Learning Self-Supervised Task Progression Metrics: a Case of Folding Clothing

Andreas Verleysen* Matthijs Biondina* Francis wyffels*

Abstract

An important challenge for smart manufacturing systems is finding relevant metrics that capture task quality and progression for process monitoring to ensure process reliability and safety. Data-driven process metrics construct features and labels from abundant raw process data, which incurs costs and inaccuracies due to the labelling process. In this work, we circumvent expensive process data labelling by distilling the task intent from video demonstrations. We present a method to express the task intent in the form of a scalar value by aligning a self-supervised learned embedding to a small set of high-quality task demonstrations. We evaluate our method on the challenging case of monitoring the progress of people folding clothing. We demonstrate that our approach effectively learns to represent task progression without manually labelling sub-steps or progress in the videos. Using case-based experiments, we find that our method learns task-relevant features and useful invariances, making it robust to noise, distractors and variations in the task and shirts. The experimental results show that the proposed method can monitor processes in domains where state representation is inherently challenging.

1 Introduction

Modern manufacturing systems are becoming increasingly complex due to high requirements on process quality and economical incentives [1]. In order to ensure the reliability and quality of the outcome of industrial processes, process monitoring techniques are utilized [2]. Among process monitoring methods, data-driven process monitoring methods are a popular approach as they do not require modelling complex physical processes and can conveniently be collected with sensors and cameras. Additionally, monitoring metrics can serve as a learning signal to automate parts of the production process using learning-based approaches such as reinforcement learning.

*IDLab-AIRO – Ghent University – imec, Technologiepark-Zwijnaarde 126, 9052 Zwijnaarde, Belgium
Corresponding author: andreas.verleysen@ugent.be

This version of the article has been accepted for publication, after peer review (when applicable) and is subject to Springer Nature's AM terms of use, but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: https://doi.org/10.1007/s10489-022-03466-8

A significant problem with data-driven process monitoring is the required labelling process associated with generating the data. For example, if the task consists of a robot or human worker folding clothing for packaging, a process monitoring system needs to perform state estimation of the cloth. However, cloth has an infinite amount of configurations due to its deformations [3]. The deformations also cause parts of the cloth to be occluded, making state estimation even more difficult. Additionally, some problems do not lend well to manually constructing a quality metric. In the given example, it is non-trivial to map the number of wrinkles to a quality measure. Implementing such quality systems to monitor task progression is non-trivial: it requires a significant engineering effort and process knowledge in order to capture many ill-defined components, some of which are hard to measure.

A solution to capture the many required details involved in capturing task progression can be found in how humans and animals acquire new skills. Primates and humans are known to possess a neuron mirror system that is at the basis of mirroring actions and behaviour of other individuals [4]. This idea has been transferred to the field of robotics [5] in which a robot can acquire new skills by imitating the behavior of the demonstrator. However, learning to solve a task from experts is suspect to copying the exact manipulations of the demonstrator. This is due to the learning agent not understanding the essence of the task. Moreover, no guidance is available when the agent arrives in unseen areas of the state space. A final problem preventing transferring expert demonstrations across actors is the correspondence problem [6]: the embodiment of the demonstrator often differs from the learning actor. For example, the kinematic chain of a delta robot differs significantly from a human arm. Consequently, learning from demonstrations requires a mapping between different morphologies. Hence, a task progression metric needs to be invariant to the actor executing the task (i.e., the correspondence problem) and needs to be able to generalize to unseen situations.

In order to construct process monitoring metrics from demonstrations that capture the task intent, we need to (1) learn task-relevant representations, (2) solve the correspondence problem, and (3) translate the representation to a metric that indicates task progression and solution quality. This task progression metric can then be used for process monitoring. Additionally, this metric can be used as a reward function in a reinforcement learning setting such that the task can be learned from environment interaction.

To address the challenge of unsupervised learning of task intent, we propose a method to learn a task progression metric from human demonstrations. We do this in a self-supervised way that does not require manually labelling video frames. Central is the idea of contrastive learning in which pairs of observations that are semantically similar are close in the embedding space compared to dissimilar observations. This can be achieved by

using time as a supervisory signal, for example, Time-Contrastive Networks (TCN)[7]. TCNs produce task-relevant features, which we then align to a small set of expert demonstrations using a modified version of Dynamic Time Warping (DTW). Finally, we distil a scalar progression metric by querying the ensemble of experts for predicting task progress.

The contributions of this paper are:

- *A novel method to generate task progression metrics from video without labelling data:*: We propose an integrated approach to overcome expensive data labelling in data-driven process monitoring in order to generate self-learned task progression metrics. Our approach also allows decoupling reward and policy learning in reinforcement learning.
- *The first solution for tracking cloth folding progression*: We provide the first results for the challenging case of quantifying cloth folding progression.
- *In-depth, case-based robustness analysis:* We demonstrate the robustness of our approach with adversarial cases to test the use-cases and limits of our proposed method.

The remainder of this paper is organized as follows. We first discuss related work in Section 2. Section 3 introduces our work by providing a high-level overview of the proposed method. Then, we explain our methodology in-depth in Section 4. We present the results on learning the task progression for folding clothing in Section 5. Finally, we discuss how these results are applicable for monitoring process quality and learning policies from crowd-sourced demonstrations in Section 6. We conclude our findings and future work in Section 7.

2 Related work

2.1 Data-driven process monitoring in smart manufacturing systems

Smart manufacturing systems allow collecting data in enterprises at high volumes and frequencies [8]. The availability of this data enables data-driven methods to train models for process monitoring and fault detection [1]. Machine learning methods, in particular, have been used to discover valuable patterns in manufacturing data by manually constructing features [9]. This way, virtual sensors can be trained to estimate product quality and process metrics based on historical measurements of easy-to-measure process variables. For example, in [10] the quality metric of a paper pulping process is inferred from chemical process features constructed from surrounding sensors. To avoid the need of manually engineering features, deep learning

methods are becoming increasingly popular for fault diagnosis [11]. In [12], they show that a deep neural network can outperform traditional process monitoring methods on three widely-used datasets. Other work looks at directly inputting process images to the neural network. For example, in [13] flame images of a furnace are used for monitoring the combustion process. In [14] infrared thermal videos are used as training data for a deep neural network to estimate the health conditions of rotating machinery. In [15] raw manufacturing data is converted to latent features learned by an autoencoder neural network. In practice, many of these applications assume the availability of process experts in order to carefully label the data [16]. However, this is costly and ambiguous to do for some domains. To avoid labelling data, existing work [17] uses semi-supervised learning to exploit both labelled and unlabelled data for predicting wafer quality during semiconductor manufacturing. Another work [18] avoids the need of labelling the remaining useful lifetime of industrial machines by compressing the input sensor data to a latent space using a recurrent neural network autoencoder. By reconstructing the latent space to a machine health index, they can match the resulting time series and use the reconstruction error to compute the health index used for estimating the system remaining useful lifetime. However, their method still requires finding example health index curves. We utilize a similar idea to leverage data under nominal operating conditions while borrowing insights from the learning from demonstration research in order to learn a task progression metric.

2.2 Learning from demonstrations

Learning from demonstrations is a prevalent domain in the robotics learning community. In the learning from demonstration survey of [5], a distinction is made between giving demonstrations and imitation learning depending on whether an embodiment mapping exists. In case the teacher executions are demonstrated, an embodiment mapping is implicit. In contrast, imitation implies that the correspondence problem needs to be solved. These definitions place our work as imitation learning from external observations: sensors external to the executing entity are used to train a learning agent that can have a different morphology. One instance of learning from demonstration is behavioural cloning, in which supervised learning is used to predict the actions an expert would do in a given state [19]. However, behavioural cloning methods are known to copy end-effector trajectories instead of understanding how actions relate to task performance. Moreover, errors accumulate when an agent takes a wrong action, which pushes him into an unseen part of the state space. A more general way to force the agent to attend to which actions increase task performance, is to learn the reward from demonstrations instead of the policy.

2.3 Inverse Reinforcement Learning

Reinforcement Learning (RL) is a domain that shares similar semantics with process monitoring: both require metrics indicating task progression and quality. In RL, the task progression metric is known as the reward function. This signal guides the learning agent towards task solutions. A sub-domain known as inverse RL [20] deals with learning reward functions from demonstration. In inverse RL, an outer loop learns the reward function while the inner loop executes a learning procedure for finding an optimal policy given the current reward function. Recent methods have looked at integrating deep neural networks as a representation layer in inverse RL [21, 22, 23]. However, much computational power is required for training due to the two loops taking place. Speeding up the training process with kinesthetic teach-in and updating instead of optimizing the reward function is explored in [21]. Unfortunately, manually moving the robot's end-effector proves to be unfeasible for tasks with difficult dynamics like knot tying or folding clothing. Other methods [22, 23] leverage expert demonstrations based on adversarial training. In these setups, the goal is to learn the task directly and not infer a reward function. In contrast, we aim to learn a reward function completely decoupled from policy optimization. This way, it can be used for multiple purposes, such as learning and process monitoring.

2.4 Self-supervised learning

An emerging method for sample-efficient learning of task-relevant features is self-supervised learning. Self-supervised learning exploits the structure present in a dataset to learn rich representations used for a downstream task such as image classification. Both natural language processing and computer vision has seen large leaps in self-supervised methods with, for example, BERT [24]. The general idea is to provide an artificial task to learn meaningful representations. Example tasks are learning to colourize images [25], reconstructing the original input [26] and predicting the relative position of two random patches [27]. An instance of self-supervised learning is contrastive learning, in which representations are learned by providing contrasting examples. In the case of video demonstrations, time can be used as a supervisory signal to provide contrasting examples. The goal then becomes to recover the temporal coherence of a video. One of the firsts works leveraging time as contrastive signal [28] inputs a sequence of frames and classifies whether the frames are in the correct order. Later work [29, 30] also frames self-supervised learning as a classification task in which the correct temporal order has to be determined. Several prior works construct reward functions, or equivalently process monitoring metrics, in latent spaces trained with time as a supervisory signal. In [31], they construct a reward function based on an image classifier trained on successful goal states reached

by teleoperating the robot towards the end state. In [32], time is used as a learned distance function for assigning environment rewards. However, their approach requires human intervention in order to select the desired goal states. [33] also uses time as a supervisory signal in videos of expert demonstrations to learn an optimal trajectory of states. However, they assume that visually removing the end-effector from the scene is not possible for all tasks. Other work [34] looks at expressing the reward function as the distance in latent space between the current state and the goal state. However, this is not possible when there is a trajectory in latent space that has to be followed in order to execute the task.

In this work, we leverage TCNs [7], which uses multi-perspective video demonstrations as input and time as a supervisory signal. It has already been shown that TCNs learn meaningful semantic embeddings, which can be used for robotic pose imitation of humans. This is done by aligning video frames using nearest neighbours in embedding space. This is problematic if a particular state machine or trajectory in embedding space has to be followed to solve the task. In [35], TCNs are trained over multiple input frames such that the network is able to encode the position and velocity of objects in the scene. Although TCNs are shown to be capable of robotic imitation of human poses, there is, to the best of our knowledge, no work that distils process monitoring metrics from self-supervised representations trained on video demonstrations.

2.5 Time series alignment

In order to match the latent space trajectory of an expert demonstration to a learning agent, the latent space progressions must be compared. Given the presence of a time dimension, a time series alignment problem arises. Time series alignment is studied extensively in natural language processing [36], bioinformatics [37] and human activity recognition [38]. Biological sequence alignment methods arrange the sequences of DNA to identify regions of similarity that may influence functional relationships. Path Similarity Analysis [37] for example, quantifies the similarity and difference between protein transition paths. In [39], a differentiable cycle consistency loss is used to align video frames based on the learned embedding space. They demonstrate impressive results on aligning video pairs of an action recognition dataset. However, it is unclear how their method behaves on long video demonstrations containing multiple, possibly suboptimal and heavily out-of-phase solutions to achieve the same task. Another broad class of algorithms for comparing a series of values with each other is Dynamic Time Warping (DTW). In DTW, the time series are assumed to be similar in amplitude but locally out of phase. DTW was introduced in [40] and had the goal to find an optimal alignment between sequences by warping the time axis iteratively. DTW has been used for a variety of domains such as

speech recognition applications [36], sign language recognition [41] and time series clustering [42]. Although many improvements on the original DTW algorithm exists [43], we experimentally show that the canonical DTW with minor adjustments can be used to align the latent space progression between expert and learner.

3 Overview of the proposed framework

We define the problem of using multi-perspective images with task demonstrations to construct a metric indicating task progression and solution quality. We want the task progression metric to increase on important moments when progression is made towards solving the task. Central in our framework is (1) generating a meaningful semantic embedding that indicates task progression and solution quality and (2) mapping this embedding to a scalar value. A high-level overview is given in Figure 1. Our method consists of three main steps. First, we use multi-perspective video frames to train an embedding using contrastive learning. This is done by using time as a self-supervisory signal. This method allows to learn useful invariances and forces the network to focus on task-relevant properties, as we will show in Section 5. We take a small sample of the demonstrations which we label *experts* as they will be used as a reference for indicating task progression. Second, we align the embeddings of the demonstrations to the task executions of experts using dynamic time warping. Third, we use this alignment to query the ensemble of experts for predicting task progress.

Our method assumes the availability of task demonstrations with corresponding process metrics. The demonstrations can range from teleoperated machinery to sensor recordings external to the executing body. We particularly focus on using recorded process metrics in the form of multi-perspective camera streams. Our method is deployable for arbitrary processes and tasks for which example demonstrations are available. These demonstrations are allowed to contain sub-par solution strategies. The method requires selecting a small part of the data, in our experiments 5%, as a reference for a good task solution. We focus on task demonstrations given by humans, but any entity solving the task can be used as input. Our methodology is applicable in settings where process monitoring is essential for output quality. We exploit temporal coherence, which requires the process to contain measurable inputs along a temporal dimension. For example, multiple cameras filming how human workers are sewing the front and back of a shirt together. Another major application we target is learning robotic manipulation skills using RL. This is because the field of RL heavily borrows the concept of expressing task progression and solution quality in a scalar value called the reward function. This reward function is used to learn an agent to solve a designated task in an unknown environment. Engineering such reward

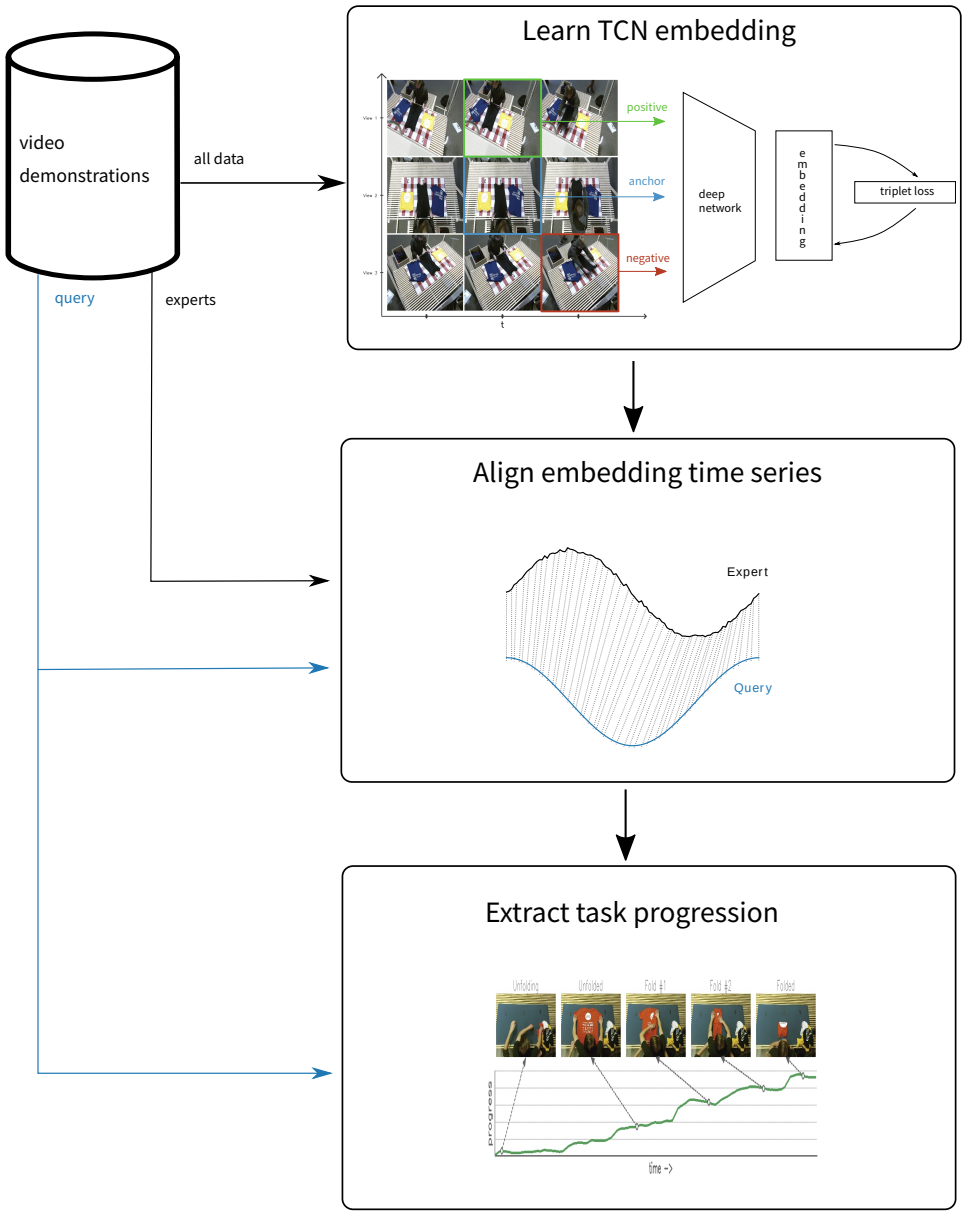


Figure 1: **High-level overview of our methodology.**

functions is difficult for some domains like folding clothing and autonomous driving. Our method allows learning a reward function without supervision, which can be used downstream for a learning agent requiring supervision in the form of a scalar value expressing task progression.

4 Generating task progression metrics from multi-perspective camera images

In Section 3, we gave a high-level overview of our framework to extract task progression metrics from crowd-sourced RGB images. Here, we discuss the framework in detail. We break down the three main steps into separate subsections.

4.1 Learning Semantic Meaningful Embeddings using TCNs

Central in the proposed framework is learning task-relevant representations containing the notion of task progression and solution quality. We use Time-Contrastive Networks [7] in which time serves as a supervisory signal. TCNs are a self-supervised method for training abstract representations of the progression of a task. The core concept is to push video frames distant in time away and pull them together when they are near in time. Multiple cameras are used to capture several perspectives of the same demonstration synchronised. This principle is shown in Figure 2. Any pair of frames from different camera angles that co-occurred must be close together in embedding space. Frames from the same camera angle separated by time are forced to be distant in embedding space. This principle encourages the network to attend to high-level features relevant to the task. Attending to irrelevant background noise or low-level features would attract negative pairs from the same perspective and repulse positive pairs from different perspectives. This way, the correspondence problem [44] for imitation learning can be solved. In case the network tries to explain the visual difference between two temporal distant frames by looking at the demonstrator, it would pull the anchor and negative close together, leading to a higher loss. The only way to achieve a lower loss is by looking at task-relevant features: what is consistently changing in the scene that cannot be attributed to changes in viewpoint, lighting, occlusion, and background.

Formally, if the embedding of an input is given by $f(x) \in \mathbb{R}^d$, we can then define the loss between an anchor x_i^a, positive x_i^p and negative frame x_i^n as [45]:

$$\left\| f\left(x_i^a\right) - f\left(x_i^p\right) \right\|_2^2 + \alpha < \left\| f\left(x_i^a\right) - f\left(x_i^n\right) \right\|_2^2, \\ \forall \left(f\left(x_i^a\right), f\left(x_i^p\right), f\left(x_i^n\right) \right) \in \mathcal{T},$$

with α being the margin enforced between positive and negative pairs. $\mathcal{T}$ represents all possible triplets, i.e. all *anchor-positive-negative* combinations. The loss we are trying to minimize then becomes:

$$\sum_{i}^{N} \left[\|f\left(x_i^a\right) - f\left(x_i^p\right)\|_2^2 - \|f\left(x_i^a\right) - f\left(x_i^n\right)\|_2^2 + \alpha \right]_+ .$$

To gather the triplets $\mathcal{T}$, we use a semi-hard triplet mining strategy with an increasing difficulty level. The goal of this strategy is to guide the training process to focus on increasingly harder *anchor-positive-negative* triplets. We do this by first sampling random anchors and positive frames from all possible perspectives. Positive frames are temporal neighbours at a maximum ϵ frames sampled around the anchor. Frames further away are labelled as negatives. This principle is visible in Figure 2. For each anchor during training, we select the most difficult positive, i.e., where the distance between anchor and positive is the largest. For this *anchor-positive* pair, we calculate the distance between the semi-hard negatives and anchor. Semi-hard negatives are defined as contrastive samples to the anchor which are of moderate difficulty: the distance between *anchor-negative* pair is marginally larger than the distance between the *anchor-positive* pair. Intuitively, this corresponds with pushing the fail-cases out of the minimal distance range one by one, starting with the easiest. We define the cost function of the batch as the average loss scores overall anchor frames. We provide the pseudocode for our training procedure in Algorithm 1.

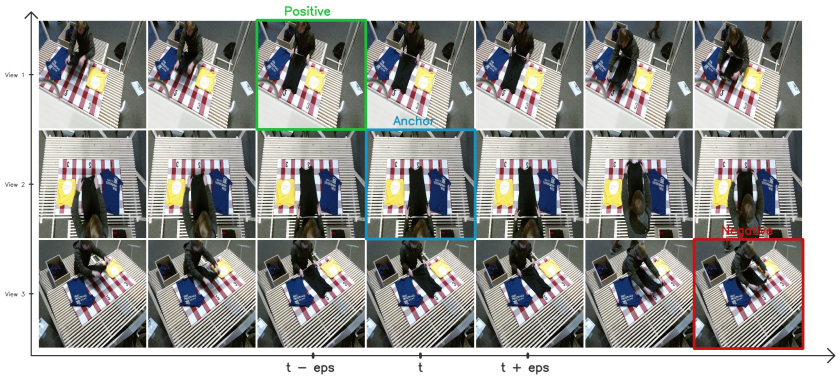


Figure 2: **Using time as a supervisory signal in TCNs.** A randomly selected anchor frame (in blue) and a nearby temporal neighbour from a different perspective (in green) are encouraged to be close in the embedding space compared to the anchor frame and a distant temporal neighbour (in red). This allows the network to learn to explain changes in the physical world.

Algorithm 1: Training loop of time-contrastive network

Input: training set of videos $\mathcal{V}$,
temporal distance ϵ,
time-contrastive neural network tcn parametrized by θ,
margin α,
mini-batch size k

foreach *epoch* **do**
  loss = 0;
  for 1 **to** k **do**
    select random video demonstration v from $\mathcal{V}$ with frames $\mathcal{F}$;
    select random anchor a from $\mathcal{F}$;
    generate positives $\mathcal{P} = \{p \in \mathcal{V} : \text{temporalDistance}\,(a, p) \leq \epsilon\}$;
    find hardest positive $p^* = \text{argmax}_{p \in \mathcal{P}}(\text{dist}\,(\text{tcn}\,(a), \text{tcn}\,(p)))$;
    generate negatives
     $\mathcal{N} = \{n \in \mathcal{V} : \text{temporalDistance}\,(a, n) > \epsilon\}$;
    generate semi-hard negatives $\mathcal{N}_{sh} =$
     $\{n \in \mathcal{N} : \text{dist}\,(\text{tcn}\,(a), \text{tcn}\,(p^*)) < \text{dist}\,(\text{tcn}\,(a), \text{tcn}\,(n)) < \text{dist}\,(\text{tcn}\,(a), \text{tcn}\,(p^*)) + \alpha\}$;
    find easiest semi-hard negative n^* with distance ϵ from a
     with $n^* = \min_{n \in \mathcal{N}_{sh}}(\text{dist}(\text{tcn}\,(a), \text{tcn}\,(n)))$;
    loss += $\text{dist}(\text{tcn}\,(a), \text{tcn}\,(p^*)) - \text{dist}(\text{tcn}\,(a), \text{tcn}\,(n^*)) + \alpha$;
  end for
  cost = loss/k;
  Perform a gradient descent step on cost with respect to the
   network parameters θ of the neural network tcn;
end foreach

4.2 Aligning Expert Video Embeddings with Query Videos

The TCN embedding trained in the previous Section 4.1 gives rise to a multivariate time series. In order to compare the time series embedding of a demonstrator $X = (x_1, \ldots, x_i, \ldots, x_N)$ to that of a chosen expert $Y = (y_1, \ldots, y_i \ldots, y_M)$ in order to judge the quality of the folding demonstration. To calculate the alignment between these time series, we use Dynamic Time Warping (DTW) [40]. DTW is an algorithm for measuring the similarity between time series under time distortions. It minimises the effects of shifting in time by allowing elastic transformations of the time series, subject to time-normalisation constraints. This allows accounting for nonlinear task execution rate differences between two demonstrations. In DTW, an optimal path P^* mapping time series X and Y are found:

$$P^* = \underset{\phi_x, \phi_y}{\operatorname{argmin}} \qquad \sum_{t=1}^{\mathrm{T}} d(x_{\phi_x(t)}, y_{\phi_y(t)}),$$

with $d()$ being a local distance function, for example, Euclidean distance. The alignment between the two time series is established through the mapping functions $(\phi_x(t), \phi_y(t))$. The warping functions are limited by certain constraints in order to be meaningful. Common warping constraints [46] are (1) start- and endpoint constraints as a clear start and end are manually specified by preprocessing the data, (2) monotonicity constraint which maintains temporal order during time normalization, and (3) local continuity constraints, also known as step patterns, to minimize loss of information. Compared to the canonical version of DTW, we relax the monotonicity constraint, which specifies that the alignment path does not go back in time. By allowing the time index of the expert time series Y (i.e., the reference signal) to go back in time, we can account for demonstrators restarting part of the task execution or even executing the task backward. This also allows for coping with a failed task execution. We also relax the formulation to open-end DTW [47] by removing the endpoint constraint to incorporate the possibility that demonstrators are not finishing the task optimally or completely. A visual interpretation on how we align a demonstration to video frames of an expert is given in Figure 3. We show the single component of two fictitious one-dimensional embeddings. The demonstration of which we need to calculate the task progression is labelled as the query. This is aligned to the embedding time series of the expert. The warping path is drawn in the global cost matrix. The coloured sections on the time series and alignment path represent subtasks. The background colours represent similar task progression or the same subtasks in the videos of the expert and query sample.

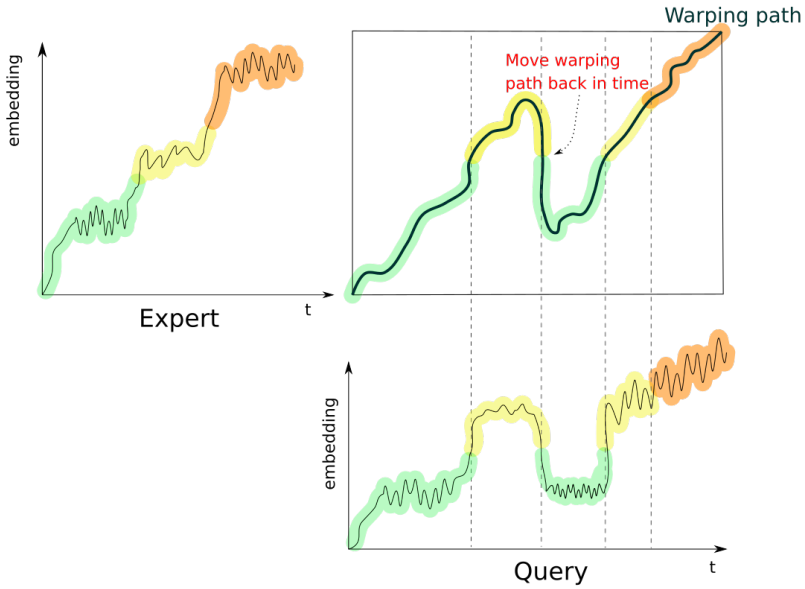


Figure 3: **Dynamic Time Warping (DTW) without monotonicity constraint.** We align a multivariate time series embedding of a demonstration (the query) to the video frames of an expert. The time index is allowed to go back in time in order to correctly align a demonstration in which executed actions undo the made progress.

4.3 Extracting Task Progression from Embeddings

The last step in our methodology is to extract task progression indicators using the aligned embeddings. First, we search which frame of each expert aligns best with each frame of the demonstration. The alignment score is expressed as the reciprocal of the cost of the best fit between them. Then, we select the temporal index of the best matching frame of every expert and express it as progress in percentage. Next, we average the experts' progress ratings by weighting with the normalised fit scores of the DTW phase of the pipeline. Finally, we remove any outliers by rejecting progress predictions of experts that deviate too much from the median of the absolute deviations [48].

5 Empirical results on folding clothing

We apply our methodology to the challenging case of assessing the task progression and quality of folding clothing. Manipulation of clothing and other deformable objects is considerably harder than rigid object manipulation due to the infinite number of configurations the object can be in. This makes the domain of deformable object manipulation relevant for our application: state estimation is challenging given the deformations exercised on the clothing [3]. In order to assess the quality of the executed manipulations, it is necessary to estimate how well the clothing is folded by identifying the final shape and detecting wrinkles. Manually performing this state estimation is time-consuming. Automated annotation processes rely on extracting contours of the textile using colour segmentation [49], utilizing depth cameras to detect wrinkles [50] and equipping the textile with tactile sensors [51]. However, these methods do not scale and generalize well as they require prior information about the state of the clothing and considerable engineering effort. In contrast, asking people to give folding demonstrations is cost-efficient to gather data.

In the remainder of this section, we present experimental results of applying our framework to extract task progression metrics based on video demonstrations of humans folding clothing. We first describe how we crowdsourced human folding demonstrations, followed by a quantitative analysis of the learned embeddings of these video demonstrations. We conclude by qualitatively examining the resulting task progression metrics functions on a test set.

5.1 Folding from demonstrations

To generate data, we crowdsourced video demonstrations of people folding clothing [52]. The setup is a dedicated folding table, visible in Figure 4. It contains three cameras mounted on top to capture multiple fixed perspec-

tives of the demonstrations. These are marked with yellow boxes on the Figure. The data is sliced into single clothing folding demonstrations: each demonstration starts with grasping an isolated piece of clothing on the left side of the table, folding in the middle, and finally stacking it on top of other clothing on the right side of the table. We only use folded t-shirts in our experiments, leading to 751 folding demonstrations. We manually select 5% of the data as experts, which will later represent the signal to align other video demonstrations using DTW. This selection was based on two simple criteria: (1) the resulting fold looks successful, and (2) the demonstrator executes substeps to solve the task in one go. The last criterion implies that to go from an initial state to a target state, the fold is executed in one flow and not paused in between or divided into multiple intermediate states. The data contains multiple sources of randomization as the recordings took place at two different locations, of which one is a public library. This makes it useful to learn multiple invariances.

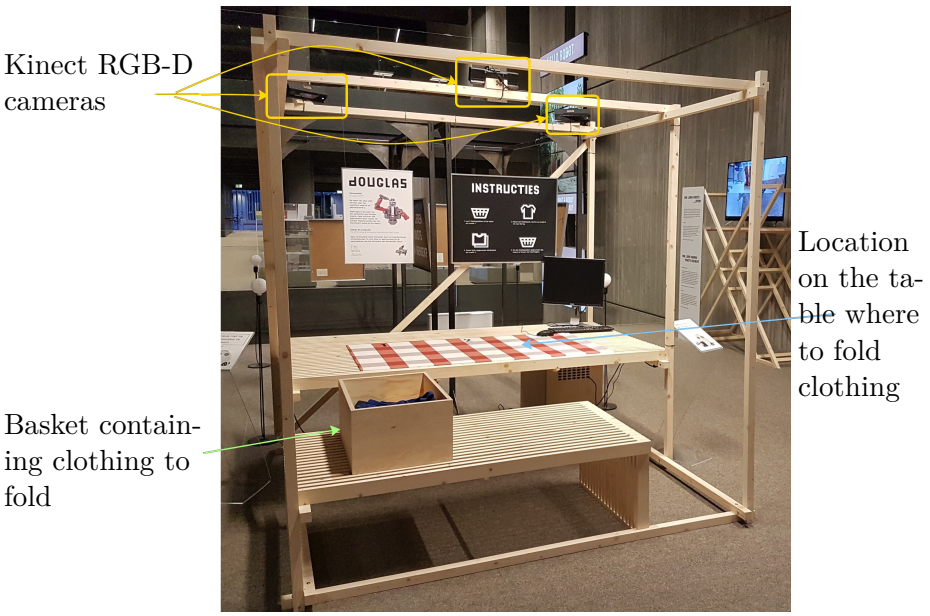


Figure 4: Picture of our folding table setup to crowdsource video demonstrations in a public library.

5.2 Training results

The first step in our methodology is to train TCNs using multi-perspective images of task demonstrations. The neural network architecture we used is given in Figure 5. We utilize the DenseNet [53] architecture pre-trained on ImageNet [54] as it already contains semantic relevant features for general-purpose vision-based tasks. Depending on the application, other neural

network architectures can be used as a backbone. We append four trainable convolutional layers to the output of the DenseNet architecture. Afterwards, the output is passed to a spatial softmax layer [55]. The "spatial soft arg-max" layer produces the expected image-space positions of the points of maximal activations of the features in the former convolutional layer. This allows decoding of the relative position of salient objects in the scene. The spatial softmax layer is succeeded by two fully-connected layers of 2048 and 32 neurons. The neurons in the last layer represent the compact low-dimensional representation of the task execution, which we will use downstream to generate a task progression metric. We train the network for 500 epochs using Adam optimization. Other fixed parameters during our experiments are given in Table 1.

To tune the hyperparameters of our method, we split the dataset into a training, validation and test set. We define the split at the level of the demonstration and not at the collective set frames. This means that a validation and test sample is a complete, novel example demonstration of folding a piece of cloth. We define a range for each identified hyperparameter in Table 2. Then, we evaluate all possible combinations of the hyperparameters using the semi-hard triplet loss, amount of semi-hard triplets and the fraction of successful hard negatives. We explain these quantitative metrics in the next paragraph. Finally, we select the set of hyperparameters that performs best on these training metrics on the validation set. The test set is used for the subsequent steps of translating the learned embedding to a scalar progression value. We provide the final hyperparameters used in our experiments in Table 2.

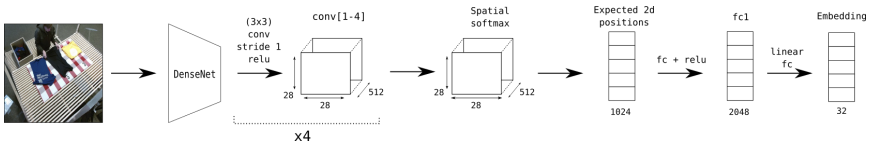


Figure 5: **Neural network architecture.** We use a pre-trained DenseNet backbone appended with four trainable convolutional layers. It is followed by a spatial softmax layer, which produces the expected 2D coordinates of the region of maximal activation in each channel of conv4. These coordinates are manipulated by a fully connected layer, which is finally passed through to the compressed embedding space.

In Figure 6, we show quantitative results of training the TCN. The loss quickly and steadily improves, as visible in Subfigure 6a. This indicates that transfer learning from the DenseNet backbone advances smoothly. However, the semi-hard triplet mining strategy presents more difficult training triplets over time to the network. This might result in oscillating loss functions making it necessary to monitor additional metrics. In particular, we monitor the amount of semi-hard triplets still available in the batches and the percent-

Table 1: Fixed settings during training of the TCN embedding.

Hyperparameter	Value
Optimizer	Adam
Weight initialization	Glorot initialization
# Epochs	500
Image size	224×224 pixels
GPU	1 NVIDIA Quadro P6000

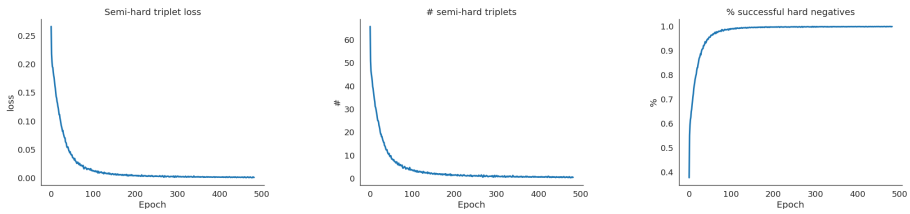
Table 2: Hyperparameters used for learning an embedding to extract task progression metrics for folding clothing items.

Hyperparameter	Best value	Value range
Learning rate	0.001	$[0.0001, 0.001, 0.01, 0.1]$
Batch size	32 samples	$[16, 32, 64]$
Embedding dimension	32 neurons	$[16, 32, 64]$
Max temporal distance between anchor and positive	3 s	0.5 s, 1 s, 3 s
Margin	0.2	$[0.1, 0.2, 0.4, 0.8]$
NN backbone	DenseNet	VGG, ResNet, DenseNet

age of successful hard negatives. Subfigure 6b shows that pushing a new semi-hard negative out of the margin does not lead to an increased amount of new semi-hard triplets re-entering the margin. Subfigure 6c examines which fraction of the *anchor-hardest positive* pairs are closer in embedding space compared to the *anchor-hardest negative* pairs. This metric steadily increases, indicating that meaningful clusters are formed in embedding space to temporally separate images from different viewpoints.

5.3 Task progression metrics

We align the resulting embeddings using DTW as described in Section 4.2. We use a symmetric stepping pattern [46] with a maximum step size up to 10 frames, corresponding with approximately 1 second. Figure 7 shows the task progression of a sample in the dataset. We annotated points in the plot with the corresponding image in the video demonstration. This shows that the task progression metric increases at meaningful moments in the demonstration. For example, when the demonstrator grasps the shirt to unfold on the table, the progression score increases from 0% to 40%. Afterwards, the progression scalar value stagnates because the demonstrator slowly moves the right sleeve to the middle. Once the sleeve is folded,



(a) (b) (c)

Figure 6: Loss and training metrics of the TCN training process with semi-hard triplet loss as optimization objective.

the progression indicator climbs. Next, the demonstrator grabs the left part of the shirt relatively quickly and moves it to the centre. This action is reflected in the progression value raising more quickly compared to the previous fold. Finally, the demonstrator receives maximum progression on completely folding the shirt. We find these results to be consistent among samples in the dataset. We provide videos of the task progression metric of other samples on `https://youtu.be/_HJhn8Hbv5s`. The whole pipeline requires 63 ms to process a single frame, with the slowest step being the alignment.

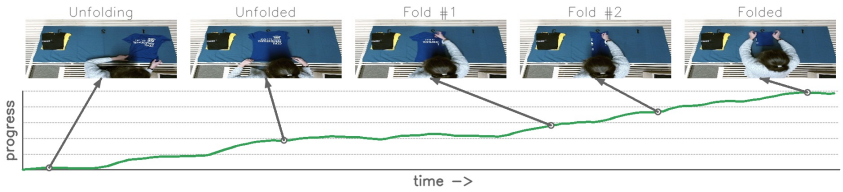


Figure 7: **Task progression plot with corresponding video frames of a single demonstration.** Each image is annotated with what is happening in the scene.

In conclusion, we find training of the embeddings in a self-supervised manner to be stable and efficient by using a semi-hard triplet function loss where we push out the easiest hard cases out of the margin first. Aligning the resulting embeddings with DTW on manually selected reference samples and using the alignment to express task progression increases progression scores on meaningful moments in the demonstration. This suggests that the embeddings encode task-relevant features. We analyze to which extent the embedding encodes important features for folding clothing in the next section.

6 Discussion

In the previous section, we have shown that TCN embeddings can be temporally aligned to extract useful task progression metrics. We now analyze the embeddings for post hoc interpretability. The goal is to discover which semantics are learned from the input images. However, neural networks lack decomposability into intuitive and understandable components. This is why we leverage the following two methods to understand what the network is encoding. First, we look at the semantic meaning of the embeddings at multiple temporal levels in Subsection 6.1. Second, we employ a case-based reasoning approach to interpret the learned representations and run robustness tests in Subsection 6.2.

6.1 Semantic Meaning of Learned TCN Embeddings

To discover the encoded semantics in the learned representations, we examine how the embedding progresses over time while linking it to what is happening in the scene. First, we examine how the embedding changes during the progress of a demonstration. To make the visualization interpretable, we project it to a lower dimension. Second, we examine whether we can extract meaningful information when searching for clusters in latent space.

To qualitatively analyze the results of training the embedding, we project the 32D embedding space to 2D using UMAP [56]. In Figure 8 we plotted the resulting projection mapped to the corresponding frame at different time shots. The time index is indicated with the colour of the scatter plot: from magenta to red, yellow, green, and blue.

A first, reoccurring observation is the projection of the embedding jumps at meaningful moments during task progression. In Figure 8, we notice that while grasping the shirt from the pile, the embedding stays in the first quadrant. Once the shirt is unfolded on the table, the embedding jumps to quadrant two. During the execution of the required folding steps, the embedding jumps to other locations. For example, the embedding gently transitions from quadrant two to three when folding the right and left sleeves. This suggests that the embedding can recognize different substeps in the folding task. When performing the final fold, the embedding is positioned between the first and fourth quadrant. We find this observation to be consistent among other samples. Consequently, the embedding at the start and the end of a trial is very similar. The explanation can be found in the observations starting and ending with a pile of unfolded shirts on the right side of the table and a pile of folded shirts on the left side. A similar start and end encoding in embedding space imply that the network potentially has problems distinguishing the start and end of the trial from a single image when there is no notion of memory.

A second observation is the embedding following a trajectory to go from grasping a piece of unfolded clothing to completely folding it on the table. Given that, in general, the third quadrant encodes folding the shirt's sleeves, it is possible that folding methods that do not fold the sleeves will not be assigned the correct progression score as there is no alignment available. In other words, for the DTW alignment to work, there needs to be a trajectory followed in embedding space in order to arrive at the solution. We explore this in Subsection 6.2.

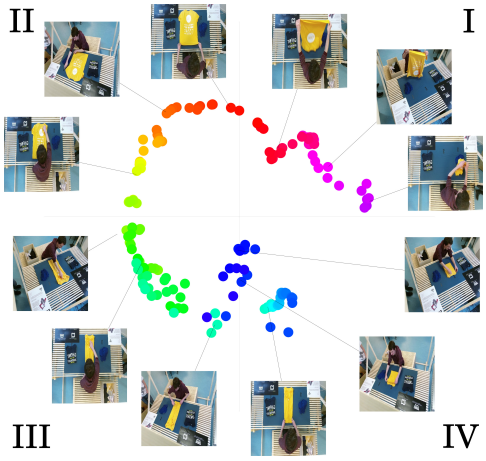


Figure 8: **2D projection of the learned embedding.** We annotate some points in the quadrants with the corresponding images to offer insight in to what is happening in the scene. Colors in the scatter plot indicate time progression from magenta to yellow, green and finally to blue.

To further analyze whether the embedding can distinguish meaningful moments during task execution, we generate clusters in the embedding hyperspace. We perform agglomerative clustering with ward linkage on each demonstration separately. Because our data is temporally sorted, we set the connectivity matrix as a square matrix with ones on the superdiagonal and subdiagonal, and zeros elsewhere. This way, we enforce the bottom-up cluster formation to consider temporal neighbours, reflecting the temporal ordering. We select five clusters to identify as they reflect the substeps in the task: grasping, flatten, fold one side, fold another side, fold in the middle. We visualize the results of the clustering in Figure 9. Here, we show the 2D projection of using UMAP on the embedding, with the colour representing the cluster membership instead of the temporal dimension. Qualitatively, we find the emergence of subtasks in the cluster membership when running agglomerative clustering in embedding space. The task starts in quadrant I in the projected embedding space. This is indicated with the red cluster membership. Once the demonstrator unfolds the shirt on the table, the cluster membership switches to green while the embedding transitions towards

quadrant II. The same process repeats when transitioning between the other subtasks. This indicates that the embedding encodes relevant aspects of the task, which downstream algorithms can use. However, we have no guarantees that the network picks up other signals from the environment during training. For example, the network might be encoding the position of the hands, or the size of the shirt. We examine which features the network is attending to in the following subsection.

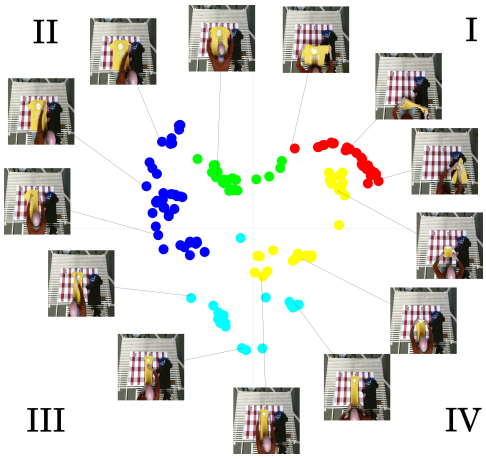


Figure 9: **Interpretation of embedding using agglomerative clustering.** Colors in the scatterplot indicate cluster membership. One point in the scatterplot represents an embedded video frame, projected onto a 2D plane using UMAP. The line links images to the corresponding embedding.

6.2 Case-based examples for post hoc interpretability

For the learned task progression metrics to be useful, they must be able to capture and generalize to specific situations that arise during the execution of the task, not seen in the training set. For example, a task executor can be performing random manipulations without actually performing meaningful contributions towards progressing the given task. This is also the case for learning purposes; many RL algorithms start with an exploration phase in which the robot acts randomly. To research the generalizability of our method, we employ a case-based approach in which we input certain scenarios and qualitatively analyze the result of our pipeline. We first describe the relevant scenario, followed by the resulting task progression scores, and offer insight into why a specific scenario succeeds or fails. Figure 10 and Figure 11 contains the visualizations of the cases we discuss below. We annotated the plotted task progression with specific frames from the demonstration to explain changes in the assigned progression. Videos of these hold-out samples are available at `https://youtu.be/ZvK0pQWH8ec`.

Change of environment with background distractions. Practical reasons and safety concerns make it possible that workers perform the designed process flow in another environment compared to the demonstrators. To test whether the learned progression metric can cope with this, we set up the folding table in a location not seen during training and fold clothing while putting random objects on the table. The images corresponding to the annotated parts of the plotted task progression in Figure 10a show that adding distractor objects such as a safety helmet, a flashlight, and a quadruped robot, do not prevent the learned task progression metric from assigning a correct progression score. There are two potential reasons for this. First, the data is recorded in a public environment where distractions are inherently present. Second, the TCNs are forced to attend high-level, task-relevant features. This filters out distractions in the image.

Meaningful manipulations compared to random behavior. In both process monitoring and scenarios where agents are learning to solve a task, non-meaningful manipulations in the process execution occur. Because fully random behaviour is not present in the data provided by the crowdsourced demonstrations, it is uncertain how the learned progression metric associate this with process quality. We explore how the progression metric evolves by first doing a meaningful manipulation of the shirt, followed by randomly moving the hands above the shirt. This experiment is displayed in Subfigure 10b. We find that the progression metric correctly assigns an intermediate score on unfolding the shirt. For the subsequent random movement of the arms of the demonstrator, no additional progress is made according to the metric. In similar experiments where demonstrators move their arms without any shirts on the table, we noticed that the network tries to distil meaningful manipulations. For example, a frequently reoccurring movement is performing the final fold where demonstrators grab the shirt at the bottom and fold it to the top. In some of those cases, we notice that the progression score increases due to the trajectory of the hands being recognized. However, the progression correctly drops again when the network detects that there is no shirt being folded. We explore this further in the next paragraph.

Attention to the essence of the task. In the previous paragraph, we examined the effect of random behaviour on the learned task progression metric. Here we compare non-meaningful behaviour to meaningful manipulations to test to which extent the neural network is paying attention to the essence of the task. We do this by setting up a scenario to test to which extent the neural network looks at the hand trajectories while monitoring the state of the textile. Concretely, we try to fool the network: we first solve a subtask and then execute the required arm trajectories to solve the task but without touching the clothing. We hypothesize that in case the

embedding is solely looking at the executed trajectories and not the clothing, the assigned task progression will increase. The result in Subfigure 10c. shows that the progression value increases when the shirt is unfolded. It is followed by stagnation of the progression score because the demonstrator is not manipulating the clothing. This demonstrates that our method looks at task-relevant features, like the state of the clothing, to indicate task progression. For RL purposes, this is relevant when compared to behavioural cloning; our method searches for the essence of the task instead of imitating end-effector trajectories.

Different task execution speed. The resulting progression metric should cope with different task execution speeds resulting from using different demonstrating entities. Given our crowdsourced dataset, this issue is already present in the training data. It is solved by aligning the resulting embedding time series with DTW. We verify that this is working as intended by performing the folds of a shirt at different speeds. We provide the results of this experiment in Subfigure 10d. We label the start and end of the folds in order to indicate the different speeds at which the task is solved. For example, the first fold is executed rapidly, leading to a quick increase in task progression values. Contrary, the second fold is executed by moving the right sleeve very slowly to the centre of the shirt. Once the sleeve is folded, a sudden increase in task progression is given.

Different task executor morphology. Industrial processes can be performed by any human actor or machine. In order for the learned progression metric to be useful, it has to generalize across actors with different morphologies. We test this by folding the shirt with two people, such that four arms are manipulating the clothing. The results, visible in Subfigure 10e, show that the resulting progression metric behaves correctly. This demonstrates that our trained embedding is invariant to the actor executing the task.

Variations in task execution. Task progression metrics should cope with different variations in executing the task. We test this in multiple ways: we fold the task by lying out the shirt diagonally, undo, and repeat some of the substeps and doing excessive wrinkle flattening. We find that the given progression value is correct and consistent in all these scenarios. In particular, we examine the case in which a shirt is folded and then unfolded again. The unfolding is executed by the demonstrator and not by playing the video in reverse. The resulting task progression, visible in Subfigure 10f, shows that the maximum progression is reached when the shirt is folded. When the demonstrator starts undoing the fold step by step, the task progression correctly drops. This hold-out sample demonstrates that our alignment step does not contain an upward drift. This is important as

the training data only exists of successful folds, leading to alignments that primarily run from start to end of the anchor and demonstrator. Hence the DTW step extracts useful information from the embedding to cope with failing task executions.

Generalization towards other folding methods. In extension to examining how well the learned task progression metric copes with variations on task execution, we look at how well they generalize when unseen folding methods are used. We test the case in which the demonstrator uses an alternative, unseen folding method where folds are executed on the table and in the air. In Subfigure 11a, we find that the assigned progression increases while folds are realized. We notice that the progression score suddenly drops on the step *unfolded square*. By looking at the embeddings, as demonstrated in Subsection 6.1, we find that is due to the embedding state transitioning to the *unfolded* step. Afterwards, the progression correctly increases to the maximum level while performing the last folding steps. We also test the case when folding only two steps instead of four. The results of this experiment are visible in Subfigure 11b. In this case, the process monitoring metric only detects the folded shirt very late in the folding process. The explanation can be found in the training data: the experts have never seen folding solutions with four steps. Given that the alignment process expects a certain trajectory to be followed in embedding space, the alignment fails.

Generalization towards other instances of the target object. We introduce shirts with other colours, textures, and reflective material to investigate how well our method generalizes to other shirt instances. We provide an example demonstration of a shirt with reflective material in Subfigure 11c. In this case, the learned process monitoring metric detects increasing progression on meaningful moments when solving the task. One instance in which the progression metric did not react appropriately is when folding tiny shirts, for example, in Subfigure 11d. When increasing the size of the shirt to normal proportions, the progression metric starts behaving appropriately. We hypothesize this is due to the embedding not reacting to very small hand movements and changes in the shirt, which lead to small pixel value changes in the image.

Task quality. We investigate to which degree the learned process monitoring metric incorporates the quality of the end result. We do this by examining how the task progression evolves when small and large disruptions are made in the resulting fold of the shirt. In Subfigure 11e, we disarrange the end fold of the shirt considerably. This leads to the demonstrator not achieving the maximum task progression at the end of the episode because the sleeves are partly hanging out of the shirt. However, in the experiment

visualized in Subfigure 11f, we apply a small perturbation of the folded shirt, which is not picked up by the process monitoring metric. We find that a rectangular folded shape with many wrinkles receives the same progression score as a perfectly flattened one. This can potentially be solved by using cameras closer to the shirt, using depth information to incorporate wrinkles in the embedding, or manually constructing triplets of end results with good and bad flattening.

In conclusion, we find that our method captures meaningful events in the task by looking at relevant features in the scene. By examining the assigned task progression values on the discussed hold-out samples, we show that the learned process monitoring metric is invariant to the demonstrator's morphology, background scene, execution speed and distractions. Our method is largely invariant to the shirt being manipulated, except that when the shirt gets too small, the resulting folds are not detected. Additionally, our method is not fooled by performing the expected arm trajectories without actually folding the shirt. We notice that the maximum progression value achieved during a successful folding demonstration consistently corresponds to the end-fold of the shirt. However, it is not possible to make meaningful comparisons in the end-quality of the fold of different demonstrations. We find meaningful reactions to highly visible disruptions in the shirt, but not to small wrinkles and small imperfections in the fold. In general, we find that the learned process monitoring metric effectively captures task progression and small degrees of output quality.

7 Conclusions

Learning the intention of a task from example demonstrations is an important step for process monitoring in manufacturing systems. In particular, evaluating the progress of folding clothing requires dealing with an infinite amount of states and occlusions caused by deformations. In this work, we proposed a method to encode task intent by assigning a progression value during task execution. We do this by learning semantically relevant features by using time as a self-supervisory signal on videos with task demonstrations captured from multiple perspectives. We align the resulting embedding to express task progression and task quality. We demonstrate the first results on expressing task progression for the challenging case of folding clothing. We find that the process monitoring metric assigns correct progression values on meaningful moments during task execution. With case-based examples, we show that our method learns task progression metrics that are invariant to noise, actor morphology and execution speed. An important characteristic is that our approach does not require labelling task progression of existing demonstrations manually. Therefore, our methodology circumvents the

Figure 10: Task progression plots and corresponding images of out-of-sample cases to specifically test properties of the learned process monitoring metrics.

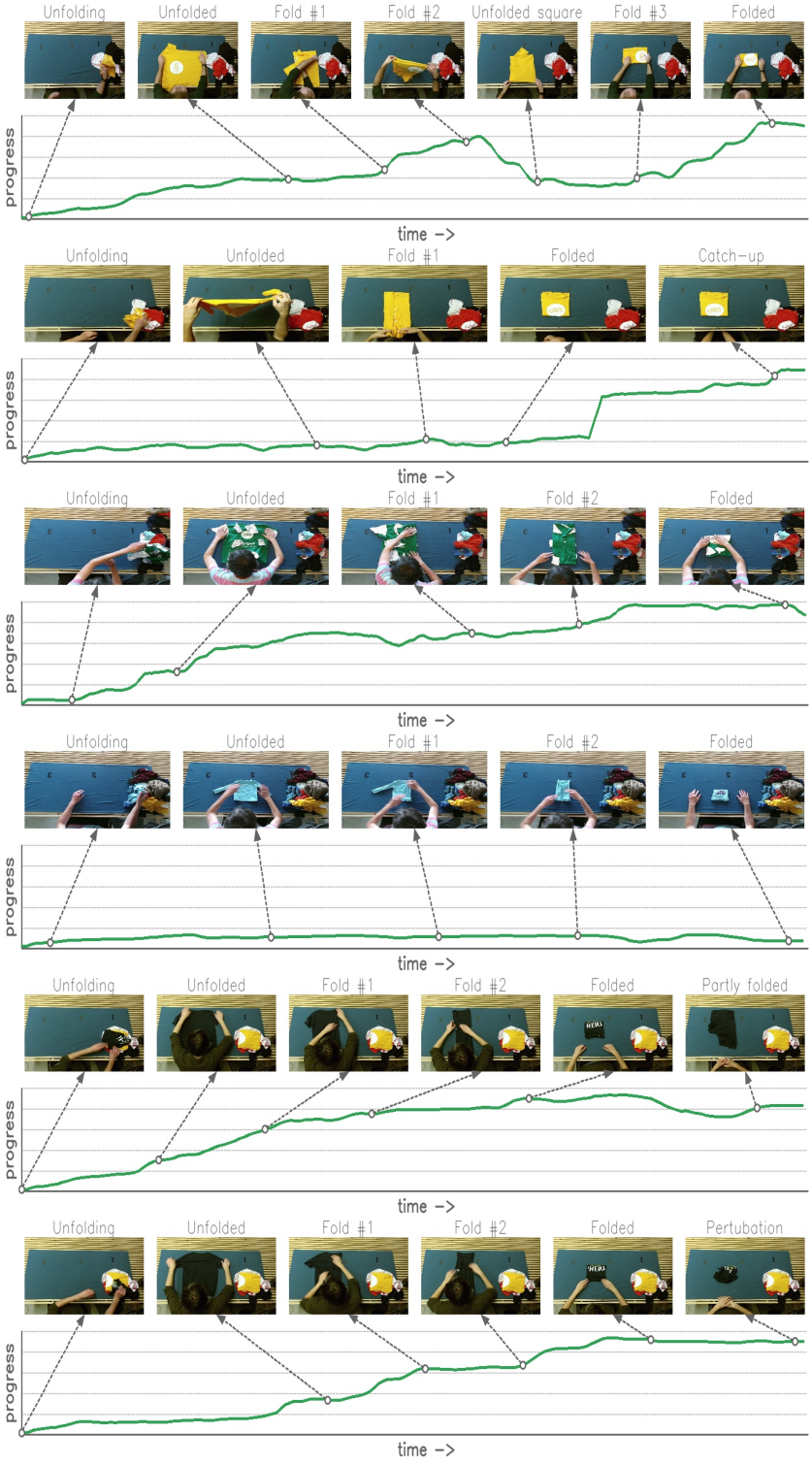


Figure 11: Task progression plots and corresponding images of out-of-sample cases to specifically test properties of the learned process monitoring metrics.

need to engineer task progression metrics by learning the task intent from existing task demonstrations. Additionally, our method can potentially be used in learning-based systems where the notion of progression needs to be incorporated in the learning signal, such as the reward function in reinforcement learning.

acknowledgements

This research is supported by the Research Foundation Flanders (FWO) under Grant number 1S54220N and Ghent University BOF under Grant number 01D21220. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Quadro P6000 GPU used for this research.

8 acknowledgements

References

[1] Shen Yin, Steven X. Ding, Xiaochen Xie, and Hao Luo. A review on basic data-driven approaches for industrial process monitoring. *IEEE Transactions on Industrial Electronics*, 61(11):6418–6428, 2014.

[2] Zhiqiang Ge, Zhihuan Song, and Furong Gao. Review of recent research on data-based process monitoring. *Industrial & Engineering Chemistry Research*, 52(10):3543–3562, 2013.

[3] Gian Luca Foresti and Felice Andrea Pellegrino. Automatic visual recognition of deformable objects for grasping and manipulation. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 34(3):325–333, Aug 2004.

[4] Vittorio Gallese, Christian Keysers, and Giacomo Rizzolatti. A unifying view of the basis of social cognition. *Trends in cognitive sciences*, 8(9):396–403, 2004.

[5] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[6] Chrystopher L Nehaniv, Kerstin Dautenhahn, et al. The correspondence problem. *Imitation in animals and artifacts*, 41, 2002.

[7] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1134–1141. IEEE, 2018.

[8] Xue Z Wang. *Data mining and knowledge discovery for process monitoring and control*. Springer Science & Business Media, 2012.

[9] Duc T Pham and Ashraf A Afify. Machine-learning techniques and their applications in manufacturing. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 219(5):395–412, 2005.

[10] Haisheng Li and Xuefeng Zhu. Application of support vector machine method in prediction of kappa number of kraft pulping process. In *Fifth World Congress on Intelligent Control and Automation (IEEE Cat. No.04EX788)*, volume 4, pages 3325–3330 Vol.4, 2004.

[11] Rui Zhao, Ruqiang Yan, Zhenghua Chen, Kezhi Mao, Peng Wang, and Robert X. Gao. Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, 115:213–237, 2019.

[12] Long Wen, Xinyu Li, Liang Gao, and Yuyan Zhang. A new convolutional neural network-based data-driven fault diagnosis method. *IEEE Transactions on Industrial Electronics*, 65(7):5990–5998, 2018.

[13] Yuting Lyu, Junghui Chen, and Zhihuan Song. Image-based process monitoring using deep learning framework. *Chemometrics and Intelligent Laboratory Systems*, 189:8–17, 2019.

[14] Olivier Janssens, Rik Van de Walle, Mia Loccufier, and Sofie Van Hoecke. Deep learning for infrared thermal image based machine health monitoring. *IEEE/ASME Transactions on Mechatronics*, 23(1):151–159, 2018.

[15] Yaguo Lei, Feng Jia, Jing Lin, Saibo Xing, and Steven X. Ding. An intelligent fault diagnosis method using unsupervised feature learning towards mechanical big data. *IEEE Transactions on Industrial Electronics*, 63(5):3137–3147, 2016.

[16] Thorsten Wuest, Daniel Weimer, Christopher Irgens, and Klaus-Dieter Thoben. Machine learning in manufacturing: advantages, challenges, and applications. *Production & Manufacturing Research*, 4(1):23–45, 2016.

[17] Pilsung Kang, Dongil Kim, and Sungzoon Cho. Semi-supervised support vector regression based on self-training with label uncertainty: An application to virtual metrology in semiconductor manufacturing. *Expert Systems with Applications*, 51:85–106, 2016.

[18] Pankaj Malhotra, Vishnu Tv, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Multi-sensor prognostics using an unsupervised health index based on lstm encoder-decoder. *1st SIGKDD Workshop on Machine Learning for Prognostics and Health Management*, 08 2016.

[19] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[20] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[21] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, pages 49–58, 2016.

[22] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 4565–4573, 2016.

[23] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.

[24] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[25] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.

[26] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[27] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.

[28] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. Shuffle and Learn: Unsupervised Learning using Temporal Order Verification. In *ECCV*, 2016.

[29] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Kumar Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequence. In *IEEE International Conference on Computer Vision*, 2017.

[30] Basura Fernando, Hakan Bilen, Efstratios Gavves, and Stephen Gould. Self-supervised video representation learning with odd-one-out networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3636–3645, 2017.

[31] Avi Singh, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine. End-to-end robotic reinforcement learning without reward engineering. *Robotics: Science and Systems*, 2019.

[32] Kristian Hartikainen, Xinyang Geng, Tuomas Haarnoja, and Sergey Levine. Dynamical distance learning for semi-supervised and unsupervised skill discovery. In *International Conference on Learning Representations*, 2019.

[33] Suraj Nair, Mohammad Babaeizadeh, Chelsea Finn, Sergey Levine, and Vikash Kumar. Trass: Time reversal as self-supervision. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 115–121, 2020.

[34] Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pages 9191–9200, 2018.

[35] Debidatta Dwibedi, Jonathan Tompson, Corey Lynch, and Pierre Sermanet. Learning actionable representations from visual observations. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1577–1584. IEEE, 2018.

[36] Cory Myers, Lawrence Rabiner, and Aaron Rosenberg. Performance tradeoffs in dynamic time warping algorithms for isolated word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(6):623–635, 1980.

[37] Sean L Seyler, Avishek Kumar, Michael F Thorpe, and Oliver Beckstein. Path similarity analysis: a method for quantifying macromolecular pathways. *PLoS Comput Biol*, 11(10):e1004568, 2015.

[38] Inês P. Machado, A. Luísa Gomes, Hugo Gamboa, Vítor Paixão, and Rui M. Costa. Human activity data discovery from triaxial accelerometer sensor: Non-supervised learning sensitivity to feature extraction parametrization. *Information Processing & Management*, 51(2):204 – 214, 2015.

[39] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Temporal cycle-consistency learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[40] Richard Bellman and Robert Kalaba. On adaptive control processes. *IRE Transactions on Automatic Control*, 4(2):1–9, 1959.

[41] Ana Kuzmanic and Vlasta Zanchi. Hand shape classification using dtw and lcss as similarity measures for vision-based gesture recognition system. In *EUROCON 2007 - The International Conference on "Computer as a Tool"*, pages 264–269, 2007.

[42] Vit Niennattrakul and Chotirat Ann Ratanamahatana. On clustering multimedia time series data using k-means and dynamic time warping. In *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*, pages 733–738, 2007.

[43] Duarte Folgado, Marília Barandas, Ricardo Matias, Rodrigo Martins, Miguel Carvalho, and Hugo Gamboa. Time alignment measurement for time series. *Pattern Recognition*, 81:268 – 279, 2018.

[44] Marcel Brass and Cecilia Heyes. Imitation: Is cognitive neuroscience solving the correspondence problem? *Trends in cognitive sciences*, 9:489–95, 11 2005.

[45] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[46] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Prentice-Hall, Inc., USA, 1993.

[47] Paolo Tormene, Toni Giorgino, Silvana Quaglini, and Mario Stefanelli. Matching incomplete time series with dynamic time warping: an algorithm and an application to post-stroke rehabilitation. *Artificial Intelligence in Medicine*, 45(1):11 – 34, 2009.

[48] Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. Detecting outliers: Do not use standard deviation

around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764 – 766, 2013.

[49] Stephen Miller, Jur van den Berg, Mario Fritz, Trevor Darrell, Ken Goldberg, and Pieter Abbeel. A geometric approach to robotic laundry folding. *The International Journal of Robotics Research*, 31(2):249–267, 2012.

[50] Andreas Doumanoglou, Jan Stria, Georgia Peleka, Ioannis Mariolis, Vladimir Petrik, Andreas Kargakos, Libor Wagner, Václav Hlaváč, Tae-Kyun Kim, and Sotiris Malassiotis. Folding clothes autonomously: A complete pipeline. *IEEE Transactions on Robotics*, 32(6):1461–1478, Dec 2016.

[51] Andreas Verleysen, Thomas Holvoet, Remko Proesmans, Cedric Den Haese, and Francis wyffels. Simpler learning of robotic manipulation of clothing by utilizing diy smart textile technology. *APPLIED SCIENCES-BASEL*, 10(12):10, 2020.

[52] Andreas Verleysen, Matthijs Biondina, and Francis wyffels. Video dataset of human demonstrations of folding clothing for robotic folding. *The International Journal of Robotics Research*, 2020.

[53] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017.

[54] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[55] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[56] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018.