Application of Toeplitz Matrix for Audio Smoothening

Abdul-Ganiy B. Usman
African Masters in Machine Intelligence
African Institute for Mathematical Sciences
Kigali, Rwanda.
ausman@aimsammi.org

Ernest Fokoue
Rochester Institute of Technology
New York, USA.
epfeqa@rit.edu

Abstract

Data preprocessing is not well regarded when using deep neural networks for training. Many often throw in inputs to model architecture and hope to get desirable outcome. In this work, we reimplemented Dai et al. proposed architectures for acoustic raw waveform and propose a prior preprocessing with toeplitz matrix. We revealed better performance with the preprocessing step. Our computations shows high reliance on computational memory and speed.

1 Introduction

Deep neural networks have made tremendous breakthrough in predictive analysis. It all started as a simple perceptron (2), a one unit neural network, to a multilayer perceptron. These types of networks allow movement of informations in one direction from the input to the output - feedforward neural networks. However, the network achieve a proper learning procedure through weights update during training known as back-propagation algorithm (17). In the case of unstructured data or data known to have grid-like topology, such as images, the networks have been proven to exhibit poor generalization performance (13). Convolutional neural networks, or CNNs, rather perform better in such cases (13).

CNNs overtime have been used majorly on images with its popular application and breakthrough on ImageNet (10) and MNIST digit (11). These are three dimensional data—depth, height and width—in their tensor representation. There has been recent outburst of CNN applications in acoustic modelling, which traditionally is in two categories: (1) feature representation of the audio data, and (2) predictive model building from such representation. The predictive task might often be suboptimal if the right representation, which is quite challenging, is not achieved. Many deep network models have made tremendous performance in speech learning and modelling (3).

A key requirement in modelling is how to effectively transform the input into a good representation to extract reasonable information from it. This is what is known as the preprocessing. Many real life data are noisy and can cause performance degradation for any type of model. Deep neural networks easily overfit to label noise data. Many methods to overcome this problem have been proposed one of which is constructing noise-tolerant training algorithm (15). A different approach is using the relationships between data features for cleaning to produce *probabilistic errors* and *probabilistic fixes* using low rank approximation (21). Preprocessing can also be devising techniques to combat large input space, "curse of dimensionality". Dimensionality reduction, while approximately preserving essential geometric properties, can be done using a number of mathematical tools, toeplitz matrix being an example (4). Toeplitz matrices have had their applications in compress sensing as a tool to recover high-dimensional sparse signals from relatively few linear observations (6).

2 Background

This section describes briefly the implementation and training approach of the architectures presented by Dai et al. in their paper along with some of the training choices we will examine experimentally.

2.1 Data

The data trained upon is the UrbanSound8k dataset. It contain a total of 8732 labelled sound data (each of length $\leq$ 4s) in ten different folds. The dataset contains 10 classes: air_conditioner, car_horn, children_playing, dog_bark, drilling, enginge_idling, gun_shot, jackhammer, siren, and street_music (18).

2.2 Setup

Each architecture takes as input a one channel 1D tensor of size 32,000. The dataset originally contains unequal audio files with well over 100,000 samples. The UrbanSound8K audio has a fixed sample rate of 44.1kHz, therefore 32,000 samples covers about 700 milliseconds. Downsampling the audio to approximately 8kHz, we get 4 seconds has a representation for the 32,000 samples. The downsampling is well achieved by taking every fifth sample of the original audio tensor, with padding applied to audio tensor not long enough to handle the downsampling. The minimum length that won't require padding is 160,000 samples. The dataset uses stereo, a two channels audio, so converting to a mono is done by taking mean over the channel.

2.3 Architecture

The architectures considered in the original paper (3) are basically five: M3, M5, M11, M18, M34-res, the last three having their parameters in millions. Table 1 briefly summarises the architectures build-up.

M3(0.2M)	M5(0.5M)	M11(1.8M)	M18(3.7M)	M34-res(4M)
Input: 32000 × 1 time-domain waveform				
[80/4, 256]	[80/4, 128]	[80/4, 64]	[80/4, 64]	[80/4, 48]
Maxpool: 4 × 1 (output: 2000 × n)				
[3, 256]	[3, 128]	[3, 64] × 2	[3, 64] × 4	$\begin{bmatrix} 3, 48 \\ 3, 48 \end{bmatrix} \times 3$
Maxpool: 4 × 1 (output: 500 × n)				
	[3, 256]	[3, 128] × 2	[3, 128] × 4	$\begin{bmatrix} 3, 96 \\ 3, 96 \end{bmatrix} \times 4$
	Maxpool: 4 × 1 (output: 125 × n)			
	[3, 512]	[3, 256] × 3	[3, 256] × 4	$\begin{bmatrix} 3, 192 \\ 3, 192 \end{bmatrix} \times 6$
	Maxpool: 4 × 1 (output: 32 × n)			
		[3, 512] × 2	[3, 512] × 4	$\begin{bmatrix} 3, 384 \\ 3, 384 \end{bmatrix} \times 3$
Global average pooling (output: 1 × n)				
Softmax				

Table 1: Model architectures

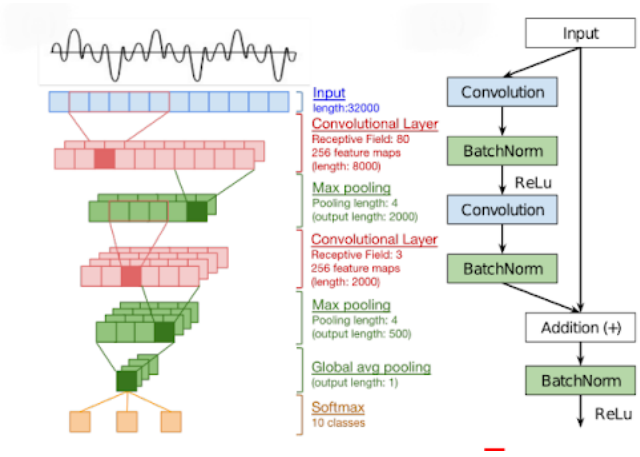


Figure 1: Pictorial representation of the model architectures

Table 1 represents the proposed fully convolutional network architectures for time-domain waveform inputs. M3 (0.2M) denotes 3 weight layers and 0.2M parameters. [80/4, 256] denotes a convolutional layer with 256 filters of size 80 and stride 4. [3, 256] has stride 1. [...] $\times k$ denotes k stacked layers. Double layers in a bracket denotes a residual block and only occur in M34-res, though we didn't consider this in our implementation. Output size after each pooling is written as m × n where m is the size in time-domain and n is the number of feature maps and which can vary across architectures. All convolutional layers are followed by batch normalization layers, omitted to avoid clutter (3).

The figure 1 further explains how the architectures build-up looks like. It clearly reveals how the input signal, represented by single feature map, is encoded by the convolutional and pooling filters at every layer. Also, the resblock formation is depicted.

2.4 Training Objective

A major problem in training deep architectures is the computational cost, which is quite difficult to accurately predict (8). Stacking many layers is quite necessary also to abstract useful information from the features. A trade-off between computational expense and high level feature abstraction, through stacking many layers, is using large receptive field in the first layer. This is followed by a very small receptive field (of 3) in the subsequent layers. Though, there has been a thorough investigation on poor performance of much smaller (19) or larger receptive field before arriving on a suitable one. In addition, the feature resolution is largely reduced by using max pooling strides limiting computational cost in the rest of the network. Li et al. in their work validated max pooling as giving a better results than other pooling method. A doubling of the number of feature map is made after the first two layers complementing for the reduction made using the pooling layers. It is a common practice to use two or more fully connected layers, after the convolutional layers, for discriminative modelling in many deep convolutional networks, this in turn results into having more parameters to train. Most learning have, however, been proven to occur in the convolutional layer if well expressed rendering the need of the fully connected layers less necessary (13). As such a fully convolutional setup is adopted followed by a single global average pooling layer that average out the feature map across the temporal dimension. Capitalizing on lowering the cost of computation, the rectified linear units (ReLU) activation is used (5). Another problem that might arise from training very deep networks is the vanishing or exploding gradient problem. Including batch normalization layer after every convolutional block will quite prevent such problem (7).

2.5 Optimization

The experiment is optimized using Adam, for its conditional local convergence property (1), with learning rate of 0.01 and $\mathbb{L}_2$ weight decay of 0.0001.The training starts off first with the specified learning rate, but decreased using a scheduler to 0.001 during training. Since the architectures are fully convolutional no dropout is used.

3 Experimental Setup

This section describes the experimental setup of our replication of Dai et al. paper.

3.1 Implementation

For starters, the implementation procedure for audio classification in the original paper (3) was done following exactly the hyperparameters given in the optimization part of section 2. However, training was done for 20 epochs as opposed to 100 stated in the original paper. This is due to the computing power available to us, as each epoch takes considerable large amount of time. As a result we were not able to get exactly the results presented in the paper revealed in table 2.

Model	Test accuracy	
	Original	Replicate
M3	56.12%	48.07%
M5	63.42%	53.77%
M11	69.07%	58.12%
M18	71.68%	60.95%

Table 2: Test accuracies for models in Table 1 on UrbanSound8k dataset

3.2 Preprocessing

What we do differently is having a preprocessing step before actually training the architectures on the audio dataset. Recent studies revealed a significant reduction in the neural network prediction errors if the input data is appropriately transformed (12). The preprocessing step convolves each and every audio signal with a special type of matrix called toeplitz matrix.

Toeplitz matrix This matrix types are quite useful for image restoration (20) and deblurring (9). The toeplitz matrix is a square matrix $T_{ij} = [t_{i-j} : i, j = 0, 1, 2, \cdots n-1]$, i.e, matrix of the form

$$T_n = \begin{bmatrix} t_0 & t_{-1} & t_{-2} & \cdots & t_{-(n-1)} \\ t_1 & t_0 & t_{-1} & \cdots & t_{-(n-1)} \\ t_2 & t_1 & t_0 & \cdots & t_{-(n-1)} \\ \vdots & & & \ddots & \vdots \\ t_{n-1} & & & \cdots & t_0 \end{bmatrix}$$

We, though, construct the elements of our toeplitz matrix using the quantity

$$T_{ij} = \sqrt{\frac{\alpha}{\pi}} e^{-\alpha(i-j)^2}, \quad i, j = 1, \ldots, N$$

.

The α is a hyperparameter, however, we choose it to be 0.5 for our experiment.

Convolution

The matrix vector multiplication has $\mathcal{O}(N^2)$ complexity. Lee, however, presented a faster algorithm for toeplitz by vector multiplication with cost $\mathcal{O}(N \log N)$. The audio signal has length of 32000, the convolution in the preprocessing step therefore is highly costly. Figure 2 shows the resulting signal, $y_i = \sum_{j=1}^{N} T_{ij} x_j$, represents a smoothened version of the original signal and this is what is used as input to the model.

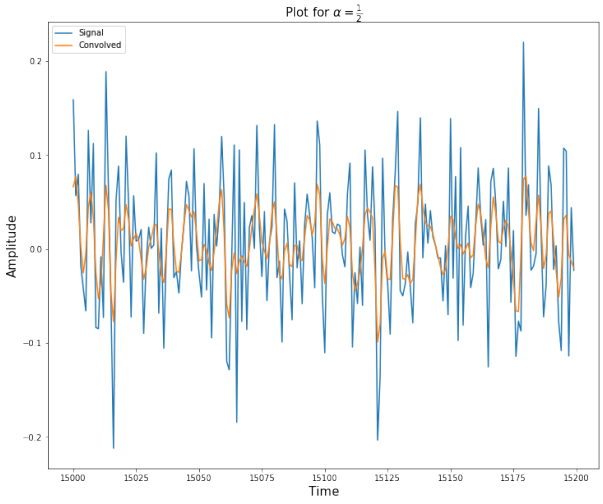


Figure 2: Original signal and convolved signal zoomed in.

3.3 Training Procedure

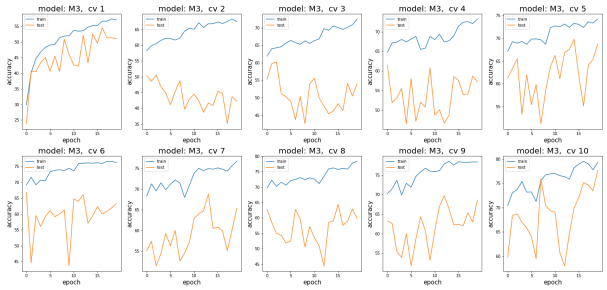
For our training process we keep the model architectures fixed exactly as described in the original paper (3). The models are implemented using the pytorch framework and trained on a server equipped

with CPU only. The training process heavily requires large computational memory and speed, the former we have but lack the later. The matrix vector product in the preprocessing step fits well in memory but takes a lot of time to execute. We, therefore, reduce the training epochs to 10 for each architecture.

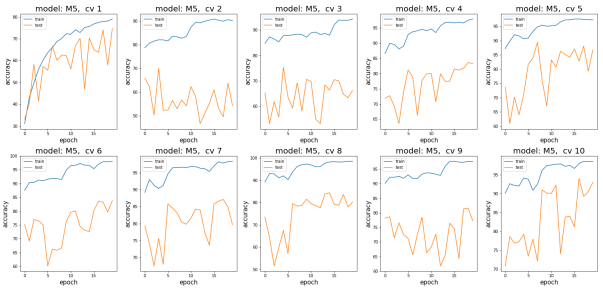
4 Result and Analysis

Figure 3 depicts the accuracy measures of each and every model in sub-figures. Each of the sub-figure 3a, 3b, 3c and 3d shows the learning process over the cross validation sets. All the models learn well with the first 9 folds and give very good performance on the test set (the 10th fold). However, this is not the case for other cross validations set. Their accuracy curves seem to strive poorly.

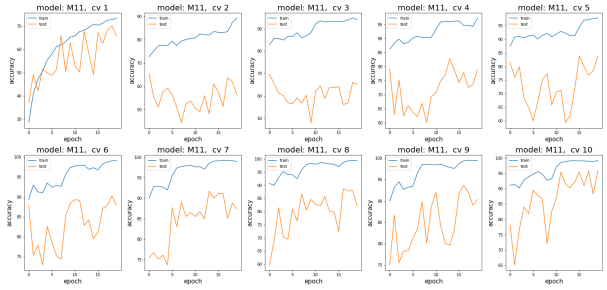
Table 3 summarises the average train and test accuracy over the 10-fold cross validation for each model architecture. A careful examination of the results reveals better learning as we go deeper. However, the accuracy of M18 drastically dropped as it probably requires more training, we could not afford, to learn well. A significant improvement can be seen comparing the test accuracies to the ones obtained in the original paper revealed in table 2. This ascertains the benefit of the preprocessing step with the toeplitz matrix.



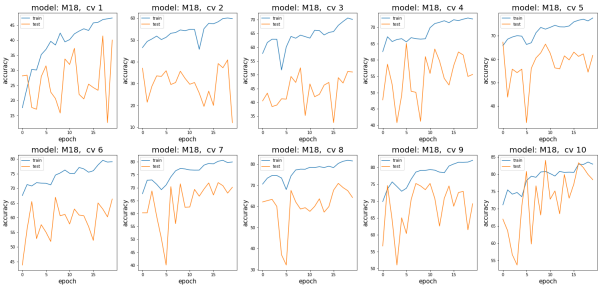
(a) Model M3



(b) Model M5



(c) Model M11



(d) Model M18

Figure 3: Train and test accuracy over the cross validation folds

Model	Accuracy	
	Train	Test
M3	73.36%	60.81%
M5	94.97%	77.95%
M11	94.85%	79.36%
M18	73.31%	56.89%

Table 3: Train and test mean accuracy over the cross validation folds for the first four model architecture in Table 1 on UrbanSound8k dataset

5 Conclusions

We constructed a special type of toeplitz matrix to preprocess the input signal by multiplying it by the toeplitz matrix. Then we train very deep convolutional neural networks (3) on the smoothened acoustic waveform inputs. The networks are optimized without the usual problem of vanishing or exploding gradients associated with deep networks. Our results reveal a better performance with the preprocess step as compare to training the models directly on acoustic waveform inputs. The performance also gets better with model depth except for the deepest model which we could not train for as long as possibly required. The Toeplitz matrix is a promising preprocessor and can be adopted for signal type input provided there is enough computing resources. Our code base implementation can be found here `https://github.com/uman95/AMMI_research`.

References

[1] Sebastian Bock and Martin Weis. [ieee 2019 international joint conference on neural networks (ijcnn) - budapest, hungary (2019.7.14-2019.7.19)] 2019 international joint conference on neural networks (ijcnn) - a proof of local convergence for the adam optimizer. 2019. ISBN 978-1-7281-1985-4. doi: 10.1109/ijcnn.2019.8852239. URL `http://gen.lib.rus.ec/scimag/index.php?s=10.1109/ijcnn.2019.8852239`.

[2] Review by: A. A. Mullin. Principles of neurodynamics.by frank rosenblatt. *American Mathematical Monthly*, 70, 05 1963. doi: 10.2307/2312103. URL `http://gen.lib.rus.ec/scimag/index.php?s=10.2307/2312103`.

[3] Wei Dai, Chia Dai, Shuhui Qu, Juncheng Li, and Samarjit Das. [ieee icassp 2017 - 2017 ieee international conference on acoustics, speech and signal processing (icassp) - new orleans, la, usa (2017.3.5-2017.3.9)] 2017 ieee international conference on acoustics, speech and signal processing (icassp) - very deep convolutional neural networks for raw waveforms. 2017. ISBN 978-1-5090-4117-6. doi: 10.1109/ICASSP.2017.7952190. URL `http://gen.lib.rus.ec/scimag/index.php?s=10.1109/ICASSP.2017.7952190`.

[4] Casper Benjamin Freksen and Kasper Green Larsen. On using toeplitz and circulant matrices for johnson–lindenstrauss transforms. *Algorithmica*, 11 2019. doi: 10.1007/s00453-019-00644-y. URL `http://gen.lib.rus.ec/scimag/index.php?s=10.1007/s00453-019-00644-y`.

[5] K. Hara, D. Saito, and H. Shouno. Analysis of function of rectified linear unit used in deep learning. 2015. doi: 10.1109/IJCNN.2015.7280578. URL `http://gen.lib.rus.ec/scimag/index.php?s=10.1109/IJCNN.2015.7280578`.

[6] J. Haupt, W.U. Bajwa, G. Raz, and R. Nowak. Toeplitz compressed sensing matrices with applications to sparse channel estimation. *IEEE Transactions on Information Theory*, 56, 2010. doi: 10.1109/tit.2010.2070191. URL `http://gen.lib.rus.ec/scimag/index.php?s=10.1109/tit.2010.2070191`.

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei, editors, *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015. URL `http://dblp.uni-trier.de/db/conf/icml/icml2015.html#IoffeS15`.

[8] Daniel Justus, John Brennan, Stephen Bonner, and Andrew Stephen McGough. Predicting the computational cost of deep learning models. *CoRR*, abs/1811.11880, 2018. URL `http://arxiv.org/abs/1811.11880`.

[9] Anoop Kalsi and Dianne P. O'Leary. Algorithms for structured total least squares problems with applications to blind image deblurring. *Journal of Research of the National Institute of Standards and Technology*, 111:113 – 119, 2006/// 2006. URL `http://nvl.nist.gov/pub/nistpubs/jres/111/2/V111.N02.A06.pdfhttp://nvl.nist.gov/pub/nistpubs/jres/111/2/V111.N02.A06.pdf`.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60, 05 2017. doi: 10.1145/3065386. URL http://gen.lib.rus.ec/scimag/index.php?s=10.1145/3065386.

[11] Ernst Kussul and Tatiana Baidyk. Improved method of handwritten digit recognition tested on mnist database. *Image and Vision Computing*, 22, 2004. doi: 10.1016/j.imavis.2004.03.008. URL http://gen.lib.rus.ec/scimag/index.php?s=10.1016/j.imavis.2004.03.008.

[12] Krystyna Kuźniar and Maciej Zając. Some methods of pre-processing input data for neural networks. *Computer Assisted Methods in Engineering and Science*, 22(2):141–151, 2017. URL https://cames.ippt.pan.pl/index.php/cames/article/view/33.

[13] Yann Lecun. *Generalization and network design strategies*. Elsevier, 1989.

[14] David Lee. Fast multiplication of a recursive block toeplitz matrix by a vector and its application. *J. Complex.*, 2(4):295–305, December 1986. ISSN 0885-064X. doi: 10.1016/0885-064X(86)90007-5. URL https://doi.org/10.1016/0885-064X(86)90007-5.

[15] Junnan Li, Yongkang Wong, Qi Zhao, and Mohan S. Kankanhalli. Learning to learn from noisy labeled data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[16] Zhi Li, Shui-Hua Wang, Rui-Rui Fan, Gang Cao, Yu-Dong Zhang, and Ting Guo. Teeth category classification via seven-layer deep convolutional neural network with max pooling and global average pooling. *International Journal of Imaging Systems and Technology*, 05 2019. doi: 10.1002/ima.22337. URL http://gen.lib.rus.ec/scimag/index.php?s=10.1002/ima.22337.

[17] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323, 1986. doi: 10.1038/323533a0. URL http://gen.lib.rus.ec/scimag/index.php?s=10.1038/323533a0.

[18] Justin Salamon, Christopher Jacoby, and Juan Pablo Bello. [acm press the acm international conference - orlando, florida, usa (2014.11.03-2014.11.07)] proceedings of the acm international conference on multimedia - mm '14 - a dataset and taxonomy for urban sound research. 2014. ISBN 9781450330633. doi: 10.1145/2647868.2655045. URL http://gen.lib.rus.ec/scimag/index.php?s=10.1145/2647868.2655045.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. URL http://arxiv.org/abs/1409.1556. cite arxiv:1409.1556.

[20] H. Yu, W. Wu, and Z. Liu. Application of toeplitz matrix in image restoration. In *2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA)*, pages 1240–1244, Sep. 2010. doi: 10.1109/BICTA.2010.5645155.

[21] Zhuoran Yu and Xu Chu. [acm press the 2019 international conference - amsterdam, netherlands (2019.06.30-2019.07.05)] proceedings of the 2019 international conference on management of data - sigmod '19 - piclean. 2019. ISBN 9781450356435. doi: 10.1145/3299869.3320214. URL http://gen.lib.rus.ec/scimag/index.php?s=10.1145/3299869.3320214.