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

Andros Tjandra\textsuperscript{1*}, Chunxi Liu\textsuperscript{2}, Frank Zhang\textsuperscript{2}, Xiaohui Zhang\textsuperscript{2}, Yongqiang Wang\textsuperscript{2}, Gabriel Synnaeve\textsuperscript{2}, Satoshi Nakamura\textsuperscript{1}, Goeffrey Zweig\textsuperscript{2}

1) NAIST, Japan  
2) Facebook AI, USA

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

Loss function is either CTC or CE (for hybrid DNN-HMM)

\[ \mathcal{L}_{\text{CTC}}(P, Y) \]
\[ \mathcal{L}_{\text{CE}}(P, Y) \]

Stack of Transformer layers

Blocks of 3x3 convolution + stride (for sub-sampling Mel-spectral)
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.

\[
P_{kl} = \text{Softmax} \left( \text{MLP}_l(Z_{kl}) \right)
\]

\[
\mathcal{L}_{\text{total}} = \text{Loss}(P_M, Y) + \lambda \sum_{l=1}^{L} \text{Loss}(P_{kl}, Y)
\]
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 $\lambda$

- $\lambda = 0.3$ vs 1.0
- $\lambda = 0.3$ 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

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

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

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

Transformer

...  

Transformer

\[ Z' \]

Transformer

\[ Z'_{12} \]

Transformer

\[ Z_{12} \]

Transformer

\[ Z_0 \]

Transformer

\[ Z_{24} \rightarrow \mathcal{L}(P_{24}, Y) \]

Transformer combines input feature and hidden state

\[ \lambda \times \mathcal{L}(P_{12}, Y) \]

Auxiliary layer to project \( Z \) to prediction \( P \)

(removed after training finished)
Result: Librispeech (CTC w/o data augmentation)

<table>
<thead>
<tr>
<th>Model</th>
<th>Config</th>
<th>dev</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>clean</td>
<td>other</td>
</tr>
<tr>
<td>CTC Baseline</td>
<td>VGG+24 Trf.</td>
<td>4.7</td>
<td>12.7</td>
</tr>
<tr>
<td>+ Iter. Loss</td>
<td>12-24</td>
<td>4.1</td>
<td>11.8</td>
</tr>
<tr>
<td></td>
<td>8-16-24</td>
<td>4.2</td>
<td>11.9</td>
</tr>
<tr>
<td></td>
<td>6-12-18-24</td>
<td>4.1</td>
<td>11.7</td>
</tr>
<tr>
<td>+ Feat. Cat.</td>
<td>12-24</td>
<td>3.9</td>
<td>10.9</td>
</tr>
<tr>
<td></td>
<td>8-16-24</td>
<td>3.7</td>
<td>10.3</td>
</tr>
<tr>
<td></td>
<td>6-12-18-24</td>
<td>3.6</td>
<td>10.4</td>
</tr>
</tbody>
</table>

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

20% test-clean & 18% test-other relative improvement
Librispeech with data augmentation

<table>
<thead>
<tr>
<th>Model</th>
<th>Config</th>
<th>LM</th>
<th>test-clean</th>
<th>test-other</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTC (Baseline)</td>
<td>VGG+24 Trf.</td>
<td>4-gram</td>
<td>4.0</td>
<td>9.4</td>
</tr>
<tr>
<td>+ Iter. Loss</td>
<td>8-16-24</td>
<td></td>
<td>3.5</td>
<td>8.4</td>
</tr>
<tr>
<td>+ Feat. Cat</td>
<td>8-16-24</td>
<td></td>
<td>3.3</td>
<td>7.6</td>
</tr>
<tr>
<td>CTC (Baseline)</td>
<td>VGG+36 Trf.</td>
<td>4-gram</td>
<td>4.0</td>
<td>9.4</td>
</tr>
<tr>
<td>+ Iter. Loss</td>
<td>12-24-36</td>
<td></td>
<td>3.4</td>
<td>8.1</td>
</tr>
<tr>
<td>+ Feat. Cat</td>
<td>12-24-36</td>
<td></td>
<td><strong>3.2</strong></td>
<td><strong>7.2</strong></td>
</tr>
</tbody>
</table>

# Librispeech with hybrid DNN-HMM

<table>
<thead>
<tr>
<th>Model</th>
<th>Config</th>
<th>LM</th>
<th>test-clean</th>
<th>test-other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hybrid (Baseline)</td>
<td>VGG+24 Trf.</td>
<td>4-gram</td>
<td>3.2</td>
<td>7.7</td>
</tr>
<tr>
<td>+ Iter. Loss</td>
<td>8-16-24</td>
<td></td>
<td>3.1</td>
<td>7.3</td>
</tr>
<tr>
<td>+ Feat. Cat</td>
<td>8-16-24</td>
<td></td>
<td>2.9</td>
<td>6.7</td>
</tr>
</tbody>
</table>

9% test-clean & 12% test-other improvement
## Video dataset

<table>
<thead>
<tr>
<th>Model</th>
<th>Config</th>
<th>video</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>curated</td>
</tr>
<tr>
<td>CTC (Baseline)</td>
<td>VGG+24 Trf.</td>
<td>14.0</td>
</tr>
<tr>
<td>+ Iter. Loss</td>
<td>8-16-24</td>
<td>13.2</td>
</tr>
<tr>
<td>+ Feat. Cat</td>
<td>8-16-24</td>
<td>12.4</td>
</tr>
<tr>
<td>CTC (Baseline)</td>
<td>VGG+36 Trf.</td>
<td>14.2</td>
</tr>
<tr>
<td>+ Iter. Loss</td>
<td>12-24-36</td>
<td>12.9</td>
</tr>
<tr>
<td>+ Feat. Cat</td>
<td>12-24-36</td>
<td><strong>12.3</strong></td>
</tr>
<tr>
<td>Hybrid (Baseline)</td>
<td>VGG+24 Trf.</td>
<td>12.8</td>
</tr>
<tr>
<td>+ Iter. Loss</td>
<td>8-16-24</td>
<td>12.1</td>
</tr>
<tr>
<td>+ Feat. Cat</td>
<td>8-16-24</td>
<td><strong>11.6</strong></td>
</tr>
</tbody>
</table>

- **13% curated**
- **8% clean**
- **6% other**

- **9% curated**
- **4% clean**
- **3% other**

Improvement:

- 13% curated
- 8% clean
- 6% other
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 😊