Weakly Supervised Construction of ASR Systems from Massive Video Data

Mengli Cheng*, Chengyu Wang*, Jun Huang, Xiaobo Wang

Alibaba Group, Hangzhou, China
{mengli.cml, chengyu.wcy, huangjun.hj, yongshu.wxb}@alibaba-inc.com

Abstract

Despite the rapid development of deep learning models, for real-world applications, building large-scale Automatic Speech Recognition (ASR) systems from scratch is still significantly challenging, mostly due to the time-consuming and financially-expensive process of annotating a large amount of audio data with transcripts. Although several self-supervised pre-training models have been proposed to learn speech representations, applying such models directly might be sub-optimal if more labeled, training data could be obtained without a large cost.

In this paper, we present VideoASR, a weakly supervised framework for constructing ASR systems from massive video data. As user-generated videos often contain human-speech audio roughly aligned with subtitles, we consider videos as an important knowledge source, and propose an effective approach to extract high-quality audio aligned with transcripts from videos based on text detection and Optical Character Recognition. The underlying ASR models can be fine-tuned to fit any domain-specific target training datasets after weakly supervised pre-training on automatically generated datasets. Extensive experiments show that VideoASR can easily produce state-of-the-art results on six public datasets for Mandarin speech recognition. In addition, the VideoASR framework has been deployed on the cloud to support various industrial-scale applications.

Index Terms: automatic speech recognition, weakly supervised learning, optical character recognition, massive video data

1. Introduction

Automatic Speech Recognition (ASR) is one of the core tasks in speech processing. The task aims to generate transcripts from speech utterances. Recently, end-to-end neural ASR models have been extensively studied, as these models do not require the explicit learning of acoustic and language models [1, 2].

Despite the success, a potential drawback is that these models require large amounts of transcribed data to produce satisfactory results [3]. Unfortunately, transcribing audio by human annotators is both time-consuming and financially-expensive [4]. Recently, self-supervised pre-training has been applied to ASR [5, 6], using unlabeled audio to pre-train ASR models. However, there exists a learning gap between pre-training objectives (such as minimizing the contrast loss [6]) and ASR training objectives. We assume that better results could be obtained if the models could optimize similar goals during pre-training and fine-tuning. Some methods generate synthetic speeches aligned with texts for training [7, 8], but the generated speeches may still be different from real ones.

In order to support real-world applications, a natural question arises: is it possible to build accurate end-to-end ASR systems without much manually labeled data? In this work, we present a weakly supervised framework to construct ASR systems from massive video data, named VideoASR. The framework is shown in Figure 1, which consists of two major stages: Weakly Supervised Pre-training (WSP) and Domain-specific Fine-tuning (DF). During WSP, based on text detection [9] and Optical Character Recognition (OCR) [10], we extract human-speech audio aligned with subtitles from user-generated videos as knowledge sources to pre-train ASR models. Here, we pre-train our models over massive videos of varied topics so that the models can capture transferable, general knowledge across domains. The underlying ASR models can be fine-tuned to fit training data (usually smaller in size) in any domains. We evaluate VideoASR over popular ASR models and six public datasets. Results show that it produces state-of-the-art results for Mandarin speech recognition. VideoASR has also been deployed in an industrial-scale distributed machine learning platform to support various applications on the cloud.

2. Related Work

End-to-end ASR Models. While hybrid ASR techniques are continuously developing (such as classical DNN-HMM-style models [11]), due to the simple model pipelines, end-to-end ASR models have gained much attention. Recurrent-style networks are naturally suitable for end-to-end ASR as they model the sequences of audio and languages [12, 1, 13]; however, they may be slow during training and inference. This reduces the application scopes of such models in industry. CNN-based approaches [14, 15] are faster in speed, but they have limited capacity for modeling long sequences. Transformer-based methods [16, 17, 18, 19] have better performance because have strong abilities to capture long-term dependencies. They also converge faster and produce more accurate results when the CTC (Connectionist Temporal Classification) loss is added as an auxiliary loss [20]. Because the architecture design is not our major focus, we do not further elaborate.

Pre-training ASR Models. Two streams of works have been proposed to reduce the requirements of manually labeled data for end-to-end ASR. One stream applies unsupervised/self-supervised pre-training models for automatic speech recognition, e.g., [21, 22]. The other stream uses unlabeled speech data to pre-train ASR models, e.g., [23, 24].

* Equal contribution.
supervised/semi-supervised methods to tackle the problem. For example, Long et al. [21] propose semi-supervised training of DNN and RNN based acoustic models. Chung et al. [22] introduce the Auto-regressive Predictive Coding for learning speech representations. The wav2vec framework [5, 6] introduces self-supervised pre-training for transformer-based ASR. Inspired by BERT [23], Baevski et al. [6] propose masked predictive coding for unsupervised pre-training of transformer encoders. Similar works include [24, 25]. Besides, a few methods aim to generate pseudo labels of unlabeled speeches for ASR training [26, 27, 28]. The other stream extracts aligned text-speech segments using existing ASR models. Lanchantin et al. [29] introduce several heuristic rules to align paragraphs of transcripts with audio to generate training data. The works [30, 31] introduce several heuristic rules to extract useful speech segments with transcripts from Youtube. Different from previous methods, our work considers vision techniques to generate high-quality utterance-text pairs.

3. VideoASR: The Proposed Framework

In this section, we introduce technical details of our framework and the ASR model architectures that we use.

3.1. Weakly Supervised Pre-training

The pipeline of WSP is illustrated in Figure 2.

**Video Acquisition.** Many user-generated videos have embedded subtitles that are almost synchronous with the audio. We regard such videos as pre-training knowledge sources. The videos that we use are of various genres provided from Youku.

**Text and Audio Spotting.** Although videos with subtitles are available to us, subtitles are generally embedded in frame images in different styles and formats. This prevents us from extracting subtitles from raw data sources directly. Hence, we first extract frame images from each video with an interval of 1/3 second. Next, we employ IncepText [9] to detect text positions from images and the OCR model [10] to recognize texts.

Given a sequence of frame images within a time window size (denoted as $s_k, s_{k+1}, \cdots, s_{j-1}, s_j$), we wish to determine whether two consecutive frames $s_k$ and $s_{k+1}$ ($i \leq k, k+1 \leq j$) can be “merged” so that a subset of such frames may correspond to the same subtitle. Subsequently, the audio within the time frames is treated as the speech that roughly matches the subtitle. We present two merging methods: Heuristics-based and Model-based.

For two consecutive frames $s_k$ and $s_{k+1}$, denote the detected texts as $t_k$ and $t_{k+1}$, respectively. Define the Relative Edit Distance (RED) between $s_k$ and $s_{k+1}$ as:

$$RED(s_k, s_{k+1}) = \frac{EditDist(t_k, t_{k+1})}{\max(\text{Len}(t_k), \text{Len}(t_{k+1}))}$$

where $EditDist(t_k, t_{k+1})$ is the edit distance between $t_k$ and $t_{k+1}$, and $\text{Len}(t_k)$ is the length of $t_k$. Heuristics-based Merging combines two frames $s_k$ and $s_{k+1}$ if $RED(s_k, s_{k+1})$ is smaller than a tuned threshold.

However, Heuristics-based Merging may ignore the corresponding relations between audio and texts. If any existing ASR model is available, we can use it to refine the merging process. Let $a_k$ be the audio segment w.r.t. the frame $s_k$. Model-based Merging employs an existing model $f$ to predict the transcript of $a_k$, denoted as $f(a_k)$. If $s_k$ and $s_{k+1}$ should not be merged, the error rate of model $f$ is computed as:

$$ERR_1(f, s_k, s_{k+1}) = CER(t_k, f(a_k)) + CER(t_{k+1}, f(a_{k+1}))$$

where $CER(t_k, f(a_k))$ is the Character Error Rate (CER) of model $f$’s predictions. If $s_k$ and $s_{k+1}$ should be merged, similarly, we have the combined error rate:

$$ERR_2(f, s_k, s_{k+1}) = \min(CER(t_k, f(a_{k+1}))), CER(t_{k+1}, f(a_{k+1}))$$

where $a_{k+1}$ concatenates $a_k$ and $a_{k+1}$. $s_k$ and $s_{k+1}$ should be merged if $ERR_1(f, s_k, s_{k+1}) > ERR_2(f, s_k, s_{k+1})$.

**Iterative Pre-training.** After extraction and merging, we obtain a large “pseudo-labeled” dataset $D = \{(a_k, t_k)\}$, consisting of audio-transcript segment pairs. We pre-train the ASR model using the way as normal training over the dataset $D$. It should be noted that the extraction process of $D$ unavoidably injects noise into the dataset due to the lack of human annotation. During pre-training, we apply a self-training strategy to filter out noisy data. In each epoch, we filter out audio-transcript segments $\{(a_k, t_k)\}$ from $D$ that are most likely to have noisy transcripts and use the remaining dataset for the next training epoch. Due to space limitation, we omit the details and refer interested readers to [32].

3.2. Domain-specific Fine-tuning

Based on the WSP learning objective, our framework could generate ready-to-use ASR models directly. However, the domains of pre-training data may be significantly different from downstream ASR tasks. Hence, given a (small) training set $D_m = \{(a_k, t_k)\}$ of domain $m$, we fine-tune the pre-trained model over $D_m$ to learn domain-adaptive parameters. One can also leverage transfer learning using both $D$ and $D_m$ to improve the fine-tuning performance, which is left as future work.

3.3. Choices of Model Architectures

Following industry practices, we consider two popular ASR models: wav2letter [14] and Speech Transformer [16], as shown in Figure 3. Wav2Letter uses one dimensional convolution networks with large kernels as encoders, and the CTC loss for training. Its efficient inference speed makes it appealing to industrial applications. Speech Transformer [16] adopts self-attention for acoustic modelling and decoding. Following [20], the CTC loss is added as an auxiliary loss to achieve faster convergence and better performance. In multi-head attention layers, we set the hidden size as 512, with 8 heads. For fast online inference, we apply a beam search of size 16 for both models.

---

2Youku (http://www.youku.com) is a popular video hosting service, a subsidiary of Alibaba Group. It holds the copyrights of these videos, and permits authors to obtain and process the data as described. Different from standard movies, the user-generated videos usually do not have subtitles provided as standalone files.

3We use a commercial Mandarin ASR service to transcribe the audios. See https://ai.aliyun.com/nls/asr.
In this section, we conduct extensive experiments to evaluate the proposed framework in various aspects. We also present the results of industrial deployment to demonstrate the effectiveness of the VideoASR framework.

4.1. Datasets and Experimental Settings

For ASR pre-training, we obtain 43,694 video clips of various topics from Youku. The total duration is around 8,000 hours. During WSP, the learning rates of wav2letter and Speech Transformer are set as 0.05 and 1.0, respectively. During fine-tuning, we set the learning rates to be 0.01 and 0.5. For both models, we normalize the utterances to 16kHz and generate the logarithm of FBank features of 80 dimensions, with a window size of 20ms and the stride of 10ms. SpecAugment [33] is applied for data augmentation. All algorithms are implemented in Tensorflow and run on GPU servers.

For evaluation, we follow the work [34] to evaluate the models on two popular Mandarin ASR datasets: AISHELL-1 and HKUST. The baselines models include the classical TDNN model [13] and the Speech Transformer with and without pre-training on unlabeled human speeches [16]. Specifically, in [16], over 10,000 unlabeled audios are employed to pre-train the model. We also consider four other public datasets, namely, ST_CMDS, AISHELL-2, AIDATANG and MagicData. The statistics of all six datasets are shown in Table 1. The datasets are varied in domains and styles and have relatively short duration compared to our WSP dataset. We keep the training, development and testing splits of all the datasets as default. In the experiments, we benchmark the two models (wav2letter and Speech Transformer) under two settings (with and without WSP) on all six datasets by ourselves.

4.2. Experimental Results and Model Analysis

General Performance. All the experimental results are summarized in Table 2. We have the following findings: i) Speech Transformer outperforms wav2letter across all the datasets. ii) The WSP technique in VideoASR effectively boosts the performance of both models on all the datasets. This phenomenon is more significant on small datasets (i.e., AIDATANG and HKUST). iii) Speech Transformer with WSP achieves state-of-the-art performance on all the six public datasets, outperforming baseline ASR approaches.

Analysis of WSP. We test both merging techniques via a manual check on 2,000 consecutive frame pairs that require proper merging. It shows model-based merging consistently produces better results (with a CER of 12.5%, compared to 33.0% for rule-based merging). This shows, even without human annotation, we can generate pre-training datasets with tolerable error rates. After text and audio spotting, we obtain a total of 1,825,927 utterances from all videos. The duration ranges from 15s to 20s. Next, we evaluate the iterative pre-training technique. We filter out part of the data (quantified by the drop ratio $\gamma$) and take the rest as the pre-training data for the next iteration. We search for the best value of $\gamma$ from 0, 0.5%, 1.0%, 2.0% and also compare our method with a classical data filtering approach [30]. We use the Mandarin ASR model from https://ai.aliyun.com/nls/asr for the implementation of [30], instead of their original English ASR model. In Table 3, we display the CER values produced by pre-trained wav2letter without fine-tuning, evaluated on the AISHELL-1 development set. It shows that WSP with $\gamma = 1.0\%$ has the best performance. After iterative pre-training, we obtain a “cleaner” pre-training ASR dataset. The CER of extracted speech-transcript pairs after noise filtering is around 6%, close to those of manually labeled datasets.11

Convergence analysis. During the DF stage, we investigate how WSP affects the DF performance. The convergence curves on HKUST are shown in Figure 5. As seen, wav2letter and Speech Transformer converge within 10 and 3 training epochs, respectively. Compared to the same models without WSP, the speed of convergence is much faster for both models, which clearly indicates WSP is able to find better parameter initialization for domain-specific ASR tasks, no matter whether there exist domain differences between the two datasets.

Error analysis and case studies. We analyze the percentages of different error types occurred in the test sets of AISHELL-1 and HKUST, shown in Table 4. The underlying ASR models are Speech Transformer w. and w/o. WSP. The majority of the errors are substitution errors caused by homophones. The WSP
Table 2: Performance of ASR models on public test datasets in terms of CER (%). * refers to our own implementation.

<table>
<thead>
<tr>
<th>Model</th>
<th>ST_CMS</th>
<th>AISHELL-1</th>
<th>AISHELL-2</th>
<th>AIDATANG</th>
<th>MagicData</th>
<th>HKUST</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDNN [3]</td>
<td>-</td>
<td>8.7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>32.7</td>
</tr>
<tr>
<td>Transformer (w/o. pre-training) [34]</td>
<td>-</td>
<td>9.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>23.8</td>
</tr>
<tr>
<td>Transformer (w. pre-training) [34]</td>
<td>-</td>
<td>7.4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>21.0</td>
</tr>
<tr>
<td>wav2letter w/o. WSP*</td>
<td>4.5</td>
<td>11.7</td>
<td>12.5</td>
<td>12.9</td>
<td>7.4</td>
<td>35.7</td>
</tr>
<tr>
<td>wav2letter w. WSP*</td>
<td>2.4</td>
<td>7.1</td>
<td>10.0</td>
<td>9.2</td>
<td>6.7</td>
<td>29.3</td>
</tr>
<tr>
<td>Transformer w/o. WSP*</td>
<td>4.4</td>
<td>6.7</td>
<td>7.4</td>
<td>7.8</td>
<td>3.6</td>
<td>23.5</td>
</tr>
<tr>
<td>Transformer w. WSP*</td>
<td>2.1</td>
<td>5.9</td>
<td>5.9</td>
<td>4.9</td>
<td>3.3</td>
<td>20.0</td>
</tr>
</tbody>
</table>

Table 3: Performance of pre-trained wav2letter with different data filtering techniques in terms of CER (%).

<table>
<thead>
<tr>
<th>Method/Iteration</th>
<th>4</th>
<th>8</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Liao et al. [30]</td>
<td>17.3</td>
<td>16.8</td>
<td>16.5</td>
</tr>
<tr>
<td>WSP (* = 0)</td>
<td>16.1</td>
<td>15.0</td>
<td>14.2</td>
</tr>
<tr>
<td>WSP (* = 0.5%)</td>
<td>15.4</td>
<td>14.4</td>
<td>13.6</td>
</tr>
<tr>
<td>WSP (* = 1.0%)</td>
<td>15.3</td>
<td>14.2</td>
<td><strong>13.3</strong></td>
</tr>
<tr>
<td>WSP (* = 2.0%)</td>
<td>15.6</td>
<td>14.9</td>
<td>14.7</td>
</tr>
</tbody>
</table>

Table 4: Error analysis in terms of CER (%).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>WSP?</th>
<th>Insertion</th>
<th>Deletion</th>
<th>Substitution</th>
</tr>
</thead>
<tbody>
<tr>
<td>AISHELL-1 No</td>
<td>0.1</td>
<td>0.2</td>
<td>6.4</td>
<td></td>
</tr>
<tr>
<td>AISHELL-1 Yes</td>
<td>0.1</td>
<td>0.2</td>
<td>5.7</td>
<td></td>
</tr>
<tr>
<td>HKUST    No</td>
<td>2.6</td>
<td>3.6</td>
<td>17.3</td>
<td></td>
</tr>
<tr>
<td>HKUST    Yes</td>
<td>2.7</td>
<td>2.6</td>
<td>14.7</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4: Cases of model prediction w. and w/o. WSP. Italic texts refer to pronunciation (spelled in Mandarin phonetic symbols).

Figure 5: Convergence curves on HKUST. (X-axis: number of epochs; Y-axis: CER on the development set.)

deployed on the EasyASR distributed machine learning platform [35], which supports efficient training and serving large-scale ASR models across multiple workers and GPUs. Users also have the options to customize the end-to-end ASR model structures easily. Hence, our solution is highly applicable for industrial-scale scenarios.

Evaluation on the in-house, e-commerce dataset. To further demonstrate the practical values of our work, we present the evaluation results on our in-house corpus. Our dataset is used in the e-commerce live-streaming domain, consisting of 350h human-speech audios with manually labeled transcripts. We hold out 2,761 audio clips (each around 15s) for evaluating the performance of wav2letter and Speech Transformer. Parameter settings are the same as in previous experiments. In the experiments, we report the performance on both model accuracy and batch inference time. Specifically, the inference time is measured using one Tesla V100 GPU (16GB), with the batch size set to be 32. The experimental results are summarized in Table 5. We can see that our pre-training technique significantly improves the model accuracy in both cases. In addition, although wav2letter is relatively inaccurate, its fast inference speed makes it appealing to real-time applications.

5. Conclusion and Future Work

In this paper, we present VideoASR and construct accurate ASR systems based on the weak supervision of massive video data. With WSP and the Speech Transformer model with our modifications, we achieve the state-of-the-art results on several public datasets. We also deploy our framework in the distributed machine learning platform and achieve desirable performance on our in-house datasets. Future work includes i) applying our approach to other languages and ASR models; ii) combining unsupervised and weakly supervised pre-training for ASR; and iii) leveraging transfer learning to improve model fine-tuning.
6. References


