Synchronous Transformers For End-To-End Speech Recognition

用于端到端语音识别的同步变压器

*田正坤*1*,*2*，姜彦毅*1*，叶白*1*,*2*，陶建华*1*,*2*,*3*，张帅*1*,*2*，郑启文*1 1 自动化研究所模式识别国家实验室，

中国科学院北京2中国科学院大学人工智能学院北京3

中国科学院脑科学与智能技术英才中心，北京，中国

# 摘要

For most of the attention-based sequence-to-sequence models, the decoder predicts the output sequence conditioned on the entire input sequence processed by the encoder. The asynchronous problem between the encoding and decoding makes these models difficult to be applied for online speech recognition. In this paper, we propose a model named synchronous transformer to address this problem, which can predict the output sequence chunk by chunk. Once a fixed-length chunk of the input sequence is processed by the encoder, the decoder begins to predict symbols immediately. During training, a forward-backward algorithm is introduced to optimize all the possible alignment paths. Our model is evaluated on a Mandarin dataset AISHELL-1. The experiments show that the synchronous transformer is able to perform encoding and decoding synchronously, and achieves a character error rate of 8.91% on the test set.

Index Terms— Asynchronous Problem, Online Speech Recognition, Synchronous Transformer, Chunk by Chunk, Forward-Backward Algorithm

对于大多数基于注意的序列到序列模型，解码器根据编码器处理的整个输入序列预测输出序列。编码和解码之间的异步问题使得这些模型难以应用于在线语音识别。针对这一问题，本文提出了一种同步变压器模型，该模型可以逐块预测输出序列。一旦编码器处理了输入序列的固定长度块，解码器就立即开始预测符号。在训练过程中，引入了一种前向-后向算法来优化所有可能的对齐路径。我们的模型在中文数据集AISHELL-1上进行了评估。实验结果表明，该同步变换器能够同步进行编码和解码，在测试集上实现了8.91%的字符错误率。

*索引项*-异步问题，在线语音识别，同步变换，逐块，前后向算法

# 1.    导言

基于注意的序列到序列模型[1，2，3，4，5，6]，特别是变压器模型[2]，在各种任务中都取得了巨大的成功，例如神经机器翻译[1，2]，图像字幕[3]和语音识别[4，5，6]。

对于传统的基于注意的序列到序列模型，推理过程可分为两个阶段。编码器首先将整个输入序列处理成高级状态序列。然后，解码器根据先前预测的符号和从整个编码状态序列中提取的上下文向量预测输出序列。这使得模型对序列进行异步编码和解码，并阻止了它应用于在线语音识别。有一些作品试图解决这个问题。Tjandra等人[7]提出了一种局部单调注意机制，该机制迫使模型在每个解码步骤预测中心位置，并仅在中心位置附近计算软注意权重。

然而，仅仅基于有限的信息很难准确预测下一个中心位置。提出了一种基于预测选择概率自适应地将编码状态序列分割成小块的单调分块注意算法[8]。但复杂而棘手的培训方法使其难以实施。触发注意[9]利用连接主义时间分类（CTC）模型产生的尖峰将序列分割成多个状态块，然后解码器以分块的方式预测输出序列。然而，触发注意需要强制对齐来辅助模型训练。大多数提出的模型引入了额外的组件，并且有非常复杂的训练方法。

本文提出了一种同步变换器模型（synctransformer），它可以同时进行编码和解码。同步变压器将变压器[6]和自我注意传感器（SA-T）[10]结合得很深。与原始变压器类似，同步变压器有一个编码器和一个解码器。为了消除自我注意机制对未来信息的依赖性，我们首先强制编码器中的每个节点只关注其左侧上下文，而完全忽略其右侧上下文。一旦编码器产生一个固定长度的状态序列块，解码器立即开始预测符号。类似于神经传感器[11，12，13]，解码器逐块生成输出序列。然而，受RNNs的时变特性的限制，神经传感器模型只能优化块序列对应的近似最佳对齐路径。相比之下，我们采用前向-后向算法来优化所有可能的对齐路径，并计算负对数损失函数，与RNN传感器[14]和SA-T[10]相同。我们在中文数据集AISHELL-1上评估了我们的同步转换器。实验结果表明，该同步变换器能够同步地对序列进行编解码，并取得了与该变换器相当的性能。

本文的其余部分组织如下。第2节描述了我们提出的同步变压器。第三节介绍了我们的实验装置和结果。结论和未来的工作将在第4节中给出。

|  |
| --- |
| **推论**  **编码器输出（右移）**            （a） 同步变压器的结构和推理过程（b）译码器的结构（c）输出概率格  图1。（a） 阐述了同步变压器的整体结构和推理过程。同步变压器由编码器和解码器组成。编码器中的每个节点只关注其左侧上下文。解码器逐块生成符号。一旦编码器处理了一个固定长度的序列块，解码器立即开始预测输出符号。（b） 说明解码器的详细信息。（c） 说明了一个输出概率格，用于通过前向-后向算法对所有可能对齐路径的概率求和。 |

# 2.    同步变压器

## 2.1.    模型

与变压器[2]类似，同步变压器由编码器和解码器组成，如图1（a）所示。编码器和解码器都是由多磁头和前馈层组成[2，6，10]。

如图1（a）所示，我们在编码器的底部放置了一个二维卷积前端，对输入语音特征序列进行简单的处理，包括维数变换（将特征维数从40变换到256）、时间轴下采样和添加正余弦位置信息[2]。作为输入特征序列，处理后的序列可以表示为，其中和分别是这两个序列的长度。*十*1： T型*s码*1： 我*T型我*

为了摆脱对整个输入状态序列的依赖，我们对原始的自注意编码器进行了以下修改，如图1（a）所示。一方面，在计算自我注意权重时，我们强制编码器中的每个节点将注意力集中在其左侧上下文上，而完全忽略其右侧上下文。虽然每个中间节点只能对局部依赖性信息进行建模，但是编码器的顶部节点仍然可以对长期依赖性进行建模。另一方面，与transformer xl[15]类似，Sync transformer的编码器逐块处理输入序列。两个相邻的块之间存在重叠，以保持块之间信息的平滑过渡。对于已处理的输入序列，编码器可以将其拆分为编码状态块。这意味着注意力权重的计算仅仅依赖于一个长度块，而不是整个输入序列。设为两个相邻块的重叠长度。和之间的关系可以表示为*s码*1： 我*米C级*1： 米*W型B类米我*

(1)

在每个解码步骤，解码器预测以先前预测的符号（0≤u≤u+1）和一个块为条件的符号。一旦预测到hi符号，解码器将切换到下一块并继续解码。这个过程可以用下面的公式来表示。*是的*0:u−1*黑色*

*第*（yu | y0:u−）=解码器（y0:u−）（2），其中（1≤m≤m）表示编码器产生的第-个块。1*，厘米*1*，厘米厘米米*

## 2.2.    培训

培训过程分为两个步骤。为了加快算法的收敛速度，首先利用训练好的变压器模型对同步变压器的参数进行初始化。然后应用下面的前后向算法来训练同步变压器。

编码器将输入序列分成块。每个块的解码过程以hi结束。很难弄清楚每个目标符号应该属于哪个块。因此，我们通过计算每个块中所有目标符号的概率来构造输出概率格图，如图1（c）所示。给定输入序列，通过对所有可能对齐路径的概率求和来计算输出的概率。（y1:U | x1，T）=X p（y12（U+M）| x1:T）（3）∈Y*米黑色十*1： T型*是的*1： U型*第年，…，y是的*

其中Y表示所有可能对齐路径的集合。通过列举所有可能的对齐路径来计算概率（y1:U | x1，T）既困难又低效。因此，与[16，10]中基于传感器的模型一样，我们引入了一种前向-后向算法来有效地计算概率。*第*

前向变量（m，u）表示所有可能路径的概率之和，这些路径以起始符号（=hi）开始，以第in个块结束。给定第个块和预测的符号序列，预测hi和的概率分别表示为（m，u−1）和（m，u−1）。对于所有1≤m≤m和1≤u≤u+1，可以使用*α是的*0 *黑色于米米是的*0:u−1*黑色于φ于*

*α*（m，u）=α（m−1，u）φ（m，u）

(4)

+α（m，u−1）yu（m，u−1）

所有路径都以（=hi）开头，表示（1，1）=1。概率（y1:U | x1，T）可以用终端节点的前向变量表示。*是的*0 *黑色α第*

*第*（y1:U | x1，T）=α（M，U+1）φ（M，U+1）(5)

类似地，后向变量（m，u）表示所有可能路径的概率之和，这些路径从第in个块开始，以最后一个块中的（=hi）结束。向后变量可以表示为*β于米于*+1*黑色*

*β*（m，u）=β（m+1，u）φ（m，u）

(6)

+β（m，u+1）yu+1（m，u），其中初始条件（m，u+1）=φ（m，u+1）。*β*

给定一个输入特征序列和一个目标序列，概率（y1:U | x1，T）等于（m，U）β（m，U）的和（m，U）在通过节点的任何左上角到右下角的对角线上。即，∀n:2≤n≤U+M+1p（y1:U | x1，T）=（M，U）β（M，U）（7）*十*1，T*是的*1： U型*第α*十*α*

（m，u）：m+u=n

我们训练模型以最小化负对数损失函数L=−ln（y1:U | x1，T）。梯度的计算与RNN-T完全相同[16]。*第*

## 2.3.    推论

推理过程如图1（a）所示。在推理期间，解码器将预测以编码状态序列的固定长度块为条件的输出符号和所有先前预测的非空白符号。它可以预测块中的一个或多个符号。一旦预测到hi，它将切换到下一个块并继续解码。解码器将重复上述步骤，直到处理完所有的块。为了简化推理过程，我们不尝试合并一些具有相同前缀的对齐路径。*黑色*

# 3.    实验和结果

## 3.1.    数据集

在本研究中，所有的实验都是在公共汉语语音语料库AISHELL-1[1][17]上进行的。这个训练集包含了大约150个小时的演讲（120098个话语），由340名演讲者录制。开发集包含大约20个小时（14326个话语），由40位演讲者录制。大约10小时（7176个话语）的语音被用作测试集。不同组的扬声器不重叠。

## 3.2.    实验装置

对于所有的实验，我们使用40维Mel滤波器组系数（Fbank）特征，这些特征是在25ms窗口上计算的，偏移10ms。每个特征被重新缩放，使每个说话人的平均值和单位方差为零。我们选择4232个字符（包括一个空白符号“hi”和一个未知符号“hi”）作为模型单位。*黑色镡*

我们利用Kaldi[2]进行数据准备。我们的SyncTransformer是建立在ESPNet[18]和warp rnnt[3]上的。它由6个编码块和6个解码器块组成。多头注意力有8个头部。2D卷积前端采用两层时间轴CNN，具有ReLU激活、步长2、通道256和内核大小3。多头注意力和前馈层的输出大小为256。为了加速收敛，我们用选通线性单元代替前馈网络中的ReLU激活函数[19]。根据经验，我们将编码器中每个节点的左上下文设置为20，右上下文设置为0。今后将探讨更多的上下文参数设置。此外，我们还采用了Adam优化器，其预热步骤为25000步，学习率调度器如[2]所述。

在解码过程中，我们使用宽度为5的波束搜索进行所有实验。并将块中生成的符号的最大长度设置为10。我们使用字符错误率（CER）来评估不同模型的性能。

## 3.3.    结果

### 3.3.1.    不同窗长的比较

我们首先探讨不同长度的区块如何影响Sync Transformer的性能。对于每个实验，相邻块的重叠范围被设置为块长度的20%。如表1所示，块长度为10的同步转换器在测试集上可以达到9.06%的CER。当长度减少到5时，同步变压器仍然表现良好。但是，如果长度大于20，则会导致性能严重下降。我们假设在一个长块中可能有多个字符，这可能使模型难以准确预测输出序列。*W型*

表1。不同窗长的比较（CER%）。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *W型* | 5 | 10 | 15 | 20 | 25 |
| 开发 | 8.64 | 7.99 | 8.57 | 8.68 | 11.04 |
| 试验 | 9.73 | 9.06 | 9.51 | 9.76 | 11.71 |

表2。不同搭接长度的比较（CER%）。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *B类* | 4 | 3 | 2 | 1 | 0 |
| 开发 | 8.60 | 7.91 | 7.99 | 9.53 | 9.61 |
| 试验 | 9.56 | 8.91 | 9.06 | 10.39 | 10.47 |

固定长度表示的语音段长度为×4×10ms，其中4表示2D卷积前端可以将语音长度缩短4倍，10ms表示帧移。同步转换器可以达到一个有竞争力的性能取决于0.4s的延迟。如果考虑到重叠，它实际上是0.32s。*W型*

当块长度为1时，Sync Transformer类似于转换器模型，它逐帧解码序列。反过来，当长度足够大时，任何话语都有一个块。在这种情况下，它相当于变压器模型。

### 3.3.2.    不同搭接长度的比较

接下来，我们试图找出块之间的重叠对性能的影响。基于之前的实验，我们将本节中所有实验的块长度设置为10。从表2中，我们发现语块之间的重叠起着重要的作用。在测试装置上，重叠长度为3的同步变压器可达到8.91%的CER。当重叠设置为1或0时，两个相邻块之间的重叠过少可能是性能下降的主要原因。因此，我们假设适当的重叠可以保持信息流在块之间的平滑过渡。当重叠设置为4时，模型的性能也会降低。我们猜测，较大的重叠将导致两个相邻块中包含的信息非常相似，这将进一步降低模型的性能。

### 3.3.3.    与其他端到端模型的比较

我们还比较了同步变压器与其他端到端模型。变压器模型是根据ESPnet[18]中的配方进行训练的，该配方的设置与我们的Sync transformer相同。第二列指示模型是否可以流式解码。第三列表示解码一个长句子所需的步数。是块的数量，是语音帧的数量。*U型米T型*

实验表明，该同步变换器可以达到与最佳变换器相当的效果，优于LAS[20]、RNN-T和我们以前的（ChunkFlow）SA-T[10]。相比之下，同步变压器可以实现在线解码只有一点性能下降。

表3。与其他模型的比较（CER%）。



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 模型 | 在线的 | 步骤 | 开发 | 试验 |
| 拉斯维加斯[20] | 不 | U型 | - | 10.56 |
| 变压器 | 不 | U型 | 7.80 | 8.64 |
| RNN-T[10] | 不 | T型 | 10.13 | 11.82 |
| SA-T[10] | 不 | T型 | 8.30 | 9.30 |
| 块流SA-T[10] | 是的 | T型 | 8.58 | 9.80 |
| 同步变压器 | 是的 | U+M公司 | 7.91 | 8.91 |



不同模型的解码步骤之间的关系是≤U+M≤T。大多数基于注意的模型，如LAS和transformer，在推理过程中需要最少的步骤。然而，由于注意力对整个输入序列的依赖性，它们不能直接应用于在线语音识别任务。块流SAT是一种无RNN的传感器模型，它逐帧解码序列。而且在训练过程中也会消耗更多的记忆。然而，与SA-T和RNN-T相比，Sync Transformer需要更少的解码步骤，这意味着更少的内存和时间开销。*U型*

# 4.    结论与讨论

在本文中，我们提出了一个同步变压器，它结合了变压器和传感器模型的优点。为了摆脱对整个输入状态序列的依赖性，在计算自我注意时，我们强制编码器中的每个节点聚焦于其左侧上下文，而完全忽略其右侧上下文。一旦编码器产生一个固定长度的状态序列块，解码器立即开始预测符号。在训练过程中，我们引入了一种前向-后向算法来求所有可能对齐路径的概率之和，并应用负logloss函数来优化同步变压器。实验结果表明，同步变压器可以实现同步编解码。更重要的是，它优于我们以前的自我关注传感器，并取得了与先进的变压器模型相当的结果。实验结果表明，同步变压器是一种很有前途的在线语音识别模型。但是，还有一些方面需要改进。例如，Sync Transformer在推理过程中可能会产生更多的delete错误，并识别出发音相似的错误单词。所有这些都阻止同步变压器立即应用。这将是我们下一步的研究方向。

# 5.    承认

这项工作得到了国家重点研究发展计划（编号：2018YFB1005003）的支持

国家自然科学基金项目（61831022号、61771472号、61773379号、61901473号）和Inria-CAS联合研究项目（173211KYSB2017 0061号、173211KYSB20190049号）。

# 6.    参考文献

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, “Neural machine translation by jointly learning to align and translate,” arXiv preprint arXiv:1409.0473, 2014.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, “Attention is all you need,” in Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.

[3] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan, “Show and tell: A neural image caption generator,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 3156–3164.

[4] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio, “Attention-based models for speech recognition,” in Advances in neural information processing systems, 2015, pp. 577–585.

[5] Suyoun Kim, Takaaki Hori, and ShinjiWatanabe, “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). IEEE, 2017, pp. 4835–4839.

[6] Linhao Dong, Shuang Xu, and Bo Xu, “Speechtransformer: a no-recurrence sequence-to-sequence model for speech recognition,” in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018, pp. 5884–5888.

[7] Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura, “Local monotonic attention mechanism for endto- end speech and language processing,” arXiv preprint arXiv:1705.08091, 2017.

[8] Chung-Cheng Chiu and Colin Raffel, “Monotonic chunkwise attention,” arXiv preprint arXiv:1712.05382, 2017.

[9] Niko Moritz, Takaaki Hori, and Jonathan Le Roux, “Triggered attention for end-to-end speech recognition,” in ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019, pp. 5666–5670.

[10] Zhengkun Tian, Jiangyan Yi, Jianhua Tao, Ye Bai, and Zhengqi Wen, “Self-attention transducers for end-toend speech recognition.,” in Interspeech, 2019, pp. 4395–4399.

[11] Navdeep Jaitly, David Sussillo, Quoc V Le, Oriol Vinyals, Ilya Sutskever, and Samy Bengio, “A neural transducer,” arXiv preprint arXiv:1511.04868, 2015.

[12] Navdeep Jaitly, Quoc V Le, Oriol Vinyals, Ilya Sutskever, David Sussillo, and Samy Bengio, “An online sequence-to-sequence model using partial conditioning,” in Advances in Neural Information Processing Systems, 2016, pp. 5067–5075.

[13] Tara N Sainath, Chung-Cheng Chiu, Rohit Prabhavalkar, Anjuli Kannan, Yonghui Wu, Patrick Nguyen, and ZhiJeng Chen, “Improving the performance of online neural transducer models,” in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018, pp. 5864–5868.

[14] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, “Speech recognition with deep recurrent neural networks,” in 2013 IEEE international conference on acoustics, speech and signal processing. IEEE, 2013, pp. 6645–6649.

[15] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov, “Transformer-xl: Attentive language models beyond a fixed-length context,” arXiv preprint arXiv:1901.02860, 2019.

[16] Alex Graves, “Sequence transduction with recurrent neural networks,” arXiv preprint arXiv:1211.3711, 2012.

[17] Hui Bu, Jiayu Du, Xingyu Na, Bengu Wu, and Hao Zheng, “Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline,” in 2017 20th Conference of the Oriental Chapter of the International Coordinating Committee on Speech Databases and Speech I/O Systems and Assessment (O-COCOSDA). IEEE, 2017, pp. 1–5.

[18] Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, et al., “Espnet: End-to-end speech processing toolkit,” arXiv preprint arXiv:1804.00015, 2018.

[19] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier, “Language modeling with gated convolutional networks,” in Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017, pp. 933–941.

[20] Changhao Shan, Weng Chao, Wang Guangsen, Su Dan, Luo Min, Yu Dong, and Xie Lei, “Component fusion: Learning replaceable language model component for end-to-end speech recognition system,” 2019.

[[1]](" \l "_ftnref1" \o ")http://www.openslr.org/13/

[[2]](" \l "_ftnref2" \o ")https://github.com/kaldi-asr/kaldi

[[3]](" \l "_ftnref3" \o ")https://github.com/1ytic/warp-rnnt