Cross-domain fault diagnosis through optimal transport for a CSTR process

Eduardo Fernandes Montesuma [*,**] **Michela Mulas** [*]
Francesco Corona [***] **Fred-Maurice Ngole Mboula** [**]

[*] *Department of Teleinformatics Engineering, Federal University of Ceará, Brazil (e-mails: eduardomontesuma@alu.ufc.br, michela.mulas@ufc.br)*
[**] *Université Paris-Saclay, CEA, List, F-91120 Palaiseau, France (email: fred-maurice.ngole-mboula@cea.fr)*
[***] *School of Chemical Engineering, Aalto University, Finland (e-mail: francesco.corona@aalto.fi)*

Abstract: Fault diagnosis is a key task for developing safer control systems, especially in chemical plants. Nonetheless, acquiring good labeled fault data involves sampling from dangerous system conditions. A possible workaround to this limitation is to use simulation data for training data-driven fault diagnosis systems. However, due to modelling errors or unknown factors, simulation data may differ in distribution from real-world data. This setting is known as cross-domain fault diagnosis (CDFD). We use optimal transport for: (i) exploring how modelling errors relate to the distance between simulation (source) and real-world (target) data distributions, and (ii) matching source and target distributions through the framework of optimal transport for domain adaptation (OTDA), resulting in new training data that follows the target distribution. Comparisons show that OTDA outperforms other CDFD methods.

Keywords: Fault Diagnosis, Optimal Transport, Transfer Learning

1. INTRODUCTION

Faults are unpermitted deviation or anomalies of characteristic properties or variables in a system (Isermann, 2006). If not timely corrected, they might evolve into serious accidents and cause significant safety, environmental and economic impacts (Chiang et al., 2000). Fault detection and diagnosis techniques provide the framework to identify such anomalies and assure safe processes. Detection aims at assessing the occurrence of a fault; diagnosis evaluates the type of fault for allowing intervention (Isermann and Balle, 1997).

This work focuses on the diagnosis task, assuming that the faults have already been detected and identified. The diagnosis is regarded as a supervised classification problem: We learn features from faulty data, thus constructing a feature space where concerned fault types can be satisfactorily classified. Examples of this approach are given by, *inter alia*, Su et al. (2014), Zhang et al. (2015) and Wang et al. (2017).

Lately, deep learning and convolutional neural networks (CNN) have been used for extracting meaningful representations from signals (He and He, 2017; Jiang et al., 2017; Hoang and Kang, 2019) showing that good classification accuracy can be achieved. Nevertheless, deep learning is known to be data-intensive, thus a large amount of labelled faulty data is needed for training. This requirement is not always met in the diagnosis tasks since acquiring faulty data can yield economic and security hazards. Instead, simulation data can be used for the design of diagnosis models which might then be deployed on real-world situations. However, even with reliable simulation models, the train and test data used for diagnosis should follow an identical probability distribution. Unknown process conditions may not satisfy this requisite. Domain adaptation (Pan and Yang, 2009) tackle the problem by transferring knowledge from a source domain with the labelled training data to a target domain. Zheng et al. (2019) intersect fault diagnosis and domain adaptation in the so-called cross-domain fault diagnosis (CDFD).

There are various methods for reducing the gap between train and test distributions (Pan et al., 2010; Ganin et al., 2016). For instance, optimal transport theory is applied for performing CDFD by Cheng et al. (2019) and minimising the gap between distributions. However, this strategy requires models to be trained from scratch, which is not always feasible due to data availability. The framework of Courty et al. (2017), however, only requires solving an optimal transport problem in the feature space, thus leveraging the existence of pre-trained models. In this work, we investigate the framework of optimal transport for domain adaptation (OTDA) (Courty et al., 2017), which has already proven itself to yield state-of-the-art performance in image and natural language processing tasks and we apply it to a benchmark continuous stirred tank reactor (CSTR). To the best of our knowledge this work is the first to apply OTDA for performing CDFD.

The paper is organised as follows: Section 2 provides the background on CDFD and optimal transport. Section 3 discusses the application on the CSTR, presenting the

system details and a comparative study of transfer learning algorithms. Section 4 draws our conclusions.

2. DESIGN OF THE FAULT DIAGNOSIS MODEL

Assuming that a mathematical model for the processes is available, simulated data are acquired and features are extracted from them. These features serve as input to a fault classifier, that uses the classification results for fault diagnosis. We discuss the steps for feature extraction and classification in section 2.1. The link between CDFD and domain adaptation is presented in section 2.2. Optimal transport for domain adaptation is defined in section 2.3.

2.1 Fault Classification

We consider the problem of fault diagnosis through a classification perspective. We use a raw signal-based support vector machine (SVM) and CNN as classifiers that serve as baseline for comparing the proposed CDFD strategies.

Consider an arbitrary process with m variables of which we assume to have available a simulation model. We create a dataset $D = \{\mathbf{x}_i, y_i\}_{i=1}^n$ of n simulations where $\mathbf{x}_i \in \mathbb{R}^{d=T\times m}$ is the concatenation of the m variables sampled through T time steps, and y_i is the label assigned to each $k \in \{1, \cdots, K\}$ class.

SVM tries to find the parameters $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ such that the separation between classes is maximal. This corresponds to solving the optimisation problem:

$$\underset{\mathbf{w},b}{\text{argmax}} \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{subject to } y_j(\mathbf{w}^T\mathbf{x}_j + b) \geq 1 - \xi_j$$

where $C \geq 0$ is a penalty term and ξ_i is the proportion of prediction on the wrong side of the hyperplane. When dealing with multiple faults, a possible strategy is using the one-versus-rest approach. This consists of fitting different classifiers for distinguishing between class k, and all other classes $j \neq k$. Each classifier has parameters $\mathbf{w}_k, b_k$. Thus, one has instead a set of K functions. The prediction can be retrieved with $\hat{y} = h(\mathbf{x}) = \underset{k=1,\ldots,K}{\text{argmax}}\, h_k(\mathbf{x})$.

CNNs rely on the convolution operation defined as

$$\mathbf{s}_\ell = (\mathbf{x} \star \mathbf{g})_\ell = \sum_{i=1}^{m'} x_{\ell+i} g_\ell,$$

where $\mathbf{x} \in \mathbb{R}^d$ is the vector containing the process variables, $\mathbf{g} \in \mathbb{R}^{m'}$ is a filter of dimension m', and $\ell = 1, \ldots, d-m'$ is an index. By stacking various convolutional layers sequentially, we extract an abstract representation vector $\mathbf{r} \in \mathbb{R}^{d'}$. This approach is known as representation learning (Bengio et al., 2013). On top of these convolutional layers, the standard approach is to stack fully connected layers, which are based on matrix-vector multiplications, that is $\mathbf{y} = \varphi(\mathbf{A}\mathbf{r} + \mathbf{b})$ for $\mathbf{A} \in \mathbb{R}^{K\times d'}$ and $\mathbf{b} \in \mathbb{R}^K$.

The function φ is the activation function, commonly a rectified linear unit (ReLU, Glorot et al. (2011)) or a softmax. While the ReLU activation is typically used in the intermediate layers, the softmax is used at the output layer as its result corresponds to a probability distribution over possible classes. In this case, a common choice of loss function for minimisation is the cross-entropy:

$$\ell(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{K} y_i \log \hat{y}_i.$$

It is important to mention that both of these algorithms assume that training and test samples are independent and identically distributed.

2.2 Cross-Domain Fault Diagnosis as Domain Adaptation

CDFD can be used when training and test faulty data follow different probability distributions. A key assumption is that, even though different, these two domains are related or similar (e.g., they refer to the same process but from new conditions). As discussed by Zheng et al. (2019), the main consequence of data following different probability distributions is a misalignment between classes in the feature space. We illustrate this idea in Figure 1. In red are shown the source domain samples and in blue the target domain samples. For simplicity, the distributional shift is illustrated as a rotation. As consequence of distributional shift, classifiers trained with source domain data may not classify target domain samples correctly.

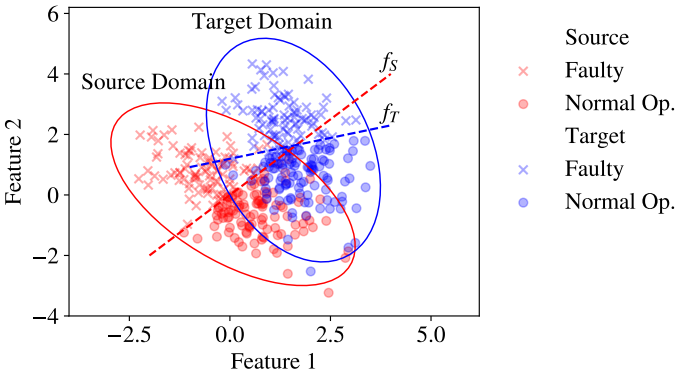

Fig. 1. Illustration example of feature misalignment.

This scenario in which feature are distributed according to different probability distributions is known as domain adaptation. We follow Pan and Yang (2009) and consider a domain to be a pair $\mathcal{D} = (\mathcal{X}, P(X))$, where $\mathcal{X}$ is a feature space and $P(X)$ its marginal distribution. Moreover, a task is a pair $\mathcal{T} = (\mathcal{Y}, f)$ where $\mathcal{Y}$ is a label space, and $f(x) = P(Y|X = x)$ is the ground-truth labelling function.

We further identify domain adaptation as the case where one has different, but related domains. Furthermore, these domains differ with respect to their probability distributions. We denote by $D_S = \{\mathbf{x}_S^i, y_S^i\}_{i=1}^{n_S}$ to the source domain data, and $D_T = \{\mathbf{x}_T^j\}_{j=1}^{n_T}$ to the target data, where n_S and n_T represent the number of source and target samples, respectively. Note that target labels y_T^j are not available at training time.

According to Pan and Yang (2009), the goal of domain adaptation is to help learning a predictive function on the target domain, by using knowledge of the source domain.

This can be done in different ways. For instance, Gong et al. (2012) proposed principal component analysis (PCA) as a naive technique for reducing the divergence between domains. The idea consists simply on projecting the data on a sub-space through a linear function $\phi(\mathbf{x}) = \mathbf{W}\mathbf{x}$, such that source and target domain share common characteristics on this sub-space.

In addition, robust techniques take a notion of divergence explicitly into account. For instance, Ganin et al. (2016) propose using the so-called $\mathcal{H}-$divergence of Ben-David et al. (2007) for guiding the representation learning of CNNs. The recent interest in optimal transport and the Wasserstein distance has inspired many advances in machine learning. We may also minimize the Wasserstein distance between source and target distributions, which corresponds to the OTDA approach (Courty et al., 2017).

2.3 Optimal Transport for Domain Adaptation

Optimal transport (OT) is a theory describing how mass can be transported between a source and a target under least effort. Note that, as there is a natural analogy between mass and probability, this theory can be used for transporting probability distributions into one another. In this section we focus on the discrete Kantorovich formulation of OT, as presented by Peyré and Cuturi (2019).

For establishing the probabilistic modeling, let P_S and P_T be the underlying probability distribution of features, from which source and target data are sampled from, that is $\mathbf{x}_S^i \sim P_S$ and $\mathbf{x}_T^j \sim P_T$. These distributions are unknown. However, given that there are enough samples, they may be approximated empirically:

$$\hat{P}_S(\mathbf{x}) = \frac{1}{n_S}\sum_{i=1}^{n_S}\delta(\mathbf{x}-\mathbf{x}_S^i),\ \hat{P}_T(\mathbf{x}) = \frac{1}{n_T}\sum_{i=1}^{n_T}\delta(\mathbf{x}-\mathbf{x}_T^i),$$

where δ is the Dirac delta function. Furthermore, we denote the support of $\hat{P}_S$ as $\mathbf{X}_S = [\mathbf{x}_S^i]_{i=1}^{n_S} \in \mathbb{R}^{n_S\times p}$ and the support of $\hat{P}_T$ as $\mathbf{X}_T = [\mathbf{x}_T^j]_{j=1}^{n_T}$. Using this notation, the samples $\mathbf{x}_S^i$ have uniform mass n_S^{-1}.

The Kantorovich formulation focus on finding a *transportation plan* $\gamma \in \mathbb{R}^{n_S\times n_T}$. Intuitively, γ_{ij} represents the amount of mass or probability transported between source sample $\mathbf{x}_S^i$ to target sample $\mathbf{x}_T^j$. This formulation further assumes that γ preserves mass. Thus, the set of all mass-preserving plans is denoted as:

$$\Pi(\hat{P}_S,\hat{P}_T) = \{\gamma|\gamma^T\mathbf{1}_{n_S} = n_T^{-1}\mathbf{1}_{n_T}, \gamma\mathbf{1}_{n_T} = n_S^{-1}\mathbf{1}_{n_S}\}$$

where $\mathbf{1}_n = [1,\cdots,1] \in \mathbb{R}^n$. Π is the set of doubly stochastic matrices. Thus, let $C_{ij} = c(\mathbf{x}_S^i,\mathbf{x}_T^j)$ be the cost of transporting $\mathbf{x}_S^i$ to $\mathbf{x}_T^j$, the optimal transport plan can be found by

$$\gamma^\star = \underset{\gamma\in\Pi(\hat{P}_S,\hat{P}_T)}{\operatorname{argmin}} \sum_{i=1}^{n_S}\sum_{j=1}^{n_T}\gamma_{ij}C_{ij}. \tag{1}$$

Note that this is a linear program with variables γ_{ij}, thus, there are known algorithms for finding $\gamma^\star$, such as the simplex algorithm developed by Dantzig et al. (1955). In this formulation, $\gamma^\star$ is the least effort plan for transporting $\hat{P}_S$ to $\hat{P}_T$. When the ground-cost c is a distance, the innermost term in 1 is proportional to the work of moving γ_{ij} units of mass from $\mathbf{x}_S^i$ to $\mathbf{x}_T^j$. This allows for the Wasserstein distance between probability distributions:

$$W_2(\hat{P}_S,\hat{P}_T)^2 = \min_{\gamma\in\Pi(\hat{P}_S,\hat{P}_T)}\sum_{i=1}^{n_S}\sum_{j=1}^{n_T}\gamma_{ij}||\mathbf{x}_S^i-\mathbf{x}_T^j||^2.$$

Intuitively, two probability distributions are far apart under the Wasserstein distance if the work needed for transporting one into another is high.

The intersection between OT theory and CDFD comes when using the Wasserstein distance as the choice of divergence for matching source and target distributions. This can be represented as an optimization problem:

$$\hat{P}_S^\star = \underset{P}{\operatorname{argmin}}\, W_p(P,\hat{P}_T).$$

Assuming that $\hat{P}_S^\star$ is supported on the same number of points as $\hat{P}_S$, with constant sample weights n_S^{-1}, this minimisation is equivalent to transforming the source data points with the barycentric mapping (Courty et al., 2017), defined as:

$$T_{\gamma^\star}(\mathbf{x}_S^i) = \hat{\mathbf{x}}_S^i = \underset{\mathbf{x}\in\mathbb{R}^d}{\operatorname{argmin}}\sum_{j=1}^{n_T}\gamma_{ij}^\star c(\mathbf{x},\mathbf{x}_T^j),$$

where $\gamma^\star$ is the optimal transport plan between $\hat{P}_S$ and $\hat{P}_T$. When the ground-cost is the Euclidean distance, $T_{\gamma^\star}$ has a closed-form expression, given by,

$$T_{\gamma^\star}(\mathbf{X}_S) = \hat{\mathbf{X}}_S = \operatorname{diag}(\gamma^\star\mathbf{1}_{n_T})^{-1}\gamma^\star\mathbf{X}_T.$$

Once $\gamma^\star$ has been estimated, the support $\mathbf{X}_S$ may be transported to $\hat{\mathbf{X}}_S$. This allows fitting a classifier on $\hat{D}_S = \{\hat{\mathbf{x}}_S^i, y_S^i\}_{i=1}^{n_S}$, which hopefully leads to a performance improvement on the target domain.

Finally, when more than one target domain is available, we use a score consisting of the average accuracy on each domain for establishing an overall comparison across all target domains. Let S and T_m refer to the source and $m-$th target domain respectively. Let h_{S,T_m} be the classifier fit on the source domain with the $m-$th target domain data used for adaptation. The score can be defined as

$$\text{score} = \frac{1}{n_d}\sum_{m=1}^{n_d}\frac{1}{n_{T_m}}\sum_{j=1}^{n_{T_m}} I[h_{S,T_m}(\mathbf{x}_T^j) = f_{T_m}(\mathbf{x}_T^j)], \tag{2}$$

where n_d is the number of domains, n_{T_m} and f_{T_m} are the number of samples and ground-truth labeling function on the $m-$th target domain respectively. Finally, $I[\cdot]$ is the indicator function.

3. CONTINUOUS STIRRED TANK REACTOR

We consider the constant holdup jacketed CSTR in Figure 2. The reactor, that carries an exothermic reaction $A \to B$, is employed by Pilario and Cao (2018) for fault detection and further studied in the context of domain adaptation by Li et al. (2020) for fault diagnosis.

The reagent A in the influent has concentration C_A and temperature T_i. The product B in the effluent has concentration C_B and temperature T. The coolant enters the

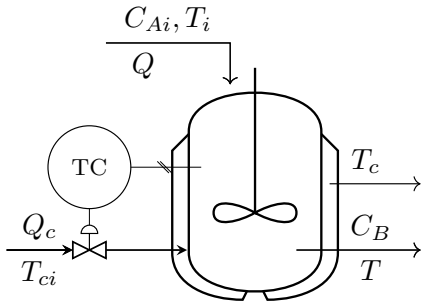

Fig. 2. Closed-loop CSTR (Li et al., 2020).

reactor jacket with flow-rate Q_c and temperature T_{ci}, T_c is its outlet temperature. The dynamics of the reactor are represented by the state variables $[C_B, T, T_c]$, the inputs $[C_A, T_i.T_{ci}]$ and the outputs $[C_B, T, T_c, Q_c]$. The state-space model is given by equations (3) where $\nu_{1,2,3} \sim \mathcal{N}(0, 10^{-2})$ are process noise and $k = k_0 \exp(-E/RT)$ is the rate constant. The process variables and parameters are listed in Table 1. Figure 2 presents the measurements and the controller of the CSTR where the coolant flow-rate Q_c is used for controlling the temperature T of the effluent as in Pilario and Cao (2018).

$$\begin{aligned}
\dot{C}_B &= \frac{Q}{V}(C_{Ai} - C_B) - akC_B^N + \nu_1 \\
\dot{T} &= \frac{Q}{V}(T_i - T) - a\frac{\Delta H_r k C_B^N}{\rho C_p} - b\frac{UA}{\rho C_p V}(T - T_c) + \nu_2 \\
\dot{T}_c &= \frac{Q_c}{V_c}(T_{ci} - T_c) + b\frac{UA}{\rho_c C_{pc} V_c}(T - T_c) + \nu_3
\end{aligned} \tag{3}$$

Table 1. Process variables and parameters.

Symbol	Description	Units	Value
Process variables			
C_A	Reagent A concentration	mol/L	
C_B	Product B concentration	mol/L	
T	Reactor temperature	K	
T_c	Jacket temperature	K	
Q_c	Coolant flow-rate	L/min	
Process inputs			
C_{Ai}	Influent concentration of A	mol/L	
T_i	Influent temperature	K	
T_{ci}	Influent coolant temperature	K	
Process parameters			
Q	Influent flow-rate	L/min	100
V	Reactor volume	L	150
V_c	Jacket volume	L	10
ΔH_r	Heat of reaction	cal/mol	$-2 \cdot 10^{-5}$
UA	Heat transfer coefficient	K·cal/min	$7 \cdot 10^5$
k_0	Pre-exponential factor of k	1/min	$7.2 \cdot 10^{10}$
E/R	Activation energy	K	10^4
ρ	Fluid density	g/L	1000
ρ_c	Coolant density	g/L	1000
C_p	Fluid heat capacity	K·cal/g	1
C_{pc}	Coolant heat capacity	K·cal/g	1
N	Reaction order	-	1

3.1 Process faults

Table 2 lists the 3 scenarios and 12 faults (Li et al., 2020) considered for performing fault diagnosis on the CSTR.

The first scenario is related to process faults, such as catalyst decay that represents a shortage in the reaction catalyst, leading to a decay in the transformation of A into B and heat fouling that represents a decay on the amount of heat exchanged between the tank and the jacket, thus leading to an increase in the tank's outlet temperature T, and a decrease on the jacket outlet temperature T_c.

Table 2. Scenarios considered for the CSTR.

Fault Class	Description		δ
Scenario I: Process Faults			
1	Catalyst decay	$a = a_0 exp(-\delta t)$	0.004
2	Heat transfer fouling	$b = b_0 exp(-\delta t)$	0.005
Scenario II: Sensor Faults			
3	Sensor bias	$\tilde{C}_i = C_i + \delta t$	0.005
4	Sensor bias	$\tilde{T}_i = T_i + \delta t$	0.1
5	Sensor bias	$\tilde{T}_{ci} = T_{ci} + \delta t$	0.1
6	Sensor bias	$\tilde{C} = C + \delta t$	0.005
7	Sensor bias	$\tilde{T} = T + \delta t$	0.1
8	Sensor bias	$\tilde{T}_c = T_c + \delta t$	0.1
9	Sensor bias	$\tilde{Q}_c = Q_c + \delta t$	-0.2
Scenario III: Process Disturbance			
10	Reactant concentration	$\Delta C_i \sim \mathcal{N}(0, \delta)$	0.005
11	Reactant temperature	$\Delta T_i \sim \mathcal{N}(0, \delta)$	5
12	Coolant temperature	$\Delta T_{ci} \sim \mathcal{N}(0, \delta)$	5

The second scenario represents malfunctions on the sensors measuring the process variables. A linear trend is added to the sensor's readings up to a saturation point, which drastically changes the statistical properties of the signal.

The third scenario considers input disturbances: A random perturbation is added every 60 minutes to C_{Ai} and temperatures T_i and T_{ci} (Pilario and Cao, 2018).

3.2 Data preparation

For CDFD, the sampling procedure of Li et al. (2020) is applied. A key assumption is to use the parameter errors to emulate model-process mismatch. This allows, to some extent, to test domain adaptation algorithms when real-world data is not available. We further assess the validity of this assumption by exploring how parameter errors influence the distributional shift phenomenon. The sampling procedure consists of the two steps described below: (i) simulation (source) domain and (ii) physical (target) domain. The CDFD is simulated by adapting the Matlab/Simulink model in Pilario and Cao (2018).

Simulation domain sampling: For each fault in Table 2, the system is simulated 100 times, resulting in a dataset with $13 \times 100 = 1300$ samples. During the simulation, the process variables are tracked through 200 minutes with time-step $\Delta t = 1$ min. Their measurements are assumed to be corrupted with noise $\eta \sim \mathcal{N}(0, 10^{-2})$. This generates a balanced dataset for classification, with 13 classes (12 faults + normal operation). As raw signals, the concatenation of the process variables is considered, hence each sample consists on a vector $\mathbf{x} \in \mathbb{R}^{200 \times 7}$.

Physical domain sampling: Following the procedure of Li et al. (2020), the model-process mismatch is emulated through parameter errors. Let $\Theta = [\theta_1, \cdots, \theta_p] \in \mathbb{R}^p$ be the vector of the $p = 20$ process parameters in Table 1. The first type of parametric error is introduced in 6 parameters, namely: $V, V_c, \Delta H_r$, UA, k_0 and E/R, through a degree of parameter mismatch $\epsilon > 0$. The

Target Domain	1	2	3	4	5	6	
Reaction Order	1.0	1.0	1.0	0.5	1.5	2.0	Score
Degree of Mismatch	10%	15%	20%	15%	15%	15%	
Raw Signals-based SVM							
Baseline	68.654 ± 0.769	64.519 ± 1.231	62.404 ± 1.154	56.923 ± 0.892	46.346 ± 1.709	32.981 ± 0.942	55.304
PCA	69.423 ± 0.490	64.519 ± 1.027	64.615 ± 0.577	57.212 ± 1.290	46.635 ± 0.804	36.058 ± 0.608	56.410
OTDA	**89.423 ± 0.680**	**87.596 ± 0.769**	**80.673 ± 1.532**	72.404 ± 1.380	**85.769 ± 1.201**	77.308 ± 1.113	82.196
Target-Only	77.788 ± 3.209	75.673 ± 1.036	70.096 ± 2.891	73.942 ± 2.875	53.365 ± 2.668	50.769 ± 1.626	66.939
Convolutional Features							
Baseline	80.192 ± 3.001	75.288 ± 2.031	74.135 ± 3.815	62.500 ± 2.544	69.808 ± 5.400	60.288 ± 5.830	70.000
DANN	86.219 ± 2.745	76.594 ± 3.284	75.219 ± 2.331	71.875 ± 3.168	79.969 ± 1.457	69.844 ± 3.205	77.000
OTDA	89.327 ± 0.360	84.519 ± 3.803	77.981 ± 2.810	**77.212 ± 2.288**	84.135 ± 2.995	**83.269 ± 1.113**	**82.740**
Target-Only	95.385 ± 4.485	91.538 ± 3.125	85.385 ± 7.750	86.538 ± 4.213	89.615 ± 3.768	84.615 ± 1.720	89.000

Table 3. Average classification accuracy with confidence intervals over a 5-fold cross-validation.

perturbed parameters are given by $\tilde{\theta}_i = \theta_i + \Delta\theta_i$ where $\Delta\theta_i \sim \mathcal{U}(-\epsilon, \epsilon)$ is uniformly distributed over the interval $[-\epsilon, \epsilon]$. $\Delta\theta_i$ is sampled independently at each simulation. The second kind of mismatch deals with the reaction order N, which is considered over the set $\{0.5, 1.5, 2.0\}$. This yields 6 different target domains, corresponding to $\epsilon = 0.1, 0.15, 0.2$ with $N = 1.0$ fixed, and $\epsilon = 0.15$ fixed, with $N = 0.5, 1.5, 2.0$. The system is simulated 20 times for each fault, yielding $13 \times 20 = 260$ samples for each target domain.

To validate the hypothesis that modelling errors induces distributional shift, we measure the Wasserstein distance between $\hat{P}_S$ and $\hat{P}_T$ for each target domain. We use the Euclidean distance as the ground-cost. Figure 3 shows the Wasserstein distance W_2 as a function of ϵ and N. Source and target domains follow different distributions ($W_2(\hat{P}_S, \hat{P}_T) \neq 0$). Furthermore, W_2 increases with an increasing degree of parameter mismatch ϵ. This indicates that the adaptation problem becomes harder. In addition, the Wasserstein distance is more sensitive to changes in the reaction order. This indicates that this parameter has a deeper impact on the system dynamics and consequently on the signals statistical properties as well.

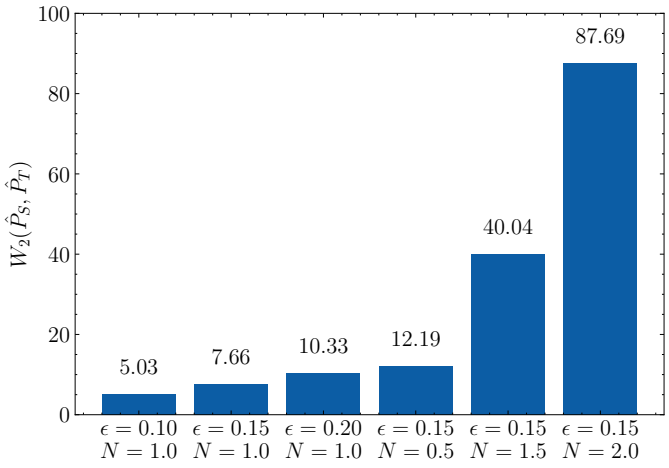

Fig. 3. Wasserstein distance between source and target domain for increasing degrees of parameter mismatch, and different values for the reaction order.

3.3 Comparative Study of Domain Adaptation Algorithms

We explore different methods of domain adaptation for performing CDFD. For this task, we compare how raw signals-based SVM performs in comparison with CNNs when the following adaptation algorithms are used: PCA (Gong et al., 2012), domain adversarial neural networks (DANN) (Ganin et al., 2016) and OTDA (Courty et al., 2017). Note that since DANN was designed for neural networks, it is not applied for the SVM classifier. We compare these algorithms with two standard training situations: The baseline and target-only cases. The baseline corresponds to applying an ill-fitted classifier on new domains. The target-only scenario is an optimistic scenario as it disregards distributional shift. Finally, the score in equation (2) gives an overall comparison across all target domains.

For the OTDA we use the network simplex solver in the optimal transport toolbox in Python (Flamary et al., 2021). As ground-cost, the Euclidean distance is used. Our evaluation is based on a 5-fold cross validation strategy that leaves a partition of the training and test dataset at each iteration. Table 3 shows the results for the mean accuracy with confidence intervals for each target domain.

First, with respect to Table 3 and Figure 3, the Wasserstein distance is negatively correlated with the baseline accuracy. This confirms that the adaptation problems have increasing difficulty. Nonetheless, this is not the only factor determining the algorithms performance, as in the target-only case, target domain 4 ($N = 0.5, \epsilon = 0.15$) poses a classification problem that is more difficult than target domain 5 ($N = 1.5, \epsilon = 0.15$). Thus domain adaptation algorithms tend to perform better in the latter domain than in the former, despite distributional shift.

Second, in overall CNNs have better performance than raw signals. This was expected, since by design this model better captures the data temporal structure. Comparing PCA with DANN, two algorithms that try to build a latent space where features share common characteristics, we note that DANN yields a better performance improvement. This indicates that, even though there is no linear sub-space where features are similar statistically, one can still build a non-linear latent space where domains coincide.

Third, OTDA has the best performance, being slightly superior with CNN features by a tight margin. This highlights the usefulness of the Wasserstein distance for defining a mapping for matching train and test probability distributions. Finally, note that OTDA outperforms other methods when the domain shift is severe (target domains 4 and 6).

4. CONCLUSION

We explored the problem of CDFD characterised by a change in distribution between training and testing data. Our concern is the application of CDFD for adapting fault diagnosis models learned on simulation data to data acquired in real-world conditions (e.g., from sensors). We follow Li et al. (2020) conducting two experiments using a CSTR benchmark process by: (i) verifying that modelling errors induce different data distributions, and (ii) comparing domain adaptation algorithms for CDFD.

The first experiment shows that an increasing level of parameter mismatch yields data following distributions farther away with respect to the Wasserstein distance. Likewise, for reaction order values different than the default ($N = 1$), the same phenomenon happens. This indeed confirms that modeling errors induce distributional shift.

The second experiment shows that the Wasserstein distance is negatively correlated with domain adaptation difficulty. Moreover, OTDA outperforms when compared with feature-based and deep learning-based techniques.

Our work further supports the power of this framework, as previously verified by Courty et al. (2017) in image processing related applications. Furthermore, it shows that OTDA can be successfully applied to CDFD problems.

REFERENCES

Ben-David, S., Blitzer, J., Crammer, K., Pereira, F., et al. (2007). Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19, 137.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8), 1798–1828.

Cheng, C., Zhou, B., Ma, G., Wu, D., and Yuan, Y. (2019). Wasserstein distance based deep adversarial transfer learning for intelligent fault diagnosis. *arXiv preprint arXiv:1903.06753*.

Chiang, L.H., Russell, E.L., and Braatz, R.D. (2000). *Fault detection and diagnosis in industrial systems*. Springer Science & Business Media.

Courty, N., Flamary, R., Tuia, D., and Rakotomamonjy, A. (2017). Optimal transport for domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9), 1853–1865. doi:10.1109/TPAMI.2016.2615921.

Dantzig, G.B., Orden, A., Wolfe, P., et al. (1955). The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*, 5(2), 183–195.

Flamary, R. et al. (2021). Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78), 1–8. URL `http://jmlr.org/papers/v22/20-451.html`.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1), 2096–2030.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 315–323. JMLR Workshop and Conference Proceedings.

Gong, B., Shi, Y., Sha, F., and Grauman, K. (2012). Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2066–2073. IEEE.

He, M. and He, D. (2017). Deep learning based approach for bearing fault diagnosis. *IEEE Transactions on Industry Applications*, 53(3), 3057–3065.

Hoang, D.T. and Kang, H.J. (2019). A survey on deep learning based bearing fault diagnosis. *Neurocomputing*, 335, 327–335.

Isermann, R. (2006). *Fault-diagnosis systems: an introduction from fault detection to fault tolerance*. Springer Science & Business Media.

Isermann, R. and Balle, P. (1997). Trends in the application of model-based fault detection and diagnosis of technical processes. *Control engineering practice*, 5(5), 709–719.

Jiang, L., Ge, Z., and Song, Z. (2017). Semi-supervised fault classification based on dynamic sparse stacked auto-encoders model. *Chemometrics and Intelligent Laboratory Systems*, 168, 72–83.

Li, W., Gu, S., Zhang, X., and Chen, T. (2020). Transfer learning for process fault diagnosis: Knowledge transfer from simulation to physical processes. *Computers & Chemical Engineering*, 106904.

Pan, S.J., Tsang, I.W., Kwok, J.T., and Yang, Q. (2010). Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2), 199–210.

Pan, S.J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10), 1345–1359.

Peyré, G. and Cuturi, M. (2019). Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6), 355–607.

Pilario, K.E.S. and Cao, Y. (2018). Canonical variate dissimilarity analysis for process incipient fault detection. *IEEE Transactions on Industrial Informatics*, 14(12), 5308–5315.

Su, Z., Tang, B., Ma, J., and Deng, L. (2014). Fault diagnosis method based on incremental enhanced supervised locally linear embedding and adaptive nearest neighbor classifier. *Measurement*, 48, 136–148.

Wang, Z., Zhang, Q., Xiong, J., Xiao, M., Sun, G., and He, J. (2017). Fault diagnosis of a rolling bearing using wavelet packet denoising and random forests. *IEEE Sensors Journal*, 17(17), 5581–5588.

Zhang, X., Chen, W., Wang, B., and Chen, X. (2015). Intelligent fault diagnosis of rotating machinery using support vector machine with ant colony algorithm for synchronous feature selection and parameter optimization. *Neurocomputing*, 167, 260–279.

Zheng, H., Wang, R., Yang, Y., Yin, J., Li, Y., Li, Y., and Xu, M. (2019). Cross-domain fault diagnosis using knowledge transfer strategy: a review. *IEEE Access*, 7, 129260–129290.