Selective Review on Adaptive Normalization Techniques

Vivek Panday [a], Rajesh Wadhvani [b] and Manasi Gyanchandani [c]

[a] *Maulana Azad National Institute of Technology, Bhopal, India*
[b] *Maulana Azad National Institute of Technology, Bhopal, India*
[c] *Maulana Azad National Institute of Technology, Bhopal, India*

Abstract

Deep Learning provides tools for time series data and are successful on many challenging forecasting tasks. But DL approaches are not always applicable for high volatile and non-stationary financial time series. Data Normalization is used to better learn from time series datasets and pre-process data. Time Series is a sequence of data points collected at regular intervals. Time Series is very changing in nature so finding best method for normalizing time series is a difficult task. Time series data analysis has been the focus of active study from many years and it is believed that it is one of the hardest challenges in data mining owing to its unique qualities. For better analysis of time series we need to preprocess it or normalize it efficiently. Normalization is a good approach which is applied for making data ready by changing the values of in the dataset to bring them to a common scale as the features in the dataset have different ranges. As most of available methods work on some expectations that do not follow for most of time series data. This review discusses all adaptive normalization techniques that have been published in the literature.

Keywords

Time series, data normalization, deep learning, machine learning, adaptive normalization, neural network.

1. Introduction

For Time Series Analysis tasks Deep learning models are used and achieve success, although if time series data is not properly normalized, the performance of models may degrade. Deep Learning have been proved effective in numerous fields where the training and testing data are collected from different domains. But, in real world applications there are more chances of applications of deep learning model to the dataset of new domain that is not available in previous training dataset [4]. Due to activation functions of neurons, in backpropagation networks it is required to preprocess data [9]. A model that can perform good on training data, cannot maintain same performance on new domain, because of cross-domain distributional shift. In DNNs due to change in the distributions of layers' inputs, it creates a problem because the layers are continuously adapting to the new distributions [5]. When the distribution of input changes, it is said to experience internal covariate shift, which we will discuss later on. In the same way, mostly used data normalization methods do not perform well for all types of data distributions. Time series data is having numerous characteristics that make it different from other types of data. Several normalization methods including Z-Score normalization, Decimal Scaling, Min- Max are not always applicable for normalization because of their limitations [1]. In the case of decimal scaling and min-max those methods depend on the minimum and maximum values, whereas Z-Score depends on mean and standard deviation. We have discussed various normalization methods as shown in figure 1.

WINS-2022: Workshop on Intelligent Systems, April 22 – 24, 2022, Chennai, India.
EMAIL: vivekpandey744@gmail.com (Vivek Panday)
ORCID: 0000-0002-4566-6960 (Vivek Panday)

If scales are not similar for different values, then higher values will have a high contribution to output. So, to bring all values to a common scale we need to normalize data. Proper normalization of input data before training has advantages:

- If a feature in dataset is large in scale compared to others then, this large scaled feature becomes dominating and as a result, predictions will not be accurate.
- Normalization tends to make the loss function more symmetrical which are easier to optimize because the gradients tend to point towards the global minimum.

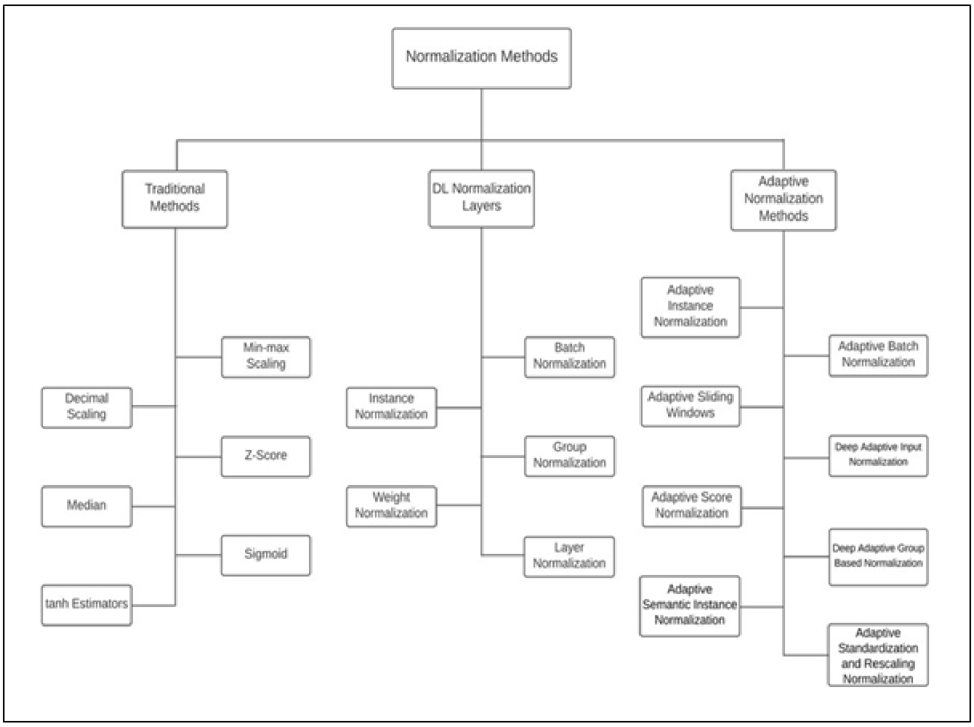
Normalization Methods
Traditional Methods
DL Normalization Layers
Adaptive Normalization Methods
Min-max Scaling
Decimal Scaling
Z-Score
Median
Sigmoid
tanh Estimators
Batch Normalization
Instance Normalization
Group Normalization
Weight Normalization
Layer Normalization
Adaptive Instance Normalization
Adaptive Batch Normalization
Adaptive Sliding Windows
Deep Adaptive Input Normalization
Adaptive Score Normalization
Deep Adaptive Group Based Normalization
Adaptive Semantic Instance Normalization
Adaptive Standardization and Rescaling Normalization

Figure 1: Normalization Methods

2. Adaptive Normalization

Adaptive Normalization method is much different from already available traditional normalization methods because it is based on learning how to execute normalization for specific task rather than depending on previously fixed normalization schemes. Also, it can beeasily applied to any new dataset without need of re-training at the same instance/time. In this paper, we have discussed several techniques of Adaptive Normalization as follows:

2.1. Adaptive Normalization for Non-Stationary Time Series

Time series data showing the non-uniform volatility (tendency to change) is said to have heteroscedasticity, if the dependent variable changes significantly from start to end. The already available technique of normalization of sliding windows does not perform well for heteroscedastic time series. After normalization, all the sliding windows show the same volatility. Fig.2 shows low volatile and high volatile time series and dataset used is U.S.D to B.R.L exchange rate. Proposed method of adaptive normalization is a variation of thistechnique. In this AN, series is first converted into a stationary sequence using concept of moving averages (SMA or EMA decided based on best Adjustment Level from all combinationsof MA and order k, for all DSWs). Then, from this stationary sequence Disjoint Sliding Windows are created and number of inputs in a sliding window is decided based on autocorrelation function. Using this sequence, global statistics can be calculated and are taken into consideration for normalization process. In this way sliding windows in adaptive normalization can be used to present different volatilities. Complete process is divided as follows:

1. By creating a sequence of DSWs (disjoint sliding windows) (non overlapping) from stationary sequence generated.
2. Removal of outliers using interquartile range.
3. Data normalization using min-max.

In this way time series properties are preserved since each sliding window represent different volatilities.

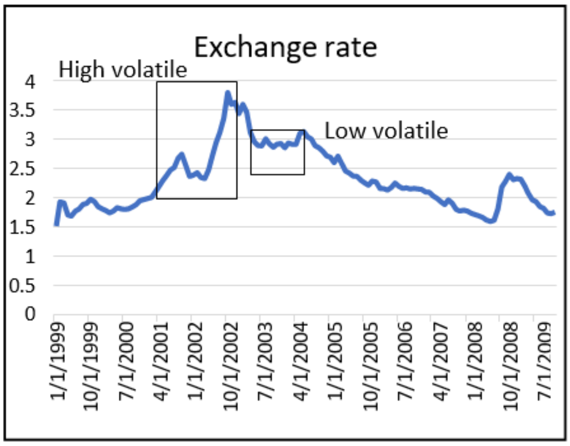

Figure 2: High and Low volatilities

2.2. Adaptive Batch Normalization

Batch normalization layers are basically used to eliminate covariate shift in a deep neural network. In Adaptive batch normalization statistics of layer are converted from source to target domain [6]. This technique of changing statistics in all layers is free from parameters whereas batch normalization requires scaling and shifting parameters. The goal of using Adaptive batch normalization is to perform domain specific normalization.

In batch normalization, mini-batches are normalized. All features in a mini-batch are normalized. Hence, it learns slope and bias for each mini-batch. BN helps in SGD (Stochastic Gradient Descent) optimization, and also helps in better learning rates. BN aligns distribution of data and eliminates the requirement of dropout and acts as a regularizer [5].

In this technique, we train a DNN model having BN and use an algorithm (Algorithm 1) to calculate mean and variance correctly. For k samples' batch for neuron j, mean and variance can be calculated as:

$$d = \mu - \mu_j \qquad (1)$$

$$\mu_j \leftarrow \mu_j + \frac{dk}{n_j},$$

$$\sigma_j^2 \leftarrow \frac{\sigma_j^2\, n_j}{n_j + k} + \frac{\sigma_j^2\, k}{n_j + k} + \frac{d^2 n_j k}{(n_j + k)^2}$$

$$n_j \leftarrow n_j + k$$

Where…

μ is mean and σ^2 is variance of the current input batch for neuron (j) and nj is stats of number of samples for neuron (j) in preceding iterations.

```
Algorithm (1) :  Adaptive Batch Normalization (AdaBN)

for neuron (j) in DNN do
    First collect outputs of neuron {xj(m)} for all images of desireddomain t,
    where xj(m) is the output for image m.
    Then, calculate mean ($\mu^t j$ ) and variance ($\sigma^t j$ ) of the targetdomain
    by Eq. (1).
end for
for neuron (j) in DNN, testing image m in target domain doCompute
BN output:

$y_j(m) = \gamma_j \frac{(x_j(m) - \mu_j^t)}{\sigma_j^t} + \beta_j$

end for
```

2.3. Adaptive Instance Normalization

In Instance normalization particular feature of each channel is normalized instead of normalizing in mini-batches as in batch normalization. Gatys et al. [7] first developed a model for Style transfer by matching statistics of feature for first time. In instance normalization, generally a pair of parameters are trained, so Dumoulin et al. [8] suggested another approach known as Conditional Instance normalization (CIN) layer. This layer gains for each style S, aset of parameters i.e. γ^s and β^s.

$$\textbf{CIN}\ (x, s) = \gamma^s \left(\frac{x - \mu(x)}{\sigma(x)}\right) + \beta_s$$

CIN layers actually have F and S additional parameters, where S is no. of styles and F is the total number of feature maps in the network.

Adaptive Instance Normalization takes two inputs : content input (a) and style input (b).AdaIN performs alignment of mean and variance of content input (a) to match to the style input (b) based on channels. By this type of transfer of statistics, AdaIN combines style and contenteffectively. Like BN, CIN, or IN, Adaptive IN has no affine parameters to be learned.

$$\textbf{AdaIN}(x, y) = \sigma(y) \left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y)$$

Content inputs are scaled and shifted with $\sigma(y)$ and $\mu(y)$ respectively.

2.4. Deep Adaptive Input Normalization

DAIN (Deep Adaptive Input Normalization) is based on a basic and effective neural layer. This neural layer is having capabilities of normalizing input dataset with taking care of data distribution i.e. adaptive normalization. DAIN results in more performance gains as being trained by back propagation. This proposed layer is put together as a sequence of 3 sub-layers[15]. The first layer does the data shifting to the suitable range of the feature space (centering).The linear scaling of data, is done by the second layer for the purpose of expanding or decreasing the variance (standardization) while, the third layer is accountable for non-linearly eliminating irrelevant and ineffective features, also termed as gating. On comparing this proposed method to other normalization methods (Z-score, sample average, instance normalization and batch normalization) after successful training of model, leads to a k-score between 0.280 to 0.345 whereas proposed method increases this k-score to 0.463.

2.5. Adaptive Standardization and Rescaling (ASR) Normalization

In this approach, an adaptive normalization method that utilizes stats assimilate by neural networks in order to increase the generalization capabilities of model, is introduced. All different forms of normalization mostly use the same formula but are actually different in the method of measuring statistics. For example, batch normalization calculate stats for each mini- batch, whereas instance, group, and layer normalizations calculate stats for each sample using various groups of channel. Switchable normalization uses the statistics of all those normalizations along with learnable weights. ASR-Norm may be considered as a general version of conventional normalizing algorithms such as BN, IN, SN and LN [17]. ASR-Norm makes use of Auto-Encoder structured neural networks to gain standardization and rescaling statistics both.

ASR-Norm comes up with a compatible computation graph between training and testing data because it adapts to each input sample [4]. ASR-Norm also incorporates a residual term for standardization statistics in order to support the learning process. When ASR-Norm and ADA (adversarial domain augmentation) are coupled together, the model may acquire robust normalization statistics that increase domain generalization capabilities.

2.6. Deep Adaptive Group-Based Normalization

Deep Reinforcement Learning approaches have provided us with excellent tools for training profitable financial time series.

However, noisy and non-stationary character of time series, requires input normalization algorithms that too precisely planned and optimized, otherwise, agents trading in financial time series will not be able to consistently execute profitable trades. To overcome this limitation, deep adaptive input normalization approach was proposed and especially designed to train DRL agents so that they can financially trade while taking input in the form of raw price directly, without prior need of any extensible pre-processing [2]. This approach makes use of two neural layers that are trainable and are capable of adaptively normalizing the input by:

1. Determining the distribution from where the data is to be sampled.
2. An approach based on grouping is used so that it can record the delicate variationsin better way.

2.7. Adaptive Semantic Instance Normalization (ASIN)

ASIN is an augmentation to the Instance Normalization. It is designed for the transfer of semantic information from specified text to the output images via the instance normalization process. Replacement of natural human language to photo-realistic visuals is always an arduous task. Batch normalization is used for most of the text-to-image type models, for speeding up and stabilizing the training process. But BN has a problem that it does not take care of individual feature differences as it works on batches and also ignores the semantic link between modalities, which is a serious issue for applications based on text-to-image.

ASIN is a method suggested to take care of these concerns. The uniqueness of created pictures is taken into consideration by ASIN method and it also incorporates text semantic details into the image normalization process, leads to a constant and semantically tight association between output images and text provided [16].

2.8. Adaptive Score Normalization

Score normalization techniques are vastly used for the purpose of speaker verification judgments and also to lessen the variation in possible ratio scores. When speaker models are taken into consideration, then there is a drift in likelihood ratio scores and to lessen the drift, the concept of Adaptive Score Normalization is used. Reason for changes in likelihood ratio scores is due to successive speaker model adaptation.

When evaluating LLR (χ_{test}, s) for utterances that depict a diversity of expected sources of variability, mostly it is taken that LLR (χ_{test}, s) have Gaussian distribution. By collecting mean estimation (μ), standard deviation (σ) and two score normalization techniques those are Z-norm and t-norm, make a way for normalized LLR score that is as follows:

$$\mathrm{LLR}(\chi_{\mathrm{test}}, \mathrm{s})_{\mathrm{norm}} = \frac{\mathrm{LLR}(\chi_{\mathrm{test}}, \mathrm{s}) - \mu}{\sigma}$$

After scrutinizing the score deviation, speaker model come up with an adaptive t-norm score normalization method. The score drifting phenomena arises in various detection challenges, involving telephone based text dependent speaker verification functions. In speaker model adaptation situations, scores are prone to drift as the amount of adaption data grows. In adaptive t-norm technique, when a model is targeted and considered then, adaptation of t-norm speaker models is done in the same manner by using utterances from t-norm speakers. We have one adaptation utterance from each of the t-norm speakers for every adaptation iteration for t-norm models [19].

This approach permits acclimation of t-norm model for a specific target speaker to be displayed as output, which results in nominal increment in complexity while validation trials.

3. Literature Review

In this section, we have discussed various methods of adaptive normalization used by different authors for performing specific tasks.

Table 1: Summary of various research papers

Research Paper	Normalization Method	Dataset	Results
Ogasawara, Eduardo S. et al. [1]	DSWs using NN-AN	U.S.D to B.R.L Exchange Rate [32]	NN-AN outperformed other traditional models.
Xun Huang, Serge Belongie [3]	Arbitrary Style Transfer using Adaptive IN	WikiArt dataset [30] and MS-COCO [31]	For unlimited no. of styles AdaIN model outperformed other models.
Fan, Xinjie, Qifei Wang et al. [4]	ASR Norm	CIFAR-10-C dataset	ASR-Norm is most accurate and outperforms BN, SN, and IN.
Li, Yanghao, Naiyan Wang et al. [6]	Adaptive BN	Office-31 dataset	For both single and multiple source adaptation, AdaBN outperforms other models.

4. Comparison of Various Normalization Techniques

In this section, we have compared several normalization techniques including traditional normalization methods, DL Layers and Adaptive Normalization methods.

Dataset used here for comparing different normalization methods is FI-2010 which contains limit order book data [33].

Table 2: Comparison between Normalization Techniques

Method	Model	Macro F1 Score
No Norm.	RNN	31.61±0.40
Z-Score Norm.	RNN	52.29±2.10
Batch Norm.	RNN	51.42±1.05
Instance Norm.	RNN	54.01±3.41
DAIN(1 layer)	RNN	55.34±2.88
DAIN(1+2 layers)	RNN	64.21±1.47
DAIN(1+2+3 layers)	RNN	63.95±1.31

We observed from the comparison above that Deep Adaptive Input Normalization (DAIN) performed best on FI-2010 dataset based on Macro F1 Score. Adaptive Normalization techniques always perform better as compared to other techniques.

5. Conclusion

The techniques for normalization are pivotal to speed up the training of DNN models and for improvement of the DNNs generalization, and are successfully utilized in different applications. Existed normalization techniques for deep neural networks like Instance normalization (IN), Group Normalization (GN), Batch normalization (BN) and Layer normalization (LN) are not truly designed for normalizing input data. These techniques are based on stats that were calculated while training, inspite of dynamically normalizing data. Hence, it proves that dynamic normalization of data is very crucial, because during implementation, we need to implement data from a different domain that is not available in the training set. Adaptive Normalization is used for improving model's Domain generalization capabilities. Adaptive normalization has also proved to be an efficient technique, as it resultedin lesser MSE and MAE when compared with traditional normalization techniques.

6. References

[1] Ogasawara, Eduardo S. et al. “Adaptive Normalization: A novel data normalization approach for non-stationary time series.” The 2010 International Joint Conference on Neural Networks (IJCNN) (2010): 1-8.

[2] A. Nalmpantis, N. Passalis, A. Tsantekidis and A. Tefas, "Improving Deep Reinforcement Learning for Financial Trading Using Deep Adaptive Group-Based Normalization," 2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP), 2021, pp. 1-6.

[3] Huang, X. and Belongie, S., 2017. Arbitrary style transfer in real-time with adaptive instance normalization. In Proceedings of the IEEE International Conference on Computer Vision (pp. 1501-1510).

[4] Fan, Xinjie, Qifei Wang, Junjie Ke, Feng Yang, Boqing Gong, and Mingyuan Zhou. "Adversarially Adaptive Normalization for Single Domain Generalization." In Proceedings of the

IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8208-8217. 2021.

[5] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In International conference on machine learning, pp. 448-

[6] 456. PMLR, 2015.

[7] Li, Yanghao, Naiyan Wang, Jianping Shi, Xiaodi Hou, and Jiaying Liu. "Adaptive batch normalization for practical domain adaptation." Pattern Recognition 80 (2018): 109-117.

[8] Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "Image style transfer using convolutional neural networks." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2414-2423. 2016.

[9] Dumoulin, Vincent, Jonathon Shlens, and Manjunath Kudlur. "A learned representation for artistic style." arXiv preprint arXiv:1610.07629 (2016).

[10] J. Sola and J. Sevilla, "Importance of input data normalization for the application of neural networks to complex industrial problems," in IEEE Transactions on Nuclear Science, vol. 44, no. 3, pp. 1464-1468, June 1997.

[11] Aghabozorgi, S., Shirkhorshidi, A.S. and Wah, T.Y., 2015. Time-series clustering–a decade review. Information Systems, 53, pp.16-38.

[12] Längkvist, Martin, Lars Karlsson, and Amy Loutfi. "A review of unsupervised feature learning and deep learning for time-series modeling." Pattern Recognition Letters 42 (2014): 11-24.

[13] Salles, Rebecca, Kele Belloze, Fabio Porto, Pedro H. Gonzalez, and Eduardo Ogasawara. "Nonstationary time series transformation methods: An experimental review." Knowledge-Based Systems 164 (2019): 274-291.

[14] Tsay, Ruey S. Analysis of financial time series. Vol. 543. John wiley & sons, 2005.

[15] Buza, Krisztian. "Time series classification and its applications." In Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics, pp. 1-4. 2018.

[16] Passalis, Nikolaos, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. "Deep adaptive input normalization for time series forecasting." IEEE transactions on neural networks and learning systems 31, no. 9 (2019): 3760-3765.

[17] Huang, Siyue, and Ying Chen. "Generative Adversarial Networks with Adaptive Semantic Normalization for text-to-image synthesis." Digital Signal Processing 120 (2022): 103267.

[18] Kim, Taesup, Inchul Song, and Yoshua Bengio. "Dynamic layer normalization for adaptive neural acoustic modeling in speech recognition." arXiv preprint arXiv:1707.06065 (2017).

[19] R. Auckenthaler, M. Carey, and H. Lloyd-Thomas, “Score normalization for text-independent speaker verification systems,”Digital Signal Processing, vol. 10, pp. 42–52, 2000.

[20] Yin, Shou-Chun, Richard Rose, and Patrick Kenny. "Adaptive score normalization for progressive model adaptation in text independent speaker verification." In 2008 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 4857-4860. IEEE, 2008.

[21] Huang, Lei, Jie Qin, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. "Normalization techniques in training dnns: Methodology, analysis and application." arXiv preprint arXiv:2009.12836 (2020).

[22] Bhanja, Samit, and Abhishek Das. "Impact of data normalization on deep neural network for time series forecasting." arXiv preprint arXiv:1812.05519 (2018).

[23] Zhang, Heng, Yuanyuan Pu, Rencan Nie, Dan Xu, Zhengpeng Zhao, and Wenhua Qian. "Multi-modal image synthesis combining content-style adaptive normalization and attentive normalization." Computers & Graphics 98 (2021): 48-57.

[24] Yin, Zixin, Jiakai Wang, Yifu Ding, Yisong Xiao, Jun Guo, Renshuai Tao, and Haotong Qin. "Improving Generalization of Deepfake Detection with Domain Adaptive Batch Normalization." In Proceedings of the 1st International Workshop on Adversarial Learning for Multimedia, pp. 21-27. 2021.

[25] Ulyanov, Dmitry, Andrea Vedaldi, and Victor Lempitsky. "Instance normalization: The missing ingredient for fast stylization." arXiv preprint arXiv:1607.08022 (2016).

[26] Tzeng, Eric, Judy Hoffman, Kate Saenko, and Trevor Darrell. "Adversarial discriminative domain adaptation." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 7167-7176. 2017.

[27] Wang, Ximei, Ying Jin, Mingsheng Long, Jianmin Wang, and Michael Jordan. "Transferable normalization: Towards improving transferability of deep neural networks." (2019).

[28] Nam, Hyeonseob, and Hyo-Eun Kim. "Batch-instance normalization for adaptively style-invariant neural networks." arXiv preprint arXiv:1805.07925 (2018).
[29] Wu, Yuxin, and Kaiming He. "Group normalization." In Proceedings of the European conference on computer vision (ECCV), pp. 3-19. 2018.
[30] Zhou, Xiao-Yun, Jiacheng Sun, Nanyang Ye, Xu Lan, Qijun Luo, Bo-Lin Lai, Pedro Esperanca, Guang-Zhong Yang, and Zhenguo Li. "Batch Group Normalization." arXiv preprint arXiv:2012.02782 (2020).
[31] Nichol. Painter by numbers, wikiart, 2016
[32] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Doll ar, and C. L. Zitnick. Microsoft coco:Common objects in context. In ECCV, 2014.
[33] E. Ogasawara, 2010. IJCNN 2010 Datasets.Dispon?vel em:http://www.cos.ufrj.br/~ogasawara/ijcnn2010. Acesso em: 24 Mar 2010.
[34] A. Ntakaris, M. Magris, J. Kanniainen, M.Gabbouj, and A. Iosifidis, “Benchmark dataset for mid-price prediction of limit order book data,” J. Forecasting, vol. 37, no. 8, pp. 852–866, 2018.