Multi-Resolution Convolutional Recurrent Networks

Jen-Tzung Chien and Yu-Min Huang
Department of Electrical and Computer Engineering, National Yang Ming Chiao Tung University, Hsinchu, Taiwan

***Abstract*—In sequential learning tasks, recurrent neural network (RNN) has been successfully developed for many years. RNN has achieved a great success in a variety of applications in presence of audio, video, speech and text data. On the other hand, temporal convolutional network (TCN) has recently drawn high attention in different works. TCN basically achieves comparable performance with RNN, but attractively TCN could work more efficient than RNN due to the parallel computation of one-dimensional convolution. A fundamental issue in sequential learning is to capture the temporal dependencies with different time scales. In this paper, we present a new sequential learning machine called the multi-resolution convolutional recurrent network (MR-CRN), which is a hybrid model of TCN encoder and RNN decoder. Utilizing the representation learned by TCN encoder in different layers with various temporal resolutions, RNN decoder can summarize the contextual information with different resolutions and time scales without modifying the original architecture. In the experiments on language modeling and action recognition, the merit of MR-CRN is illustrated for sequential learning and prediction in terms of latent representation, model perplexity and recognition accuracy.**

I. Introduction

Deep machine learning has been achieving a great success in many real-world applications. With these successful applications, we are able to extend the applications to other unknown tasks and accomplish the desirable performance by deep neural networks (DNNs) through collecting sufficient training data and optimizing a specialized objective function with the well-defined observation inputs and target outputs. Among different learning machines, sequential learning is one of the most popular and influential mechanisms which are developed to characterize the temporal dependencies and patterns from sequence data. The sequence data, e.g. natural sentences, audio and speech signals and video streams are everywhere in our daily life. Deep learning models have been extensively constructed to represent the underlying temporal relationship between sequence samples and their corresponding target outputs. Basically, the temporal dependencies in the mapping between inputs and outputs are complicated. It is especially difficult to capture long-term dependencies in long sequence data. Accordingly, a fundamental issue in sequential learning is to identify the sequential patterns with different lengths. Many researchers have been dedicating to study and overcome these challenges and difficulties.

Recurrent neural network (RNN) [1], [2], [3], [4], [5] have been developed to characterize sequential patterns for many years. RNNs dynamically calculate the hidden states and propagate them to the next time steps. This sequential machine summarizes the history information from the past inputs $\{\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}\}$ and distills the information to predict the next sequential output $\mathbf{y}_t$ conditioned on the current input $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$. Using RNNs, the hidden state $\mathbf{h}_t$ is continuously updated by a series of linear and nonlinear transformations which provide an avenue to characterize the complicated probability distribution over sequential data. RNN based models are fitted to sequential learning. Some variants of sequential learning machines, like long short term memory (LSTM) [2] and gated recurrent unit (GRU) [6] and transformer [7], have been proposed to improve the robustness of performance via the gating and attention mechanisms. In particular, the gating mechanism makes the temporal dependencies sufficiently preserved in hidden states compared with the standard RNN. Recurrent networks with gating mechanism are seen as a kind of mainstream method in deep learning.

Despite of the success of RNNs, convolutional neural network (CNN) [8] has emerged as a rising approach to handle learning representation of sequence data. CNN has been popular in the areas of computer vision due to its powerfulness of capturing the local information especially in spatial data. Recently, CNN has been developed to capture local information in temporal data. When the one-dimensional convolution operation is applied in time domain, the resulting model, called the temporal convolutional networks (TCN) [9], [10], has been exploited. Further, the tricks of dilation, residual connection, and causality were imposed to TCNs to improve sequential learning. Compared with RNN, TCN takes the advantage on the parallelism in computation. The inference using TCN is much faster than that using RNN. Meanwhile, the performance of TCN is also comparable to RNN so that TCN related works are now drawing more and more attention in the related areas in recent years.

No matter how the learning representation is based on TCNs or RNNs, it is important to capture long-term as well as short-term dependencies in sequence data. Basically, RNNs with gating mechanism can capture long-term dependency better than a vanilla RNN. But, RNNs are still suffering from the problem of gradient vanishing. As a result, many researches are devoted to deal with this issue. On the other hand, the temporal hierarchy using TCNs is meaningful and attractive. The context with different sizes of receptive fields can be learned in different layers of TCNs in a bottom-up manner. Local information is utilized sufficiently. However, the temporal patterns with infinite length cannot be handled by TCNs. This paper presents a new model, called the multi-resolution convolutional recurrent network, which would like to boost the strength of TCNs and RNNs. We use TCNs as local feature extractor to encode the information with multiple

time scales, and then feed this information to RNNs where local and global features are learned. This model relaxes the loading of RNNs by allowing TCNs share the responsibility to learn local features. With the preprocessing done by TCN, each RNN can focus on the modeling of temporal dependency from context with a certain scale.

II. Sequential Learning

A. Background and motivation

In sequential learning, it is challenging to preserve long-term dependencies without forgetting short-term memories in a sequence of observations. An effective approach to tackle this challenge is to implement the recurrent networks by merging with temporal information from multiple time scales. The related works include the clockwork RNNs [11], phased LSTMs [12], hierarchical multi-scale RNNs [13]. In [14], the dilated RNNs were proposed to characterize the temporal patterns with multiple resolutions by means of the dilation in recurrent connections. Dilation here means the skipping of inputs with a certain step size. Dilation is an important scheme which was first proposed for CNNs in [15]. Due to dilated connections, the receptive fields in the representation can be expanded exponentially by increasing the depth of a deep model. Such a dilation was not only employed in RNNs [14] but also in temporal convolutional networks (TCNs) [9], [10]. In [16], [17], the long and short-term time-series network (LSTNet) and the hybrid CNN, LSTM and DNN (called CLDNN) were proposed to strengthen deep models by combining different model structures. Both LSTNet and CLDNN adopted CNNs and RNNs to represent the multivariate time series. However, two-dimensional or multi-dimensional convolution in CNNs was not clearly fitted to sequential learning when compared with one-dimensional convolution in TCNs. This paper presents a hybrid network architecture by capturing the multi-scale contextual information in TCN and simultaneously aggregating the long-term information via multiple RNNs. Since TCNs are good at extracting local features and RNNs are fitted to forecast time series, the proposed hybrid model is comprehensive and reasonable. Moreover, due to the dilated connection in TCNs, there is no need to apply skip connection for RNNs like LSTNet. The multivariate sequence data are directly characterized without modifying the architecture of RNNs. Furthermore, the proposed method waives the need of skipping the state updating like skip RNN [18]. The detailed computation model is addressed in what follows.

B. Convolutional recurrent networks

Basically, TCNs represent the sequence data based on one-dimensional convolution with dilation while the receptive field of a time sequence is exponentially expanded by increasing the depth of neural network. Because of the stacked dilation layers, there is a temporal hierarchy where the representation learned in the upper layers contains the larger span of context from sub-sequences. This representation is strictly increasing with the height of a deep neural structure. Since we would like to capture temporal dependency with various time scales, this property is attractive to accomplish this goal. However, the size of receptive field is still bounded by the number of layers in TCN. Meanwhile, recurrent neural networks (RNNs) are good at temporal modeling, but may suffer from gradient vanishing. RNNs with gating mechanism [2], [6], [19] generally capture the temporal patterns without the limitation in the length of sequential modeling. As a result, it is natural to introduce an RNN as the top layer to relax the limitation in TCN. The temporal hierarchy is extended. The so-called convolutional recurrent network (CRN) [20], [21] is therefore proposed.

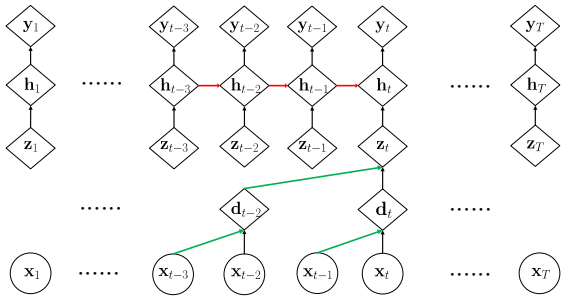

Fig. 1: Graphical model of convolutional recurrent network.

The graphical representation of CRN is shown in Figure 1 which is seen as a concatenation of TCN and RNN. In this hybrid model, TCN basically calculate one-dimensional convolution with dilation to find $\mathbf{d}_t^l$ at time-step t and layer l which summarizes the input sequence $\mathbf{x}_{1:t}$ as follows:

$$\mathbf{d}_t^l = \mathrm{Conv}^{(l)}\left(\mathbf{d}_t^{l-1}, \mathbf{d}_{t-j}^{l-1}\right) \tag{1}$$

where $j = 2^{l-1}$ and $\mathbf{d}_t^0 = \mathbf{x}_t$ denote the features of first layer for every time steps t. The information is propagated in a bottom-up direction. Different layer of convolution $\mathrm{Conv}^{(l)}(\cdot)$ adopts individual parameters. The upper layers summarize the context with larger span of time scales. After the TCN calculation with L layers, the outputs of TCN $\mathbf{d}_{1:T}^L$ are then used as inputs to feed into RNN. This RNN is able to transmit the information horizontally along the whole time horizon. Here, LSTM is used to implement this RNN by

$$\begin{aligned}
\mathbf{i}_t &= \sigma\left(\mathbf{W}_i\mathbf{d}_t^l + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i\right)\\
\mathbf{f}_t &= \sigma\left(\mathbf{W}_f\mathbf{d}_t^l + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f\right)\\
\mathbf{o}_t &= \sigma\left(\mathbf{W}_o\mathbf{d}_t^l + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o\right)\\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \sigma_c\left(\mathbf{W}_c\mathbf{d}_t^l + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c\right)\\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh\left(\mathbf{c}_t\right)
\end{aligned} \tag{2}$$

where $\mathbf{h}_t$ denotes the recurrent state, $\odot$ denotes the element-wise product, $\sigma(\cdot)$ denotes the sigmoid function, $\{\mathbf{W}_i, \mathbf{U}_i, \mathbf{b}_i\}$, $\{\mathbf{W}_f, \mathbf{U}_f, \mathbf{b}_f\}$, $\{\mathbf{W}_o, \mathbf{U}_o, \mathbf{b}_o\}$ and $\{\mathbf{W}_c, \mathbf{U}_c, \mathbf{b}_c\}$ denote the parameters from the input of TCN $\mathbf{d}_t^l$ to input gate, forget gate, output gate and cell, respectively. Basically, using this hybrid model, TCN acts as a local context feature extractor or encoder while RNN is viewed as a decoder with global view. In this two-stage model, TCN embeds the local information first, which simplifies the task for RNN

decoding. One-dimensional convolution in TCN is to capture local features while RNN is good at temporal information representation. Therefore, such a mixture of RNN and TCN is beneficial to acquire short-term and long-term information for prediction. This model also extends the temporal hierarchy, which releases the limitation and increases the capability of capturing structural temporal dependency. The dilation in TCN allows efficient computation in the implementation.

III. Extended Studies

A. Generalization and interpretation

Convolutional recurrent network (CRN) is a generalization of both TCN and RNN where a kind of encoder-decoder network is configured for sequential learning. This CRN is simplified as TCN in case that the decoder doesn't depend on the previous state $\mathbf{h}_{t-1}$. Accordingly, in this case, the sequential machine only captures the temporal dependencies within a limited length because the temporal information is not propagated in LSTM. This machine partially forgets past information. On the other hand, in case that the TCN encoder has the specialized wights such that the condition of aligning TCN features in different layers

$$\mathbf{d}_t^l = \mathrm{Conv}^{(l)}\left(\mathbf{d}_t^{l-1}, \mathbf{d}_{t-j}^{l-1}\right) = \mathbf{d}_t^{l-1} \tag{3}$$

is met. This learning machine is then realized as LSTM which implies that the convolution layers only propagate the inputs to the upper layers. This simplification weakens model capability because the local dependencies of sequence data via convolution calculation are missing. As illustrated in Figure 1, if the green computational path is suspended, this CRN will become an RNN. And if the red computational path is stopped, this CRN will realize as TCN. The general model using CRN considerably extends the temporal hierarchy using TCN. The incorporation of RNN as upper layers in CRN sufficiently captures longer temporal dependencies in sequence data with larger time scales or wider receptive fields. CRN is viewed as an extension of temporal hierarchy in TCN by merging with a recurrent machine using LSTM.

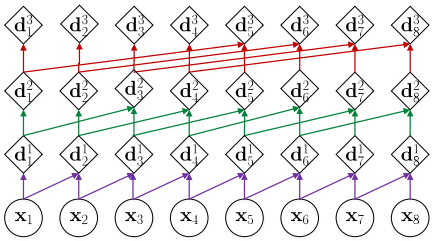

Fig. 2: Encoder in multi-resolution convolutional recurrent network.

B. Multi-resolution convolutional recurrent networks

Convolutional recurrent network is configured as a hybrid network structure which integrates the merits of TCN and RNN by using TCN as an encoder and RNN as a decoder.

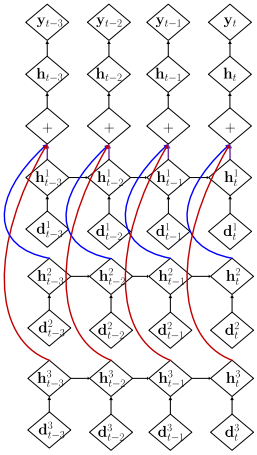

Fig. 3: Decoder in multi-resolution convolutional recurrent network.

However, we challenge that this architecture may not completely acquire and utilize the highest benefit from the combination of TCN and RNN. This paper further extends CRN by strengthening its capability via a so-called multi-resolution convolutional recurrent network (MR-CRN). Figures 2 and 3 illustrate the graphical representation of the hierarchies of encoder and decoder of MR-CRN, respectively. The examples of three-layer encoder and decoder in multi-resolution CRN with $L = 3$ are shown. Encoder is seen as a standard TCN with three layers for calculation of outputs $\mathbf{d}_{1:T}^3$ given input sequence $\mathbf{x}_{1:T}$. Dilation is applied in different layers. The first, second and third layers of TCN span the receptive fields with sizes 2, 4 and 8, respectively. Local information in a short subsequence is learned first and then gradually propagates to upper layers where the longer temporal dependencies are represented. The information with different sizes of receptive fields in encoder is seen as the multi-resolution information or multi-time-scale temporal dependency which is adopted in decoder for MR-CRN.

The main difference between CRNs without and with multi-resolution lies on the decoder. Generally, the local features $\mathbf{d}_{1:T}^l$ learned in different layers l of TCN are provided as the inputs to a deep recurrent neural network with L layers by calculating the hidden states in different layers

$$\mathbf{h}_{1:T}^l = \mathrm{RNN}^{(l)}\left(\mathbf{d}_{1:T}^l\right), \qquad \text{for} \quad l = 1, \ldots, L \tag{4}$$

where $\mathrm{RNN}^{(l)}(\cdot)$ denotes the individual parameters of RNN in different layers l. The temporal structure in different layers of TCN captures different resolutions of local features in an observation sequence. The RNN $\mathrm{RNN}^{(l)}(\cdot)$ summarizes these local features as a set of global features. After this calculation, different hidden states $\mathbf{h}_{1:T}^l$ in L layers of RNNs are added to produce a single sequence of hidden states $\mathbf{h}_{1:T}$ with an integration of multi-level resolutions in a form of

$$\mathbf{h}_t = \sum_{l=1}^{L} \mathbf{h}_t^l, \qquad \text{for} \quad t = 1, \ldots, T \tag{5}$$

To construct the supervised classification network, this single state sequence $\mathbf{h}_{1:T}$ is transformed to find the posterior probabilities corresponding to K different classes $\widehat{\mathbf{y}}_{1:T}$ which are matched with the one-hot target sequences $\mathbf{y}_{1:T}$ by minimizing the cross-entropy error function

$$\mathcal{L}(\mathbf{y}, \widehat{\mathbf{y}}) = -\sum_{t=1}^{T}\sum_{k=1}^{K} y_{tk} \log(\widehat{y}_{tk}). \quad (6)$$

Stochastic gradient descent algorithm [22], [23] is implemented to estimate the parameters of L layers in a backward manner from RNN to TCN through the error backpropagation procedure. Basically, using CRN, the representation of the last layer in TCN is encoded and fed into an RNN decoder. However, the individual hidden features $\mathbf{d}^l_{1:T}$ in different layers l reflect temporal dependencies or embeddings with different lengths and different levels of resolutions. The context in lower layers encode more local information. MR-CRN utilizes these valuable embeddings as different degrees or resolutions of local information, and then calculates the corresponding global features using RNN decoders in individual layers. The disentangled global features with different temporal resolutions are calculated by layer-dependent RNNs and integrated to estimate the sophisticated values of class posteriors $\widehat{\mathbf{y}} = \{\widehat{y}_{tk}\}$ to match with one-hot class targets $\mathbf{y} = \{y_{tk}\}$ for classification. MR-CRN captures richer information than the dilated RNN [14] where structural local features and multi-level resolutions were missing. From an alternative perspective, this multi-resolution CRN is seen as a composition of multiple CRNs in various layers which naturally learn multi-resolutions in an integrated network. The proposed MR-CRN is investigated for sequential learning for word prediction and action recognition.

IV. Experiments

This study conducts two sets of experiments which would like to investigate the effect of various temporal information in different domain data.

A. Evaluation for language modeling

Penn Treebank (PTB) [24] dataset was used to evaluate the performance of sequential learning using different models. PTB was widely used in natural language processing for evaluation of word prediction. There were 10K words in the dictionary. The sentences for training, validation and test consisted of 929K, 73K, 82K words, respectively. The capital letters, numbers and punctuations were removed in text preprocessing. Each input sentence was trimmed to length of 20 words in order to learn long sequence in an efficient way. This dataset was used to examine word-level prediction for language model. Latent spaces using different methods were analyzed. Perplexity was measured to investigate how well a probability distribution or language model, normalized by the length of sentence, predicts the future words. Lower perplexity generally implies better performance in word prediction. For comparison, LSTM (here denoted as RNN) [2], TCN [9], [10], stochastic TCN (STCN) [25] and the proposed CRN [20], [21] and MR-CRN with $L = 3$ were implemented by running twenty epochs using SGD algorithm. For comparative study, the recurrent convolutional network (denoted as RCN) was carried out as RNN encoder and TCN decoder. The recurrent recurrent network (denoted as RRN) implemented a 2-layer RNN. The mini-batch size was twenty. Gradient clipping [26] was applied to mitigate the gradient vanishing in learning procedure. All parameters were uniformly initialized between -1 and 1. Recurrent dropout was used in each layer's outputs with a dropout rate 0.5 [27]. The size of hidden states and the amount of kernels were adjusted in different models to obtain desirable performance using validation data. The size of hidden states was 450 for all models using LSTM. Model size was included in the evaluation.

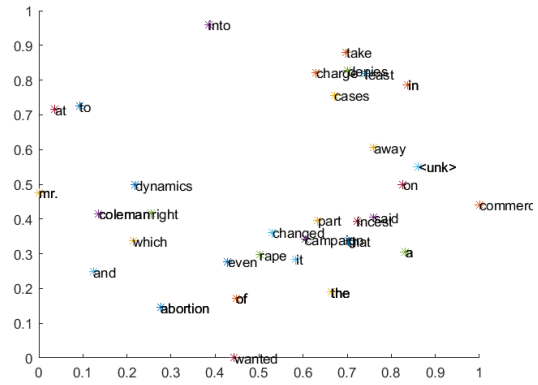

Fig. 4: Latent space of word embeddings in RNN.

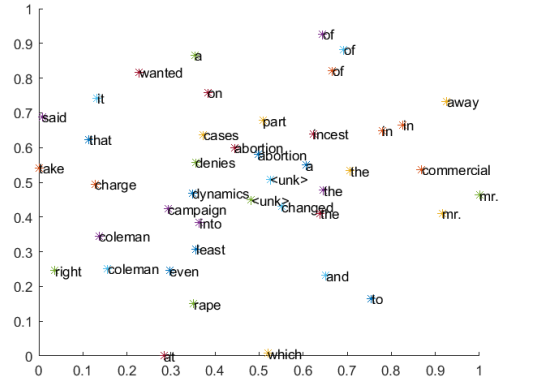

Fig. 5: Latent space of word embeddings in RRN.

For illustration, we investigate a sentence from PTB which is "that commercial which said mr. coleman wanted to take away the right of abortion even in cases of rape and incest a charge mr. coleman denies changed the dynamics of the campaign ⟨unk⟩ it at least in part into a ⟨unk⟩ on abortion" where ⟨unk⟩ means an unknown word. RNN, RRN and CRN are compared in Figures 4, 5 and 6 where the standard word embeddings, RNN word embedding and TCN word embed-

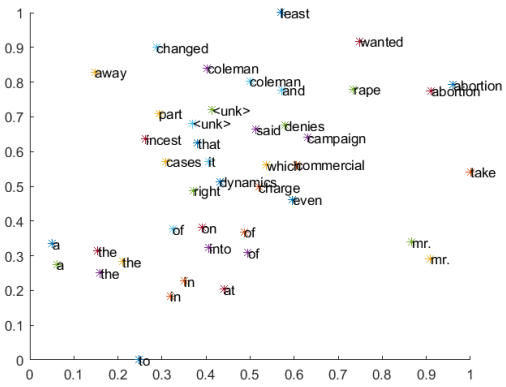

Fig. 6: Latent space of word embeddings in CRN.

ding are visualized after the dimension reduction using the t-distributed stochastic neighbor embedding (t-SNE), respectively [28]. Typically, each word is projected to a fixed vector. The words appear more than once in the sentence will only show one time in RNN where standard word embeddings are examined. Standard word embeddings ignore local information because RNN just focus on single words. The words are not clearly separated in latent space using RNN. However, the word embeddings in RRN and CRN are different from those in RNN. Depending on the context using RRN and CRN, a word can be projected to similar but different vectors in RNN and TCN word embeddings, e.g. the words 'the', 'of', 'mr.', 'coleman', etc. Interestingly, the proposed CRN uses the embeddings of 'which' and 'commercial' which are very closely. This implies that CRN can understand the relation between these two words where the word 'which' connects to the word 'commercial' in this sentence. This is a kind of local information where TCN can capture. On the other hand, RNN and RRN cannot recognize such a word relation. Therefore, local information via TCN encoding in the first stage is useful. RNN encoder may partially capture local information, but is not as good as TCN encoder. In this comparison, the word embeddings in CRN perform well and reflect local information for RNN decoder which is helpful for language modeling.

TABLE I: Perplexity (in three phases) and model size using PTB.

Model	#Params	Train	Validation	Test
RNN	8.8M	101	128	123
RCN	9.5M	125	129	125
RRN	10.4M	90	126	122
TCN [9], [10]	7.0M	100	138	131
STCN [25]	13.1M	97	125	120
CRN	11.1M	79	119	116
MR-CRN	18.5M	59	108	**98**

Table I compares the model size and the perplexity in training, validation and test phases by using different models including RCN, RRN, RNN, TCN, STCN, CRN and MR-CRN. The best result is shown in bold. RRN obtains lower perplexities than RCN. The performance of RCN is degraded and even performs worse than RNN. This result confirms the selection of model architecture with the order of TCN encoder and RNN decoder. CRN outperforms RNN, RCN and RRN in terms of perplexity. The strengths of TCN and RNN are complementary. The order in configuration of cascading TCN and RNN is important. CRN boosts the strengths of TCN and RNN. CRN captures the temporal dependencies better than RNN and TCN. TCN obtains higher perplexity but consumes smaller number of model parameters than RNN. This is due to the convolution calculation in TCN. In addition, stochastic variant of TCN performs better than TCN, but the model size is increased significantly due to additional memory consumption from prior network and inference network. Nevertheless, CRN obtains lower perplexity than STCN with even with smaller model size. Among these methods, the lowest test perplexity is achieved by using MR-CRN while the model size is increased as well due to multi-level resolutions using multiple layers of encoders and decoders. MR-CRN resolves the possible model overfitting in CRN. With the multi-time-scale contextual information, the proposed MR-CRN sufficiently capture long-term and short-term temporal patterns, and desirably behaves and performs for word prediction.

TABLE II: Classification accuracy and model size using UCF101.

Model	#Params	Accuracy
RNN	6.3M	77.9%
RRN	8.8M	79.1%
TCN [9], [10]	6.1M	81.8%
CRN	9.7M	82.7%
MR-CRN	11.9M	**84.5%**

B. Evaluation for action recognition

Sequential learning is further evaluated for human action recognition using UCF101 [29], [30]. UCF101 consisted of 13,320 video clips with diverse forms of camera motion and illustration from 101 action classes. Each video clip has a frame rate of 25 (frames per second) with various lengths. The resolution of 320 x 240 pixels was recorded. Different from language modeling using TCN encoder, the task of video classification adopted an CNN which transformed a video stream into a sequence of local features. Next, the TCN encoder and RNN decoder were applied to capture the short-term and long-term temporal information and implement the CRN for classification outputs for action classification. In this set of evaluation, RNN, RRN, TCN, CRN and MR-CRN ($L = 3$) were implemented. Different models were trained by twenty epochs with Adam optimizer[31]. The size of hidden states or the amount of kernels was set as 512 for all models. A pretrained ResNet [32] from ImageNet was used as the CNN to find a set of embeddings for sequential learning. The training batch size was fixed at 40 and the initial learning rate was set at 10^3. The results are shown in Table II. In action recognition, local information is more important than long-term dependency since TCN performs better than RNN and RRN with reduced model size. It is important to characterize local dynamics in adjacent frames. The hybrid model CRN

obtains higher accuracy than individual RNN and CRN. In this comparison, the highest classification accuracy is achieved by using MR-CRN. In summary, the proposed multi-resolution convolutional recurrent network consistently performs better than other individual and hybrid methods in two tasks with data in different domains.

V. Conclusions

This paper presented a new sequential learning method based on the multi-resolution convolutional recurrent network. The local and global features as the temporal information in sequence data were captured via a cascade of temporal convolutional network and recurrent neural network, respectively. The convolutional recurrent network was accordingly proposed as a general framework of TCN and RNN where the advantages of both individual models were captured. In CRN, the local information was reflected by TCN encoder while RNN played a role of decoder with a global view. In particular, the proposed CRN was further strengthened by incorporating the information of multi-level time scales in representation of multivariate time series. The multi-resolution CRN was proposed to sufficiently utilize different scales of temporal information. The experiments on sequential prediction in language modeling and action recognition showed that the hybrid models using CRN and multi-resolution CRN performed better than stand-alone models and other hybrid models with different order of stacking or cascading in model integration. Future work will be extended by introducing stochastic modeling and topic modeling [33], [34], [35], [36] in multi-resolution CRN.

References

[1] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[3] J.-T. Chien and Y.-C. Ku, "Bayesian recurrent neural network for language modeling," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 2, pp. 361–374, 2015.

[4] G. Saon and J.-T. Chien, "Bayesian sensing hidden markov models," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 43–54, 2011.

[5] J.-T. Chien and C.-Y. Kuo, "Markov recurrent neural network language model," in *Proc. of IEEE Automatic Speech Recognition and Understanding Workshop*, 2019, pp. 807–813.

[6] K. Cho, Bart Van M., C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

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

[8] Y. LeCun, Y. Bengio, et al., "Convolutional networks for images, speech, and time series," *The Handbook of Brain Theory and Neural Networks*, 1995.

[9] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[10] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[11] J. Koutnik, K. Greff, F. Gomez, and J. Schmidhuber, "A clockwork RNN," *arXiv preprint arXiv:1402.3511*, 2014.

[12] D. Neil, M. Pfeiffer, and S.-C. Liu, "Phased LSTM: Accelerating recurrent network training for long or event-based sequences," in *Advances in Neural Information Processing Systems*, 2016, pp. 3882–3890.

[13] J. Chung, S. Ahn, and Y. Bengio, "Hierarchical multiscale recurrent neural networks," *arXiv preprint arXiv:1609.01704*, 2016.

[14] S. Chang, Y. Zhang, W. Han, M. Yu, X. Guo, W. Tan, X. Cui, M. Witbrock, M. A. Hasegawa-Johnson, and T. S. Huang, "Dilated recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 77–87.

[15] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[16] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *Proc. of ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 95–104.

[17] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015, pp. 4580–4584.

[18] V. Campos, B. Jou, X. Giró-i Nieto, J. Torres, and S.-F. Chang, "Skip RNN: Learning to skip state updates in recurrent neural networks," *arXiv preprint arXiv:1708.06834*, 2017.

[19] C. Tallec and Y. Ollivier, "Can recurrent neural networks warp time?," *arXiv preprint arXiv:1804.11188*, 2018.

[20] J.-T. Chien and Y.-M. Huang, "Stochastic convolutional recurrent networks," in *Proc. of International Joint Conference on Neural Networks*, 2020, pp. 1–6.

[21] J.-T. Chien and Y.-M. Huang, "Stochastic convolutional recurrent networks for language modeling.," in *Proc. of Annual Conference of International Speech Communication Association*, 2020, pp. 3640–3644.

[22] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, pp. 400–407, 1951.

[23] J. Kiefer, J. Wolfowitz, et al., "Stochastic estimation of the maximum of a regression function," *The Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 462–466, 1952.

[24] T. Mikolov and G. Zweig, "Context dependent recurrent neural network language model," in *Proc. of IEEE Workshop on Spoken Language Technology*, 2012, pp. 234–239.

[25] E. Aksan and O. Hilliges, "STCN: Stochastic temporal convolutional networks," *arXiv preprint arXiv:1902.06568*, 2019.

[26] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[27] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," *Advances in Neural Information Processing Systems*, vol. 29, pp. 1019–1027, 2016.

[28] L. Van der Maaten and G. Hinton, "Visualizing data using *t*-SNE," *Journal of Machine Learning Research*, vol. 9, no. 11, 2008.

[29] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[30] Y.-M. Huang, H.-H. Tseng, and J.-T. Chien, "Stochastic fusion for multi-stream neural network in video classification," in *Proc. of Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, 2019, pp. 69–74.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[32] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, Inception-ResNet and the impact of residual connections on learning," in *Proc. of AAAI Conference on Artificial Intelligence*, 2017.

[33] J.-T. Chien and C.-H. Chueh, "Latent dirichlet language model for speech recognition," in *Proc. of IEEE Spoken Language Technology Workshop*, 2008, pp. 201–204.

[34] J.-T. Chien and Y.-L. Chang, "Bayesian sparse topic model," *Journal of Signal Processing Systems*, vol. 74, no. 3, pp. 375–389, 2014.

[35] J.-T. Chien, "Hierarchical Pitman–Yor–Dirichlet language model," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 8, pp. 1259–1272, 2015.

[36] J.-T. Chien, "Hierarchical theme and topic modeling," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 3, pp. 565–578, 2015.