# WeNet 2.0: More Productive End-to-End Speech Recognition Toolkit

**Binbin Zhang**\(^{1,3}\), **Di Wu**\(^{1,3}\), **Zhendong Peng**\(^{1,3}\), **Xingchen Song**\(^{1,3}\), **Zhuoyuan Yao**\(^2\), **Hang Li**\(^{2,3}\), **Lei Xie**\(^{2,*}\), **Chao Yang**\(^{1,3}\), **Fuping Pan**\(^1\), **Jianwei Niu**\(^1\)

\(^1\)Horizon Robotics, Beijing, China

\(^2\)Audio, Speech and Language Processing Group (ASLP@NPU), School of Computer Science, Northwestern Polytechnical University, Xi’an, China

\(^3\)WeNet Open Source Community

binbin.zhang@horizon.ai, hanglv@nwpu-aslp.org, lxie@nwpu.edu.cn

### Abstract

Recently, we made available WeNet\(^1\), a production-oriented end-to-end speech recognition toolkit, which introduces a unified two-pass (U2) framework and a built-in runtime to address the streaming and non-streaming decoding modes in a single model. To further improve ASR performance and facilitate various production requirements, in this paper, we present WeNet 2.0 with four important updates. (1) We propose U2++, a unified two-pass framework with bidirectional attention decoders, which includes the future contextual information by a right-to-left attention decoder to improve the representative ability of the shared encoder and the performance during the rescoring stage. (2) We introduce an n-gram based language model and a WFST-based decoder into WeNet 2.0, promoting the use of rich text data in production scenarios. (3) We design a unified contextual biasing framework, which leverages user-specific context (e.g., contact lists) to provide rapid adaptation ability for production and improves ASR accuracy in both with-LM and without-LM scenarios. (4) We design a unified IO to support large-scale data for effective model training. In summary, the brand-new WeNet 2.0 achieves up to 10% relative recognition performance improvement over the original WeNet on various corpora and makes available several important production-oriented features.

**Index Terms:** U2++, Language Model, Contextual Biasing, UIO, Toolkit

### 1. Introduction

End-to-end (E2E) methods, such as connectionist temporal classification (CTC)\(^2\), recurrent neural network transducer (RNN-T)\(^{3,4,5}\), and attention based encoder-decoder (AED)\(^{6,7,8,9}\), have drawn immense interest in the last few years in automatic speech recognition (ASR). Recent works\(^{10,11,12,13,14}\) show the E2E systems not only extremely simplify the speech recognition pipeline, but also surpass the conventional hybrid ASR systems in accuracy.

Considering the advantages\(^{15}\) of E2E models, deploying the emerging E2E ASR framework into real-world productions with stable and highly efficient resources becomes necessary. However, almost all of the well-known E2E speech recognition toolkits, such as ESPNet\(^{16}\) and SpeechBrain\(^{17}\), are research-oriented rather than production-oriented.

In\(^1\), we present a production first and production ready E2E speech recognition toolkit, WeNet\(^1\), which focuses on addressing the production problems of transformer\(^{18}\) and conformer\(^{19}\) based E2E models. Specifically, WeNet adopts a joint CTC/AED structure as the basic model structure. And then, a unified two-pass (U2) framework is proposed to solve the streaming problem, where a dynamic chunk masking strategy is applied in the training procedure to unify the streaming and non-streaming modes in a unified neural model. A built-in runtime is provided in which developers can run x86 and android E2E system for speech recognition out of the box. WeNet dramatically reduces the workload of deploying an E2E model in real applications, so it is widely adopted by researchers and developers.

In this paper, we present **WeNet 2.0**, which introduces the recent developments and solutions updated in WeNet for production-oriented speech recognition. The key updates of WeNet 2.0 are as follows.

1. **U2++**: we upgrade the original U2 framework to U2++, which simultaneously utilizes the left-to-right and right-to-left bidirectional contextual information of the labeled sequences to learn richer context information during the training stage, and combines the forward and backward predictions to achieve more accurate results during the decoding stage. The experiments show that U2++ achieves up to 10% relative reduction over the original U2 framework in error rate.

2. **Production language model solution**: we support an optional n-gram LM, which is composed with the E2E model in a weighted finite state transducer (WFSST)\(^{20}\) based decoding graph, during the streaming CTC decoding stage. The n-gram LM can be trained rapidly on rich production-accumulated text data. Experiments show that the n-gram language model can provide up to 8% relative performance improvement.

3. **Contextual biasing**: we design a unified contextual biasing framework which provides the opportunity to leverage the user-specific contextual information with or without LM during the streaming decoding stage. Utilizing user-specific contextual information (e.g., contact lists, particular dialog state, conversation topic, location, etc) plays a great role in both improving the ASR accuracy and providing rapid adaptation ability. Experiments show that our contextual biasing solution can bring clear dramatic improvement for both with-LM and without-LM cases.

4. **Unified IO (UIO)**: we design a unified IO system to support the large-scale dataset for effective model training. The UIO system provides a unified interface for different storage media (i.e., local disk or cloud) and datasets for different scale datasets. For small datasets, UIO keeps the
sample-level random access ability. While, for large datasets, UIO aggregates the data samples to shards (refer to Figure 4 for more details) and provides the shard-level random access ability. Thanks to UIO, WeNet 2.0 can elastically support training with a few hours to millions of hours of data.

In summary, our brand-new WeNet 2.0 makes available several important production-oriented features, which achieves substantial ASR improvement over the original WeNet and makes itself a more productive E2E speech recognition toolkit.

2. WeNet 2.0

In this section, we will describe the production-oriented key updates: U2++ framework, production language model solution, contextual biasing, and unified IO in detail, respectively.

2.1. U2++

U2++, a unified two-pass joint CTC/AED framework with bidirectional attention decoders, provides an ideal solution to unify streaming and non-streaming modes. As shown in Figure 1, U2++ consists of four parts. 1) A Shared Encoder that models the information of acoustic features. The Shared Encoder consists of multiple Transformer [18] or Conformer [19] layers which only takes limited right contexts into account to keep a balanced latency. 2) A CTC Decoder that models the frame-level alignment information between acoustic features and token units. The CTC Decoder consists of a linear layer, which transforms the Shared Encoder output to the CTC activation. 3) A Left-to-Right Attention Decoder (L2R) that models the ordered token sequence from left to right to represent the past contextual information. 4) A Right-to-Left Attention Decoder (R2L) that models the reversed token sequence from right to left to represent the future contextual information. The L2R and R2L Attention Decoders consist of multiple Transformer decoder layers. During the decoding stage, the CTC Decoder runs in the streaming mode in the first pass and the L2R and R2L Attention Decoders do rescoring in the non-streaming mode to improve the performance in the second pass.

![Figure 1: U2++ framework. The dotted line denotes the second pass rescoring during the decoding stage.](image)

Compared with U2 [1], we add an extra right-to-left attention decoder to enhance the context modeling ability of our model, so that the context information comes not only from the past (left-to-right decoder) but also from the future (right-to-left decoder). This improves the representative ability of the shared encoder, the generalization ability of the whole system, and the performance during the rescoring stage.

The combined CTC and AED loss is used to train U2++:

\[ L_{\text{combined}}(x, y) = \lambda L_{\text{CTC}}(x, y) + (1 - \lambda)(L_{\text{AED}}(x, y)) \]  

(1)

where \( x \) denotes the acoustic features, and \( y \) denotes the corresponding labels, \( \lambda \) is a hyper parameter which balances the importance of the CTC loss \( L_{\text{CTC}}(x, y) \) and AED loss \( L_{\text{AED}}(x, y) \).

To incorporate an extra R2L attention decoder into our model, we introduce a hyper parameter \( \alpha \) inside \( L_{\text{AED}}(x, y) \) to adjust the contributions of two unidirectional decoders:

\[ L_{\text{AED}}(x, y) = (1 - \alpha)L_{\text{R2L}}(x, y) + \alpha L_{\text{L2R}}(x, y). \]  

(2)

Similar to U2, the dynamic chunk masking strategy is used to unify the streaming and non-streaming modes. During the training stage, we firstly sample a random chunk size \( C \) from a uniform distribution, \( C \sim U(1, \text{max batch length} \cdot T) \). And then, the input is split into several chunks with the chosen chunk size. At last, the current chunk does bidirectional chunk-level attention to itself and previous/following chunks in L2R/R2L attention decoder respectively in training. During the decoding stage, the n-best results achieved from the first-pass CTC decoder are rescored by the L2R and R2L attention decoder with its corresponding acoustic information generated by the Shared Encoder. The final results are obtained by fusing the scores of the two attention decoders and the CTC decoder.

Empirically, a larger chunk size results in better results with higher latency. Thanks to the dynamic strategy, U2++ learns to predict with arbitrary chunk size so that balancing the accuracy and latency is simplified by tuning the chunk size in decoding.

2.2. Language Model

To promote the use of rich text data in production scenarios, we provide a unified LM integration framework in WeNet 2.0, which is shown in Figure 2.

As a unified LM and LM-free system, CTC is employed to generate the first-pass n-best results. When LM is not provided, CTC prefix beam search is applied to get n-best candidates. When LM is supplied, WeNet 2.0 compiles the n-gram LM \( G \), lexicon \( L \), and end-to-end modeling CTC topology \( T \) into a WFST-based decoding graph \( TLG \)

\[ TLG = T \circ \min(\det(L \circ G)), \]  

(3)

and then CTC WFST beam search is applied to get n-best candidates. Finally, the n-best candidates are rescoped by the Attention Rescoring module to find the best candidate.

We reuse the algorithm and code in Kaldi[21] for decoding, which is denoted as CTC WFST beam search in the WeNet 2.0 framework. To speedup decoding, blank frame skipping [22] technique is adopted.

2.3. Contextual Biasing

Utilizing user-specific contextual information (e.g., contact lists, driver’s navigation) plays a vital role in speech production, which always improves the accuracy significantly and provides the rapid adaptation ability. Contextual biasing technique is investigated in [23, 24, 25] for both traditional and E2E systems.

In WeNet 2.0, for leveraging the user-specific contextual information in both LM and LM-free cases during the streaming decoding stage, inspired by [24], we construct a contextual WFST graph on the fly when a set of biasing phrases are known in advance. Firstly, the biasing phrases are split into biasing units according to the E2E modeling units in the LM-free case or the vocabulary words in the with-LM case. And then, the contextual WFST graph is built on the fly as follows: (1) Each biasing unit with a boosted score is placed on a corresponding intermediate state of the acceptable chain. (2) For each
with a negative accumulated boosted score is added. The failure arcs are used to remove the boosted scores when only partial biasing units are matched rather than the entire phrase. In Figure 3, a char (E2E modeling unit) level context graph in the LM-free case and a word-level context graph in the with-LM case are shown, respectively. Finally, during the streaming decoding stage, a boosted score is added immediately when the beam search result matches with a biasing unit through the contextual WFST graph, as shown in

$$y^* = \arg \max \log P(y|x) + \lambda \log P_C(y),$$

where $$P_C(y)$$ is the biasing score, $$\lambda$$ is a tunable hyper-parameter to control how much the contextual LM influences the overall model score. Especially, when some biasing phrases share the same prefix, we do a greedy match to simplify the implementation.

(a) Char level context graph without LM

(b) Word level context graph with LM

Figure 3: Context graph for Chinese phrase “不忘初心”. The failure arcs are highlighted in red lines.

2.4. UIO

Typically, the production-scale speech dataset is over tens of thousands of hours labeled speech, which always comprises massive small size files. The massive small files will cause the following problems:

• Out of memory (OOM): we have to keep the index information for all of the small files, which is very memory-consuming for large datasets.

• Lower training speed: reading data becomes the bottleneck of the training due to the intolerable time-consuming during the training to access massive small files.

To address the aforementioned problems for large-scale production datasets and keep the high efficiency for small datasets at the same time, we design a unified IO system, which provides a unified interface for different storage (i.e., local disk or cloud) and different scale datasets (i.e., small or large datasets). The whole framework is shown in Figure 4.

For large datasets, inspired by TFRecord (Tensorflow) [26] and AllStore [27], we pack each set of samples (e.g., 1000 samples) with their associated metadata information into a corresponding bigger shard, which is done by GNU tar, an open-source, open-format, ubiquitous and widely-supported archival tool. Tar files significantly save memory and overcome the OOM problem effectively. During the training stage, on-the-fly decompression is performed in the memory, and the data in the same compressed shard is read sequentially, which solves the time-consuming random data access problem and speeds up the training process. At the same time, different shards can be read randomly to ensure the global randomness of data. For small datasets, we can also load the training samples directly.

Especially, when using shards for big datasets, we support loading the shards from both the local disk and the distributed storage (e.g., S3, OSS, HDFS, etc). Similar to TFRecord, chain operations for processing data on-the-fly is designed, so that the unified IO is extensible and easy to debug.

3. Experiments

In this section we describe our experimental setup, test sets, and analyze the experimental results. Most of the experimental setups are available in the WeNet recipes.

The experiments are carried on all of or some of the following corpora: AISHELL-1 [28], AISHELL-2 [29], LibriSpeech [30], GigaSpeech [31], and recent released Wenet-Speech [32]. The five corpora include different languages (English and Mandarin), recording environments (clean and noisy), and sizes (100 - 10000 hours).

3.1. U2++

To evaluate the effectiveness of our U2++ model, we conduct experiments on all of the 5 ASR corpora listed above. For most experiments, 80-dimensional log Mel-filter banks (FBANK) with a 25ms window and a 10ms shift are used as acoustic features. SpecAugment [33] is applied on-the-fly for data augmentation. Two convolution subsampling layers with kernel size 3*3 and stride 2 are used in the front of the encoder. We use 12 conformer layers for the encoder. To keep comparable parameters for U2/U2++, we use 6 transformer decoder layers for

\[https://wenet.org.cn/wenet/UIO.html\]
Table 1: U2++/U2 comparison on various open-source ASR corpora. * denotes the model is trained with dynamic mask, and it could be decoded with different chunk size (full/16 in the table). † denotes the model is trained with full attention.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Language</th>
<th>Hours</th>
<th>Unit</th>
<th>Metric</th>
<th>Test Sets</th>
<th>U2</th>
<th>U2++</th>
</tr>
</thead>
<tbody>
<tr>
<td>AISHELL-1 *</td>
<td>Mandarin</td>
<td>170</td>
<td>Char</td>
<td>CER</td>
<td>test(full)/test(16)</td>
<td>4.97/5.45</td>
<td>4.63/5.05</td>
</tr>
<tr>
<td>AISHELL-2 *</td>
<td>Mandarin</td>
<td>1000</td>
<td>Char</td>
<td>CER</td>
<td>test ios(full)/test ios(16)</td>
<td>6.08/6.46</td>
<td>5.39/5.78</td>
</tr>
<tr>
<td>LibriSpeech †</td>
<td>English</td>
<td>960</td>
<td>BPE</td>
<td>WER</td>
<td>test clean/test other</td>
<td>2.85/7.24</td>
<td>2.66/6.53</td>
</tr>
<tr>
<td>WenetSpeech †</td>
<td>Mandarin</td>
<td>10000</td>
<td>Char</td>
<td>MER</td>
<td>Test Net/Test Meeting</td>
<td>9.70/15.59</td>
<td>9.25/16.18</td>
</tr>
<tr>
<td>GigaSpeech †</td>
<td>English</td>
<td>10000</td>
<td>BPE</td>
<td>WER</td>
<td>dev/test</td>
<td>11.30/11.20</td>
<td>10.70/10.60</td>
</tr>
</tbody>
</table>

U2, 3 left-to-right and 3 right-to-left decoder layers for U2++. Moreover, we obtain our final model by model averaging. In AISHELL-1 and AISHELL-2, the attention layer uses attention dimension 256, feed-forward 2048, and 4-head attention. In LibriSpeech, GigaSpeech and WenetSpeech, the attention layer uses attention dimension 512, feed-forward 2048, and 8-head attention. The kernel size of the convolution module is 8/8/31/31/15 for the five corpora, respectively. Accumulating gradient is used to stabilize training.

In Table 1, we report the character error rate (CER), word error rate (WER) or mixed error rate (MER) for each corpus. It shows that U2++ outperforms U2 on most of the corpora, and achieves up to 10% relative improvement over U2. From the results, we approve that U2++ shows superior performance in various types and sizes of ASR corpora consistently.

3.2. N-gram Language Model

The language model solution is evaluated on AISHELL-1, AISHELL-2, and LibriSpeech on the models listed in Section 3.1. For AISHELL-1 and AISHELL-2, a tri-gram trained on its own training corpus is used. For LibriSpeech, the official pre-trained four-gram large language model (fglarge) is used. To speed up the decoding, when the probability of the “blank” symbol of a frame is greater than 0.98, the frame will be skipped.

Table 2: N-gram LM evaluation

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Test Sets</th>
<th>without LM</th>
<th>with LM</th>
</tr>
</thead>
<tbody>
<tr>
<td>AISHELL-1</td>
<td>test</td>
<td>4.63</td>
<td>4.40</td>
</tr>
<tr>
<td>AISHELL-2</td>
<td>test_ios</td>
<td>5.39</td>
<td>5.35</td>
</tr>
<tr>
<td>LibriSpeech</td>
<td>test_clean/test_other</td>
<td>2.66/6.53</td>
<td>2.65/5.98</td>
</tr>
</tbody>
</table>

Table 2 exhibits the comparison results on the three corpora in with-LM and without-LM scenarios. The with-LM solution shares 5% gain on AISHELL-1, comparable result on AISHELL-2, comparable result in test_clean on LibriSpeech, and superior improvement (8.42%) in test_other on LibriSpeech.

In conclusion, our production LM solution shows consistent gain on the three corpora, which illustrates n-gram language model works well when integrated to an E2E speech recognition system. Benefitting from this solution, the production system can enjoy the rich production-accumulated text data.

3.3. Contextual Biasing

We evaluate contextual biasing in a contact scenario, and we design two test sets selected from AISHELL-2 test sets.

- **test_p**: positive test set with its relevant context. We select 107 utterances with person names, and all the names are attached as contextual biasing phrases in decoding.
- **test_n**: negative test set without any contextual biasing phrases (i.e., names) in the positive testset. We randomly select 107 eligible utterances as the negative test set.

We carry on the test on the AISHELL-1 U2++ model with and without LM. We show that the strength of contextual biasing can be controlled by switching the boosted score. In our experiments, the boosted score is varied from 0 to 10, where 0 means context biasing is not applied. As shown in Table 3, our solution reduces the error rate on the positive test set and a bigger boosted score usually gives better improvement. Furthermore, when a proper boosted score is set, we can improve the performance on the positive test set and avoid the performance degradation on negative test set at the same time.

Table 3: Context Biasing evaluation

<table>
<thead>
<tr>
<th>Test Sets</th>
<th>boosted score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>NO test_p</td>
<td>14.94</td>
</tr>
<tr>
<td>NO test_n</td>
<td>7.45</td>
</tr>
<tr>
<td>YES test_p</td>
<td>13.95</td>
</tr>
<tr>
<td>YES test_n</td>
<td>7.20</td>
</tr>
</tbody>
</table>

3.4. UIO

We evaluate the performance of UIO on AISHELL-1 [28] testset and WenetSpeech [32] testsets (i.e., test_net/test_meeting), including the accuracy, scalability and training speed. Specifically, we use 8 GPUs on one machine for AISHELL-1, and 24 GPUs on three machines for WenetSpeech.

As shown in Table 4, the UIO shows close accuracy in raw and shard modes on AISHELL-1, while the shard mode gets about 9.27% speedup in terms of the training speed. For WenetSpeech, since it is too slow to only use raw mode for training, the training speed is not shown. We adopt the result of ESPnet (marked with ‡) as our baseline, which has the same configuration as our model. The result of the model trained on WenetSpeech by shard mode is comparable to ESPnet, which further illustrates the effectiveness of UIO.

Table 4: Unified IO evaluation

<table>
<thead>
<tr>
<th>Dataset</th>
<th>metric</th>
<th>raw</th>
<th>shard</th>
</tr>
</thead>
<tbody>
<tr>
<td>AISHELL-1</td>
<td>speed(per epoch)</td>
<td>593 sec</td>
<td>538 sec</td>
</tr>
<tr>
<td></td>
<td>WER</td>
<td>4.63</td>
<td>4.67</td>
</tr>
<tr>
<td>WenetSpeech</td>
<td>speed(per epoch)</td>
<td>NA</td>
<td>75 min</td>
</tr>
<tr>
<td></td>
<td>WER</td>
<td>8.90/15.90(‡)</td>
<td>9.70/15.59</td>
</tr>
</tbody>
</table>

4. Conclusion and Future Work

In this paper, we proposed our more productive E2E speech recognition toolkit, WeNet 2.0, which introduces several important production-oriented features and achieves substantial ASR performance improvement.

We are working on WeNet 3.0, which mainly focuses on unsupervised self-learning, on-device model exploration and optimization, and other characteristics for production-level ASR.
References


