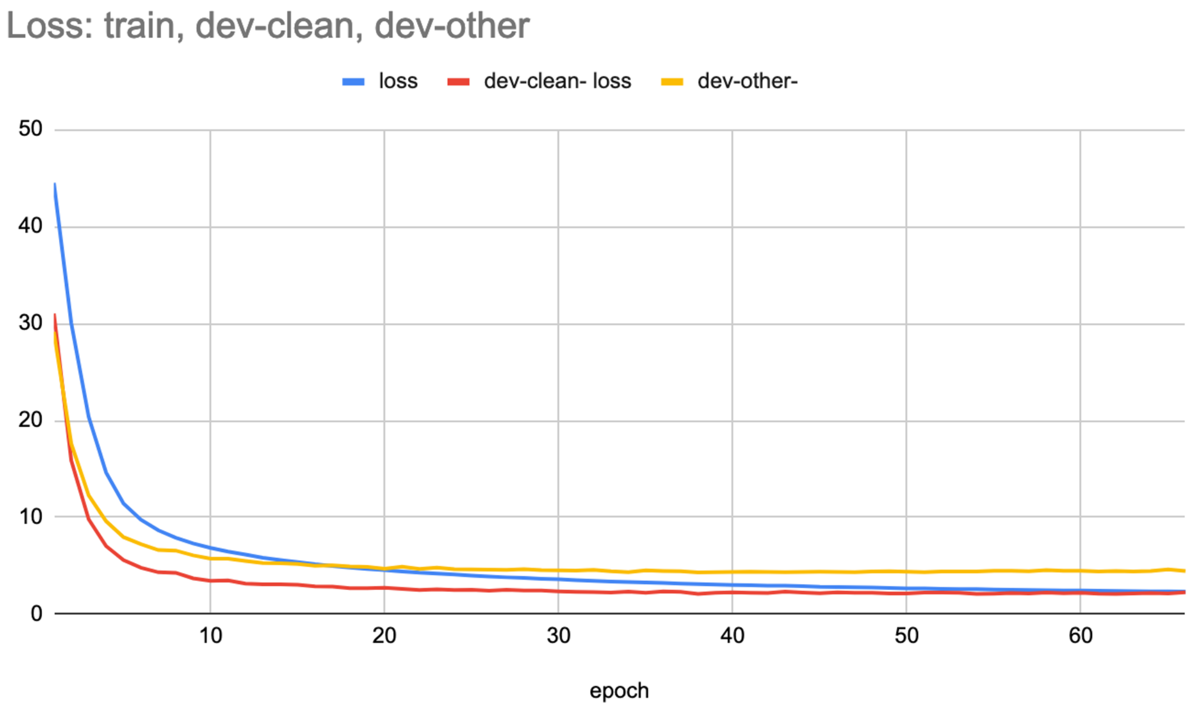
```
epoch:       100 | nupdates:      869913 | lr: 0.400000 | lrcriterion: 0.400000 | runtime:
01:47:48 | bch(ms): 736.20 | smp(ms): 0.66 | fwd(ms): 120.91 | crit-fwd(ms): 8.84 |
bwd(ms): 570.29 | optim(ms): 44.00 | loss:     1.98598 | train-TER:  1.55 | train-WER:
3.57 | dev-clean-loss:           2.20011 | dev-clean-TER:  2.16 | dev-clean-WER:  5.29 |
dev-other-loss:       4.64947 | dev-other-TER:  7.09 | dev-other-WER: 13.92 | avg-isz:
1229 | avg-tsz: 040 | max-tsz: 096 | hrs:  960.51 | thrpt(sec/sec): 534.53
```

```
$ gpustat
dev007                     Sat May 23 05:22:38 2020  440.33.01
[0] Tesla V100-PCIE-16GB | 37'C,  99 % | 12019 / 16160 MB | root(12007M)
[1] Tesla V100-PCIE-16GB | 36'C,  99 % | 11975 / 16160 MB | root(11963M)
[2] Tesla V100-PCIE-16GB | 37'C,  99 % | 12105 / 16160 MB | root(12093M)
[3] Tesla V100-PCIE-16GB | 35'C, 100 % | 12077 / 16160 MB | root(12065M)
```

Train - WER on LibriSpeech



Train - Loss on LibriSpeech



Test:

```
# wav2letter/build/Test --am /home/om/w2l/librispeech_clean_trainlogs/chk-pt-90epochs-7-may-2020/001_model_lists#dev-clean.lst.bin  --maxload 10 --test lists/dev-clean.lst -show
```

Decode:

```
# wav2letter/build/Decoder --am /home/om/w2l/librispeech_clean_trainlogs/chk-pt-90epochs-7-may-2020/001_model_list/w2l/am/lexicon.txt --lm=/home/omprakash.s/w2l/lm/3-gram.arpa -show --tokens=/home/omprakash.s/w2l/am/tokens.txt --lexicon=/home/omprakash.s/
```

Setting up Docker - Issue:

```
(base) om@dev007:~$ sudo docker run --runtime=nvidia --rm -itd --ipc=host -v /home/omprakash.s/w2l-libri-local/:/w2l-libri-local/ -v /home/omprakash.s/w2l-libri-local/wav2letter:/root/wav2letter/ --name w2l-new-local wav2letter/wav2letter:cuda-latest

C17fcc7506fe0ba67b02c5cbed0271355d9ffeb4e65fb528078cc711615f949b

(base) om@ldev007:~$ sudo docker exec -it w2l-new-local bash
```

Audio file read

- While preparing lst files, the process used to terminate if there is any error in reading file
- Had to use 3rd party tool to verify read is working for all audio files before preparing lst files

Seq2Seq model and ASG training

- Was not able to run as the GPU memory was insufficient

Learning rate ramp up

- There was an issue where learning rate didn’t increase and loss was not decreasing
- It was fixed

Adjust AM and LM weight parameter

- Adjusted lm-weight parameter in configuration file

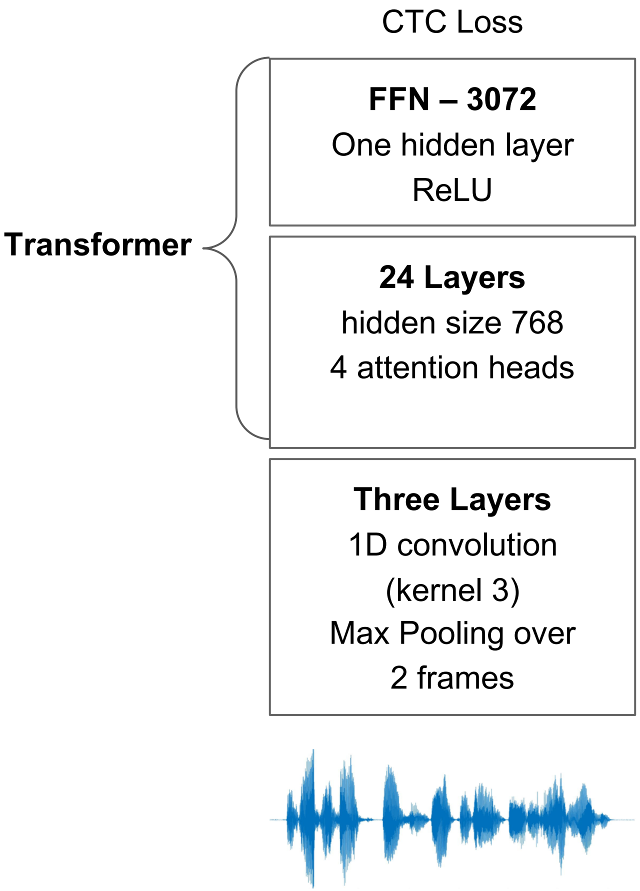
Beam Size

- Reduced the beam size from 250 to 100, and didn’t get degradation

Reduce experimentation time

- Reduced training set to half by removing low confidence samples (from existing ASR system)
- This helped in doing architectural change validation faster

Architecture Evaluation



No. of Layers: 16, 20, 24, 40, 32, 36

Embedding Dimension: 512, 768, 1024, 1536, 2048

No. of heads: 4, 8

Other parameters:

- Beam size, Language Model Weight, Learning rate

4. Which is fed to as 768 embedding dimension input to transformer

3. After 2nd and 3rd convolution and max pooling get frame of 20ms

2. After first convolution and max pool over two frames - get 4 frames of 80ms

1. Eight frames each of 20ms (total 160ms)

Input feature 40, 80, 120 Filterbank

am_transformer_ctc_36_768_3072_8h_30july.arch

```
V -1 1 NFEAT 0
WN 3 C NFEAT 2048 3 1 -1
GLU 2
DO 0.2
WN 3 C 1024 2048 3 1 -1
GLU 2
DO 0.2
WN 3 C 1024 2048 3 1 -1
GLU 2
DO 0.2
WN 3 C 1024 2048 3 1 -1
GLU 2
DO 0.2
WN 3 C 1024 2048 3 1 -1
GLU 2
DO 0.2
WN 3 C 1024 2048 3 1 -1
GLU 2
DO 0.2
M 8 1 81
RO 2 0 3 1
POSEMB 768 460 0.2
TR 768 3072 8 460 0.2 0.2 (36 times)
POSEMB 768 460 0.2
L 768 NLABEL
```

am_transformer_ctc_48x512x2048.arch

```
V -1 1 NFEAT 0
WN 3 C NFEAT 2048 3 1 -1
GLU 2
DO 0.2
WN 3 C 1024 2048 3 1 -1
GLU 2
DO 0.2
WN 3 C 1024 2048 3 1 -1
GLU 2
DO 0.2
WN 3 C 1024 2048 3 1 -1
GLU 2
DO 0.2
WN 3 C 1024 2048 3 1 -1
GLU 2
DO 0.2
WN 3 C 1024 2048 3 1 -1
GLU 2
DO 0.2
M 8 1 81
RO 2 0 3 1
POSEMB 512 460 0.2
TR 512 2048 8 460 0.2 0.2 (48 times)
POSEMB 512 460 0.2
L 512 NLABEL
```

References

[1] LibriSpeech- An ASR Corpus Based On Public Domain Audio Books, Vassil Panayotov, et. al.

[2] Ashish Vaswani et al. Attention is all you need. Adv. NIPS, 2017

[3] A Comparative Study on Transformers vx RNN in Speech Applications IEEE Automatic Speech Recognition and Understanding Workshop 2019.

[4] Jason Li et. al. Jasper: An end-to-end convolutional neural acoustic model. In Interspeech, 2019.

[5] Awni Hannun et. al. Sequence-to-sequence speech recognition with time-depth separable convolutions. Interspeech 2019

[6] Yongqiang Wang, et. al. Transformer-based acoustic modeling for hybrid speech recognition, 2019

[7] Kahn Jacob et. al..Self-training for end-to-end speech recognition. ICASSP 2020

[8] Karel Vesely, et. al.. Semi-supervised DNN training with word selection for ASR. In Interspeech, 2017.

[9] End-to-End ASR: From Supervised to Semi Supervised Learning with Modern Architecture

Thank You for Attending Part-I

For more information visit

www.DeepThinking.AI or email omisonie@gmail.com