TABSEQ: A FRAMEWORK FOR DEEP LEARNING ON TABULAR DATA VIA SEQUENTIAL ORDERING*

Al Zadid Sultan Bin Habib[1], **Kesheng Wang**[2], **Mary-Anne Hartley**[3], **Gianfranco Doretto**[4], **Donald A. Adjeroh**[5]

[1,4,5]Lane Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV 26505, USA
ah00069@mix.wvu.edu[1], gianfranco.doretto@mail.wvu.edu[4], donald.adjeroh@mail.wvu.edu[5]
[2]College of Nursing, University of South Carolina, Columbia, SC 29208, USA
kesheng@mailbox.sc.edu[2]
[3]Yale University School of Medicine, New Haven, CT 06510, USA
mary-anne.hartley@yale.edu[3]

ABSTRACT

Effective analysis of tabular data still poses a significant problem in deep learning, mainly because features in tabular datasets are often heterogeneous and have different levels of relevance. This work introduces TabSeq, a novel framework for the sequential ordering of features, addressing the vital necessity to optimize the learning process. Features are only sometimes equally informative, and for certain deep learning models, their random arrangement can hinder the model's learning capacity. Finding the optimum sequence order for such features could improve the deep learning models' learning process. The novel feature ordering technique, which we provide in this work, is based on clustering and incorporates both local ordering and global ordering. It is designed to be used with a multi-head attention mechanism in a denoising autoencoder network. Our framework uses clustering to align comparable features and improve data organization. Multi-head attention focuses on essential characteristics, whereas denoising autoencoder highlights important aspects by rebuilding from distorted inputs. This method improves the capability to learn from tabular data while lowering redundancy. Our research demonstrating improved performance through appropriate feature sequence rearrangement utilizing raw antibody microarray and two other real-world biomedical datasets validates the impact of feature ordering. These results demonstrate that feature ordering can be a viable approach to improved deep learning of tabular data.

Keywords Deep Learning · Tabular Data · Feature Ordering

1 Introduction

Deep learning has transformed how we handle and comprehend diverse data types, resulting in unparalleled progress in numerous fields. Deep learning models have outperformed conventional techniques in audio analysis, picture identification, and Natural Language Processing (NLP), opening the door to new applications previously thought impractical. For example, Convolutional Neural Networks (CNNs) have emerged as the mainstay of image-processing applications, demonstrating exceptional performance in picture classification, object recognition, and other applications [15], [22]. In NLP, Transformer-like models have established new benchmarks for text summarization, machine translation, and question-answering systems [36], [11]. Additionally, deep learning has helped audio processing by advancing speech recognition and synthesis, greatly enhancing user interaction with technology [4], [25], [30]. These achievements demonstrate how deep learning is an essential tool for applications where standard feature engineering fails because it can grasp intricate patterns and relationships inside high-dimensional data.

The quest for an optimal deep learning architecture for tabular data, crucial in sectors like finance, healthcare, and retail, remains ongoing. Unlike image, text, and audio data, tabular data's structure, rows representing samples, and

*This paper has been accepted for presentation at the 27th International Conference on Pattern Recognition (ICPR 2024) in Kolkata, India.

columns as features present distinct challenges, especially in modeling complex feature relationships that lack spatial or sequential correlation. Innovative model architectures and data representation methods are essential to address tabular data's unique aspects. Models such as TabNet[5], Neural Oblivious Decision Ensembles (NODE)[27], and TabTransformer[18] have emerged as practical solutions alongside popular gradient-boosting tree models. However, gaps remain in handling scenarios with high-dimensional features against smaller sample sizes, such as genomic or other medical data.

We introduce TabSeq, a framework for deep learning on tabular data, using the feature ordering to optimize tabular data utilization. Our approach is motivated by methods of band ordering often used in the efficient analysis of hyperspectral images. Adapting band ordering from hyperspectral images[32] to tabular data involves comparing dataset features to spectral bands, where features, like bands, vary in informational value. This approach uses statistical and machine learning methods to prioritize significant features and reduce redundancy, enhancing dataset efficiency similar to compression in hyperspectral imaging. The bandwidth minimization problem in communication networks[38] focuses on optimizing data transmission order or compressing data to meet bandwidth limits, akin to arranging features in tabular data for deep learning models. The novel contributions of our paper are as follows:

1. We present a novel feature ordering technique that combines local ordering and global ordering to optimize feature sequences and clustering to group comparable features. This innovative method systematically improves learning and significantly improves the model's performance on tabular datasets by prioritizing features according to their relevance and informative content.
2. Our framework enables a Denoising Autoencoder (DAE) architecture to incorporate the Multi-Head Attention (MHA) mechanism smoothly. This integration highlights important characteristics, eliminates redundancy by rebuilding inputs from partially corrupted versions, and allows for dynamic attention to vital elements.
3. Our studies using raw antibody microarray and other datasets show that our feature ordering approach substantially improves the performance of deep learning models. The outcomes demonstrate how feature sequencing is crucial for training and validating the potential of feature ordering in tabular data processing inside deep learning frameworks.

These contributions collectively address the challenges of heterogeneous feature relevance in tabular data, setting a new precedent for data preprocessing and model optimization in deep learning applications.

2 Related Work

Feature ordering in tabular datasets is essential for improving machine learning models' interpretability, accuracy, and efficiency, particularly in deep learning. Models that recognize and rank important features can learn new information more quickly, require less training time, and exhibit better generalization on unobserved data [41]. While feature ordering is essential for all tabular data types, including numerical data, it influences models that use data structures, such as attention mechanisms or specific autoencoders.

Attention-based Models: TabNet[5] employs an attention mechanism for feature selection in tabular data, enhancing performance and interpretability without rearranging features. TabTransformer[18] uses contextual embeddings to improve accuracy in handling categorical data, though it requires pre-training and fine-tuning. AutoInt[31] specializes in Click-Through Rate (CTR) prediction by learning feature interactions with a self-attentive network despite assuming unordered features. ASENN[26] predicts pavement deterioration with multi-dropout attention layers, offering efficient infrastructure maintenance solutions. Attention-based models might find it difficult to determine how important a particular feature is to the model's predictions; feature ordering can help with this problem by highlighting the elements with the most significant impact.

Tree-based Models: The Tree Ensemble Layer (TEL)[14] by Hazimeh et al. enhances neural networks with the efficiency of tree ensembles through "soft trees" and sparse activations, improving performance. TEL, however, does not perform well in capturing complex feature interactions. NODE[27] by Popov et al. combined deep learning flexibility with gradient-boosted with the benefit of decision trees. They achieved superior outcomes via differentiable trees and entmax transformation, albeit with potential limitations in capturing nuanced feature interactions. Tree-based models might struggle to explain complex feature associations; feature ordering fills this gap by arranging features to clarify their relationships.

LLM-based Models: TabLLM[16], developed by Hegselmann et al., leverages LLMs for few-shot categorization of tabular data by translating tables into natural language, showing superior performance over traditional techniques with limited data. MediTab[40] by Wang et al. introduced a "learn, annotate, refine" approach combined with LLMs for medical data predictions, achieving high performance and excellent zero-shot capabilities without fine-tuning. IngesTables[43], presented by Yak et al., creates scalable tabular foundation models, addressing key issues, such as large cardinality and semantic relevance, through an attention-based method with LLMs, offering cost-effective alternatives

to conventional models for clinical trial predictions. Implicit data hierarchies may be a problem for LLM-based models; feature ordering might help by arranging data to represent underlying importance and relationships.

Graph-based Models: Ruiz et al. introduced PLATO[28], leveraging an auxiliary knowledge graph for model regularization in MLPs. This improved learning on high-dimensional tabular datasets, reducing over-fitting by connecting features to knowledge graph nodes. T2G-FORMER[44], by Yan et al., enhances structured feature interactions through a Graph Estimator and a Transformer network, offering superior interaction modeling and prediction accuracy over conventional deep neural networks. Chen et al. proposed HyTrel[7], a model using hypergraphs to capture tabular data's structural properties, outperforming existing methods with minimal pretraining by integrating inductive biases about data structure. Graph-based models may miss linear feature correlations; feature ordering can address this limitation by better aligning features to depict linear trends and relationships. In general, graph-based approaches provide improvements in tabular data analysis by capturing possible interactions between features. However, they suffer from high computational costs, limiting their applicability.

Autoencoder-based Models: ReConTab, developed by Chen et al., is a deep learning framework for automatic representation learning from tabular data, utilizing contrastive learning and an asymmetric autoencoder with regularization to enhance classification models like Random Forest and XGBoost[8]. ReMasker, introduced by Du et al., employs masked autoencoding for imputing missing values in tabular data. This improved the results through randomization in masking extra values and offering competitive performance, especially with increased missing data[12]. SwitchTab offers a self-supervised learning approach to identify less apparent dependencies in tabular data, using an asymmetric encoder-decoder to improve prediction tasks and provide interpretable insights via its embeddings[42]. Autoencoder-based models may need to be more efficient in prioritizing influential features, a limitation that feature ordering can overcome by arranging features to enhance model focus and interpretability.

Other Models: GrowNet introduces a method leveraging shallow neural networks within a gradient-boosting framework for various machine learning tasks, limited by static feature selection [6]. Using Scaled Exponential Linear Units (SELU), Self-Normalizing Neural Networks (SNNN) aim for automatic activation normalization to stabilize deep learning but face restrictions due to reliance on SELUs [20]. DCN V2 advances the integration of feature interactions, constrained by its static interaction framework [39]. Gorishniy et al.'s critique of deep learning models for tabular data, including the FT-Transformer, highlights the need for dynamic feature sequencing to enhance model performance [13]. Static feature integration is a potential problem for these models, which feature ordering can solve by dynamically modifying feature sequences to maximize learning and performance. Also see the transformer-based model, TabPFN, and the regularization-based model, TANGOS, in [17] and [19] respectively.

The literature review highlights innovative strategies for enhancing deep learning model performance on tabular data. Our feature ordering approach uniquely merges clustering with local ordering and global ordering in an MHA-augmented DAE framework, focusing on the ordered arrangement of features based on their importance and information content. This strategy contrasts with traditional methods that often do not consider the arrangement of features.

3 Methodology

This section presents our deep learning architecture, designed primarily to analyze tabular data effectively, as seen in Figure 1. The process begins with feature clustering and then moves to local ordering and global ordering to improve input feature arrangements. These rearranged features are fed into an MHA mechanism and a DAE, ultimately leading to feature extraction and classification model decisions. Our methodology introduces a novel feature ordering technique to improve the analysis of tabular datasets. This methodology seeks to improve prediction accuracy and robustness for various applications by capturing intricate feature interactions and underlying patterns in the data.

3.1 Feature Ordering

Feature ordering is finding an optimal arrangement of features within and across clusters to minimize a defined cost function that reflects the disorganization of feature positioning. This involves computing permutations that best sequence the features according to their relationships. Given a dataset $X \in \mathbb{R}^{n \times m}$ with n samples and m features, we define a set of graphs $\{G_1, G_2, ..., G_k\}$, where k is the number of clusters, and each graph $G_c = (V_c, E_c)$ for cluster c has vertices $v_i \in V_c$ corresponding to features within that cluster. The edges $(v_i, v_j) \in E_c$ represent significant relationships between features i and j within the cluster. For each cluster c, the goal is to find a permutation π_c of its features that minimizes a local cost function F_c:

$$F_c(\pi_c) = \sum_{(v_i, v_j) \in E_c} |i - j|$$

where $i = \pi_c(v_i)$ and $j = \pi_c(v_j)$ represent the indices of features v_i and v_j in the permutation π_c, minimizing the disorganization within the cluster. The overall goal is to find a global permutation Π that integrates the local

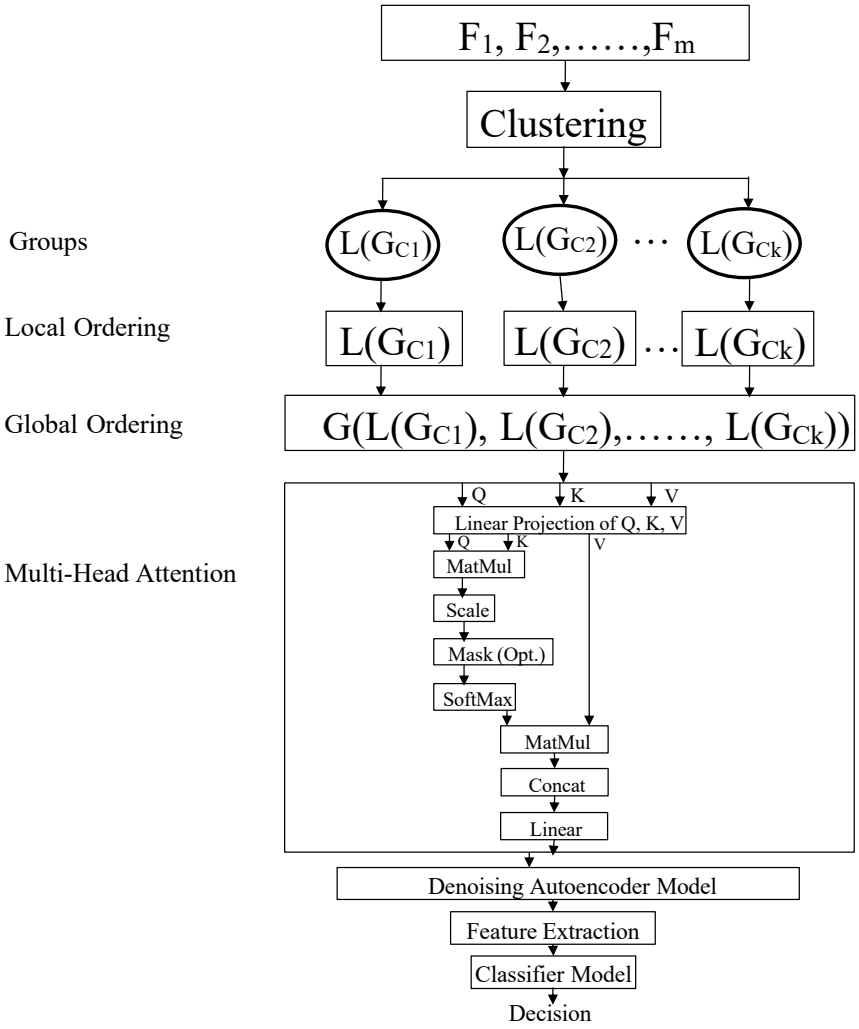


Figure 1: Overview of the TabSeq framework.

permutations π_c's and minimizes a global cost function F_G:

$$F_G(\Pi) = \sum_{c=1}^{k} \alpha_c \cdot F_c(\pi_c)$$

where α_c represents the weight or importance of cluster c in the global context with $\sum_c \alpha_c = 1$. The optimal permutation Π^* minimizes $F_G(\Pi)$:

$$\Pi^* = \arg\min_{\Pi} F_G(\Pi)$$

Local Ordering Function: Local ordering is the computation of a permutation that minimizes the sum of absolute differences in positions of related features within a cluster, thereby reducing feature dispersion.

$$L(G_c) = \text{argmin}_{\pi_c} \sum_{(v_i, v_j) \in E_c} |\pi_c(i) - \pi_c(j)|$$

Here, $L(G_c)$ outputs the permutation π_c for cluster c that minimizes the feature dispersion, where $\pi_c(i)$ is the position of feature i in the permutation π_c, and the sum quantifies the total dispersion of features in the cluster.

Global Ordering Function: Global ordering is the process of integrating local permutations from all clusters into a global permutation that minimizes the weighted sum of within-cluster feature dispersion to enhance the deep learning

model's performance.

$$G(\{\pi_1, \pi_2, ..., \pi_k\}) = \text{argmin}_{\Pi} \sum_{c=1}^{k} \alpha_c \cdot D_{\pi_c}(G_c)$$

G integrates the local permutations π_c of all clusters into a global ordering Π, minimizing the weighted sum of feature dispersion $D_{\pi_c}(G_c)$ within each cluster G_c, where α_c represents the weight or importance of cluster c. In these functions, $D_{\pi_c}(G_c)$ represents a measure of feature dispersion within cluster c based on the permutation π_c, and Π is the global permutation that integrates these local orderings into a dataset-wide global feature ordering that aims to improve the performance of the model.

Feature Dispersion: In the context of feature ordering for a tabular dataset, the term "Feature Dispersion" describes the degree to which features with a strong relationship (or dependency) are placed far apart in the ordering. The goal would be to minimize this dispersion so that related features are positioned closer together, which could be advantageous for specific deep learning models that can benefit from the structure of the data (See statistical dispersion in [21], [33]).

For instance, a generalized feature dispersion for a cluster G_c could be defined as:

$$D(\pi_c) = \sum_{(v_i, v_j) \in E_c} w_{ij} \cdot |\pi_c(i) - \pi_c(j)|$$

Where, π_c is the permutation of features within cluster c, w_{ij} is a weight assigned to the edge between features i and j (which could be based on the strength of the relationship between the features e.g., correlation or mutual information), $|\pi_c(i) - \pi_c(j)|$ is the absolute difference in the ordered positions of features i and j within the permutation π_c.

Feature Dispersion and Variance: We adopt variance as a metric to guide the ordering of features locally [29]. This approach is based on the premise that organizing features to minimize their dispersion within clusters can enhance model performance by affecting the variance of these features in a beneficial manner. We understand that feature dispersion within a cluster, $D(\pi_c)$, reflects how spread out the features are in terms of their arrangement or ordering based on certain criteria (e.g., importance, similarity, etc.). Variance, $\text{Var}(X_i)$, measures the spread of values for a given feature i across the dataset or within clusters. The goal is to understand how minimizing $D(\pi_c)$ influences $\text{Var}_c(X_i)$ for features within the same cluster. A decrease in $D(\pi_c)$ (i.e., reduced dispersion or more closely arranged features based on their relationships) leads to an increase in the homogeneity of feature values within the cluster. This homogeneity, in turn, can lead to a more meaningful and possibly reduced variance ($\text{Var}_c(X_i)$) for the features within the cluster, as related features that behave similarly or have strong relationships are positioned closer together, thus reflecting their actual data distribution more accurately. The conceptual relationship can be summarized as:

$$\text{Var}_c(X_i) \propto \frac{1}{D(\pi_c)}$$

This expression suggests that as feature dispersion within a cluster decreases (making $D(\pi_c)$ smaller), the variance of features within that cluster ($\text{Var}_c(X_i)$) becomes more meaningful of the true data distribution. The inverse proportionality indicates that lower dispersion (closer grouping of related features) leads to a more stable or accurate variance representation, underlining the importance of thoughtful feature arrangement in enhancing model understanding and performance. Algorithm 1 captures the general procedure for our feature ordering, including both steps of local and global ordering.

3.2 MHA

MHA or Multi-Head Attention, inspired from[36], is the integration that serves as a cornerstone for enhancing the model's capacity to capture complex dependencies and interactions within the input data. This mechanism's key feature is its capacity to narrow down an input sequence through several attention heads at once, which enables the model to pay attention to data from various representation subspaces at various points in time. Formally, for each head h, we perform linear transformations on the input X to obtain queries Q_h, keys K_h, and values V_h using parameter matrices W_h^Q, W_h^K, and W_h^V, respectively:

$$Q_h = XW_h^Q, \quad K_h = XW_h^K, \quad V_h = XW_h^V$$

Subsequently, we compute the scaled dot-product attention for each head. The attention function operates on queries, keys, and values and scales the dot products of queries with all keys by $\frac{1}{\sqrt{d_k}}$, where d_k is the dimensionality of the keys and queries. This scaling factor helps stabilize the gradients during training. The attention scores are then passed through a softmax function to obtain the weights on the values:

$$\text{Attention}(Q_h, K_h, V_h) = \text{softmax}\left(\frac{Q_h K_h^T}{\sqrt{d_k}}\right) V_h$$

Algorithm 1 Feature Ordering

- Preprocessed dataset $X_{\text{train}} \in \mathbb{R}^{n \times m}$ with n samples and m features.
- Clustering Algorithm: Choose from k-means, DBSCAN, HDBSCAN, or a Custom Algorithm.
- Sorting Order: Ascending or Descending.
- Number of clusters (num_clusters), required for k-means and optional for other algorithms.

Output:

- Reordered dataset $new\_training\_set \in \mathbb{R}^{n \times m}$.

Procedure

Initialize the clustering model based on the selected algorithm.
if Clustering Algorithm is k-means **then**
    Specify num_clusters.
else
    Use default or custom settings.
end if
Apply clustering to X_{train} to get cluster labels.
Append cluster labels to X_{train}.
for each cluster, excluding noise **do**
    Select data for the current cluster.
    Calculate and order feature dispersion based on the sorting order.
    Record feature order for the cluster.
end for
Combine feature orders from all clusters into an overall feature order.
Reorder X_{train} columns according to the overall feature order.
Assess model performance with the reordered dataset.

The final output of the MHA layer is created by concatenating and linearly transforming each head's output, each of which captures a unique feature of the incoming data. The information acquired by each head is combined by this concatenation procedure, maintaining the diversity of the attended features:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O$$

where, $\text{head}_h = \text{Attention}(Q_h, K_h, V_h)$, and W^O is the parameter matrix for the output linear transformation. Using this sophisticated attention mechanism, our model gains a more sophisticated capacity to identify and use the complex patterns in the data. The MHA mechanism provides insights into the diverse elements of the data that different heads focus on, hence improving the interpretability and expressive capacity of the model.

3.3 DAE

TabSeq leveraged an MHA layer in addition to DAE [37] architecture to overcome the difficulties associated with learning robust representations from high-dimensional tabular data. The DAE enhances the model's performance on ensuing tasks by lowering noise and extracting significant features. The DAE architecture comprises an encoder and a decoder, where the encoder maps the input data X to a latent space representation Z, and the decoder reconstructs the input from Z. The MHA layer improves the encoder's capacity to focus on relevant information by allocating different levels of attention to different data segments. Formally stated, the encoding procedure is as follows:

$$Z = f_{\text{encoder}}(X) = \text{ReLU}(W_e \cdot \text{MHA}(X) + b_e)$$

Where X is the input data, W_e and b_e are the weights and bias of the encoding layer, respectively, and ReLU denotes the Rectified Linear Unit activation function. The $\text{MHA}(X)$ function represents the output of the MHA layer applied to X. The decoder, aiming to reconstruct the input data from the latent representation Z, is given by:

$$\hat{X} = f_{\text{decoder}}(Z) = \text{Sigmoid}(W_d \cdot Z + b_d)$$

where, $\hat{X}$ is the reconstructed data, W_d and b_d are the decoder weights and bias, and Sigmoid is the activation function facilitating reconstruction. The loss function for the DAE, aiming to minimize the reconstruction error, is defined as the Mean Squared Error (MSE) between the original input X and its reconstruction $\hat{X}$:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N} \sum_{i=1}^{N} (X_i - \hat{X}_i)^2$$

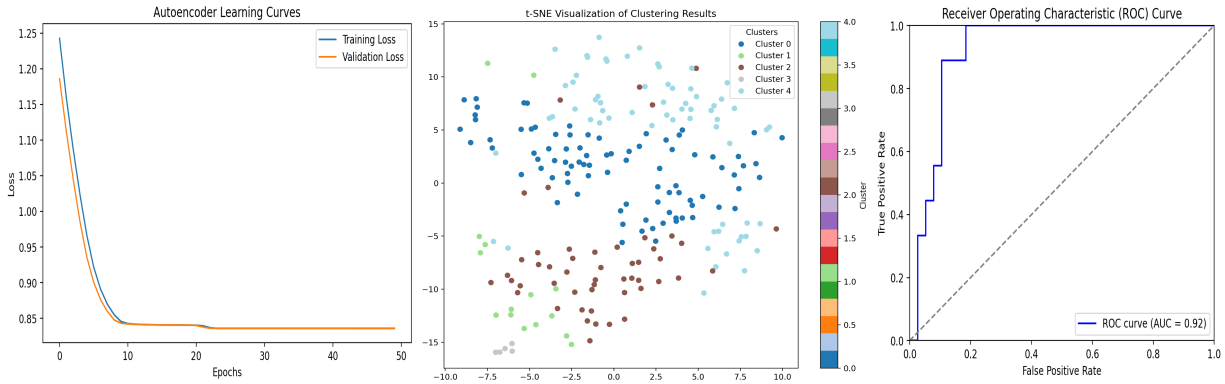


Figure 2: Visualization of model performance on the autoimmune diseases dataset.

where N is the number of samples in the dataset. The MHA layer added to the DAE architecture improves the encoder's capacity to identify and highlight the most informative characteristics by utilizing the attention mechanism [45], [34], [9], [35]. This leads to a less noisy and more discriminative representation in the latent space, which is essential for tasks that come after, like classification. Using a sequential network with dense layers, ReLU activation [23] for hidden layers, and sigmoid activation [10] for the output layer, optimized for classification tasks, the final classification model is trained on the encoded representations. Our methodology efficiently tackles the problem of learning from high-dimensional and noisy tabular data by incorporating the MHA layer within the DAE framework, significantly improving the model's predictive performance and robustness.

3.4 Feature Extraction and Classifier Model

In the TabSeq framework, the DAE is instrumental in preprocessing the input data by denoising and enhancing feature salience through its robust feature extraction process, where the encoder component transforms the corrupted input into a refined, lower-dimensional representation. These enhanced features are then utilized by the classifier, which is specifically configured with a softmax activation for multi-class scenarios to generate class probability distributions or a sigmoid activation for binary classification to yield a probability of class membership. This setup ensures that the classifier operates on high-quality features extracted post-DAE, thereby optimizing the model's accuracy and adaptability to different classification tasks. In a nutshell, the DAE processes the input data, which is then followed by a feature extraction process to extract robust, noise-reduced features, which are then utilized by the classifier to ensure precise predictions based on clean and relevant information, illustrating an essential sequential information flow where the classifier's efficacy is significantly enhanced by the high-quality features provided by the DAE.

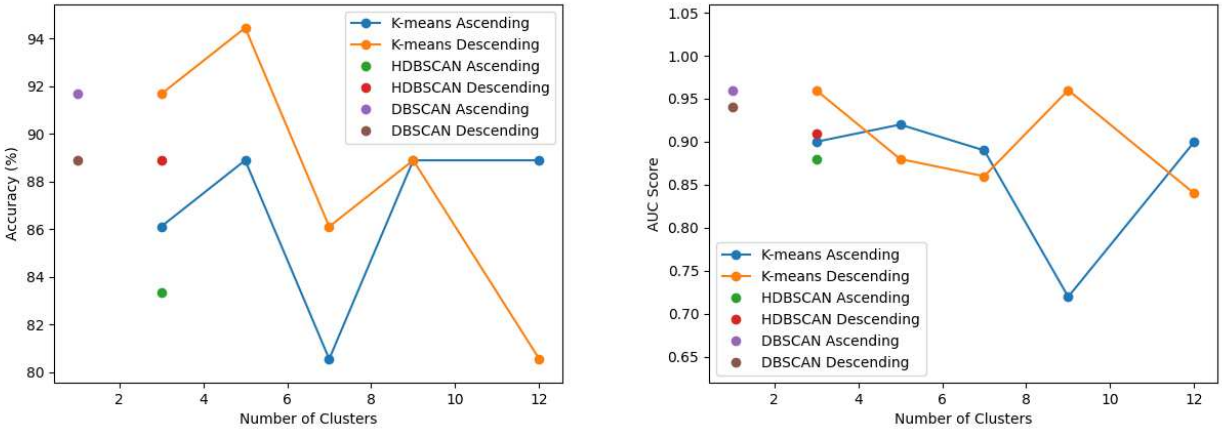
4 Experimental Results

4.1 Datasets and Model Hyperparameters

In our research, the autoimmune diseases dataset used in [24] and publicly released in [3] contains 393 features targeting five disease classes of 316 samples, detailing each antibody's signal intensity. The ADNI dataset [1] includes 177 samples and 263 features with target attributes like AD123, ABETA12, and AV45AB12, representing various stages of Alzheimer's disease and captured through DTI analysis for white matter integrity. Lastly, the WDBC dataset [2] offers 32 features derived from breast mass images, aiming to differentiate between 357 benign and 212 malignant cases. Each dataset was partitioned into training, validation, and testing subsets following a 70:15:15 split, focusing on specific target attributes for comprehensive classification and analysis. TabSeq model with feature ordering integrates an MHA mechanism with four heads and dimensionality of 32 alongside a DAE comprising dense ReLU-activated layers. It was trained over 50 epochs with a batch size of 32 using the Adam optimizer, and the model employs MSE loss for the DAE and binary cross-entropy loss for the classifier. This configuration facilitates nuanced feature extraction and robust classification, as evidenced by the model's validation performance, optimizing computational efficiency and learning effectiveness. In our analysis, feature ordering was uniformly applied across baseline models using k-means clustering with 5 clusters in ascending order for the autoimmune diseases and ADNI datasets and 3 clusters for the WDBC dataset. This consistent methodology underscores the effectiveness of feature ordering in enhancing model performance across the board, with TabSeq demonstrating particularly notable improvements in accuracy and AUC with feature ordering.

4.2 Experiments with Autoimmune Diseases' Dataset

Two cases were included in our investigation (Table 1), which showed how feature ordering affected model performance. In Case 1, NODE's accuracy sharply declined, whereas TabSeq's increased from 85.42% to 87.23% with feature ordering. Case 2 demonstrated how feature ordering significantly improved TabNet's accuracy, raising it to 94.44%.



(a) Case 2: Accuracy

(b) Case 2: AUC

Figure 3: Results' plots for TabSeq with different feature ordering configurations.

TabSeq demonstrated its practical usage of feature ordering with consistently good performance. These findings highlight the context and model-specificity of feature ordering's efficacy, with TabSeq consistently outperforming other models in the dataset on autoimmune disorders. Visualizations in Fig. 2 confirm effective clustering for feature ordering and model generalization and show the ROC curve's high binary classification accuracy. From Table 1, it is evident that feature ordering significantly affects model performance, especially in sequence-dependent architectures (e.g., autoencoders and LSTM), where the feature ordering aligns features in a meaningful sequence to improve learning. However, tree-based models, such as NODE, could not benefit as much because of their built-in feature selection methods. The local receptive field of 1D CNNs limits the effect of feature ordering. In contrast, TabNet's attention mechanism already prioritizes relevant features, which may cause inconsistent feature ordering performance. TabTransformer consistently enhances AUC across datasets; TANGOS shows marked improvements in AUC with feature ordering but does not demonstrate strong accuracy. These results underline TabSeq's capacity to discern complex patterns in high-dimensional genomic datasets, advocating their potential for data analysis. Based on ablation studies, we chose k-means with 5 clusters for feature ordering with features sorted in ascending order (Table 1 for Case 1-2 and Fig. 2 for Case 1).

4.3 Ablation Studies

We assessed the TabSeq model's performance using the autoimmune diseases' dataset, mainly how the clustering algorithms affected feature ordering. Various clustering algorithms affected AUC and model accuracy in different scenarios. For example, in Case 1, the maximum accuracy of 87.23% and the highest AUC of 0.92 were obtained using DBSCAN with a single cluster with the features sorted in ascending order. Case 2, on the other hand, achieved 94.44% accuracy and optimal performance using k-means at five clusters with features sorted in descending order. Surprisingly, DBSCAN achieved perfect accuracy and an AUC of 100% and 1.00 in Case 4, with features sorted in descending order. The significance of the number of clusters, specific clustering algorithm, and sorting order were also observed for these cases. Fig. 3 shows the assessment for Case 2, whereas the figures for other cases also looked similar. These results highlight the crucial role that clustering configurations in feature ordering play in improving the predictive power of the TabSeq model, indicating that the model's ability to distinguish intricate patterns of autoimmune diseases is greatly influenced by strategic cluster formation and feature ordering.

4.4 Experiments with ADNI and WDBC Dataset

The studies done on the ADNI dataset (Table 2) show how effective feature ordering is in several models, such as TabNet, NODE, and TabSeq. Across two target attributes, such as AD123 and ABETA12, the addition of feature ordering resulted in significant gains in accuracy and AUC scores; nonetheless, AD123 presented a multi-class classification challenge. For example, the TabSeq model's accuracy improved from 66.67% to 67.68% for the AD123 target. Similarly, substantial improvements were noted for the ABETA12 target respectively 64.44% to 75.13% for TabSeq. By efficiently selecting and rearranging features according to their informative contribution to the predictive goal, feature ordering improved the performance of deep learning models. These results highlight the importance of adopting feature ordering to refine model predictions for complex datasets. The studies conducted with the WDBC dataset (Table 2)

Table 1: Comparative results on autoimmune diseases' dataset for different models (# = without feature ordering, * = with feature ordering, Acc. = Accuracy).

Model	Acc.#	AUC#	Acc.*	AUC*
Case 1: H vs SLE+RA+SS+SV				
Linear SVM [24]	N/A	0.94	N/A	N/A
TabSeq (ours)	85.42%	0.92	87.23%	0.92
LSTM	82.11%	0.88	85.11%	0.91
DAE-LSTM	75.79%	0.80	82.98%	0.80
DAE	76.84%	0.80	80.85%	0.86
1D CNN	72.92%	0.84	79.17%	0.44
TabNet [5]	77.08%	0.85	83.33%	0.50
NODE [27]	89.58%	0.92	20.83%	0.40
TabTransformer [18]	82.98%	0.84	87.23%	0.75
TANGOS [19]	82.98%	0.61	82.98%	0.65
Case 2: SLE vs RA+SS+SV				
Linear SVM [24]	N/A	0.96	N/A	N/A
TabSeq (ours)	86.11%	0.87	91.67%	0.96
LSTM	81.94%	0.86	86.11%	0.93
DAE-LSTM	79.17%	0.81	86.11%	0.87
DAE	84.72%	0.84	86.11%	0.91
1D CNN	80.56%	0.81	80.56%	0.31
TabNet [5]	91.67%	0.60	94.44%	0.74
NODE [27]	83.33%	0.82	80.56%	0.68
TabTransformer [18]	86.11%	0.77	80.56%	0.54
TANGOS [19]	86.12%	0.39	86.12%	0.73

Table 2: Performance on ADNI and WDBC datasets.

Dataset	Model	Acc.#	AUC#	Acc.*	AUC*
ADNI	**Target Attribute: AD123**				
	TabNet [5]	59.26%	0.68	66.67%	0.54
	NODE [27]	59.26%	0.68	66.67%	0.54
	TabSeq (ours)	66.67%	0.67	67.68%	0.61
	TabTransformer [18]	66.67%	0.40	74.07%	0.70
	TANGOS [19]	74.07%	0.70	74.07%	0.73
	Target Attribute: ABETA12				
	TabNet [5]	51.85%	0.52	74.07%	0.67
	NODE [27]	59.26%	0.57	59.26%	0.68
	TabSeq (ours)	64.44%	0.55	75.13%	0.71
	TabTransformer [18]	59.26%	0.50	59.26%	0.50
	TANGOS [19]	44.45%	0.50	44.45%	0.62
WDBC	TabNet [5]	91.86%	0.91	94.18%	0.99
	NODE [27]	91.86%	0.99	93.71%	0.98
	TabSeq (ours)	94.65%	0.91	94.71%	0.98
	TabTransformer [18]	87.08%	0.91	60%	0.37
	TabPFN [17]	94.19%	0.94	40.69%	0.50
	TANGOS [19]	60%	0.76	60%	0.85

demonstrate how feature ordering can improve model accuracy and AUC scores. The TabNet, NODE, and TabSeq versions performed better when feature ordering was used. Notably, compared to its excellent performance without feature ordering, the TabSeq model's accuracy increased, reaching 94.71% with feature ordering. TabTransformer boosts AUC; TANGOS shows lower accuracy; TabPFN excels in feature-limited datasets like WDBC but without feature ordering. We employed k-means clustering with 5 clusters for the ADNI dataset and 3 for the WDBC dataset, ordering them ascendingly. As Fig. 3 shows, increasing cluster numbers initially enhanced performance for most of the cases but eventually declined after specific points. These results affirm that our feature ordering approach boosts model accuracy and AUC across datasets, noting its importance in advancing deep learning for tabular data.

5 Conclusion

This paper introduces TabSeq, a novel framework that employs feature ordering to enhance deep learning's performance on tabular datasets significantly. By integrating local ordering and global ordering within a DAE equipped with an MHA mechanism, our method systematically optimizes feature sequences to improve learning efficacy. Studies conducted on raw antibody microarray data and other medical datasets have underscored the capability of strategic feature sequencing to yield substantial performance gains. These empirical results reinforce feature ordering's potential as a game-changing technique in deep learning for tabular data. Its effectiveness is predominantly seen in sequence-based architectures, and its performance on low-dimensional datasets remains to be determined. Further research is needed to refine and extend the method's applicability to various architectures and datasets.

Acknowledgement

This work was supported in part by grants from the US National Science Foundation (Award #1920920, #2125872, and #2223793).

References

[1] ADNI | Alzheimer's Disease Neuroimaging Initiative. https://adni.loni.usc.edu/, (Accessed on 03/07/2024)

[2] Breast Cancer Wisconsin (Diagnostic) - UCI Machine Learning Repository. https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic, (Accessed on 03/07/2024)

[3] Data_set_190503.xlsx. https://figshare.com/s/3bd3848a28ef6e7ae9a9, (Accessed on 03/07/2024)

[4] Amodei, D., et al.: Deep Speech 2: End-to-End Speech Recognition in English and Mandarin. In: Proc. Int. Conf. on Machine Learning. pp. 173–182. PMLR (2016)

[5] Arik, S.Ö., Pfister, T.: TabNet: Attentive Interpretable Tabular Learning. In: Proc. AAAI Conf. on Artificial Intelligence. vol. 35, pp. 6679–6687 (2021)

[6] Badirli, S., et al.: Gradient Boosting Neural Networks: GrowNet. arXiv preprint arXiv:2002.07971 (2020)

[7] Chen, P., et al.: HYTREL: Hypergraph-Enhanced Tabular Data Representation Learning. Advances in Neural Information Processing Systems **36** (2024)

[8] Chen, S., et al.: ReConTab: Regularized Contrastive Representation Learning for Tabular Data. arXiv preprint arXiv:2310.18541 (2023)

[9] Chen, Y., et al.: MAMA Net: Multi-Scale Attention Memory Autoencoder Network for Anomaly Detection. IEEE Trans. Med. Imaging **40**(3), 1032–1041 (2020)

[10] DE, R.: Learning Representations by Back-Propagation Errors. Nature **323**, 533–536 (1986)

[11] Devlin, J., et al.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805 (2018)

[12] Du, T., et al.: ReMasker: Imputing Tabular Data with Masked Autoencoding. arXiv preprint arXiv:2309.13793 (2023)

[13] Gorishniy, Y., et al.: Revisiting Deep Learning Models for Tabular Data. Advances in Neural Information Processing Systems **34**, 18932–18943 (2021)

[14] Hazimeh, H., et al.: The Tree Ensemble Layer: Differentiability Meets Conditional Computation. In: Proc. Int. Conf. on Machine Learning. pp. 4138–4148. PMLR (2020)

[15] He, K., et al.: Deep Residual Learning for Image Recognition. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition. pp. 770–778 (2016)

[16] Hegselmann, S., et al.: TabLLM: Few-Shot Classification of Tabular Data with Large Language Models. In: International Conference on Artificial Intelligence and Statistics. pp. 5549–5581. PMLR (2023)

[17] Hollmann, N., et al.: TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second. In: NeurIPS 2022 First Table Representation Workshop

[18] Huang, X., et al.: TabTransformer: Tabular Data Modeling Using Contextual Embeddings. arXiv preprint arXiv:2012.06678 (2020)

[19] Jeffares, A., et al.: TANGOS: Regularizing Tabular Neural Networks through Gradient Orthogonalization and Specialization. In: The Eleventh International Conference on Learning Representations

[20] Klambauer, G., et al.: Self-Normalizing Neural Networks. Advances in Neural Information Processing Systems **30** (2017)

[21] Kostal, L., et al.: Measures of Statistical Dispersion Based on Shannon and Fisher Information Concepts. Information Sciences **235**, 214–223 (2013)

[22] Krizhevsky, A., et al.: ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems **25** (2012)

[23] Nair, V., Hinton, G.E.: Rectified Linear Units Improve Restricted Boltzmann Machines. In: ICML 2010 (2010)

[24] Ohlsson, M., et al.: Proteomic Data Analysis for Differential Profiling of the Autoimmune Diseases SLE, RA, SS, and ANCA-Associated Vasculitis. Journal of Proteome Research **20**(2), 1252–1260 (2020)

[25] Oord, A.v.d., et al.: WaveNet: A Generative Model for Raw Audio. arXiv preprint arXiv:1609.03499 (2016)

[26] Philip, B., et al.: ASENN: Attention-Based Selective Embedding Neural Networks for Road Distress Prediction. Journal of Big Data **10**(1), 164 (2023)

[27] Popov, S., et al.: Neural Oblivious Decision Ensembles for Deep Learning on Tabular Data. arXiv preprint arXiv:1909.06312 (2019)

[28] Ruiz, C., et al.: High Dimensional, Tabular Deep Learning with an Auxiliary Knowledge Graph. Advances in Neural Information Processing Systems **36** (2024)

[29] de Sá, C.R.: Variance-Based Feature Importance in Neural Networks. In: International Conference on Discovery Science. pp. 306–315. Springer (2019)

[30] Senior, A., et al.: Deep Neural Networks for Acoustic Modeling in Speech Recognition. IEEE Signal Processing Magazine (2012)

[31] Song, W., et al.: AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks. In: Proc. 28th ACM Int. Conf. on Information and Knowledge Management. pp. 1161–1170 (2019)

[32] Tate, S.R.: Band Ordering in Lossless Compression of Multispectral Images. IEEE Transactions on Computers **46**(4), 477–483 (1997)

[33] Th. Gries, S.: Analyzing Dispersion. In: A Practical Handbook of Corpus Linguistics, pp. 99–118. Springer (2021)

[34] Tian, T., Fang, Z.F.: Attention-Based Autoencoder Topic Model for Short Texts. Procedia Computer Science **151**, 1134–1139 (2019)

[35] Tihon, S., et al.: DAEMA: Denoising Autoencoder with Mask Attention. In: International Conference on Artificial Neural Networks. pp. 229–240. Springer (2021)

[36] Vaswani, A., et al.: Attention Is All You Need. Advances in Neural Information Processing Systems **30** (2017)

[37] Vincent, P., et al.: Extracting and Composing Robust Features with Denoising Autoencoders. In: Proc. 25th Int. Conf. on Machine Learning. pp. 1096–1103 (2008)

[38] Wang, C., et al.: Bandwidth Minimization Problem. In: MOSIM 2014, 10ème Conférence Francophone de Modélisation, Optimisation et Simulation (2014)

[39] Wang, R., et al.: DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-Scale Learning to Rank Systems. In: Proc. Web Conf. 2021. pp. 1785–1797 (2021)

[40] Wang, Z., et al.: MediTab: Scaling Medical Tabular Data Predictors via Data Consolidation, Enrichment, and Refinement. arXiv preprint arXiv:2305.12081 (2023)

[41] Wojtas, M., Chen, K.: Feature Importance Ranking for Deep Learning. Advances in Neural Information Processing Systems **33**, 5105–5114 (2020)

[42] Wu, J., et al.: SwitchTab: Switched Autoencoders Are Effective Tabular Learners. arXiv preprint arXiv:2401.02013 (2024)

[43] Yak, S., et al.: IngesTables: Scalable and Efficient Training of LLM-Enabled Tabular Foundation Models. In: NeurIPS 2023 Table Represent. Learning Wkshp (2023)

[44] Yan, J., et al.: T2G-Former: Organizing Tabular Features into Relation Graphs Promotes Heterogeneous Feature Interaction. In: AAAI Conf. on AI (2023)

[45] Zhou, J.P., et al.: TAFA: Two-Headed Attention Fused Autoencoder for Context-Aware Recommendations. In: Proceedings of the 14th ACM Conference on Recommender Systems. pp. 338–347 (2020)