DOI 10.5162/sensoren2019/P2.9

# Comparison of deep feature extraction techniques for varying-length time series from an industrial piercing press

Christian Thiel[1,2], Carolin Steidl[2], Bernd Henning[2]
[1] BENTELER Steel/Tube GmbH, Residenzstraße 1, 33104 Paderborn, Germany
[2] Measurement Engineering Group, Paderborn University, Warburger Str. 100, 33098 Paderborn, Germany

## Abstract

The continuous refinement of sensor technologies enables the manufacturing industry to capture increasing amounts of data during the production process. As processes take time to complete, sensors register large amounts of time-series-like data for each product. In order to make this data usable, a feature extraction is mandatory. In this work, we discuss and evaluate different network architectures, input pre-processing and cost functions regarding, among other aspects, their suitability for time series of different lengths.

**Keywords: Feature Extraction, Dynamic Time Warping, Masking, Neural Networks**

## 1 Introduction

Companies in the manufacturing industry measure increasing amounts of product and process data in order to improve their processes. This data is used to stabilize production, identify factors on the product quality and improve process efficiency. As processes take time to complete, sensors register large amounts of time-series-like data for each product. In contrast to single-value features measured once for each product, temporally or locally resolved measurements provide much richer information about the underlying process. The high complexity and dimensionality of these time series of different lengths point towards the necessity of Feature Extraction (FE) methods.
Traditionally, domain experts would design handcrafted composite features for each time series. This process is prone to errors and unlikely to capture every detail. Recent advances in deep learning have allowed various Neural-Network(NN)-based auto encoding FE techniques to emerge, allowing FE in an objective unified way.
These new architectures include some, that can natively handle varying length inputs. Those architectures include Recurrent Neural Networks (RNNs), which we discuss in section 4. Besides RNNs, also Fully Convolutional Networks are suited for varying length inputs [7]. However, their hidden dimensionality is not constant, so that less features are produced for shorter inputs. As this simply shifts the problem of unequal dimensionality to the next step of analysis, we do not evaluate these methods further.
For common NN-based Autoencoders (AE) however, handling varying-length time series still poses major problems, as only fixed-dimensional inputs are natively supported. While the use of zeropadding or interpolation already enables the use of common AE, we show that naive application of these methods is not ideal as new patterns are introduced to the original time series. In this work we discuss and compare two possibilities to overcome this shortcommming:

1. Zeropadding introduces new patterns and discontinuities to time series. We propose the use of masking to avoid training on these artificially created patterns in subsection 3.1.
2. Interpolation stretches and compresses patterns in time series. This effect can be mitigated using elastic cost functions such as Dynamic Time Warping (DTW) as discussed in subsection 3.2.

The outline of the paper is as follows: In section 2 the process and data preprocessing is discussed. Our proposed modifications to common AE architectures are then motivated and presented in section 3. With RNNs a more complex network architecture is discussed in section 4. Finally our experiments are presented in section 5.

## 2 Process and Data Description

In the tube production process a piercing press is used to form a hole in a solid block of hot metal. The piercing step is crucial in production and strongly influences many critical quality factors such as the eccentricity of the final tube. Thus, data from the piercing process is of importance in many analysis tasks.
As the process duration of the piercing press depends on the size of the block as well as the chosen process speed, signals resulting from equidistant measurements in time inherently have different

lengths. In this work we evaluate all approaches on 100000 instances of pressing force time series from the tube production process. The sampling rate for all time series is 20 milliseconds. The resulting time series have lengths between 156 and 258 values.
Signals from industrial processes often contain outliers due to the rough environments. In order to be able to use non-robust estimators such as means or standard deviations, a Hampel-Filter is applied to each time series during preprocessing. All identified outliers are then replaced using interpolation.
The importance of time series normalization when measuring time series similarities has been shown multiple times in the literature [6]. When using DTW or the Mean Squared Error (MSE) it is evident that already a small offset between two time series leads to potentially large distances. This problem becomes more significant, the longer the time series gets. Usually, each time series is thus normalized to zero mean and unit variance. For the piercing press data used in this work we deviate from this approach slightly and use trimmed mean and standard deviation as more robust estimators for normalization. This mainly results in the normalization focusing on the center part of the signal without taking edge effects such as very low pressing forces at the start or end into account.

# 3 Adjustments for common architectures

Common AE based techniques, as for example Deep Convolutional Autoencoders (DCAE) and Fully Connected Autoencoders (FCAE), only support fixed-dimensional inputs [4]. This restriction originates from the matrix multiplication in Fully-Connected (FC) Layers.
In order to use DCAE and other architectures that require a fixed input dimensionality, simply equalizing the length of time-series prior to FE naturally comes to mind. In this work we consider zero-padding and interpolation as two means to equalize the lengths of time series. While this already enables the use of powerful techniques such as the FCAE and DCAE, the results might not be perfect. We provide two intuitive explanations that support this assumption:

I. Zeropadding introduces new patterns to the time series. This does not only relate to the perfectly straight line in the zeropadded part of the signal, but especially to the transition from signal end to zero. Even for standardized time series with zero mean and unit variance significant discontinuities can occur as seen in the curve on top of Figure 1. Even small inaccuracies of the discontinuities position in the AE's reconstruction yield high losses. Thus the AE spents a reasonable amount of information in the latent representation to exactly model the discontinuity while performance in meaningful areas suffers.

II. Interpolation causes characteristic local patterns in the original time series to be stretched or compressed depending on the difference of time series length to chosen input dimensionality. These alterations of the original pattern can not be matched as good as the original pattern by a single filter mask. Thus either more filters representing different alterations of the same original pattern are required or the overall performance can be expected to drop.

We propose the use of masking when zeropadding is used to counteract I in subsection 3.1. Also Dynamic Time Warping (DTW) based losses in combination with interpolation to counteract II are discussed in subsection 3.2.

## 3.1 Masking layer

Masking is already used to simplify training of RNNs with inputs of varying dimensionality. It is implemented for RNNs in popular frameworks such as TensorFlow (`tf.keras.layers.masking`).
We propose to use masking in combination with zeropadding also for common architectures such as DCAE. In our masking layer, we simply force all values in the padded part of the reconstructed signal to be zero before calculating the loss and performing back propagation. Errors in the reconstruction of the padded parts as well as the artificially introduced discontinuity no longer contribute to the cost function and thus do not affect training. The overall DCAE model including the masking layer is shown in Figure 1.

## 3.2 Dynamic Time Warping as loss

Dynamic Time Warping (DTW) is widely used as an elastic similarity measure in time series analysis. While DTW is already able to compare time series of unequal lengths, this feature has no relevance for common architectures as the output always has a fixed dimensionality so that unequal lengths never occur. However, as an elastic measure, DTW is very well suited to compensate for stretched or compressed patterns introduced by interpolation. Following, we firstly give a brief introduction to the classic DTW approach in subsubsection 3.2.1, as we have already shown its suitability for our data in [10]. As this classic DTW approach is not differentiable, it cannot be used for training using backpropagation. Secondly we introduce the recently published soft-DTW algorithm in subsubsection 3.2.2, which is a differentiable modification of DTW [2].

### 3.2.1 Classic Dynamic Time Warping

To calculate the DTW distance, firstly the signal difference matrix $\mathbf{S}$ containing all pairwise distances of two time series $\vec{x} = \{x_1, \ldots, x_M\}$ and $\vec{y} = \{y_1, \ldots, y_N\}$:

$$\mathbf{S}_{m,n} = d(x_m, y_n) \quad \forall\, m = 1, \ldots, M \quad n = 1, \ldots, N, \quad (1)$$

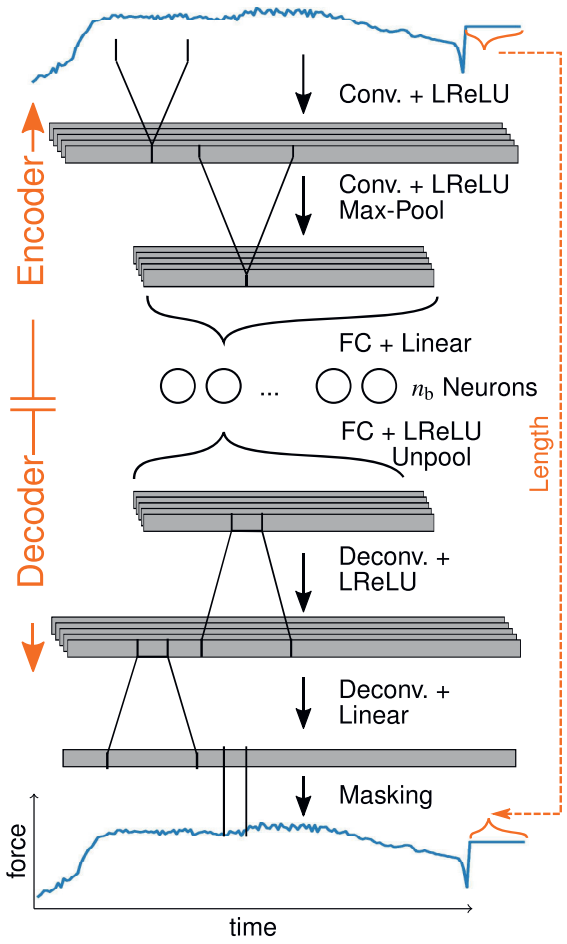



***Figure 1:*** *Deep Convolutional Autoencoder (DCAE) with additional masking to avoid training on zero-padded parts of the input signal.*

where $d$ is usually the Euclidean Distance (ED). For brevity we denote $d(x_m, y_n)$ as $d_{m,n}$. The DTW alignment of the series is defined by a warping function $\mathcal{W} = \{\vec{w}_1, ..., \vec{w}_K\}$ through $\mathbf{S}$, which results in the lowest cumulated (weighted) distances:

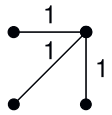
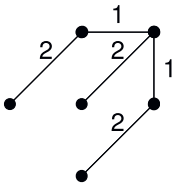
$$\mathcal{W}^* = \underset{\vec{w}_1,...,\vec{w}_K}{\operatorname{argmin}} \sum_{k=1}^{K} d(x_{w_{k,1}}, y_{w_{k,2}}), \quad \vec{w}_k = \{w_{k,1}, w_{k,2}\}, \quad (2)$$

where each warping function has to fullfill the well-known conditions [8]

1. Monotony: $w_{k,1} \geq w_{k-1,1}$ and $w_{k,2} \geq w_{k-1,2}$
2. Continuity: $w_{k,1} - w_{k-1,1} \leq 1$ and $w_{k,2} - w_{k-1,2} \leq 1$
3. Boundary conditions: $w_1 = (1,1)$ and $w_K = (M,N)$

These conditions allow a reformulation as a Dynamic Programming (DP) algorithm using Bellman's equation through a cumulated signal difference matrix $\tilde{\mathbf{S}}$

***Table 1:*** *Constraining the slope of DTW warping functions using stepping patterns [8].*

| Formula for $\tilde{\mathbf{S}}_{m,n} =$ | Schema |
|---|---|
| Symmetric-P0 (SymP0) | |
| $\min \begin{bmatrix} \tilde{\mathbf{S}}_{m,n-1} + d_{m,n} \\ \tilde{\mathbf{S}}_{m-1,n-1} + d_{m,n} \\ \tilde{\mathbf{S}}_{m-1,n} + d_{m,n} \end{bmatrix}$ |   |
| Symmetric-P1 (SymP1) | |
| $\min \begin{bmatrix} \tilde{\mathbf{S}}_{m-1,n-2} + 2d_{m,n-1} + d_{m,n} \\ \tilde{\mathbf{S}}_{m-1,n-1} + 2d_{m,n} \\ \tilde{\mathbf{S}}_{m-2,n-1} + 2d_{m-1,n} + d_{m,n} \end{bmatrix}$ |   |

calculated as follows:

$$\tilde{\mathbf{S}}_{0,0} = 0, \quad \tilde{\mathbf{S}}_{m,0} = \infty \; \forall \; m > 0, \quad \tilde{\mathbf{S}}_{0,n} = \infty \; \forall \; n > 0 \quad (3)$$

$$\tilde{\mathbf{S}}_{i,j} = \min \begin{bmatrix} \tilde{\mathbf{S}}_{m,n-1} + d_{m,n} \\ \tilde{\mathbf{S}}_{m-1,n-1} + d_{m,n} \\ \tilde{\mathbf{S}}_{m-1,n} + d_{m,n} \end{bmatrix}, \quad (4)$$

where Equation 4 is known as the stepping pattern. The commonly used unrestricted Symmetric-P0 stepping pattern can easily lead to unreasonable mappings for real world data, as it allows arbitrary many points of one time series to map to just one point of the other. The original authors of DTW proposed the use of more robust steppings such as Symmetric-P1, which restrict the slope of the warping function and thus prohibits degenerated alignments [8, 10]. A visualization of the Symmetric-P1 in comparison to the default Symmetric-P0 stepping is given in Table 1. Finally, the overall DTW cost is equivalent to the corner element $\tilde{\mathbf{S}}_{M,N}$ of the cumulated signal difference matrix. It is equivalent to the cumulated costs of the optimal warping function $\mathcal{W}^*$:

$$\mathbf{dtw}(\vec{x}, \vec{y}) = \tilde{\mathbf{S}}_{M,N} = \min_{\vec{w}_1,...,\vec{w}_K} \sum_{k=1}^{K} d(x_{w_{k,1}}, y_{w_{k,2}}) \quad (5)$$

### 3.2.2 Differentiable Dynamic Time Warping

DTW, as presented in subsection 3.2.1, is not continuously differentiable due to the minimum operator in Equation 2 and Equation 4 respectively. While for a fixed warping path a derivative with respect to one of the time series can easily be calculated, the minimum operator leads to discontinuities for values of $\vec{x}$, where small changes result in a different warping path. Following the idea of the Global Alignment Kernel (GAK) introduced in 2011 [1], $\gamma$-soft-DTW considers the set

of all possible alignments $\mathcal{W}_{M,N}$ instead of only the one alignment resulting in minimal costs $\mathcal{W}^*$ [2].
In order to incorporate all warping paths, the authors introduce a generalized soft minimum operator with smoothing parameter $\gamma \geq 0$ [2]:

$$\min^{\gamma}\{a_1,\ldots,a_n\} := \begin{cases} \min_{i\leq n} a_i, & \gamma = 0, \\ -\gamma \log \sum_{i=1}^{n} e^{-a_i/\gamma} & \gamma > 0. \end{cases}, \tag{6}$$

which for $\gamma = 0$ results in a regular minimum operator and for $\gamma > 0$ in a soft-minimum incorporating all values in the given set. With the generalized soft minimum defined, $\gamma$-soft-DTW follows as:

$$\mathbf{dtw}_{\gamma}(\vec{x},\vec{y}) := \min^{\gamma}\left\{\sum_{k=1}^{K} d(x_{w_{k,1}}, y_{w_{k,2}}), \{\vec{w}_1,\ldots\vec{w}_K\} \in \mathcal{W}_{M,N}\right\}, \tag{7}$$

where $\mathcal{W}_{M,N}$ denotes the set of all possible alignment paths for two signals of lengths $M$ and $N$. For the reformulation as a DP algorithm, the minimum operator of Equation 4 is simply replaced by the soft minimum operator with the corresponding parameter $\gamma$.
Finally, for $\gamma > 0$, Equation 7 can be explicitly differentiated as shown in [2]. This enables its use as a loss in any gradient based setting, such as backpropagation in NN.

# 4 Recurrent Network Architectures

RNNs are inherently able to handle varying lengths inputs. In contrast to common architectures where the whole input sequence is fed to the network simultaneously, RNNs are fed sequentially. In a time series setting, one value of the time series is fed after the other. RNNs are obtained by using neurons outputs of one time step as inputs of the next. In bidirectional RNNs also the outputs from future time steps are used as inputs, at the cost of loosing causality [9]. Additionally, memory blocks are often introduced, to capture dependencies over time explicitly.
To train RNNs, gradients do not only have to be propagated through the network itself but also through all time steps. The algorithm used for training is referred to as Backpropagation-Through-Time (BPTT). Due to the unfolding in time, RNNs can be seen as very deep networks, although with strong parameter sharing. RNNs high depth leads to vanishing and exploding gradient problems. Multiple techniques have been introduced to counteract these problems. One architecture that gained significant attention are Long-Short Term Memory Networks (LSTMs) [5].
For FE, the encoder is realized as a Bidirectional LSTM (Bi-LSTM). To obtain a fixed dimensional representation for the varying length inputs, the last outputs of the deepest LSTM layers are interpreted as features. The concatenated outputs of the forward and backward models is the overall representation of the input time series in the feature space. For the decoder, this representation is repeated to match the length of the input time series, and then fed to the decoding forward and backwards models in each timestep. The final prediction for a timestep is obtained by collecting the outputs of the deepest layer of the forward and backward LSTM respectively, and feeding them into a single neuron with linear activation. The weights of the neuron is identical for every timestep. The number of steps for which the decoder is run is equivalent to the length of the input.

***Table 2:*** *Model Architectures. Encoder architecture is abbreviated as EA. The decoder is build transposed to the encoder without weight sharing. Activations before output and feature representation are always linear. Convolutional Layers are denoted as Conv(Filters, Filtersize, Stride).*

| Model | Architecture |
|---|---|
| Bi-LSTM | Activation: tanh<br>Recurrent Activation: hard sigmoid<br>EA: $2\text{x}50 \rightarrow 2\text{x}50 \rightarrow 2\text{x}5$<br>Feature Dimension: 10<br>Number of Parameters: 170,981 |
| DCAE | Activation: Leaky-ReLU<br>EA: $\text{Conv}(8,6,2) \rightarrow \text{Conv}(16,6,2) \rightarrow 10$<br>Feature Dimension: 10<br>Pooling: Not used.<br>Number of Parameters: 23,515 |
| FCAE | Activation: Leaky-ReLU<br>EA: $200 \rightarrow 200 \rightarrow 10$<br>Feature Dimension: 10<br>Number of Parameters: 169,020 |

# 5 Experiments

In order to validate our proposed adjustments from section 3, experiments on zeropadded (Experiment 1) and interpolated (Experiment 2) signals are performed. Results for the RNN are obtained using the original (normalized) signals without zeropadding or interpolation.

## 5.1 Experiment Description

Besides experiment specific performance measures discussed below, also epochs until convergence and training time per epoch are taken into account. In this work we define convergence as the last epoch which lead to an improvement of more than one percent compared to the previously best result. If no such improvement occurs for more than 5 epochs, the training is stopped. All architectures used for the experiments are shown in Table 2. All models are trained using the Adamax optimizer with a batch size of 200.
**Experiment 1:** In order to asses the influence of masking for zeropadded signals, a comparable metric for models trained with and without masking is

required. We use the mean squared error (MSE) between the prediction and the true signal in the non-zeropadded part of the signal. We motivate this twofold: Firstly, the accuracy of the zeropadded part is simply stated irrelevant. Secondly, as discussed in subsection 3.1, the length of the signal can be seen as an additional feature and is thus known. Using this knowledge any error in the reconstruction of the zeropadded part can be rectified.

**Experiment 2:** Strict dynamic time warping using stepping restrictions and warping windows result in reasonable alignments for pressing force signals [10]. However, neither the score of classic DTW using other steppings than SymP0 nor the soft DTW score can be interpreted easily as some *average error of the prediction in the original domain of the data*, following the MSE notion. In order to obtain such interpretable errors, we use the resulting MSE of the *aligned* signals. The alignment is obtained using classic DTW with SymP1 steppings and a Sakoe-Chiba bands of width 30. We denote the resulting metric as $\mathrm{MSE}_{\mathrm{DTW}}$. During training we use the soft-DTW implementation by the original authors written as a C-Extension for Python.

## 5.2 Results

The results of both experiments and the reference Bi-LSTM are shown in Table 3.

**Experiment 1:** For both, the DCAE and FCAE masking leads to overall better results. The convergence time of both models is significantly lower and additionally also the overall MSE reduced. These results directly support assumption I from section 3. While training time increases when masking is used, the overall training is still extremely fast.

**Experiment 2:** Using soft-DTW in training leads to significantly lower DTW losses, even for strict steppings and warp path regulations which were not used during training. This shows that soft-DTW is a suitable replacement in training for non-differentiable strict DTW. Please note that as described in subsection 5.1 $\mathrm{MSE}_{\mathrm{DTW}}$ follows the notion of average error per point in the original data domain, and is thus directly comparable to the MSE. Supporting assumption II, $\mathrm{MSE}_{\mathrm{DTW}}$ is consistently lower when soft-DTW is used for training. The significant increase in training time results from the high complexity of soft-DTW and even supersedes the slow training of RNNs.

***Table 3:*** *Results of the experiments. Columns: Proposed Method Applied (P), Epochs until Convergence (EC), training Time per Epoch (TE), Mean Squared Error (MSE), Mean Squared Error after constrained DTW alignment ($\mathrm{MSE}_{\mathrm{DTW}}$). The proposed method in Experiment 1 is masking, in Experiment 2 training with soft-DTW.*

| Model | P | EC | TE | MSE | $\mathrm{MSE}_{\mathrm{DTW}}$ |
|---|---|---|---|---|---|
| Experiment 1 (zeropadded data) | | | | | |
| DCAE | yes | 19 | 2 s | **0.10** | |
| DCAE | no | 30 | 1,7 s | 0.12 | |
| FCAE | yes | **12** | 1,4 s | 0.14 | |
| FCAE | no | 26 | **1,1 s** | 0.18 | |
| Experiment 2 (interpolated data) | | | | | |
| DCAE | yes | 29 | 25 min | 0.19 | **0.08** |
| DCAE | no | 20 | 1,7 s | **0.13** | 0.10 |
| FCAE | yes | 18 | 25 min | 0.20 | 0.10 |
| FCAE | no | **8** | **1,1 s** | 0.15 | 0.12 |
| RNN | | | | | |
| Bi-LSTM | no | 8 | 16 min | 0.14 | 0.12 |

# 6 Literature


[1] M. Cuturi: Fast global alignment kernels, *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 929-936.

[2] M. Cuturi, M. Blondel: Soft-DTW: a Differentiable Loss Function for Time-Series, *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70, Mar. 2017 .

[3] H. A. Dau, E. Keogh, K. Kamgar, C. Yeh, Y. Zhu, G. Shaghayegh, C. Ratanamahatana, A. Chotirat, Yanping and B. Hu: The UCR Time Series Classification Archive, Oct. 2018.

[4] J. Geng, J. Fan, H. Wang, X. Ma, B. Li and F. Chen: High-Resolution SAR Image Classification via Deep Convolutional Autoencoders, *Proceedings of the IEEE conference on Geoscience and Remote Sensing Letters*, 2015.

[5] S. Hochreiter and J. Schmidhuber: Long short-term memory, *Neural computation*, 9.8, 1997, pp. 1735-1780.

[6] E. Keogh, S. Kasettey: On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration, *Data Mining and Knowledge Discovery*, Vol. 7, Oct. 2003.

[7] J. Long, E. Shelhamer and T. Darrell: Fully Convolutional Networks for Semantic Segmentation, *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[8] H. Sakoe and S. Chiba: Dynamic programming algorithm optimization for spoken word recognition, IEEE transactions on acoustics, speech, and signal processing, Vol. 26, No. 1, 1978, pp. 43-49

[9] M. Schuster and K. Paliwal, Bidirectional recurrent neural networks, *IEEE Transactions on Signal Processing*, Vol. 45, No. 11, 1997, pp. 2673–2681.

[10] C. Thiel, N. Feldmann and B. Henning: Extraction of Interpretable Features from Temporal Measurements using Approximate Prototypes, *Sensors and Measuring Systems*, 19th ITG/GMA-Symposium, Nuremberg, Germany, 2018, pp. 536-540.