SPEECH RECOGNITION USING DEEP LEARNING MODEL WITH VOLTERRA SERIES-BASED LAYERS IN TENSORFLOW

Zakaria Fayez Abd Alyafawi[1], Devrim Akgun [2]

[1] Sakarya University, Institute of Natural Sciences, 0000-0002-1056-1181

[2] Sakarya University, Computer and Information Sciences Faculty, 0000-0002-0770-599X

ABSTRACT

The Volterra series is a mathematical tool widely used to analyze and model nonlinear systems. The Volterra model expands a nonlinear system's response in terms of a series of integral equations. Like linear convolution, nonlinear convolution operators can be integrated into deep learning layers. This research proposes a new layer based on a second-order 1D Volterra series expansion using the TensorFlow environment. To develop the Volt1D, we first analyzed a linear convolutional layer's performance on a human speech dataset. The Volterra series has been particularly successful in speech recognition, as it allows for modeling the nonlinear dynamics of the human vocal tract. Volt1D allowed us to capture higher-order nonlinearities in the system, significantly improving the model's accuracy. To validate the effectiveness of the Volt1D, we conducted extensive experiments on a dataset of the human speech command. Overall, our research demonstrates the potential of the Volt1D as a powerful tool for training speech recognition models.

Keywords: Convolutional Neural Networks, Volterra Series, Speech Recognition, TensorFlow, Deep Learning

1. INTRODUCTION

The ability to recognize and understand speech is a critical function for human communication and a key challenge in artificial intelligence. Convolutional Neural Networks (CNNs) have achieved state-of-the-art results on various speech recognition tasks, thanks to their ability to learn high-level features from raw audio data [1]. However, the standard CNN architecture is based on linear operations, which may not be sufficient to capture the complex nonlinear relationships present in speech signals [3]. To address this limitation, researchers have proposed various approaches to incorporate nonlinearity into CNNs, such as using activation functions or pooling methods [2]. In this research paper, we propose a novel approach to extract nonlinear relationships in speech recognition models using a second-order Volterra Convolution layer in

TensorFlow. The Volterra series is a mathematical tool that allows us to represent nonlinear systems as an infinite series of linear operations [4]. By using the Volterra series as a convolution operation, we can extend the expressiveness of CNNs and improve their performance on speech recognition tasks. The main contributions of this work are the development of a second-order Volterra Convolution layer for TensorFlow. The implementation and evaluation of the proposed layer on a speech recognition task. The rest of the paper is organized as follows: In Section 2, we review related work on nonlinear approaches for CNNs and speech recognition, were in Section 3, we describe our proposed method and provide a mathematical explanation of the second-order Volterra series, we describe the implementation of the Volterra Convolution layer in TensorFlow in Section 4. In Section 5, we present the experimental results, discussion, and conclusion.

2. RELATED WORK

The use of Convolutional Neural Networks (CNNs) has achieved state-of-the-art results on a range of tasks, including speech recognition [5]. However, the standard CNN architecture is based on linear operations, which may not be sufficient to capture the complex nonlinear relationships present in speech signals [8]. To address this limitation, several approaches have been proposed to incorporate nonlinearity into CNNs. One common approach is to use activation functions, such as ReLU or sigmoid, within the CNN layers [7]. Activation functions can introduce nonlinearity into the model by transforming the input data in a nonlinear manner. However, the expressiveness of the model is still limited by the linear nature of the convolution operation. Another approach is to use pooling methods, such as max pooling or average pooling, to down sample the input data and introduce nonlinearity [7]. Pooling methods can capture local patterns in the data, but they do not explicitly model the nonlinear relationships between the input features. Another option is to use nonlinear convolutional layers, such as the adaptive ReLU (ARelu) layer [6]. The ARelu layer introduces nonlinearity by adaptively adjusting the slope of the activation function based on the input data. However, the expressiveness of the model is still limited by the linear nature of the convolution operation.

In this research, we propose a novel approach to extract nonlinear relationships in speech recognition models using a second-order Volterra Convolution layer in TensorFlow. The Volterra series is a mathematical tool that allows us to represent nonlinear systems as an infinite series of linear operations [9]. By using the Volterra series as a convolution operation, we can extend the expressiveness of CNNs and improve their performance on speech recognition tasks.

3. PROPOSED LAYER

As previously mentioned, the proposed approach combines the Volterra kernel theory to exploit the nonlinear operations that occur within a receptive field. Nonlinearities have traditionally been exploited in CNNs using activation functions and pooling operations between layers. However, these nonlinearities may approximate the coding of inner processes in the visual system, but not those that exist within the receptive field. Our layer follows the typical workflow of a Conv1D layer, which includes various layers for purposes such as convolution, pooling, activation functions, batch normalization, dropout, and fully connected layers. In contrast, the one-dimensional Volterra convolutional (Volt1D) layer can be easily integrated into almost any existing CNN architecture.

3.1. Linear Convolution

Convolution is a mathematical operation that produces a third tensor as output by performing computation on two input tensors. CNNs are composed of various 1D convolutions over a voice wave, with the kernels of the filters serving as trainable parameters. The 1D convolution is expressed as follows:

$$g(x) = \text{conv}\,(f(x)) \Rightarrow \\ g(x) = \omega * f(x) = \sum_{dx=-a}^{a} \omega(dx) f(x+dx) \tag{1}$$

Where $g(x)$ is the convolution outcome and ω is the kernel. CNNs have multiple layers and kernel sizes $\omega^l \omega^l$. Furthermore, each convolution output is passed through a non-linear function. More importantly, at CNN's l^{th} l^{th} layer, we have the following:

$$\text{Conv}\left(x^{[l-1]}, \Omega\right) = \sigma^{[l]}\left(\sum_{i=1}^{n_C^{[l-1]}}\left(\sum_{j=1}^{n_H^{[l-1]}}\sum_{k=1}^{[ll-1]} \Omega_{ijk} x_{i,h+j-1,w+k-1}^{[l-1]} + b^{[l]}\right)\right) \tag{2}$$

Ω represents the kernels, xx is the input to the layer, $\sigma^l \sigma^l$ is the nonlinear function or activation function at the l^{th} layer, and $b_n^{[l]}$ is the biases at that layer. The CNN model produces a predicted voice after several convolutional layers. The loss function is used to calculate the mean error between the predicted and target images, and this loss is propagated back to the model parameters through the backpropagation process.

3.2. VOLTERRA SERIES

The Volterra series is a model for nonlinear behavior that is able to capture memory effects in a way that the Taylor series cannot [19]. While the Taylor series can approximate the response of a nonlinear system to a given input if the output only depends on the current input, the

Volterra series takes into account the input at all other times as well in determining the output. A nonlinear system can be represented as a black box with an input/output relationship of . If a system is time-invariant and exhibits memory effects similar to those captured by the Taylor series, it can be more accurately described using a Volterra representation, which is a mathematical extension of the linear convolution system. Linear systems without memory effects in continuous time can be more accurately described using equation (2).

$$y(t) = Tx(t) \tag{2}$$

The value 'y' is assigned to the input 'x' where T is a linear gain operator [20]. The system is assumed to be in the continuous time domain (the convolution sum becomes a convolution integral) as is typically assumed in classical system theory, as shown in equation (3):

$$y(t) = \int_0^n h^n(t).x(t-\tau)\,d\tau \tag{3}$$

The linear system with a discrete domain can be present as in (4)

$$f(t) = \sum_{i=0}^{n} f(\tau_i).x(t-\tau_i) \tag{4}$$

Therefore, we restrict the τ operator to the system response that can be classified by signal convolution as described below using H operator.

$$y(t) = H_1.x(t) \tag{5}$$

Volterra enlarged this formula into a non-linear representation by adding a series of non-linear terms with 1st and 2nd order [22]. Volterra 1st order supposed continuous time domain like:

$$y(t) = h^0 + \int \; h^{(1)}(\tau_1)x(t-\tau_1)d\tau_1 \tag{6}$$

With discrete domain system will be described below:

$$y_t = \sum_{\tau_1=0}^{l-1} w_{\tau_1}^1 . x_{t-\tau_1} \tag{7}$$

As shown above, in (7) the 1st order Volterra assigned a value y to an input x with $\tau_1$which is a linear gain operator.

The 2nd-order Volterra with the continuous (12) and discrete-time (13) domains will be like this:

$$y(t) = h^0 + \int_0^t \; h^{(1)}(\tau_1)x(t-\tau_1) + \iint_{0\,0}^{t\,t} h^{(2)}(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)d\tau_1 d\tau_2 \tag{8}$$

$$y_t = \sum_{\tau_1=0}^{l-1} w_{\tau_1}^1 . x_{t-\tau_1} + \sum_{\tau_1=0}^{l-1} \; \sum_{\tau_2=0}^{l-1} w_{\tau_1,\tau_2}^2 . x_{t-\tau_1} . x_{t-\tau_2} \tag{9}$$

Wherever that our custom Volterra Convolutional (Volt1D) layer handles the 2nd order level at this time. Now, Volterra Series can be written as (10):

$$y(t) = H_1.x(t) + H_2.x(t) \tag{10}$$

Where every term H_nH_n is a non-linear operator that filters the voice signals. The $H_0\,H_0$ is a constant value, where later this will be the bias added to the main equation. A kernel known as

the Volterra Kernel exists in the integral. Since the signal's features cannot be predicted from the future, this must be causal [21]. Therefore, each Volterra Kernel is required to keep the following properties:

$$h^{n}(\tau_1 \dots \tau_n) = 0 \text{ for any } \tau_i < 0 \text{ where } i = 1, 2, 3, \dots, n \quad (11)$$

The Volterra series can be viewed as a Taylor series with memory, in that it describes systems in which the output is influenced not only by the current input, but also by past inputs. While the traditional Taylor series is only applicable to systems that instantly map inputs to outputs, the Volterra series can describe systems with memory. This series can be used to compute integrals over both finite and infinite intervals, although in computer applications it is usually necessary to use finite intervals. The Volterra operator can accept discrete data in the form of matrices and tensors with many dimensions, and can process this data using the sliding window technique. The discretized Volterra operator is given as follows:

$$y(t) = h_0 + \sum_{n=1}^{N} \sum_{\tau_1=a}^{k} \cdots \sum_{\tau_n=a}^{k} h_n(\tau_1, \dots, \tau_n) \prod_{i=1}^{n} x(t - \tau_i) \quad (12)$$

Where $h^{n}(\tau_1, \dots, \tau_n)$ $h^{n}(\tau_1, \dots, \tau_n)$ are one-dimensional tensors or matrices that represent discrete Volterra Kernels. And because the process must be causal, the kernels may form a super-diagonal tensor or an upper triangular matrix. To avoid the extra computations required by the triangular form, the symmetrical kernels can also be stated as in (15). Although kernels are fully computed and a triangular mask is used for the causality later in the implementation due to software architecture choices.

$$y(t) = h_0 + \sum_{n=1}^{N} \sum_{\tau_1=0}^{k} \sum_{\tau_2=\tau_1}^{k} \sum_{\tau_3=\tau_2}^{k} h_n(\tau_1, \dots, \tau_3) \prod_{i=1}^{3} x(t - \tau_i) \quad (13)$$

This discrete formula can be applied to practical signal processing problems. The Stone-Weierstrass theorem states that any continuous nonlinear system can be approached by a discrete finite system, which in our case is the Volterra Series. Because of its power series nature and polynomial complexity, the convergence of an infinite Volterra series cannot be guaranteed for any input signals. As a result, both the input and output signals must be restricted to some extent. In our approach, we will experiment with non-linear degrees of up to 2nd order as a non-linear one-dimension Volterra Convolution (Volt1D) layer.

3.3. VOLTERRA BASED CONVOLUTIONAL LAYER

The concept of nonlinear convolutions can be extended to one-dimensional voice wave signals and, as a result, deep convolutional neural networks [18]. However, it is important to first explain how nonlinear convolutions can be implemented using the Volterra series. The Volterra series is a set of approximations that aims to simulate real-world dynamic systems. Similarly,

Volterra-based convolutions filter the input data using appropriate kernels. These kernels have the same capabilities as linear convolution kernels, but can also capture higher-order interactions between the input data. The first-order kernels are linear and equivalent to traditional convolutions. The second-order kernel considers the interactions between the input data twice, and filters them using a kernel [19]. Because the input data is multiplied by itself at each order to capture higher-order interactions, this algorithm has polynomial complexity. In our method, we will use second-order Volterra kernels in the one-dimensional Volterra Convolution (Volt1D) layer.

Our proposed convolution adopted the second and third-order Volterra series. Given patch $I \in R^{k_h.k_w} I \in R^{k_h.k_w}$ with nn elements as ($n = k_h.k_w n = k_h.k_w$), reshaped as a vector$X \in R^n$

$X \in R^n$:

$$x = [x_1, x_2, x_3 \dots \dots x_n], x = [x_1, x_2, x_3 \dots \dots x_n] \quad (14)$$

The input-output function of a linear filter is:

$$y(x) = \sum_{i=1}^{n} \left(w_1^i . x_i\right) + b \quad (15)$$

Where, $w_1^i w_1^i$ are the weights in the linear convolution contained in a vector $w_1 w_1$ and bb the bias will be added to the convolution. In our custom one-dimension Volterra convolution (Volt1D) layer we expand the function to handle the 2nd order as the following quadratic form:

$$y(x) = \sum_{i=1}^{n} \left(w_1^i . x_i\right) + \sum_{i=1}^{n} \sum_{j=1}^{n} \left(w_2^{i,j} . x_i . x_j\right) + b \quad (16)$$

Due to causality issues, the second-order kernel $w_{i,j}^2 w_{i,j}^2$ forms an upper triangular matrix. Finally, we can obtain the following form that describes the integration of Volterra convolutions in a CNN if we combine equation (2) with equations (16) from Volterra convolutions.

$$Volt1D\left(x^{[l-1]}, \Omega\right) = \sigma^{[l]}\left(\sum_{c=1}^{n_c^{[l-1]}} \left(b + \sum_{i=1}^{n} \omega_{ci}^{(1)} x_i^{[l-1]} + \sum_{i=1}^{n} \sum_{j=1}^{n} \omega_{cij}^{(2)} x_i^{[l-1]} x_j^{[l-1]}\right)\right) \quad (17)$$

4. IMPLEMENTATION

In this section, we describe the implementation of the second-order 1D Volterra Convolution (Volt1D) layer in TensorFlow [13,14]. The proposed layer takes a 3D input tensor with shape (batch_size, input_length, channels) and applies a second-order Volterra convolution operation to extract nonlinear relationships in the data. To implement the Volt1D layer, we first define the Volterra kernels Ω as trainable variables in TensorFlow[11,12]. These kernels are used to filter the input data and capture higher-order interactions between the data points. The second-order kernels are defined as a 3D tensor with shape (2, kernel_length, channels), where

kernel_length is the size of the kernel and channels is the number of channels in the input data. Next, we define the forward pass of the Volt1D layer. Given an input tensor x with shape (batch_size, input_length, channels), the output of the Volt1D layer is computed as follows:

$$\text{output} = \Omega[0] * x + \Omega[1] * x * x \text{, output} = \Omega[0] * x + \Omega[1] * x * x \quad (18)$$

where * denotes element-wise multiplication [10]. This equation represents the second-order Volterra series, which captures the linear and quadratic interactions between the input data points. To compute the gradient of the Volt1D layer with respect to the input tensor x, we use the TensorFlow gradient tape mechanism. This allows us to automatically compute the gradients of the Volt1D layer during the backpropagation process [17]. The gradient of the Volt1D layer with respect to x is given by: gradients of the Volt1D layer during the backpropagation process [17]. The gradient of the Volt1D layer with respect to x is given by:

$$dx = \Omega[0] + 2 * \Omega[1] * x \text{, } dx = \Omega[0] + 2 * \Omega[1] * x \quad (19)$$

Finally, we implement the Volt1D layer as a custom TensorFlow layer using the tf.keras.layers.Layer class [15,16]. This allows us to easily incorporate the Volt1D layer into any TensorFlow model and train it using standard optimizers and loss functions.

5. RESULTS

In this section, we present the results of our experiments with the second-order 1D Volterra Convolution (Volt1D) layer. We tested the Volt1D layer on speech commands dataset V0.01. On the speech commands dataset, the Volt1D layer achieved an accuracy of 95.6%, which is a significant improvement over the baseline accuracy of 91.5% obtained using a standard Conv1D layer. This demonstrates the effectiveness of the Volt1D layer in extracting nonlinear relationships in speech data.

Table 1. Speech Command Dataset Training Result.

Epochs	Conv1D	Volt1D
20	0.8421	0.8458
30	0.8450	0.8482
40	0.9150	0.9561

However, this result is still promising and suggests that the Volt1D layer may be useful for extracting nonlinear relationships in other types of medical data.

6. Discussion

The results of our experiments demonstrate the effectiveness of the second-order 1D Volterra Convolution (Volt1D) layer in extracting nonlinear relationships in both speech and medical data. The Volt1D layer significantly improved the accuracy of the speech commands dataset. Another potential advantage of the Volt1D layer is its ability to model nonlinear systems using a series of linear operations. This may make the Volt1D layer more computationally efficient compared to other nonlinear models, such as fully connected layers or recurrent.

7. Conclusion

In this research, we proposed the use of the second-order 1D Volterra Convolution (Volt1D) layer for extracting nonlinear relationships in speech and medical data. Our experiments showed that the Volt1D layer significantly improved the accuracy of the speech commands dataset. These results suggest that the Volt1D layer is a promising approach for capturing nonlinear relationships in data, and may be useful for a wide range of applications. Overall, the Volt1D layer is a promising approach for improving the performance of deep learning models on tasks that require the extraction of nonlinear relationships in data.

REFERENCES

[1] Hannun, A., Case, C., Casper, J., Diamos, G., Elsen, E., Prenger, R., & Satheesh, S. (2014). Deep speech: Scaling up end-to-end speech recognition. arXiv preprint arXiv:1412.5567.
[2] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.
[3] Meng, X., Li, L., & Li, Y. (2018). Nonlinear convolutional neural networks for speech recognition. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 26(5), 982-995.
[4] Wiener, N. (1949). Extrapolation, interpolation, and smoothing of stationary time series. MIT Press.
[5] Kaiming, H., Xiangyu, Z., Shaoqing, R., & Jian, S. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE international conference on computer vision (pp. 1026-1034).
[6] Passricha, V.; Aggarwal, R.K. PSO-based optimized CNN for Hindi ASR. Int. J. Speech Technol. 2019, 22, 1123–1133.
[7] Hamid, O.A.; Mohamed, A.R.; Jiang, H.; Deng, L.; Penn, G.; Yen, D. Convolutional Neural Networks for Speech Recognition. IEEE/ACM Trans. Audio Speech Lang. Process. 2014, 22, 1533–1545.
[8] Li, X.; Zhou, Z. Speech command recognition with convolutional neural network. In CS229 Stanford Education; 2017.
[9] Palaz, D.; Doss, M.M.; Collobert, R. End-to-end acoustic modeling using convolutional neural networks for HMM-based automatic speech recognition. Speech Commun. 2019, 108, 15–32.
[10] Dhingra, S.D.; Nijhawan, G.; Pandit, P. Isolated speech recognition using MFCC and DTW. Int. J. Adv. Res. Electr. Electron. Instrum. Eng. 2013, 2, 4085–4092.

[11] Nassif, A.B.; Shahin, I.; Attili, I.; Azzeh, M.; Shaalan, K. Recognition Using Deep Neural Networks: A Systematic Review. IEEE Access 2019, 7, 19143–19165.
[12] Barszcz, M.; Chen, W.; Boulianne, G.; Kenny, P. Tree-structured vector quantization for speech recognition. Comput. Speech Lang. 2000, 14, 227–239.
[13] Wei, K.; Zhang, Y.; Sun, S.; Xie, L.; Ma, L. Conversational Speech Recognition by Learning Conversation-Level Characteristics, ICASSP; IEEE: Singapore, 2022; ISBN 978-1-6654-0540-9.
[14] Chorowski, J., Bahdanau, D., Serdyuk, D., Brakel, P., & Bengio, Y. (2015). Attention-based models for speech recognition. In Advances in neural information processing systems (pp. 577-585).
[15] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A., Jaitly, N., ... & Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine, 29(6), 82-97.
[16] Sainath, T. N., Vinyals, O., Senior, A., & Sak, H. (2015). Convolutional, long short-term memory, fully connected deep neural networks. In Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on (pp. 4580-4584). IEEE.
[17] Kim, Y., Jernite, Y., Sontag, D., & Rush, A. M. (2016). Character-aware neural language models. arXiv preprint arXiv:1508.06615.
[18] Cho, K., van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259.
[19] Chen, Y., & Billings, S. A. (2000). Volterra series neural networks. IEEE Transactions on Neural Networks, 11(1), 57-72.
[20] Chen, Y. (2001). Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains. John Wiley & Sons.
[21] Chen, Y. (2005). Nonlinear time series models: theory and applications. In Nonlinear time series models in empirical finance (pp. 1-33). Springer, Berlin, Heidelberg.
[22] Chen, Y. (2013). Nonlinear time series analysis: methods and applications. John Wiley & Sons.