Benchmarking Domain Adaptation for Chemical Processes on the Tennessee Eastman Process

Eduardo Fernandes Montesuma[1], Michela Mulas[2], Fred Ngolè Mboula[1],
Francesco Corona[3], and Antoine Souloumiac[1]

[1] Université Paris-Saclay, CEA, LIST, F-91120, Palaiseau, France
[2] Department of Teleinformatics Engineering, Federal University of Ceará, Brazil
[3] School of Chemical Engineering, Aalto University, Finland

Abstract. In system monitoring, automatic fault diagnosis seeks to infer the systems' state based on sensor readings, e.g., through machine learning models. In this context, it is of key importance that, based on historical data, these systems are able to generalize to incoming data. In parallel, many factors may induce changes in the data probability distribution, hindering the possibility of such models to generalize. In this sense, domain adaptation is an important framework for adapting models to different probability distributions. In this paper, we propose a new benchmark, based on the Tennessee Eastman Process of Downs and Vogel (1993), for benchmarking domain adaptation methods in the context of chemical processes. Besides describing the process, and its relevance for domain adaptation, we describe a series of data processing steps for reproducing our benchmark. We then test 11 domain adaptation strategies on this novel benchmark, showing that optimal transport-based techniques outperform other strategies[4].

Keywords: Transfer Learning · Domain Adaptation · Optimal Transport · Tennessee Eastman Process.

1 Introduction

Within process supervision, faults are unpermitted deviations of a characteristic property or variables of a system [16]. Furthermore, there is an increasing demand on reliability and safety of technical plants, motivating the necessity of methods for supervision and monitoring. These are Fault Detection and Diagnosis (FDD) methods, which comprise the *detection*, i.e., if and when a fault has occurred, and the *diagnosis*, i.e., the determination of *which fault* has occurred. In this paper, we focus on Automatic Fault Diagnosis (AFD) systems, assuming that faults were previously detected accordingly.

In parallel, Machine Learning (ML) is a field of artificial intelligence, that defines predictive models based on data. Nonetheless, these models make an implicit assumption, that training and test data come from the same probability

---

[4] Our code is open sourced at `https://github.com/eddardd/tep-domain-adaptation`

distribution, which is seldom verified in practice [26], as both training and test data may be collected under heterogeneous conditions that drive shifts in probability distributions. This phenomenon motivates the field of Transfer Learning (TL) [23] to propose algorithms that are robust to *distributional shift*.

There is a straightforward link between ML and AFD systems, as one can understand fault diagnosis as a classification problem. In this sense, one uses sensor data (e.g., temperature, concentration, flow-rate) as inputs to a classifier, which predicts the corresponding fault, or its absence [34]. Further, TL is a broad field within ML, in which knowledge must be *transferred* from a source to a target context. Within TL, Domain Adaptation (DA) is a common framework where one has access to labeled data from a source domain, and unlabeled data from a target domain. Thus, DA seeks improving classification accuracy on target domain data. In many cases, source data is itself heterogeneous, following multiple probability distributions. This setting is known as Multi-Source DA (MSDA).

In this paper, we propose a new benchmark, based on the Tennessee Eastman Process (TEP) [7, 28], a complex, large-scale chemical process used by the chemical engineering community for benchmarking control systems, as well as FDD techniques. This process is interesting for DA, as it may operate at different modes of production. As we show in our case study (section 3), the different modes of production induce different data probability distributions, thus the need for DA techniques for improving generalization. We further benchmark existing techniques in DA, either based on pre-extracted features (shallow DA), or through deep learning (deep DA).

The rest of this paper is divided as follows. Section 2 covers the theoretical foundations of our work. Section 3 presents a case study of the TEP. In this section, we present the system, analyze the properties of the different modes of production, and benchmark different strategies in DA. Finally, section 4 concludes this paper.

2   Classification and Domain Adaptation

In supervised learning, one is provided with a dataset $\{\mathbf{x}_i^{(P)}, y_i^{(P)}\}_{i=1}^n$, where $\mathbf{x}_i^{(P)} \overset{iid}{\sim} P$, and $y_i^{(P)} = h_0(\mathbf{x}_i^{(P)})$, for a distribution P and a ground-truth labeling function $h_0 : \mathcal{X} \to \mathcal{Y}$. $\mathcal{X}$ is called *feature space*, such as $\mathbb{R}^d$, and $\mathcal{Y}$ *label space*, in this case $\{1, \cdots, n_c\}$. The goal of classification is finding, among a family of functions $\mathcal{H}$, $\hat{h}$ such that,

$$\hat{h} = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^n \mathcal{L}(h(\mathbf{x}_i^{(P)}), y_i^{(P)}), \tag{1}$$

where $\mathcal{L}$ is a loss function, such as the Cross-entropy (CCE), $\mathrm{CCE}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{c=1}^{n_c} y_c \log \hat{y}_c$. This approach, known as empirical risk minimization, has the desirable property that $\hat{h}$ correctly predicts on unseen samples from P. This property is known as *generalization*. We refer readers to [27] for a review on the theory of generalization.

In this paper, we consider deep neural nets composed of 2 parts: an encoder network ϕ, and a classifier h. The encoder maps data $\mathbf{x} \in \mathcal{X}$ into a latent representation $\mathbf{z} \in \mathcal{Z}$, whereas the classifier maps the representation into a label space $\mathcal{Y}$. Hence, $\hat{y}_i^{(P)} = h(\phi(\mathbf{x}_i^{(P)}))$. As such, eq. 1 is minimized with respect the parameters of the encoder, θ_ϕ and classifier θ_h.

$$\mathbf{x}_i^{(P)} \; \rightarrow \; \underset{\phi}{\phi} \; \rightarrow \; \mathbf{z}_i^{(P)} \; \rightarrow \; \underset{h}{h} \; \rightarrow \; \hat{y}_i^{(P)} \rightarrow \boxed{\mathcal{L}}$$

<div align="center">

ϕ h

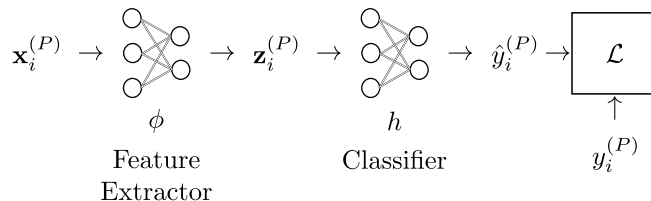Feature   Classifier $y_i^{(P)}$

Extractor

</div>

Fig. 1: **Illustration of a deep neural net,** where data $\mathbf{x}_i^{(P)}$ are mapped into latent representation vectors $\mathbf{z}_i^{(P)}$ through an encoder ϕ. The latent representation is then used to predict a class, i.e., $\hat{y}_i^{(P)}$.

The main challenge faced by ML models is generalizing beyond samples from P. In this sense, it is desirable that $\hat{h}$ generalizes to different, but related distributions Q, which is known as TL [23]. Within TL, DA is a popular framework where one seeks to improve performance on a target domain based on knowledge available in a source domain. Especially, a domain is a pair $\mathcal{D} = (\mathcal{X}, P)$, where P is a distribution the feature space $\mathcal{X}$. Likewise, a task is a pair $\mathcal{T} = (\mathcal{Y}, h_0)$, where $h_0 : \mathcal{X} \to \mathcal{Y}$ is a ground-truth labeling function. Given a source domain and task $(\mathcal{D}_S, \mathcal{T}_S)$, and a target domain and task, $(\mathcal{D}_T, \mathcal{T}_T)$, in DA one has $\mathcal{T}_S = \mathcal{T}_T$, but $P_S \neq P_T$. As a consequence, $\mathcal{D}_S \neq \mathcal{D}_T$. The goal of DA can be summarized as follows: given labeled samples from the source domain, and unlabeled samples from the target domain, find a classifier $\hat{h}$ that generalizes to samples from P_T.

In addition, one may have a scenario where source domain data is heterogeneous. In this case, one assumes that this domain is composed of several distributions, i.e., $P_{S_1}, \cdots, P_{S_N}$, for $N > 1$. This case is known in the literature as MSDA. Besides the challenge of having $P_{S_\ell} \neq P_T$, one has inter-domain shifts, i.e., $P_{S_\ell} \neq P_{S_{\ell'}}$, for $\ell \neq \ell'$.

Given our discussion so far, one needs a notion of *closeness* between P_S and P_T for having generalization to new distributions Q [27, Theorem 10]. We thus focus on DA methods that seek to reduce the distance between distributions P_S and P_T through data transformations. In a nutshell, these methods apply a mapping to $\mathbf{x}_i^{(P_S)}$, so that $\{T(\mathbf{x}_i^{(P_S)})\}_{i=1}^n$ is distributed in the same way as $\{\mathbf{x}_j^{(P_T)}\}$. This idea is illustrated in Fig. 2. This alignment supposes a criterion of *dissimilarity* between these objects. In this sense, one may use *probability metrics*, which are distances in the space of probability distributions. In our experiments, we consider three prominent metrics, namely, the $\mathcal{H}$-distance, the Maximum Mean Discrepancy (MMD) and the Wasserstein distance.
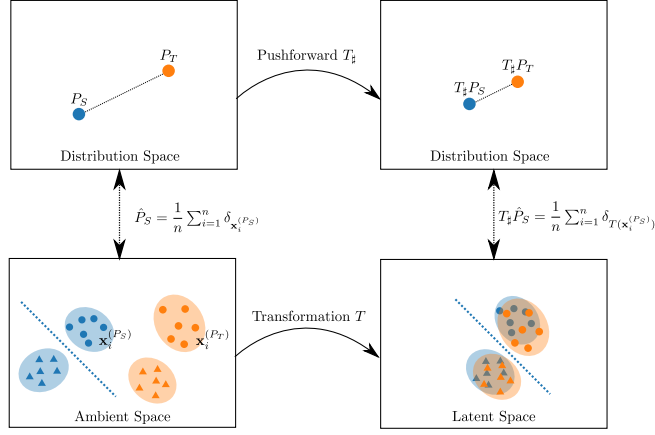
Fig. 2: **Domain adaptation based on data transformation.** In an ambient space, source and target data follow different probability distributions. As a result, a classifier learned on the source (blue straight line on the left) is not able to generalize on data from the target domain (orange elements). In this paper we consider methods that align the distributions through a data transformation T, which maps data into a latent space.

The $\mathcal{H}$-distance has its roots on DA theory [2]. This distance measures how likely a classifier can separate samples from these distributions. Hence, let $h \in \mathcal{H}$ be a classifier,

$$d_{\mathcal{H}}(\hat{P}_S, \hat{P}_T) = 2\left(1 - \min_{h \in \mathcal{H}} \left(\frac{1}{n}\sum_{i=1}^{n}\log(1 - h(\mathbf{x}_i^{(P_S)})) + \frac{1}{m}\sum_{j=1}^{m}\log h(\mathbf{x}_j^{(P_T)})\right)\right).$$
$$(2)$$

Note that the $d_{\mathcal{H}}$ can be easily estimated from samples $\{\mathbf{x}_i^{(P_S)}\}_{i=1}^{n}$ and $\{\mathbf{x}_j^{(P_T)}\}_{j=1}^{m}$, by learning a classifier that predicts the domain of a given sample (e.g., 0 for P_S, and 1 for P_T).

The MMD has its roots on kernel theory [13], and was initially proposed to test if two samples come from the same distribution. Let $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a kernel, the MMD can be defined as,

$$\text{MMD}_k(\hat{P}_S, \hat{P}_T)^2 = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}k(\mathbf{x}_i^{(P_S)}, \mathbf{x}_j^{(P_S)}) + \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}k(\mathbf{x}_i^{(P_T)}, \mathbf{x}_j^{(P_T)})$$
$$- \frac{2}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}k(\mathbf{x}_i^{(P_S)}, \mathbf{x}_j^{(P_T)}),$$
$$(3)$$

examples of kernels include the linear kernel $k(\mathbf{x}_i^{(P_S)}, \mathbf{x}_j^{(P_T)}) = (\mathbf{x}_i^{(P_S)})^{\top}\mathbf{x}_j^{(P_T)}$, and the Gaussian kernel, $k(\mathbf{x}_i^{(P_S)}, \mathbf{x}_i^{(P_T)}) = \exp{(-\gamma\|\mathbf{x}_i^{(P_S)} - \mathbf{x}_j^{(P_T)}\|_2^2)}$, for a pa-

rameter $\gamma > 0$. Intuitively, the MMD is a distance between the means of distributions in an embedding space defined by the kernel k.

Finally, the Wasserstein distance W_p is rooted on the theory of Optimal Transport (OT). In its modern computational treatment [25, 10], the OT problem can be phrased as,

$$\gamma^\star = \operatorname*{argmin}_{\gamma \in \Gamma} \sum_{i=1}^n \sum_{j=1}^m \gamma_{ij} \|\mathbf{x}_i^{(P_S)} - \mathbf{x}_j^{(P_T)}\|_2^p, \qquad (4)$$

where $\gamma \in \mathbb{R}^{n \times m}$ is called OT plan, and Γ is the set of mass preserving plans, i.e., matrices γ such that their row sum $\sum_{i=1}^n \gamma_{ij} = m^{-1}$, and column sum $\sum_{j=1}^m \gamma_{ij} = n^{-1}$. Problem 4 is a linear program, which can be solved exactly through the Simplex method [6]. Based on $\gamma^\star$, the Wasserstein distance is

$$W_p(\hat{P}_S, \hat{P}_T)^p = \sum_{i=1}^n \sum_{j=1}^m \gamma_{ij}^\star \|\mathbf{x}_i^{(P_S)} - \mathbf{x}_j^{(P_T)}\|_2^p.$$

Let $\mathcal{N}(\mu, \Sigma)$ denote the Gaussian distribution with mean $\mu \in \mathbb{R}^d$, and covariance matrix $\Sigma \in \mathcal{S}_+^d$, i.e., a $d \times d$ symmetric and positive semi-definite matrix. For $p = 2$, $P_S = \mathcal{N}(\mu_S, \Sigma_S)$ and $P_T = \mathcal{N}(\mu_T, \Sigma_T)$, the Wasserstein distance is,

$$W_2(P_S, P_T)^2 = \|\mu_S - \mu_T\|_2^2 + \mathcal{B}(\Sigma_S, \Sigma_T), \qquad (5)$$

where $\mathcal{B}$ is the Bures-metric between covariance matrices [29]. While OT-based DA methods use equation 4, equation 5 is commonly used for estimating the Wasserstein distance given samples. In this case, the parameters $(\mu_S, \Sigma_S, \mu_T, \Sigma_T)$ are the sample mean and covariance from each domain.

Table 1: Description of shallow and deep domain adaptation methods alongside the notion of distance they minimize during training.
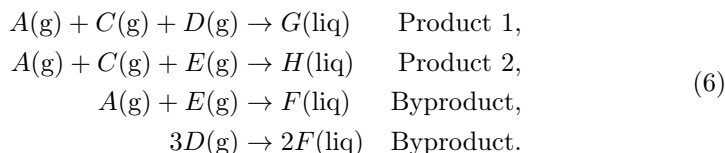
| Method | Single Source | | | Method | Multi Source | | |
	Distance	Category	Reference		Distance	Category	Reference
TCA	MMD	Shallow	[23]	M3SDA	MMD	Deep	[24]
OTDA	W_2	Shallow	[4]	M3SDA$_\beta$			
JDOT	W_2	Shallow	[3]	WJDOT	W_2	Shallow	[30]
MMD	MMD	Deep	[12]	WBT$_{reg}$	W_2	Shallow	[9, 8]
DANN	$d_{\mathcal{H}}$	Deep	[11]	DaDiL-R	W_2	Shallow	[21]
DeepJDOT	W_2	Deep	[5]	DaDiL-E			

A final distinction between DA methods is with respect their strategy. First, we consider shallow DA methods. These strategies apply transformations to pre-extracted features, in the hope of aligning the data distributions. For instance, Transfer Component Analysis (TCA) [23] projects data into a lower dimensional

space while minimizing the MMD. With respect the architecture shown in Fig. 1, these methods keep the parameters of the encoder network ϕ frozen during adaptation, and fine-tune the classifier h on the adapted data. Second, we consider deep DA methods, which rely on the encoder network ϕ for aligning the data. The principle is to minimize the distance in distribution between $\phi_\sharp P_S$ and $\phi_\sharp P_T$ (c.f., Fig. 2). This is the case of DeepJDOT [5], which minimizes the Wasserstein distance between the aforementioned distributions. In total, we consider 11 methods, as shown in Table 1. We refer readers to the original papers for further details on these algorithms.

3   Case Study: the Tennessee Eastman Process

In this section, we present our case study, the Tennessee Eastman Process (TEP). This chemical process was first introduced by [7], with the intent to serve as a realistic benchmark for the design of control and monitoring systems. From the perspective of fault detection and diagnosis [20], this system is widely used by the academic community. Henceforth, we follow the description of the TEP by [28]. The TEP consists on the production of two liquid product components, G and H, from 4 gaseous reactants, A, C, D and E, with an additional inert B and a byproduct F, which are related through 4 exothermic and irreversible reactions,

$$
\begin{aligned}
A(\text{g}) + C(\text{g}) + D(\text{g}) &\to G(\text{liq}) &&\text{Product 1,} \\
A(\text{g}) + C(\text{g}) + E(\text{g}) &\to H(\text{liq}) &&\text{Product 2,} \\
A(\text{g}) + E(\text{g}) &\to F(\text{liq}) &&\text{Byproduct,} \\
3D(\text{g}) &\to 2F(\text{liq}) &&\text{Byproduct.}
\end{aligned}
\tag{6}
$$

The TEP system is composed by five major process units: reactor, product condenser, vapor-liquid separator, recycle compressor and product stripper, shown in Fig. 3. Based on the reactions in equation 6, there are 6 different *modes of operation*, which correspond to 3 different G/H mass ratios, and a desired product rate. The different modes of operation are shown in Table 2.

From the perspective of DA, each mode of operation induces changes in the statistical properties of the data. As a result, a model learned with historical data from a set of operation modes (e.g., $1, \cdots, 5$) may not generalize to a new operation mode (e.g., 6). At the same time, collecting labeled data at the new operation mode is costly. MSDA is thus a natural solution, where one leverages historical data from previous modes to learn a better model on the new mode, only requiring unlabeled data on the new operation conditions. In section 3.1, we describe a methodology for building a MSDA benchmark on top of TEP simulations provided by [28].

To build an AFD system, we need to collect data from a set of sensors, then categorize the data into a set of faults. In this paper, we use the data provided by [28]. In their simulations, there are 53 sensors in the overall plant, corresponding to different physical and chemical quantities. We group these variables into measurements (denoted $\text{XME}(i)$, for the $i-$th measurement) and manipulated (denoted $\text{XMV}(j)$, for the $j-$th manipulation), as shown in Table 3.
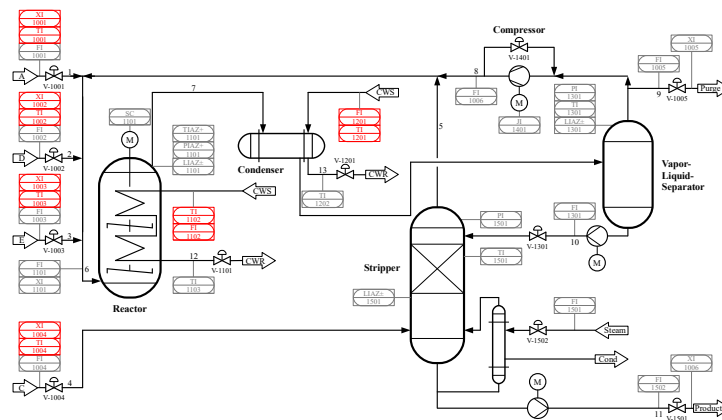
Fig. 3: **P&ID diagram for the TEP.** Figure reproduced from [1], which shows the main components of the process. Measurements originally introduced by [7] are shown in gray, whereas the measurements introduced by [1] are shown in red. A simulation environment, based on this diagram, is described in [28].

Table 2: TEP operation modes, as described in [7]. In our experiments, each mode of operation corresponds to a different domain.

Mode	Mass Ratio	Production rate
1	50/50	7038 kg h^{-1} G and 7038 kg h^{-1} H
2	10/90	1408 kg h^{-1} G and 12,669 kg h^{-1} H
3	90/10	10,000 kg h^{-1} G and 1111 kg h^{-1} H
4	50/50	maximum production rate
5	10/90	maximum production rate
6	90/10	maximum production rate

In this dataset, the TEP system is simulated for a 100 hours, with a sampling rate of 3 minutes. As such, we use each simulation as a sample in our MSDA benchmark. In each simulation, faults are introduced after 600 time steps (i.e., 30 hours). Concerning the type of faults, in their initial publication, [7] presents 20 types of process disturbances (faults 1 through 20 in Table 4), affecting different process variables. In addition to these initial faults, [1] proposed 8 additional faults under the type *random variation*, as shown in Table 4.

3.1 Benchmark preparation

Data Cleaning. For each mode, the simulations provided by [28] are divided into 3 groups: set-point variation, mode transitions and single fault. In the first case, the authors change the initial simulation set-point using a step or ramp function. In the second case, the simulation changes from one mode to another

Table 3: Description of process variables of the TEP. Variables are divided into measurements (XME) and manipulated (XMV).

Variable	Description	Variable	Description	Variable	Description	Variable	Description
XME(1)	A Feed (kscmh)	XME(15)	Stripper Level (%)	XME(29)	Component A in Purge (mol %)	XMV(2)	E Feed (%)
XME(2)	D Feed (kg/h)	XME(16)	Stripper Pressure (kPa gauge)	XME(30)	Component B in Purge (mol %)	XMV(3)	A Feed (%)
XME(3)	E Feed (kg/h)	XME(17)	Stripper Underflow (m³/h)	XME(31)	Component C in Purge (mol %)	XMV(4)	A & C Feed (%)
XME(4)	A & C Feed (kg/h)	XME(18)	Stripper Temp (°C)	XME(32)	Component D in Purge (mol %)	XMV(5)	Compressor recycle valve (%)
XME(5)	Recycle Flow (kscmh)	XME(19)	Stripper Steam Flow (kg/h)	XME(33)	Component E in Purge (mol %)	XMV(6)	Purge valve (%)
XME(6)	Reactor Feed rate (kscmh)	XME(20)	Compressor Work (kW)	XME(34)	Component F in Purge (mol %)	XMV(7)	Separator liquid flow (%)
XME(7)	Reactor Pressure (kscmh)	XME(21)	Reactor Coolant Temp (°C)	XME(35)	Component G in Purge (mol %)	XMV(8)	Stripper liquid flow (%)
XME(8)	Reactor Level (%)	XME(22)	Separator Coolant Temp (°C)	XME(36)	Component H in Purge (mol %)	XMV(9)	Stripper steam valve (%)
XME(9)	Reactor Temperature (°C)	XME(23)	Component A to Reactor (mol %)	XME(37)	Component D in Product (mol %)	XMV(10)	Reactor coolant (%)
XME(10)	Purge Rate (kscmh)	XME(24)	Component B to Reactor (mol %)	XME(38)	Component E in Product (mol %)	XMV(11)	Condenser Coolant (%)
XME(11)	Product Sep Temp (°C)	XME(25)	Component C to Reactor (mol %)	XME(39)	Component F in Product (mol %)	XMV(12)	Agitator Speed (%)
XME(12)	Product Sep Level (%)	XME(26)	Component D to Reactor (mol %)	XME(40)	Component G in Product (mol %)		
XME(13)	Product Sep Pressure (kPa gauge)	XME(27)	Component E to Reactor (mol %)	XME(41)	Component H in Product (mol %)		
XME(14)	Product Sep Underflow (m³/h)	XME(28)	Component F to Reactor (mol %)	XMV(1)	D Feed (%)		

Table 4: Description and types of faults for the TEP in the simulation environment of [28]. Faults are grouped into 4 types: step, random variation (RV), sticking and unknown.

Fault	Variable	Type	Fault Class	Variable	Type
1	A/C feed ratio, B composition constant	Step	15	Water outlet temperature (separator)	Sticking
2	B composition, A/C ratio constant	Step	16	Variation coefficient of the steam supply of the heat exchange of the stripper	RV
3	D feed temperature	Step	17	Variation coefficient of heat transfer (reactor)	RV
4	Water inlet temperature (reactor)	Step	18	Variation coefficient of heat transfer (condenser)	RV
5	Water inlet temperature (condenser)	Step	19	Unknown	Unknown
6	A feed loss	Step	20	Unknown	RV
7	C header pressure loss	Step	21	A feed temperature	RV
8	A/B/C composition of stream 4	RV	22	E feed temperature	RV
9	D feed temperature 4	RV	23	A feed flow	RV
10	C feed temperature	RV	24	D feed flow	RV
11	Water outlet temperature (reactor)	RV	25	E feed flow	RV
12	Water outlet temperature (separator)	RV	26	A & C feed flow	RV
13	Reaction kinetics	RV	27	Water flow (reactor)	RV
14	Water outlet temperature (reactor)	Sticking	28	Water flow (condenser)	RV

at an instant in time. In the third case, as previously mentioned, a fault is introduced at time step 600, i.e., after 30 hours of simulation. For each fault, there are multiple intensities available (e.g., 25%, 50%, 75% and 100% fault magnitude). For magnitudes 25%, 50% and 75%, the system is simulated 100 times, whereas for 100%, the system is simulated 200 times. As a result, for the single-fault scenario only, the data provided by [28] contains,

$$28 \text{ faults} \times 6 \text{ modes} \times 500 \text{ simulations} = 84000 \text{ simulations}.$$

Nonetheless, one should note that some simulations terminate earlier than $100h$, due to forced plant-shutdown. As a result, we adopt the following strategy: for each fault, we keep the first 100 simulations of highest magnitude that terminate successfully. For each selected simulation, we crop the signal into 2 parts. The first 30h correspond to the steady state, determined by the set point of the mode of operation. This first part of the signal characterizes the healthy state of the system (i.e., faultless state). We further sub-sample the number of faultless state signals to keep a balanced dataset (i.e., 100 per mode of operation). The second part consists on the next 30h of simulation. Since faults are introduced

exactly at the 601th time step, the second part of the signal characterize each fault. This process generates a slightly imbalanced dataset of 17289 samples[5]. We summarize the division of samples among modes of operation in Table 5.

Table 5: Number and percentage of samples from each mode of operation.

Mode of Operation	# of Samples	% of Samples
1	2900	16.77
2	2845	16.45
3	2899	16.76
4	2865	16.57
5	2883	16.67
6	2897	16.75
Total	17289	100

Variable selection and pre-processing. Out of the 53 variables presented in Table 3 some of these variables are not continuous (e.g., XME(23) through XME(41)). Given this remark, we follow [28], and consider a sub-set of 34 continuous signals as input to our neural nets. These are measurements XME(1) through XME(22), and manipulated variables XMV(1) through XMV(12). We thus have multi-variate time series of shape $(34, 600)$, where 34 is the number of sensor readings (i.e., considered variables), and 600 corresponds to the number of time steps (T). We further perform a standardization along each variable, within each mode, $x_{i,j,t}^{(P_\ell)} = (x_{i,j,t}^{(P_\ell)} - \mu_j^{(P_\ell)})/\sigma_j^{(P_\ell)}$, where,

$$\mu_j^{(P_\ell)} = \frac{1}{n_\ell T} \sum_{i=1}^{n_\ell} \sum_{t=1}^{T} x_{i,j,t}^{(P_\ell)}, \text{ and } \sigma_{i,j,t}^{(P_\ell)} = \sqrt{\frac{1}{n_\ell T} \sum_{i=1}^{n_\ell} \sum_{t=1}^{T} (x_{i,j,t}^{(P_\ell)} - \mu_j^{(P_\ell)})^2},$$

where n_ℓ is the number of samples for mode $\ell = 1, \cdots, 6$ (c.f., Table 5).
Neural Network Backbone. In DA, it is common to choose a backbone upon which methods will rely on. For instance, in image processing, residual networks [14] are widely used. In the context of time series, In our paper, we employ a Fully Convolutional Network (FCN) [19, 32, 17], which consists on three convolutional blocks followed by a Global Average Pooling (GAP) layer. Each convolutional block has a convolutional layer, and a normalization layer. In our experiments we verified that instance normalization [31] improves stability and performance over other normalization layers such as batch normalization [15].

3.2   Exploratory Data Analysis

Qualitative Analysis. We analyze the pairwise correlations of variables, conditioned on the type of fault. In Fig. 4, we illustrate a change in the pattern of

---

[5] The data used in our experiments is available on Kaggle: `https://www.kaggle.com/datasets/eddardd/tennessee-eastman-process-domain-adaptation/data`

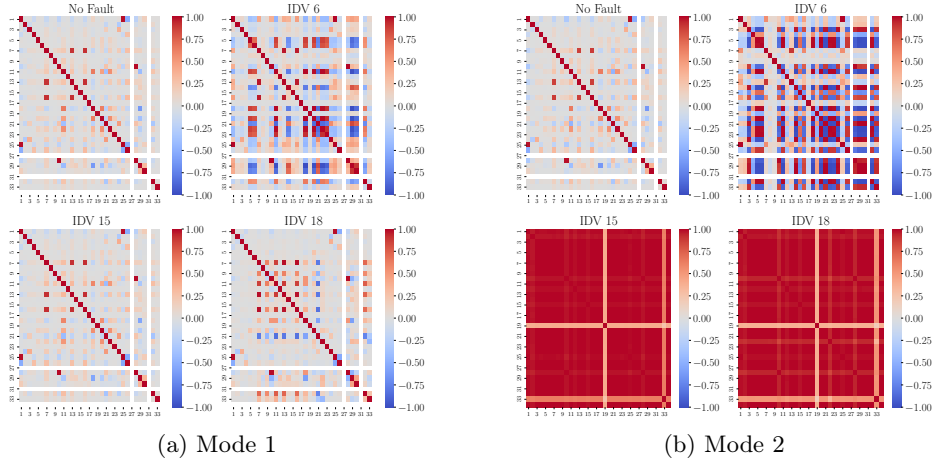(a) Mode 1   (b) Mode 2

Fig. 4: **Qualitative analysis of distributional shift.** In (a) and (b), we show the correlation between different variables in TEP, for modes 1 and 2, for no fault, and IDVs 6, 15 and 18.

correlations between variables, conditioned on the fault type, for modes 1 and 2 for a 4 cases: no fault, IDV 6, 15 and 18. From this figure, we conclude that the mode of production deeply impacts the dynamic of the system, which creates a shift in distribution between data from these modes.

(a) Pairwise W_2.   (b) Mode embeddings.
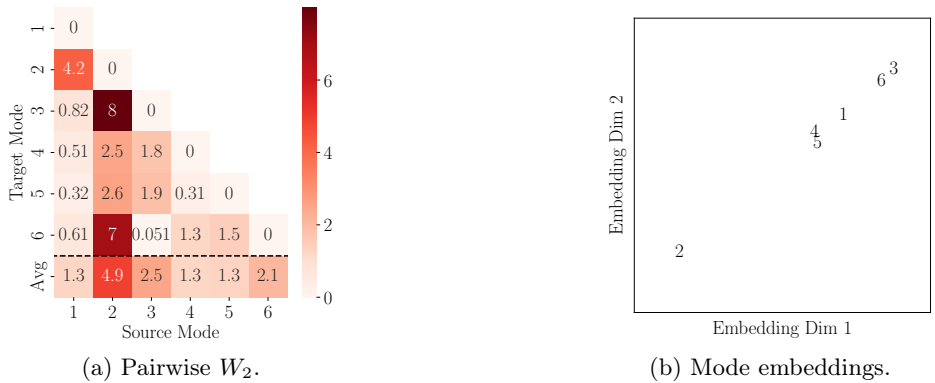
Fig. 5: **Quantitative analysis of distributional shift**. Pairwise Wasserstein distance between modes (a). Mode embeddings based on MDS (b).

Quantitative Analysis. We quantify the shift between pairs of modes through the probability metrics introduced in section 2. We estimate the pairwise Wasserstein distances between modes using eq. 5. In Fig. 5 (a), we show the pairwise

distance in probability distribution between different modes. On one hand, the most different mode with respect others is Mode 2, which is especially far from modes 1, 3 and 5. On the other hand, the most similar modes are 3 and 6. We can have a better picture about the level of similarity of these different domains by embedding them on the plane, as shown in Fig. 5 (b). We obtain these embeddings through Multi-Dimensional Scaling (MDS) [18], which defines the points in $\mathbb{R}^2$ while preserving the pairwise distances between the embeddings.

From our qualitative and quantitative analysis, we expect lower performances with respect the adaptation towards mode 2, as it is the most dissimilar from other modes (c.f., Fig. 5 (a), average row). In contrast, adaptation between modes $(3, 6)$, and $(1, 4, 5)$ should work well as these modes share statistical characteristics. We verify these indications empirically in the next sections.

3.3   Single-Source Domain Adaptation

In this section, we explore single-source DA, i.e., when adaptation is done from a single source mode, to a single target mode. On the one hand, we refer to *generalization*, to the ability of a classifier to perform well on unseen data from an unseen domain. On the other hand, we refer to *adaptation*, when a classifier performs well on unseen data from the target domain. In this context, we have
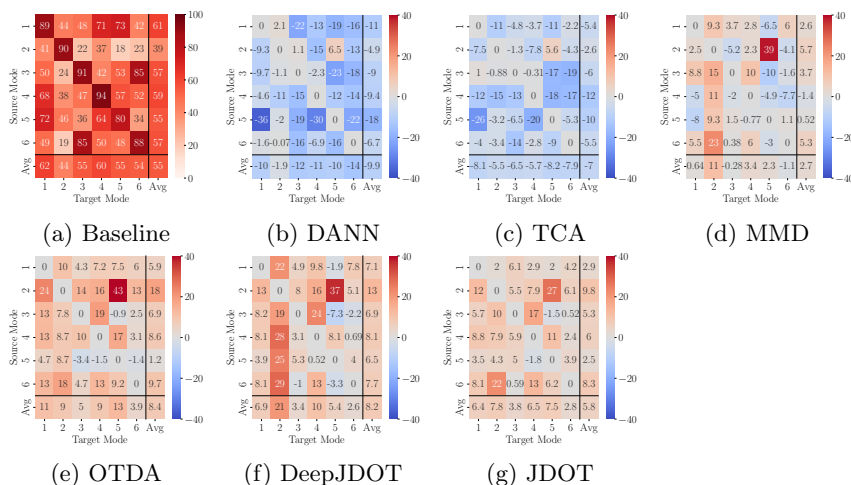

Fig. 6: Baseline (a) and single-source domain adaptation (b) algorithms.

2 baselines. The first, *source-only*, considers that a classifier is learned exclusively with source domain data (i.e., no adaptation). This corresponds to the off-diagonals of Figure 6 (a). Second, we have *target-only*, which trains and evaluates models on the target domain (i.e., no distributional shift). Note that the target-only scenario has an advantage over other methods, as it has access to

labeled data in the target domain. This baseline can be seen as an upper bound in the adaptation performance.. With respect these scenarios, note that we verify our previous remarks, i.e., generalization towards mode 2 is much more difficult than other domains, and the clusters of similar domains (e.g., $(3, 6)$ and $(1, 4, 5)$) generalize well.

We further compare the single-source DA methods presented in Table 1, which are shown in Fig. 6 (b) through (g). Overall, we find that OT-based methods have a higher performance than other metrics (e.g., the MMD). This is similar to previous findings on smaller scale problems, such as [22]. The best performing method is Optimal Transport Domain Adaptation (OTDA) [4], which maps source domain points to the target domain points through the OT plan (c.f., eq. 4). Nonetheless, one should be mindful of *negative transfer* [33] between similar modes (e.g., $3 \rightarrow 5$), which may result in performance degradation.

3.4   Multi-Source Domain Adaptation

In this section, we explore multi-source DA, i.e., when adaptation is done from multiple source domains towards a single target. Here, note that the models have much more labeled available data, as all domains are considered at once. We start our discussion by comparing the performance of single, source-only baselines, and the corresponding multi-source baseline for each target mode, which is shown in Fig. 7. Overall, the multi source-only baseline improve over single single-only for the same target. These baselines have similar performance when there are pairs of highly similar modes (e.g., modes 3 and 6), showing that extra data from additional modes is not as informative for generalization.
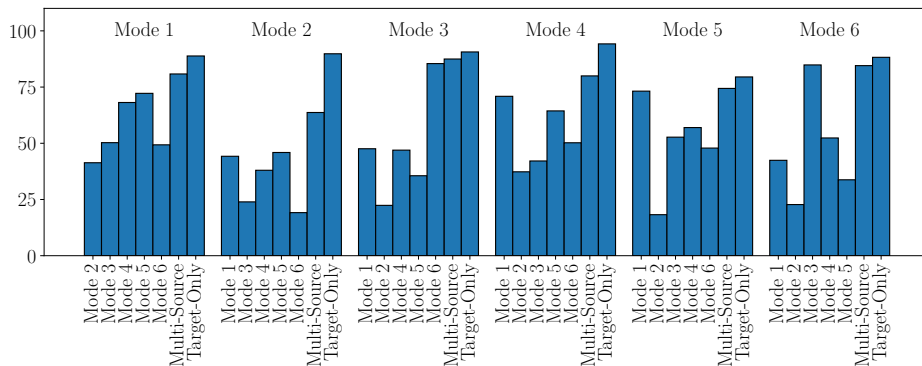

Fig. 7: **Multi and single-source baseline comparison.** On top, we show the target domain. In the abscissa, we show the corresponding baseline. The multi-source scenario generally improves over the single source-only case.

We now consider the performance of DA algorithms in the multi-source setting. Besides native MSDA algorithms, i.e., algorithms that suppose the source

as composed by different domains, we also consider single-source algorithms with access to the concatenation of all source domains. Our comparison is shown in Fig. 8. A first question is whether access to additional data is beneficial to adaptation. For instance, in single-source DA, methods exhibited negative transfer in the task $3 \to 5$. When provided access to data from all domains, all single-source adaptation method performance improved over the single-source baseline. As a result, even though data from multiple domains may not improve generalization, it does improves adaptation.
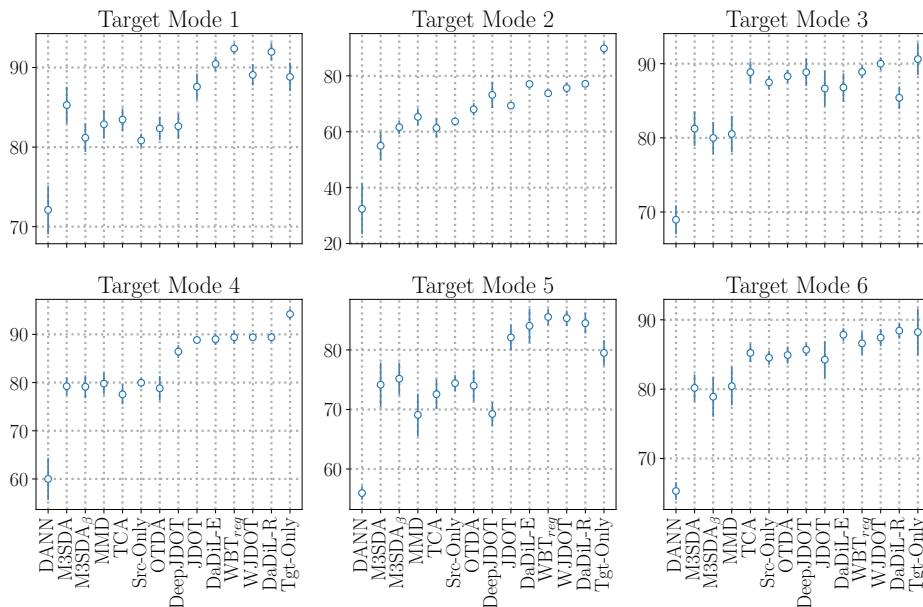

Fig. 8: **Multi-source domain adaptation results.** We compare all algorithms with access to labeled data from all source domains, except the target mode, from which we have access to unlabeled data. Methods in the abcisssa are ordered by average performance on all modes.

With respect Fig. 8, from the perspective of MSDA, methods that weight sources in a linear space, such as WJDOT, or in a Wasserstein space, such as WBT and DaDiL outperform the weighting of classifiers' predictions, such as $M3SDA_\beta$ [24]. On the one hand, WJDOT can filter undesirable information during adaptation by assigning small weights to domains and samples. On the other hand, WBT and DaDiL combine the information in the sources non-linearly. These two strategies are effective in domain adaptation.

Finally, from Fig. 9, we can see that shallow DA methods (e.g., JDOT) generally improve over deep DA methods (e.g., DeepJDOT). Indeed, deep DA methods learn features that are invariant to the domain shift between different

modes. As a result, these features may be less useful for classification. In a general note (both single, and multi-source methods), OT-based techniques outperform methods based on other distances, such as the MMD and $d_{\mathcal{H}}$. This remark agrees with previous studies on smaller scale systems [22].
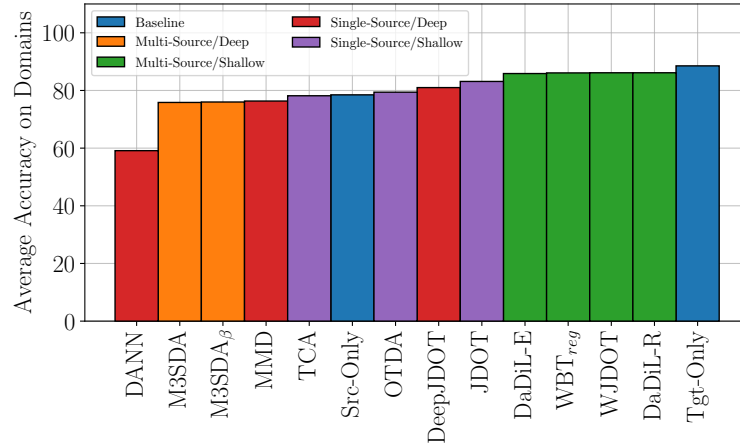

Fig. 9: Comparison of average adaptation performance of DA algorithms in the multi-source setting.

4   Conclusion

In this paper, we introduce a new benchmark for domain adaptation algorithms based on the Tennessee Eastman process [7]. The present benchmark is created by applying pre-processing steps on the simulations provided by [28] (c.f., section 3), thus creating a large scale dataset of time series. These time series are associated with different modes of production. Based on each mode of production, the statistical properties of the time series change (c.f., Fig. 4) creating a shift in the data probability distribution (c.f., Fig. 5). As a result, data trained on a specific mode may not generalize well to other modes of production, thus the need for domain adaptation. Through a series of experiments with single-source and multi-source domain adaptation methods, we show that OT-based methods outperform methods that rely on the maximum mean discrepancy, and $\mathcal{H}-$distances, which agrees with previous findings on smaller scale systems [22]. Besides providing the open source code for the reproduction of our benchmark, with this work we hope to encourage research on the intersection between domain adaptation and fault diagnosis [34].

References

1. Bathelt, A., Ricker, N.L., Jelali, M.: Revision of the tennessee eastman process model. IFAC-PapersOnLine **48**(8), 309–314 (2015)
2. Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., Vaughan, J.W.: A theory of learning from different domains. Machine learning **79**(1-2), 151–175 (2010)
3. Courty, N., Flamary, R., Habrard, A., Rakotomamonjy, A.: Joint distribution optimal transportation for domain adaptation. Advances in neural information processing systems **30** (2017)
4. Courty, N., Flamary, R., Tuia, D., Rakotomamonjy, A.: Optimal transport for domain adaptation. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(9), 1853–1865 (2017). https://doi.org/10.1109/TPAMI.2016.2615921
5. Damodaran, B.B., Kellenberger, B., Flamary, R., Tuia, D., Courty, N.: Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 447–463 (2018)
6. Dantzig, G.B., Orden, A., Wolfe, P., et al.: The generalized simplex method for minimizing a linear form under linear inequality restraints. Pacific Journal of Mathematics **5**(2), 183–195 (1955)
7. Downs, J.J., Vogel, E.F.: A plant-wide industrial process control problem. Computers & chemical engineering **17**(3), 245–255 (1993)
8. Fernandes Montesuma, E., Mboula, F.M.N.: Wasserstein barycenter for multi-source domain adaptation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16785–16793 (2021)
9. Fernandes Montesuma, E., Mboula, F.M.N.: Wasserstein barycenter transport for acoustic adaptation. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 3405–3409. IEEE (2021)
10. Fernandes Montesuma, E., Mboula, F.N., Souloumiac, A.: Recent advances in optimal transport for machine learning. arXiv preprint arXiv:2306.16156 (2023)
11. Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V.: Domain-adversarial training of neural networks. The journal of machine learning research **17**(1), 2096–2030 (2016)
12. Ghifary, M., Kleijn, W.B., Zhang, M.: Domain adaptive neural networks for object recognition. In: Pacific Rim international conference on artificial intelligence. pp. 898–904. Springer (2014)
13. Gretton, A., Borgwardt, K.M., Rasch, M., Schölkopf, B., Smola, A.J.: A kernel approach to comparing distributions. In: Proceedings of the National Conference on Artificial Intelligence. vol. 22, p. 1637. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999 (2007)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
15. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning. pp. 448–456. pmlr (2015)
16. Isermann, R.: Fault-diagnosis systems: an introduction from fault detection to fault tolerance. Springer Science & Business Media (2006)
17. Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Deep learning for time series classification: a review. Data mining and knowledge discovery **33**(4), 917–963 (2019)

18. Kruskal, J.B.: Multidimensional scaling by optimizing goodness of fit to a non-metric hypothesis. Psychometrika **29**(1), 1–27 (1964)
19. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3431–3440 (2015)
20. Melo, A., Câmara, M.M., Clavijo, N., Pinto, J.C.: Open benchmarks for assessment of process monitoring and fault diagnosis techniques: A review and critical analysis. Computers & Chemical Engineering **165**, 107964 (2022)
21. Montesuma, E.F., Mboula, F., Souloumiac, A.: Multi-source domain adaptation through dataset dictionary learning in wasserstein space. In: European Conference on Artificial Intelligence. pp. 1739–1745 (09 2023). https://doi.org/10.3233/FAIA230459
22. Montesuma, E.F., Mulas, M., Corona, F., Mboula, F.M.N.: Cross-domain fault diagnosis through optimal transport for a cstr process. IFAC-PapersOnLine **55**(7), 946–951 (2022)
23. Pan, S.J., Tsang, I.W., Kwok, J.T., Yang, Q.: Domain adaptation via transfer component analysis. IEEE transactions on neural networks **22**(2), 199–210 (2010)
24. Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., Wang, B.: Moment matching for multi-source domain adaptation. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 1406–1415 (2019)
25. Peyré, G., Cuturi, M.: Computational optimal transport: With applications to data science. Foundations and Trends® in Machine Learning **11**(5-6), 355–607 (2019)
26. Quiñonero-Candela, J., Sugiyama, M., Lawrence, N.D., Schwaighofer, A.: Dataset shift in machine learning. Mit Press (2009)
27. Redko, I., Morvant, E., Habrard, A., Sebban, M., Bennani, Y.: Advances in domain adaptation theory. Elsevier (2019)
28. Reinartz, C., Kulahci, M., Ravn, O.: An extended tennessee eastman simulation dataset for fault-detection and decision support systems. Computers & Chemical Engineering **149**, 107281 (2021)
29. Takatsu, A.: Wasserstein geometry of Gaussian measures. Osaka Journal of Mathematics **48**(4), 1005 – 1026 (2011)
30. Turrisi, R., Flamary, R., Rakotomamonjy, A., et al.: Multi-source domain adaptation via weighted joint distributions optimal transport. In: The 38th Conference on Uncertainty in Artificial Intelligence (2022)
31. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022 (2016)
32. Wang, Z., Yan, W., Oates, T.: Time series classification from scratch with deep neural networks: A strong baseline. In: 2017 International joint conference on neural networks (IJCNN). pp. 1578–1585. IEEE (2017)
33. Zhang, W., Deng, L., Zhang, L., Wu, D.: A survey on negative transfer. IEEE/CAA Journal of Automatica Sinica **10**(2), 305–329 (2022)
34. Zheng, H., Wang, R., Yang, Y., Yin, J., Li, Y., Li, Y., Xu, M.: Cross-domain fault diagnosis using knowledge transfer strategy: a review. IEEE Access **7**, 129260–129290 (2019)