Heterogeneous transfer learning for highly non-linear regression tasks with application to the hydrotreatment of tire pyrolysis feedstocks

Youba Abed, Julien Jacques, Victor Costa, Benoît Celse, Denis Guillaume, Julian Per Becker

Heterogeneous transfer learning for highly non-linear regression tasks with application to the hydrotreatment of tire pyrolysis feedstocks

Youba Abed[1,2], Julien Jacques[2], Victor Costa[1], Benoît Celse[1], Denis Guillaume[1], Julian Per Becker[1]

[1]IFP Energies nouvelles (IFPEN), Rond-point de l'échangeur de Solaize, France
[2]Université Lumière Lyon 2, Universite Claude Bernard Lyon 1, ERIC, 69007, Lyon, France

Address for correspondence: Youba Abed, IFP Energies nouvelles, Lyon, France.
Email: youba.abed@ifpen.fr

Abstract

Hydrotreatment is a crucial step in removing impurities, such as nitrogen, from feedstocks in order to improve the hydrocracking reaction and avoid early catalyst deactivation. The objective of this work is to predict the nitrogen concentration after the hydrotreatment step under a scarce data regime. In particular, for renewable feedstocks, the available data is limited, whereas rich data sets are often available in the fossil domain, where they can be leveraged to improve the prediction task. This motivates the use of transfer learning, especially heterogeneous transfer learning, since the feature spaces of the two domains differ. Three new heterogeneous transfer learning methods for regression tasks have been developed, achieving substantially lower prediction errors than classical methods both with and without transfer learning on simulated and real data sets.

Keywords: Heterogeneous Transfer Learning, Regression Tasks, Early Layer Fine-Tuning, Kinetic Models, Hydrotreatment.

1 Introduction

1.1 Industrial challenge

In the refining industry, crude oil is transformed into valuable products, primarily fuels for transportation, such as gasoline, aviation kerosene, and diesel. Additionally, high-purity chemicals like propylene, butadiene, and aromatics are produced for plastic manufacturing. The transformation process begins with an atmospheric distillation, which separates crude oil into different fractions based on their boiling points. This step, carried out at high temperature and atmospheric pressure, yields light fractions (light gases such as propane and butane, naphtha), middle distillates (kerosene, diesel/gasoil), and heavy fractions (atmospheric gasoil - AGO, residual fuel oil). However, the residual fuel still contains high boiling point hydrocarbons that cannot be further distilled under atmospheric conditions. To process these heavy residuals, a vacuum distillation is used. By operating under reduced pressure, this method lowers the boiling points, allowing heavier fractions to be distilled. This step produces vacuum gas oil (VGO), vacuum residue, and other fractions.

One of the most common refinery processes to produce fuels are the Hydrocracking (HCK) units (Speight 2011), which convert heavy feedstocks like the VGO into valuable fractions, such as naphta, jet fuel, and diesel. These units are composed of two sections: purification and conversion. The purification section, known as hydrotreatment, removes impurities like sulfur and nitrogen. The conversion section, transforms the purified heavy high boiling-point molecules into lighter, more valuable molecules. The underlying reactions in both sections take place in chemical reactors under high hydrogen pressure in the presence of catalysts, which lower temperature and pressure requirements while improving reaction selectivity. Because the catalyst in the HCK section is highly sensitive to impurities, the hydrotreatment section is essential to protect against early deactivation and ensure an efficient conversion process. The main reactions that are promoted in the first section

are the HydroDeNitration (HDN) and the HydroDeSulfurization (HDS) reactions.

The work of this paper focus on HDN, more precisely the prediction of the nitrogen concentration after the hydrotreating stage. This is crucial for improving global efficiency; it allows operators to tune operating conditions to achieve the desired low concentrations and helps to design new installations of these units. The most widely used models for this prediction are kinetic models (Oliveira et al. 2016; Cao et al. 2020; Becker et al. 2024), which describe the rates of chemical reactions and their dependencies on factors such as composition, temperature, and pressure. These models which are based on mass balance equations coupled with reaction rate expressions, involve solving one or more ordinary differential equations (ODEs). These equations combine features (data) and parameters: the features consist of descriptors selected by experts to explain the evolution of the nitrogen concentration, while the parameters are adjusted to fit the observed outputs. Fitting the parameters of these equations requires a substantial amount of expensive experimental data, with each data point being costly to obtain. Furthermore, every time a new catalyst is introduced (typically every two to three years), the parameters must be re-fitted, resulting in the development of new models. This process is both time-consuming and resource-intensive.

The challenge becomes even greater when dealing with alternative renewable feedstocks, such as bio-based products or blends of fossil and bio-derived materials. In such cases, existing models are often no longer valid and must either be re-parameterized and structurally modified by adding or removing features and/or parameters. In the latter case, the appropriate model structure is often unknown, as these feedstocks are relatively new and remain under active investigation. These feedstocks are hereafter referred to as NET (New Environment Technologies).

This paper investigates the development of predictive models for NET feedstocks using as little data as possible, under two scenarios: when the NET kinetic model structure is known, and when it is unknown. To support this, we aim to leverage knowledge from historical data sets obtained using fossil feed, in order to improve prediction for these new NET feedstocks. One of the advantages of this approach is that the fundamental chemical mechanisms governing HDN reactions generally remain consistent, even when transitioning from fossil to renewable feedstocks. This shared foundation motivates the application of transfer learning to the regression problem.

1.2 Transfer Learning

Before introducing the transfer learning concept, some key notions are defined below:

Definition 1 (Domain): A domain $\mathcal{D}$ consists of two components: a feature space $\mathcal{X}$ and a marginal probability distribution $\mathbb{P}(\boldsymbol{x})$ where $\boldsymbol{x} = \{x_1, \cdots, x_n\} \in \mathcal{X}$. In other words, $\mathcal{D} = \{\mathcal{X}, \mathbb{P}(\boldsymbol{x})\}$.

Definition 2 (Task): A task $\mathcal{T}$ (denoted by $\mathcal{T} = \{\mathcal{Y}, f\}$) involves an output space $\mathcal{Y}$ and an objective prediction function $f(\cdot)$ which is not observed but can be learned from the training data consisting of pairs $\{\boldsymbol{x}_i, y_i\}$, where $\boldsymbol{x}_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. The function $f(\cdot)$ is used to predict the corresponding output $y_i = f(\boldsymbol{x}_i)$.

Given a source domain and task ($\mathcal{D}_\text{S}$, $\mathcal{T}_\text{S}$), a target domain and task ($\mathcal{D}_\text{T}$, $\mathcal{T}_\text{T}$), transfer learning (Yang et al. 2020) aims to leverage knowledge from source domain and task to improve the objective prediction function of the target task. Based on the consistency between the source and target feature spaces, transfer learning can be categorized into homogeneous and heterogeneous transfer learning. Homogeneous transfer learning, also known as Domain Adaptation (DA) (Zhuang et al. 2021), refers to the case where the source and target domains are represented by the same feature space ($\mathcal{X}_\text{S} = \mathcal{X}_\text{T}$) and output space ($\mathcal{Y}_\text{S} = \mathcal{Y}_\text{T}$). In contrast, Heterogeneous Transfer Learning (HTL) also known as Heterogeneous Domain Adaptation (HDA) (Day et al. 2017) refer to the case where either the feature spaces are non-equivalent ($\mathcal{X}_\text{S} \neq \mathcal{X}_\text{T}$) and/or the output spaces are non-equivalent ($\mathcal{Y}_\text{S} \neq \mathcal{Y}_\text{T}$).

In our regression setting, the output spaces are identical (nitrogen concentration), whereas the feature spaces differ, as the NTE may include either additional or fewer features. Consequently, the problem is formulated as a heterogeneous transfer learning (HTL) task for regression.

1.3 Related work

In the literature, there are mainly two approaches to solve HTL problems (Bao et al. 2024): data-based and model-based. The data-based methods focus on transferring the original data or their transformed features to

the target domain. This process allows the target model to be trained with enriched data, thus expanding the data available within the target domain. The data-based approach comprises multiple subcategories, among which feature mapping (symmetric (Wu et al. 2021; Li et al. 2019a; Shi et al. 2013) and asymmetric (Yao et al. 2019; Zhou et al. 2019)) and feature augmentation (Li et al. 2020; Yan et al. 2017) are the most prominent.

In contrast, model-based approaches focus on transferring or adapting a model or its underlying knowledge from a source domain to improve prediction in a different target domain. This often involves training a model on the source data to learn optimal parameters, which are then transferred to the target model using techniques such as parameter regularization (Shu et al. 2015; Ye et al. 2020; Silvestrin et al. 2023) or parameter tuning (Lee et al. 2023; Xiao et al. 2023). While many of these methods rely on parametric models, some approaches also transfer knowledge from non-parametric models (Wu et al. 2020).

Before moving further, it is worth noting that the vast majority of HTL methods in the literature were designed for classification problems, with very few addressing the challenge of regression problems. These regression methods are not suitable to our problem. As an example, the method proposed by Silvestrin et al. 2023 considers a case where both the source and target models are linear, which does not apply to our highly non-linear setting. Another HTL method for regression is from Shi et al. 2013. First, the authors construct a common feature space using a spectral embedding-based method. In this common feature space, source samples that are close to the target samples are selected using a KL divergence-based framework and then construct new outputs for those selected samples (Shi et al. 2010). Finally, a target model is trained using the projected target samples with their original outputs and the selected projected source samples with their constructed outputs. However, the feature transformation is done independently from the target (source) outputs. This decoupled approach may fail to preserve the mathematical relationships in ODE-based regression where outputs emerge from integrating complex ODEs. Wu et al. 2020 proposed another HTL regression method using multiple kernel learning. Their boosting approach selects between source and target models at each iteration, which helps avoid negative transfer. Negative transfer occurs where transfer learning methods undesirably reduce learning performance compared to non-transfer learning (Zhang et al. 2023). The method was primarily developed for scenarios involving multiple source domains. In the present work, only one source domain is available; therefore, existing methods must be adapted to this more constrained setting. Model-based approaches are adopted for this purpose.

On the subject of model-based, particularly in parameter regularization, the target model parameters are updated while staying close to the source model parameters using a regularization function. These methods can be categorized into three subcategories: Bayesian transfer learning (Suder et al. 2023), Physics-informed machine learning (PIML) (Bradley et al. 2022), and other approaches such as optimal transport (Ye et al. 2020) or direct regularization (Shu et al. 2015). To the best of our knowledge, no existing HTL methods in the literature fall into the first two subcategories. For Bayesian transfer learning, the idea is to guide the learning process in the target domain using prior knowledge from the source domain. These methods operate in the parametric space, meaning they focus on learning parameters. One can use a prior distribution that encourage the target model parameters to stay close to those of the source. Iapteff et al. 2023 proposed a Bayesian transfer learning method for DA problem.

In contrast, parameter tuning methods (often applied to neural networks) focus on refining a pre-trained source model to perform well on the target task. This typically involves two key stages: an initial training phase and a subsequent adaptation phase, also known as fine-tuning. Several fine-tuning strategies exist in the literature. One common approach is full fine-tuning, where the weights of all layers in the source neural network are updated. Another approach is partial fine-tuning, where only a subset of the source network's layers is updated. In partial fine-tuning, last layers are often the ones fine-tuned (Basha et al. 2021). However, some works suggest that fine-tuning earlier layers can be beneficial, especially for input-level shifts, though this has been primarily demonstrated for convolutional and transformer architectures (Lee et al. 2023). When the target domain differs in input dimensionality, containing either more or fewer features, new input weight vectors can be added or removed accordingly, and the corresponding weights trained to address the heterogeneity.

1.4 Paper organization

This paper is organized as follows. The data sets and the case study kinetic models (ODEs) are described in Section 2. Three HTL frameworks for regression tasks are proposed in Section 3. The first framework concerns the case where the target kinetic model structure is supposed to be known. The other two frameworks address the case where the target kinetic model is supposed to be unknown. The experimentation is described in Section 4, which includes the data sets simulation algorithm, the real data sets, the comparison methodology, and hyperparameter tuning with training strategies. The results for both simulated and real data sets are presented in Section 5. Finally, Section 6 concludes the paper.

2 Data sets and models

2.1 Source domain

The source (fossil) data set contains 7 features selected by experts used to predict the output nitrogen N. All these features are quantitative, and they are divided into two types:

- Features that describe the operating conditions (OPC), represented by T, ppH_2, and $LHSV$.
- Features that describe the feedstock properties (FP), represented by TMP, N_0, S_0, and Res_0.

Table 1 summarizes these features. The OPC tend to have more variability compared to the FP in pilot plant

Feature	Description	Type of Feature
$LHSV$	Liquid Hourly Space Velocity: the ratio of liquid volume flow per hour to reactor volume (in h^{-1}). It is the inverse of the residence time.	Operating Conditions (OPC)
T	Temperature of the hydrotreating reactor (in Kelvin (K)).	
ppH_2	Hydrogen partial pressure (in bar).	
TMP	Weighted average of the simulated distillation (in Kelvin (K)).	Feedstock Properties (FP)
N_0	Nitrogen content in feedstock (in ppm).	
S_0	Sulfur content in feedstock (in mass percent).	
Res_0	Resins content in feedstock (in mass percent).	
N (to be predicted)	Nitrogen content after hydrotreating (in ppm). The variable of interest that we want to model using the other features.	Output Nitrogen

Table 1: Source features description.

data sets, as they can be adjusted by operators to achieve the desired performance (e.g., low output nitrogen). One of the features that describe the FP is the nitrogen content in the feedstock (N_0). This nitrogen content evolves over time, and the goal is to predict the remaining nitrogen content (N) after the hydrotreatment process given the OPC and the FP. The relation between these features and the output nitrogen is given by the following ODE-based kinetic modeling (Cao et al. 2020; Becker et al. 2024):

$$\frac{dy}{dt} = -k_0 \frac{exp\left(-\frac{E_a}{R_g}\left(\frac{1}{T}-\frac{1}{T_{ref}}\right)\right)\left(\frac{ppH_2}{ppH_{2,ref}}\right)^m y^n}{(1+A_0\times Res_0)\left(1+\frac{C_0\times N_0}{1+S_0}\right)}\times \left(1-u\cdot exp\left(-\frac{b}{R_g}\left(\frac{1}{T}-\frac{1}{T_{ref}}\right)\right)\left(\frac{ppH_2}{ppH_{2,ref}}\right)^\alpha\left(\frac{TMP}{TMP_{ref}}\right)^v y^l\right). \tag{1}$$

Here, y denotes nitrogen (y is used instead of N to remain consistent with standard machine learning notation). $\boldsymbol{\theta}_S = (k_0, E_a, m, n, \alpha, b, A_0, C_0, u, l, v)^\top \in \mathbb{R}^{11}$ are the kinetic parameters to optimize using the source data set, in order to fit the observed outputs. The boundaries of these parameters are constrained to maintain physical validity but are not disclosed due to confidentiality restrictions. Additionally, the parameters are defined on different scales, which may affect their relative influence and interpretation. $\{R_g, T_{ref}, ppH_{2,ref}, TMP_{ref}\}$

Symbol	Value
R_g	1.98 cal/mol/K
T_{ref}	648.15 K
$ppH_{2,ref}$	32.50 bar
TMP_{ref}	643.15 K

Table 2: Reference values of the source kinetic model.

are fixed reference values (Becker et al. 2024) as presented in Table 2. The second term (below the first term in the equation) is the return term to account for thermodynamic equilibrium limitations. This reversible formulation allows the model to capture thermodynamic limits that can be observed when the system approaches equilibrium. In our case, this term remains non-negative.

The output nitrogen is calculated by integrating the ODE between 0 and $\frac{1}{LHSV}$:

$$y = \int_0^{\frac{1}{LHSV}} \left(\frac{dy}{dt}\right) dt. \tag{2}$$

2.2 Target domain

The target (NTE) data set features are similar to the source data features, but include an additional quantitative feature $\boldsymbol{Tire}$ in the denominator of the first fraction, which refers to the tire pyrolysis oil content in the feedstock (a property of the feedstock (FP)). The case study for the target ODE is defined as follows:

$$\begin{aligned}\frac{dy}{dt} = &- k_0 \frac{exp\left(-\frac{E_a}{R_g}\left(\frac{1}{T}-\frac{1}{T_{ref}}\right)\right)\left(\frac{ppH_2}{ppH_{2,ref}}\right)^m y^n}{(1+A_0\times Res_0)\left(1+\frac{C_0\times N_0}{1+S_0}\right)(1+\boldsymbol{p}\times\boldsymbol{Tire})}\times \\ &\left(1-u\cdot exp\left(-\frac{b}{R_g}\left(\frac{1}{T}-\frac{1}{T_{ref}}\right)\right)\left(\frac{ppH_2}{ppH_{2,ref}}\right)^\alpha\left(\frac{TMP}{TMP_{ref}}\right)^v y^l\right).\end{aligned} \tag{3}$$

As observed, the new input feature $\boldsymbol{Tire}$ is related to a new parameter $\boldsymbol{p}$ in the denominator when compared to Equation (1). The target parameters are then: $\boldsymbol{\theta}_{\mathrm{T}} = (k_0, E_a, m, n, \alpha, b, A_0, C_0, u, l, v, \boldsymbol{p})^\top \in \mathbb{R}^{12}$. The common parameters between the two ODEs do not have the same values, but their boundaries are the same. For p, the boundary is also fixed. The reference values are also the same (Table 2).

The same type of ODE is retained because the fundamental hydrocarbon chemistry remains unchanged. The introduction of tire material primarily affects the apparent reaction rates, for instance through inhibitory effects, rather than modifying the underlying reaction mechanisms.

To illustrate the inhibition effect of the tire through simulation, Figure 1 shows the comparison between the evolution of the output nitrogen (N) with increasing temperature on both source and target, while keeping the common features and parameters fixed (same values).

Our goal is to build a regression model to predict y in the target domain using as little data as possible:

$$y_i = f(\boldsymbol{x}_i) + \epsilon_i, \tag{4}$$

where the structure of f may either be known or considered as a black-box function. Knowledge of the structure of f refers to knowledge of the target ODE structure, since integrating it yields f. In this case, we only need to estimate the ODE parameters.

3 Proposed HTL frameworks

We propose three HTL frameworks. The first, named Heterogeneous Bayesian Transfer Learning (HBTL), is inspired by Iapteff et al. 2023 and concerns the case where the target ODE structure is known. The other two frameworks, which do not require this knowledge, are named Deep Heterogeneous Transfer Learning with Fine-Tuning (DHTLFT) and Deep Heterogeneous Transfer Learning with Multi-Task Learning (DHTLM), respectively. They are inspired by the works of Lee et al. 2023; Shu et al. 2015 on classification tasks.

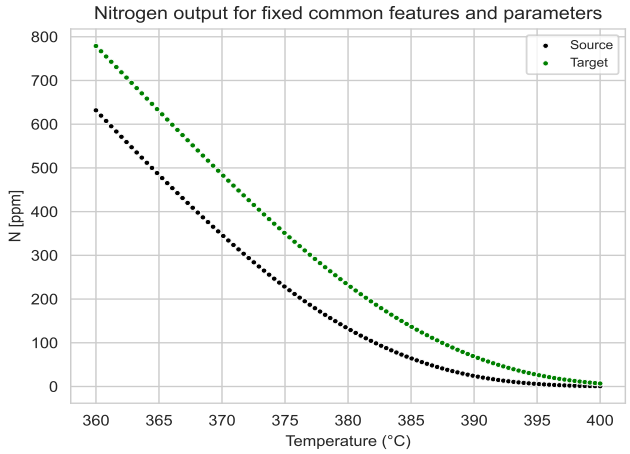

Figure 1: Simulation example of the inhibition effect of tire on nitrogen content evolution with increasing temperature. The black plot represents the source domain, and the green plot represents the target domain, with common features and parameters fixed at the same values.

3.1 HTL with knowledge of the target ODE structure

This section presents the HTL approach in the context of knowing the target ODE structure. The main challenge is to fit the target parameters, as fitting them will enable the prediction of the output nitrogen content y, which can be calculated by solving the Equation (3) with a numerical solver such as LSODE (Radhakrishnan et al. 1993).

We first outline the formulation of the modeling problem, before presenting our HBTL framework. The HBTL is based on the Metropolis-Hastings within Gibbs (MHwG) algorithm (Tierney 1994).

3.1.1 Modeling problem

The objective is to optimize parameters $\boldsymbol{\theta} \in \mathbb{R}^K$ such that the following quadratic weighted loss:

$$\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{y_i} \tag{5}$$

is minimized, where K is the number of parameters to optimize and n is the size of the training data set. This loss help to prevent larger output nitrogen values from disproportionately influencing the model compared to smaller ones during parameter optimization.

Minimizing this loss is equivalent to estimating the following model (Iapteff et al. 2023):

$$y_i = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + \epsilon_i, \tag{6}$$

$$\epsilon_i \sim \mathcal{N}(0, \sigma_i^2),$$

$$\sigma_i^2 = \sigma \times y_i, \tag{7}$$

where σ is a fixed parameter.

Given a prior $\pi(\boldsymbol{\theta})$ on $\boldsymbol{\theta}$, the posterior distribution is given as follows:

$$\pi(\boldsymbol{\theta} \mid \boldsymbol{y}) = \frac{\pi(\boldsymbol{\theta}) \times \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{y}, X)}{\int \pi(\boldsymbol{\theta}) \times \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{y}, X) \mathrm{d}\boldsymbol{\theta}} = \frac{g(\boldsymbol{\theta})}{\int g(\boldsymbol{\theta}) d\boldsymbol{\theta}} \tag{8}$$

where $\mathcal{L}$ is the likelihood function.

3.1.2 HBTL framework

Since the posterior distribution is difficult to calculate analytically (due to the high-dimensional complex integral in the denominator), the MHwG algorithm is employed to sample from the posterior distribution using only

Algorithm 1 Metropolis-Hastings within Gibbs (MHwG)

Input: Data set $\{X, \boldsymbol{y}\}$; incomplete posterior distribution $g(\boldsymbol{\theta})$; proposal distributions $q_i(\cdot \mid \cdot)$ for $i = 1, 2, \ldots, K$; number of iteration T.
Initialization: Choose $\boldsymbol{\theta}_{(0)} = (\theta^1_{(0)}, \ldots, \theta^K_{(0)})$.
for $t = 1$ to T **do**
  for $i = 1$ to K **do**
    Sample a proposal $\theta^i_* \sim q_i(\theta^i_* \mid \theta^i_{(t-1)})$.
    The proposal parameter vector: $\boldsymbol{\theta}_* \leftarrow (\theta^0_{(t-1)}, \cdots, \theta^i_*, \cdots, \theta^K_{t-1})$
    Compute the Metropolis-Hastings acceptance ratio:

$$\tau = \frac{g(\boldsymbol{\theta}_*)\, q_i(\theta^i_{(t-1)} \mid \theta^i_*)}{g(\boldsymbol{\theta}_{t-1})\, q_i(\theta^i_* \mid \theta^i_{(t-1)})}.$$

    Draw $w \sim \text{Uniform}(0, 1)$.
    if $w \leq \tau$ **then**
      Accept: $\theta^i_{(t)} \leftarrow \theta^i_*$.
    else
      Reject: $\theta^i_{(t)} \leftarrow \theta^i_{(t-1)}$.
    end if
    $\boldsymbol{\theta}_{(t-1)} \leftarrow (\theta^1_{(t)}, \ldots, \theta^i_{(t)}, \ldots, \theta^K_{(t-1)})$
  end for
  Update the parameter vector of iteration (t): $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{(t-1)}$
end for
Return: Samples $\{\theta_{(t)}\}_{t=1}^T$.

$g(\boldsymbol{\theta})$. Algorithm 1 summarize the MHwG. For each parameter, a normal distribution is used as the proposal distribution for sampling. Each proposal distributions $q_i(\cdot \mid \cdot)$ $(i = 1, 2, \ldots, K)$ is centered on the previous draw and with a certain standard deviation s^i that can be fixed. However, as well known, the acceptance ratio should be neither too small nor too high (Gelman et al. 1996). To achieve this, in our experiments, the standard deviations of the proposal distributions are initially set to 10% of the initial parameter values and are then adjusted every J iterations according to the following rule:

- If the acceptance rate percentage over the last J iterations $\tau^i_J\%$ is higher than 70%, s^i is increased, since larger standard deviations tend to reduce the acceptance rate.
- If $\tau^i_J\% \leq 30\%$, s^i is decreased, since smaller standard deviations tend to increase the acceptance rate.
- If $\tau^i_J\% \in$ [30% and 70%[, s^i remains unchanged.

After the burn-in period, the estimates $\hat{\boldsymbol{\theta}}$ and $Var(\hat{\boldsymbol{\theta}})$ are obtained from the mean and covariance matrix of the remaining samples, respectively.

Given a source data set, the source parameters mean $\hat{\boldsymbol{\theta}}_{\text{S}}$ and covariance matrix $Var(\hat{\boldsymbol{\theta}}_{\text{S}})$ are estimated using the MHwG algorithm with a weakly informative prior. σ_{S} (Eq. (7)) was fitted with the source parameters: $\boldsymbol{\theta}'_{\text{S}} = (\boldsymbol{\theta}_{\text{S}}, \sigma_{\text{S}}) \in \mathbb{R}^{12}$. The prior was chosen to be a Gaussian distribution with a large variance: $\pi(\boldsymbol{\theta}'_{\text{S}}) \sim \mathcal{N}(0_{\mathbb{R}^{12}}, m \times \mathbb{I}_{12})$, where m is a large value, typically 10^{100}. The parameters drawn outside their boundary were rejected (the parameter boundaries are assumed to be known as mentioned earlier). Additionally, parameter sets that result in a negative thermodynamic return term were also rejected.

Note that as normal distributions are symmetric, $q_i(\theta^i_{(t-1)} \mid \theta^i_*)$ and $q_i(\theta^i_* \mid \theta^i_{(t-1)})$ simplify in the acceptance ration τ (Algorithm 1).

Once the source parameter mean and covariance matrix are estimated, and given a target training subset, our HBTL estimates the target parameters by following similar steps. To incorporates the knowledge we have

from the source domain model parameters, we suppose $\sigma_t = \hat{\sigma}_S$ with $\hat{\sigma}_S$ the estimated mean of σ_S, and instead of a weakly informative prior we incorporates the following Gaussian prior for the target parameters:

$$\pi(\boldsymbol{\theta}_T) \sim \mathcal{N}\left(\begin{pmatrix}\hat{\boldsymbol{\theta}}_S \\ 0\end{pmatrix}, g \times \begin{pmatrix}\text{Var}(\hat{\boldsymbol{\theta}}_S) & \mathbf{0}_{\mathbb{R}_{11}} \\ \mathbf{0}_{\mathbb{R}_{11}}^\top & m\end{pmatrix}\right), \tag{9}$$

where g is a positive hyperparameter that controls the importance of this prior as in the Zellner's prior or g-prior (Zellner 1986), and m a large value (10^{100}) reflecting our lack of information about the parameter p. As g increases, the prior tends to be weakly informative, and if g decreases to 0, the shared target parameters tend to $\hat{\boldsymbol{\theta}}_S$ and p (target specific parameter) to 0.

$\boldsymbol{\theta}_{T_{(0)}} = (\hat{\boldsymbol{\theta}}_S, p_0)$ with p_0 sampled randomly from a uniform distribution within p boundaries.

As the source parameters do not share the same scale, $\text{Var}(\hat{\boldsymbol{\theta}}_S)$ is poorly scaled (variances with different magnitudes), and consequently the covariance matrix of the prior (9) is also poorly scaled, which can lead to numerical issues when inverting it during the evaluation of the prior. To address this, the prior is constructed and evaluated using scaled sampled parameters. The scaling factors are defined as the estimated standard deviations of the source parameters, i.e., $\sqrt{\text{Var}(\hat{\boldsymbol{\theta}}_S)_{ii}}$ for $i = 1, \ldots, 11$. These parameters represent those shared between the two domains. No scaling is applied to p, as its prior covariance with the other parameters is set to zero, and it is not shared between the domains. This scaling standardizes the diagonal entries of the covariance matrix of the shared parameters across domains in prior (9) to one, which helps prevent numerical issues during inversion. Importantly, this transformation does not affect the values taken by the acceptance ratio in the MHwG algorithm, since the scaling factors cancel out in the acceptance ratio.

Often, the ODE structure of NET feedstocks is unknown or the known structure is imperfect. It is important then to develop HTL methods that do not rely on it. For this reason, we propose two HTL methods that do not require this knowledge.

3.2 HTL without knowledge of the target ODE structure

In this section, two HTL methods that do not require prior knowledge of the target ODE structure are proposed: DHTLFT and DHTLM. The first method relies on fine-tuning the early layers of the neural network, while the second is inspired by multi-task learning (MTL) Caruana 1997, in which the source and target domains are trained simultaneously.

Both methods use the multi-layer perceptron (MLP) (Rumelhart et al. 1986). The ReLU activation function (Agarap 2019) is used in the hidden layers, as well as in the output layer (since nitrogen concentration is always positive).

3.2.1 DHTLFT framework

The first HTL approach proposed in this context is DHTLFT. In this method, two MLPs are used for the source and target cases, where each MLP contains N hidden layers with n^i neurons in the i-th layer. The MLP of the source is trained separately on the source data with random initialization using the source regression loss ℓ_S given by Equation (5). Then the target MLP is reinitialized with the pre-trained source MLP except for the weights associated to the input feature $\boldsymbol{Tire}$, which are randomly initialized. Finally, the weights of the first $N_1 \geq 1$ hidden layers of the target MLP are fine-tuned, while the weights of the last $N_2 = N - N_1$ hidden layers and the output layer are frozen. The fine-tuning is performed using a target training subset and the target regression loss ℓ_T (Equation (5)). We keep $N_1 \geq 1$ so at least the input weight are trained to help avoiding the negative transfer.

Figure 2 summarizes the method. N, N_1 (or N_2), n^i for $i \in \{1, \ldots N\}$ are hyperparameters of this method that need to be determined.

The earlier layers are fine-tuned since the main difference (shift) across domains is characterized by the additional input feature ($\boldsymbol{Tire}$ content), and consequently the earlier layers must be domain-specific to capture this new information without excessive influence from the source domain. This contrasts with many existing works, especially for image classification, that fine-tune the last layers while freezing the earlier convolutional

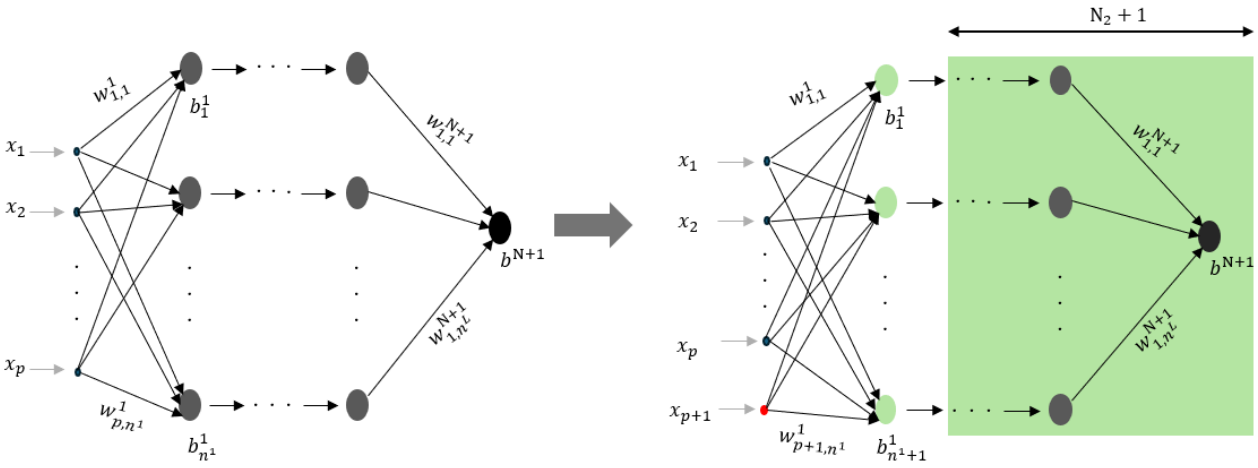

Figure 2: Architecture of DHTLFT. The source MLP (left) is trained separately from the target MLP (right). The red node in the target MLP represents the specific target feature ***Tire***, which is referred to here as x_{p+1}. The weights of the first N_1 hidden layers of the target MLP are fine-tuned using the target training subset, while the weights of the last N_2 hidden layers and the output layer are frozen. The green band represent the freezing. In this example, only one feature (x_{p+1}) is specific to the target domain, but the method can also be applied when the target contains more or fewer features than the source domain.

layers, as those earlier layers learn general low-level features (edges, textures) that are transferable across domains. However, in our case, the source domain has 7 input features while the target domain has 8 features, with the extra feature being critical for the target task. Since the additional input feature introduces a structural change rather than a domain shift in learned representations, the input layer and early hidden layers must adapt to process this additional dimension, justifying our decision to fine-tune the first layers rather than the last ones.

Directly forcing some weights of the target MLP to be equal to the source MLP can be harmful, especially with heterogeneous domains. To implement a softer constraint, we propose DHTLM in the following section.

3.2.2 DHTLM framework

The method proposed by Shu et al. 2015, which was designed for a classification problem was adapted to the regression context. By modifying both the architecture and the loss functions, we propose DHTLM. Moreover, the source domain is trained simultaneously with the target domain which was not the case of the method in Shu et al. 2015, but rather used to propagate source labels to the target domain through a translator function.

The proposed DHTLM architecture consists of two separate MLPs, one for each domain (similar to DHTLFT), with identical structures: N hidden layers and n^i neurons in the i-th layer. Each MLP has its own input and output corresponding to its respective domain (source or target).

We propose the following objective function to be minimized:

$$\mathcal{L} = \ell_{\mathrm{S}} + \ell_{\mathrm{T}} + \frac{\gamma}{2}\Omega \tag{10}$$

where ℓ_{S} and ℓ_{T} are the regression losses for the source and target domains respectively, given by Equation (5). Ω is a regularization term that links the MLPs across domains by forcing the last N_2 hidden layers' weights and output weights for the two MLPs to be close to each other. Ω is given as follows:

$$\Omega = \sum_{i=\mathrm{N}-\mathrm{N}_2+1}^{\mathrm{N}+1} \left(\| W_{\mathrm{T}}^{(i)} - W_{\mathrm{S}}^{(i)} \|_F^2 + \| \boldsymbol{b}_{\mathrm{T}}^{(i)} - \boldsymbol{b}_{\mathrm{S}}^{(i)} \|_2^2 \right) \tag{11}$$

where $W_{\mathrm{S}}^{(i)}$, $\boldsymbol{b}_{\mathrm{S}}^{(i)}$, $W_{\mathrm{T}}^{(i)}$, $\boldsymbol{b}_{\mathrm{T}}^{(i)}$ denote the weight matrices and bias vectors of the i-th layer for the source and target domains, respectively. γ is a hyperparameter that controls the importance of this term, and it plays the

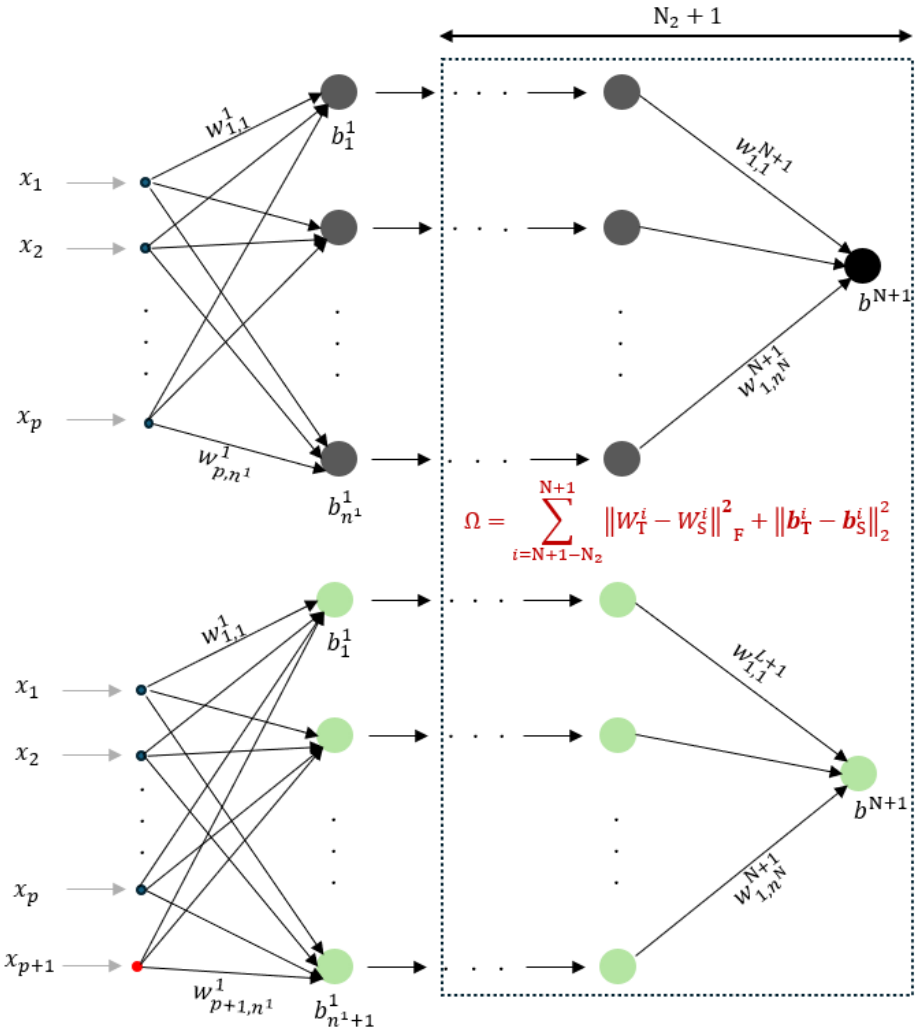

Figure 3: Architecture of DHTLM. The source MLP (above) is trained simultaneously with the target MLP (below). Ω regularizes the weights of the last N_2 hidden layers and the output weights across domains. The weights of the first $N_1 = N - N_2$ hidden layers are kept domain-specific to help avoid negative transfer, as in DHTLFT. Here, only one feature (x_{p+1}) is specific to the target domain, but the method can also be applied when the target contains more or fewer features than the source domain.

inverse role of g in HBTL: as it increases, the target model moves toward the source model, and vice versa. This regularization term serves to transfer knowledge from the source MLP weights to the target MLP weights.

Figure 3 shows the DHTLM architecture. We set $N_2 < N$, keeping the first $N_1 = N - N_2 \geq 1$ layers domain-specific (to help avoid negative transfer). The first N_1 target layers remain unregularized and are fine-tuned using only the target regression loss ℓ_T while freezing the source network and target regularized layers. This fine-tuning alternates with minimizing the objective function (10). Algorithm 2 summarize the optimization of the method. Both MLPs initialize from a pre-trained source model (trained with random initialization on source data) except target input weights for the additional feature x_{p+1} which are randomly initialized.

The hyperparameters of this method that need to be determined are: N, N_1 (or N_2), n^i for $i \in \{1, \ldots, N\}$, and γ.

The framework of DHTLM is very flexible: it allows the inclusion of any specific constraint (such as a physical constraint, e.g. PIML (Bradley et al. 2022)) in the target MLP, since it is not merged with the source MLP, which facilitates the optimization process.

Algorithm 2 Deep Heterogeneous Transfer Learning with Multi-Task Learning (DHTLM)

Input: Source train data set $\{X_{\mathrm{S}}, \boldsymbol{y}_{\mathrm{S}}\}$; Target train data set $\{X_{\mathrm{T}}, \boldsymbol{y}_{\mathrm{T}}\}$; Hyperparameters ($\gamma$, number of iterations n_{iter}, N_2 layers for the regularization term Ω), neurons per layer (n^i, $i \in \{1, \cdots, \mathrm{N}\}$)).
Weights Initialization: Pre-train the source MLP with random initialization using the regression loss ℓ_{S}; Reinitialize the target MLPs with the pre-trained source MLP weights, except for the weights associated with the additional input feature x_{p+1}, which are randomly initialized.
for $t = 1$ to n_{epoch} **do**
  for $i = 1$ to $\mathrm{N} + 1$ **do**
    Update $W_{\mathrm{S}}^{(i)}$, $W_{\mathrm{T}}^{(i)}$ $\boldsymbol{b}_{\mathrm{S}}^{(i)}$, $\boldsymbol{b}_{\mathrm{T}}^{(i)}$, using gradient descent to minimize the objective function $\mathcal{L}$.
  end for
  for $i = 1$ to N_1 **do**
    Fine-tune $W_{\mathrm{T}}^{(i)}$ and $\boldsymbol{b}_{\mathrm{T}}^{(i)}$ using the target regression loss ℓ_{T}.
  end for
  if early stopping **then**
    Break
  end if
end for
Return: $\{ W_{\mathrm{T}}^{(i)}, \boldsymbol{b}_{\mathrm{T}}^{(i)} \}_{i=1}^{\mathrm{N}+1}$

4 Experiments settings

This section presents the materials used to evaluate the proposed methods. The methods were applied to both simulated and real data sets. First, we describe the data simulation algorithm. Second, we introduce briefly the real data sets. Third, we introduce the comparison methodology. Finally, we discuss the training and hyperparameters tuning.

4.1 Simulated data sets

The algorithm to simulate the data sets is based on the Kennard-Stone algorithm (KS) (Kennard et al. 1969). Given an initial cloud of points, the KS algorithm is initialized with a training subset composed of the two most distant points, and a test subset composed of the remaining points. At each iteration, one point $\boldsymbol{x}_i$ is moved from the test subset to the training subset, following the criterion:

$$\boldsymbol{x}_i = \arg \max_{\boldsymbol{x}_i \in \text{Test}} \left(\min_{\boldsymbol{x}_{i'} \in \text{Training}} \text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) \right) \tag{12}$$

until the training subset reaches a predefined size. This approach ensures a representative coverage over the feature space for the training set. The training set is the subset selected by the algorithm.

We generate multiple target data sets with different samples in order to robustly evaluate the performance of our proposed methods under diverse data configurations, and only one source data set is generated.

Each data set contains a specified number of total points (n) with a number of distinct Feedstock Properties or FPs (n_{FP}), where each FP represents a unique combination of feedstock property variables. The number of points per FP is denoted as n_{PFP} (n_{PFP} can be derived when fixing n and n_{FP}), representing all observations that share the same fixed FP combination. The values of n, n_{FP}, and n_{PFP} are defined prior to the simulation.

In order to run the simulation, we must first obtain the kinetic parameters to compute the outputs. The source ODE's parameters $\boldsymbol{\theta}_{\mathrm{S}}$ were fitted from pilot plant examples. The target parameters shared with the source were obtained by increasing the source parameters by 20% of their respective value: $\theta_{\mathrm{T}}^i = 1.2 \times \theta_{\mathrm{S}}^i$ for $i = 1, 2, \ldots, 11$. For the parameter p, it was assigned the same value as A_0 of the target to ensure it does not have a smaller influence than A_0. This choice of adding 20% to the source parameters aims to create a target domain that is different but close to the source domain. A 20% increase is not negligible, especially given that the parameters are on different scales as mentioned earlier.

Then, a feature search is performed over the features of both the source and target. Specifically, a variation step δ is defined for each feature over a specified interval (Table 3), generating multiple combinations of feature

Feature	δ	Interval	Type of Feature
$LHSV$	$0.25h^{-1}$	$[0.5h^{-1}, 4.25h^{-1}]$	Operating Conditions (OPC)
T	$10°C$	$[360°C, 410°C]$	
ppH_2	10 bar	[90 bar, 150 bar]	
TMP	$20°C$	$[450°C, 550°C]$	Feedstock Property (FP)
N_0	150 ppm	[500 ppm, 3100 ppm]	
S_0	0.5 %m/m	[0.5 %m/m, 4 %m/m]	
Res_0	0.75 %m/m	[5 %m/m, 15.5 %m/m]	
$\boldsymbol{Tire}$	2.25 %	[0.5 %, 30 %]	

Table 3: Increments and intervals used for the grid search in the data set simulation. The feature $\boldsymbol{Tire}$ only concerns the target domain.

Algorithm 3 Data sets simulation

Input: Number of data sets M, length of each data sets n; Number of FP n_{FP}.
while $t \leq$ M **do**
  Select n_{FP} FP points with KS.
  Combine each FP point with all possible OPC $\Rightarrow$ this results in n_{FP} mini data sets.
  Compute output nitrogen and retain only the points with $y \in [5 \text{ ppm}, 500 \text{ ppm}]$.
  For each filtered mini data set, apply KS on the features to select n_{PFP} points.
  Concatenate all the n_{PFP} points to form the final data set $(\mathbf{X}_t, \boldsymbol{y}_t)$.
  Add uniform noise $U[-0.15\, y_i, 0.15\, y_i]$ to the outputs $\boldsymbol{y}$ and force them to be between $[5 \text{ ppm}, 500 \text{ ppm}]$.
  Remove FP_1
  $t = t + 1$
end while
Return: $\{(\mathbf{X}_i, \boldsymbol{y}_i)\}_{i=1}^{\text{M}}$ data sets

values.

To generate a data set, we first created all possible FP points from the predefined variations and selected n_{FP} of them using the Kennard–Stone algorithm. Each selected FP was then combined with all possible operating condition values to form a data subset corresponding to that FP. The nitrogen outputs were computed, and only points within $[5, 500]$ ppm nitrogen were retained. For each data subset, the KS algorithm was applied to select a representative sample of n_{PFP} points. All selected n_{PFP} points from all data subsets were concatenated to form one complete data set containing n_{FP} distinct FPs and n_{PFP} points for each FP. To simulate multiple distinct data sets (as in the case of the target domain), the procedure was repeated, removing the first FP (FP_1) from the selection in each iteration. This ensures that the Kennard-Stone algorithm selects a different combination of points in each data set, thereby generating diverse data sets configurations for robust performance assessment. Finally, uniform noise of $\pm 15\%$ was added to the outputs, clipped to $[5, 500]$ ppm nitrogen to account for measurement uncertainty.

Algorithm 3 summarizes these steps. We simulated one source data set containing 10,000 points with 8 points per FP, resulting in 1,250 different FPs. For the target domain, we simulated 20 different data sets of 10,020 points each, with 5 points per FP. This means each target data set contains 2,004 different FPs. As will be detailed later, only a few points of each target data set are used for training, and the rest for testing.

To assess the differences between the simulated target data sets, Figure 4 shows the matrix of common samples between them. Only a few data sets share a significant number of points, confirming that the generated data sets are sufficiently diverse. This diversity ensures that the evaluation of the proposed methods is robust and not dependent on a specific data configuration.

4.2 Real data sets

The source data set comes from the HydroDeNitration (HDN) of VGO using a fossil-based catalyst (Becker et al. 2015), while the target data set comes from the HDN of tire co-processing (Badlaoui et al. 2025). Both

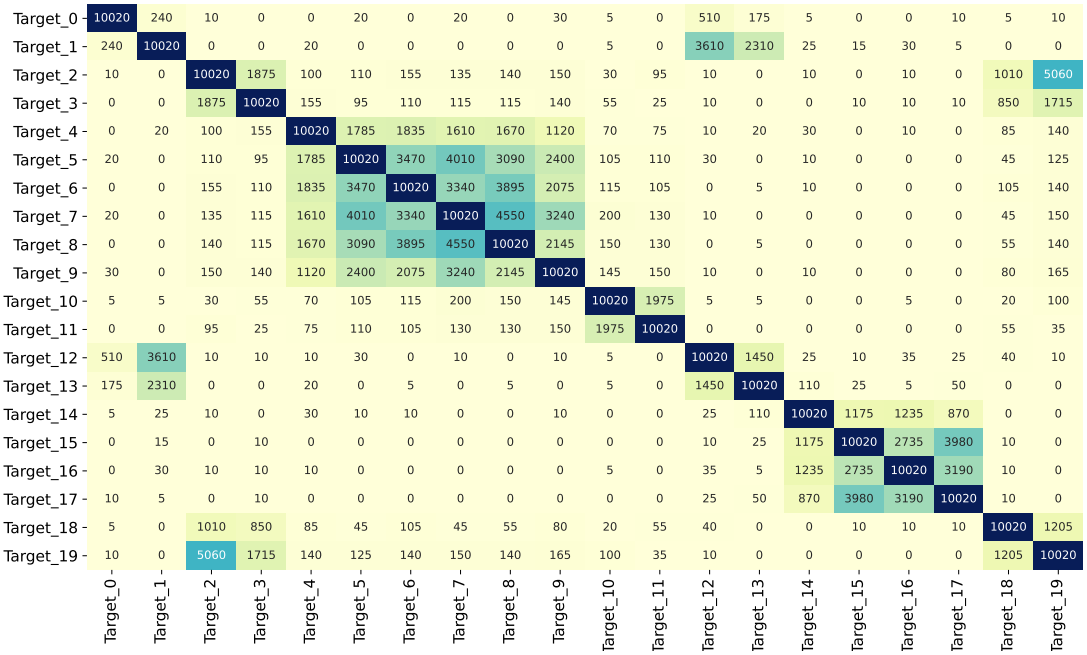

Figure 4: Matrix of common samples between target data sets. Each target data set contains 10020 points.

data sets were filtered to retain only the points with $y \in [5, 500 \text{ ppm}]$ to account for measurement uncertainty. The source data set contains 121 points with 20 FP, whereas the target data set contains 221 points with 6 FP.

4.3 Comparison methodology

We compare each proposed method against relevant competitors. The first set of competitors operates without knowledge transfer using only the target data, while the second set incorporates knowledge transfer techniques.

Without knowledge transfer, HBTL is compared with MHwG using a weakly informative prior and tuning the standard deviations of the proposal distribution s^i (cf. section 3.1.2), and the classical gradient-based optimization method L-BFGS-B (Byrd et al. 1995). For DHTLM and DHTLFT, the comparison is made with an MLP.

With knowledge transfer, we consider the full fine-tuning settings, where the previous competitors are initialized with the pre-trained source model's parameters (same initialization as the proposed methods). These competitors are denoted as $\text{MHwG}_{\text{init}}$, $\text{L-BFGS-B}_{\text{init}}$, and MLP_{init}.

Additionally, we consider fine-tuning the last layers (instead of earlier layers) as a transfer learning competitor for the MLP-based methods, denoted FTLL, which stands for Fine-Tuning Last Layers. In this case, the first N_1 layers' weights are frozen, and only the last N_2 layers and output weights are fine-tuned (cf. Section 3.2.1). The input weights are always frozen for FTLL. However, we study two scenarios: the weights related to the input feature $\boldsymbol{Tire}$ are either trained or frozen. When trained, we denote the competitor as $\text{FTLL}_{\text{tire}}$.

The comparison between the different methods and the competitors on the test sets is performed using the Mean Absolute Error (MAE), defined as

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|. \tag{13}$$

It is worth noting that each method and the competitors are optimized independently. For example, in FTLL, given a training size, the N_1 value can be different from that of the DHTLFT method. Also, the architecture of DHTLM can be different from DHTLFT. The optimization process and the training are detailed in the following section.

4.4 Hyperparameters tuning and training

Here, we define the different hyperparameters used and the training strategy for both the proposed methods and their respective competitors. Table 4 summarizes the different hyperparameters used. Some hyperparameters were already mentioned within each section of the proposed methods, such as g for the prior in HBTL, the number of iterations J for tuning the standard deviation s^i within the MHwG algorithm, N or N_1 or n^i in DHTLFT and DHTLM.

Hyperparameters \ Methods	HBTL	$\mathrm{MHwG}_{\mathrm{init}}$ (or MHwG)	$\mathrm{L\text{-}BFGS\text{-}B}_{\mathrm{init}}$ (or L-BFGS-B)	DHTLM	DHTLFT (FTLL)	$\mathrm{MLP}_{\mathrm{init}}$ (or MLP)
$g \in \{1, 10, 100, 1000, 10000\}$	✓	–	–	–	–	–
$J = 100$	✓	✓	–	–	–	–
$\epsilon \in \{0.01, 0.001, 0.0001\}$	–	–	–	✓	✓	✓
$\lambda \in \{0.99, 0.999\}$	–	–	–	✓	✓	✓
$\gamma \in \{1, 10, 100, 10000, 100000\}$	–	–	–	✓	–	–
$\mathrm{N} \in \{1, 2, 3\}$	–	–	–	✓	✓	✓
$\mathrm{N}_1 \in \{1, \cdots, \mathrm{N}\}$	–	–	–	✓	✓	–
$n^i \in \{16, 32, 64, 128, 256\}, i \in \{1, \cdots, \mathrm{N}\}$	–	–	–	✓	✓	✓

Table 4: Hyperparameters of the different methods and the baselines.

The training process and the selection of these hyperparameters are described below:

- **Target train size**: each $n_t \in \{5, 10, 15, \ldots, 100\}$ is selected multiple times using the KS algorithm for each target simulated data set, or 20 times randomly for the real data set. The remaining samples are reserved for testing.

- **Cross-validation and hyperparameter selection**: for each hyperparameter configuration and n_t, cross-validation is performed with a fold size f_{size} depending on the training size ($5 \leq n_t \leq 20$, $f_{\text{size}} = 1$; $20 < n_t \leq 30$, $f_{\text{size}} = 2$; $35 \leq n_t \leq 100$, $f_{\text{size}} = 5$). Model selection is based on the regression loss (Equation (5)), with the best hyperparameters chosen according to the smallest mean validation loss.

- **Data normalization**: For MLP-based methods and their competitors, all input features are standardized using the Z-score normalization:

$$\mathbf{x}_{\text{norm}} = \frac{\mathbf{x} - \boldsymbol{\mu}}{\boldsymbol{\sigma}} \tag{14}$$

  where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are the mean and standard deviation vectors computed on the training set. The same normalization parameters are applied to validation and test sets to prevent data leakage.

- **Test evaluation**: after selecting the best hyperparameters, each fold produces a trained model whose performance is evaluated on the test set using MAE. For each n_t, the test MAEs from all folds are averaged to obtain a single test score. Finally, the mean and minimum–maximum interval of these test scores are reported across repetitions for each n_t.

- **Optimizer**: Adam optimizer (Kingma et al. 2017) was used for the MLP-based methods and their competitors, with different learning rate ϵ. An exponential learning rate schedule (Li et al. 2019b) was also applied, with varying decay factors λ: $\epsilon_t = \epsilon \times \lambda^t$ where ϵ is the initial learning rate.

- **MLP-based methods optimization:** using cross-validation to find optimal hyperparameters for MLP-based methods and their competitors across all n_t values is computationally prohibitive. To efficiently explore the hyperparameter search space without exhaustively evaluating all combinations, we employ Optuna (Akiba et al. 2019), a Bayesian optimization framework that uses the Tree-structured Parzen Estimator (TPE) algorithm. TPE intelligently suggests hyperparameter combinations based on previous trial performance, balancing exploration of new parameter regions with exploitation of promising areas, thereby significantly reducing computational cost compared to exhaustive grid search while maintaining optimization effectiveness. We conduct up to 500 trials with early stopping after 200 consecutive trials without improvement.

- **Methods initialization**: each proposed method and some competitors (MHwG$_{\text{init}}$, L-BFGS-B$_{\text{init}}$, MLP$_{\text{init}}$, DHTLFT$_{\text{last}}$, and FTLL) are initialized from a pre-trained source model. This source model is trained using the source data. To train it, we select a subset of data using the KS algorithm, while the remaining data is reserved as a validation set that serves for early stopping in the case of MLP-based methods and their corresponding competitors, and also for selecting the best initialization in the case of HBTL, MHwG$_{\text{init}}$, and L-BFGS-B$_{\text{init}}$. Indeed, the last two are initialized similarly to HBTL with $(\hat{\boldsymbol{\theta}}_{\text{S}}, p_0)$ as mentioned earlier, and to find $\hat{\boldsymbol{\theta}}_{\text{S}}$, we use three random uniform initialization, and the best one is chosen with the remaining validation samples.

  For the source simulated data set, the training set consists of 9,000 samples for the MLP-based methods and the corresponding competitors, or 150 samples for the other since running the MHwG on huge source training set is computationally prohibitive; 150 samples are sufficient for inferring the source parameters (Iapteff et al. 2023). For the real data set, the source training set consists of 110 samples.

  For MLP-based methods, we fixed the learning rate at $\epsilon = 0.01$ and the decay factor at $\lambda = 0.999$ for the pre-trained source model, to reduce computational complexity in Optuna.

- **Iteration (epochs)**: we ran 20,000 iterations for HBTL, MHwG, MHwG$_{\text{init}}$, and the last 1,000 iterations is used for inference. For MLP-based methods and their competitors, we used 500 epochs with early stopping patience of 30 epochs. For L-BFGS-B, 1000 different random initializations are used, and the best one is chosen with the cross-validation.

- **Batch size**: for the MLP-based methods, a batch size of 2 is used for the target domain. For the pre-trained source model, a batch size of 64 is used for the simulated data set and 32 for the real data set. For DHTLM, since the source is trained simultaneously with the target, its batch size is adjusted according to the target training size each time, ensuring that all available source samples are used.

- **Tuning the standard deviations** s^i: for HBTL, MHwG, and MHwG$_{\text{init}}$, the s^i are tuned every J iterations (cf. section 3.1.2) in order to have $\tau_J\% \in [30\%, 70\%[$. We use the following tuning strategy:

  - $\tau_J\% \in [0, 10\%] \rightarrow s^i_{\text{new}} = 0.25\times\ s^i$
  - $\tau_J\% \in [10, 20\%] \rightarrow s^i_{\text{new}} = 0.5\times\ s^i$
  - $\tau_J\% \in [20, 30\%] \rightarrow s^i_{\text{new}} = 0.8\times\ s^i$
  - $\tau_J\% \in]70, 80\%] \rightarrow s^i_{\text{new}} = 1.25\times\ s^i$

- **Experimentation materials**: The methods were coded in Python. All experiments were conducted on IFPEN's Orion computing cluster running Red Hat Enterprise Linux 9.4, equipped with dual AMD EPYC 9534 64-Core Processors (256 total CPU cores) and 755 GB of RAM. The supercomputer's job scheduling system allowed us to submit multiple combination jobs in parallel, significantly reducing overall computation time. However, due to job submission limitations, the complete execution of all combinations still took a considerable amount of time. The resolution of the ODEs was performed using the LSODE algorithm from the SLATEC numerical library (Radhakrishnan et al. 1993), implemented through a FORTRAN code compiled as a dynamic link library (.dll) and interfaced with Python. This allows faster ODE resolution (≈ 5 times faster than using classical Python solvers). The original code of the SLATEC library can be found online. For L-BFGS-B optimization, the implementation from the scipy.optimize library (Virtanen et al. 2020) was used with default hyperparameters and the bounds of the parameters. For implementing the MLP-based methods and their competitors, PyTorch (Paszke et al. 2019) was used.

5 Results

This section presents the results of the proposed methods and their competitors. First, the simulated data sets results are presented followed by the results for real data sets.

5.1 Results on simulated data sets

Starting from HBTL and its competitors which leverage the target ODE structure, Figure 5 shows the comparison between the different competitors on the test sets. We observe that $\text{MHwG}_{\text{init}}$ consistently provides the best mean prediction errors and tighter min–max intervals across all training sizes (left panel). Interestingly, the classical gradient descent method L-BFGS-B with 1000 different initializations outperforms both the randomly initialized MHwG and even its transfer learning variant $\text{L-BFGS-B}_{\text{init}}$. This suggests that using more initializations improves results, though at the expense of computational time. However, gradient descent methods remain vulnerable to local minima, as evidenced by $\text{L-BFGS-B}_{\text{init}}$, which shows poor predictions even with 100 training points. The multi-start approach L-BFGS-B faces similar challenges, visible in the large min–max interval at $n_t = 100$. While increasing the number of multi-starts can alleviate this issue, it requires substantial computational resources. In this context, transfer learning emerges as an efficient alternative for rapidly building accurate prediction models.

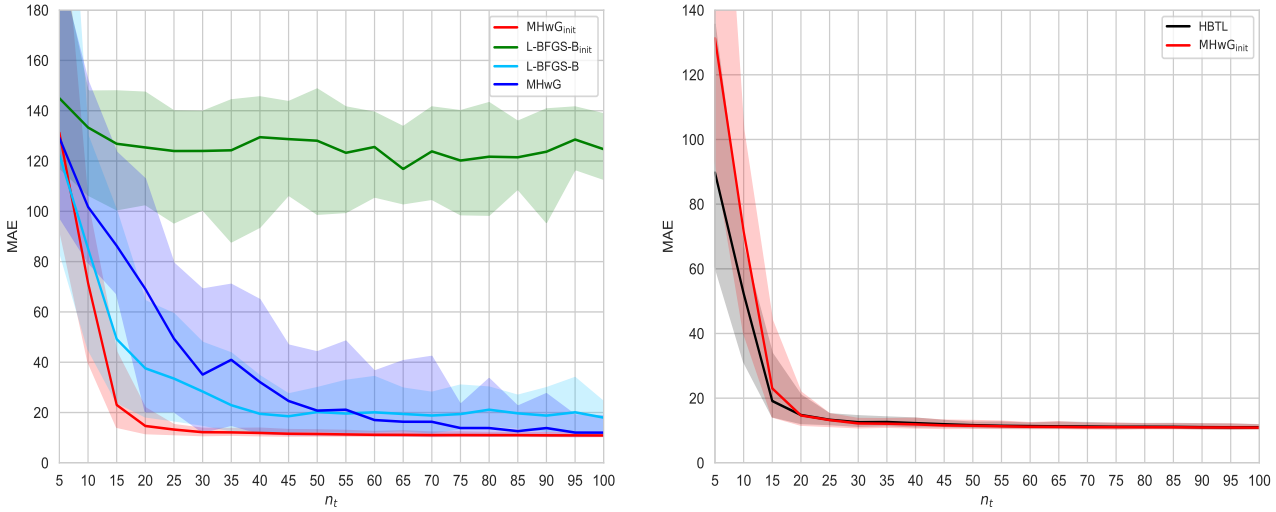

Figure 5: **Left:** Comparison between the different competitors of HBTL on the simulated data sets. The graphs show the evolution of the average MAE on the target test sets as a function of the target training sample size. The shaded areas represent the min–max intervals. **Right:** Comparison between HBTL and the best competitor $\text{MHwG}_{\text{init}}$ on the simulated test sets.

When no knowledge transfer is used, the classical methods can produce very poor results. The best competitor $\text{MHwG}_{\text{init}}$ is then compared to our proposed method HBTL on the test sets (right panel), which highlights the superiority of the heterogeneous Bayesian transfer learning both in mean prediction error and min–max intervals for training sizes $n_t \leq 20$. As the training size increases beyond this threshold, the two methods converge to similar prediction errors and min–max intervals.

Similarly, the results of the different competitors for the MLP-based methods are shown in Figure 6, where MLP_{init} emerges as the best competitor in terms of both mean prediction errors and min–max intervals. The worst performer is MLP (left panel), as it does not leverage knowledge transfer. Additionally, both fine-tuning last layers methods (FTLL and $\text{FTLL}_{\text{Tire}}$) yield worse mean prediction errors compared to the full fine-tuning approach MLP_{init}, along with highly unstable min–max prediction intervals. However, $\text{FTLL}_{\text{Tire}}$ outperforms FTLL since it trains the weights related to the $\boldsymbol{Tire}$ feature. This suggests that fine-tuning the last layers is not an effective strategy for knowledge transfer in our case, as the main domain shift occurs at the input level due to the new feature $\boldsymbol{Tire}$ (cf. Section 3.2.1). This hypothesis is confirmed in the right panel, where the proposed methods DHTLM and DHTLFT, which fine-tune the earlier layers rather than the last layers, surpass even the best competitor MLP_{init} for most training sizes n_t. The min-max intervals are quite similar. Comparing DHTLM and DHTLFT, we observe that DHTLM yields lower prediction errors for most n_t, particularly for $n_t \geq 30$, which highlights the importance of the soft constraint over hard freezing (as mentioned earlier in Section 3.2.2).

Finally, DHTLM is compared to HBTL in Figure 7, which shows the superiority of the Bayesian method,

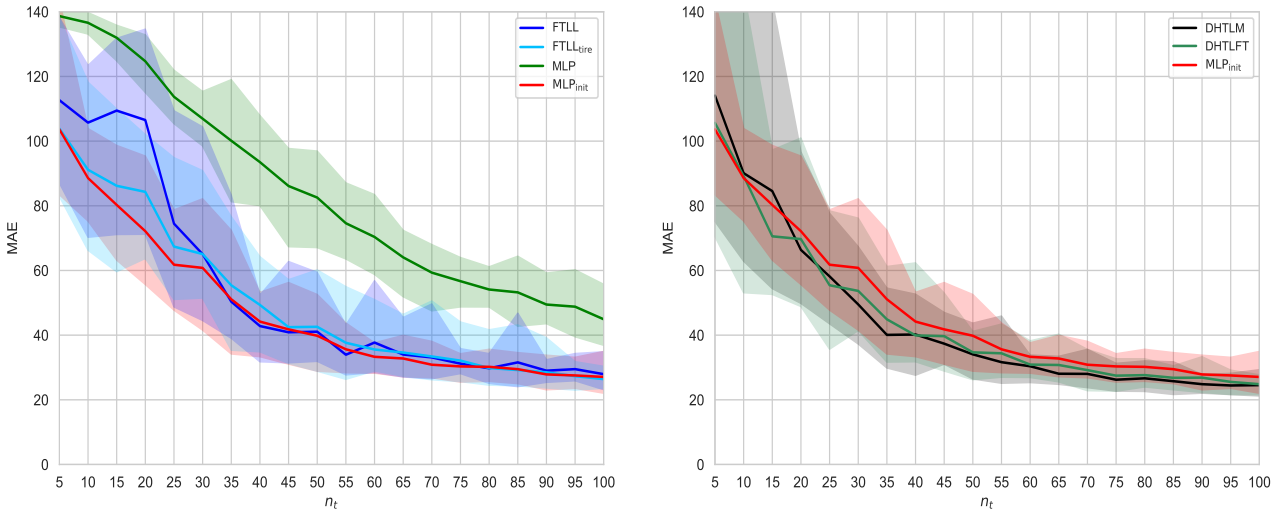

Figure 6: **Left:** Comparison between the different competitors of DHTLM and DHTLFT on the simulated test sets. **Right:** Comparison between the proposed methods DHTLM and DHTLFT against the best competitor MLP$_{\text{init}}$ on the simulated test sets.

as it leverages the knowledge of the target ODE.

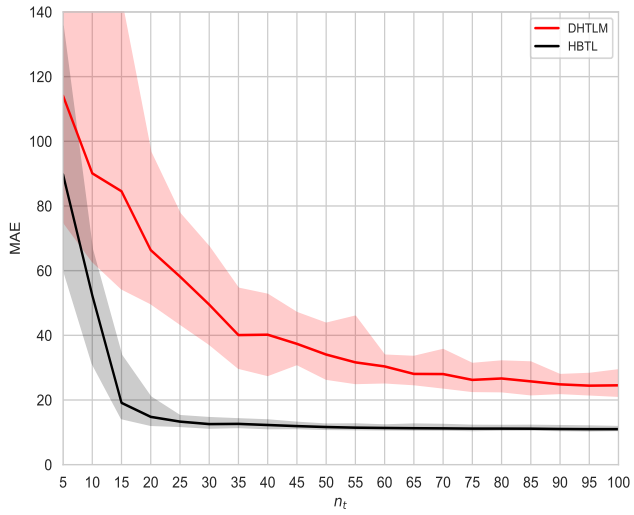

Figure 7: Comparison between HBTL and DHTLM on the simulated test sets.

In other words, HBTL uses physical information across domains (source and target ODEs), highlighting the importance of incorporating such information when available.

5.2 Results on real data sets

This section discusses the results on the real data sets in the same manner as for the simulated data sets. Figure 8 shows the results for the different competitors of HBTL. In contrast to the simulated data sets, L-BFGS-B$_{\text{init}}$ provides the best prediction performance (left panel) for $n_t \leq 15$, while it performs similarly to MHwG$_{\text{init}}$ for larger n_t, where both methods stand out as the best competitors. This outcome may occur but is not guaranteed for L-BFGS-B$_{\text{init}}$ (single initialization). Given that for larger n_t the methods perform comparably, we choose L-BFGS-B$_{\text{init}}$ as the best competitor. The classical multi-start gradient descent L-BFGS-B remains superior to the one-start based algorithm MHwG, consistent with the simulated data sets results.

HBTL is then compared against its best competitor in the right panel, which shows the superiority of the proposed heterogeneous Bayesian transfer learning in both mean prediction error and min–max intervals.

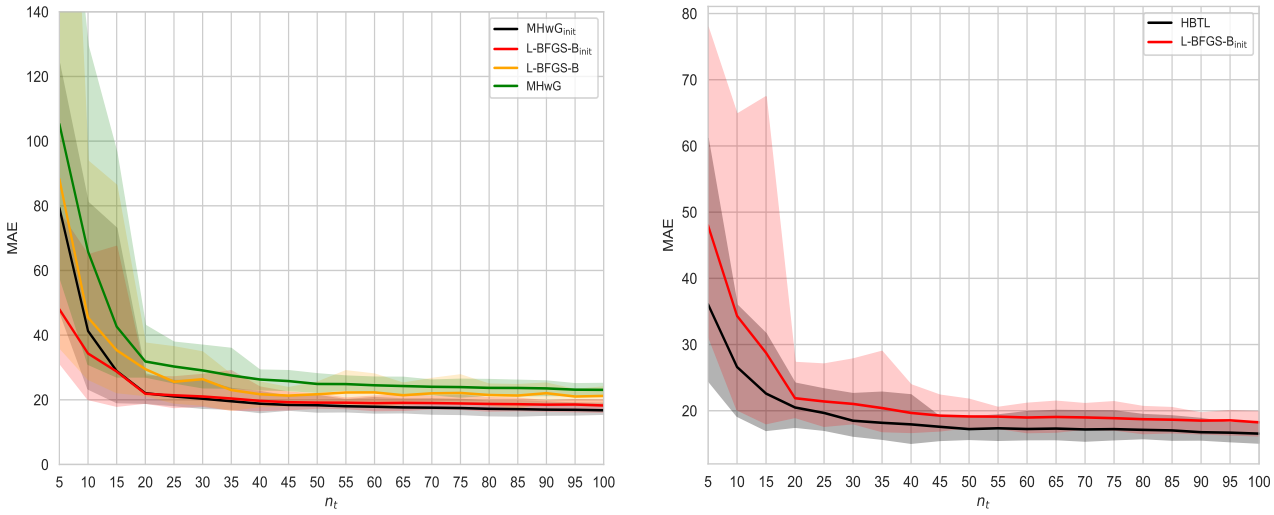

Figure 8: **Left:** Comparison between the different competitors of HBTL on the real test sets. **Right:** Comparison between HBTL and the best competitor L-BFGS-B$_{\text{init}}$ on the real test sets.

Back to the MLP-based methods, the competitors are compared against each other in Figure 9, where MLP$_{\text{init}}$ emerges as the best performer in both mean and min–max prediction errors (left panel). The other two methods yield unstable min–max prediction intervals, confirming that fine-tuning the last layers is not an effective choice in cases such as ours as mentioned earlier. MLP$_{\text{init}}$ is then compared against the proposed methods in the right panel where DHTLM emerges as the best method in both mean and min-max intervals prediction across most n_t. MLP$_{\text{init}}$ shows better min-max intervals predictions errors compared to DHTLFT, as the latter suffers from instability, which concludes the importance of the soft-constraint compared to the hard constraint (freezing layers).

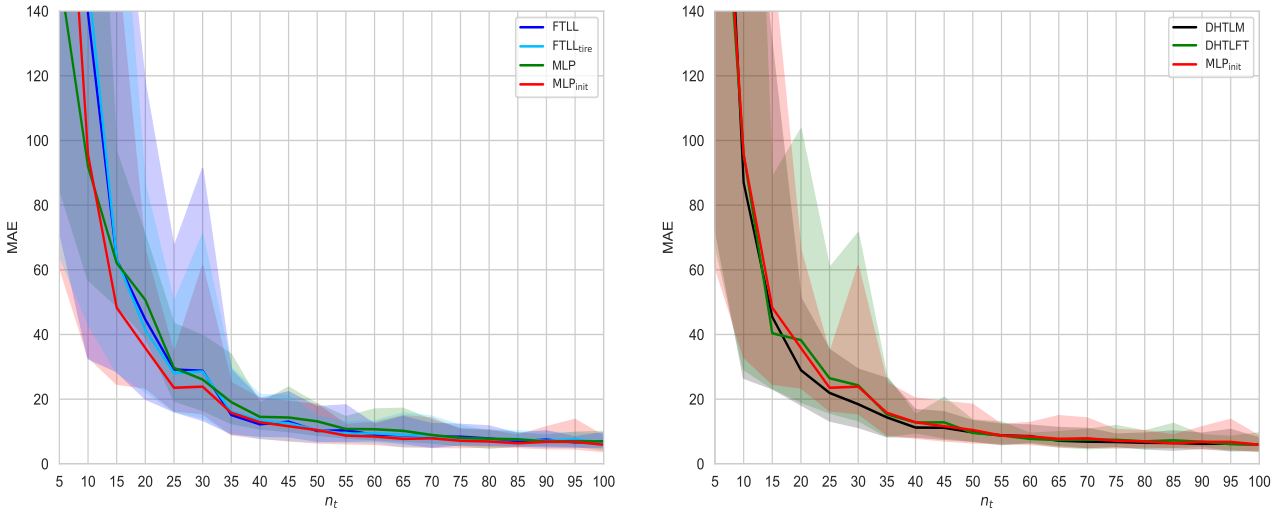

Figure 9: **Left:** Comparison between the competitors of DHTLM and DHTLFT on real test sets. **Right:** Comparison between the proposed methods DHTLM and DHTLFT against the best competitor MLP$_{\text{init}}$ on the real test sets.

An interesting result is that the data-driven methods outperformed the HBTL method, even though HBTL uses the target ODE structure (Figure 10). In the early stages of developing kinetic models, the ODE is often not as well-defined as it becomes after extensive study. This highlights the importance of developing predictive models that do not rely on knowledge of the ODE structure.

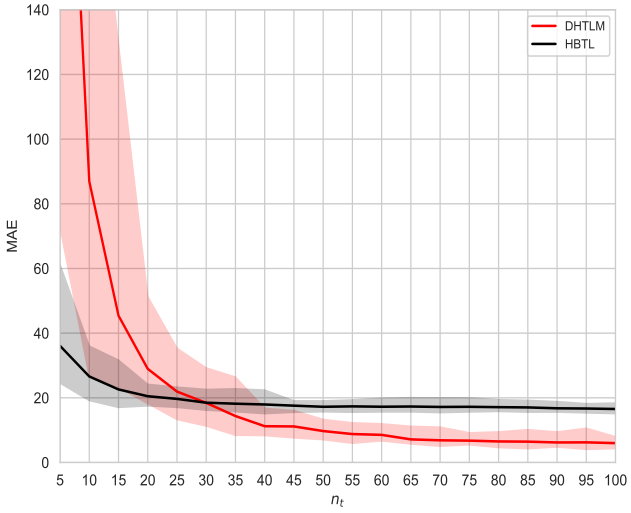

Figure 10: Comparison between the proposed methods HBTL and DHTLM on the real test sets.

6 Conclusion and perspective

The objective was to develop predictive models for nitrogen concentration in hydrotreatment reactors. The target task focused on renewable feedstocks, particularly tire co-processing feedstocks. Since the available data are limited, a rich source data set from fossil feedstocks was leveraged to improve the constructed predictive model. This paper employs transfer learning, specifically heterogeneous transfer learning, because the feature spaces of the two domains differ. Three heterogeneous transfer learning methods for regression tasks were proposed, addressing a challenge that has been rarely explored in the literature. The proposed methods were developed for two scenarios: with and without knowledge of the target ODE structure. HBTL was proposed for the first case, while DHTLM and DHTLFT were proposed for the second. The proposed methods achieve better results than both classical non-transfer competitors and classical transfer learning competitors (including full fine-tuning and fine-tuning of last layers) on both simulated and real-world data sets. HBTL performs best overall among the three methods since it leverages both the target and source ODE structures. However, for larger training sizes, the data-driven methods surpass it on real data sets. This is because the assumed ODE for the target domain is imperfect, which highlights the importance of the other two methods. DHTLM outperforms DHTLFT because it imposes a soft transfer-based constraint compared to the hard freezing of the second.

The proposed methods not only address the industrial problem at hand but also fill an important gap in the literature on heterogeneous transfer learning for regression tasks. Looking ahead, our short perspective is to further improve the HTL data-driven approaches, particularly DHTLM and DHTLFT, by incorporating prior physical knowledge with the source ODE. Combining the prior physical information into the proposed methods can be addressed by PIML frameworks (Bradley et al. 2022). To the best of our knowledge, no existing HTL-PIML method in the literature. Although the source ODE does not capture all the essential physics of the target domain due to heterogeneous physics, it can still provide valuable prior information. To achieve this, the kinetic parameters of the source ODE must be refitted using target data to ensure they become relevant and useful for the target domain.

7 Code availability

The code for all methods is available at `https://github.com/Youba655/Heterogeneous-transfer-learning-for-highly-non-linear-regression-tasks`.

8 Data Availability

The simulated and real data sets used in this study cannot be made publicly available due to confidentiality restrictions related to proprietary industrial processes at IFP Energies nouvelles. The code provided allows

reproduction of the methods on similar data structures.

Funding

This work was supported by IFP Energies nouvelles (Y.A.).

References

Agarap, Abien Fred (2019). *Deep Learning using Rectified Linear Units (ReLU)*. URL: https://arxiv.org/abs/1803.08375.

Akiba, Takuya, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama (2019). *Optuna: A Next-generation Hyperparameter Optimization Framework*. URL: https://arxiv.org/abs/1907.10902.

Badlaoui, Meryem, Ngoc Yen Phuong Cao, Warumporn Pejpichestakul, Benoit Celse, Bertrand Guichard, Minh Tuan Nguyen, Nadège Charon, and Joris W. Thybaut (2025). "Performance comparison of tire pyrolysis oils in hydrotreating toward high-quality fuel". In: *Chemical Engineering Journal* 520, p. 165115. ISSN: 1385-8947. DOI: https://doi.org/10.1016/j.cej.2025.165115.

Bao, Runxue, Yiming Sun, Yuhe Gao, Jindong Wang, Qiang Yang, Zhi-Hong Mao, and Ye Ye (2024). *A Recent Survey of Heterogeneous Transfer Learning*. URL: https://arxiv.org/abs/2310.08459.

Basha, S.H. Shabbeer, Sravan Kumar Vinakota, Viswanath Pulabaigari, Snehasis Mukherjee, and Shiv Ram Dubey (Jan. 2021). "AutoTune: Automatically Tuning Convolutional Neural Networks for Improved Transfer Learning". In: *Neural Networks* 133, pp. 112–122. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2020.10.009.

Becker, Per Julian, Benoit Celse, Denis Guillaume, Hugues Dulot, and Victor Costa (2015). "Hydrotreatment modeling for a variety of VGO feedstocks: A continuous lumping approach". In: *Fuel* 139, pp. 133–143. ISSN: 0016-2361. DOI: https://doi.org/10.1016/j.fuel.2014.08.032.

Becker, Per Julian, Warumporn Pejpichestakul, and Benoit Celse (2024). *Applications of Monte Carlo Markov Chains in Kinetic Modelling of Hydroprocessing with Transfer Learning: Parameter Identification from Pilot Plant and Industrial Data*. Available at SSRN. SSRN Electronic Journal. DOI: 10.2139/ssrn.5131444.

Bradley, William, Jinhyeun Kim, Zachary Kilwein, Logan Blakely, Michael Eydenberg, Jordan Jalvin, Carl Laird, and Fani Boukouvala (2022). "Perspectives on the integration between first-principles and data-driven modeling". In: *Computers Chemical Engineering* 166, p. 107898. ISSN: 0098-1354. DOI: https://doi.org/10.1016/j.compchemeng.2022.107898.

Byrd, Richard H., Peihuang Lu, Jorge Nocedal, and Ciyou Zhu (Sept. 1995). "A limited memory algorithm for bound constrained optimization". English. In: *SIAM Journal on Scientific Computing* 16, pp. 1190–1208. ISSN: 1064-8275. DOI: 10.1137/0916069.

Cao, Ngoc-Yen-Phuong, Benoit Celse, Denis Guillaume, Isabelle Guibard, and Joris W. Thybaut (2020). "Accelerating Kinetic Parameter Identification by Extracting Information from Transient Data: A Hydroprocessing Study Case". In: *Catalysts* 10.4. ISSN: 2073-4344. DOI: 10.3390/catal10040361.

Caruana, Rich (1997). "Multitask Learning". In: *Machine Learning* 28, pp. 41–75. URL: https://api.semanticscholar.org/CorpusID:45998148.

Day, Oscar and Taghi Khoshgoftaar (Sept. 2017). "A survey on heterogeneous transfer learning". In: *Journal of Big Data* 4, p. 29. DOI: 10.1186/s40537-017-0089-0.

Gelman, A., G. O. Roberts, and W. R. Gilks (1996). "Efficient Metropolis jumping rules". In: *Bayesian Statistics*. Ed. by J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith. Oxford University Press, Oxford, pp. 599–608.

Iapteff, Loïc, Julien Jacques, Benoit Celse, and Victor Costa (2023). "Reducing the Number of Experimental Points to Fit Kinetic Models: A Bayesian Approach". In: *Industrial & Engineering Chemistry Research* 62.28, pp. 10903–10914. DOI: 10.1021/acs.iecr.2c03862.

Kennard, R. W. and L. A. Stone (1969). "Computer Aided Design of Experiments". In: *Technometrics* 11.1, pp. 137–148. DOI: 10.1080/00401706.1969.10490666.

Kingma, Diederik P. and Jimmy Ba (2017). *Adam: A Method for Stochastic Optimization*. URL: https://arxiv.org/abs/1412.6980.

Lee, Yoonho, Annie S. Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn (2023). *Surgical Fine-Tuning Improves Adaptation to Distribution Shifts*. URL: https://arxiv.org/abs/2210.11466.

Li, Haoliang, Sinno Jialin Pan, Shiqi Wang, and Alex C. Kot (2020). "Heterogeneous Domain Adaptation via Nonlinear Matrix Factorization". In: *IEEE Transactions on Neural Networks and Learning Systems* 31.3, pp. 984–996. DOI: 10.1109/TNNLS.2019.2913723.

Li, Jingjing, Mengmeng Jing, Ke Lu, Lei Zhu, and Heng Tao Shen (2019a). "Locality Preserving Joint Transfer for Domain Adaptation". In: *IEEE Transactions on Image Processing* 28.12, pp. 6103–6115. DOI: 10.1109/TIP.2019.2924174.

Li, Zhiyuan and Sanjeev Arora (2019b). *An Exponential Learning Rate Schedule for Deep Learning*. URL: https://arxiv.org/abs/1910.07454.

Oliveira, Luís Pereira de, Damien Hudebine, Denis Guillaume, and Jan J. Verstraete (2016). "A Review of Kinetic Modeling Methodologies for Complex Processes". In: *Oil & Gas Science and Technology – Revue d'IFP Energies nouvelles* 71, p. 45. URL: https://api.semanticscholar.org/CorpusID:55457448.

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala (2019). "PyTorch: an imperative style, high-performance deep learning library". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc.

Radhakrishnan, Krishnan and Alan C. Hindmarsh (1993). "Description and use of LSODE, the Livermore Solver for Ordinary Differential Equations". In: URL: https://api.semanticscholar.org/CorpusID:53752439.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (1986). "Learning internal representations by error propagation". In: URL: https://api.semanticscholar.org/CorpusID:62245742.

Shi, Xiaoxiao, Qi Liu, Wei Fan, Qiang Yang, and Philip S. Yu (2010). "Predictive Modeling with Heterogeneous Sources". In: *Proceedings of the 2010 SIAM International Conference on Data Mining (SDM)*, pp. 814–825. DOI: 10.1137/1.9781611972801.71.

Shi, Xiaoxiao, Qi Liu, Wei Fan, and Philip S. Yu (2013). "Transfer across Completely Different Feature Spaces via Spectral Embedding". In: *IEEE Transactions on Knowledge and Data Engineering* 25.4, pp. 906–918. DOI: 10.1109/TKDE.2011.252.

Shu, Xiangbo, Guo-Jun Qi, Jinhui Tang, and Jingdong Wang (2015). "Weakly-Shared Deep Transfer Networks for Heterogeneous-Domain Knowledge Propagation". In: *Proceedings of the 23rd ACM international conference on Multimedia*. URL: https://api.semanticscholar.org/CorpusID:207223936.

Silvestrin, Luis Pedro, Harry van Zanten, Mark Hoogendoorn, and Ger Koole (Dec. 2023). "Transfer learning across datasets with different input dimensions: An algorithm and analysis for the linear regression case". In: *Journal of Computational Mathematics and Data Science* 9, p. 100086. ISSN: 2772-4158. DOI: 10.1016/j.jcmds.2023.100086.

Speight, J.G. (Jan. 2011). *The Refinery of the Future*. DOI: 10.1016/C2009-0-20064-X.

Suder, Piotr M., Jason Xu, and David B. Dunson (2023). *Bayesian Transfer Learning*. URL: https://arxiv.org/abs/2312.13484.

Tierney, Luke (1994). "Markov Chains for Exploring Posterior Distributions". In: *The Annals of Statistics* 22.4, pp. 1701–1728. DOI: 10.1214/aos/1176325750.

Virtanen, P., R. Gommers, T. E. Oliphant, Matteo Cucchi, et al. (2020). "SciPy 1.0: fundamental algorithms for scientific computing in Python". In: *Nat. Methods* 17, pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

Wu, Di, Boyu Wang, Doina Precup, and Benoit Boulet (2020). "Multiple Kernel Learning-Based Transfer Regression for Electric Load Forecasting". In: *IEEE Transactions on Smart Grid* 11.2, pp. 1183–1192. DOI: 10.1109/TSG.2019.2933413.

Wu, Hanrui, Hong Zhu, Yuguang Yan, Jiaju Wu, Yifan Zhang, and Michael K. Ng (2021). "Heterogeneous Domain Adaptation by Information Capturing and Distribution Matching". In: *IEEE Transactions on Image Processing* 30, pp. 6364–6376. DOI: 10.1109/TIP.2021.3094137.

Xiao, Guangxuan, Ji Lin, and Song Han (2023). *Offsite-Tuning: Transfer Learning without Full Model*. URL: https://arxiv.org/abs/2302.04870.

Yan, Yuguang, Wen Li, Michael Ng, Mingkui Tan, Hanrui Wu, Huaqing Min, and Qingyao Wu (2017). "Learning Discriminative Correlation Subspace for Heterogeneous Domain Adaptation". In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pp. 3252–3258. DOI: 10.24963/ijcai.2017/454.

Yang, Qiang, Yu Zhang, Wenyuan Dai, and Sinno Jialin Pan (2020). *Transfer Learning*. Cambridge University Press.

Yao, Yuan, Yu Zhang, Xutao Li, and Yunming Ye (2019). *Heterogeneous Domain Adaptation via Soft Transfer Network*. URL: https://arxiv.org/abs/1908.10552.

Ye, Han-Jia, De-Chuan Zhan, Yuan Jiang, and Zhi-Hua Zhou (May 2020). "Heterogeneous Few-Shot Model Rectification With Semantic Mapping". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PP, pp. 1–1. DOI: 10.1109/TPAMI.2020.2994749.

Zellner, Arnold (1986). "On assessing prior distributions and Bayesian regression analysis with g-prior distributions". In: *Bayesian Inference and Decision Techniques: Essays in Honor of Bruno de Finetti*. Ed. by Prem Goel and Arnold Zellner. Elsevier Science Publishers, pp. 233–243.

Zhang, Wen, Lingfei Deng, Lei Zhang, and Dongrui Wu (Feb. 2023). "A Survey on Negative Transfer". In: *IEEE/CAA Journal of Automatica Sinica* 10.2, pp. 305–329. ISSN: 2329-9274. DOI: 10.1109/jas.2022.106004.

Zhou, Joey Tianyi, Sinno Jialin Pan, and Ivor W. Tsang (2019). "A deep learning framework for Hybrid Heterogeneous Transfer Learning". In: *Artificial Intelligence* 275, pp. 310–328. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2019.06.001.

Zhuang, Fuzhen, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He (2021). "A Comprehensive Survey on Transfer Learning". In: *Proceedings of the IEEE* 109.1, pp. 43–76. DOI: 10.1109/JPROC.2020.3004555.