Progressive Down-Sampling for Acoustic Encoding

Anonymous ACL submission

Abstract

In acoustic encoding, the fine-grained frame-level features are not suited for capturing global dependencies. But condensing them into a semantically complete representation by stacked down-sampling does not work well. We find that the condensation leads to the degraded correlation of the representations in adjacent positions, which poses the risk of information loss in the stacked method. In this work, we propose a new method, progressive down-sampling (PDS), for encoding the context sufficiently before each condensation. Also, we develop a representation fusion method to alleviate information loss by combining the multi-scale representations. Experimental results on the 960h LibriSpeech automatic speech recognition task show that, for a strong Conformer-based system, our method down-samples the input speech features to 1/32 of the initial length, while yielding an improvement of 0.47 WER with a speedup of 1.42×. It also achieves the state-of-the-art BLEU score (25.8) on the MuST-C En-De speech translation benchmark with no additional training data.

1 Introduction

Despite the success in speech processing tasks like automatic speech recognition (ASR) (Lu et al., 2020; Zhang et al., 2021) and speech translation (ST) (Xu et al., 2021), how to encode the speech features effectively is an open problem. Different from modeling based on discrete units in natural language processing, a standard paradigm for acoustic encoding is taking as input the continuous frame-level features with a very short shift.

Framing generates a very long sequence consisting of fine-grained features. For example, a framing-based feature sequence is in general tens of times longer than the sub-word sequence in a transcription (see Figure 1). For encoding, such a problem leads to the difficulties of capturing long-distance dependencies and distributing the attention weights across semantically incomplete modeling units (Han et al., 2019). Previous work (Al-Rfou et al., 2019) also demonstrates that the fine-grained character-level models yield significantly inferior performance compared with word-level counterparts. In addition, the long sequence also results in prohibitive computation costs due to the quadratic complexity of self-attention.

A popular method is down-sampling (DS) fine-grained feature to form a more meaningful representation by stacking multiple strided convolution layers before encoding (Dong et al., 2018; Berard et al., 2018). Unfortunately, it does not work well when the down-sampling ratio increases. Intuitively, it is difficult to condense dozens of frames into one unit straightly (Sayood, 2018). It is something like that a few principal components cannot preserve all the information in the classical principal component analysis method (Wold et al., 1987).

For shedding light on the reason of failure, we analyze the condensation process. We find that the correlation of representations in adjacent positions degrades due to down-sampling, which increases the difficulty for subsequent condensation. This leads to the non-trivial issue of information loss in the stacked method.

To address this issue, we propose a Progressive Down-Sampling (PDS) method. For the input speech features, a single layer of down-sampling is employed to aggregate the consecutive representa-

Figure 1: The distribution of the length ratio between the speech features (frame-level) and corresponding transcriptions (sub-word level).
tions into more informative units. Then the model encodes the context for high correlation of the representation over the sequence. Repeating the above process, we aggregate the frame-level features into more semantically complete units in a progressive manner.

In this way, the multi-scale representations of different granularities are obtained. The fine-grained representations, on the other hand, may contain information that is lost during condensation. To further address the problem, we align the multi-scale representations to the same shape, and then combine them by a lightweight representation fusion method.

PDS is a general method for acoustic encoding. It is easy to make a trade-off between computational speedup and performance. We evaluate it on ASR and End-to-End ST tasks. Experiments on LibriSpeech ASR show that our method achieves a high down-sampling ratio up to 32. Also, it is beneficial to both system speedup and performance improvement. It outperforms the stacked counterparts by 0.47 WER with a speedup of 1.42×. On a more challenging task of ST, our method helps model convergence and achieves the state-of-the-art BLEU score of 25.8 on the MuST-C En-De benchmark without additional resources.

2 Related Work

Unlike text that has explicit boundaries, audio is in general represented in continuous signals. Although researchers have explored models based on the raw audio signal (Schneider et al., 2019), the popular method for segmentation is framing with a frame size of 25ms and a frame shift of 10ms (Oppenheim, 1999). The short frame shift allows the continuity of the speech signal, and the overlapping segments help to avoid the information loss between consecutive frames.

However, the fine-grained frame-level features may not be suitable for the state-of-the-art architectures (Vaswani et al., 2017). The long sequences composed of semantically incomplete units lead to the difficulties of capturing long-distance dependencies and distributing the attention weights across the most related positions. Researchers (Salesky et al., 2019; Salesky and Black, 2020) investigate phoneme-level methods. For example, one can average frame-level features within phoneme-like units. But this needs a non-trivial recognizer for phoneme alignment.

Motivated by the work in efficient models, researchers alleviate the modeling difficulty by the improved self-attention mechanisms (Han et al., 2019; Alastruey et al., 2021; Papi et al., 2021). However, they ignore the inherent problem of fine-grained modeling and the cross-attention module still suffers from the same issue.

A natural idea is to down-sample the fine-grained features to generate a more meaningful representation (Chan et al., 2015; Bahdanau et al., 2016). To do this, a popular method is to pass the features through a stack of strided convolutional layers before encoding (Dong et al., 2018; Berard et al., 2018). But the stacked method does not work well in practice due to the loss of information in consecutive convolutional operations. As a way to address this, several research groups use the progressive method to down-sample the acoustic sequence (Peddinti et al., 2018; Huang et al., 2020; Han et al., 2020; Burchi and Vielzeuf, 2021). However, there is still no in-depth analysis on this problem.

Another open problem for acoustic encoding is the variable information caused by silence or noise. Researchers develop adaptive selection (Zhang et al., 2020a) or dynamic down-sampling methods (Na et al., 2019; Zhang et al., 2019) for avoiding useless features. However, the granularity of the filtered representation is still far from ideal. Here we explicitly discuss the problem and focus on effective down-sampling with a fixed ratio.

3 The Method

3.1 Why Is Information Lost?

Down-sampling generates more semantically complete units by aggregating the adjacent frame-
level features. Following previous work in down-sampling (Dong et al., 2018; Berard et al., 2018), speech features are fed into a stack of 2 convolutions with a stride of 2. The convolution layers are followed by a number of encoder layers (see Figure 3 (a)). For a large down-sampling ratio, it is natural to stack more down-sampling layers.

We study the changes of representation during down-sampling. We define the correlation of representation as the average cosine similarity of each unit to the surrounding units within a small window. High correlation means that the representations of the adjacent positions are similar.

We train a Transformer-based (Vaswani et al., 2017) ASR model with 4 stacked down-sampling on the 960h LibriSpeech dataset and show the correlation of the test-clean test set. As shown in Figure 2 (Left), the input speech features have an extremely high correlation due to the overlapping framing. However, the correlation degrades sharply after each down-sampling. The subsequent down-sampling processes are difficult to condense the diverse representation while preserving the information completely. We call this issue information loss caused by stacked down-sampling.

Now a new question arises: how to increase the correlation of the representation and alleviate the information loss in down-sampling? An intuitive conjecture is that the context modeling increases the correlation due to the strong preference of the short-distance dependency (Sperber et al., 2018; Xu et al., 2021). Figure 2 (Right) shows the correlation in each layer of the encoder of the standard Transformer with a down-sampling ratio of 4. Obviously, the correlation increases from bottom to top, as we expected. This motivates us to develop a progressive method for encoding context information sufficiently after each down-sampling.

3.2 Progressive Down-Sampling

We propose a Progressive Down-Sampling (PDS) method to condense the fine-grained features into the semantically complete units. See Figure 3 (b) for an overview of PDS. The encoding is divided to two processes: a representation down-sampling process and a context interaction process.

For the input speech features like MFCC or Mel filter bank, an overlapping down-sampling condenses it by a simple convolution 1D module. Like framing, the overlapping in convolution alleviates the information loss. It also enforces the model to capture local modeling. To deal with varied sequence lengths, the position encoding is introduced into the representation after layer normalization.

Inspired by the finding presented in Section 3.1, the down-sampled representation requires sufficient context interaction for high correlation. Here we simply use the multiple identity layers to capture the dependencies.

Each run of down-sampling and encoding is called a stage. The model runs for $M$ stages and obtains more meaningful representations $\{H_1, H_2, \cdots, H_M\}$.

A merit of PDS is that it offers a trade-off between computational efficiency and performance. One can stack more stages for extreme down-sampling. This decreases the computational cost significantly but may lead to the performance drop due to the inevitable information loss. On the other hand, fewer down-sampling processes preserve the information for better performance but cannot provide sufficient speedups. Note that the stacked method can be seen as a specific case of PDS: it consists of two stages and the number of layers in the first stage is zero.

PDS is also similar to the typical backbones in the field of computer vision (CV), like CNN (He, 2020).
et al., 2016) and Transformer (Wang et al., 2021). Both of them employ the same design concept, i.e., aggregating the fine-grained input into the more informative representation by progressive down-sampling. This paradigm is widely used in CV tasks. Here we explore it in the field of speech processing.

3.3 Representation Fusion

As the nature of down-sampling, the information loss still occurs inevitably although we use the progressive method. Motivated by previous methods to make full use of the multi-level representations (Wang et al., 2018, 2019), a way to further address the problem is to fuse the finer-grained representations (Zhao et al., 2017; Zhang et al., 2020b). Then the final output representation $H^o$ can be defined as:

$$H^o = \mathcal{F}(H_1, \ldots, H_M)$$

(1)

where $\mathcal{F}(\cdot)$ is the fusion function\(^1\). But this raises a new question: how to combine the multi-scale representations effectively?

The first step is to align the different scales to the same one. We resort to a simple but effective non-overlapping convolution operation to transform the finer-grained representations outputted in bottom stages to the shape of $H_M$. The stride for the representation $H_k$ is set to the multiplication of the subsequent down-sampling ratios\(^2\).

Drawing on the design of the convolution module in Conformer, the representation fusion method with alignment is shown in Figure 4. We use a simple linear combination of the representations that are already in the same shape. The output $\mathcal{F}(\cdot)$ is defined as:

$$\mathcal{F}(H_1, \ldots, H_M) = \sum_{k=1}^{M} W_k \text{LN}(A(H_k))$$

(2)

where $A(\cdot)$ is the alignment function and LN(\cdot) is the layer normalization function. $W_k \in \mathbb{R}$ is a learnable scalar to weight the aligned representations. The weights are initialized to the same values, and are updated as other parameters during training.

3.4 PDS Settings

In this work, we choose 8 settings with different depths and down-sampling ratios (see Table 1). We basically follow the design in He et al. (2016) but do adaptations for acoustic encoding:

- In down-sampling, a bigger window size means that more context information is involved. But this also increases the difficulty for down-sampling due to the lower correlation. We use an empirical setting of kernel size = 5.
- We do not increase hidden dimensions as the growth of the model depth, i.e., we use the same hidden dimensions for all stages. Modeling with a small dimension in the bottom stage is obviously not a good idea due to the high dimension of the initial features (typically 80-dimension).
- The bottom stages have fewer layers for efficient computation due to the longer sequence. The major computations are concentrated on the intermediate stages. This leads to computation acceleration and sufficient encoding.

Table 1: Settings of PDS. "Stack-4" represents the standard method. "PDS-Base-R" and "PDS-Deep-R" denote an encoder of 12 layers and 30 layers with a down-sampling ratio of $R$ respectively. "Stride" and "Layer" separated by "," represent the stride of the down-sampling module and the number of layers in each stage from bottom to top.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Stride</th>
<th>Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stack-4</td>
<td>2-2</td>
<td>0-12</td>
</tr>
<tr>
<td>PDS-Base-8</td>
<td>2-2-1-2</td>
<td>3-3-3-3</td>
</tr>
<tr>
<td>PDS-Base-16</td>
<td>2-2-2-2</td>
<td>2-2-6-2</td>
</tr>
<tr>
<td>PDS-Base-32</td>
<td>2-2-2-2-2</td>
<td>2-2-3-3-2</td>
</tr>
<tr>
<td>Stack-4</td>
<td>2-2</td>
<td>0-30</td>
</tr>
<tr>
<td>PDS-Deep-8</td>
<td>2-2-1-2</td>
<td>7-7-7-9</td>
</tr>
<tr>
<td>PDS-Deep-16</td>
<td>2-2-2-2</td>
<td>5-5-12-8</td>
</tr>
<tr>
<td>PDS-Deep-32</td>
<td>2-2-2-2-2</td>
<td>5-5-7-7-6</td>
</tr>
</tbody>
</table>

\(^1\)We drop the input feature because it is extracted by signal processing rather than the encoding model.

\(^2\)The stride for $H_M$ is set to 1.
4 Experiments
We evaluate PDS on ASR and ST tasks.

4.1 Datasets and Preprocessing
The datasets are from two benchmarks:

- LibriSpeech is a publicly available read English ASR corpus, which consists of 960-hour of training data (Panayotov et al., 2015). The development and test data are divided into clean and other subsets according to the speech quality. We select the model on the dev-clean set and report results on all four subsets.

- MuST-C En-De is a multilingual speech translation corpus extracted from the TED talks (Gangi et al., 2019). We train our systems on the English-German speech translation dataset of 400-hour speech. We select (and tune) the model on the dev set and report results on the tst-COMMON set.

For preprocessing, we follow the common recipes in fairseq toolkit\(^3\), we remove the utterances of more than 3,000 frames or fewer than 5 frames. Speed perturbation is not used in all models. The 80-channel Mel filter bank features are extracted by a 25ms window with a stride of 10ms. We learn SentencePiece\(^4\) segmentation with a size of 10,000 for the two datasets. For ST, we use a shared vocabulary for source and target languages.

4.2 Model Settings
We use the encoder-decoder framework and implement the method based on the fairseq toolkit. We use the Adam optimizer and adopt the default learning schedule in fairseq. We apply dropout with a rate of 0.1 and label smoothing \(\epsilon_{ls} = 0.1\) for regularization. SpecAugment (Park et al., 2019) is applied in the input speech features for better generalization and robustness.

For ASR tasks, we evaluate our method on Transformer (Vaswani et al., 2017) and Conformer (Gulati et al., 2020). The settings of the encoder for ASR models are shown in Table 2. The decoder consists of 6 Transformer layers and the settings are same to the encoder. CTC (Graves et al., 2006) multitask learning is not used due to the very modest improvement in our preliminary experiments.

For ST tasks, we evaluate our method on Transformer and SATE (Xu et al., 2021). Knowledge distillation is not used for simplicity. The encoder consists of 12 layers for Transformer. SATE has an acoustic encoder of 12 layers and a textual encoder of 6 layers. Each layer comprises 256 hidden units, 4 attention heads, and 2,048 feed-forward hidden units. We use pre-training for more challenging ST task, where the ASR and MT models are pre-trained with the MuST-C En-De data. Different from the ASR model, CTC is employed with a weight of 0.3 for better convergence.

All the models are trained for 100 epochs. We early stop training when there is no performance improvement on the development set for 10 consecutive checkpoints. We use beam search decoding with a beam size of 5 for all models. The CTC and language model rescoring methods are not used. WER and case-sensitive SacreBLEU are reported for ASR and ST respectively.

4.3 Results of ASR
Table 2 shows the results on the 960h LibriSpeech corpus. We compare the methods on Transformer and Conformer with different encoder layers and hidden dimensions. We use Stack-4 as the baseline model (see Table 1 for the setting). The amount of model parameters in each group is similar for a fair comparison.

For the popular setting of 12 encoder layers with 256 hidden dimensions in rows (A), PDS is the first to achieve a very high down-sampling ratio of 32 with no performance drop. It yields a speedup of 1.20 ×. As the down-sampling ratio decreases, performance improves significantly. Similar phenomena are observed on the wider Transformer with 512 hidden dimensions in rows (B), and this bigger model benefits more in speedup.

Interestingly, we find that the deep models with 30 encoder layers in rows (C) eliminate the performance gap when different down-sampling ratios are employed. PDS condenses the representation to 1/32 of the initial length, while achieving a considerable relative improvement of 0.75 WER points. We conjecture that the deep model allows more sufficient modeling in each stage and preserves the information even in an extreme case of downsampling. This has a practical advantage in industrial scenarios wherein more speedups are required.

For Conformer, the behavior of the base model with 12 encoder layers in rows (D) is similar to

\(^3\)https://github.com/pytorch/fairseq
\(^4\)https://github.com/google/sentencepiece
Table 2: WER on 960h LibriSpeech ASR corpus. L: the number of encoder layers. \(d_h\): the hidden dimension. \(d_{ff}\): the feed-forward dimension. \(h\): the number of the attention heads. #Params: the number of parameters. The speedup is computed during inference on test-clean set with the batch size of 100k and beam size of 5.

<table>
<thead>
<tr>
<th>Setting</th>
<th>L.</th>
<th>(d_h)</th>
<th>(d_{ff})</th>
<th>h</th>
<th>#Params</th>
<th>dev clean</th>
<th>other clean</th>
<th>test clean</th>
<th>other clean</th>
<th>Avg.</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Transformer</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(A) Stack-4</td>
<td>12</td>
<td>256</td>
<td>2048</td>
<td>4</td>
<td>30M</td>
<td>3.88</td>
<td>9.26</td>
<td>4.49</td>
<td>9.42</td>
<td>6.76</td>
<td>1.00×</td>
</tr>
<tr>
<td>PDS-Base-8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30M</td>
<td>3.57</td>
<td>8.63</td>
<td>3.85</td>
<td>8.58</td>
<td>6.16 (-0.60)</td>
<td>0.99×</td>
</tr>
<tr>
<td>PDS-Base-16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30M</td>
<td>3.71</td>
<td>8.73</td>
<td>3.74</td>
<td>9.02</td>
<td>6.30 (-0.46)</td>
<td>1.14×</td>
</tr>
<tr>
<td>PDS-Base-32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>31M</td>
<td>4.13</td>
<td>9.31</td>
<td>4.21</td>
<td>9.31</td>
<td>6.74 (-0.02)</td>
<td>1.20×</td>
</tr>
<tr>
<td>(B) Stack-4</td>
<td>12</td>
<td>512</td>
<td>2048</td>
<td>8</td>
<td>71M</td>
<td>3.53</td>
<td>8.15</td>
<td>3.67</td>
<td>7.96</td>
<td>5.83</td>
<td>1.00×</td>
</tr>
<tr>
<td>PDS-Base-8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>75M</td>
<td>3.17</td>
<td>7.46</td>
<td>3.47</td>
<td>7.47</td>
<td>5.39 (-0.44)</td>
<td>1.08×</td>
</tr>
<tr>
<td>PDS-Base-16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>76M</td>
<td>3.34</td>
<td>7.73</td>
<td>3.57</td>
<td>7.85</td>
<td>5.57 (-0.26)</td>
<td>1.34×</td>
</tr>
<tr>
<td>PDS-Base-32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>82M</td>
<td>3.32</td>
<td>7.94</td>
<td>3.64</td>
<td>7.85</td>
<td>5.69 (-0.14)</td>
<td>1.47×</td>
</tr>
<tr>
<td>(C) Stack-4</td>
<td>30</td>
<td>256</td>
<td>2048</td>
<td>4</td>
<td>53M</td>
<td>3.80</td>
<td>8.51</td>
<td>4.33</td>
<td>8.61</td>
<td>6.31</td>
<td>1.00×</td>
</tr>
<tr>
<td>PDS-Base-8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>53M</td>
<td>3.34</td>
<td>7.90</td>
<td>3.50</td>
<td>7.79</td>
<td>5.63 (-0.68)</td>
<td>1.03×</td>
</tr>
<tr>
<td>PDS-Base-16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>54M</td>
<td>3.15</td>
<td>7.83</td>
<td>3.38</td>
<td>7.79</td>
<td>5.53 (-0.78)</td>
<td>1.19×</td>
</tr>
<tr>
<td>PDS-Base-32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>55M</td>
<td>3.26</td>
<td>7.77</td>
<td>3.33</td>
<td>7.88</td>
<td>5.56 (-0.75)</td>
<td>1.27×</td>
</tr>
</tbody>
</table>

| Conformer    |    |         |            |    |         |           |            |           |            |       |         |
| -------------|----|---------|------------|    |         |           |            |           |            |       |         |
| (D) Stack-4  | 12 | 256     | 2048       | 4  | 45M     | 3.64      | 8.17       | 3.74       | 8.10       | 5.91  | 1.00×   |
| PDS-Base-8   |    |         |            |    | 46M     | 3.10      | 7.41       | 3.23       | 7.59       | 5.33 (-0.58) | 0.98×   |
| PDS-Base-16  |    |         |            |    | 46M     | 3.06      | 7.27       | 3.12       | 7.58       | 5.26 (-0.65) | 1.16×   |
| PDS-Base-32  |    |         |            |    | 47M     | 2.99      | 7.57       | 3.17       | 7.76       | 5.37 (-0.54) | 1.20×   |
| (E) Stack-4  | 12 | 512     | 2048       | 8  | 109M    | 3.67      | 7.66       | 3.79       | 7.60       | 5.56  | 1.00×   |
| PDS-Base-8   |    |         |            |    | 113M    | 2.90      | 6.71       | 3.10       | 6.70       | 4.85 (-0.71) | 1.00×   |
| PDS-Base-16  |    |         |            |    | 114M    | 3.11      | 6.74       | 3.57       | 6.90       | 5.03 (-0.53) | 1.33×   |
| PDS-Base-32  |    |         |            |    | 119M    | 2.96      | 7.10       | 3.18       | 7.11       | 5.09 (-0.47) | 1.42×   |

Table 3 shows an obvious performance gap of 2.5 BLEU between the stacked and PDS methods when the auxiliary CTC and pre-training methods are not used. This indicates that PDS helps convergence and improves ST when transcription is not available. When pre-training and CTC are available, better performance is achieved by good initialization and strong supervision. Also, PDS always outperforms the stacked method significantly.

<table>
<thead>
<tr>
<th>Setting</th>
<th>CTC</th>
<th>#Params</th>
<th>w/o PT</th>
<th>w/ PT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Transformer</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stack-4</td>
<td>✓</td>
<td>30M</td>
<td>20.3</td>
<td>22.9</td>
</tr>
<tr>
<td>PDS-Base-8</td>
<td>✓</td>
<td>32M</td>
<td>23.5</td>
<td>24.0</td>
</tr>
<tr>
<td>SATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stack-4</td>
<td>✓</td>
<td>40M</td>
<td>24.3</td>
<td>25.4</td>
</tr>
<tr>
<td>PDS-Base-8</td>
<td>✓</td>
<td>40M</td>
<td>24.9</td>
<td>25.8</td>
</tr>
</tbody>
</table>

4.4 Results of ST

End-to-end ST has become popular recently (Duong et al., 2016; Berard et al., 2016). However, unlike ASR, annotated speech-to-translation data is scarce, which prevents well-trained ST models. Therefore, we use CTC and pre-training for sufficient training (Bahar et al., 2019). According to the experimental results on ASR, we use PDS-Base-8 to investigate the effects of PDS on both performance and model convergence.

Table 3 shows an obvious performance gap of 2.5 BLEU between the stacked and PDS methods when the auxiliary CTC and pre-training methods are not used. This indicates that PDS helps convergence and improves ST when transcription is not available. When pre-training and CTC are available, better performance is achieved by good initialization and strong supervision. Also, PDS always outperforms the stacked method significantly.

On SATE (Xu et al., 2021), consistent improvements are achieved. The encoder of SATE is composed of an acoustic encoder and a textual encoder. We only employ PDS in the first encoder. Although an adaptor is introduced for adaptive representation, the length inconsistency issue is not solved. As a popular method, the shrink mechanism filters the acoustic representation based on the CTC predic-
Table 4: Impact of representation fusion. "F" represents the representation fusion method. We report the average WER of all 4 sets on LibriSpeech.

<table>
<thead>
<tr>
<th>Stack</th>
<th>F</th>
<th>Ratio</th>
<th>Stride</th>
<th>Layer</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>/</td>
<td>2</td>
<td>2-2</td>
<td>12</td>
<td>7.11</td>
</tr>
<tr>
<td></td>
<td>/</td>
<td>4</td>
<td>2-2</td>
<td>0-12</td>
<td>6.76</td>
</tr>
<tr>
<td></td>
<td>/</td>
<td>8</td>
<td>2-2-2</td>
<td>0-0-12</td>
<td>7.48</td>
</tr>
<tr>
<td></td>
<td>/</td>
<td>16</td>
<td>2-2-2-2</td>
<td>0-0-0-12</td>
<td>9.22</td>
</tr>
<tr>
<td>PDS</td>
<td>(B)</td>
<td>✓</td>
<td>4</td>
<td>2-2-1-1</td>
<td>2-3-3-3</td>
</tr>
<tr>
<td></td>
<td>(C)</td>
<td>✓</td>
<td>8</td>
<td>2-2-1-2</td>
<td>2-3-3-3</td>
</tr>
<tr>
<td></td>
<td>(D)</td>
<td>✓</td>
<td>8</td>
<td>2-2-1-2</td>
<td>2-2-6-2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓</td>
<td>16</td>
<td>2-2-2-2</td>
<td>2-2-6-2</td>
</tr>
<tr>
<td></td>
<td>(E)</td>
<td>✓</td>
<td>32</td>
<td>2-2-2-2</td>
<td>2-2-3-3-2</td>
</tr>
</tbody>
</table>

Table 5: Impact of the number of layers in each stage. We report the results of Transformer with PDS-Base-16 and PDS-Deep-16 settings.

<table>
<thead>
<tr>
<th>Layer</th>
<th>dev</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>clean</td>
<td>other</td>
<td>clean</td>
</tr>
<tr>
<td>2-2-2-6</td>
<td>3.91</td>
<td>3.57</td>
</tr>
<tr>
<td>2-2-4-4</td>
<td>3.83</td>
<td>3.57</td>
</tr>
<tr>
<td>2-2-6-2</td>
<td>3.71</td>
<td>3.74</td>
</tr>
<tr>
<td>5-5-10-10</td>
<td>3.19</td>
<td>3.57</td>
</tr>
<tr>
<td>5-5-12-8</td>
<td>3.15</td>
<td>3.38</td>
</tr>
<tr>
<td>5-5-15-5</td>
<td>3.27</td>
<td>3.60</td>
</tr>
</tbody>
</table>

Figure 5: The fusion weights of the output representation in each stage. We consider three settings of the number of layers under PDS-Base-16.

5 Analysis

Next, we study a number of interesting problems on 960h LibriSpeech.

5.1 Impact of Representation Fusion

To investigate the effect of information loss and the need of representation fusion, we compare the results with different down-sampling ratios (see Table 4).

For the vanilla stacked method in rows (A), the popular setting is to down-sample the input with a lower ratio of 4. This setting also achieves the best performance. Choosing a ratio of 2 leads to inferior WER because long but fine-grained features face the challenges of modeling. As the ratio of down-sampling increases, the performance drops significantly. This supports the point that information loss is serious in the stacked method.

For PDS, the system outperforms the stacked counterpart under the same setting of ratio = 4. However, we find that the fusion method does not obtain significant improvement. This might be because that the light condensation is lossless and cannot benefit from the fusion of representations.

Interestingly, the fusion method achieves consistent improvements when higher down-sampling ratios are adopted. To study it further, we add another set of experiments with a special setting in rows (D): the down-sampling ratio decreases from 16 to 8 by setting the stride of the final stage to 1. Then, we see a better WER of 6.59, which indicates less information loss under the lighter condensation. With the help of the fusion method, they achieve similar performances. This is consistent with what we expected.

5.2 Impact of Model Depth

We compare the results of the different number of layers in each stage. Table 5 shows the results on base and deep models.

For the model of 12 encoder layers, we use 2 fixed layers in the bottom 2 stages for less computation. As we described in Section 3.4, PDS achieves better performance as the number of layers increases in the intermedia stage. We think that there are two reasons. Firstly, the information loss is less compared with the top stages, and thus the encoding layers are more helpful. Secondly, sufficient encoding helps the down-sampling in the next stage, but it is not the case for the top stage. This is consistent with the previous conclusion (Huang et al., 2020). We also show the fusion weights in
Figure 5. The weight increases as the number of layers increases, and vice versa. The results are agreed with our design.

We also compare the results in a deep Transformer model with a 30-layer encoder. Due to the sufficient encoding in each stage, the deep model is robust in the design of the number of layers in each stage. There is no obvious performance gap across three different settings. It is very meaningful to combine PDS with the popular deep model (Pham et al., 2019) in the follow-up work.

5.3 Impact on Correlation

Unlike the stacked method, PDS runs a context interaction process after each down-sampling process. Figure 6 shows the correlation across different model architectures.

In Transformer, high correlation (about 60% - 80%) alleviates the information loss under the setting of PDS-Base-8. As the down-sampling ratio increases, fewer layers in each stage cannot capture the context information sufficiently and thus make the degraded correlation. This leads to the performance drops, as shown in Section 4.3.

Conformer and deep Transformer show similar performance under the different settings of down-sampling. This motivates us to investigate their behavior. Despite the limited layers in each stage, Conformer always shows a high correlation by the explicit local modeling. This also demonstrates the effectiveness of Conformer. The deep Transformer alleviates the issue straightly by stacking more layers.

One interesting finding is that the correlation of top layers is very high (> 90%) across all architectures. It may be affected by the strong supervision of the decoder. This inspires us to explore multitask learning in the future.

5.4 Distribution of Attention Weights

PDS generates the semantically complete units on the top of the encoder. We suppose that this informative representation has a greater effect on decoding. Refer to Zhang et al. (2020a), Figure 7 shows the distribution of summed cross-attention weights for each encoder representation.

Due to the fine-grained representations in the stacked method, the smaller attention weights spread across multiple relevant representations. In PDS, each representation receives greater attention as the down-sampling ratio increases. Although our method does not explicitly filter uninformative features, we argue that the stronger condensation forces the model to prefer more meaningful representations.

6 Conclusion

In this paper, we investigate the down-sampling process and shed light on the issue of information loss in the popular stacked method. This inspires us to propose a Progressive Down-Sampling method, which encodes the context information after each down-sampling. Furthermore, we develop a representation fusion method to combine the multi-scale information. Results on ASR and ST tasks demonstrate the effects of our method.
References


Kyuj H. Han, Jing Huang, Yun Tang, Xiaodong He, and Bowen Zhou. 2019. Multi-stride self-attention for speech recognition. In Interspeech


Yu Zhang, Daniel S. Park, Wei Han, James Qin, Anmol Gulati, Joel Shor, Aren Jansen, Yuanzhong Xu, Yanping Huang, Shibo Wang, Zongwei Zhou, Bo Li, Min Ma, William Chan, Jiahui Yu, Yongqiang Wang, Lianliang Cao, Khe Chai Sim, Bhuvana Ramabhadran, Tara N. Sainath,