LEARNING A GENERATIVE INVERSE TO ADAPT THE FORWARD: SOURCE-FREE DOMAIN ADAPTATION FOR SEQUENCE REGRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Source-Free Domain Adaptation (SFDA) has emerged as a pragmatic paradigm for adapting models to practical scenarios without access to source data. Most existing SFDA methods focus on image classification and rely on entropy minimization or pseudo-labeling under the cluster assumption. Sequence regression instead has a continuous output space and does not provide discrete category codes. Simply extending classification-style methods to regression often fails to yield satisfactory results and may even lead to negative transfer. To address this issue, we propose Generative Inverse Regression Adaptation (GIRA), a general-purpose SFDA framework designed for sequence regression. Specifically, GIRA defines generalized classes over the continuous output space and uses a conditional generative model to learn the inverse mapping of the pre-trained model, generating associated target-style inputs. These resulting samples provide adaptation signals for updating the regression model on the target domain. Furthermore, uncertainty threshold filtering and feature scale constraints are integrated to mitigate negative transfer and ensure training stability. Across nine transfer scenarios in three tasks, GIRA reduces MAE by 20.87% on average relative to zero-shot and achieves the highest MAE improvement among the compared methods in most tasks.

1 INTRODUCTION

Deep neural networks have achieved remarkable success in various sequence regression tasks, such as non-intrusive load monitoring (NILM) (Zoha et al., 2012), time series forecasting (Lim & Zohren, 2021), and speech enhancement (Wang & Chen, 2018). A critical challenge arises when deploying these models in real-world scenarios where the practical environment (target domain) differs significantly from the training environment (source domain). These distributional discrepancies between the two domains, also known as domain shift, severely degrade model performance on the target domain (Wilson & Cook, 2020).

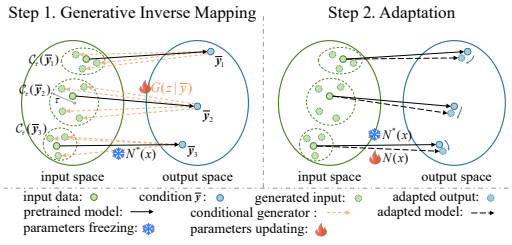


Figure 1: GIRA overview. Step 1 learns the generative inverse mapping, and Step 2 updates the regression model with input–class pairs.

Transfer learning has emerged as a predominant solution for domain shifts. While fine-tuning is a widely adopted strategy, it remains limited by the prohibitive annotation costs (Yosinski et al., 2014). Unsupervised Domain Adaptation (UDA) addresses this limitation by aligning feature distributions without target supervision (Ganin et al., 2016; Long et al., 2017). However, standard UDA methods typically require access to source data, which is often impractical in real-world applications due to strict privacy and security concerns (Liang et al., 2020).

Consequently, Source-Free Domain Adaptation (SFDA) has gained prominence as a pragmatic paradigm, which solely uses unlabeled data on the target domain to perform adaptation (Liang et al., 2020). The majority of existing SFDA methods primarily focus on visual classification problems. These approaches typically employ self-supervised objectives, such as entropy minimization (Wang

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

et al., 2021) or pseudo-labeling (Liang et al., 2020; Lee & Lee, 2023), to exploit the intrinsic structure of target data for implicit feature alignment. Furthermore, generative frameworks have also been explored to synthesize surrogate data (Kurmi et al., 2021; Li et al., 2020). These methods are mostly developed for discrete category codes under the cluster assumption, whereas regression has a continuous output space. Consequently, directly extending classification-style methods to regression often yields unsatisfactory results and may cause negative transfer.

To address this challenge, we propose **G**enerative **I**nverse **R**egression **A**daptation (GIRA), a general-purpose SFDA framework for sequence regression tasks. GIRA defines generalized classes (GC) over the continuous output space to construct adaptation signals without discrete category codes. The overall design philosophy is illustrated in Figure 1. In **Step 1**, we define a GC $\mathcal{C}_\tau(\bar{\boldsymbol{y}}_i)$ as the target inputs whose pre-trained predictions lie near a given prediction $\bar{\boldsymbol{y}}_i$, which serves as its identifier. We use target-domain inputs and corresponding $\bar{\boldsymbol{y}}_i$ to train a conditional generative model G. The generator learns an inverse mapping from each GC to associated target-style inputs. In **Step 2**, the mapping of the pre-trained model N is updated using the resulting input–GC pairs. To further enhance adaptation stability and performance, mechanisms such as uncertainty threshold filtering and feature scale constraints are integrated into the framework. The proposed approach can be conveniently applied to any encoder-decoder-type model and most sequence regression scenarios.

The main contributions of this paper are summarized as follows:

- We formulate generalized classes for continuous regression outputs and develop a source-free adaptation method that does not rely on discrete category codes. We further provide a target-risk bound analysis for the proposed method.
- We propose the GIRA adaptation approach, which learns a conditional inverse mapping from generalized classes to target-style inputs. Its generator is trained on unlabeled target data to match the target input distribution, and the resulting input–GC pairs guide updates of the pre-trained regressor without target labels.
- We evaluate GIRA on three tasks and nine transfer scenarios. Relative to the corresponding pre-adaptation models, it reduces task-average MAE by 26.93% for non-intrusive load monitoring and 38.60% for electric load forecasting, while increasing average SegSNR from 13.04 to 13.55 dB for speech enhancement.

2 Related Works

Unsupervised Domain Adaptation. Unsupervised domain adaptation aligns labeled source and unlabeled target domains through distribution-discrepancy objectives, including MMD (Long et al., 2015), JAN (Long et al., 2017), and Deep CORAL (Sun & Saenko, 2016). Adversarial approaches, including DANN (Ganin et al., 2016), ADDA (Tzeng et al., 2017), CDAN (Long et al., 2018), and MADA (Pei et al., 2018), learn domain-invariant features through domain discrimination. Representation Subspace Distance (RSD) further shows that feature-scale changes can impair regression adaptation (Chen et al., 2021). These approaches generally require source data during adaptation, which limits their real-world application, especially in privacy-sensitive scenarios.

Source-Free Domain Adaptation. Source-free domain adaptation adapts a pre-trained model using unlabeled target data without access to source samples. SHOT exemplifies classification-oriented adaptation through information maximization and self-supervised pseudo-labeling (Liang et al., 2020). Pseudo-source generation has also been explored to recover adaptation signals without source data (Kurmi et al., 2021; Li et al., 2020). These classification-oriented approaches use discrete category codes, which do not directly describe the continuous outputs of sequence regression. Only a limited number of studies have extended SFDA to other tasks, such as image regression (Dinsdale et al., 2023; He et al., 2024; Adachi et al., 2025) or time-series classification (Ragab et al., 2023; Zhong et al., 2025; Kim et al., 2024). For sequence regression, source-free adaptation has been studied in time-series forecasting (Yan et al., 2026) and remaining useful life prediction (Cao et al., 2024; Chen et al., 2025; Hou et al., 2025). Self-supervised sequence restoration has been used to adapt non-intrusive load monitoring models (Zhong et al., 2024), while latent pseudo-labels have enabled test-time adaptation for speech enhancement (Raichle et al., 2026). Overall, SFDA methods specifically designed for sequence regression remain relatively few, and most focus on individual application domains.

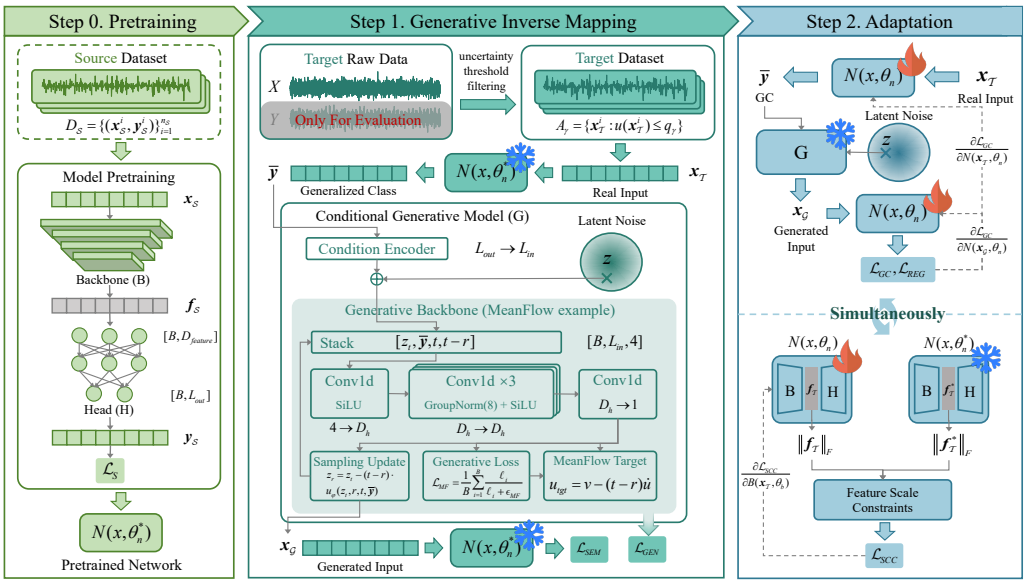


Figure 2: Overall pipeline of GIRA. Step 0 pretrains the source model using labeled source data. Step 1 filters unreliable target inputs, constructs generalized-class identifiers, and learns a conditional generative inverse mapping. Step 2 updates the regression model using $\mathcal{L}_{GC}$, with $\mathcal{L}_{REG}$ and $\mathcal{L}_{SCC}$ used for parameter and feature-scale stability. The upper two N_θ boxes in Step 2 denote forward passes through the same regression model with shared parameters.

3 Methodology

3.1 Problem Formulation

In sequence regression tasks, let $\mathcal{D}_\mathcal{S} = \{(\boldsymbol{x}_\mathcal{S}^i, \boldsymbol{y}_\mathcal{S}^i)\}_{i=1}^{N_\mathcal{S}}$ denote the labeled source domain, consisting of $N_\mathcal{S}$ pairs where $\boldsymbol{x}_\mathcal{S}^i \in \mathcal{X}_\mathcal{S} \subset \mathbb{R}^{L_{\text{in}}}$ represents an input sequence of length L_{in} and $\boldsymbol{y}_\mathcal{S}^i \in \mathcal{Y}_\mathcal{S} \subset \mathbb{R}^{L_{\text{out}}}$ represents its continuous regression target of length L_{out}. Conversely, the target domain is defined as an unlabeled set $\mathcal{D}_\mathcal{T} = \{\boldsymbol{x}_\mathcal{T}^j\}_{j=1}^{N_\mathcal{T}}$, where inputs $\boldsymbol{x}_\mathcal{T}^j \in \mathcal{X}_\mathcal{T}$ share the same length as the source. Consistent with standard domain adaptation assumptions, there exists a distributional shift between the domains, manifested as $P_\mathcal{S} \neq P_\mathcal{T}$. Given a source model $f_\mathcal{S} : \mathcal{X}_\mathcal{S} \to \mathcal{Y}_\mathcal{S}$ that has been well-trained on $\mathcal{D}_\mathcal{S}$, the objective of Source-Free Domain Adaptation is to learn an adapted target model $h : \mathcal{X}_\mathcal{T} \to \mathcal{Y}_\mathcal{T}$ using exclusively the unlabeled target data $\mathcal{D}_\mathcal{T}$ and the pre-trained parameters of $f_\mathcal{S}$, without any access to the source data $\mathcal{D}_\mathcal{S}$ during the adaptation process.

3.2 Method Overview

We propose GIRA to address the data restriction issues in sequence regression tasks. As illustrated in Figure 2, the pipeline involves three steps, namely *Pre-training*, *Generative Inverse Mapping*, and *Adaptation*. In **Step 0**, the pre-trained model parameters are obtained using source domain data. This process can be conducted offline, which implies that any checkpoint already trained on the web can also be used, provided that the model satisfies the encoder-decoder structure. In **Step 1**, Uncertainty Threshold Filtering (UTF) first removes unreliable generalized-class assignments, after which a conditional generative model learns an inverse mapping from retained GC to target-style inputs. In **Step 2**, the model parameters are updated using the resulting input–GC pairs.

3.3 Step 1: Generative Inverse Mapping

With the pre-trained model fixed, we enable its Dropout layers during inference to obtain stochastic predictions N_ω^*, where ω indexes the random forward pass. Distances in the continuous output space are measured by

$$d_y(\boldsymbol{a}, \boldsymbol{b}) = \frac{1}{L_{\text{out}}} \|\boldsymbol{a} - \boldsymbol{b}\|_2^2. \tag{1}$$

The diagnostic results in Table 14 show that high predictive uncertainty is associated with less reliable generalized-class assignments. We therefore introduce Uncertainty Threshold Filtering before generator training. For each target input, we approximate the predictive mean and uncertainty with K stochastic forward passes as

$$\bar{\boldsymbol{y}}(\boldsymbol{x}) = \mathbb{E}_\omega N^*_\omega(\boldsymbol{x}), \qquad u(\boldsymbol{x}) = \mathbb{E}_\omega d_y(N^*_\omega(\boldsymbol{x}), \bar{\boldsymbol{y}}(\boldsymbol{x}))\,. \tag{2}$$

With uncertainty threshold q_γ, UTF retains the set and conditional target distribution

$$A_\gamma = \{\boldsymbol{x} : u(\boldsymbol{x}) \le q_\gamma\}, \qquad P^\gamma_{\mathcal{T}} = P_{\mathcal{T}}(\cdot \mid A_\gamma), \tag{3}$$

where $\gamma = P_{\mathcal{T}}(A_\gamma)$ is the retained proportion. For models without Dropout, including the Task C speech-enhancement model, we use Test-Time Augmentation (TTA) (Shanmugam et al., 2020) and inject Gaussian noise into $\boldsymbol{x}_{\mathcal{T}}$ to obtain the stochastic predictions. The theoretical relation between UTF and the generalized-class radius is given in Section A.2.

For each retained target input $\boldsymbol{x}^i_{\mathcal{T}}$, the prediction $\bar{\boldsymbol{y}}_i = N^*(\boldsymbol{x}^i_{\mathcal{T}})$ serves as a *generalized-class identifier*. The corresponding generalized class contains target inputs whose pre-trained predictions lie within τ of this identifier:

$$\mathcal{C}_\tau(\bar{\boldsymbol{y}}_i) = \{\boldsymbol{x} \in \mathcal{X}_{\mathcal{T}} : d_y(N^*(\boldsymbol{x}), \bar{\boldsymbol{y}}_i) \le \tau\}\,, \tag{4}$$

where $\tau > 0$ is a normalized squared output tolerance corresponding to the Euclidean output radius $\sqrt{L_{\text{out}}\tau}$. Given a latent variable sampled from a standard Gaussian distribution $\boldsymbol{z} \sim P_{\mathcal{Z}}$, the conditional generator produces an associated target-style input $\boldsymbol{x}_{\mathcal{G}} = G(\boldsymbol{z} \mid \bar{\boldsymbol{y}})$. The semantic consistency loss softly encourages the generated input to remain in the inverse-mapping neighborhood of its conditioning generalized class:

$$\mathcal{L}_{SEM}(G) = \mathbb{E}_{\substack{\boldsymbol{x}_{\mathcal{T}} \sim P^\gamma_{\mathcal{T}} \\ \boldsymbol{z} \sim P_{\mathcal{Z}}}} d_y(N^*(G(\boldsymbol{z} \mid \bar{\boldsymbol{y}}(\boldsymbol{x}_{\mathcal{T}}))), \bar{\boldsymbol{y}}(\boldsymbol{x}_{\mathcal{T}}))\,. \tag{5}$$

The framework permits different conditional generator families, as evaluated in Section C.2. For the main experiments, we instantiate G with Conditional MeanFlow (CMF) (Geng et al., 2026). For a target input $\boldsymbol{x}_{\mathcal{T}}$ and Gaussian noise $\boldsymbol{\epsilon}$ of the same dimension, the interpolation state and velocity are

$$\boldsymbol{z}_t = (1-t)\boldsymbol{x}_{\mathcal{T}} + t\boldsymbol{\epsilon}, \qquad \boldsymbol{v} = \boldsymbol{\epsilon} - \boldsymbol{x}_{\mathcal{T}}. \tag{6}$$

The velocity network $u_\phi(\boldsymbol{z}_t, r, t, \bar{\boldsymbol{y}})$, parameterized by ϕ, receives four channels formed by $\boldsymbol{z}_t$, the length-aligned condition $\widetilde{\boldsymbol{y}}$, $t\mathbf{1}$, and $(t-r)\mathbf{1}$. The scalars r and t denote the starting and ending times of a sampled MeanFlow interval. The directional derivative and detached MeanFlow target are

$$\dot{\boldsymbol{u}} = \frac{\partial u_\phi}{\partial \boldsymbol{z}_t}\boldsymbol{v} + \frac{\partial u_\phi}{\partial t}, \qquad \boldsymbol{u}_{\text{tgt}} = \text{stopgrad}[\boldsymbol{v} - (t-r)\dot{\boldsymbol{u}}]\,, \tag{7}$$

where $\boldsymbol{u}_{\text{tgt}}$ is the detached target velocity used to train the MeanFlow velocity network. The implemented adaptive MeanFlow loss is

$$\mathcal{L}_{MF} = \frac{1}{B}\sum_{i=1}^{B} \frac{\ell_i}{\text{stopgrad}[(\ell_i + \epsilon_{MF})^p]}, \qquad \ell_i = \left\| u_\phi(\boldsymbol{z}^i_t, r_i, t_i, \bar{\boldsymbol{y}}_i) - \boldsymbol{u}^i_{\text{tgt}} \right\|_2^2, \tag{8}$$

where B is the minibatch size, $p = 1$, $\epsilon_{MF} = 0.01$, and stopgrad prevents gradients through the adaptive denominator. Starting from $\boldsymbol{z}_0 \sim P_{\mathcal{Z}}$, CMF generates an input through four reverse Euler steps,

$$\boldsymbol{z}_{k+1} = \boldsymbol{z}_k - (t_k - r_k)u_\phi(\boldsymbol{z}_k, r_k, t_k, \bar{\boldsymbol{y}}), \qquad k = 0, \ldots, 3, \tag{9}$$

and returns $G_\phi(\boldsymbol{z}_0 \mid \bar{\boldsymbol{y}}) = \boldsymbol{z}_4$.

The Step 1 objective comprises semantic consistency and a generator-specific training loss:

$$\mathcal{L}_G = \alpha_{SEM}\mathcal{L}_{SEM} + \mathcal{L}_{GEN}, \tag{10}$$

where α_{SEM} is the semantic-consistency weight and $\mathcal{L}_{GEN}$ trains the selected generator to model the target-input distribution. For CMF, the generator-specific loss is

$$\mathcal{L}_{GEN} = \mathcal{L}_{MF} + \alpha_{END}\mathcal{L}_{END} + \alpha_{TV}\mathcal{L}_{TV}. \tag{11}$$

The MeanFlow loss $\mathcal{L}_{MF}$ learns the conditional velocity field, $\mathcal{L}_{END}$ matches the generated endpoint with its paired target input, and $\mathcal{L}_{TV}$ aligns their local-variation statistics. The complete network, losses, weight schedule, and sampling procedure are reported in Section B.3.

3.4 Step 2: Adaptation

In Step 2, we fix the generator G and update the regression model N, keeping the initial identifier $\bar{\boldsymbol{y}}(\boldsymbol{x}_{\mathcal{T}}) = N^*(\boldsymbol{x}_{\mathcal{T}})$ fixed for each retained input. Conditioning G on the current prediction $N(\boldsymbol{x}_{\mathcal{T}})$ yields an input paired with this identifier. We use these input–GC pairs in the adaptation loss

$$\mathcal{L}_{GC}(N) = \mathbb{E}_{\substack{\boldsymbol{x}_{\mathcal{T}} \sim P_{\mathcal{T}}^{\gamma} \\ \boldsymbol{z} \sim P_{\mathcal{Z}}}} d_y(N(G(\boldsymbol{z} \mid N(\boldsymbol{x}_{\mathcal{T}}))), \bar{\boldsymbol{y}}(\boldsymbol{x}_{\mathcal{T}})). \tag{12}$$

The inner and outer occurrences of N denote the same adapting model. Although the generator parameters are fixed, the generator remains in the computational graph, so gradients propagate through the outer model, the generator input, and the inner prediction. This construction supplies adaptation gradients even though target labels are unavailable. We additionally use the weight regularization loss $\mathcal{L}_{REG}$ and scale constraint loss $\mathcal{L}_{SCC}$ to stabilize the update. The Step 2 objective is

$$\mathcal{L}_A = \alpha_{GC}\mathcal{L}_{GC} + \alpha_{REG}\mathcal{L}_{REG} + \alpha_{SCC}\mathcal{L}_{SCC}, \tag{13}$$

where α_{GC}, α_{REG}, and α_{SCC} denote the corresponding weighting coefficients.

3.5 Weight Regularization

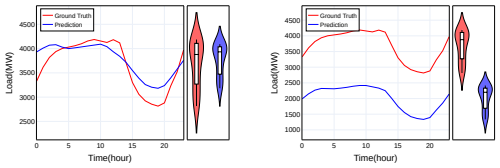
The ablation results in Table 4 show that adaptation with unreliable input–GC pairs can cause parameter drift and degrade target-domain performance. We introduce a weight regularization loss to constrain this drift:

$$\mathcal{L}_{REG}(N) = \frac{1}{n_\theta} \|\boldsymbol{\theta}_n - \boldsymbol{\theta}_n^*\|_2^2, \tag{14}$$

where $\boldsymbol{\theta}_n^*$ and $\boldsymbol{\theta}_n$ denote the model parameters before and during adaptation, and n_θ is the number of model parameters. This negative-feedback constraint limits excessive deviation from the initial model.

3.6 Feature Scale Constraints

We observed that adaptation can introduce a constant offset between the output and the ground truth, as illustrated in Figure 3b. This phenomenon occurs because the output of regression tasks relies heavily on the absolute magnitude of the feature matrix (Chen et al., 2021). During the adaptation phase driven by synthesized samples, the scale of the feature matrix may be inadvertently altered, causing a deviation in the overall prediction magnitude. We therefore introduce a scale constraint loss that regularizes the feature matrix scale through its Frobenius norm:



(a) (b)

Figure 3: Examples of model outputs for Task #4 before and after adaptation (without scale constraint loss). (a) Zero-Shot. (b) After adaptation.

$$\mathcal{L}_{SCC}(B) = \mathbb{E}_{\boldsymbol{x}_{\mathcal{T}} \sim P_{\mathcal{T}}^{\gamma}} \Big[\|B(\boldsymbol{x}_{\mathcal{T}})\|_F - \|B^*(\boldsymbol{x}_{\mathcal{T}})\|_F \Big]^2, \tag{15}$$

where B and B^* denote the backbones of the adapting and initial models, respectively. This loss constrains the feature-matrix scale during adaptation and mitigates shifts in the output statistics.

3.7 Theorem Analysis

Theorem 3.1 (GIRA target-risk bound)**.** *Let h be the adapting model, let $f_{\mathcal{T}}$ be the target regression function, and let the normalized squared loss d_y be bounded by M on the output range under consideration. The target risk $\mathcal{L}_{\mathcal{T}}(h, f_{\mathcal{T}})$ satisfies*

$$\boxed{\mathcal{L}_{\mathcal{T}}(h, f_{\mathcal{T}}) \le \gamma \big[2\mathcal{R}_G(h) + \mathrm{disc}_{\mathcal{L}}(P_G^h, P_{\mathcal{T}}^{\gamma}) + 2\lambda_h \big] + (1-\gamma)M,} \tag{16}$$

where $\gamma = P_{\mathcal{T}}(A_\gamma) > 0$ is the retained probability, $\mathcal{R}_G(h) = \mathcal{L}_{GC}$ is the generated-pair fitting risk, $\mathrm{disc}_{\mathcal{L}}(P_G^h, P_{\mathcal{T}}^{\gamma})$ is the generated–target discrepancy, λ_h is the generated-pair generalized-class error, and $(1-\gamma)M$ is the filtering coverage cost.

The proof, the expansion into the GIRA objectives, and the relation between UTF and τ are provided in Section A.2.

Table 1: Results of Task A: Non-Intrusive Load Monitoring. Best and second-best results among adapted UDA and SFDA methods are shown in bold and underlined, respectively. "Improvement" denotes the relative gain of GIRA compared to the zero-shot model. "-" denotes that method failed to converge during training.

Method	T#1: D→U			T#2: U→I			T#3: I→D			MAE.avr↓
	MAE↓	SAE↓	F1↑	MAE↓	SAE↓	F1↑	MAE↓	SAE↓	F1↑	
Zero-Shot	36.57	27.44	0.52	45.13	38.87	0.40	38.35	34.66	0.10	40.02
DeepCORAL	30.50±1.20	20.22±0.17	0.44±0.00	47.39±1.84	40.53±1.23	0.39±0.01	35.39±0.22	35.06±0.13	0.13±0.00	37.76
DANN	28.87±1.53	18.70±1.10	0.50±0.01	43.60±1.30	36.70±1.60	0.40±0.00	34.13±0.12	18.73±0.20	0.30±0.00	35.53
MDD	29.30±1.40	19.33±0.26	0.51±0.00	43.36±1.70	36.38±1.41	0.40±0.01	34.27±0.06	19.32±0.53	0.28±0.00	35.64
MMDA	30.16±1.20	20.16±0.11	0.46±0.01	48.87±1.63	42.18±0.33	0.38±0.00	33.75±0.22	12.31±0.12	0.05±0.00	37.59
CDAN	27.20±0.50	19.05±0.91	0.65±0.00	36.53±1.12	30.97±0.01	0.48±0.01	36.31±0.04	**8.75**±0.13	0.35±0.00	33.35
RSD	-	-	-	44.10±2.30	37.33±0.29	0.41±0.00	34.91±0.28	20.07±0.09	0.19±0.01	—
SHOT	64.34±3.45	53.06±3.11	0.13±0.01	78.52±2.13	65.57±0.64	0.15±0.02	36.14±0.08	34.30±1.50	0.05±0.00	59.67
NRC	48.56±0.52	41.19±1.39	0.02±0.00	53.91±2.29	47.73±0.25	0.31±0.02	35.05±0.20	30.34±0.65	0.06±0.00	45.84
AaD	60.74±2.47	54.12±1.13	0.17±0.00	60.76±1.40	52.03±2.82	0.17±0.00	35.22±0.00	15.48±0.10	0.10±0.00	52.24
SSA	35.23±0.53	26.98±0.88	0.57±0.02	36.98±0.47	28.83±0.49	0.43±0.01	33.54±0.25	17.23±0.20	0.26±0.01	35.25
MAPU	37.35±2.28	37.34±0.63	0.01±0.00	40.57±0.32	34.44±0.40	0.33±0.00	33.71±0.03	13.04±0.24	0.01±0.00	37.21
SFHarmony	36.27±1.04	28.01±1.84	0.04±0.00	34.26±0.17	23.72±1.13	0.44±0.00	33.49±0.34	10.47±0.03	0.20±0.00	34.67
GIRA (Ours)	**25.45**±0.21	**17.37**±0.43	**0.67**±0.00	**29.93**±0.72	**23.45**±0.23	**0.51**±0.00	**32.34**±0.09	11.03±0.01	**0.37**±0.00	**29.24**
Improvement	*30.41%*	*36.70%*	*28.85%*	*33.68%*	*39.67%*	*27.50%*	*15.67%*	*68.18%*	*270.00%*	*26.93%*

Table 2: Results of Task B: Electric Load Forecasting.

Method	T#4: N→G		T#5: G→P		T#6: P→N		MAE.avr↓
	MAE↓	RMSE↓	MAE↓	RMSE↓	MAE↓	RMSE↓	
Zero-Shot	498.13	584.75	130.70	153.73	161.10	208.07	263.31
DeepCORAL	458.71±3.99	540.58±3.86	103.90±0.10	134.43±1.11	153.19±0.46	192.37±1.11	238.60
DANN	454.25±5.40	537.13±5.67	110.02±0.89	143.27±0.17	145.02±3.41	181.44±4.11	236.43
MDD	454.30±1.30	537.12±1.57	111.02±0.56	144.45±1.58	144.37±3.14	180.72±2.12	236.56
MMDA	451.11±1.06	533.55±0.76	104.46±0.71	135.20±0.40	143.73±2.47	180.63±3.17	233.10
CDAN	279.81±1.55	341.00±1.00	109.84±0.65	141.25±0.08	147.31±0.73	185.27±2.86	178.99
RSD	456.85±4.22	539.59±5.77	111.36±0.16	144.53±1.95	144.24±2.99	180.54±1.05	237.48
SHOT	693.38±11.72	829.55±4.91	247.58±1.70	318.35±0.42	157.02±1.98	232.03±0.04	365.99
NRC	484.28±1.51	563.89±1.10	228.19±0.22	295.67±3.71	157.76±3.31	199.20±0.80	290.08
AaD	404.19±2.06	528.45±1.19	196.75±0.03	273.20±3.50	149.97±2.88	205.78±3.65	250.30
SSA	305.25±1.28	381.93±1.48	114.64±0.50	156.52±0.63	138.24±1.31	179.85±0.86	186.04
MAPU	388.84±1.34	440.19±2.78	163.17±0.14	209.55±2.14	153.01±0.36	187.04±2.06	235.01
SFHarmony	327.03±3.36	403.74±2.61	107.10±0.60	138.51±1.18	145.83±1.07	182.28±1.32	193.32
GIRA (Ours)	**259.22**±1.67	**334.96**±1.89	**94.07**±0.20	**125.05**±0.55	**131.73**±0.69	**178.24**±0.50	**161.67**
Improvement	*47.96%*	*42.72%*	*28.03%*	*18.66%*	*18.23%*	*14.34%*	*38.60%*

4 EXPERIMENTS

To validate the efficacy of GIRA on sequence regression tasks, we selected three tasks: non-intrusive load monitoring, electric load forecasting, and speech enhancement. These tasks represent three typical scenarios characterized by low-frequency step signals, low-frequency periodic signals, and high-frequency signals, respectively.

4.1 TASKS & SETTINGS

A. Non-Intrusive Load Monitoring. This task aims to extract the power consumption sequence of a target appliance from the total household load sequence. We chose three public datasets for this task: U.K.DALE (**U**) (Kelly & Knottenbelt, 2015), REFIT (**I**) (Murray et al., 2017), and Pecan Street Dataport (**D**) (Pecan Street Inc., 2025), and selected the fridge as the target appliance. We employed a ResNet-seq2seq architecture (Zhang et al., 2023) as the pre-trained model. Besides MAE, we adopted Signal Aggregate Error (SAE) and F1-score as evaluation metrics. SAE measures the total error over a period and is defined as $SAE = \frac{1}{L_{\text{out}}} \left| \sum_{i=1}^{L_{\text{out}}} \hat{y}_i - \sum_{i=1}^{L_{\text{out}}} y_i \right|$. F1 measures on/off-state detection performance.

B. Electric Load Forecasting. This task aims to forecast future electricity demand based on historical load data. We picked three public datasets for this task: NYISO Actual Load (**N**) (Operator, 2026), GEFCom2014 (**G**) (Hong et al., 2016), and Panama (**P**) (Madrid, 2021). Similar to NILM, this task

Table 3: Results of Task C: Speech Enhancement. For SegSNR, the improvement is reported as an absolute gain in dB.

Method	T#7: V+O→A+M		T#8: A+O→J+M		T#9: J+O→V+M		SegSNR.avr↑
	MAE(10^{-3})↓	SegSNR(dB)↑	MAE(10^{-3})↓	SegSNR(dB)↑	MAE(10^{-3})↓	SegSNR(dB)↑	
Zero-Shot	7.55	12.53	10.82	13.38	10.84	13.22	13.04
DeepCORAL	7.24±0.03	12.70±0.00	10.73±0.02	13.50±0.00	10.79±0.00	13.27±0.00	13.16
DANN	7.15±0.03	13.04±0.00	10.50±0.00	13.74±0.00	10.60±0.01	13.44±0.01	13.41
MDD	7.11±0.04	13.15±0.01	10.47±0.03	13.77±0.01	10.62±0.02	13.38±0.00	13.43
MMDA	7.33±0.03	12.74±0.01	10.79±0.05	13.39±0.01	10.83±0.06	13.23±0.01	13.12
CDAN	7.23±0.01	12.88±0.01	10.75±0.03	13.48±0.00	**10.57**±0.01	**13.47**±0.00	13.28
RSD	7.20±0.02	12.83±0.00	10.50±0.00	13.70±0.01	10.88±0.01	13.24±0.00	13.26
SHOT	7.53±0.04	12.61±0.01	11.34±0.09	12.86±0.02	11.12±0.01	12.80±0.02	12.76
NRC	7.53±0.03	12.60±0.02	11.07±0.05	13.13±0.01	11.04±0.03	13.01±0.00	12.91
AaD	7.36±0.01	12.94±0.02	10.94±0.01	13.35±0.01	10.82±0.00	13.18±0.00	13.16
SSA	7.14±0.01	13.05±0.00	10.84±0.02	13.39±0.00	10.63±0.01	13.46±0.01	13.30
MAPU	7.48±0.01	12.73±0.00	10.77±0.03	13.48±0.00	10.73±0.03	13.32±0.00	13.18
SFHarmony	7.09±0.01	13.16±0.00	10.61±0.00	13.69±0.00	10.68±0.01	13.43±0.00	13.43
GIRA (Ours)	**6.92**±0.02	**13.40**±0.00	**10.45**±0.02	**13.79**±0.00	10.61±0.01	13.46±0.01	**13.55**
Improvement	*8.34%*	*0.87*	*3.42%*	*0.41*	*2.12%*	*0.24*	*0.51*

employed the ResNet-seq2seq as the pre-trained model. We used MAE and RMSE as the evaluation metrics.

C. Speech Enhancement. This task aims to recover clean speech from noisy signals, thereby improving quality and intelligibility. We employed three clean speech datasets: VCTK (**V**) (Yamagishi et al., 2019), AISHELL-3 (**A**) (Shi et al., 2020), and JSUT (**J**) (Sonobe et al., 2017), in conjunction with two noise datasets: NOISEX-92 (**O**) (Varga et al., 1993) and MUSAN (**M**) (Snyder et al., 2015). To exacerbate domain shifts, clean speech was mixed exclusively with **O** for the source domain and with **M** for the target domain. We adopted the Demucs denoiser (Defossez et al., 2020) as the pre-trained model. Besides MAE, we utilized Segmental SNR (SegSNR) as an evaluation metric. SegSNR calculates the average signal-to-noise ratio over short frames to better reflect the perceptual quality of non-stationary speech signals, defined as $SegSNR = \frac{1}{M}\sum_{m=1}^{M} SNR_m$.

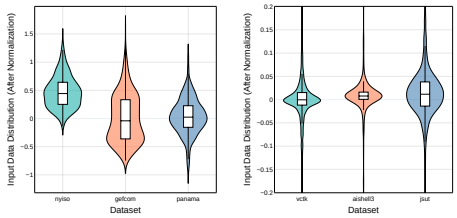
Detailed experimental settings are provided in Section B.

4.2 Baselines

Due to the limited availability of open-source SFDA frameworks for sequence regression, we benchmark against compatible methods from related fields. While we strive to ensure a fair comparison, these baselines may not achieve theoretically optimal performance, as they are not originally designed for sequence regression. For UDA, we compare GIRA with DeepCORAL (Sun & Saenko, 2016), DANN (Ganin et al., 2016), MDD (Zhang et al., 2019), MMDA (Rahman et al., 2020), CDAN (Long et al., 2018), and RSD (Chen et al., 2021). For SFDA, we include SHOT (Liang et al., 2020), NRC (Yang et al., 2021), AaD (Yang et al., 2022), SSA (Adachi et al., 2025), MAPU (Ragab et al., 2023), and SFHarmony (Dinsdale et al., 2023). The classification objectives of SHOT, NRC, and AaD do not directly apply to continuous outputs. Their regression implementations retain applicable method-specific mechanisms and use a prediction-variance consistency loss across Gaussian-perturbed target inputs. MAPU also retains feature reconstruction and uses this variance consistency loss, while SSA uses regression-specific subspace alignment.

4.3 Experimental Results

Results of Non-Intrusive Load Monitoring. For this task, we designed three transfer cases: **D**→**U**, **U**→**I**, and **I**→**D**. Table 1 presents the performance of GIRA compared to previous works across three evaluation metrics. In this case, GIRA consistently achieves performance gains over the pre-trained model and secures the best results in 8 out of 9 metrics. In particular, in case T#2, GIRA improves



(a) (b)

Figure 4: Data distribution of (a) Task B and (b) Task C (after normalization).

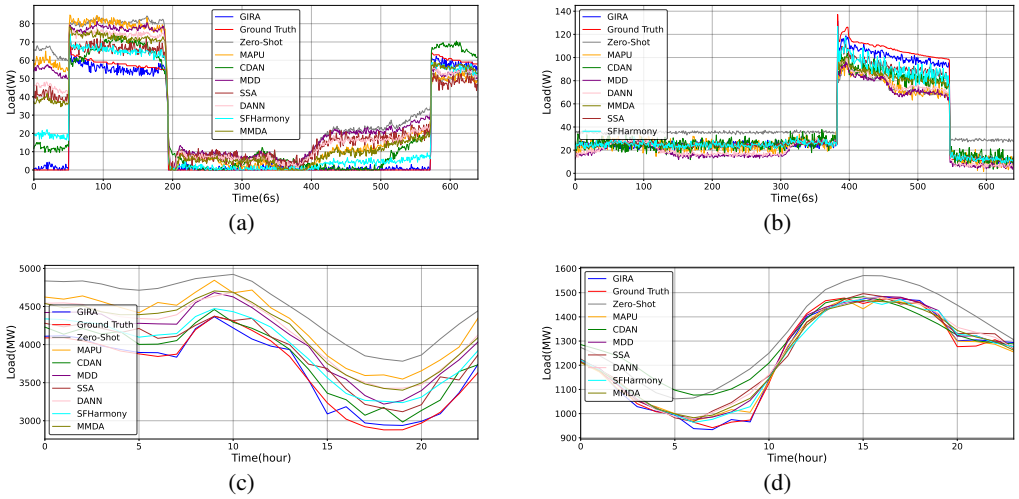


Figure 5: Example results for Tasks A and B. (a) Task A, T#2. (b) Task A, T#3. (c) Task B, T#4. (d) Task B, T#5.

the MAE by 33.68% compared to the pre-trained model and by 12.64% compared to the runner-up, SFHarmony. Overall, GIRA achieves an average MAE of 29.24 across the three cases, outperforming all UDA methods that utilize source domain data.

Results of Electric Load Forecasting. For this task, we designed three transfer cases: **N**→**G**, **G**→**P**, and **P**→**N**. Table 2 displays the experimental results. GIRA achieves the best performance among the compared methods across all evaluation metrics. In case T#4, the model delivers an impressive 47.96% improvement in MAE. With an average MAE of 161.67 across the three cases, the proposed method outperforms all comparative approaches.

Results of Speech Enhancement. For this task, we designed three cases: **V+O**→**A+M**, **A+O**→**J+M**, and **J+O**→**V+M**, and the results are summarized in Table 3. GIRA outperforms all comparative methods in cases T#7 and T#8. Across the three cases, GIRA achieves an average SegSNR of 13.55, getting an average improvement of 0.51 dB in speech quality relative to the pre-trained model. Notably in case T#9, the performance of GIRA and SFHarmony lags behind some UDA-based methods. The large discrepancy between **J** and **V** (as shown in Figure 4b) may make the basic-performance condition in Appendix Assumption A.3 less reliable. Conversely, UDA methods can explicitly align the feature spaces, thereby mitigating the impact of such large domain shifts.

Furthermore, we evaluated GIRA on various pre-trained architectures, with the results provided in Section C.1.

4.4 Visualization

To provide a more intuitive understanding of the proposed method's performance, we selected some representative samples for visual analysis.

Figures 5a and 5b show that GIRA follows the ground-truth appliance activity more closely and maintains a stable near-zero output during inactive periods. Figures 5c and 5d show that GIRA best follows the waveform trends, while $\mathcal{L}_{SCC}$ helps maintain feature-scale consistency during adaptation. This constraint enables GIRA to achieve lower mean errors compared to the baseline methods in Figure 5c.

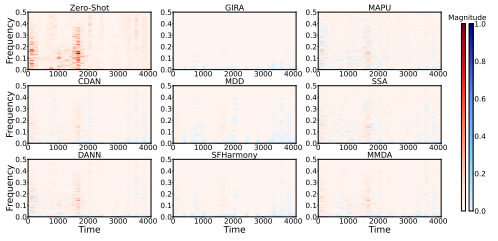


Figure 6: Differential spectrograms of denoising results for Task C. Red shows the magnitude of the difference from clean speech, while blue highlights regions with greater error than in the zero-shot output. The sample is selected from case T#7.

Table 4: Ablation study results on T#1, T#4, and T#5.

Methods	T#1: D→U			T#4: N→G		T#5: G→P	
	MAE	SAE	F1	MAE	RMSE	MAE	RMSE
Zero-Shot	36.57	27.44	0.52	498.13	584.75	130.70	153.73
$\mathcal{L}_{GC}$ Only	35.14	26.10	0.48	1603.67	1764.16	135.83	159.15
w/o $\mathcal{L}_{REG}$	32.08	24.62	0.56	362.85	537.88	126.31	146.15
w/o $\mathcal{L}_{SCC}$	45.10	32.49	0.55	1592.41	1777.05	124.40	149.68
w/o UTF	30.17	24.15	0.52	313.12	430.14	98.13	131.97
GIRA (Ours)	**25.45**	**17.37**	**0.67**	**259.22**	**334.96**	**94.07**	**125.05**

Table 5: Relationship between CMF training quality and model performance on T#4. "Improvement" denotes the relative gain compared to the zero-shot model.

Training step	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
Semantic MAE	1625.55	1219.45	603.11	458.16	489.10	229.92	144.99	131.39	118.45	100.97
Semantic RMSE	1678.99	1268.31	704.37	562.33	579.60	287.82	186.60	166.50	155.60	131.62
MMD^2	4.13	3.92	3.84	1.55	0.13	0.09	0.06	0.05	0.05	0.05
Score↑	-206.527	-163.588	-103.774	-66.5245	-54.735	-26.787	-17.1795	-15.3945	-14.2025	-12.1295
MAE	603.98	564.09	355.80	319.04	300.48	278.43	268.00	262.81	260.62	259.22
Imp.	-21.25%	-13.24%	28.57%	35.95%	39.68%	44.10%	46.20%	47.24%	47.68%	47.96%

Figure 6 displays the differential spectrograms of the speech enhancement results for Task C. This visualization quantifies the spectral distribution of the difference between the enhanced signal and clean signal, where smaller red spots indicate stronger denoising performance and blue spots represent regions with increased error relative to zero-shot. It can be observed that while models such as MAPU and SSA mitigate noise, they inadvertently introduce artifacts (blue spots) in other regions. Additionally, most models fail to effectively suppress the noise burst around $t = 1500$. In contrast, the proposed method reduces these high-intensity error regions to a nearly invisible level while introducing minimum artifacts.

4.5 Ablation Study

To further investigate the contribution of each component within the proposed framework, we conducted ablation studies on T#1, T#4, and T#5. As shown in Table 4, removing either $\mathcal{L}_{REG}$ or UTF degrades performance relative to the full GIRA model. In T#5, adding $\mathcal{L}_{REG}$ reduces MAE from 126.31 to 94.07. Removing $\mathcal{L}_{SCC}$ causes a severe degradation on T#4 but a smaller decline on T#5. The larger discrepancy between **N** and **G** in T#4, shown in Figure 4a, makes feature-scale control more important.

Furthermore, we analyzed the impact of CMF quality on performance by varying training iterations. The results are presented in Table 5. We quantify the quality of the generated samples using the Score in Equation (75), which combines semantic matching and generated–target distribution discrepancy. At 1000 and 2000 training steps, the adapted MAE exceeds the pre-adaptation MAE, indicating negative transfer. Beyond 2000 steps, the adapted MAE decreases as generator quality improves. The gains become smaller at later steps, with MAE decreasing from 262.81 at 8000 steps to 259.22 at 10000 steps.

5 Conclusion

We presented Generative Inverse Regression Adaptation, a source-free domain adaptation framework for sequence regression tasks. GIRA constructs target-style input–GC pairs through generalized classes and generative inverse mapping to support adaptation without target labels. Across nine transfer scenarios, GIRA reduces MAE by 20.87% on average relative to the corresponding pre-adaptation models and achieves the best task-average result among the compared source-free methods on all three tasks. Future work will focus on robust adaptation under larger domain shifts and less reliable source-model predictions.

AI USE STATEMENT

Large language models were used to assist with literature search tasks, code implementation, and manuscript language polishing. The authors independently reviewed and verified the cited sources, scientific claims, technical details, experimental protocols, and reported results.

REPRODUCIBILITY STATEMENT

The manuscript specifies the method and optimization objectives in the main text, and provides the baseline settings, conditional-generator configurations, task preprocessing, evaluation protocols, and computational environment in Sections B.2 to B.5.

REFERENCES

Kazuki Adachi, Shin'ya Yamaguchi, Atsutoshi Kumagai, and Tomoki Hamagami. Test-time adaptation for regression by subspace alignment. In *International Conference on Learning Representations*, volume 2025, pp. 77850–77874, 2025.

Yudong Cao, Jichao Zhuang, Qiuhua Miao, Minping Jia, Ke Feng, Xiaoli Zhao, Xiaoan Yan, and Peng Ding. Source-free domain adaptation for transferable remaining useful life prediction of machine considering source data absence. *Reliability Engineering & System Safety*, 246:110079, 2024.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. volume 31, 2018.

Xinyang Chen, Sinan Wang, Jianmin Wang, and Mingsheng Long. Representation subspace distance for domain adaptation regression. In *ICML*, pp. 1749–1759, 2021.

Zhuohang Chen, Jinglong Chen, Tongyang Pan, and Jingsong Xie. Uncertainty estimation pseudo-label-guided source-free domain adaptation for cross-domain remaining useful life prediction in iiot. *IEEE Internet of Things Journal*, 12(1):236–249, 2025. doi: 10.1109/JIOT.2024.3464854.

Alexandre Defossez, Gabriel Synnaeve, and Yossi Adi. Real time speech enhancement in the waveform domain. *arXiv preprint arXiv:2006.12847*, 2020.

Nicola K Dinsdale, Mark Jenkinson, and Ana IL Namburete. Sfharmony: Source free domain adaptation for distributed neuroimaging analysis. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11460–11471. IEEE, 2023.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of machine learning research*, 17(59):1–35, 2016.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. volume 38, pp. 75460–75482, 2026.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The journal of machine learning research*, 13:723–773, 2012.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Tianlang He, Zhiqiu Xia, Jierun Chen, Haoliang Li, and S-H Gary Chan. Target-agnostic source-free domain adaptation for regression tasks. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pp. 1464–1477. IEEE, 2024.

Tao Hong, Pierre Pinson, Shu Fan, Hamidreza Zareipour, Alberto Troccoli, and Rob J Hyndman. Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond, 2016.

Yubo Hou, Yucheng Wang, Min Wu, Chee-Keong Kwoh, Xiaoli Li, and Zhenghua Chen. Source-free calibrated uncertainty for rul adaptation with incomplete degradation. *Mechanical Systems and Signal Processing*, 233:112718, 2025. doi: 10.1016/j.ymssp.2025.112718.

Jack Kelly and William Knottenbelt. The uk-dale dataset, domestic appliance-level electricity demand and whole-house demand from five uk homes. *Scientific data*, 2(1):150007, 2015.

Dongmin Kim, Sunghyun Park, and Jaegul Choo. When model meets new normals: Test-time adaptation for unsupervised time-series anomaly detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 13113–13121, 2024.

Vinod K Kurmi, Venkatesh K Subramanian, and Vinay P Namboodiri. Domain impression: A source data free domain adaptation method. In *2021 IEEE Winter conference on applications of computer vision (WACV)*, pp. 615–625. IEEE, 2021.

JoonHo Lee and Gyemin Lee. Feature alignment by uncertainty and self-training for source-free unsupervised domain adaptation. *Neural Networks*, 161:682–692, 2023.

Rui Li, Qianfen Jiao, Wenming Cao, Hau-San Wong, and Si Wu. Model Adaptation: Unsupervised Domain Adaptation Without Source Data. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9638–9647, 2020.

Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International conference on machine learning*, pp. 6028–6039. PMLR, 2020.

Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical transactions of the royal society a: mathematical, physical and engineering sciences*, 379(2194), 2021.

Jun Lin, Jin Ma, Jianguo Zhu, and Huishi Liang. Deep domain adaptation for non-intrusive load monitoring based on a knowledge transfer learning network. *IEEE Transactions on Smart Grid*, 13 (1):280–292, 2022.

Yinyan Liu, Li Zhong, Jing Qiu, Junda Lu, and Wei Wang. Unsupervised domain adaptation for nonintrusive load monitoring via adversarial and joint adaptation network. *IEEE Transactions on Industrial Informatics*, 18(1):266–277, 2022.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pp. 97–105. PMLR, 2015.

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *International conference on machine learning*, pp. 2208–2217. PMLR, 2017.

Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael Jordan. Conditional adversarial domain adaptation. *Advances in neural information processing systems*, 31, 2018.

A Madrid. Short-term electricity load forecasting (panama case study). *Mendeley Data*, 1:2021, 2021.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

David Murray, Lina Stankovic, and Vladimir Stankovic. An electrical load measurements dataset of united kingdom households from a two-year longitudinal study. *Scientific data*, 4(1):160122, 2017.

New York Independent System Operator. Nyiso load data. `https://www.nyiso.com/load-data`, 2026. Accessed: 2026-09-25.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

Pecan Street Inc. Pecan street dataport. https://www.pecanstreet.org/dataport/, 2025. Accessed: 2026-09-25.

Zhongyi Pei, Zhangjie Cao, Mingsheng Long, and Jianmin Wang. Multi-adversarial domain adaptation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Mohamed Ragab, Emadeldeen Eldele, Min Wu, Chuan-Sheng Foo, Xiaoli Li, and Zhenghua Chen. Source-free domain adaptation with temporal imputation for time series data. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 1989–1998, 2023.

Mohammad Mahfujur Rahman, Clinton Fookes, Mahsa Baktashmotlagh, and Sridha Sridharan. On minimum discrepancy estimation for deep domain adaptation. In *Domain Adaptation for Visual Understanding*, pp. 81–94. Springer, 2020.

Tobias Raichle, Niels Edinger, and Bin Yang. Test-time adaptation for speech enhancement via domain invariant embedding transformation. *IEEE Open Journal of Signal Processing*, 7:134–143, 2026. doi: 10.1109/OJSP.2026.3656059.

Divya Shanmugam, Davis Blalock, Guha Balakrishnan, and John Guttag. When and why test-time augmentation works. *arXiv preprint arXiv:2011.11156*, 1(3):4, 2020.

Yao Shi, Hui Bu, Xin Xu, Shaoji Zhang, and Ming Li. Aishell-3: A multi-speaker mandarin tts corpus and the baselines. *arXiv preprint arXiv:2010.11567*, 2020.

David Snyder, Guoguo Chen, and Daniel Povey. Musan: A music, speech, and noise corpus. *arXiv preprint arXiv:1510.08484*, 2015.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

Ryosuke Sonobe, Shinnosuke Takamichi, and Hiroshi Saruwatari. Jsut corpus: free large-scale japanese speech corpus for end-to-end speech synthesis. *arXiv preprint arXiv:1711.00354*, 2017.

Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision*, pp. 443–450. Springer, 2016.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 2962–2971. Ieee, 2017.

A Varga, HJM Steeneken, et al. Noisex-92: A database and an experiment to study the effect of additive noise on speech recognition systems. *Speech Commun*, 12(3):247–253, 1993.

DeLiang Wang and Jitong Chen. Supervised speech separation based on deep learning: An overview. *IEEE/ACM transactions on audio, speech, and language processing*, 26(10):1702–1726, 2018.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=uXl3bZLkr3c.

Garrett Wilson and Diane J Cook. A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(5):1–46, 2020.

Junichi Yamagishi, Christophe Veaux, and Kirsten MacDonald. Cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit (version 0.92). *The Rainbow Passage which the speakers read out can be found in the International Dialects of English Archive:(http://web. ku. edu/˜ idea/readings/rainbow. htm).*, 2019.

Kangjia Yan, Chenxi Liu, Hao Miao, Xinle Wu, Yan Zhao, Chenjuan Guo, and Bin Yang. Invariant representation learning for source-free time series forecasting with LLM-centric proxy denoising. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=65uC8Ac9hm.

Shiqi Yang, Joost Van de Weijer, Luis Herranz, Shangling Jui, et al. Exploiting the intrinsic neighborhood structure for source-free domain adaptation. *Advances in neural information processing systems*, 34:29393–29405, 2021.

Shiqi Yang, Shangling Jui, Joost Van De Weijer, et al. Attracting and dispersing: A simple approach for source-free domain adaptation. *Advances in neural information processing systems*, 35:5802–5815, 2022.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

Yuanshi Zhang, Wenyan Qian, Yujian Ye, Yang Li, Yi Tang, Yu Long, and Meimei Duan. A novel non-intrusive load monitoring method based on resnet-seq2seq networks for energy disaggregation of distributed energy resources integrated with residential houses. *Applied Energy*, 349:121703, 2023.

Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. Bridging theory and algorithm for domain adaptation. In *International conference on machine learning*, pp. 7404–7413. PMLR, 2019.

Feichi Zhong, Zihan Shan, Gangquan Si, Aoming Liu, Gerui Zhao, and Bo Li. Source-free domain adaptation with self-supervised learning for nonintrusive load monitoring. *IEEE Transactions on Instrumentation and Measurement*, 73:1–13, 2024. doi: 10.1109/TIM.2024.3480230.

Yingyi Zhong, Wen'an Zhou, and Liwen Tao. Source-free time series domain adaptation with wavelet-based multi-scale temporal imputation. *Neural Networks*, 188:107428, 2025.

Ahmed Zoha, Alexander Gluhak, Muhammad Ali Imran, and Sutharshan Rajasegarar. Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey. *Sensors*, 12(12):16838–16866, 2012.

A Additional Theoretical Analysis

A.1 Supporting Assumptions

The following assumptions describe the problem setting used by the proposed method and the intuitive visualization in Section A.3. They do not replace the explicit sufficient conditions of Theorem 3.1.

Assumption A.1. (Non-injective Assumption). Let $f : \mathcal{X} \to \mathcal{Y}$ denote the forward process of the target task. The mapping f is non-injective, so its inverse $f^{-1} : \mathcal{Y} \to \mathcal{X}$ is multi-valued.

GIRA uses this multi-valued inverse mapping to explore inputs associated with a generalized class. Evidently, classification tasks inherently satisfy this assumption, and the majority of sequence regression tasks—including the three selected in this study—also adhere to this condition.

Assumption A.2. (SFDA Basic Assumption). The source and target joint distributions are distinct, while their conditional distributions remain approximately invariant. That is, $P_{\mathcal{S}}(\boldsymbol{x}, \boldsymbol{y}) \neq P_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{y})$ and $P_{\mathcal{T}}(\boldsymbol{y} \mid \boldsymbol{x}) \approx P_{\mathcal{S}}(\boldsymbol{y} \mid \boldsymbol{x})$.

Assumption A.3. (Basic Performance Assumption). The pre-trained model $f(\boldsymbol{x}, \boldsymbol{\theta}^*)$ retains basic inference capability on the target domain.

A.2 Proof of Theorem 3.1

Notation and conditions. All predictors and generators are measurable on a common input space $\mathcal{X}$ containing the target and generated inputs. If $f_{\mathcal{T}}$ is specified only on $\mathcal{X}_{\mathcal{T}}$, assume that it admits a fixed extension to $\mathcal{X}$ satisfying the conditions below. The generalized class in Equation (4) remains restricted to $\mathcal{X}_{\mathcal{T}}$; generated inputs are not assumed to belong to that set. For a distribution P over $\mathcal{X}$, define

$$\mathcal{L}_P(g, g') = \mathbb{E}_{\boldsymbol{x}\sim P} d_y(g(\boldsymbol{x}), g'(\boldsymbol{x}))\,, \qquad \mathcal{L}_{\mathcal{T}} = \mathcal{L}_{P_{\mathcal{T}}}, \tag{17}$$

$$\mathrm{disc}_{\mathcal{L}}(P, Q) = \sup_{g,g'\in\mathcal{H}} |\mathcal{L}_P(g, g') - \mathcal{L}_Q(g, g')|\,. \tag{18}$$

Write $\|\boldsymbol{y}\|_y = \|\boldsymbol{y}\|_2/\sqrt{L_{out}}$, so that $d_y(\boldsymbol{a}, \boldsymbol{b}) = \|\boldsymbol{a}-\boldsymbol{b}\|_y^2$. Throughout, $N^*(\boldsymbol{x}) = \bar{\boldsymbol{y}}(\boldsymbol{x}) = \mathbb{E}_\omega N^*_\omega(\boldsymbol{x})$ denotes the fixed predictive mean in Equation (2). Distinguish it from $N_0 = N(\cdot, \boldsymbol{\theta}^*)$, evaluated using the same inference convention as $h = N(\cdot, \boldsymbol{\theta})$, and define

$$\varepsilon_{inf} = \mathbb{E}_{P^\gamma_{\mathcal{T}}} d_y(N_0(\boldsymbol{x}), N^*(\boldsymbol{x})) < \infty. \tag{19}$$

Thus $\varepsilon_{inf} = 0$ when the two reference predictors coincide; equality of their checkpoint parameters alone is not assumed to imply this. Fix a finite UTF threshold $q_\gamma \geq 0$ and let $\gamma = P_{\mathcal{T}}(A_\gamma) \in (0, 1]$ be its actual retained probability.

For independent $\boldsymbol{x} \sim P^\gamma_{\mathcal{T}}$ and $\boldsymbol{z} \sim P_{\mathcal{Z}}$, define

$$\begin{aligned} P^*_G(A) &= \Pr[G(\boldsymbol{z} \mid \bar{\boldsymbol{y}}(\boldsymbol{x})) \in A]\,, \quad P^h_G(A) = \Pr[G(\boldsymbol{z} \mid h(\boldsymbol{x})) \in A]\,, \\ D_G &= \mathrm{MMD}_k(P^*_G, P^\gamma_{\mathcal{T}}), \quad D_h = \mathrm{MMD}_k(P^h_G, P^\gamma_{\mathcal{T}}), \end{aligned} \tag{20}$$

where $A \subseteq \mathcal{X}$ is measurable. Let

$$\begin{aligned} \mathbb{E}_{\boldsymbol{x},\boldsymbol{z}}[\cdot] &:= \mathbb{E}_{\substack{\boldsymbol{x}\sim P^\gamma_{\mathcal{T}} \\ \boldsymbol{z}\sim P_{\mathcal{Z}}}}[\cdot], \\ \boldsymbol{x}^*_G &= G(\boldsymbol{z} \mid \bar{\boldsymbol{y}}(\boldsymbol{x})), \qquad \boldsymbol{x}^h_G = G(\boldsymbol{z} \mid h(\boldsymbol{x})), \\ \mathcal{R}_G(h) &= \mathcal{L}_{GC} = \mathbb{E}_{\boldsymbol{x},\boldsymbol{z}} d_y\big(h(\boldsymbol{x}^h_G), \bar{\boldsymbol{y}}(\boldsymbol{x})\big)\,, \\ \Delta_h &= \mathbb{E}_{P^\gamma_{\mathcal{T}}} d_y(h(\boldsymbol{x}), N^*(\boldsymbol{x}))\,, \\ \lambda_h &= \mathbb{E}_{\boldsymbol{x},\boldsymbol{z}} d_y\big(\bar{\boldsymbol{y}}(\boldsymbol{x}), f_{\mathcal{T}}(\boldsymbol{x}^h_G)\big)\,. \end{aligned} \tag{21}$$

These are joint-sampling expectations; no inverse function G^{-1} is required.

Let k be a bounded measurable positive-semidefinite kernel with well-defined mean embeddings in its reproducing kernel Hilbert space (RKHS) $\mathcal{H}_k$. For $\ell_{g,g'}(\boldsymbol{x}) = d_y(g(\boldsymbol{x}), g'(\boldsymbol{x}))$ and $e_*(\boldsymbol{x}) = d_y(N^*(\boldsymbol{x}), f_{\mathcal{T}}(\boldsymbol{x}))$, assume

$$\begin{aligned} &h, N^*, f_{\mathcal{T}} \in \mathcal{H}, \qquad \mathcal{H} \subseteq \{g : \mathcal{X} \to \mathcal{Y}\}, \qquad \sup_{\boldsymbol{a},\boldsymbol{b}\in\mathcal{Y}} d_y(\boldsymbol{a}, \boldsymbol{b}) \leq M, \\ &\sup_{g,g'\in\mathcal{H}} \|\ell_{g,g'}\|_{\mathcal{H}_k} \leq B_{\mathcal{H}}, \qquad \|e_*\|_{\mathcal{H}_k} \leq B_*. \end{aligned} \tag{22}$$

The finite constants $B_{\mathcal{H}}, B_*$ are RKHS norm bounds, not consequences of loss normalization. Writing $\phi_k(\boldsymbol{x}) = k(\boldsymbol{x}, \cdot)$, assume that, for the conditioning outputs under consideration and almost every $\boldsymbol{z}$,

$$\begin{aligned} \|\phi_k(G(\boldsymbol{z} \mid \boldsymbol{y}_1)) - \phi_k(G(\boldsymbol{z} \mid \boldsymbol{y}_2))\|_{\mathcal{H}_k} &\le L_{kG}\|\boldsymbol{y}_1 - \boldsymbol{y}_2\|_y, \\ \|N^*(G(\boldsymbol{z} \mid \boldsymbol{y}_1)) - N^*(G(\boldsymbol{z} \mid \boldsymbol{y}_2))\|_y &\le L_{NG}\|\boldsymbol{y}_1 - \boldsymbol{y}_2\|_y. \end{aligned} \tag{23}$$

Here L_{kG}, L_{NG} bound the sensitivity to the generator condition.

For the scale analysis, write $\boldsymbol{\theta} = (\boldsymbol{\beta}, \boldsymbol{\eta})$, $h = H_{\boldsymbol{\eta}} \circ B_{\boldsymbol{\beta}}$, and $N_0 = H_{\boldsymbol{\eta}^*} \circ B_{\boldsymbol{\beta}^*}$. The backbone output is the feature tensor regularized in Equation (15) and supplies all inputs to the head. Identify it with its vectorization $\boldsymbol{b}$, so that $\|\boldsymbol{b}\|_2 = \|B_{\boldsymbol{\beta}}(\boldsymbol{x})\|_F$. Write $\boldsymbol{b} = r\boldsymbol{v}$ and $\boldsymbol{b}^* = r^*\boldsymbol{v}^*$, with unit directions; at zero norm choose any fixed unit direction. Assume, uniformly on the retained inputs and parameter region under consideration,

$$\begin{aligned} \|H_{\boldsymbol{\eta}}(\boldsymbol{s}) - H_{\boldsymbol{\eta}}(\boldsymbol{t})\|_2 &\le L_H\|\boldsymbol{s} - \boldsymbol{t}\|_2, \\ \|H_{\boldsymbol{\eta}}(\boldsymbol{s}) - H_{\boldsymbol{\eta}^*}(\boldsymbol{s})\|_2 &\le L_\eta\|\boldsymbol{\eta} - \boldsymbol{\eta}^*\|_2, \\ rr^*\|\boldsymbol{v} - \boldsymbol{v}^*\|_2^2 &\le L_{dir}^2\|\boldsymbol{\beta} - \boldsymbol{\beta}^*\|_2^2. \end{aligned} \tag{24}$$

The first two bounds hold for all relevant head inputs $\boldsymbol{s}, \boldsymbol{t}$. The last bound controls weighted directional drift; a sufficient condition is $\|B_{\boldsymbol{\beta}}(\boldsymbol{x}) - B_{\boldsymbol{\beta}^*}(\boldsymbol{x})\|_F \le L_B\|\boldsymbol{\beta} - \boldsymbol{\beta}^*\|_2$, with $L_{dir} = L_B$, as follows from Equation (30) below. Finally, assume the error–uncertainty envelope

$$\mathbb{E}[e_*(\boldsymbol{x}) \mid u(\boldsymbol{x})] \le a\, u(\boldsymbol{x}) + b, \qquad a, b \ge 0. \tag{25}$$

This is an additional condition on the reference error, not a property implied by MC Dropout. All stability and norm bounds are finite, nonnegative, and uniform over the models being compared; comparison of generator checkpoints additionally requires common bounds in Equation (23).

Proof. The normalized squared loss satisfies

$$d_y(\boldsymbol{a}, \boldsymbol{c}) \le 2d_y(\boldsymbol{a}, \boldsymbol{b}) + 2d_y(\boldsymbol{b}, \boldsymbol{c}). \tag{26}$$

Using the indicator $\mathbf{1}_A$ of an event A,

$$\begin{aligned} \mathcal{L}_{\mathcal{T}}(h, f_{\mathcal{T}}) &= \mathbb{E}_{P_{\mathcal{T}}}\left[\mathbf{1}_{A_\gamma}\ell_{h,f_{\mathcal{T}}}\right] + \mathbb{E}_{P_{\mathcal{T}}}\left[\mathbf{1}_{A_\gamma^c}\ell_{h,f_{\mathcal{T}}}\right] \\ &\le \gamma\mathcal{L}_{P_{\mathcal{T}}^\gamma}(h, f_{\mathcal{T}}) + (1-\gamma)M. \end{aligned} \tag{27}$$

This also applies when $\gamma = 1$. By Equations (18) and (26),

$$\begin{aligned} \mathcal{L}_{P_{\mathcal{T}}^\gamma}(h, f_{\mathcal{T}}) &= \mathcal{L}_{P_G^h}(h, f_{\mathcal{T}}) + \left[\mathcal{L}_{P_{\mathcal{T}}^\gamma}(h, f_{\mathcal{T}}) - \mathcal{L}_{P_G^h}(h, f_{\mathcal{T}})\right] \\ &\le \mathcal{L}_{P_G^h}(h, f_{\mathcal{T}}) + \left|\mathcal{L}_{P_{\mathcal{T}}^\gamma}(h, f_{\mathcal{T}}) - \mathcal{L}_{P_G^h}(h, f_{\mathcal{T}})\right| \\ &\le \mathcal{L}_{P_G^h}(h, f_{\mathcal{T}}) + \mathrm{disc}_{\mathcal{L}}(P_G^h, P_{\mathcal{T}}^\gamma) \\ &\le 2\mathbb{E}_{\boldsymbol{x},\boldsymbol{z}}d_y\big(h(\boldsymbol{x}_G^h), \bar{\boldsymbol{y}}(\boldsymbol{x})\big) \\ &\quad + 2\mathbb{E}_{\boldsymbol{x},\boldsymbol{z}}d_y\big(\bar{\boldsymbol{y}}(\boldsymbol{x}), f_{\mathcal{T}}(\boldsymbol{x}_G^h)\big) + \mathrm{disc}_{\mathcal{L}}(P_G^h, P_{\mathcal{T}}^\gamma) \\ &= 2\mathcal{R}_G(h) + 2\lambda_h + \mathrm{disc}_{\mathcal{L}}(P_G^h, P_{\mathcal{T}}^\gamma). \end{aligned} \tag{28}$$

Together with Equation (27), this proves Equation (16). We next expand its terms.

(1) Representativeness and adaptation stability. The generated-pair fitting term is exactly

$$\mathcal{R}_G(h) = \mathcal{L}_{GC}. \tag{29}$$

For the parameter and feature-scale controls, the radial–directional decomposition gives

$$\begin{aligned} \|\boldsymbol{b} - \boldsymbol{b}^*\|_2^2 &= r^2 + (r^*)^2 - 2rr^*\langle \boldsymbol{v}, \boldsymbol{v}^*\rangle \\ &= (r - r^*)^2 + rr^*\|\boldsymbol{v} - \boldsymbol{v}^*\|_2^2. \end{aligned} \tag{30}$$

Thus Equation (24) yields

$$\begin{aligned} d_y(h(\boldsymbol{x}), N_0(\boldsymbol{x})) &= \frac{1}{L_{out}} \|H_{\boldsymbol{\eta}}(\boldsymbol{b}) - H_{\boldsymbol{\eta}^*}(\boldsymbol{b}^*)\|_2^2 \\ &\le \frac{2}{L_{out}} \Big[\|H_{\boldsymbol{\eta}}(\boldsymbol{b}) - H_{\boldsymbol{\eta}}(\boldsymbol{b}^*)\|_2^2 + \|H_{\boldsymbol{\eta}}(\boldsymbol{b}^*) - H_{\boldsymbol{\eta}^*}(\boldsymbol{b}^*)\|_2^2 \Big] \\ &\le \frac{2}{L_{out}} \Big[L_H^2 (r - r^*)^2 + L_H^2 r r^* \|\boldsymbol{v} - \boldsymbol{v}^*\|_2^2 + L_\eta^2 \|\boldsymbol{\eta} - \boldsymbol{\eta}^*\|_2^2 \Big] \\ &\le \frac{2}{L_{out}} \Big[L_H^2 (r - r^*)^2 + L_H^2 L_{dir}^2 \|\boldsymbol{\beta} - \boldsymbol{\beta}^*\|_2^2 + L_\eta^2 \|\boldsymbol{\eta} - \boldsymbol{\eta}^*\|_2^2 \Big]. \end{aligned} \tag{31}$$

With

$$C_{REG} = \frac{2n_\theta}{L_{out}} \max\{L_\eta^2, L_H^2 L_{dir}^2\}, \qquad C_{SCC} = \frac{2L_H^2}{L_{out}}, \tag{32}$$

we obtain

$$\mathbb{E}_{P_\mathcal{T}^\gamma} d_y(h, N_0) \le C_{REG} \mathcal{L}_{REG} + C_{SCC} \mathcal{L}_{SCC} =: S_\theta. \tag{33}$$

The triangle inequality in the space of square-integrable vector-valued functions gives

$$\begin{aligned} \sqrt{\Delta_h} &\le \sqrt{\mathbb{E}_{P_\mathcal{T}^\gamma} d_y(h, N_0)} + \sqrt{\mathbb{E}_{P_\mathcal{T}^\gamma} d_y(N_0, N^*)} \\ &\le \sqrt{S_\theta} + \sqrt{\varepsilon_{inf}}, \\ \Delta_h &\le \Big(\sqrt{S_\theta} + \sqrt{\varepsilon_{inf}} \Big)^2 =: R_\theta. \end{aligned} \tag{34}$$

When $N_0 = N^*$, $R_\theta = S_\theta$. SCC controls the radial term in Equation (30); directional and head changes still require the other stability conditions.

(2) Generalization. For the kernel mean embedding (Gretton et al., 2012),

$$\begin{aligned} \mu_P = \mathbb{E}_{\boldsymbol{x} \sim P} \phi_k(\boldsymbol{x}), \qquad \mathrm{MMD}_k(P, Q) = \|\mu_P - \mu_Q\|_{\mathcal{H}_k}, \\ |\mathbb{E}_P v - \mathbb{E}_Q v| = |\langle v, \mu_P - \mu_Q \rangle_{\mathcal{H}_k}| \\ \le \|v\|_{\mathcal{H}_k} \mathrm{MMD}_k(P, Q), \qquad v \in \mathcal{H}_k. \end{aligned} \tag{35}$$

Consequently,

$$\begin{aligned} \mathrm{disc}_\mathcal{L}(P_G^h, P_\mathcal{T}^\gamma) &= \sup_{g, g' \in \mathcal{H}} \Big| \mathbb{E}_{P_G^h} \ell_{g,g'} - \mathbb{E}_{P_\mathcal{T}^\gamma} \ell_{g,g'} \Big| \\ &\le \sup_{g, g' \in \mathcal{H}} \|\ell_{g,g'}\|_{\mathcal{H}_k} D_h \le B_\mathcal{H} D_h. \end{aligned} \tag{36}$$

Coupling $\boldsymbol{x}_G^h$ and $\boldsymbol{x}_G^*$ through the same $(\boldsymbol{x}, \boldsymbol{z})$ and using Equation (23),

$$\begin{aligned} \mathrm{MMD}_k(P_G^h, P_G^*) &= \big\| \mathbb{E}_{\boldsymbol{x}, \boldsymbol{z}} [\phi_k(\boldsymbol{x}_G^h) - \phi_k(\boldsymbol{x}_G^*)] \big\|_{\mathcal{H}_k} \\ &\le \mathbb{E}_{\boldsymbol{x}, \boldsymbol{z}} \|\phi_k(\boldsymbol{x}_G^h) - \phi_k(\boldsymbol{x}_G^*)\|_{\mathcal{H}_k} \\ &\le L_{kG} \mathbb{E}_{P_\mathcal{T}^\gamma} \|h(\boldsymbol{x}) - \bar{\boldsymbol{y}}(\boldsymbol{x})\|_y \le L_{kG} \sqrt{\Delta_h}, \\ \mathbb{E}_{\boldsymbol{x}, \boldsymbol{z}} d_y(N^*(\boldsymbol{x}_G^h), N^*(\boldsymbol{x}_G^*)) &\le L_{NG}^2 \mathbb{E}_{P_\mathcal{T}^\gamma} \|h(\boldsymbol{x}) - \bar{\boldsymbol{y}}(\boldsymbol{x})\|_y^2 = L_{NG}^2 \Delta_h. \end{aligned} \tag{37}$$

Hence

$$\begin{aligned} D_h &\le \mathrm{MMD}_k(P_G^h, P_G^*) + \mathrm{MMD}_k(P_G^*, P_\mathcal{T}^\gamma) \\ &\le D_G + L_{kG} \sqrt{\Delta_h} \le D_G + L_{kG} \sqrt{R_\theta}, \end{aligned} \tag{38}$$

$$\mathrm{disc}_\mathcal{L}(P_G^h, P_\mathcal{T}^\gamma) \le B_\mathcal{H} \Big(D_G + L_{kG} \sqrt{R_\theta} \Big). \tag{39}$$

(3) Intrinsic Error. By Equation (26),

$$\begin{aligned} \lambda_h &= \mathbb{E}_{\boldsymbol{x}, \boldsymbol{z}} d_y\big(\bar{\boldsymbol{y}}(\boldsymbol{x}), f_\mathcal{T}(\boldsymbol{x}_G^h)\big) \\ &\le 2 \mathbb{E}_{\boldsymbol{x}, \boldsymbol{z}} d_y\big(\bar{\boldsymbol{y}}(\boldsymbol{x}), N^*(\boldsymbol{x}_G^h)\big) + 2 \mathbb{E}_{P_G^h} e_*(\boldsymbol{x}), \end{aligned} \tag{40}$$

$$\begin{aligned}\mathbb{E}_{\boldsymbol{x},\boldsymbol{z}} d_y\big(\bar{\boldsymbol{y}}(\boldsymbol{x}), N^*(\boldsymbol{x}_G^h)\big) &\le 2\mathbb{E}_{\boldsymbol{x},\boldsymbol{z}} d_y\big(N^*(\boldsymbol{x}_G^h), N^*(\boldsymbol{x}_G^*)\big) \\ &\quad + 2\mathbb{E}_{\boldsymbol{x},\boldsymbol{z}} d_y(N^*(\boldsymbol{x}_G^*), \bar{\boldsymbol{y}}(\boldsymbol{x})) \\ &\le 2L_{NG}^2 \Delta_h + 2\mathcal{L}_{SEM}.\end{aligned} \tag{41}$$

Since $A_\gamma = \{u \le q_\gamma\}$ is determined by u, Equations (25) and (35) imply

$$\begin{aligned}\mathbb{E}_{P_{\mathcal{T}}^\gamma} e_*(\boldsymbol{x}) &= \mathbb{E}[\mathbb{E}[e_*(\boldsymbol{x}) \mid u(\boldsymbol{x})] \mid A_\gamma] \\ &\le a\,\mathbb{E}[u(\boldsymbol{x}) \mid A_\gamma] + b \le a q_\gamma + b, \\ \mathbb{E}_{P_G^h} e_*(\boldsymbol{x}) &\le \mathbb{E}_{P_{\mathcal{T}}^\gamma} e_*(\boldsymbol{x}) + \left|\mathbb{E}_{P_G^h} e_*(\boldsymbol{x}) - \mathbb{E}_{P_{\mathcal{T}}^\gamma} e_*(\boldsymbol{x})\right| \\ &\le a q_\gamma + b + B_* D_h.\end{aligned} \tag{42}$$

Combining these inequalities gives

$$\begin{aligned}\lambda_h &\le 4L_{NG}^2 \Delta_h + 4\mathcal{L}_{SEM} + 2(a q_\gamma + b) + 2B_* D_h \\ &\le 4\mathcal{L}_{SEM} + 4L_{NG}^2 R_\theta + 2(a q_\gamma + b) + 2B_*\Big(D_G + L_{kG}\sqrt{R_\theta}\Big).\end{aligned} \tag{43}$$

(4) Expanded GIRA Bound. Let $C_D = B_{\mathcal{H}} + 4B_*$. Substitution into Equation (16) yields

$$\begin{aligned}\mathcal{L}_{\mathcal{T}}(h, f_{\mathcal{T}}) &\le \gamma\Big[2\mathcal{L}_{GC} + B_{\mathcal{H}}(D_G + L_{kG}\sqrt{R_\theta}) \\ &\qquad + 8\mathcal{L}_{SEM} + 8L_{NG}^2 R_\theta + 4(a q_\gamma + b) + 4B_*(D_G + L_{kG}\sqrt{R_\theta})\Big] + (1-\gamma)M \\ &= \gamma\Big[2\mathcal{L}_{GC} + 8\mathcal{L}_{SEM} + 4a q_\gamma + 4b + C_D D_G \\ &\qquad + 8L_{NG}^2 R_\theta + C_D L_{kG}\sqrt{R_\theta}\Big] + (1-\gamma)M.\end{aligned} \tag{44}$$

(5) Connection to the optimized objectives. For positive adaptation weights in Equation (13), put

$$\begin{aligned}\nu_A = \max\left\{\frac{C_{REG}}{\alpha_{REG}}, \frac{C_{SCC}}{\alpha_{SCC}}\right\}, \qquad R_A(t) = \left(\sqrt{\nu_A t} + \sqrt{\varepsilon_{inf}}\right)^2, \quad t \ge 0, \\ \mathcal{L}_{GC} \le \frac{\mathcal{L}_A}{\alpha_{GC}}, \qquad S_\theta \le \nu_A \mathcal{L}_A, \qquad R_\theta \le R_A(\mathcal{L}_A).\end{aligned} \tag{45}$$

For a generator-family-independent connection, use the population counterpart of the negative checkpoint Score in Equation (75). Let T_y be the fixed output transformation to the scoring units and define

$$\begin{aligned}\boldsymbol{e}_G &= T_y(N^*(\boldsymbol{x}_G^*)) - T_y(\bar{\boldsymbol{y}}(\boldsymbol{x})), \\ m_G &= \mathbb{E}_{\boldsymbol{x},\boldsymbol{z}} \frac{\|\boldsymbol{e}_G\|_1}{L_{out}}, \qquad r_G = \left(\mathbb{E}_{\boldsymbol{x},\boldsymbol{z}} \frac{\|\boldsymbol{e}_G\|_2^2}{L_{out}}\right)^{1/2}, \\ Q_G &= w_M m_G + w_R r_G + w_D D_G^2, \qquad (w_M, w_R, w_D) = (0.05, 0.05, 10).\end{aligned} \tag{46}$$

The proof applies to any positive w_M, w_R, w_D. Assume $\|\boldsymbol{p} - \boldsymbol{q}\|_2 \le s_y \|T_y(\boldsymbol{p}) - T_y(\boldsymbol{q})\|_2$ on the relevant output range. Here $s_y = 1$ for identical units; for an invertible affine transformation $T_y(\boldsymbol{p}) = A_y \boldsymbol{p} + \boldsymbol{c}_y$, one may take $s_y = \|A_y^{-1}\|_{op}$, where $\|\cdot\|_{op}$ is the Euclidean operator norm. Nonnegativity gives

$$\begin{aligned}\mathcal{L}_{SEM} \le s_y^2 r_G^2 \le \frac{s_y^2}{w_R^2} Q_G^2, \qquad D_G \le \sqrt{\frac{Q_G}{w_D}}, \\ 8\mathcal{L}_{SEM} + C_D D_G \le \frac{8 s_y^2}{w_R^2} Q_G^2 + \frac{C_D}{\sqrt{w_D}}\sqrt{Q_G}.\end{aligned} \tag{47}$$

Thus

$$\begin{aligned}\mathcal{L}_{\mathcal{T}}(h, f_{\mathcal{T}}) \le \gamma\Big[\frac{2}{\alpha_{GC}}\mathcal{L}_A + 8L_{NG}^2 R_A(\mathcal{L}_A) + C_D L_{kG}\sqrt{R_A(\mathcal{L}_A)} \\ + \frac{8 s_y^2}{w_R^2} Q_G^2 + \frac{C_D}{\sqrt{w_D}}\sqrt{Q_G} + 4a q_\gamma + 4b\Big] + (1-\gamma)M.\end{aligned} \tag{48}$$

For a fixed reference model, filtering rule, and common constants, this upper envelope is nondecreasing in the population quantities $\mathcal{L}_A$ and Q_G. Consequently, decreasing either quantity while holding the other fixed decreases, or leaves unchanged, this envelope. The generator connection is through semantic quality and the checkpoint-selection criterion, not an asserted equivalence between a generator-specific training loss and MMD. □

Corollary A.4 (UTF and the generalized-class radius)**.** *For $j \in \{*, h\}$, let*

$$\eta_G^j = \Pr[\boldsymbol{x}_G^j \notin \mathcal{X}_\mathcal{T}], \qquad V_G^* = \mathcal{L}_{SEM}, \qquad V_G^h = 2\mathcal{L}_{SEM} + 2L_{NG}^2 R_\theta. \tag{49}$$

For $\tau > 0$, with probability taken over $\boldsymbol{x} \sim P_\mathcal{T}^\gamma$, the stochastic prediction realization ω, and $\boldsymbol{z} \sim P_\mathcal{Z}$,

$$\begin{aligned} &\Pr\left[d_y(N_\omega^*(\boldsymbol{x}), \bar{\boldsymbol{y}}(\boldsymbol{x})) > \tau \ \vee \ \boldsymbol{x}_G^j \notin \mathcal{C}_\tau(\bar{\boldsymbol{y}}(\boldsymbol{x}))\right] \\ &\qquad \le \eta_G^j + \frac{q_\gamma + V_G^j}{\tau}. \end{aligned} \tag{50}$$

For a desired failure probability $\delta \in (\eta_G^j, 1)$, it suffices to choose

$$\tau > 0, \qquad \tau \ge \frac{q_\gamma + V_G^j}{\delta - \eta_G^j}. \tag{51}$$

If the numerator vanishes, any positive τ suffices.

Proof. For $\boldsymbol{x} \in A_\gamma$, Markov's inequality gives

$$\Pr_\omega[d_y(N_\omega^*(\boldsymbol{x}), \bar{\boldsymbol{y}}(\boldsymbol{x})) > \tau] \le \frac{u(\boldsymbol{x})}{\tau} \le \frac{q_\gamma}{\tau}. \tag{52}$$

Define $\ell_{SEM}^j(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{z}} d_y(N^*(\boldsymbol{x}_G^j), \bar{\boldsymbol{y}}(\boldsymbol{x}))$. By the definition of $\mathcal{C}_\tau$,

$$\begin{aligned} \Pr_{\boldsymbol{z}}[\boldsymbol{x}_G^j \notin \mathcal{C}_\tau(\bar{\boldsymbol{y}}(\boldsymbol{x}))] &\le \Pr_{\boldsymbol{z}}[\boldsymbol{x}_G^j \notin \mathcal{X}_\mathcal{T}] + \Pr_{\boldsymbol{z}}[d_y(N^*(\boldsymbol{x}_G^j), \bar{\boldsymbol{y}}(\boldsymbol{x})) > \tau] \\ &\le \Pr_{\boldsymbol{z}}[\boldsymbol{x}_G^j \notin \mathcal{X}_\mathcal{T}] + \frac{\ell_{SEM}^j(\boldsymbol{x})}{\tau}. \end{aligned} \tag{53}$$

Moreover, Equations (34) and (41) imply

$$\begin{aligned} &\mathbb{E}_{P_\mathcal{T}^\gamma} \ell_{SEM}^*(\boldsymbol{x}) = \mathcal{L}_{SEM} = V_G^*, \\ &\mathbb{E}_{P_\mathcal{T}^\gamma} \ell_{SEM}^h(\boldsymbol{x}) \le 2\mathcal{L}_{SEM} + 2L_{NG}^2 \Delta_h \le V_G^h. \end{aligned} \tag{54}$$

For the events $E_u = \{d_y(N_\omega^*(\boldsymbol{x}), \bar{\boldsymbol{y}}(\boldsymbol{x})) > \tau\}$ and $E_G^j = \{\boldsymbol{x}_G^j \notin \mathcal{C}_\tau(\bar{\boldsymbol{y}}(\boldsymbol{x}))\}$,

$$\begin{aligned} \Pr(E_u \cup E_G^j) &\le \Pr(E_u) + \Pr(E_G^j) \\ &\le \frac{q_\gamma}{\tau} + \eta_G^j + \frac{\mathbb{E}_{P_\mathcal{T}^\gamma} \ell_{SEM}^j(\boldsymbol{x})}{\tau} \\ &\le \eta_G^j + \frac{q_\gamma + V_G^j}{\tau}. \end{aligned} \tag{55}$$

This proves Equations (50) and (51). For the actual adaptation samples ($j = h$), set

$$\delta_u = \frac{q_\gamma}{\tau}, \qquad \delta_G = \frac{2\mathcal{L}_{SEM} + 2L_{NG}^2 R_\theta}{\tau}. \tag{56}$$

Then Equation (51) gives $\eta_G^h + \delta_u + \delta_G \le \delta$, and

$$8\mathcal{L}_{SEM} + 8L_{NG}^2 R_\theta + 4aq_\gamma = 4\tau(\delta_G + a\delta_u). \tag{57}$$

□

The simpler bound $(q_\gamma + \mathcal{L}_{SEM})/\tau$ is recovered for $j = *$ only when $\eta_G^* = 0$. The corollary is an average guarantee over retained inputs, not a uniform per-input guarantee. Here τ is a normalized squared output tolerance, corresponding to Euclidean output radius $\sqrt{L_{out}\tau}$; it does not bound an input-space radius.

Scope and implemented loss conventions. The RKHS, conditional-sensitivity, feature-stability, and error–uncertainty conditions are explicit sufficient conditions, not consequences of Assumptions A.1 to A.3 alone. In particular, b may retain systematic reference bias even at zero uncertainty, and ε_{inf} must be retained unless the inference conventions agree. The semantic evaluator in the losses and Score must use the same N^* as the generalized-class definition; using a different evaluator requires a corresponding mismatch term as well. The argument is at the population level: empirical losses, finite-K uncertainty estimates, empirical filtering proportions, and finite-sample checkpoint scores require additional estimation bounds before they can replace the displayed population quantities. Neither a nonvacuous numerical certificate nor strict improvement over the source model follows without further conditions.

For the Huber semantic loss used in Section C.2, the squared-loss quantities above are not identified with the implemented Huber loss. For unit-threshold Huber loss

$$\rho_1(t) = \begin{cases} t^2/2, & |t| \le 1, \\ |t| - 1/2, & |t| > 1, \end{cases} \qquad |t| \le R_{max} \implies t^2 \le C_\rho \rho_1(t), \quad C_\rho = 2\max\{1, R_{max}\}. \tag{58}$$

Indeed, the small-residual case is an equality with factor 2; otherwise $\rho_1(t) \ge |t|/2$ and $t^2 \le R_{max}|t|$. Averaging over output coordinates and samples gives

$$\mathcal{L}_{SEM} \le C_\rho \mathcal{L}_{SEM}^{Huber}, \qquad \mathcal{L}_{GC} \le C_\rho \mathcal{L}_{GC}^{Huber}, \tag{59}$$

where the second relation is needed only if the adaptation fitting loss also uses Huber, and the residual bound must hold for each converted loss in the same units. In that case, Equation (48) holds with the Huber-based adaptation objective and $2/\alpha_{GC}$ replaced by $2C_\rho/\alpha_{GC}$; the REG/SCC controls are unchanged. No control of MMD by the temporal first-difference loss $\mathcal{L}_{TV}$ alone is asserted.

A.3 Visualization and Intuitive Explanation of GIRA

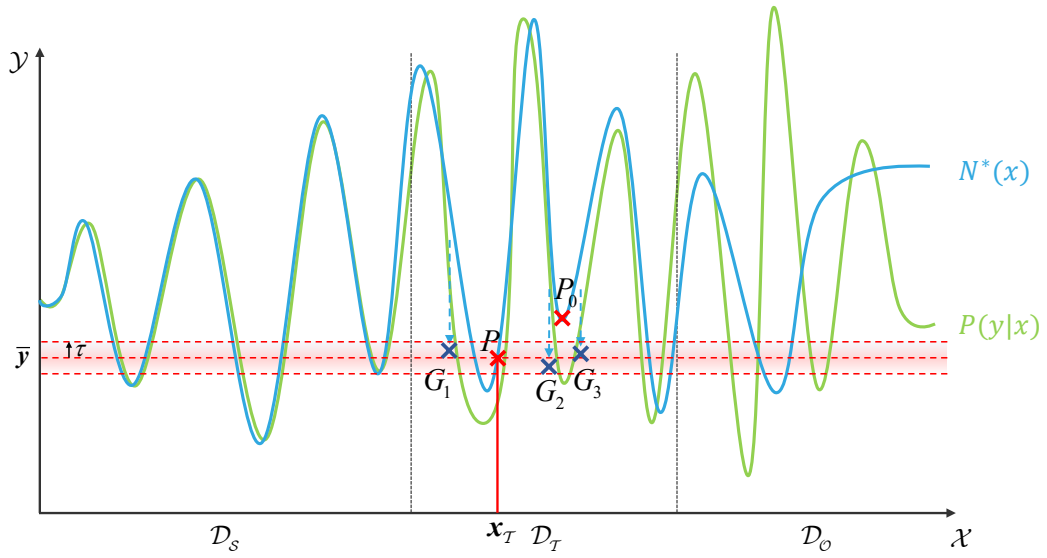


Figure 7: Visual explanation of the GIRA working mechanism. The horizontal and vertical axes denote one-dimensional projections of the input and output spaces, respectively. The green and blue curves represent $P(y \mid x)$ and $N^*(x)$, respectively.

This section provides a visual explanation to facilitate an intuitive understanding of the GIRA framework and addresses potential inquiries. It should be noted that this discussion aims to enhance conceptual comprehension and does not constitute a rigorous mathematical derivation.

The essence of a regression task lies in learning an input-to-output mapping function $N^*(x)$ to approximate the underlying conditional relationship $P(y \mid x)$. As illustrated in Figure 7, the

horizontal and vertical axes represent the one-dimensional projections of the input and output spaces, respectively. On the horizontal axis, a greater distance from the origin signifies a larger domain discrepancy relative to the source domain. Consequently, the schematic input space is divided into the source domain $\mathcal{D}_{\mathcal{S}}$, the target domain $\mathcal{D}_{\mathcal{T}}$, and other domains $\mathcal{D}_{\mathcal{O}}$.

We denote $P(y \mid x)$ and $N^*(x)$ using green and blue curves, respectively. It is worth noting that while depicting $N(x)$ as a continuous curve is accurate due to the nature of deep learning models, $P(y|x)$ is generally not a continuous function. Nevertheless, we adopt this representation for visual clarity. It can be observed that $N^*(x)$ is not an injective function, consistent with Assumption A.1. Furthermore, the same conditional probability distribution governs both the source and target domains, consistent with Assumption A.2. $N^*(x)$ demonstrates a strong fit on $\mathcal{D}_{\mathcal{S}}$. On $\mathcal{D}_{\mathcal{T}}$, the fit degrades because of the domain shift, while the model retains basic inference capability as stated in Assumption A.3. On $\mathcal{D}_{\mathcal{O}}$, the discrepancy is too large for the schematic model to provide reliable predictions. The displayed widths of these regions are illustrative and do not represent their sample counts or geometric measures.

We use this figure to demonstrate the forward propagation process in Step 2. After sampling $\boldsymbol{x}_{\mathcal{T}}$ from $\mathcal{D}_{\mathcal{T}}$, a vertical line from $\boldsymbol{x}_{\mathcal{T}}$ intersects $N^*(x)$ at point P. The ordinate of P corresponds to the generalized-class identifier $\overline{\boldsymbol{y}}$. Subsequently, $\overline{\boldsymbol{y}}$ is fed into G. The generator approximates the conditional inverse mapping of N^* and incorporates a random latent variable as input. Through random sampling of $\boldsymbol{z}$, G produces candidates such as G_1, G_2, and G_3. The red band schematically represents the output tolerance associated with $\mathcal{C}_\tau(\overline{\boldsymbol{y}})$. The y-coordinates of the candidate points need not exactly be the same as $\bar{\boldsymbol{y}}$, but their generated inputs are encouraged to lie in this inverse-mapping neighborhood under N^*. Although these points may not perfectly align with $P(y \mid x)$ because of generalized-class error λ_h and imperfect generator training, they provide adaptation signals at unlabeled positions. Finally, $\mathcal{L}_{GC}$ reshapes the curve N using the generated input–GC pairs, making it adhere more closely to the morphology of $P(y|x)$.

Discussion on Key Inquiries:

Q1: Why do the samples generated by G remain in the target-domain region?
The generator-specific objective $\mathcal{L}_{GEN}$ encourages the generated-input distribution to approach the target-input distribution. Therefore, candidate points tend to fall within the target domain $\mathcal{D}_{\mathcal{T}}$ rather than the source domain $\mathcal{D}_{\mathcal{S}}$ or other domains $\mathcal{D}_{\mathcal{O}}$.

Q2: Why are the potential solutions searched by G likely to be located near $P(y \mid x)$?
This schematic is a dimensionality-reduced representation. In reality, the high-dimensional space implies that $\overline{\boldsymbol{y}}$ likely has infinite intersections with $P(y|x)$. Thus, finding some potential solutions via G is feasible. Even if the generated points do not perfectly align with the true $P(y \mid x)$, they still provide directional references for updating N.

Q3: If N yields poor predictions at certain points (e.g., P_0), will these points cause N to deviate further from $P(y \mid x)$?
Unreliable generalized-class assignments can potentially lead to negative transfer. GIRA mitigates this risk by using *Uncertainty Threshold Filtering* to exclude high-risk samples and Equation (14) to prevent excessive parameter drift.

B Experimental Details

B.1 Hyperparameter Settings

The hyperparameter configurations for the three steps (pre-training, conditional-generator training, and adaptation) are summarized in Table 6. GIRA uses the Adam optimizer in its generator-training and adaptation stages for all three tasks. The sensitivity of the four loss coefficients is evaluated in Section C.5.

B.2 Task Settings and Preprocessing

Table 7 summarizes the specific configurations for the three tasks, including dataset details, preprocessing pipelines, and evaluation metrics.

Table 6: Hyperparameter settings for the pre-training, conditional-generator training, and adaptation stages.

Step / Parameter	Value	Description
Step 0: Pre-training		
Learning Rate	1×10^{-4} (A), 1×10^{-3} (B), 3×10^{-4} (C)	Using StepLR
Batch Size	256	-
Epochs	50	-
Early Stop Patience	20	Stop if no improvement
Val Ratio	0.1	Validation set split
Weight Decay	1×10^{-2}	L2 Regularization
Dropout Rate	0.1	-
Step 1: Training CMF		
K	10	Sampling times for UTF
γ	0.8	Keep ratio for UTF
Generator Updates	10000	Total training iterations
Generator LR	9.8567×10^{-5}	-
Noise Channels	1	One Gaussian noise value per input time step
α_{SEM}	3.3×10^{-5}	Semantic loss weight
MeanFlow Hidden Channels	128	Width of the velocity network
MeanFlow Layers	4	Number of hidden Conv1d stages
Sampling Steps	4	Number of generation steps
Semantic Warm-up	2000	Updates before ramping α_{SEM}
Step 2: Adaptation		
Adaptation Epochs	40	-
Adaptation LR	5.6×10^{-4}	-
α_{GC}	1.4×10^{-4}	Generalized-class adaptation loss weight
α_{REG}	0.83	Regularization weight
α_{SCC}	4.1×10^{-4}	Scale constraint weight

Table 7: Detailed settings and preprocessing pipelines for the three tasks.

Item	Task A: NILM	Task B: Load Forecasting	Task C: Speech Enhancement
Datasets	UK-DALE, REFIT, Dataport	NYISO, GEFCom, Panama	VCTK, AISHELL-3, JSUT
Source Size	100000	100000	50000
Target Size	5000	5000	5000
Input Length	640 time steps	160 time steps	4096 time steps
Output Length	640 time steps	24 time steps (Future)	4096 time steps
Sampling Rate	Resampled to 6s	Resampled to 60min	16kHz
Preprocessing	1. Fill missing (ffill, limit 180). 2. Window stride: **U**-16, **I**-64, **D**-32. 3. Norm: Mean-Std(Only input).	1. Linear interpolation for missing. 2. Window stride: **N**-16, **G**-8, **P**-4. 3. Norm: Mean-Std(Only input).	1. Window stride: 4096. 2. Mix with Noise (**O/M**) with random SNR (10 ± 2 dB). 3. Norm: RMS normalization.
Metrics	MAE, SAE, F1-Score	MAE, RMSE	MAE, SegSNR

Target-data protocol. The source pre-training sequences are split into training and validation portions using a 90%/10% ratio, and a separate source-side test stream is used for pre-training evaluation. For each transfer scenario, up to 5,000 windows from the target-domain test stream form the unlabeled target pool used for SFDA adaptation. The same target windows are evaluated offline with their labels, so the default main experiments do not use a disjoint labeled target test split. The unlabeled target pool is observed during adaptation, while target labels are reserved for offline evaluation and are not used for optimization, UTF filtering, checkpoint selection, or hyperparameter tuning. The source checkpoint is selected using source-validation performance before adaptation. In the default Conditional MeanFlow runs, generator training and Step 2 adaptation use the UTF-retained target inputs, while the generator Score is computed on a sampled target batch without a separate held-out scoring split. Only the conditional-generator compatibility experiment reserves 500 aligned input–condition pairs from the UTF-retained pool for shared checkpoint scoring. Those 500 pairs are excluded from generator updates, and the remaining retained pairs are used for generator training.

Evaluation parameters. For Task A, the fridge on/off states are obtained from predicted and reference power using a 50 W threshold, a minimum on-duration of 10 samples, and a minimum

off-duration of 2 samples before binary F1 is computed. For Dropout-based source models, GIRA estimates uncertainty from ten stochastic forward passes with dropout enabled and averages the predictive variance over output steps. The generalized-class identifier is the mean of those predictions. The UTF retained proportion γ is empirical and is not an overall probability derived from τ. For Task C, the deterministic model is evaluated on ten Gaussian-perturbed views with noise standard deviation 0.01. The prediction mean and per-sample uncertainty are computed across these views in evaluation mode, with the variance averaged over output coordinates. The Gaussian-view variance used by the regression baselines is a separate two-view consistency term defined in Section B.4.

Reporting convention. Per-transfer adapted-method results in Tables 1–3 are reported as means $\pm$ standard deviations over seeds 2024, 2025, and 2026 whenever the corresponding standard deviations are available. Zero-Shot values and task-average columns remain point estimates because the accompanying standard-deviation files do not contain those entries.

B.3 Model Architectures

The architectures of the pre-trained models and the conditional generator used by GIRA are described in Table 8.

Table 8: Architectures of the pre-trained models and GIRA components.

Model Component	Architecture Details	Parameters
Pre-trained Model (Task A/B) (Zhang et al., 2023)	**ResNet-seq2seq** Encoder: 3 × (MaxPool1d + 2 ResBlocks) Bottleneck: 3 CBRBlocks with Dropout Decoder: 4 × (ConvTranspose1d + ResBlock) with skip connections Head: Linear → ReLU → Linear → ReLU	Backbone Hidden: 32 Backbone Kernel size: 8 Head Hidden: 128
Pre-trained Model (Task C) (Defossez et al., 2020)	**Demucs Denoiser** Encoder: 5 layers, Stride 4, GLU activation Bottleneck: Bi-directional LSTM Decoder: 5 layers with skip connections	Initial Conv Channels: 48 Growth Factor: 2 Resample: 4x
Conditional Generator (G)	**CMF** (Geng et al., 2026) Input: $\boldsymbol{z}_t$, $\widetilde{\boldsymbol{y}}$, $t\mathbf{1}$, and $(t-r)\mathbf{1}$ Velocity Network: Conv1d + GroupNorm + SiLU Output: 1×1 Conv1d	Hidden Channels: 128 Hidden Conv1d Stages: 4 Kernel Size: 5 Sampling Steps: 4

Conditional MeanFlow implementation. For a generalized-class condition $\bar{\boldsymbol{y}} \in \mathbb{R}^{L_{\text{out}}}$, its length-aligned representation is

$$\widetilde{\boldsymbol{y}} = \begin{cases} W_y \bar{\boldsymbol{y}} + \boldsymbol{b}_y, & L_{\text{out}} \neq L_{\text{in}}, \\ \bar{\boldsymbol{y}}, & L_{\text{out}} = L_{\text{in}}. \end{cases} \tag{60}$$

The velocity network input is

$$\text{stack}[\boldsymbol{z}_t, \widetilde{\boldsymbol{y}}, t\mathbf{1}, (t-r)\mathbf{1}] \in \mathbb{R}^{4 \times L_{\text{in}}}. \tag{61}$$

The network contains a $\text{Conv1d}(4, 128, 5)$ layer followed by SiLU, three $\text{Conv1d}(128, 128, 5)$ stages followed by eight-group GroupNorm and SiLU, and a final $\text{Conv1d}(128, 1, 1)$ layer. Thus, the default configuration has four hidden Conv1d stages.

To sample the MeanFlow interval, we draw

$$s_1, s_2 \sim \mathcal{N}(-0.4, 1), \qquad \widetilde{t} = \sigma(s_1), \qquad \widetilde{r} = \sigma(s_2), \tag{62}$$

and set

$$t = \max(\widetilde{t}, \widetilde{r}), \qquad r = \min(\widetilde{t}, \widetilde{r}). \tag{63}$$

For the first 75% of each minibatch, we set $r = t$ as in the implemented data-proportion schedule. Given $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I)$ with the same shape as $\boldsymbol{x}_{\mathcal{T}}$, the interpolation path is

$$\boldsymbol{z}_t = (1-t)\boldsymbol{x}_{\mathcal{T}} + t\boldsymbol{\epsilon}, \qquad \boldsymbol{v} = \boldsymbol{\epsilon} - \boldsymbol{x}_{\mathcal{T}}. \tag{64}$$

The Jacobian–vector product used by the implementation is

$$\dot{\boldsymbol{u}} = \frac{\partial u_\phi}{\partial \boldsymbol{z}_t}\boldsymbol{v} + \frac{\partial u_\phi}{\partial t}, \qquad \boldsymbol{u}_{\text{tgt}} = \text{stopgrad}[\boldsymbol{v} - \text{clip}(t-r, 0, 1)\dot{\boldsymbol{u}}]\,. \tag{65}$$

For a minibatch of size B, define

$$\ell_i = \left\| u_\phi(\boldsymbol{z}_t^i, r_i, t_i, \bar{\boldsymbol{y}}_i) - \boldsymbol{u}_{\text{tgt}}^i \right\|_2^2. \tag{66}$$

The implemented adaptive MeanFlow loss is

$$\mathcal{L}_{MF} = \frac{1}{B} \sum_{i=1}^{B} \frac{\ell_i}{\text{stopgrad}[(\ell_i + \epsilon_{MF})^p]}, \qquad p = 1, \quad \epsilon_{MF} = 0.01. \tag{67}$$

Generation starts from an external Gaussian sample $\boldsymbol{z}_0 \sim P_{\mathcal{Z}}$. When the latent dimension exceeds one, a learned linear layer projects each latent vector to one scalar at each sequence position. The default four-step reverse Euler solver uses

$$t_k = 1 - \frac{k}{K_s}, \qquad r_k = 1 - \frac{k+1}{K_s}, \qquad K_s = 4, \tag{68}$$

$$\boldsymbol{z}_{k+1} = \boldsymbol{z}_k - (t_k - r_k) u_\phi(\boldsymbol{z}_k, r_k, t_k, \bar{\boldsymbol{y}}), \qquad k = 0, \ldots, K_s - 1, \tag{69}$$

and returns $\widehat{\boldsymbol{x}} = G_\phi(\boldsymbol{z}_0 \mid \bar{\boldsymbol{y}}) = \boldsymbol{z}_{K_s}$.

Let the elementwise Smooth-L1 function be

$$\rho_{\text{SL1}}(a) = \begin{cases} \frac{1}{2}a^2, & |a| < 1, \\ |a| - \frac{1}{2}, & |a| \geq 1. \end{cases} \tag{70}$$

The endpoint loss is

$$\mathcal{L}_{END} = \frac{1}{B L_{\text{in}}} \sum_{i=1}^{B} \sum_{j=1}^{L_{\text{in}}} \rho_{\text{SL1}}(\widehat{x}_{i,j} - x_{i,j}). \tag{71}$$

For the local-variation constraint, define the mean absolute first differences

$$q_i^G = \frac{1}{L_{\text{in}} - 1} \sum_{j=1}^{L_{\text{in}}-1} |\widehat{x}_{i,j+1} - \widehat{x}_{i,j}|, \qquad q_i^T = \frac{1}{L_{\text{in}} - 1} \sum_{j=1}^{L_{\text{in}}-1} |x_{i,j+1} - x_{i,j}|. \tag{72}$$

Let (μ_G, σ_G) and (μ_T, σ_T) denote the minibatch means and population standard deviations of $\{q_i^G\}$ and $\{q_i^T\}$. The TV loss is

$$\mathcal{L}_{TV} = \rho_{\text{SL1}}(\mu_G - \mu_T) + \rho_{\text{SL1}}(\sigma_G - \sigma_T). \tag{73}$$

This term aligns local-variation statistics and is not assumed to control the complete generated–target distribution discrepancy by itself.

The population notation in Equation (5) uses the normalized squared output distance d_y. In implementation, its empirical counterpart uses the regression criterion of the corresponding task. Task A uses Huber loss with $\delta = 1$, Task B uses MSE, and Task C uses L1 loss plus 0.5 times a multi-resolution STFT loss. The STFT loss uses FFT sizes $\{512, 1024, 2048\}$, hop sizes $\{64, 128, 256\}$, and window lengths $\{512, 1024, 2048\}$. At generator update m, the semantic weight is

$$\alpha_{SEM}^{(m)} = \begin{cases} 0, & 1 \leq m \leq 2000, \\ \alpha_{SEM} \dfrac{m - 2000}{2000}, & 2000 < m \leq 4000, \\ \alpha_{SEM}, & m > 4000, \end{cases} \qquad \alpha_{SEM} = 3.3 \times 10^{-5}. \tag{74}$$

The remaining weights are $\alpha_{END} = 1$ and $\alpha_{TV} = 2$. The generator is trained for 10,000 updates with Adam and a learning rate of 9.8567×10^{-5}. The MMD term in the Score of Equation (75) is used for checkpoint selection and does not receive gradients from Equation (10).

B.4 Baseline Settings

We compared GIRA with several representative domain adaptation methods. Their main hyperparameters are summarized in Table 9.

Table 9: Adaptation settings for baseline methods. Methods use the same backbone, target data, batch size, and adaptation budget unless a method-specific setting is listed.

Method	LR (Adaptation)	Key Hyperparameters / Mechanism
DeepCORAL	1×10^{-4}	CORAL weight increases linearly over adaptation epochs
DANN	1×10^{-8}	GRL ramp $0 \to 0.1$ ($\alpha = 1.0$), domain classifier (hidden=1024), adversarial weight 1
MDD	1×10^{-8}	margin-disparity weight $\alpha_{mdd} = 1.0$
MMDA	1×10^{-4}	multi-statistics discrepancy weight $\alpha_{mmda} = 1.0$
CDAN	1×10^{-3}	randomized multilinear map (dim=1024), $\alpha_{cdan} = 7 \times 10^{-4}$
RSD	1×10^{-8}	$\alpha_{rsd} = 1 \times 10^{-3}$
SHOT	1×10^{-3}	frozen head, lowest-uncertainty 10%, teacher agreement
NRC	1×10^{-3}	reciprocal and expanded neighbors, $k = 5$
AaD	1×10^{-3}	attraction and repulsion, $k = 5$, repulsion weight 0.1
SSA	1×10^{-3}	source subspace alignment, rank 64
MAPU	1×10^{-3}	frozen source-trained feature imputer
SFHarmony	1×10^{-4}	harmony/discrepancy weight $w_{harmony} = 0.1$

Shared regression protocol. SHOT, NRC, AaD, SSA, and MAPU retain continuous predictions without introducing softmax categories. Their target adaptation uses SGD with momentum 0.9, Nesterov updates, and the learning rates in Table 9. Source-only artifacts are prepared before adaptation, and no source examples are accessed during target adaptation. Target labels are used for offline evaluation only. For SHOT, NRC, AaD, and MAPU, two input views are generated as $\boldsymbol{x}^{(v)} = \boldsymbol{x} + 0.01\boldsymbol{\epsilon}^{(v)}$ with independent standard Gaussian noise. Their variance term is $\mathcal{V} = \mathrm{mean}_t \, \mathrm{Var}_{v=1,2}[N(\boldsymbol{x}^{(v)})_t / s] / 0.01^2$, where s is the standard deviation of source-model predictions stored before adaptation. The variance term has weight 0.01, with dropout disabled and BatchNorm held in evaluation mode during its calculation.

SHOT. The prediction head is frozen, and a frozen copy of the source model supplies teacher predictions. The target bank stores features, predictions, and uncertainty from two Gaussian-perturbed teacher views. The threshold setting 0.9 retains the 10% of target samples with the lowest uncertainty, not samples whose predicted probability exceeds 0.9. The selected samples use Smooth-L1 agreement between student and teacher predictions normalized by s, together with the shared variance term.

NRC and AaD. Both methods use a target feature bank with $k = 5$ cosine-nearest neighbors, a normalized Smooth-L1 anchor to the frozen teacher, and the shared variance term. NRC weights reciprocal neighbors more strongly, downweights neighbors by their prediction disagreement, and adds expanded-neighbor agreement with coefficient 0.5. AaD applies attraction to neighboring prediction sequences after centering and normalization. Its repulsion penalizes non-neighbor similarity above 0.2 with coefficient 0.1. The feature and prediction banks are updated after each target optimization step.

MAPU. Before target adaptation, a feature imputer is trained on source inputs to reconstruct clean-input features from masked-input features. One of eight input segments is masked, and the imputer is subsequently frozen. Target adaptation minimizes the MSE between imputed masked-input features and clean-input features, together with the shared variance term. Only the regression model is updated using target inputs.

SSA. Source preparation stores feature statistics from at most 4,096 examples and forms a rank-64 PCA subspace. The alignment weights reflect the frozen source model's output sensitivity along each principal direction. Target adaptation minimizes a sensitivity-weighted symmetric KL divergence between diagonal Gaussian source and target projections. SSA does not use the Gaussian-view variance term.

SFHarmony. Source preparation fits the source feature statistics used by its GMM-based harmony objective. Target adaptation updates the regression model with the target feature distribution and uses SGD with the learning rate listed in Table 9.

B.5 Computational Environment

All experiments are conducted on a workstation with the following configuration:

- **Hardware**: 4 RTX4090 GPUs on Ubuntu 24.04 remote server.
- **Software Stack**: Python 3.7, PyTorch 1.13.1, Torchvision 0.14.1.
- **Key Dependencies**: NumPy 1.21.6, Pandas 1.3.5, Librosa 0.10.2 (for Task C).

C Supplementary Experiments

C.1 Performance Analysis on Different Pre-trained Architectures

Table 10: Architectures of the additional pre-trained frameworks.

Model	Architecture Details	Parameters
CNN (Liu et al., 2022)	Backbone: 5 Conv1d Layers (Replicate Padding+ReLU) Head: Linear → ReLU → Linear → ReLU	Filters: 32→32→64→64→128 Kernel Sizes: 9, 7, 5, 5, 5 Head Hidden: 1024
Bi-LSTM	Backbone: 2-layer Bi-LSTM Head: Linear → ReLU → Linear → ReLU	Bi-LSTM Hidden: 256 Bidirectional: True Head Hidden: 512
TCN (Lin et al., 2022)	Backbone: 5 TCN Residual Blocks (Dilated Conv + WeightNorm + ReLU) Head: Linear → ReLU → Linear → ReLU → Linear → ReLU	Filters: 128 Kernel Size: 3 Dilation: $d \in \{1, 2, 4, 8, 16\}$ Head Hidden: 1024, 128

To assess the robustness and universality of the proposed framework across different deep learning architectures, we extended our evaluation to include alternative sequence-to-sequence models. Specifically, we benchmarked GIRA against comparative methods using a pure CNN (Liu et al., 2022), a Bi-LSTM, and a TCN (Lin et al., 2022). Detailed architectural specifications and hyperparameter settings are provided in Table 10, while all other experimental configurations remain consistent with those in Table 6 and Table 7. We conducted experiments on tasks T#1 and T#4, with results summarized in Table 11.

GIRA improves all reported metrics over the corresponding pre-adaptation model for every architecture. It obtains the lowest MAE on both T#1 and T#4 across all four backbones. The relative strength of the source checkpoints varies by task: Bi-LSTM and CNN have lower zero-shot MAE than ResNet-seq2seq on T#1, but neither leads on T#4. On T#4, it also obtains the lowest RMSE with ResNet-seq2seq, CNN, and Bi-LSTM, while SSA performs better on TCN RMSE. These results show that the adaptation gains are not restricted to the default pre-trained architecture.

C.2 Compatibility with Conditional Generative Models

The proposed framework requires a conditional generator that maps a generalized-class condition and a random latent variable to a target-style input, *i.e.*, $G(\boldsymbol{z} \mid \bar{\boldsymbol{y}}) \to \hat{\boldsymbol{x}}_{\mathcal{G}}$. To examine whether the adaptation mechanism depends on a particular generator architecture, we instantiate this interface with a conditional generative adversarial network (CGAN) (Mirza & Osindero, 2014), a conditional variational autoencoder (CVAE) (Sohn et al., 2015), a conditional Wasserstein GAN with gradient penalty (CWGAN-GP) (Gulrajani et al., 2017), a conditional normalizing flow (CNF) (Papamakarios et al., 2021; Chen et al., 2018), and Conditional MeanFlow (Geng et al., 2026). This experiment tests whether the same adaptation framework works with different conditional-generator instances, not a controlled ranking of generator families at matched model capacity. We conduct the comparison on T#1 and T#4, which represent the NILM and load-forecasting tasks, respectively.

Controlled data protocol. For each task, all variants start from the same best source-model checkpoint used in the main experiment and the same 5,000 target sequences. We perform $K = 10$ stochastic forward passes with dropout enabled, use their mean as the generalized-class identifier, and use the mean predictive variance in Equation (76) as the sample uncertainty. The 20% most uncertain

Table 11: Experimental results on different pre-trained architectures (ResNet-seq2seq, CNN, Bi-LSTM, and TCN). "—" denotes that method failed to converge during training.

Method	T#1: D→U			T#4: N→G		T#1: D→U			T#4: N→G	
	MAE↓	SAE↓	F1↑	MAE↓	RMSE↓	MAE↓	SAE↓	F1↑	MAE↓	RMSE↓
	Architecture: ResNet-seq2seq					**Architecture: CNN**				
Zero-Shot	36.57	27.44	0.52	498.13	584.75	28.50	18.11	0.55	515.96	605.63
DeepCORAL (Sun & Saenko, 2016)	30.50	20.22	0.44	458.71	540.58	26.15	16.49	0.47	495.96	575.92
DANN (Ganin et al., 2016)	28.87	18.70	0.50	454.25	537.13	24.57	16.34	0.55	438.42	530.49
MDD (Zhang et al., 2019)	29.30	19.33	0.51	454.30	537.12	24.55	16.18	0.54	434.09	551.89
MMDA (Rahman et al., 2020)	30.16	20.16	0.46	451.11	533.55	25.50	17.16	0.41	433.40	556.21
CDAN (Source-avail.) (Long et al., 2018)	27.20	19.05	0.65	279.81	341.00	22.75	12.99	0.64	302.22	367.26
RSD (Chen et al., 2021)	—	—	—	456.85	539.59	23.76	15.98	0.56	418.64	492.95
SHOT (Liang et al., 2020)	64.34	53.06	0.13	693.38	829.55	27.98	17.13	0.32	504.18	593.11
NRC (Yang et al., 2021)	48.56	41.19	0.02	484.28	563.89	27.30	17.22	0.51	517.81	602.22
AaD (Yang et al., 2022)	60.74	54.12	0.17	404.19	528.45	26.68	15.73	0.55	437.64	578.56
SSA (Adachi et al., 2025)	35.23	26.98	0.57	305.25	381.93	21.61	13.61	**0.66**	301.25	339.85
MAPU (Ragab et al., 2023)	37.35	37.34	0.01	388.84	440.19	23.16	12.33	0.46	468.93	530.90
SFHarmony (Dinsdale et al., 2023)	36.27	28.01	0.04	327.03	403.74	22.18	11.92	0.16	344.54	397.40
GIRA (Ours)	**25.45**	**17.37**	**0.67**	**259.22**	**334.96**	**20.22**	**11.41**	0.59	**283.79**	**338.37**
Improvement	*30.41%*	*36.70%*	*28.85%*	*47.96%*	*42.72%*	*29.05%*	*37.00%*	*7.27%*	*45.00%*	*44.13%*
	Architecture: Bi-LSTM					**Architecture: TCN**				
Zero-Shot	28.19	17.64	0.49	970.15	1119.16	38.34	30.77	0.44	856.22	1576.85
DeepCORAL (Sun & Saenko, 2016)	28.03	17.75	0.51	556.55	713.08	32.06	21.70	0.41	556.94	743.80
DANN (Ganin et al., 2016)	25.98	14.36	0.43	516.62	650.33	34.38	25.75	0.50	485.43	681.92
MDD (Zhang et al., 2019)	25.45	14.72	0.43	510.29	646.93	32.23	20.50	0.48	469.21	684.70
MMDA (Rahman et al., 2020)	27.09	15.88	0.38	403.96	587.30	28.54	23.29	0.46	377.34	607.08
CDAN (Source-avail.) (Long et al., 2018)	22.44	**12.28**	**0.61**	401.56	492.75	27.76	**13.37**	**0.64**	353.49	606.14
RSD (Chen et al., 2021)	—	—	—	447.21	685.21	35.52	24.89	0.47	—	—
SHOT (Liang et al., 2020)	46.57	39.97	0.06	1083.93	1543.16	63.34	53.56	0.15	897.70	1599.60
NRC (Yang et al., 2021)	54.16	51.52	0.32	1225.27	1644.83	44.64	39.32	0.20	628.76	1084.73
AaD (Yang et al., 2022)	39.07	19.74	0.48	632.87	817.08	51.00	44.85	0.16	621.89	779.47
SSA (Adachi et al., 2025)	23.58	18.10	0.49	428.57	564.35	29.05	20.99	0.58	418.27	**552.06**
MAPU (Ragab et al., 2023)	46.69	40.20	0.02	1071.93	1460.59	31.21	25.90	0.09	481.09	676.54
SFHarmony (Dinsdale et al., 2023)	23.97	14.40	0.21	433.91	489.25	37.64	32.81	0.07	360.03	597.17
GIRA (Ours)	**21.01**	12.99	0.52	**391.97**	**458.81**	**26.48**	18.94	0.52	**337.27**	574.74
Improvement	*25.47%*	*26.36%*	*6.12%*	*59.60%*	*59.00%*	*30.93%*	*38.45%*	*18.18%*	*60.61%*	*63.55%*

samples are removed once; consequently, every generator receives exactly the same retained pairs $(\boldsymbol{x}_{\mathcal{T}}, \bar{\boldsymbol{y}})$. With the compatibility protocol enabled and seed 2025, we reserve the same 500 correctly paired retained inputs and conditions for label-free checkpoint scoring in all five variants and use the remaining pairs for generator updates. The scoring set is identical across the five generators and is not used for gradient updates. Target ground-truth labels are not accessed when constructing either set.

When $L_{out} \neq L_{in}$, CMF aligns the condition with a linear layer, while CGAN, CVAE, CWGAN-GP, and CNF use a linear layer followed by ReLU. CGAN applies its learned condition projection even when the input and output lengths are equal. Except for the global CVAE latent, all stochastic inputs have shape $L_{in} \times 1$, so each generated sample has shape L_{in}. The main configurations are summarized in Table 12, with implementation details given below. CGAN and CMF reuse the architectural backbones and defaults of the main codebase, while the other variants are used only in this compatibility experiment.

Shared semantic objective. All variants use $\mathcal{L}_{SEM} = \ell(N^*(G(\boldsymbol{z} \mid \bar{\boldsymbol{y}})), \bar{\boldsymbol{y}})$, where ℓ is Huber loss with $\delta = 1$ on T#1 and MSE on T#4. The base semantic coefficient is $\alpha_{SEM} = 3.3 \times 10^{-5}$. For CVAE, this term uses a conditional-prior sample, and for CNF it uses a forward ODE sample. The 10,000-update budget counts generator or encoder–decoder updates, excluding the additional critic updates of CWGAN-GP.

CGAN. The generator concatenates a length-L_{in} Gaussian noise sequence with the aligned condition and applies a two-layer unidirectional LSTM with hidden width 500, followed by Linear$(500, 1)$. The discriminator concatenates the input and condition, then applies a two-layer LSTM, spectral-normalized linear projections, and a time-wise sigmoid. Its BCE loss uses real, generated, and mismatched input–condition pairs with labels 0.9, 0.1, and 0.1, respectively. The mismatched condition $\bar{\boldsymbol{y}}^{\mathrm{cyc}}$ is formed by cyclically shifting the conditions by one position within each batch. The generator uses BCE, semantic consistency weighted by $100\alpha_{SEM}$ on T#1 and α_{SEM} on T#4, and input mean and standard-deviation matching weighted by $w_{\mathrm{STA}} = 0.08269$. The discriminator

Table 12: Key configurations of the conditional generators in the compatibility experiment. The architecture and objectives are detailed in the accompanying text.

Generator	Architecture and condition handling	Objective and optimization
CGAN	Length-aligned condition concatenated with Gaussian sequence, 2-layer LSTM, width 500	BCE $+\mathcal{L}_{SEM}+$ statistics matching, Adam with G/D learning rates 9.8567×10^{-5} and 1.5181×10^{-5}, 10,000 updates
CVAE	Length-aligned condition with Conv encoder–decoder, latent dimension 64	Reconstruction + KL $+\mathcal{L}_{SEM}$ with $\beta_{KL} = 10^{-3}$, Adam with learning rate 10^{-4}, 10,000 updates
CWGAN-GP	Same conditional LSTM generator as CGAN	Wasserstein + GP $+\mathcal{L}_{SEM}$ with $\lambda_{GP} = 10$, Adam with learning rate 10^{-4}, 10,000 generator updates and 5 critic updates per generator update
CNF	Length-aligned condition and time channel, 4 Conv blocks, width 128	NLL $+\mathcal{L}_{SEM}$ with RK4 using 8 steps, Adam with learning rate 10^{-4}, 10,000 updates
CMF	Four-channel input with aligned condition, 4 Conv stages, width 128	$\mathcal{L}_{MF} + \alpha_{SEM}\mathcal{L}_{SEM} + \mathcal{L}_{END} + 2\mathcal{L}_{TV}$, Adam with learning rate 9.8567×10^{-5}, 10,000 updates

updates when the real–fake score difference is below 0.56938 and always updates during the first 500 iterations.

CVAE. The encoder concatenates the target input and aligned condition and applies four stride-2 Conv1d stages with channels 32, 64, 128, and 128, followed by global average pooling and separate latent mean and log-variance heads. A 64-dimensional latent is projected to 64 channels, repeated along time, concatenated with the condition, and decoded by Conv1d stages with channels 65, 128, 128, 64, and 1. The objective combines reconstruction, KL divergence, and semantic consistency. Reconstruction uses Huber loss on T#1 and MSE on T#4. The KL coefficient increases linearly from zero to 10^{-3} over the first 2,000 updates. Training uses one reparameterized posterior sample per input, and generation draws from the conditional prior.

CWGAN-GP. Its generator has the same two-layer LSTM architecture as CGAN. The critic uses a two-layer conditional LSTM without sigmoid or spectral normalization and averages its time-wise outputs to produce a sequence score. The critic minimizes the difference between generated and real scores plus a gradient penalty with coefficient $\lambda_{GP} = 10$. The generator minimizes the negative generated score plus $\alpha_{SEM}\mathcal{L}_{SEM}$. Five critic updates are performed per generator update with the condition fixed during interpolation. Both networks use Adam with $\beta_1 = 0$, $\beta_2 = 0.9$, and learning rate 10^{-4}.

CNF. Its length-L_{in} Gaussian base is transformed by a conditional vector field that concatenates the current state, aligned condition, and broadcast time channel. The vector field uses four 128-channel Conv1d blocks and a 1×1 output convolution. Conditional negative log likelihood is computed by the continuous change-of-variables formula using one Rademacher Hutchinson vector per sample, and $\alpha_{SEM}\mathcal{L}_{SEM}$ is added. A fixed-step RK4 solver with eight steps is used for both likelihood evaluation and generation.

Conditional MeanFlow. The four input channels, four 128-channel convolutional stages, four-step sampling, and the $\mathcal{L}_{MF} + \alpha_{SEM}\mathcal{L}_{SEM} + \mathcal{L}_{END} + 2\mathcal{L}_{TV}$ objective follow the main-experiment implementation in Section B.3.

Generator training and checkpoint selection. All generator variants use batch size 256 and seed 2025. They are trained for exactly 10,000 generator updates and evaluated every 100 updates on the fixed 500-sample scoring set. Within each run, the same generalized-class conditions and a fixed set of random latent variables are used at every scoring event. The checkpoint with the largest label-free

generator-quality score is retained. Specifically,

$$\text{Score} = -0.05\,\text{MAE}(N^*(\boldsymbol{x}_\mathcal{G}), \bar{\boldsymbol{y}}) - 0.05\,\text{RMSE}(N^*(\boldsymbol{x}_\mathcal{G}), \bar{\boldsymbol{y}}) - 10\,\text{MMD}^2(\boldsymbol{x}_\mathcal{G}, \boldsymbol{x}_\mathcal{T})\,, \tag{75}$$

where a larger value indicates better joint semantic and distributional matching. MAE and RMSE are computed after inverse-transforming the outputs to their original task units, whereas MMD is computed between normalized generated and target inputs. The MMD bandwidth is determined once from the scoring target inputs using the median heuristic and is then shared by all generator variants. Each normalized sequence is treated as one vector sample. We use an equal-weight five-bandwidth RBF mixture with $\sigma_j = \widetilde{\sigma} 2^{-1+j/2}$ for $j = 0, \ldots, 4$, where $\widetilde{\sigma}$ is the median pairwise Euclidean distance among the scoring target inputs. Thus, the kernel is $k(\boldsymbol{x}, \boldsymbol{x}') = \sum_{j=0}^{4} \exp\big(-\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2/(2\sigma_j^2)\big)$. The reported MMD^2 uses the biased estimator with diagonal terms included and m^{-2} normalization for a scoring batch of m samples.

Step 2 adaptation and timing. For every selected generator checkpoint, we reload the same source checkpoint, freeze the generator, and optimize only the regression model. Step 2 uses Adam with learning rate 5.6×10^{-4}, batch size 256, 20 parameter-update iterations per epoch, and 40 epochs. At each update, a target minibatch is drawn, its current model predictions are passed to the frozen generator together with newly sampled noise, and the resulting 256 input–GC pairs are used in $\alpha_{GC}\mathcal{L}_{GC} + \alpha_{REG}\mathcal{L}_{REG} + \alpha_{SCC}\mathcal{L}_{SCC}$. Thus, every variant generates and consumes 5,120 synthetic samples per epoch and 204,800 over the complete adaptation run. We use the nominal coefficients in Table 6, clip the regression-model gradient norm at 1, and report the model after epoch 40 rather than selecting an epoch with target labels.

Time is the mean wall-clock duration of the Step 2 parameter-update loop over epochs 2–40. We call `torch.cuda.synchronize()` immediately before and after each timed loop. Data loading, end-of-epoch evaluation, logging, and checkpoint I/O are outside the timed region; generator training time is also excluded. Thus, Time measures the cost of Step 2 with an already selected generator and does not measure end-to-end training time from target inputs. All variants are timed in separate processes on the same RTX 4090 using FP32 with automatic mixed precision disabled.

Figure 8 provides a qualitative diagnostic of generator compatibility. Black and orange curves denote target-domain and generated inputs, while blue and red curves denote target and generated semantics. The generated curves generally follow the target-domain style and their semantic counterparts, with differences across generator instances.

Table 13: Compatibility of GIRA with different conditional generative models on T#1 and T#4. Time denotes the average Step 2 adaptation time per epoch in seconds and excludes generator training.

Generator	T#1: D→U					T#4: N→G			
	Score↑	Time↓	MAE↓	SAE↓	F1↑	Score↑	Time↓	MAE↓	RMSE↓
Zero-Shot	–	–	36.57	27.44	0.52	–	–	498.13	584.75
CGAN	**-5.16**	6.89	26.97	18.28	0.58	-16.73	6.83	260.49	**328.22**
CVAE	-17.49	8.92	26.68	21.72	0.56	-37.02	8.25	279.86	368.28
CWGAN-GP	-20.20	5.64	27.41	22.72	0.58	-12.59	**5.31**	262.14	338.07
CNF	-15.83	5.44	26.09	20.53	0.56	-55.14	5.35	291.59	349.36
CMF	-8.57	**5.11**	**25.45**	**17.37**	**0.67**	**-12.13**	5.36	**259.22**	334.96

The numerical results are reported in Table 13. All five conditional generators improve the corresponding pre-adaptation model on both tasks, supporting the compatibility of the inverse-regression framework with different generator families. CMF obtains the lowest MAE on both tasks, the lowest T#1 SAE, the highest T#1 F1, and the highest T#4 Score; it is also the fastest variant on T#1. CGAN achieves the highest T#1 Score and the lowest T#4 RMSE, while CWGAN-GP has the lowest T#4 adaptation time. The ranking of Score therefore does not perfectly match every downstream metric, which motivates reporting generator quality, task performance, and computation separately.

C.3 Generalized-Class Error, Uncertainty, and Filtering

UTF assumes that predictive uncertainty provides a useful ranking of unreliable generalized-class assignments. We examine this assumption from three complementary perspectives: the empirical

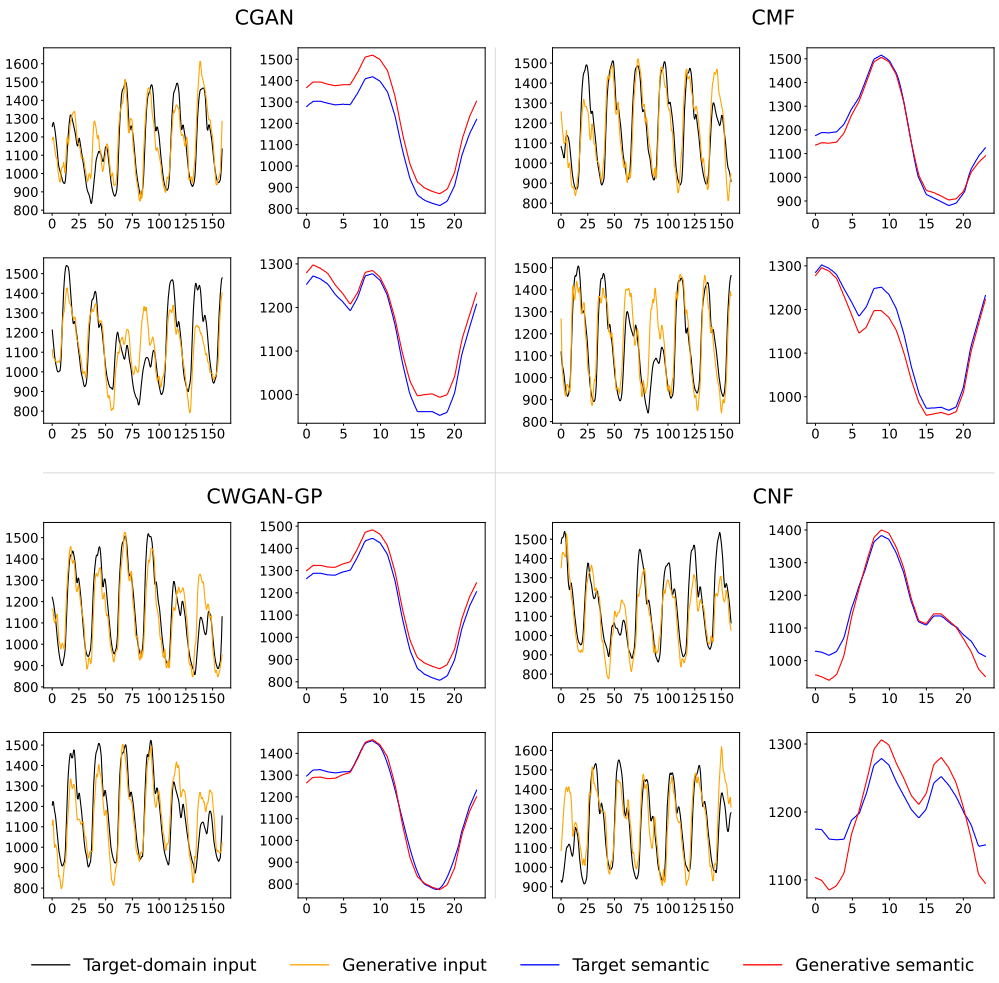


Figure 8: Examples of style and semantic similarity for samples generated by different conditional models on T#4.

relation between generalized-class error and uncertainty, the amount of target data removed, and the contribution of uncertainty-based ranking beyond the effect of retaining fewer samples.

C.3.1 Error–Uncertainty Relationship

For each T#1 target sample $\boldsymbol{x}_i$, we load the same source checkpoint and target split as in the main experiment and perform ten stochastic forward passes with dropout enabled. The generalized-class identifier is the pointwise mean $\bar{\boldsymbol{y}}_i = \frac{1}{10}\sum_{m=1}^{10} N_m^*(\boldsymbol{x}_i)$. For an output sequence of length L, the sample-level uncertainty is

$$u_i = \frac{1}{L}\sum_{t=1}^{L} \mathrm{Var}_{m=1,\ldots,10}\, N_m^*(\boldsymbol{x}_i)_t. \tag{76}$$

and the diagnostic sample-level generalized-class error is $e_i = \frac{1}{L}\sum_{t=1}^{L} |\bar{y}_{i,t} - y_{i,t}|$. We partition all target samples into the four fixed MAE intervals shown in Table 14 and compute the sample proportion and uncertainty statistics within each interval. Target labels are accessed only to compute e_i for this post hoc analysis; u_i depends only on the source model and unlabeled target inputs and is the quantity available to UTF during source-free adaptation.

Table 14: Relationship between generalized-class error and predictive uncertainty on T#1.

Generalized-class MAE bin	Sample proportion	Mean uncertainty	Median uncertainty	Maximum uncertainty
0–25	61.3%	0.08	0.06	0.27
25–40	24.7%	0.17	0.15	0.46
40–65	10.5%	0.31	0.29	0.73
> 65	3.5%	0.58	0.47	0.97

As shown in Table 14, both the mean and median uncertainty increase across the generalized-class MAE bins. The highest-error bin contains only 3.5% of the samples but has substantially larger uncertainty than the lower-error bins. These observations support uncertainty as a practical ranking signal for identifying a concentrated subset of high-risk generalized-class assignments; they do not require uncertainty to be a calibrated estimate of the numerical prediction error.

C.3.2 Effect of Filtering Ratio

We vary the removed ratio over $\{0\%, 10\%, 20\%, 30\%, 40\%, 50\%\}$, where

$$\text{removed ratio} = 1 - \text{keep ratio}. \tag{77}$$

For each of T#1 and T#4, we first compute the ten-pass generalized-class identifiers and uncertainties once and sort the 5,000 target samples in ascending order of u_i. For a removal ratio r, we retain the first $\lfloor (1-r)N_T \rfloor$ samples; the 0% condition uses all target samples. We then train a new CMF from its random initialization for 10,000 updates, select its checkpoint by Equation (75), reload the same source checkpoint, and run a new 40-epoch Step 2 adaptation. No generator or adapted-model checkpoint is shared across removal ratios. MeanFlow and Step 2 settings, including batch size 256, 20 Step 2 updates per epoch, optimizer, and loss coefficients, are fixed to Table 6. The target pool is evaluated offline after each epoch, but its labels do not enter optimization or checkpoint selection.

Table 15: Effect of the UTF filtering ratio on adaptation performance.

Removed ratio	T#1 MAE↓	T#1 SAE↓	T#1 F1↑	T#4 MAE↓	T#4 RMSE↓
0%	30.17	24.15	0.52	313.12	430.14
10%	26.62	22.23	0.59	263.49	339.34
20%	**25.45**	**17.37**	**0.67**	**259.22**	**334.96**
30%	28.83	20.60	0.56	259.35	335.09
40%	31.91	24.86	0.51	274.22	367.12
50%	31.19	25.97	0.48	319.39	387.87

The filtering-ratio results are reported in Table 15. Filtering 20% of the samples yields the lowest MAE and SAE and the highest F1 on T#1, as well as the lowest MAE and RMSE on T#4. The 20% and 30% settings perform similarly on T#4, whereas removing 40–50% of the samples degrades both tasks. These results reflect the trade-off between excluding uncertain generalized-class assignments and retaining sufficient target-domain coverage.

C.3.3 Effect of Uncertainty Ranking

To distinguish ranking quality from the effect of using fewer samples, we compare four removal rules at the same 20% ratio against an unfiltered model. Random filtering uses a permutation generated with seed 2025 and controls for sample quantity. Removing the lowest-uncertainty samples reverses the UTF ordering and tests whether the direction of that ordering is meaningful. UTF removes the highest-uncertainty samples and is the deployable strategy. The Oracle ranks samples by e_i and removes the highest-error 20%; because this construction accesses target labels, it is an inadmissible SFDA method and serves only as a label-informed diagnostic reference.

We evaluate the five rows in Table 16 independently on T#1 and T#4. For each condition, we apply the specified filtering rule, train CMF from scratch on the retained pairs, and then reload the same source checkpoint for 40 epochs of adaptation. No generator or adapted-model parameters are transferred between filtering strategies. The no-filtering row retains all 5,000 samples, whereas every other row retains 4,000 samples. Target labels are used only to construct the Oracle ranking and are not used by the other strategies.

Table 16: Comparison of filtering quantity and ranking quality. The Oracle ranking uses target labels only for post hoc diagnosis.

Filtering strategy	Removed ratio	T#1 MAE↓	SAE↓	F1↑	T#4 MAE↓	RMSE↓
No filtering	0%	30.17	24.15	0.52	313.12	430.14
Random filtering	20%	30.44	24.93	0.51	283.43	352.75
Remove lowest-uncertainty	20%	36.47	27.60	0.43	380.99	429.85
UTF: remove highest-uncertainty	20%	25.45	**17.37**	**0.67**	259.22	334.96
Oracle: remove highest-error	20%	**24.65**	20.17	0.61	**254.64**	**321.85**

UTF outperforms no filtering, random filtering, and the reversed uncertainty ranking on every metric. Random removal improves T#4 MAE and RMSE over using all samples, but degrades T#1 MAE, SAE, and F1. Removing the lowest-uncertainty samples causes the largest degradation. The Oracle has lower MAE on both tasks and lower T#4 RMSE, while UTF has lower T#1 SAE and higher F1. These results show that the gain of UTF comes from the uncertainty ordering rather than from reducing the sample count alone.

C.4 Dependence on Source-model Quality

We evaluate how the benefit of GIRA changes with the quality of the source model on T#4. Source checkpoints saved every five epochs from epoch 10 to epoch 50 provide a controlled quality trajectory while preserving the source architecture, source training run, and target split. This isolates checkpoint quality from differences in initialization or source data.

Each source checkpoint is processed independently as follows. First, we reset the random seed to 2025 and compute its generalized-class identifiers and uncertainties using ten stochastic passes over all target samples. Second, we remove the 20% highest-uncertainty samples. Third, we initialize a new CMF and train it for 10,000 updates using the configuration in Table 12; its checkpoint is selected by Equation (75). Fourth, we initialize Step 2 from the same source checkpoint, freeze the selected generator, and perform 40 epochs with 20 updates of batch size 256 per epoch. Finally, we evaluate the epoch-40 model with MAE and RMSE. Accordingly, the generalized-class identifiers, uncertainty threshold, retained samples, generator, and adapted model are all specific to a source checkpoint; in particular, no generator trained with the epoch-50 source model is reused for an earlier checkpoint.

For a source checkpoint with pre-adaptation error E_{before} and post-adaptation error E_{adapted}, the relative MAE gain is

$$\text{MAE gain} = \frac{E_{\mathrm{before}} - E_{\mathrm{adapted}}}{E_{\mathrm{before}}} \times 100\%. \tag{78}$$

The source-quality results are reported in Table 17. Adaptation is beneficial from epoch 15 onward, but the weakest epoch-10 source model undergoes a small MAE degradation of 1.63%. The gain is non-monotonic among the early checkpoints and becomes substantially larger once the source model reaches epoch 30, increasing to 45.00–47.96% for epochs 40–50. These results indicate that GIRA

Table 17: Dependence of adaptation benefit on source-model quality on T#4.

Source epoch	Zero-Shot MAE↓	Zero-Shot RMSE↓	Adapted MAE↓	Adapted RMSE↓	MAE gain↑
10	910.24	1038.57	925.09	1035.50	-1.63%
15	780.46	892.13	689.27	776.70	11.68%
20	692.18	801.37	644.10	733.02	6.95%
25	628.37	734.82	537.64	625.28	14.44%
30	582.15	681.44	398.94	501.73	31.47%
35	541.02	632.27	372.14	454.89	31.22%
40	516.08	603.95	283.84	344.65	45.00%
45	505.07	591.23	264.21	339.63	47.69%
50	498.13	584.75	**259.22**	**334.96**	**47.96%**

tolerates a range of source-model qualities but still requires a minimally reliable source predictor to avoid negative transfer.

C.5 Hyperparameter Sensitivity

We perform a sensitivity analysis on T#4 for α_{SEM}, α_{GC}, α_{REG}, and α_{SCC}. Their nominal values are 3.3×10^{-5}, 1.4×10^{-4}, 0.83, and 4.1×10^{-4}, respectively. For each coefficient, we multiply its nominal value by $\{0.25, 0.50, 0.75, 1.00, 1.25, 1.50, 1.75\}$ while holding the other three coefficients fixed.

For each α_{SEM} setting, we train CMF from scratch for 10,000 updates and select its checkpoint using Equation (75) before running Step 2. For the α_{GC}, α_{REG}, and α_{SCC} settings, we reuse the CMF trained with the nominal α_{SEM} and rerun only Step 2. Every Step 2 run starts from the same T#4 source checkpoint and uses seed 2025, the same UTF-retained subset, batch size 256, 20 updates per epoch, Adam with learning rate 5.6×10^{-4}, and 40 epochs. Each cell in Table 18 reports the MAE of the epoch-40 model.

Table 18: Sensitivity analysis of the loss coefficients on T#4. Each cell reports MAE; one coefficient is varied while all others remain at their nominal values.

Varied coefficient	0.25×	0.50×	0.75×	1.00×	1.25×	1.50×	1.75×
α_{SEM}	453.88	311.30	260.43	**259.22**	259.53	283.63	274.86
α_{GC}	325.21	294.56	263.89	**259.22**	262.24	265.56	264.74
α_{REG}	291.26	263.49	260.06	**259.22**	263.58	264.19	271.04
α_{SCC}	342.90	273.49	263.08	**259.22**	259.70	264.42	261.99

The nominal setting gives the lowest MAE for all four coefficients. Performance is relatively stable around 0.75–$1.25\times$, particularly for α_{SCC}, but degrades when the coefficients are strongly underweighted. The largest degradation occurs for α_{SEM} at $0.25\times$, showing that insufficient semantic supervision most strongly affects the method within the tested ranges.