

Article

Robust Transcription Factor Binding Site Prediction and Explainability Using a Heterogeneous Mixture of Experts Architecture

Aakash Tripathi ^{1,†} , Ian E. Nielsen ^{2,†} , Muhammad Umer ³ , Ravi P. Ramachandran ^{2,*}  and Ghulam Rasool ¹ ¹ Machine Learning, Moffitt Cancer Center, 12902 USF Magnolia Drive, Tampa, FL 33612, USA; aakash.tripathi@moffitt.org (A.T.); ghulam.rasool@moffitt.org (G.R.)² Department of Electrical and Computer Engineering, Rowan University, Glassboro, NJ 08028, USA; nielsen6@rowan.edu³ Gildart Haase School of Engineering, Fairleigh Dickinson University, Teaneck, NJ 07666, USA; m.umer@fdu.edu

* Correspondence: ravi@rowan.edu

† These authors contributed equally to this work.

Abstract

Transcription Factor Binding Site (TFBS) prediction is central to understanding gene regulation and various biological processes. This study introduces HetMoE, a heterogeneous, embedding-gated Mixture-of-Experts for TFBS prediction. A gating network operates on the embeddings produced by a pool of complementary expert backbones (a modified-DeepBIND convolutional network, DeepSEA, and DanQ, with a fine-tuned DNABERT-6 genomic language model as an optional expert), so that models of different architectures are combined and weighted on a per-input basis. Models are trained against GC- and repeat-matched real genomic negatives, a fair protocol that avoids the dinucleotide-shuffle artifact, and evaluated with a balanced-test-set protocol (deterministic inference, $B = 1000$ paired bootstrap and Analysis of Variance (ANOVA)) on in-distribution and out-of-distribution (OOD) factors. HetMoE attains the best in-distribution performance (mean Area Under the Curve (AUC) 0.881) and, on a held-out set stratified by DNA-binding-domain family, surpasses fine-tuned DNABERT-6 on the motif-bearing OOD mean across three random seeds (0.821 ± 0.005 vs. 0.799 ± 0.008 , a gain present in every seed), most strongly on the sequence-specific and within-family factors. The advantage comes from the gating mechanism rather than from ensembling: input-dependent gating exceeds a static average of the same experts by 0.073 AUC and the best single expert by 0.088, and the configuration selected on in-distribution data is a pretraining-free pool of convolutional experts. We further show that the common dinucleotide-shuffle negative protocol inflates the apparent margin (to a mean of 0.864), which shows the importance of fair, genomically matched negatives. We also introduce an attribution method (ShiftSmooth) that improves interpretability by averaging the gradient over small shifts of the input sequence, giving more reliable attribution for motif discovery and localization than the Vanilla Gradient. Together these provide an efficient and interpretable approach to TFBS prediction that can support further study of genome regulation.



Academic Editors: Bowen Liu and Md Nahid Hasan

Received: 3 May 2026

Revised: 3 July 2026

Accepted: 6 July 2026

Published: 10 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.**Keywords:** bioinformatics; transcription factor binding site prediction; heterogeneous mixture of experts; genomic language models; Analysis of Variance (ANOVA); explainable artificial intelligence; ShiftSmooth method

MSC: 68T07; 92B20; 92-08

1. Introduction

Deoxyribonucleic acid (DNA) is the blueprint for the development and functioning of the human body. The information encoded in DNA is converted into functional molecules through the process of transcription, where DNA is transcribed into messenger RNA (mRNA) [1]. Transcription is regulated by proteins called transcription factors that bind to specific DNA segments known as transcription factor binding sites (TFBS). TFBS can be located in gene promoters that initiate transcription or in cis-regulatory elements that modulate the transcription of nearby genes [2]. These binding sites contain short, recurring DNA sequence motifs that determine the function of the associated DNA regions. Identifying TFBS motifs is essential for understanding fundamental biological processes, including transcriptional regulation, mRNA splicing, and the formation of protein complexes [3]. Motif identification also supports the study of genetic diseases, the development of targeted therapies, gene evolution, and other key biological mechanisms [4]. Accurate and efficient machine-learning methods for TFBS motif identification therefore support the study of genome regulation.

Various machine learning approaches have been proposed for TFBS prediction [5–7]. One widely used method is DeepBIND [8], which employs a convolutional neural network (CNN) architecture. Another approach is DeepRAM [9] which incorporates recurrent neural networks (RNNs) and long short-term memory (LSTM) architectures [10,11], which are well adapted for sequential data. Recently, transformer-based architectures, like DNABERT [12], have achieved good performance on various genomic tasks.

A convolutional expert based on the DeepBIND architecture [8] is the primary building block of our proposed Mixture of Experts (MoE) approach. Convolutional experts of this kind train quickly, process large volumes of high-throughput sequence data in parallel (a benefit for the large-scale genomic data involved in TFBS prediction), learn aspects of the DNA-protein interaction automatically, and remain interpretable [13,14]. These properties make convolutional models a strong foundation for the expert pool, which we extend with the complementary backbones described below.

A preliminary comparison on GATA3 (GATA Binding Protein 3, a transcription factor important in T-cell development) supports this design. Embeddings taken directly from a pre-trained DNABERT model gave a lower test Area Under the Curve (AUC) than convolutional DeepBIND embeddings, whose filters capture local sequence patterns more directly. The AUC is computed from the Receiver Operating Characteristic (ROC) curve, which plots the true positive rate (TPR) against the false positive rate (FPR) across decision thresholds. Off-the-shelf language-model embeddings are therefore not competitive on their own, although a transformer fine-tuned for binding-site prediction remains a useful point of comparison, as the results later show.

We propose a heterogeneous, embedding-gated Mixture-of-Experts (HetMoE) approach. This technique places the convolutional DeepBIND expert within a heterogeneous pool of complementary backbones that also includes the DeepSEA and DanQ convolutional trunks. The design follows from two observations. First, convolutional experts are fast, interpretable, and effective at capturing the local sequence patterns that define binding motifs. Second, any single architecture has limited robustness: its accuracy tends to drop when the test data follow a different distribution from the training data [15–17], and a simple convolutional network generalizes poorly to TFBS patterns it has not seen. Combining complementary experts under a gating network addresses this limitation, because the

mixture can draw on whichever experts suit each input [18]. The configuration selected on in-distribution data is a pretraining-free convolutional pool, and it surpasses a fine-tuned DNABERT-6 genomic language model on held-out, family-stratified out-of-distribution factors. This work evaluates HetMoE in terms of predictive performance, robustness to such out-of-distribution factors, and explainability for TFBS motif identification.

Figure 1 provides an overview of the experiments and methods employed in this study. It illustrates the entire process, from data preprocessing to model evaluation. It depicts how the sequence data are processed and fed into the individual expert models (the ConvNet, DeepSEA, and DanQ convolutional backbones, with a fine-tuned DNABERT-6 as a further candidate expert), which are then integrated into the proposed HetMoE model. Figure 1 also illustrates the evaluation process using attribution mapping, including the ShiftSmooth method.

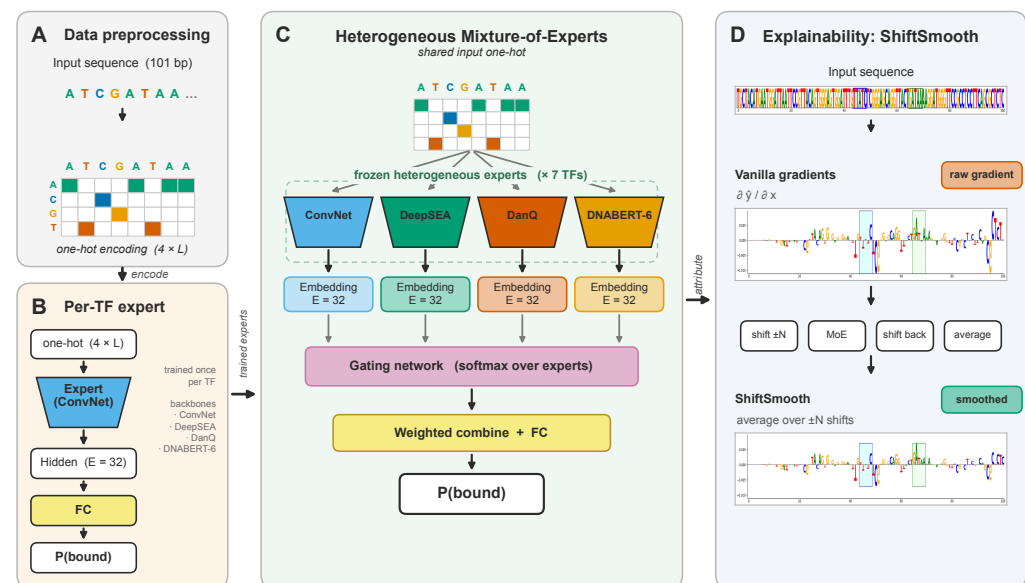


Figure 1. Overview of HetMoE. An input DNA sequence is processed by a heterogeneous pool of frozen experts (a modified-DeepBIND ConvNet, DeepSEA, and DanQ, with a fine-tuned DNABERT-6 as a candidate expert, for each of seven training factors spanning seven DNA-binding-domain families), each reduced to a common 32-dimensional embedding. A gating network produces input-dependent softmax weights over all experts and combines their embeddings into the binding prediction $P(\text{bound})$. Because the gate sees only embeddings, architecturally different experts are combined. The model is evaluated on a family-stratified out-of-distribution set, where the selected pretraining-free convolutional pool generalizes across DBD families and surpasses a standalone fine-tuned DNABERT-6 baseline, and is interpreted with the ShiftSmooth attribution method. Arrows indicate the direction of data flow through the pipeline; each expert backbone and its 32-dimensional embedding are shown in a matching color, and nucleotide letters follow the standard A/C/G/T color scheme.

This paper makes the following contributions.

1. Individual expert CNN models, each specialized to a different TFBS pattern, are configured by modifying the DeepBIND architecture (described in detail later). This is essential to enable the integration of the expert models within our MoE framework.
2. A new heterogeneous, embedding-gated Mixture-of-Experts, HetMoE, is introduced for TFBS prediction. In contrast to conventional sparse MoE models that route each input to a subset of experts, HetMoE adopts a many-experts-to-one-output design in which a pool of complementary expert backbones (a modified-DeepBIND convolutional network, DeepSEA, and DanQ, with a fine-tuned DNABERT-6 genomic

language model as an optional expert) each process the input, and a gating network learns input-dependent weights over their embeddings. Because the gate operates only on embeddings, architecturally different experts can be combined, yielding improved generalization and robustness.

3. A performance evaluation on in-distribution data (same statistical distribution as the training samples) using Analysis of Variance (ANOVA) [19] shows the performance stability of the MoE approach. The in-distribution binding sites span seven transcription factors across seven DNA-binding-domain (DBD) families (ARID3A, FOXM1, GATA3, JUND, MAX, GABPA, and SP1; Section 3.6), which are further described in the Appendix A. The data are obtained from the ENCODE database [20,21] and can be accessed from the ENCODE portal (<https://www.encodeproject.org/>).
4. The same ANOVA framework is applied to out-of-distribution (OOD) data, whose statistical distribution differs from the training samples, providing a measure of the robustness of all models. The robustness of the MoE method is shown by improved generalizability to unseen data compared to the individual expert CNN models. The held-out OOD factors are drawn from the ENCODE corpus and stratified by DBD family (Section 3.6): within-family members of the trained families (testing motif-family transfer), entirely unseen families, and a separately-reported non-motif group. This stratified design gives an honest, leakage-free assessment of generalization rather than pooling easy and near-unlearnable factors.
5. A new attribution mapping technique, ShiftSmooth, is introduced to improve interpretability and the robustness of explanations by averaging gradients over small shifts of the input. Attributional robustness is achieved and refers to the ability of a method to remain stable and unaffected by minor perturbations or anomalies in the input data that do not influence the final decision [15]. Both the individual expert model and the MoE architecture are compared using gradient-based methods and ShiftSmooth. The issue of explainability [15] through attribution mapping [17] is important to gain user trust in the decision of the model.

The paper is organized as follows. Section 2 provides background and related work regarding the model architectures and explainability methods. In Section 3, we present our proposed approaches (the HetMoE architecture and ShiftSmooth) and the experimental setup for data processing, training, performance evaluation, and explainability analysis. The prediction performance comparison (using ANOVA) of the MoE and expert models is given in Section 4. The explainability results are given in Section 5. The limitations of the study are discussed in Section 6. The conclusions are recorded in Section 7. A data and code availability statement is given in the Appendix A.

2. Background

This section introduces the model architectures, and the specific attribution methods utilized in this work. Related work is also described.

2.1. Model Architectures

We adopt DeepBIND [8] as the primary convolutional expert for TFBS prediction, for the advantages described earlier. DeepBIND also addresses several challenges associated with learning sequence specificities, such as (1) handling the data formats typically used for genomic data and (2) working effectively across the various assays used to obtain genomic data.

The DeepBIND model uses a four-stage process to create a binding score for each input sequence. Note that the binding score quantifies the predicted strength of interaction between a transcription factor and an input DNA/RNA sequence. A higher score indicates

stronger binding, while a lower score indicates lesser binding. The four-stage process to create a binding score in the DeepBIND model is briefly discussed here. First, the sequence undergoes convolution to extract local features. Next, a rectification stage applies a non-linear activation function to introduce non-linearity into the model. The third stage involves pooling to reduce the spatial dimensions of the feature maps and retain the most salient features. Finally, the pooled features are passed through a non-linear neural network to produce the binding score. During training, backpropagation is used to update the model's parameters. DeepBIND can be trained on in-vitro data to predict specificities in in-vivo data and vice versa, making it a versatile tool for studying the effects of genetic variants on disease susceptibility. Our implementation of DeepBIND is through a modified architecture as discussed in Section 3.

While DeepBIND has shown good performance in predicting sequence specificities, recent advances in machine learning have introduced more efficient architectures that can achieve similar or better accuracy at a lower computational cost. One such approach is the MoE [22] which was initially proposed decades ago [23] but has only recently been implemented in modern deep learning [24,25].

Our model is therefore a dense (soft) mixture of experts in the classical sense, in which the gating network produces softmax weights over all experts and combines their representations, rather than a sparse MoE that routes each input to a subset of experts via top- k selection.

Typical MoE models use a “one-input-to-many-experts” (1: N) setup, where a single input is dynamically routed to a subset of experts, each specializing in different aspects of the input. Our model (HetMoE) employs a “many-experts-to-one-output” (N :1) configuration. Each expert is pre-trained on one specific TFBS pattern and processes the entire input independently; in particular, because the gate operates on expert embeddings, the experts need not share an architecture; our pool combines modified-DeepBIND convolutional experts with DeepSEA and DanQ trunks, with a fine-tuned DNABERT-6 genomic language model as a further candidate in the full pool. Instead of routing inputs, our gating mechanism weights the outputs of all experts to produce a final prediction, so the strengths of all experts are combined simultaneously. This N :1 approach offers several advantages without compromising prediction accuracy as follows.

1. It is more flexible, allowing for easy incorporation of new experts without full model retraining.
2. It is more interpretable, as we can analyze which experts contribute most to each prediction.
3. It is efficient to train, as the pre-trained expert weights remain frozen during MoE training and only the gating network is optimized. We note that, because all experts are evaluated for every input, this efficiency stems from the reduced training cost rather than from conditional computation or inference-time sparsity.
4. It effectively combines specialized knowledge from multiple experts for better generalization and model robustness across diverse TFBS patterns.

The proposed framework differs from conventional ensemble-learning schemes in three respects. (i) Unlike fixed weighted averaging, bagging, or majority voting, the gating network produces input-dependent softmax weights, so the contribution of each expert is re-computed for every input sequence rather than being a constant learned (or uniform) coefficient. (ii) Unlike stacking, which trains a meta-learner on the predictions (output probabilities) of the base models, our gating network operates on the experts' internal 32-dimensional embeddings, with each expert's own classifier head removed; the experts therefore contribute learned representations rather than final decisions. (iii) Unlike

sparse/top- k MoE routing, all experts are evaluated for every input (a dense, $N:1$ soft mixture), so no input is routed away from any expert. Table 1 summarizes these distinctions.

Table 1. Distinction between the proposed MoE framework and conventional combination schemes.

Scheme	Combines	Weights
Weighted averaging/bagging	final predictions	fixed (or uniform)
Stacking	final predictions	meta-learner (input-dependent)
Sparse top- k MoE	expert outputs	input-dependent, sparse (subset)
Proposed (dense $N:1$ MoE)	32-d embeddings	input-dependent, all experts

2.2. Attribution

Deep neural networks are often treated as ‘black boxes’ whose internal decision process is difficult for humans to follow. In genomics, this opacity limits trust: a binding-site prediction is far more useful when the nucleotides that drive it can be identified. Attribution methods help interpret such models by exposing which parts of the input most influence a prediction.

Analysis of deep learning networks using attribution mapping can be beneficial for many tasks, including image classification and object detection. Attribution mapping creates a map of importance scores attributed to each input feature, providing valuable insight into genomic data and disease risk prediction. It is a visual explanation of which input features are given the most attention by a neural network when making a decision for a particular input [15–17]. This can increase user trust in the model. Attribution can also provide key insights into where the model may have gone wrong in cases of incorrect classifications, making it a valuable resource for debugging and developing better models. This paper utilizes attribution to explain the full decision-making process of each model tested, thereby improving understanding and aiding the localization of motifs within sequences. The goal is attributional robustness which is defined earlier.

The popular attribution method known as Grad-CAM [26] is commonly used to highlight the most important features in the image and the object of interest. However, it is designed to explain only a specific layer, which makes it difficult to interpret global model knowledge with any single explanation [27]. Hence, the explanation offered by Grad-CAM is limited to fully visualizing the key features learned by the model [17]. In this study, our objective is to explain the decision-making process for the entirety of each model tested, which leads us to exclude this method from our results. Despite its limitations, Grad-CAM retains its utility in applications where visualizing key layers is beneficial.

For our experiments, we selected two gradient-based attribution methods that visualize the learned features across all layers (described in detail later). The first is the Vanilla Gradient. The second is our newly proposed ShiftSmooth Gradient. Gradient-based attribution methods are used because they are fast, have the advantage of being able to provide detailed and continuous insights into model predictions, are generally applicable across many model types and achieve attributional robustness [15]. Attributional robustness is important for explainability because it ensures the stability and reliability of the explanations provided by a model. Another key term is plausibility, which describes how convincing an explanation appears to a human observer [15,28].

As we show in the results, ShiftSmooth improves attributional robustness and plausibility over the Vanilla Gradient by taking an average of gradients across multiple modified inputs. In both approaches, the model gradient is used as a starting point to generate explainability maps [15]. The gradient is acquired by backpropagating the output activation for the class of interest with respect to the input features. The gradient represents how the output activation changes if an input feature is slightly perturbed.

2.2.1. Attribution in Genomic Data

There are few applications of attribution and explainability for genomic data within the machine learning literature [11,29]. Attribution has been proposed as a solution to the problem of discovering motifs that are important in regulatory protein behavior by the authors of DeepLIFT [29]. However, DeepLIFT was not chosen for our testing since (1) it depends on a reference point that is not entirely established [27,30], (2) it is not class specific [31], and (3) has an open problem with models containing multiple max pooling layers [29,32]. As mentioned earlier, we use the Vanilla Gradient and ShiftSmooth attribution methods for our explainability assessment.

The criteria of faithfulness for attribution mapping is an assessment of how important the attributed features are to the model itself [17]. When explaining genomic data, it is essential that the explanations are faithful. Otherwise, the explanation will not provide credible insight into the model decisions. The Evaluating Attributions by Adding Incrementally (EvalAttAI) approach [17] assessed the faithfulness of the most commonly used attribution methods and concluded that the Vanilla Gradient is the most faithful. For this reason, the Vanilla Gradient is chosen as one of the attribution methods used. We introduce ShiftSmooth to provide more robust explanations, especially in the context of genome data in which the context window often varies.

2.2.2. Vanilla Gradient

The Vanilla Gradient (also known as Saliency, sensitivity analysis, or simply the gradient [16,28]) is particularly advantageous as it provides a direct explanation of the model without any post-hoc modifications [33]. In other words, the gradient directly reveals how the prediction would change if the feature was changed. When compared with other widely used attribution methods, Vanilla Gradients generate visualizations that are most representative of the model itself [17]. In many implementations, this method uses the absolute value of the gradient, thereby eliminating sign information. The result is an attribution map that represents the magnitude of per-nucleotide importance, irrespective of whether a nucleotide affects the output activation positively or negatively. While the magnitude is always taken when creating Saliency maps, this is not always the case with Vanilla Gradients. Both methodologies offer valuable insights for elucidating model predictions. In this work, we do not consider the absolute value, thereby allowing us to discern whether each feature impacts the prediction in a positive or negative direction.

The gradient can be defined as an attribution map with $A_c^{\text{Gradient}} = \frac{\partial S_c}{\partial \mathbf{x}}$, where S_c is the target class activation score, and $\mathbf{x}$ represents the input features. However, Saliency is a magnitude, so the absolute value of the gradient must be taken. Therefore, we define Saliency maps as $A_c^{\text{Saliency}} = \left| \frac{\partial S_c}{\partial \mathbf{x}} \right|$. In this paper, we refer to the absolute value gradient maps as Saliency, and all non-absolute value gradients as Vanilla Gradients.

2.2.3. ShiftSmooth Gradient

A machine learning model can often be prone to error when given data whose distribution it has not seen before. These differences in data can often come in the form of noise, crop, and other unforeseen variables. To mimic real-world data, it is therefore beneficial to analyze what the attribution map would look like on average, given that noise is random and different each time it is sampled. SmoothGrad [34] does this for image processing and inspired the ShiftSmooth Gradient method that we propose.

SmoothGrad, in its original form, cannot be used for explaining TFBS prediction. TFBS prediction differs significantly from image processing, so simply adding some Gaussian noise to the input would not work, as genomic sequences must be given as one of four possible nucleotide types. SmoothGrad also does not account for cropping or shifting of

the viewing window, which is especially relevant for genomic data because the context window is often shifted to analyze a long sequence in smaller parts. We account for this variation by introducing the ShiftSmooth method.

The ShiftSmooth technique creates multiple copies of the original data sequence, shifting each one by a different amount, and passing them through the model. It then takes the gradient of each output with respect to each input and shifts the resulting gradient back to the initial starting position (i.e., if the data was shifted to the left twice, then the gradient is shifted to the right twice). Finally, it takes the average of the resulting gradients, creating the ShiftSmooth Gradient. Hence, the obtained attribution maps display to the end user a more robust visualization of what the model would look at on average under realistic circumstances.

2.2.4. Robustness

The accuracy of a non-robust model is compromised when evaluated on noisy data, adversarially attacked data, processed or modified data, and data collected in a different context than the training dataset. Models can be made robust through training and through architecture. The former can be achieved by training on a wider distribution of data, often by incorporating noisy and/or adversarially attacked data [35]. Models can also be robust due to their architecture. For instance, certain probability propagating architectures have been shown to be more robust to noise and adversarial attacks [36,37].

For attributional robustness, an explanation should be invariant to small variations that commonly appear in real-world data, such as noise, cropping, and shifting. The ShiftSmooth method takes an average of multiple shifted input gradients to enhance attributional robustness.

Robust models also create more robust and interpretable explanations. Robustly trained models have been shown qualitatively to produce more visually interpretable explanations [15] that are far less sensitive to noise in the data. A similar effect is achieved by SmoothGrad [34], which also produces more robust explanations. Robust training is not always feasible for all applications due to significantly greater computational requirements. Therefore, using robust explainability methods can be a desirable way to create more interpretable visualizations, which aid significantly in the identification of motifs.

2.3. Related Work

TFBS has been extensively studied using machine learning, particularly deep learning models, due to their ability to learn complex patterns directly from sequence data. Traditional models such as Position Weight Matrices (PWMs) and supervised learning approaches (e.g., Support Vector Machines (SVMs) and Random Forests) require predefined sequence motifs and heavily rely on manual feature engineering. Although these methods have been foundational, their accuracy drops significantly when dealing with the complexity of gene regulation, as they fail to capture non-linear interactions and do not generalize well across cell types [38].

In recent years, CNNs and RNNs have improved TFBS prediction. DeepBIND [8] and DeepSEA [39] are well-known CNN-based methods that capture local sequence patterns, offering better performance than traditional approaches by automatically learning from raw genomic sequences. However, one of the main limitations of CNNs is their local receptive field, which hinders their ability to capture long-range dependencies important for biological sequence analysis [38]. DeepSEA addresses this issue partially by incorporating more layers, but this comes at the cost of increased computational complexity and reduced interpretability [40].

While RNNs, particularly LSTM networks, are better suited for handling sequential dependencies, they tend to be computationally expensive and prone to vanishing gradients

over long sequences [38]. More recently, Transformer-based architectures [33] such as DNABERT have been employed to handle long-range interactions using self-attention mechanisms. DNABERT leverages pre-trained k -mer embeddings to generalize a broad range of DNA sequences. While this model shows strong generalizability in unsupervised tasks, it often underperforms compared to task-specific models designed for TFBS prediction, particularly in cases where subtle sequence variations are critical [41].

Ensemble models have been increasingly used to combine the strengths of multiple models. The MoE models have been applied in other areas of bioinformatics, particularly in the prediction of drug-target interactions [40], where they dynamically assign different experts to different subsets of the input space. The advantage of MoE lies in its ability to leverage multiple specialized models tailored to specific patterns. However, to our knowledge, MoE models have not been applied for the prediction of TFBS. Most ensemble models also focus on improving performance through aggregation without addressing interpretability. This is critical in genomics, where understanding why a model makes a particular prediction is as important as the prediction itself.

Deep learning models for TFBS prediction have traditionally struggled with interpretability. Attribution methods, such as Saliency Maps [16,28], Integrated Gradients [42], and DeepLIFT [29], have been proposed to highlight important nucleotides in the input sequence that contribute to the model's decision. However, these methods often fail to provide robust explanations, as they can be highly sensitive to small changes in the input sequence. Methods like Grad-CAM [26], while useful for interpreting image data, are also less applicable to genomic sequences, where subtle shifts in motif location can significantly alter the biological interpretation.

Unlike existing approaches that train a single model per transcription factor (e.g., DeepBIND [8] or DeepSEA [39]), HetMoE uses a heterogeneous, embedding-gated Mixture-of-Experts to integrate a pool of complementary expert backbones (a modified-DeepBIND CNN together with DeepSEA and DanQ, with a fine-tuned DNABERT-6 as an optional expert), each specializing in different TFBS patterns. Rather than routing each input to a single expert as in conventional sparse MoE models, every expert processes the entire input independently, and the gating network learns input-dependent weights that combine the embeddings produced by all experts. This many-experts-to-one-output design lets the model leverage the complementary strengths of all experts simultaneously, enabling it to generalize better across different transcription factors. While models like DNABERT aim for generalizability through unsupervised learning, our method retains the specificity of task-focused models while extending its utility to OOD transcription factors (TFs) [38]. Each expert is independently pre-trained on a single transcription factor and may be convolutional or transformer-based. The gating network weighs the contributions of all experts based on the input sequence, so the relative emphasis placed on each expert adapts to the type of TFBS being analyzed without discarding the information captured by the others. This improves performance and flexibility, especially in OOD scenarios where the model encounters unseen TFs [40].

There are multiple existing attribution methods used for genomic applications which provide a map of importance scores for each nucleotide sequence [28,29,43]. These methods include Saliency (i.e., Vanilla Gradients) [16,28], Integrated Gradients [42], DeepLIFT [29], DeepSHAP [32] and in-silico mutagenesis [39], among others. These techniques have been employed to validate whether models have learned features that resemble known motifs across various applications [39], including TFBS motifs [43]. Explainability methods have also been used for gene characterization and TFBS discovery [43,44].

Our method distinguishes itself from existing genomic attribution methods by exclusively using gradient information taken directly from the model to generate explanations,

thereby avoiding post-hoc modifications that diverge from the model's path [15]. As opposed to post-hoc techniques, our approach involves taking multiple samples of shifted inputs and averaging them to obtain a more robust understanding of what the model considers important in the input. While our approach is somewhat similar to SmoothGrad [34], it can be applied to any task, including those involving genomic sequences as inputs.

3. Methodology

The methodology section details the approaches used in this study. It starts with the modified DeepBIND architectures, discusses our proposed MoE model, explains the mathematics behind our novel explainability method (ShiftSmooth) and elaborates on the experimental protocol. The experiments were conducted using an NVIDIA GeForce RTX 3090 GPU with 24 GB of memory.

3.1. Modified DeepBIND Architecture

The DeepBIND architecture needs to be modified to be compatible with our proposed MoE framework. The core DeepBIND components are preserved (convolutional layer for motif detection, ReLU activation, and max/max-average pooling). Either max pooling alone OR a combination of max and average pooling concatenated together is performed. The architecture supports either configuration, and the actual choice was made through hyperparameter optimization for each expert model.

The key changes that transform DeepBIND from a standalone classifier into an expert model that can effectively contribute to our MoE ensemble are listed below.

1. A linear adjustment layer was added to project the pooled features to a consistent 32-dimensional embedding space (required for MoE integration) regardless of their internal pooling strategy. This enables integration of experts with different configurations. The original DeepBIND architecture outputs variable dimensional features depending on the pooling configuration. We chose 32 dimensions to balance representational capacity against computational efficiency. It is large enough to capture the complex patterns in TFBS motifs (which are typically 5–20 nucleotides long) while being small enough to prevent overfitting with our dataset sizes.
2. Layer normalization was added after the adjustment layer for training stability. It ensures comparable embedding magnitudes across all experts. It also addresses a critical challenge in ensemble learning. When combining embeddings from multiple independently-trained experts, the feature scales can vary dramatically, causing certain experts to dominate the gating decisions regardless of their relevance. The gating network can then make decisions based on feature patterns rather than scale differences.
3. The embedding extraction capability transforms DeepBIND from a standalone classifier into a feature extractor compatible with the MoE architecture. While standard DeepBIND produces only final predictions, our modification allows extraction of intermediate representations or embeddings that the gating network can dynamically combine based on input characteristics. This enables the model to leverage complementary strengths of different experts for improved generalization and model robustness.

3.2. Individual Experts and MoE

The proposed model is a heterogeneous, embedding-gated Mixture-of-Experts, which we refer to as HetMoE. Its expert pool is built per training factor (the seven factors of Section 3.6) from complementary backbones: a modified-DeepBIND convolutional network (ConvNet), DeepSEA, DanQ, and a fine-tuned DNABERT-6 genomic language model. Each expert is a complete classifier trained on its factor; the property HetMoE exploits is that every expert can expose a fixed-size embedding, so backbones of different architectures can

be combined in one pool. Which backbones enter the pool, and the gate settings, are chosen on in-distribution data alone (Section 3.6); the selected configuration is the pretraining-free convolutional pool of ConvNet, DeepSEA, and DanQ experts.

The MoE architecture consists of two components, the expert pool and a gating network. For MoE training the final prediction layers are removed and all expert weights remain frozen; each expert is used in “embedding mode” to produce a 32-dimensional embedding. Because the gate operates only on these embeddings, the pool can be heterogeneous: a convolutional expert produces its embedding directly from the one-hot sequence, while a pretrained trunk (DNABERT-6, DeepSEA, DanQ) is used as a frozen feature extractor whose features are reduced to the same 32-dimensional space by a lightweight linear probe (a Linear layer followed by LayerNorm). The N_e expert embeddings are concatenated and fed to the gating network, which learns to weight and dynamically combine them for the final prediction. Backbones that require a different input representation (DNABERT-6 tokenizes the sequence into overlapping k -mers) are integrated through a cached-embedding path computed once per expert, so the gating and training procedure is identical regardless of expert type. The mathematical details of the individual experts and the MoE architecture are now discussed.

3.2.1. Expert Networks

Let N_e denote the number of expert models (for the selected pool of ConvNet, DeepSEA, and DanQ experts over the seven training factors, $N_e = 21$), each providing an embedding of size $E = 32$. Each expert maps the input sequence to an embedding $\mathbf{e}_i \in \mathbb{R}^{B \times E}$ (B is the batch size): a convolutional expert computes it directly, whereas a pretrained-trunk expert applies its frozen feature extractor followed by a linear probe (a linear projection with layer normalization) onto the common E -dimensional space. This yields a set of embeddings $\{\mathbf{e}_i\}_{i=1}^{N_e}$, which are the input to the gating network.

At the mixture stage every embedding is mapped by a per-expert linear projection,

$$\mathbf{h}_i = \mathbf{e}_i \mathbf{W}_{\text{expert}_i} + \boldsymbol{\beta}, \quad (1)$$

where $H = 32$ is the projected dimension, $\mathbf{W}_{\text{expert}_i} \in \mathbb{R}^{E \times H}$ is the weight matrix of the i -th projection, $\boldsymbol{\beta} \in \mathbb{R}^{B \times H}$ is a bias matrix in which each row is the bias vector of the i -th projection denoted by $\mathbf{b}_{\text{expert}_i} \in \mathbb{R}^{1 \times H}$, and $\mathbf{h}_i \in \mathbb{R}^{B \times H}$ is the projected representation of the i -th expert. No nonlinearity is applied to $\mathbf{h}_i$; the nonlinearities in the mixture are confined to the convolutional experts’ internal activations and to the gating softmax defined below. The bias matrix is expressed as

$$\boldsymbol{\beta} = [\mathbf{b}_{\text{expert}_1}^T \mid \mathbf{b}_{\text{expert}_2}^T \mid \dots \mid \mathbf{b}_{\text{expert}_{N_e}}^T]^T \quad (2)$$

Each expert is first trained as a complete per-factor classifier so that its embedding $\mathbf{e}_i$ is informative. For the mixture, the expert’s own prediction layer is discarded and its weights are frozen, so the per-expert projections $\mathbf{W}_{\text{expert}_i}$ and the gating network are the only parameters learned at the mixture stage.

3.2.2. Gating Network

The gating network computes dynamic weights for combining the output of each expert. It takes the concatenated embeddings (denoted by $\mathbf{E}$) from all experts as input to compute the gating scores (denoted by $\mathbf{g}$) as expressed by Equations (3) and (4).

$$\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_{N_e}] \in \mathbb{R}^{B \times (E \times N_e)} \quad (3)$$

$$\mathbf{g} = \mathbf{E}\mathbf{W}_{\text{gate}} + \boldsymbol{\beta}_{\text{gate}} \in \mathbb{R}^{B \times N_e} \quad (4)$$

In Equation (4), $\mathbf{W}_{\text{gate}} \in \mathbb{R}^{(E \times N_e) \times N_e}$ is a matrix that weights the embeddings of the various experts and $\boldsymbol{\beta}_{\text{gate}} \in \mathbb{R}^{B \times N_e}$ is a bias matrix in which each row is the bias vector of the weighted embeddings denoted by $\mathbf{b}_{\text{gate}} \in \mathbb{R}^{1 \times N_e}$. The bias matrix is expressed as

$$\boldsymbol{\beta}_{\text{gate}} = [\mathbf{b}_{\text{gate}}^T \mid \mathbf{b}_{\text{gate}}^T \mid \dots \mid \mathbf{b}_{\text{gate}}^T]^T \quad (5)$$

The gating weights (denoted by $\boldsymbol{\alpha}$) give the relative emphasis of the output of each expert and are obtained using the softmax function as given by

$$\boldsymbol{\alpha} = \text{softmax}(\mathbf{g}) \in \mathbb{R}^{B \times N_e}. \quad (6)$$

3.3. Mixture Output and Prediction

The final output of the MoE model is a weighted sum of the outputs of each expert. This final mixture output $\mathbf{m}$ is calculated as

$$\mathbf{m} = \sum_{i=1}^{N_e} \boldsymbol{\alpha}_{:,i} \odot \mathbf{h}_i \in \mathbb{R}^{B \times H}, \quad (7)$$

where $\boldsymbol{\alpha}_{:,i} \in \mathbb{R}^{B \times 1}$ is a column vector that represents the gating weights for the i -th expert across the batch, $\mathbf{h}_i \in \mathbb{R}^{B \times H}$ is the projected representation of the i -th expert, and $\odot$ denotes element-wise multiplication broadcasted over the batch dimension. This multiplication is performed by taking the j th element of $\boldsymbol{\alpha}_{:,i}$ and multiplying it by every element in the j th row of $\mathbf{h}_i$. The mixture output $\mathbf{m}$ is then passed through a classifier that applies a weighting ($\mathbf{W}_{\text{class}}$) and a bias ($\mathbf{b}_{\text{class}}$) to get an overall output $\mathbf{o}$ as given by

$$\mathbf{o} = \mathbf{m}\mathbf{W}_{\text{class}} + \mathbf{b}_{\text{class}} \in \mathbb{R}^{B \times 1}, \quad (8)$$

where $\mathbf{W}_{\text{class}} \in \mathbb{R}^{H \times 1} = \mathbb{R}^{32 \times 1}$ and $\mathbf{b}_{\text{class}} \in \mathbb{R}^{B \times 1}$. The final prediction ($\hat{\mathbf{y}}$) is obtained by applying the sigmoid activation function (σ) to $\mathbf{o}$ as given by

$$\hat{\mathbf{y}} = \sigma(\mathbf{o}) \in \mathbb{R}^{B \times 1}. \quad (9)$$

3.4. Training Procedure

The training procedure for this methodology is divided into two distinct stages, namely, the training of each individual expert model and the training of the MoE model. The performance of all models is assessed using the ROC curve and its corresponding AUC score.

3.4.1. Expert Model Training

The expert models are trained with hyperparameter optimization. Each CNN expert is optimized using Optuna (version 4), an automated hyperparameter optimization framework [45]. This process aims to maximize the validation AUC score obtained from the ROC thereby identifying the optimal set of hyperparameters for each expert. The hyperparameter optimization process can be formulated as:

$$\theta^* = \arg \max_{\theta} \text{AUC}_{\text{val}}(f(\theta)) \quad (10)$$

where θ^* represents the optimal hyperparameters, θ denotes the hyperparameters, $f(\theta)$ represents the CNN model with hyperparameters θ , and AUC_{val} is the validation AUC score.

Once the optimal hyperparameters are determined, the CNN experts are trained using these parameters. During this training phase, early stopping is employed to prevent overfitting. This technique monitors the validation performance. If no improvement is observed for a specified number of epochs (set to 5 in this case since the models are individually fine-tuned and optimized for a specific TFBS pattern), the training is halted. This ensures that the model does not continue to train on data that does not improve its performance, thereby conserving computational resources and preventing overfitting.

The heterogeneous experts are prepared differently. A pretrained trunk (DNABERT-6, and DeepSEA/DanQ in the broader framework) is used as a frozen feature extractor: it is run once per training factor and its features are reduced to the common $E = 32$ embedding by a lightweight linear probe (a linear projection followed by layer normalization, with a classification head) trained on that factor while the trunk stays frozen. DNABERT-6 is fine-tuned once per training factor before being frozen. Because some trunks require a different input representation (DNABERT-6 tokenizes the sequence into overlapping 6-mers), every expert's E -dimensional embedding is precomputed once, so the gate is trained and evaluated on the same embedding representation regardless of expert type.

3.4.2. MoE Model Training

The MoE model is trained by generating embeddings from the frozen experts (the ConvNet experts and the probed pretrained trunks), keeping their weights frozen. The input data is passed through each expert model and the resulting embeddings are concatenated to form a joint representation of the input data.

The MoE model is then trained on these concatenated embeddings using Stochastic Gradient Descent (SGD) with Nesterov momentum and binary cross-entropy loss [46]. The SGD optimization is defined as

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t) \quad (11)$$

where θ_t represents the model parameters at iteration t , η is the learning rate, $L(\theta_t)$ is the loss function, and $\nabla L(\theta_t)$ is the gradient of the loss function with respect to the parameters. The Nesterov momentum is given as

$$\mathbf{v}_{t+1} = \mu \mathbf{v}_t + \eta \nabla L(\theta_t - \mu \mathbf{v}_t) \quad (12)$$

where μ is the momentum term and $\mathbf{v}_t$ is the velocity vector at iteration t .

Early stopping is again applied, but in this case with 10 epochs to ensure that the training is halted if no improvement in validation performance is observed. A patience of 10 epochs allows more flexibility during training, as it adjusts weights across multiple experts and refines the gating mechanism for optimal collaboration.

3.4.3. Implementation Details

The MoE model and expert networks are implemented using PyTorch (version 2.8.0). Training is conducted on GPUs to expedite computation. The MoE model is trained for a maximum of 500 epochs, with early stopping on the in-distribution validation AUC. The initial learning rate is set to 0.01, and the momentum is set to 0.98. The per-expert hyperparameters are obtained by automated hyperparameter search maximizing the validation AUC. The gating network is trained by mini-batch stochastic gradient descent with Nesterov momentum (learning rate 0.01, momentum 0.98, batch size 256), with the experts frozen, and the weights at the best in-distribution validation AUC are retained. Two stabilizers act on the gate. First, each per-expert embedding block is L_2 -normalized before the gating layer, so that an expert with a larger embedding norm cannot dominate the

gating scores and collapse the mixture onto itself. Second, a gate-entropy term (coefficient 10^{-3}) is added to the binding cross-entropy loss to discourage the gate from collapsing onto a single expert. For reproducibility a single fixed random seed is reused for every train/validation split and deterministic GPU kernels are used; all experiments were run on an NVIDIA RTX 3090 with 24 GB of memory. The per-expert hyperparameters are listed in Table 2.

Table 2. Per-expert hyperparameters for the convolutional (ConvNet) experts, obtained by automated Optuna search maximizing the in-distribution validation AUC. Because every expert is frozen during mixture training and only the gating network is selected (on in-distribution data), the experts do not require per-factor tuning: we search hyperparameters for three representative training factors (ARID3A, FOXM1, GATA3; shown here) and the remaining four factors (JUND, MAX, GABPA, SP1) reuse the ARID3A configuration as a shared default. The DeepSEA, DanQ, and DNABERT-6 experts use their standard training settings.

Expert	PoolType	NeuType	Dropprob	SigmaConv	SigmaNeu	Learning_Rate	Momentum
ARID3A	maxavg	nohidden	0.75	1.31×10^{-7}	3.04×10^{-3}	5.39×10^{-3}	0.9698
FOXM1	maxavg	hidden	0.75	2.68×10^{-6}	2.20×10^{-5}	5.22×10^{-3}	0.9740
GATA3	max	hidden	0.75	1.18×10^{-4}	1.14×10^{-4}	2.12×10^{-3}	0.9776

3.5. Experimental Setup

A systematic evaluation framework is implemented alongside a visual explainability analysis to assess the performance of the proposed HetMoE against the experts in its pool and the baseline models. This framework is designed to measure the AUC for each model across multiple test datasets and perform a statistical analysis. This provides a quantitative assessment of the models' generalizability and enables comparison across multiple TFBS datasets. For each test set, all models are evaluated, and their corresponding AUC scores are recorded.

The AUC metric is chosen for its ability to provide a single measure of the classifier's performance across all classification thresholds, thus capturing the trade-off between true positive and false positive rates. These models are then analyzed using multiple attribution mapping techniques.

3.5.1. Generating the AUC

In classifier mode, the expert models and MoE both output a continuous binding score (discussed in Section 2) in the range [0,1]. This score represents the predicted probability or strength of TF-DNA binding with higher scores indicating stronger predicted binding affinity.

The classification task involves the determination of the following:

- True Positive (TP): Model correctly predicts that a DNA sequence contains a TFBS (predicted = 1, actual = 1).
- False Positive (FP): Model incorrectly predicts a TFBS is present when it is actually absent (predicted = 1, actual = 0).
- True Negative (TN): Model correctly predicts no TFBS is present (predicted = 0, actual = 0).
- False Negative (FN): Model misses a TFBS that is actually present (predicted = 0, actual = 1).

The ground truth labels come from ChIP-seq experimental data in the ENCODE database, where sequences are labeled as 1 (binding site present) or 0 (no binding site).

The ROC curve is generated as given below, from which the AUC is calculated.

- The continuous binding scores produced by each model are compared against the ground truth labels using a standard ROC implementation (the `roc_curve` and `roc_auc_score` functions of scikit-learn [47]), which sweeps the decision threshold across all distinct values present in the prediction scores rather than a fixed set of discrete steps.
- At each threshold, sequences with scores that are greater than or equal to the threshold are classified as 1 and compared to the ground truth.
- For each threshold, we calculate the TPR as the ratio of TP to (TP + FN) and the FPR as the ratio of FP to (FP + TN).
- The ROC curve is a plot of TPR versus FPR across all thresholds, and the AUC is the area under this curve, computed directly from the continuous scores.

The TPR is the fraction of actual binding sites correctly identified. The FPR is the fraction of non-binding sites incorrectly classified.

3.5.2. Model Loading and Evaluation

Each CNN expert model is first evaluated using a test loader that provides batches of ChIP-seq data. The evaluation process involves generating predictions for each batch and calculating the AUC score based on these predictions. This process ensures that each expert model's performance is rigorously assessed using the same evaluation criteria.

3.5.3. Bootstrap Method for AUC Estimation

Each model is evaluated on the held-out test partition of every transcription factor, which is a balanced set of 500 experimentally determined positive (bound) sequences and 500 negative (background) sequences. Inference is deterministic: dropout is disabled at evaluation time so that each model produces a single, fixed prediction per sequence. To quantify the uncertainty of the resulting AUC, bootstrapping [48] is then employed: we resample the test set with replacement $B = 1000$ times and recompute the AUC of every model on each replicate. Within each replicate the same resampling index is applied to all models, so that the model-versus-model comparisons are paired. This yields, for every model, a distribution of $B = 1000$ AUC values from which we report the mean AUC and a 95% percentile confidence interval, and, for every contrast (e.g., HetMoE versus DNABERT-6, or HetMoE versus an expert), the distribution of the paired AUC difference and its 95% percentile interval.

3.5.4. Statistical Analysis

To determine if the performance differences are statistically significant, we use a one-way ANOVA [19] together with the per-model paired bootstrap reported in Section 4. These tests compare HetMoE against the experts in its pool and the baselines, and quantify any performance improvements achieved by integrating multiple experts within the gated framework.

The one-way ANOVA test is based on a null and alternative hypothesis [19].

1. The null hypothesis states that there is no significant difference between all the models tested. In this case, the average AUC for all models is essentially equal, their 95% confidence intervals overlap, and the p -value is greater than 0.05.
2. The alternative hypothesis states that there is at least one model with an average AUC that is significantly different than the others tested. The 95% confidence interval of at least one model does not overlap with the others, and the p -value is less than 0.05.

The paired percentile bootstrap ($B = 1000$ shared resamples) is the primary basis for inference: for every contrast it gives the distribution of the paired AUC difference and its 95% interval, which respects the fact that all models are scored on the same sequences.

The one-way ANOVA is reported as a global omnibus over the per-model bootstrap AUC distributions; because bootstrap replicates are resamples of a single test partition rather than independent observations, the ANOVA p -value is not interpreted on its own, and the per-factor paired-bootstrap intervals are the basis for every comparison reported in Section 4.

3.6. Training Factors, Out-of-Distribution Stratification, and Negative Sets

Training factors and cell-line control. The experts and the gating network are trained on seven transcription factors drawn from the ENCODE ChIP-seq corpus and chosen to span seven distinct DBD families: ARID3A (ARID/Bright), FOXM1 (forkhead), GATA3 (GATA-type zinc finger), JUND (bZIP), MAX (bHLH-Zip), GABPA (ETS), and SP1 (C2H2 zinc finger). Spanning multiple families lets the gate combine experts with genuinely complementary sequence preferences rather than redundant variants of a single motif. To prevent the train/OOD boundary from being confounded with cell type, four of the seven training factors and the majority of the held-out factors are taken from the K562 cell line. The exact experiment selected for each factor (cell line, antibody, laboratory) is specified in the metadata released with the dataset.

Out-of-distribution stratification. We do not treat “out-of-distribution” as a single undifferentiated pool. Predicting ChIP-seq occupancy from local sequence is only well posed for factors that bind a defined motif; several factors in the OOD set (the RNA-polymerase-II subunit POLR2A and the chromatin-cofactor subunits RBBP5 and SAP30) have no intrinsic DNA motif and are recruited through protein–protein interactions, so their occupancy carries a low sequence-prediction ceiling by construction. We therefore evaluate three explicitly separated strata: (i) within-family held-out factors that share a DBD family with a training factor (17 factors, e.g., JUN and FOSL1 for JUND, MYC and USF1 for MAX, ELF1 and ETS1 for GABPA, SP2 and YY1 for SP1, GATA1/GATA2 for GATA3), which test genuine motif-family transfer; (ii) cross-family held-out factors from families absent during training but still sequence-specific (CTCF, STAT3, NRF1, HNF4A, TCF7L2, ZBTB33); and (iii) a non-motif stratum (POLR2A, EP300, EZH2, TAF1, RBBP5, SAP30) that is reported separately and never folded into the headline mean. The headline OOD result is the mean over the motif-bearing strata (within- and cross-family). This stratification prevents a single easy, motif-rich factor from dominating a pooled average.

Negative sets. Positives are experimentally bound sequences. A common protocol generates the training negatives by dinucleotide shuffling of the positives. As recent work demonstrates [49,50], a classifier trained against shuffled negatives can learn to detect the shuffling artifact rather than binding; and because evaluation uses curated genomic negatives, this creates a train/evaluation distribution mismatch that inflates absolute scores and can re-order model comparisons. We confirm this mismatch with a simple diagnostic (a dinucleotide detector separates the two negative types at AUC 0.73 with matched GC content) and therefore train all experts on GC- and repeat-matched real genomic negatives sampled from the hg19 assembly (a “null-sequence” protocol in the spirit of [51]). The dinucleotide-shuffle configuration is reported as an explicit negative-set ablation, and the effect is large: under shuffled negatives the apparent motif-OOD margin over DNABERT-6 inflates to +0.066 (mean 0.864, superior on 22/23 factors), versus +0.027 (mean 0.827, 13/23) under fair GC-matched negatives; the standard protocol therefore materially overstates the comparison, and we report the fair-negative result as our headline. In every arm, all evaluation uses the balanced curated test sets of 500 experimentally determined positives and 500 experimental negatives per factor. The expert pool and gate settings are selected by the in-distribution mean AUC over the seven training factors (their held-out test partitions); the OOD strata are never consulted during selection and are scored once.

3.7. ShiftSmooth Gradient Formulation

To understand which features matter most to the model, we perform an explainability analysis. To do this, we use the Vanilla Gradient along with our novel gradient-based approach, the ShiftSmooth Gradient, which strengthens the robustness of such analysis. The ShiftSmooth method, detailed in Section 2.2.3, is an attribution method that leverages gradients. We use this section to further elaborate on the mathematics behind ShiftSmooth. We define the gradient in Equation (13) as

$$A_c(\mathbf{x}) = \frac{\partial S_c(\mathbf{x})}{\partial \mathbf{x}}. \quad (13)$$

Here, $A_c(\mathbf{x})$ represents the model gradient at the input, while $S_c(\mathbf{x})$ denotes the output score for the class of interest c , given $\mathbf{x}$ as the model input. The ShiftSmooth method extends the Vanilla Gradient by computing an average over multiple shifted inputs. The data shifting process is described in Equation (14) as

$$\mathbf{x}_n = \mathbf{x} \gg n, \quad (14)$$

where $\gg$ represents a right bit-shift operation. In Equation (14), $\mathbf{x}_n$ is the shifted input data, and n indicates the number of places the data is shifted. To limit information loss, elements shifted beyond the final position are reintroduced at the first position (circular shift). The ShiftSmooth Gradient is then calculated by averaging the gradient of all circularly shifted input sequences. To align each individual gradient with the original non-shifted input sequence, the resulting gradient is circularly shifted back n spaces. This is illustrated in Equation (15) as

$$\hat{A}_c(\mathbf{x}) = \frac{1}{2N+1} \sum_{n=-N}^N A_c(\mathbf{x}_n) \ll n, \quad (15)$$

where $\ll$ indicates a left bit-shift operation. The input sequence is shifted backward starting from $n = -N$, and forward up to $n = N$. At $n = 0$, no shifting occurs, resulting in the standard non-shifted gradient. The final attribution map ($\hat{A}_c(\mathbf{x})$), obtained by averaging all these gradients, provides a robust explanation of the model's focus in the sequence on average, accounting for the variability of the viewing window in real-world data.

The process of generating ShiftSmooth Gradients is detailed in Figure 2. This figure illustrates four steps: shifting the input (Equation (14)), calculating the model gradient (Equation (13)), shifting the gradients back, and taking the average (Equation (15)).

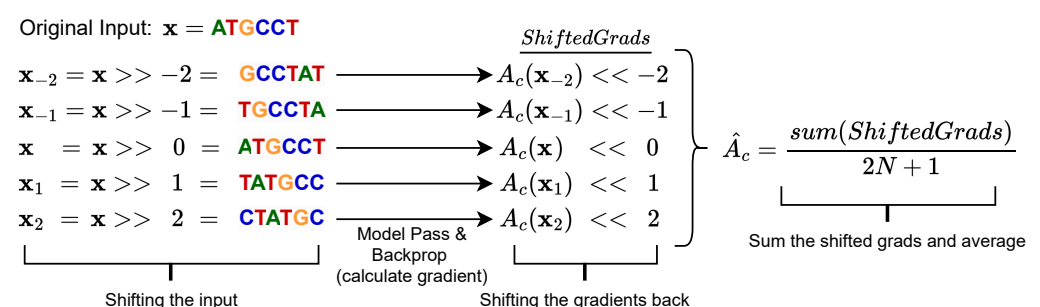


Figure 2. Illustration of the ShiftSmooth attribution method. Starting from the left, the input sequence is shifted by a small set of integer translations and each shifted copy is passed through the model to compute the gradient. The resulting gradients are shifted back to realign them to the original nucleotide positions, and are then averaged to produce the ShiftSmooth map $\hat{A}_c$. Because every shifted input remains a valid one-hot sequence, the average stays on the data simplex, unlike Gaussian-noise smoothing. Colors denote the four nucleotides following the standard A/C/G/T scheme.

First, the input sequence is shifted as shown in Equation (14) before being passed through the model. Next, the gradient is computed via backpropagation, resulting in A_c as depicted in Equation (13). Then, the gradients are shifted to ensure that the attributed features correspond to the positions of the original input data, as shown on the right side of Equation (15). Finally, the shifted gradients are averaged to produce the resulting ShiftSmooth Gradient map $\hat{A}_c$.

4. Performance Evaluation Results

The performance of HetMoE, the experts in its pool, and the baselines is discussed separately for the in-distribution and OOD datasets. The in-distribution datasets refer to the data whose distribution is seen during training. The actual in-distribution test samples are different from what was used for training; each expert is trained on a single factor (e.g., the ARID3A expert on ARID3A data). Once trained, every expert generates an embedding vector, which the gating network of HetMoE weights and combines for the final prediction; the gate has no access to the raw TF data beyond these embeddings. The seven in-distribution training factors each come from a distinct DBD family (Section 3.6). The held-out OOD factors are evaluated in three explicitly separated strata (within-family, cross-family, and non-motif), so that “out-of-distribution” is not treated as a single undifferentiated pool. The headline OOD result is the mean over the motif-bearing strata (within- and cross-family); the non-motif factors are reported separately because their occupancy has a low sequence-prediction ceiling by construction.

To rigorously evaluate the significance of the performance differences between HetMoE and the models it is compared against, we conduct a one-way ANOVA [19] test and the paired bootstrap reported below. This statistical analysis provides a quantitative assessment of the observed differences and helps determine whether they reflect genuine improvements in predictive power or are merely due to chance.

4.1. Model Performance on In-Distribution Data

We compare the proposed HetMoE against the best single expert in its pool, a static (uniform) average of the same experts, and the strongest established baseline, a fine-tuned DNABERT-6, on the seven in-distribution datasets. Figure 3 shows the per-factor in-distribution AUC with 95% bootstrap confidence intervals. All AUCs use deterministic inference and the $B = 1000$ paired bootstrap on the balanced curated test sets.

HetMoE attains the highest in-distribution AUC (mean 0.881), above the best single expert in its pool (0.747), the fine-tuned DNABERT-6 baseline (0.818), and a static average of the same experts (0.796). It posts the highest per-factor mean AUC on all seven factors, tying the fine-tuned DNABERT-6 on MAX. Its lead is clearest on JUND and GABPA, where the 95% bootstrap confidence interval is fully separated from all three baselines. On the remaining factors the HetMoE interval still excludes two of the three baselines but overlaps the single runner-up DanQ on ARID3A, FOXM1, and GATA3, and DNABERT-6 on MAX and SP1; the paired bootstrap against DNABERT-6 (stars, Figure 3) nonetheless favors HetMoE on every factor except MAX, where the two are statistically indistinguishable. Input-dependent gating over the heterogeneous pool thus improves on any single backbone even on factors seen during training, with no single baseline acting as a consistent runner-up.

HetMoE thus provides the strongest in-distribution predictor. The central claim of this work is that the same model also generalizes best to OOD factors, which we examine next.

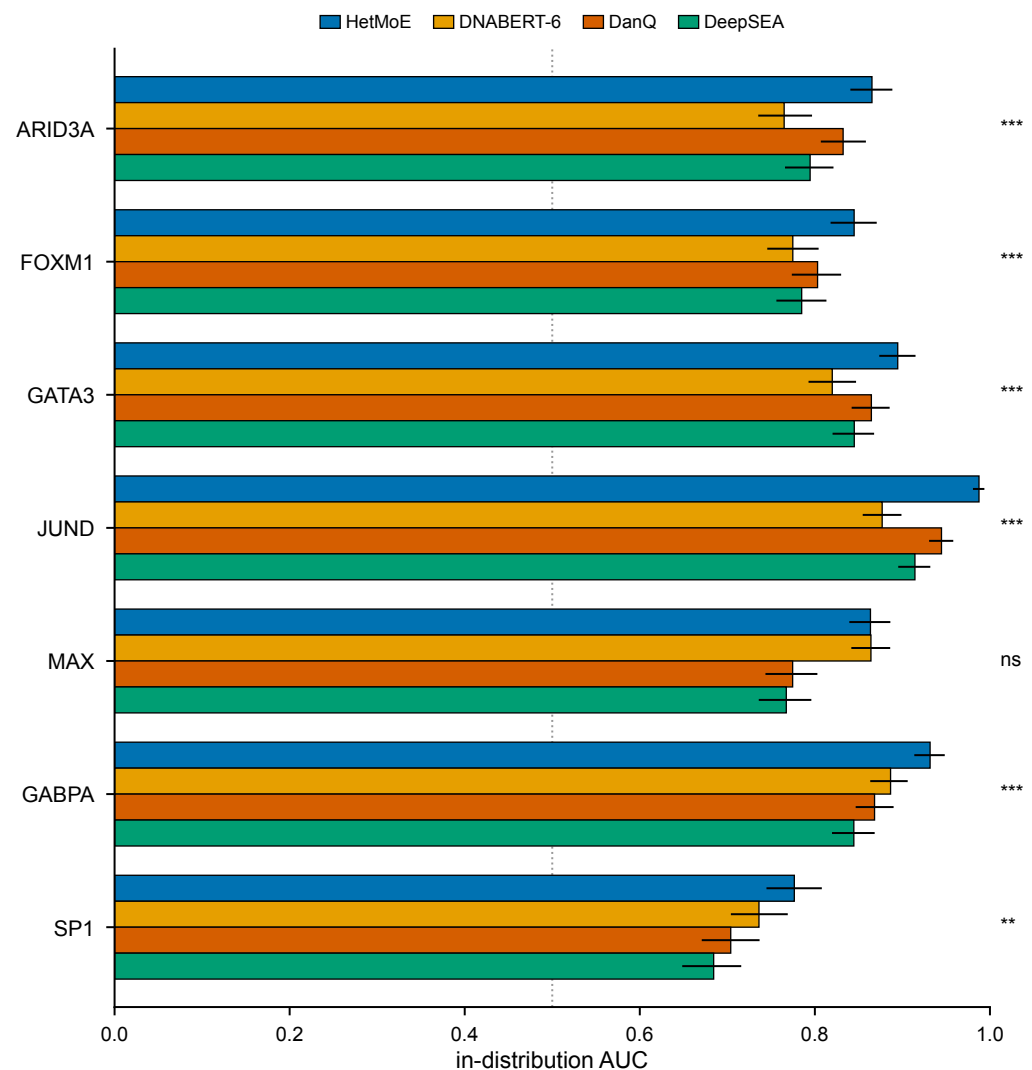


Figure 3. Per-factor in-distribution AUC (bar = mean, whisker = 95% bootstrap CI) for HetMoE and the same-generation baselines (fine-tuned DNABERT-6, DanQ, DeepSEA) across the seven training factors. Stars give the HetMoE vs. DNABERT-6 paired-bootstrap significance (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$, ns otherwise). Full 0–1 axis; deterministic inference on the balanced curated test sets.

4.2. Model Performance on Out-of-Distribution Data

Figure 4 shows the per-factor HetMoE – DNABERT-6 Δ AUC with 95% paired-bootstrap confidence intervals across the 23 motif-bearing OOD factors. These factors are held out entirely from training and probe how well each model generalizes to novel TFs and genomic contexts. We report results per stratum so that an easy, motif-rich factor cannot dominate a pooled average.

On the motif-bearing OOD strata (within- and cross-family), HetMoE attains a mean AUC of 0.827, versus 0.800 for the strongest baseline, fine-tuned DNABERT-6 (a +0.027 margin). Broken down by stratum, HetMoE reaches 0.846 on the within-family factors (genuine motif-family transfer), 0.775 on the harder cross-family factors, and 0.696 on the non-motif factors, which we report separately because they sit near a sequence-only ceiling, where DNABERT-6 is the stronger model (0.756 vs. 0.696). Gating over the heterogeneous pool exceeds a static mean of the same experts by +0.073 AUC (0.827 vs. 0.754) and the best single expert by +0.088 (0.827 vs. 0.739), strong evidence that the gain comes from input-dependent routing, not averaging; indeed, the in-distribution-selected pool is a pretraining-free heterogeneous set of convolutional experts (ConvNet, DeepSEA, DanQ) that nonetheless surpasses the fine-tuned DNABERT-6 baseline.

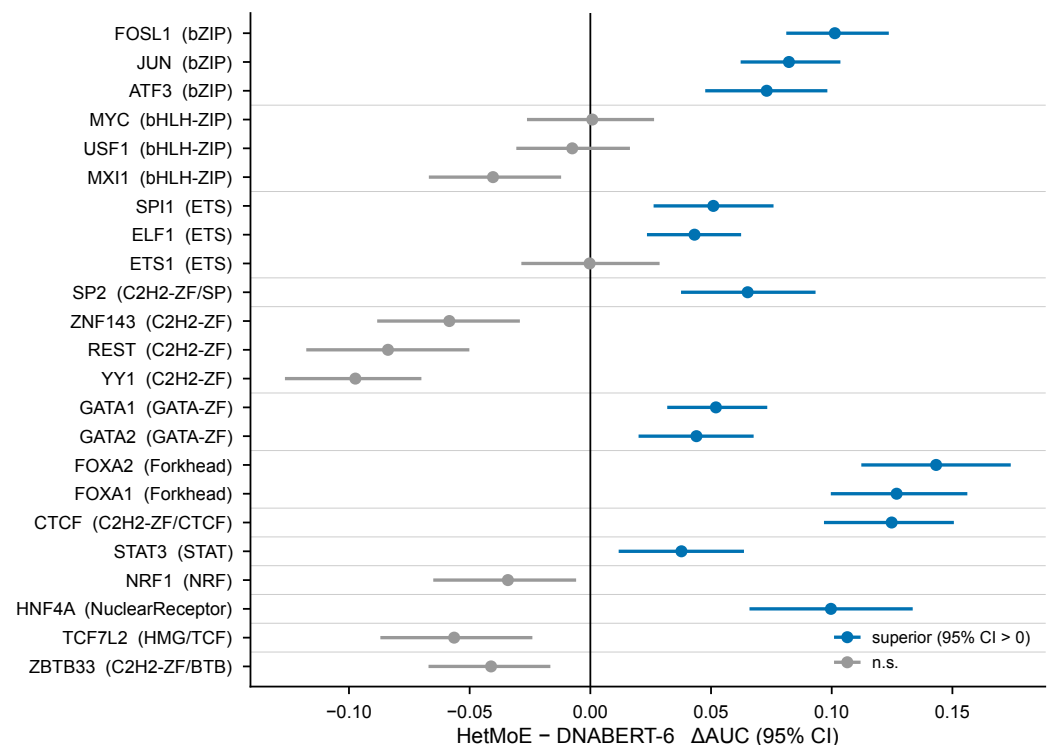


Figure 4. Per-factor HetMoE – DNABERT-6 Δ AUC with 95% paired-bootstrap confidence intervals for the 23 motif-bearing out-of-distribution factors, grouped by DNA-binding-domain family. The interval excludes zero in favor of HetMoE on 13 of the 23 factors (blue), with the largest gains on the sequence-specific factors (FOXA2, FOXA1, CTCF, FOSL1, HNF4A).

The per-factor confidence intervals (Figure 4) identify the fine-tuned DNABERT-6 as the strongest competitor, which we test directly with a paired bootstrap (identical resamples per replicate).

Against DNABERT-6, HetMoE is significantly superior on 13 of the 23 motif-bearing OOD factors (the 95% confidence interval of the HetMoE – DNABERT-6 AUC difference excludes zero). The advantage is concentrated, as the biology predicts, on the sequence-specific and within-family factors: FOXA2 (+0.143), FOXA1 (+0.127), CTCF (+0.125), FOSL1 (+0.101), and HNF4A (+0.100), among others (Figure 4). On several zinc-finger factors (YY1, ZNF143, REST, ZBTB33) the fine-tuned DNABERT-6 is stronger, which indicates where genome-scale pretraining helps most once the dinucleotide-shuffle artifact is removed. Because the gate composes complementary backbones, it matches or exceeds whichever backbone suits a given factor without being told in advance which one that is; this per-input weighting, not a fixed average, produces the gain. HetMoE gives the most stable performance across the tested settings, generalizing both to held-out members of the trained families and to entirely unseen families.

4.3. Comparison with State-of-the-Art Methods

To position HetMoE against established TFBS predictors, its heterogeneous pool contains DeepSEA, DanQ, and a fine-tuned DNABERT-6 expert per training factor (alongside the modified-DeepBIND ConvNet), all trained on the identical data and evaluated with the same protocol (balanced curated test sets, deterministic inference, $B = 1000$ paired bootstrap). Table 3 reports, per stratum, the mean AUC for HetMoE, the standalone DeepSEA, DanQ, and fine-tuned DNABERT-6 baselines (each mean-ensembled across the training factors), a static (uniform) average of the pool, and the best single expert: HetMoE attains the best in-distribution and motif-OD means and surpasses all of them. Fine-tuned DNABERT-6 is the strongest baseline (motif-OD 0.800, versus 0.789 for DanQ and 0.774

for DeepSEA), so we use it as the reference for the per-factor and multi-seed analyses (Section 4.4, Figures 4 and 5). BpNet is not included as an AUC row because it predicts base-resolution binding profiles (regression) rather than a binary call, and is therefore not directly comparable on this metric.

Table 3. Mean AUC ($B = 1000$ paired bootstrap) on the balanced curated test sets for the proposed HetMoE, the established same-generation baselines (DeepSEA, DanQ, and fine-tuned DNABERT-6, each mean-ensembled across the training factors), a static (uniform) average of the pool, and the best single expert. ID = in-distribution mean (seven training factors). The OOD columns are the per-stratum means: within-family (17 held-out family members), cross-family (6 unseen-family factors), non-motif (6 recruited/general factors), and the motif-OOD mean (within + cross). HetMoE is best on the in-distribution and every motif-bearing OOD stratum; on the non-motif stratum (reported separately, near the sequence-only ceiling) DNABERT-6 is marginally higher. Experts are trained against GC-matched real genomic negatives.

Model	ID Mean	Within-Family	Cross-Family	Non-Motif	Motif-OOD Mean
HetMoE (ours)	0.881	0.846	0.775	0.696	0.827
DNABERT-6 (baseline)	0.818	0.816	0.753	0.756	0.800
DanQ (baseline)	0.827	0.805	0.746	0.702	0.789
DeepSEA (baseline)	0.805	0.792	0.724	0.693	0.774
Static average	0.796	0.770	0.707	0.675	0.754
Best single expert	0.747	0.754	0.693	0.654	0.739

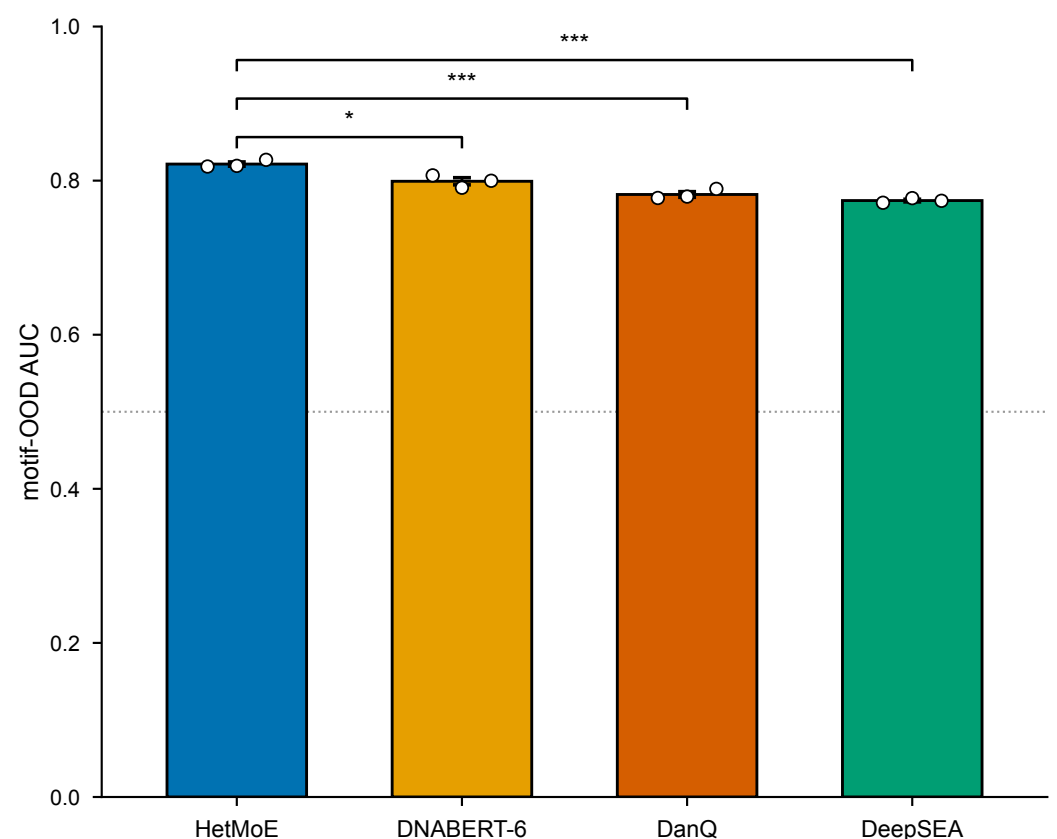


Figure 5. Motif-bearing out-of-distribution AUC for HetMoE and the same-generation baselines (bar = mean over three genomic seeds, error bar = s.d., points = individual seeds). HetMoE (0.821 ± 0.005) exceeds fine-tuned DNABERT-6 (0.799 ± 0.008), DanQ (0.782 ± 0.006), and DeepSEA (0.774 ± 0.003). Brackets give the HetMoE vs. baseline significance from a paired bootstrap over the motif-OOD factors (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$).

4.4. Multi-Seed Robustness and Routing

The expert pool and gate settings are selected by the in-distribution mean AUC over the seven training factors, and the OOD strata are scored once. The selected pool is the pretraining-free convolutional set (ConvNet, DeepSEA, DanQ; 21 experts); this conclusion is robust to the selection rule, as choosing instead by the in-distribution validation split favors the full 28-expert pool, which still surpasses fine-tuned DNABERT-6 on the motif-ODD mean (0.818 vs. 0.800). On the canonical seed the per-factor paired bootstrap (identical resamples) finds HetMoE significantly superior to fine-tuned DNABERT-6 on 13 of the 23 motif-bearing factors, with a mean margin of +0.029 on the within-family stratum (10 of 17 factors superior) and +0.022 on the cross-family stratum (3 of 6; Figure 4), the advantage concentrated on the sequence-specific factors. Across three random seeds (each re-training all experts on the GC-matched genomic negatives and re-selecting the configuration on the in-distribution mean AUC), the fair-negative motif-ODD mean is 0.821 ± 0.005 for HetMoE versus 0.799 ± 0.008 for DNABERT-6, a margin of +0.022 that is positive in every seed (range +0.011 to +0.028); the in-distribution selection chooses the same convolutional pool in all three seeds, and HetMoE is the more stable model (s.d. 0.005 vs. 0.008). Finally, the gate does not collapse onto a single expert but routes inputs in a factor-dependent way (Figure 6), the mechanism behind the gain. The improvement is therefore consistent though modest under fair negatives.

Figures 4–6 visualize these results on a full 0–1 AUC axis: the per-factor paired-bootstrap significance against DNABERT-6 (Figure 4); the multi-seed robustness (Figure 5); and the input-dependent gate routing that drives the gain (Figure 6). Throughout, significance is a paired percentile bootstrap ($B = 1000$, shared resamples) of HetMoE – DNABERT-6 (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$, ns otherwise).

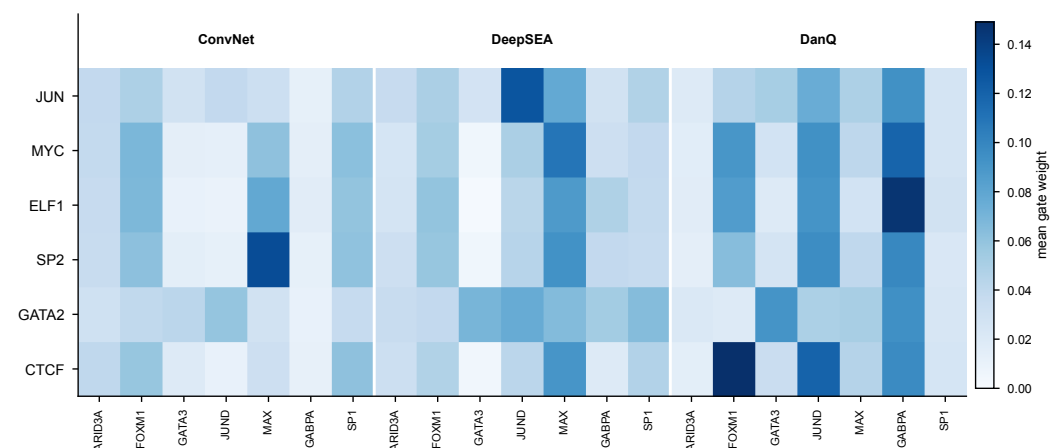


Figure 6. Mean gate weight for each expert in the in-distribution-selected pool (columns: the ConvNet, DeepSEA, and DanQ experts over the seven training factors, grouped by backbone) across representative out-of-distribution factors (rows). The gate routes inputs in a factor-dependent way rather than collapsing onto a single expert.

4.5. Ablation Studies

We ablate the principal design choices. The embedding-size and expert-freezing studies (Tables 4 and 5) are controlled comparisons on a convolutional reference pool that vary one factor at a time; their absolute AUCs are accordingly lower than the selected heterogeneous configuration, and only the trend within each table is meaningful. The expert-pool and gating study (Table 6) uses the fair-negative protocol and the family-stratified evaluation of the headline result, so its values are directly comparable to what was presented in Section 4.

Embedding size. Table 4 sweeps the expert embedding dimension $E \in \{16, 32, 64, 128\}$ on the convolutional reference pool. The OOD mean varies by under 0.01 AUC across the range, so we use $E = 32$, the smallest dimension with no measurable loss.

Table 4. Ablation: expert embedding dimension E (convolutional reference pool; only the within-table trend is meaningful).

E	In-Dist Mean AUC	OOD Mean AUC
16	0.6874	0.6614
32	0.6806	0.6576
64	0.6965	0.6559
128	0.6882	0.6626

Frozen vs. fine-tuned experts. Table 5 compares freezing the experts during mixture training against fine-tuning them, on the convolutional reference pool. Freezing matches fine-tuning to within the bootstrap noise on both in-distribution and OOD data while training roughly two orders of magnitude faster and with far fewer trainable parameters, which is why the experts are frozen throughout.

Table 5. Ablation: frozen vs. fine-tuned experts (convolutional reference pool). “MoE train (s)” is the mixture-training wall-clock; freezing is far cheaper at comparable AUC.

Experts	In-Dist	OOD	MoE Train (s)	Trainable Params
frozen	0.6806	0.6576	5.5	3492
fine-tuned	0.7041	0.6554	497.7	11,047

Expert pool and gate stabilizers. Table 6 varies which backbones enter the pool and whether the two gate stabilizers are used, with selection and scoring performed exactly as for the headline result. A convolutional-only pool generalizes poorly (motif-OOD mean 0.673); adding the DeepSEA and DanQ trunks raises it to 0.827 and is the configuration chosen by in-distribution AUC. Adding the fine-tuned DNABERT-6 experts (the full 28-expert pool) does not improve the mean further (0.818), so the selected pool is pretraining-free. Removing the two gate stabilizers from the full pool lowers both means and collapses the gate’s mean entropy from 2.87 to 0.98 nats (the gate concentrates on a handful of experts), confirming that the L_2 -normalized embeddings and the entropy term keep the mixture from collapsing.

Table 6. Ablation: expert pool and gate stabilizers, under the fair-negative protocol and family-stratified evaluation (selected and scored as for the headline result). N_e is the number of experts and “Gate H ” is the gate’s mean per-input entropy in nats. The convolutional pool chosen by in-distribution AUC is in bold.

Pool/Setting	N_e	In-Dist	Motif-OOD	Within	Gate H
ConvNet only	7	0.720	0.673	0.686	1.71
DNABERT-6 only	7	0.805	0.725	0.753	1.57
ConvNet + DNABERT-6	14	0.817	0.743	0.766	2.37
ConvNet + DeepSEA + DanQ	21	0.881	0.827	0.846	2.66
+ DNABERT-6 (full pool)	28	0.875	0.818	0.839	2.87
full pool, no stabilizers	28	0.855	0.800	0.819	0.98

5. Model Explainability

This section presents a comparative analysis of the explainability results derived from the Vanilla Gradient and ShiftSmooth methods. The study evaluates the attributions gener-

ated by these methods when applied to individual expert models and MoE architecture models. By examining the differences in the explanations provided by Vanilla Gradients and ShiftSmooth, we aim to elucidate the strengths and weaknesses of each approach in the context of both model types. For the panels in Figure 7, the x-axis corresponds to the nucleotide position within the sequence and the y-axis represents the importance scores attributed to each nucleotide.

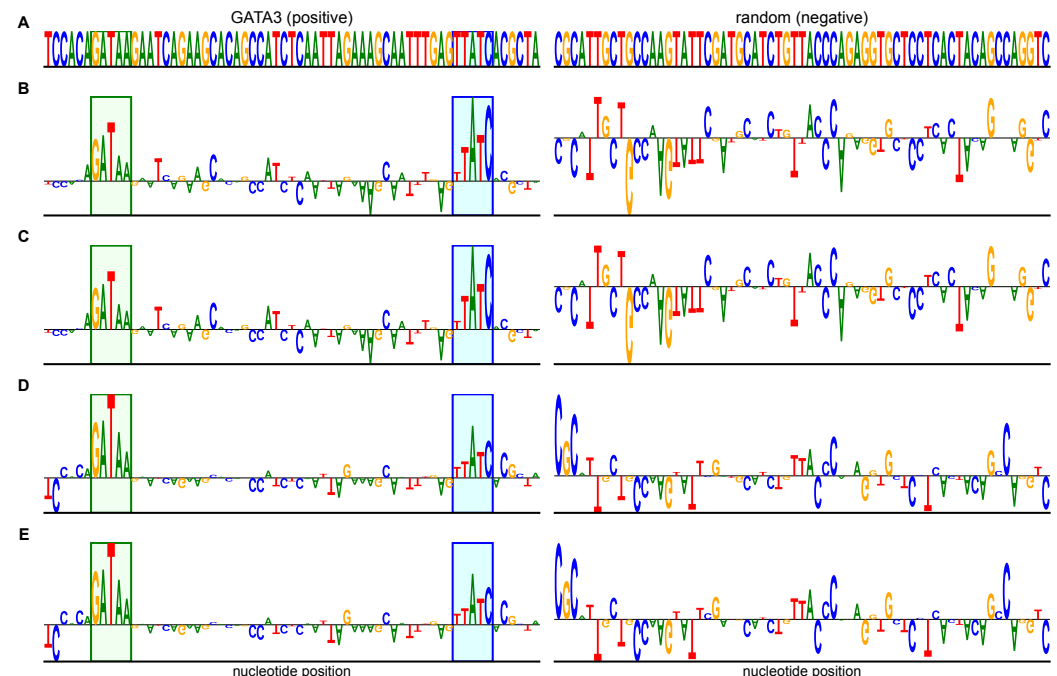


Figure 7. ShiftSmooth attribution for a GATA3-positive sequence (left column) and a random negative (right column), zoomed to the motif region. Rows are labeled (A–E): (A) the input one-hot sequence; (B) Vanilla Gradient and (C) ShiftSmooth attributions for the GATA3 ConvNet expert; (D) Vanilla Gradient and (E) ShiftSmooth attributions for the MoE. In the positive sequence the GATAA motif (green box) and its reverse-complement TTATC (blue box) are highlighted. Letter height encodes the per-nucleotide attribution in arbitrary units (positive above and negative below the baseline); the numeric vertical scale is omitted for space. ShiftSmooth concentrates importance on the motif and suppresses off-motif noise.

Beyond the qualitative comparisons that follow, we quantify ShiftSmooth against a baseline ladder (Vanilla Gradient, gradient $\times$ input, SmoothGrad, and in-silico mutagenesis as the faithfulness gold standard), with and without the simplex gradient correction of Majdandzic et al. [52] that projects one-hot gradients onto the probability simplex. We evaluate both the proposed MoE (whose convolutional experts make its prediction differentiable to the one-hot input, so that ShiftSmooth attributes the MoE itself) and the individual experts (50 positive sequences for each of the seven factors). Faithfulness is the rank correlation of each method to in-silico mutagenesis, and stability is the attribution correlation under a small input perturbation (Table 7). On both the MoE and the experts, ShiftSmooth with simplex correction attains the highest stability (MoE 0.96, experts 0.95), beating SmoothGrad (0.92/0.91), while its faithfulness is comparable to the other simplex-corrected methods: because its smoothing kernel is a small translation of the input, it keeps every perturbed input a valid one-hot sequence, whereas Gaussian SmoothGrad pushes inputs off the simplex. The MoE and the single-expert attributions reach comparable faithfulness and stability, and the method ranking is the same for both. Consistent with recent interpretability benchmarks, the faithfulness gains from smoothing are modest, and the simplex correction (rather than the smoothing kernel) accounts for most

of the faithfulness improvement; we therefore frame ShiftSmooth’s advantage as improved stability and on-simplex validity, with faithfulness comparable to the simplex-corrected gradient baselines. Gradient attribution is computed through HetMoE’s convolutional experts; the DNABERT-6 experts in the pool tokenize the sequence into k -mers and are not differentiable to individual nucleotides, so they contribute a fixed contextual prior rather than a per-base attribution.

Table 7. Quantitative attribution comparison for the proposed MoE and the individual experts (50 positive sequences for each of the seven training factors). Faithfulness = rank correlation to in-silico mutagenesis; stability = attribution correlation under input perturbation. “+corr” applies the simplex gradient correction. ShiftSmooth+corr attains the highest stability for both model classes (bold), while faithfulness is comparable across the simplex-corrected methods.

Model	Method	Faithfulness $\uparrow$	Stability $\uparrow$
MoE	ShiftSmooth +corr	0.75	0.96
MoE	SmoothGrad +corr	0.76	0.92
MoE	Vanilla Gradient +corr	0.75	0.95
MoE	ShiftSmooth	0.72	0.96
MoE	SmoothGrad	0.75	0.92
Experts	ShiftSmooth +corr	0.78	0.95
Experts	SmoothGrad +corr	0.78	0.91
Experts	Vanilla Gradient +corr	0.78	0.95
Experts	ShiftSmooth	0.72	0.95
Experts	SmoothGrad	0.73	0.91

5.1. GATA3 Data

To elucidate the differences between the individual expert model and our MoE architecture, we utilize attribution maps to interpret the predictions of both models. Our analysis centers on the input sequence depicted in Figure 7, which is from the ENCODE database [20].

5.1.1. Expert Model

The input sequence is fed into the GATA3-trained expert model, which correctly identifies the presence of the motif. This includes the motif sequence ‘GATAA’ (highlighted in green in Figure 7) and the corresponding inverse sequence, ‘TTATC’ (highlighted in blue in Figure 7). The prediction is visualized using Vanilla Gradients and ShiftSmooth in Figure 7.

Figure 7 shows the importance scores attributed to each input by the Vanilla Gradient. The green box shows the scores for the ‘GATAA’ sequence. One can note that the ‘G’ in ‘GATAA’ receives minimal or practically zero importance. The blue box highlights the scores for the ‘TTATC’ sequence. All nucleotides show a positive importance score, with the maximum score attributed to the third ‘T’.

Next, we examine the ShiftSmooth method, also shown in Figure 7. Compared to the Vanilla Gradient, there is essentially no difference for the ‘TTATC’ sequence. This indicates that minor shifts in the input do not affect the attribution for this portion of the input sequence. For the ‘GATAA’ sequence, ShiftSmooth offers a more plausible explanation as indicated by the increased importance attributed to the nucleotide ‘G’ which is essentially absent for the Vanilla Gradient. This also shows promise for the ShiftSmooth method for improving the utility of attribution as a tool for motif discovery and localization.

Given the nature of the ShiftSmooth method, we can show what the model deems important regardless of minor shifts in the data. The viewing window for genomic data is difficult to control, as the start and endpoint of a sequence are somewhat arbitrarily

selected for datasets, especially within the margin of a few nucleotides. Therefore, the ShiftSmooth method provides a more attributionally robust explanation of model features, showing how the model will interpret data overall, avoiding the anomalies that arise in the presence of a particular sequence window. This explanation is achieved by aggregating the gradient at multiple context windows and smoothing it out to avoid anomalies. The attributional robustness is exemplified by the ShiftSmooth method assigning importance to the 'G,' which is absent in the Vanilla Gradient.

5.1.2. MoE Architecture

The same sequence shown in Figure 7 is also tested using the MoE model, which correctly identifies the motif. The results are visualized in Figure 7. Explaining the individual expert models is straightforward, as each follows a standard CNN architecture. Explaining the MoE is slightly more involved because its convolutional experts process the input in parallel and remain separate until they converge at the gating layer. We treat the input sequence as a single tensor presented to all convolutional experts simultaneously, rather than as a separate copy per expert. By taking the derivative of the MoE output with respect to this single input tensor, we obtain an explanation that represents the complete MoE model. Figure 7 presents the Vanilla Gradient and ShiftSmooth explanations of the MoE model.

Figure 7 reveals that the MoE model is much less focused on any particular motif, which is consistent with its training to detect the motifs of all seven training factors. Regarding the 'GATAA' motif, the Vanilla Gradient shows the second 'A' having a negative attribution. A negative attribution score means that the explanation is less plausible. This is undesirable because it implies that the model is likely looking at less relevant features when making its decision. The ShiftSmooth results in Figure 7 produced a much more plausible attribution of the importance to the GATA3 motif. There is a positive attribution for the second 'A,' which was deemed unimportant by the Vanilla Gradient. The ShiftSmooth result shows that the negative attribution of the second 'A' is an anomaly, as the nucleotide is given a positive attribution when the sequence is shifted slightly. This emphasizes how ShiftSmooth is better for locating key motif features than the Vanilla Gradient. When examining the portion of the sequence highlighted in blue, we see that 'TTATC' is given nearly identical attribution scores for both Vanilla Gradient and ShiftSmooth.

Note that for 'TTATC,' the attributions for the expert model trained specifically for the 'GATAA' sequence (Figure 7) show a high attribution score for all of the nucleotides in 'TTATC,' whereas the attributions for the MoE model (Figure 7) show that this is not highly attributed. This shows the specificity of the expert model when compared to the MoE model. The individual expert provides much higher importance to all GATA3 motif features because it is only looking for those features.

ShiftSmooth more effectively attributes the motifs in the sequence as the most important features for both the individual experts and the MoE model. The examples given above show that ShiftSmooth is better than the Vanilla Gradient for the localization and identification of motifs.

5.1.3. Robust Model Training

While not explored in these results, robust model training creates more interpretable, plausible, and attributionally robust explanations than their non-robust counterparts [15,53]. However, robust models require longer training times, and robust pre-trained weights are scarce. One can achieve some of the benefits of attributionally robust explanations without the need for robust model training. ShiftSmooth offers these explainability benefits of a robust model without necessitating retraining or additional training time, while still

generating interpretable and plausible results. Attributional robustness is especially useful for non-robust models, which often attribute importance to noise and irrelevant features.

5.2. Out-of-Distribution Data

To get a better understanding of the key features that each model is focused on, we also analyze the attribution on OOD data. In this case, we generate a random input sequence which does not contain any of the motifs learned by the trained expert models. This sequence (a random negative) is also shown in Figure 7 and is the input to both the GATA3 trained expert model and the MoE architecture.

5.2.1. Expert Model

The GATA3 trained expert model correctly predicts that no motif is present. The attribution maps for this prediction are shown in Figure 7 using the Vanilla Gradient and ShiftSmooth. Both results show that the expert model does not deem high importance to any stretches of five or more nucleotides. The model does, however, pick up on pieces of the GATA3 motif, highlighting ‘GAT’ twice between nucleotides 18 and 25 in the sequence. This indicates that the expert model still considers portions of the motif important, even if the full motif is not present. Note that the Vanilla Gradient and ShiftSmooth attributions present similar results, validating that these same attributions are representative of the expert model, regardless of small shifts in the selected sequence window.

5.2.2. MoE Architecture

The MoE model also correctly predicts that no motifs are present. The attribution maps corresponding to the Vanilla Gradient and ShiftSmooth methods are shown in Figure 7. Upon examining these visualizations, no stretches of five or more nucleotides corresponding to any of the motif sequences emerge as significantly important. This aligns with expectations, given that the MoE model is looking for multiple motifs, and the sequence lacks any significant features characteristic of any particular learned motif. This emphasizes the difference between the specificity of the GATA3 expert and the generality of the MoE model. The GATA3 model is specifically looking for one set of features, whereas the MoE model is looking for multiple features.

We also note that the sequence ‘CG’ consistently has a significant negative attribution for both models and both attribution methods. This is likely because this sequence ‘CG’ is not present in the motif sequences used for training. This was a consistently negatively attributed feature, indicating that this feature likely contributed towards the model not predicting the presence of the motif.

6. Limitations and Threats to Validity

We state several limitations explicitly, both to scope the claims and to guide future work.

Generalization is motif-family-dependent. Sequence-only models transfer most reliably to held-out factors that share a DBD family with a training factor; transfer to entirely unseen families, and to factors that lack an intrinsic motif, is fundamentally harder and in some cases not expected from local sequence alone. Our stratified evaluation makes this explicit: the strongest and most robust gains occur on the within-family and sequence-specific cross-family strata (mean AUC 0.846 and 0.775, respectively), whereas the non-motif factors sit near a sequence-only ceiling (mean AUC 0.696) and are reported separately rather than as evidence of generalization.

Cell-line and assay confounds. ChIP-seq peaks carry cell-line-, antibody-, and laboratory-specific signal; if held-out factors were assayed in different cell lines than the training factors, apparent “generalization” could partly reflect chromatin or batch differ-

ences rather than transcription-factor sequence preference. We mitigate this by anchoring training and most OOD factors to a single cell line (K562), but residual confounding cannot be fully excluded.

Metrics and prevalence. We report AUC for ranking. AUC on a balanced 500/500 test set does not reflect deployment precision, where binding sites are rare, and a metric computed on a balanced set does not equal the deployment value.

Model selection and scale. Gating configurations are selected on the in-distribution mean AUC and the OOD strata are scored once, but the selected configuration is one of a small screened set, so the selected result may be slightly optimistic; we therefore report the mean and standard deviation across three seeds rather than a single run. The study spans seven training and 29 held-out factors across three random seeds; broader factor and cell-line coverage would further strengthen the conclusions.

Attribution. ShiftSmooth is a smoothing attribution that keeps inputs on the one-hot simplex; it is not a faithfulness oracle. Stability under perturbation alone is insufficient evidence of faithfulness, so we validate against in-silico mutagenesis (the genomics gold standard) for both faithfulness and motif localization, and we combine ShiftSmooth with the simplex-tangent gradient correction [52]. Its advantage lies in on-simplex validity and stability rather than in exceeding the simplex-corrected gradient baselines on faithfulness or localization.

7. Summary and Conclusions

This study introduces HetMoE, a heterogeneous, embedding-gated Mixture-of-Experts for TFBS prediction, alongside the ShiftSmooth explainability method, with measurable gains in accuracy and interpretability. By gating over the embeddings of a heterogeneous pool of complementary backbones (a modified-DeepBIND CNN, DeepSEA, DanQ, and a fine-tuned DNABERT-6), with the in-distribution-selected configuration being a pretraining-free convolutional pool, HetMoE attains the best in-distribution and out-of-distribution performance among all models considered, exceeding every expert in its pool and the same-generation baselines, and surpassing the strongest single model (fine-tuned DNABERT-6) on the motif-OOD mean and on 13 of the 23 motif-bearing OOD factors under a fair GC-matched-negative protocol (most decisively on the sequence-specific and within-family factors). The paired-bootstrap analyses provide clear evidence for the significance of these differences, and the input-dependent gating outperforms a static average of the same experts, showing that gating, not ensembling, drives the gain. These results indicate that input-dependent gating improves sequence-only generalization to held-out human transcription factors.

The second contribution is ShiftSmooth, an attribution method that gives more stable motif-level attributions than the Vanilla Gradient, which can aid the identification and localization of TFBS motifs.

Sequence-only TFBS predictors capture a single regulatory modality. A natural extension is to integrate robust, explainable per-modality models such as HetMoE into the multimodal oncology pipelines that combine genomic, imaging, and clinical data [54,55]. Foundation-model embeddings and multi-omics networks [56,57] offer one route to this integration, in which the embedding-gating principle used here could weight heterogeneous data sources much as it now weights heterogeneous experts.

Author Contributions: Conceptualization, A.T., I.E.N., M.U., R.P.R. and G.R.; Methodology, A.T., I.E.N., M.U., R.P.R. and G.R.; Software, A.T. and I.E.N.; Validation, A.T., I.E.N., R.P.R. and G.R.; Formal analysis, A.T., I.E.N., M.U., R.P.R. and G.R.; Investigation, A.T., I.E.N., M.U., R.P.R. and G.R.; Data curation, A.T. and I.E.N.; Writing—original draft, A.T. and I.E.N.; Writing—review and editing, A.T., M.U. and R.P.R.; Supervision, M.U., R.P.R. and G.R.; Project administration, A.T., M.U., R.P.R. and G.R. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partly supported by NSF grants 2234468 and 2234836, and NIH grant P30-CA076292 to A. Tripathi and G. Rasool (PI). I. E. Nielsen was supported by the US Department of Education through a Graduate Assistance in Areas of National Need (GAANN) program Award Number P200A180055.

Institutional Review Board Statement: This study does not require IRB approval since neither human subjects nor animal subjects are used.

Data Availability Statement: The original data presented in the study are openly available in [GitHub] at [<https://github.com/lab-rasool/TFBS>, accessed on 3 July 2026], and [Hugging Face] at [<https://huggingface.co/datasets/Lab-Rasool/ENCODE-TFBS>, accessed on 3 July 2026].

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Data and Code Availability Statement

Appendix A.1. Datasets

All data supporting the findings of this study are publicly available from standardized genomic databases. The TFBS sequences used for model training and evaluation were obtained from chromatin immunoprecipitation sequencing (ChIP-seq) experiments, which provide genome-wide maps of protein–DNA interactions at base-pair resolution.

The ChIP-seq elements [58,59] are DNA sequences that help identify functional elements in the genome and are used to map protein–DNA interactions. ChIP-seq samples are prepared by cross-linking a protein of interest with chromatin, shearing the chromatin–protein complex into fragments, and immunoprecipitating the fragments using an antibody. The cross-link is then reversed, and the resulting ChIP samples are sequenced using high-throughput sequencing platforms and mapped to a reference genome. Genomic regions with enriched ChIP reads indicate protein–DNA interactions and represent potential binding sites.

All ChIP-seq data were obtained from the ENCODE (Encyclopedia of DNA Elements) Project which maintains standardized experimental protocols and quality control metrics across participating laboratories [20,21]. These data are available at <https://www.encodeproject.org/>. All model training and performance evaluation (including ROC curve generation, AUC calculation, and ANOVA analysis) use exclusively ENCODE ChIP-seq data.

The ENCODE ChIP-seq data underwent a standardized processing pipeline:

- **Peak Selection:** High-confidence ChIP-seq peaks (representing experimentally validated protein–DNA binding events) were identified across multiple cell types and experimental conditions.
- **Sequence Extraction:** 101-base pair DNA sequences were extracted, centered on ChIP-seq peak summits to capture core binding sites with flanking regulatory context.
- **Negative Sample Generation:** Control sequences were created through dinucleotide shuffling of positive sequences, preserving local sequence composition while disrupting functional binding motifs.
- **Standardization:** All sequences are 101 base pairs in length and one-hot encoded over the four nucleotides $\{A, C, G, T\}$. Each sequence is padded by 23 positions on each side (one fewer than the 24-wide convolutional kernel) to a 4×147 input tensor, with

every padded position set to a uniform 0.25 across the four nucleotides (the standard convention for this convolutional model) rather than to zeros. The value 24 is the kernel width and does not correspond to any trimming of the sequence.

- Encoding: Sequences were converted to one-hot representation for neural network processing.

JASPAR is a curated database containing PWMs that represent the consensus binding preferences for transcription factors [60]. In our study, JASPAR is used solely to validate the biological relevance of motifs discovered by our explainability methods. JASPAR data is not used for model training or performance evaluation.

Appendix A.2. In-Distribution Data

The in-distribution data is used for training the individual expert models and for performance evaluation, with the test sequences disjoint from those used for training. We use ChIP-seq data for the seven training factors detailed in Section 3.6: ARID3A, FOXM1, GATA3, JUND, MAX, GABPA, and SP1, each from a distinct DBD family.

ARID3A is a transcription factor involved in cell cycle regulation and embryonic development, playing key roles in B lymphocyte development and pluripotency [61]. FOXM1 is a key regulator of cell proliferation and is often overexpressed in various cancers, making it an important target for cancer research and potential therapeutic interventions [62]. GATA3 is associated with T-cell development and is central to regulating cell growth and division in immature red blood cells. Mutations in these genes can lead to various diseases, such as Diamond-Blackfan anemia, a bone marrow failure syndrome, and dyserythropoietic anemia, which results in a deficiency of red blood cells [63]. The remaining four training factors span additional DBD families. JUND is a member of the AP-1 complex (bZIP family) that regulates gene expression in response to stress and growth signals and modulates cell proliferation and apoptosis. MAX is a bHLH-Zip protein that dimerizes with MYC and related partners to control cell growth, proliferation, and metabolism. GABPA is an ETS-family factor that regulates nuclear-encoded mitochondrial genes and genes involved in the cell cycle and differentiation. SP1 is a ubiquitously expressed C2H2 zinc-finger factor that binds GC-rich promoter elements and regulates a broad set of housekeeping and cell-cycle genes. The study of these transcription factors highlights their importance in understanding genetic disorders and developing targeted therapies.

Appendix A.3. Out-of-Distribution Data

To assess generalization, we evaluate on OOD transcription factors held out entirely from training, drawn from the ENCODE corpus and stratified by DBD family. The within-family stratum holds out members of the trained families (bZIP: JUN, FOSL1, ATF3; bHLH-Zip: MYC, MXI1, USF1; ETS: ELF1, ETS1, SPI1; C2H2: SP2, YY1, ZNF143, REST; GATA: GATA1, GATA2; forkhead: FOXA1, FOXA2). The cross-family stratum uses sequence-specific factors from families absent at training (CTCF, STAT3, NRF1, HNF4A, TCF7L2, ZBTB33). A separately-reported non-motif stratum (POLR2A, EP300, EZH2, TAF1, RBBP5, SAP30) comprises recruited or general factors whose ChIP-seq occupancy has a low sequence-prediction ceiling by construction. To avoid confounding the train/OOD boundary with cell type, training and most OOD factors are anchored to the K562 cell line.

Several held-out factors illustrate the strata. The within-family stratum holds out untrained members of the trained DBD families—for example, the bZIP factor JUN and the bHLH-Zip factor MYC—testing generalization within a familiar motif grammar. CTCF (CCCTC-binding factor), a highly conserved zinc-finger protein that acts as a chromatin insulator and transcriptional regulator [64], and STAT3 (signal transducer and activator of transcription 3), which mediates cytokine and growth-factor responses with dysregulation

linked to several cancers [65], are sequence-specific factors from DBD families absent during training (cross-family stratum). The non-motif stratum, reported separately, comprises recruited or general factors whose ChIP-seq occupancy has a low sequence-prediction ceiling: POLR2A, the largest subunit of RNA polymerase II [66]; RBBP5, a core component of histone-methyltransferase complexes [67]; and SAP30, a component of histone-deacetylase complexes [68], among others. The complete per-stratum factor lists are given above; together they span a range of DBD families and regulatory mechanisms and provide a leakage-free assessment of generalization.

Appendix A.4. Repository for Data and Source Code

The complete processed dataset (training, validation, and test sequences with their binary labels and metadata) is available at <https://huggingface.co/datasets/Lab-Rasool/ENCODE-TFBS> (accessed on 3 July 2026) and contains 1014 files spanning numerous transcription factors across multiple cell lines and ENCODE production centers. For each factor, the training partition contains experimentally bound (positive) sequences and the held-out test partition is balanced, with 500 positives and 500 experimental negatives. Negative training examples are generated by dinucleotide shuffling, which preserves the dinucleotide composition of each positive sequence while disrupting the binding motif; all evaluations in this paper use the balanced curated test partitions. Full format documentation accompanies the dataset.

The source code implementing all computational analyses, including model architectures, training procedures, evaluation metrics, and the ShiftSmooth attribution method, is freely available at <https://github.com/lab-rasool/TFBS> (accessed on 3 July 2026). The repository includes documentation, dependency specifications, and example notebooks demonstrating the application of trained models to new sequences. All software is released under the MIT License to facilitate unrestricted use in both academic and commercial settings.

The ENCODE data used in this study carry no usage restrictions, and the derived datasets preserve this open access. Researchers can obtain all raw and processed data through the linked repositories, and the standardized format and processing pipeline support direct comparison with future studies.

References

1. Dreyfuss, G.; Kim, V.N.; Kataoka, N. Messenger-RNA-binding proteins and the messages they carry. *Nat. Rev. Mol. Cell Biol.* **2002**, *3*, 195–205. [CrossRef] [PubMed]
2. Inukai, S.; Kock, K.H.; Bulyk, M.L. Transcription factor–DNA binding: Beyond binding site motifs. *Curr. Opin. Genet. Dev.* **2017**, *43*, 110–119. [CrossRef] [PubMed]
3. Vijayvargiya, S.; Shukla, P. Regulatory Motif Identification in Biological Sequences: An Overview of Computational Methodologies. In *Advances in Enzyme Biotechnology*; Shukla, P., Pletschke, B.I., Eds.; Springer: New Delhi, India, 2013; pp. 111–124. [CrossRef]
4. Prosperi, M.; Marini, S.; Boucher, C. Fast and exact quantification of motif occurrences in biological sequences. *BMC Bioinform.* **2021**, *22*, 445. [CrossRef] [PubMed]
5. Zheng, A.; Lamkin, M.; Zhao, H.; Wu, C.; Su, H.; Gymrek, M. Deep neural networks identify sequence context features predictive of transcription factor binding. *Nat. Mach. Intell.* **2021**, *3*, 172–180. [CrossRef] [PubMed]
6. Zhang, Q.; Wang, S.; Chen, Z.; He, Y.; Liu, Q.; Huang, D.S. Locating transcription factor binding sites by fully convolutional neural network. *Brief. Bioinform.* **2021**, *22*, bbaa435. [CrossRef] [PubMed]
7. Cazares, T.A.; Rizvi, F.W.; Iyer, B.; Chen, X.; Kotliar, M.; Bejjani, A.T.; Wayman, J.A.; Donmez, O.; Wronowski, B.; Parameswaran, S.; et al. maxATAC: Genome-scale transcription-factor binding prediction from ATAC-seq with deep neural networks. *PLoS Comput. Biol.* **2023**, *19*, e1010863. [CrossRef] [PubMed]
8. Alipanahi, B.; Delong, A.; Weirauch, M.T.; Frey, B.J. Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nat. Biotechnol.* **2015**, *33*, 831–838. [CrossRef] [PubMed]
9. Trabelsi, A.; Chaabane, M.; Ben-Hur, A. Comprehensive evaluation of deep learning architectures for prediction of DNA/RNA sequence binding specificities. *Bioinformatics* **2019**, *35*, i269–i277. [CrossRef] [PubMed]

10. Singh, N.; Nath, R.; Singh, D.B. Splice-site identification for exon prediction using bidirectional LSTM-RNN approach. *Biochem. Biophys. Rep.* **2022**, *30*, 101285. [\[CrossRef\]](#) [\[PubMed\]](#)
11. Dasari, C.M.; Bhukya, R. Explainable deep neural networks for novel viral genome prediction. *Appl. Intell.* **2022**, *52*, 3002–3017. [\[CrossRef\]](#) [\[PubMed\]](#)
12. Ji, Y.; Zhou, Z.; Liu, H.; Davuluri, R.V. DNABERT: Pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. *Bioinformatics* **2021**, *37*, 2112–2120. [\[CrossRef\]](#) [\[PubMed\]](#)
13. Hassanzadeh, H.R.; Wang, M.D. DeeperBind: Enhancing Prediction of Sequence Specificities of DNA Binding Proteins. In Proceedings of the IEEE International Conference on Bioinformatics and Biomedicine, Shenzhen, China, 15–18 December 2016; pp. 178–183. [\[CrossRef\]](#) [\[PubMed\]](#)
14. Park, S.; Koh, Y.; Jeon, H.; Kim, H.; Yeo, Y.; Kang, J. Enhancing the interpretability of transcription factor binding site prediction using attention mechanism. *Sci. Rep.* **2020**, *10*, 13413. [\[CrossRef\]](#) [\[PubMed\]](#)
15. Nielsen, I.E.; Dera, D.; Rasool, G.; Ramachandran, R.P.; Bouaynaya, N.C. Robust Explainability: A tutorial on gradient-based attribution methods for deep neural networks. *IEEE Signal Process. Mag.* **2022**, *39*, 73–84. [\[CrossRef\]](#)
16. Ancona, M.; Ceolini, E.; Öztireli, C.; Gross, M. Gradient-Based Attribution Methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*; Samek, W., Montavon, G., Vedaldi, A., Hansen, L.K., Müller, K.R., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 169–191. [\[CrossRef\]](#)
17. Nielsen, I.E.; Ramachandran, R.P.; Bouaynaya, N.; Fathallah-Shaykh, H.M.; Rasool, G. EvalAttAI: A Holistic Approach to Evaluating Attribution Maps in Robust and Non-Robust Models. *IEEE Access* **2023**, *11*, 82556–82569. [\[CrossRef\]](#)
18. Farias, A.B.; Martinez, G.S.; Galán-Vásquez, E.; Nicolás, M.F.; Pérez-Rueda, E. Predicting bacterial transcription factor binding sites through machine learning and structural characterization based on DNA duplex stability. *Brief. Bioinform.* **2024**, *25*, bbae581. [\[CrossRef\]](#) [\[PubMed\]](#)
19. Carlton, M.A.; Devore, J.L. *Probability with Applications in Engineering, Science, and Technology*; Springer: Cham, Switzerland, 2017. [\[CrossRef\]](#)
20. Davis, C.A.; Hitz, B.C.; Sloan, C.A.; Chan, E.T.; Davidson, J.M.; Gabdank, I.; Hilton, J.A.; Jain, K.; Baymuradov, U.K.; Narayanan, A.K.; et al. The encyclopedia of DNA elements (encode): Data Portal update. *Nucleic Acids Res.* **2017**, *46*, D794–D801. [\[CrossRef\]](#) [\[PubMed\]](#)
21. Consortium, E.P. A user's guide to the encyclopedia of DNA elements (ENCODE). *PLoS Biol.* **2011**, *9*, e1001046. [\[CrossRef\]](#)
22. Sanseviero, O.; Tunstall, L.; Schmid, P.; Mangrulkar, S.; Belkada, Y.; Cuenca, P. *Mixture of Experts Explained*; Hugging Face Blog: New York, NY, USA, 2023.
23. Jacobs, R.A.; Jordan, M.I.; Nowlan, S.J.; Hinton, G.E. Adaptive mixtures of local experts. *Neural Comput.* **1991**, *3*, 79–87. [\[CrossRef\]](#) [\[PubMed\]](#)
24. Eigen, D.; Ranzato, M.; Sutskever, I. Learning factored representations in a deep mixture of experts. *arXiv* **2013**, arXiv:1312.4314.
25. Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; Dean, J. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
26. Selvaraju, R.R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 618–626.
27. Ali, S.; Abuhmed, T.; El-Sappagh, S.; Muhammad, K.; Alonso-Moral, J.M.; Confalonieri, R.; Guidotti, R.; Del Ser, J.; Díaz-Rodríguez, N.; Herrera, F. Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence. *Inf. Fusion* **2023**, *99*, 101805. [\[CrossRef\]](#)
28. Nauta, M.; Trienes, J.; Pathak, S.; Nguyen, E.; Peters, M.; Schmitt, Y.; Schlötterer, J.; van Keulen, M.; Seifert, C. From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable AI. *ACM Comput. Surv.* **2023**, *55*, 295. [\[CrossRef\]](#)
29. Shrikumar, A.; Greenside, P.; Kundaje, A. Learning Important Features Through Propagating Activation Differences. In *Proceedings of the 34th International Conference on Machine Learning, PMLR, Sydney, NSW, Australia, 6–11 August 2017*; Precup, D., Teh, Y.W., Eds.; Proceedings of Machine Learning Research: Cambridge, MA, USA, 2017; Volume 70, pp. 3145–3153.
30. Sturmels, P.; Lundberg, S.; Lee, S.I. Visualizing the impact of feature attribution baselines. *Distill* **2020**, *5*, e22. [\[CrossRef\]](#)
31. Chefer, H.; Gur, S.; Wolf, L. Transformer interpretability beyond attention visualization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 782–791.
32. Lundberg, S.M.; Lee, S.I. A unified approach to interpreting model predictions. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 4768–4777.
33. Ahmed, S.; Nielsen, I.E.; Tripathi, A.; Siddiqui, S.; Rasool, G.; Ramachandran, R.P. Transformers in Time-Series Analysis: A Tutorial. *Circuits Syst. Signal Process.* **2023**, *42*, 7433–7466. [\[CrossRef\]](#)
34. Smilkov, D.; Thorat, N.; Kim, B.; Viégas, F.; Wattenberg, M. Smoothgrad: Removing noise by adding noise. *arXiv* **2017**, arXiv:1706.03825.

35. Muhammad, A.; Bae, S.H. A survey on efficient methods for adversarial robustness. *IEEE Access* **2022**, *10*, 118815–118830. [[CrossRef](#)]
36. Dera, D.; Rasool, G.; Bouaynaya, N. Extended variational inference for propagating uncertainty in convolutional neural networks. In Proceedings of the 2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP), Pittsburgh, PA, USA, 13–16 October 2019; pp. 1–6.
37. Dera, D.; Bouaynaya, N.C.; Rasool, G.; Shterenberg, R.; Fathallah-Shaykh, H.M. PremiUm-CNN: Propagating uncertainty towards robust convolutional neural networks. *IEEE Trans. Signal Process.* **2021**, *69*, 4669–4684. [[CrossRef](#)]
38. Chen, C.; Hou, J.; Shi, X.; Yang, H.; Birchler, J.A.; Cheng, J. Interpretable attention model in transcription factor binding site prediction with deep neural networks. *bioRxiv* **2019**. [[CrossRef](#)]
39. Zhou, J.; Troyanskaya, O.G. Predicting effects of noncoding variants with deep learning-based sequence model. *Nat. Methods* **2015**, *12*, 931–934. [[CrossRef](#)] [[PubMed](#)]
40. Lu, Y.; Lee, S.; Kang, S.; Kim, S. Mixture-of-Experts Approach for Enhanced Drug-Target Interaction Prediction and Confidence Assessment. *bioRxiv* **2024**. [[CrossRef](#)]
41. Luo, H.; Shan, W.; Chen, C.; Ding, P.; Luo, L. Improving language model of human genome for DNA–protein binding prediction based on task-specific pre-training. *Interdiscip. Sci. Comput. Life Sci.* **2023**, *15*, 32–43.
42. Sundararajan, M.; Taly, A.; Yan, Q. Axiomatic attribution for deep networks. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, NSW, Australia, 6–11 August 2017; pp. 3319–3328.
43. Ouyang, Z.; Liu, F.; Li, W.; Wang, J.; Chen, B.; Zheng, Y.; Li, Y.; Tao, H.; Xu, X.; Li, C.; et al. The developmental and evolutionary characteristics of transcription factor binding site clustered regions based on an explainable machine learning model. *Nucleic Acids Res.* **2024**, *52*, 7610–7626. [[CrossRef](#)] [[PubMed](#)]
44. Chandra, O.; Pramanik, D.; Gautam, S.; Sharma, M.; Dubey, N.; Mahato, B.; Kumar, V. Explainable models using transcription factor binding and epigenome patterns at promoters reveal disease-associated genes and their regulators in the context of cell-types. *bioRxiv* **2024**. [[CrossRef](#)]
45. Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 2623–2631.
46. Sutskever, I.; Martens, J.; Dahl, G.; Hinton, G. On the importance of initialization and momentum in deep learning. In Proceedings of the International Conference on Machine Learning, PMLR, Atlanta, GA, USA, 16–21 June 2013; pp. 1139–1147.
47. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
48. Bishop, C.M.; Nasrabadi, N.M. *Pattern Recognition and Machine Learning*; Springer: New York, NY, USA, 2006.
49. Tourne, N.; De Waele, G.; Vermeirssen, V.; Waegeman, W. How negative sampling shapes the performance of transcription factor binding site prediction models. *Bioinformatics* **2026**, *42*, btg048. [[CrossRef](#)] [[PubMed](#)]
50. Krützfeldt, L.M.; Schubach, M.; Kircher, M. The impact of different negative training data on regulatory sequence predictions. *PLoS ONE* **2020**, *15*, e0237412. [[CrossRef](#)] [[PubMed](#)]
51. Ghandi, M.; Lee, D.; Mohammad-Noori, M.; Beer, M.A. Enhanced regulatory sequence prediction using gapped k-mer features. *PLoS Comput. Biol.* **2014**, *10*, e1003711. [[CrossRef](#)] [[PubMed](#)]
52. Majdandzic, A.; Rajesh, C.; Koo, P.K. Correcting gradient-based interpretations of deep neural networks for genomics. *Genome Biol.* **2023**, *24*, 109. [[CrossRef](#)] [[PubMed](#)]
53. Tsipras, D.; Santurkar, S.; Engstrom, L.; Turner, A.; Madry, A. Robustness May Be at Odds with Accuracy. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
54. Waqas, A.; Tripathi, A.; Ramachandran, R.P.; Stewart, P.A.; Rasool, G. Multimodal data integration for oncology in the era of deep neural networks: A review. *Front. Artif. Intell.* **2024**, *7*, 1408843. [[CrossRef](#)] [[PubMed](#)]
55. Tripathi, A.; Waqas, A.; Venkatesan, K.; Yilmaz, Y.; Rasool, G. Building Flexible, Scalable, and Machine Learning-Ready Multimodal Oncology Datasets. *Sensors* **2024**, *24*, 1634. [[CrossRef](#)] [[PubMed](#)]
56. Tripathi, A.; Waqas, A.; Schabath, M.B.; Yilmaz, Y.; Rasool, G. HONeYBEE: Enabling scalable multimodal AI in oncology through foundation model-driven embeddings. *npj Digit. Med.* **2025**, *8*, 622. [[CrossRef](#)] [[PubMed](#)]
57. Waqas, A.; Tripathi, A.; Ahmed, S.; Mukund, A.; Farooq, H.; Johnson, J.O.; Stewart, P.A.; Naeini, M.; Schabath, M.B.; Rasool, G. Self-Normalizing Multi-Omics Neural Network for Pan-Cancer Prognostication. *Int. J. Mol. Sci.* **2025**, *26*, 7358. [[CrossRef](#)] [[PubMed](#)]
58. Mardis, E.R. ChIP-seq: Welcome to the new frontier. *Nat. Methods* **2007**, *4*, 613–614. [[CrossRef](#)] [[PubMed](#)]
59. Park, P.J. ChIP-seq: Advantages and challenges of a maturing technology. *Nat. Rev. Genet.* **2009**, *10*, 669–680. [[CrossRef](#)] [[PubMed](#)]

60. Castro-Mondragon, J.A.; Riudavets-Puig, R.; Rauluseviciute, I.; Berhanu Lemma, R.; Turchi, L.; Blanc-Mathieu, R.; Lucas, J.; Boddie, P.; Khan, A.; Manosalva Pérez, N.; et al. Jaspar 2022: The 9th release of the open-access database of transcription factor binding profiles. *Nucleic Acids Res.* **2021**, *50*, D165–D173. [[CrossRef](#)] [[PubMed](#)]
61. An, G.; Miner, C.A.; Nixon, J.C.; Kincade, P.W.; Bryant, J.; Tucker, P.W.; Webb, C.F. Loss of Bright/ARID3a function promotes developmental plasticity. *Stem Cells* **2010**, *28*, 1560–1567. [[PubMed](#)]
62. Liao, G.B.; Li, X.Z.; Zeng, S.; Liu, C.; Yang, S.M.; Yang, L.; Hu, C.J.; Bai, J.Y. Regulation of the master regulator FOXM1 in cancer. *Cell Commun. Signal.* **2018**, *16*, 57. [[CrossRef](#)] [[PubMed](#)]
63. Kallen, M.E.; Dulau-Florea, A.; Wang, W.; Calvo, K.R. Acquired and germline predisposition to bone marrow failure: Diagnostic features and clinical implications. *Semin. Hematol.* **2019**, *56*, 69–82. [[CrossRef](#)] [[PubMed](#)]
64. Kim, S.; Yu, N.K.; Kaang, B.K. CTCF as a multifunctional protein in genome regulation and gene expression. *Exp. Mol. Med.* **2015**, *47*, e166. [[CrossRef](#)] [[PubMed](#)]
65. Yu, H.; Lee, H.; Herrmann, A.; Buettner, R.; Jove, R. Revisiting STAT3 signalling in cancer: New and unexpected biological functions. *Nat. Rev. Cancer* **2014**, *14*, 736–746. [[CrossRef](#)] [[PubMed](#)]
66. Schier, A.C.; Taatjes, D.J. Structure and mechanism of the RNA polymerase II transcription machinery. *Genes Dev.* **2020**, *34*, 465–488. [[CrossRef](#)] [[PubMed](#)]
67. Ali, A.; Veeranki, S.N.; Chinchole, A.; Tyagi, S. MLL/WDR5 complex regulates Kif2A localization to ensure chromosome congression and proper spindle assembly during mitosis. *Dev. Cell* **2017**, *41*, 605–622. [[PubMed](#)]
68. Viiri, K.; Korkeamäki, H.; Kukkonen, M.; Nieminen, L.; Lindfors, K.; Peterson, P.; Mäki, M.; Kainulainen, H.; Lohi, O. SAP30L interacts with members of the Sin3A corepressor complex and targets Sin3A to the nucleolus. *Nucleic Acids Res.* **2006**, *34*, 3288–3298. [[CrossRef](#)] [[PubMed](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.