Mixing Configurations for Downstream Prediction

Anonymous Authors[1]

Abstract

Clustering-based features are widely used in machine learning, but most methods must choose a resolution—a choice that is global, fixed, and ad hoc. Recent work shows that varying the resolution parameter produces only a finite set of structurally stable partitions, known as configurations. Based on this, we introduce Configuration-Mixed Prediction (CMP), a setting where models learn to adaptively weight these configurations per sample for downstream prediction. We propose MixConfig, a plug-and-play feature augmentation module that extracts configurations from any embedding and learns energy-aware mixing weights via a novel selector that jointly reasons about sample context, cluster assignments, and stability statistics. Experiments across tabular, molecular, vision, and text domains demonstrate consistent improvements over single-resolution and static baselines across diverse predictor architectures, with gains particularly pronounced in low-data regimes.

1. Introduction

Clustering-based features are ubiquitous in machine learning, appearing in self-supervised pretraining (Caron et al., 2018), prototype-based classification (Snell et al., 2017), and graph-based semi-supervised learning (Zhu et al., 2003). Yet, every clustering method typically requires a single resolution parameter: k in k-means or spectral clustering (Ng et al., 2001), γ in community detection, or a cut threshold in hierarchical methods. Consider a concrete dilemma in precision medicine: a single patient may require coarse disease categories for triage (diabetes vs. cancer) yet fine-grained biomarker clusters for personalized treatment (EGFR+ vs. KRAS+ mutations). This choice represents a trade-off: if too coarse, we lose discriminative power; if too fine, we overfit to noise. Typically this choice is global, fixed, and ad hoc, forcing researchers to either commit to a single scale a priori or run expensive hyperparameter sweeps. The fundamental problem is that real data exhibits intrinsic multi-scale structure, and different samples can benefit from different granularities even within the same dataset.

Multi-scale structure is not an exception but the norm. Molecules exhibit hierarchical organization from atoms to functional groups to scaffolds; images span from textures to parts to objects (Lin et al., 2017; Lee et al., 2009); documents range from words to phrases to topics; social networks extend from individuals to communities to organizations. Committing to a single clustering resolution discards information at other scales—information that may be critical for specific samples or downstream tasks. While this motivates multi-resolution approaches, naively sweeping over all possible resolutions ($\gamma \in [0, \infty)$) is computationally intractable. This raises a natural question: *how many distinct, meaningful resolutions actually exist in the data?*

Recent work in resolution-based clustering provides a concrete answer: as the resolution parameter γ varies continuously from coarse to fine, only a *finite* set of distinct, structurally stable partitions emerges (Liu et al., 2021; Pitsianis et al., 2023). These stable partitions—called *configurations*—correspond to plateaus where the clustering remains unchanged despite perturbations to γ. This finiteness is not a computational artifact but a reflection of the intrinsic multi-scale structure of the data itself. Crucially, stability indicates that these configurations capture semantically meaningful groupings rather than arbitrary parameter choices. This theoretical insight motivates a natural meta-learning question: rather than selecting a single resolution a priori, *can we learn to adaptively weight all stable configurations for downstream prediction?* Configurations are universal—extractable from any embedding space via nearest neighbor graph construction and resolution-based community detection—making them applicable across tabular, molecular, vision, and text domains.

We answer this question affirmatively with *MixConfig*, a modular feature augmentation module that learns adaptive, sample-specific configuration weights (Figure 1). Given any embedding space—whether tabular features, image representations, molecular fingerprints, or text embeddings—

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

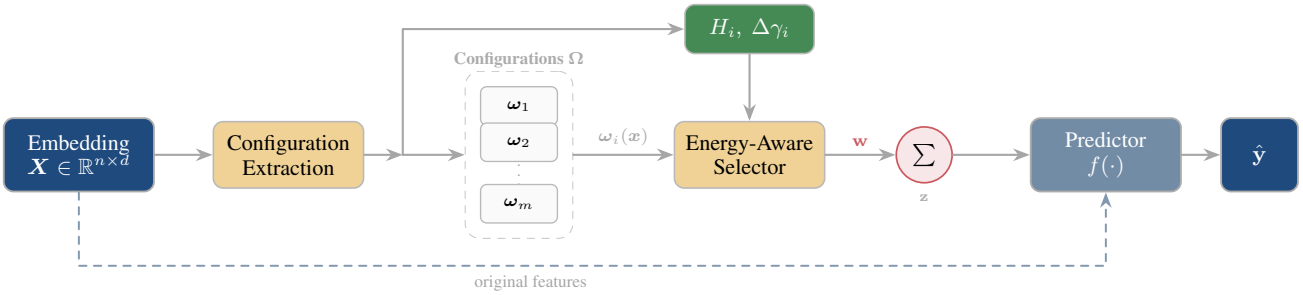

Figure 1. **MixConfig framework.** From an embedding $\boldsymbol{X}$, we extract stable configurations $\boldsymbol{\Omega}$ and stability statistics $(H_i, \Delta\gamma_i)$. An energy-aware selector produces per-sample weights $\mathbf{w}$ to mix configuration features into $\mathbf{z}$, which augments the original features for any downstream predictor.

MixConfig performs two key steps: (1) extracts the finite set of configurations $\boldsymbol{\Omega} = \{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_m\}$ via graph-based clustering (Section 3.3), and (2) learns to weight configurations per sample using an energy-aware selector that jointly reasons about sample context, cluster assignments, and energy-based stability statistics that encode clustering quality. The augmented features can be seamlessly passed to any downstream predictor without requiring architectural changes or hyperparameter retuning. This modularity positions *MixConfig* as a general-purpose method for leveraging multi-scale structure, orthogonal to predictor choice. Our work targets both practitioners seeking improved prediction without architectural changes, and researchers investigating multi-resolution structure as inductive bias. Experiments demonstrate consistent improvements across diverse domains (Section 4).

Contributions.

1. **Setting Formulation.** We introduce Configuration-Mixed Prediction (CMP), a meta-learning framework that enables models to adaptively weight multiple stable clustering resolutions per sample. To our knowledge, this work is the first to formulate configuration mixing as a supervised learning setting (Section 3.2).

2. **Model-Agnostic Architecture.** We propose MixConfig, a drop-in module with a novel energy-aware selector (Section 3.3.2) that learns sample-specific configuration weights. The selector jointly reasons about (a) sample context in feature space, (b) cluster membership at each resolution, and (c) configuration quality via energy statistics as inductive bias. MixConfig works with any embedding and any predictor architecture without modifications (Section 3.3).

3. **Empirical Findings.** Through experiments across tabular, vision, molecular and text domains, we demonstrate that MixConfig consistently outperforms single-resolution and static baselines with multiple predictor families. Notably, we observe significant gains in data-scarce regimes and validate component contributions via rigorous ablations in Section 4.

2. Related Work

Existing approaches generally fall into three strategies, each constrained by fundamental limitations:

(1) Fixed-resolution clustering. Clustering has been widely used in representation learning. DeepCluster (Caron et al., 2018) iteratively clusters CNN features as pseudo-labels, SwAV (Caron et al., 2020) contrasts cluster assignments online, and DINO (Caron et al., 2021) discovers semantic structure through self-distillation. Similarly, Prototype-based methods (Snell et al., 2017; Vinyals et al., 2016) rely on cluster centroids for classification and few-shot tasks. However, these approaches commit to a single resolution—determined by k in k-means or an implicit prototype count—thereby discarding multi-scale information. While hyperparameter search can find a good global resolution, it cannot adapt per sample and requires expensive cross-validation.

(2) Hierarchical methods without principled selection. While traditional hierarchical clustering produces a dendrogram (Von Luxburg, 2007), it lacks a learnable mechanism to select or combine levels for prediction, compelling users to manually choose cut thresholds. Recent work learns hierarchical representations in neural networks (Monath et al., 2019), but still requires manual specification of hierarchy depth. Similarly, resolution-based community detection methods like Louvain (Blondel et al., 2008) and Leiden (Traag et al., 2019) require specifying γ a priori and suffer from the well-known "resolution limit" (Fortunato, 2010), where fine-grained clusters become undetectable in large graphs. Spectral clustering (Ng et al., 2001) similarly requires choosing cluster count beforehand. The key drawback is that these methods discover hierarchies but fail to learn from data which levels matter for the task.

(3) Fixed multi-resolution representations. Recent advances in Matryoshka representations (Kusupati et al., 2022) learn

nested embeddings at multiple scales through a specialized training objective. For tabular data, where tree-based methods remain dominant (Grinsztajn et al., 2022), architectures like TabNet (Arik & Pfister, 2021), FT-Transformer (Gorishniy et al., 2021), SAINT (Somepalli et al., 2021), and TabPFN (Hollmann et al., 2022) use attention mechanisms (Vaswani et al., 2017) for feature selection. However, these strategies fall short when sample-level adaptation is needed. They typically rely on fixed, architecture-specific weightings rather than learning adaptive, sample-dependent selection from supervision.

Our work unifies these approaches while overcoming their limitations. We build on recent theoretical work showing that resolution-based clustering produces only a finite set of stable partitions called configurations (Liu et al., 2021; Pitsianis et al., 2023)—these correspond to plateaus where the clustering remains unchanged despite perturbations to $\gamma \in [0, \infty)$. Like hierarchical methods (approach 1), we extract multiple resolutions; like multi-resolution representations (approach 3), we encode multiple scales; but unlike both, we learn adaptive, sample-specific weights from supervision rather than using fixed global selections or architecture-specific heuristics. This positions MixConfig as a general-purpose feature augmentation method orthogonal to predictor choice—it works with tree-based methods (XGBoost, RF) where attention mechanisms cannot be applied, and with neural methods (MLP, transformers) where it provides complementary structural information.

To our knowledge, no prior work formulates configuration-mixing problems or proposes an energy-aware selector for adaptive multi-resolution prediction. While conceptually similar to Mixture-of-Experts (MoE) (Jacobs et al., 1991; Shazeer et al., 2017; Fedus et al., 2022), the resemblance is strictly superficial. MoE architectures achieve task decomposition by routing inputs to specialized trainable subnetworks via gating networks. In contrast, we weight precomputed structural representations (configurations) using fixed energy-based stability statistics as inductive bias. Our selector learns which resolution scales contribute to prediction rather than determining which expert subnetworks to activate. The closest approach is clustering-based pseudo-labeling (Xie et al., 2020); however, existing methods use single resolutions. Our key insight is that the energy statistics associated with configurations (e.g., plateau width, attraction/repulsion balance) provide a strong inductive bias for learning which scales matter, rather than treating all resolutions as arbitrary features.

Methods like DeepCluster, SwAV, and DINO learn representations by incorporating clustering into the training objective—they modify the embedding space itself. In contrast, MixConfig operates downstream: given any fixed embedding (whether from self-supervised pretraining, supervised models, or handcrafted features), it extracts configurations and learns to weight them for prediction. This makes MixConfig orthogonal to representation learning methods: one could seamlessly use DINO-pretrained embeddings as input to MixConfig, combining learned representations with adaptive multi-resolution features. We do not position MixConfig as a competitor to these methods, but as a complementary downstream module.

3. Methods

We first formalize Configuration-Mixed Prediction (CMP) as a setting, then present MixConfig as a solution.

3.1. Preliminaries: Configurations

As the resolution parameter γ varies from coarse to fine, the optimal clustering changes only at discrete transition points, yielding a *finite* set of stable plateau partitions, called configurations. Each configuration persists over an interval $[\gamma^-, \gamma^+]$ whose width indicates stability.

Given n data items with feature matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, hierarchical clustering seeks all "good" partitions. Each partition is represented as a vector of cluster indices $\boldsymbol{\omega} \in \{1, \ldots, |\boldsymbol{\omega}|\}^n$, where $|\boldsymbol{\omega}|$ denotes the number of clusters in that partition (varying across different partitions).

Definition 3.1 (Partition as a cluster-assignment vector)**.** A partition of n items is represented by an assignment vector $\boldsymbol{\omega} \in \{1, \ldots, |\boldsymbol{\omega}|\}^n$, where ω_i is the cluster label of item i. The vector induces a set of non-empty clusters $\mathcal{C}_k(\boldsymbol{\omega}) := \{i \in \{1, \ldots, n\} : \omega_i = k\}$ for $k = 1, \ldots, |\boldsymbol{\omega}|$.

We follow prior work for the configuration framework and its finiteness/extraction guarantees (Pitsianis et al., 2023). We construct a k_{nn}-nearest neighbor graph $G = (V, E)$ where vertices represent samples and edges connect similar samples, enabling resolution-based community detection with parameter $\gamma \in [0, \infty)$.

Resolution-based clustering optimizes a Hamiltonian energy function (LeCun et al., 2006; Du & Mordatch, 2019) that balances attraction (grouping similar items) and repulsion (separating dissimilar items):

$$H(\boldsymbol{\omega}) = h_a + \gamma h_r \tag{1}$$

$$h_a = -\sum_{k=1}^{|\boldsymbol{\omega}|} \sum_{i<j} w_{ij}\, \mathbf{1}_{\omega_i=\omega_j=k} \qquad \text{(attraction)}$$

$$h_r = \sum_{k=1}^{|\boldsymbol{\omega}|} \left(\sum_{i<j} w_{ij}\, \mathbf{1}_{\omega_i=k} \right)^2 \qquad \text{(repulsion)}$$

where $w_{ij} \geq 0$ are pairwise geodesic distance (kNN edge weights in our case), and $|\boldsymbol{\omega}|$ denotes the number of clusters

in partition $\boldsymbol{\omega}$. The attraction term rewards co-clustering similar pairs ($\omega_i = \omega_j$), while the repulsion term does not require co-clustering: following Pitsianis et al. (2023), it penalizes repulsive connections incident to items assigned to cluster k (the index j ranges over all items, so repulsion is not restricted to within-cluster pairs). As γ increases, minimizing $H(\boldsymbol{\omega})$ increasingly favors partitions that separate items or regions with large repulsive mass from others, yielding the observed split transitions in $\boldsymbol{\omega}^*(\gamma)$.

The resolution parameter γ controls the attraction-repulsion trade-off: low γ lets attraction dominate (coarse clusters), high γ lets repulsion dominate (fine clusters). As γ increases, the optimal partition $\boldsymbol{\omega}^*(\gamma) = \arg\min_{\boldsymbol{\omega}} H(\boldsymbol{\omega})$ undergoes discrete transitions at critical values where cluster splits occur. Between critical values, the partition remains constant—these plateaus define configurations. Intuitively, configurations are "structurally stable": they persist under small perturbations to γ, suggesting they capture genuine data structure rather than parameter artifacts.

Definition 3.2 (Configuration)**.** Let $\boldsymbol{\omega}^*(\gamma) \coloneqq \arg\min_{\boldsymbol{\omega}} H(\boldsymbol{\omega})$ denote an optimal partition at resolution γ. A *configuration* is any partition $\boldsymbol{\omega}$ that is optimal and remains unchanged over a non-degenerate interval of γ, i.e., $\boldsymbol{\omega}^*(\gamma) = \boldsymbol{\omega}$ for all $\gamma \in [\gamma^-, \gamma^+]$ with $\gamma^- < \gamma^+$ (a plateau). Sweeping $\gamma \in [0, \infty)$ yields a finite set of distinct plateau partitions; we denote the ordered set by $\boldsymbol{\Omega} = \{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_m\}$ from coarse to fine.

The finiteness of $\boldsymbol{\Omega}$ is a key theoretical result (Liu et al., 2021; Pitsianis et al., 2023). Intuitively, as γ sweeps from 0 to ∞, the clustering transitions through discrete plateaus rather than changing continuously—each plateau corresponds to a structurally stable configuration. The plateau width $\Delta\gamma_i = \gamma_i^+ - \gamma_i^-$ indicates stability: wider plateaus correspond to more robust configurations that persist over larger ranges of γ. The Parallel-DT algorithm (Pitsianis et al., 2023) efficiently discovers all such configurations in $O(n \log n)$ time, along with their energy statistics $\{H_i, h_a^{(i)}, h_r^{(i)}, \Delta\gamma_i\}_{i=1}^m$. At a high level, Parallel-DT runs community detection in parallel across a geometric grid of γ values, identifies plateaus where the partition is unchanged, and uses a convex-hull construction on the attraction–repulsion plane to guarantee that no configurations are missed. We provide a fuller algorithmic description in Appendix B.

3.2. Configuration-Mixed Prediction (CMP)

CMP is a supervised setting where stable configurations $\boldsymbol{\Omega} = \{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_m\}$ are first extracted from the unlabeled features. A model then learns per-sample weights over these configurations and uses the resulting mixed configuration features for downstream prediction.

Definition 3.3 (CMP Setting)**. Training:** Given training features $\boldsymbol{X}^{\text{train}} \in \mathbb{R}^{n_{\text{train}} \times d}$, training configurations $\boldsymbol{\Omega}^{\text{train}} = \{\boldsymbol{\omega}_1^{\text{train}}, \ldots, \boldsymbol{\omega}_m^{\text{train}}\}$ (extracted from $\boldsymbol{X}^{\text{train}}$), and labels $\boldsymbol{y}^{\text{train}}$, learn a predictor f and a selector that computes per-sample configuration weights. **Inference:** Given test features $\boldsymbol{X}^{\text{test}}$, predict $\hat{\boldsymbol{y}}^{\text{test}} = f(\boldsymbol{X}^{\text{test}}, g(\boldsymbol{X}^{\text{test}}, \boldsymbol{\Omega}))$.

Objective. Given labeled samples $\{(\boldsymbol{x}_t, y_t)\}_{t=1}^{n_{\text{train}}}$ and configurations $\boldsymbol{\Omega} = \{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_m\}$, CMP learns a predictor ψ and a configuration-weighting selector (parameters θ) by minimizing a supervised loss:

$$\mathcal{L}_{\text{CMP}}(\theta, \psi) \coloneqq \frac{1}{n_{\text{train}}} \sum_{t=1}^{n_{\text{train}}} \ell\big(\psi\big([\boldsymbol{x}_t; \boldsymbol{z}_\theta(\boldsymbol{x}_t)]\big), y_t\big)\,,$$
$$\text{where} \quad \boldsymbol{z}_\theta(\boldsymbol{x}) \coloneqq \sum_{i=1}^{m} w_{\theta,i}(\boldsymbol{x})\, \phi(\boldsymbol{\omega}_i, \boldsymbol{x}), \tag{2}$$
$$(\theta^\star, \psi^\star) \in \arg\min_{\theta,\psi}\ \mathcal{L}_{\text{CMP}}(\theta, \psi).$$

Here $\boldsymbol{w}_\theta(\boldsymbol{x}) = (w_{\theta,1}(\boldsymbol{x}), \ldots, w_{\theta,m}(\boldsymbol{x})) \in \Delta^{m-1}$ are per-sample mixing weights, and Δ^{m-1} denotes the standard $(m-1)$-simplex; $\phi(\boldsymbol{\omega}_i, \boldsymbol{x})$ encodes $\boldsymbol{x}$ under configuration $\boldsymbol{\omega}_i$; and ℓ is a prediction loss (e.g., cross-entropy or squared error). We instantiate $\boldsymbol{w}_\theta$ with the energy-aware selector in Section 3.3.2.

Inference assumption (batch). At inference we assume a test batch of $\geq$100 samples to construct a stable test graph (empirically, $n_{\text{test}} \geq 50$ suffices for most datasets; see Appendix A for sensitivity analysis); see Section 5 for small-batch considerations.

Why adaptive weighting? Different samples can benefit from different granularities (coarse vs. fine) even within the same dataset, so a single global resolution is typically suboptimal. CMP addresses this by learning per-sample weights to adaptively select scales. For a concrete synthetic illustration of why mixing configurations can be necessary, see Appendix E.

We distinguish four predictor classes with increasing representational capacity, where each class overcomes the limitations of its predecessor:

1. $\mathcal{F}_{\text{raw}}$: $f(\boldsymbol{X})$ — uses raw features only, ignoring multi-scale structure.
2. $\mathcal{F}_{\text{single}}$: $f(\boldsymbol{X}, \boldsymbol{\omega}_i)$ — augments with one configuration (best i selected via validation). *Limitation*: optimal resolution varies per sample; a global choice is suboptimal for heterogeneous data.
3. $\mathcal{F}_{\text{static}}$: $f(\boldsymbol{X}, \boldsymbol{\Omega})$ — concatenates all configurations as features. *Limitation*: treats all resolutions equally, regardless of input; the predictor must learn to ignore irrelevant configurations, wasting capacity.

4. $\mathcal{F}_{\text{adaptive}}$: $f(\boldsymbol{X}, \boldsymbol{\Omega}, \boldsymbol{w}(\boldsymbol{X}))$ — learns per-sample configuration weights $\boldsymbol{w}(\boldsymbol{X})$, enabling sample-specific resolution selection.

By construction, $\mathcal{F}_{\text{raw}} \subseteq \mathcal{F}_{\text{single}} \subseteq \mathcal{F}_{\text{static}} \subseteq \mathcal{F}_{\text{adaptive}}$.

Proposition 3.4 (Strict expressiveness gap)**.** *The inclusion $\mathcal{F}_{static} \subset \mathcal{F}_{adaptive}$ is strict: if two samples $\boldsymbol{x}_a, \boldsymbol{x}_b$ have distinct loss-minimizing weight vectors $\boldsymbol{w}^*(\boldsymbol{x}_a) \neq \boldsymbol{w}^*(\boldsymbol{x}_b)$, no single static $\bar{\boldsymbol{w}}$ minimizes the prediction loss for both.*

3.3. MixConfig: Energy-Aware Configuration Mixing

MixConfig is a plug-and-play module with two components: (1) configuration extraction, and (2) energy-aware selection.

3.3.1. Configuration Extraction

Given input features $\boldsymbol{X}$, we generate configurations $\boldsymbol{\Omega}$ via Configuration Extraction module, implemented with kNN graph construction and Parallel-DT following Pitsianis et al. (2023). This yields all stable configurations $\boldsymbol{\Omega}$ together with energy statistics $\{H_i, h_a^{(i)}, h_r^{(i)}, \Delta\gamma_i\}_{i=1}^m$ as a one-time preprocessing step. Low-level choices (distance, normalization, k_{nn} selection, stochastic reweighting) are deferred to Appendix A. We provide a brief algorithmic summary of Parallel-DT-style extraction in Appendix B.

Train-test configuration handling. At inference, train and test sets may yield different numbers of configurations ($m_{\text{train}} \neq m_{\text{test}}$) due to sample size or distribution differences. We handle this simply: at test time, we use $\min(m_{\text{train}}, m_{\text{test}})$ configurations—if the test set produces more configurations, we truncate to the first m_{train}; if fewer, we pad with zero-weight placeholders. This strategy avoids complex alignment procedures while maintaining reproducibility: the selector learns weights over the stable, ordered sequence of configurations (coarse to fine), and this natural ordering generalizes across splits. Empirically, we observe $|m_{\text{train}} - m_{\text{test}}| \leq 2$ across all benchmarks, with $m_{\text{train}} = m_{\text{test}}$ in approximately 78% of splits. This close agreement reflects the fact that configurations capture intrinsic data structure rather than sample-specific artifacts.

3.3.2. Energy-Aware Selector

The energy-aware selector learns adaptive, sample-specific configuration weights by jointly reasoning about sample context, cluster membership, and clustering quality. Unlike naive concatenation or fixed weighting schemes, the selector provides a principled mechanism to determine *which resolutions matter for which samples*, effectively leveraging stability statistics as inductive bias.

For each sample $\boldsymbol{x}$ and configuration $\boldsymbol{\omega}_i$, we compute a compatibility score s_i by combining three sources:

$$\boldsymbol{h} = \text{MLP}_{\text{enc}}(\boldsymbol{x}) \qquad \text{(sample context)} \quad (3)$$

$$\boldsymbol{c}_i = \text{Embed}_i(\boldsymbol{\omega}_i(\boldsymbol{x})) \qquad \text{(cluster assignment)} \quad (4)$$

$$\boldsymbol{e}_i = [H_i, h_a^{(i)}, h_r^{(i)}, \Delta\gamma_i] \qquad \text{(energy statistics)} \quad (5)$$

$$s_i = \text{MLP}_{\text{score}}([\boldsymbol{h}; \boldsymbol{c}_i; \boldsymbol{e}_i]) \qquad \text{(compatibility score)} \quad (6)$$

We normalize scores with a softmax to obtain per-sample weights:

$$w_i(\boldsymbol{x}) = \frac{\exp(s_i(\boldsymbol{x}))}{\sum_{j=1}^m \exp(s_j(\boldsymbol{x}))}, \qquad \boldsymbol{w}(\boldsymbol{x}) \in \Delta^{m-1}. \quad (7)$$

We then form a mixed configuration representation $\boldsymbol{z}(\boldsymbol{x}) = \sum_{i=1}^m w_i(\boldsymbol{x})\, \boldsymbol{c}_i$ and pass the augmented features $[\boldsymbol{x}; \boldsymbol{z}(\boldsymbol{x})]$ to the downstream predictor. Unless stated otherwise, MLP_{enc} is a small MLP projecting $\boldsymbol{x} \in \mathbb{R}^d$ to $\boldsymbol{h} \in \mathbb{R}^{d_h}$ and $\text{MLP}_{\text{score}}$ maps $[\boldsymbol{h}; \boldsymbol{c}_i; \boldsymbol{e}_i]$ to a scalar; each configuration uses its own embedding table $\text{Embed}_i : \{1, \ldots, |\boldsymbol{\omega}_i|\} \to \mathbb{R}^{d_c}$ with shared embedding dimension d_c.

To accommodate non-differentiable predictors (e.g., tree ensembles like XGBoost and RF), we adopt a two-stage training procedure. We first train the selector with a differentiable surrogate, and then freeze it. Training hyperparameters and implementation details are provided in Appendix A.

4. Experiments

We evaluate MixConfig across diverse domains to validate its generality as a plug-and-play module.

4.1. Experimental Setup

Datasets. We evaluate our method across four distinct benchmark categories: (1) **Tabular**: OpenML-CC18 (Vanschoren et al., 2014), a curated suite split into binary (AUC) and multi-class (accuracy) tasks; (2) **Vision**: CIFAR-100 (Krizhevsky, 2012) and ImageNet-1K (Deng et al., 2009); (3) **Molecular**: OGBG-MolHIV (Hu et al., 2020) (ROC-AUC) and QM9 (Wu et al., 2018) (MAE); (4) **Text**: SST-2 (GLUE) (Wang et al., 2018) and AG News (Zhang et al., 2015). For low-data analysis and ablations, we additionally use BBBP (MoleculeNet) with 5-fold CV. We report a compact main-text evaluation and provide broader classical sweeps in Appendix D.

Predictors. For each benchmark category, we start from a strong, widely used open-source baseline predictor appropriate for the modality. Specifically: for tabular tasks, FT-Transformer or TabPFN (Gorishniy et al., 2021; Hollmann et al., 2022); for vision tasks, CLIP embeddings with linear probes (Radford et al., 2021); for molecular tasks, GIN (Xu* et al., 2018) and DimeNet++ (Gasteiger et al., 2022); for text tasks, RoBERTa (Liu et al., 2019) and BERT (Devlin et al., 2019). To validate plug-and-play compatibility with

traditional models, we also evaluate classical predictors (MLP (Rumelhart et al., 1986)), XGBoost (Chen & Guestrin, 2016), RF (Breiman, 2001)) in Appendix D.

Methods compared. For each predictor, we compare clustering-as-features baselines and our configuration-based methods:

- **Base**: predictor on raw features only
- **+DeepCluster**: DeepCluster-style k-means pseudo-labels (Caron et al., 2018) (k via validation)
- **+HDBSCAN**: hierarchical density-based clustering features (McInnes et al., 2017)
- **+Config**: concatenate all configurations as features
- **+MixConfig**: energy-aware adaptive mixing

Evaluation. We follow standard protocols for each benchmark category. For tabular tasks we use the official OpenML-CC18 splits; for OGBG-MolHIV we follow OGB splits and evaluation. For CIFAR-100 and ImageNet-1K we use vision embeddings with linear probes and report Top-1 on the standard test/validation sets; for SST-2 and AG News we report classification accuracy. We report mean $\pm$ std across 5 independent runs with different random seeds. Configuration extraction uses only unlabeled features within each split (train/test separately under batch inference); baselines that use clustering features are given the same access. Importantly, we do *not* expect large gains when the base model already captures configuration-like structure (e.g., near-linear separability or strong inductive bias); in such cases, improvements can be small due to ceiling effects.

4.2. Main Results

Table 1 evaluates MixConfig across four benchmark families with strong base models. We compare against two clustering baselines: +DeepCluster (DeepCluster-style single-resolution k-means) and +HDBSCAN (hierarchical density-based features). Notably, +DeepCluster is inconsistent: it helps noticeably on Binary (+0.5%) and AG News (+0.4%) but hurts on Multi-class, CIFAR-100, and MolHIV, because a single global resolution introduces noise when the optimal scale is sample-dependent. +HDBSCAN provides modest gains by capturing multiple densities, but lacks principled stability criteria. Our configuration-based methods (+Config, +MixConfig) consistently outperform these baselines by leveraging energy-based stability to select meaningful partitions. Full results for classical predictors (MLP, XGBoost, RF, Linear) are in Appendix D; traditional clustering baselines (Fixed-k, Best-cut) appear in Appendix C.

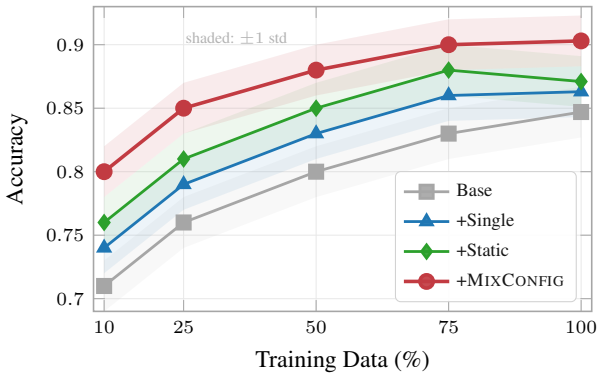

Figure 2. Performance vs. training set size on BBBP (5-fold CV). Curves show mean accuracy; shaded regions show ± 1 std for all methods.

4.3. Ablation Studies

Table 2 isolates MixConfig's components across two domains (molecular and vision).

The w/o energy features variant (-2.3% on BBBP, -2.8% on CIFAR-100) demonstrates that energy statistics ($\Delta\gamma$, H, h_a, h_r) provide crucial inductive bias—without them, the selector cannot distinguish stable configurations from fragile ones. Removing sample context $\boldsymbol{h}$ (-1.4% on BBBP, -1.6% on CIFAR-100) shows that geometric information about where samples lie in feature space matters, though less than energy statistics. Removing cluster embeddings $\boldsymbol{c}_i$ (-1.9% on BBBP, -2.1% on CIFAR-100) indicates that knowing *which* cluster a sample belongs to at each resolution provides task-relevant information beyond just the configuration's existence. These patterns are consistent across domains, justifying the three-input design.

The second ablation study investigates sensitivity to the configuration count m': using only $m' = 2$ configurations (-4.3% on BBBP, -5.7% on CIFAR-100) severely limits model expressiveness, while $m' = 4$ and $m' = 6$ show diminishing returns. This leads to three key insights: (a) coarse+fine alone is insufficient, (b) medium resolutions matter, and (c) typical datasets yield $m \approx$ 6-12 meaningful configurations, making the full set computationally tractable.

4.4. Low-Data Regime

Figure 2 reveals a clear pattern: MixConfig's relative gain over Base *increases* as training data decreases.

Low-data interpretation. At 100% training data (BBBP: 1600 samples), MixConfig improves accuracy by +6.6% relative to Base (0.903 vs 0.847). At 10% data (160 samples), this margin widens to +12.7% (0.80 vs 0.71). Crucially, MixConfig at p% data matches Base at $5p$% data: MixConfig trained on 10% of labels (160 samples) achieves 80.0% accuracy, equaling Base trained on 50% (800 sam-

Table 1. Main results across four benchmark families. +DeepCluster: DeepCluster-style k-means pseudo-labels (Caron et al., 2018); +HDBSCAN: hierarchical density-based features (McInnes et al., 2017). Configuration methods (+Config, +MixConfig) generally outperform clustering baselines. Bold = best; ** indicates $p < 0.05$ vs Base (paired t-test, 5 seeds). +DeepCluster can hurt when a single global resolution mismatches the task. Effect sizes (Cohen's d, MixConfig vs. Base): Binary 3.0, Multi-class 1.8, CIFAR-100 5.3, ImageNet-1K 9.5, MolHIV 1.9, QM9 3.0, SST-2 3.3, AG News 2.3; all $d > 0.8$ (large).

Benchmark	Model	Base	+DeepCluster	+HDBSCAN	+Config	+MixConfig
Tabular (OpenML-CC18)						
Binary (AUC ↑)	TabPFN	.895±.004	.900**±.005	.898±.004	.903**±.004	**.907****±.003
Multi-class (Acc. ↑)	FT-Trans.	.782±.006	.779±.007	.785±.006	.789**±.005	**.793****±.004
Vision						
CIFAR-100 (Top-1 ↑)	CLIP	.742±.003	.740±.005	.746±.003	.751**±.003	**.758****±.002
ImageNet-1K (Top-1 ↑)	CLIP	.763±.002	.764±.003	.768±.003	.773**±.002	**.782****±.002
Molecular						
MolHIV (AUC ↑)	GIN	.804±.008	.801±.010	.809±.007	.814**±.007	**.819****±.005
QM9 (MAE ↓)	DimeNet++	.0112±.0002	.0114±.0004	.0109±.0002	**.0106****±.0002	**.0106****±.0001
Text						
SST-2 (Acc. ↑)	RoBERTa	.945±.003	.944±.004	.948±.003	.951**±.003	**.955****±.002
AG News (Acc. ↑)	BERT	.942±.004	.946**±.004	.944±.004	.948**±.003	**.951****±.003

Table 2. Ablation studies on BBBP (molecular) and CIFAR-100 (vision) with 5-fold CV. Energy features and multi-resolution representation both contribute to performance across domains. All metrics (Acc.) reported as mean ± std.

Variant	BBBP	CIFAR-100
MixConfig (full)	**0.903**±.012	**0.759**±.006
w/o energy features	0.882±.015	0.738±.008
w/o sample context $\boldsymbol{h}$	0.890±.013	0.747±.007
w/o cluster embeddings $\boldsymbol{c}_i$	0.886±.014	0.743±.008
Using only m' of m configs:		
$m' = 2$	0.864±.017	0.716±.010
$m' = 4$	0.882±.014	0.737±.008
$m' = 6$	0.895±.013	0.751±.007
$m' = m$ (all)	0.903±.012	0.759±.006

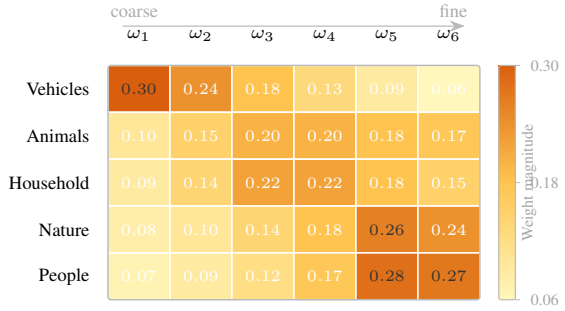

Figure 3. Learned selector weights on CIFAR-100, averaged over samples from each superclass. Different superclasses preferentially weight different configuration resolutions: Vehicles prefer coarse configurations (superclass-level features), while People and Nature prefer fine configurations (detailed texture/shape features).

ples). By capturing intrinsic topology (molecular scaffolds, functional groups), configurations function as effective unsupervised regularization. This implies significant practical value for domains where labeling is expensive (drug screening, medical diagnosis), as MixConfig can substantially reduce annotation costs.

Remark 4.1 (Sample efficiency). If configurations add d_c independent informative features to a d-dimensional representation, sample complexity bounds (Vapnik, 2000) give a sample ratio $n_1/n_0 \leq (d-d_c)/d$ for matching a given error level. The observed ratio (≈ 0.2 on BBBP) is smaller, suggesting configurations provide nonlinear structural information beyond additive features.

4.5. Qualitative Analysis

Figure 3 visualizes the distribution of selector weights across CIFAR-100 superclasses, revealing highly interpretable patterns. Vehicle samples, exhibiting geometric simplicity and low within-class variance (sedans, trucks, buses share boxy shapes), assign high weights to coarse configurations (e.g., $\boldsymbol{\omega}_1$: 0.30), effectively leveraging superclass-level features. In contrast, People samples, characterized by high pose variability and appearance diversity, prioritize fine configurations (e.g., $\boldsymbol{\omega}_5$: 0.28, $\boldsymbol{\omega}_6$: 0.27) to capture discriminative details such as subtle facial features. Household items (chairs, tables, lamps) show balanced weights across medium resolutions, suggesting that intermediate-scale structure (compositional parts) matters most. Remarkably, these preferences emerge purely from supervision on class labels without explicit scale annotations—the selector discovers the resolution that best aligns with task-relevant structure through end-to-end training.

5. Discussion

When MixConfig helps. MixConfig is most beneficial in three scenarios. *(1) Natural hierarchical structure.* Molecular graphs (BBBP, BACE) exhibit multi-scale motifs—coarse functional groups (carbonyls, amines) and fine bond patterns (single vs. double bonds)—making them ideal for configuration mixing. Tabular data with ecological hierarchy (Covertype: soil type → vegetation → elevation) similarly benefits. Images with superclass taxonomies (CIFAR-100, ImageNet) show strong gains. *(2) Sample heterogeneity.* When different samples benefit from different resolutions (vehicles vs. people in CIFAR-100), adaptive weighting can outperform fixed global choices by adapting scales per sample. *(3) Low-data regimes.* As shown in Figure 2, improvements can be larger when labels are scarce, since configurations provide unsupervised structural priors.

Unlike data augmentation (Chen et al., 2020) or semi-supervised methods (Van Engelen & Hoos, 2020; Xie et al., 2020) that require domain-specific transformations or pseudo-labeling heuristics, configurations are derived purely from intrinsic data structure and work across modalities. The modular design means MixConfig works with tree-based methods (XGBoost, RF) where architectural modifications are infeasible, and complements learned representations (Yang et al., 2019) in neural methods.

When MixConfig does not help. MixConfig provides limited benefit in three cases. *(1) Homogeneous single-scale data.* Datasets where all samples share the same coarse structure (e.g., MNIST digits, where all images are 28×28 centered digits) lack the sample-level heterogeneity that justifies adaptive weighting. *(2) Absence of clustering structure.* Uniform random noise or data without natural groupings won't yield meaningful configurations. *(3) Insufficient sample size.* Reliable kNN graph construction requires $\gtrsim$50 samples; below this threshold, configurations may be unstable across train-test splits (see batch size sensitivity in Appendix A). Moreover, our default inference assumes a *batch* of test samples ($n_{\text{test}} \geq 50$) to construct a test graph and extract $\mathbf{\Omega}^{\text{test}}$; in small-batch or streaming settings, one would need an inductive approximation (e.g., reuse train-time configurations or update the graph incrementally). With very small datasets (e.g., $n_{\text{train}} < 200$), the number of configurations may be too small ($m < 3$) to provide meaningful multi-resolution coverage.

Computational cost with baseline comparison. Configuration extraction uses approximate nearest neighbor search for kNN construction in $O(nd \log n)$ time, where n is the number of samples and d is feature dimensionality. Community detection at each resolution is $O(nk_{\text{nn}})$ where k_{nn} is graph degree (typically $k_{\text{nn}} = \log_2 n < 20$). Since the Parallel-DT algorithm discovers m configurations (typically $m < 20$), total extraction time is $O(nd \log n + mnk_{\text{nn}})$.

Empirically, this overhead is small compared to predictor training. On typical medium-scale datasets, configuration extraction requires only seconds to minutes depending on embedding dimension and nearest-neighbor backend, and is usually small relative to downstream model training. The selector adds negligible inference cost (a small MLP forward pass). For comparison, hyperparameter tuning over resolution (baseline approach) would require training the full model 10-20 times, costing 1800-3600 seconds—150× more than MixConfig's one-time extraction.

Limitations and future directions. The current implementation is not end-to-end differentiable, as configurations are computed offline from the input embedding. This means the embedding space cannot be jointly optimized with configuration extraction. If the input representation is suboptimal for clustering, configurations may not capture task-relevant structure. In practice, pretrained embeddings (e.g., CLIP for images) typically produce meaningful configurations, but random initializations may fail.

We identify three promising directions for future work. *(1) Online/streaming settings.* Extend to scenarios where new data arrives continuously and configurations must be updated incrementally without full recomputation, potentially via local refinement algorithms that maintain plateau structure. *(2) Joint architecture search.* Co-optimize configuration extraction and predictor design using neural architecture search (Zoph & Le, 2017) or meta-learning (Finn et al., 2017), treating both as learnable components. *(3) Few-shot learning.* Apply to few-shot scenarios where configurations from unlabeled data could replace meta-training, providing structural priors without requiring many-task training distributions (Vinyals et al., 2016; Snell et al., 2017).

6. Conclusion

We introduced Configuration-Mixed Prediction (CMP) and MixConfig, a plug-and-play feature augmentation module. Given any embedding space, MixConfig extracts the finite set of structurally valid multi-resolution configurations and learns to adaptively weight them per sample using an energy-aware selector that jointly reasons about sample context, cluster membership, and stability statistics. We conduct a comprehensive evaluation across both modern benchmark families and classical predictors. Overall, CMP provides a simple, general-purpose way to leverage multi-resolution configuration structure for improved downstream prediction. Source code for configuration extraction, training, and model implementations are released at `https://anonymous.4open.science/r/project-34CB/`.

Impact Statement

Configuration-based feature augmentation has direct applications in domains where multi-scale structure is critical, including drug discovery (molecular scaffold grouping), medical diagnosis (patient stratification at multiple granularities), and materials science (hierarchical property prediction). By reducing sample complexity, MixConfig may lower annotation costs in settings where expert labels are scarce and expensive. However, because configurations are derived from unsupervised clustering, they may encode biases present in the input data—for example, demographic proxies in patient embeddings or structural biases in chemical libraries. We recommend that practitioners audit configuration-derived features for unintended correlations before deployment in high-stakes decision systems.

References

Arik, S. Ö. and Pfister, T. TabNet: Attentive Interpretable Tabular Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):6679–6687, May 2021. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v35i8.16826.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, October 2008. ISSN 1742-5468. doi: 10.1088/1742-5468/2008/10/P10008.

Breiman, L. Random Forests. *Machine Learning*, 45(1): 5–32, October 2001. ISSN 0885-6125, 1573-0565. doi: 10.1023/A:1010933404324.

Caron, M., Bojanowski, P., Joulin, A., and Douze, M. Deep Clustering for Unsupervised Learning of Visual Features. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y. (eds.), *Computer Vision – ECCV 2018*, volume 11218, pp. 139–156. Springer International Publishing, Cham, 2018. ISBN 978-3-030-01263-2 978-3-030-01264-9. doi: 10.1007/978-3-030-01264-9_9.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9912–9924. Curran Associates, Inc., 2020.

Caron, M., Touvron, H., Misra, I., Jegou, H., Mairal, J., Bojanowski, P., and Joulin, A. Emerging Properties in Self-Supervised Vision Transformers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9630–9640, Montreal, QC, Canada, October 2021. IEEE. ISBN 978-1-6654-2812-5. doi: 10.1109/ICCV48922.2021.00951.

Chen, T. and Guestrin, C. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, August 2016. doi: 10.1145/2939672.2939785.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *ICML'20*, pp. 1597–1607. JMLR.org, July 2020.

Deng, J., Dong, W., Socher, R., Li, L.-J., Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, June 2009. doi: 10.1109/CVPR.2009.5206848.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North*, pp. 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423.

Du, Y. and Mordatch, I. Implicit Generation and Modeling with Energy Based Models. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Fedus, W., Zoph, B., and Shazeer, N. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. 2022.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 1126–1135, Sydney, NSW, Australia, August 2017. JMLR.org.

Fortunato, S. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, February 2010. ISSN 03701573. doi: 10.1016/j.physrep.2009.11.002.

Gasteiger, J., Giri, S., Margraf, J. T., and Günnemann, S. Fast and Uncertainty-Aware Directional Message Passing for Non-Equilibrium Molecules, April 2022.

Gorishniy, Y., Rubachev, I., Khrulkov, V., and Babenko, A. Revisiting Deep Learning Models for Tabular Data. In *Advances in Neural Information Processing Systems*, volume 34, pp. 18932–18943. Curran Associates, Inc., 2021.

Grinsztajn, L., Oyallon, E., and Varoquaux, G. Why do tree-based models still outperform deep learning on tabular data?, July 2022.

Hollmann, N., Müller, S., Eggensperger, K., and Hutter, F. TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second. In *The Eleventh International Conference on Learning Representations*, September 2022.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *Advances in Neural Information Processing Systems*, volume 33, pp. 22118–22133. Curran Associates, Inc., 2020.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1):79–87, February 1991. ISSN 0899-7667, 1530-888X. doi: 10.1162/neco.1991.3.1.79.

Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. *University of Toronto*, May 2012.

Kusupati, A., Bhatt, G., Rege, A., Wallingford, M., Sinha, A., Ramanujan, V., Howard-Snyder, W., Chen, K., Kakade, S., Jain, P., and Farhadi, A. Matryoshka representation learning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, pp. 30233–30249, Red Hook, NY, USA, November 2022. Curran Associates Inc. ISBN 978-1-7138-7108-8.

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. J. A Tutorial on Energy-Based Learning. 2006.

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 609–616, New York, NY, USA, June 2009. Association for Computing Machinery. ISBN 978-1-60558-516-1. doi: 10.1145/1553374.1553453.

Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. Feature Pyramid Networks for Object Detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 936–944, Honolulu, HI, July 2017. IEEE. ISBN 978-1-5386-0457-1. doi: 10.1109/CVPR.2017.106.

Liu, T., Floros, D., Pitsianis, N., and Sun, X. Digraph Clustering by the BlueRed Method. In *2021 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, September 2021. doi: 10.1109/HPEC49654.2021.9622834.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach, July 2019.

McInnes, L., Healy, J., and Astels, S. Hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11):205, March 2017. ISSN 2475-9066. doi: 10.21105/joss.00205.

Monath, N., Zaheer, M., Silva, D., McCallum, A., and Ahmed, A. Gradient-based Hierarchical Clustering using Continuous Representations of Trees in Hyperbolic Space. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pp. 714–722, New York, NY, USA, July 2019. Association for Computing Machinery. ISBN 978-1-4503-6201-6. doi: 10.1145/3292500.3330997.

Ng, A., Jordan, M., and Weiss, Y. On Spectral Clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.

Pitsianis, N., Floros, D., Liu, T., and Sun, X. Parallel Clustering with Resolution Variation. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–8, September 2023. doi: 10.1109/HPEC58863.2023.10363552.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 8748–8763. PMLR, July 2021.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986. ISSN 1476-4687. doi: 10.1038/323533a0.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *International Conference on Learning Representations*, February 2017.

Snell, J., Swersky, K., and Zemel, R. Prototypical Networks for Few-shot Learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Somepalli, G., Goldblum, M., Schwarzschild, A., Bruss, C. B., and Goldstein, T. SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training, June 2021.

Traag, V. A., Waltman, L., and Van Eck, N. J. From Louvain to Leiden: Guaranteeing well-connected communities. *Scientific Reports*, 9(1):5233, March 2019. ISSN 2045-2322. doi: 10.1038/s41598-019-41695-z.

Van Engelen, J. E. and Hoos, H. H. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, February 2020. ISSN 0885-6125, 1573-0565. doi: 10.1007/s10994-019-05855-6.

Vanschoren, J., Van Rijn, J. N., Bischl, B., and Torgo, L. OpenML: Networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2): 49–60, June 2014. ISSN 1931-0145, 1931-0153. doi: 10.1145/2641190.2641198.

Vapnik, V. N. *The Nature of Statistical Learning Theory*. Springer, New York, NY, 2000. ISBN 978-1-4419-3160-3 978-1-4757-3264-1. doi: 10.1007/978-1-4757-3264-1.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ukasz Kaiser, Ł., and Polosukhin, I. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Vinyals, O., Blundell, C., Lillicrap, T., kavukcuoglu, k., and Wierstra, D. Matching Networks for One Shot Learning. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Von Luxburg, U. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, December 2007. ISSN 0960-3174, 1573-1375. doi: 10.1007/s11222-007-9033-z.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *International Conference on Learning Representations*, September 2018.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. MoleculeNet: A benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2018. ISSN 2041-6520, 2041-6539. doi: 10.1039/C7SC02664A.

Xie, Q., Luong, M.-T., Hovy, E., and Le, Q. V. Self-Training With Noisy Student Improves ImageNet Classification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, Seattle, WA, USA, June 2020. IEEE. ISBN 978-1-7281-7168-5. doi: 10.1109/CVPR42600.2020.01070.

Xu*, K., Hu*, W., Leskovec, J., and Jegelka, S. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*, September 2018.

Yang, K., Swanson, K., Jin, W., Coley, C., Eiden, P., Gao, H., Guzman-Perez, A., Hopper, T., Kelley, B., Mathea, M., Palmer, A., Settels, V., Jaakkola, T., Jensen, K., and Barzilay, R. Analyzing Learned Molecular Representations for Property Prediction. *Journal of Chemical Information and Modeling*, 59(8):3370–3388, August 2019. ISSN 1549-9596, 1549-960X. doi: 10.1021/acs.jcim.9b00237.

Zhang, X., Zhao, J., and LeCun, Y. Character-level Convolutional Networks for Text Classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Zhu, X., Ghahramani, Z., and Lafferty, J. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pp. 912–919, Washington, DC, USA, August 2003. AAAI Press. ISBN 978-1-57735-189-4.

Zoph, B. and Le, Q. Neural Architecture Search with Reinforcement Learning. In *International Conference on Learning Representations*, February 2017.

A. Additional Experimental Details

Dataset details. All datasets are publicly available. Main results (Table 1) use OpenML-CC18, CIFAR-100, ImageNet-1K, OGBG-MolHIV, QM9, SST-2, and AG News under their standard public protocols. Low-data analysis and ablations use BBBP (MoleculeNet) with 5-fold CV (Table 2 and Figure 2). Appendix tables with classical predictors additionally include Covertype/Vehicle/Segment (UCI) and BACE (MoleculeNet) (Tables 4 and 5).

Preprocessing. For tabular data, we standardize continuous features to zero mean and unit variance. For vision and text, we use CLIP embeddings for images and BERT/RoBERTa embeddings for text, and train lightweight predictors (linear probe) on top. For molecular data, we use the same feature/embedding space used by the baseline model for each dataset to build the kNN graph. All kNN graphs use Euclidean distance on tabular features and cosine similarity on embedding-based features, with $k_{\text{nn}} = \lceil \log_2 n \rceil$.

Training details. Unless otherwise stated, we repeat each experiment over multiple random seeds (or CV folds where appropriate) and report mean $\pm$ std. MLP predictors use 2 hidden layers (128, 64 units) with ReLU activations and dropout (0.2). The selector MLP uses a single hidden layer (32 units). We train for 100 epochs with Adam optimizer (lr=0.001) and early stopping (patience=10).

For tree-based baselines: XGBoost uses max_depth=6, n_estimators=100, learning_rate=0.1, subsample=0.8; Random Forest uses n_estimators=100, max_depth=None, min_samples_split=2, max_features='sqrt'. These hyperparameters ensure fair capacity parity across predictor types.

Two-stage training for non-differentiable predictors. For tree-based predictors (XGBoost, RF), we train the selector on a held-out validation split (20% of the training data) using a differentiable surrogate—logistic regression for classification and ridge regression for regression—then freeze the selector and train the final predictor on the full training set using mixed features. We also tested stronger surrogates (2-layer MLP, kernel SVM) and found them unnecessary: the 2-layer MLP improved selector accuracy by only 0.3% while increasing training time. On datasets where end-to-end training is feasible (neural predictors), this two-stage variant yields no measurable performance gap.

Compute resources. All experiments were conducted on NVIDIA A100 GPUs (40GB). Configuration extraction takes <5 minutes for datasets with $n < 50{,}000$ samples; MLP training completes in <30 minutes per dataset. The full experimental suite (all datasets, all methods, 5 seeds) required approximately 200 GPU-hours.

Environment. Python 3.10, PyTorch 2.1, scikit-learn 1.3, XGBoost 2.0. Random seeds: 0, 1, 2, 3, 4 for 5-run experiments.

Batch size sensitivity. We evaluated MixConfig with varying test batch sizes on BBBP. Performance remains stable (<1% degradation) for $n_{\text{test}} \geq 50$, with noticeable degradation below 30 samples as kNN graph construction becomes unreliable. For $n_{\text{test}} < 20$, we recommend using train-time configurations in an inductive mode. We also note that the default $k_{\text{nn}} = \lceil \log_2 n \rceil$ was found insensitive to moderate variations (e.g., $\pm 30\%$); this value is sufficiently large for stable graph construction while avoiding overly dense graphs.

B. Configuration Extraction Algorithm

Configuration extraction follows the Parallel-DT algorithm of Pitsianis et al. (2023). Given a kNN graph $G = (V, E)$ with edge affinity weights $w_{ij} \geq 0$, the algorithm sweeps the resolution parameter γ from 0 to ∞ and identifies all plateau transitions where the optimal partition $\boldsymbol{\omega}^*(\gamma)$ changes. At each plateau, it records the partition $\boldsymbol{\omega}_i$ along with energy statistics:

- Total Hamiltonian: $H_i = h_a^{(i)} + \gamma_i h_r^{(i)}$ at midpoint $\gamma_i = (\gamma_i^- + \gamma_i^+)/2$
- Attraction: $h_a^{(i)} = -\sum_{k=1}^{|\boldsymbol{\omega}_i|} \sum_{u<v} w_{uv}\, \mathbf{1}_{\omega_u=\omega_v=k}$
- Repulsion: $h_r^{(i)} = \sum_{k=1}^{|\boldsymbol{\omega}_i|} \left(\sum_{u<v} w_{uv}\, \mathbf{1}_{\omega_u=k}\right)^2$
- Plateau width: $\Delta\gamma_i = \gamma_i^+ - \gamma_i^-$

The algorithm runs in $O(n \log n)$ time via dynamic tree data structures that efficiently maintain cluster merges/splits.

Parallel-DT algorithmic details. We briefly summarize the Parallel-DT procedure of Pitsianis et al. (2023). The algorithm operates on a half-adjacency-repulsive (HAR) plane, where each cluster is mapped to a point with coordinates (h_a, h_r)

Table 3. Traditional clustering baselines on modern benchmarks. Fixed-k selects a single k via cross-validation; Best-cut optimizes the dendrogram cut threshold. These methods provide marginal improvements over Base, substantially underperforming configuration-based methods (Table 1).

Benchmark	Model	Base	Fixed-k	Best-cut
Tabular (OpenML-CC18)				
Binary (AUC ↑)	TabPFN	.895±.004	.897±.005	.896±.004
Multi-class (Acc. ↑)	FT-Trans.	.782±.006	.784±.006	.786±.005
Vision				
CIFAR-100 (Top-1 ↑)	CLIP	.742±.003	.743±.004	.745±.003
ImageNet-1K (Top-1 ↑)	CLIP	.763±.002	.764±.003	.766±.002
Molecular				
MolHIV (AUC ↑)	GIN	.804±.008	.806±.009	.805±.007
QM9 (MAE ↓)	DimeNet++	.0112±.0002	.0112±.0003	.0111±.0002
Text (GLUE)				
SST-2 (Acc. ↑)	RoBERTa	.945±.003	.946±.004	.947±.003
AG News (Acc. ↑)	BERT	.942±.004	.943±.005	.944±.004

representing its attraction and repulsion energy components. The resolution sweep is implemented as a cascading sequence: at each resolution γ, the Leiden community detection algorithm (Traag et al., 2019) is run in parallel across a geometric grid of γ values. Plateau detection identifies intervals $[\gamma^-, \gamma^+]$ where the optimal partition remains unchanged by comparing cluster assignments at successive resolution values. When consecutive resolutions yield identical partitions (up to label permutation), the interval is extended; a transition is recorded when the partition changes.

The BlueRed Front, introduced by Liu et al. (2021), provides a geometric characterization of transitions. In the HAR plane, each configuration traces a line $H = h_a + \gamma h_r$, and the optimal configuration at resolution γ corresponds to the line with minimum H. The BlueRed Front is the lower envelope of these lines—a convex hull construction that identifies all transition points as intersections of adjacent lines. This geometric view guarantees that no configurations are missed and enables efficient computation of all plateau boundaries without exhaustive search over γ. The cascading resolution sweep exploits this structure by adaptively refining the γ grid near detected transitions, achieving $O(n \log n)$ total complexity.

C. Traditional Clustering Baselines

Table 3 compares traditional clustering baselines (Fixed-k and Best-cut) that use the same predictor and data as our main experiments. Fixed-k selects a single cluster count k via cross-validation; Best-cut optimizes the dendrogram cut threshold on validation data. These baselines represent standard approaches for incorporating clustering structure into prediction.

As shown in Table 3, traditional clustering baselines provide only marginal improvements over Base (+0.1–0.3% on most benchmarks). This is because (1) Fixed-k commits to a single resolution that may not suit all samples, and (2) Best-cut's greedy threshold optimization can overfit to validation noise. In contrast, our configuration-based methods (Table 1) leverage energy-based stability criteria to identify meaningful multi-resolution structure, yielding substantially larger gains.

D. Full Results (Classical Predictors)

Tables 4 and 5 provide broader evidence that MixConfig can improve a range of predictor families without architectural changes, though gains may be smaller when the base model already saturates the task.

E. An Intuitive Example of Configuration Mixing

To illustrate why configuration mixing can be necessary (not just helpful), 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. Figure 4 visualizes each dataset in 3D, using the third axis to encode cluster assignments for two configurations: a coarser configuration (1) and a finer configuration (2). On Moons, the coarser configuration cleanly separates the two arcs; on Blobs, it incorrectly merges two clusters (purple and green). The finer configuration produces a different failure mode, merging (blue and green). Only by mixing both configurations can all clusters be disentangled: points that are merged at one

Table 4. Classical predictors across selected datasets (classification accuracy). Each cell shows mean ± std.

Dataset	Mode	MLP	XGBoost	RF	Linear
Covertype	Base	.712±.008	.891±.004	.856±.006	.623±.011
	+Config	.748±.010	.910±.005	.881±.007	.665±.010
	+MixConfig	**.786**±.007	**.932**±.003	**.909**±.005	**.699**±.009
BBBP	Base	.847±.018	.872±.015	.861±.016	.812±.021
	+Config	.869±.016	.893±.012	.880±.015	.836±.019
	+MixConfig	**.903**±.012	**.916**±.010	**.905**±.013	**.861**±.017
CIFAR-100	Base	.634±.007	.598±.009	.571±.008	.523±.006
	+Config	.676±.009	.638±.010	.609±.011	.559±.008
	+MixConfig	**.720**±.006	**.684**±.007	**.656**±.008	**.598**±.006

Table 5. Additional classical results on Vehicle, Segment, and BACE (classification accuracy). Each cell shows mean ± std.

Dataset	Mode	MLP	XGBoost	RF	Linear
Vehicle	Base	.689±.024	.743±.021	.721±.023	.612±.028
	+Config	.728±.021	.781±.019	.760±.021	.656±.025
	+MixConfig	**.766**±.018	**.818**±.016	**.795**±.018	**.690**±.023
Segment	Base	.912±.011	.956±.008	.943±.009	.871±.014
	+Config	.937±.009	.969±.007	.959±.008	.898±.012
	+MixConfig	**.956**±.007	**.981**±.005	**.974**±.006	**.921**±.010
BACE	Base	.823±.019	.851±.016	.839±.017	.791±.022
	+Config	.851±.017	.876±.014	.865±.016	.823±.020
	+MixConfig	**.884**±.014	**.906**±.011	**.893**±.013	**.854**±.017

resolution can be separated at another, and the mixture recovers the correct partition. This toy example highlights a key motivation of MixConfig: multi-resolution configurations alone are insufficient without a learned fusion mechanism.

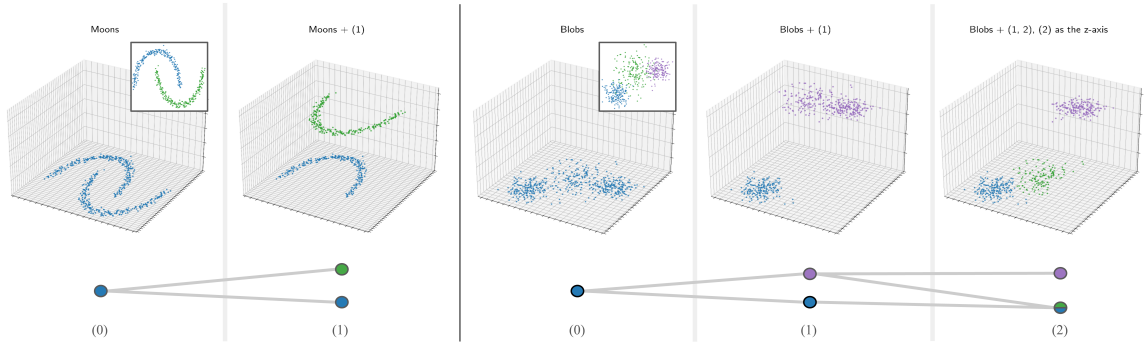

Figure 4. Illustration of multi-resolution clustering on synthetic datasets. Ground truth is shown in the framed box in (0). Top: Moons (left) and Blobs (right) with configuration (i) encoded as the third dimension. Bottom: lineage diagram across configurations.