Mixing Configurations for Downstream Prediction

Juntang Wang[1,†] **Hao Wu**[2,†] **Runkun Guo**[3] **Yihan Wang**[1] **Dongmian Zou**[1] **Shixin Xu**[1*]
[1]Division of Natural and Applied Sciences, Duke Kunshan University, China
[2]School of Cyber Science and Engineering, Sichuan University, China
[3]Boston University, USA

Abstract

Humans possess an innate ability to group objects by similarity—a cognitive mechanism that clustering algorithms aim to emulate. Recent advances in community detection have enabled the discovery of *configurations*—valid hierarchical clusterings across multiple resolution scales—without requiring labeled data. In this paper, we formally characterize these configurations and identify similar emergent structures in register tokens within Vision Transformers. Unlike register tokens, configurations exhibit lower redundancy and eliminate the need for ad hoc selection. They can be learned through unsupervised or self-supervised methods, yet their selection or composition remains specific to the downstream task and input. Building on these insights, we introduce GraMixC, a plug-and-play module that extracts configurations, aligns them using our novel Reverse Merge/Split (RMS) technique, and fuses them via attention heads before forwarding them to any downstream predictor. On the DSNI 16S rRNA cultivation-media prediction task, GraMixC improves the R^2 from 0.6 to 0.9 on various methods, setting a new state-of-the-art. We further validate GraMixC across standard tabular benchmarks, where it consistently outperforms single-resolution and static-feature baselines.

1 Introduction

Learning general-purpose features that enhance downstream tasks has been a long-standing goal in machine learning. One prominent example is clustering (*i.e.*, community detection) in unsupervised learning, which groups entities into clusters of similar objects while separating dissimilar ones, without using labels. [1]–[3]. Interestingly, this paradigm demonstrates remarkable similarities to human-like behaviors. Decades of cognitive science studies show that even infants have the ability to group objects by similarity [4], [5]. In particular, they often organize them at different abstraction levels [6], [7]. Inspired by this, recent advances in community detection have extended clustering to the discovery of *configurations*—hierarchical clusterings that span multiple resolution scales [8]. For example, as illustrated in the lineage diagram of Fig. 1, in the CIFAR10 dataset [9], coarse configurations may separate vehicles from animals, while finer configurations distinguish between birds, cats, and dogs. These multi-resolution representations reveal rich hierarchical structures that could provide stronger priors or inductive biases for deep models. However, despite their potential, such configurations remain largely underexplored in deep learning, especially in challenging domains where labels are sparse.

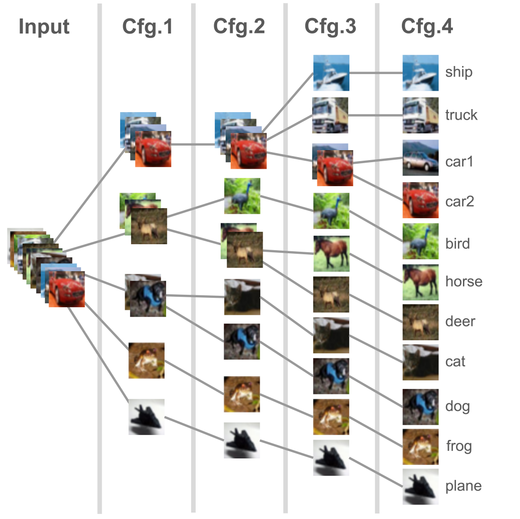


Figure 1: Illustration of CIFAR10 configurations. Each column represents a configuration—clustering at a specific resolution.

One such domain is 16S ribosomal RNA (rRNA) gene sequencing, a widely used tool in microbiome studies for identifying and classifying bacteria. Analyzing 16S rRNA data has consistently confronted significant challenges in downstream prediction tasks within label-scarce environments. Previous works in 16S rRNA representation learning have demonstrated

*Corresponding author: `shixin.xu@dukekunshan.edu.cn`

substantial benefits for bacterial taxonomic profiling and microbial community analysis [10]–[12]. Notably, Johnson et al. [13] showed that full-length sequencing combined with appropriate clustering of intragenomic sequence variation can provide more accurate representation of bacterial species in microbiome datasets. These findings underscore the importance of learning clustered representations without relying on labels.

Recent methodologies typically transform clustering results into pseudo-labels to enhance downstream prediction performance. For instance, DeepCluster [14] iteratively clusters CNN-extracted visual features and leverages these cluster assignments to guide network parameter updates. Graph-based methods such as [15] employ structural clustering to overcome limitations of traditional contrastive learning approaches that depend on positive and negative sample pairs. Their method captures structural relationships among nodes in heterogeneous information networks, establishing a self-supervised pre-training framework that learns robust network representations from unlabeled data. Nevertheless, aforementioned approaches predominantly focus on a single configuration type, overlooking the potential benefits of mixing configurations across multiple resolution scales.

In this paper, we introduce GraMixC, a plug-and-play module that extracts, aligns and mixes graph-based configurations for downstream prediction. The main contributions of the paper are as follows:

- We identify three key characteristics of clustering configurations through systematic experimental analysis, providing a novel perspective on enhancing downstream prediction via mixing configurations.
- We propose GraMixC, a plug-and-play module based on mixed configurations. We apply it to a novel 16S rRNA cultivation-media prediction task, setting a new state-of-the-art.
- We further conduct extensive experiments on multiple standard tabular benchmarks to validate GraMixC's effectiveness, where it consistently outperforms single-resolution and static-feature baselines.

The remainder of this paper is organized as follows. Section 2 analyzes behavioral patterns of configurations. Section 3 details our proposed GraMixC. Section 4 evaluates GraMixC's performance through extensive experiments. Finally, Section 5 concludes the paper.

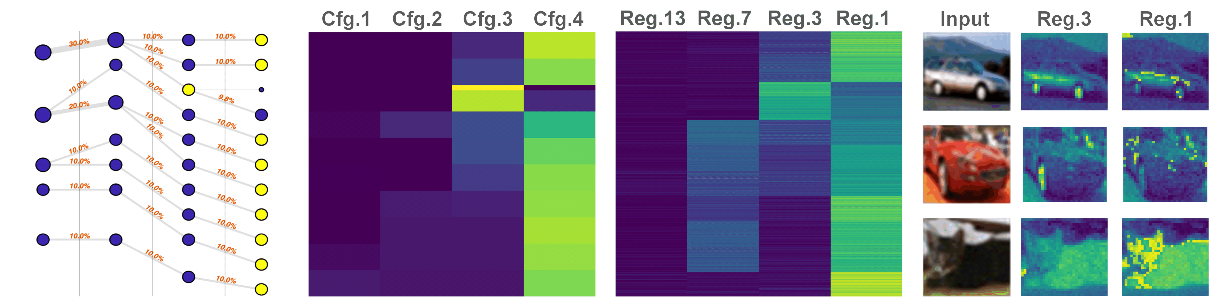
2 Preliminary results

We first present preliminary experimental results on configurations using CIFAR10. Specifically, we compare patterns of configurations with those of the learnable "register" tokens in a recent vision transformer DINOv2-reg [16]. Fig. 2 shows the attention maps from our configurations and their register tokens. Moreover, Fig. 3 shows qualitative behaviors of our configurations and their quantitative advantages over registers in terms of feature importance and neighborhood similarity. From these results, we identify three key properties:

Configurations emerge via unsupervised or self-supervised learning. We define Near ground truth (GT) balls as balls selected with the highest clustering scores, marked yellow in Fig. 2a. As shown in Fig. 2b, the attention map, acquired by feeding configurations as tokens to attention heads for linear probing, yields high norm regions substantially overlap with GT balls. On another hand, DINOv2-reg exhibits similar attention map patterns in selected registers (see Fig. 2c), which might be related to registers activating different areas in Fig. 2d, similar to slot attention [16]–[19]. Thus, based on the similar attention map behavior, register token can be considered as a latent configuration.

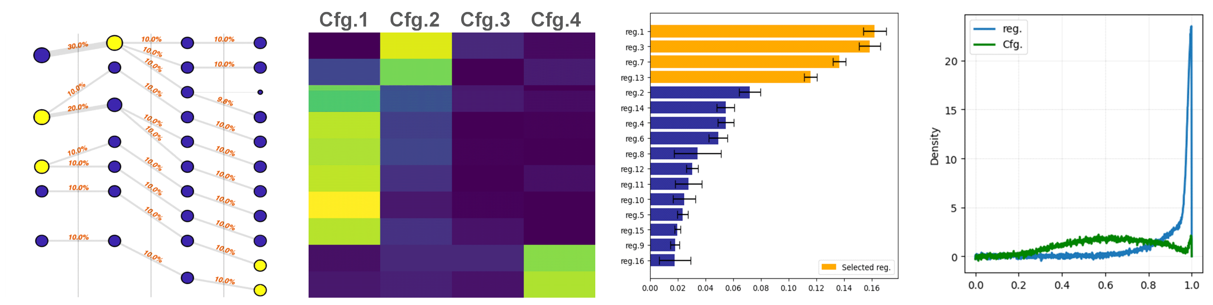
Configurations are selected and mixed based on input and task. *Configuration selection and mixing* refers to learning which resolution scales to focus on for a given downstream task. We visualize this via attention maps over configuration tokens, where high-norm regions indicate the selected scales. In Fig. 2b, attention norms vary across rows, showing that each input sample triggers different resolution scales. Without any change to the configurations, we merge the original labels into coarser classes (Fig. 3a) and plot the new attention map (Fig. 3b). The attention shifts to align with the coarser GT, whereas DINOv2-reg register tokens remain unchanged unless re-trained. These observations confirm that configuration selection and mixing are input- and task-dependent.

Configurations are more informative and less redundant than register tokens. Register tokens can help extract configurations, similar to object detection [20], [21], but selecting a fixed number by feature importance is arbitrary and non-rigorous (see Fig. 3c). Furthermore, register tokens exhibit high redundancy—cosine similarity between their embeddings and their 2 neighbors embeddings is heavily skewed toward 1—whereas configurations yield information less redundant (see Fig. 3d).



(a) Configurations (b) Cfg. attention map (c) Reg. attention map (d) Reg. attention imgs

Figure 2: Comparison of attention maps obtained from configurations and registers, rows for samples. **(a)**: Lineage diagram for configurations, near GT balls are marked yellow. **(b)**: Attention map of configuration tokens in an attention-based linear probing. **(c)**: Attention map of DINOv2-reg register tokens, mean of all patch norms is used. **(d)**: Attention maps over the register tokens, as images.



(a) Configurations (b) Cfg. attention map (c) Feature importance (d) Neighbors similarity

Figure 3: Illustration of another two properties of configurations, grouped by left two and right two. **(a)**: Lineage diagram where coarser classes are used for GT. **(b)**: Attention map in linear probing the coarser classes. **(c)**: Distribution of feature vector importance over the register tokens querying, mean of all patch importance is used. **(d)**: Distribution of cosine similarity between query embeddings of register and configuration tokens and their 2 neighbors, mean of all patch similarities is used.

3 Methodology

Having these characterizations, we hypothesize that unsupervised methods can produce hierarchical *multi-resolution clusterings*, and that task- and input-specific *selection and mixing* of these configurations represent *global information* beneficial to downstream tasks. Building on the hypothesis, we propose a lightweight module *GraMixC*, that treats configurations as tokens (`[CFG]`) and incorporates a novel alignment layer plus learnable attention heads [22] after the configuration extraction model, enabling task- and input-specific mixing of configurations via end-to-end back-propagation.

Fig. 4 illustrates GraMixC. Given an input matrix $\boldsymbol{X} \in \mathbb{R}^{N\times d}$ (with N samples and feature dimension d), GraMixC pass $\boldsymbol{X}$ to two branches: (1) a path to unsupervised learning box that extracts configurations, and (2) a direct path to the downstream predictor. If at inference, we apply *Reverse Merge & Split* (RMS) alignment on the configurations. Then we pass them to positional encoding (PE) and attention heads. The final concatenation is passed to a downstream predictor for the prediction $\tilde{\boldsymbol{y}}$.

Except for the downstream predictor, the GraMixC model can be divided into three parts: the unsupervised learning of configurations, the Reverse Merge & Split (RMS) for alignment, and attention heads for fusion. In the attention heads part, following Darcet et al. [16], we append register tokens `[REG]` after `[CFG]` and `[CLS]` for a clean attention map, that can be used backwards to guide configuration selection. Below we detail the rest two components in Section 3.1 and Section 3.2.

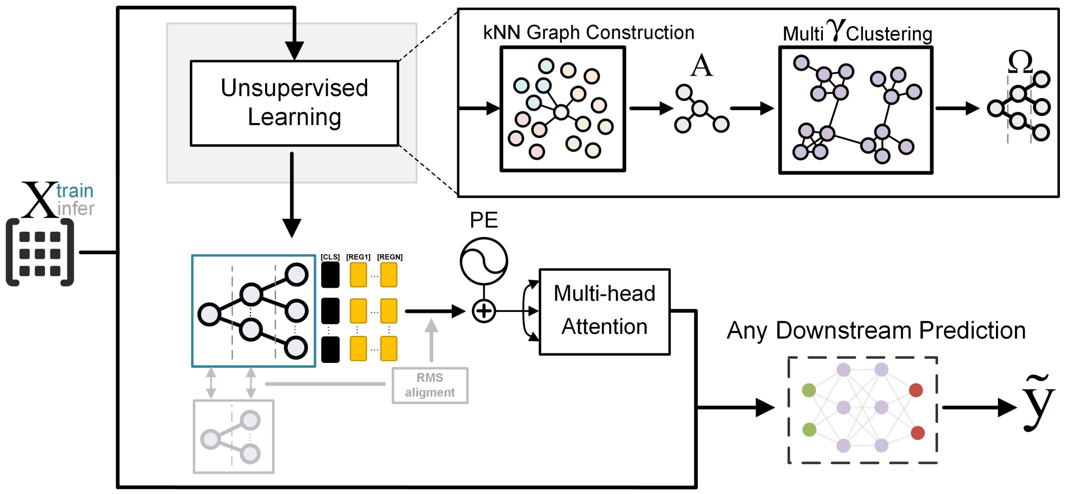


Figure 4: Illustration of the proposed GraMixC module and resulting model. The input data branches into (upper) a path to unsupervised learning box that extracts configurations, and (lower) a direct path to the downstream predictor. Their outcomes concatenate and pass to the downstream predictor. The components occur only during training and inference are colored in blue and gray, respectively.

3.1 Multi-resolution graph-based clustering

Given $\boldsymbol{X}$, multi-resolution clustering seeks to extract *configurations*—valid hierarchical clusterings across multiple resolution scales—which we denote as $\boldsymbol{\Omega} \in \mathbb{N}^{N\times m}$, where m denotes the number of valid resolution levels. To preserve the latent manifold structure in data, ease parameter sensitivity, and prevent other problems with traditional clustering methods (see Appendix C), we choose the resolution parameter ($\gamma \in \mathbb{R}_+$)-based community detection as our core clustering method. While BlueRed [23] can conduct graph clustering without problems like resolution limit or parameter sensitivity in traditional methods, recent work by Pitsianis et al. [8] further demonstrates the elimination of γ selection, and enabled the unsupervised discovery of $\boldsymbol{\Omega}$ and the corresponding set of all valid γ, which is denoted as $\Gamma = \{\gamma_1^*, \gamma_2^*, \ldots, \gamma_m^*\} \subseteq [0, \infty)$. Inspired by these works, the unsupervised box in Fig. 4 unfolds into two steps: **(1) k-nearest neighbors (kNN)** [24] **graph construction**, which return a directed graph $G = (V, E)$, usually represented as adjacency matrix $\boldsymbol{A} \in \mathbb{R}_+^{N\times N}$, and **(2) multi-$\gamma$ clustering** on the resulted graph, *i.e.* modularity based community detection with unsupervised Γ learning, which return the wanted $\boldsymbol{\Omega}$. The details for each of these two steps are:

(1) kNN graph construction. We construct a kNN graph with $k = \log_{10} N$ as convention, using Euclidean distance for simplicity. Such pair-wise geometric distance between two different vertexes is denoted $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ where $i \neq j$ and $x_i \in \mathbb{R}^d$ is the i-th feature vector. We then have the adjacency matrix $\boldsymbol{A}$ formulated as: $A_{ij} = d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ if $(\boldsymbol{x}_i, \boldsymbol{x}_j) \in E$, 0 otherwise, where E is the edge set of the kNN graph and A_{ij} denotes the i-th row and j-th column element of the adjacency matrix. Then we force *column stochastic* by dividing each column in the constructed $\boldsymbol{A}$ with the column sum. The resulted graph is sparse stochastic, and we can apply Stochastic Graph t-SNE (SG-t-SNE) reweighting [25], which proved to remedy skewed degree distribution, that is not promised by conventional t-SNE [26]. From the original work, the key equations for SG-t-SNE reweighting are:

$$w(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{1}{\lambda} \exp\left(-\frac{d^2(\boldsymbol{x}_i, \boldsymbol{x}_j)}{2\sigma_i^2}\right), \quad \text{with} \quad \lambda = \sum_{\boldsymbol{x}_j : (\boldsymbol{x}_i, \boldsymbol{x}_j) \in E} \exp\left(-\frac{d^2(\boldsymbol{x}_i, \boldsymbol{x}_j)}{2\sigma_i^2}\right),$$

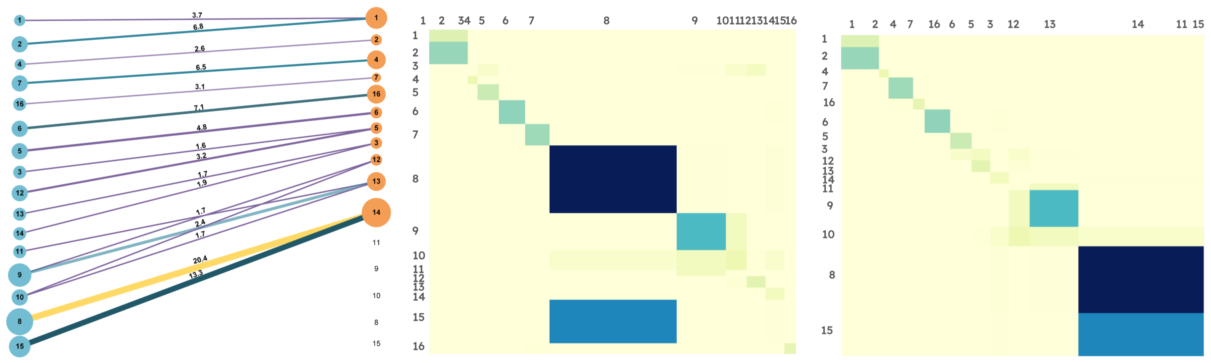
where λ is a non-negative parameter constant, which we simply set to 15 as previous work show that it is not so sensitive to the choice of λ [25], and σ_i is a variable to be numerically solved with bisection method. After giving value of w to d, we have $\boldsymbol{A}$ with less skewed degree distribution, which avoids problems like numerical instability and bias towards hubs in downstream clustering.

(2) multi-γ community detection. Then one may simply pass the reweighted $\boldsymbol{A}$ to γ-based community detection method, such as Leiden algorithm [27], to get one pseudo-configuration vector $\boldsymbol{\omega}_\gamma \in \{1, \ldots, N\}^N$ ("pseudo" for not sure to be valid). However, such γ falls in the range of $[0, \infty)$, and searching over all possible γ is exhausting. Therefore, we incorporate the BlueRed method with parallel descending triangulation (parallel-DT) [8], in order to automatically discover all valid $\gamma^* \in \Gamma$. Given a fixed γ, BlueRed find the optimal configuration $\boldsymbol{\omega}_\gamma$ by the following optimization:

$$\boldsymbol{\omega}_\gamma = \underset{\boldsymbol{\omega}\in\{1,\dots,N\}^N}{\arg\min}\left[-\sum_{k=1}^{|\boldsymbol{\omega}|_\infty}\sum_{(i,j)\in E} d(\boldsymbol{x}_i,\boldsymbol{x}_j)\mathbf{1}_{\omega_i=\omega_j=k} + \gamma\sum_{k=1}^{|\boldsymbol{\omega}|_\infty}\sum_{(i,j)\in E} d^2(\boldsymbol{x}_i,\boldsymbol{x}_j)\mathbf{1}_{\omega_i=k},\right],$$

where ω_i denotes the i-th element of $\boldsymbol{\omega}$, $|\boldsymbol{\omega}|_\infty = \max_{i\le N}\omega_i$ is a inf-norm, and $\mathbf{1}$ denotes the indicator gate which take value 1 if its subscript condition holds, 0 otherwise. Pitsianis et al. [8] describe the first term as attraction and the second term as repulsion. Optimizing each solely yields all-in-one configuration $\boldsymbol{\omega}_0 = [1,1,\dots,1]$ and all-lonely configuration $\boldsymbol{\omega}_\infty = [1,2,\dots,N]$. Between these two configurations, parallel-DT allows forming BlueRed Front (BRF) [8] by segmenting $(0,\infty)$ into m ranges, among which each has a dominant γ_i^* yields lower HAR [8]—the sum of first term and the negative second term—which means "local minimum" on that range. Thus desired $\boldsymbol{\Omega}$ is formed.

3.2 RMS: reverse merge & split alignment



(a) Bipartite graph (b) Proportional confusion matrix (c) Reordered confusion matrix

Figure 5: Example of the RMS alignment process applied to clustering results and ground truth (both treated as configurations) on the Salinas dataset [28]. **(a)**: Bipartite graph representation, where blue nodes correspond to predicted clusters and red nodes to ground truth clusters. Node labels indicate cluster indices; edge labels show the proportion of samples shared between clusters. **(b)**: Proportional confusion matrix $\boldsymbol{C}$ comparing predicted clusters (horizontal axis) to ground truth clusters (vertical axis). **(c)**: Confusion matrix $\boldsymbol{C}_{\text{tw}}$ reordered via the two-walk Laplacian. Notable splits, such as ground truth cluster 8 being divided into clusters 8 and 15 in the prediction, can be resolved through the reverse merge/split procedure.

Multi-resolution clustering on different datasets $\boldsymbol{X}_{\text{train}}$ and $\boldsymbol{X}_{\text{test}}$ often naturally produces misaligned configurations, that either (1) have different value of m or $|\boldsymbol{\omega}|_\infty$, or (2) have different cluster labels. While (2) is not a problem as re-assigning fix it, (1) could be problematic as the length and position of configurations influence the downstream fusion. One possible interpretation is that some clusters are further merged or split in another configuration, leading to this mismatch. To address this, we propose Reverse Merge & Split (RMS), which identifies an optimal alignment, allowing re-merging and re-splitting, between two configurations, $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_j$. First of all, an alignment score is defined:

$$\text{SCORE}(\boldsymbol{\omega}_i,\boldsymbol{\omega}_j) = \text{ARI}(\boldsymbol{\omega}_i,\boldsymbol{\omega}_j) - \theta\left|\frac{|\boldsymbol{\omega}_i|_\infty - |\boldsymbol{\omega}_j|_\infty}{|\boldsymbol{\omega}_i|_\infty + |\boldsymbol{\omega}_j|_\infty}\right|.$$

where θ is a hyperparameter to balance the weights of the two terms, which we set to 0.1, ARI is the adjusted rand index as defined in Hubert and Arabie [29]. By this punished ARI design, we consider different labels, merge and split during scoring the alignment between two partition, but also avoids too much difference in number of clusters (one extreme case is $\boldsymbol{\omega}_0$ and $\boldsymbol{\omega}_\infty$ has ARI of 1).

However, the SCORE itself does not convey the mapping we need for reassigning. In *RMS* alignment, we construct a confusion matrix $\boldsymbol{C}\in\mathbb{N}^{|\boldsymbol{\omega}_i|_\infty\times|\boldsymbol{\omega}_j|_\infty}$ between $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_j$. Fig. 5 illustrates this process with a concrete example, showing how the confusion matrix captures the relationship between predicted and ground truth clusters, including cases where clusters are split or merged across configurations. As an assignment problem with a rectangle cost matrix $-\boldsymbol{C}$[2], it is solvable by twisting existing Hungarian algorithm methods [30]–[32]. Because $\boldsymbol{C}$ *is the adjacency matrix of a bipartite graph*, spectral reordering via its graph Laplacian is preferred, since it encodes global connectivity and reveals

[2]The negative of the confusion matrix is used to frame the assignment problem (minimizing the diagonal).

coherent split–merge structures rather than merely optimizing diagonal entries. As the Fiedler vector reordering [33] assumes symmetric positive semi-definite, it is not directly applicable to $\boldsymbol{C}$. Inspired by a recent work of Floros, Pitsianis, and Sun [34], we introduce a *two-walk Laplacian*, which is defined as:

$$\boldsymbol{L}_{\text{tw}} = \boldsymbol{D} - \boldsymbol{C}_{\text{tw}}, \quad \text{with} \quad \boldsymbol{C}_{\text{tw}} = \begin{bmatrix} \boldsymbol{C}\boldsymbol{C}^\top & \boldsymbol{C} \\ \boldsymbol{C}^\top & \boldsymbol{C}^\top\boldsymbol{C} \end{bmatrix},$$

where $\boldsymbol{D} = \operatorname{diag}(\boldsymbol{C}_{\text{tw}}\mathbf{1})$ is the diagonal degree matrix of $\boldsymbol{C}_{\text{tw}}$. We remap $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_j$ by using, respectively, the first $\|\boldsymbol{\omega}_i\|_\infty$ and the last $\|\boldsymbol{\omega}_j\|_\infty$ entries in the Fiedler eigenvector of $\boldsymbol{L}_{\text{tw}}$, which is the eigenvector corresponds to smallest positive eigenvalue. We further reverse split and merge simply by reassigning the redundant columns or rows who has element larger than its diagonal entry.

In GraMixC, we carry a small portion (0.1%) of train samples as *anchors* during inference, and the portion of $\boldsymbol{\Omega}_{\text{train}}$ and $\boldsymbol{\Omega}_{\text{test}}$ corresponding to the anchors are used to calculate the SCORE. Given m is usually small, we exhaustively test pairs $(\boldsymbol{\omega}_i, \boldsymbol{\omega}_j)$ then iteratively pick the pair yielding the highest SCORE for each $\boldsymbol{\omega}_i$. For each pair, we apply the mapping from RMS$(\boldsymbol{\omega}_i, \boldsymbol{\omega}_j)$. The final alignments is then used to match the configurations. See Appendix D for alignment examples and more implementation details.

4 Experiments

In this section, we evaluate the proposed plug-and-play module by training baseline models with and without GraMixC (*GMC*). We also test a static variant (*GC*), which use aligned configurations as extra features, without attention mechanism. We expect the performance to follow a general trend

$$\text{baseline} < \text{baseline+GC} < \text{baseline+GMC}.$$

We then ablate the number of configurations used to check that they cause a performance regression.

4.1 Implementation details and experimental setup

Our module was implemented with MATLAB, Python 3.12, PyTorch 2.6. We run trainings on a GeForce RTX 3090Ti GPU. Models were trained with the Adam optimizer [35] at a fixed learning rate of 10^{-3}. Unless otherwise noted, we used a batch size of 100 and trained for up to 100 epochs.

Ahead of diving into the experimental details, we briefly summarize the datasets and metrics used.

DSNI-pH and DSNI-Temp. We collected the DSNI dataset from DSMZ [36] and NIH [37]. It comprises six relational tables (STRAINS, MEDIA, SOLUTIONS, INGREDIENTS, STEPS, GAS) covering taxonomic and protocol information. We use approximately 65 000 samples with 16S rRNA sequence (500–1 500 nucleotides), cultivation temperatures (2–103 °C), and pH (0.1–11). The task is to predict optimal temperature (DSNI-Temp) and pH (DSNI-pH) from the 16S rRNA sequence.

Following Çelikkanat, Masegosa, and Nielsen [38] and related works [39], [40], we encode each 16S rRNA sequence as a 7-mer count vector in $\mathbb{N}^{16\,384}$, yielding a dataset of shape $65\,023 \times 16\,384$. We perform an 80/20 split (52,018 train / 13,005 test), which preserves the skewed pH (6–8) and temperature (20–40 °C) distributions. Appendix B provides an illustration for target value (y_{train} and y_{test}) distribution. Preprocessing—robust scaling, variance thresholding, and selection of the top 1,000 features—was fitted on the training set and then applied to both splits to avoid data leakage.

Additional benchmarks. We further evaluate on QM9 [41] for molecular property regression, on Boston Housing [42], and on MNIST [43] and CIFAR10 for classification (some in Appendix E).

Evaluation metrics. For regression we use mean squared error (MSE), mean absolute error (MAE; used for QM9 for comparability with SOTA) for training, and report coefficient of determination (R^2). For classification we use cross-entropy loss (CE) for training and report top-1 accuracy (Acc).

For each benchmark, we include three classical decision tree models for reference: Random Forest (RF) [44], XG-Boost [45], CatBoost [46]. As both GMC and GC are plug-and-play modules, they can be easily applied to various downstream predictors. We first evaluate a 3-layer perceptron (3LP) with hidden dims [256,128,64]. Because our inputs combine numerical features with categorical configurations, we naturally consider tabular models: TabNet (TabN) [47], TabTransformer (TabT) [48], FT-Transformer (FTT) [49] were all run with their default settings from the official implementations.

Table 1: Regression performance on DSNI-pH, DSNI-Temp and QM9. Values are mean±std from runs with different random seeds; best results per baseline are bold; best results per metric are underlined.

	DSNI-pH		DSNI-Temp		QM9	
	MSE ↓	R^2	MSE ↓	R^2	MAE ↓	R^2
RF	0.198±0.000	0.601±0.001	17.759±0.276	0.393±0.009	0.015±0.000	0.979±0.000
XGBoost	0.196±0.001	0.604±0.003	18.212±0.543	0.377±0.018	0.014±0.001	0.978±0.001
CatBoost	0.193±0.001	0.610±0.002	17.375±0.398	0.406±0.013	0.014±0.000	0.978±0.002
3LP	0.201±0.002	0.595±0.006	18.484±0.183	0.368±0.006	0.018±0.001	0.958±0.001
3LP+GC	0.097±0.004	0.804±0.008	6.520±0.360	0.777±0.012	0.016±0.003	0.974±0.000
3LP+GMC	**0.023**±0.002	**0.953**±0.004	**2.277**±0.061	**0.922**±0.002	**0.010**±0.003	**0.990**±0.002
TabN	0.184±0.004	0.629±0.007	13.290±0.244	0.545±0.008	0.015±0.001	0.962±0.002
TabN+GC	0.086±0.003	0.825±0.007	7.997±0.210	0.726±0.007	0.012±0.002	0.983±0.001
TabN+GMC	**0.020**±0.001	**0.959**±0.002	**0.989**±0.361	**0.966**±0.012	**0.008**±0.000	**0.995**±0.002
TabT	0.256±0.007	0.483±0.014	18.910±0.247	0.353±0.008	0.434±0.008	0.921±0.008
TabT+GC	0.106±0.002	0.786±0.005	8.280±0.303	0.717±0.010	0.212±0.004	0.961±0.008
TabT+GMC	**0.017**±0.002	**0.964**±0.005	**2.785**±0.540	**0.904**±0.018	**0.009**±0.000	**0.998**±0.001
FTT	0.218±0.003	0.561±0.006	13.571±0.069	0.536±0.002	0.085±0.005	0.984±0.006
FTT+GC	0.070±0.003	0.858±0.007	5.915±0.277	0.797±0.009	0.034±0.002	0.993±0.003
FTT+GMC	**0.007**±0.005	**0.984**±0.009	**1.480**±0.120	**0.949**±0.004	**0.026**±0.001	**0.995**±0.003

4.2 Evaluation of the proposed module

As shown in Fig. 2 and Fig. 3, we demonstrate, with attention maps, the learned mixing of configurations by training models with self-attention head on aligned configurations. In order to quantify the quality of such mixing, for each baseline, we set up the evaluation in three modes: standalone (baseline), with static configuration concatenation (baseline+GC), and with attention-based fusion via GraMixC (baseline+GMC). Table 1 reports regression results on our main benchmarks; Appendix E (Table 2) shows the rest results. Across all models and tasks, adding GC yields consistent gains, and incorporating GMC provides further significant improvements, confirming our initial hypothesis.

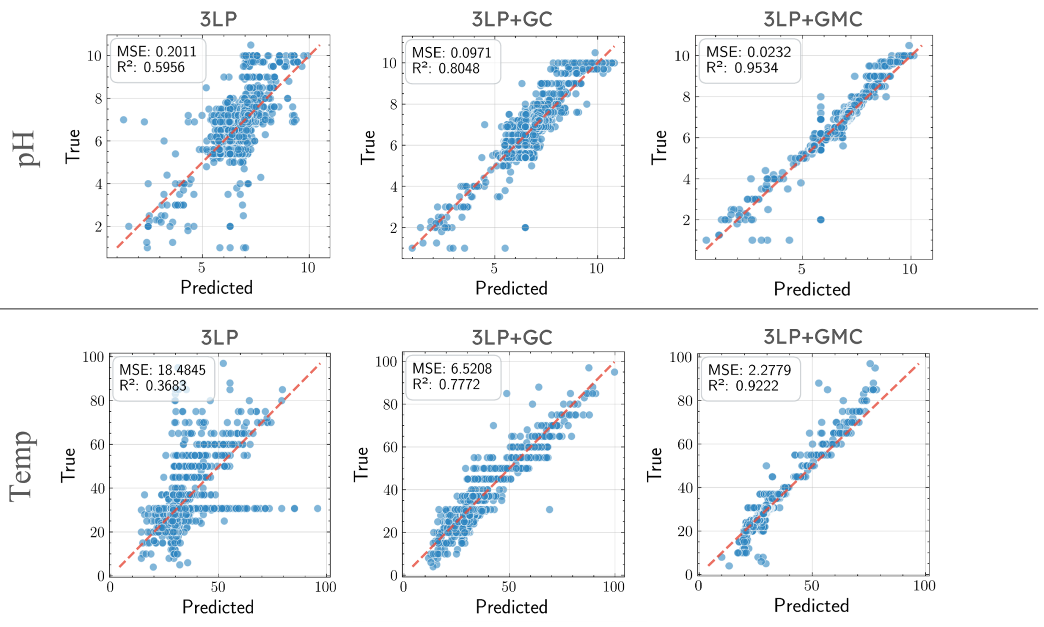


Figure 6: Illustration of the regression performance improvement example in 3LP by adding GC or GMC. Each column plots predicted vs. actual pH (top) or temperature (bottom). 3LP+GC (middle) outperforms the 3LP baseline (left), while 3LP+GMC (right) further boosts R^2 up to > 0.9.

Performance improvement. Table 1 shows that adding GC and GMC yields consistent gains across all baselines. Among these observed improvements, the scores increasing on DSNI is quite satisfying. Prior specialized growth-media regression methods are not convincing with $R^2 \leq 0.8$ (e.g., 0.75 [50]). We confirm this with our base models score R^2 between 0.3 and 0.6 on DSNI-pH and DSNI-Temp. However, even without tailoring the baseline model design, we bring the score to a new high by simply adding GC or GMC. Fig. 6 illustrates some examples of such improvement. We see the model's predictions align more closely with the ideal regression line and better handle rare cases, by incorporating configurations and probably capturing the latent manifold structure. Incorporating GC and further GMC raises R^2 to 0.98 (pH) and 0.97 (Temp). Which not only is considered very satisfying in application of bacterial cultivation but also set the new state-of-the-art (SOTA) for growth-media prediction. On QM9, GraMixC achieves an MAE of 0.008, nearly matching the SOTA (w/o extra training data) of 0.007 [51], and represents the best result among non-GNN models.

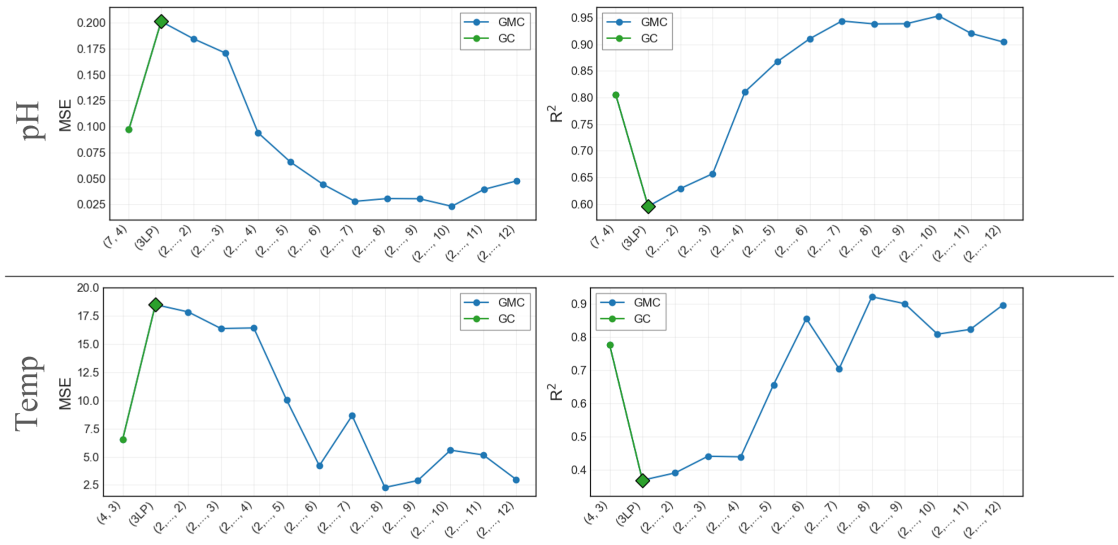
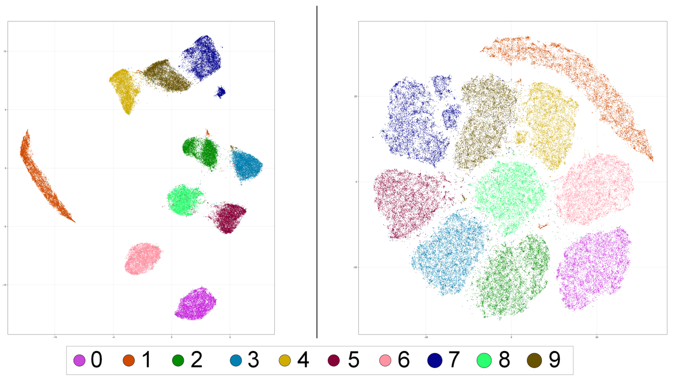


Figure 7: Ablation study on the number of configurations used on DSNI. On the blue curves (GMC), $[2, \ldots, i]$ denote fusing configurations from 2 through i via GraMixC. On the green curves (GC), (i, j) denote the best train/test configuration pair used in static concatenation. Incrementally mixing configurations improves performance and outperforms static concatenation.

Number of configurations used. We ablate the number of configuration levels in GMC. Fig. 7 shows that more configurations generally decreases MSE and increases R^2, confirming the value of multi-resolution information. Importantly, GMC often needs more than half as many total configurations to outperform GC, and performance plateaus—or even slightly declines—when including the last few configurations. These aligns with Pitsianis et al. [8], who report a finite set of optimal configurations rather than continuous gains at infinite resolutions. Using all configurations available is still preferred.

4.3 Qualitative evaluation of configurations.

Our final experiment compares configurations against standard representation-extraction methods. As discussed in Section 1, configurations can be viewed as special unsupervised representation learning. Fig. 3 already shows their advantage over self-supervised register tokens. Here, we replace GC/GMC with PCA [52], UMAP [53], and a vanilla autoencoder (AE), each embed into dimensions the same number of as our configurations. We visualize these embeddings on MNIST (Fig. 8a; additional views in Appendix E.2). Qualitatively, SG-t-SNE (the reduction step in GraMixC) yields more uniform, well-separated clusters that respect global kNN connectivity rather than forming hubs. Fig. 8b quantifies downstream classification accuracy, where GC and GMC strongly outperform PCA, UMAP, and AE given the same embedding budget. These results confirm that mixed configurations provide a more expressive yet compact representation for downstream tasks.



(a) 2D visualization of embeddings learned.

	CE ↓	Acc
3LP+PCA	0.157	0.971
3LP+UMAP	0.181	0.975
3LP+AE	0.158	0.969
3LP+GC	0.046	0.992
3LP+GMC	**0.028**	**0.993**

(b) Classification performance.

Figure 8: **(a)**: Illustration of 2D embeddings of MNIST using UMAP (left) and SG-t-SNE (right). **(b)**: Classification performance on MNIST using features from PCA, UMAP, autoencoder (AE), static configurations (GC), and GraMixC (GMC) at equal embedding dimensions. SG-t-SNE embeddings integrated via GC or GMC exploit multi-resolution structure to notably outperform other methods.

5 Conclusion

In this study, we investigate the functional mechanisms of configurations in downstream prediction tasks and identify three key properties. Based on this, we propose GraMixC, which dynamically mixes configurations through attention head. We apply it to the challenging task of 16S rRNA cultivation-media prediction task, and set a new state-of-the-art. Further validation across multiple standard tabular data benchmarks consistently reveals that GC (a static version of GraMixC) enhances baseline performance, while GraMixC demonstrates even more substantial improvements. Our results suggest that harnessing rich manifold priors via attention-driven fusion opens promising avenues for interpretable and robust learning in both scientific and conventional domains.

In future work, we plan to extend mixed configurations to more expressive networks and dynamically learn configuration alignment through end-to-end differentiable modules. Additionally, we will focus on exploring adaptive clustering for evolving data streams where train and test distributions may shift, which could further enhance the resilience of multi-resolution approaches.

References

[1] J. MacQueen. “Some Methods for Classification and Analysis of Multivariate Observations”. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Vol. 5.1. University of California Press, Jan. 1, 1967, pp. 281–298.

[2] Jianbo Shi and J. Malik. “Normalized Cuts and Image Segmentation”. In: *IEEE Trans. Pattern Anal. Machine Intell.* 22.8 (Aug. 2000), pp. 888–905.

[3] A. Ng, M. Jordan, and Y. Weiss. “On Spectral Clustering: Analysis and an Algorithm”. In: *Advances in Neural Information Processing Systems*. Vol. 14. MIT Press, 2001.

[4] P. C. Quinn and P. D. Eimas. “Perceptual Cues That Permit Categorical Differentiation of Animal Species by Infants”. In: *J Exp Child Psychol* 63.1 (Oct. 1996), pp. 189–211. PMID: 8812045.

[5] M. H. Bornstein, M. E. Arterberry, and C. Mash. “Infant Object Categorization Transcends Diverse Object-Context Relations”. In: *Infant Behav Dev* 33.1 (Feb. 2010), pp. 7–15. PMID: 20031232.

[6] L. Zaadnoordijk, T. R. Besold, and R. Cusack. “Lessons from Infant Learning for Unsupervised Machine Learning”. In: *Nat Mach Intell* 4.6 (June 2022), pp. 510–520.

[7] L. Muttenthaler, K. Greff, F. Born, B. Spitzer, S. Kornblith, M. C. Mozer, K.-R. Müller, T. Unterthiner, and A. K. Lampinen. *Aligning Machine and Human Visual Representations across Abstraction Levels*. Oct. 29, 2024. arXiv: 2409.06509 [cs]. URL: http://arxiv.org/abs/2409.06509 (visited on 05/11/2025). Pre-published.

[8] N. Pitsianis, D. Floros, T. Liu, and X. Sun. “Parallel Clustering with Resolution Variation”. In: *2023 IEEE High Performance Extreme Computing Conference (HPEC)*. 2023 IEEE High Performance Extreme Computing Conference (HPEC). Boston, MA, USA: IEEE, Sept. 25, 2023, pp. 1–8.

[9] A. Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Toronto, ON, Canada, 2009, pp. 32–33.

[10] J. M. Janda and S. L. Abbott. "16S rRNA Gene Sequencing for Bacterial Identification in the Diagnostic Laboratory: Pluses, Perils, and Pitfalls". In: *J Clin Microbiol* 45.9 (Sept. 2007), pp. 2761–2764. PMID: 17626177.

[11] Q. Wang, G. M. Garrity, J. M. Tiedje, and J. R. Cole. "Naive Bayesian Classifier for Rapid Assignment of rRNA Sequences into the New Bacterial Taxonomy". In: *Appl Environ Microbiol* 73.16 (Aug. 2007), pp. 5261–5267. PMID: 17586664.

[12] J. De Vrieze, A. J. Pinto, W. T. Sloan, and U. Z. Ijaz. "The Active Microbial Community More Accurately Reflects the Anaerobic Digestion Process: 16S rRNA (Gene) Sequencing as a Predictive Tool". In: *Microbiome* 6.1 (Apr. 2, 2018), p. 63.

[13] J. S. Johnson, D. J. Spakowicz, B.-Y. Hong, L. M. Petersen, P. Demkowicz, L. Chen, S. R. Leopold, B. M. Hanson, H. O. Agresta, M. Gerstein, E. Sodergren, and G. M. Weinstock. "Evaluation of 16S rRNA Gene Sequencing for Species and Strain-Level Microbiome Analysis". In: *Nat Commun* 10.1 (Nov. 6, 2019), p. 5029.

[14] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. *Deep Clustering for Unsupervised Learning of Visual Features*. Version 2. Mar. 18, 2019. arXiv: 1807.05520 [cs]. URL: http://arxiv.org/abs/1807.05520 (visited on 05/13/2025). Pre-published.

[15] Y. Yang, Z. Guan, Z. Wang, W. Zhao, C. Xu, W. Lu, and J. Huang. *Self-Supervised Heterogeneous Graph Pre-training Based on Structural Clustering*. Apr. 12, 2023. arXiv: 2210.10462 [cs]. URL: http://arxiv.org/abs/2210.10462 (visited on 05/13/2025). Pre-published.

[16] T. Darcet, M. Oquab, J. Mairal, and P. Bojanowski. *Vision Transformers Need Registers*. Apr. 12, 2024. arXiv: 2309.16588 [cs]. URL: http://arxiv.org/abs/2309.16588 (visited on 03/25/2025). Pre-published.

[17] F. Locatello, D. Weissenborn, T. Unterthiner, A. Mahendran, G. Heigold, J. Uszkoreit, A. Dosovitskiy, and T. Kipf. *Object-Centric Learning with Slot Attention*. Oct. 14, 2020. arXiv: 2006.15055 [cs]. URL: http://arxiv.org/abs/2006.15055 (visited on 05/11/2025). Pre-published.

[18] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. *Emerging Properties in Self-Supervised Vision Transformers*. May 24, 2021. arXiv: 2104.14294 [cs]. URL: http://arxiv.org/abs/2104.14294 (visited on 03/25/2025). Pre-published.

[19] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. *DINOv2: Learning Robust Visual Features without Supervision*. Feb. 2, 2024. arXiv: 2304.07193 [cs]. URL: http://arxiv.org/abs/2304.07193 (visited on 03/25/2025). Pre-published.

[20] O. Siméoni, G. Puy, H. V. Vo, S. Roburin, S. Gidaris, A. Bursuc, P. Pérez, R. Marlet, and J. Ponce. *Localizing Objects with Self-Supervised Transformers and No Labels*. Sept. 29, 2021. arXiv: 2109.14279 [cs]. URL: http://arxiv.org/abs/2109.14279 (visited on 03/25/2025). Pre-published.

[21] H. Zhang, F. Li, S. Liu, L. Zhang, H. Su, J. Zhu, L. M. Ni, and H.-Y. Shum. *DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection*. July 11, 2022. arXiv: 2203.03605 [cs]. URL: http://arxiv.org/abs/2203.03605 (visited on 03/25/2025). Pre-published.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. ukasz Kaiser, and I. Polosukhin. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

[23] T. Liu, D. Floros, N. Pitsianis, and X. Sun. "Digraph Clustering by the BlueRed Method". In: *2021 IEEE High Performance Extreme Computing Conference (HPEC)*. 2021 IEEE High Performance Extreme Computing Conference (HPEC). Sept. 2021, pp. 1–7.

[24] J. B. Tenenbaum, V. de Silva, and J. C. Langford. "A Global Geometric Framework for Nonlinear Dimensionality Reduction". In: *Science* 290.5500 (Dec. 22, 2000), pp. 2319–2323.

[25] N. Pitsianis, A.-S. Iliopoulos, D. Floros, and X. Sun. "Spaceland Embedding of Sparse Stochastic Graphs". In: *2019 IEEE High Performance Extreme Computing Conference (HPEC)*. 2019 IEEE High Performance Extreme Computing Conference (HPEC). Sept. 2019, pp. 1–8.

[26] L. Van der Maaten and G. Hinton. "Visualizing Data Using T-SNE." In: *Journal of machine learning research* 9.11 (2008), pp. 2579–2605.

[27] V. A. Traag, L. Waltman, and N. J. Van Eck. "From Louvain to Leiden: Guaranteeing Well-Connected Communities". In: *Sci. Rep.* 9.1 (Mar. 26, 2019), p. 5233.

[28] A. Plaza and J. Tilton. "Automated Selection of Results in Hierarchical Segmentations of Remotely Sensed Hyperspectral Images". In: *Proceedings. 2005 IEEE International Geoscience and Remote Sensing Symposium, 2005. IGARSS '05.* . 2005 IEEE International Geoscience and Remote Sensing Symposium, 2005. IGARSS '05. Vol. 7. July 2005, pp. 4946–4949.

[29] L. J. Hubert and P. Arabie. "Comparing Partitions". In: *Journal of Classification* 2.2–3 (1985), pp. 193–218.

[30] H. W. Kuhn. "The Hungarian Method for the Assignment Problem". In: *Nav. Res. Logist. Q.* 2.1–2 (Mar. 1955), pp. 83–97.

[31] R. Jonker and A. Volgenant. "A Shortest Augmenting Path Algorithm for Dense and Sparse Linear Assignment Problems". In: *Computing* 38.4 (Dec. 1, 1987), pp. 325–340.

[32] D. P. Bertsekas. "Auction Algorithms for Network Flow Problems: A Tutorial Introduction". In: *Comput Optim Applic* 1.1 (Oct. 1992), pp. 7–66.

[33] M. Fiedler. "Algebraic Connectivity of Graphs". In: *Czech. Math. J.* 23.2 (1973), pp. 298–305.

[34] D. Floros, N. Pitsianis, and X. Sun. "Algebraic Vertex Ordering of a Sparse Graph for Adjacency Access Locality and Graph Compression". In: *2024 IEEE High Performance Extreme Computing Conference (HPEC)*. 2024 IEEE High Performance Extreme Computing Conference (HPEC). Sept. 2024, pp. 1–7.

[35] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv:1412.6980 [cs.LG]* (Jan. 30, 2017). arXiv: 1412.6980 [cs.LG].

[36] German Collection of Microorganisms and Cell Cultures GmbH. *DSMZ – German Collection of Microorganisms and Cell Cultures*. 2025.

[37] *National Institutes of Health (NIH)*. National Institutes of Health (NIH). URL: https://www.nih.gov/ (visited on 02/04/2025).

[38] A. Çelikkanat, A. R. Masegosa, and T. D. Nielsen. "Revisiting K-mer Profile for Effective and Scalable Genome Representation Learning". In: ().

[39] *Kraken: Ultrafast Metagenomic Sequence Classification Using Exact Alignments | Genome Biology | Full Text*. URL: https://genomebiology.biomedcentral.com/articles/10.1186/gb-2014-15-3-r46 (visited on 02/04/2025).

[40] *How to Apply de Bruijn Graphs to Genome Assembly | Nature Biotechnology*. URL: https://www.nature.com/articles/nbt.2023 (visited on 02/04/2025).

[41] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. von Lilienfeld. "Quantum Chemistry Structures and Properties of 134 Kilo Molecules". In: *Sci Data* 1.1 (Aug. 5, 2014), p. 140022.

[42] D. Harrison and D. L. Rubinfeld. "Hedonic Housing Prices and the Demand for Clean Air". In: *Journal of Environmental Economics and Management* 5.1 (Mar. 1, 1978), pp. 81–102.

[43] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-Based Learning Applied to Document Recognition". In: *Proceedings of the IEEE* 86.11 (Nov. 1998), pp. 2278–2324.

[44] L. Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32.

[45] T. Chen and C. Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Aug. 13, 2016, pp. 785–794. arXiv: 1603.02754 [cs.LG].

[46] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. "CatBoost: Unbiased Boosting with Categorical Features". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018.

[47] S. O. Arik and T. Pfister. *TabNet: Attentive Interpretable Tabular Learning*. Dec. 9, 2020. arXiv: 1908.07442 [cs]. URL: http://arxiv.org/abs/1908.07442 (visited on 05/14/2025). Pre-published.

[48] X. Huang, A. Khetan, M. Cvitkovic, and Z. Karnin. *TabTransformer: Tabular Data Modeling Using Contextual Embeddings*. Dec. 11, 2020. arXiv: 2012.06678 [cs]. URL: http://arxiv.org/abs/2012.06678 (visited on 05/05/2025). Pre-published.

[49] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko. *Revisiting Deep Learning Models for Tabular Data*. Oct. 26, 2023. arXiv: 2106.11959 [cs]. URL: http://arxiv.org/abs/2106.11959 (visited on 05/14/2025). Pre-published.

[50] D. B. Sauer and D.-N. Wang. "Predicting the Optimal Growth Temperatures of Prokaryotes Using Only Genome Derived Features". In: *Bioinformatics* 35.18 (Sept. 15, 2019), pp. 3224–3231.

[51] X. Fang, L. Liu, J. Lei, D. He, S. Zhang, J. Zhou, F. Wang, H. Wu, and H. Wang. "Geometry-Enhanced Molecular Representation Learning for Property Prediction". In: *Nat Mach Intell* 4.2 (Feb. 2022), pp. 127–134.

[52] *Principal Component Analysis*. Springer Series in Statistics. New York: Springer-Verlag, 2002.

[53] L. McInnes, J. Healy, and J. Melville. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction". In: *arXiv:1802.03426 [stat.ML]* (Sept. 18, 2020). arXiv: 1802.03426 [stat.ML].

A An Intuitive Example of Configuration Mixing

To illustrate the necessity of fusing valid clusterings across resolution scales, we use two synthetic point-cloud datasets from scikit-learn: “Moons” and “Blobs.” The Blobs dataset is tuned so that no single clustering resolution recovers all three clusters. Fig. 9 visualizes each dataset in 3D, using the third axis to encode cluster assignments for the corresponding configuration: coarser configuration (1) and finer configuration (2). Configuration (1), by lifting some dots above the plane, cleanly separates the two Moon arcs but merges two (purple and green) of the Blobs clusters. Configuration (2), by itself, fails the Blobs with a different merge (blue and green). Only by fusing both configurations can all clusters be disentangled—the purple dots in (1) that falls down in (2), emerges correct as the green cluster. This toy example shows that multi-resolution clusterings alone are insufficient without a fusion mechanism. Our GraMixC use attention-based fusion to integrate these scales. While just one demonstration, it highlights the broader advantage of mixing configurations in complex settings.

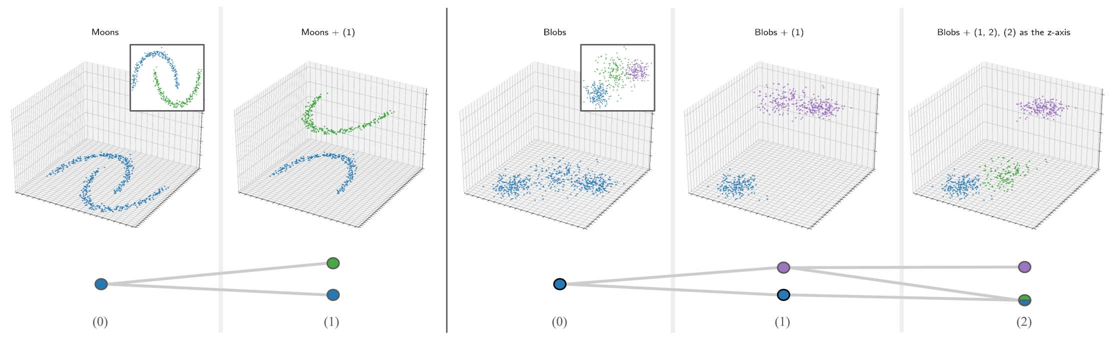


Figure 9: Illustration of multi-resolution clustering on synthetic datasets. GT is shown in the framed box in (0). Upper is the embedding of Moons (left) and Blobs (right) with corresponding configuration (i) as third dimension; lower is the lineage diagram of the configurations.

B DSNI dataset distribution

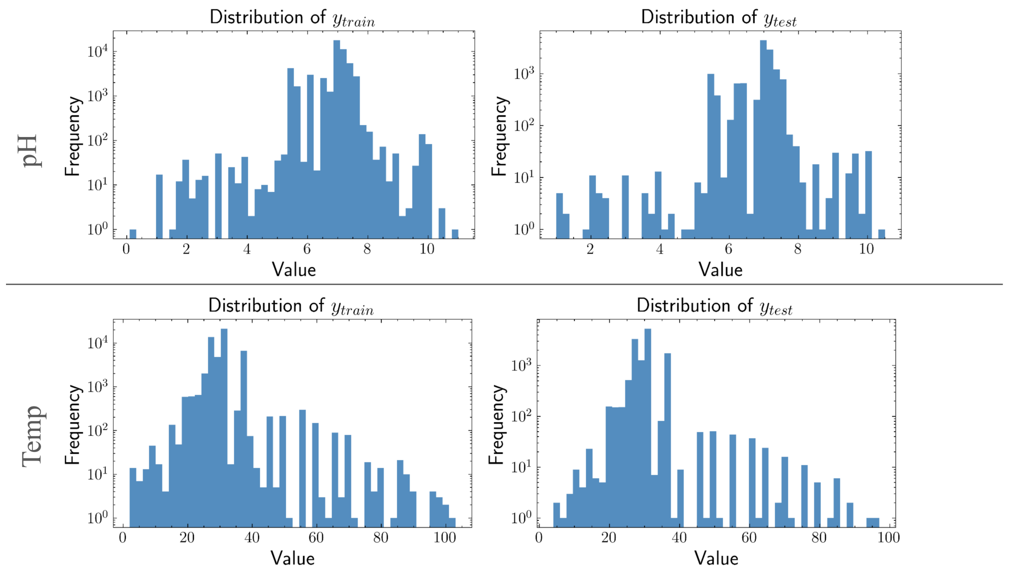


Figure 10: Illustration of target value distributions across train-test splits in DSNI dataset. The first row represents pH distributions and the second row represents temperature distributions. The first column represents the training set (y_{train}) and the second column represents the test set (y_{test}).

C Synthetic Clustering Benchmarks

In this section, we further discuss the limitations of conventional clustering methods raised in Section 3.1. We compare our modularity-based clustering strategy, which is used as the unsupervised layer in GraMixC, against widely-used clustering algorithms on synthetic 2D datasets.

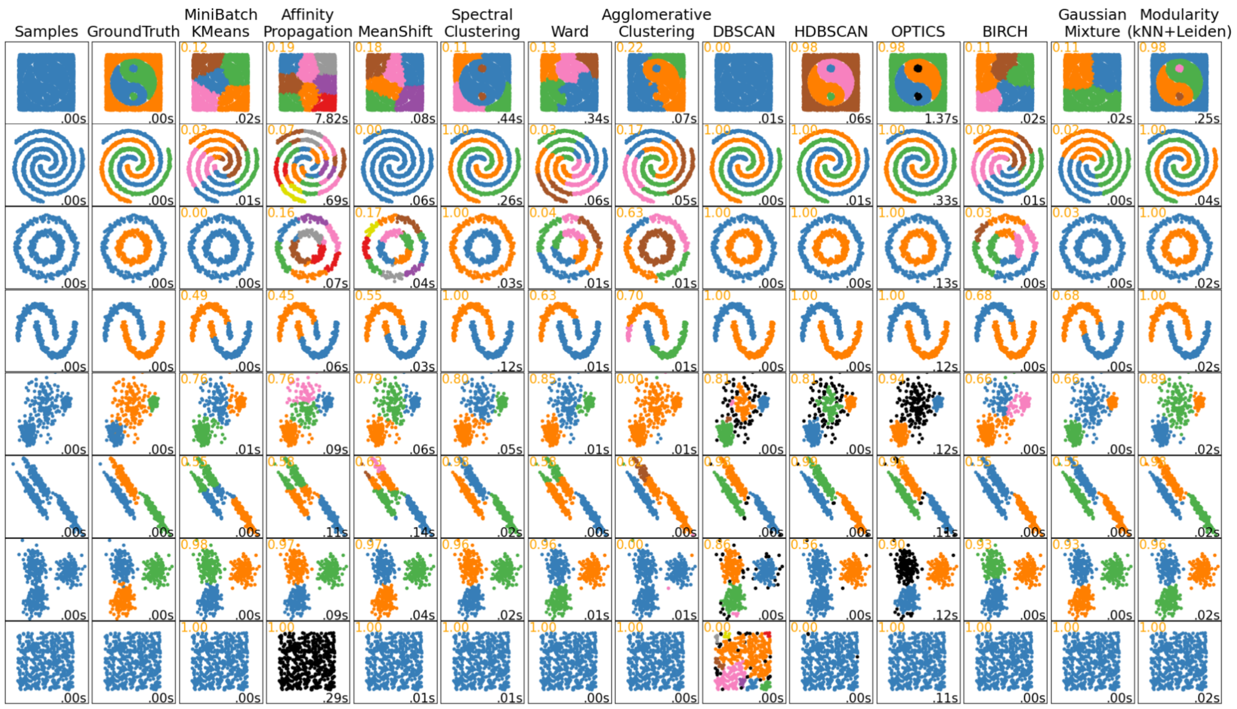


Figure 11: Illustration of clustering methods comparisons across multiple synthetic datasets. Rows correspond to different 2D point clouds—the first row is custom, others are from scikit-learn. Each method's result is labeled with ARI (top-left in yellow) and execution time (bottom-right in black). *Modularity: kNN+Leiden* (far right) accurately recovers ground-truth structures across different shapes and densities, with robustness to noise, anisotropy, and distribution variation.

Each row in Fig. 11 presents a distinct synthetic dataset distribution, ranging from custom-designed to standard scikit-learn datasets, including *Taiji*, spirals, circles, moons, varied blobs, anisotropy, blobs, and isotropic noise. Each column represents the result of one clustering method, annotated with Adjusted Rand Index (ARI) and execution time.

Unlike traditional clustering methods, the approach we adopted (last column: Modularity, implemented via kNN graph + Leiden community detection) consistently uncovers the underlying structure—even in challenging cases involving non-convex geometries, anisotropic spreads, or uneven density distributions. This comparison underscores the reliability and manifold sensitivity of our unsupervised segmentation approach, even before introducing multi-resolution fusion or downstream learning tasks.

D RMS Alignment Details

In Section 3.2 we introduced the Reverse Merge & Split (RMS) procedure for aligning multi-resolution configurations between train and test sets. Below we provide the full pseudo-code in Algorithm 1, using the same notation as the main text.

Implementation notes.

- We set $\theta = 0.1$ and compute ARI as in Hubert and Arabie [29].
- We use $0.1\,\%$ of the train samples as anchors to form $\mathcal{A}$.
- The greedy matching loops over each train configuration $\boldsymbol{\omega}_i$ to find its best-scoring test partner $\boldsymbol{\omega}_j$, applies the label mapping, and removes both from further consideration to ensure one-to-one alignment.

The details for SCORE and $\boldsymbol{L}_{\text{tw}}$ are covered in Algorithm 1 so we skip them here.

Algorithm 1 Reverse Merge & Split (RMS) Alignment

```
Require: $\boldsymbol{\Omega}_{\text{train}} \in \mathbb{N}^{N\times m_t}$, $\boldsymbol{\Omega}_{\text{test}} \in \mathbb{N}^{N\times m_s}$, anchor indices $\mathcal{A} \subset \{1,\dots,N\}$, $\theta$
Ensure: Aligned $\boldsymbol{\Omega}_{\text{test}}$
 1: $\mathbb{U} \leftarrow \{1,\dots,m_t\}$,    $\mathbb{V} \leftarrow \{1,\dots,m_s\}$
 2: for $i$ in $\mathbb{U}$ do                                          ▷ for each train configuration $\boldsymbol{\omega}_i$
 3:     best_score $\leftarrow -\infty$, best_j $\leftarrow$ null
 4:     $\boldsymbol{\omega}_i \leftarrow \boldsymbol{\Omega}_{\text{train}}[\mathcal{A}, i]$
 5:     for $j$ in $\mathbb{V}$ do                                      ▷ find best test configuration $\boldsymbol{\omega}_j$
 6:         $\boldsymbol{\omega}_j \leftarrow \boldsymbol{\Omega}_{\text{test}}[\mathcal{A}, j]$
 7:         $s \leftarrow \text{SCORE}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_j, \theta)$
 8:         if $s >$ best_score then
 9:             best_score $\leftarrow s$,  best_j $\leftarrow j$
10:         end if
11:     end for
12:     $M \leftarrow$ PAIR_MAPPING $(\boldsymbol{\Omega}_{\text{train}}[:, i],\ \boldsymbol{\Omega}_{\text{test}}[:, \text{best\_j}])$
13:     for $p = 1$ to $N$ do
14:         $\boldsymbol{\Omega}_{\text{test}}[p, \text{best\_j}] \leftarrow M\big(\boldsymbol{\Omega}_{\text{test}}[p, \text{best\_j}]\big)$
15:     end for
16:     Remove $i$ from $\mathbb{U}$,  remove best_j from $\mathbb{V}$
17: end for
18: return $\boldsymbol{\Omega}_{\text{test}}$
19: function PAIR_MAPPING($\boldsymbol{\omega}_i, \boldsymbol{\omega}_j$)
20:     $n_i \leftarrow \|\boldsymbol{\omega}_i\|_\infty$,    $n_j \leftarrow \|\boldsymbol{\omega}_j\|_\infty$
21:     for $p = 1$ to $N$ do                                           ▷ build confusion matrix $\boldsymbol{C} \in \mathbb{N}^{n_i\times n_j}$
22:         $\boldsymbol{C}[\boldsymbol{\omega}_i[p], \boldsymbol{\omega}_j[p]] \mathrel{+}= 1$
23:     end for
24:     Construct two-walk Laplacian $\boldsymbol{L}_{\text{tw}}$
25:     $\mathcal{F} \leftarrow$ Fiedler vector of $\boldsymbol{L}_{\text{tw}}$
26:     Split $\mathcal{F} \rightarrow (\mathcal{F}_i \in \mathbb{R}^{n_i},\ \mathcal{F}_j \in \mathbb{R}^{n_j})$
27:     $\pi_i \leftarrow \text{argsort}(\mathcal{F}_i),\ \pi_j \leftarrow \text{argsort}(\mathcal{F}_j)$
28:     return mapping $k \mapsto \pi_i\left[\pi_j^{-1}(k)\right]$ for $k = 1,\dots,\min(n_i, n_j)$
29: end function
```

E Additional Experimental Results

In Section 4 we introduced our experimental setup and high-level results. Here, we provide the full details and qualitative analyses that couldn't fit into the main body, including:

- Downstream task performance on three other benchmarks.
- Qualitative illustration of prediction versus true value on the three tabular baseline models.
- Embeddings from PCA and AE.

E.1 Additional evaluation of proposed module

Table 2 extends our evaluation to three additional benchmarks: Boston Housing (regression), MNIST and CIFAR-10 (classification). We compare classical ensembles (RF, XGBoost, CatBoost), a 3-layer MLP (3LP), and three neural tabular architectures (TabNet, TabTransformer, FT-Transformer) in three modes: baseline, static configuration concatenation (GC), and attention-based fusion (GMC).

Across almost all models and datasets, GC consistently improves performance over the raw baselines, and GMC provides further gains.

The sole exception is TabTransformer on Boston Housing, where GC yields only a marginal R^2 increase (0.811→0.813), but GMC degrades it (to 0.671), suggesting that attention-based fusion may disrupt already well-structured features in this case.

Table 2: Regression/classification performance on Boston Housing (BHouse), MNIST, and CIFAR10.

Dataset	BHouse		MNIST		CIFAR10	
Metric	MSE ↓	R^2	CE ↓	Acc	CE ↓	Acc
RF	0.022	0.884	0.247	0.969	1.681	0.463
XGBoost	0.022	0.881	0.066	0.980	1.296	0.539
CatBoost	0.016	0.913	0.096	0.975	1.230	0.567
3LP	0.023	0.879	0.141	0.970	1.428	0.524
3LP+GC	0.022	0.882	0.046	0.992	0.480	0.844
3LP+GMC	**0.017**	**0.909**	**0.028**	**0.993**	**0.220**	**0.949**
TabN	0.033	0.822	0.130	0.964	1.499	0.463
TabN+GC	0.021	0.888	0.225	0.941	0.377	0.876
TabN+GMC	**0.012**	**0.936**	**0.017**	**0.995**	**0.077**	**0.978**
TabT	0.035	0.811	0.192	0.980	1.028	0.706
TabT+GC	**0.035**	**0.813**	0.040	0.993	1.049	0.704
TabT+GMC	0.061	0.671	**0.018**	**0.994**	**0.458**	**0.911**
FTT	0.032	0.826	0.098	0.980	0.415	0.874
FTT+GC	0.030	0.838	0.029	0.993	0.437	0.870
FTT+GMC	**0.026**	**0.860**	**0.018**	**0.995**	**0.157**	**0.955**

On MNIST, GC lifts accuracy above 99%, and GMC pushes it to 99.3–99.5%. On CIFAR-10, GC delivers dramatic gains (e.g. TabTransformer from 46.3% to 87.6%), and GMC further improves all models, with FT-Transformer+GMC reaching 95.5% accuracy. These results underscore that configuration integration via GraMixC is broadly effective, with only one minor counterexample.

E.2 Additional qualitative evaluation of configurations

In Section 4.3 we provided the embedding of MNIST digits using UMAP and SG-t-SNE (Fig. 8a). Here we provide the missing illustration of embedding with PCA and autoencoder (AE) in Fig. 12. As expected, they do not provide representation with clusters as separated as the former two methods.

With the final figure (Fig. 13) we visualize predicted vs. actual values from the tabular baselines on DSNI, filling in what is missing from Fig. 6.

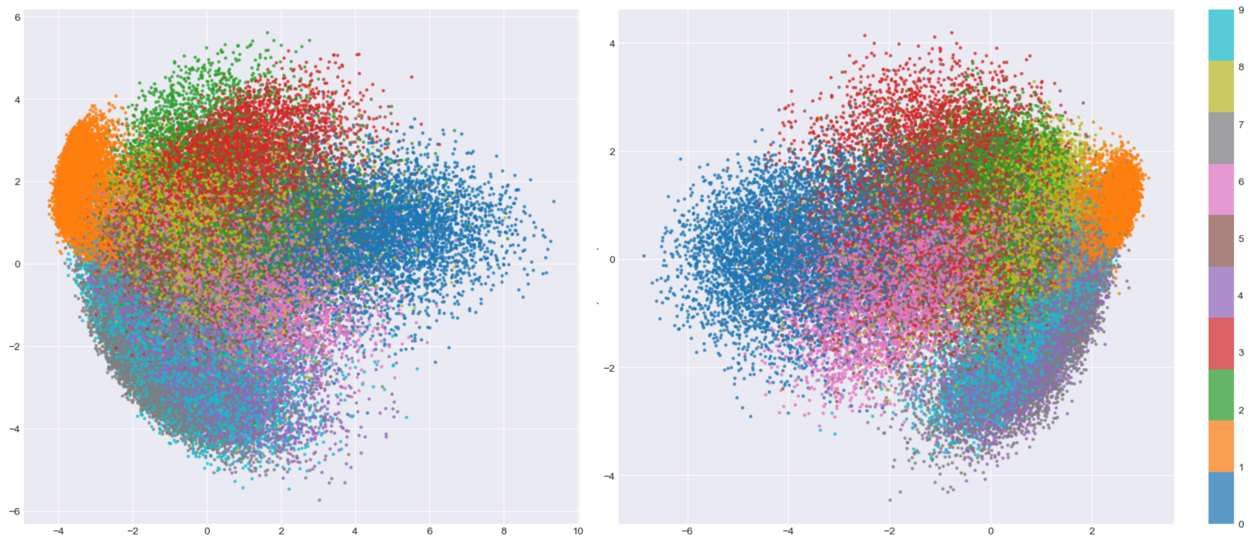


Figure 12: Illustration of 2D embeddings learned by PCA (left) and AE (right) on MNIST.

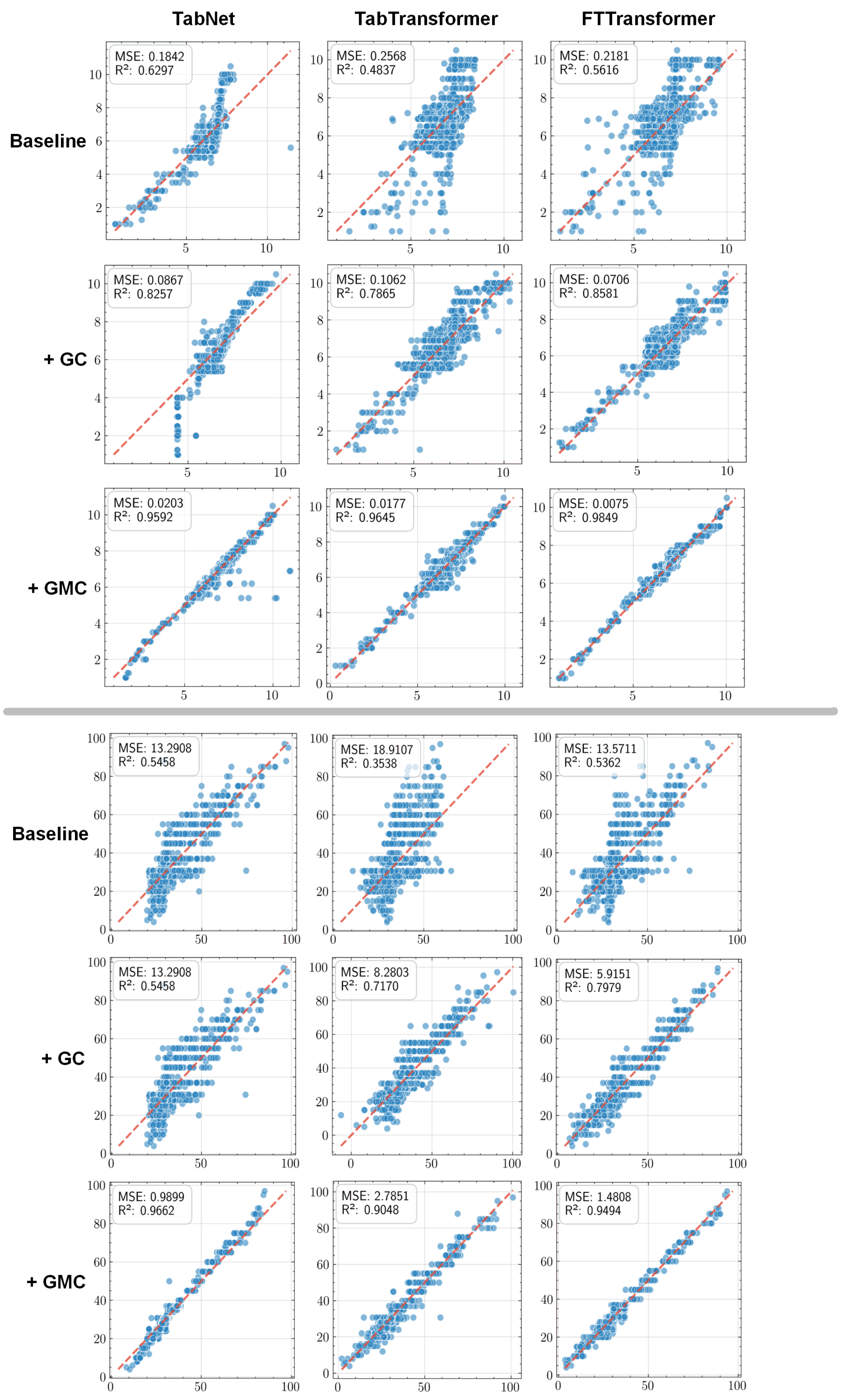


Figure 13: Illustration of the regression performance improvement example in TabNet, TabTransformer and FT-Transformer by adding GC or GMC. Each plots predicted vs. actual value.