Adaptive Detrending for Accelerating the Training of Convolutional Recurrent Neural Networks

Minju Jung (P)[1,2], and Jun Tani[2]

[1] School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Korea
[2] Cognitive Neurorobotics Research Unit, Okinawa Institute of Science and Technology, Japan
E-mail: tani1216jp@gmail.com

Abstract— Convolutional recurrent neural networks (ConvRNNs) provide robust spatio-temporal information processing capabilities for contextual video recognition, but require extensive computation that slows down training. Inspired by detrending methods, we propose "adaptive detrending" (AD) for temporal normalization in order to accelerate the training of ConvRNNs, especially of convolutional gated recurrent unit (ConvGRU).

Keywords—Detrending, normalization, internal covariate shift, convolutional recurrent neural networks (ConvRNNs)

1 Introduction

The current paper focuses on the time domain in order to accelerate training of convolutional recurrent neural networks (ConvRNNs). Much of time series analysis and many forecasting methods can be applied only to stationary time series. Detrending transforms non-stationary time series to stationary series by identifying the change as a trend and removing it. The current research applies this method to normalize sequences of neurons in recurrent neural networks (RNNs). Our key insight here is that the hidden state of a gated recurrent unit (GRU) [1] can be considered as a trend that can be approximated by the form of an exponential moving average with an adaptively changing decay factor. Based on this insight, we propose a novel temporal normalization method, "adaptive detrending" (AD), for use with GRU and convolutional gated recurrent unit (ConvGRU). The implications of AD are fourfold:

- AD is easy to implement, reducing computational cost and consuming less memory than competing methods.
- AD eliminates temporal internal covariate shift.
- AD controls the degree of detrending (or normalization) through decay factor adaptability.
- AD is fully compatible with existing normalization methods.

2 Model

2.1 Gated Recurrent Unit

The gated recurrent unit (GRU) was proposed by Cho et al. [1] to overcome the vanishing gradient problem by using a gating mechanism. Specifically, GRU has two gating units, called a reset gate $\mathbf{r}$ and an update gate $\mathbf{z}$, and is defined as follows:

$$\mathbf{h}_t = \mathbf{z}_t \odot \tilde{\mathbf{h}}_t + (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} \tag{1}$$

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1} + \mathbf{b}_z) \tag{2}$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_h \mathbf{x}_t + \mathbf{r}_t \odot \mathbf{U}_h \mathbf{h}_{t-1} + \mathbf{b}_h) \tag{3}$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1} + \mathbf{b}_h) \tag{4}$$

where $\sigma(\cdot)$ is a sigmoid function and $\odot$ is an element-wise multiplication.

The convolutional gated recurrent unit (ConvGRU) is a natural extension of GRU by replacing the weight multiplication of GRU with convolution.

2.2 Gated Recurrent Unit Normalization in the Spatial Domain

Following Ba et al. [2], in this paper we apply recurrent batch normalization (recurrent BN) [3] and layer normalization (LN) [2] to GRU. We refer to recurrent BN and LN as "spatial" normalization methods to differentiate the present approach from normalization in the time domain reviewed above. The following equations represent GRU normalization in the spatial domain:

$$\mathbf{r}_t = \sigma(\mathrm{N}_{\boldsymbol{\gamma},\boldsymbol{\beta}}(\mathbf{W}_r \mathbf{x}_t) + \mathrm{N}_{\boldsymbol{\gamma}}(\mathbf{U}_r \mathbf{h}_{t-1})) \tag{5}$$

$$\mathbf{z}_t = \sigma(\mathrm{N}_{\boldsymbol{\gamma},\boldsymbol{\beta}}(\mathbf{W}_z \mathbf{x}_t) + \mathrm{N}_{\boldsymbol{\gamma}}(\mathbf{U}_z \mathbf{h}_{t-1})) \tag{6}$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathrm{N}_{\boldsymbol{\gamma},\boldsymbol{\beta}}(\mathbf{W}_h \mathbf{x}_t) + \mathbf{r}_t \odot \mathrm{N}_{\boldsymbol{\gamma}}(\mathbf{U}_h \mathbf{h}_{t-1})) \tag{7}$$

$$\mathbf{h}_t = \mathbf{z}_t \odot \tilde{\mathbf{h}}_t + (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} \tag{8}$$

where $\mathrm{N}_{\boldsymbol{\gamma},\boldsymbol{\beta}}(\cdot)$ represents the normalization followed by an affine transformation with two learnable parameters (gain $\boldsymbol{\gamma}$ and bias $\boldsymbol{\beta}$) for recurrent BN and LN, and $\mathrm{N}_{\boldsymbol{\gamma}}(\cdot)$ is the same as $\mathrm{N}_{\boldsymbol{\gamma},\boldsymbol{\beta}}(\cdot)$ except for an affine transformation with only the gain $\boldsymbol{\gamma}$ to remove the bias redundancy within an equation.

2.3 Adaptive Detrending

In statistics, a MA is widely used to extract long-term trends from noisy time series by filtering out fluctuations. Among these variants, an exponential moving average (EMA) is preferred when the MA needs to quickly respond to recent data because past data decay exponentially over time. The value of the EMA μ_t at time step t is calculated by

$$\mu_t = \alpha \cdot x_t + (1 - \alpha) \cdot \mu_{t-1} \tag{9}$$

where x_t is the current input value and α is a constant decay factor or smoothing factor between 0 and 1.

Detrending is a method that removes a slowly changing component, called a "trend", in order to render time series stationary. We think that detrending can be applied to RNNs to eliminate temporal internal covariate shift. Notice that the definition of EMA in (9)

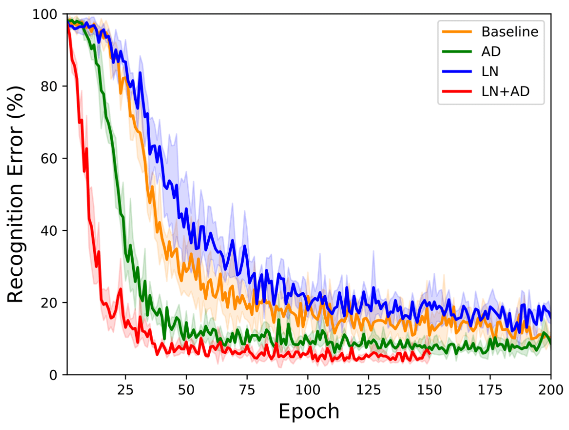

Figure 1: Graph of test recognition error averaged over three splits versus training epochs on the object-related with modifier (OA-M) recognition dataset.

Table 1: Comparison of convergence speed on the OA-M recognition dataset.

Model	Epochs to Baseline's max accuracy	Acceleration
Baseline	200	$\times 1.0$
AD	63	$\times 3.2$
LN+AD	28	$\times 7.1$

is the same as that for the hidden state $\mathbf{h}$ of GRU in (1) when, rather than being fixed, the decay factor α is continuously changing at each time step as shown in (2). By considering the hidden state $\mathbf{h}$ as a trend of the candidate hidden state $\tilde{\mathbf{h}}$, we can apply detrending to GRU for temporal normalization, as follows:

$$\mathbf{y}_t = \tilde{\mathbf{h}}_t - \mathbf{h}_t \tag{10}$$

where $\mathbf{y}_t$ is the detrended output at time step t, and is input into the next layer.

As mentioned above, the proposed detrending method uses the update gate $\mathbf{z}$ in (2) as the decay factor α in (9), but it is adaptively changed over time rather than fixed. Hence, we call this method "adaptive detrending" (AD) to differentiate it from conventional detrending methods that employ a pre-defined or fixed setting to estimate a trend.

3 Object-Related Action with Modifier Recognition Experiment

We tested AD on the object-related action with modifier (OA-M) recognition dataset. The dataset for OA-M recognition consisted of 840 videos in 42 object-action-modifier combination classes created by non-exhaustively and non-redundantly combining four objects, four actions, and six modifiers. Each object-action-modifier combination class was performed by 10 subjects two times each with a randomly selected distractor present in each video.

Surprisingly, LN performs worse than the baseline in terms of training speed and recognition accuracy (Fig. 1). We hypothesize that the statistics estimation error plaguing LN when implemented in CNNs is accumulated through time in ConvRNNs, leading to significant decrease in convergence speed.

AD improves convergence speed significantly as well as increasing recognition accuracy over those of the baseline and LN (Fig. 1) and needs 3.2 times fewer epochs than the baseline (Table 1). These results imply that the time domain is more critical than the spatial domain when normalizing RNNs. Furthermore, by solving the limitation of LN with neuron-wise normalization of AD, LN+AD shows the most significant improvements in both training speed and generalization over the baseline, as well as over LN or AD, alone. Specifically, LN+AD requires 7.1 and 2.2 times fewer epochs, and improves recognition accuracy by 4.3% and 1.8% compared with those of the baseline and AD. These results show that utilizing the time domain as well as the spatial domain for normalization generates beneficial synergy.

4 Conclusion

This paper proposes a novel temporal normalization method, "adaptive detrending" (AD), to accelerate training of recurrent neural networks (RNNs) by removing the temporal internal covariate shift. Although several normalization methods employing batch normalization (BN) [4] have been proposed to accelerate training of RNNs, these methods utilize only the spatial domain and neglect the time domain for statistical estimation. The key insight of this paper is to view the hidden state of the gated recurrent unit (GRU) as a trend with an exponential moving average. With this in mind, and with simple modifications, we were able to implement AD in GRU. AD has several advantages over other normalization methods: It is highly efficient in terms of computational and memory requirements. Unlike conventional detrending methods that require manual parameter setting, AD learns and estimates trends automatically. AD is generally applicable to both GRU and ConvGRU, which is not the case for either BN or for layer normalization (LN).

References

[1] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder–decoder for statistical machine translation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014.

[2] L. J. Ba, R. Kiros, G. E. Hinton, Layer normalization, CoRR abs/1607.06450.

[3] T. Cooijmans, N. Ballas, C. Laurent, A. Courville, Recurrent batch normalization, in: International Conference on Learning Representations (ICLR), 2017.

[4] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Proceedings of the 32nd International Conference on Machine Learning, 2015.