School of Informatics

Informatics Research Review
Transfer Learning for Time Series Classification: A Review

Oisín Nolan

Abstract

This paper reviews a variety of methods and results in transfer learning for the task of time series classification. The methods include pre-training regimes for deep neural network models, as well as methods in source selection. Through examining and comparing experimental results, the review identifies a number of factors that can improve knowledge transfer between time series datasets, including the size of the source set, the similarity between the source and target domains, and the suitability of the loss function to the available data.

1 Introduction

Time series are ubiquitous: most-real world data has a temporal component [1]. The task of *classifying* time series data appears in many domains, such as healthcare [2, 3], human activity recognition [4], and various audio tasks [5]. Thus, advances in time series classification can have positive downstream effects on many machine learning tasks. The UCR archive [6], a large collection of labelled time series classification datasets, serves as a testament to the wide range of data types that fall into the category of time series. This archive is used as a benchmark to evaluate the performance of many time series classification models, and provides a common data source for all the experiments reviewed in this paper. We focus in this review on practicalities around employing *transfer learning* [7] in this task of time series classification. Transfer learning has found great success recently in Natural Language Processing [8, 9], and Computer Vision [10], reaching new state-of-the-art performance on many NLP tasks in particular. This success has generated interest in applying transfer learning to other domains, such as time series [11].

The primary research question explored by this paper is thus: *"to what degree, and under what conditions, does knowledge transfer across time series datasets result in improved classification performance?"*. In order to answer this, we examine a number of recent papers on this topic, each of which were chosen to meet the following criteria:

(i) It proposes a novel time series classification model. The goal is to compare the model characteristics and pre-training regimes, rather than the ways in which they have been applied.

(ii) It contains experiments on transferability of learned representations produced by the model. The setup and results of these experiments will be the main focus of the literature review.

(iii) It uses datasets from the UCR archive to perform experiments. This gives us a way to ground the experimental results, enabling easier comparison across papers.

A comparison of experimental methods and results of these papers helps shed light on the current state-of-the-art in this domain. The optimal choice of model and pre-training regime is likely to depend on contextual factors, such as labelled data availability and dataset sparsity, and so this review aims to cover both supervised and unsupervised training regimes. Firstly, we provide background information on time series classification, deep learning, and transfer learning. In each case we define key terms that are used throughout the paper. Next is the main body of the literature review, in which we describe and compare models and pre-training regimes from the chosen papers. Finally, we explore recent methods in source selection, which help to optimise the conditions for successful knowledge transfer.

2 Background

2.1 Time Series Classification

Definition 1: A *time series* is an ordered set of values $X = [x_1, x_2, ..., x_T]$. In the univariate case, x_i is a scalar, whereas in the multivariate case x_i is a vector [11]. Examples could include stock prices over time, heart rate measurements, and audio signals.

Definition 2: A time series *dataset* $D = \{(X_1, Y_1), (X_2, Y_2), ..., (X_N, Y_N)\}$ is a set of pairs of time series and associated class labels (X_i, Y_i). Y_i can take on K possible values, corresponding to the class labels, and is typically one-hot encoded [12]. This one-hot encoding can be seen as a probability distribution over the classes where all of the probability mass is assigned to the correct label.

Definition 3: *Time series classification* is the task of mapping from the input space of a given dataset to a probability distribution over the K possible class labels [12].

2.2 Deep Learning

The models discussed in this paper are all *deep neural networks*. In effect, a deep neural network is a parameterized function, $f(x;\theta) = y$, consisting of a composition of linear and non-linear transformations [13]. x is the input to the network, in this case a time-series, y is the class-label assigned to x by the network, and θ is the set of network parameters that determine how the input data is transformed to produce a class-label prediction. In the context of time series classification, these networks can be thought to consist of an encoder and classifier: the encoder serves to transform the input into a fixed-length vector representation that can easily be classified, and the classifier then maps this representation to a probability distribution over class labels. The encoders consist mostly of convolutional neural networks (CNNs) [14]. These models are characterized by the convolution operation, which can be seen as a sliding filter passing over the time series. Having extracted a number of features from the input, these models then typically aggregate the data along the time dimension to create a fixed-length representation. Recurrent Neural Networks (RNNs) [15] are also used as encoders for time series classification. These networks pass over the data sequentially, updating a latent representation one input at a time. Support vector machines and softmax regression models are typically used as the classifier.

2.3 Transfer Learning

Definition 4: Given a source domain D_S, source task T_S, target domain D_T, and target task T_T, *transfer learning* aims to improve the learning of a predictive function $f_T(.)$ that performs task T_T given inputs from D_T, using knowledge from D_S and T_S, where $D_S \neq D_T$ or $T_S \neq T_T$ [16].

Thus, TL involves *pre-training* on some source domain, a dataset or set of datasets related but not equal to a target domain, and later applying the knowledge learned to a target task. Some algorithms involve pre-training on labelled data, and others define clever loss functions that enable them to train on unlabelled data in a self-supervised manner. Using the terminology from above, the encoder might be learned via pre-training on a large source domain, with the goal of producing universally useful representations of time series data. This encoder should then produce useful representations of data in the target domain which can be easily classified, performing the target task.

3 Literature Review

The following sections contain the main literature review. This consists first of a brief description of the architecture for the models used in each of the transfer learning experiments, summarized in Table 1. Then, the setup and results of the pre-training regimes used in each experiment are discussed and compared. Finally, two recent methods in source selection are explored, serving as a remedy for some of the common issues found by the experiments.

	Base model	Supervision	Loss function	Residual connections	Activation	Temporal aggregation
FCN [17]	CNN	Supervised	Cross-entropy		ReLU	GAP
Encoder [18]	CNN	Supervised	Cross-entropy		PReLU	Attn.
ConvTimeNet [19]	CNN	Supervised	Cross-entropy	✓	ReLU	GAP
TimeNet [22]	RNN	Unsupervised	Reconstruction error		Tanh	-
Causal-CNN [20]	CNN	Unsupervised	Triplet Loss	✓	Leaky ReLU	GMP
TS2Vec [21]	CNN	Unsupervised	Hierarchical contrastive loss	✓	GELU	-

Table 1: A comparative summary of the model architectures discussed in Section 3.1.

3.1 Models

Convolutional neural networks are by far the most popular model architecture for time series classification, with the SOTA models in both the supervised and unsupervised cases consisting mostly of convolutional layers. The supervised CNN-based models reviewed are the Fully Convolutional Network (FCN) [17], Encoder model [18], and ConvTimeNet (CTN) [19]. In terms of unsupervised CNN models we consider Causal-CNN [20] and TS2Vec [21]. We also examine TimeNet [22], a slightly older model which uses RNN as the encoder, employing Gated Recurrent Units [23]. The models typically use ReLU-like activations, including standard ReLUs, PReLUs, leaky ReLUs, and GELUs [24]. The SOTA models tend to be deeper, and thus use residual connections [25] to improve gradient flow during training. Each model contains some form of normlization layer in their convolutional blocks: batch normalization [26] in FCN and CTN, instance normalization [27] in Encoder, and weight normalization [28] in Causal-CNN. When using CNNs to encode time-series, in order to produce a fixed-length vector encoding one must include some form of temporal aggregation. A variety of methods in temporal aggregation can be seen across the models studied, including Global Average Pooling [29], Global Max Pooling, and attention mechanisms. In the case of TS2Vec no aggregation is included directly, but rather a same-length encoding is produced, which could be aggregated in various ways depending on the task. A summary of these architectural details is presented in Table 1.

3.2 Pre-Training Regimes

The following sections compare the experiments in transferability done on each of the models outlined above. This consists in, for each experiment, specifying the source datasets, loss function, and details of the training procedure, which we group under the term *pre-training regime* for conciseness. Only relative performance improvement due to transfer of each model is discussed here, as each paper uses their own distinct experiments for evaluating transferability.

3.2.1 Supervised

For each of the supervised models, the cross-entropy loss function was used. This function computes a similarity between two probability distributions, in this case comparing the predicted distribution and the true distribution [30]. The predicted distribution is $p(y = k)$, the probabilities assigned to each possible class label k by the model. The true distribution, which is the label associated the input sample, assigns all probability mass to the correct answer, and so the distribution is essentially a binary vector with a 1 at the correct label and 0s elsewhere. The experiments in transferability done on the supervised models typically involve training the model on one or more source datasets, and using the obtained encoder parameters as an initialization to train on the target dataset. A new randomly-initialized classifier will then be trained on these encodings to perform the target classification task. The encoder may be further trained during the target task training, or it may remain *frozen*, which is more computationally efficient but typically less accurate, as will be seen in the result from the papers discussed next.

Fawaz et al. [31] pre-trains on each UCR dataset individually, and then fine-tunes both the encoder and classifier on every other UCR dataset. This paper thus pre-trains, fine-tunes, and evaluates 7140 models. Interestingly, Fawaz et al. find that performance on any dataset in UCR can be either improved or disimproved depending on the source set, known as *positive* or *negative* transfer, respectively. This uncertainty motivates smart source selection (see 3.3). Serrà et al. [18] looks at dataset semantics to explore whether Encoder's representations can be transferred across time series data *types*, for example EEG signals or spectrographs. They identify seven such data types in the UCR archive. Their experiments then involve using six of the seven types as the source set, with the seventh constituting the target set. Their pre-training regime uses multiple softmax classification heads on a single encoder and iterates through all six source sets during training. A fresh task-specific classifier is then fitted to a target task. They evaluate both frozen and fine-tuned encoder, finding that they get a 3% relative improvement from fine-tuning. The frozen encoder still performs well, however, significantly outperforming classical baselines, and thus providing some evidence that Encoder learns universal representations. They also compare the pre-trained Encoder to one trained from scratch, finding that the pre-trained model achieved a 2% relative improvement. ConvTimeNet (CTN) [19] uses the entire UCR archive as a source set, using multiple classification heads during training, as was the case with Encoder's regime. Having pre-trained CTN, the encoder and a new classifier are fine-tuned to each of the UCR training sets and evaluated on the associated test set. Thus, the target domain and task are subsets of the source domain and task. In this sense, the transfer is less strict than that seen in the FCN and Encoder experiments. A *from-scratch* CTN model was also trained and evaluated on each set for comparison, and overall it was observed that the pre-trained CTN had a 4% average relative improvement in accuracy versus the from-scratch model.

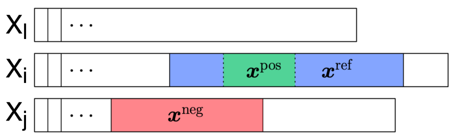
3.2.2 Unsupervised

The following pre-training regimes are *unsupervised*: the loss is calculated as a function of data in the input domain only, and so labels are not required. This permits larger source domains: both labelled and unlabelled data can be used in this regime. Unlabelled data is also typically easier to acquire, and, as such, has enabled the creation of huge pre-trained models in other domains, such as NLP [8, 9], which have been very successful. An intuition behind the success of such models is that, in seeing such a huge volume of input data, the models can learn representations that encode universal structural features, leading to success in a variety of tasks. A good encoder will cast the input into a representation space in which the inputs belonging to different classes are easily separable by the classification head. Note that these regimes could also be considered *self-supervised*, in that their loss functions essentially create labels using data from the input domain [32]. Unlike in the supervised regimes, which all used cross-entropy loss, each of the following models uses a different loss function.

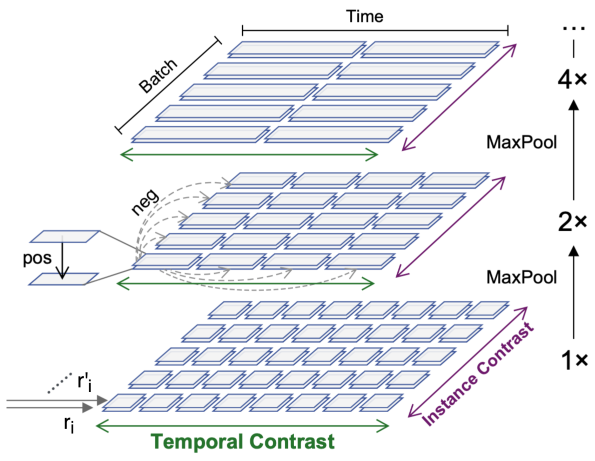
TimeNet [22] uses reconstruction error loss for pre-training. In effect, TimeNet's encoder is trained as a *sequence auto-encoder* (SAE), whose task is to encode the input sequence as a fixed-length vector representation from which the input can be decoded with minimal squared error. In order to succeed in this auto-encoding task, TimeNet's encoder must identify those features of the input that contain as much information as possible. Transferability experiments on this model involved training the SAE described above on 18 datasets chosen from UCR, each of which had length greater than 512. A further 6 datasets were used for validation of hyper-parameter settings, including the length of the embedding vector. A further 30 distinct UCR datasets were used for evaluation: fitting an SVM classifier to TimeNet encodings of the training set inputs, and evaluating accuracy on the corresponding test sets. A from-scratch SAE was also trained and evaluated on each evaluation set for comparison, and it was found that TimeNet had a 10% relative improvement on average. That being said, the standard deviation of relative change in accuracy between TimeNet and the from-scratch SAE was 22%, indicating that while there was a significant improvement overall, negative transfer is still a potential issue.

Causal-CNN uses triplet loss, which draws inspiration from Word2Vec [33], a model which generates vector representations of words based on their distribution in natural language. This loss function involves taking a number of contiguous sequences of values from the training data: first, the reference sequence, x^{ref}, then *positive samples*, x^{pos}, which are sub-sequences x^{ref}, and finally *negative samples*, x^{neg}, which are drawn randomly from other instances in the training data. Each of these is then encoded as same-length vectors representations, $f(x^{ref})$, and so on. The loss function then incurs lower loss when $f(x^{ref})$ is similar to $f(x^{pos})$, and higher loss when $f(x^{ref})$ is similar to $f(x^{neg})$. Thus, analogous to Word2Vec, sequences that appear in the same context should have similar representations, and sequences that appear in different contexts should have dissimilar representations. This should create a vector space in which similar time series features are nearby one another, and dissimilar features are far away, enabling classifiers to distinguish more easily between inputs of different classes. The pre-training regime used for Causal-CNN used a single source set from UCR, the *FordA* dataset, and then all other UCR sets as target tasks. While they only pre-trained on a single set, FordA is considerably large, with 1320 training instances each of length 500. As with [31], this resulted in some positive transfer and some negative, but never significantly degraded the performance, indicating that Causal-CNN does learn transferable representations. Interestingly, however, whether the FordA source results in positive or negative transfer appears to differ for FCN and Causal-CNN. On the *Wine* set, for example, pre-training on FordA resulted in a significant accuracy increase for FCN, but a significant decrease for Causal-CNN. This provides some evidence that, in addition to source and target domain similarity, differences in architecture or pre-training regime can affect the outcome of the transfer.

TS2Vec [21] uses a hierarchical contrastive loss function. This involves taking some sequence from a training sample, x_i, and duplicating it, x_i'. Then some values are randomly masked and cropped out of both x_i and x_i' such that they are no longer identical. The edited sequences are then encoded by TS2Vec to form two vector representations r_i and r_i'. Representations of the same time stamps in these vectors are then considered positive samples for one another, and the loss function incentivises them to be nearby in representation space. Max-pooling is then applied a number of times to these representations to make further abstract representations which cover longer timescales, each of which are also incentivised to be nearby, hence the term *hierarchical*. Negative samples are are taken from other time-steps in the same instance, as well as other instances at the same time step. Negative sampling also occurs in the max-pooled abstract representations. The motivation behind this is to capture features at multiple semantic levels and across longer time spans in the time series. The experiments in representation transferability



(a) Triplet loss, adapted from [20].



(b) Hierarchical contrastive loss, adapted from [21].

Figure 1: Illustrations of positive and negative sampling methods used by Causal-CNN and TS2Vec.

on TS2Vec are the same as for Causal-CNN: using FordA as a source set and each other UCR dataset as targets. Whether transfer is positive or negative again differs between TS2Vec and Causal-CNN, despite the architectures being quite similar – the primary difference between the two is in the loss function.

3.3 Source Selection

It is clear from the experiments reviewed in the previous sections that transfer learning can result in either and improvement or disimprovement in performance if source sets are chosen naively. Motivated by this drawback, the following sections will discuss two methods of *source selection* – techniques for choosing source sets that increase the likelihood of positive transfer. These techniques shed light on exactly those *conditions* in which transfer learning can be successful in time series classification. The methods discussed are both model-agnostic, and so can be applied to any of the models described in this paper, and any future models that fit the encoder-classifier schema. Note that the primary advantage in these methods is that they are more computationally efficient than training and evaluating on many possible source domains. The ideal method of source selection, given enough compute, would be to pre-train on all possible subsets of available and choose the one that gives best performance on the target task, although this is typically computationally infeasible.

3.3.1 Inter-dataset Similarity

Fawaz et al. [31] propose *Inter-Dataset Similarity* (IDS) as a method of source selection. This uses the Dynamic Time Warping (DTW) algorithm [34] as a measure of similarity between two time series. DTW aims to identify structural similarities in the time series, accounting for slight misalignments in similar features. IDS calculates a *prototype* time series for each class in a given dataset by averaging over the set of time series belonging to that class. DTW Barycenter Averaging (DBA) [35] finds the time series which has the minimum DTW distance to all other time series in the set that it's averaging over, and is used to calculate this class prototype for each class in the dataset. The distance between two datasets is then calculated as the minimum distance between prototypes, for all prototypes in that set. Using this method, they achieved positive transfer in 71/85 cases, rather than approximately 50/50 pos neg transfer. This provides evidence that this method can help increase confidence that transfer will improve performance on future samples – that good conditions are those in which source domain and target domain are similar.

3.3.2 Source Model Selection

Meiseles and Rokach [36] propose *Source Model Selection* (SMS), a method that helps choose between a number of pre-trained encoders. In this sense, their method takes a more holistic view of source selection, considering both the source domain on which the encoder was trained, as well as the application of that pre-trained encoder on the target task. This is achieved by comparing the *Mean Silhouette Coefficient* (MSC) [37] of clusters in representation space produced by the input from the target domain. The intuition is that if a given encoder produces well-defined, easily separable clusters, then it will likely lead to good performance on the task. The MSC for a dataset is calculated as the average silhouette coefficient over representations of all samples in the training set of the target task. The silhouette coefficient for a representation vector r_i is calculated by comparing the average cosine similarity of those representations in the same class as r_i to that of the representations from the next nearest cluster. In order for the MSC of a dataset to be high, representation vectors must be similar to those vectors in the same class and different to those vectors in other classes. Meiseles and Rokach. use top-1 accuracy as a performance metric, finding that SMS ranks the best source first in 7.1% of cases, whereas IDS only does so in 4.7%. Thus, SMS shows a significant performance increase on IDS. This method differs from IDS in important ways, however, making the comparison somewhat unfair. In particular, SMS is to be used in the case that one is choosing between a number of possible existing pre-trained models, rather than choosing which dataset(s) to pre-train on. If one already has the pre-trained models, this is useful. If not, it would be more computationally efficient to use IDS, rather than pre-training on the many possible source domains we might want to consider. Given that computational efficiency is a primary motivator of methods in source selection, this is significant. An optimal strategy might employ both methods, using IDS to refine the space of possible source domains, pre-training a few models on the source domains with high IDS, and then using MSC on the representations to choose a final model.

4 Discussion & Future Work

The experiments described above cover a wide range of pre-training regimes, and some patterns emerge within the results. In cases with small source datasets, e.g. FCN, Causal-CNN, and TS2Vec, the ratio of positive to negative transfer was approximately even. In cases with large source datasets the transfer was mostly positive, with CTN in particular causing negative transfer in only 6/41 cases, indicating that larger source sets may decrease the likelihood of negative transfer. Overall, it is clear from the experiments discussed that TSC can benefit from transfer learning. The experiments reviewed give approximate answers to the research questions: pre-trained models typically increased relative accuracy by 2-10%, and source selection methods like IDS can help understand how to optimise the data or model conditions for transfer learning. However, further experimentation is needed to reconcile certain findings across the literature. For example, IDS source selection served as an indication of which source datasets might result in positive transfer for FCN, but as mentioned in section 3.2.2, using the same source dataset and target task can result in positive or negative transfer depending on the choice of model.

Progress in this direction could be made via an empirical study that uses a standard evaluation framework to compare the effects of pre-training regimes and architectural choices on the level of transferability achieved by the encoder. Such a study could shed light on a number of research questions, for example: given a set of possible sources and a target, should supervised or unsupervised pre-training be preferred? The choice of model might be affected by such things as label sparsity [20], data type, and similarity between source and target domains. Another question could further study dataset semantics: Serrà et al. [18] showed that representations can be transferred *across types* of dataset, but would transfer *within types* of dataset result in significant performance gains? Furthermore, the relationship between dataset type and measures of datatset similarity (e.g. IDS) could be studied: does a common dataset type serve simply as an indication of dataset similarity? In NLP, massive unsupervised pre-trained models have produced state-of-the-art results in many tasks. Further work could explore whether such universal representations of time-series data can be learned by, for example, TS2Vec [21]. Finally, labels are required to compute both IDS and cluster MSC, making these source selection methods less suitable for the unsupervised case. Future work could look at developing and evaluating unsupervised dataset similarity metrics, for example, calculating a prototype time series for the entire dataset, rather than per class.

5 Conclusion

This review has presented and compared a range of methods in pre-training time series classifiers for transfer learning. In particular, it has explored the gains in classification performance that have been achieved using a variety of pre-training regimes, focusing in each case on the specific conditions that led to those gains. It was discovered that relevant factors include the similarity of the source and target domains, the size of the source domain, and whether the encoder is further trained during fine-tuning. It is also clear in the experiments that transfer may have either a positive or negative effect on target tasks if small source datasets are used. As source datasets scaled up, transfer tended to be more often positive. Methods in source selection were also reviewed. These methods involved computing similarity metrics between datasets, as well as directly assessing the quality of the representations produced by pre-trained encoders. Using these methods, one can significantly increase the likelihood of positive transfer, thus shedding light on exactly those conditions in which knowledge may be productively transferred from one domain to another. The primary conclusion of this work is that, while transfer learning shows potential in the time series classification, the optimal choice of model and source domain given a specific task remains somewhat nebulous due to a lack of comparable experimental methods. In order to fully understand the conditions under which transfer learning is most likely to improve performance, this paper suggests carrying out an empirical study using a standard set of experiments on the models discussed above.

References

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

[2] Alvin Rajkomar, Eyal Oren, Kai Chen, Andrew M Dai, Nissan Hajaj, Michaela Hardt, Peter J Liu, Xiaobing Liu, Jake Marcus, Mimi Sun, et al. Scalable and accurate deep learning with electronic health records. *NPJ Digital Medicine*, 1(1):1–10, 2018.

[3] Daniel Severo, Flávio Amaro, Estevam R Hruschka Jr, and André Soares de Moura Costa. Ward2icu: A vital signs dataset of inpatients from the general ward. *arXiv preprint arXiv:1910.00752*, 2019.

[4] Henry Friday Nweke, Ying Wah Teh, Mohammed Ali Al-Garadi, and Uzoma Rita Alo. Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Systems with Applications*, 105:233–261, 2018.

[5] Daniele Barchiesi, Dimitrios Giannoulis, Dan Stowell, and Mark D Plumbley. Acoustic scene classification: Classifying environments from the sounds they produce. *IEEE Signal Processing Magazine*, 32(3):16–34, 2015.

[6] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

[7] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[9] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[10] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12299–12310, 2021.

[11] Manuel Weber, Maximilian Auch, Christoph Doblander, Peter Mandl, and Hans-Arno Jacobsen. Transfer learning with time series data: A systematic mapping study. *IEEE Access*, 2021.

[12] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[14] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2, 1989.

[15] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[16] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[17] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE, 2017.

[18] Joan Serrà, Santiago Pascual, and Alexandros Karatzoglou. Towards a universal neural network encoder for time series. In *CCIA*, pages 120–129, 2018.

[19] Kathan Kashiparekh, Jyoti Narwariya, Pankaj Malhotra, Lovekesh Vig, and Gautam Shroff. Convtimenet: A pre-trained deep convolutional neural network for time series classification. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[20] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. Unsupervised scalable representation learning for multivariate time series. *arXiv preprint arXiv:1901.10738*, 2019.

[21] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. Ts2vec: Towards universal representation of time series. *arXiv preprint arXiv:2106.10466*, 2021.

[22] Pankaj Malhotra, Vishnu TV, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Timenet: Pre-trained deep recurrent neural network for time series classification. *arXiv preprint arXiv:1706.08838*, 2017.

[23] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[24] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[26] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

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

[28] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29:901–909, 2016.

[29] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[30] Kevin P Murphy. *Machine learning: a probabilistic perspective.* MIT press, 2012.

[31] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Transfer learning for time series classification. In *2018 IEEE international conference on big data (Big Data)*, pages 1367–1376. IEEE, 2018.

[32] Yu Sun, Eric Tzeng, Trevor Darrell, and Alexei A Efros. Unsupervised domain adaptation through self-supervision. *arXiv preprint arXiv:1909.11825*, 2019.

[33] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[34] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

[35] François Petitjean, Alain Ketterlin, and Pierre Gançarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern recognition*, 44(3):678–693, 2011.

[36] Amiel Meiseles and Lior Rokach. Source model selection for deep learning in the time series domain. *IEEE Access*, 8:6190–6200, 2020.

[37] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.