Interlocking-free Selective Rationalization Through Genetic-based Learning

Anonymous ACL submission

Abstract

A popular end-to-end architecture for selective rationalization is the select-then-predict pipeline, comprising a generator to extract highlights fed to a predictor. Such a cooperative system suffers from suboptimal equilibrium minima due to the dominance of one of the two modules, a phenomenon known as *interlocking*. While several contributions aimed at addressing interlocking, they only mitigate its effect, often by introducing feature-based heuristics, sampling, and ad-hoc regularizations. We present GenSPP, the first interlocking-free architecture for selective rationalization that does not require any learning overhead, as the above-mentioned. GenSPP avoids interlocking by performing disjoint training of the generator and predictor via genetic global search. Experiments on a synthetic and a real-world benchmark show that our model outperforms several state-of-the-art competitors.

1 Introduction

Selective rationalization is the process of learning by providing highlights (or rationales) as explanation, a type of explainable AI approach that has gained momentum in high-stakes scenarios (Wiegreffe and Marasovic, 2021), such as fact-checking and legal analytics. Highlights are a subset of input texts meant to be interpretable by a user and faithfully describe the inference process of a classification model (Herrewijnen et al., 2024). Among the several contributions, the select-then-predict (SPP) selective rationalization framework of Lei et al. (2016) has gained popularity due to its inherent property of defining a faithful self-explainable model. In SPP, a classification model comprises a generator and a predictor. The generator generates highlights from input texts, i.e., it selects a portion of input text tokens, which are fed to the predictor to address a task. To define interpretable highlights, the generator performs discrete selections of input tokens while regularization objectives control the quality of generated highlights.

This discretization process introduces an optimization issue between the generator and the predictor, hindering training stability and increasing the chances of falling into local minima, a phenomenon denoted as *interlocking* (Yu et al., 2021). To account for this issue, several contributions have been proposed to facilitate information flow between the generator and predictor and avoid overfitting on sub-optimal highlights. Notable examples include differentiable discretization via sampling (Bao et al., 2018; Bastings et al., 2019), weight sharing between generator and predictor (Liu et al., 2022), and external guidance via soft rationalization (Yu et al., 2021; Huang et al., 2021; Sha et al., 2023; Hu and Yu, 2024). However, these methods only mitigate interlocking by introducing ad-hoc regularization.

A few attempts have been proposed to eliminate interlocking. These solutions either rely on feature-based heuristics to pre-train the generator (Jain et al., 2020) or partially address interlocking by introducing multiple independent training stages (Li et al., 2022). However, these methods present several limitations, including the use of heuristics for guiding the generator, limited information flow between the generator and the predictor, and introduce additional optimization issues.

We propose **Gen**etic-**SPP** (GenSPP), the first selective rationalization framework that eliminates interlocking without requiring heuristics and architectural changes. GenSPP breaks interlocking by splitting the optimization process into two stages, optimized via genetic-based search. First, a generator instance is defined independently of a given predictor. Second, a predictor is trained from scratch while keeping the defined generator frozen. Genetic-based search allows for local and global exploration of the generator's parameters, significantly reducing the risk of getting stuck into local minima. Furthermore, genetic-based search does not require differentiable learning objectives, allowing for a more accurate model evaluation accounting for both classification performance and highlight quality.

We evaluate GenSPP on two benchmarks: a controlled synthetic dataset that we introduce to assess selective rationalization frameworks and a popular real-world dataset on hate speech. Experimental results show that GenSPP achieves superior highlight quality while maintaining comparable classification performance.

To summarize, our contributions are:

- We introduce GenSPP, the first interlocking-free selective rationalization framework that does not require sub-optimal heuristics and additional regularizations.
- We design a robust evaluation objective to account for classification and rationalization capabilities equally.
- We build a novel controlled synthetic dataset to study selective rationalization frameworks.
- We carry out an extensive, robust, and reproducible experimental setting to compare GenSPP with several competitive selective rationalization frameworks.

We make our data and code available for research.[1]

2 Preliminaries

We overview two fundamental concepts to understand our method: (i) selective rationalization and (ii) genetic-based search.

2.1 Selective Rationalization

Selective rationalization denotes a self-explainable classification model capable of extracting discrete highlights from an input text. The typical architecture for selective rationalization is based on the select-then-predict (SPP) architecture (Lei et al., 2016). In SPP, the classification model is split into a generator (g_θ) and a predictor (f_ω), where θ and ω are the parameter sets. Given an input text $x = \{x^1, x^2, \dots, x^n\}$ comprising n tokens and its corresponding ground-truth label y, the generator g_θ produces a binary mask $m = g_\theta(x) = \{m^1, m^2, \dots, m^n\}$ where $m^i \in \{0, 1\}$. The mask m indicates which tokens of x are selected. We denote the mask generation process as *rationalization*. A masked input text $\tilde{x}$ is then defined by applying m on x as follows: $\tilde{x} = x \odot m$. The masked text $\tilde{x}$ is fed to the predictor f_ω for classification. Generally, the selective rationalization architecture is trained to minimize the classification empirical error on an annotated dataset, without providing supervision on generated m. This setting is often denoted as *unsupervised rationalization*, which is formalized as follows:

$$\mathcal{L}_t = \min_{\theta,\omega} \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \mathcal{L}_{ce}\big(f_\omega(g_\theta(x) \odot x), y\big), \quad (1)$$

where $\mathcal{D}$ is a textual dataset annotated for classification and $\mathcal{L}_{ce}$ is the classification loss.

Controlled Rationalization. A self-explainable model should produce meaningful highlights in addition to accurate predictions. Lei et al. (2016) introduced regularization objectives to prefer sparse and coherent highlights for better interpretability. Formally, the regularizer is denoted as follows:

$$\Omega(m) = \lambda_s \underbrace{\sum_{i=0}^{n} m^i}_{\mathcal{L}_s} + \lambda_c \underbrace{\sum_{i=1}^{n} |m^i - m^{i-1}|}_{\mathcal{L}_c}, \quad (2)$$

where $\mathcal{L}_s$ controls the level of sparsity (sparsity constraint), $\mathcal{L}_c$ reduces highlights fragmentation (contiguity constraint), and $\lambda_s, \lambda_c \in \mathbb{R}$ are scalar coefficients that balance the regularization. Effectively controlling the regularization effect of $\mathcal{L}_s$ to not outweigh $\mathcal{L}_t$ is non-trivial. To simplify the optimization process, Chang et al. (2020) relax the sparsity constraint to achieve a specific sparsity level: $\mathcal{L}_s = |\alpha - \frac{1}{n}\sum_{i=0}^{n} m^i|$, where $\alpha \in [0, 1]$ regulates the degree of sparsity. By including the regularizer, Eq. 1 can then be rewritten as follows:

$$\mathcal{L} = \mathcal{L}_t + \Omega(m) \quad (3)$$

Interlocking. Yu et al. (2021) showed that when performing unsupervised rationalization in an end-to-end fashion, the selective rationalization architecture suffers from sub-optimal equilibrium minima. This occurs when either the generator g_θ or the predictor f_ω are in a sub-optimal state. If g_θ is stuck on generating a sub-optimal m, f_ω is fine-tuned on that m, further enforcing g_θ to maintain that selection. Similarly, if f_ω is a remarkably bad

[1] `https://anonymous.4open.science/r/gen-spp-8D2F/`

predictor, it is further encouraged to exhibit lower classification error on a sub-optimal m compared to the ground-truth one m^*.

2.2 Genetic Algorithms

Genetic Algorithms (GAs) constitute a class of search algorithms for finding optima in optimization problems. They are based on population-based search relying on the concept of survival of the fittest (Katoch et al., 2021). Formally, a population $\mathcal{P}$ contains a set of I individuals, $\mathcal{P} = \{c_1, c_2, \ldots, c_I\}$, where each individual $c \in \mathbb{R}^d$ is a parameter vector representing a candidate solution to the problem of interest.

Initially, a population $\mathcal{P}_0$ of I individuals is initialized randomly to cover the solution search space. The individuals are evaluated by a fitness function $h : \mathbb{R}^d \rightarrow \mathbb{R}$ that is the optimization objective of GAs. A portion of individuals is then selected based on their fitness scores with selection probability p^{sl}. An intermediate population $\tilde{\mathcal{P}}_0$ is built by generating individuals from selected ones, either by modifying a portion of individual parameters (*mutation*) or by mixing parameters between individual pairs (*crossover*). We denote p^m and p^c the mutation and crossover probabilities, respectively. The population for the next iteration $\mathcal{P}_i$ is built by performing a second individual selection phase, denoted as survival selection, to keep the number of individuals equal to I across generations. We denote p^{su} the survival probability of each individual. The population-based search is iterated for G generations or stopped preemptively if a certain fitness score is reached.

Neuroevolution. GAs have been successfully applied to solve a wide variety of tasks (Alhijawi and Awajan, 2024), including image processing, scheduling, clustering, natural language processing, and, in particular, neural network optimization, known as *neuroevolution* (Galván and Mooney, 2021). Neuroevolution denotes the process of (i) neural network architecture search and (ii) parameter optimization by employing genetic algorithms. In the second scenario, each individual c in a population $\mathcal{P}$ denotes the parameters of a neural network. In addition to having interesting properties, such as parallel computation and reduced likelihood of getting stuck into local minima, neuroevolution also shows correspondence with gradient descent, as proved by Whitelam et al. (2021).

3 Related Work

Lei et al. (2016) introduce Rationalizing Neural Predictions (RNP), the first SPP framework, whereby the generator and predictor components are trained via reinforcement learning (Williams, 1992). Several contributions have explored ways to improve RNP, including end-to-end optimizations, external guidance to mitigate spurious correlations, regularizations for faithful rationalization, and attempts to break interlocking.

Improved Optimization. Bao et al. (2018) propose an end-to-end architecture by leveraging the Gumbel softmax trick (Jang et al., 2017) for generating differentiable discrete masks m. Similarly, Bastings et al. (2019) adopt rectified Kumaraswamy distributions to replace sampling from Bernoulli distributions. Parameterized sampling provides a regularization effect to mitigate interlocking, but it requires additional calibration effort to find the best trade-off between sampling stability and exploration. In contrast, genetic-based search does not require sampling to define discrete selection masks and has superior optimization stability with respect to standard reinforcement learning algorithms (Salimans et al., 2017). Contributions have also explored solutions to ease the learning process. Liu et al. (2022) propose to share embedding weights between the generator and predictor to increase information flow between the two modules. Liu et al. (2023d) employ different learning rates for g_θ and f_ω to mitigate selection mask overfitting. Liu et al. (2023b) use multiple generators to improve rationalization exploration to reduce the chance of interlocking. While, in principle, some of these design choices, like weight sharing, may be included in our framework, they are not required as GenSPP avoids interlocking.

External Guidance. Another class of contributions leverages information from the input text to guide selective rationalization. Yu et al. (2021) define an attention-based predictor that performs soft selections to mitigate interlocking. Chang et al. (2019) propose a generator-discriminator adversarial training to learn class-wise highlights. Paranjape et al. (2020) propose a sparsity regularization objective based on information bottleneck to trade-off performance accuracy and highlight coherence. Huang et al. (2021) define a guider module that acts as a teacher for f_ω and propose an embedding-based regularization between the embedded input x

and the generated highlight $\tilde{x}$ to guide g_θ. Yue et al. (2022) propose a mutual information regularization to exploit information from non-selected tokens by leveraging an additional predictor. Sha et al. (2023) introduce the InfoCal framework, where an additional predictor trained on the input text x provides guidance through a regularization objective based on the information bottleneck principle. Liu et al. (2023a) use an additional predictor trained on the original texts and fixed during rationalization to guide f_ω. Hu and Yu (2024) employ an end-to-end guidance module with information from the original input text to guide f_ω while also providing importance scores for weighting tokens to guide g_θ. In contrast to all these approaches, GenSPP does not require the integration of additional neural modules and regularizations to guide g_θ since genetic-based search alleviates selective rationalization from getting stuck into sub-optimal minima.

Breaking Interlocking. Few attempts have explored breaking interlocking. Jain et al. (2020) employ importance score features derived from post-hoc explainable tools like LIME (Ribeiro et al., 2016) to first pre-train g_θ. Subsequently, f_ω is trained on the dataset produced in the previous stage. Compared to our work, the solution of Jain et al. (2020) has two limitations. First, it requires external feature extraction tools that act as heuristics for training g_θ in a supervised fashion. Second, information learned when training f_ω does not flow to g_θ for improvement. In contrast, the generator g_θ in GenSPP is trained via a heuristic fitness function that only involves learning objectives concerning classification performance and highlight quality (Eq. 3). A recent contribution is the 3-stage framework of Li et al. (2022) for multi-aspect rationalization (Antognini et al., 2021; Antognini and Faltings, 2021). In the first stage, g_θ and f_ω are first trained end-to-end, and then g_θ is discarded. In the second stage, f_ω is frozen, and a new generator is trained. Likewise, in the third stage, the trained new generator is frozen while f_ω is fine-tuned. While this framework avoids interlocking by iteratively freezing g_θ or f_ω, it presents two main limitations. First, it is not completely interlocking-free since interlocking may still occur in the first stage, leading to a sub-optimal f_ω. Second, it does not offer good guarantees for reaching an optimal solution due to two independent training stages. In contrast, GenSPP is interlocking-free, characterized by stable convergence properties due to global search.

4 Motivation

We motivate our work by discussing how existing contributions only mitigate interlocking. The analysis of (Yu et al., 2021) underlines that the quality of the selective rationalization solution strongly depends on the system's capability to avoid the interlocking effect, thus reducing the probability of incurring local minima during training. Interlocking affects the following optimization problem:

$$\min_\theta \min_\omega \mathcal{L}(f_\omega(g_\theta(x) \odot x), y) \tag{4}$$

A major cause of interlocking is the generation of a discrete binary mask m to define a faithful and interpretable model. The discretization of m induces a discrepancy in how g_θ and f_ω learn during training. As pointed out by Yu et al. (2021), f_ω tends to overfit to a certain sub-optimal mask m, causing the interlocking. More precisely, while the predictor's parameters ω change smoothly at each gradient step thanks to the continuous nature of the learning objective, the generator g_θ contains a discrete function (i.e., rounding) that makes its policy a piecewise constant function with respect to its parameters θ. Even by applying smoothing techniques (e.g., sampling) to mitigate the issue and achieve differentiability, the generated binary mask m might remain unchanged (or change too slowly) over multiple gradient steps, thus, leading f_ω to overfit on m.

To address this issue, contributions have proposed sampling-based methods to allow for differentiable discretization (Bao et al., 2018; Bastings et al., 2019), external guidance by introducing an additional soft rationalization system (Chang et al., 2019; Yu et al., 2021; Sha et al., 2023; Liu et al., 2023a; Hu and Yu, 2024), multi-stage training procedures (Liu et al., 2023b), and weight sharing between g_θ and f_ω for increased information flow (Liu et al., 2022). However, none of these methods solves interlocking, and the likelihood of rapidly falling into a local optimum is only mitigated at the cost of added optimization issues, such as increased variance.

Given the side effect caused by the unequal joint training of the two models via stochastic gradient descent (SGD), a logical and straightforward way to break the interlocking between g_θ and f_ω is to split the dual minimization problem of Eq. 4. Formally, let ω^* be the optimal predictor's parameters, and let l be its optimal solution:

$$l = \mathcal{L}(f_{\omega^*}(x), y) \tag{5}$$

Eq. 4 can be reformulated as a disjoint training by minimizing:

$$\min_{\theta} \Omega(m)$$
$$s.t. \min_{\omega} \mathcal{L}\big(f_\omega(g_\theta(x) \odot x), y\big) \leq l + \epsilon \quad (6)$$

for a tolerance ϵ. This formulation is equivalent to finding the optimal highlight (according to the applied regularization), such that f_ω achieves a comparable performance to a predictor trained on x, up to a certain level of approximation regulated by ϵ. Equivalently, g_θ is trained to filter out uninformative information from input text x. Given the structure of Eq. 6, the disjoint optimization cannot be addressed via SGD and, therefore, we propose genetic algorithms to address the minimization problem.

5 The GenSPP Framework

We introduce GenSPP, a novel SPP framework optimized via genetic-based search. GenSPP presents several advantages over selective rationalization based on SGD. First, GenSPP is interlocking-free by splitting the optimization process into two stages (Eq. 6): each individual c embodies a different generator g_θ, which is then evaluated through a unique predictor f_ω. Second, GenSPP leverages genetic-based search, allowing for both local (via mutation) and global (via crossover) search in the θ parameter space to avoid local minima. Third, genetic-based search does not require a differentiable learning objective, allowing for more accurate training regularizations. We describe GenSPP and discuss its advantages over other selective rationalization frameworks in detail.

5.1 Method

GenSPP follows the same architecture of Lei et al. (2016) where hard rationalization is performed via rounding and is trained via neuroevolution. In particular, individual evaluation is a two-stage process. First, a population $\mathcal{P}$ of individuals, each representing a configuration of the generator's parameters, is defined. Second, each individual is evaluated via a fitness function h. In particular, a predictor is initialized from scratch for each individual c and trained to minimize the task classification loss via SGD while keeping the parameters of c frozen to avoid interlocking. We compute h on each trained model, and we build a new population by selecting individuals based on their fitness scores. The process is iterated until convergence or a fixed budget of generations G is reached. Algorithm 1 summarizes GenSPP algorithm.

Algorithm 1 GenSPP Algorithm

```
Input: Population $\mathcal{P}$, fitness function $h$, selection probability $p^{sl}$, crossover probability $p^c$, mutation probability $p^m$, survival probability $p^{su}$, $G$ generations, task threshold $l$.
Output: Optimal individual $c^*$.
 1: Initialize $\mathcal{P}_0 = \{c_1, \dots, c_I\}$ of $I$ individuals
 2: Initialize memory weights $p_{|M|} = p^0_{|M|}$
 3: for individual $c \in \mathcal{P}_0$ do
 4:     Train a predictor $f_\omega$ to minimize $\mathcal{L}_t$
 5:     Evaluate $c$ via fitness function $h$
 6: end for
 7: while current generation $g < G$ do
 8:     Determine crossover pairs with selection probability $p^c_i = \frac{h(c_i)}{\sum_j^I h(c_j)}$
 9:     Generate $\frac{I}{2}$ new individuals via one-point crossover
10:     Perform mutation on newly generated individuals with mutation probability $p^m$
11:     for individual $c$ in generated individuals do
12:         Train a predictor $f_\omega$ to minimize $\mathcal{L}_t$
13:         Evaluate $c$ via fitness function $h$
14:     end for
15:     Perform survival selection to obtain $\mathcal{P}_{g+1}$
16: end while
```

5.2 Individual Evaluation

We identify two major issues in Eq. 3 for model evaluation. First, finding a balance between $\mathcal{L}_t$ and $\Omega(m)$ is non-trivial, potentially leading to sub-optimal solutions that only minimize one of the two. Second, the joint learning formulation is not a reasonable candidate for optimization, collapsing substantially different solutions to the same cost value. Consider two instances of the learning problem, one with $\mathcal{L}_t = 0.0$ and $\Omega(m) = 1.0$, and another with $\mathcal{L}_t = 0.5$ and $\Omega(m) = 0.5$. Notably, both instances have the same average cost of 0.5, but the first does not satisfy our objective of defining a faithful rationalization framework (see Appendix A for a graphical comparison). Therefore, the two instances should be evaluated differently to favor solutions that are both accurate and interpretable.

To allow for more robust individual evaluation, we propose the following objective function:

$$\tilde{h} = \begin{cases} 1 - \mathcal{L}, & \text{if } \mathcal{L}_t < l + \epsilon \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

where $\mathcal{L} = \sqrt{(1-\Omega(m)) \times (1-min(\mathcal{L}_t, 1))}$. To account for the maximization problem in genetic search, we define the fitness function h in GenSPP as follows:

$$h = \frac{1}{\tilde{h} + \hat{\epsilon}}, \tag{8}$$

where $\hat{\epsilon}$ is a small constant to ensure computational stability. Eq. 7 guides the learning process by initially favoring $\mathcal{L}_t$ and progressively shifting toward a state where $\mathcal{L}_t$ is stable while $\Omega(m)$ is optimized. We do not require weight balancing since learning objectives are normalized and equally important.

5.3 GenSPP Genetic Algorithm

We describe the genetic algorithm for training GenSPP. Given a population $\mathcal{P}_0$ of I individuals, each representing a different generator instance, we perform individual selection and recombination as follows. We initially evaluate $\mathcal{P}_0$ by computing the fitness score of each individual in the population. We apply the roulette-wheel selection strategy, a stochastic process where individuals are sampled proportionally to their fitness score (Lipowski and Lipowska, 2012), to pair individuals for recombination. In total, $\frac{I}{2}$ pairs are selected. We employ one-point crossover (Poli and Langdon, 1998) to generate I new individuals from selected pairs. This crossover strategy swaps parameters between two individuals by randomly choosing a swap point from a uniform distribution. We then mutate each generated individual parameter with probability p^m by inserting Gaussian noise. The intermediate population $\tilde{\mathcal{P}_0}$ comprises the original I individuals and the I newly generated ones. To build the population $\mathcal{P}_1$ of I individuals for the next generation, we evaluate the fitness score of $\tilde{\mathcal{P}_0}$ and then perform survival selection via the half-elitism strategy (Michalewicz, 1996). In particular, we select the $\frac{I}{2}$ with the highest fitness score, while the remaining $\frac{I}{2}$ is sampled via roulette-wheel selection.

5.4 Advantages

Optimizing Eq. 6 via GAs introduces several advantages over selective rationalization based on SGD, which we discuss in detail.

Disjoint Training. A joint training of the selective rationalization system based on SGD involves a dependency between g_θ and f_ω: the quality of a highlight mask m is also dependent on the quality of the current employed f_ω (e.g., good masks may be evaluated badly if f_ω has already overfitted to a previously generated mask). In contrast, the proposed disjoint training allows the optimization of g_θ by searching in the space of parameters that minimize Ω, while yielding the highest performance in classification. More precisely, the f_ω depends on g_θ, while the opposite does not hold.

Global Search. Population-based search in GAs reduces the chances of converging towards local minima, a common issue in optimization independently from interlocking. Mutation and crossover offer two ways to perform local and global search space, respectively, alleviating the risk of getting stuck into a local optimum.

Non-differentiable Objective. Differentiable sampling (e.g., via Gumbel softmax (Jang et al., 2017)) introduces noise, potentially making the optimization process of g_θ unstable depending on the chosen sampling hyper-parameters. In contrast, genetic-based search does not require gradient computation for optimization, ensuring a more robust training procedure. Additionally, the optimization objective of GenSPP (Eq. 8) can be designed without defining surrogate losses (Eq. 7). This is a crucial advantage of GenSPP since it is not subject to dataset-specific hyperparameter-tuning (e.g., α in $\mathcal{L}_s$). In contrast, SGD-based approaches require heavy fine-tuning to find a reasonable α value.

6 Experimental Settings

We compare GenSPP to several competitors for unsupervised selective rationalization[2] on two benchmarks. We describe the data, models, and evaluation metrics in detail. See Appendix B for additional details.

Toy Dataset. We build and release a controlled toy dataset of random strings. We define three classification classes, each corresponding to a unique character-based highlight: *aba*, *baa*, *abc*. We design highlights to ensure that all their characters have to be selected in order to determine the corresponding class. To avoid degenerate solutions in which only a portion of the highlight is sufficient for classification, we contaminate generated strings with randomly sampled chunks of other class highlights. Lastly, we enforce that a single highlight is contained in each string. Generated strings not

[2]We recall that ground-truth highlights are only used for model evaluation and not provided as input.

compliant with the aforementioned rules are discarded. We set the generated string length to 20 characters. In total, we generate 10k random strings and split them into train (6.4k), validation (1.6k), and test (2k) partitions.

HateXplain Dataset. A dataset of $\sim$20k English posts from social media platforms like X and Gab (Mathew et al., 2021). Each post is annotated from three different perspectives: hate speech (*hate*, *offensive*, *normal*), the target community victim of hate speech, and the rationales which the labeling decision about hate speech is based on. To account for annotation subjectivity, each post is annotated by at least three annotators (Waseem, 2016; Sap et al., 2022). We notice that annotations vary significantly among annotators regarding the number of selected tokens. This might hinder rationalization evaluation since longer highlights might be preferred. For this reason, we employ a majority voting strategy to merge annotators' highlights and identify top relevance tokens. As a side effect, extracted ground-truth highlights are less cohesive. We filter out texts longer than 30 tokens to reduce the computational overhead. The dataset is split into train ($\sim$10k), validation ($\sim$1.3k), and test ($\sim$1.3k) partitions. We consider hate speech as a binary classification problem by merging *hate* and *offensive* classes.

Models. We consider the architecture of Yu et al. (2021) for all models, including ours, described as follows. An input text x is encoded via a frozen pre-trained embedding layer. We use one-hot encoding for Toy and 25-dimension GloVe embeddings (Pennington et al., 2014) pre-trained on Twitter for HateXplain. The generator g_θ comprises a RNN layer with a dense layer on top for token selection. The predictor f_ω comprises a RNN layer followed by a max-pooling layer and a final linear layer for classification. We set the RNN layer to a biGRU for baselines and GRU for GenSPP, respectively. We consider the following baselines. FR (Liu et al., 2022), an end-to-end SPP framework using Gumbel softmax for discrete mask generation, where g_θ and f_ω share the same RNN layers. MGR (Liu et al., 2023b), an SPP framework where multiple generators are considered to extract distinct highlights that are fed to a single predictor. At inference time, only the first generator is considered since all generators eventually align on the same mask m. MCD (Liu et al., 2023c), a guidance-based SPP framework, where an additional predictor trained using the original input text x is used to guide selective rationalization towards better highlights. G-RAT (Hu and Yu, 2024), a recent guidance-based SPP framework, where an attention-based soft SPP framework is used as guidance.

Evaluation Metrics. We focus on classification performance and rationalization quality (Chang et al., 2019; Yu et al., 2021). Regarding classification performance, we report macro F1-score averaged over all classes (**Clf-F1**). Regarding generated highlights, we report binary token-level F1-score (**Hl-F1**), selection ratio (R), and selection size (S).

7 Results

We consider two sets of experiments. The first evaluates models when trained from scratch to assess their capability to avoid local minima. The second measures how good a method is at recovering from interlocking. See Appendix C for additional results.

Benchmark Evaluation. Table 1 reports results. We observe that GenSPP significantly outperforms all competitors in selecting high-quality highlights (+10.3% Hl-F1 in Toy and +6.5% Hl-F1 in HateXplain), while reporting comparable classification performance. Additionally, GenSPP shows reduced variance across seed runs compared to competitors, especially in the Toy dataset, where MGR and G-RAT present notable instability. Regarding highlight regularization, GenSPP selects highlights that are more sparse and accurate compared to baseline models. Interestingly, GenSPP learns to not select any highlight for negative examples in HateXplain, while keeping valuable selections for positive examples, a flexibility that baseline models cannot achieve since they are subject to satisfy a certain sparsity threshold. Overall, these results show the advantage of GenSPP in performing a disjoint optimization problem via genetic-based search to break interlocking.

Synthetic Skewing. We follow Liu et al. (2022) and train a skewed g_θ for $K = 10$ epochs using the classification label as supervision for selecting the first token x^1. To evaluate GenSPP on this experiment, we include one skewed individual in the initial population $\mathcal{P}_0$, while randomly initializing the remaining individuals. We experiment with $G \in [100, 150]$ since convergence may require more time due to recombinations with the

	Toy				HateXplain			
Model	Clf-F1 $\uparrow$	Hl-F1 $\uparrow$	$R \downarrow$	$S \downarrow$	Clf-F1 $\uparrow$	Hl-F1 $\uparrow$	$R \downarrow$	$S \downarrow$
FR	$99.78_{\pm 0.20}$	$54.07_{\pm 4.02}$	$14.80_{\pm 0.24}$	$2.96_{\pm 0.05}$	$72.14_{\pm 1.12}$	$31.15_{\pm 2.56}$	$25.55_{\pm 0.72}$	$3.46_{\pm 0.11}$
MGR	$99.92_{\pm 0.04}$	$50.34_{\pm 11.23}$	$15.05_{\pm 0.80}$	$3.01_{\pm 0.16}$	$71.14_{\pm 1.16}$	$29.38_{\pm 4.83}$	$25.30_{\pm 1.04}$	$3.42_{\pm 0.06}$
MCD	$99.90_{\pm 0.04}$	$65.70_{\pm 3.76}$	$15.18_{\pm 0.17}$	$3.04_{\pm 0.03}$	$70.37_{\pm 1.06}$	$27.92_{\pm 1.66}$	$25.07_{\pm 1.52}$	$3.50_{\pm 0.20}$
G-RAT	$99.36_{\pm 0.82}$	$50.22_{\pm 7.78}$	$14.81_{\pm 0.51}$	$2.96_{\pm 0.10}$	$73.85_{\pm 1.05}$	$36.17_{\pm 1.62}$	$24.68_{\pm 0.86}$	$3.34_{\pm 0.08}$
GenSPP (Ours)	$99.00_{\pm 0.25}$	$^{**}\mathbf{76.02_{\pm 0.64}}$	$11.47_{\pm 0.49}$	$\mathbf{2.29_{\pm 0.08}}$	$69.71_{\pm 0.40}$	$^{**}\mathbf{42.62_{\pm 0.73}}$	$6.51_{\pm 0.58}$	$\mathbf{0.75_{\pm 0.05}}$

Table 1: Benchmark evaluation test results. We report average macro F1-score (**Clf-F1**) for classification, while we report binary token-level F1-score (**Hl-F1**), selection rate (**R**) and size (**S**) for rationalization. Best results are highlighted in bold.
(**) ≤ 0.01 denotes Wilcoxon statistical significance on the best baseline.

	Toy				HateXplain			
Model	Clf-F1 $\uparrow$	Hl-F1 $\uparrow$	$R \downarrow$	$S \downarrow$	Clf-F1 $\uparrow$	Hl-F1 $\uparrow$	$R \downarrow$	$S \downarrow$
FR	$99.85_{\pm 0.11}$	$58.91_{\pm 3.18}$	$14.57_{\pm 0.12}$	$2.91_{\pm 0.02}$	$71.00_{\pm 0.76}$	$7.22_{\pm 2.29}$	$26.85_{\pm 0.92}$	$3.47_{\pm 0.08}$
MGR	$97.75_{\pm 4.25}$	$37.45_{\pm 12.33}$	$14.99_{\pm 0.42}$	$3.00_{\pm 0.08}$	$71.09_{\pm 1.00}$	$14.45_{\pm 4.84}$	$27.75_{\pm 1.36}$	$3.57_{\pm 0.11}$
MCD	$99.93_{\pm 0.06}$	$62.94_{\pm 2.39}$	$15.10_{\pm 0.66}$	$3.02_{\pm 0.13}$	$70.93_{\pm 0.95}$	$13.88_{\pm 10.13}$	$25.84_{\pm 1.87}$	$3.46_{\pm 0.16}$
G-RAT	$99.85_{\pm 0.14}$	$47.53_{\pm 12.77}$	$14.56_{\pm 0.36}$	$2.91_{\pm 0.07}$	$73.15_{\pm 0.35}$	$34.33_{\pm 1.22}$	$25.43_{\pm 0.87}$	$3.40_{\pm 0.09}$
GenSPP ($G = 100$)	$98.93_{\pm 0.47}$	$70.52_{\pm 0.15}$	$13.04_{\pm 0.12}$	$2.60_{\pm 0.02}$	$67.02_{\pm 0.57}$	$39.89_{\pm 0.73}$	$7.45_{\pm 0.48}$	$0.96_{\pm 0.05}$
GenSPP ($G = 150$)	$99.46_{\pm 0.36}$	$^{**}\mathbf{74.28_{\pm 0.61}}$	$10.11_{\pm 0.42}$	$1.99_{\pm 0.04}$	$69.89_{\pm 0.43}$	$^{**}\mathbf{42.81_{\pm 0.65}}$	$6.74_{\pm 0.67}$	$0.87_{\pm 0.07}$
GenSPP$_{sk}$ ($G = 100$)	$98.74_{\pm 0.43}$	$63.45_{\pm 0.36}$	$8.03_{\pm 0.40}$	$1.58_{\pm 0.06}$	$66.41_{\pm 0.35}$	$35.52_{\pm 0.46}$	$8.17_{\pm 0.62}$	$1.06_{\pm 0.07}$

Table 2: Synthetic skew test set results. We report average macro F1-score (**Clf-F1**) for classification, while we report binary token-level F1-score (**Hl-F1**) and selection rate (**R**) and size (**S**) for rationalization. Best results are highlighted in bold.
(**) ≤ 0.01 denotes Wilcoxon statistical significance on the best baseline.

skewed individual in the earlier generations. Additionally, to stress test GenSPP, we consider a more degenerated setting where we initialize $\mathcal{P}_0$ with variants of the skew individual by adding Gaussian noise. We denote this configuration as GenSPP$_{sk}$. Table 2 reports results conducted on both datasets. We observe that G-RAT and MCD are the best-performing baselines on HateXplain and Toy datasets, respectively. In general, baseline models suffer from high variance, showing that these methods are not able to break the interlocking state in many seed runs. In contrast, GenSPP recovers from the degenerated state and outperforms baseline models, achieving comparable performance to the one reported in Table 1. In particular, performing a parameter search with an increased budget (e.g., $G = 150$) leads to the best results.

Discussion. Breaking interlocking in GenSPP comes with some limitations. Intuitively, genetic-based search requires more computational time than solutions based on SGD since I predictors are trained at each generation. On average, a seed run of GenSPP takes $\sim$36min in Toy and $\sim$78min in HateXplain. In contrast, a seed run for baseline models requires $\sim$8min and $\sim$4min, respectively. Nonetheless, we remark on two aspects regarding our implementation: (i) individuals are evaluated sequentially, and (ii) we make use of standard genetic operations for individual evaluation and selection. More efficient implementations (e.g., allowing parallel computation of individuals) and advanced algorithms, such as the CMA-ES (Hansen and Ostermeier, 2001), can significantly reduce convergence time. We leave these improvements as future work. This drawback is mitigated by two main properties of GenSPP. First, GenSPP has low variance, avoiding, in principle, multiple seed runs for evaluation. Second, global search via crossover allows for employing lighter and yet more efficient models. Compared to competitors, GenSPP has the same size as the smallest model (i.e., FR), which is 2-4x smaller than other baselines.

8 Conclusions

We have introduced GenSPP, the first selective rationalization framework that breaks interlocking via genetic-based search. GenSPP does not require differentiable surrogate learning objectives, additional regularization tuning, and architectural changes. Our results on two benchmarks, a controlled synthetic one that we curate, and a real-world dataset for hate speech, show the advantage of GenSPP, outperforming several competitors. Furthermore, our robust evaluation underlines the increased variance that affects competitors' models, a phenomenon that was not sufficiently explored in selective rationalization. Future research directions regard exploring more efficient genetic algorithms and implementations to reduce computational overhead and scale to more complex neural architectures.

Limitations

Data This study is based on two datasets, one of which is synthetic. A broader analysis of GenSPP on several datasets could strengthen our contribution. Nonetheless, despite our efforts, we could not find other high-quality datasets for text classification with a relatively sufficient number of samples and multiple annotations for robust evaluation.

Models All models follow a specific architecture, which is not the only possible one. For instance, Hu and Yu (2024) was first proposed with transformer-based models. Our study could include other backbone architectures for a more exhaustive evaluation of selective rationalization frameworks.

References

Bushra Alhijawi and Arafat Awajan. 2024. Genetic algorithms: theory, genetic operators, solutions, and applications. *Evol. Intell.*, 17(3):1245–1256.

Diego Antognini and Boi Faltings. 2021. Rationalization through concepts. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 761–775. Association for Computational Linguistics.

Diego Antognini, Claudiu Musat, and Boi Faltings. 2021. Multi-dimensional explanation of target variables from documents. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 12507–12515. AAAI Press.

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Yujia Bao, Shiyu Chang, Mo Yu, and Regina Barzilay. 2018. Deriving machine attention from human rationales. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1903–1913, Brussels, Belgium. Association for Computational Linguistics.

Jasmijn Bastings, Wilker Aziz, and Ivan Titov. 2019. Interpretable neural predictions with differentiable binary variables. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2963–2977, Florence, Italy. Association for Computational Linguistics.

Shiyu Chang, Yang Zhang, Mo Yu, and Tommi S. Jaakkola. 2019. A game theoretic approach to class-wise selective rationalization. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 10055–10065.

Shiyu Chang, Yang Zhang, Mo Yu, and Tommi S. Jaakkola. 2020. Invariant rationalization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1448–1458. PMLR.

Edgar Galván and Peter Mooney. 2021. Neuroevolution in deep neural networks: Current trends and future challenges. *IEEE Trans. Artif. Intell.*, 2(6):476–493.

Nikolaus Hansen and Andreas Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*.

Elize Herrewijnen, Dong Nguyen, Floris Bex, and Kees van Deemter. 2024. Human-annotated rationales and explainable text classification: a survey. *Frontiers Artif. Intell.*, 7.

Shuaibo Hu and Kui Yu. 2024. Learning robust rationales for model explainability: A guidance-based approach. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 18243–18251. AAAI Press.

Yongfeng Huang, Yujun Chen, Yulun Du, and Zhilin Yang. 2021. Distribution matching for rationalization. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 13090–13097. AAAI Press.

Sarthak Jain, Sarah Wiegreffe, Yuval Pinter, and Byron C. Wallace. 2020. Learning to faithfully rationalize by construction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4459–4473, Online. Association for Computational Linguistics.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *ICLR*.

Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. 2021. A review on genetic algorithm: past, present, and future. *Multim. Tools Appl.*, 80(5):8091–8126.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Austin, Texas. Association for Computational Linguistics.

Shuangqi Li, Diego Antognini, and Boi Faltings. 2022. Interlock-free multi-aspect rationalization for text classification. *CoRR*, abs/2205.06756.

Adam Lipowski and Dorota Lipowska. 2012. Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications*, 391(6):2193–2196.

Wei Liu, Haozhao Wang, Jun Wang, Zhiying Deng, Yuankai Zhang, Cheng Wang, and Ruixuan Li. 2023a. Enhancing the rationale-input alignment for self-explaining rationalization. *CoRR*, abs/2312.04103.

Wei Liu, Haozhao Wang, Jun Wang, Ruixuan Li, Xinyang Li, Yuankai Zhang, and Yang Qiu. 2023b. MGR: multi-generator based rationalization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 12771–12787. Association for Computational Linguistics.

Wei Liu, Haozhao Wang, Jun Wang, Ruixuan Li, Chao Yue, and Yuankai Zhang. 2022. FR: folded rationalization with a unified encoder. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Wei Liu, Jun Wang, Haozhao Wang, Ruixuan Li, Zhiying Deng, Yuankai Zhang, and Yang Qiu. 2023c. D-separation for causal self-explanation. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Wei Liu, Jun Wang, Haozhao Wang, Ruixuan Li, Yang Qiu, Yuankai Zhang, Jie Han, and Yixiong Zou. 2023d. Decoupled rationalization with asymmetric learning rates: A flexible lipschitz restraint. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, pages 1535–1547. ACM.

Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2021. Hatexplain: A benchmark dataset for explainable hate speech detection. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 14867–14875. AAAI Press.

Zbigniew Michalewicz. 1996. *Genetic Algorithms + Data Structures = Evolution Programs, Third Revised and Extended Edition*. Springer.

Bhargavi Paranjape, Mandar Joshi, John Thickstun, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. An information bottleneck approach for controlling conciseness in rationale extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1938–1952, Online. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL.

Riccardo Poli and W. B. Langdon. 1998. Genetic programming with one-point crossover. In *Soft Computing in Engineering Design and Manufacturing*, pages 180–189, London. Springer London.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144. ACM.

Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. 2017. Evolution strategies as a scalable alternative to reinforcement learning. *CoRR*, abs/1703.03864.

Maarten Sap, Swabha Swayamdipta, Laura Vianna, Xuhui Zhou, Yejin Choi, and Noah A. Smith. 2022. Annotators with attitudes: How annotator beliefs and identities bias toxic language detection. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5884–5906, Seattle, United States. Association for Computational Linguistics.

Lei Sha, Oana-Maria Camburu, and Thomas Lukasiewicz. 2023. Rationalizing predictions by adversarial information calibration. *Artif. Intell.*, 315:103828.

Zeerak Waseem. 2016. Are you a racist or am I seeing things? annotator influence on hate speech detection on Twitter. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 138–142, Austin, Texas. Association for Computational Linguistics.

Stephen Whitelam, Viktor Selin, Sang-Won Park, and Isaac Tamblyn. 2021. Correspondence between neuroevolution and gradient descent. *Nature Communications*.

Sarah Wiegreffe and Ana Marasovic. 2021. Teach me to explain: A review of datasets for explainable natural language processing. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256.

Mo Yu, Yang Zhang, Shiyu Chang, and Tommi S. Jaakkola. 2021. Understanding interlocking dynamics of cooperative rationalization. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 12822–12835.

Linan Yue, Qi Liu, Yichao Du, Yanqing An, Li Wang, and Enhong Chen. 2022. DARE: disentanglement-augmented rationale extraction. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

A Loss Landscape Comparison

Fig. 1 compares the landscape of Eq. 7 to Eq. 3.

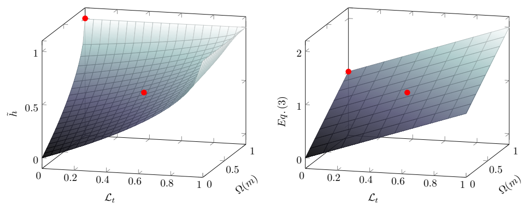

Figure 1: Loss landscape comparison between our fitness function $\tilde{h}$ (left) and the regularized selective rationalization objective (Eq. 3). Red markers denote points (0.0, 1.0) and (0.5, 0.5), highlighting the difference between the two losses.

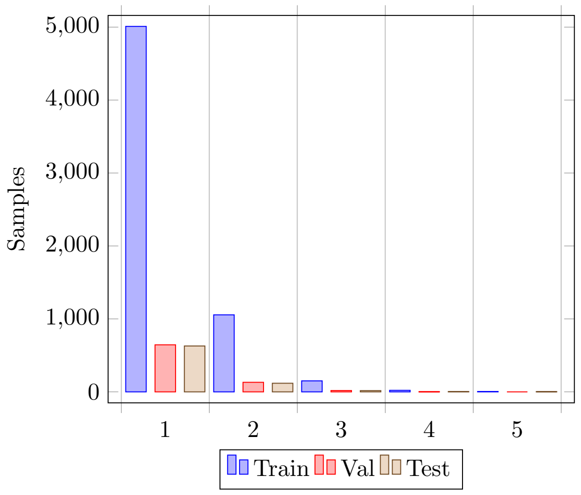

Figure 2: Number of contiguous highlights (i.e., connected token groups) in HateXplain.

B Experimental Settings

B.1 Data

We report additional details regarding the presented datasets.

Toy Dataset. To assess the quality of our toy dataset, we evaluate string-matching baselines for selective rationalization. Intuitively, the baseline that selects the right highlight for each class should achieve perfect rationalization performance. In contrast, other selections should lead to much lower selection performance. We consider the following string-matching baselines: {aba, baa, abc}, {abc, baa, aba}, and {ba, aa, bc}. The baselines achieve 100%, 33.33% and 53.57% Hl-F1 score, respectively.

HateXplain Dataset. Aggregating annotators' provided highlights via majority voting produces fragmented highlights. Therefore, the contiguity constraint $\mathcal{L}_c$ may lead to sub-optimal solutions. We compute the number of contiguous highlights in each example x to systematically analyze the impact of our design choice (see Fig. 2). Additionally, we compute the average highlight size and sparsity percentage. On average, $S = 1.57_{\pm 2.52}$ which corresponds to $R = 0.12_{\pm 0.17}$.

B.2 Training Setup

We carry out a repeated train-and-test evaluation routine using the provided dataset partitions. We evaluate models in five distinct seed runs. We consider layer norm (Ba et al., 2016), and early stopping on validation loss with patience set to 30 epochs as regularization methods. We train models using batch size 64 and Adam optimizer (Kingma and Ba, 2015) with learning rate set to 10^{-3}. All baseline models are trained with SGD following Eq. 3 as training objective, where $\mathcal{L}_{ce}$ is the categorical cross-entropy. We set $\lambda_s = 1.0$ and $\lambda_c = 2.0$ in the Toy dataset, while we set $\lambda_c = 0$ for HateXplain since highlights are inherently more fragmented (Fig. 2). We set the sparsity threshold $\alpha = 0.15$ in the Toy dataset. This value of α encourages $\sum_{i=0}^{n} m^i = 3$, which is the length of all character-based highlights in the Toy dataset. Conversely, we set $\alpha = 0.22$ in HateXplain based on training data statistics of ground-truth highlights.

Regarding GenSPP, we set $G = 100$ and $I = 50$, with mutation and crossover probabilities $p^m = p^c = 1.0$ and selection and survival rates $p^{sl} = p^{su} = 0.5$. We perform mutation by adding a Gaussian noise sample from $\mathcal{N}(0.0, 0.05)$. We train predictors during the genetic-based search for 3 epochs with batch size 64 and learning rate of 10^{-2}. We set evaluation tolerance $l + \epsilon$ equal to 0.1 and 0.6 for Toy and HateXplain case studies, respectively.

B.3 Model Details

Table 3 reports the full list of model hyperparameters employed in our experiments, while Table 4 and Table 5 report model configurations in Toy and HateXplain datasets, respectively.

B.4 Hardware and Implementation Details

For our experiments, we implemented all baselines and methods in PyTorch (Paszke et al., 2019), relying on open-source frameworks like PyTorch Lightning.[3] We will release all of the code and data to reproduce our experiments in an MIT-licensed public repository. All experiments were run on a private machine with an NVIDIA 3060Ti GPU with 8 GB dedicated VRAM.

C Results

We report additional experimental results for each presented experiment.

Benchmark Evaluation Table 6 and Table 7 report extensive results conducted on Toy and HateXplain datasets, respectively. In addition to baseline models, we consider a random baseline to assess the complexity of the rationalization task.

Synthetic Skewing Table 8 reports synthetic skew results when considering $K \in [5, 10, 15, 20]$.

Running Time and Model Size Table 9 reports training running time and model size for each selective rationalization evaluated in our experiments. It is worth noting that for GenSPP, we only report f_ω trainable parameters, which are the only ones trained during individual evaluation. If we consider g_θ parameters, the GenSPP size equals the one of FR.

[3] https://github.com/Lightning-AI/pytorch-lightning.

Name	Description
`emb_dim`	Input embedding dimension
`emb_type`	Pre-trained embedding matrix type
`num_classes`	Number of classification classes
`hidden_size`	Number of units in RNN layers
`cell_type`	Type of RNN layer for encoding
`num_generators`	Number of generators in MGR
λ_s	Coefficient for sparsity regularization $\mathcal{L}_s$
λ_c	Coefficient for contiguity regularization $\mathcal{L}_c$
λ_{kl}	Kullback-Lieber divergence coefficient in MCD
λ_{jsd}	Jensen-Shannon divergence coefficient in G-RAT
λ_g	Guider coefficient in G-RAT
`pretrain`	Number of guider pre-training epochs in G-RAT
g_decay	Guider regularization decay coefficient in G-RAT
σ	Attention noise in guider model in G-RAT
G	Number of genetic-based search generations in GenSPP
I	Population size in GenSPP
p^m	Mutation probability in GenSPP
p^c	Crossover probability in GenSPP
p^{sl}	Selection probability in GenSPP
p^{su}	Survival probability in GenSPP

Table 3: List of hyper-parameters in employed selective rationalize models.

Model	General	g_θ	f_ω	Learning
FR	emb_dim: 25 emb_type: 1-hot num_classes: 3	hidden_size: 8 cell_type: biGRU	hidden_size: 8 cell_type: biGRU	λ_s: 1.0 λ_c: 1.0
MGR	emb_dim: 25 emb_type: 1-hot num_classes: 3	hidden_size: 8 cell: biGRU num_generators: 3	hidden_size: 8 cell: biGRU	λ_s: 1.0 λ_c: 1.0
MCD	emb_dim: 25 emb_type: 1-hot num_classes: 3	hidden_size: 8 cell_type: biGRU	hidden_size: 8 cell_type: biGRU	λ_s: 1.0 λ_c: 1.0 λ_{kl}: 1.0
G-RAT	emb_dim: 25 emb_type: 1-hot num_classes: 3	hidden_size: 8 cell_type: biGRU	hidden_size: 8 cell_type: biGRU	λ_s: 1.0 λ_c: 1.0 λ_{jsd}: 1.0 λ_g: 1.0 pretrain: 10 g_decay: $1e^{-05}$ σ: 1.0
GenSPP	emb_dim: 25 emb_type: 1-hot num_classes: 3	hidden_size: 8 cell_type: GRU	hidden_size: 8 cell: GRU	G: 100 I: 50 p^m: 1.0 p^c: 1.0 p^{sl}: 0.5 p^{su}: 0.5

Table 4: Model hyper-parameters for Toy dataset.

Model	General	g_θ	f_ω	Learning
FR	`emb_dim`: 25 `emb_type`: GloVe `num_classes`: 2	`hidden_size`: 16 `cell_type`: biGRU	`hidden_size`: 16 `cell`: biGRU	λ_s: 1.0 λ_c: 0.0
MGR	`emb_dim`: 25 `emb_type`: GloVe `num_classes`: 2	`hidden_size`: 16 `cell_type`: biGRU `num_generators`: 3	`hidden_size`: 16 `cell`: biGRU	λ_s: 1.0 λ_c: 0.0
MCD	`emb_dim`: 25 `emb_type`: GloVe `num_classes`: 2	`hidden_size`: 16 `cell_type`: biGRU	`hidden_size`: 16 `cell_type`: biGRU	λ_s: 1.0 λ_c: 0.0 λ_{kl}: 1.0
G-RAT	`emb_dim`: 25 `emb_type`: GloVe `num_classes`: 2	`hidden_size`: 16 `cell_type`: biGRU	`hidden_size`: 16 `cell_type`: biGRU	λ_s: 1.0 λ_c: 0.0 λ_{jsd}: 1.0 λ_g: 2.5 `pretrain`: 10 `g_decay`: $1e^{-05}$ σ: 1.0
GenSPP	`emb_dim`: 25 `emb_type`: GloVe `num_classes`: 2	`hidden_size`: 16 `cell_type`: GRU	`hidden_size`: 16 `cell`: GRU	G: 100 I: 50 p^m: 1.0 p^c: 1.0 p^{sl}: 0.5 p^{su}: 0.5

Table 5: Model hyper-parameters for HateXplain dataset.

Model	Clf-F1 ↑	Hl-F1 ↑	R ↓	S ↓
FR ($\alpha = 0.10$)	$99.14_{\pm 1.23}$	$50.23_{\pm 8.32}$	$9.74_{\pm 0.47}$	$1.95_{\pm 0.09}$
MGR ($\alpha = 0.10$)	$99.56_{\pm 0.21}$	$40.02_{\pm 8.90}$	$10.03_{\pm 0.40}$	$2.01_{\pm 0.08}$
MCD ($\alpha = 0.10$)	$99.89_{\pm 0.05}$	$62.89_{\pm 2.20}$	$10.12_{\pm 0.38}$	$2.02_{\pm 0.08}$
G-RAT ($\alpha = 0.10$)	$99.42_{\pm 0.94}$	$50.33_{\pm 10.34}$	$9.98_{\pm 0.21}$	$2.00_{\pm 0.04}$
FR ($\alpha = 0.15$)	$99.78_{\pm 0.20}$	$54.07_{\pm 4.02}$	$14.80_{\pm 0.24}$	$2.96_{\pm 0.05}$
MGR ($\alpha = 0.15$)	$99.92_{\pm 0.04}$	$50.34_{\pm 11.23}$	$15.05_{\pm 0.80}$	$3.01_{\pm 0.16}$
MCD ($\alpha = 0.15$)	$99.90_{\pm 0.04}$	$65.70_{\pm 3.76}$	$15.18_{\pm 0.17}$	$3.04_{\pm 0.03}$
G-RAT ($\alpha = 0.15$)	$99.36_{\pm 0.82}$	$50.22_{\pm 7.78}$	$14.81_{\pm 0.51}$	$2.96_{\pm 0.10}$

Table 6: Test results on Toy when varying sparsity threshold α.

Model	Clf-F1 ↑	Hl-F1 ↑	$R\downarrow$	$S\downarrow$
FR ($\alpha = 0.10$)	$70.80_{\pm 1.15}$	$22.52_{\pm 14.64}$	$13.02_{\pm 0.87}$	$1.57_{\pm 0.07}$
MGR ($\alpha = 0.10$)	$69.74_{\pm 1.81}$	$28.13_{\pm 10.99}$	$13.76_{\pm 0.58}$	$1.64_{\pm 0.06}$
MCD ($\alpha = 0.10$)	$68.52_{\pm 2.79}$	$21.17_{\pm 15.03}$	$12.96_{\pm 0.57}$	$1.65_{\pm 0.03}$
G-RAT ($\alpha = 0.10$)	$71.33_{\pm 1.14}$	$40.40_{\pm 3.25}$	$13.00_{\pm 0.50}$	$1.58_{\pm 0.07}$
FR ($\alpha = 0.16$)	$71.90_{\pm 1.47}$	$27.34_{\pm 13.41}$	$19.31_{\pm 1.35}$	$2.49_{\pm 0.12}$
MGR ($\alpha = 0.16$)	$71.03_{\pm 0.70}$	$31.57_{\pm 5.60}$	$20.28_{\pm 1.25}$	$2.54_{\pm 0.08}$
MCD ($\alpha = 0.16$)	$70.03_{\pm 0.97}$	$25.60_{\pm 6.98}$	$19.65_{\pm 0.87}$	$2.60_{\pm 0.07}$
G-RAT ($\alpha = 0.16$)	$71.68_{\pm 1.23}$	$38.45_{\pm 2.31}$	$19.73_{\pm 0.78}$	$2.52_{\pm 0.04}$
FR ($\alpha = 0.22$)	$72.14_{\pm 1.12}$	$31.15_{\pm 2.56}$	$25.55_{\pm 0.72}$	$3.46_{\pm 0.11}$
MGR ($\alpha = 0.22$)	$71.14_{\pm 1.16}$	$29.38_{\pm 4.83}$	$25.30_{\pm 1.04}$	$3.42_{\pm 0.06}$
MCD ($\alpha = 0.22$)	$70.37_{\pm 1.06}$	$27.92_{\pm 1.66}$	$25.07_{\pm 1.52}$	$3.50_{\pm 0.20}$
G-RAT ($\alpha = 0.22$)	$73.85_{\pm 1.05}$	$36.17_{\pm 1.62}$	$24.68_{\pm 0.86}$	$3.34_{\pm 0.08}$
FR ($\alpha = 0.28$)	$73.09_{\pm 0.75}$	$29.41_{\pm 1.32}$	$31.08_{\pm 1.75}$	$4.35_{\pm 0.14}$
MGR ($\alpha = 0.28$)	$72.41_{\pm 0.95}$	$27.03_{\pm 3.28}$	$31.11_{\pm 1.72}$	$4.33_{\pm 0.12}$
MCD ($\alpha = 0.28$)	$70.26_{\pm 1.15}$	$25.98_{\pm 0.76}$	$30.29_{\pm 0.71}$	$4.34_{\pm 0.14}$
G-RAT ($\alpha = 0.28$)	$73.60_{\pm 0.71}$	$32.20_{\pm 0.96}$	$31.26_{\pm 0.80}$	$4.36_{\pm 0.11}$

Table 7: Test results on HateXplain when varying sparsity threshold α.

		Toy				**HateXplain**			
	Model	Clf-F1 ↑	Hl-F1 ↑	$R\downarrow$	$S\downarrow$	Clf-F1 ↑	Hl-F1 ↑	$R\downarrow$	$S\downarrow$
$K = 5$	FR	$99.94_{\pm 0.06}$	$50.32_{\pm 9.81}$	$14.62_{\pm 0.32}$	$2.92_{\pm 0.06}$	$70.86_{\pm 1.45}$	$14.91_{\pm 7.50}$	$27.35_{\pm 1.01}$	$3.48_{\pm 0.08}$
	MGR	$99.08_{\pm 1.54}$	$41.26_{\pm 18.58}$	$14.96_{\pm 0.68}$	$2.99_{\pm 0.14}$	$72.85_{\pm 0.77}$	$21.84_{\pm 9.56}$	$27.62_{\pm 1.13}$	$3.42_{\pm 0.12}$
	MCD	$99.94_{\pm 0.04}$	$65.18_{\pm 5.43}$	$15.12_{\pm 0.45}$	$3.02_{\pm 0.09}$	$70.02_{\pm 1.56}$	$22.72_{\pm 9.03}$	$25.54_{\pm 1.32}$	$3.53_{\pm 0.09}$
	G-RAT	$99.20_{\pm 1.37}$	$46.79_{\pm 12.37}$	$14.91_{\pm 0.13}$	$2.98_{\pm 0.03}$	$73.34_{\pm 0.34}$	$33.51_{\pm 1.20}$	$26.31_{\pm 0.99}$	$3.45_{\pm 0.11}$
$K = 10$	FR	$99.85_{\pm 0.11}$	$58.91_{\pm 3.18}$	$14.57_{\pm 0.12}$	$2.91_{\pm 0.02}$	$71.00_{\pm 0.76}$	$7.22_{\pm 2.29}$	$26.85_{\pm 0.92}$	$3.47_{\pm 0.08}$
	MGR	$97.75_{\pm 4.25}$	$37.45_{\pm 12.33}$	$14.99_{\pm 0.42}$	$3.00_{\pm 0.08}$	$71.09_{\pm 1.00}$	$14.45_{\pm 4.84}$	$27.75_{\pm 1.36}$	$3.57_{\pm 0.11}$
	MCD	$99.93_{\pm 0.06}$	$62.94_{\pm 2.39}$	$15.10_{\pm 0.66}$	$3.02_{\pm 0.13}$	$70.93_{\pm 0.95}$	$13.88_{\pm 10.13}$	$25.84_{\pm 1.87}$	$3.46_{\pm 0.16}$
	G-RAT	$99.85_{\pm 0.14}$	$47.53_{\pm 12.77}$	$14.56_{\pm 0.36}$	$2.91_{\pm 0.07}$	$73.15_{\pm 0.35}$	$34.33_{\pm 1.22}$	$25.43_{\pm 0.87}$	$3.40_{\pm 0.09}$
$K = 15$	FR	$99.85_{\pm 0.15}$	$51.44_{\pm 10.06}$	$14.90_{\pm 0.59}$	$2.98_{\pm 0.12}$	$70.95_{\pm 1.51}$	$8.21_{\pm 1.49}$	$25.85_{\pm 1.54}$	$3.39_{\pm 0.13}$
	MGR	$93.29_{\pm 11.14}$	$22.10_{\pm 8.79}$	$15.04_{\pm 0.48}$	$3.01_{\pm 0.10}$	$72.35_{\pm 0.69}$	$10.90_{\pm 5.51}$	$25.99_{\pm 1.66}$	$3.42_{\pm 0.07}$
	MCD	$99.91_{\pm 0.07}$	$62.78_{\pm 2.01}$	$14.94_{\pm 0.33}$	$2.99_{\pm 0.07}$	$69.58_{\pm 0.81}$	$11.04_{\pm 8.03}$	$25.37_{\pm 1.40}$	$3.41_{\pm 0.06}$
	G-RAT	$99.84_{\pm 0.21}$	$42.84_{\pm 8.17}$	$14.75_{\pm 0.47}$	$2.95_{\pm 0.09}$	$73.55_{\pm 0.32}$	$33.43_{\pm 2.25}$	$26.12_{\pm 1.80}$	$3.36_{\pm 0.13}$
$K = 20$	FR	$99.63_{\pm 0.45}$	$48.51_{\pm 13.52}$	$14.49_{\pm 0.36}$	$2.90_{\pm 0.07}$	$71.30_{\pm 1.25}$	$8.53_{\pm 2.21}$	$27.18_{\pm 0.87}$	$3.54_{\pm 0.08}$
	MGR	$89.05_{\pm 11.15}$	$18.53_{\pm 5.19}$	$16.07_{\pm 0.64}$	$3.21_{\pm 0.13}$	$71.30_{\pm 1.11}$	$13.26_{\pm 4.31}$	$27.31_{\pm 0.42}$	$3.53_{\pm 0.11}$
	MCD	$99.91_{\pm 0.08}$	$62.26_{\pm 2.50}$	$14.80_{\pm 0.35}$	$2.96_{\pm 0.07}$	$69.75_{\pm 1.35}$	$16.58_{\pm 9.54}$	$26.44_{\pm 2.40}$	$3.48_{\pm 0.20}$
	G-RAT	$66.95_{\pm 40.28}$	$29.43_{\pm 10.21}$	$21.43_{\pm 8.72}$	$4.29_{\pm 1.74}$	$73.57_{\pm 0.55}$	$33.71_{\pm 1.06}$	$26.53_{\pm 0.88}$	$3.47_{\pm 0.07}$

Table 8: Synthetic skew experiment results when varying skew pre-training epochs K.

Model	Single (min.)	Total (min.)	No. Parameters
		Toy	
FR	$7.41_{\pm 2.68}$	38.42	1797
MGR	$8.94_{\pm 2.67}$	46.04	7001
MCD	$6.55_{\pm 0.99}$	34.07	3477
G-RAT	$8.45_{\pm 4.08}$	43.19	6538
GenSPP	~ 36.00	~ 180.00	891 (f_ω)
		HateXplain	
FR	$2.52_{\pm 0.30}$	13.49	4324
MGR	$3.34_{\pm 0.13}$	18.11	17032
MCD	$2.84_{\pm 0.29}$	15.11	8452
G-RAT	$4.80_{\pm 0.43}$	25.39	16840
GenSPP	~ 78.00	~ 390.00	2098 (f_ω)

Table 9: Training running time and model size. We report single seed run running time (Single) and total running time over five seed runs (Total). Running time is measured in minutes. Additionally, we report the total number of trainable parameters.