Using DenseNet for IoT multivariate time series classification

Joseph Azar*, Abdallah Makhoul† and Raphaël Couturier‡

FEMTO-ST Institute/CNRS
Univ. Bourgogne Franche-Comté
Belfort, France
Email: *joseph.azar@univ-fcomte.fr, †abdallah.makhoul@univ-fcomte.fr, ‡raphael.couturier@univ-fcomte.fr

***Abstract*—Nowadays, most Internet of Things (IoT) devices collect multiple features and produce multivariate time series. In an IoT application, the mining and classification of the collected data have become crucial tasks. Hybrid LSTM-fully convolutional networks (MLSTM-FCN) provide state-of-the-art classification results on multivariate time series benchmarks. This paper examines the use of the DenseNet architecture, originally proposed for computer vision applications, for the classification of multivariate time series. More precisely, this paper proposes a hybrid LSTM-DenseNet model that is able to achieve the performance of the state-of-the-art models and surpass them in many situations, based on the results obtained from various experiments on 15 benchmark datasets. Thus, this paper suggests the 1D DenseNet as a potential tool to be considered by machine learning engineers and data scientists for IoT time series classification task.**

***Index Terms*—IoT, time series classification, multivariate time series, deep learning, DenseNet, long short-term memory**

I. Introduction

An Internet of Things (IoT) is a self-configuring, complex, and adaptive network connecting recognizable "things" to the internet. The "things" have capabilities for sensing and potential programmability. They are defined by information such as the identity, status, or location and offer services, with or without human intervention, through data collection, communication, and ability to take actions [1]. IoT applications are boundless, and the emergence of edge computing has enabled more optimized data processing, real-time analytics, and the use of artificial intelligence in many applications such as the energy and smart grid industries, connected vehicles and transportation, manufacturing, wearables, implantations, and medical devices [2].

Efficient real-time edge analytics covers numerous functionalities in data mining, such as classification, clustering, outlier detection, time series analysis and association analysis, and has recently been attracting many researchers [3] [4]. Many of the data collected from IoT and Industrial IoT (IIOT) applications have been gathered over the course of time, constituting a time series. Time Series Classification (TSC) has been used to solve recognition tasks in different areas, from healthcare to science and industry. These tasks include signature verification, driver guidance, activity recognition, cardiovascular disease detection, and neurological disorders [5]–[7]. With the development of the IoT and 5G, the learning representations and the TSC become more and more relevant.

A time series is a set of ordered real values. Nowadays, many of the IoT devices can collect multiple features, resulting in Multivariate Time Series (MTS). A MTS $T = [t^1, t^2, ..., t^n]$ consists of n univariate time series, where $t^i \in IR^L$ and L is the length of each time series t^i. Various time series classification methods were proposed in the literature [8]. The most famous classifiers are the distance-based classifiers such as k-nearest neighbor with Dynamic Time Warping (DTW) distance and dictionary-based classifiers in which a time series is converted into representative words. An example of dictionary-based techniques is the bag-of-SFA-symbols (BOSS). Certain classifiers are shapelet-based, such as the Fast Shapelets (FS) and the Shapelet Transform (ST), and ensemble-based, such as the Collective of Transformation Ensembles (COTE), which is a combination of classifiers in different domains. The aforementioned categories and methods of classification are described in detail in [8] [9].

Another approach to time series classification which has proved to be effective in recent years is end-to-end deep learning [10]. This approach consists in learning the hidden features from raw time series using non-linear transformations without the need for extensive feature engineering. Different architectures have been proposed in the literature that are based on Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Multi Layer Perceptron (MLP), and Echo State Network [10] [11].

This paper proposes a Deep Neural Network (DNN) model that is based on the densely connected convolutional network, namely DenseNet, and the Long Short-Term Memory (LSTM) RNN. Note that the objective of this paper is not to compare the proposed deep learning model with classifiers in different categories (distance-based, feature-based, ensemble, etc.), but rather to show that DenseNets are candidates which should be taken into account when choosing a deep learning architecture for TSC. Note that when processing time series, DenseNets can be used in a stand-alone way, or combined with LSTMs and this paper will consider both strategies.

The paper's structure is as follows. Section II offers an insight into convolutional and recurrent neural networks. Section III presents the most effective deep learning-based time

series classifiers proposed in the literature. Section IV explains the proposed DenseNet-based deep learning model. Section V outlines the results obtained and compares them with those obtained from the models discussed in section III. Section VI concludes this paper.

II. Background

This section provides a background on CNNs and RNNs, as most of the proposed TSC models are based on convolution blocks and LSTM units.

A. Convolutional neural networks

Normally, CNNs are used to analyze image data. With multidimensional data, such as images and MTS in our case, this kind of network scales well. A CNN is a feed forward network consisting of many layers including convolutional filters with batch normalization [12] and layers for rectification and pooling. A CNN typically adopts one or more fully-connected layers after the last pooling layer in order to convert 2-D feature maps into a 1-D vector to allow for final classification.

CNNs have proven to be effective in solving computer vision issues, including achieving state-of-the-art results on tasks such as image classification and captioning, object localization and more. This is done by working directly on raw data such as raw pixel values instead of on manually derived features extracted from the raw data. In order to solve the addressed problem, the model learns how to automatically extract the useful features from the raw data. This type of learning is referred to as representation learning, and the CNN does this in a way that extracts the features regardless of how they appear in the data. CNNs' ability to learn and derive features automatically from raw input data can be extended to TSC problems. A series of observations can be viewed as an image that can be read and transformed into the most salient elements by a CNN model. In this case, a convolution can be seen as applying and sliding a one-dimensional filter over the time series to perform a non-linear transformation.

B. Recurrent neural networks

Recurrent neural networks such as the Long Short-Term Memory Network (LSTM) provide the explicit handling of order between the input samples while learning a mapping function from inputs to outputs that CNNs do not provide. Typically, RNNs are used with training samples that have clear interdependencies and relevant representation to hold information about what happened in previous time steps. In other words, these models are providing output by leveraging two input sources, the current and the recent past.

RNNs suffer from the so-called long-term dependencies problem. This means that when attempting to link previous information to a present task they can not reach far previous memory. LSTM networks can be effective for TSC problems because they can overcome the long-term dependencies issue by incorporating weights and gates [13], and they have the ability to learn long-term associations in a sequence. LSTM networks do not require a pre-specified time window and are able to model complex multivariate sequences accurately. However, even if an LSTM is expected to capture the long-term dependencies better than the RNN, it appears to become forgetful. This issue is tackled by *Bahdanau et al* in [14]. *Bahdanau et al* proposed the attention mechanism to address the long-term dependencies problem in neural-based machine translation of texts. The goal was to assign some of the input words more significance compared to others while the sentence was being translated. The authors have done this simply by building a context vector and taking a weighted sum of the hidden states. The core idea is an attempt to mimic the humain brain by selectively concentrating on a few relevant things, while ignoring others. The authors in [15] detail the use of the attention mechanism for TSC.

III. Related work

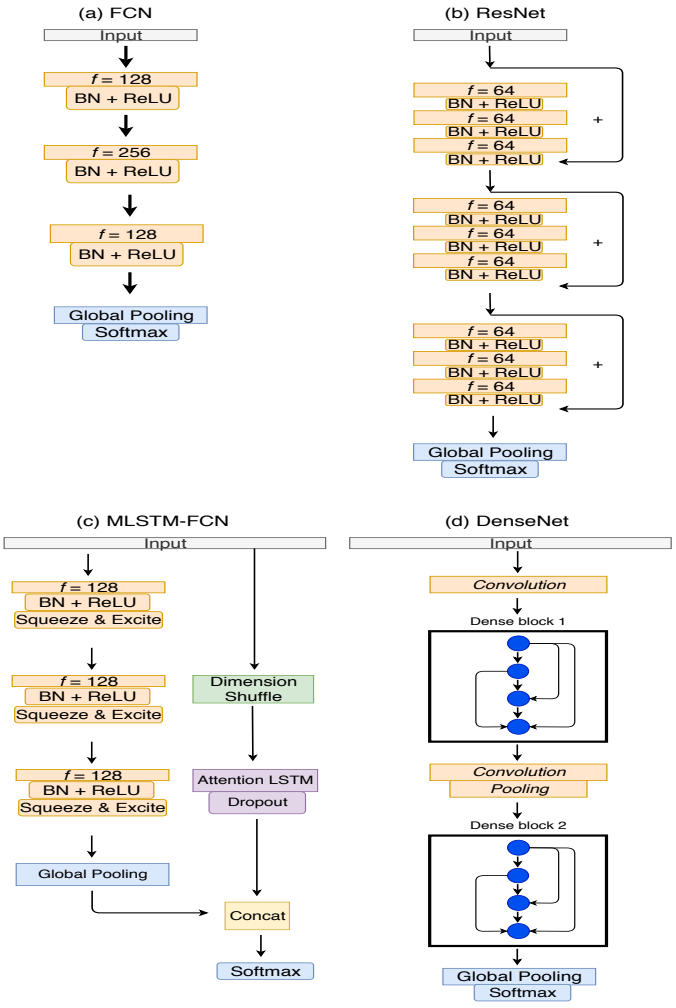

Fig. 1: The network structure of FCN, ResNet, MLSTM-FCN, and DenseNet

Different neural network architectures for multivariate TSC were proposed in the literature. In the following we will present the most effective architectures based on the results obtained on the TSC benchmarks, namely the UCR/UEA

archive [8] [16] and the MTS archive [17], in addition to the conclusions derived from detailed surveys [10] [11].

The Fully Convolutional Networks (FCNs) were originally proposed for segmentation tasks [18] [19] and are very effective in extracting features from input data. The FCN used for TSC is formed by stacking three blocks, each consisting of a convolutional layer with filters, preceded by a batch normalization layer and a ReLU activation layer as shown in Figure 1-a. Then a global average pooling layer is applied to the features after the first three convolutional blocks, effectively reducing the number of weights. Finally, the softmax layer produces the final output.

The Residual Network (ResNet) is a deep architecture used for TSC. There are many variants. The version presented in Figure 1-b consists of 9 convolutional layers followed by global average pooling and softmax layers. This architecture uses shortcut connections between successive convolutional layers for training and adds linear shortcuts to connect a residual block's output to its input thus resulting in an easier training.

The authors in [15] proposed the Multivariate LSTM Fully Convolutional Network (MLSTM-FCN). This proposed architecture has yielded so far the best results on the multivariate UEA archive. The MLSTM-FCN model, as shown in Figure 1-c, consists of a fully convolutional block and a LSTM block. The input multivariate time series is transferred into a shuffle layer and subsequently passing it into a LSTM block with attention mechanism followed by dropout. The attention LSTM layer output is concatenated with the global pooling layer output, and the final results are obtained from the softmax layer. An interesting approach used by the authors is the addition of the squeeze and excite block proposed and detailed in [20] after each convolutional block, thus enhancing the performance of the model.

The Densely Connected Convolutional Networks (DenseNet) was proposed in [21]. The proposed architecture attempts to solve the problem of vanishing gradient by introducing connection from one layer to all its consequent layers in a feed forward manner. The authors in [22] adapted the DenseNet architecture for univariate TSC. The motivation behind using DenseNet for time series is that this architecture facilitates the propagation of features and enables feature reuse, which greatly reduces the number of parameters. However, the main objective of the work in [22] is to study the normalization techniques used with DenseNet and not to compare its performance with the state of the art DNN techniques used for TSC.

The rest of this paper introduces an adapted version of DenseNet for the processing of multivariate time series (Figure 1-d), and a new architecture inspired by the work described in [15] in which the DenseNet is used with LSTM. Notice that the authors in [15] referred to their architecture as MALSTM-FCN, where they use the attention mechanism with LSTM. We will refer to this architecture in this work as MLSTM-FCN.

IV. Proposed models

This section presents the two DNN models considered in this paper. Similarly to [22], the first model is an adapted version of the DenseNet for multivariate time series, and is illustrated in Figure 2. It takes a signal input of size (m, n) where m is the number of timesteps and n the number of features. The input goes first through a 1D convolution layer with a kernel size of 3 then through two dense blocks and a global pooling step. Each dense block consists of four convolution steps. Each step applies four operations, namely batch normalization, ReLU activation, squeeze and excite operation inspired from [15] as well as a 1D convolution with kernel size of 3. The number of filters is initialized to 32 and incremented by 16 after each step. It may be noticed that, after each step, the number of parameters in each dense block increases by more than 16. This is due to the concatenation operation, where features of the k^{th} layer are concatenated with features of previous layers and are provided as input to the next layer. Between both dense blocks, a down-sampling layer that reduces the size of feature maps is used. This layer is referred to as the transition layer and it applies batch normalization, ReLU activation, 1D convolution with kernel size 1 and 1D average pooling with pooling size 2 to the input. The last dense block's output is passed through batch normalization, ReLU activation, 1D global average pooling, and softmax.

The second model uses the same DenseNet architecture of the first, and adds more complexity to it. It is inspired from the work proposed in [15] and illustrated in Figure 1-c. As shown in Figure 3, a signal input of size (m, n) is fed to the previously described dense architecture, in addition to a LSTM layer with 8 units and an attention mechanism followed by a dropout operation. The number of units has been chosen based on the results achieved in [15]. Note that the temporal dimension of the input is shuffled before being fed to the LSTM layer when the number of timesteps is greater than the number of features. Given that the number of timesteps m is usually greater than the number of features n in a time series, the LSTM will involve n timesteps to handle m features, which is more efficient, when shuffling the dimension and providing the LSTM an input of dimension (n, m). The output of the attention LSTM layer is then concatenated with the output of the global pooling layer of the DenseNet block and passed through a softmax layer. Both models use the variant of stochastic gradient descent 'Adam' [23] with a learning rate of 0.001 as optimizer[1].

V. Experiments

This paper uses the Keras library [24] with the Tensorflow backend [25] to train the aforementioned models. In the experiments of this paper, 15 datasets coming from [17], [26], [27] are used. Note that the authors in previous works have made their experiments on more datasets, such as in [10],

[1]The codes and weights of some models are available at https://github.com/josephazar/MLSTM-DenseNet

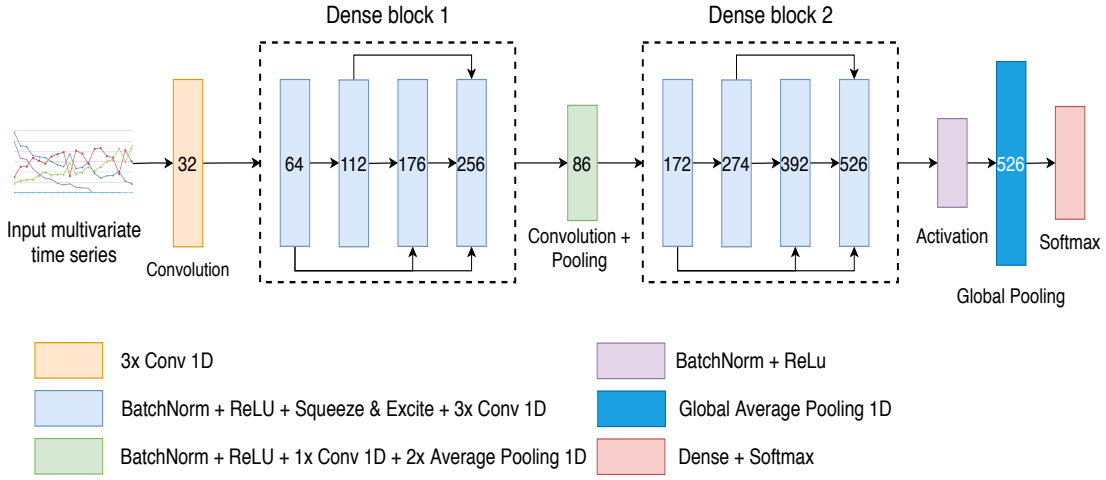

Fig. 2: The adapted 1D-DenseNet architecture for multivariate time series classification. The numbers in each box denote the number of parameters at each step.

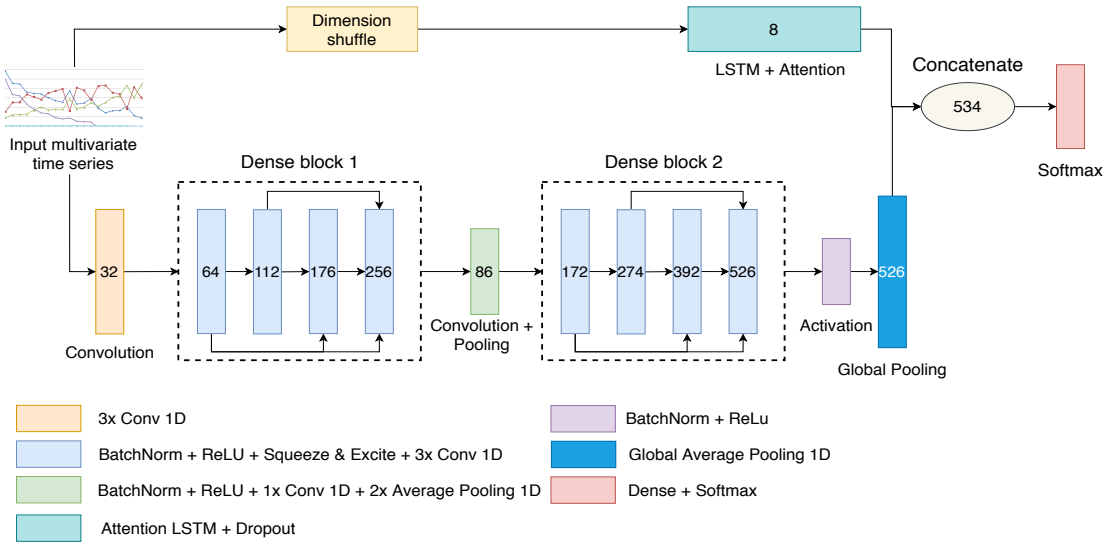

Fig. 3: The adapted MLSTM-FCN architecture proposed in [15] where the FCN is replaced by the 1D-DenseNet. The numbers in each box denote the number of parameters at each step.

[15]. The previously proposed models were able to achieve very high accuracy on multiple datasets, ranging from 95% to 100%. Such datasets are omitted in our experiments where very high accuracy is reached by most of the DNN models, since the features in these datasets can be easily learned by a simple classifier and there is nothing to improve. The computations were performed on an NVIDIA Tesla Titan X GPU to train the models on these datasets.

A DNN model's input is a tensor with shape (s, m, n), where s is the number of samples, m is the length of each sample's time series (or number of timesteps), and n is the number of features. The samples of each dataset are already divided into train and test sets. Hyperparameters such as the number of epochs and batch size are specified for each dataset, and the option of standardizing the numerical features to have a mean 0 and unit variance using standard scalar is based on the numerical ranges of the features and the various empirical tests performed. Table I shows the properties and parameters used for each dataset.

The accuracy metric was used to evaluate the performance of the five DNN models discussed in this paper, namely FCN, ResNet, MLSTM-FCN and the two models described in section IV: DenseNet and MLSTM-DenseNet. Important to note that the accuracy value achieved in the experiments by the state-of-the-art models may vary from the one achieved in the original papers, and this is due to the change in the hyperparameters, and the standardization of some datasets in addition to the stochastic nature of deep learning algorithms. Though, we assured that all models are trained and evaluated on each dataset with the same settings and conditions.

Table II shows the classification accuracy value obtained by implementing the 1D-DenseNet and MLSTM-DenseNet,

Dataset	# classes	# features	Timesteps	# samples	Task	Train-test	Batch size	Epochs	Standardize	Source
Uwave	8	3	315	4478	Gesture recognition	20-80	16	2000	yes	[17]
Gesture Phase	5	18	214	396	Gesture recognition	50-50	16	2000	yes	[26]
LP3	4	6	15	47	Robot failure recognition	36-64	8	2000	no	[26]
LP5	5	6	15	164	Robot failure recognition	39-61	16	2000	no	[26]
Ozone	2	72	291	344	Weather classification	50-50	16	2000	no	[26]
ECG	2	2	152	200	ECG classification	50-50	16	2000	yes	[17]
EEG	2	13	117	128	EEG classification	50-50	16	2000	no	[26]
KickvsPunch	2	62	841	26	Action recognition	62-38	4	2000	yes	[17]
Netflow	2	4	997	1337	Action recognition	60-40	128	2000	yes	[17]
Action 3D	20	570	100	567	Action recognition	47-53	16	2000	yes	[27]
LP2	5	6	15	47	Robot failure recognition	36-64	8	2000	no	[26]
HT Sensor	3	11	5396	100	Food classification	50-50	8	2000	no	[26]
Movement AAL	2	4	119	314	Movement classification	50-50	64	2000	yes	[26]
Occupancy	2	5	3758	117	Occupancy classification	35-65	16	2000	yes	[26]
AREM	7	7	480	82	Activity recognition	51-49	16	2000	no	[26]

TABLE I: Properties of all the datasets used in this paper and originated from [17], [26], [27]. The batch size and number of epochs used to train the DNN models are shown for each dataset

Dataset	FCN	ResNet	DenseNet	MLSTM-DenseNet	MLSTM-FCN
Uwave	93.4	92.6	**93.6**	89.7	89.3
Gesture Phase	45.9	47.9	48.9	**57.0**	55.5
LP3	46.6	43.3	70.0	**83.3**	80.0
LP5	68.0	65.0	68.0	**69.0**	67.0
Ozone	81.5	83.2	81.9	**86.0**	82.0
ECG	90.0	91.0	92.0	**93.0**	**93.0**
EEG	64.0	62.5	62.5	**67.0**	65.6
KickvsPunch	60.0	80.0	**100**	**100**	**100**
Netflow	95.5	95.8	**96.6**	95.1	87.2
Action 3D	55.5	57.5	52.1	**77.1**	75.4
LP2	46.3	30.0	76.6	**83.3**	**83.3**
HT Sensor	84.0	**88.0**	82.0	86.0	82.0
Movement AAL	61.1	59.2	58.6	77.0	**79.0**
Occupancy	71.0	72.3	72.3	**79.0**	77.0
AREM	84.6	89.7	87.18	**94.8**	92.3

TABLE II: Achieved accuracy by the five DNN models on the 15 multivariate time series datasets. The best performance is shown in bold

and regenerating the FCN, ResNet, and MLSTM-FCN models based on their actual implementation on github. The proposed MLSTM-DenseNet model was able to outperform the other models on 8 datasets and provide the highest accuracy on 3 datasets, namely ECG, LP2, and KickvsPunch, alongside the MLSTM-FCN. The 1D-DenseNet model yielded the best accuracy on the Uwave and Netflow datasets, the ResNet model on the HT Sensor dataset, and the MLSTM-FCN model on the Movement AAL dataset. This means that the performance of the proposed MLSTM-DenseNet model over various datasets is generally higher than that of other state-of-the-art models.

VI. Conclusion

The proposed MLSTM-DenseNet classification model shows that replacing the Fully Convolutional Network (FCN) with a DenseNet is a potential solution that needs to be considered for the classification of multivariate time series. The benefit of DenseNet is that every layer has direct access to the gradients from the loss function and the original input signal, resulting in an enhanced flow of information and gradients across the network, as well as a regularizing effect that decreases overfitting on tasks with limited training set sizes. Experiments have been conducted on 15 multivariate time series datasets, and the results show that the proposed approach is capable of achieving the performance of the state-of-the-art models and bypassing them in several cases. For some datasets, such as the ones in which the number of timesteps is high, the standalone 1D DenseNet showed outperformed the hybrid models. Considering the 1D DenseNet as a standalone model or as a replacement for FCN in the MLSTM-FCN for the classification of multivariate time series can thus help to improve classification performance.

For future work, we intend to perform further experiments and compare the models with regard to other metrics than the accuracy such as using the mean per-class error, training and prediction time and the complexity of the models. Moreover, we intend to better understand why the proposed model does not match the FCN and ResNet performance on the Uwave dataset.

Acknowledgment

This work has been supported by the EIPHI Graduate School (contract "ANR-17-EURE-0002"). Computations have been performed on the supercomputer facilities of the "Mésocentre de calcul de Franche-Comté"

References

[1] "Towards a definition of the internet of things (iot)," https://iot.ieee.org/definition.html, accessed: 2020-01-13.

[2] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, Jan 2017.

[3] J. Azar, A. Makhoul, M. Barhamgi, and R. Couturier, "An energy efficient iot data compression approach for edge machine learning," *Future Generation Computer Systems*, vol. 96, pp. 168 – 175, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X18331716

[4] N. Harth, C. Anagnostopoulos, and D. Pezaros, "Predictive intelligence to the edge: impact on edge analytics," *Evolving Systems*, vol. 9, no. 2, pp. 95–118, Jun 2018. [Online]. Available: https://doi.org/10.1007/s12530-017-9190-z

[5] J. Tervonen, V. Isoherranen, and M. Heikkil, "A review of the cognitive capabilities and data analysis issues of the future industrial internet-of-things," in *2015 6th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, Oct 2015, pp. 127–132.

[6] V. Kurbalija, M. Radovanovi, M. Ivanovi, D. Schmidt, G. L. von Trzebiatowski, H.-D. Burkhard, and C. Hinrichs, "Time-series analysis in the medical domain: A study of tacrolimus administration and influence on kidney graft function," *Computers in Biology and Medicine*, vol. 50, pp. 19 – 31, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010482514000936

[7] E. Florido, F. Martnez-lvarez, A. Morales-Esteban, J. Reyes, and J. Aznarte-Mellado, "Detecting precursory patterns to enhance earthquake prediction in chile," *Computers & Geosciences*, vol. 76, pp. 112 – 120, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0098300414002805

[8] A. Bagnall, A. Bostrom, J. Large, and J. Lines, "The great time series classification bake off: An experimental evaluation of recently proposed algorithms. extended version," 2016.

[9] A. Bagnall, J. Lines, J. Hills, and A. Bostrom, "Time-series classification with cote: The collective of transformation-based ensembles," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 9, pp. 2522–2535, Sep. 2015.

[10] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, p. 917963, Mar 2019. [Online]. Available: http://dx.doi.org/10.1007/s10618-019-00619-1

[11] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," *CoRR*, vol. abs/1611.06455, 2016. [Online]. Available: http://arxiv.org/abs/1611.06455

[12] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014.

[15] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate lstm-fcns for time series classification," *CoRR*, vol. abs/1801.04503, 2018. [Online]. Available: http://arxiv.org/abs/1801.04503

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

[17] "Mustafa baydogan - multivariate time series classification data sets," http://www.mustafabaydogan.com/files/viewcategory/20-data-sets.html, accessed: 2020-02-04.

[18] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CoRR*, vol. abs/1411.4038, 2014. [Online]. Available: http://arxiv.org/abs/1411.4038

[19] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[20] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks," 2017.

[21] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," 2016.

[22] G. Richard, G. Hébrail, M. Mougeot, and N. Vayatis, "Densenets for time series classification: towards automation of time series pre-processing with cnns."

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[24] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[25] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[26] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[27] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3d points," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, June 2010, pp. 9–14.