CTC Network with Statistical Language Modeling for Action Sequence Recognition in Videos

Mengxi Lin, Nakamasa Inoue, Koichi Shinoda
Tokyo Institute of Technology, Tokyo, Japan
{mengxi,inoue}@ks.c.titech.ac.jp,shinoda@c.titech.ac.jp

ABSTRACT

We propose a method for recognizing an action sequence in which several actions are concatenated and their boundaries are not given. The proposed method combines Connectionist Temporal Classification (CTC) and a statistical language model. CTC can learn the nature of each element action given no boundary information in an end-to-end manner. The statistical language model can learn the relationship between actions. We evaluate our method on the *Breakfast* dataset. When we use the trigram as the language model, its accuracy rate is 43.4%, which is better than the state-of-the-art ECTC method by 6.7 percentage points.

KEYWORDS

connectionist temporal classification; action sequence recognition; statistical language model; weakly supervised learning

1 INTRODUCTION

In the past few decades, a lot of studies in computer vision have been devoted to human action understanding in videos. These studies have developed many effective visual representations and models, in order to perceive human actions in videos [1–4]. Despite their great progress, they mainly focus on the recognition paradigm which heavily relies on the pre-segmented videos that contain only one action in one clip during training. It requires us to take expensive labor effort to annotate the boundaries between different actions as well as between actions and non-action in videos for training the action models. With hundreds of thousands of videos produced in every day, the cost of temporal boundary annotation refrains us from leveraging these resources for building up more accurate recognition systems.

To solve this problem, several works [5, 6] have been proposed to recognize an action sequence that consists of more than one action, where no temporal boundaries between different actions are given both in the training and test phases. Since the temporal order annotations of actions are easy to generate, it paves a way for utilizing larger amounts of data for model learning by avoiding expensive action boundary annotations.

In previous works [5, 7], Recurrent Neural Network (RNN) trained with Connectionist Temporal Classification (CTC) [8] has shown its effectiveness on modeling video data when only given temporal order supervision. CTC views the outputs of an RNN in all time steps of a video as a whole, and enables the RNN to learn the mapping between the video and the target sequence directly. The Extended Connectionist Temporal Classification (ECTC) method proposed in [5] applies CTC to action segmentation and action sequence recognition. It takes into account the visual similarity between frames to guide the training in CTC.

One major problem of the CTC framework for action sequence recognition is that it hardly captures the relationship among actions, in spite of the fact that the relationship is strongly presented in action sequences; e.g. the action *pourWater* is more likely to take place than *pourCoffee* after *addTeabag*. The reason is that the RNN on which CTC is applied still has difficulty in encoding long-term information to cover enough context for capturing the action relations, even with a sophisticated gating design of LSTM [9]. Some evidence given in Singh et al.'s work [10] showed that a typical Bi-Directional-LSTM [11] can only remember 120 frames on average as context. However, the temporal interval between actions is usually longer than 120 frames. For instance, the action *pourDough* may happen after the action *stirDough* that takes about 2 *mins* to complete. This imposes difficulty for the RNN to capture the relationship between *pourDough* and *stirDough*, as well as the relationship between *pourDough* and *pourMilk* that precedes *stirDough*, since it requires the RNN to remember at least 2 *mins* context, i.e. 180 frames under 1.5fps. Some works based on the hierarchical temporal pooling on RNN [12] may alleviate the problem to some extent without significantly increasing the model size and computational cost, but the temporal pooling results in the loss of temporal resolution, which is disadvantageous for learning the nature of element actions, especially in the case of recognizing fine-grained actions [13].

Recently, attention-based RNN [14] has achieved satisfying results on video captioning [15], which can be formulated as the learning problem of mapping a source sequence (a video sequence in video captioning) to a target sequence (a word sequence in video captioning) without any alignment information during training, as in our problem. It has the merit that the relationship among the labels in the target sequence is likely to be captured by allowing the network to

ThematicWorkshops'17, October 23–27, 2017, Mountain View, CA, USA.
© 2017 ACM. ISBN 978-1-4503-5416-5/17/10...$15.00
DOI: https://doi.org/10.1145/3126686.3126755

attend to all time steps of the source sequence and memorizing the previous outputs in the target sequence. However, in the problem of action sequence recognition, actions have more rigid time structures than the natural language, which the attention model may fail to capture due to its over-flexibility.

To address these difficulties, we propose to model the relationship between actions explicitly using a statistical n-gram language model, and combine it with a CTC-trained RNN for action sequence recognition. While the statistical n-gram language model can be easily trained from action sequences, it can well capture the distribution of action sequences and generalize to the ones that have never been seen in the training with backoff smoothing. Combining the n-gram language model with the CTC network helps to effectively identify incorrect action sequences that do not conform to either the CTC network or the statistical n-gram language model, leading to better performance of action sequence recognition.

We evaluate our method in the realistic *Breakfast* dataset [13], which records the activities of preparing breakfast. A video in the dataset contains a sequence of various actions. The actions in one video may appear in the other videos as well, but with different combinations or temporal orders. Our method only exploits the temporal order annotations of actions to construct the model, and the experimental results show that our method substantially outperforms the other state of the art.

This paper is organized as follows. In Section 2, we review some of the related work. After introducing Connectionist Temporal Classification (CTC) and statistical n-gram language models in Section 3 and 4 respectively, we present our approach in Section 5. Finally, we show our experimental results and analysis in Section 6, followed by our conclusion in Section 7.

2 RELATED WORK

2.1 Action Learning from Pre-segmented Videos.

There has been a large body of research work devoted to the recognition of human actions in pre-segmented videos [1, 2, 16, 17]. In order to understand the visual patterns of human actions, researchers have developed many sophisticated hand-crafted spatio-temporal features [1, 16, 18–20]. Among these hand-crafted features, Improved Dense Trajectory [1] proposed by Wang et al. outperforms the others in many realistic datasets, e.g. HMDB51 [21] and UCF101 [22]. They proposed to densely sample points from videos and track them across frames, followed by extracting several low-level features around the points and aggregating them using Fisher Vector Encoding [23] or VLAD [24] to generate a representation of the video clip for classification with SVMs.

Recently, deep visual representations learned from deep neural networks and big data achieved surprising results in image classification [25, 26]. Inspired by the work in the image classification field, many researchers turned to developing deep neural network architectures to learn effective representations for human actions [2–4]. Gan et al. [2] exploited Convolutional Neural Network (CNN) that was trained in ImageNet [27] and fine-tuned in the target datasets for event detection in videos. Ji. et al. [28] and Tran. et al. [4] extended the 2D convolution to 3D to additionally learn the short-term motion patterns of human actions. In [3, 17], a two-stream CNN architecture was proposed, which takes RGB and optical flow images of videos as the input to two independent 2D-CNNs respectively, and fuses the learned representations from both the CNNs for predicting the actions in a video.

Besides the achievements got by creating advanced feature representations, some studies showed the effectiveness of temporal modeling in action recognition. Shi et al. [29], Kuehne et al. [13] and Chen et al. [30] proposed to employ Hidden Markov Models to capture the transitions of action states. Gaidon et al. [31] utilized finer annotations of an action to learn temporal templates for the action. Recently, neural-network-based methods made a breakthrough in modeling the long-term temporal information in videos. 1D temporal convolutional filters were utilized in [32–34], which were coupled with a 2D/3D-CNN for action detection and action segmentation. Some researchers [10, 35–38] experimented on LSTM-based Recurrent Neural Networks [9], obtaining the state-of-the-art performance in many action recognition datasets.

In addition to the dedicated effort made for modeling the temporal structures of video frames, some prior works attempted to model the temporal dependency of action labels [39–41]. They trained their classification models in pre-segmented videos and created grammars or statistical n-gram language models of action labels, to guide their classification models for action detection and parsing.

In spite of the progress made by the above works, they all rely on the segmentation of videos for learning. The cost of boundary annotations refrains their models from scaling up to exploit larger amounts of data.

2.2 Action Learning from Unsegmented Videos.

Compared with the methods that perform learning from pre-segmented videos, the approaches focusing on learning from unsegmented videos are relatively few. Given action labels along with unsegmented videos, UntrimmedNet [42] proposed by Wang et al. relies on sorts of proposal mechanisms to generate short segments from the videos for training the action models. Bojanowski et al. [43, 44] formulated the learning from temporal order annotations as a discriminative clustering problem. Their method used the temporal order annotations as constraints to cluster frames in unsegmented videos, and simultaneously learned a classifier that classified each frame in videos independently. Regardless the simplicity of the above methods, they can only deal with short-term information for action recognition in videos.

Besides the works of [43, 44], the method proposed by Kuehne et al. [6] can learn their models under temporal order supervision as well, while being able to take into account the temporal structures of video frames. Inspired by the success of

Hidden Markov Model (HMM) applying in speech recognition without phoneme boundary annotations, Kuehne et al. [6] evaluated it on video data for action segmentation. The HMM they employed is able to model the temporal structures to some extent. However, it relies on the Markov assumption and has to associate a fixed parametric distribution, e.g. Gaussian Mixture Model, to its states, which brings a limit to its application to action sequence recognition that presents complicated temporal and visual patterns.

Recently, Recurrent Neural Network (RNN) drew researchers' attentions, for its power in modeling temporal dynamics and learning feature representation more flexibly than the traditional temporal modeling methods like HMM. Along with RNNs, Connectionist Temporal Classification [8] proposed by Graves et al. makes the sequence learning possible when the boundaries of labels in a sequence are not given. This advantage of CTC opens up the possibility to exploit large amounts of unsegmented sequence data for building RNN models, leading to the state-of-the-art performance in many fields [7, 45, 46]. In the field of human action understanding, Huang et al. [5] proposed to adapt the CTC framework to take into account the visual similarity between frames for action segmentation and action sequence recognition. They achieved the state-of-the-art result when the segmentation of training videos was not available. However, as discussed in Section 1, their method still can hardly catch the relationship between actions.

2.3 Statistical Language Model

A statistical language model is able to estimate the probability of a sentence, revealing how likely a sentence is generated from a distribution. Therefore, a statistical language model can provide useful information to assist many tasks related to the natural language, including machine translation [47], speech recognition [48], text retrieval [49] and etc, where the estimation of sequence likelihoods can help to constrain the inference of the final results.

Researchers have developed many kinds of statistical language models. N-gram statistical language model is the most widely used one. It makes Markov assumption and assigns the probability of a word according to its preceding $(n-1)$ words, which can be estimated by simply counting the occurrences of word subsequences in the training set. Despite its simplicity, it is effective given the fact that sequences usually present strong local structures in most of the problems, meeting with its Markov assumption. In some cases, some of the word subsequences may have zero count in the training set, leading to the estimation of the probability being under-estimated to zero. To amend it, several smoothing methods have been proposed [50–55]. Recently, Richard et al. [41] proposed to apply the n-gram model to model the relationship of action labels for assisting action detection in videos. However, their framework only allowed them to combine the n-gram model with action classifiers that were trained in segmented videos.

Besides the n-gram language model, several other statistical language models were also proposed [56–59]. Despite their relative success in modeling the language sequence, they have been found to give only comparable or moderate improvement over the n-gram language model when combining with applications like speech recognition [48].

3 CONNECTIONIST TEMPORAL CLASSIFICATION

Let $\boldsymbol{X} = (\boldsymbol{x}^1, \boldsymbol{x}^2, ..., \boldsymbol{x}^T)$ denote a sequence of activations of a video output by an RNN, and $\boldsymbol{y} = (y^1, y^2, ..., y^N)$ denote its action sequence, where the number of the activations is not smaller than the number of the actions, i.e. $T \geq N$. CTC defines the probability of an action sequence $\boldsymbol{y}$ given the activations $\boldsymbol{X}$:

$$p(\boldsymbol{y}|\boldsymbol{X}) = \sum_{\boldsymbol{\pi} \in \{\boldsymbol{\pi}:\mathfrak{R}(\boldsymbol{\pi})=\boldsymbol{y}\}} p(\boldsymbol{\pi}|\boldsymbol{X}), \tag{1}$$

where $\boldsymbol{\pi} = (\pi^1, \pi^2, ..., \pi^T)$ is a path denoting the emission of action labels along $\boldsymbol{X}$, and $\mathfrak{R}$ is an operator to remove consecutively repeated action labels in $\boldsymbol{\pi}$, e.g. both the action paths $[A,A,B,C]$ and $[A,B,C,C]$ are mapped to the same action sequence $[A,B,C]$. In this way, every $\boldsymbol{x}^t$ in $\boldsymbol{X}$ contributes to the generation of each label in $\boldsymbol{y}$ by aligning different paths to it through $\mathfrak{R}$. Then, CTC makes an assumption that each π^t in $\boldsymbol{\pi}$ is conditionally independent given $\boldsymbol{X}$:

$$p(\boldsymbol{\pi}|\boldsymbol{X}) = \prod_{t=1}^{T} p(\pi^t|\boldsymbol{X}). \tag{2}$$

The value of $p(\pi^t|\boldsymbol{X})$ is given by the soft-max output of an RNN at time t by mapping $\boldsymbol{x}^t$ to the probabilities of action classes via a linear and soft-max layer.

Note that the complexity of computing $p(\boldsymbol{y}|\boldsymbol{X})$ grows exponentially with respect to the length of $\boldsymbol{X}$, if we apply the summation in Equation 1 directly in a brute-force way. For that, Graves et al. [8] proposed an efficient dynamic programming method to compute it. Specifically, let $\boldsymbol{\pi}_1^t$ denote the partial path of $\boldsymbol{\pi}$ from time 1 to t, and $\boldsymbol{y}_1^k$ denote the partial action label sequence composed by the first k labels in order from $\boldsymbol{y}$. Then we can define a variable $\alpha^t(k)$, which is the sum of the probabilities of $\boldsymbol{\pi}_1^t$ that can be aligned to $\boldsymbol{y}_1^k$:

$$\alpha^t(k) = \sum_{\boldsymbol{\pi} \in \{\boldsymbol{\pi}:\mathfrak{R}(\boldsymbol{\pi}_1^t)=\boldsymbol{y}_1^k\}} p(\boldsymbol{\pi}_1^t|\boldsymbol{X}). \tag{3}$$

By taking the conditional independence assumption in Equation 2, we can derive $\alpha^t(k)$ recursively:

$$\alpha^t(k) = [\alpha^{t-1}(k-1) + \alpha^{t-1}(k)]s^t(y^k), \tag{4}$$

where $s^t(y^k)$ is the probability of emitting the label y^k at time t, i.e. $s^t(y^k) = p(\pi^t|\boldsymbol{X})$, for all the $\boldsymbol{\pi}$ that have a label y^k at time t. Intuitively, Equation 4 defines the mapping of $\boldsymbol{X}$ to $\boldsymbol{y}$ up to time t and the k-th label, allowing only the transitions from the $(k-1)$-th and the k-th labels at time $t-1$.

By dynamic programming, CTC is capable to compute the probability $p(\boldsymbol{y}|\boldsymbol{X})$ by deriving $\alpha^T(N)$ in linear time with respect to the length of the video. Using the probability

$p(\boldsymbol{y}|\boldsymbol{X})$, CTC defines the loss for an RNN as:

$$L = -\log p(\boldsymbol{y}|\boldsymbol{X}) \tag{5}$$

For the sake of calculating the gradients of the loss with respect to the parameters of an RNN efficiently, CTC introduces another variable $\beta^t(k)$ that defines the mapping of $\boldsymbol{X}$ to $\boldsymbol{y}$ up to time t and k-th label, but from the end of $\boldsymbol{X}$ and $\boldsymbol{y}$ instead of from the beginning as when we calculate $\alpha^t(k)$. With $\alpha^t(k)$ and $\beta^t(k)$, it is possible to differentiate the loss function in Equation 5 with respect to every output $s^t(m)$, i.e. the soft-max output of the RNN for the m-th action class at time t, which allows us to train the network using backpropagation.

Generally, the output of a CTC network can be decoded with the best path [5, 8], i.e.

$$\begin{aligned}\boldsymbol{y}^* &= \arg\max_{\boldsymbol{y}} \sum_{\boldsymbol{\pi} \in \{\boldsymbol{\pi}:\Re(\boldsymbol{\pi})=\boldsymbol{y}\}} p(\boldsymbol{\pi}|\boldsymbol{X}) \\ &\approx \arg\max_{\boldsymbol{y}} \max_{\boldsymbol{\pi} \in \{\boldsymbol{\pi}:\Re(\boldsymbol{\pi})=\boldsymbol{y}\}} p(\boldsymbol{\pi}|\boldsymbol{X}) \\ &= \Re(\arg\max_{\boldsymbol{\pi}} p(\boldsymbol{\pi}|\boldsymbol{X})),\end{aligned} \tag{6}$$

by approximating $p(\boldsymbol{y}|\boldsymbol{X})$ with $p(\boldsymbol{\pi}|\boldsymbol{X})$ where $\boldsymbol{\pi}$ is the most probable path corresponding to $\boldsymbol{y}$. Therefore, in the best path decoding, we only need to find out the label with the largest probability at each time step and concatenate them to form a path, followed by applying $\Re(\cdot)$ operator to delete the consecutively repeated labels in the path to form the action sequence. Despite the simplicity of the best path decoding, it does not allow the incorporation of any other external information for inference.

4 STATISTICAL N-GRAM LANGUAGE MODEL

Let $\boldsymbol{y} = (y^1, y^2, .., y^N)$ denote a sequence of length N. A statistical n-gram language model can be described as follow:

$$p(\boldsymbol{y}) = \prod_{k=1}^{N} p(y^k|\boldsymbol{y}_1^{k-1}) = \prod_{k=1}^{N} p(y^k|\boldsymbol{y}_{k-n+1}^{k-1}), \tag{7}$$

where y^k denotes the k-th label in $\boldsymbol{y}$ and $\boldsymbol{y}_l^k$ denotes a subsequence composed by the l-th to k-th labels in order from $\boldsymbol{y}$. In the n-gram language model, we call a subsequence of length n an *n-gram*, and the probability of emitting a label in a sequence is assumed to be only dependent on its $n-1$ preceding labels. We can count the number of the occurrence of $\boldsymbol{y}_{k-n+1}^{k}$ and the occurrence of $\boldsymbol{y}_{k-n+1}^{k-1}$ in the sequences of the training set. The Maximum Likelihood (ML) estimation of the probability $p(y^k|\boldsymbol{y}_{k-n+1}^{k-1})$ is given by:

$$p(y^k|\boldsymbol{y}_{k-n+1}^{k-1}) = \frac{C(\boldsymbol{y}_{k-n+1}^{k})}{C(\boldsymbol{y}_{k-n+1}^{k-1})}, \tag{8}$$

where $C(\cdot)$ denotes the corresponding counts of subsequences appearing in the training set.

Since the training set is usually too small to give robust ML estimation (Equation 8), and under-estimates the probability to zero when an n-gram is unseen in the training set, some of the methods [50–54] are proposed to smooth the language model. Among them, an empirical study conducted by Chen et al. [60] showed that the Katz backoff model [52] and the Jelinek-Mercer [51] interpolation model perform consistently well, with the Katz backoff model being slightly better. Therefore, we adopt the Katz backoff model to smooth the language model. The basic idea behind the Katz backoff model with discounting is to steal some probability mass from the n-gram observed in the training data and re-distribute it to the unseen ones by referring back to the lower order n-grams, i.e. $(n-1)$gram. The following shows the recursive equation of the backoff model to calculate the probability estimation of an n-gram:

$$p^*(y^k|\boldsymbol{y}_{k-n+1}^{k-1}) = \begin{cases} \tilde{p}(y^k|\boldsymbol{y}_{k-n+1}^{k-1}), & \text{if } C(\boldsymbol{y}_{k-n+1}^{k}) > 0, \\ \gamma(\boldsymbol{y}_{k-n+1}^{k-1}) p^*(y^k|\boldsymbol{y}_{k-n+2}^{k-1}), & \text{otherwise.} \end{cases} . \tag{9}$$

where $p^*(y^k|\boldsymbol{y}_{k-n+1}^{k-1})$ is the smoothed estimation of an n-gram probability, $\tilde{p}(y^k|\boldsymbol{y}_{k-n+1}^{k-1})$ is the discounted probability of an n-gram that can be observed in the training, $\gamma(\boldsymbol{y}_{k-n+1}^{k-1})$ is the renormalization factor that guarantees the smoothed conditional probabilities sum up to one over all the label types. For $\tilde{p}(y^k|y_{k-n+1}^{k-1})$, the Witten-Bell discounting method [55] can be applied, which is more effective than the other discounting methods [61] when dealing with a small corpus like the one consisting of action sequences in our experiments. It additionally regards the first occurrences of the n-grams in the training data as the occurrences of unseen n-grams, leading to:

$$\tilde{p}(y^k|\boldsymbol{y}_{k-n+1}^{k-1}) = \frac{C(\boldsymbol{y}_{k-n+1}^{k})}{C(\boldsymbol{y}_{k-n+1}^{k-1}) + \delta(\boldsymbol{y}_{k-n+1}^{k-1})}, \tag{10}$$

where $\delta(\boldsymbol{y}_{k-n+1}^{k-1})$ denotes the number of label types with the preceding context $\boldsymbol{y}_{k-n+1}^{k-1}$ observed in the training data. For theoretical justification of Witten-Bell discounting, see [55].

5 OUR APPROACH

5.1 Framework

Figure 1 shows the framework of our approach when recognizing an action sequence in an unsegmented video. It consists of four components, namely feature extraction module, RNN prediction module, statistical language module and decoder module. In recognition stage, feature extraction module extracts appearance and motion feature for each frame in an unsegmented video. The extracted features are fed into the CTC-trained RNN to model the temporal structure of the video, producing the probabilities of the action labels at each frame. Meanwhile, a statistical language model is learned from the action sequences in the training data to capture action relationship. Finally, the decoder module combines the knowledge from the network and the statistical language model to derive the most probable action sequence that involves in the video. Our main contribution falls into combining the statistical language model of actions and CTC network output for the decoder module.

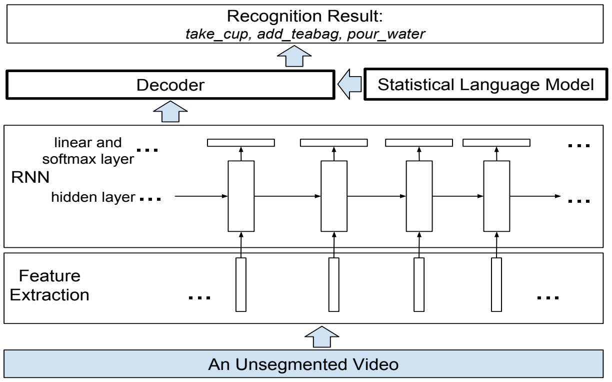

Figure 1: The framework of our approach: when performing action sequence recognition, our framework extracts the appearance and motion features for frames in the unsegmented video, which are fed into an RNN to produce probabilities of action labels at each frame. Then a decoder will take the output of the network and incorporate the information of a statistical language model of actions to generate the recognition result.

5.2 Combining CTC Network with Statistical Language Model

Basically, we want to find the action sequence that maximizes a posterior probability given an RNN encoded sequence of a video:

$$\begin{aligned}\boldsymbol{y}^* &= \arg\max_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{X}) \\ &= \arg\max_{\boldsymbol{y}} \frac{p(\boldsymbol{X}|\boldsymbol{y})p(\boldsymbol{y})}{p(\boldsymbol{X})} \\ &= \arg\max_{\boldsymbol{y}} p(\boldsymbol{X}|\boldsymbol{y})p(\boldsymbol{y}),\end{aligned} \tag{11}$$

where $\boldsymbol{y}$ denotes an action sequence and $\boldsymbol{X}$ denotes an RNN encoded sequence of a video. Here, Bayes Rule is applied to convert the posterior probability $p(\boldsymbol{y}|\boldsymbol{X})$ to the product of the conditional probability $p(\boldsymbol{X}|\boldsymbol{y})$ and the prior probability $p(\boldsymbol{y})$, normalized by an observation probability $p(\boldsymbol{X})$. The conditional probability $p(\boldsymbol{X}|\boldsymbol{y})$ can be interpreted as an *action model* that explains the video data given that the action sequence is known, while the prior probability $p(\boldsymbol{y})$ is interpreted as a *relation model* that explains the relationship among the actions. Since the observation probability $p(\boldsymbol{X})$ is independent of action sequences, we ignore it when doing inference.

Given that a CTC network is able to learn the nature of each element action in a sequence, the output of the CTC network naturally provides a way to model the term $p(\boldsymbol{X}|\boldsymbol{y})$. However, since the CTC network directly estimates a posterior of an action sequence, i.e. $p(\boldsymbol{y}|\boldsymbol{X})$, we are required to convert it to the form of conditional probability, i.e. $p(\boldsymbol{X}|\boldsymbol{y})$, before plugging it into Equation 11 as the *action model*. Specifically, following the same procedure of the DNN-HMM framework for speech recognition [62, 63], where the posterior probability of the DNN is converted to the conditional probability by scaling the posterior probability using the HMM state priors, we scale the CTC network output by the path label priors, i.e.

$$p(\boldsymbol{X}|\boldsymbol{y}) \propto \prod_{t=1}^{T} \frac{p(\pi^t|\boldsymbol{X})}{p(\pi^t)}, \tag{12}$$

where we apply the best path approximation for $\boldsymbol{y}$, and π^t is the t-th label in the best path of $\boldsymbol{X}$, whose length is T. Here $p(\pi^t|\boldsymbol{X})$ is the soft-max probability directly given by the CTC network and $p(\pi^t)$ is the path label prior, which can be estimated by forcedly aligning training videos and their corresponding action sequences using Viterbi algorithm [64], followed by simply counting, i.e. Maximum Likelihood (ML) estimation. As pointed out in [65], CTC network tends to give very peaky label prediction, with the majority 'background' label dominating the best paths, leading to that the prior estimated by ML is rather skew and not robust. Therefore, we follow the suggestion of [65] to discount the number of 'background' labels and smooth the ML estimation for $p(\pi^t)$.

The *relation model* $p(\boldsymbol{y})$ in Equation 11 is estimated by the statistical language model of actions and the prior about the length of an action sequence:

$$p(\boldsymbol{y}) \propto p_{\mathrm{lm}}^{\alpha}(\boldsymbol{y})N^{\beta}. \tag{13}$$

where $p_{\mathrm{lm}}(\boldsymbol{y})$ is the statistical language model of actions, N is the length of $\boldsymbol{y}$, α and β are two tunable parameters. In addition to the statistical language model for modeling the action relationship, we propose to add the prior about the global length of an action sequence. This should complement the statistical language model, since the statistical language model can only reason about the action relationship by locally looking at several preceding actions at a time, and lacks the knowledge about the global length of an action sequence.

Combining Equation 12, 13 into Equation 11, and taking the logarithm, we can get:

$$\boldsymbol{y}^* = \arg\max_{\boldsymbol{y}} E(\boldsymbol{y}|\boldsymbol{X}), \tag{14}$$

where

$$E(\boldsymbol{y}|\boldsymbol{X}) = \sum_{t=1}^{T} \log \frac{p(\pi^t|\boldsymbol{X})}{p(\pi^t)} + \alpha \log p_{\mathrm{lm}}(\boldsymbol{y}) + \beta \log(N). \tag{15}$$

Hence, for incorporating the information of action relations into the CTC network, our goal is to find the action sequence $\boldsymbol{y}$ that maximizes Equation 15. We refer $\sum_{t=1}^{T} \log \frac{p(\pi^t|\boldsymbol{X})}{p(\pi^t)}$ as *CTC term*, $\alpha \log p_{\mathrm{lm}}(\boldsymbol{y})$ as *language model term* and $\beta \log(N)$ as *length term*.

Inspired by the work in speech recognition [45, 66, 67], we introduce a beam search decoding mechanism to find the optimal $\boldsymbol{y}$ in Equation 15. Algorithm 1 describes the details. Basically, we maintain a set Y_{best}^{t}, which is composed of several action subsequences that are best matching with the partial time sequence of the video up to the t-th time step. Then we proceed to the next time step $t+1$, constructing Y_{best}^{t+1} based on the set Y_{best}^{t}. At time $t+1$, for each action subsequence $\boldsymbol{y}$ that is in Y_{best}^{t}, we either keep it unchanged by making the network to repeatedly predict the last action of it, or extend it by a new action. Given these candidate action sequences matching to the time steps from 1 to $t+1$, we perform a beam pruning to compose Y_{best}^{t+1} with the ones that have maximum values in Equation 15. The above procedure is repeated until we reach to the last frame. The recognition result is then generated by choosing the best action sequence that is in Y_{best}^{T}, where T is the length of the video.

Algorithm 1 CTC Decoding with Beam Search

```
Inputs:
    (1) softmax outputs of the CTC network $s^t(m)$,
        $t = 1...T, m \in V$, where $T$ is the length of the video,
        $V$ is the set of the action types.
    (2) path label prior $p(c)$.
    (3) a language model $p_{\text{lm}}(l|\boldsymbol{h})$, where $l$ is an action
        label and $\boldsymbol{h}$ is the action subsequence preceding it.
Parameters: language model term factor $\alpha$, length term
factor $\beta$ and beam pruning width $w$.
Output: the most probable action sequence $\boldsymbol{y}^*$.
Notations: $Y_{\text{best}}^t$ is the set of action subsequences that are
best matching from time 1 to $t$, $D^t(\boldsymbol{y}|\boldsymbol{X})$ is the value of
Equation 15 excluding the length term for the action
subsequence $\boldsymbol{y}$ up to time $t$.
Procedure:
$Y_{\text{best}}^0 \leftarrow \{\epsilon\}$, $D^t(\epsilon|X) \leftarrow 0$
for $t = 1...T$ do
    $Y_{\text{cand}}^t = \emptyset$
    for each $\boldsymbol{y} \in Y_{\text{best}}^{t-1}$ do
        $c \leftarrow \boldsymbol{y}^{|\boldsymbol{y}|}$          ▷ Get the last action label in $\boldsymbol{y}$.
                               ▷ Collapse the repeated action label.
        $D^t(\boldsymbol{y}|\boldsymbol{X}) \leftarrow \log \frac{s^t(c)}{p(c)} + D^{t-1}(\boldsymbol{y}|\boldsymbol{X})$
        $Y_{\text{cand}}^t \leftarrow Y_{\text{cand}}^t \cup \{\boldsymbol{y}\}$
        for each $l \in V \wedge l \neq c$ do
                               ▷ Extend the action sequence at time $t$.
            $\boldsymbol{y}^+ \leftarrow append(\boldsymbol{y}, l)$
            $D^t(\boldsymbol{y}^+|\boldsymbol{X}) \leftarrow \log \frac{s^t(l)}{p(l)} + \log p_{\text{lm}}(l|\boldsymbol{y}) + D^{t-1}(\boldsymbol{y}|\boldsymbol{X})$
            $Y_{\text{cand}}^t \leftarrow Y_{\text{cand}}^t \cup \{\boldsymbol{y}^+\}$
        end for
    end for
    Add the length term to $D^t(\boldsymbol{y}|\boldsymbol{X})$ for each $\boldsymbol{y}$ in $Y_{\text{cand}}^t$
    and choose the best $w$ ones to constitute $Y_{\text{best}}^t$
end for
return the best $\boldsymbol{y}$ in $Y_{\text{best}}^T$
```

6 EXPERIMENTS

6.1 Dataset

We evaluate our method on the *Breakfast* [13] dataset, which is a large-scaled video dataset that records realistic human actions of preparing breakfast. It consists of 1,712 video clips with totally 66.7 hours. Each clip records a person performing a goal-directed activity, such as *preparing cereal* and *preparing sandwiches*. The activity is further decomposed into several action units in temporal order. In total, there are 48 kinds of action units in the dataset. Two different video clips may respectively contain different action units in different temporal order, i.e. two different action sequences, regardless they belong to the same activity or not. An example action sequence of preparing cereals is (*pour_cereals*, *pour_milk*, *stir_cereals*). Note that we only use the temporal order annotations of the action units in our experiments, though the dataset also provides temporal boundaries between them. We use the training/testing splits of the dataset defined in [13].

Evaluation Metric. We follow the evaluation protocol of [13] to evaluate our method under *unit accuracy*. Specifically, we ignore the leading and trailing 'SIL' units and match the recognized action sequence for a video against its groundtruth using Dynamic Time Warping (DTW). The matching using DTW results in three types of error, namely insertion error I, deletion error D and substitution error S. Then the *unit accuracy* for the video is calculated as:

$$\text{Acc.}(\boldsymbol{y}^*, \boldsymbol{y}) = 1 - \frac{I + D + S}{N}, \qquad (16)$$

where $\boldsymbol{y}^*$ denotes the recognized action sequence, $\boldsymbol{y}$ denotes the groundtruth action sequence and N denotes the length of the groundtruth. The average accuracy over the whole test set is used as the final evaluation.

6.2 Experimental Setup

We down-sample the videos from 15fps to 1.5fps for the sake of efficiency, and extract a Fisher Vector of Improved Dense Trajectory [1] for every frame in a way described in [68]. As a result, each frame is featured with a 64-dim descriptor.

For the RNN module, we employ a one-layer Bi-Directional-LSTM [11] with 256 hidden units, which has two independent chains to go forward and backward respectively along a video sequence for encoding temporal information. We train the RNN with CTC using Rmsprop [69] under the initial learning rate being 0.001. We decay the learning rate by every epoch and clip the gradients within [-0.5, 0.5]. A weight decay is also applied to regularize the network, and the batch size is set to be 128.

For the statistical language model module, we explore the bigram, trigram as well as quadgram language model in our experiments. The language models are built from the action sequences in the training set by using the Srilm toolkits [61]. When combining the statistical language model into the decoding of the CTC network, a width of 7 is set for the beam search. For the factor α and β in Equation 15, we set them using HyperOpt [70]. We smooth the CTC path label prior in Equation 12 by discounting the number of the majority 'background' label by a factor of 4 in the action sequences and increase the counts of actions that are below the average count to the average.

6.3 Exprimental Results and Discussion

6.3.1 Effectiveness of statistical language models for capturing action relationship. We first investigate how good our language model can be in merely explaining action sequences,

Table 1: The perplexity of different n-gram language models in the test set. The "Random" model makes prediction by uniformly selecting any action type.

Model	Perplexity
Random	48.00
Unigram	24.03
Bigram	4.217
Trigram	**3.176**
Quadgram	3.236

by using perplexity. The perplexity of a language model in a corpus of action sequences is defined as:

$$\text{Perp} = \sqrt[N]{\frac{1}{\prod_{\boldsymbol{y}\in C} p(\boldsymbol{y})}}, \tag{17}$$

where C is a corpus, N is the total number of actions in the corpus, $p(\boldsymbol{y})$ is the estimated probability given by the language model. Intuitively, the perplexity describes on average how many independent actions the model has to uniformly choose from, in order to give a correct prediction for an action in an action sequence. Therefore, a lower perplexity indicates a stronger language model.

Table 1 shows the perplexity of our unigram, bigram, trigram and quadgram language models on the test set. We can see that the bigram, trigram, as well as quadgram, substantially outperform the unigram, although the unigram has already been able to give a much stronger prediction, compared with the random model that uniformly selects an action from 48 action types. The effectiveness of the bigram, trigram and quadgram in terms of perplexity suggests the existence of strong relations among the actions, and they are able to well capture the relationship. Usually, a higher order n-gram model is able to give better performance by providing more context information. However, we find that the perplexity of the quadgram is slightly higher than the trigram. We attribute it to the lack of training data, which makes the quadgram cannot be well generalized.

6.3.2 Ablation study of combining a statistical language model for action sequence recognition. To investigate how effective our statistical language model components are in the decoding of the CTC network for action sequence recognition, we conduct an ablation study with the *unit accuracy* results shown in Table 2.

The first baseline (CTC) is merely a CTC network, which cannot leverage any knowledge except the one learned by itself. For the second baseline (CTC+Len), we only include the *length term* along with the *CTC term* in Equation 15, which allows us to add a prior about the length of action sequences when decoding the CTC output. Empirically, we find a CTC network tends to be overwhelmed by the 'background' label in the emission of every time step. This results in relatively short predicted sequences when collapsing the repeated 'background' labels. We attribute this bias to the overwhelming number of the 'background' class during training, and the introduction of the *length term* can help to alleviate the impact, leading to approximately 1% point improvement.

We also investigate the effect of the *language model term* in Equation 15, by excluding the *length term.* Row 3 to Row 5 in Table 2 show the results when the *language model term* is given by a bigram, trigram and quadgram respectively. One can see that by incorporating a statistical language model it helps to boost the performance significantly, supporting our opinion that the action relations captured by a statistical language model benefit the recognition of the CTC network. Among them, the trigram model gives the best result. This is consistent with the perplexity performance shown in Table 1, which suggests a trigram captures the action relationship best, and by nature provides more reliable information for the CTC network.

Finally, we show the performance of the complete model in the last three rows in Table 2. The combination of the *length term* and the *language model term* gives even larger improvement than the sum of the improvement given by each single component. For example, CTC+Bigram+Len gives 4.5% points improvement while the sum of the improvement given by a single CTC+Len and a single CTC+Bigram is 3.2% points. This suggests the *length term* and the *language model term* can interact well, supporting our idea that the *length term* is complementary to the statistical language model, in the sense that the statistical language model lacks the knowledge about the global length of an action sequence. Our proposed CTC+Trigram+Len achieves the best result (0.434).

In Figure 2, we show several qualitative results comparing our proposed CTC+Trigram+Len and the CTC baseline. From the results, we can see that our method is able to correct the errors of the CTC network that do not conform the general human action orders. For instance, *takePlate* is corrected to *pourDough2pan* in the first example, since *pourDough2pan* is more likely to happen given the preceding actions being *pourMilk* and *stirDough.* So is the third example correcting *pourMilk* to *pourWater*. Moreover, our method helps to discover some actions that are not well recognized by the CTC network, e.g. *putPancake2plate* at the end of the sequence in the first example, *addSaltnpepper* in the second example and *takeGlass* in the forth example. However, we still find it challenge to recognize some very short actions that do not have strong dependency on its context, e.g. *butterPan* in the second example and *takePlate*, *takeKnife* in the forth example.

6.3.3 Comparison with the state-of-the-art. In Table 3, we compare our CTC+Trigram+Len method with three state-of-the-art methods, namely Ordered Constrained Discriminative Clustering (OCDC) [43], Hidden Markov Model equipped with a network-based Grammar (HMM+Grammar) [6] and Extended Connectionist Temporal Classification (ECTC), under the *unit accuracy* metric. All of them are capable of learning models from unsegmented videos and perform action sequence recognition. For OCDC and ECTC methods, we show their performances that are reported in [5]. Since Kuehne et al. [6] only reported the frame-wise segmentation accuracy for HMM+Grammar, we build the HMM+Grammar

Table 2: Ablation study of our proposed method under *unit accuracy* on *Breakfast* dataset. CTC refers to the CTC baseline. Len refers to including the *length term* in Eq. 15 for CTC decoding. Bigram, Trigram and Quadgram refer to including the *language model term* in Eq. 15 and it is estimated by a bigram, trigram and quadgram respectively.

Model	Unit Acc.
CTC	0.370
CTC+Len	0.381
CTC+Bigram	0.391
CTC+Trigram	0.415
CTC+Quadgram	0.412
CTC+Bigram+Len	0.415
CTC+Trigram+Len	**0.434**
CTC+Quadgram+Len	0.429

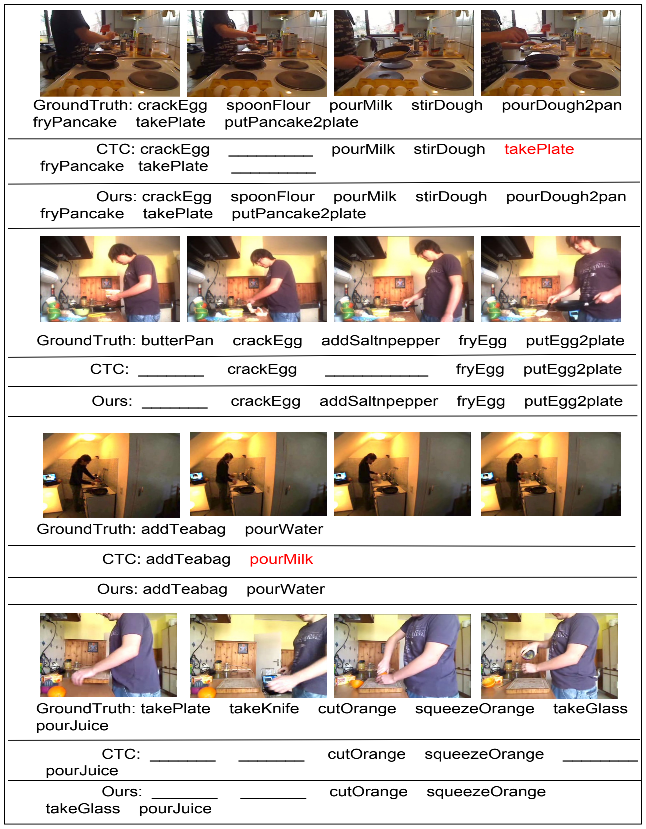
GroundTruth: crackEgg spoonFlour pourMilk stirDough pourDough2pan fryPancake takePlate putPancake2plate
CTC: crackEgg ________ pourMilk stirDough takePlate fryPancake takePlate ________
Ours: crackEgg spoonFlour pourMilk stirDough pourDough2pan fryPancake takePlate putPancake2plate
GroundTruth: butterPan crackEgg addSaltnpepper fryEgg putEgg2plate
CTC: _______ crackEgg ___________ fryEgg putEgg2plate
Ours: _______ crackEgg addSaltnpepper fryEgg putEgg2plate
GroundTruth: addTeabag pourWater
CTC: addTeabag pourMilk
Ours: addTeabag pourWater
GroundTruth: takePlate takeKnife cutOrange squeezeOrange takeGlass pourJuice
CTC: _______ _______ cutOrange squeezeOrange ________ pourJuice
Ours: _______ _______ cutOrange squeezeOrange takeGlass pourJuice

Figure 2: Qualitative results on *Breakfast* dataset. Our method (CTC+Trigram+Len) is compared with the CTC baseline. The underscores denote deletion error and the red color on the action labels denotes substitution error.

baseline and report its *unit accuracy*. Note that the frame-wise segmentation accuracy of our HMM+Grammar baseline is comparable to the one reported in [6]. The comparison is fair since all the three methods share the same feature extraction module [68] as ours.

Table 3: The *unit accuracy* of our CTC+Trigram+Len method, compared with the state of the art on *Breakfast* dataset.

Model	Unit Acc.
OCDC [43]	0.104
HMM+Grammar [6]	0.207
ECTC [5]	0.367
CTC+Trigram+Len	**0.434**

From the results, we can see that our method substantially outperforms all of the three state-of-the-art methods, surpassing the best one among them by 6.7% points. OCDC utilizes temporal order constraint to train a linear classifier that classifies each frame independently, thus being not able to make use of the temporal structure of videos. Compared with OCDC, our method is built upon Bi-Directional-LSTM, which has the power to capture temporal dynamics of video frames, and achieves a much better result (0.434 Vs. 0.104). Our proposed method also exceeds HMM+Grammar method, which can seizures the temporal information by an HMM, and has the ability to model the relationship of actions by a network-based grammar. Our method has a better performance than the ECTC method as well, which adopts the CTC framework as we do. Note that our CTC baseline has already been slightly better than ECTC (0.370 Vs. 0.367. The CTC baseline performance reported in [5] is 0.363). We ascribe it to our better engineering. Adding a trigram language model and a length prior to the CTC network brings us a boost, making our method surpass ECTC by 6.7 points.

7 CONCLUSION

In this paper, we introduce to equip a CTC network with a statistical language model for action sequence recognition. The whole framework does not rely on any segmentation of human actions in videos to learn the model. We demonstrate the effectiveness of our method by experimenting on the realistic *Breakfast* dataset. The experimental results show that our method gives a good performance and outperforms the current state of the art.

8 ACKNOWLEDGMENT

This work was supported by JST CREST Grant Number JPMJCR1687, Japan.

REFERENCES

[1] H. Wang and C. Schmid. Action recognition with improved trajectories. In *Proc. ICCV*. pp.3551-3558. 2013.
[2] C. Gan et al. Devnet: A deep event network for multimedia event detection and evidence recounting. In *Proc. CVPR*. pp.2568-2577. 2015.
[3] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Proc. NIPS*. pp.568-576. 2014.
[4] D. Tran et al. Learning spatiotemporal features with 3d convolutional networks. In *Proc. ICCV*. pp.4489-4497. 2015.
[5] D.A. Huang et al. Connectionist temporal modeling for weakly supervised action labeling. In *Proc. ECCV*. pp.137-153. 2016.
[6] H. Kuehne et al. Weakly supervised learning of actions from transcripts. In *CVIU*. 2017.

[7] P. Molchanov et al. Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network. In *Proc. CVPR*. pp.4207-4215. 2016.

[8] A. Graves et al. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proc. ICML*. pp.369-376. 2006.

[9] S. Hochreiter and J. Schmidhuber. Long short-term memory. In *Neural Computation*. Vol.9, No.8, pp.1735-1780. 1997.

[10] B. Singh et al. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In *Proc. CVPR*. pp.1961-1970. 2016.

[11] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*. Vol.18, No.5, pp.602-610. 2005.

[12] P. Pan et al. Hierarchical recurrent neural encoder for video representation with application to captioning. In *Proc. CVPR*. pp.1029-1038. 2016.

[13] H. Kuehne et al. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proc. CVPR*. pp.780-787. 2014.

[14] D. Bahdanau et al. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*. 2015.

[15] L. Yao et al. Describing videos by exploiting temporal structure. In *Proc. ICCV*. pp.4507-4515. 2015.

[16] I. Laptev. On space-time interest points. In *IJCV* Vol.64, No.2-3, pp.107-123. 2005.

[17] C. Feichtenhofer et al. Convolutional two-stream network fusion for video action recognition. In *Proc. CVPR*. pp.1933-1941. 2016.

[18] P. Scovanner et al. A 3-dimensional sift descriptor and its application to action recognition. In *Proc. ACM MM*. pp.357-360. 2007.

[19] A. Klaser et al. A spatio-temporal descriptor based on 3d-gradients. In *Proc. BMVC*. 2008.

[20] G. Willems et al. An efficient dense and scale-invariant spatio-temporal interest point detector. In *Proc. ECCV*. pp.650-663. 2008.

[21] H. Kuehne et al. HMDB: a large video database for human motion recognition. In *Proc. ICCV*. pp.2556-2563. 2011.

[22] K. Soomro et al. UCF101: A dataset of 101 human actions classes from videos in the wild. In *Technical Report: CRCV-TR-12-01*. 2012.

[23] J. Sanchez et al. Image classification with the fisher vector: Theory and practice. In *IJCV* Vol.105, No.3, pp.222-245. 2013.

[24] R. Arandjelovic and A. Zisserman. All about VLAD. In *Proc. CVPR*. pp.1578-1585. 2013.

[25] A. Krizhevsky et al. Imagenet classification with deep convolutional neural networks. In *Proc. NIPS*. pp.1097-1105. 2012.

[26] C. Szegedy et al. Going deeper with convolutions. In *Proc. CVPR*. pp.1-9. 2015.

[27] J. Deng et al. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*. pp.248-255. 2009.

[28] S. Ji et al. 3D convolutional neural networks for human action recognition. In *PAMI*. Vol.35, No.1, pp.221-231. 2013.

[29] Q. Shi et al. Discriminative human action segmentation and recognition using semi-Markov model. In *Proc. CVPR*. pp.1-8. 2008.

[30] C.C. Chen and J.K. Aggarwal. Modeling human activities as speech. In *Proc. CVPR*. pp.3425-3432. 2011.

[31] A. Gaidon et al. Temporal localization of actions with actoms. In *PAMI*. Vol.35, No.11, pp.2782-2795. 2013.

[32] C. Lea et al. Temporal convolutional networks: A unified approach to action segmentation. In *Proc. ECCV Workshops*. pp.47-54. 2016.

[33] C. Lea et al. Segmental spatiotemporal CNNs for fine-grained action segmentation. In *Proc. ECCV*. pp.36-52. 2016.

[34] Z. Shou et al. CDC: Convolutional-De-Convolutional networks for precise temporal action localization in untrimmed videos. In *Proc. CVPR*. 2017.

[35] M. Baccouche et al. Sequential deep learning for human action recognition. In *Proc. HBU*. pp.29-39. 2011.

[36] J. Donahue et al. Long-term recurrent convolutional networks for visual recognition and description.In *Proc. CVPR*. pp.2625-2634. 2015.

[37] V. Veeriah et al. Differential recurrent neural networks for action recognition. In *Proc. ICCV*. pp.4041-4049. 2015.

[38] Q. Li et al. Action recognition by learning deep multi-granular spatio-temporal video representation. In *Proc. ICMR*. pp.159-166. 2016.

[39] H. Pirsiavash and D. Ramanan. Parsing videos of actions with segmental grammars. In *Proc. CVPR*. pp.612-619. 2014.

[40] N.N. Vo and A.F. Bobick. Sequential interval network for parsing complex structured activity. In *CVIU* Vol.143, pp.147-158. 2016.

[41] A. Richard and J. Gall. Temporal action detection using a statistical language model. In *Proc. CVPR*. pp.3131-3140. 2016.

[42] L. Wang et al. UntrimmedNets for weakly supervised action recognition and detection. In *Proc. CVPR*. 2017.

[43] P. Bojanowski et al. Weakly supervised action labeling in videos under ordering constraints. In *Proc. ECCV*. pp.628-643. 2014.

[44] J.B. Alayrac et al. Unsupervised learning from narrated instruction videos. In *Proc. CVPR*. pp.4575-4583. 2016.

[45] D. Amodei et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *Proc. ICML*. 2016.

[46] T. Bluche et al. Framewise and CTC training of neural networks for handwriting recognition. In *Proc. ICDAR*. 2015.

[47] M.T. Luong et al. Deep neural language models for machine translation. In *Proc. CoNLL*. pp.305-309. 2015.

[48] D. Soutner et al. On a hybrid NN/HMM speech recognition system with a RNN-based language model. In *Proc. SPECOM*. pp.315-321. 2014.

[49] F. Song and W.B. Croft. A general language model for information retrieval. In *Proc. CIKM*. pp.316-321. 1999.

[50] G.J. Lidstone. Note on the general case of the Bayes-Laplace formula for inductive or a posteriori probabilities. In *Trans. of the Faculty of Actuaries*. pp.182-192. 1920.

[51] F. Jelinek and R. Mercer. Interpolated estimation of Markov source parameters from sparse data. In *Proc. of Workshop on Pattern Recognition in Practice*. pp.381-397. 1980.

[52] S. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. In *ASSP*. Vol.35, No.3, pp.400-401. 1987.

[53] I.J. Good. The population frequencies of species and the estimation of population parameters. In *Biometrika*. Vol.40, No.3/4, pp.237-264. 1953.

[54] H. Ney et al. On the estimation of small probabilities by leaving-one-out. In *PAMI*. pp.1202-1212. 1995.

[55] T.C. Bell et al. Text compression. Prentice-Hall, Inc. 1990.

[56] D. Guthrie et al. A closer look at skip-gram modelling. In *Proc. LREC*. pp.1-4. 2006.

[57] P.F. Brown et al. Class-based n-gram models of natural language. In *Computational Linguistics*. Vol.18, No.4, pp.467-479. 1992.

[58] D. Gildea and T. Hofmann. Topic-based language models using EM. In *Proc. EUROSPEECH*. pp.2167-2170. 1999.

[59] T. Mikolov et al. Recurrent neural network based language model. In *Proc. Interspeech*. pp.1045-1048. 2010.

[60] S.F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *Proc. ACL*. pp.310-318. 1996.

[61] A. Stolcke et al. SRILM-an extensible language modeling toolkit. In *Proc. Interspeech*. pp.901-904. 2002.

[62] Y. Miao, and F. Metze. Improving low-resource CD-DNN-HMM using dropout and multilingual DNN training. In *Proc. Interspeech*. pp.2237-2241. 2013.

[63] J. Pan et al. Investigation of deep neural networks (DNN) for large vocabulary continuous speech recognition: Why DNN surpasses GMMS in acoustic modeling. In *Proc. ISCSLP*. pp.301-305. 2012.

[64] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proc. of the IEEE*. Vol.77, No.2, pp.257-286. 1989.

[65] Y. Miao et al. EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. In *Proc. ASRU*. pp.167-174. 2015.

[66] A.Y. Hannun et al. First-pass large vocabulary continuous speech recognition using bi-directional recurrent DNNs. *arXiv preprint arXiv:1408.2873*. 2014.

[67] A.L. Maas et al. Lexicon-Free conversational speech recognition with neural networks. In *Proc. NAACL*. pp.345-354. 2015.

[68] H. Kuehne et al. An end-to-end generative framework for video segmentation and recognition. In *Proc. WACV*. 2016.

[69] T. Tieleman and G.E. Hinton. Lecture 6.5RmsProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*. 2012.

[70] J. Bergstra et al. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Proc. SciPy*. pp.13-20. 2013.