Regularizing Fully Convolutional Networks for Time Series Classification by Decorrelating Filters

Kaushal Paneri,[1] **Vishnu TV,**[2] **Pankaj Malhotra,**[2] **Lovekesh Vig,**[2] **Gautam Shroff**[2]
[1]Northeastern University, Boston, MA, 02115
[2]TCS Research, New Delhi, India
[1]Corresponding Author: paneri.k@husky.neu.edu

Abstract

Deep neural networks are prone to overfitting, especially in small training data regimes. Often, these networks are overparameterized and the resulting learned weights tend to have strong correlations. However, convolutional networks in general, and fully convolution neural networks (FCNs) in particular, have been shown to be relatively parameter efficient, and have recently been successfully applied to time series classification tasks. In this paper, we investigate the application of different regularizers on the correlation between the learned convolutional filters in FCNs using Batch Normalization (BN) as a regularizer for time series classification (TSC) tasks. Results demonstrate that despite orthogonal initialization of the filters, the average correlation across filters (especially for filters in higher layers) tends to increase as training proceeds, indicating redundancy of filters. To mitigate this redundancy, we propose a strong regularizer, using simple yet effective filter decorrelation. Our proposed method yields significant gains in classification accuracy for 44 diverse time series datasets from the UCR TSC benchmark repository.

Introduction

Time Series Classification (TSC) has been a fundamental task in signal processing with applications in multiple domains such as healthcare and finance. Traditionally, distance based methods such as Dynamic Time Warping (DTW) have been employed for TSC tasks. Recently, deep networks have consistently outperformed DTW (Malhotra et al. 2017). In contrast to standard deep networks, Fully Convolutional Networks (FCNs) utilize fewer parameters and have yielded very promising results for TSC tasks (Wang, Yan, and Oates 2017). However, these networks still tend to be overparameterized resulting in redundant parameters after training. Orthogonal Initialization (OI) (Mishkin and Matas 2015) of filters in convolutional layers has also been demonstrated to improve performance compared to random initialization. Further, decorrelation of filters (OrthoReg) (Rodríguez et al. 2016) and network activations (Cogswell et al. 2015) were proposed for regularization of CNNs. Specifically, OrthoReg tries to reduce the redundancy of the filters by minimizing pairwise filter similarities. Leveraging the property of decorrelating filters, we propose FCN with filter decorrelations imposed via a loss function along with the standard classification loss and show that it leads to better classification performance. To the best of our knowledge, this is the first attempt to show the effectiveness of using filter decorrelation loss for time series classification.

Decorrelating Filters in FCNs

Let L be the number of convolutional layers, and p_l be the number of filters in layer l. Let $\mathbf{w}_i^l \in \mathbb{R}^{n_l}$ represent the weights of the i-th filter in the l-th layer. Here $n_l = t_l \times 1 \times p_{l-1}$, where t_l is the length of the filter, p_{l-1} is the number of filters in the $(l-1)$-th layer. The decorrelation loss is given by,

$$\mathcal{L}_{Decorr} = \sum_{l=1}^{L} \sum_{i=1}^{p_l} \sum_{j=i+1}^{p_l} \left(\frac{\mathbf{w}_i^l \cdot \mathbf{w}_j^l}{||\mathbf{w}_i^l||_2 ||\mathbf{w}_j^l||_2}\right)^2$$

where $||.||_2$ denotes the L_2 norm, and $\mathbf{x} \cdot \mathbf{y}$ denotes the dot-product of the vectors $\mathbf{x}$ and $\mathbf{y}$. For a K-class classification problem, let $\mathcal{L}_{CE} = -\sum_{n=1}^{N} \sum_{k=1}^{K} y_n^k \cdot log(\hat{y}_n^k)$, where $\hat{y}_k^n$ denotes the probability of k-th class for n-th training instance, as given by softmax layer with K units, and N is the number of training instances. The total loss is then given by $\mathcal{L} = \mathcal{L}_{CE} + \lambda \mathcal{L}_{Decorr}$.

Experimental Evaluation and Observations

We evaluated our approach on 44 univariate time series datasets taken from publicly available UCR TSC benchmark dataset (Chen et al. 2015). To select the best hyperparameter setting, we held out 25% of training data via stratified

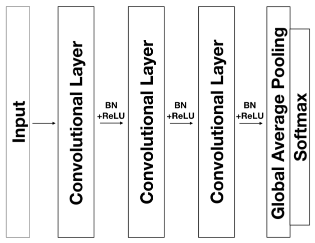

Figure 1: FCN Architecture

sampling. As shown in Fig. 1, FCN model contains 3 convolutional layers with batch normalization (BN) followed by ReLU activation, and ends with Global Average Pooling (Lin, Chen, and Yan 2013) and Softmax layer. With mini-batch size of 10% of the training data, we trained it with Adam (lr= 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$) till 600 epochs with early-stopping tolerance of 30. We then performed grid search over $p_l = [8, 16, 32, 64]$ and $t_l = [0.05T, 0.1T]$, where T is the length of time series.

We considered the following baselines for comparison: i) using *BN*, ii) using *BN* and Dropout (a random subset of filters is dropped at each layer in each iteration of training) (*BN+DO*), iii) using BN and the proposed decorrelation loss (*BN+Decorr*). We also incorporated regularization factors into grid search in the following way: for case ii) dropout factor $= [0.05, 0.15, 0.25]$, and for case iii) $\lambda = [1, 0.1, 0.01, 0.001]$.

To summarize the classification errors of 3 models for 44 datasets, let

$$I = \frac{\#\, training\ instances}{\#\, labels\ in\ training\ data}$$

We group the datasets by I. *Small* contains datasets with $I \leq 11$ (up to the 33^{rd} Percentile), *Medium* contains datasets with $11 < I \leq 33$ ($33^{rd} - 66^{th}$ percentile) and *Large* contains datasets with $I > 33$ (above the 66^{th} percentile).

Table 1: Average Classification error rates on 44 UCR datasets grouped by I

Group	**BN**	**BN+DO**	**BN+Decorr**
Small ($I \leq 11$)	0.2961	0.6050	**0.2509**
Medium ($11 < I \leq 33$)	0.1500	0.49405	**0.1320**
Large ($I > 33$)	0.1522	0.4047	**0.1468**

Observations

- From Table 1, we observe that for Small and Medium datasets, BN+Decorr has significant gain over the others. It shows that Decorr is reducing overfitting in the case of small training data.
- We observed that BN+Decorr improved performance in 26/44 datasets and had same performance in 5/44 datasets with BN, out of which 4 were having zero classification error. Dropout is usually known to be effective for dense fully connected layers after the convolutional layers and not effective when applied to filters. We observed the same behavior as BN+DO performs significantly worse than BN alone.
- For datasets where BN+Decorr and BN are similar, we further explored the effect of Decorr regularizer by reducing the training dataset size, and observe that when only a fraction (30%) of the training data is available, BN+Decorr performed significantly better than BN alone implying better regularization with average percentage gain of 8.61%.
- To understand the working of decorrelation loss, we analyzed the layer-wise correlation of filters. As shown in Figure 2, randomly initialized filters are reasonably decorrelated to begin with, but over several training epochs, the correlation between filters tends to increase if $\mathcal{L}_{Decorr}$ is not included, whereas the filters tend to remain decorrelated if $\mathcal{L}_{Decorr}$ is included in the loss function.

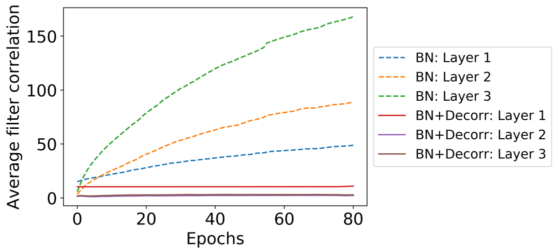

Figure 2: Layer-wise filter correlations averaged over all datasets with and without using decorrelation loss.

Conclusion

In this work, we have explored regularization of FCNs with application to time series classification. The proposed regularizer attempts to learn filters that are decorrelated and is simple to implement, but is effective nevertheless. We observed that the decorrelation loss added to the loss function yielded significant performance gain in terms of classification error rates on diverse TSC benchmark datasets. In the future, we plan to explore the effectiveness of the proposed regularizer for multivariate time series applications, and further explore methods for disentangled representation (Bengio 2013) for time series.

References

Bengio, Y. 2013. Deep learning of representations: Looking forward. In Dediu, A.-H.; Martín-Vide, C.; Mitkov, R.; and Truthe, B., eds., *Statistical Language and Speech Processing*, 1–37. Berlin, Heidelberg: Springer Berlin Heidelberg.

Chen, Y.; Keogh, E.; Hu, B.; Begum, N.; Bagnall, A.; Mueen, A.; and Batista, G. 2015. The ucr time series classification archive. www.cs.ucr.edu/~eamonn/time_series_data/.

Cogswell, M.; Ahmed, F.; Girshick, R. B.; Zitnick, L.; and Batra, D. 2015. Reducing overfitting in deep networks by decorrelating representations. *CoRR* abs/1511.06068.

Lin, M.; Chen, Q.; and Yan, S. 2013. Network in network. *arXiv preprint arXiv:1312.4400*.

Malhotra, P.; TV, V.; Vig, L.; Agarwal, P.; and Shroff, G. 2017. Timenet: Pre-trained deep recurrent neural network for time series classification. *CoRR* abs/1706.08838.

Mishkin, D., and Matas, J. 2015. All you need is a good init. *CoRR* abs/1511.06422.

Rodríguez, P.; Gonzàlez, J.; Cucurull, G.; Gonfaus, J. M.; and Roca, F. X. 2016. Regularizing cnns with locally constrained decorrelations. *CoRR* abs/1611.01967.

Wang, Z.; Yan, W.; and Oates, T. 2017. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International Joint Conference on Neural Networks (IJCNN)*, 1578–1585.