# GFSNetwork: Differentiable Feature Selection via Gumbel-Sigmoid Relaxation

Witold Wydmański<sup>1,2</sup> (⋈) and Marek Śmieja<sup>1</sup>

- <sup>1</sup> Faculty of Mathematics and Computer Science, Jagiellonian University wwydmanski@gmail.com
  - <sup>2</sup> Malopolska Centre of Biotechnology, Jagiellonian University

**Abstract.** Feature selection in deep learning remains a critical challenge, particularly for high-dimensional tabular data where interpretability and computational efficiency are paramount. We present GFSNetwork, a novel neural architecture that performs differentiable feature selection through temperature-controlled Gumbel-Sigmoid sampling. Unlike traditional methods, where the user has to define the requested number of features, GFSNetwork selects it automatically during an end-toend process. Moreover, GFSNetwork maintains constant computational overhead regardless of the number of input features. We evaluate GFS-Network on a series of classification and regression benchmarks, where it consistently outperforms recent methods including DeepLasso, attention maps, as well as traditional feature selectors, while using significantly fewer features. Furthermore, we validate our approach on realworld metagenomic datasets, demonstrating its effectiveness in highdimensional biological data. Concluding, our method provides a scalable solution that bridges the gap between neural network flexibility and traditional feature selection interpretability. We share our python implementation of GFSNetwork at https://github.com/wwydmanski/ GFSNetwork, as well as a PyPi package (gfs\_network).

**Keywords:** feature selection, gumbel-softmax, classification, regression

#### 1 Introduction

Feature selection remains a fundamental challenge in deep learning, particularly for high-dimensional tabular data where interpretability and computational efficiency are crucial. Traditional feature selection methods, such as L1 regularization, attention-based mechanisms, and classical statistical techniques, often struggle with scalability and robustness in deep learning settings. As neural networks typically operate in high-dimensional spaces, selecting a minimal yet informative subset of features without compromising predictive performance is an open problem.

Existing approaches to feature selection can broadly be categorized into filter [6,17,14], wrapper [9,11], and embedded methods [15,18]. Filter methods rank features based on statistical relevance but remain independent of the learning

model, potentially overlooking complex feature interactions. Wrapper methods iteratively select features using a model's predictive performance as a criterion, but they suffer from high computational costs. Embedded methods, such as L1 regularization or attention-based mechanisms, integrate feature selection within the learning process but may introduce instability or lack fine-grained control over feature importance.

Most of the feature selection algorithms have complexity linearly dependent on the number of input features, thus making them slow for larger dimensions of the data. Moreover, typical methods treat the final number of features as a hyperparameter – selecting it wrongly may lead to suboptimal results, requiring multiple retrainings of the final model.

To address these limitations, we propose **GFSNetwork**, a novel neural architecture that performs automatic and efficient feature selection using temperature-controlled Gumbel-Sigmoid sampling. Unlike traditional methods, GFSNetwork autonomously selects the most relevant features during training, eliminating the need for manual selection or iterative retraining. Moreover, our approach maintains a constant computational overhead, ensuring that the model does not slow down as the number of features increases. This makes GFSNetwork highly scalable and well-suited for high-dimensional datasets where traditional methods often become computationally prohibitive.

We evaluate GFSNetwork on a range of standardized classification and regression benchmarks (Section 4.2), where it consistently outperforms existing feature selection techniques while using significantly fewer features. Additionally, we demonstrate its effectiveness in real-world metagenomic datasets (Section 4.3), highlighting its applicability in high-dimensional biological data. To further show the real-world potential, we analyze the computational complexity of our method and compare it with other feature selection algorithms (section 4.4). Furthermore, we visualize the selected features on the MNIST dataset (Section 4.5), providing insights into the interpretability of our method in image-based feature selection. By bridging the gap between neural network flexibility and interpretable feature selection, our method provides a scalable and robust solution to feature selection in deep learning.

Our work makes the following key contributions:

- We introduce GFSNetwork, a novel neural architecture that performs automatic and differentiable feature selection using temperature-controlled Gumbel-Sigmoid sampling.
- We show that our method maintains constant computational overhead, ensuring efficiency even as the number of input features increases, making it highly scalable for high-dimensional datasets.
- We demonstrate that GFSNetwork selects a minimal yet informative subset of features without compromising predictive performance, outperforming traditional feature selection methods on standardized benchmarks.
- We validate our approach on an openml-based benchmark, as well as on real-world metagenomic datasets, highlighting its effectiveness in highdimensional biological data analysis.

 We showcase the interpretability of GFSNetwork by visualizing its selected features on the MNIST dataset, demonstrating its applicability to imagebased feature selection.

# 2 Related Work

Authors of [5] introduce feature importance analysis by means of attention maps of a tabular transformer, while [10,7] show that appropriately constrained neural network can be used for selecting important features. In [2], authors explore the importance of feature selection, focusing on filter, wrapper, and embedded methods. The study highlights how feature selection can improve model performance, reduce redundancy, minimize overfitting, and enhance computational efficiency. Additionally, it discusses the integration of feature selection with deep learning and explainable AI to address scalability and fairness in large-scale applications.

A dynamic feature selection (DFS) method [4] is dependent on sequential queries of features based on conditional mutual information. This approach, theoretically appealing, outperforms existing methods by efficiently selecting features without relying on reinforcement learning. Authors of [16] introduce a Sequential Attention mechanism for feature selection, utilizing attention weights as proxies for feature importance. This method achieves state-of-the-art results and offers theoretical insights by connecting to the Orthogonal Matching Pursuit algorithm, thereby enhancing our understanding of attention mechanisms in neural networks.

Addressing computational challenges in large datasets, [13] extend the adaptive query model to Orthogonal Matching Pursuit for non-submodular functions. Their algorithm achieves exponentially fast parallel runtime and incorporates fairness constraints into the feature selection process, providing strong approximation guarantees applicable to various parametric models.

These developments reflect the ongoing evolution of feature selection methodologies, highlighting the importance of adapting and integrating various techniques to meet the complex demands of modern machine learning applications.

# 3 The proposed model

In this section we introduce GFSNetwork, a neural network approach for automatic selection of features, which are relevant for a given machine learning task. First, we give a brief overview of GFSNetwork. Next, we describe its main building blocks. Finally, we summarize the training algorithm and inference phase. We conclude this section with a discussion of GFSNetwork.

#### 3.1 Overview of GFSNetwork

GFSNetwork is a neural network approach to differentiable feature selection. It retrieves features which are the most informative for solving a given classification or regression task.

#### 4 W. Wydmański, M. Śmieja

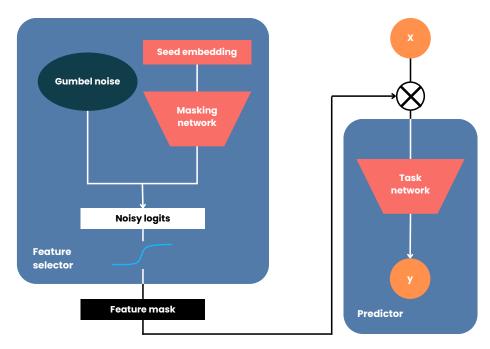


Fig. 1: Architecture of GFSNetwork. Our method consists of two parts - masking and task network. The first one, in conjunction with Gumbel noise, creates a binary mask which is then used by the second network to output either a class or real number. This way we simultaneously optimize the number of features and performance of a classifier that's based on them.

The architecture of GFSNetwork is split into two components: masking and task networks, see Figure 1 for the illustration. While the masking network returns a mask representing selection of the features, the task network solves the underlying task using only the ones which were not masked out by the masking network. The loss function of GFSNetwork combines the cross-entropy (for classification) or mean-square error (for regression) with the penalty term, which reduces the number of selected features. In consequence, task network plays a role of discriminator, which verifies the usefulness of the selected features for a given task.

In contrast to traditional methods for feature selection, which iteratively selects or reduce features, GFSNetwork uses a differentiable mechanism for learning a mask based on Gumbel-Sigmoid relaxation of discrete distribution [8]. This design ensures that the computational time remains nearly constant regardless of input dimensionality, making it particularly efficient for high-dimensional data.

The following sections describe details behind GFSNetwork.

#### 3.2 Masking network

Masking network  $f: \mathbb{R}^{D_e} \to \mathbb{R}^D$  is responsible for generating a mask that indicates features selected for a given dataset  $\{(x_i,y_i)\}_{i=1}^N \subset \mathbb{R}^D$ . Given a randomly initialized input embedding  $e \in \mathbb{R}^{D_e}$ , the network f outputs D-dimensional vector  $w = f_{\phi}(e) \in \mathbb{R}^D$ , which determines the mask. More precisely, the output vector  $w = (w_1, \dots, w_D)$  is transformed via a sequence of D Gumbel-Sigmoid functions to the (non-binary) mask vector  $m = (m_1, \dots, m_D)$ , where  $m_i = GS(w_i; \tau)$  is given by the Gumbel-Sigmoid function with the temperature parameter  $\tau > 0$ . Let us recall that Gumbel-Sigmoid function given is by:

$$GS(w_i; \tau) = \sigma\left(\frac{w_i + g_i}{\tau}\right)$$

where  $g_i \sim -\log(-\log(u))$  with  $u \sim \text{Uniform}(0,1)$  is the Gumbel noise,  $\sigma$  is the sigmoid function, and  $\tau > 0$  is the temperature parameter.

In consequence, the mask  $m = (m_1, \ldots, m_D)$  sampled from the GS distribution takes a continuous (non-binary) form. As  $\tau$  decreases, the mask approaches to binary vector, which represents final discrete mask. Slow decrease of the temperature  $\tau$  allows us to learn optimal mask during network training.

#### 3.3 Task network

To learn the optimal mask, we need to verify whether it is informative for the underlying task (e.g. classification). To this end, we first apply a mask m to the input example x, by the element-wise multiplication  $x_m = m \odot x$ . Next, we feed a task network  $g: \mathbb{R}^D \to Y$  with  $x_m$  to obtain the final output  $g(x_m)$ . By applying a typical loss function  $\mathcal{L}_{task}(y; g(x_m))$ , e.g. cross-entropy or mean-square error, we quantify the importance of features selected by m. Additionally, to encourage the model to eliminate redundant features, we penalize the model for every added feature by:

$$\mathcal{L}_{select} = \sum_{j} m_{j}.$$

The complete loss function is then given by:

$$\mathcal{L}_{total} = \mathcal{L}_{task} + \lambda \mathcal{L}_{select},$$

where  $\lambda$  is hyperparameter. Thanks to the Gumbel-Sigmoid relaxation of the discrete mask distribution, we can learn the mask during end-to-end differentiable training.

#### 3.4 Training process

Let us summarize the complete training algorithm, see Algorithm 1.

The training starts with a fixed temperature  $\tau = \tau_0$  and randomly initialized embedding e. Given an embedding e, the masking network f returns a mask

vector  $m = (m_1, \ldots, m_D)$  using Gumbel-Sigmoid functions. Every continuous mask vector m sampled from GS is then applied to a mini-batch to construct the reduced vectors  $x_m = m \odot x$ . This vector goes to the task network g, which returns the response for a given task  $g(x_m)$ . The loss function  $\mathcal{L}_{total}$  is calculated and the gradient is propagated to: (1) embedding vector e, (2) weights of f and g. In particular, by learning the parameters of  $f_{\phi}$ , we learn the mask vector.

### Algorithm 1 GFSNetwork Training Procedure

```
1: Input: Dataset \mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N, batch size B, initial temperature \tau_0 = 2.0, decay
       rate \alpha = 0.997, total epochs E, feature selection balance parameter \lambda
  2: Initialize: Embedding vector \mathbf{e} \in \mathbb{R}^{d_e}, masking network f_{\phi}, task network g_{\theta}
 3: \tau \leftarrow \tau_0
  4: for epoch = 1 to E do
             for each mini-batch \mathbf{X} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^B \subset \mathcal{D} \ \mathbf{do}
  5:
                                                                                                ▷ Compute logits for feature mask
  6:
                   \mathbf{g} \leftarrow -\log(-\log(\mathbf{u})) where \mathbf{u} \sim \text{Uniform}(0,1)
  7:
                                                                                                                    ▷ Sample Gumbel noise
                   \mathbf{m} \leftarrow \sigma \left( (\mathbf{w} + \mathbf{g}) / \tau \right)
 8:
                                                                                         ▷ Generate mask via Gumbel-Sigmoid
                   X_{\mathrm{masked}} \leftarrow X \odot m
 9:
                                                                                                                       ▶ Mask input features
                    \hat{\mathbf{Y}} \leftarrow g_{\theta}(\mathbf{X}_{\text{masked}})
10:
                                                                                          ▷ Forward pass through task network
11:
                   \mathcal{L}_{\text{task}} \leftarrow \begin{cases} -\sum_{i=1}^{B} \sum_{c=1}^{C} y_{i,c} \log(\hat{y}_{i,c}) & \text{for classification} \\ \frac{1}{B} \sum_{i=1}^{B} (y_i - \hat{y}_i)^2 & \text{for regression} \end{cases}
12:
                    \mathcal{L}_{\text{select}} \leftarrow \frac{1}{D} \sum_{j=1}^{D} m_{j}
\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{task}} + \lambda \cdot \mathcal{L}_{\text{select}}
13:
14:
                                                                               ▷ Combine task and feature selection losses
15:
                    \mathbf{e} \leftarrow \mathbf{e} - \eta_1 \nabla_{\mathbf{e}} \mathcal{L}_{\mathrm{total}}
16:
                                                                                                                         ▶ Update embedding
                    \phi \leftarrow \phi - \eta_1 \nabla_{\phi} \mathcal{L}_{total}
                                                                                                              ▶ Update masking network
17:
                    \theta \leftarrow \theta - \eta_2 \nabla_{\theta} \mathcal{L}_{total}
                                                                                                                      ▷ Update task network
18:
             end for
19:
20:
             \tau \leftarrow \tau \cdot \alpha
                                                                                                                       ▶ Anneal temperature
21: end for
22: Output:
                                                                                                              \mathbf{s} = (s_1, \dots, s_D)
                                                          importance
                                                                                         scores
                                                                                                                                                      where
       s_j = \sum_{i=1}^{N} \mathbf{1}[\sigma(f_{\phi}(\mathbf{e}))_j > 0.5]
```

This approach allows the network to initially explore the feature space broadly and gradually commit to specific features as training progresses. The feature selection regularization term  $\mathcal{L}_{select}$  can be configured in two modes: either penalizing deviation from a target number of features, or directly encouraging sparsity by minimizing the number of active features.

A critical aspect of our algorithm is the temperature annealing schedule. We begin with a high temperature ( $\tau=2.0$ ), which produces soft masks that allow gradient flow to all features. As training progresses, the temperature decays exponentially (typically with  $\alpha=0.997$ ), causing the masks to become increasingly binary. This gradual transition serves multiple purposes:

- It allows the network to initially explore the full feature space.

- It enables progressive commitment to more discrete feature selection decisions.
- It leads to convergence on a nearly binary feature selection mask by the end
  of training.

The annealing process effectively functions as a curriculum, starting with easier optimization (continuous selection) and progressively transitioning to harder optimization (discrete selection).

#### 3.5 Feature Importance Quantification

After training, we quantify the importance of each feature (see the last line in Algorithm 1) by directly applying the learned selection mechanism with hard Gumbel-Sigmoid activation:

- 1. Computing the feature logits from the trained embedding:  $\mathbf{w} = \mathbf{f}_{\phi}(\mathbf{e})$ .
- 2. Applying a hard threshold at  $\tau = 0$ :  $\mathbf{m} = \sigma(\mathbf{w}) > 0.5$ .
- 3. Interpreting the resulting binary vector as the feature selection mask.

This process produces a deterministic feature selection that clearly identifies relevant features for the task. Since our feature selection mechanism is parameterized by a single embedding vector that is independent of specific input examples, the selected features remain constant across the entire dataset.

The resulting binary mask can be directly used to filter features, or features can be ranked by their logit values when a specific top-k selection is desired. Importantly, since the selection mechanism was jointly optimized with the task objective, the selected features capture both individual importance and interactive effects relevant to the specific task.

## 3.6 Discussion

The key innovation in our approach is the application of the Gumbel-Sigmoid trick to generate differentiable binary masks. This technique allows us to perform discrete selection during the forward pass while maintaining gradient flow during backpropagation.

During training, the feature mask is generated by adding Gumbel noise to the logits and applying a sigmoid function with a temperature parameter. This process can be conceptualized as a continuous relaxation of binary sampling that becomes increasingly discrete as the temperature decreases. The temperaturecontrolled sampling enables our model to transition smoothly from exploration (high temperature) to exploitation (low temperature).

This exploration-exploitation tradeoff parallels fundamental concepts in reinforcement learning. The feature selection problem can be framed as a contextual multi-armed bandit, where each feature represents an "arm" with an unknown reward distribution. Initially, with high temperature, our model explores the feature space broadly – similar to how RL agents employ  $\epsilon$ -greedy or softmax

Statis the best falls.								
FS method	AL	EY	GE	HE	HI	JA	ОТ	Median rank
No FS	0.941	0.538	0.466	0.366	0.798	0.703	0.773	11.0
Univariate	0.96	0.575	0.515	0.379	0.811	0.715	0.808	5.0
Lasso	0.949	0.547	0.458	0.380	0.812	0.715	0.805	8.5
1L Lasso	0.952	0.564	0.474	0.375	0.811	0.715	0.796	8.5
AGL	0.958	0.578	0.473	0.386	0.810	0.718	0.799	6.0
LassoNet	0.954	0.552	0.495	0.385	0.811	0.715	0.783	7.0
AM	0.953	0.554	0.498	0.382	0.813	0.722	0.801	6.0
RF	0.955	0.589	0.594	0.386	0.814	0.720	0.806	3.0
XGBoost	0.956	0.59	0.502	0.385	0.812	0.720	0.805	4.0
Deep Lasso	0.959	0.573	0.485	0.383	0.814	0.720	0.802	5.0
GFSNetwork	0.96	0.599	0.507	0.374	0.809	0.733	0.807	2.0

Table 1: Classification accuracy for the case of random features. GFSNetwork obtains the best rank.

policies with high entropy to explore their environment. As training progresses and temperature decreases, the model increasingly exploits high-value features, analogous to how RL agents converge toward optimal policies.

The temperature annealing schedule thus functions as an adaptive exploration strategy—initially permitting broad sampling of the feature space before gradually committing to the most informative features. This approach prevents premature convergence to suboptimal feature subsets, a common challenge in both feature selection and reinforcement learning. Furthermore, the end-to-end training with the primary task provides an implicit reward signal that guides the feature selection policy, where improvements in task performance reinforce the selection of beneficial features through gradient updates to the selection parameters.

# 4 Experiments

To evaluate the effectiveness of GFSNetwork, we conducted extensive experiments across multiple datasets and compared our approach against state-of-the-art feature selection methods. Our evaluation focused on two key aspects: (1) performance on classification and regression tasks, and (2) application to high-dimensional metagenomic datasets. Additionally, we analyzed the computational efficiency of our method compared to existing approaches.

We follow a recent benchmark introduced in [3]. Reported results were achieved by extending the code base with GFSNetwork. The benchmark consists of three parts. Each of them introduces some type of noise to the given datasets – either fully random features, features corrupted with gaussian noise, or a set of second-order features, created by multiplying randomly selected features from the original dataset. We analyzed a scenario in which 50% of the features in each

between the best rank and outperforms officer methods on most datasets.								
	AL	$\mathbf{E}\mathbf{Y}$	GE	$_{ m HE}$	HI	JA	OT	Median
FS method								$\operatorname{rank}$
No FS	0.946	0.557	0.525	0.37	0.802	0.703	0.778	11.0
Univariate	0.955	0.556	0.514	0.346	0.81	0.717	0.795	9.0
Lasso	0.955	0.548	0.512	0.382	0.813	0.713	0.796	6.0
1L Lasso	0.955	0.566	0.515	0.382	0.812	0.718	0.795	8.0
AGL	0.953	0.588	0.538	0.386	0.813	0.722	0.796	4.0
LassoNet	0.955	0.57	0.556	0.382	0.811	0.719	0.795	7.0
AM	0.955	0.583	0.527	0.381	0.814	0.722	0.797	4.0
RF	0.951	0.574	0.568	0.383	0.81	0.724	0.788	6.0
XGBoost	0.954	0.583	0.51	0.385	0.815	0.722	0.803	3.5
Deep Lasso	0.955	0.577	0.525	0.388	0.815	0.721	0.801	4.5
GFSNetwork	0.957	0.604	0.552	0.366	0.815	0.736	0.801	2.0

Table 2: Classification accuracy for the case of corrupted features. GFSNetwork obtains the best rank and outperforms other methods on most datasets.

dataset were created artificially. By applying feature selection algorithms, we aim to eliminate redundant features and improve performance on a downstream task.

#### 4.1 Experimental Setup

Baseline Methods: Following [3], we compared GFSNetwork against ten established feature selection methods:

- No Feature Selection (No FS): Using all available features
- Univariate Selection: Statistical tests for feature ranking
- Lasso: L1-regularized linear models
- 1L Lasso: Single-layer neural network with L1 regularization
- AGL: Adaptive Group Lasso [7]
- LassoNet: Neural network with hierarchical sparsity [10]
- AM: Attention Maps for feature importance [5]
- RF: Random Forest importance
- XGBoost: Gradient boosting importance [1]
- Deep Lasso: Deep neural network with L1 regularization [3]

All of the feature selection methods use MLP as a downstream classifier.

**Evaluation Metrics:** For each dataset and method, we report performance metrics specific to the task (accuracy for classification, mean squared error for regression). We also compute the median rank across all datasets to provide an overall performance assessment.

# 4.2 Classification and Regression Performance

Table 1 presents the classification performance across seven benchmark datasets with random features. GFSNetwork achieves the best median rank (2.0) among all methods, outperforming the second-best method (RF) by a significant margin.

FS method	AL	EY	GE	HE	HI	JA	ОТ	Median rank
No FS	0.96	0.631	0.605	0.383	0.811	0.719	0.8	7.0
Univariate	0.961	0.584	0.582	0.357	0.817	0.724	0.798	10.0
Lasso	0.955	0.608	0.59	0.366	0.816	0.724	0.806	8.0
1L Lasso	0.959	0.634	0.571	0.38	0.815	0.728	0.808	6.0
AGL	0.961	0.637	0.594	0.383	0.807	0.73	0.806	3.5
LassoNet	0.959	0.641	0.611	0.379	0.816	0.724	0.797	7.0
AM	0.961	0.622	0.604	0.381	0.819	0.73	0.802	5.0
RF	0.958	0.639	0.619	0.37	0.818	0.735	0.801	3.0
XGBoost	0.87	0.635	0.604	0.373	0.818	0.734	0.805	5.5
Deep Lasso	0.961	0.648	0.6	0.384	0.815	0.733	0.805	4.0
GFSNetwork	0.959	0.623	0.606	0.367	0.815	0.736	0.807	7.0

Table 3: Classification accuracy for the case of second-order features.

Notably, our algorithm achieves the highest scores on three datasets (AL, EY, and JA) and competitive performance on the remaining datasets.

Similarly, for datasets with corrupted features (Table 2), GFSNetwork again achieves the best median rank (2.0), demonstrating its robustness to noisy data. The method particularly excels on datasets with simple feature interactions, where it outperforms traditional methods by identifying complementary feature subsets.

Interestingly, when evaluating on datasets with second-order features (Table 3), we observed a notable decline in GFSNetwork's performance. While maintaining strong performance on individual metrics (achieving the highest score on the JA dataset), its overall median rank dropped to 7.0, significantly behind RF (3.0) and Deep Lasso (4.0).

This performance degradation suggests that GFSNetwork struggles with datasets containing engineered second-order features (products of randomly selected original features). In such scenarios, information becomes redundantly encoded across both original and derived features. The presence of these engineered features creates a more challenging selection landscape where the optimal strategy isn't simply identifying relevant features, but rather selecting an efficient subset from groups of correlated or redundant features. This scenario highlights a limitation in our current approach when dealing with datasets containing significant feature redundancy through engineered interactions.

The results presented in Table 4 show that our method achieves 3rd place of the ranking, which proves that GFSNetwork can be successfully applied in a wide range of machine learning tasks including both classification and regression.

#### 4.3 Metagenomic Dataset Analysis

To evaluate GFSNetwork's effectiveness on high-dimensional biological data, we applied it to 24 metagenomic datasets obtained from Curated Metagenomics

4.5

method in regression benefinark.								
	СН	НО	MI	YE	Median			
FS method					rank			
No FS	-0.475	-0.607	-0.909	-0.797	10.0			
Univariate	-0.451	-0.62	-0.92	-0.828	11.0			
Lasso	-0.449	-0.602	-0.903	-0.795	7.5			
1L Lasso	-0.447	-0.581	-0.902	-0.78	4.8			
AGL	-0.45	-0.561	-0.902	-0.78	4.8			
LassoNet	-0.452	-0.551	-0.905	-0.777	5.0			
AM	-0.449	-0.555	-0.905	-0.78	4.8			
RF	-0.453	-0.565	-0.904	-0.786	7.5			
XGBoost	-0.454	-0.553	-0.892	-0.779	2.5			
Deep Lasso	-0.447	-0.567	-0.895	-0.776	2.8			

Table 4: Regression – MSE for the case of corrupted features. GFSNetwork is third best method in regression benchmark.

Data [12]. These datasets represent a particularly challenging domain with high feature dimensionality (308-718 features) and complex biological interactions. Both versions - before and after feature selection - use downstream Random Forest classifier.

-0.583

-0.894

-0.784

The results presented in Table 5 demonstrate that GFSNetwork consistently achieves performance gains while drastically reducing feature dimensionality. On average, GFSNetwork selected only 5.8% of the original features while not diminshing the results in a statistically significant way. In 17 out of 24 datasets, our method improved or maintained performance compared to using all features. Figure 2 illustrates the process of feature selection. Observe that GFSNetwork deeply explores the space of all features and selects the optimal set of features at the end of the training.

#### 4.4 Computational Complexity Estimation

GFSNetwork

-0.446

The estimated computational complexity reveal striking differences across feature selection methods, see Figure 3. Denoting time complexity as an exponential function of number of features  $t \approx D^{\alpha}$ , our empirical analysis shows that GFSNetwork demonstrates near-constant time scaling ( $\alpha \approx 0.08$ ), significantly outperforming all competing approaches as dimensionality increases.

ANOVA F-value and Mutual Information exhibit linear scaling ( $\alpha \approx 1.0$ ), while Random Forest shows sub-linear behavior ( $\alpha \approx 0.53$ ). In contrast, Recursive Feature Elimination with Linear SVC demonstrates superlinear scaling ( $\alpha \approx 1.41$ ), causing its performance to degrade more rapidly with increasing feature dimensions. The confidence intervals over 5 runs (RHS or Figure 3) indicate these estimates are statistically robust across tested dimensionalities. This assessment provides compelling evidence for GFSNetwork's exceptional efficiency advantage in high-dimensional feature selection tasks, with its nearly

Table 5: Performance on metagenomic data reduced with GFSNetwork. Although GFSNetwork heavily reduces data dimensionality it usually does not lead to the deterioration of the results. Each dataset's name is derived from the first author's surname and the year of publication.

dataset dataset	Score on full data	Score on GFSNetwork	No. of selected features	Original dimension
WirbelJ 2018	0.776	0.821	22	537
Rubel $\overline{MA}$ 2020	0.775	0.796	27	370
GuptaA = 2019	0.875	0.938	20	308
$ m JieZ\_2017$	0.762	0.770	37	683
$Thomas AM\_2018a$	0.817	0.917	33	477
$QinJ_2012$	0.616	0.622	40	651
$FengQ_2015$	0.833	0.889	22	606
$HanniganGD_2017$	0.817	0.533	21	292
$ZellerG_2014$	0.652	0.871	27	652
$AsnicarF_2021$	0.500	0.500	59	639
LiJ_2017	0.561	0.432	29	436
ZhuF_2020	0.768	0.739	30	480
$KeohaneDM\_2020$	0.469	0.531	31	381
$LifeLinesDeep\_2016$	0.500	0.500	66	646
$QinN_2014$	0.833	0.855	26	645
LiJ_2014	0.500	0.508	36	621
$VogtmannE\_2016$	0.694	0.694	30	540
YachidaS_2019	0.636	0.608	57	718
$LeChatelierE\_2013$	0.549	0.620	44	526
$ThomasAM_2019c$	0.627	0.764	28	519
$ThomasAM_2018b$	0.586	0.586	28	503
$YuJ_2015$	0.674	0.646	22	575
$NagySzakalD_2017$	0.917	0.958	17	438
$NielsenHB\_2014$	0.711	0.634	30	606

constant-time behavior representing a significant algorithmic advancement over conventional methods.

#### 4.5 Feature Selection Visualization on MNIST

To provide intuitive insights into how GFSNetwork selects discriminative features, we conducted visualization experiments on the MNIST handwritten digit dataset. Figure 4 (left) show the entropy of the selected features. It is evident that GFSNetwork focuses on more discriminative features (higher entropy) – the mean entropy of selected features equals 1.98 while the mean entropy of all features equals 1.43. Figure 4 (right) illustrates the pixels selected for MNIST dataset. Clearly, the model pay more attention to the center of the image, ignoring background regions. Finally, Figure 5 examines individual selected features for a sample digit, comparing their class-conditional activation distributions with

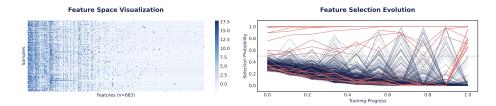


Fig. 2: Figure 3: Feature selection analysis showing the feature space representation (left) and selection probability evolution (right). The heatmap displays the distribution of features across samples, with color intensity indicating feature values. The evolution plot tracks selection probabilities throughout training progress, highlighting distinct patterns between consistently selected features (red) that maintain high probabilities either from initialization or emerge at later stages, versus non-selected features (blue) that exhibit diminishing selection probabilities over time.

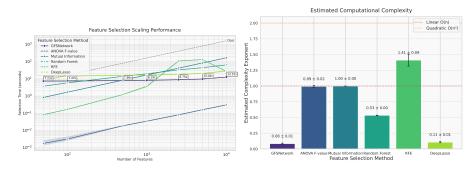


Fig. 3: The time requirements of GFSNetwork does not substantially increase with raising number of features.

non-selected pixels. Selected features consistently show more discriminative patterns across digit classes, with lower entropy values indicating higher information content. These visualizations demonstrate that GFSNetwork selects features in a manner that aligns with human intuition about discriminative regions for digit recognition.

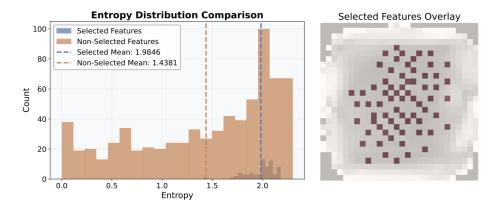


Fig. 4: Average entropy of selected features is significantly higher than the entropy of all features, which means that GFSNetwork selected features with discriminative potential (left). Moreover, selected pixel are localized in the center region of the image (right).

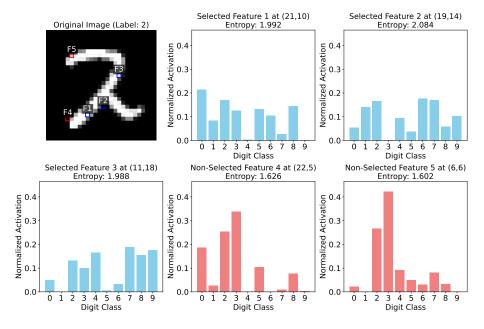


Fig. 5: Analysis of sample features (top-left) from MNIST dataset shows that entropy of selected features (F1-F3) is much higher than their non-selected counterparts (F4, F5). It confirms that GFSNetwork selects the most discriminative features.

#### 5 Conclusion

We presented GFSNetwork, a novel neural architecture for differentiable feature selection using temperature-controlled Gumbel-Sigmoid sampling. Our experiments demonstrate that GFSNetwork offers a compelling approach to feature selection, particularly for classification tasks with high-dimensional data. The method's near-constant computational scaling ( $\alpha \approx 0.08$ ) represents a significant advantage over traditional approaches, making it particularly valuable for datasets with thousands of features – such as omics datasets. The end-to-end learning approach allows GFSNetwork to optimize feature selection specifically for the task at hand, while automatically determining the appropriate number of features to select. As demonstrated in our MNIST visualization experiments, the selected features align with human intuition, providing interpretability that is valuable in domains like healthcare and biology.

However, our experiments with second-order features revealed limitations in handling engineered feature interactions. This suggests opportunities for future work to extend our approach to better capture complex relationships between features, perhaps by incorporating mechanisms that can model feature interdependencies.

In conclusion, GFSNetwork represents a significant step forward in automated feature selection, offering a scalable, end-to-end approach that bridges the gap between neural network flexibility and traditional feature selection interpretability.

Acknowledgments. Witold Wydmański is supported by the Ministry of Science grant no. PN/01/0195/2022 and NCN Sonata BIS grant number 2020/38/E/NZ2/00598. Marek Śmieja is supported by the NCN grant no. 2023/50/E/ST6/00169.

For the purpose of Open Access, the authors have applied a CC-BY public copyright license to any Author Accepted Manuscript (AAM) version arising from this submission.

**Disclosure of Interests.** The authors have no competing interests to declare that are relevant to the content of this article.

#### References

- 1. Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 785–794. KDD '16, ACM (2016)
- 2. Cheng, X.: A Comprehensive Study of Feature Selection Techniques in Machine Learning Models. Artificial Intelligence and Digital Technology 1(1), 65–78 (Nov 2024)
- 3. Cherepanova, V., Levin, R., Somepalli, G., Geiping, J., Bruss, C.B., Wilson, A.G., Goldstein, T., Goldblum, M.: A Performance-Driven Benchmark for Feature Selection in Tabular Deep Learning

- Covert, I.C., Qiu, W., Lu, M., Kim, N.Y., White, N.J., Lee, S.I.: Learning to maximize mutual information for dynamic feature selection. In: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (eds.) Proceedings of the 40th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 202, pp. 6424–6447. PMLR (23–29 Jul 2023)
- Gorishniy, Y., Rubachev, I., Khrulkov, V., Babenko, A.: Revisiting Deep Learning Models for Tabular Data (Oct 2023)
- 6. Guyon, I., Elisseeff, A.: An Introduction to Variable and Feature Selection
- 7. Ho, L.S.T., Richardson, N., Tran, G.: Adaptive Group Lasso Neural Network Models for Functions of Few Variables and Time-Dependent Data (Dec 2021)
- 8. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with gumbel-softmax (2017)
- 9. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artificial Intelligence 97(1), 273–324 (Dec 1997)
- Lemhadri, I., Ruan, F., Abraham, L., Tibshirani, R.: LassoNet: A Neural Network with Feature Sparsity (Jun 2021)
- 11. Maldonado, S., Weber, R.: A wrapper method for feature selection using Support Vector Machines. Information Sciences 179(13), 2208–2217 (Jun 2009)
- Pasolli, E., Schiffer, L., Manghi, P., Renson, A., Obenchain, V., Truong, D.T., Beghini, F., Malik, F., Ramos, M., Dowd, J.B., Huttenhower, C., Morgan, M., Segata, N., Waldron, L.: Accessible, curated metagenomic data through ExperimentHub. Nat. Methods 14(11), 1023–1024 (oct 2017)
- Quinzan, F., Khanna, R., Hershcovitch, M., Cohen, S., Waddington, D., Friedrich, T., Mahoney, M.W.: Fast feature selection with fairness constraints. In: Ruiz, F., Dy, J., van de Meent, J.W. (eds.) Proceedings of The 26th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 206, pp. 7800–7823. PMLR (25–27 Apr 2023)
- 14. Śmieja, M., Warszycki, D., Tabor, J., Bojarski, A.J.: Asymmetric clustering index in a case study of 5-ht1a receptor ligands. PloS one 9(7), e102069 (2014)
- 15. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological) **58**(1), 267–288 (1996)
- 16. Yasuda, T., Bateni, M., Chen, L., Fahrbach, M., Fu, G., Mirrokni, V.: (Apr 2023), arXiv:2209.14881 [cs]
- 17. Yu, L., Liu, H.: Efficient Feature Selection via Analysis of Relevance and Redundancy
- 18. Zou, H., Hastie, T.: Regularization and Variable Selection Via the Elastic Net. Journal of the Royal Statistical Society Series B: Statistical Methodology **67**(2), 301–320 (Mar 2005)