DANNTe: a case study of a turbo-machinery sensor virtualization under domain shift

Abstract

We propose an adversarial learning method to tackle a Domain Adaptation time series regression task (DANNTe). The task concerns the virtualization of a physical sensor of a turbine with aim to build a reliable virtual sensor working on operating conditions not considered during the training phase. Our approach is directly inspired by the need to have a domain-invariant representation of the features to correct the covariate shift present in the data. The learner has access to both a labeled source data and unlabeled target data (Unsupervised DA) and is trained on both, exploiting the minmax game between a task regressor neural network and a domain classifier neural network. Both models share the same feature representation in terms of a feature extractor neural network. This work is based on the work of Ganin et al. [6]; we present an extension suitable to be applied to time series data. The results report a significant improvement in regression performance, compared to the base model trained on the source domain only.

1 Introduction

Unsupervised Domain Adaptation [13], with generalization bounds stated by Ganin et al. [3], [4] is a type of transfer learning [11] where the task remains the same while the domains are different (transductive transfer learning). Formally, the learner has access to a labeled source dataset $S = \{(x_i^s, y_i^s)\}_{i=1}^{n_s}$ and an unlabeled target dataset $T = \{(x_i^t)\}_{i=1}^{n_t}$, where datapoints x^s and x^t are sample respectively from a source distribution P_s and a target distribution P_t both over X.

We seek to build a domain-invariant feature representation, where the emergent features are invariant with respect to the domain. We expect that a model based on domain-invariant features will perform with good performance in both domains, so that the difference between them is not very significant. In particular, we apply the unsupervised domain adaptation method to an industrial turbo-machinery context providing practical results and showing that domain adaptation can be an answer also applied to complex timeseries application, even in presence of a non-independently and identically distributed assumption.

1.1 Related works

Several line of research address the unsupervised domain adaptation task. The line of *Domain Invariant feature* aims to learn a domain-invariant feature representation, typically in the form of a feature extractor neural network [7], [15], [12], [1]. The line of *Domain Mapping* aims to learn a mapping from one domain to another. The map is typically created at the pixel level [2], or trough a specific GAN [10] , where a generator performs adaptation. The line of *Normalization statistics* exploits the batch normalization layer [8] to learn domain knowledge [14]. The line of *Ensemble methods* consists of using multiple models [5] averaging their output to keep domains separated.

Submitted to NeurIPS 2021 AI for DistShift Workshop.

1.2 Use case

In turbo-machinery applications it is common to observe a domain shift. Domain shift can be generated from operative conditions not observed during test phase or from the environmental characteristics of the customer site, often different from those where the prototype is validated (test rig). In our specific case, a model of a physical sensor is learnt from prototype data and needs to be applied to fleet data, where that physical sensor is not present. In other words, our challenge is to combine the lack of the sensor in target data and the domain shift due to different features distribution. With this statement we are providing the needs to a unsupervised DA approach to deal with the observed domain shift.

1.3 DataSet

To validate our implementation (called DANNTe) we tested on prototype data, splitting source/target between winter and summer period respectively. In particular, the source domain is represented by timeseries collected during a winter period and the target domain acquired during a summer period. Dataset has been acquired from 30 sensors installed on a turbine running for test from December 2019 to August 2020. Winter data from December to February is used as labeled source dataset, and the summer data from June to July as unlabeled target dataset. Both the datasets are collected from the same machine in a prototype state where the conditional operation might be different and the environmental conditions generate two different distributions.

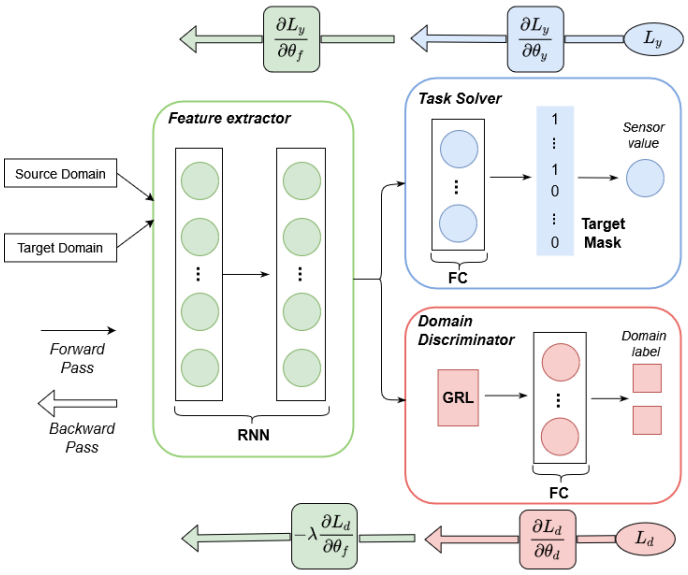


Figure 1: Architecture of our proposed approach (DANNTe), based on Ganin et al. [6]. Feature extractor weights are modified by both the task solver (in our case, a regressor trying to minimize the reconstruction loss) and the domain classifier (trying to minimize the source vs target domain classification loss). The gradient reversal layer acts so that a minimization problem is solved (instead of a min-max one), just reversing the sign of the domain classifier gradient during backpropagation.

.

2 The implemented library

2.1 Model

We adapted the Domain-Adversarial Neural Networks (DANN) by Ganin et al. [6] to a regression task (**D**omain-**A**dversarial **N**eural **N**etworks applied to **T**imeseri**e**s, DANNTe). The architecture is composed by a feature extractor recurrent network stacked on two networks (see Fig. 1). The first head is a task solver (previously "label predictor") neural network, a regressor whose goal is to minimize the reconstruction loss for the source domain, where y is available. Its loss is not affected

by target domain examples, which are skipped because of the target mask layer. The second head is a domain classifier neural network, aiming at discriminate examples coming from source from those coming from target domain, exploiting the x information, the only available in both domains. To correctly discriminate, a new dataset $U = \{(x_i, 0)\}_{i=1}^{n_s} \cup \{(x_i, 1)\}_{i=1}^{n_t}$ is built, where samples from source domain are labelled with 0 and samples from target domain are labelled with 1.

The task predictor loss L_y is the MSE, while the domain classifier loss L_d is the Negative Log Loss defined as:

$$L_d = -\frac{1}{n}\sum_{i=1}^{n}(y_i log(p(y_i)) + (1 - y_i) log(1 - p(y_i))) \tag{1}$$

where y_i denotes the binary variable (domain label) for the i-th sample, which indicates whether it comes from the source distribution (y_i = 0) or from the target distribution (y_i = 1).

The total loss combines the contribution of the losses of the two branches, defined as:

$$L_{tot} = L_y - \lambda L_d \tag{2}$$

where λ is the domain loss multiplier; λ influences the contribution of the domain classifier loss during backpropagation. Once trained, only the feature extraction and task solver parts are kept and used for inference.

2.2 Model adaptation

Ganin et al. ([6]) propose to use the data in different formats according to the branch: the *task solver* expects the training data $\{(x_i^s, y_i^s)\}_{i=1}^{n_s}$, and the *domain classifier* expects the training data $\{(x_i^s, 0)\}_{i=1}^{n_s} \cup \{(x_i^t, 1)\}_{i=1}^{n_t}$. Using two different datasets to train the model causes the need to perform multiple forward and backward passes, making training computationally demanding. To reduce the computational complexity, we propose a solution based on the addition of a target mask layer. The **target mask** layer modifies the loss L_y contribution of the *task predictor* for the samples belonging to the target domain, by assigning them a loss weight of 0. This approach is equivalent to computing the loss L_y first using only transformed samples from the source domain, and then computing the loss L_d using the combined batches.

Another proposal by Ganin et al. ([6]) is to build the datasets by i.i.d from P_s and P_t. In our specific use case, since we use an RNN (LSTM) as a *feature extractor*, generating the datasets by uniformly and identically distributed sampling would lose the temporal dependencies within the data and we cannot apply. The solution we propose is to train the model by creating **equally divided batches** where half of each batch is filled with samples from the source domain, and half with samples from the target domain, keeping the temporal order. The reason why we select this approach is to hold the sequential behaviour of measurements in successive time slots, granted by having portion of batch with consecutive measurements.

3 Results

3.1 Performance assessment strategy

Model performance has been evaluated by exploiting the variable y which in this simplified use case is available in both domains. We remind once again that this ground truth will not be available in our real use case, instead.

The DANNTe regression performance has been compared to:

- an upper-bound performance, given by the case when ground truth is available also in target set, so that both source and target sets are used for a fully supervised training (*fully-supervised model*);
- a lower-bound performance, given by the case when we simply perform supervised training on source target data and directly apply the model to target domain data (*baseline model*).

3.2 Model performance and comparison

Performance comparison summarized in Tab. 1 shows the improvement achieved using DANNTe, with respect to the baseline model. Related uncertainties are referring to the variance achieved trough cross validation phase.

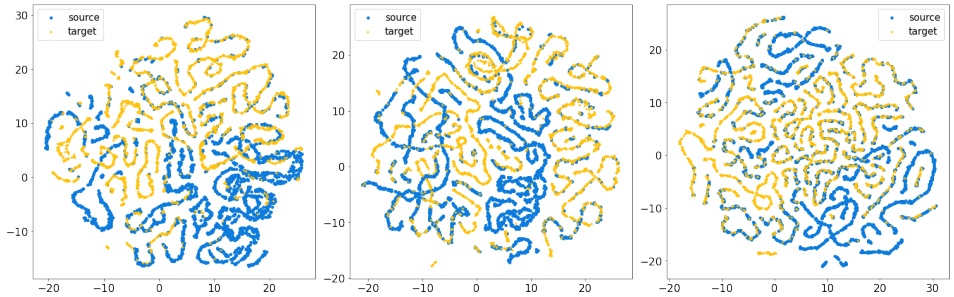
	MSE Source	MSE Target
Baseline model	**8.4** $\pm$ 0.1	3567.6 $\pm$ 0.9
DANNTe model	120.6 $\pm$ 0.2	**1398.2** $\pm$ 0.2
Fully-supervised model	16.4 $\pm$ 0.7	17.8 $\pm$ 0.8

Table 1: DANNTe performance compared to baseline and fully-supervised models. Metric MSE refers to the Mean Squared Error.

We found that the hyperparameter λ, which is the constant multiplier of the domain classifier's loss during backpropagation, plays a key role in feature extraction. The higher its value, the stronger is the push towards domain invariance in the feature extractor. For our experiments, we found that a value equal to 1, yielded the best performance.

3.3 Features encoding

To get a graphic insight about how the feature embedding changes with DANNTe, we use the UMAP method [9] to lower the embedding dimension to 2 (see Figure 2). From the comparison of Figure 2a corresponding to the baseline model and Figure 2b corresponding to the fully supervised model, we observe a clear clustering between source and target datasets. However, when we use the proposed approach, the representation in Figure 2c shows a lower ability to distinguish between clusters.



(a) Baseline model (b) Fully supervised model (c) Proposed approach

Figure 2: Data representation reduced to 2 dimensions using UMAP. We can observe a clear separation between the two domains (blue and yellow, summer and winter) using by the baseline model (a). Our DANN adaptation to timeseries (DANNTe) seems to construct a less discriminative feature representation.

4 Conclusions and Future work

The results are promising and allowed us to improve the reliability of a model in two different domains, correcting the domain shift present in the data, showing how improvement is guaranteed if virtualization is based on features that combine discriminativeness and domain invariance. We adapted DANN method to a regression task applied to a industrial use case. Results report that the DANNTe approach improves performance. Despite there is still room for improvement in order to achieve results as close as possible to the fully-supervised approach, our findings show that is a promising approach also in real applications.

References

[1] David Acuna et al. “f-Domain-Adversarial Learning: Theory and Algorithms”. In: (2021).

[2] Shrivastava Ashish et al. “Learning from simulated and unsupervised images through adversarial training”. In: (2017).

[3] Shai Ben-David et al. “Analysis of Representations for Domain Adaptation”. In: *Advances in Neural Information Processing Systems 19 (NIPS 2006)* (2007), pp. 137–144.

[4] Shai Ben-David et al. “Impossibility Theorems for Domain Adaptation”. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, PMLR 9* (2010), pp. 129–136.

[5] Mostafa El Habib Daho et al. “Weighted vote for trees aggregation in Random Forest”. In: *IEEE:10.1109* (2014).

[6] Yaroslav Ganin et al. “Domain-Adversarial Training of Neural Networks”. In: *arXiv:1505.07818* (2016).

[7] Zhao Han et al. “On Learning Invariant Representations for Domain Adaptation”. In: (2019), pp. 7523–7532.

[8] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: (2015).

[9] McInnes Leland, Healy John, and Melville James. “UMAP: Uniform ManifoldApproximation and Projection forDimension Reduction”. In: *arXiv:1802.03426* (2018).

[10] Julian Alberto Palladino, Diego Fernandez Slezak, and Enzo Ferrante. “Unsupervised Domain Adaptation via CycleGAN for White Matter Hyperintensity Segmentation in Multicenter MR Images”. In: (2020).

[11] S. J. Pan and Q. Yang. “A Survey on Transfer Learning”. In: (2010), pp. 1345–1359.

[12] Eric Tzeng et al. “Adversarial Discriminative Domain Adaptation”. In: (2017).

[13] Garret Wilson and Diane J. Cook. “A Survey of Unsupervised Deep Domain Adaptation”. In: (2020).

[14] Li Yanghao et al. “Adaptive Batch Normalization for practical domain adaptation”. In: (2018), pp. 109–117.

[15] Chaohui Yu et al. “Transfer Learning with Dynamic Adversarial Adaptation Network”. In: (2019).