Signal, Image and Video Processing
https://doi.org/10.1007/s11760-022-02292-2

ORIGINAL PAPER

Handling irregularly sampled signals with gated temporal convolutional networks

Fatih Aslan[1] · S. Serdar Kozat[1]

Received: 16 March 2022 / Revised: 14 June 2022 / Accepted: 15 June 2022

Abstract
We investigate the sequential modeling problem and introduce a novel gating mechanism into the temporal convolutional network architectures. In particular, we introduce the gated temporal convolutional network architecture with elaborately tailored gating mechanisms. In our implementation, we alter the way in which the gradients flow and avoid the vanishing or exploding gradient and the dead ReLU problems. The proposed GTCN architecture is able to model the irregularly sampled sequences as well. In our experiments, we show that the basic GTCN architecture is superior to the generic TCN architectures in various benchmark tasks requiring the modeling of long-term dependencies and irregular sampling intervals. Moreover, we achieve the state-of-the-art results on the permuted sequential MNIST and the sequential CIFAR10 benchmarks with the basic structure.

Keywords Sequential learning · Irregular sampling · Temporal convolutional networks · Time series classification

1 Introduction

1.1 Preliminaries

We study the classification and prediction of sequential data using deep learning architectures. This problem, sequential modeling, is extensively studied in the signal processing and machine learning [1] studies since it arises in various real-life applications including finance [2], medicine [3], and cyber-security [4]. In most of these applications, nonlinear approaches are preferred when the linear modeling is inadequate [5]. Until recent years, recurrent architectures such as recurrent neural networks (RNN) [6] and their more advanced versions such as long-short term memory (LSTM) [7] and gated recurrent unit (GRU) [8] were considered as the default architectures for signal modeling tasks by deep learning researchers and practitioners [9]. Contrarily, the recent results on sequential data processing have shown that convolutional architectures can outperform recurrent architectures in modeling the long-term dependencies in sequential learning tasks [1]. However, despite their success at reaching the state-of-the-art results in sequential learning tasks, the ability of convolutional architectures to model the irregularly sampled signals is not investigated apart from the continuous kernel convolutions [10]. The sequences in most real-life applications are sampled irregularly or have missing values [11]. For instance, in medical applications, the condition of a patient is recorded at varying intervals [12]. In this paper, we introduce a novel architecture capable of modeling long-term dependencies as well as the irregularities in the sampling intervals. In particular, we introduce novel gating mechanisms to the generic temporal convolutional networks (TCN) [1] to enable the architectures to model the sampling irregularities alongside long-term dependencies. We show that the proposed architecture, namely gated temporal convolutional networks (GTCNs), outperforms the generic TCN architectures when the sampling intervals are regular or irregular, especially in very deep architectures. Furthermore, the results on real-world and artificial benchmark datasets indicate that the basic GTCNs outperform the state-of-the-art architectures in various sequential modeling tasks with regular and irregular sampling intervals.

The code for implementing the GTCN architecture and the datasets used in the experiments are available at https://github.com/aslanfth/gatedtcn.

✉ Fatih Aslan
faslan@ee.bilkent.edu.tr

S. Serdar Kozat
kozat@ee.bilkent.edu.tr

[1] Bilkent University, Ankara, Turkey

Published online: 06 July 2022

1.2 Prior art and comparisons

Sequence modeling using deep learning architectures can be divided into recurrent architectures and convolutional architectures. Recurrent architectures model the sequences using inherent recurrent connections and process the sequential data, while convolutional architectures model the sequences using stacked causal convolutions. Mainly, in recurrent architectures inherent characteristics of the sequence are summarized in a state vector and this state vector is updated as the new observations arrive. Whereas, in convolutional architectures different parts of the sequence is passed through convolution filters, i.e., for feature extraction, and the output is produced at each time step. There also exist architectures that combine the recurrent and convolutional architectures such as convolutional LSTM [13]. Due to the sequential state update, the gradient flows in the same direction with time in the recurrent architectures. Hence, these architectures suffer from vanishing or exploding gradients as the sequence length increases and fall short to model the long-term dependencies [1]. On the other hand, convolutional architectures reduce the effects of the vanishing or exploding gradients by propagating the gradient through the deeper layers of the architecture, i.e., not sequentially in time. Convolutional architectures utilize causal convolutions to uphold the causality constraint of the sequential learning task and stack dilated convolutions on top of each other to cover more observations in the past [1,14,15]. There are architectures that use the ensemble technique to benefit from both the recurrent and convolutional architectures as well [16].

Various sequential modeling applications in real-life contain irregular sampling intervals either because of the nature of the application or missing values in regular sampling [12]. The Time Gated LSTM (TG-LSTM) architecture [17] incorporates the sampling interval by concatenating the interval information to the input as an additional channel. Other recurrent architectures such as [18] utilize ordinary differential equations to model the irregular sampling. Recently, an architecture that utilizes continuous kernel convolutions to model the irregularly sampled sequences is proposed [10].

In [1], it is shown that the generic TCNs outperform most generic recurrent architectures in modeling long-term dependencies. While extensions of RNNs such as Dilated RNN [19] reduce the effects of the vanishing or exploding gradients and outperform the generic TCN on certain tasks, we focus on convolutional architectures and demonstrate that our architecture outperforms those extensions as well. By using ReLU [20] activation functions, TCNs alleviate the vanishing or exploding gradients. However, ReLU activation functions die in very deep architectures which is needed when the dependency is extensive in sequential modeling, e.g., when we have long period seasonalities [21]. To remedy, convolutional architectures with gating mechanisms are proposed such as [22–24]. Gating mechanisms are employed to the outputs of the convolutional architectures in [22,23], while [24] utilizes the gating functions from [15] in the building blocks of the two dimensional convolutional network. Contrary, we introduce an elaborately tailored gating mechanism into the building blocks of the generic TCN architectures. Furthermore, this new gating mechanism enables the architecture to model the irregularly sampled sequences as well.

1.3 Contributions

Our main contributions are as follows.

1. We introduce a novel gating mechanism into the generic TCN [1] architectures to model very long-term dependencies by altering the way of gradient flow and alleviating vanishing or exploding gradients and the dead ReLU problems together.
2. We demonstrate that the proposed GTCN architecture is robust to missing values and irregularly sampled sequential data.
3. Through an extensive set of experiments, we compare the basic GTCN architectures with the generic TCN architectures [1] and show that the GTCNs are superior to the generic TCNs in well-known benchmark datasets.
4. We achieve the state-of-the-art results in the sequential CIFAR10 [25], permuted sequential MNIST [26], and speech commands [27] datasets using our basic architecture.

The organization of this paper is as follows. We define our problem setting in Sect. 2. In Sect. 3, we provide the generic TCN architectures, give the motivation behind the proposed architecture, and then introduce our GTCN architecture. Section 4 compares the performance of the GTCN architecture with the state-of-the-art architectures under the irregular and regular sampling scenarios. We conclude the paper with final remarks in Sect. 5.

2 Problem description

All vectors are column vectors and denoted by boldface lower case letters. Matrices are represented by boldface capital letters. Δt_i denotes the sampling intervals, i.e., $\Delta t_i = t_i - t_{i-1}$. For a vector $\mathbf{u}$, $\mathbf{u}^T$ is the ordinary transpose. The time index is given as subscript, e.g., $\mathbf{u}_{t_i}$ is the vector at time step t_i, and the layer index is given as superscript. $[\mathbf{U}, \mathbf{V}]$ denotes that the matrices $\mathbf{U}$ and $\mathbf{V}$ are concatenated on the temporal dimension. $[\mathbf{U}; \mathbf{V}]$ denotes that the matrices $\mathbf{U}$ and $\mathbf{V}$ are concatenated on the channel dimension. $\sigma(\mathbf{U})$ denotes that

the sigmoid function is applied to the matrix $\mathbf{U}$ element-wise and $\otimes$ denotes the element-wise multiplication. $(\mathbf{U})^l$ denotes the Hadamard power, and $f'(\mathbf{U})$ denotes the derivative of the function $f(\mathbf{U})$.

We study the sequence modeling task, where we observe an input sequence $\mathbf{X}_{t_k} = [\mathbf{x}_{t_0}, \mathbf{x}_{t_1}, \ldots, \mathbf{x}_{t_k}]$ and produce predictions $\hat{\mathbf{Y}}_{t_k} = [\hat{\mathbf{y}}_{t_0}, \hat{\mathbf{y}}_{t_1}, \ldots, \hat{\mathbf{y}}_{t_k}]$ of the output sequence $\mathbf{Y}_{t_k} = [\mathbf{y}_{t_0}, \mathbf{y}_{t_1}, \ldots, \mathbf{y}_{t_k}]$, where $\mathbf{x}_{t_i} \in \mathbb{R}^m$ and $\mathbf{y}_{t_i}, \hat{\mathbf{y}}_{t_i} \in \mathbb{R}^n$. While producing the outcome $\hat{\mathbf{y}}_{t_i}$, we only use the observed inputs up to step t_i, i.e., the goal is to estimate the possibly nonlinear mapping from the input sequence to the output sequence, $\hat{\mathbf{y}}_{t_i} = f([\mathbf{X}_{t_i}, \mathbf{Y}_{t_{i-1}}])$. The nonlinear mapping is found by minimizing the accumulated loss between the actual output sequence $\mathbf{y}_{t_i}$ and the predicted sequence $\hat{\mathbf{y}}_{t_i}$, $\bar{L} = \frac{1}{k}\sum_{i=1}^{k} L(\mathbf{y}_{t_i}, \hat{\mathbf{y}}_{t_i})$.

The sampling intervals of the input signal may not be regular and Δt_i may vary throughout the sequence. This implementation provides us flexibility when dealing with the sequences with missing values. For instance, if some inputs from the regularly sampled sequence $\mathbf{X}_{t_k}$ are missing, we discard the missing values and rearrange the sampling intervals accordingly.

3 Novel gating mechanism for TCN architectures

After a brief description of the generic TCN, we provide the motivation and gating mechanism of our algorithm incorporating the sampling interval information while keeping the gradients stable on all hidden layers.

3.1 Generic TCN architectures

The generic TCN architectures are first introduced in [1] and outperformed most generic recurrent architectures such as LSTM [7] and GRU [8] on sequence modeling tasks. TCNs utilize dilated and causal convolutions similar to the other convolutional sequence modeling architectures, such as Wavenet [14] and GLU [15].

Similar to the other convolutional architectures for sequence modeling, TCNs stack dilated convolutions on top of each other to increase the receptive field exponentially. The receptive field of the TCN is calculated using the number of hidden layers (n), the dilation coefficient (d), and the kernel size (k) of the architecture. The exact calculation of the receptive field (h) is given as: $h = k \times d^n$. TCNs do not alter the dilation size and keep it fixed, e.g., $d = 2$. Therefore, to increase the receptive field of the TCN, one can either increase the kernel size or the number of hidden layers. To increase the receptive field exponentially while keeping the number of parameters low, one needs to increase the number of hidden layers in the architecture.

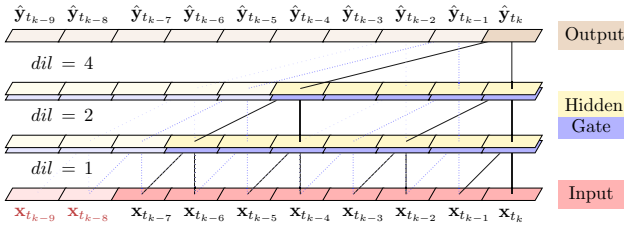


Fig. 1 GTCN architecture with $d = 2$, $k = 2$, $n = 2$, and the receptive field $h = 8$.dil denotes the dilation of the dilated convolutional filters in each layer

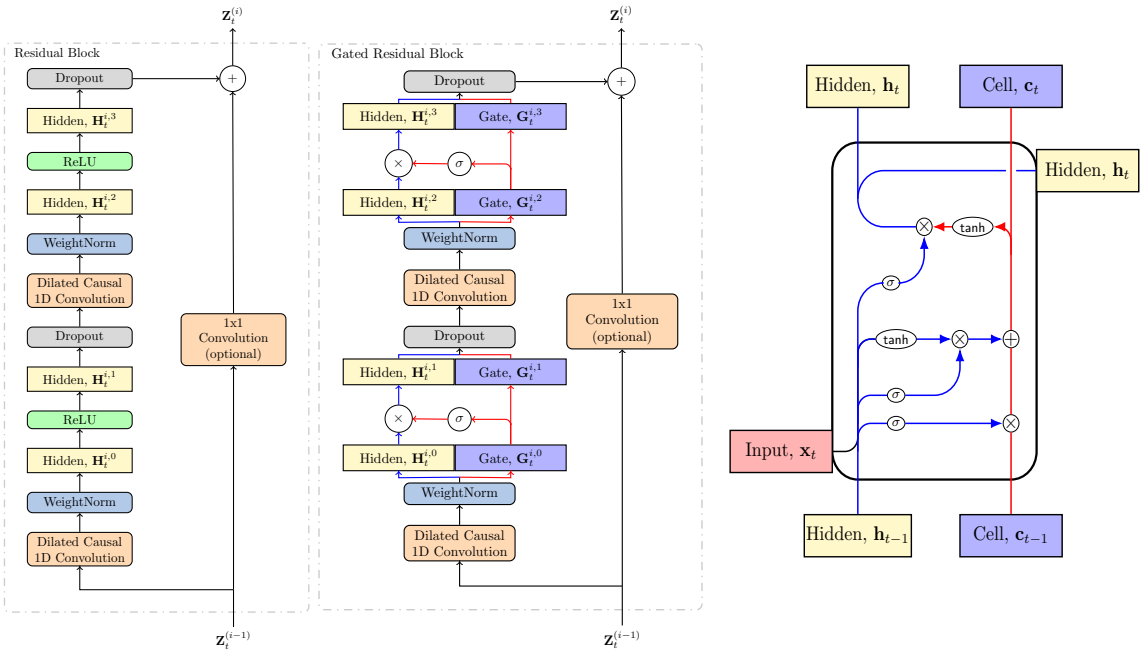
Contrary to the other convolutional architectures for sequence modeling, the generic TCN architecture does not use any gating mechanisms and uses ReLU [20] activations on the hidden layers. Furthermore, a temporal convolutional architecture with gated activations is reported to have no significant performance improvement over the generic TCN architectures [1]. Even though convolutional architectures decrease the negative effects of vanishing and exploding gradients by flowing the gradients in a different direction instead of temporal direction, i.e., the gradients flow just through the hidden layers, very deep architectures still suffer from vanishing or exploding gradients especially as we increase the number of stacked layers. By having ReLU activations rather than gating mechanisms, TCNs avoid the vanishing or exploding gradient problem. However, ReLU activation functions still die in very deep architectures or under improper initializations [21].

We adopt the generic TCN [1] architecture and add gated activation functions to increase the performance. The generic GTCN architecture is shown in Fig. 1. The receptive field of the architecture in Fig. 1 is $h = 2 \times 2^2 = 8$. Thus, the architecture can learn up to 8 previous steps, e.g., $\hat{\mathbf{y}}_{t_k} = f([\mathbf{x}_{t_{k-7}}, \mathbf{x}_{t_{k-6}}, \ldots, \mathbf{x}_{t_k}])$. As in Fig. 1, unlike the TCN, we propagate the gate information alongside the hidden representations to the deeper layers of the architecture. The building blocks of the generic TCN and the GTCN is shown in Fig. 2a, b, respectively.

3.2 Motivation

In order to model very long-term dependencies, the receptive field of the architecture needs to cover inputs from all previous steps that affect the output. The receptive field is increased by increasing the number of hidden layers. However, as the number of hidden layers is increased, generic TCN architectures suffer from dead ReLU, whereas gated convolutional architectures suffer from vanishing or exploding gradients.

In order to incorporate the sampling interval information, one can directly concatenate the sampling intervals to the input matrices. In this direct implementation, the sampling intervals have additive effects on the output of the architec-



(a) Residual block [1] **(b)** Gated residual block **(c)** An LSTM cell [7]

Fig. 2 Residual blocks used in building generic TCNs (**a**), and proposed GTCN architecture (**b**), along with a generic LSTM cell (**c**). Rectangles with sharp corners represent the hidden representations, gate information, hidden state, and cell state. Red arrows in **b** and **c** represent the propagation of the gate information and cell state, while blue arrows represent the propagation of the hidden representation and hidden state (color figure online)

ture when the generic TCN architectures are used. Using Taylor series expansion, Sahin and Kozat [17] prove that the sampling intervals have a scaling effect on the actual output. However, due to Taylor series approximation, proof in [17] only stands for sufficiently small sampling intervals. Larger sampling intervals can have both additive and scaling effects on the actual output.

The GTCN architecture is inspired by the LSTMs and combines powerful aspects of the convolutional and gated architectures. As in Fig. 2b, c, the gate information of the GTCN architectures operate similarly to the cell states of the LSTM architectures.

3.3 Gated temporal convolutional network

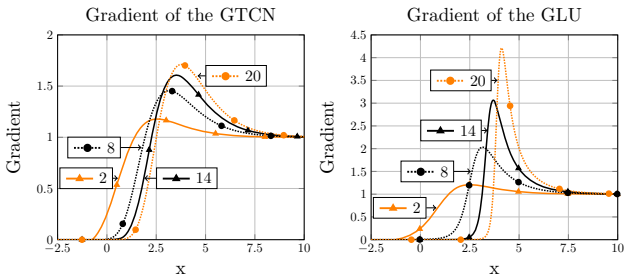
As shown in Fig. 2a, b, contrary to the generic TCN architectures, our implementation contains gated activations instead of the ReLU activations.

The gradient in our architecture flows similar to the GLU [15] and Wavenet [14], with the exception of propagating the gate information to the deeper layers as shown in Fig. 2b. It is shown that the GLU reduces the negative effects of the vanishing or exploding gradients compared to the Wavenet [15]. The GTCNs reduce the problem of vanishing or exploding gradients even further. As shown in [15], the gradient of the single layer gated linear unit

$$\nabla[\mathbf{X} \otimes \sigma(\mathbf{X})] = \nabla\mathbf{X} \otimes \sigma(\mathbf{X}) + \mathbf{X} \otimes \sigma'(\mathbf{X})\nabla\mathbf{X} \quad (1)$$

has a path $\nabla\mathbf{X} \otimes \sigma(\mathbf{X})$ without downscaling, i.e., this path avoids the vanishing or exploding gradients as more units are stacked. In (1), the one-dimensional convolution operations are omitted following the notation in [15]. The gradient of the gated temporal convolutional network with l hidden layers $\nabla[\mathbf{X} \otimes (\sigma(\mathbf{X}))^l] = \nabla\mathbf{X} \otimes (\sigma(\mathbf{X}))^l + l\mathbf{X} \otimes (\sigma(\mathbf{X}))^{l-1} \otimes \sigma'(\mathbf{X})\nabla\mathbf{X}$ has no path with downscaling and the gradients remain more stable. The gradients of the GTCN and GLU for the scalar case is provided in Fig. 3 with alternating number of hidden layers. As demonstrated in Fig. 3, the gradient of the GTCN remains stable as the number of hidden layers increase, which is needed when the sequences are long.

Gated activations also allow the architecture to model the scaling effects of the sampling intervals. Our implementation of the gated activations inspired by the GLUs [15] and TG-LSTMs [17]. In addition, the GTCN architectures model the additive effects of the sampling intervals by concatenating the hidden representation and the gate information before the one-dimensional convolution operation. The one-dimensional convolution operation can be considered as a linear multiplication on the channel dimension and rewritten as $(\mathbf{w}_t)^T[\mathbf{h}_t; \mathbf{g}_t]$, enabling $\mathbf{h}_t$ and $\mathbf{g}_t$ to additively affect each other. Here, $\mathbf{w}_t$ denotes the column vector consisting of the weights of the one-dimensional convolutional filter along the channel dimension, we omit the layer superscript for notational simplicity.



(a) GTCN. **(b)** GLU.

Fig. 3 The gradients of the GTCN and GLU for the scalar case with alternating number of hidden layers. The number of hidden layers is provided as legend entries

Our implementation consists of the gated residual blocks, which are stacked on top of each other. As shown in Fig. 2b, the gated residual block on level i takes the input $\mathbf{Z}_t^{i-1}$ and propagates the output $\mathbf{Z}_t^i$ to the gated residual block at level $i+1$. In a gated residual block, the hidden representation and the gate information are chunked on the channel dimension to employ the gated activation and concatenated again before the dropout operation. We follow the generic TCN implementation of [1] and utilize weight normalizations, dropouts, and residual connections. The gated residual block on level i operates on the input sequence as follows:

$$
\begin{aligned}
[\mathbf{H}_t^{i,0}; \mathbf{G}_t^{i,0}] &= \mathrm{Conv1}D([\mathbf{H}_t^{i-1,3}; \mathbf{G}_t^{i-1,3}]),\\
[\mathbf{H}_t^{i,1}; \mathbf{G}_t^{i,1}] &= [\mathbf{H}_t^{i,0} \otimes \sigma(\mathbf{G}_t^{i,0}); \mathbf{G}_t^{i,0}],\\
[\mathbf{H}_t^{i,2}; \mathbf{G}_t^{i,2}] &= \mathrm{Conv1}D([\mathbf{H}_t^{i,1}; \mathbf{G}_t^{i,1}]),\\
[\mathbf{H}_t^{i,3}; \mathbf{G}_t^{i,3}] &= [\mathbf{H}_t^{i,2} \otimes \sigma(\mathbf{G}_t^{i,2}); \mathbf{G}_t^{i,2}].
\end{aligned}
$$

We denote the one-dimensional dilated causal convolution with $\mathrm{Conv1}D(\cdot)$ and omit the weights of the convolutions. We omit the weight normalizations, the dropouts, and the residual connection in the equations for notational simplicity.

4 Experiments

We provide experiments on different sequence modeling tasks with regular and irregular sampling and illustrate the performance of the GTCN architecture by comparing it with the TCN [1] and other state-of-the-art architectures.

We first compare the performance of the GTCN architecture with the TCN [1] architecture on benchmark datasets that are widely used for evaluating the ability to model the long-term dependencies of the architectures. We also report the results obtained with the state-of-the-art architectures for comparison. We set the kernel size and the number of layers as 8 to ensure that the receptive field covers the entire sequence, i.e., the receptive field is $8 \times 2^7 = 1024$.

The sequential MNIST dataset [26] is a well-established benchmark for testing the sequential architectures. The task is based on the classical well-known MNIST [30] dataset. In the sequential version of the MNIST dataset, grayscale MNIST images are flattened and fed to the architectures sequentially. In order to have a fair comparison between the architectures, we set the number of parameters of both the GTCN and TCN to be around 300 k. Accuracy scores on the test data are given in Table 1 under the column sMNIST alongside the state-of-the-art results. Since the task is relatively simple, there is no significant difference between the performances of the TCN and GTCN architectures. However, both of the architectures perform near the state-of-the-art and outperform most of the existing architectures.

In the permuted sequential MNIST dataset [26], each flattened pixel array of the sequential MNIST dataset is permuted based on a fixed random permutation before fed to the architectures. Due to the random permutation, local properties in the image cannot be captured, and the task requires the architectures to model the dependencies on the more distant past. Accuracy scores of the GTCN architecture and the state-of-the-art architectures are provided in Table 1, under the column pMNIST. In the permuted sequential MNIST dataset, the GTCN outperforms the current state-of-the-art CKCNN-Big [10]. Table 1 shows that the performance difference between the TCN and the GTCN architectures increases as the task gets more complex.

A more challenging sequential modeling test is the sequential CIFAR10 [25] benchmark. Similar to the sequential MNIST dataset, RGB-colored images of the original CIFAR10 dataset [31] are flattened and then fed to the architectures sequentially, having 3 input channels. Comparative results are shown in Table 1 under the column sCIFAR10. The GTCN outperforms the current state-of-the-art. Due to the longer sequences, the difference between the performances of the GTCN and other architectures becomes more significant.

We next compare the TCN and the GTCN under an irregular sampling scenario with the Speech Commands dataset [27]. For comparisons, we provide the state-of-the-art results as well. By having sequences with a length of 16,000, the Speech Commands dataset is an excellent choice to assess the architectures based on the ability to model the long-term dependencies alongside the irregularities in the sampling intervals. There are voice recordings of various short commands in the Speech Commands dataset. We follow the work of [10] and choose ten different commands to predict. Contrary to [32], we do not employ any preprocessing on the speech data other than normalization. This dataset is referred to as the Raw Speech Commands dataset in [10]. We follow the methodology of [10,32] to compose the irregularly sampled sequences by dropping a part of each sequence randomly, determined by the drop rate. When the portion of the

Table 1 Accuracy (%) scores on the sequential CIFAR10 (sCIFAR10), sequential MNIST (sMNIST), and permuted sequential MNIST (pMNIST) datasets

Architecture	Accuracy		
	sCIFAR10	sMNIST	pMNIST
LSTM (70 k) [1]	–	87.2	85.7
GRU (70 k) [1]	–	96.2	87.3
TCN (70 k) [1]	–	99.0	97.2
Dense-IndRNN (257 k) [28]	–	**99.48**	97.2
Self-Attention (500 k) [25]	62.2	98.9	97.9
GTCN (50 k)	70.11	99.3	98.1
TrellisNet (8 m) [29]	73.42	99.2	98.13
CKCNN-Big (1 m) [10]	63.74	99.32	98.54
TCN (300 k) [1]	70.66	99.39	98.44
GTCN (300 k)	**74.37**	99.42	**98.69**

Highest accuracy is shown with bold
Number of parameters for each architecture is shown in parentheses

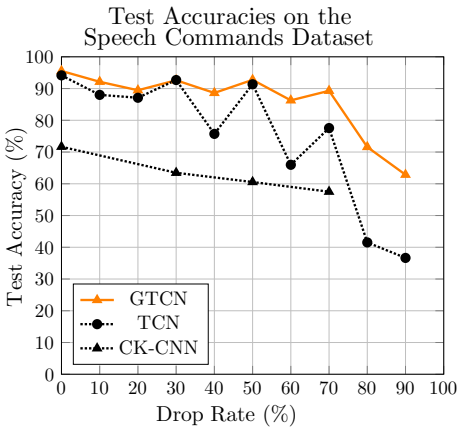


Fig. 4 Accuracy (%) scores on the Speech Commands dataset with respect to drop rates. Accuracy scores of CK-CNN is from [10]

sequence is dropped, we append the sampling interval information as an additional channel to the input sequence and fed the resulting sequences to the models. We normalize the sampling intervals using the mean and the standard deviation of the sampling intervals in the training set. The TCN and GTCN algorithms are set to have 12 layers and the kernel size of 8 when the sequences are observed entirely. As shown in Fig. 4, both the TCN and the GTCN architectures are able to model the signals when the drop rate is less than 50%, while the GTCN outperforms other architectures as the drop rate is increased. Due to having stable gradients even with the very deep networks, e.g., the GTCNs used in the experiments have at least 22 hidden layers, and being able to model the additive and scaling effects of the sampling intervals, the GTCN offers more balance in terms of modeling long-term dependencies and sampling irregularities together.

5 Conclusion

We have studied sequential modeling of signals and introduced a novel gating mechanism to be used on the basic TCN [1] architectures. The GTCN architecture is able to model the long-term dependencies and the irregularities in the sampling intervals in the sequential data. By this gated architecture, we avoid dying activation functions and alleviated the vanishing or exploding gradient problem by altering the way of gradient flow. The GTCN algorithm offers a balance between modeling very long-term dependencies and sampling irregularities. Through extensive set of experiments, we demonstrate that the basic GTCN architecture is capable of modeling long-term dependencies even when the sequences have 16,000 observations and able to handle the sequences with missing values even when the 90% of the sequences are dropped. We achieve significant performance gain in various well-known benchmarks and real-life applications compared to the state-of-the-art results.

Acknowledgements This study is partially supported by Turk Telekom within the framework of 5G and Beyond Joint Graduate Support Programme coordinated by Information and Communication Technologies Authority and Tubitak Contract No: 117E153.

References

1. Bai, S., Kolter, J.Z., Koltun, V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint (2018)
2. Machado, J.T., Duarte, F.B., Duarte, G.M.: Analysis of financial indices by means of the windowed Fourier transform. SIViP **6**(3), 487–494 (2012)
3. Koç, E., Türkoğlu, M.: Forecasting of medical equipment demand and outbreak spreading based on deep long short-term memory network: the covid-19 pandemic in Turkey. SIViP **16**, 613–621 (2021)
4. Fortunati, S., et al.: An improvement of the state-of-the-art covariance-based methods for statistical anomaly detection algorithms. SIViP **10**(4), 687–694 (2016)
5. Singer, A.C., Wornell, G.W., Oppenheim, A.V.: Nonlinear autoregressive modeling and estimation in the presence of noise. Digit. Signal Process. **4**(4), 207–221 (1994)

6. Elman, J.L.: Finding structure in time. Cogn. Sci. **14**(2), 179–211 (1990)
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
8. Cho, K., van Merrienboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: encoder–decoder approaches, pp. 103–111 (2014)
9. Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y.: Deep Learning, vol. 1. MIT Press, Cambridge (2016)
10. Romero, D.W., Kuzina, A., Bekkers, E.J., Tomczak, J.M. Hoogendoorn, M., Ckconv: Continuous kernel convolution for sequential data. arXiv preprint arXiv:2102.02611 (2021)
11. Khan, N.A., Butt, N.R. Jakobsson, A.: Iterative missing data recovery algorithm for non-stationary signals. SIViP 1–8 (2022)
12. Lipton, Z.C., Kale, D.C., Elkan, C. Wetzel, R.C.: Learning to diagnose with LSTM recurrent neural networks (2016)
13. Shi, X., et al.: Convolutional LSTM network: a machine learning approach for precipitation nowcasting, pp. 802–810 (2015)
14. van den Oord, A., et al.: WaveNet: a generative model for raw audio, vol. 125. ISCA, Kolkata (2016)
15. Dauphin, Y.N., Fan, A., Auli, M., Grangier, D.: Language modeling with gated convolutional networks. In: International Conference on Machine Learning, pp. 933–941. PMLR (2017)
16. Nanni, L., Lumini, A., Manfe, A., Brahnam, S., Venturin, G.: Gated recurrent units and temporal convolutional network for multilabel classification. arXiv preprint arXiv:2110.04414 (2021)
17. Sahin, S.O., Kozat, S.S.: Nonuniformly sampled data processing using LSTM networks. IEEE Trans. Neural Netw. Learn. Syst. **30**(5), 1452–1461 (2018)
18. Lechner, M., Hasani, R.M.: Learning long-term dependencies in irregularly-sampled time series . arXiv preprint arXiv:2006.04418 (2020)
19. Chang, S., et al.: Dilated recurrent neural networks. In: Guyon, I., et al. (eds.) NeurIPS Proceedings, pp. 77–87 (2017)
20. Hahnloser, R.H., Sarpeshkar, R., Mahowald, M.A., Douglas, R.J., Seung, H.S.: Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. Nature **405**(6789), 947–951 (2000)
21. Lu, L., Shin, Y., Su, Y., Karniadakis, G.E.: Dying relu and initialization: theory and numerical examples. arXiv preprint arXiv:1903.06733 (2019)
22. Yudistira, N., Kurita, T.: Gated spatio and temporal convolutional neural network for activity recognition: towards gated multimodal deep learning. EURASIP J. Image Video Process. **2017**(1), 1–12 (2017)
23. Hong, S., Wang, C., Fu, Z.: Gated temporal convolutional neural network and expert features for diagnosing and explaining physiological time series: a case study on heart rates. Comput. Methods Prog. Biomed. **200**, 105847 (2021)
24. Dai, P., Ji, S., Zhang, Y.: Gated convolutional networks for cloud removal from bi-temporal remote sensing images. Remote Sens. **12**(20), 3427 (2020)
25. Trinh, T.H., Dai, A.M., Luong, T., Le, Q.V.: Learning longer-term dependencies in RNNs with auxiliary losses. In: Proceedings of Machine Learning Research, vol. 80, pp. 4972–4981. PMLR (2018)
26. Le, Q.V., Jaitly, N., Hinton, G.E.: A simple way to initialize recurrent networks of rectified linear units. arXiv preprint arXiv:1504.00941 (2015)
27. Warden, P.: Speech commands: a dataset for limited-vocabulary speech recognition. arXiv preprint arXiv:1804.03209 (2018)
28. Li, S., Li, W., Cook, C., Gao, Y., Zhu, C.: Deep independently recurrent neural network (IndRNN). arXiv preprint arXiv:1910.06251 (2019)
29. Bai, S., Kolter, J.Z., Koltun, V.: Trellis Networks for Sequence Modeling. OpenReview.net, Toronto (2019)
30. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
31. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
32. Kidger, P., Morrill, J., Foster, J., Lyons, T.J.: Neural controlled differential equations for irregular time series. NeurIPS Proceedings, pp. 6696–6707 (2020)