Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Benchmarking Data Augmentation Strategies for Drug--Target Interaction Prediction under Class Imbalance: A Controlled Comparative Study

Vitor Magalhaes Silva

silva.vitor_magalhaes@univ.teluq.ca

TELUQ University

Khadidja Henni

TELUQ University

Neila Mezghani

TELUQ University

Research Article

Keywords: Drug--target interaction, data augmentation, matrix factorization, generative AI, diffusion models

Posted Date: November 5th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11026477/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Benchmarking Data Augmentation Strategies for Drug–Target Interaction Prediction under Class Imbalance: A Controlled Comparative Study

Vitor Magalhaes Silva[1*], Khadidja Henni[1,2*] and Neila Mezghani[1,2*]

[1*] Department of Information Technology, TELUQ University, Montreal, Quebec, Canada.
[2*] Imaging and Orthopedics Research Laboratory (LIO), CHUM Research Center, Montreal, Canada, TELUQ University, Montreal, Quebec, Canada.

*Corresponding author(s). E-mail(s):
silva.vitor_magalhaes@univ.teluq.ca; Khadidja.Henni@teluq.ca;
Neila.Mezghani@teluq.ca;

Abstract

Drug–target interaction (DTI) prediction is central to computational drug discovery, yet it remains constrained by extreme class imbalance, sparse interaction matrices, and the limited availability of experimentally validated interactions. Data augmentation is increasingly used to address these challenges, but the literature applies heterogeneous strategies and evaluation practices, limiting comparability and obscuring robustness.

This study presents a controlled benchmarking analysis of data augmentation strategies for DTI prediction, supported by related work from the literature. The literature analysis covers resampling techniques, feature-space transformations, matrix factorization–based approaches, generative models, Transformer-based architectures, and diffusion models, and is used to identify recurring methodological issues related to evaluation protocols, leakage control, and robustness under imbalance. Fourteen augmentation, resampling, filtering, and generative methods are then evaluated within a unified experimental pipeline across four protein families from the Yamanishi benchmark [1].

Results show that boundary-aware cleaning and latent-structure–aware augmentation can improve F1-score and PR-AUC in several settings, while naive oversampling tends to inflate recall at the expense of precision.

Keywords: Drug–target interaction, data augmentation, matrix factorization, generative AI, diffusion models

1 Introduction

Drug–target interaction (DTI) prediction is central to computational drug discovery, where experimental validation remains limited relative to the size of the possible drug–protein interaction space. The cost and duration of drug development further motivate computational approaches for prioritizing candidate interactions and reducing dependence on resource-intensive experimental screening [2].

A persistent obstacle in DTI prediction is *structural class imbalance*. In benchmark and real-world datasets, known positive interactions represent only a small fraction of possible drug–target pairs, producing sparse interaction matrices and skewed label distributions. Models trained under these conditions may favor the majority class, show limited sensitivity to true interactions, and produce misleading performance estimates [3, 4]. Evaluation practices also differ across studies, particularly in relation to data splitting, negative sampling, and leakage control, which limits comparability and weakens conclusions about methodological robustness.

Data augmentation is one strategy for addressing sparsity, imbalance, and limited feature diversity in DTI prediction. Classical resampling and boundary-cleaning methods are widely used to address imbalanced classification problems [3], while SMOTE-based approaches generate minority-class observations through interpolation or boundary-aware sampling [5, 6]. Matrix factorization and representation learning methods provide another way to address sparse interaction data by reconstructing latent drug and target representations from interaction matrices and similarity information [7–9]. Traditional deep learning models, including convolutional and recurrent neural networks, further support DTI prediction by combining chemical structures, protein sequences, and learned feature representations [10, 11].

Recent augmentation approaches extend these methods through neural generative models, Transformer-based architectures, and hybrid pipelines. Variational autoencoders and generative adversarial networks have been used to synthesize samples or latent interaction profiles for imbalanced DTI datasets [12–14]. Transformer-based and large language model approaches, including BERT-like architectures, use masked language modeling, token-level perturbation, attention mechanisms, and pretraining-based regularization to improve molecular and protein sequence representations under data scarcity [15–20]. Hybrid and cross-domain approaches combine deep learning, generative modeling, matrix factorization, and Transformer-based representations to improve generalization in sparse and imbalanced settings [10, 21]. Diffusion models have also been proposed as a generative alternative, particularly in latent spaces where sampling stability and diversity can be controlled [22, 23].

Despite this methodological expansion, several limitations remain. Augmentation methods are often evaluated in isolation and under non-comparable experimental pipelines. Synthetic data generation may also support the construction of anonymized and statistically representative datasets, which is relevant for data sharing and privacy

constraints in biomedical research [24]. However, such benefits require careful validation to ensure that artificial samples preserve meaningful statistical and biological structure [24, 25].

This study presents a controlled benchmarking analysis of data augmentation strategies for DTI prediction under class imbalance, supported by related work from the literature. Fourteen augmentation, resampling, filtering, and generative methods are evaluated within a unified experimental pipeline across four protein families from the Yamanishi benchmark [1]. The experiments use fixed data partitions, predefined class-distribution configurations, and a common downstream classifier to support controlled comparison across methods.

2 Related Work

2.1 DTI Prediction Background

DTI prediction is commonly formulated as the inference of unknown links between drugs and targets from sparse interaction matrices. This setting creates two related problems: many drug–target pairs are unobserved, and confirmed interactions represent a small proportion of possible pairs. As a result, models must distinguish between true non-interactions and missing information, while also learning from an imbalanced label distribution.

Several methods address this problem by using similarity information and latent representations. Weighted K-Nearest Known Neighbors (WKNKN) is used as a preprocessing step to complement sparse interaction matrices by assigning likelihood scores to unknown pairs from the nearest known drug and target neighbors [26]. NRLMF incorporates neighborhood regularization into logistic matrix factorization for DTI prediction [9]. Other approaches extend relational modelling through graph-based representations and graph regularization, including GNN/LSTM models and graph-regularized non-negative matrix factorization [16, 27]. These examples show that sparsity and imbalance are not only data issues; they shape the choice of representation, augmentation strategy, and evaluation protocol in DTI prediction.

2.2 Baseline Approaches for DTI Prediction

Traditional deep learning models, including convolutional and recurrent neural networks (CNNs and RNNs), have been widely applied to drug–target interaction (DTI) prediction and computational drug discovery [10, 11]. These models learn nonlinear mappings between drug and target representations and interaction outcomes.

Given a drug–target pair (d_i, t_j), the prediction score is typically modeled as:

$$\hat{Y}(d_i, t_j) = f_\theta(x_{d_i}, x_{t_j}) \tag{1}$$

where x_{d_i} and x_{t_j} denote the feature representations of the drug and target, and f_θ is a neural network parameterized by θ.

Feature-based DTI approaches incorporate chemical structures, drug fingerprints, protein sequences, and learned representations as model inputs [28]. Neural networks employing attention mechanisms can also identify drug and target features that contribute to interaction prediction [29].

Attention-based interaction modeling is defined by:

$$\alpha_{ij} = \frac{\exp\left((h_{d_i} W_q)(h_{t_j} W_k)^\top\right)}{\sum_k \exp\left((h_{d_i} W_q)(h_{t_k} W_k)^\top\right)} \tag{2}$$

where α_{ij} represents the attention weight between drug d_i and target t_j.

The corresponding interaction representation follows:

$$z_{ij} = \alpha_{ij} \cdot (h_{t_j} W_v) \tag{3}$$

where h_{d_i} and h_{t_j} are embeddings of drug d_i and target t_j, and W_q, W_k, W_v are learnable parameters.

Matrix factorization techniques, such as Collaborative Matrix Factorization (CMF), incorporate drug and target similarity features alongside interaction data, creating a robust framework to address noisy and incomplete datasets. For example, the application of soft regularization improves the ability to integrate heterogeneous data sources, enabling large-scale DTI predictions [7].

Matrix factorization models approximate the interaction matrix as:

$$Y \approx UV^\top \tag{4}$$

where $U \in \mathbb{R}^{n_d \times k}$ and $V \in \mathbb{R}^{n_t \times k}$ are latent representations of drugs and targets.

A regularized objective is defined as:

$$\min_{U,V} \|Y - UV^\top\|_F^2 + \lambda \left(\|U\|_F^2 + \|V\|_F^2\right) \tag{5}$$

where λ controls the regularization strength.

The fusion of structural and sequence features provides joint drug–target representations for predictive modelling. Li *et al.* combine drug fingerprint information and protein sequence information for DTI prediction [28].

Feature fusion is represented as:

$$x_{ij} = [\phi(d_i) \,\|\, \psi(t_j)] \tag{6}$$

where $\phi(d_i)$ and $\psi(t_j)$ denote feature encodings of drugs and targets, and $\|$ denotes concatenation.

Autoencoder-based DTI methods combine representation learning with interaction modelling; for example, denoising autoencoders have been integrated with matrix factorization for DTI prediction [30]. Variational autoencoders provide a probabilistic latent-space formulation that supports sampling from learned representations [31].

An autoencoder learns the mapping:

$$z = f_\theta(x), \quad \hat{x} = g_\phi(z) \tag{7}$$

where z is the latent representation and $\hat{x}$ is the reconstruction.

The reconstruction loss is defined as:

$$\mathcal{L} = \|x - \hat{x}\|^2 \tag{8}$$

For VAEs, the objective becomes:

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{q(z|x)}[\log p(x|z)] - D_{KL}(q(z|x)\|p(z)) \tag{9}$$

where $q(z|x)$ is the encoder distribution and $p(z)$ the prior.

These formulations provide latent representations that can be used for reconstruction, sampling, or downstream prediction.

Matrix factorization methods are commonly used for feature extraction, including Multiple Similarities Collaborative Matrix Factorization (MSCMF) [32] and Neighborhood Regularized Logistic Matrix Factorization (NRLMF) [9]. Other approaches include dual-network integrated logistic matrix factorization [8] and heterogeneous-network integration for DTI prediction and drug repositioning [33]. Neural network-based techniques, such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) models, appear in some studies, reflecting an increasing use of deep learning methods [34, 35].

Preprocessing, representation-learning, and augmentation mechanisms vary across DTI studies. Examples include WKNKN preprocessing [26], denoising autoencoders [30], and generative latent-space approaches based on VAEs [31]. Other studies use soft regularization or multiple drug and target similarities to address sparsity without explicit data augmentation [7, 32]. Graph regularization provides another mechanism for incorporating structural information without generating additional samples [27].

Prediction methodologies vary across the DTI literature. Collaborative Matrix Factorization and related matrix-factorization approaches remain widely represented [7, 32]. Ensemble learning is also used for DTI prediction [36, 37], while neural-network and broad-learning approaches provide alternative predictive architectures [12, 26]. Graph-regularized matrix factorization and heterogeneous-network integration provide additional mechanisms for incorporating relational information [27, 33]. These approaches illustrate the diversity of representation and prediction strategies used in DTI modelling.

These representation and prediction strategies provide inputs for downstream neural classifiers. DNN-based DTI models then learn nonlinear mappings between drug and target representations and interaction outcomes [10, 29].

2.3 Data Augmentation for Imbalanced DTI Datasets

Methods addressing data scarcity and class imbalance can operate at the sample, feature, or prediction level. In this review, the relevant strategies are grouped into synthetic data generation, feature perturbation, and ensemble-based prediction. Synthetic generation and feature perturbation modify the training data or its representation, whereas ensemble methods combine multiple model outputs without generating additional observations.

Several mechanisms are relevant to imbalanced DTI learning. Synthetic generation creates additional observations from learned data distributions, most commonly through adversarial or autoencoder-based models [12, 14, 30]. For a standard GAN, the generator and discriminator are optimized through:

$$\min_G \max_D \; \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z)))] \tag{10}$$

where G denotes the generator, D the discriminator, and p_{data} the real data distribution.

The Wasserstein GAN with gradient penalty (WGAN-GP) formulation is given by:

$$\min_G \max_D \; \mathbb{E}_{x\sim P_{\text{real}}}[D(x)] - \mathbb{E}_{\tilde{x}\sim P_G}[D(\tilde{x})] - \lambda\, \mathbb{E}_{\hat{x}} \left[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2 \right] \tag{11}$$

where λ is the gradient penalty coefficient and $\hat{x}$ represents interpolated samples between real and generated observations.

Feature-space perturbation provides a simpler mechanism by introducing controlled variation directly into continuous representations [25]. A Gaussian perturbation can be written as:

$$\tilde{x} = x + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2) \tag{12}$$

where ϵ represents Gaussian noise with variance σ^2.

Ensemble methods address imbalance at the prediction level rather than by modifying the training distribution. Ezzat *et al.* [36] and Zhan *et al.* [37] combine multiple predictors to improve DTI classification under imbalanced conditions. The aggregated prediction is expressed as:

$$\hat{y} = \frac{1}{M} \sum_{m=1}^{M} f_m(x) \tag{13}$$

where f_m denotes the m-th model in the ensemble.

2.3.1 Data Selection and Resampling Methods

Random Over-Sampling (ROS) and Random Under-Sampling (RUS) provide simple baselines for class rebalancing. ROS replicates minority-class observations and can increase overfitting when repeated samples dominate training, whereas RUS reduces the majority class at the cost of potentially discarding informative observations [3, 38]. Hybrid resampling and cleaning methods address these limitations by combining synthesis or selection with boundary cleaning [3, 39].

The SMOTE (Synthetic Minority Over-sampling Technique) family of models represents an advancement in addressing the challenge of class imbalance in machine learning. Introduced by Chawla *et al.* [5], the original SMOTE algorithm improves classifier performance by generating synthetic minority class examples through interpolation between existing samples and their nearest neighbors. This method increases the diversity of the minority class without simply duplicating instances, thus reducing the risk of overfitting. However, the original SMOTE has limitations, especially when dealing with noisy datasets, overlapping classes, or mixed data types. To address these shortcomings, a range of SMOTE variants have been proposed—each introducing targeted heuristics or enhancements [6, 40–42].

Table 1 Comparison of SMOTE variants and related resampling methods.

Method	Sampling Type	Description
Random Over-Sampling	Over	Randomly replicates minority samples
Random Under-Sampling	Under	Randomly removes majority samples
Tomek Links	Under	Removes noisy majority examples
One-Sided Selection (OSS)	Under	CNN + Tomek Links for refined selection
CNN + Tomek Links	Under	CNN followed by Tomek boundary cleaning
Neighborhood Cleaning Rule	Under	ENN-based cleaning of noisy samples
SMOTE	Over	Interpolation using nearest neighbors
SMOTE + Tomek Links	Over	SMOTE plus Tomek-based cleaning
SMOTE + ENN	Over	SMOTE with ENN for deeper cleaning
Basic SMOTE	Over	Original SMOTE interpolation scheme
Borderline-SMOTE	Over	Focus near decision boundary
ADASYN	Over	Focus on hard-to-learn minority samples
k-Means SMOTE	Over	Clustering before SMOTE

2.3.2 Transformation-Based Data Augmentation

Transformation-based augmentation modifies drug, protein, feature, or latent representations while preserving the original class label. For molecules, SMILES enumeration generates alternative valid strings for the same compound by changing atom ordering without changing the molecular structure [43]. For graph-based molecular models, edge dropping, node masking, and subgraph sampling create alternative graph views and have been used in self-supervised and contrastive molecular learning [44, 45]. Protein representation learning also uses sequence masking during language-model pretraining. ProtTrans/ProtBERT, for example, uses self-supervised protein sequence modelling to learn protein representations [46].

Transformation can also be applied directly in feature or embedding space. Controlled noise perturbation is a general data-augmentation mechanism for introducing variation into continuous representations [25]. Mixup interpolates feature vectors and labels to smooth the decision boundary [47]. Latent-space transformations rely on encoded representations, where autoencoders or VAEs allow sampling or perturbation of learned embeddings [31]. Contrastive learning follows a related principle by creating

multiple views of the same entity and using them as positive pairs for representation learning [48].

2.3.3 Synthetic, Generative, and Diffusion-Based Data Augmentation

Synthetic and generative augmentation methods aim to create new samples or latent representations rather than only modifying existing inputs. Prior studies indicate that augmentation can support generalization under data scarcity and class imbalance, including settings where pre-finetuning and augmentation are combined for imbalanced learning [25, 49]. In DTI prediction, these methods are relevant because confirmed interactions are limited and the minority class is often sparse.

GAN-based methods learn to generate synthetic samples from the data distribution and have been combined with matrix factorization or representation learning to produce interaction patterns or latent samples for downstream prediction [12, 14, 25]. VAEs provide another generative path by learning a latent representation and sampling from it to generate reconstructed or perturbed examples [31]. In this study, these methods are treated as augmentation strategies whose value depends on the quality of the latent representation and on whether generated samples preserve the structure of the DTI task.

Diffusion Models for Data Augmentation in DTI Prediction

At their core, diffusion models simulate a process that gradually corrupts data by adding noise in the forward direction and then learn to reverse this process to recover the underlying data distribution. This denoising process is performed through a parameterized neural network trained to estimate the reverse transition from corrupted samples [22, 23].

Diffusion models include Denoising Diffusion Probabilistic Models (DDPMs), score-based models, and stochastic differential equation (SDE)-based formulations. These approaches generate samples through iterative denoising and provide an alternative to adversarial and variational generative models [22, 23, 50].

DDPMs formulate generation through a sequence of forward noising and learned reverse denoising transitions [22], while score-based approaches estimate the score of the data distribution and can be formulated through stochastic differential equations [50]. These formulations provide the theoretical basis for diffusion-based sample generation.

Diffusion processes can also be performed in a compressed latent space rather than directly in the original feature space. Latent diffusion reduces the dimensionality in which the denoising process operates while retaining a learned representation of the original data [51, 52]. This principle is relevant to DTI datasets represented through learned embeddings, where drug–target pairs can first be mapped to latent representations before generative augmentation is applied.

Within a DTI augmentation pipeline, latent representations produced through matrix factorization or other representation-learning methods can serve as the input space for diffusion-based generation. In the present study, diffusion augmentation

is applied to latent drug–target representations, and the generated samples are subsequently incorporated into downstream classification.

Compared with GAN-based augmentation, diffusion models avoid adversarial optimization and instead rely on iterative probabilistic denoising [22, 23]. Their main limitation is the computational cost associated with repeated denoising steps, although latent-space formulations reduce the dimensionality of the generation process [51, 52].

2.4 LLMs, Transformers, and Attention Mechanisms for DTI Prediction

Large Language Models (LLMs) and Transformer architectures have been adapted to DTI prediction because drugs and proteins can be represented as sequences, such as SMILES strings and amino acid sequences [19–21, 46, 53]. The Transformer architecture introduced self-attention as a mechanism for modelling dependencies between sequence elements without recurrence [54]. This principle supports BERT-like models, where masked language modelling (MLM) is used to learn contextual representations by masking tokens and predicting them from surrounding sequence information [17]. In DTI and related biomedical tasks, MLM has been used for molecular and protein representation learning, including SMILES-BERT and ProtTrans/ProtBERT [20, 46].

Transformer-based DTI models use attention to represent interactions between drug and target sequences. Models such as MolTrans, ArkDTA, MMDG-DTI, and DTI-BERT use self-attention or cross-modal attention in DTI modelling [53, 55–57], while SMILES-BERT applies Transformer-based representation learning to molecular sequences [20]. Cross-modal formulations are used to combine drug and target embeddings, while attention weights can help identify sequence regions contributing to the prediction [53, 56, 58]. These models extend earlier CNN- and RNN-based DTI architectures by introducing attention-based sequence modelling [10, 21].

Data augmentation and representation learning in Transformer-based models can be applied at the token, sequence, or representation level. Masked language modelling introduces token-level corruption during pretraining and is used in BERT, ProtTrans/ProtBERT, and SMILES-BERT [17, 20, 46]. SMILES-specific linguistic information has also been incorporated into chemical language models [59]. In DTI-focused architectures, attention, multimodal feature fusion, and domain-generalization mechanisms are used to combine or regularize drug and target representations [53, 55–57].

Existing work spans heterogeneous model architectures and representation strategies, which limits direct comparison across DTI and related biomedical applications [10, 21]. Transformer-based bioinformatics research also considers methodological challenges associated with applying language-model architectures across different biological sequence-modelling tasks [21]. These considerations motivate the controlled benchmarking adopted in this study, in which augmentation strategies are evaluated under a common experimental protocol.

2.5 Hybrid Approaches and Emerging Trends

Recent DTI research combines multiple representation and prediction paradigms rather than relying on a single model family. Sequence- and fingerprint-based representations provide direct drug and protein inputs for deep learning [28], while Transformer-based DTI models use attention and contextual sequence representations [53, 55, 56]. Multimodal approaches also combine heterogeneous drug and target features within shared prediction frameworks [57].

A second trend is the integration of latent representation learning with generative or reconstruction-based methods. Logistic matrix factorization has been combined with generative adversarial networks for DTI prediction [12], while matrix factorization has also been combined with denoising autoencoders [30]. Multiple Similarity Collaborative Matrix Factorization provides another mechanism for integrating interaction and similarity information into latent representations [32].

Transformer and language-model approaches extend representation learning to molecular and protein sequences. BERT-based DTI models, SMILES-BERT, and protein language models demonstrate the use of contextual sequence representations in DTI and related molecular or protein tasks [20, 46, 53]. Multimodal and attention-based DTI models further combine these learned representations with interaction-specific architectures [55–57].

These studies increasingly combine multiple data representations and learning mechanisms within DTI pipelines. Evaluating augmentation methods under a common experimental protocol reduces variation attributable to differences in data partitioning, downstream classification, and evaluation procedures.

3 Methods

3.1 Study Design and Experimental Setting

This study was designed as a controlled computational benchmark of data augmentation, resampling, filtering, and generative strategies for drug–target interaction prediction under class imbalance. Four protein families from the Yamanishi benchmark were evaluated: Enzyme, G-protein-coupled receptor (GPCR), Ion Channel, and Nuclear Receptor.

All experimental configurations were evaluated using the same fixed stratified five-fold outer cross-validation partitions and a common downstream DNN classification protocol. Within each outer-training fold, a fixed stratified 80/20 split was used to define the inner-training and validation subsets. Matrix-factorization models were fitted using inner-training labels only, with validation and outer-test labels excluded from the factorization loss. Augmentation, resampling, and filtering procedures were applied exclusively to the inner-training observations.

The benchmark compared 14 augmentation, resampling, filtering, and generative methods. Classical methods were evaluated using MSCMF representations, while WGAN-GP and VAE were additionally evaluated with NNMF and LMF

representations. Latent Diffusion was evaluated with MSCMF. Four target non-interaction/interaction class distributions were considered: 50/50, 80/20, 85/15, and 90/10.

3.2 Data Representation and Types

To train a classifier for the prediction of drug-target interactions (DTIs), a set of known interactions is required. The objective is to infer whether a given drug–target pair interacts based on observed data. In addition to interaction labels, auxiliary information describing drugs and targets is necessary to construct meaningful representations for predictive modeling.

In this study, we rely on the Yamanishi benchmark datasets [1], which provide gold standard drug–target interaction data across four protein families: enzymes, ion channels, G-protein-coupled receptors (GPCRs), and nuclear receptors. These datasets include binary interaction data as well as similarity matrices for both compounds and proteins. The interaction information is obtained from multiple publicly available databases, including KEGG BRITE, BRENDA, SuperTarget, and DrugBank. Although these databases are continuously updated, the benchmark datasets used in this study correspond to a fixed snapshot and are not dynamically updated.

3.2.1 Interaction Data

The interaction data are provided both as a binary relation list and as an adjacency matrix. For the experimental pipeline, the interaction matrix is represented as $Y \in \mathbb{R}^{n_d \times n_t}$, where rows correspond to drugs and columns to targets, and n_d and n_t denote the number of drugs and targets, respectively. When required, the raw matrix is transposed to conform to this orientation before model fitting.

Each entry Y_{ij} is defined as:

$$Y_{ij} = \begin{cases} 1 & \text{if the interaction is recorded in the Yamanishi gold-standard dataset} \\ 0 & \text{if the interaction is not recorded in the gold-standard dataset.} \end{cases}$$

This matrix corresponds to a bipartite graph in which nodes represent drugs and targets, and edges indicate known interactions. As is typical in DTI datasets, the matrix is sparse due to the limited number of experimentally validated interactions relative to the total number of possible pairs.

For the experimental evaluation, the complete set of drug–target pairs is partitioned using fixed stratified five-fold outer cross-validation. Within each outer-training fold, a stratified 80/20 split defines the inner-training and validation subsets. Validation and outer-test labels are excluded from matrix-factorization fitting, and augmentation is applied only to the inner-training subset. The outer-test fold is retained exclusively for final prediction.

3.2.2 Drug and Target Data

In addition to interaction data, the Yamanishi benchmark provides similarity matrices for both compounds and proteins.

Drug similarity is represented through a compound structure similarity matrix $S_d \in \mathbb{R}^{n_d \times n_d}$. The similarity scores are computed using the SIMCOMP method, which measures structural similarity between chemical compounds based on graph alignment techniques [1].

Target similarity is represented through a protein sequence similarity matrix $S_t \in \mathbb{R}^{n_t \times n_t}$. These similarity scores are computed using the normalized Smith–Waterman alignment score, which evaluates sequence similarity between proteins based on local alignment [1].

Together, the interaction matrix Y and the similarity matrices (S_d, S_t) define the core data structure used for DTI prediction. These matrices support a wide range of computational approaches, including similarity-based inference, matrix factorization, graph-based modeling, and deep learning methods, particularly in settings characterized by sparse and imbalanced data.

3.3 Problem Formulation

Let $Y \in \{0,1\}^{n_d \times n_t}$ denote the drug–target interaction matrix, where n_d and n_t are the numbers of drugs and targets, respectively. An entry $Y_{ij} = 1$ denotes a validated interaction between drug i and target j, whereas $Y_{ij} = 0$ denotes an unobserved interaction. For the binary classification experiments, zero-valued entries are treated operationally as non-interactions; they are not assumed to represent experimentally confirmed negative interactions.

For a drug–target pair (i, j), the classifier estimates a score

$$f_\theta(i,j) \approx \Pr\left(Y_{ij} = 1 \mid \mathbf{x}_{ij}\right), \tag{14}$$

where $\mathbf{x}_{ij}$ denotes the latent representation of the pair and θ represents the parameters of the DNN classifier.

The latent representation $\mathbf{x}_{ij}$ is constructed from the embeddings obtained through matrix factorization. Because validated interactions represent a small fraction of the complete drug–target matrix, the resulting classification problem is highly imbalanced. The augmentation and resampling strategies examined in this study therefore modify the class distribution only within the inner-training data, while the validation and outer-test subsets remain unchanged.

3.4 Experimental Pipeline

The experimental pipeline follows the matrix-factorization, generative-augmentation, and DNN framework introduced in our previous work [60], with an explicit data-partitioning stage to separate representation learning, augmentation, classifier training, and final evaluation

For each protein family, the complete drug–target pair dataset is partitioned using fixed stratified five-fold outer cross-validation. The same outer-fold assignments are used for every experimental configuration. Within each outer-training fold, a second stratified split assigns 80% of the observations to inner training and 20% to validation. This inner partition is also fixed across methods. The outer-test fold is used only for final prediction.

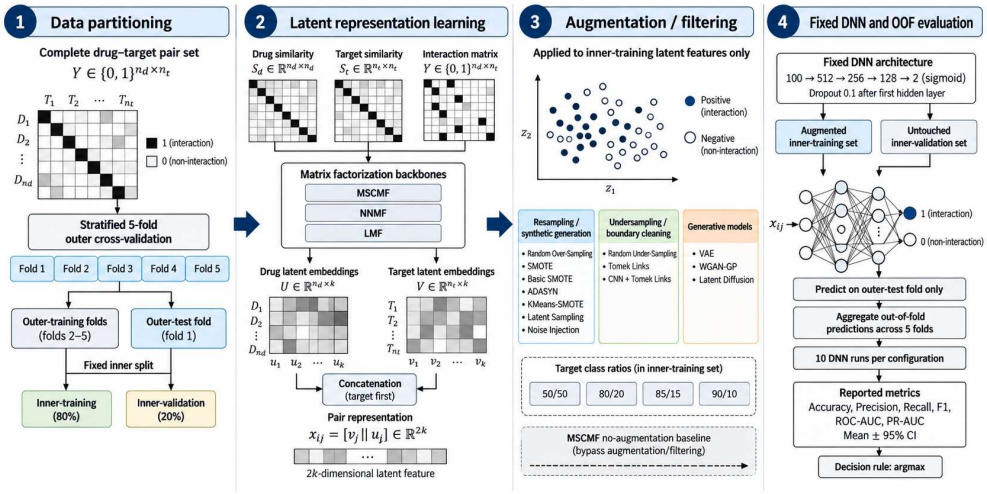


Fig. 1 Experimental pipeline used for DTI prediction. The complete drug–target pair dataset is partitioned using fixed stratified five-fold outer cross-validation. Within each outer fold, the outer-training observations are further divided into inner-training and validation subsets. Matrix factorization is fitted using only labels permitted for representation learning, with validation and outer-test labels masked from the factorization loss. Augmentation, resampling, or noise reduction is applied exclusively to the inner-training subset. The DNN is trained on the resulting training data and evaluated on the untouched outer-test fold. Predictions from the five outer folds are concatenated to obtain one full-family out-of-fold prediction vector per stochastic run. The complete procedure is repeated for ten stochastic runs under the same fixed data partitions.

Matrix factorization is fitted using the inner-training labels. Labels belonging to the validation and outer-test subsets are masked from the factorization loss and are not replaced by zeros. The resulting latent embeddings are therefore estimated without using the labels subsequently employed for validation or evaluation.

Three factorization backbones are considered: Multiple Similarity Collaborative Matrix Factorization (MSCMF), Non-negative Matrix Factorization (NNMF), and Logistic Matrix Factorization (LMF). The latent dimension is fixed at 50 for each entity type. A drug–target pair is represented by concatenating the corresponding 50-dimensional target and drug embeddings, producing a 100-dimensional feature vector. In accordance with the implementation, the target representation precedes the drug representation:

$$x_{ij} = [v_j \| u_i], \tag{15}$$

where u_i denotes the latent representation of drug i, v_j denotes the latent representation of target j, and $\|$ denotes vector concatenation.

For MSCMF, drug and target similarity information enters through graph-Laplacian regularization [32]. Let Ω_{tr} denote the set of interaction entries assigned to the inner-training subset. The masked objective can be written as

$$\min_{U,V} \left\| \mathcal{P}_{\Omega_{\mathrm{tr}}} \left(Y - UV^\top \right) \right\|_F^2 + \lambda_d \operatorname{Tr}(U^\top L_d U) + \lambda_t \operatorname{Tr}(V^\top L_t V) + \gamma \left(\|U\|_F^2 + \|V\|_F^2 \right), \tag{16}$$

where $\mathcal{P}_{\Omega_{\text{tr}}}$ restricts the reconstruction loss to inner-training entries, L_d and L_t denote the drug and target graph Laplacians, and λ_d, λ_t, and γ are regularization parameters. NNMF and LMF provide alternative latent representations and are fitted under the same train/validation/test masking protocol. Their formulations are described in Sections 3.5.2 and 3.5.3.

For a given protein family, outer fold, and factorization backbone, the latent representations are fitted once and retained for the downstream experiments. They are not re-estimated during the repeated DNN runs.

Figure 2 illustrates this representation-learning stage.

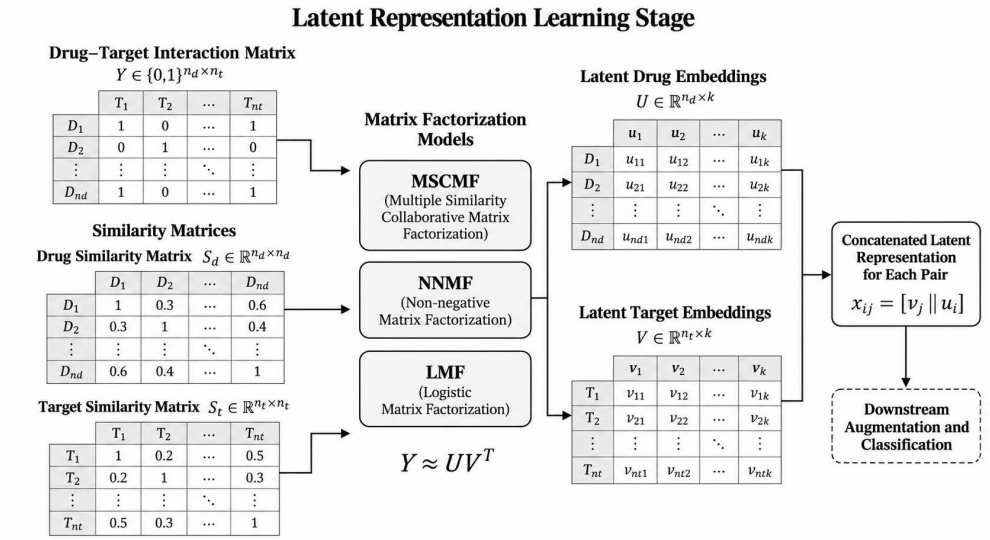


Fig. 2 Latent representation learning stage. Matrix factorization is fitted using inner-training labels, while validation and outer-test labels are masked from the factorization loss. MSCMF incorporates drug and target similarity information through graph-Laplacian regularization; NNMF and LMF provide alternative latent factorizations. Each drug–target pair is represented by the concatenation of 50-dimensional target and drug embeddings.

Augmentation, resampling, and filtering are performed after latent representation learning and only on the inner-training observations. Neither the validation subset nor the outer-test fold is modified. The evaluated procedures are Random Over-Sampling, Random Under-Sampling, Tomek Links, CNN + Tomek Links, SMOTE, SMOTE + Tomek Links, Basic SMOTE, ADASYN, KMeans-SMOTE, Latent Sampling, Noise Injection, VAE, WGAN-GP, and Latent Diffusion.

Four target class distributions are evaluated: 50/50, 80/20, 85/15, and 90/10. The first value denotes the proportion of non-interactions and the second the proportion of interactions in the resulting inner-training data. The validation and outer-test class distributions are not altered.

The augmentation operation can be expressed at the training-set level as

$$\widetilde{\mathcal{T}}_{\text{train}} = \mathcal{A}\left(\mathcal{T}_{\text{train}}\right), \tag{17}$$

where $\mathcal{A}$ denotes the augmentation, resampling, or filtering procedure and $\widetilde{\mathcal{T}}_{\text{train}}$ is the training set supplied to the classifier.

For interpolation-based SMOTE variants, synthetic observations follow

$$x_{\text{new}} = x_i + \delta(x_{zi} - x_i), \qquad \delta \in [0, 1], \tag{18}$$

where x_i is a minority-class observation and x_{zi} is one of its selected neighbors. Noise Injection uses

$$\tilde{x} = x + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma^2). \tag{19}$$

The formulations used for WGAN-GP, VAE, and Latent Diffusion are given in Sections 3.7 and the corresponding method-specific equations.

The backbone comparison is not identical for every augmentation method. The classical resampling, filtering, interpolation, latent-sampling, and noise-injection methods are evaluated using MSCMF representations. WGAN-GP and VAE are evaluated with MSCMF, NNMF, and LMF representations, while Latent Diffusion is evaluated with MSCMF. This produces 18 augmentation–backbone combinations. Each combination is evaluated at the four target class ratios, and an MSCMF configuration without data augmentation is retained as the common baseline, resulting in 73 configurations per protein family.

Figure 3 summarizes the training-only augmentation stage.

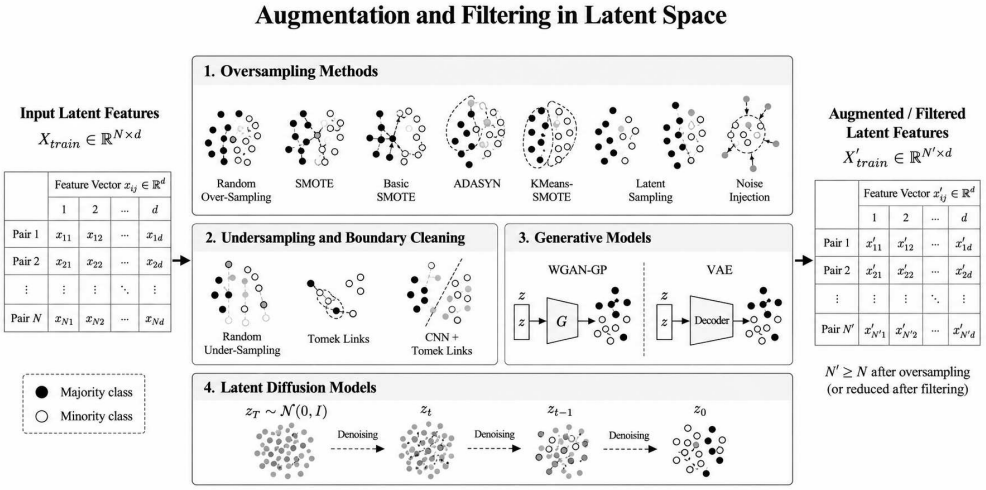


Fig. 3 Training-only augmentation stage. Augmentation, resampling, filtering, and generative procedures operate on latent representations from the inner-training subset. Validation and outer-test observations remain unchanged. Four non-interaction/interaction target distributions are evaluated: 50/50, 80/20, 85/15, and 90/10.

3.5 Matrix Factorization Methods

The benchmark uses three matrix-factorization backbones: Multiple Similarity Collaborative Matrix Factorization (MSCMF), Non-negative Matrix Factorization (NNMF), and Logistic Matrix Factorization (LMF). The latent dimension is fixed at $k = 50$ for all three methods.

Let Ω_{tr} denote the interaction entries assigned to the inner-training subset for a given outer fold. Factorization losses are evaluated only on entries in Ω_{tr}. Labels belonging to the validation subset and the outer-test fold are therefore excluded from factorization rather than entered as zero-valued observations. The learned drug and target embeddings are subsequently combined according to Equation 15 to form the 100-dimensional pair representation used by the downstream augmentation and classification stages.

3.5.1 Multiple Similarity Collaborative Matrix Factorization (MSCMF)

MSCMF combines interaction reconstruction with graph regularization derived from drug and target similarity information [32]. Let

$$U \in \mathbb{R}^{n_d \times k}, \qquad V \in \mathbb{R}^{n_t \times k} \tag{20}$$

denote the drug and target latent-factor matrices, respectively. The reconstructed interaction matrix is

$$\hat{Y} = UV^\top. \tag{21}$$

Drug and target similarity matrices are denoted by S_d and S_t. Their graph Laplacians are

$$L_d = D_d - S_d, \qquad L_t = D_t - S_t, \tag{22}$$

where D_d and D_t are the corresponding diagonal degree matrices.

Under the leakage-safe protocol, reconstruction is restricted to the inner-training entries. The MSCMF objective is

$$\min_{U,V} \left\| \mathcal{P}_{\Omega_{\text{tr}}} \left(Y - UV^\top \right) \right\|_F^2 + \lambda_d \operatorname{Tr}\left(U^\top L_d U \right) + \lambda_t \operatorname{Tr}\left(V^\top L_t V \right) + \gamma \left(\|U\|_F^2 + \|V\|_F^2 \right), \tag{23}$$

where $\mathcal{P}_{\Omega_{\text{tr}}}$ retains only entries belonging to the inner-training subset. Parameters λ_d and λ_t control the graph regularization terms, while γ controls the magnitude of the latent factors.

For a drug d_i and target t_j, the reconstructed interaction score is

$$\hat{Y}_{ij} = u_i^\top v_j, \tag{24}$$

where u_i and v_j are the corresponding latent vectors.

3.5.2 Non-negative Matrix Factorization (NNMF)

NNMF provides a factorization in which the drug and target latent factors are constrained to be non-negative [27]. The approximation is

$$\hat{Y} = UV^\top, \tag{25}$$

subject to

$$U \geq 0, \qquad V \geq 0. \tag{26}$$

As with MSCMF, validation and outer-test labels are excluded from the factorization loss. NNMF therefore minimizes the reconstruction error only over the inner-training entries:

$$\min_{U,V \geq 0} \left\| \mathcal{P}_{\Omega_{\text{tr}}} \left(Y - UV^\top \right) \right\|_F^2 . \tag{27}$$

No additional latent-factor regularization term is introduced in the NNMF implementation used in the benchmark.

The reconstructed score for pair (i, j) is

$$\hat{Y}_{ij} = u_i^\top v_j. \tag{28}$$

NNMF does not use the drug and target similarity matrices employed by MSCMF. Its role in the benchmark is to provide an alternative latent representation under the same data partitions, latent dimension, and masking protocol.

3.5.3 Logistic Matrix Factorization (LMF)

LMF models the binary interaction probability directly from the inner product of the drug and target latent factors [9]. For drug d_i and target t_j, the estimated interaction probability is

$$p_{ij} = P(Y_{ij} = 1 \mid u_i, v_j) = \sigma\left(u_i^\top v_j\right), \tag{29}$$

with

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{30}$$

The loss is evaluated only for interaction entries contained in Ω_{tr}:

$$\mathcal{J}_{\text{LMF}} = - \sum_{(i,j) \in \Omega_{\text{tr}}} \left[Y_{ij} \log(p_{ij}) + (1 - Y_{ij}) \log(1 - p_{ij}) \right]. \tag{31}$$

The optimization problem is therefore

$$\min_{U,V} \mathcal{J}_{\text{LMF}}. \tag{32}$$

No additional L2 regularization term is included in the LMF implementation used in the benchmark. As for MSCMF and NNMF, validation and outer-test labels do not contribute to parameter estimation.

LMF uses the same latent dimension, $k = 50$, as MSCMF and NNMF. The resulting drug and target embeddings are concatenated to obtain the pair-level representation defined in Equation 15.

3.6 Resampling, Synthetic Oversampling, and Boundary Cleaning Methods

The non-generative benchmark includes random resampling, interpolation-based oversampling, adaptive oversampling, cluster-based oversampling, latent-distribution sampling, perturbation-based augmentation, and boundary cleaning. All procedures operate only on the inner-training subset defined in Section 3.4. Validation and outer-test observations are not modified.

3.6.1 Oversampling Methods

Random Over-Sampling increases the number of interaction observations by sampling with replacement from the positive class [3]. If $\mathcal{X}_+$ denotes the set of positive training observations, additional samples are drawn from $\mathcal{X}_+$ until the required class distribution is reached. The feature values of the replicated observations are not altered.

SMOTE generates synthetic positive observations by interpolation between a sample and one of its minority-class neighbors [5]. For a positive observation x_i and a selected positive neighbor x_{zi},

$$x_{\text{new}} = x_i + \delta(x_{zi} - x_i), \qquad \delta \in [0, 1]. \tag{33}$$

Two SMOTE implementations are retained in the benchmark. In the configuration denoted SMOTE, the latent features are standardized using statistics estimated from the inner-training data before synthetic generation and transformed back to the original latent-feature scale afterwards. Basic SMOTE applies the same interpolation procedure directly to the latent features without this preliminary standardization.

ADASYN also generates synthetic minority observations, but allocates more samples to positive observations located in regions with a higher proportion of neighboring majority-class observations [40]. The sampling allocation for observation i is proportional to its local imbalance measure:

$$r_i = \frac{\Delta_i}{\sum_{j=1}^{N_+} \Delta_j}, \tag{34}$$

where Δ_i measures the local majority-class concentration around positive observation x_i and N_+ is the number of positive observations considered during resampling.

KMeans-SMOTE first partitions the latent observations into clusters and subsequently applies SMOTE within eligible clusters [42]. In the implementation used here, clustering is performed with $K = 5$.

For an observation x_i, cluster assignment follows

$$c_i = \arg\min_k \|x_i - \mu_k\|_2^2, \tag{35}$$

where μ_k denotes the centroid of cluster k. Synthetic observations are then generated within clusters satisfying the resampling criterion.

Latent Sampling differs from interpolation-based methods. A Gaussian mixture model with one component and full covariance is fitted to the positive inner-training representations. Synthetic positive observations are sampled from the estimated distribution

$$x_{\text{new}} \sim \mathcal{N}(\mu_+, \Sigma_+), \tag{36}$$

where μ_+ and Σ_+ are estimated from the positive latent-feature vectors.

Noise Injection generates additional positive observations by perturbing existing positive latent representations:

$$\tilde{x} = x + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma^2 I), \tag{37}$$

with $\sigma = 0.01$ in the experiments.

3.6.2 Undersampling and Boundary Cleaning Methods

Random Under-Sampling reduces the number of non-interaction observations by sampling the majority class without replacement [3]. Let $\mathcal{X}_-$ denote the non-interaction training observations. The retained majority subset satisfies

$$\mathcal{X}'_- \subset \mathcal{X}_-, \tag{38}$$

with its size determined by the target class distribution.

Tomek Links identify pairs of observations from opposite classes that are mutual nearest neighbors [39]. A pair (x_i, x_j) forms a Tomek link when

$$d(x_i, x_j) = \min_{k \neq i} d(x_i, x_k) \tag{39}$$

and

$$d(x_i, x_j) = \min_{k \neq j} d(x_j, x_k), \tag{40}$$

with $y_i \neq y_j$. The corresponding cleaning operation removes observations associated with overlapping class boundaries.

The CNN + Tomek configuration first applies Condensed Nearest Neighbor reduction to the inner-training data, followed by Tomek-Link cleaning. The resulting cleaned set defines the admissible pool of real training observations. This pool is then adjusted to the required 50/50, 80/20, 85/15, or 90/10 class distribution using the benchmark ratio-control procedure. The reported CNN + Tomek configurations therefore correspond to boundary cleaning followed by class-ratio adjustment.

SMOTE + Tomek combines synthetic minority oversampling with subsequent Tomek-Link cleaning. The latent features are standardized using the inner-training data, SMOTE is applied in the standardized feature space, Tomek Links remove overlapping observations, and the resulting features are transformed back to the original latent-feature scale.

These procedures modify only the inner-training observations. Their outputs are subsequently adjusted to the corresponding 50/50, 80/20, 85/15, or 90/10 target distribution before DNN training.

3.7 Generative Augmentation Models

Three generative methods are evaluated: WGAN-GP, VAE, and Latent Diffusion. WGAN-GP and VAE are applied to MSCMF, NNMF, and LMF representations, whereas Latent Diffusion is evaluated with MSCMF. In all cases, generative models are fitted to positive observations from the inner-training subset only.

3.7.1 Wasserstein Generative Adversarial Networks with Gradient Penalty (WGAN-GP)

WGAN-GP generates synthetic positive-class representations through adversarial training using the Wasserstein objective with gradient regularization [12, 14, 61, 62]. The generator maps a random vector z to the 100-dimensional factorization feature space:

$$\tilde{x} = G(z), \qquad z \sim \mathcal{N}(0, I). \tag{41}$$

The gradient penalty is evaluated on interpolated observations

$$\hat{x} = \epsilon x + (1 - \epsilon)\tilde{x}, \qquad \epsilon \sim \mathcal{U}(0, 1), \tag{42}$$

where x is a real positive-class observation. The penalty coefficient is fixed at $\lambda = 10$. The complete WGAN-GP objective is given in Equation 11.

Positive inner-training observations are standardized before adversarial training. Generated samples are transformed back to the original factorization-feature scale before being added to the training set.

3.7.2 Variational Autoencoders (VAE)

The VAE models positive-class factorization representations through an encoder–decoder structure [31]. The encoder estimates a Gaussian latent distribution,

$$q_\phi(z|x) = \mathcal{N}\left(z; \mu_\phi(x), \mathrm{diag}(\sigma^2_\phi(x))\right), \tag{43}$$

from which latent vectors are sampled using

$$z = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I). \tag{44}$$

The decoder maps the sampled representation back to the factorization feature space:

$$\hat{x} = D_\theta(z). \tag{45}$$

Because the input features are continuous, reconstruction is measured by squared error. The objective is

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{q_\phi(z|x)} \left[\|x - D_\theta(z)\|_2^2 \right] + D_{\text{KL}} \left(q_\phi(z|x) \| \mathcal{N}(0, I) \right). \tag{46}$$

Synthetic samples are obtained by drawing from the latent prior and decoding:

$$z_{\text{new}} \sim \mathcal{N}(0, I), \qquad \tilde{x} = D_\theta(z_{\text{new}}). \tag{47}$$

As for WGAN-GP, standardization is estimated from the positive inner-training observations and reversed after generation.

3.7.3 Latent Diffusion Models

Latent Diffusion applies the diffusion process in a compressed representation space rather than directly to the 100-dimensional MSCMF features [22, 23, 51, 52]. A β-VAE first maps the MSCMF pair representation x to a 32-dimensional latent vector:

$$z_0 = E_\phi(x). \tag{48}$$

The β-VAE is optimized using

$$\mathcal{L}_{\beta\text{-VAE}} = \|x - \hat{x}\|_2^2 + 0.1\, D_{\text{KL}} \left(q_\phi(z|x) \| \mathcal{N}(0, I) \right). \tag{49}$$

The forward diffusion process is

$$q(z_t|z_{t-1}) = \mathcal{N}\left(z_t; \sqrt{1 - \beta_t}\, z_{t-1}, \beta_t I \right), \tag{50}$$

with

$$\bar{\alpha}_t = \prod_{s=1}^{t} (1 - \beta_s). \tag{51}$$

A noisy latent vector can therefore be written as

$$z_t = \sqrt{\bar{\alpha}_t}\, z_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I). \tag{52}$$

The denoising network is trained to estimate the injected noise:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{z_0, \epsilon, t} \left[\|\epsilon - \epsilon_\theta(z_t, c, t)\|_2^2 \right], \tag{53}$$

where c denotes the conditioning representation. Synthetic samples are produced through iterative reverse denoising and decoded back to the MSCMF feature space:

$$\tilde{x} = D_\theta(z_0). \tag{54}$$

Table 2 Implementation parameters for generative augmentation models.

Parameter	WGAN-GP	VAE	Latent Diffusion
Input dimension	100	100	100
Generated feature dimension	100	100	100
Generative latent dimension	100	16	32
Main hidden dimensions	G: 256–512–1024	256–64 / 64–256	β-VAE: 128–64 / 64–128
Denoising hidden dimensions	–	–	128–128
Training epochs	100	4	10 (β-VAE), 50 (diffusion)
Batch size	256	77	–
Learning rate	10^{-4}	Adam default	10^{-3} (β-VAE), 10^{-4} (diffusion)
Gradient penalty	10	–	–
Diffusion steps	–	–	1000
Noise schedule	–	–	10^{-4} to 0.02
Reverse sampling steps	–	–	100

Table 2 summarizes the implementation parameters used for the three generative models.

3.8 DNN classification and Evaluation

Before classifier fitting, feature standardization is estimated from the final inner-training data only. The same transformation is then applied to the fixed validation subset and the corresponding outer-test fold.

A common DNN architecture is used for all configurations. For a pair representation x_{ij}, the classifier produces a two-dimensional output,

$$\hat{y}_{ij} = f_\theta(x_{ij}), \qquad \hat{y}_{ij} \in [0,1]^2, \tag{55}$$

where f_θ denotes the neural classifier. Labels are represented using two output classes. The network contains fully connected hidden layers of 512, 256, and 128 units with ReLU activation. A dropout layer with rate 0.1 follows the first hidden layer, and the output layer contains two sigmoid units. Discrete class predictions are obtained by selecting the output unit with the highest score using argmax. Accuracy, precision, recall, and F1-score are computed from these predicted classes. ROC-AUC and PR-AUC are computed using the second output score, corresponding to the interaction class. PR-AUC is calculated by trapezoidal integration of the precision–recall curve.

The DNN is trained for 12 epochs using Adam and binary cross-entropy. The fixed inner-validation subset is supplied explicitly during training; no additional validation split is created after augmentation and no early-stopping procedure is used. Table 3 reports the parameters shared across the four protein families.

3.9 Statistical Analysis

Each configuration is evaluated through ten stochastic DNN training runs under the same fixed outer and inner partitions. A run comprises all five outer folds. After

Table 3 DNN architecture and training parameters used across protein families.

Parameter	**Enzyme | GPCR | IC | NR**
Input dimension	100
Hidden layers	512–256–128
Hidden activation	ReLU
Dropout	0.1 (after first hidden layer)
Output units	2
Final activation	Sigmoid
Label encoding	One-hot
Loss function	Binary cross-entropy
Optimizer	Adam
Learning rate	0.001
Adam β_1	0.9
Adam β_2	0.999
Epsilon	1×10^{-7}
L2 regularization	None
Epochs	12
Early stopping	None
Batch size	256 | 128 | 128 | 32

training on a given fold, predictions are generated only for that fold's untouched outer-test observations. The five test-fold prediction vectors are then concatenated, yielding one out-of-fold prediction for every original drug–target pair in the protein family.

Accuracy, precision, recall, F1-score, ROC-AUC, and PR-AUC are computed once from this pooled full-family out-of-fold vector. Fold-level metrics are not averaged to produce the reported result. The procedure yields ten run-level values for each metric and experimental configuration, from which the final mean and 95% confidence interval are calculated.

The final result for each experimental configuration is obtained from the ten run-level out-of-fold evaluations. Results are reported as the mean and 95% confidence interval across these ten stochastic runs. Confidence intervals are calculated using a two-tailed Student's t-distribution with $df = 9$ and $t_{0.975,9} = 2.262$. These intervals quantify stochastic DNN-training variability under the fixed data partitions; they do not quantify variability associated with alternative cross-validation partitions. Each metric is computed separately for each run before averaging, so the reported mean F1-score does not necessarily equal the harmonic mean of the reported mean precision and recall.

4 Results

4.1 Result Reporting and Table Structure

Results are reported for each method–ratio configuration. The 50/50, 80/20, 85/15, and 90/10 settings denote the non-interaction/interaction class distribution of the inner-training data after augmentation or resampling. Validation and outer-test observations retain their original class distributions.

For each configuration, predictions from the five outer-test folds are concatenated to form one out-of-fold prediction vector covering the complete protein-family dataset. Accuracy, precision, recall, F1-score, ROC-AUC, and PR-AUC are computed from this pooled vector. The procedure is repeated for ten stochastic DNN training runs under the same fixed data partitions.

Tables 5–8 report the results for Enzyme, GPCR, Ion Channel, and Nuclear Receptor, respectively. Values are given as the mean and 95% confidence interval across the ten run-level out-of-fold evaluations. Confidence intervals use a two-tailed Student's t-distribution with $df = 9$ and $t_{0.975,9} = 2.262$. They describe variability across stochastic training runs under fixed data partitions.

Each metric is computed separately for each run before averaging. The reported mean F1-score therefore does not necessarily equal the harmonic mean of the reported mean precision and recall.

4.2 Family-Level Performance

Performance differs across protein families. Enzyme and Ion Channel show the highest baseline performance, with relatively small changes after augmentation. For Enzyme, the MSCMF configuration without augmentation reaches an F1-score of 0.7777. The highest F1-score, 0.7824, is obtained with k-Means SMOTE at 90/10, while Latent Sampling at the same ratio gives the highest PR-AUC (0.7416). Tomek Links with ratio adjustment at 50/50 produces the highest ROC-AUC (0.9589), but its lower F1-score shows that the improvement in ranking performance does not translate into higher discrete classification performance.

For Ion Channel, the no-augmentation MSCMF configuration reaches an F1-score of 0.8440 and a PR-AUC of 0.8737. Random Over-Sampling at 80/20 increases the F1-score to 0.8527, while SMOTE + Tomek Links at 85/15 gives the highest PR-AUC (0.8768). Differences among the highest-performing configurations are small.

GPCR shows greater variation across augmentation methods. The MSCMF baseline reaches an F1-score of 0.5656, compared with 0.5833 for Tomek Links with ratio adjustment at 90/10. MSCMF + VAE at 80/20 gives the highest PR-AUC (0.5606), whereas Basic SMOTE at 85/15 gives the highest ROC-AUC (0.8949).

For Nuclear Receptor, the MSCMF baseline gives an F1-score of 0.3568, while Random Under-Sampling at 90/10 reaches 0.3705. Noise Injection at 50/50 provides the highest PR-AUC (0.3865), and Latent Diffusion at 80/20 produces the highest ROC-AUC (0.8231). As for GPCR, the highest-performing configuration depends on the evaluation metric.

4.3 Trends Across Methods and Class Ratios

No augmentation method or target ratio performs best across all four protein families. The highest F1-score is obtained with k-Means SMOTE at 90/10 for Enzyme, Tomek Links with ratio adjustment at 90/10 for GPCR, Random Over-Sampling at 80/20 for Ion Channel, and Random Under-Sampling at 90/10 for Nuclear Receptor. Rankings also vary by metric: MSCMF with VAE gives the highest GPCR PR-AUC, while Latent Diffusion gives the highest Nuclear Receptor ROC-AUC.

The effect of the target ratio also varies across methods. WGAN-GP, VAE, and Latent Diffusion show limited variation across the four ratios, whereas some undersampling and boundary-cleaning configurations show larger changes in precision and recall. CNN + Tomek Links exhibits some of the largest shifts across ratios. The MSCMF configuration without augmentation also remains close to the best augmented configurations, particularly for Enzyme and Ion Channel. These results show that augmentation performance depends on the protein family, class ratio, representation, and evaluation metric.

4.4 Performance of the Reference Framework

The MSCMF + WGAN-GP + DNN framework from our previous work [60] is retained as a reference configuration. For each protein family, Table 4 reports the best observed MSCMF + WGAN-GP configuration by mean F1-score among the four evaluated ratios.

The selected configurations achieve accuracies of 99.62%, 97.85%, 99.00%, and 94.02% for Enzyme, GPCR, Ion Channel, and Nuclear Receptor, respectively. Compared with DeepLSTM with SPCA [35] and the fingerprint- and sequence-based model [28], the reference MSCMF + WGAN-GP + DNN configuration reports higher accuracy across all four families, while ROC-AUC differences vary by dataset.

Because the compared studies use different representations, architectures, and evaluation protocols, these values should be interpreted as contextual rather than head-to-head comparisons.

Table 4 Baseline comparison with prior work and the reference MSCMF + WGAN-GP + DNN framework. For the reference framework, the ratio with the highest mean F1-score among the four MSCMF + WGAN-GP configurations is reported for each protein family. Prior baselines did not use data augmentation.

Method	protein family	Ratio	Acc. (%)	ROC-AUC (%)
DeepLSTM with SPCA [35]	Enzyme	–	92.9	99.5
	GPCR	–	91.8	99.5
	Ion Channel	–	92.0	97.1
	Nuclear Receptor	–	91.1	92.1
Fingerprint and sequence-based [28]	Enzyme	–	89.2	94.7
	GPCR	–	82.2	86.5
	Ion Channel	–	86.0	91.5
	Nuclear Receptor	–	71.7	78.0
Reference MSCMF + WGAN-GP	Enzyme	50/50	99.62	94.17
	GPCR	90/10	97.85	88.45
	Ion Channel	90/10	99.00	97.13
	Nuclear Receptor	85/15	94.02	79.30

Table 5 Performance comparison of augmentation and resampling configurations for the Enzyme family. Values are reported as mean [95% CI] over 10 runs.

Method	Ratio	Acc.	Prec.	Rec.	F1	ROC-AUC	PR-AUC
MSCMF (No DA)	No aug.	0.9961 [0.9959, 0.9962]	0.8819 [0.8680, 0.8958]	0.6957 [0.6930, 0.6985]	0.7777 [0.7721, 0.7833]	0.9422 [0.9395, 0.9449]	0.7342 [0.7287, 0.7398]
MSCMF (WGAN-GP)	50/50	0.9962 [0.9961, 0.9962]	0.8944 [0.8870, 0.9018]	0.6938 [0.6914, 0.6963]	0.7814 [0.7785, 0.7844]	0.9417 [0.9395, 0.9439]	0.7377 [0.7351, 0.7403]
	80/20	0.9961 [0.9960, 0.9962]	0.8933 [0.8824, 0.9042]	0.6923 [0.6886, 0.6959]	0.7800 [0.7758, 0.7841]	0.9423 [0.9397, 0.9449]	0.7357 [0.7317, 0.7397]
	85/15	0.9961 [0.9961, 0.9962]	0.8927 [0.8887, 0.8967]	0.6924 [0.6884, 0.6963]	0.7799 [0.7773, 0.7824]	0.9431 [0.9409, 0.9452]	0.7357 [0.7320, 0.7395]
	90/10	0.9961 [0.9960, 0.9962]	0.8896 [0.8805, 0.8987]	0.6940 [0.6896, 0.6983]	0.7796 [0.7758, 0.7835]	0.9437 [0.9408, 0.9466]	0.7373 [0.7340, 0.7407]
MSCMF (VAE)	50/50	0.9961 [0.9961, 0.9962]	0.8912 [0.8837, 0.8987]	0.6940 [0.6902, 0.6979]	0.7803 [0.7781, 0.7824]	0.9410 [0.9373, 0.9448]	0.7366 [0.7325, 0.7407]
	80/20	0.9961 [0.9961, 0.9962]	0.8934 [0.8856, 0.9013]	0.6938 [0.6887, 0.6990]	0.7810 [0.7783, 0.7837]	0.9414 [0.9381, 0.9447]	0.7412 [0.7369, 0.7455]
	85/15	0.9961 [0.9960, 0.9962]	0.8887 [0.8741, 0.9034]	0.6923 [0.6887, 0.6960]	0.7782 [0.7730, 0.7834]	0.9424 [0.9396, 0.9452]	0.7401 [0.7369, 0.7434]
	90/10	0.9961 [0.9960, 0.9962]	0.8870 [0.8808, 0.8933]	0.6944 [0.6893, 0.6995]	0.7789 [0.7753, 0.7826]	0.9439 [0.9417, 0.9460]	0.7406 [0.7364, 0.7449]
NNMF (WGAN-GP)	50/50	0.9957 [0.9957, 0.9957]	0.9558 [0.9524, 0.9592]	0.5943 [0.5915, 0.5970]	0.7329 [0.7311, 0.7346]	0.9426 [0.9412, 0.9440]	0.7110 [0.7088, 0.7132]
	80/20	0.9957 [0.9956, 0.9958]	0.9544 [0.9502, 0.9587]	0.5951 [0.5886, 0.6016]	0.7330 [0.7281, 0.7380]	0.9422 [0.9401, 0.9444]	0.7100 [0.7058, 0.7141]
	85/15	0.9957 [0.9957, 0.9958]	0.9518 [0.9463, 0.9572]	0.5968 [0.5934, 0.6002]	0.7336 [0.7302, 0.7370]	0.9449 [0.9442, 0.9457]	0.7124 [0.7101, 0.7146]
	90/10	0.9957 [0.9956, 0.9958]	0.9462 [0.9314, 0.9609]	0.5977 [0.5945, 0.6009]	0.7325 [0.7288, 0.7362]	0.9447 [0.9433, 0.9461]	0.7127 [0.7109, 0.7145]
LMF (WGAN-GP)	50/50	0.9960 [0.9959, 0.9961]	0.9316 [0.9206, 0.9426]	0.6445 [0.6401, 0.6488]	0.7618 [0.7577, 0.7659]	0.9053 [0.9032, 0.9073]	0.6951 [0.6931, 0.6971]
	80/20	0.9960 [0.9959, 0.9961]	0.9299 [0.9212, 0.9386]	0.6468 [0.6443, 0.6493]	0.7629 [0.7592, 0.7667]	0.9050 [0.9034, 0.9065]	0.6953 [0.6936, 0.6970]
	85/15	0.9960 [0.9959, 0.9961]	0.9259 [0.9113, 0.9404]	0.6484 [0.6423, 0.6544]	0.7625 [0.7586, 0.7664]	0.9052 [0.9025, 0.9079]	0.6953 [0.6923, 0.6984]
	90/10	0.9961 [0.9960, 0.9961]	0.9320 [0.9218, 0.9421]	0.6487 [0.6457, 0.6516]	0.7649 [0.7611, 0.7686]	0.9061 [0.9046, 0.9077]	0.6963 [0.6946, 0.6980]
NNMF (VAE)	50/50	0.9956 [0.9955, 0.9958]	0.9431 [0.9206, 0.9656]	0.5972 [0.5946, 0.5998]	0.7311 [0.7244, 0.7379]	0.9441 [0.9424, 0.9457]	0.7111 [0.7081, 0.7142]
	80/20	0.9957 [0.9957, 0.9957]	0.9523 [0.9471, 0.9574]	0.5943 [0.5920, 0.5966]	0.7318 [0.7298, 0.7339]	0.9446 [0.9436, 0.9456]	0.7116 [0.7095, 0.7138]
	85/15	0.9957 [0.9957, 0.9957]	0.9553 [0.9513, 0.9592]	0.5950 [0.5907, 0.5994]	0.7333 [0.7304, 0.7361]	0.9443 [0.9431, 0.9454]	0.7118 [0.7092, 0.7145]
	90/10	0.9957 [0.9957, 0.9958]	0.9491 [0.9418, 0.9563]	0.5988 [0.5965, 0.6011]	0.7343 [0.7319, 0.7368]	0.9431 [0.9414, 0.9447]	0.7151 [0.7136, 0.7166]
LMF (VAE)	50/50	0.9959 [0.9958, 0.9960]	0.9172 [0.9013, 0.9330]	0.6452 [0.6408, 0.6495]	0.7573 [0.7528, 0.7619]	0.8988 [0.8965, 0.9010]	0.6900 [0.6881, 0.6920]
	80/20	0.9958 [0.9957, 0.9960]	0.9082 [0.8879, 0.9284]	0.6452 [0.6374, 0.6531]	0.7541 [0.7472, 0.7611]	0.9017 [0.8999, 0.9035]	0.6869 [0.6821, 0.6916]
	85/15	0.9960 [0.9959, 0.9960]	0.9222 [0.9113, 0.9332]	0.6480 [0.6429, 0.6532]	0.7611 [0.7571, 0.7651]	0.9024 [0.9007, 0.9040]	0.6914 [0.6887, 0.6940]
	90/10	0.9959 [0.9958, 0.9960]	0.9193 [0.9062, 0.9324]	0.6462 [0.6427, 0.6498]	0.7589 [0.7547, 0.7631]	0.9032 [0.9009, 0.9054]	0.6918 [0.6894, 0.6941]
Diffusion Model	50/50	0.9961 [0.9960, 0.9962]	0.8851 [0.8768, 0.8935]	0.6951 [0.6932, 0.6971]	0.7787 [0.7756, 0.7818]	0.9437 [0.9415, 0.9458]	0.7384 [0.7360, 0.7408]
	80/20	0.9961 [0.9960, 0.9962]	0.8913 [0.8813, 0.9014]	0.6921 [0.6867, 0.6975]	0.7791 [0.7742, 0.7840]	0.9461 [0.9434, 0.9487]	0.7392 [0.7336, 0.7449]
	85/15	0.9961 [0.9959, 0.9962]	0.8822 [0.8675, 0.8968]	0.6947 [0.6909, 0.6986]	0.7772 [0.7724, 0.7819]	0.9463 [0.9442, 0.9483]	0.7400 [0.7374, 0.7426]
	90/10	0.9961 [0.9960, 0.9962]	0.8888 [0.8791, 0.8985]	0.6949 [0.6927, 0.6971]	0.7800 [0.7764, 0.7835]	0.9454 [0.9433, 0.9475]	0.7410 [0.7378, 0.7442]
ADASYN	50/50	0.9954 [0.9951, 0.9956]	0.8041 [0.7838, 0.8245]	0.7042 [0.7006, 0.7078]	0.7506 [0.7422, 0.7590]	0.9403 [0.9382, 0.9424]	0.6964 [0.6869, 0.7058]
	80/20	0.9955 [0.9952, 0.9959]	0.8253 [0.7930, 0.8577]	0.6982 [0.6944, 0.7019]	0.7558 [0.7423, 0.7693]	0.9391 [0.9364, 0.9418]	0.7124 [0.7049, 0.7199]
	85/15	0.9958 [0.9957, 0.9959]	0.8548 [0.8416, 0.8679]	0.6959 [0.6922, 0.6997]	0.7671 [0.7617, 0.7725]	0.9396 [0.9375, 0.9417]	0.7238 [0.7177, 0.7299]
	90/10	0.9959 [0.9958, 0.9960]	0.8677 [0.8567, 0.8787]	0.6952 [0.6900, 0.7005]	0.7719 [0.7674, 0.7764]	0.9384 [0.9356, 0.9411]	0.7259 [0.7227, 0.7292]
Latent Sampling	50/50	0.9960 [0.9958, 0.9962]	0.8798 [0.8602, 0.8993]	0.6922 [0.6859, 0.6985]	0.7746 [0.7666, 0.7826]	0.9348 [0.9304, 0.9392]	0.7346 [0.7300, 0.7392]
	80/20	0.9961 [0.9960, 0.9962]	0.8849 [0.8744, 0.8954]	0.6963 [0.6928, 0.6998]	0.7793 [0.7769, 0.7816]	0.9405 [0.9377, 0.9433]	0.7389 [0.7354, 0.7425]
	85/15	0.9961 [0.9959, 0.9962]	0.8869 [0.8709, 0.9030]	0.6922 [0.6895, 0.6950]	0.7775 [0.7715, 0.7834]	0.9408 [0.9379, 0.9438]	0.7394 [0.7349, 0.7439]
	90/10	0.9960 [0.9959, 0.9962]	0.8800 [0.8664, 0.8936]	0.6963 [0.6917, 0.7009]	0.7773 [0.7717, 0.7829]	0.9408 [0.9378, 0.9438]	0.7416 [0.7391, 0.7441]
Noise Injection	50/50	0.9952 [0.9951, 0.9954]	0.7835 [0.7715, 0.7956]	0.7158 [0.7137, 0.7178]	0.7480 [0.7432, 0.7528]	0.9419 [0.9398, 0.9440]	0.7383 [0.7355, 0.7410]
	80/20	0.9954 [0.9952, 0.9955]	0.8024 [0.7885, 0.8163]	0.7093 [0.7069, 0.7116]	0.7528 [0.7467, 0.7590]	0.9431 [0.9416, 0.9445]	0.7356 [0.7306, 0.7406]
	85/15	0.9955 [0.9954, 0.9957]	0.8142 [0.8005, 0.8279]	0.7101 [0.7085, 0.7116]	0.7585 [0.7529, 0.7641]	0.9443 [0.9426, 0.9460]	0.7401 [0.7361, 0.7441]
	90/10	0.9956 [0.9955, 0.9957]	0.8243 [0.8114, 0.8373]	0.7078 [0.7059, 0.7098]	0.7615 [0.7565, 0.7666]	0.9431 [0.9407, 0.9456]	0.7371 [0.7322, 0.7421]
Random Over-Sampling	50/50	0.9954 [0.9953, 0.9956]	0.8057 [0.7923, 0.8192]	0.7118 [0.7101, 0.7134]	0.7557 [0.7502, 0.7613]	0.9401 [0.9374, 0.9429]	0.7283 [0.7223, 0.7343]
	80/20	0.9955 [0.9954, 0.9957]	0.8175 [0.8045, 0.8305]	0.7080 [0.7056, 0.7104]	0.7587 [0.7536, 0.7639]	0.9433 [0.9409, 0.9457]	0.7287 [0.7265, 0.7309]
	85/15	0.9956 [0.9955, 0.9957]	0.8223 [0.8096, 0.8349]	0.7080 [0.7064, 0.7095]	0.7608 [0.7550, 0.7665]	0.9449 [0.9423, 0.9475]	0.7333 [0.7291, 0.7375]
	90/10	0.9957 [0.9955, 0.9958]	0.8309 [0.8171, 0.8447]	0.7068 [0.7055, 0.7080]	0.7637 [0.7580, 0.7694]	0.9428 [0.9399, 0.9456]	0.7313 [0.7248, 0.7378]
SMOTE	50/50	0.9957 [0.9955, 0.9958]	0.8274 [0.8160, 0.8389]	0.7101 [0.7081, 0.7121]	0.7642 [0.7595, 0.7689]	0.9408 [0.9379, 0.9437]	0.7289 [0.7253, 0.7324]
	80/20	0.9956 [0.9954, 0.9958]	0.8239 [0.8068, 0.8411]	0.7073 [0.7059, 0.7087]	0.7610 [0.7537, 0.7683]	0.9438 [0.9416, 0.9460]	0.7316 [0.7279, 0.7353]
	85/15	0.9958 [0.9957, 0.9959]	0.8442 [0.8353, 0.8530]	0.7073 [0.7059, 0.7087]	0.7696 [0.7666, 0.7727]	0.9435 [0.9418, 0.9452]	0.7335 [0.7310, 0.7361]
	90/10	0.9959 [0.9958, 0.9960]	0.8518 [0.8420, 0.8615]	0.7062 [0.7045, 0.7078]	0.7721 [0.7684, 0.7758]	0.9443 [0.9414, 0.9472]	0.7357 [0.7309, 0.7406]
Basic SMOTE	50/50	0.9955 [0.9955, 0.9956]	0.8169 [0.8098, 0.8240]	0.7090 [0.7076, 0.7103]	0.7591 [0.7561, 0.7621]	0.9415 [0.9376, 0.9455]	0.7313 [0.7278, 0.7348]
	80/20	0.9957 [0.9955, 0.9958]	0.8307 [0.8183, 0.8432]	0.7075 [0.7060, 0.7090]	0.7641 [0.7588, 0.7694]	0.9433 [0.9411, 0.9455]	0.7306 [0.7257, 0.7355]
	85/15	0.9957 [0.9956, 0.9958]	0.8350 [0.8219, 0.8482]	0.7072 [0.7059, 0.7085]	0.7657 [0.7603, 0.7712]	0.9422 [0.9385, 0.9459]	0.7324 [0.7275, 0.7372]
	90/10	0.9958 [0.9957, 0.9959]	0.8443 [0.8316, 0.8569]	0.7058 [0.7039, 0.7077]	0.7688 [0.7638, 0.7737]	0.9429 [0.9414, 0.9444]	0.7283 [0.7230, 0.7336]
k-Means SMOTE	50/50	0.9961 [0.9960, 0.9961]	0.8811 [0.8721, 0.8901]	0.6963 [0.6933, 0.6993]	0.7778 [0.7741, 0.7816]	0.9440 [0.9416, 0.9464]	0.7376 [0.7329, 0.7423]
	80/20	0.9961 [0.9961, 0.9962]	0.8884 [0.8808, 0.8960]	0.6944 [0.6892, 0.6997]	0.7795 [0.7767, 0.7822]	0.9430 [0.9408, 0.9452]	0.7371 [0.7322, 0.7419]
	85/15	0.9961 [0.9960, 0.9961]	0.8788 [0.8700, 0.8875]	0.6976 [0.6937, 0.7015]	0.7777 [0.7733, 0.7821]	0.9426 [0.9397, 0.9454]	0.7368 [0.7333, 0.7403]
	90/10	0.9961 [0.9961, 0.9962]	0.8853 [0.8764, 0.8941]	0.7011 [0.6995, 0.7026]	0.7824 [0.7796, 0.7853]	0.9444 [0.9422, 0.9467]	0.7372 [0.7342, 0.7402]
SMOTE + Tomek Links	50/50	0.9955 [0.9954, 0.9957]	0.8173 [0.8068, 0.8278]	0.7090 [0.7076, 0.7104]	0.7592 [0.7549, 0.7636]	0.9396 [0.9373, 0.9418]	0.7310 [0.7280, 0.7340]
	80/20	0.9956 [0.9951, 0.9960]	0.8263 [0.7877, 0.8649]	0.7058 [0.7040, 0.7076]	0.7605 [0.7433, 0.7778]	0.9441 [0.9424, 0.9458]	0.7326 [0.7300, 0.7353]
	85/15	0.9957 [0.9956, 0.9958]	0.8296 [0.8184, 0.8408]	0.7075 [0.7058, 0.7091]	0.7636 [0.7592, 0.7680]	0.9464 [0.9446, 0.9481]	0.7348 [0.7315, 0.7381]
	90/10	0.9958 [0.9957, 0.9959]	0.8496 [0.8390, 0.8603]	0.7049 [0.7037, 0.7061]	0.7705 [0.7663, 0.7746]	0.9437 [0.9418, 0.9456]	0.7326 [0.7299, 0.7352]
CNN + Tomek Links	50/50	0.9208 [0.9130, 0.9286]	0.0883 [0.0809, 0.0957]	0.7376 [0.7263, 0.7489]	0.1575 [0.1459, 0.1691]	0.8763 [0.8731, 0.8794]	0.5432 [0.5109, 0.5754]
	80/20	0.9561 [0.9533, 0.9590]	0.1408 [0.1328, 0.1488]	0.6662 [0.6617, 0.6706]	0.2322 [0.2214, 0.2429]	0.8370 [0.8355, 0.8385]	0.5322 [0.5144, 0.5500]
	85/15	0.9602 [0.9561, 0.9642]	0.1510 [0.1385, 0.1635]	0.6414 [0.6305, 0.6522]	0.2439 [0.2274, 0.2605]	0.8101 [0.8086, 0.8116]	0.4629 [0.4515, 0.4743]
	90/10	0.9715 [0.9686, 0.9743]	0.1965 [0.1772, 0.2157]	0.5924 [0.5781, 0.6068]	0.2938 [0.2744, 0.3133]	0.7768 [0.7736, 0.7799]	0.4236 [0.3959, 0.4513]
Random Under-Sampling	50/50	0.9753 [0.9730, 0.9776]	0.2598 [0.2411, 0.2784]	0.7918 [0.7849, 0.7988]	0.3904 [0.3700, 0.4107]	0.9563 [0.9544, 0.9582]	0.6854 [0.6632, 0.7077]
	80/20	0.9924 [0.9917, 0.9932]	0.6008 [0.5638, 0.6378]	0.7334 [0.7272, 0.7397]	0.6592 [0.6388, 0.6797]	0.9483 [0.9463, 0.9503]	0.7347 [0.7302, 0.7392]
	85/15	0.9935 [0.9931, 0.9939]	0.6574 [0.6332, 0.6815]	0.7259 [0.7225, 0.7294]	0.6895 [0.6774, 0.7015]	0.9474 [0.9450, 0.9497]	0.7323 [0.7245, 0.7401]
	90/10	0.9943 [0.9939, 0.9946]	0.7086 [0.6860, 0.7313]	0.7211 [0.7194, 0.7227]	0.7144 [0.7030, 0.7258]	0.9465 [0.9444, 0.9485]	0.7269 [0.7107, 0.7430]
Tomek Links	50/50	0.9738 [0.9711, 0.9766]	0.2493 [0.2280, 0.2706]	0.7958 [0.7896, 0.8021]	0.3787 [0.3550, 0.4023]	0.9589 [0.9568, 0.9611]	0.6447 [0.6272, 0.6623]
	80/20	0.9923 [0.9916, 0.9929]	0.5912 [0.5586, 0.6237]	0.7335 [0.7299, 0.7371]	0.6537 [0.6349, 0.6726]	0.9501 [0.9482, 0.9520]	0.7388 [0.7328, 0.7449]
	85/15	0.9931 [0.9925, 0.9937]	0.6366 [0.6014, 0.6718]	0.7289 [0.7246, 0.7332]	0.6786 [0.6598, 0.6974]	0.9483 [0.9467, 0.9500]	0.7325 [0.7241, 0.7409]
	90/10	0.9943 [0.9940, 0.9946]	0.7103 [0.6919, 0.7287]	0.7180 [0.7145, 0.7216]	0.7139 [0.7046, 0.7233]	0.9476 [0.9455, 0.9496]	0.7356 [0.7306, 0.7405]

Table 6 Performance comparison of augmentation and resampling configurations for the GPCR family. Values are reported as mean [95% CI] over 10 runs.

Method	Ratio	Acc.	Prec.	Rec.	F1	ROC-AUC	PR-AUC
MSCMF (No DA)	No aug.	0.9787 [0.9784, 0.9790]	0.7285 [0.7161, 0.7410]	0.4627 [0.4497, 0.4756]	0.5656 [0.5565, 0.5747]	0.8864 [0.8811, 0.8918]	0.5517 [0.5475, 0.5558]
MSCMF (WGAN-GP)	50/50	0.9778 [0.9773, 0.9783]	0.7032 [0.6865, 0.7198]	0.4494 [0.4361, 0.4628]	0.5480 [0.5373, 0.5588]	0.8796 [0.8745, 0.8846]	0.5372 [0.5271, 0.5472]
	80/20	0.9779 [0.9772, 0.9786]	0.7009 [0.6770, 0.7249]	0.4620 [0.4496, 0.4745]	0.5563 [0.5470, 0.5656]	0.8822 [0.8775, 0.8869]	0.5472 [0.5366, 0.5579]
	85/15	0.9783 [0.9778, 0.9787]	0.7192 [0.7026, 0.7358]	0.4517 [0.4437, 0.4596]	0.5546 [0.5485, 0.5607]	0.8849 [0.8807, 0.8891]	0.5476 [0.5419, 0.5533]
	90/10	0.9785 [0.9782, 0.9787]	0.7242 [0.7126, 0.7357]	0.4565 [0.4441, 0.4690]	0.5597 [0.5522, 0.5671]	0.8845 [0.8792, 0.8897]	0.5537 [0.5451, 0.5624]
MSCMF (VAE)	50/50	0.9780 [0.9775, 0.9786]	0.7038 [0.6819, 0.7256]	0.4644 [0.4474, 0.4814]	0.5587 [0.5482, 0.5693]	0.8828 [0.8777, 0.8879]	0.5523 [0.5439, 0.5607]
	80/20	0.9786 [0.9781, 0.9791]	0.7311 [0.7166, 0.7455]	0.4531 [0.4344, 0.4718]	0.5589 [0.5443, 0.5734]	0.8878 [0.8851, 0.8906]	0.5606 [0.5493, 0.5720]
	85/15	0.9785 [0.9781, 0.9790]	0.7180 [0.7036, 0.7325]	0.4690 [0.4567, 0.4813]	0.5671 [0.5576, 0.5765]	0.8879 [0.8832, 0.8927]	0.5600 [0.5516, 0.5683]
	90/10	0.9782 [0.9778, 0.9785]	0.7168 [0.7023, 0.7313]	0.4509 [0.4326, 0.4691]	0.5529 [0.5407, 0.5651]	0.8842 [0.8788, 0.8895]	0.5522 [0.5405, 0.5640]
NNMF (WGAN-GP)	50/50	0.9749 [0.9743, 0.9756]	0.6370 [0.6155, 0.6584]	0.3839 [0.3723, 0.3956]	0.4786 [0.4681, 0.4892]	0.8364 [0.8330, 0.8397]	0.4224 [0.4131, 0.4317]
	80/20	0.9748 [0.9744, 0.9753]	0.6405 [0.6263, 0.6547]	0.3672 [0.3543, 0.3802]	0.4665 [0.4553, 0.4777]	0.8343 [0.8308, 0.8378]	0.4099 [0.3999, 0.4200]
	85/15	0.9751 [0.9746, 0.9756]	0.6445 [0.6267, 0.6623]	0.3778 [0.3699, 0.3857]	0.4761 [0.4687, 0.4834]	0.8332 [0.8294, 0.8371]	0.4152 [0.4079, 0.4225]
	90/10	0.9746 [0.9742, 0.9750]	0.6288 [0.6120, 0.6456]	0.3748 [0.3614, 0.3882]	0.4690 [0.4609, 0.4771]	0.8364 [0.8312, 0.8415]	0.4093 [0.4009, 0.4177]
LMF (WGAN-GP)	50/50	0.9762 [0.9758, 0.9765]	0.7071 [0.6871, 0.7270]	0.3520 [0.3364, 0.3676]	0.4691 [0.4583, 0.4800]	0.7910 [0.7884, 0.7937]	0.4330 [0.4236, 0.4424]
	80/20	0.9765 [0.9759, 0.9771]	0.7031 [0.6696, 0.7366]	0.3791 [0.3629, 0.3953]	0.4913 [0.4799, 0.5028]	0.7945 [0.7912, 0.7977]	0.4480 [0.4363, 0.4598]
	85/15	0.9760 [0.9756, 0.9765]	0.6855 [0.6618, 0.7092]	0.3753 [0.3522, 0.3984]	0.4835 [0.4664, 0.5006]	0.7938 [0.7916, 0.7961]	0.4400 [0.4298, 0.4503]
	90/10	0.9766 [0.9762, 0.9771]	0.7094 [0.6913, 0.7274]	0.3756 [0.3638, 0.3874]	0.4908 [0.4806, 0.5009]	0.7919 [0.7886, 0.7952]	0.4489 [0.4399, 0.4578]
NNMF (VAE)	50/50	0.9752 [0.9745, 0.9758]	0.6478 [0.6264, 0.6692]	0.3784 [0.3650, 0.3918]	0.4772 [0.4661, 0.4883]	0.8359 [0.8316, 0.8403]	0.4219 [0.4171, 0.4267]
	80/20	0.9747 [0.9743, 0.9752]	0.6301 [0.6154, 0.6448]	0.3805 [0.3617, 0.3993]	0.4738 [0.4597, 0.4878]	0.8357 [0.8307, 0.8408]	0.4173 [0.4067, 0.4278]
	85/15	0.9747 [0.9740, 0.9753]	0.6276 [0.6042, 0.6510]	0.3861 [0.3726, 0.3997]	0.4774 [0.4679, 0.4868]	0.8375 [0.8342, 0.8407]	0.4223 [0.4154, 0.4293]
	90/10	0.9747 [0.9743, 0.9750]	0.6271 [0.6170, 0.6371]	0.3833 [0.3710, 0.3956]	0.4755 [0.4655, 0.4856]	0.8360 [0.8321, 0.8400]	0.4222 [0.4125, 0.4320]
LMF (VAE)	50/50	0.9760 [0.9754, 0.9765]	0.6809 [0.6518, 0.7100]	0.3797 [0.3570, 0.4024]	0.4857 [0.4716, 0.4998]	0.7923 [0.7884, 0.7963]	0.4417 [0.4283, 0.4552]
	80/20	0.9759 [0.9751, 0.9767]	0.6829 [0.6418, 0.7241]	0.3784 [0.3570, 0.3998]	0.4846 [0.4737, 0.4956]	0.7911 [0.7880, 0.7941]	0.4423 [0.4298, 0.4548]
	85/15	0.9767 [0.9761, 0.9773]	0.7136 [0.6921, 0.7352]	0.3732 [0.3594, 0.3870]	0.4897 [0.4762, 0.5032]	0.7915 [0.7885, 0.7945]	0.4488 [0.4371, 0.4605]
	90/10	0.9766 [0.9761, 0.9770]	0.7021 [0.6849, 0.7192]	0.3808 [0.3625, 0.3990]	0.4931 [0.4779, 0.5083]	0.7927 [0.7903, 0.7950]	0.4498 [0.4401, 0.4594]
Diffusion Model	50/50	0.9778 [0.9775, 0.9781]	0.7007 [0.6873, 0.7142]	0.4524 [0.4374, 0.4674]	0.5494 [0.5397, 0.5591]	0.8839 [0.8778, 0.8899]	0.5296 [0.5175, 0.5417]
	80/20	0.9776 [0.9765, 0.9787]	0.6899 [0.6551, 0.7247]	0.4630 [0.4494, 0.4766]	0.5532 [0.5386, 0.5679]	0.8799 [0.8719, 0.8880]	0.5309 [0.5098, 0.5520]
	85/15	0.9779 [0.9771, 0.9788]	0.6976 [0.6718, 0.7234]	0.4687 [0.4548, 0.4825]	0.5602 [0.5459, 0.5746]	0.8839 [0.8779, 0.8899]	0.5416 [0.5241, 0.5591]
	90/10	0.9783 [0.9778, 0.9788]	0.7226 [0.7079, 0.7372]	0.4488 [0.4318, 0.4659]	0.5534 [0.5390, 0.5677]	0.8816 [0.8746, 0.8886]	0.5426 [0.5293, 0.5559]
ADASYN	50/50	0.9775 [0.9763, 0.9787]	0.6867 [0.6516, 0.7219]	0.4658 [0.4530, 0.4787]	0.5538 [0.5432, 0.5644]	0.8893 [0.8848, 0.8939]	0.5256 [0.5153, 0.5359]
	80/20	0.9785 [0.9780, 0.9790]	0.7177 [0.7003, 0.7350]	0.4674 [0.4590, 0.4758]	0.5658 [0.5591, 0.5726]	0.8884 [0.8837, 0.8932]	0.5549 [0.5459, 0.5639]
	85/15	0.9786 [0.9781, 0.9792]	0.7216 [0.7037, 0.7395]	0.4688 [0.4569, 0.4808]	0.5681 [0.5576, 0.5785]	0.8897 [0.8868, 0.8926]	0.5505 [0.5409, 0.5602]
	90/10	0.9786 [0.9780, 0.9792]	0.7160 [0.6952, 0.7369]	0.4767 [0.4656, 0.4878]	0.5720 [0.5619, 0.5821]	0.8847 [0.8810, 0.8883]	0.5541 [0.5465, 0.5617]
Latent Sampling	50/50	0.9768 [0.9761, 0.9774]	0.6612 [0.6419, 0.6805]	0.4631 [0.4465, 0.4798]	0.5442 [0.5315, 0.5568]	0.8701 [0.8653, 0.8750]	0.5101 [0.4999, 0.5204]
	80/20	0.9780 [0.9776, 0.9784]	0.6944 [0.6813, 0.7075]	0.4780 [0.4685, 0.4874]	0.5660 [0.5587, 0.5732]	0.8780 [0.8741, 0.8820]	0.5329 [0.5220, 0.5439]
	85/15	0.9774 [0.9767, 0.9780]	0.6727 [0.6529, 0.6924]	0.4803 [0.4711, 0.4896]	0.5601 [0.5514, 0.5688]	0.8824 [0.8780, 0.8868]	0.5367 [0.5298, 0.5435]
	90/10	0.9780 [0.9777, 0.9784]	0.6882 [0.6753, 0.7011]	0.4891 [0.4752, 0.5030]	0.5714 [0.5631, 0.5798]	0.8804 [0.8751, 0.8857]	0.5368 [0.5299, 0.5438]
Noise Injection	50/50	0.9777 [0.9773, 0.9781]	0.6736 [0.6587, 0.6886]	0.4983 [0.4874, 0.5092]	0.5724 [0.5678, 0.5770]	0.8866 [0.8815, 0.8917]	0.5445 [0.5337, 0.5554]
	80/20	0.9787 [0.9783, 0.9790]	0.7081 [0.6971, 0.7191]	0.4906 [0.4795, 0.5016]	0.5793 [0.5721, 0.5865]	0.8926 [0.8880, 0.8973]	0.5600 [0.5544, 0.5656]
	85/15	0.9783 [0.9778, 0.9788]	0.7031 [0.6828, 0.7234]	0.4819 [0.4693, 0.4945]	0.5712 [0.5650, 0.5774]	0.8916 [0.8880, 0.8952]	0.5535 [0.5474, 0.5595]
	90/10	0.9782 [0.9774, 0.9790]	0.7054 [0.6799, 0.7310]	0.4699 [0.4550, 0.4848]	0.5634 [0.5506, 0.5763]	0.8910 [0.8858, 0.8962]	0.5546 [0.5394, 0.5697]
Random Over-Sampling	50/50	0.9782 [0.9778, 0.9787]	0.6947 [0.6775, 0.7118]	0.4915 [0.4819, 0.5011]	0.5753 [0.5697, 0.5809]	0.8870 [0.8812, 0.8927]	0.5503 [0.5416, 0.5590]
	80/20	0.9782 [0.9779, 0.9785]	0.6943 [0.6827, 0.7059]	0.4901 [0.4811, 0.4991]	0.5743 [0.5693, 0.5793]	0.8902 [0.8865, 0.8940]	0.5523 [0.5437, 0.5608]
	85/15	0.9789 [0.9784, 0.9793]	0.7240 [0.7065, 0.7415]	0.4792 [0.4680, 0.4904]	0.5763 [0.5693, 0.5832]	0.8901 [0.8847, 0.8955]	0.5582 [0.5496, 0.5669]
	90/10	0.9785 [0.9778, 0.9791]	0.7127 [0.6905, 0.7348]	0.4748 [0.4613, 0.4883]	0.5694 [0.5579, 0.5809]	0.8905 [0.8872, 0.8938]	0.5544 [0.5429, 0.5658]
SMOTE	50/50	0.9780 [0.9772, 0.9788]	0.6972 [0.6711, 0.7232]	0.4756 [0.4652, 0.4860]	0.5647 [0.5569, 0.5725]	0.8914 [0.8869, 0.8958]	0.5408 [0.5320, 0.5496]
	80/20	0.9783 [0.9779, 0.9787]	0.7016 [0.6853, 0.7179]	0.4816 [0.4739, 0.4892]	0.5708 [0.5674, 0.5742]	0.8944 [0.8894, 0.8995]	0.5457 [0.5394, 0.5520]
	85/15	0.9781 [0.9776, 0.9787]	0.7044 [0.6812, 0.7277]	0.4699 [0.4556, 0.4843]	0.5630 [0.5548, 0.5711]	0.8933 [0.8896, 0.8971]	0.5496 [0.5415, 0.5577]
	90/10	0.9783 [0.9779, 0.9787]	0.7108 [0.6948, 0.7268]	0.4652 [0.4500, 0.4804]	0.5618 [0.5521, 0.5716]	0.8880 [0.8838, 0.8923]	0.5404 [0.5319, 0.5489]
Basic SMOTE	50/50	0.9779 [0.9770, 0.9789]	0.6930 [0.6628, 0.7231]	0.4803 [0.4646, 0.4960]	0.5662 [0.5562, 0.5762]	0.8917 [0.8865, 0.8969]	0.5395 [0.5260, 0.5530]
	80/20	0.9777 [0.9771, 0.9782]	0.6834 [0.6625, 0.7042]	0.4767 [0.4641, 0.4892]	0.5611 [0.5516, 0.5707]	0.8901 [0.8863, 0.8939]	0.5397 [0.5271, 0.5523]
	85/15	0.9782 [0.9776, 0.9788]	0.7020 [0.6795, 0.7244]	0.4778 [0.4673, 0.4883]	0.5681 [0.5597, 0.5765]	0.8949 [0.8917, 0.8981]	0.5536 [0.5432, 0.5640]
	90/10	0.9783 [0.9777, 0.9789]	0.7050 [0.6832, 0.7268]	0.4757 [0.4657, 0.4858]	0.5676 [0.5610, 0.5741]	0.8938 [0.8901, 0.8975]	0.5496 [0.5377, 0.5615]
k-Means SMOTE	50/50	0.9779 [0.9775, 0.9784]	0.6937 [0.6789, 0.7085]	0.4731 [0.4609, 0.4853]	0.5622 [0.5534, 0.5709]	0.8926 [0.8868, 0.8984]	0.5446 [0.5335, 0.5556]
	80/20	0.9780 [0.9775, 0.9786]	0.6992 [0.6807, 0.7178]	0.4704 [0.4603, 0.4805]	0.5622 [0.5529, 0.5715]	0.8889 [0.8834, 0.8945]	0.5309 [0.5172, 0.5446]
	85/15	0.9783 [0.9779, 0.9788]	0.7099 [0.6903, 0.7295]	0.4707 [0.4573, 0.4841]	0.5654 [0.5588, 0.5720]	0.8904 [0.8870, 0.8939]	0.5425 [0.5364, 0.5485]
	90/10	0.9779 [0.9773, 0.9784]	0.6937 [0.6750, 0.7123]	0.4699 [0.4536, 0.4863]	0.5597 [0.5477, 0.5718]	0.8868 [0.8842, 0.8894]	0.5382 [0.5293, 0.5471]
SMOTE + Tomek Links	50/50	0.9782 [0.9776, 0.9788]	0.6973 [0.6752, 0.7193]	0.4844 [0.4745, 0.4943]	0.5711 [0.5654, 0.5768]	0.8910 [0.8841, 0.8979]	0.5454 [0.5342, 0.5565]
	80/20	0.9786 [0.9783, 0.9789]	0.7137 [0.7036, 0.7238]	0.4787 [0.4687, 0.4888]	0.5729 [0.5662, 0.5795]	0.8933 [0.8888, 0.8978]	0.5494 [0.5396, 0.5593]
	85/15	0.9784 [0.9782, 0.9786]	0.7088 [0.7013, 0.7163]	0.4746 [0.4615, 0.4878]	0.5683 [0.5598, 0.5768]	0.8945 [0.8905, 0.8985]	0.5522 [0.5434, 0.5609]
	90/10	0.9780 [0.9776, 0.9784]	0.6966 [0.6799, 0.7133]	0.4742 [0.4585, 0.4899]	0.5636 [0.5541, 0.5732]	0.8913 [0.8874, 0.8951]	0.5474 [0.5364, 0.5584]
CNN + Tomek Links	50/50	0.9166 [0.9108, 0.9225]	0.1867 [0.1748, 0.1985]	0.5241 [0.5163, 0.5319]	0.2748 [0.2623, 0.2873]	0.7880 [0.7815, 0.7946]	0.3977 [0.3859, 0.4096]
	80/20	0.9570 [0.9539, 0.9602]	0.2978 [0.2698, 0.3259]	0.3065 [0.2950, 0.3179]	0.3007 [0.2851, 0.3163]	0.7131 [0.7102, 0.7161]	0.2593 [0.2493, 0.2693]
	85/15	0.9629 [0.9595, 0.9664]	0.3292 [0.2681, 0.3904]	0.1882 [0.1710, 0.2054]	0.2341 [0.2197, 0.2484]	0.6566 [0.6476, 0.6656]	0.1758 [0.1599, 0.1918]
	90/10	0.9701 [0.9692, 0.9710]	0.5461 [0.4469, 0.6453]	0.0795 [0.0698, 0.0893]	0.1371 [0.1225, 0.1518]	0.6296 [0.6258, 0.6335]	0.1448 [0.1338, 0.1559]
Random Under-Sampling	50/50	0.9161 [0.9111, 0.9212]	0.2138 [0.2022, 0.2254]	0.6655 [0.6615, 0.6695]	0.3233 [0.3100, 0.3367]	0.8706 [0.8687, 0.8724]	0.4148 [0.4032, 0.4263]
	80/20	0.9699 [0.9675, 0.9723]	0.5029 [0.4712, 0.5346]	0.5899 [0.5783, 0.6015]	0.5414 [0.5258, 0.5570]	0.8820 [0.8764, 0.8875]	0.5450 [0.5324, 0.5575]
	85/15	0.9738 [0.9727, 0.9748]	0.5640 [0.5428, 0.5852]	0.5635 [0.5454, 0.5815]	0.5629 [0.5529, 0.5728]	0.8823 [0.8768, 0.8877]	0.5426 [0.5246, 0.5607]
	90/10	0.9763 [0.9757, 0.9769]	0.6235 [0.6076, 0.6394]	0.5310 [0.5199, 0.5421]	0.5732 [0.5663, 0.5800]	0.8854 [0.8810, 0.8898]	0.5472 [0.5373, 0.5571]
Tomek Links	50/50	0.9202 [0.9149, 0.9255]	0.2248 [0.2134, 0.2363]	0.6726 [0.6642, 0.6810]	0.3366 [0.3242, 0.3490]	0.8767 [0.8745, 0.8789]	0.3997 [0.3858, 0.4136]
	80/20	0.9690 [0.9670, 0.9711]	0.4898 [0.4649, 0.5148]	0.5929 [0.5796, 0.6062]	0.5353 [0.5242, 0.5463]	0.8847 [0.8809, 0.8885]	0.5459 [0.5370, 0.5549]
	85/15	0.9740 [0.9728, 0.9752]	0.5673 [0.5458, 0.5887]	0.5683 [0.5525, 0.5842]	0.5671 [0.5546, 0.5797]	0.8878 [0.8832, 0.8924]	0.5563 [0.5453, 0.5674]
	90/10	0.9772 [0.9767, 0.9777]	0.6453 [0.6320, 0.6585]	0.5329 [0.5182, 0.5476]	0.5833 [0.5743, 0.5924]	0.8859 [0.8834, 0.8885]	0.5481 [0.5371, 0.5591]

Table 7 Performance comparison of augmentation and resampling configurations for the Ion Channel family. Values are reported as mean [95% CI] over 10 runs.

Method	Ratio	Acc.	Prec.	Rec.	F1	ROC-AUC	PR-AUC
MSCMF (No DA)	No aug.	0.9900 [0.9898, 0.9902]	0.9137 [0.9072, 0.9201]	0.7843 [0.7799, 0.7887]	0.8440 [0.8406, 0.8474]	0.9718 [0.9694, 0.9742]	0.8737 [0.8708, 0.8766]
MSCMF (WGAN-GP)	50/50	0.9897 [0.9894, 0.9900]	0.9154 [0.9055, 0.9254]	0.7726 [0.7640, 0.7811]	0.8378 [0.8328, 0.8429]	0.9682 [0.9648, 0.9716]	0.8640 [0.8591, 0.8688]
	80/20	0.9899 [0.9897, 0.9902]	0.9155 [0.9081, 0.9229]	0.7801 [0.7718, 0.7885]	0.8423 [0.8383, 0.8464]	0.9716 [0.9701, 0.9731]	0.8706 [0.8663, 0.8750]
	85/15	0.9898 [0.9895, 0.9901]	0.9168 [0.9105, 0.9230]	0.7753 [0.7651, 0.7856]	0.8400 [0.8345, 0.8456]	0.9685 [0.9658, 0.9711]	0.8659 [0.8606, 0.8712]
	90/10	0.9900 [0.9897, 0.9904]	0.9160 [0.9071, 0.9249]	0.7826 [0.7737, 0.7915]	0.8440 [0.8384, 0.8495]	0.9713 [0.9698, 0.9727]	0.8703 [0.8667, 0.8740]
MSCMF (VAE)	50/50	0.9894 [0.9888, 0.9900]	0.9049 [0.8924, 0.9174]	0.7728 [0.7602, 0.7854]	0.8335 [0.8236, 0.8435]	0.9672 [0.9651, 0.9692]	0.8622 [0.8574, 0.8670]
	80/20	0.9899 [0.9896, 0.9902]	0.9104 [0.9022, 0.9186]	0.7839 [0.7797, 0.7880]	0.8424 [0.8380, 0.8468]	0.9707 [0.9687, 0.9726]	0.8702 [0.8658, 0.8746]
	85/15	0.9898 [0.9896, 0.9900]	0.9137 [0.9051, 0.9223]	0.7774 [0.7695, 0.7854]	0.8400 [0.8368, 0.8431]	0.9710 [0.9683, 0.9738]	0.8694 [0.8659, 0.8729]
	90/10	0.9901 [0.9900, 0.9902]	0.9195 [0.9159, 0.9230]	0.7797 [0.7743, 0.7852]	0.8438 [0.8417, 0.8459]	0.9712 [0.9696, 0.9728]	0.8724 [0.8696, 0.8752]
NNMF (WGAN-GP)	50/50	0.9866 [0.9863, 0.9869]	0.8580 [0.8466, 0.8693]	0.7327 [0.7219, 0.7434]	0.7902 [0.7852, 0.7951]	0.9542 [0.9528, 0.9557]	0.8095 [0.8071, 0.8119]
	80/20	0.9864 [0.9861, 0.9867]	0.8523 [0.8418, 0.8628]	0.7328 [0.7266, 0.7390]	0.7879 [0.7840, 0.7918]	0.9547 [0.9529, 0.9565]	0.8054 [0.8002, 0.8106]
	85/15	0.9865 [0.9863, 0.9867]	0.8566 [0.8472, 0.8660]	0.7308 [0.7180, 0.7436]	0.7885 [0.7835, 0.7934]	0.9557 [0.9548, 0.9567]	0.8086 [0.8045, 0.8126]
	90/10	0.9863 [0.9861, 0.9866]	0.8490 [0.8409, 0.8571]	0.7346 [0.7254, 0.7437]	0.7875 [0.7836, 0.7915]	0.9543 [0.9531, 0.9555]	0.8070 [0.8047, 0.8093]
LMF (WGAN-GP)	50/50	0.9847 [0.9842, 0.9851]	0.8488 [0.8367, 0.8609]	0.6761 [0.6639, 0.6883]	0.7524 [0.7444, 0.7605]	0.9508 [0.9493, 0.9522]	0.7850 [0.7772, 0.7929]
	80/20	0.9848 [0.9845, 0.9851]	0.8319 [0.8219, 0.8420]	0.7011 [0.6906, 0.7115]	0.7607 [0.7559, 0.7655]	0.9521 [0.9513, 0.9530]	0.7935 [0.7884, 0.7987]
	85/15	0.9851 [0.9847, 0.9856]	0.8469 [0.8338, 0.8601]	0.6946 [0.6829, 0.7063]	0.7630 [0.7554, 0.7706]	0.9527 [0.9523, 0.9532]	0.7936 [0.7883, 0.7988]
	90/10	0.9848 [0.9843, 0.9853]	0.8402 [0.8247, 0.8557]	0.6907 [0.6777, 0.7037]	0.7578 [0.7500, 0.7657]	0.9511 [0.9499, 0.9523]	0.7856 [0.7730, 0.7981]
NNMF (VAE)	50/50	0.9863 [0.9861, 0.9866]	0.8560 [0.8471, 0.8649]	0.7263 [0.7150, 0.7376]	0.7856 [0.7808, 0.7905]	0.9536 [0.9515, 0.9557]	0.8084 [0.8050, 0.8117]
	80/20	0.9864 [0.9861, 0.9866]	0.8513 [0.8417, 0.8609]	0.7323 [0.7205, 0.7441]	0.7871 [0.7819, 0.7923]	0.9555 [0.9536, 0.9575]	0.8091 [0.8051, 0.8132]
	85/15	0.9863 [0.9860, 0.9866]	0.8505 [0.8401, 0.8608]	0.7320 [0.7201, 0.7440]	0.7866 [0.7814, 0.7918]	0.9553 [0.9537, 0.9570]	0.8139 [0.8114, 0.8164]
	90/10	0.9865 [0.9863, 0.9868]	0.8597 [0.8523, 0.8672]	0.7275 [0.7231, 0.7319]	0.7881 [0.7847, 0.7914]	0.9554 [0.9542, 0.9566]	0.8123 [0.8098, 0.8148]
LMF (VAE)	50/50	0.9845 [0.9840, 0.9850]	0.8233 [0.8021, 0.8445]	0.7026 [0.6889, 0.7164]	0.7576 [0.7514, 0.7638]	0.9501 [0.9489, 0.9514]	0.7877 [0.7804, 0.7951]
	80/20	0.9850 [0.9845, 0.9854]	0.8452 [0.8340, 0.8564]	0.6909 [0.6810, 0.7008]	0.7602 [0.7532, 0.7671]	0.9500 [0.9487, 0.9514]	0.7879 [0.7791, 0.7967]
	85/15	0.9849 [0.9844, 0.9854]	0.8406 [0.8233, 0.8579]	0.6931 [0.6822, 0.7040]	0.7594 [0.7529, 0.7659]	0.9507 [0.9501, 0.9513]	0.7862 [0.7794, 0.7930]
	90/10	0.9847 [0.9843, 0.9851]	0.8374 [0.8262, 0.8487]	0.6892 [0.6774, 0.7009]	0.7559 [0.7489, 0.7629]	0.9503 [0.9486, 0.9520]	0.7872 [0.7786, 0.7958]
Diffusion Model	50/50	0.9893 [0.9890, 0.9896]	0.9018 [0.8909, 0.9128]	0.7740 [0.7618, 0.7862]	0.8328 [0.8276, 0.8379]	0.9683 [0.9661, 0.9704]	0.8629 [0.8587, 0.8671]
	80/20	0.9896 [0.9892, 0.9900]	0.9057 [0.8939, 0.9176]	0.7806 [0.7664, 0.7947]	0.8382 [0.8314, 0.8450]	0.9697 [0.9683, 0.9712]	0.8655 [0.8600, 0.8711]
	85/15	0.9898 [0.9895, 0.9901]	0.9098 [0.9014, 0.9183]	0.7815 [0.7765, 0.7865]	0.8408 [0.8359, 0.8456]	0.9684 [0.9659, 0.9709]	0.8680 [0.8634, 0.8727]
	90/10	0.9897 [0.9892, 0.9902]	0.9043 [0.8909, 0.9177]	0.7848 [0.7782, 0.7913]	0.8402 [0.8328, 0.8476]	0.9686 [0.9654, 0.9719]	0.8656 [0.8598, 0.8715]
ADASYN	50/50	0.9899 [0.9894, 0.9903]	0.8993 [0.8866, 0.9119]	0.7951 [0.7887, 0.8014]	0.8438 [0.8378, 0.8499]	0.9680 [0.9654, 0.9705]	0.8563 [0.8499, 0.8626]
	80/20	0.9899 [0.9896, 0.9901]	0.9163 [0.9092, 0.9234]	0.7766 [0.7690, 0.7841]	0.8406 [0.8360, 0.8451]	0.9708 [0.9692, 0.9724]	0.8670 [0.8640, 0.8700]
	85/15	0.9900 [0.9896, 0.9903]	0.9151 [0.9093, 0.9209]	0.7810 [0.7709, 0.7910]	0.8426 [0.8365, 0.8487]	0.9703 [0.9680, 0.9726]	0.8675 [0.8616, 0.8733]
	90/10	0.9900 [0.9897, 0.9902]	0.9090 [0.9002, 0.9179]	0.7888 [0.7845, 0.7931]	0.8446 [0.8412, 0.8479]	0.9714 [0.9697, 0.9732]	0.8689 [0.8629, 0.8750]
Latent Sampling	50/50	0.9895 [0.9891, 0.9898]	0.8966 [0.8847, 0.9086]	0.7848 [0.7810, 0.7887]	0.8369 [0.8323, 0.8416]	0.9633 [0.9604, 0.9661]	0.8624 [0.8577, 0.8671]
	80/20	0.9898 [0.9896, 0.9900]	0.9003 [0.8952, 0.9054]	0.7907 [0.7848, 0.7965]	0.8419 [0.8387, 0.8451]	0.9659 [0.9637, 0.9682]	0.8674 [0.8642, 0.8707]
	85/15	0.9898 [0.9895, 0.9901]	0.9017 [0.8904, 0.9130]	0.7897 [0.7795, 0.7999]	0.8418 [0.8371, 0.8464]	0.9672 [0.9652, 0.9692]	0.8678 [0.8650, 0.8706]
	90/10	0.9896 [0.9893, 0.9900]	0.9000 [0.8917, 0.9082]	0.7872 [0.7751, 0.7992]	0.8396 [0.8337, 0.8456]	0.9671 [0.9650, 0.9692]	0.8679 [0.8638, 0.8721]
Noise Injection	50/50	0.9898 [0.9893, 0.9902]	0.8819 [0.8677, 0.8962]	0.8126 [0.8083, 0.8169]	0.8457 [0.8404, 0.8510]	0.9719 [0.9704, 0.9734]	0.8720 [0.8674, 0.8766]
	80/20	0.9903 [0.9901, 0.9906]	0.9034 [0.8952, 0.9117]	0.8053 [0.7994, 0.8111]	0.8515 [0.8480, 0.8550]	0.9716 [0.9697, 0.9735]	0.8759 [0.8731, 0.8788]
	85/15	0.9900 [0.9897, 0.9903]	0.8997 [0.8905, 0.9090]	0.7982 [0.7931, 0.8033]	0.8458 [0.8420, 0.8497]	0.9714 [0.9693, 0.9735]	0.8718 [0.8673, 0.8763]
	90/10	0.9903 [0.9900, 0.9905]	0.9076 [0.8992, 0.9160]	0.7993 [0.7963, 0.8022]	0.8500 [0.8466, 0.8533]	0.9715 [0.9697, 0.9732]	0.8730 [0.8682, 0.8778]
Random Over-Sampling	50/50	0.9903 [0.9900, 0.9906]	0.9008 [0.8930, 0.9085]	0.8085 [0.8049, 0.8122]	0.8521 [0.8477, 0.8565]	0.9716 [0.9695, 0.9737]	0.8734 [0.8693, 0.8774]
	80/20	0.9905 [0.9903, 0.9907]	0.9148 [0.9076, 0.9219]	0.7987 [0.7918, 0.8056]	0.8527 [0.8497, 0.8558]	0.9737 [0.9720, 0.9754]	0.8755 [0.8718, 0.8793]
	85/15	0.9903 [0.9900, 0.9906]	0.9074 [0.8983, 0.9165]	0.7995 [0.7948, 0.8043]	0.8500 [0.8461, 0.8539]	0.9717 [0.9689, 0.9745]	0.8732 [0.8676, 0.8788]
	90/10	0.9901 [0.9897, 0.9905]	0.9101 [0.9041, 0.9160]	0.7915 [0.7789, 0.8040]	0.8465 [0.8396, 0.8534]	0.9719 [0.9696, 0.9741]	0.8712 [0.8657, 0.8767]
SMOTE	50/50	0.9900 [0.9897, 0.9904]	0.9027 [0.8914, 0.9139]	0.7970 [0.7917, 0.8022]	0.8464 [0.8419, 0.8509]	0.9692 [0.9661, 0.9723]	0.8704 [0.8644, 0.8764]
	80/20	0.9899 [0.9893, 0.9905]	0.9016 [0.8838, 0.9195]	0.7938 [0.7898, 0.7977]	0.8441 [0.8362, 0.8520]	0.9722 [0.9703, 0.9740]	0.8715 [0.8654, 0.8776]
	85/15	0.9902 [0.9899, 0.9905]	0.9113 [0.9038, 0.9188]	0.7928 [0.7857, 0.7999]	0.8479 [0.8432, 0.8526]	0.9700 [0.9669, 0.9732]	0.8717 [0.8681, 0.8753]
	90/10	0.9902 [0.9900, 0.9904]	0.9130 [0.9051, 0.9210]	0.7907 [0.7841, 0.7972]	0.8474 [0.8440, 0.8507]	0.9714 [0.9697, 0.9730]	0.8717 [0.8678, 0.8756]
Basic SMOTE	50/50	0.9900 [0.9895, 0.9905]	0.9008 [0.8850, 0.9165]	0.7979 [0.7920, 0.8038]	0.8460 [0.8397, 0.8523]	0.9678 [0.9654, 0.9703]	0.8619 [0.8537, 0.8702]
	80/20	0.9904 [0.9902, 0.9906]	0.9197 [0.9119, 0.9276]	0.7907 [0.7837, 0.7976]	0.8502 [0.8471, 0.8533]	0.9703 [0.9679, 0.9727]	0.8724 [0.8699, 0.8749]
	85/15	0.9904 [0.9903, 0.9905]	0.9163 [0.9117, 0.9210]	0.7948 [0.7905, 0.7991]	0.8512 [0.8495, 0.8529]	0.9719 [0.9693, 0.9746]	0.8742 [0.8698, 0.8786]
	90/10	0.9899 [0.9896, 0.9901]	0.9079 [0.8994, 0.9164]	0.7854 [0.7785, 0.7924]	0.8421 [0.8385, 0.8458]	0.9708 [0.9680, 0.9735]	0.8679 [0.8639, 0.8720]
k-Means SMOTE	50/50	0.9899 [0.9894, 0.9904]	0.9016 [0.8844, 0.9188]	0.7939 [0.7868, 0.8011]	0.8441 [0.8385, 0.8497]	0.9685 [0.9654, 0.9717]	0.8598 [0.8505, 0.8690]
	80/20	0.9899 [0.9896, 0.9902]	0.9076 [0.9002, 0.9151]	0.7883 [0.7805, 0.7961]	0.8437 [0.8389, 0.8485]	0.9714 [0.9695, 0.9732]	0.8657 [0.8616, 0.8697]
	85/15	0.9900 [0.9895, 0.9904]	0.9089 [0.8977, 0.9200]	0.7886 [0.7842, 0.7930]	0.8444 [0.8383, 0.8505]	0.9716 [0.9692, 0.9739]	0.8681 [0.8633, 0.8728]
	90/10	0.9901 [0.9899, 0.9903]	0.9108 [0.9003, 0.9212]	0.7905 [0.7825, 0.7985]	0.8462 [0.8431, 0.8494]	0.9715 [0.9689, 0.9741]	0.8686 [0.8650, 0.8722]
SMOTE + Tomek Links	50/50	0.9901 [0.9897, 0.9904]	0.9022 [0.8918, 0.9126]	0.7993 [0.7923, 0.8064]	0.8475 [0.8426, 0.8525]	0.9718 [0.9705, 0.9731]	0.8718 [0.8688, 0.8749]
	80/20	0.9896 [0.9890, 0.9902]	0.8949 [0.8761, 0.9137]	0.7920 [0.7840, 0.8000]	0.8400 [0.8326, 0.8474]	0.9717 [0.9703, 0.9731]	0.8690 [0.8646, 0.8735]
	85/15	0.9903 [0.9900, 0.9906]	0.9116 [0.9027, 0.9205]	0.7968 [0.7910, 0.8026]	0.8503 [0.8462, 0.8543]	0.9730 [0.9711, 0.9750]	0.8768 [0.8730, 0.8805]
	90/10	0.9902 [0.9899, 0.9905]	0.9112 [0.8985, 0.9239]	0.7930 [0.7868, 0.7991]	0.8478 [0.8439, 0.8518]	0.9732 [0.9719, 0.9745]	0.8743 [0.8704, 0.8782]
CNN + Tomek Links	50/50	0.9416 [0.9349, 0.9483]	0.3466 [0.3210, 0.3722]	0.7555 [0.7481, 0.7629]	0.4740 [0.4496, 0.4983]	0.8962 [0.8935, 0.8988]	0.7095 [0.7008, 0.7181]
	80/20	0.9634 [0.9604, 0.9663]	0.4795 [0.4472, 0.5118]	0.6124 [0.6051, 0.6197]	0.5365 [0.5182, 0.5549]	0.8011 [0.7991, 0.8031]	0.5875 [0.5774, 0.5976]
	85/15	0.9651 [0.9630, 0.9672]	0.4966 [0.4676, 0.5255]	0.5113 [0.4962, 0.5264]	0.5032 [0.4840, 0.5224]	0.7716 [0.7676, 0.7755]	0.4890 [0.4603, 0.5176]
	90/10	0.9736 [0.9728, 0.9743]	0.7498 [0.7120, 0.7876]	0.3543 [0.3404, 0.3683]	0.4800 [0.4695, 0.4906]	0.7200 [0.7175, 0.7225]	0.4459 [0.4360, 0.4557]
Random Under-Sampling	50/50	0.9642 [0.9601, 0.9683]	0.4930 [0.4617, 0.5242]	0.8707 [0.8629, 0.8784]	0.6282 [0.6037, 0.6527]	0.9727 [0.9716, 0.9738]	0.7839 [0.7710, 0.7968]
	80/20	0.9861 [0.9856, 0.9867]	0.7837 [0.7719, 0.7954]	0.8253 [0.8205, 0.8302]	0.8039 [0.7971, 0.8107]	0.9748 [0.9735, 0.9761]	0.8584 [0.8528, 0.8640]
	85/15	0.9878 [0.9872, 0.9884]	0.8213 [0.8077, 0.8349]	0.8256 [0.8234, 0.8278]	0.8234 [0.8162, 0.8305]	0.9746 [0.9727, 0.9764]	0.8630 [0.8548, 0.8711]
	90/10	0.9888 [0.9884, 0.9892]	0.8537 [0.8396, 0.8678]	0.8159 [0.8069, 0.8248]	0.8341 [0.8290, 0.8393]	0.9738 [0.9725, 0.9752]	0.8678 [0.8622, 0.8733]
Tomek Links	50/50	0.9666 [0.9626, 0.9706]	0.5136 [0.4792, 0.5480]	0.8692 [0.8606, 0.8778]	0.6441 [0.6192, 0.6691]	0.9739 [0.9730, 0.9748]	0.8092 [0.7943, 0.8241]
	80/20	0.9857 [0.9846, 0.9868]	0.7714 [0.7468, 0.7961]	0.8341 [0.8275, 0.8408]	0.8011 [0.7893, 0.8129]	0.9753 [0.9739, 0.9767]	0.8579 [0.8514, 0.8645]
	85/15	0.9882 [0.9877, 0.9887]	0.8353 [0.8220, 0.8486]	0.8194 [0.8134, 0.8255]	0.8271 [0.8213, 0.8330]	0.9745 [0.9728, 0.9761]	0.8679 [0.8638, 0.8720]
	90/10	0.9889 [0.9884, 0.9894]	0.8581 [0.8446, 0.8717]	0.8117 [0.8045, 0.8188]	0.8341 [0.8280, 0.8402]	0.9739 [0.9722, 0.9756]	0.8671 [0.8617, 0.8726]

Table 8 Performance comparison of augmentation and resampling configurations for the Nuclear Receptor family. Values are reported as mean [95% CI] over 10 runs.

Method	Ratio	Acc.	Prec.	Rec.	F1	ROC-AUC	PR-AUC
MSCMF (No DA)	No aug.	0.9404 [0.9386, 0.9422]	0.5805 [0.5473, 0.6137]	0.2578 [0.2480, 0.2675]	0.3568 [0.3424, 0.3711]	0.7898 [0.7798, 0.7998]	0.3719 [0.3657, 0.3780]
MSCMF (WGAN-GP)	50/50	0.9390 [0.9371, 0.9410]	0.5556 [0.5227, 0.5886]	0.2567 [0.2508, 0.2625]	0.3508 [0.3399, 0.3617]	0.7985 [0.7908, 0.8062]	0.3724 [0.3598, 0.3849]
	80/20	0.9400 [0.9382, 0.9419]	0.5737 [0.5387, 0.6088]	0.2611 [0.2534, 0.2688]	0.3584 [0.3473, 0.3694]	0.8006 [0.7917, 0.8095]	0.3793 [0.3713, 0.3873]
	85/15	0.9402 [0.9388, 0.9417]	0.5757 [0.5487, 0.6028]	0.2656 [0.2547, 0.2764]	0.3628 [0.3517, 0.3740]	0.7930 [0.7845, 0.8015]	0.3712 [0.3618, 0.3806]
	90/10	0.9397 [0.9387, 0.9407]	0.5657 [0.5474, 0.5840]	0.2611 [0.2465, 0.2757]	0.3568 [0.3422, 0.3714]	0.7998 [0.7940, 0.8056]	0.3708 [0.3630, 0.3787]
MSCMF (VAE)	50/50	0.9343 [0.9317, 0.9370]	0.4817 [0.4441, 0.5193]	0.2656 [0.2474, 0.2837]	0.3413 [0.3216, 0.3611]	0.7773 [0.7679, 0.7868]	0.3123 [0.2948, 0.3298]
	80/20	0.9361 [0.9350, 0.9372]	0.5034 [0.4877, 0.5191]	0.2711 [0.2611, 0.2812]	0.3523 [0.3412, 0.3633]	0.7829 [0.7756, 0.7902]	0.3366 [0.3260, 0.3472]
	85/15	0.9346 [0.9330, 0.9363]	0.4845 [0.4640, 0.5050]	0.2667 [0.2537, 0.2796]	0.3431 [0.3340, 0.3523]	0.7739 [0.7623, 0.7854]	0.3226 [0.3097, 0.3356]
	90/10	0.9361 [0.9346, 0.9377]	0.5057 [0.4831, 0.5283]	0.2711 [0.2570, 0.2852]	0.3521 [0.3405, 0.3636]	0.7679 [0.7582, 0.7776]	0.3335 [0.3185, 0.3484]
NNMF (WGAN-GP)	50/50	0.9251 [0.9218, 0.9284]	0.3931 [0.3659, 0.4203]	0.3011 [0.2890, 0.3132]	0.3404 [0.3255, 0.3552]	0.7773 [0.7733, 0.7812]	0.2666 [0.2564, 0.2767]
	80/20	0.9242 [0.9218, 0.9267]	0.3824 [0.3632, 0.4015]	0.2911 [0.2762, 0.3060]	0.3299 [0.3175, 0.3423]	0.7738 [0.7665, 0.7812]	0.2589 [0.2509, 0.2670]
	85/15	0.9249 [0.9210, 0.9287]	0.3887 [0.3604, 0.4169]	0.2878 [0.2776, 0.2980]	0.3298 [0.3166, 0.3429]	0.7720 [0.7632, 0.7808]	0.2580 [0.2465, 0.2695]
	90/10	0.9250 [0.9222, 0.9278]	0.3868 [0.3666, 0.4070]	0.2833 [0.2697, 0.2970]	0.3263 [0.3147, 0.3379]	0.7826 [0.7768, 0.7885]	0.2668 [0.2588, 0.2748]
LMF (WGAN-GP)	50/50	0.9286 [0.9230, 0.9342]	0.4062 [0.3261, 0.4863]	0.1778 [0.1492, 0.2063]	0.2407 [0.2139, 0.2674]	0.6829 [0.6763, 0.6895]	0.2229 [0.1933, 0.2525]
	80/20	0.9263 [0.9204, 0.9322]	0.3829 [0.3176, 0.4482]	0.2167 [0.1873, 0.2460]	0.2740 [0.2384, 0.3095]	0.6805 [0.6711, 0.6900]	0.2251 [0.1964, 0.2538]
	85/15	0.9273 [0.9245, 0.9301]	0.3743 [0.3426, 0.4060]	0.1944 [0.1658, 0.2230]	0.2535 [0.2260, 0.2811]	0.6804 [0.6735, 0.6873]	0.2304 [0.2086, 0.2522]
	90/10	0.9306 [0.9278, 0.9333]	0.4124 [0.3726, 0.4522]	0.1867 [0.1483, 0.2250]	0.2528 [0.2156, 0.2901]	0.6828 [0.6766, 0.6890]	0.2276 [0.2057, 0.2495]
NNMF (VAE)	50/50	0.9213 [0.9178, 0.9248]	0.3644 [0.3379, 0.3908]	0.3000 [0.2824, 0.3176]	0.3285 [0.3099, 0.3471]	0.7667 [0.7603, 0.7732]	0.2405 [0.2282, 0.2528]
	80/20	0.9227 [0.9211, 0.9244]	0.3651 [0.3494, 0.3808]	0.2778 [0.2638, 0.2918]	0.3154 [0.3013, 0.3294]	0.7710 [0.7627, 0.7793]	0.2292 [0.2223, 0.2362]
	85/15	0.9212 [0.9195, 0.9229]	0.3616 [0.3478, 0.3754]	0.2978 [0.2896, 0.3060]	0.3265 [0.3168, 0.3362]	0.7633 [0.7502, 0.7763]	0.2374 [0.2283, 0.2465]
	90/10	0.9231 [0.9219, 0.9243]	0.3706 [0.3601, 0.3811]	0.2867 [0.2673, 0.3061]	0.3228 [0.3082, 0.3374]	0.7661 [0.7553, 0.7769]	0.2352 [0.2265, 0.2440]
LMF (VAE)	50/50	0.9181 [0.9124, 0.9238]	0.3002 [0.2599, 0.3405]	0.1967 [0.1653, 0.2280]	0.2337 [0.2063, 0.2612]	0.6855 [0.6797, 0.6912]	0.1633 [0.1515, 0.1751]
	80/20	0.9204 [0.9175, 0.9234]	0.3057 [0.2759, 0.3356]	0.1889 [0.1616, 0.2162]	0.2323 [0.2064, 0.2582]	0.6906 [0.6836, 0.6976]	0.1822 [0.1690, 0.1955]
	85/15	0.9189 [0.9131, 0.9248]	0.3275 [0.2971, 0.3579]	0.2367 [0.2017, 0.2716]	0.2706 [0.2450, 0.2962]	0.6916 [0.6859, 0.6972]	0.1854 [0.1753, 0.1955]
	90/10	0.9182 [0.9132, 0.9232]	0.2929 [0.2566, 0.3292]	0.1856 [0.1585, 0.2126]	0.2243 [0.1991, 0.2496]	0.6803 [0.6751, 0.6855]	0.1763 [0.1566, 0.1961]
Diffusion Model	50/50	0.9371 [0.9350, 0.9393]	0.5228 [0.4874, 0.5582]	0.2667 [0.2561, 0.2773]	0.3522 [0.3408, 0.3636]	0.8136 [0.7992, 0.8281]	0.3793 [0.3691, 0.3895]
	80/20	0.9387 [0.9369, 0.9404]	0.5476 [0.5175, 0.5777]	0.2656 [0.2577, 0.2735]	0.3571 [0.3480, 0.3662]	0.8231 [0.8178, 0.8283]	0.3818 [0.3751, 0.3885]
	85/15	0.9385 [0.9364, 0.9405]	0.5463 [0.5041, 0.5886]	0.2611 [0.2534, 0.2688]	0.3525 [0.3417, 0.3634]	0.8117 [0.8035, 0.8199]	0.3741 [0.3677, 0.3804]
	90/10	0.9390 [0.9372, 0.9409]	0.5552 [0.5229, 0.5875]	0.2678 [0.2590, 0.2765]	0.3602 [0.3551, 0.3654]	0.8100 [0.8026, 0.8173]	0.3794 [0.3735, 0.3852]
ADASYN	50/50	0.9374 [0.9356, 0.9392]	0.5254 [0.4983, 0.5525]	0.2678 [0.2583, 0.2773]	0.3542 [0.3439, 0.3645]	0.7744 [0.7669, 0.7819]	0.3490 [0.3386, 0.3594]
	80/20	0.9401 [0.9390, 0.9412]	0.5727 [0.5501, 0.5952]	0.2633 [0.2515, 0.2752]	0.3603 [0.3490, 0.3716]	0.7861 [0.7782, 0.7939]	0.3744 [0.3668, 0.3821]
	85/15	0.9404 [0.9391, 0.9416]	0.5786 [0.5539, 0.6034]	0.2633 [0.2568, 0.2699]	0.3616 [0.3543, 0.3689]	0.7940 [0.7891, 0.7988]	0.3776 [0.3732, 0.3820]
	90/10	0.9392 [0.9377, 0.9406]	0.5559 [0.5309, 0.5809]	0.2578 [0.2480, 0.2675]	0.3521 [0.3390, 0.3651]	0.7984 [0.7935, 0.8033]	0.3753 [0.3699, 0.3806]
Latent Sampling	50/50	0.9330 [0.9309, 0.9350]	0.4675 [0.4435, 0.4915]	0.3056 [0.2942, 0.3170]	0.3689 [0.3583, 0.3794]	0.7940 [0.7780, 0.8101]	0.3573 [0.3477, 0.3668]
	80/20	0.9321 [0.9308, 0.9333]	0.4536 [0.4391, 0.4682]	0.2900 [0.2758, 0.3042]	0.3534 [0.3410, 0.3658]	0.7903 [0.7824, 0.7982]	0.3577 [0.3469, 0.3685]
	85/15	0.9356 [0.9346, 0.9366]	0.4967 [0.4845, 0.5089]	0.2822 [0.2703, 0.2942]	0.3596 [0.3494, 0.3697]	0.7885 [0.7785, 0.7986]	0.3595 [0.3487, 0.3702]
	90/10	0.9364 [0.9345, 0.9383]	0.5093 [0.4825, 0.5360]	0.2856 [0.2749, 0.2962]	0.3653 [0.3541, 0.3766]	0.7942 [0.7851, 0.8032]	0.3664 [0.3600, 0.3729]
Noise Injection	50/50	0.9365 [0.9351, 0.9378]	0.5088 [0.4905, 0.5270]	0.2811 [0.2757, 0.2865]	0.3620 [0.3534, 0.3707]	0.8102 [0.8021, 0.8183]	0.3865 [0.3791, 0.3939]
	80/20	0.9392 [0.9376, 0.9408]	0.5558 [0.5278, 0.5837]	0.2689 [0.2578, 0.2800]	0.3616 [0.3517, 0.3716]	0.7998 [0.7933, 0.8063]	0.3783 [0.3703, 0.3862]
	85/15	0.9397 [0.9381, 0.9412]	0.5629 [0.5374, 0.5884]	0.2711 [0.2634, 0.2788]	0.3656 [0.3558, 0.3754]	0.7969 [0.7873, 0.8065]	0.3791 [0.3707, 0.3875]
	90/10	0.9399 [0.9389, 0.9408]	0.5657 [0.5492, 0.5822]	0.2700 [0.2581, 0.2819]	0.3652 [0.3532, 0.3773]	0.8021 [0.7940, 0.8102]	0.3802 [0.3742, 0.3862]
Random Over-Sampling	50/50	0.9368 [0.9357, 0.9378]	0.5130 [0.4965, 0.5295]	0.2789 [0.2719, 0.2858]	0.3611 [0.3534, 0.3689]	0.8115 [0.8047, 0.8184]	0.3850 [0.3808, 0.3891]
	80/20	0.9392 [0.9383, 0.9402]	0.5529 [0.5368, 0.5690]	0.2756 [0.2673, 0.2838]	0.3676 [0.3585, 0.3767]	0.8047 [0.7963, 0.8131]	0.3792 [0.3692, 0.3891]
	85/15	0.9382 [0.9373, 0.9392]	0.5380 [0.5213, 0.5547]	0.2667 [0.2592, 0.2742]	0.3563 [0.3493, 0.3633]	0.7978 [0.7900, 0.8056]	0.3805 [0.3717, 0.3892]
	90/10	0.9381 [0.9371, 0.9391]	0.5370 [0.5190, 0.5550]	0.2578 [0.2505, 0.2651]	0.3480 [0.3409, 0.3552]	0.7911 [0.7800, 0.8021]	0.3723 [0.3652, 0.3795]
SMOTE	50/50	0.9378 [0.9361, 0.9396]	0.5309 [0.5035, 0.5584]	0.2700 [0.2545, 0.2855]	0.3574 [0.3406, 0.3742]	0.7943 [0.7850, 0.8036]	0.3712 [0.3624, 0.3799]
	80/20	0.9387 [0.9371, 0.9403]	0.5472 [0.5191, 0.5753]	0.2678 [0.2583, 0.2773]	0.3589 [0.3501, 0.3676]	0.8052 [0.7982, 0.8122]	0.3731 [0.3648, 0.3814]
	85/15	0.9401 [0.9386, 0.9415]	0.5726 [0.5462, 0.5989]	0.2644 [0.2474, 0.2815]	0.3610 [0.3435, 0.3785]	0.8006 [0.7937, 0.8075]	0.3740 [0.3672, 0.3808]
	90/10	0.9383 [0.9372, 0.9394]	0.5410 [0.5212, 0.5607]	0.2578 [0.2473, 0.2682]	0.3487 [0.3390, 0.3585]	0.7939 [0.7854, 0.8024]	0.3582 [0.3489, 0.3675]
Basic SMOTE	50/50	0.9379 [0.9358, 0.9400]	0.5322 [0.5008, 0.5637]	0.2744 [0.2626, 0.2863]	0.3617 [0.3463, 0.3772]	0.8035 [0.7952, 0.8119]	0.3754 [0.3716, 0.3791]
	80/20	0.9385 [0.9372, 0.9399]	0.5444 [0.5214, 0.5674]	0.2622 [0.2497, 0.2748]	0.3534 [0.3413, 0.3654]	0.7927 [0.7798, 0.8057]	0.3694 [0.3572, 0.3816]
	85/15	0.9390 [0.9372, 0.9407]	0.5522 [0.5213, 0.5832]	0.2611 [0.2470, 0.2753]	0.3541 [0.3375, 0.3707]	0.7954 [0.7871, 0.8038]	0.3624 [0.3535, 0.3712]
	90/10	0.9371 [0.9358, 0.9384]	0.5190 [0.4992, 0.5388]	0.2667 [0.2583, 0.2750]	0.3522 [0.3414, 0.3629]	0.7843 [0.7723, 0.7962]	0.3616 [0.3539, 0.3693]
k-Means SMOTE	50/50	0.9385 [0.9377, 0.9393]	0.5399 [0.5273, 0.5524]	0.2800 [0.2683, 0.2917]	0.3685 [0.3571, 0.3799]	0.7906 [0.7848, 0.7964]	0.3788 [0.3738, 0.3838]
	80/20	0.9410 [0.9398, 0.9421]	0.5891 [0.5682, 0.6101]	0.2622 [0.2529, 0.2716]	0.3628 [0.3512, 0.3743]	0.7694 [0.7568, 0.7821]	0.3472 [0.3361, 0.3583]
	85/15	0.9397 [0.9385, 0.9410]	0.5677 [0.5439, 0.5915]	0.2578 [0.2460, 0.2695]	0.3540 [0.3420, 0.3661]	0.7737 [0.7657, 0.7817]	0.3573 [0.3523, 0.3624]
	90/10	0.9396 [0.9386, 0.9406]	0.5636 [0.5467, 0.5806]	0.2567 [0.2415, 0.2719]	0.3523 [0.3363, 0.3683]	0.7860 [0.7778, 0.7942]	0.3595 [0.3523, 0.3666]
SMOTE + Tomek Links	50/50	0.9368 [0.9352, 0.9385]	0.5153 [0.4906, 0.5401]	0.2756 [0.2638, 0.2873]	0.3586 [0.3465, 0.3707]	0.8001 [0.7916, 0.8087]	0.3765 [0.3711, 0.3819]
	80/20	0.9385 [0.9362, 0.9407]	0.5429 [0.5075, 0.5783]	0.2678 [0.2551, 0.2805]	0.3582 [0.3410, 0.3755]	0.7978 [0.7900, 0.8056]	0.3753 [0.3698, 0.3808]
	85/15	0.9402 [0.9393, 0.9410]	0.5740 [0.5588, 0.5892]	0.2611 [0.2465, 0.2757]	0.3584 [0.3442, 0.3726]	0.7919 [0.7818, 0.8020]	0.3727 [0.3651, 0.3804]
	90/10	0.9395 [0.9385, 0.9406]	0.5594 [0.5416, 0.5772]	0.2689 [0.2599, 0.2779]	0.3630 [0.3525, 0.3735]	0.7979 [0.7885, 0.8072]	0.3667 [0.3599, 0.3735]
CNN + Tomek Links	50/50	0.7904 [0.7805, 0.8003]	0.1012 [0.0963, 0.1062]	0.2867 [0.2769, 0.2964]	0.1495 [0.1437, 0.1552]	0.6103 [0.6028, 0.6178]	0.2334 [0.2165, 0.2503]
	80/20	0.8877 [0.8798, 0.8956]	0.2108 [0.1880, 0.2337]	0.2667 [0.2561, 0.2773]	0.2346 [0.2181, 0.2511]	0.6047 [0.5941, 0.6153]	0.2720 [0.2599, 0.2841]
	85/15	0.9249 [0.9217, 0.9280]	0.3697 [0.3384, 0.4010]	0.2389 [0.2225, 0.2553]	0.2897 [0.2697, 0.3097]	0.5630 [0.5578, 0.5681]	0.2170 [0.2013, 0.2327]
	90/10	0.9325 [0.9281, 0.9369]	0.4519 [0.3185, 0.5853]	0.0856 [0.0790, 0.0921]	0.1402 [0.1293, 0.1510]	0.5232 [0.5181, 0.5284]	0.1535 [0.1408, 0.1662]
Random Under-Sampling	50/50	0.8021 [0.7970, 0.8071]	0.1647 [0.1609, 0.1684]	0.5122 [0.4961, 0.5283]	0.2491 [0.2440, 0.2543]	0.7608 [0.7567, 0.7650]	0.2011 [0.1960, 0.2063]
	80/20	0.9174 [0.9156, 0.9191]	0.3472 [0.3368, 0.3575]	0.3267 [0.3190, 0.3343]	0.3364 [0.3297, 0.3432]	0.7866 [0.7814, 0.7917]	0.3575 [0.3488, 0.3661]
	85/15	0.9217 [0.9197, 0.9236]	0.3688 [0.3548, 0.3827]	0.3100 [0.2973, 0.3227]	0.3365 [0.3259, 0.3471]	0.7963 [0.7900, 0.8027]	0.3573 [0.3498, 0.3647]
	90/10	0.9375 [0.9359, 0.9391]	0.5249 [0.5022, 0.5476]	0.2867 [0.2804, 0.2929]	0.3705 [0.3618, 0.3793]	0.7977 [0.7907, 0.8048]	0.3720 [0.3654, 0.3787]
Tomek Links	50/50	0.8056 [0.8021, 0.8090]	0.1649 [0.1611, 0.1687]	0.5000 [0.4870, 0.5130]	0.2479 [0.2424, 0.2535]	0.7586 [0.7548, 0.7625]	0.1994 [0.1957, 0.2032]
	80/20	0.9160 [0.9130, 0.9189]	0.3409 [0.3262, 0.3555]	0.3289 [0.3196, 0.3382]	0.3342 [0.3275, 0.3410]	0.7880 [0.7842, 0.7919]	0.3627 [0.3563, 0.3692]
	85/15	0.9254 [0.9238, 0.9269]	0.3971 [0.3843, 0.4099]	0.3156 [0.3055, 0.3256]	0.3515 [0.3421, 0.3609]	0.8018 [0.7951, 0.8086]	0.3626 [0.3583, 0.3669]
	90/10	0.9355 [0.9332, 0.9377]	0.4979 [0.4699, 0.5259]	0.2911 [0.2736, 0.3086]	0.3662 [0.3520, 0.3803]	0.7948 [0.7864, 0.8032]	0.3676 [0.3556, 0.3796]

4.5 Ablation Study

The ablation analysis considers two comparisons supported by the experimental design: MSCMF with and without generative augmentation, and the effect of the latent backbone for WGAN-GP and VAE using MSCMF, NNMF, and LMF under the same downstream protocol.

Within MSCMF, generative augmentation produces only limited changes in F1-score. For WGAN-GP and VAE, the values reported here correspond to the highest mean F1-score observed across the four evaluated class ratios. For Enzyme, the no-augmentation value is 0.7777, compared with 0.7814 for WGAN-GP and 0.7810 for VAE. For GPCR, the corresponding values are 0.5656, 0.5597, and 0.5671; for Ion Channel, 0.8440, 0.8440, and 0.8438; and for Nuclear Receptor, 0.3568, 0.3628, and 0.3523.

The backbone comparison is more consistent. For both WGAN-GP and VAE, MSCMF gives the highest F1-score among the three latent representations in each protein family. The effect of generative augmentation therefore depends on both the dataset and the representation used for sample generation.

This ablation is not a complete factorial comparison of the three matrix-factorization backbones, since no-augmentation baselines are available only for MSCMF. Backbone effects are therefore assessed through the matched WGAN-GP and VAE configurations.

5 Discussion

5.1 Augmentation under Class Imbalance

The results do not identify a single augmentation method or class ratio that is consistently preferable across protein families. Performance also depends on the evaluation metric, since the configurations maximizing F1-score do not necessarily maximize ROC-AUC or PR-AUC.

The target class distribution affects the balance between precision and recall. Across several configurations, changes in the training ratio produce higher recall at the cost of precision, or the reverse. The 80/20 and 90/10 settings frequently produce higher F1-scores than the more balanced 50/50 setting. This variability is consistent with the sensitivity of resampling methods to class overlap and synthetic-sample placement [3, 5, 6], and indicates that the augmentation ratio should be treated as part of the model configuration.

The MSCMF configuration without augmentation remains close to the best augmented results, particularly for Enzyme and Ion Channel, while the improvements observed for GPCR and Nuclear Receptor are also limited. Augmentation therefore does not provide a consistent improvement over the non-augmented MSCMF baseline.

5.2 Role of the Latent Representation and Generative Augmentation

Generative models do not consistently outperform classical methods. WGAN-GP and VAE are competitive but do not achieve the highest F1-score in any protein family,

although VAE leads GPCR PR-AUC and Latent Diffusion leads Nuclear Receptor ROC-AUC. This shows that gains in ranking metrics do not necessarily translate into higher F1-score.

The ablation results further show that performance depends on the latent representation. For both WGAN-GP and VAE, MSCMF achieves higher F1-scores than NNMF and LMF across all four protein families, indicating that generative augmentation cannot compensate for weaker underlying representations.

The MSCMF + WGAN-GP + DNN framework introduced in our previous work [60] remains competitive but does not dominate the expanded benchmark. Its contribution in the present study is better interpreted as one generative configuration within a broader controlled comparison rather than as a universally superior augmentation strategy.

Latent Diffusion follows a similar pattern. Its best result is obtained for Nuclear Receptor ROC-AUC, but it does not lead F1-score or PR-AUC across the four families. Increasing generative-model complexity therefore does not, by itself, produce consistent improvements over simpler resampling or perturbation methods for these latent DTI representations.

5.3 Implications for Evaluation

The differences between F1-score, ROC-AUC, and PR-AUC rankings are relevant under strong class imbalance. High accuracy is observed in several configurations, particularly for Enzyme and Ion Channel, but accuracy alone provides limited information about interaction detection because non-interactions constitute the majority class. Precision, recall, F1-score, and PR-AUC provide complementary information about the minority-class predictions.

Under the fixed outer cross-validation protocol, each drug–target pair contributes to the reported metrics only when it belongs to an outer-test fold. The corresponding interaction label is excluded from matrix-factorization fitting, and the pair is not used for augmentation or DNN training in that fold. Predictions from the five outer-test folds are pooled to obtain one out-of-fold prediction for each original pair.

The confidence intervals quantify stochastic training variability under these fixed partitions. They do not capture variability that would arise from selecting different cross-validation partitions. This distinction is important when differences between configurations are small, as occurs for several of the strongest Enzyme and Ion Channel results.

6 Conclusion

This study compared 14 augmentation, resampling, filtering, and generative methods for DTI prediction across Enzyme, GPCR, Ion Channel, and Nuclear Receptor datasets. The methods were evaluated under fixed stratified five-fold outer cross-validation, common DNN training conditions, and four target non-interaction/interaction ratios.

No augmentation method or class ratio dominates across protein families or evaluation metrics. Classical methods produce the highest F1-score in all four families, while

VAE and Latent Diffusion lead selected PR-AUC and ROC-AUC results. The MSCMF configuration without augmentation remains competitive, showing that modifying the class distribution does not necessarily improve classification performance.

The generative-model comparisons further show that augmentation performance is linked to the latent representation. For both WGAN-GP and VAE, MSCMF gives the highest mean F1-score among the three evaluated matrix-factorization backbones across the four protein families. Overall, the results support evaluating augmentation jointly with the representation, class ratio, and performance metric rather than treating augmentation as an independent preprocessing step.

Declarations

Abbreviations

DTI: Drug–Target Interaction; MSCMF: Multiple Similarities Collaborative Matrix Factorization; NNMF: Non-negative Matrix Factorization; LMF: Logistic Matrix Factorization; DNN: Deep Neural Network; ROS: Random Over-Sampling; RUS: Random Under-Sampling; SMOTE: Synthetic Minority Over-sampling Technique; ADASYN: Adaptive Synthetic Sampling; CNN: Condensed Nearest Neighbor; VAE: Variational Autoencoder; WGAN-GP: Wasserstein Generative Adversarial Network with Gradient Penalty; ROC-AUC: Area Under the Receiver Operating Characteristic Curve; PR-AUC: Area Under the Precision–Recall Curve.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Availability of data and materials

The datasets analyzed in this study are publicly available through the Yamanishi benchmark repository at http://web.kuicr.kyoto-u.ac.jp/supp/yoshi/drugtarget/. The source code used to implement the experimental pipeline is publicly available at https://github.com/vitormag-gh/DTI_2026.

Competing interests

The authors declare no competing interests.

Funding

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) through a Discovery Grant (RGPIN-2025-06792) and internal funding from TELUQ University (FAR3 program).

Authors' contributions

V.M.S. conducted most of the experiments and wrote the main manuscript text. K.H. provided methodological guidance and reviewed and edited the manuscript. N.M. served as principal investigator, supervised the research, and reviewed and edited the manuscript. All authors reviewed and approved the final manuscript.

Acknowledgements

AI-based tools were used for language editing and proofreading. They were not used to generate experimental results or technical methods.

References

[1] Yamanishi, Y., Araki, M., Gutteridge, A., Honda, W., Kanehisa, M.: Prediction of drug–target interaction networks from the integration of chemical and genomic spaces. Bioinformatics **24**, 232–240 (2008) https://doi.org/10.1093/bioinformatics/btn162

[2] Sertkaya, A., Beleche, T., Jessup, A., Sommers, B.D.: Costs of drug development and research and development intensity in the us, 2000–2018. JAMA Netw. Open **7**(6), 2415445 (2024) https://doi.org/10.1001/jamanetworkopen.2024.15445

[3] Batista, G., Prati, R.C., Monard, M.: A study of the behavior of several methods for balancing machine learning training data. SIGKDD Explorations **6**(1), 20–29 (2004)

[4] Ezzat, A., Wu, M., Li, X.-L., Kwoh, C.-K.: Drug-target interaction prediction via class imbalance-aware ensemble learning. BMC Bioinformatics **17**, 509 (2016) https://doi.org/10.1186/s12859-016-1377-y

[5] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research **16**, 321–357 (2002)

[6] Han, H., Wang, W.-Y., Mao, B.-H.: Borderline-smote: A new over-sampling method in imbalanced data sets learning. In: ICIC, pp. 878–887 (2005)

[7] Gao, L.-G., Yang, M.-Y., Wang, J.-X.: Collaborative matrix factorization with soft regularization for drug-target interaction prediction. J. Comput. Sci. Technol. **36**, 310–322 (2021) https://doi.org/10.1007/s11390-021-0844-8

[8] Hao, M., Bryant, S.H., Wang, Y.: Predicting drug-target interactions by dual-network integrated logistic matrix factorization. Sci. Rep. **7**, 40376 (2017) https://doi.org/10.1038/srep40376

[9] Liu, Y., Wu, M., Miao, C., Zhao, P., Li, X.-L.: Neighborhood regularized logistic matrix factorization for drug-target interaction prediction. PLoS Comput. Biol.

(2016) https://doi.org/10.1371/journal.pcbi.1004760

[10] Kim, J., Park, S., Min, D., Kim, W.: Comprehensive survey of recent drug discovery using deep learning. Int. J. Mol. Sci. **22**, 9983 (2021) https://doi.org/10.3390/ijms22189983

[11] Askr, H., Elgeldawi, E., Aboul Ella, H., Elshaier, Y.A., Gomaa, M.M., Hassanien, A.E.: Deep learning in drug discovery: an integrative review and future challenges. Artificial Intelligence Review **56**, 5975–6037 (2023) https://doi.org/10.1007/s10462-022-10306-1

[12] Abbou, S.I., Bouziane, H., Chouarfia, A.: Logistic matrix factorisation and generative adversarial neural network based method for predicting drug target interactions. Molecular Diversity **25**, 1497–1516 (2021) https://doi.org/10.1007/s11030-021-10273-9

[13] Abdelliche, Y.M.: Identification des interactions médicaments-cibles par une méthode basée sur l'apprentissage profond. PhD thesis, École nationale Supérieure d'Informatique (2022)

[14] Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Bengio, Y.: Generative Adversarial Networks (2014). https://arxiv.org/abs/1406.2661

[15] Aldahdooh, J., Vähä-Koskela, M., Tang, J., Tanoli, Z.: Using bert to identify drug-target interactions from whole pubmed. BMC Bioinformatics **23**, 245 (2022) https://doi.org/10.1186/s12859-022-04768-x

[16] Benazzoug, E.N.: Exploration of drug-protein interactions through graph neural networks and lstm. PhD thesis, École nationale Supérieure d'Informatique (2024)

[17] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (2019). https://arxiv.org/abs/1810.04805

[18] Hamitouche, T.: Utilisation de bert pour la prédiction des interactions médicament-protéine. Master's thesis, École nationale Supérieure d'Informatique (2024)

[19] Kang, H., Goo, S., Lee, H., Chae, J.-W., Yun, H.-Y., Jung, S.: Fine-tuning of bert model to accurately predict drug–target interactions. Pharmaceutics **14**, 1710 (2022) https://doi.org/10.3390/pharmaceutics14081710

[20] Wang, S., Guo, Y., Wang, Y., Sun, H., Huang, J.: Smiles-bert: Large scale unsupervised pre-training for molecular property prediction. In: ACM BCB, pp. 429–436 (2019). https://doi.org/10.1145/3307339.3342186

[21] Zhang, S., Dong, L., Li, X., Zhang, S., Sun, X., Wang, S., Wang, G.: Applications of transformer-based language models in bioinformatics: a survey. Bioinformatics Advances **3**, 001 (2023) https://doi.org/10.1093/bioadv/vbad001

[22] Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: NeurIPS (2020)

[23] Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., Zhang, W., Cui, B., Yang, M.-H.: Diffusion models: A comprehensive survey of methods and applications. ACM Computing Surveys **56**(4), 1–39 (2024) https://doi.org/10.1145/3626235

[24] Lu, Y., Shen, M., Wang, H., Wang, X., Rechem, C., Wei, W.: Machine Learning for Synthetic Data Generation: A Review (2024). https://arxiv.org/abs/2302.04062

[25] Mumuni, A., Mumuni, F.: Data augmentation: A comprehensive survey of modern approaches. Array **16**, 100258 (2022) https://doi.org/10.1016/j.array.2022.100258

[26] Xu, W., Yang, X., Guan, Y., Cheng, X., Wang, Y.: Integrative approach for predicting drug-target interactions via matrix factorization and broad learning systems. Math. Biosci. Eng. **21**, 2608–2625 (2024) https://doi.org/10.3934/mbe.2024115

[27] Zhang, J., Xie, M.: Graph regularized non-negative matrix factorization with prior knowledge consistency constraint for drug–target interactions prediction. BMC Bioinformatics **23**, 564 (2022) https://doi.org/10.1186/s12859-022-05119-6

[28] Li, Y., Huang, Y.-A., You, Z.-H., Li, L.-P., Wang, Z.: Drug-target interaction prediction based on drug fingerprint information and protein sequence. Molecules **24**, 2999 (2019) https://doi.org/10.3390/molecules24162999

[29] Gao, K.Y., Fokoue, A., Luo, H., Iyengar, A., Dey, S., Zhang, P.: Interpretable drug target prediction using deep neural representation. In: IJCAI, pp. 3371–3377 (2018)

[30] Sajadi, S.Z., Zare Chahooki, M.A., Tavakol, M., Gharaghani, S.: Matrix factorization with denoising autoencoders for prediction of drug–target interactions. Molecular Diversity **27**, 1333–1343 (2023) https://doi.org/10.1007/s11030-022-10492-8

[31] Gómez-Bombarelli, R., Wei, J.N., Duvenaud, D., Hernández-Lobato, J.M., Sánchez-Lengeling, B., Aguilera-Iparraguirre, J., Aspuru-Guzik, A.: Automatic chemical design using a data-driven continuous representation of molecules. ACS Cent. Sci. **4**, 268–276 (2018)

[32] Zheng, X., Ding, H., Mamitsuka, H., Zhu, S.: Collaborative matrix factorization

with multiple similarities for predicting drug-target interactions. In: KDD, pp. 1025–1033 (2013). https://doi.org/10.1145/2487575.2487670

[33] Luo, Y., Zhao, X., Zhou, J., Yang, J., Zhang, Y., Kuang, W., Zeng, J.: A network integration approach for drug-target interaction prediction and computational drug repositioning from heterogeneous information. Nat. Commun. **8**, 573 (2017) https://doi.org/10.1038/s41467-017-00680-8

[34] Öztürk, H., Özgür, A., Ozkirimli, E.: Deepdta: deep drug–target binding affinity prediction. Bioinformatics **34**, 821–829 (2018) https://doi.org/10.1093/bioinformatics/bty593

[35] Wang, Y.-B., You, Z.-H., Yang, S., Yi, H.-C., Chen, Z.-H., Zheng, K.: A deep learning-based method for drug–target interaction prediction based on long short-term memory neural network. BMC Med. Inform. Decis. Mak. **20**, 49 (2020) https://doi.org/10.1186/s12911-020-1052-0

[36] Ezzat, A., Wu, M., Li, X.-L., Kwoh, C.-K.: Drug-target interaction prediction using ensemble learning and dimensionality reduction. Methods **129**, 81–88 (2017) https://doi.org/10.1016/j.ymeth.2017.05.016

[37] Zhan, X., You, Z.-H., Cai, J., Li, L., Yu, C., Pan, J., Kong, J.: Prediction of drug-target interactions by ensemble learning method from protein sequence and drug fingerprint. IEEE Access **8**, 185465–185476 (2020) https://doi.org/10.1109/ACCESS.2020.3026479

[38] Drummond, C., Holte, R.C.: Class imbalance, and cost sensitivity: Why under-sampling beats over-sampling. In: Workshop on Learning from Imbalanced Datasets II (2003)

[39] Tomek, I.: Two modifications of cnn. IEEE Trans. Syst. Man Cybern. **SMC-6**, 769–772 (1976)

[40] He, H., Bai, Y., Garcia, E.A., Li, S.: Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In: IJCNN (2008)

[41] Nguyen, H.M., Cooper, E.W., Kamei, K.: Borderline over-sampling for imbalanced data classification. Int. J. Knowl. Eng. Soft Data Paradigms, 4–21 (2011)

[42] Last, F., Douzas, G., Bacao, F.: Oversampling for Imbalanced Learning Based on K-Means and SMOTE (2017). https://arxiv.org/abs/1711.00837

[43] Bjerrum, E.J.: SMILES Enumeration as Data Augmentation for Neural Network Modeling of Molecules (2017). https://arxiv.org/abs/1703.07076

[44] You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., Shen, Y.: Graph contrastive

learning with augmentations. In: NeurIPS (2020)

[45] Rong, Y., Bian, Y., Xu, T., Xie, W., Wei, Y., Huang, W., Huang, J.: Self-Supervised Graph Transformer on Large-Scale Molecular Data (2020). https://arxiv.org/abs/2007.02835

[46] Elnaggar, A., Heinzinger, M., Dallago, C., Rehawi, G., Wang, Y., Jones, L., Gibbs, T., Feher, T., Angerer, C., Steinegger, M., Bhowmik, D., Rost, B.: Prottrans: Toward understanding the language of life through self-supervised learning. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(10), 7112–7127 (2022) https://doi.org/10.1109/TPAMI.2021.3095381

[47] Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: Beyond empirical risk minimization. In: ICLR (2018)

[48] Wang, Y., Wang, J., Cao, Z., Farimani, A.B.: Molecular Contrastive Learning of Representations via Graph Neural Networks (2022). https://arxiv.org/abs/2207.08845

[49] Shi, Y., Valizadeh Aslani, T., Wang, J., Ren, P., Zhang, Y., Hu, M., Liang, H.: Improving Imbalanced Learning by Pre-finetuning with Data Augmentation (2022)

[50] Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. In: ICLR (2021)

[51] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: CVPR, pp. 10684–10695 (2022)

[52] Vahdat, A., Kreis, K., Kautz, J.: Score-based generative modeling in latent space. In: NeurIPS (2021)

[53] Zheng, J., Xiao, X., Qiu, W.-R.: Dti-bert: Identifying drug-target interactions in cellular networking based on bert and deep learning method. Front. Genet. **13**, 859188 (2022) https://doi.org/10.3389/fgene.2022.859188

[54] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017)

[55] Huang, K., Xiao, C., Glass, L.M., Sun, J.: Moltrans: Molecular interaction transformer for drug–target interaction prediction. Bioinformatics, 830–836 (2021)

[56] Gim, M., Choe, J., Baek, S., Park, J., Lee, C., Ju, M., Kang, J.: Arkdta: attention regularization guided by non-covalent interactions for explainable drug–target binding affinity prediction. Bioinformatics **39**, 448–457 (2023) https://doi.org/10.1093/bioinformatics/btad207

[57] Hua, Y., Feng, Z., Song, X., Wu, X.-J., Kittler, J.: Mmdg-dti: Drug–target interaction prediction via multimodal feature fusion and domain generalization. Pattern Recognit. **157**, 110887 (2025) https://doi.org/10.1016/j.patcog.2024.110887

[58] Zhao, Q., Zhao, H., Zheng, K., Wang, J.: Hyperattentiondti: improving drug–protein interaction prediction by sequence-based deep learning with attention mechanism. Bioinformatics **38**, 655–662 (2022) https://doi.org/10.1093/bioinformatics/btab715

[59] Lee, I., Nam, H.: Infusing Linguistic Knowledge of SMILES into Chemical Language Models (2022). https://arxiv.org/abs/2205.02089

[60] Silva, V.M., Henni, K., Abdelliche, Y., Mezghani, N.: A matrix factorization and generative modeling framework for drug–target interaction prediction. In: Proceedings of the 19th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2026) – Volume 2: BIODEVICES, BIOIMAGING, BIOINFORMATICS, pp. 688–695 (2026). https://doi.org/10.5220/0014484300004070

[61] Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 214–223 (2017)

[62] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.: Improved training of wasserstein gans. In: Advances in Neural Information Processing Systems, vol. 30, pp. 5767–5777 (2017)