DEJA-VU: DOUBLE FEATURE PRESENTATION AND ITERATED LOSS IN DEEP TRANSFORMER NETWORKS

Andros Tjandra[1*], Chunxi Liu[2], Frank Zhang[2], Xiaohui Zhang[2], Yongqiang Wang[2], Gabriel Synnaeve[2], Satoshi Nakamura[1], Goeffrey Zweig[2]

1) NAIST, Japan 2) Facebook AI, USA

facebook Artificial Intelligence

* This work was done while Andros was a research intern at Facebook

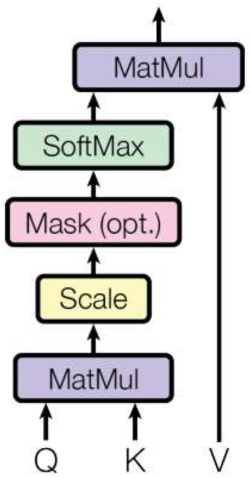
Nara Institute of Science and Technology NAIST

Motivation

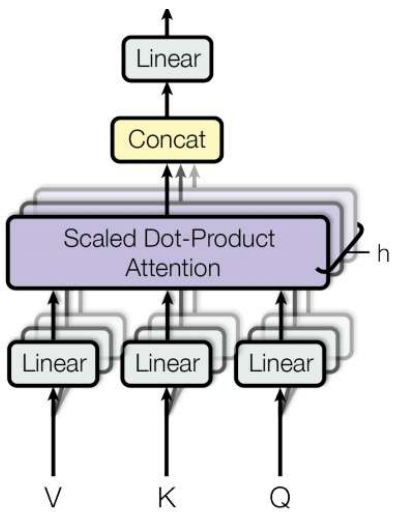
- Make feature processing adaptive to what is being said.
  - Different feature processing, depending on what words need to be differentiated in light of a specific utterance.
- To achieve this, we allow a Transformer Network to (re)-attend to the audio features, using intermediate layer activations as the Query.
- Imposing the objective function on the intermediate layer ensures that it has meaningful information – and trains much faster.
- Net – using these two methods lowers error rate 10-20% for Librispeech and Video ASR datasets.

Review: Self-attention in Transformers

Dot Product Attention



Multihead Self Attention



Transformer module

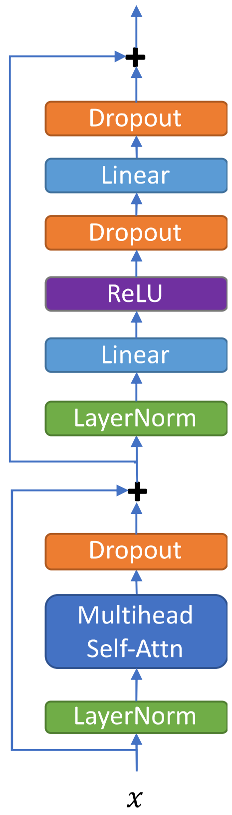
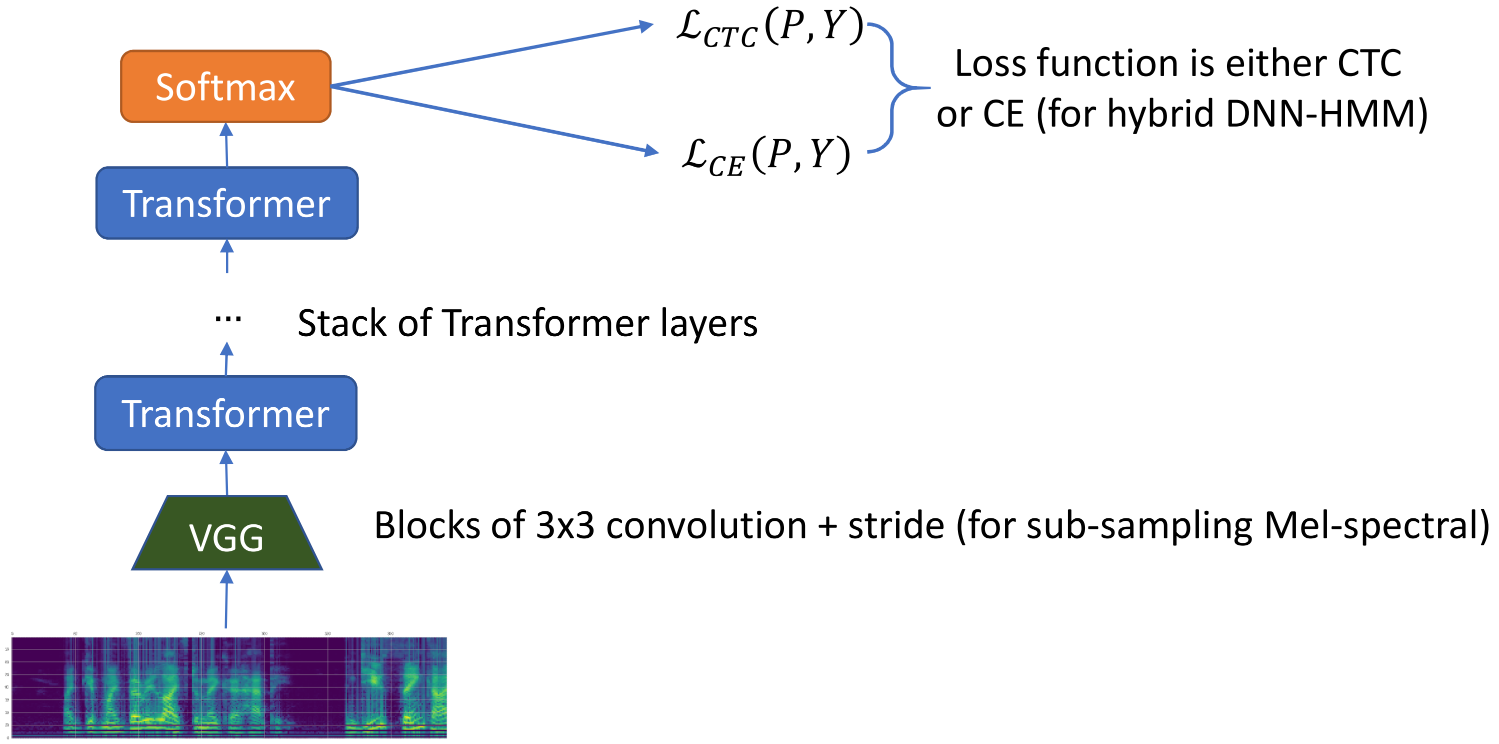


Image ref: Attention is all you need (Vaswani et al., NIPS 2017)

Review: VGG + Transformer Acoustic Model



Problems?

- Stacking more and more layers has empirically give better result.
  - Computer vision: AlexNet (<10 layers) -> VGGNet (20 layers) -> ResNet (>100 layers).

- However, training such deep models are difficult.

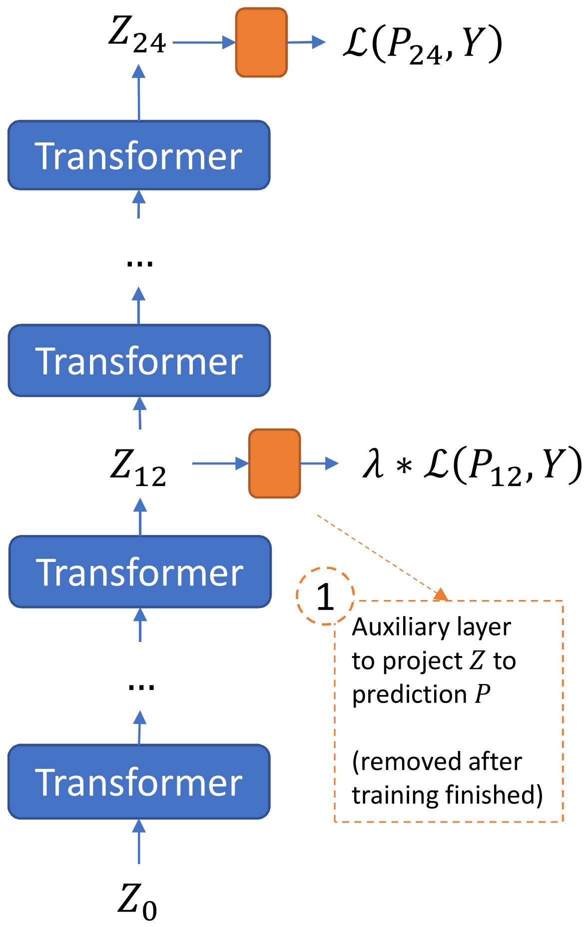
- With improvements in this paper, we can reliably train up to 36 layer networks.

Idea #1: Iterated Loss

- In the deep neural network, the loss are always the furthest node from the input.
- Early nodes (layers) might received less feedback (due to vanishing gradients).
- We add auxiliary loss in the intermediate node.

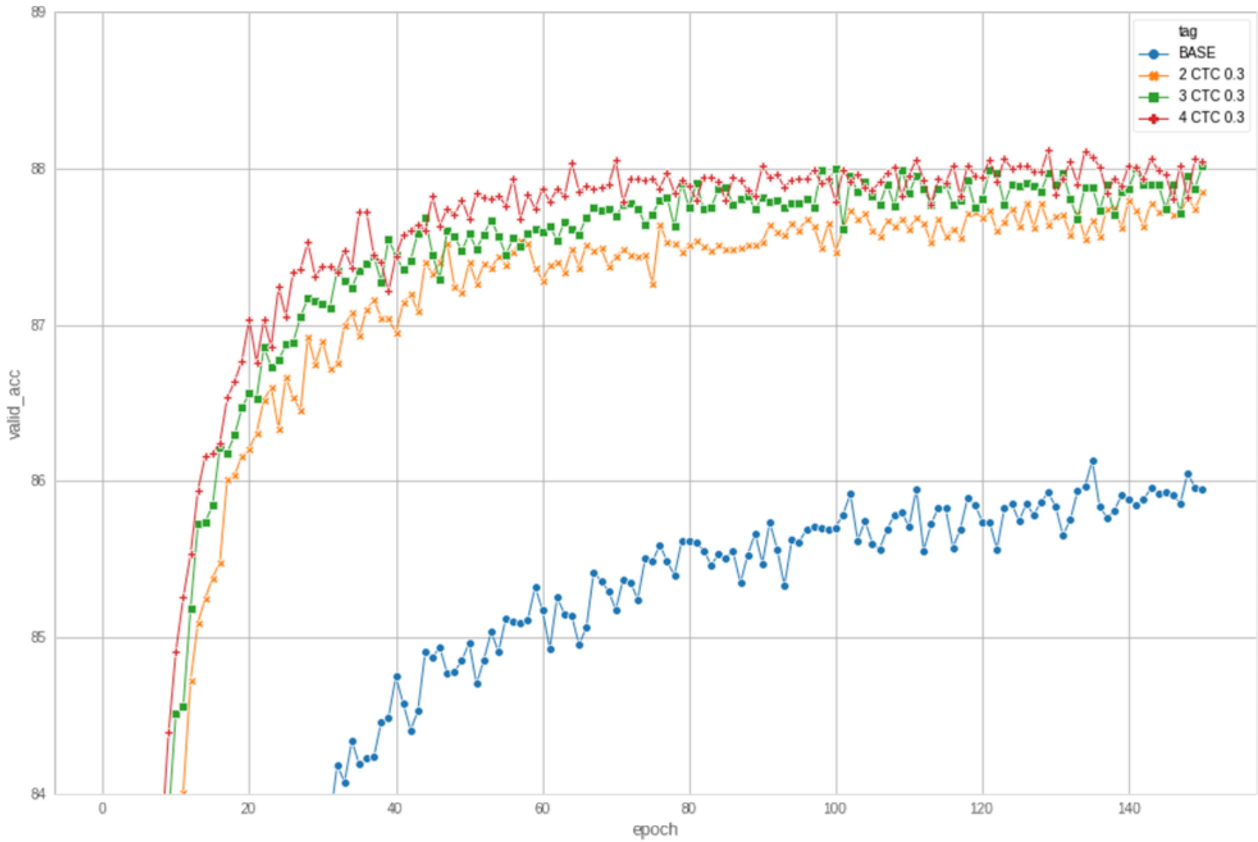
$$P_{k_l} = Softmax\left(MLP_l(Z_{k_l})\right)$$

$$\mathcal{L}_{total} = Loss(P_M, Y) + \lambda \sum_{l=1}^{L} Loss\left(P_{k_l}, Y\right)$$



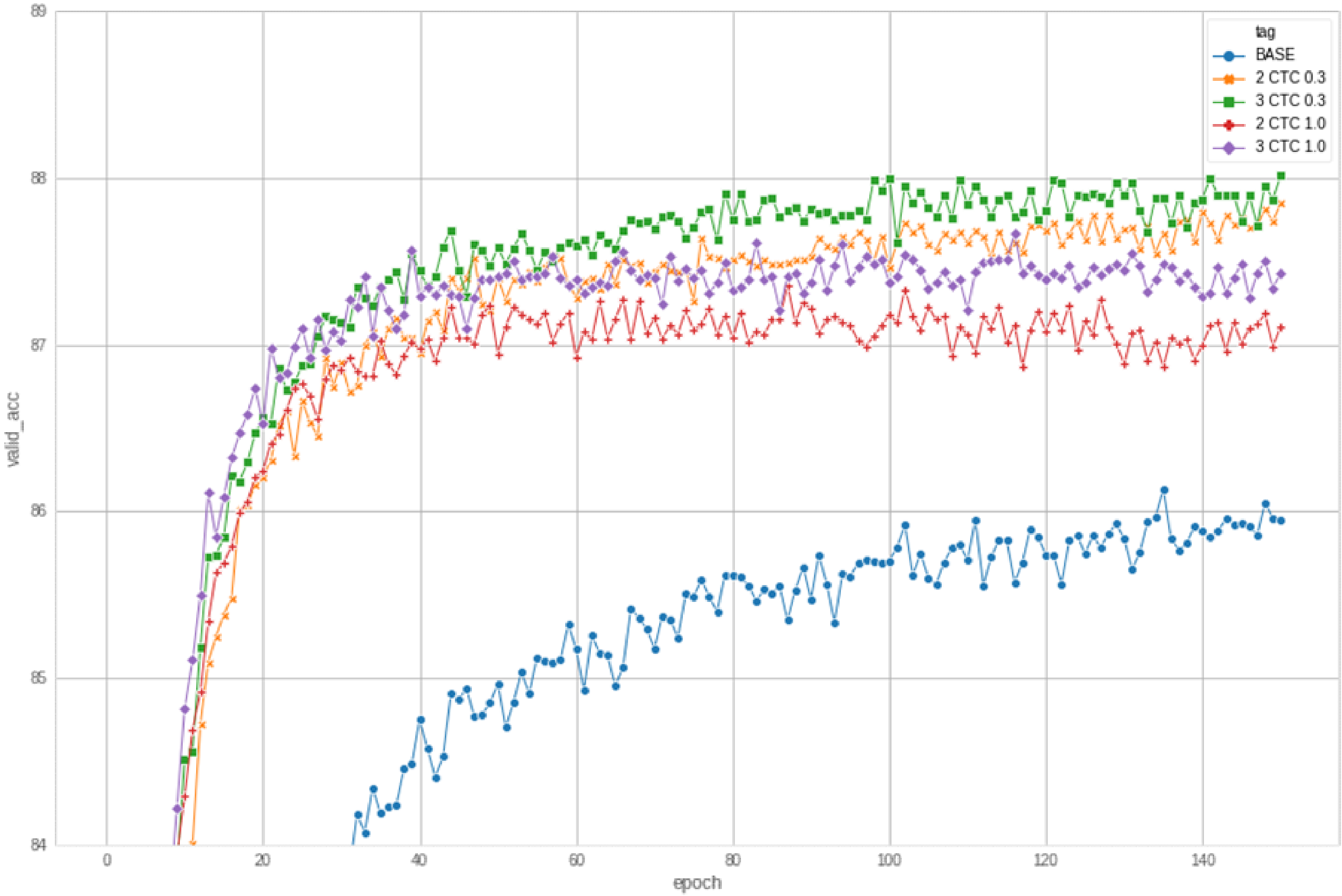
Effect of Iterated Loss

- Comparison:
  - Baseline 1 CTC (24)
  - 2 CTC (12–24)
  - 3 CTC (8-16-24)
  - 4 CTC (6-12-18-24)
- Coeff $\lambda = 0.3$



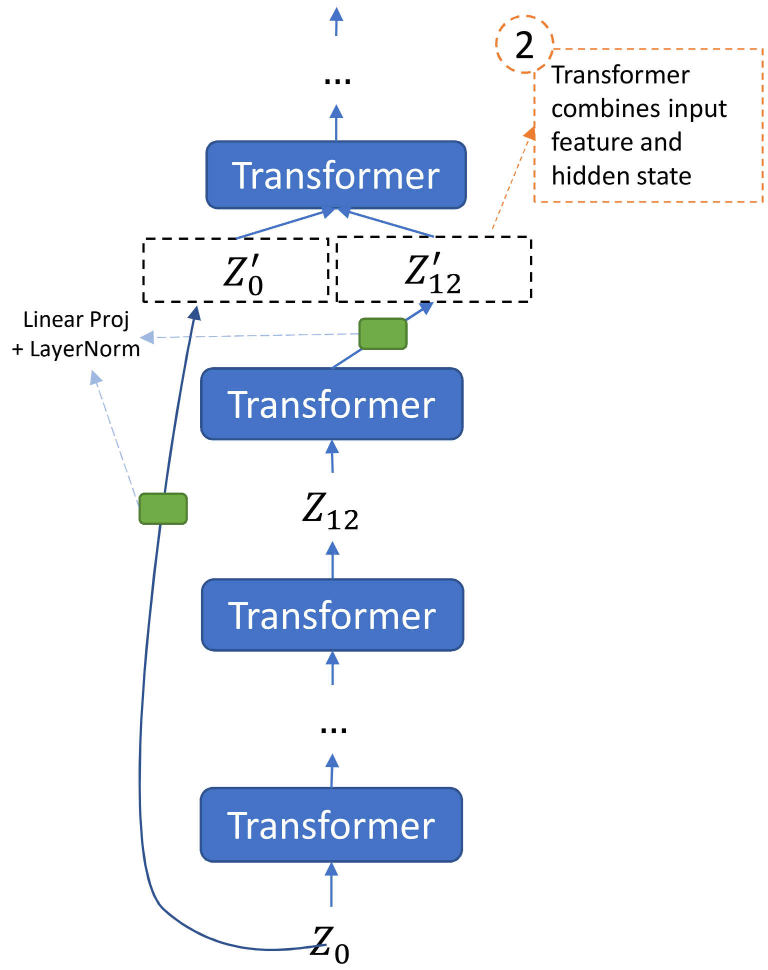
Effect of λ

- λ = 0.3 vs 1.0
- λ = 0.3 consistent better compared to 1.0 on 2 CTC and 3 CTC



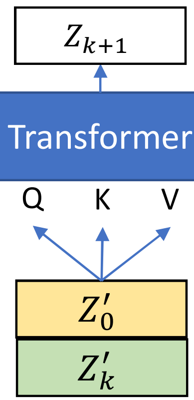
Idea #2: Feature Re-presentation

- After the iterated loss, we want to dynamically integrate the input features.
- Why?
  - The layer after iterated loss might have partial hypothesis.
  - We could find correlated features based on the partial hypothesis.
- There are several ways we have explored (next slide ->)



(Cont.) Feature Concatenation

- (Top) Feature axis. concatenation



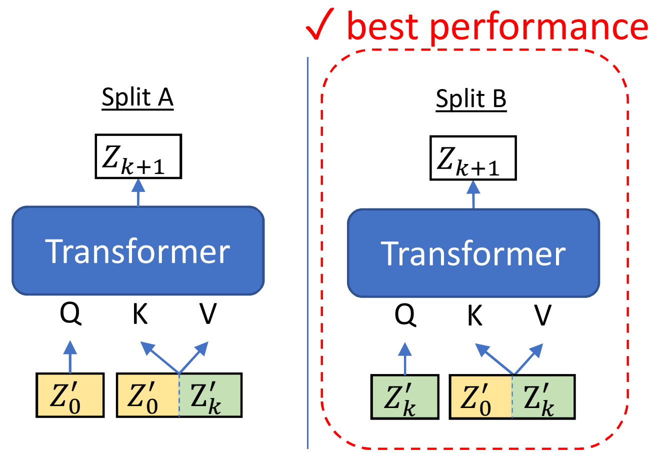
Linear proj. + LN

$$Z_0' = \mathtt{cat}([\mathtt{LayerNorm}(Z_0 W_1), E], \mathtt{dim} = 1)$$
$$Z_k' = \mathtt{cat}([\mathtt{LayerNorm}(Z_k W_2), E], \mathtt{dim} = 1)$$

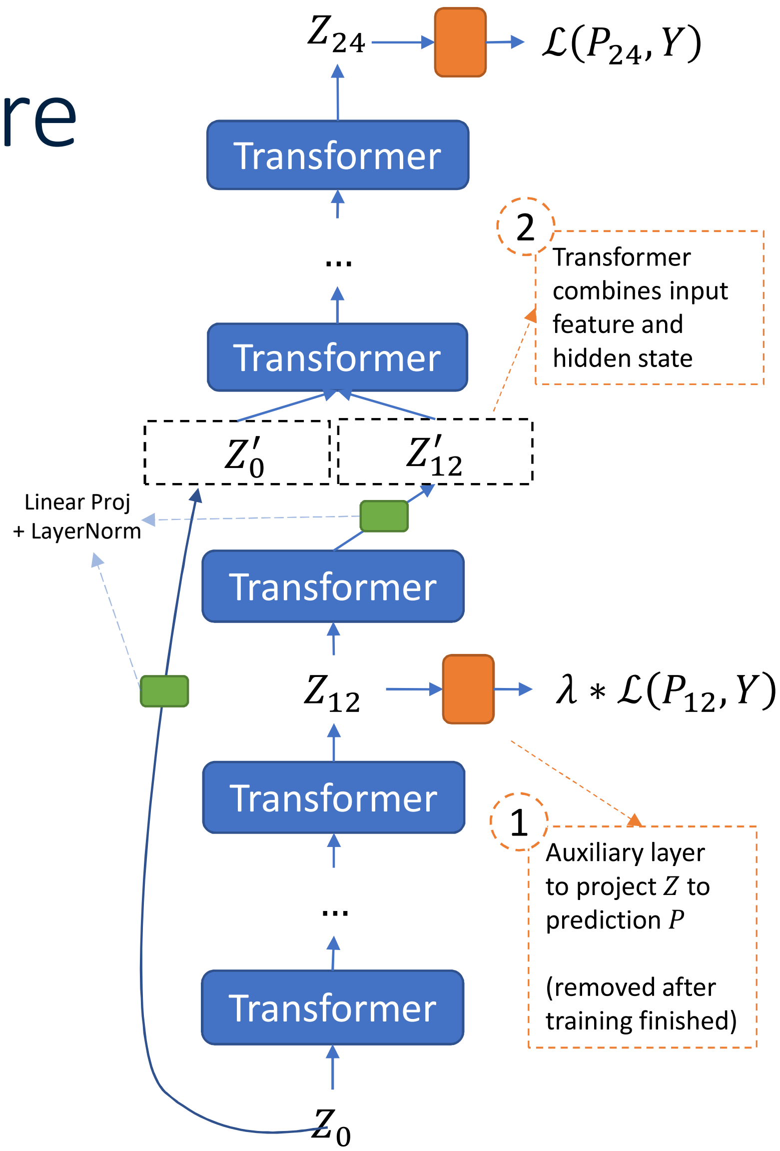
- (Btm) Time axis. Concatenation
  - Split A : input as Query
  - Split B : hidden state as Query

Time Cat + Post Projection

$$O = \mathtt{cat}([Z_0', Z_k'], \mathtt{dim} = 0) \in \mathbb{R}^{2S \times (d_c + d_e)}$$
$$Z_{k+1}' = \begin{cases} \mathtt{Transformer}(\mathtt{Q} = Z_0', \mathtt{K} = O, \mathtt{V} = O), & \text{split A} \\ \mathtt{Transformer}(\mathtt{Q} = Z_k', \mathtt{K} = O, \mathtt{V} = O), & \text{split B} \end{cases}$$
$$Z_{k+1} = \mathtt{LayerNorm}(\mathtt{ReLU}(Z_{k+1}' W_3))$$



Final architecture



Result: Librispeech (CTC w/o data augmentation)

Model	Config	dev		test	
		clean	other	clean	other
CTC Baseline	VGG+24 Trf.	4.7	12.7	5.0	13.1
+ Iter. Loss	12-24	4.1	11.8	4.5	12.2
	8-16-24	4.2	11.9	4.6	12.3
	6-12-18-24	4.1	11.7	4.4	12.0
+ Feat. Cat.	12-24	3.9	10.9	4.2	11.1
	8-16-24	3.7	10.3	4.1	**10.7**
	6-12-18-24	3.6	10.4	**4.0**	10.8

12% test-clean & 8% test-other relative improvement

20% test-clean & 18% test-other relative improvement

Librispeech with data augmentation

Model	Config	LM	test-clean	test-other
CTC (Baseline)	VGG+24 Trf.		4.0	9.4
+ Iter. Loss	8-16-24	4-gram	3.5	8.4
+ Feat. Cat	8-16-24		3.3	7.6
CTC (Baseline)	VGG+36 Trf.		4.0	9.4
+ Iter. Loss	12-24-36	4-gram	3.4	8.1
+ Feat. Cat	12-24-36		**3.2**	**7.2**

Without **iter-loss** & **feat-cat**, increasing Transformer layers doesn't improve performance

With **iter-loss** & **feat-cat**, we still get improvement with deeper Transformer

Librispeech with hybrid DNN-HMM

Model	Config	LM	test-clean	test-other
Hybrid (Baseline)	VGG+24 Trf.	4-gram	3.2	7.7
+ Iter. Loss	8-16-24		3.1	7.3
+ Feat. Cat	8-16-24		2.9	6.7

9% test-clean &
12% test-other
improvement

Video dataset

Model	Config	video		
		curated	clean	other
CTC (Baseline)	VGG+24 Trf.	14.0	17.4	23.6
+ Iter. Loss	8-16-24	13.2	16.7	22.9
+ Feat. Cat	8-16-24	12.4	16.2	22.3
CTC (Baseline)	VGG+36 Trf.	14.2	17.5	23.8
+ Iter. Loss	12-24-36	12.9	16.6	22.8
+ Feat. Cat	12-24-36	**12.3**	**16.1**	**22.3**
Hybrid (Baseline)	VGG+24 Trf	12.8	16.1	22.1
+ Iter. Loss	8-16-24	12.1	15.7	21.8
+ Feat. Cat	8-16-24	**11.6**	**15.4**	**21.4**

13% curated
8% clean
6% other
improvement

9% curated
4% clean
3% other
improvement

Conclusion

- We have proposed a method for re-processing the input features in light of the information available at an intermediate network layer.
- To integrate the features from different layers, we proposed self-attention across layers by concatenating two sequences in time-axis.
- Adding iterated loss in the middle of deep transformers helps the performance (tested on hybrid ASR as well).
- Librispeech: 10-20% relative improvements
- Video: 3.2-13% relative improvements

End of presentation

☺ Thank you for your attention ☺