Towards a Universal Neural Network Encoder for Time Series

Joan Serrà [a,1], Santiago Pascual [b] and Alexandros Karatzoglou [a]
[a] *Telefónica Research, Barcelona*
[b] *Universitat Politècnica de Catalunya, Barcelona*

Abstract. We study the use of a time series encoder to learn representations that are useful on data set types with which it has not been trained on. The encoder is formed of a convolutional neural network whose temporal output is summarized by a convolutional attention mechanism. This way, we obtain a compact, fixed-length representation from longer, variable-length time series. We evaluate the performance of the proposed approach on a well-known time series classification benchmark, considering full adaptation, partial adaptation, and no adaptation of the encoder to the new data type. Results show that such strategies are competitive with the state-of-the-art, often outperforming conceptually-matching approaches. Besides accuracy scores, the facility of adaptation and the efficiency of pre-trained encoders make them an appealing option for the processing of scarcely- or non-labeled time series.

Keywords. Neural networks, time series, classification, representation learning, multi-task learning, transfer learning, generalization.

1. Introduction

Time series data present a number of characteristics that motivate specific processing strategies. Besides the importance of attribute ordering, temporal correlations, periodicities, and drifts [13], time series algorithms typically deal with variable lengths, high-dimensional inputs, and scarcely labeled data. For instance, the UEA/UCR time series classification repository [3] contains data sets of sizes ranging between 40 and 16,637 instances, and lengths/dimensionalities between 24 and 2,709 samples. Apart from classification [1], other important tasks in time series are clustering [20], segmentation [18], motif discovery [24], anomaly detection [7], and forecasting [16].

In this paper, we study the use of an encoder to tackle the aforementioned challenges of variable length, high-dimension, and few labeled data. To overcome the first two challenges, we couple a time-wise attention mechanism with convolutional neural networks. The attention mechanism summarizes variable-length representations into fixed-length vectors, while convolutions deal with local/temporal correlations. To overcome the latter challenge, we propose to learn a universal network, trained with a variety of data sets, that can deal with new data types without further intervention or training. Overall, our objective is to develop and train an encoder network that converts variable-length time

[1] Corresponding author: Joan Serrà, Telefónica Research, Pl. Ernest Lluch i Martín 5, 08019 Barcelona; E-mail: `joan.serra@telefonica.com`.

series to a fixed-length, low-dimensional representation which, when interchanged with the raw time series or other features extracted from it, improve a reference task. Importantly, we want the learned representations to generalize to unseen data types, with minimal or even no adaptation of the encoder network to the novel data. This last point, the generalization of learned representations to unseen data types, is an active area of research within machine learning which, to the best of our knowledge, has not received much attention in the time series domain.

Although the usage of the proposed encoder and its representations aim at general time series problems, in this paper, we restrict ourselves to the problem of time series classification [1], as it allows for a clear and objective evaluation, and also well represents the aforementioned challenges in time series processing [13]. Moreover, there exist a reasonable amount and variety of time series classification data sets [3], organized by data type, with which we can conveniently train an encoder and then test it with an unseen type. Under this setting, a pre-trained universal encoder should produce representations that are useful to automatically label, for instance, an electrocardiogram (ECG) data set, without having seen any ECG instance in the training phase. Such labeling should be performed with minimal adaptation to the target data or, in the extreme case, without any learning over such data.

2. Related Work

Multi-task learning [5], in which commonalities across multiple related tasks are exploited to better solve some target task(s), has a long tradition. In the main setting, multi-task learning uses a shared representation that is learnt in parallel across several tasks, including the target one(s). This can be an unrealistic scenario, as target data sets may not be available beforehand, they may not have labels, or it simply may become unfeasible to re-train in parallel with all data sets every time we find a new target task [31]. Transfer learning [27] is an interesting alternative, in which a pre-trained model is adapted to a new target task, with less effort and better results than training from scratch on the new task. Transfer learning typically does not reuse previous data in the adaptation step but, nonetheless, it assumes labeled data for the target task. Notice also that, under a sequential or lifelong learning scenario [31], repeated transfer learning may yield to the phenomenon of catastrophic forgetting [23], in which the knowledge of previous tasks progressively vanishes.

In order to have sufficient knowledge to accomplish any task, and in order to be applicable in the absence of labeled data or even without adaptation/re-training, researchers have been increasingly adopting the generic concept of universal encoders, specially within the text processing domain [6,9,12] (note that related concepts also exist in other domains [8,11,34]). The basic idea is to train a model (the encoder) that learns a common representation which is useful for a variety of tasks and that, at the same time, can be reused for novel tasks with minimal or no adaptation. While it would seem that classical autoencoders and other unsupervised models should perfectly fit this purpose, recent research in sentence encoding shows that, with current means, encoders learnt with a sufficiently large set of supervised tasks [9], or mixing supervised and unsupervised data [6], consistently outperform their purely unsupervised counterparts.

Despite time series classification offers an interesting testbed for universal encoders, to the best of our knowledge, only Malhorta et al. [22] learn time series encoders whose outputs are later exploited to perform new classification tasks. In particular, they consider seq2seq [30] autoencoders, and train them to reconstruct time series, either with single or multiple data sets. Adaptation to the new (supervised) data set is done through support vector machine classifiers with radial basis function kernels. They report accuracies marginally over the typical nearest-neighbor classifier using a dynamic time warping (DTW) distance.

Deep neural networks are progressively being introduced to the problem of time series classification, with promising results [10,14,36]. However, due to the diversity in time series lengths and the low number of instances in the training sets (often under 100), these type of algorithms seem to struggle to catch up with more competitive approaches [21]. In general, ensemble approaches with multiple classifiers, features, and distances are the most competitive ones [1]. Two successful algorithms of this kind are COTE [2] and HIVE-COTE [21]. Canonical baseline approaches using the raw time series are based on nearest-neighbor classifiers with elastic distances [28], such as the aforementioned DTW distance, or feature-based classifiers on top of the raw time series [1]. Their accuracies are always significantly below the ones achieved by competitive ensemble methods.

3. Towards a Universal Encoder for Time Series

3.1. Architecture

In the design of the encoder network we strive for simplicity and efficiency. That is, we strive for a model that is both conceptually straightforward and computationally lightweight. The former is interesting for implicit regularization and ease of explanation, understanding, and deployment. The latter is important for pre-trained model transfer between users and speed of operation.

The model we consider as encoder is a standard convolutional network, with a convolutional attention mechanism to summarize the time axis, and a final fully-connected layer to set the desired representation dimensionality (Fig. 1). The convolutional network is formed by three convolutional blocks with two 2-factor max-pooling layers between them. A convolutional block is formed by a 1-dimensional convolution, followed by an instance normalization layer [32], a parametric rectified linear unit (PReLU) [15] activation, and a dropout layer (Fig. 1, bottom left). After the first part of the network, half of the filters are input to a time-wise softmax activation, which acts as an attention mechanism for the other half of the filters. That is, for a single filter,

$$h = \mathbf{h} \cdot \mathbf{a}, \tag{1}$$

where $\cdot$ denotes a dot product, $\mathbf{h}$ is the result of a single 1-dimensional convolutional filter over a time-wise signal, and $\mathbf{a}$ is the time-wise attention vector (independent for each filter). The result of the attention mechanism for all filters is finally passed through a fully-connected and an instance normalization layers (Fig. 1, top right). We found instance normalization to facilitate training and to provide more consistent value ranges in the encoder's output. The dimensionality of the output is denoted by k, which is a parameter whose impact we study below (Sec. 5.2).

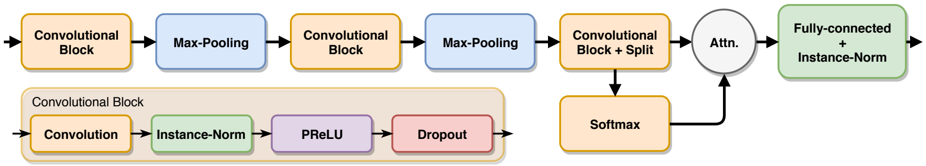

Figure 1. Architecture diagram of the proposed encoder (top) and the convolutional block (bottom left).

3.2. Implementation Details

For the three convolutional layers we use[2], respectively, 128, 256, and 512 filters, with kernels of 5, 11, and 21 (stride of 1), and same-length padding of 3, 5, and 10. Instance normalization includes the affine transformation [17], and PReLUs are multi-parametric, that is, they have one slope parameter per filter. We use a dropout of 0.2 in all layers. Half of the 512 filters of the last convolutional layer are input to the softmax layer and later used to compute the filter-wise dot product with the remaining half (Eq. 1).

3.3. Training

For learning the weights of the encoder network, we take all training data sets one by one and backpropagate the error on each data set batch-wise. That is, for every data set, we take a single batch of examples and do a forward pass with an extra classification, fully-connected layer that learns to map the encoder representation to the number of classes of the given data set. We then take the softmax of that output and measure the error with categorical cross-entropy. This strategy corresponds to a so-called multi-head output [4], typically used in multitask or sequential learning. After the forward pass, we backpropagate the error to both the extra layer and the encoder network. We repeat this process using single random batches for every single data set 20 times. That is, if there are n data sets in the training set, we do $20n$ single-batch forward-backward passes. This defines a training epoch.

To update the weights of the networks we use plain stochastic gradient descent with a learning rate of 0.005. We reduce the learning rate by a factor of 3 if we do not observe an improvement in the validation loss for more than 10 epochs, and stop training when we hit a learning rate below 10^{-4}. As validation loss we use an average of the per-data set losses, using all validation data. We use a batch size of 12.

4. Evaluation Methodology

4.1. Data and Splitting

To assess the quality of the representations, we consider the task of time series classification [1]. In particular, we consider the 85 data sets of the UEA/UCR time series classification repository [3]. To assess the generalization capabilities of the learned representations, we form encoder train/test splits according to the data type. This way, at test time, we evaluate the encoder with a data type that has not been used for training. The repos-

[2] Unless stated otherwise, we use PyTorch [25] version 0.3.1 with default parameters.

itory contains 7 data types: electric devices (6 data sets), ECGs (7), image outlines (29), motion capture (14), sensor readings (16), spectrographs (7), and simulated/artificial data (6). Therefore, we follow a 7-fold training procedure. When learning the parameters of the encoder, we leave out all the data sets corresponding to one data type for testing, and split the rest of the data sets into train/validation following a per-data set, non-stratified 80/20% rule. At test time, we take the left out data sets (corresponding to one data type) and use the original single train/test split provided by the repository. We use the train split to fine-tune the parameters of the encoder (if needed), and to learn the mapping from the representations to the specific class labels. The test split is solely used to compute the reported accuracy scores. Following common practice, all time series are pre-normalized to have zero mean and unit variance.

4.2. Measures

In addition to the raw accuracy score (in %), we consider the normalized accuracy ratio

$$R_i = \frac{A_i - A_i^{\mathrm{M}}}{100 - A_i^{\mathrm{M}}},$$

where i denotes the i-th data set, A_i is the accuracy obtained with the current classifier, and A_i^{M} is the accuracy of a majority-based classifier. This way, R is a quantity that is normalized by both the number of classes and the relative difficulty of the prediction task with respect to the class distribution. Apart from A and R, we also report the average rank of the considered approaches, including the baselines evaluated in the repository, and the number of times an approach is the best across all approaches and baselines. We find a total of 36 baselines in the repository, including some of the most competitive existing approaches [1].

4.3. Encoder Adaptation

To assess the goodness of the learned representations in the case of no adaptation, we consider the performance of a one nearest-neighbor (1NN) classifier. The 1NN classifier is the main choice to evaluate time series similarity measures [28], and almost always outperforms other classifiers when considering the raw time series [1]. In our case, the 1NN classifier performs no further adaptation or learning (it only retrieves closest points), and exploits the Euclidean distance between representations, which we believe is an interesting proxy for other unsupervised tasks like clustering or motif discovery.

To assess the goodness of the learned representations in the case of performing some adaptation, we consider the performance of two classifiers[3]: a logistic regression classifier (LR; with regularization or complexity parameter $C = 0.1$) and a support vector machine with a radial basis function kernel (SVM; $C = 100$). In all previous cases, the parameters of the encoder remain frozen while the classifiers learn to map representations to class labels. We only normalize the representation components to have zero mean and unit variance.

A further case we consider is the adaptation of both encoder and mapping to the new task. For that we take the pre-trained encoder and fine-tune it, together with a fully-

[3] Unless stated otherwise, we use scikit-learn [26] version 0.19.1 with default parameters.

Approach	$\bar{A}$	$\bar{R}$	Rank	Wins
Euclidean-1NN	70.9	0.504	29.7	1
DTW-Rn-1NN	75.9	0.580	23.4	2
TWE-1NN	76.4	0.580	22.4	3
Encoder-1NN	**76.5**	**0.599**	**22.7**	**2**
MSM-1NN	77.3	0.593	20.1	2
RotF	77.6	0.608	17.8	6
Encoder-LR	**79.8**	**0.650**	**17.3**	**5**
Encoder-SVM	**80.3**	**0.667**	**15.6**	**5**
BOSS	81.0	0.676	14.3	15
Encoder-NEW	**81.3**	**0.682**	**11.9**	**16**
ST	82.2	0.694	11.9	17
Encoder-ADAPT	**82.9**	**0.708**	**8.7**	**26**
COTE	83.8	0.715	7.7	18

Table 1. Average performance of selected approaches. Values are computed by considering the original single splits of all the 85 data sets and 36 baselines of the UCR/UEA repository, together with the encoder-based approaches. However, due to space constrains, we do not show all baselines and individual data set values. The encoder-based classifiers use $k = 256$.

connected layer with softmax activation (ADAPT). Finally, to assess the benefit of encoder pre-training, we also consider an additional encoder network trained from scratch solely on the new target task (NEW). Training for ADAPT and NEW is done with Adam [19] for 100 epochs with empirically-chosen learning rates of $5 \cdot 10^{-5}$ and 10^{-4}, respectively. In pre-analysis, we made sure that both ADAPT and NEW were able to converge to a stable solution with this amount of training.

5. Results

5.1. *Accuracy and Ranking*

As mentioned, we compute the evaluation measures for every encoder-based approach on all the 85 data sets of the UCR/UEA repository, and then compare against all 36 baselines available in the same repository (Sec. 4). However, due to space constrains, and for ease of summarization, we only report average measures and focus on selected baseline approaches (Table 1). First of all, we observe that using the raw learned representation without adaptation (Encoder-1NN) is already a very competitive strategy. It clearly outperforms the Euclidean distance baseline, and has a better accuracy than classical distance measures like DTW. Moreover, it obtains accuracies comparable to the top-scoring similarity measures (TWE and MSM).

These results are interesting because we are using plain Euclidean distance over learned representations. Given that results are comparable to or better than current distance-based approaches, the advantage of using the encoder-based representation over the raw time series is essentially threefold. First, representations are generally more compact than the raw time series. Here use representations of $k = 256$ numbers, which corresponds to a reduced-size representation for more than half of the training sets available in the repository. Second, representations are fast to compute, in the order of milliseconds

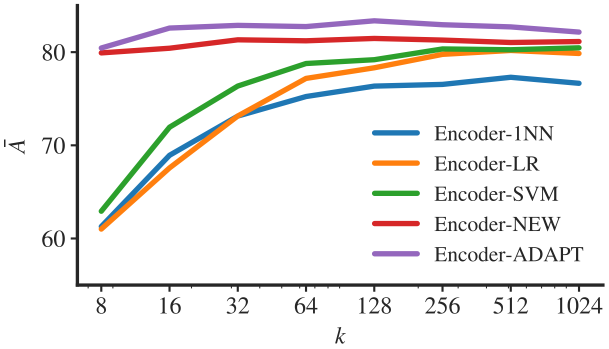

Figure 2. Effect of the encoding size. Average accuracy $\bar{A}$ as a function of representation dimensionality k. We do not consider values of $k > 1024$ as then the time series representation would be larger than almost all raw time series, thus expanding the size of the data instead of compacting it.

with a Titan Xp GPU for a hundred time series. Third, the use of Euclidean distance is quite appealing, as it is already implemented in almost all data processing libraries, with efficient methods to deal with nearest-neighbor queries.

Going back to the main results (Table 1), we observe that, if we learn a mapping from representations to classes while keeping the encoder weights frozen, the encoder-based architectures outperform dedicated classifiers with the raw time series as input (RotF). If we further adapt the encoder weights to a specific classification task, we observe that the resulting approach (Encoder-ADAPT) is competitive with the state-of-the-art. Only the best baselines beat the obtained classifiers (COTE and, in principle, HIVE-COTE [21], which is not available in the repository). These are ensemble-based methods that, compared to adapting the encoder architecture, might presumably be significantly less efficient, both at training and at testing time [1]. A further interesting thing to note is that Encoder-ADAPT outperforms COTE in number of wins, but overall has a lower average rank. This indicates that Encoder-ADAPT can perform well on a number of data sets but, nonetheless, performs poorly on others. In future work, we plan to gain insight on this question. Finally, we also observe that starting from a pre-trained encoder (Encoder-ADAPT) is better than training from scratch the exactly same architecture only with the target data set (Encoder-NEW).

5.2. Effect of Representation Size

We can also study how the size of the representations k affects the final accuracy (Fig. 2). Overall, we observe two trends, which correspond to the fact of adapting or not adapting the encoder network to the target test set. If we do not adapt the encoder network (Encoder-1NN, Encoder-LR, and Encoder-SVM), we see that, the lower the representation dimensionality, the lower the performance of the classifiers. This is to be expected, as with lower k the encoder is forced to tradeoff potentially relevant information for compactness. Contrastingly, if we adapt the encoder network (Encoder-NEW and Encoder-ADAPT), we see that the representation dimensionality does not have a clear effect on the results. There seems to be a marginally optimal operation point between $k = 64$ and $k = 256$, but the difference with the rest of operation points might not be significant.

5.3. Informal Report of Alternative Architectures

To develop the proposed encoder architecture, we started from the successful convolutional network by Wang et al. [36]. However, we found that the proposed attention strategy outperformed the original global average pooling strategy, specially for Encoder-1NN. In addition, we replaced batch normalization by instance normalization, and added a final instance normalization layer. We again found the latter to substantially help in the case of Encoder-1NN, Encoder-LR, and Encoder-SVM. An additional change with respect to that work is the introduction of max-pooling, which increased the efficiency of the encoder, and the use of larger convolutional kernel sizes, which we found yield slightly better accuracies.

In addition to the aforementioned architectures, we also experimented with a number of alternative strategies. One of the non-successful strategies we tried was to substitute the attention mechanism by a recurrent neural network. With that, we could achieve marginally better accuracies in the validation set that, nonetheless, did not generalize well to the out-of-type test sets. A further non-successful architecture change we considered was the use of causal dilated convolutions [33] with padding.

6. Conclusion and Future Work

We have studied the use of a universal encoder for time series in the specific case of classifying an out-of-sample data set of an unseen data type. We have considered the cases of no-adaptation, mapping adaptation, and full adaptation. In all cases we achieve performances that are competitive with the state-of-the-art that, in addition, involve a compact reusable representation and few training iterations. We have also studied the effect of the representation dimensionality, showing that small representations have an impact to no-adaptation and mapping adaptation approaches, but not much to full adaptation ones.

In the future, we plan to refine the encoder architecture, as well as optimizing some of the parameters we empirically use in our experiments. A very interesting direction for future research is the adoption of one-shot learning schemas [29,35], which we find very suitable for the current setting in time series classification problems. A further option to enhance the performance of a universal encoder is data augmentation, specially considering recent linear instance/class interpolation approaches [37].

References

[1] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, 2017.

[2] A. Bagnall, J. Lines, J. Hills, and A. Bostrom. Time-series classification with COTE: the collective of transformation-based ensembles. *IEEE Trans. on Knowledge and Data Engineering*, 27(9):2522–2535, 2015.

[3] A. Bagnall, J. Lines, W. Vickers, and E. Keogh. The UEA & UCR time series classification repository, 2017. URL: `http://www.timeseriesclassification.com`.

[4] B. Bakker and T. Heskes. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research*, 4:83–99, 2003.

[5] R. Caruana. Multi-task learning. *Machine Learning*, 28:41–75, 1997.

[6] D. Cer, Y. Yang, S.-Y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y.-H. Sung, B. Strope, and R. Kurzweil. Universal sentence encoder. *ArXiv*, 1803.11175, 2018.

[7] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: a survey. *ACM Computing Surveys*, 41(3):15, 2009.

[8] Y.-A. Chung, C.-C. Wu, C.-H. Shen, H.-Y. Lee, and L.-S. Lee. Audio word2vec: unsupervised learning of audio segment representations using sequence-to-sequence autoencoder. In *Proc. of the Int. Speech Communication Association Conf. (INTERSPEECH)*, pages 765–769, 2016.

[9] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680, 2017.

[10] Z. Cui, W. Chen, and Y. Chen. Multi-Scale Convolutional Neural Networks for Time Series Classification. *ArXiv*, 1603.06995, 2016.

[11] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 1126–1135, 2017.

[12] T.-L. Ha, J. Niehues, and A. Waibel. Toward multilingual neural machine translation with universal encoder and decoder. In *Proc. of the Int. Workshop on Spoken Language Translation (IWSLT)*, 2016.

[13] J. Han and M. Kamber. *Data mining: concepts and techniques*. Morgan Kaufmann, Waltham, USA, 2005.

[14] N. Hatami, Y. Gavet, and J. Debayle. Classification of time-series images using deep convolutional neural networks. *ArXiv*, 1710.00886, 2017.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: surpassing human-level performance on ImageNet classification. In *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[16] R. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, Melbourne, Australia, 2013.

[17] S. Ioffe and C. Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 448–456, 2015.

[18] E. Keogh, S. Chu, H. Hart, and M. Pazzani. Segmenting time series: a survey and a novel approach. In M. Last, A. Kandel, and H. Bunke, editors, *Data Mining In Time Series Databases*, volume 57 of *Series in Machine Perception and Artificial Intelligence*, chapter 1, pages 1–22. World Scientific, Singapore, 2004.

[19] D. P. Kingma and J. L. Ba. Adam: a method for stochastic optimization. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2015.

[20] T. W. Liao. Clustering of time series data: a survey. *Pattern Recognition*, 38(11):1857–1874, 2005.

[21] J. Lines, S. Taylor, and A. Bagnall. HIVE-COTE : the hierarchical vote collective of transformation-based ensembles for time series classification. In *Proc. of the IEEE Int. Conf. on Data Mining (ICDM)*, pages 1041–1046, 2016.

[22] P. Malhotra, V. TV, L. Vig, P. Agarwal, and G. Shroff. TimeNet: pre-trained deep recurrent neural network for time series classification. In *Proc. of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, 2017.

[23] M. McCloskey and N. Cohen. Catastrophic interference in connectionist networks: the sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165, 1989.

[24] A. Mueen. Time series motif discovery: dimensions and applications. *WIREs Data Mining and Knowledge Discovery*, 4(2):152–159, 2014.

[25] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *NIPS Workshop on The Future of Gradient-based Machine Learning Software & Techniques (NIPS-Autodiff)*, 2017.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[27] L. Y. Pratt. Discriminability-based transfer between neural networks. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 204–211. Curran Associates Inc., 1993.

[28] J. Serrà and J. Ll. Arcos. An empirical evaluation of similarity measures for time series classification.

Knowledge-Based Systems, 67:305–314, 2014.

[29] J. Snell, K. Swersky, and R. S. Zemel. Prototypical networks for few-shot learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 4077–4087. Curran Associates Inc., 2017.

[30] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112. Curran Associates Inc., 2014.

[31] S. Thrun and T. Mitchell. Lifelong robot learning. *Robotics and Autonomous Systems*, 15:25–46, 1995.

[32] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: the missing ingredient for fast stylization. *ArXiv*, 1607.08022, 2016.

[33] A. Van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. WaveNet: a generative model for raw audio. *ArXiv*, 1609.03499, 2016.

[34] A. Van den Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning. *ArXiv*, 1711.00937, 2017.

[35] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 3630–3638. Curran Associates Inc., 2016.

[36] Z. Wang, W. Yan, and T. Oates. Time series classification from scratch with deep neural networks: a strong baseline. *ArXiv*, 1611.06455, 2016.

[37] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: beyond empirical risk minimization. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2018.