MASARYKOVA UNIVERZITA

Přírodovědecká fakulta

Ústav matematiky a statistiky

Vysvětlitelnost modelů pro zpracování obrazu pomocí lokální selekce vlastností

Bakalářská práce

David Halmazňa

Vedoucí práce: doc. RNDr. Tomáš Brázdil, Ph.D., MBA

Brno 2026

Bibliografický záznam

Autor:	David Halmazňa Přírodovědecká fakulta, Masarykova univerzita Ústav matematiky a statistiky
Název práce:	Vysvětlitelnost modelů pro zpracování obrazu pomocí lokální selekce vlastností
Studijní program:	Matematika
Studijní obor:	Modelování a výpočty
Vedoucí práce:	doc. RNDr. Tomáš Brázdil, Ph.D., MBA
Akademický rok:	2025/2026
Počet stran:	viii + 63
Klíčová slova:	vysvětlitelná AI, zpracování obrazu, selekce příznaků, strojové učení, neuronové sítě

Bibliographic Entry

Author:	David Halmazňa Faculty of Science, Masaryk University Department of Mathematics and Statistics
Title of Thesis:	Explainability of Image Processing Models using Local Feature Selection
Degree Programme:	Mathematics
Field of Study:	Modelling and Calculations
Supervisor:	doc. RNDr. Tomáš Brázdil, Ph.D., MBA
Academic Year:	2025/2026
Number of Pages:	viii + 63
Keywords:	explainable AI, image processing, feature selection, machine learning, neural networks

Abstrakt

Hluboké neuronové sítě dosahují špičkových výsledků v řadě úloh počítačového vidění, jejich netransparentní způsob rozhodování však komplikuje nasazení v citlivých oblastech, jako je lékařské zobrazování nebo autonomní řízení. Současné black-box metody vysvětlitelné umělé inteligence (XAI) zpravidla produkují rozptýlené teplotní mapy nebo nespojité segmenty, jež je obtížné interpretovat jako ucelený důvod predikce.

Tato práce představuje framework formulující extrakci vysvětlení jako problém diskrétní kombinatorické optimalizace nad grafem sousednosti superpixelů. Pomocí heuristických vyhledávacích algoritmů odvozených od Monte Carlo Tree Search a Upper Confidence Bounds framework identifikuje prostorově souvislý podgraf s maximálním vlivem na výstup modelu. Framework hodnotíme vůči referenčním XAI metodám na syntetických, přirozených i patologických datech; ve věrnosti vysvětlení se srovnatelným metodám vyrovná nebo je překonává—přičemž jako jediný zaručuje prostorovou spojitost výstupních oblastí.

Abstract

Deep neural networks achieve state-of-the-art performance across many vision tasks, but their opaque decision-making limits adoption in high-stakes domains such as medical imaging and autonomous driving. Existing black-box Explainable AI (XAI) methods typically produce scattered heatmaps or disconnected segments that are difficult to interpret as a coherent structural reason for a prediction.

This thesis introduces a framework that formulates explanation extraction as a discrete combinatorial optimization problem over a region adjacency graph of superpixels. Using heuristic search algorithms based on Monte Carlo Tree Search and Upper Confidence Bounds, the framework discovers the spatially connected subgraph that most strongly influences the model's output. We evaluate the approach against state-of-the-art baselines across synthetic, natural-image, and digital pathology datasets, matching or outperforming established competitors in faithfulness metrics while providing a strict spatial connectedness guarantee that none of the evaluated baselines enforce.

MUNI
SCI

MASARYKOVA UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA
KOTLÁŘSKÁ 2, 611 37 BRNO
IČ: 00216224
DIČ: CZ00216224

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Akademický rok: 2025/2026

Ústav:	Ústav matematiky a statistiky
Student:	David Halmazňa
Program:	Matematika
Specializace:	Modelování a výpočty

Ředitel *ústavu* PřF MU Vám ve smyslu Studijního a zkušebního řádu MU určuje bakalářskou práci s názvem:

Název práce:	Vysvětlitelnost modelů pro zpracování obrazu pomocí lokální selekce vlastností
Název práce anglicky:	Explainability of Image Processing Models using Local Feature Selection
Jazyk práce:	angličtina

Oficiální zadání:
Cílem práce je navrhnout a implementovat algoritmus vysvětlující chování modelů pro zpracování obrazu. Jedná se o modifikaci a rozšíření metody LIME, které by mohlo vést k vylepšení reflexe prostorového uspořádání částí vstupního obrázku. Student se zapojí do aktivního výzkumu; v jeho rámci navrhne modifikovaný algoritmus, vytvoří prototypovou implementaci a porovná je s existujícími metodami. Výsledný systém bude aplikován na modely natrénované na obecných obrázcích z databáze ImageNet a modely trénované pro diagnostiku v digitální patologii. Součástí práce bude také popis teorie (např. teorie informace) použité pro návrh algoritmu a jeho zhodnocení.

Literatura: Novovičová, J., Somol, P., Haindl, M., Pudil, P. Conditional Mutual Information Based Feature Selection for Classification Task. In: Rueda, L., Mery, D., Kittler, J. (eds) Progress in Pattern Recognition, Image Analysis and Applications. CIARP 2007. Lecture Notes in Computer Science, vol 4756. Springer. 2007
Ribeiro, M. T., Singh, S., & Guestrin, C. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1135–1144. ACM. 2016

Vedoucí práce:	doc. RNDr. Tomáš Brázdil, Ph.D., MBA
Konzultant:	Mgr. Ondřej Pokora, Ph.D.
Datum zadání práce:	1. 10. 2025
V Brně dne:	26. 3. 2026

Zadání bylo schváleno prostřednictvím IS MU.

David Halmazňa, 18. 10. 2025
doc. RNDr. Tomáš Brázdil, Ph.D., MBA, 20. 10. 2025
RNDr. Jan Vondra, Ph.D., 20. 10. 2025

Poděkování

Rád bych na tomto místě poděkoval svému vedoucímu práce, doc. RNDr. Tomáši Brázdilovi, Ph.D., MBA, za odborné rady a především za to, že mi pomohl udržet si nadhled a ukázal mi, že k bakalářské práci lze přistupovat i bez zbytečného stresu. Mé poděkování patří také konzultantovi Mgr. Ondřeji Pokorovi, Ph.D., za jeho čas, ochotu a cenné připomínky, a to zejména ohledně formální stránky textu.

Mé další díky směřují do laboratoře RationAI, na jejíchž seminářích vznikaly samotné zárodky této práce. Jmenovitě bych chtěl poděkovat Vojtěchu Kůrovi za ochotu kdykoliv konzultovat libovolný problém a za rady ohledně struktury textu. RNDr. Vítu Musilovi, Ph.D., děkuji za nasměrování ve chvílích, kdy jsem nevěděl kudy dál, a Adamu Kukučkovi za trpělivost s mým kódem a pečlivé revize mých pull requestů.

Děkuji také všem svým kamarádům za to, že se mi dokázali nesmát ve chvílích, kdy jsem s psaním teprve začínal, zatímco oni už měli své práce téměř hotové.

V neposlední řadě děkuji své rodině za obrovskou podporu během celého mého studia, a obzvláště za to, že jsem byl dva víkendy před odevzdáním práce zproštěn povinnosti umývat nádobí.

Prohlášení

Prohlašuji, že jsem svoji bakalářskou práci vypracoval samostatně pod vedením vedoucího práce s využitím informačních zdrojů, které jsou v práci citovány.

Dále prohlašuji, že jsem při zpracování práce využil následující nástroje založené na umělé inteligenci:

- Gemini: ke korektuře anglické gramatiky, celkovému stylistickému vylepšení anglického textu a k asistenci při formulaci tohoto prohlášení.
- GitHub Copilot: jako pomoc při psaní zdrojového kódu a pro code review.
- Claude: k asistenci při programování, formátování v prostředí LaTeX (zejména při tvorbě grafiky pomocí nástroje TikZ) a pro podporu při práci s platformou MLflow (extrakce dat a tvorba vizualizací).

Výpočetní zdroje byly poskytnuty projektem e-INFRA CZ (ID:90140), podporovaným Ministerstvem školství, mládeže a tělovýchovy České republiky.

Brno, 11. května 2026 David Halmazňa

Contents

1 Introduction 1

2 Background 3
- 2.1 Neural Networks for Image Classification . . . 3
- 2.2 Image Segmentation and Adjacency Graphs . . . 4
- 2.3 Search Algorithms for Combinatorial Problems . . . 5

3 Related Work 8
- 3.1 White-Box Methods . . . 8
- 3.2 Black-Box Methods . . . 8
- 3.3 Graph-Based Explanations . . . 9

4 Problem Formulation 10
- 4.1 Problem Setting . . . 10
- 4.2 The Perturbation-Based Objective Function . . . 10
- 4.3 The Connected Region Explanation Problem . . . 12

5 Proposed Framework 13
- 5.1 Framework Overview . . . 13
- 5.2 Initial Segment Scoring and Seed Selection . . . 15
- 5.3 Region Expansion Algorithms . . . 15

6 Implementation 22
- 6.1 Technology Stack . . . 22
- 6.2 System Architecture . . . 22
- 6.3 Pipeline Execution . . . 23
- 6.4 Computational Optimizations . . . 23
- 6.5 Replacement Image Strategies . . . 24

7 Experiments 26
- 7.1 Experimental Setup . . . 26
- 7.2 Baselines and Competitors . . . 28
- 7.3 Hyperparameter Optimization . . . 30
- 7.4 Stability of Stochastic Algorithms . . . 30
- 7.5 Internal Method Comparison . . . 31
- 7.6 State-of-the-Art Comparison . . . 31

8 Discussion 36
8.1 Algorithmic Trade-offs 36
8.2 Stability of Stochastic Algorithms 36
8.3 Comparison with State of the Art 37
8.4 Limitations 38

9 Conclusion 39
9.1 Summary of Contributions 39
9.2 Future Directions 39

Bibliography 41

A MCTS Implementation Details 46

B Search Algorithm Configurations 49

C Competitor Configurations 55

D Full Internal Comparison Results 57

E FunnyBirds Evaluation Protocols 63

1 Introduction

Deep neural networks have achieved remarkable success in complex computer vision tasks, particularly in image classification. Consequently, they are increasingly deployed in high-stakes fields such as autonomous driving, finance, and medicine. However, their complex and non-linear nature makes them difficult to interpret. In critical applications, high accuracy alone is insufficient. A doctor, for example, cannot blindly trust an AI's diagnosis without understanding the visual reasoning behind it. This need for transparency has driven research in Explainable AI (XAI), which aims to highlight the specific image regions responsible for a model's prediction.

While existing XAI methods successfully highlight discriminative image features, they often produce fragmented heatmaps or scattered, disconnected segments. These disjointed explanations fail to express a cohesive, intuitive reason for a prediction.

This lack of spatial connectivity is especially critical in domains like histopathology, where a tumor presents as a contiguous mass of cells rather than a random collection of isolated pixels. If an XAI method highlights disconnected patches across a scan, the explanation lacks structural coherence. To be genuinely trustworthy, an explanation must capture the true semantic boundaries and contiguous nature of the target object.

To address this limitation, this thesis introduces a novel framework for generating connected, region-based explanations. By abstracting the input image into a region adjacency graph of superpixels, we preserve local image structure and formulate the explanation extraction as a discrete combinatorial optimization problem. To solve this, we propose multiple heuristic search algorithms to systematically explore the graph and discover the optimal connected subgraph. An overview of the full pipeline is given in Figure 1.1.

The main contributions of this thesis are the following:

- We formalize explanation extraction as a combinatorial optimization problem over a region adjacency graph of superpixels, enforcing spatial connectedness as a hard constraint throughout the search rather than as a post-processing step.

- We design and implement five heuristic search algorithms—Lookahead, Potential, Monte Carlo Tree Search (MCTS), Monte Carlo Graph Search (MCGS), and UCB—releasing them as an extensible, open-source Python library engineered so that new search strategies can be integrated via a single, unified interface.

- We benchmark the framework across synthetic (FunnyBirds [1]), natural (ImageNet-S [2]), and digital pathology (PatchCamelyon [3]) datasets. Notably, our approach surpasses several gradient-based, white-box methods in empirical faithfulness on

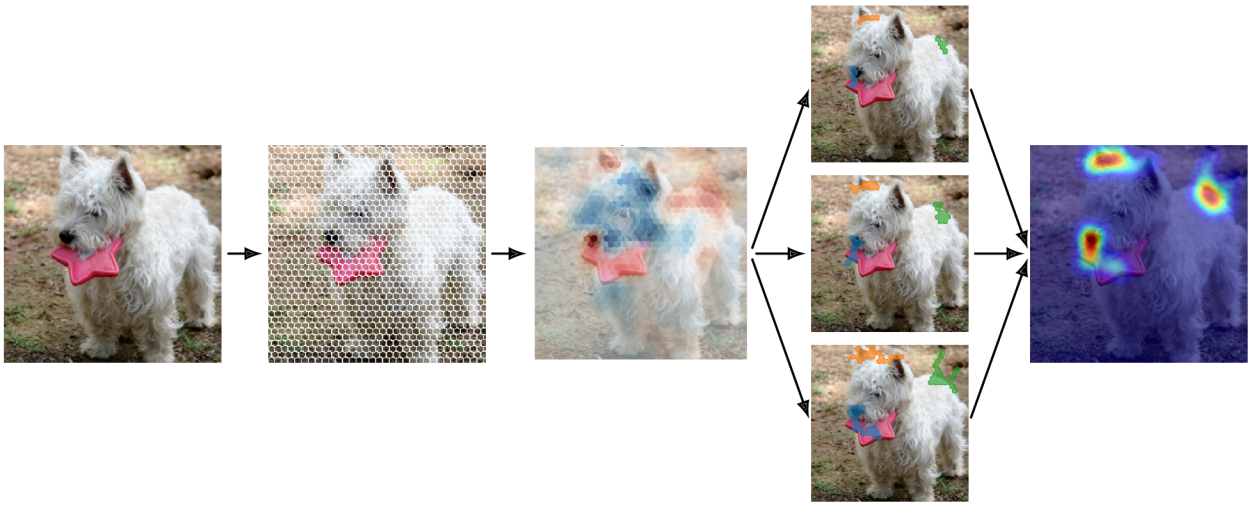


Figure 1.1: The explanation pipeline: (1) input image, (2) superpixel segmentation, (3) seed selection heatmap, (4) connected regions with different target lengths, (5) final saliency map.

FunnyBirds. Furthermore, it achieves the highest Insertion AUC among black-box competitors on ImageNet-S and performs competitively on PCam, all while providing a strict spatial connectedness guarantee that none of the evaluated baselines enforce.

The remainder of this thesis is structured as follows. Chapter 2 provides the necessary theoretical background on neural networks, graph structures, and search algorithms. Chapter 3 reviews existing XAI literature and establishes the limitations of current baseline methods. Chapter 4 mathematically formalizes the connected region explanation problem. Chapter 5 details the core mechanics of our proposed framework. Chapter 6 discusses specific implementation details and code optimizations. Chapter 7 compares different search algorithms and presents the evaluation of our method against state-of-the-art methods. Chapter 8 interprets the experimental results, examining the computational trade-offs and limitations of the framework. Finally, Chapter 9 summarizes the contributions of the thesis and outlines directions for future work.

2 Background

2.1 Neural Networks for Image Classification

Image classification tasks are primarily solved using architectures such as convolutional neural networks [4] and vision transformers [5]. While many XAI methods require access to the internal weights or gradients of these models, this requirement makes them strictly architecture-dependent. Furthermore, the internal mechanics of many state-of-the-art commercial models are proprietary. To ensure compatibility across any current or future model architectures and to work independently of commercial restrictions, a more robust approach is to treat the classifier entirely as a black box, interacting with it solely through its inputs and outputs.

When evaluating a black-box model, correctly handling its output space is critical. Most neural networks natively output unnormalized, unbounded class scores—the so-called logits. For each class $c \in \{1, \ldots, K\}$, we denote its logit by z_c. These are typically transformed into a valid probability distribution $p \in (0,1)^K$ by applying the softmax function, where each element p_c is defined as

$$p_c = \frac{\exp(z_c)}{\sum_{j=1}^{K} \exp(z_j)}. \tag{2.1}$$

For explanation methods that evaluate the model by masking or perturbing the input image, working directly with post-softmax probabilities is problematic. As Guo et al. showed, state-of-the-art models frequently suffer from overconfidence, meaning the output probability of the predicted class is often heavily saturated ($p_c \approx 1$) [6]. Consequently, masking out small semantic features in the input image might result in microscopic, numerically unstable drops in the probability space.

While working directly with raw class scores avoids the saturation problem, it fundamentally ignores class competition.

Neural networks also use features as negative evidence—to suppress competing classes, not just to support the target. Consider an image classified as class c, where masking a feature leaves z_c unchanged while causing the competing class score z_k to increase significantly. A method watching only z_c would see no change and declare the feature irrelevant. Yet, the spike in z_k enlarges the denominator, which consequently collapses p_c. The feature matters, but only the probabilities reveal it.

To resolve both the saturation of probabilities and the relative blindness of raw class scores, we employ a unified, model-agnostic metric. Let X denote the space of all valid input images. We model the classifier as a black-box function $p : X \to (0,1)^K$ mapping

any given image $I \in X$ to a probability distribution over K classes; if the network natively outputs logits, they are first normalized via softmax. This formulation naturally captures the competitive shifts between classes. We then map the probability of the target class $p_c(I)$ back to an unbounded interval $(-\infty, \infty)$ using the log-odds transformation:

$$f_c(I) = \log\left(\frac{p_c(I)}{1 - p_c(I)}\right). \tag{2.2}$$

Throughout the remainder of this text, we will refer to this transformed metric as the *log-odds score*. By evaluating the shift in this log-odds score rather than raw probabilities or uncalibrated class scores, we ensure the underlying changes in model confidence remain measurable, sensitive to microscopic shifts, and consistent across any black-box classifier, as depicted in Figure 2.1.

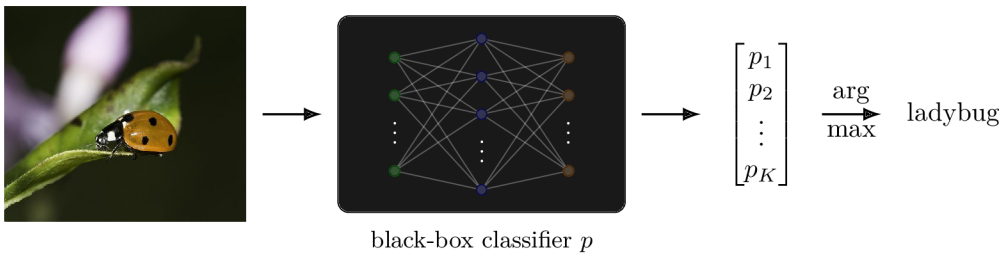


Figure 2.1: The black-box classifier $p : X \to (0,1)^K$. The model's weights and architecture are inaccessible; only input–output queries are permitted.

2.2 Image Segmentation and Adjacency Graphs

While a digital image is fundamentally composed of a discrete grid of individual pixels, evaluating the importance of each pixel independently is computationally intractable and provides limited semantic meaning. To address this, images are commonly abstracted into *superpixels*—contiguous groups of pixels that form localized, meaningful segments. By utilizing superpixels as the foundational building blocks rather than individual pixels, the dimensionality of the image space is reduced while preserving local spatial structures.

The process of partitioning an image into these segments is known as image segmentation. Data-driven algorithms such as Simple Linear Iterative Clustering (SLIC) [7] group nearby pixels with similar colors into compact segments, producing boundaries that follow image edges. An efficient alternative is to partition the image into a regular geometric grid of square or hexagonal tiles.

To systematically represent the spatial relationships between these segments, the image is abstracted into a region adjacency graph. Formally, a region adjacency graph is an undirected graph $G = (V, E)$. The set of vertices V represents the collection of unique superpixels. The edge set $E \subseteq V \times V$ represents spatial connectivity; an edge $(i, j) \in E$ exists if and only if superpixel i and superpixel j share a physical boundary in the

segmented image. This graph structure translates the image into a mathematical format suitable for our search algorithms, as illustrated in Figure 2.2.

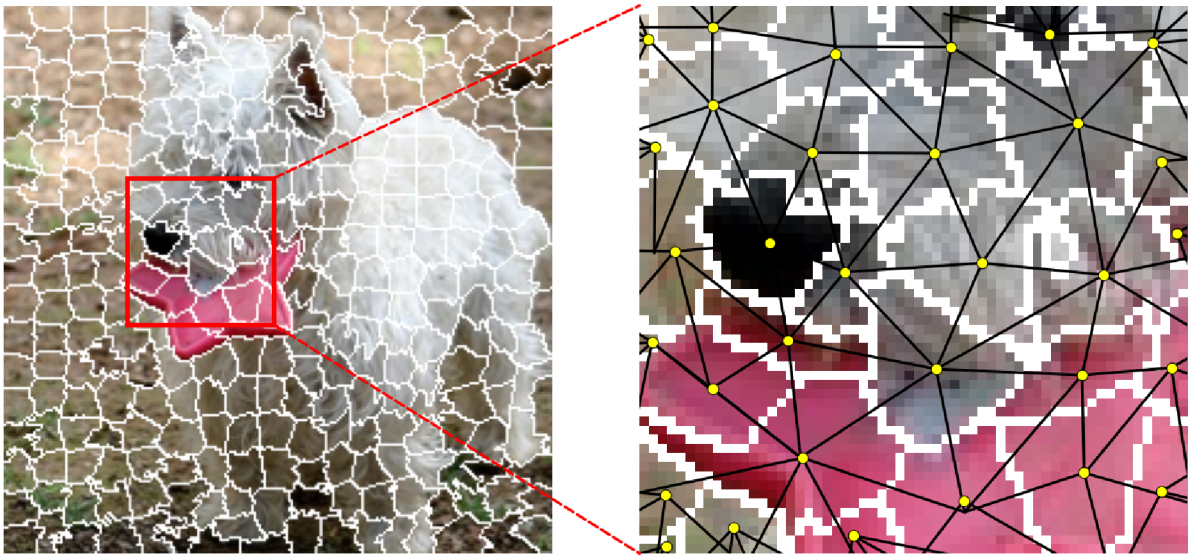


Figure 2.2: Superpixel segmentation and its region adjacency graph. Each superpixel becomes a vertex; shared boundaries become edges.

2.3 Search Algorithms for Combinatorial Problems

2.3.1 Exploration-Exploitation Tradeoff

Many combinatorial optimization problems possess search spaces so large that evaluating every possible state is computationally intractable. To navigate these spaces effectively, an algorithm must balance two competing objectives: exploiting the most promising known actions to maximize immediate outcomes, and exploring untested actions that might yield superior, undiscovered results.

This dilemma, known as the exploration-exploitation tradeoff, is a fundamental challenge in search algorithms and reinforcement learning.

2.3.2 Multi-Armed Bandits and UCB1

The classic mathematical model for this tradeoff is the multi-armed bandit problem. In this framework, an agent faces a set of choices (analogous to a row of slot machines), each characterized by an unknown, underlying reward distribution. Given a limited computational budget, the agent must strategically select actions to maximize cumulative reward based solely on past observations.

The Upper Confidence Bound (UCB1) algorithm [8] solves this by evaluating each potential action a according to the formula

$$\mathrm{UCB1}(a) = \bar{x}_a + c\sqrt{\frac{\ln n}{n_a}}, \tag{2.3}$$

where $\bar{x}_a$ is the empirical mean reward of action a, n is the total number of times any action has been selected, n_a is the number of times action a has been selected, and c is an exploration constant that controls the balance between the two terms.

This formula mathematically unifies the tradeoff. The left term drives exploitation by favoring actions that have yielded the highest average returns. The right term acts as an exploration bonus. Because n_a is in the denominator, infrequently selected actions receive a larger bonus. Furthermore, the slowly growing numerator $\ln n$ ensures that even initially unpromising actions are eventually re-evaluated, theoretically preventing the algorithm from becoming permanently trapped in a local optimum.

2.3.3 Monte Carlo Tree Search

MCTS [9] is a highly effective heuristic search algorithm, popularized largely by its success in mastering complex board games with enormous branching factors, such as chess and Go.

Before explaining the algorithm, we must formalize the search environment. Let the search space be defined by a set of states $\mathcal{S}$ and a set of valid actions $\mathcal{A}(s)$ for any state $s \in \mathcal{S}$. Executing an action maps the current state to a subsequent one. MCTS incrementally builds a search tree where each node v represents a state, the root node v_0 represents the initial state, and directed edges represent applied actions. A terminal state is defined as a state from which no further actions can be generated. Each node tracks its visit count $n(v)$ and its mean reward $Q(v)$.

MCTS navigates this space by iteratively repeating four phases until a predefined computational budget is exhausted:

- **Selection:** Starting from the root v_0, the algorithm traverses down the tree by selecting child nodes according to a specific policy (such as Upper Confidence Bound for Trees (UCT), detailed in Section 2.3.4). This traversal stops when it reaches an expandable node v_e (a node with at least one unexplored valid action).
- **Expansion:** If v_e is not a terminal state, one of its unexplored actions is selected, and a corresponding new child node v_c is added to the tree.
- **Simulation (Rollout):** From the new node v_c, a fast rollout policy (typically choosing actions uniformly at random) is applied sequentially until a terminal state is reached, producing a final reward.
- **Backpropagation:** The obtained reward is propagated back up the tree along the exact traversal path, updating the $n(v)$ and $Q(v)$ statistics for every node from v_c up to the root v_0.

After the algorithm finishes, the child of the root with the highest visit count or highest mean value is selected. Figure 2.3 shows one complete iteration.

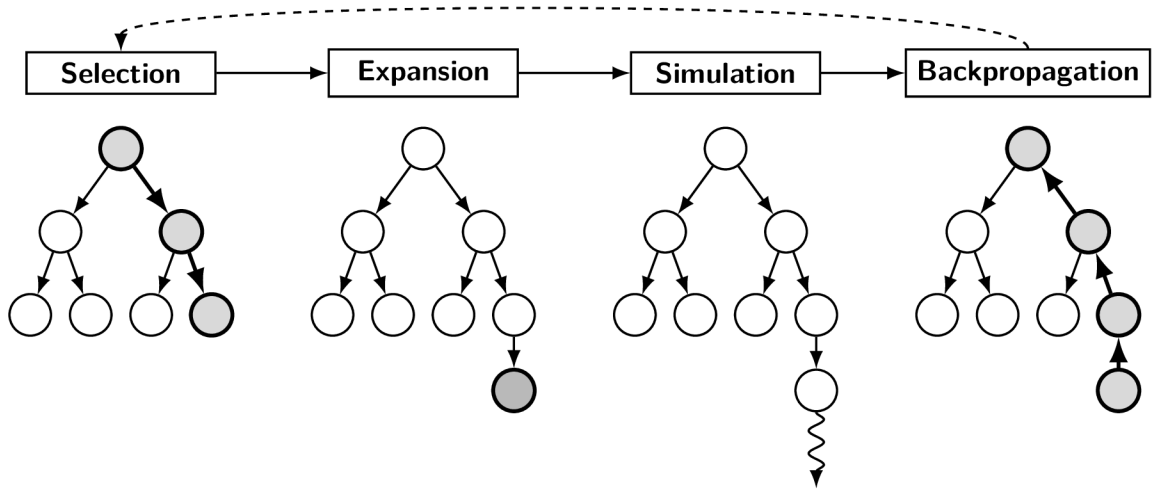


Figure 2.3: One iteration of Monte Carlo Tree Search. The four phases—Selection, Expansion, Simulation, and Backpropagation—are repeated until the computational budget is exhausted.

2.3.4 UCT and Single-Player Optimization

The efficiency of the MCTS Selection phase depends entirely on its traversal policy. The standard approach is UCT [9], which treats the selection of child nodes at each step as a multi-armed bandit problem. By applying the UCB1 formula at every node, UCT continuously balances the exploitation of high-reward paths with the exploration of less-visited branches. Kocsis and Szepesvári [9] showed that UCT directly inherits the mathematical properties of UCB1. Given rewards scaled to $[0, 1]$, it is theoretically guaranteed that the probability of selecting a suboptimal action at the root converges to zero at a polynomial rate.

While traditional MCTS is designed for two-player games—where the algorithm ultimately returns the most robust action at the root—applying MCTS to combinatorial optimization is fundamentally a single-player configuration [10]. In this context, there is no opponent and thus no uncertainty about its next action. Consequently, rather than selecting the safest root action, the algorithm can simply track and return the single globally optimal state discovered anywhere in the tree during the entire search process.

2.3.5 Monte Carlo Graph Search

A major structural limitation of standard MCTS is permutation invariance. In many domains, a state depends entirely on the set of chosen actions, not the order in which they were executed. In a standard search tree, multiple distinct paths leading to the exact same state are evaluated independently, wasting computational budget. MCGS [11] resolves this by replacing the underlying tree with a directed acyclic graph. By utilizing a transposition table, MCGS caches and merges identical states across different branches, allowing statistics to be shared globally.

3 Related Work

This chapter surveys the XAI methods most relevant to our work. We identify two recurring limitations: existing approaches often require white-box access to the model, or they fail to enforce the spatial connectivity and compactness of the highlighted area. Our framework is explicitly designed to address these shortcomings.

The field of XAI can be categorized based on the level of access required to the underlying model. This distinction between white-box and black-box approaches determines not only the computational efficiency of the method but also its flexibility across different model architectures.

3.1 White-Box Methods

White-box methods require full access to the model's architecture, weights, and biases. Because neural networks are differentiable, these methods often rely on backpropagated gradients to identify which parts of the input most strongly influence the final prediction. While computationally efficient, this approach is strictly tied to specific network architectures.

Although our framework is strictly model-agnostic, two notable white-box methods share our goal of generating compact explanations and are highly relevant to our problem formulation.

Grad-CAM++ [12] builds on Grad-CAM [13], which produces a coarse heatmap by pooling the gradients of the class score with respect to the last convolutional feature maps. Both methods are therefore applicable only to convolutional neural networks. Grad-CAM++ refines this by weighting each feature map channel with second-order gradients, producing sharper and better-localized activation maps. As white-box methods, they require full model access and produce continuous heatmaps with no strict guarantee of compactness or connectivity.

Meaningful Perturbations [14] learns a smooth mask by optimizing it via backpropagation to minimally preserve model confidence, under smoothness and sparsity regularization. This shares our objective of finding a compact, influential area, but its reliance on gradient-based optimization fundamentally limits it to differentiable models.

3.2 Black-Box Methods

Black-box methods treat the model purely as a mapping from inputs to outputs. They evaluate feature importance by systematically perturbing (altering or hiding) parts of the

image and observing the resulting change in the model output. This approach is highly practical, as it allows for the explanation of any model architecture, including commercial APIs where internal weights remain hidden.

LIME (Local Interpretable Model-agnostic Explanations) [15] is a foundational black-box method that serves as a primary inspiration for our approach. LIME segments an image into superpixels and generates a surrogate dataset by masking random subsets of these segments. A simple, interpretable model (typically a linear regression with LASSO regularization) is trained on this surrogate dataset to locally approximate the neural network's behavior. While highly influential, LIME is often unstable. Because it treats superpixels as largely independent variables, the resulting explanations frequently consist of scattered, fragmented patches rather than cohesive objects.

Occlusion Sensitivity [16] evaluates feature importance by sliding a fixed-size mask across the image and recording the drop in the model's prediction. The results are aggregated into a continuous heatmap. While this method successfully captures local importance, it does not explicitly account for the semantic boundaries of objects or the connectivity of the highlighted features.

3.3 Graph-Based Explanations

While traditional image-based XAI operates directly on pixel grids or independent superpixels, a parallel domain focuses on explaining Graph Neural Networks (GNNs). Because our framework abstracts segmented images into a region adjacency graph, the search algorithms designed for the GNN domain are highly relevant to our methodology.

MAGE [17] identifies influential structural motifs using complex interaction indices. To evaluate vision models, the authors transformed images into superpixel adjacency graphs—validating the structural abstraction adopted in this thesis. However, while MAGE focuses on node-to-node interactions, our framework is specifically designed to optimize for the single most influential contiguous region.

SubgraphX [18] identifies explanatory subgraphs by utilizing MCTS to iteratively prune the least important nodes from the full graph. While it shares our use of MCTS, SubgraphX performs a top-down reduction. In contrast, our framework employs a bottom-up search. By incrementally building a connected explanation outward from a single seed superpixel, we can more effectively focus the computational budget on the most influential spatial components from the very first evaluation.

4 Problem Formulation

4.1 Problem Setting

Let $X \subset \mathbb{R}^{H \times W \times C}$ be the space of input images, where H, W, C are the height, width, and color channels, and K is the number of target classes. Let $p : X \to (0,1)^K$ represent the black-box neural network classifier, mapping an input image to a probability distribution across all classes. For a given image $I \in X$, let $p_c(I)$ denote the predicted probability for a specific target class $c \in \{1, \ldots, K\}$.

To evaluate the model in an architecture-agnostic space, we use the log-odds score introduced in Section 2.1. We restate it here for completeness: $f_c(I) \in \mathbb{R}$ is defined as

$$f_c(I) = \log\left(\frac{p_c(I)}{1 - p_c(I)}\right). \tag{4.1}$$

We assume the target class c is fixed for a given explanation task; in practice, c is typically the class with the highest predicted probability on the unoccluded image.

As established in Section 2.2, we abstract the input image I into a region adjacency graph $G = (V, E)$, where V represents the set of superpixels and E represents their spatial adjacency.

We define an *explanation region* R as a subset of vertices $R \subseteq V$ such that the subgraph induced by R, denoted $G[R]$, is connected. Formally, R is a valid region if for any pair of vertices $u, v \in R$, there exists a path between them such that all intermediate vertices also belong to R.

Let $m \in \mathbb{N}$ be the maximum allowed size (number of segments) of an explanation. The search space of all valid explanations is the set of all connected induced subgraphs bounded by size m, denoted by

$$\Omega_m = \{R \subseteq V \mid G[R] \text{ is connected and } |R| \leq m\}. \tag{4.2}$$

In practice, the search algorithms attempt to grow the region to the target size m; the inequality accommodates cases where the frontier is exhausted before reaching m, as illustrated in Figure D.6 and discussed in Section D.3.

4.2 The Perturbation-Based Objective Function

To identify an optimal explanation, we evaluate the importance of a region by occluding it and observing the resulting change in the target log-odds score f_c. If the log-odds score

drops significantly after occlusion, the region provides *positive evidence* for the class. Conversely, if the log-odds score rises, the region provides *negative evidence*, indicating that its presence suppressed the model's confidence.

Formally, let $\mathcal{P} = \{1, \ldots, H\} \times \{1, \ldots, W\}$ denote the set of pixel coordinates, and let $\phi : \mathcal{P} \to V$ be the assignment map that sends each pixel to its containing superpixel. We define a binary mask $M_R \in \{0,1\}^{H \times W}$ that indicates the pixels belonging to the selected region R as

$$M_R(h, w) = \begin{cases} 1 & \text{if } \phi(h, w) \in R, \\ 0 & \text{otherwise.} \end{cases} \tag{4.3}$$

While one could evaluate R in isolation by occluding its background (a preservation approach), we consistently employ a deletion approach throughout this work, evaluating regions by occluding them and measuring the resulting drop in model confidence.

Given the original image I and a replacement image $I_{rep} \in X$ that acts as the perturbation baseline, the occluded image $I_R \in X$ is defined by applying the mask across all C color channels. For each coordinate (h, w, k), where $k \in \{1, \ldots, C\}$, the value is given by

$$I_R(h, w, k) = (1 - M_R(h, w)) \cdot I(h, w, k) + M_R(h, w) \cdot I_{rep}(h, w, k). \tag{4.4}$$

A detailed discussion regarding the specific strategies for choosing I_{rep} is provided in Section 6.5. An example occlusion is shown in Figure 4.1.

(a) Original image I with region R outlined.

(b) Replacement image I_{rep} (solid color).

(c) Occluded image I_R seen by the classifier.

Figure 4.1: Region occlusion. The selected region R is replaced by the corresponding pixels from I_{rep}, producing the perturbed image I_R that is passed to the classifier.

We quantify the importance of the region via the base score function $S(R)$ as the difference in the model's output before and after occlusion, expressed as

$$S(R) = f_c(I) - f_c(I_R). \tag{4.5}$$

To unify the search for both positive and negative evidence under a single maximization problem, we introduce a sign parameter $\sigma \in \{-1, 1\}$ to define the parameterized objective function

$$g_\sigma(R) = \sigma \cdot S(R). \tag{4.6}$$

Setting $\sigma = 1$ directs the algorithm to maximize the log-odds score drop (positive evidence), while $\sigma = -1$ maximizes the log-odds score increase (negative evidence).

4.3 The Connected Region Explanation Problem

The goal of our framework is to solve a discrete combinatorial optimization problem. For a given input image I, replacement image I_{rep}, target class c, budget m, and evidence direction $\sigma \in \{-1, 1\}$, we aim to find the optimal connected region R^* that maximizes the objective function

$$R^* = \arg\max_{R \in \Omega_m} g_\sigma(R). \tag{4.7}$$

Since g_σ places no structural assumptions on f_c, the classifier could in principle return an arbitrary value for every candidate region. Without such assumptions, exhaustive evaluation would be necessary to guarantee optimality. The number of connected subgraphs of size at most m grows exponentially with m, making exact search computationally intractable. Furthermore, evaluating $g_\sigma(R)$ requires a forward pass through the neural network, so the objective is non-additive—the importance of a region is not simply the sum of its parts—and offers no guarantees of submodularity, which would otherwise allow greedy algorithms with provable approximation bounds. Heuristic search is therefore necessary.

Assuming the objective behaves completely arbitrarily would, however, be overly pessimistic. Deep networks trained on visual data learn spatial and semantic coherence, so adding a single superpixel to a region typically changes the prediction by only a small amount. This implicit smoothness is what makes heuristic search effective in practice: a high-scoring partial region tends to be a good stepping stone toward a high-scoring complete region. This well-behavedness is loosely related to Lipschitz continuity, but we do not formally assume it.

5 Proposed Framework

5.1 Framework Overview

The primary objective of the proposed framework is to generate visual explanations by identifying contiguous components within an image that strongly influence a model's prediction for a specific class. To achieve this, our pipeline executes three core stages: (i) segmenting the image into superpixels to construct a spatial region adjacency graph, (ii) identifying a promising starting node (*seed*), and (iii) iteratively expanding a connected region that maximizes the objective function $g_\sigma(R)$ defined in Section 4.2. An end-to-end illustration of the pipeline is given in Figure 1.1.

While the formal optimization problem in Chapter 4 focuses on identifying a single optimal region R^*, complex visual reasoning often involves multiple distinct features. We therefore adopt an iterative approximation: once we identify an influential region, we remove its constituent vertices from the graph and repeat the search on the remaining graph to find the next disjoint component. Algorithm 1 summarizes this process, Figure 5.1 illustrates the mechanism, and Figure 5.2 shows real examples of the resulting output.

Algorithm 1 Graph-Constrained Iterative Region Extraction

Require: Image I, replacement image I_{rep}, model p, target class c, evidence direction σ, size limit m, region count k, evaluation budget T

Ensure: A set of disjoint explanatory regions $\mathcal{R}$

```
P ← SegmentImage(I)
G ← BuildAdjacencyGraph(P) /* Construct initial graph G = (V, E) */
R ← ∅
for i = 1 to k do
    v_seed ← SelectSeed(G, p, I, c) /* Detailed in Section 5.2 */
    R_i ← GrowRegion(v_seed, G, g_σ, m, T) /* Detailed in Section 5.3 */
    R ← R ∪ {R_i}
    G ← G \ R_i /* Prune R_i to ensure disjointness */
end for
return R
```

Figure 5.1: Iterative identification of multiple disjoint explanatory regions. Each region is found by repeating the search on the graph after removing the previously found region from the adjacency graph.

Figure 5.2: Real examples of multi-region output on ImageNet images.

5.2 Initial Segment Scoring and Seed Selection

The effectiveness of any region-growing algorithm depends heavily on identifying an optimal starting node. Since initiating a full combinatorial search from every possible superpixel is computationally prohibitive, we employ a localized heuristic.

Evaluating a single, isolated superpixel often fails to capture the spatial context required by deep neural networks. To address this, we evaluate superpixels within their local neighborhood. Let $N(i) = \{j \in V \mid (i,j) \in E\}$ denote the set of direct neighbors of superpixel i in G. We define the cluster $\mathcal{N}_i = \{i\} \cup N(i)$ and calculate its objective score $g_\sigma(\mathcal{N}_i)$.

Because these 1-hop neighborhoods overlap, a single superpixel j is included in the evaluation of multiple clusters. We leverage this overlap to estimate the marginal contribution of superpixel j by averaging the scores of all clusters that contain it. This effectively acts as a spatial mean filter on the graph, reducing segmentation noise. Formally, the smoothed score $\bar{g}_\sigma(j)$ is defined as

$$\bar{g}_\sigma(j) = \frac{1}{|C_j|} \sum_{i \in C_j} g_\sigma(\mathcal{N}_i), \tag{5.1}$$

where $C_j = \{i \mid j \in \mathcal{N}_i\}$ is the set of cluster indices containing j. The seed node is then selected as

$$v_{seed} = \arg\max_{v \in V} \bar{g}_\sigma(v). \tag{5.2}$$

Figure 5.3 shows an example of the resulting heatmap and the selected seed.

5.3 Region Expansion Algorithms

Once an initial seed is selected, the framework iteratively grows the region by selecting adjacent vertices from G. We define the frontier $F(R) = \{v \in V \setminus R \mid \exists\, u \in R : (u,v) \in E\}$ as the set of vertices not in R that are adjacent to at least one vertex in R; expansion steps always select from $F(R)$. We propose and evaluate multiple expansion strategies. All algorithms can be interpreted as approximate solvers for the spatial optimization problem, differing primarily in how they navigate the combinatorial search space.

5.3.1 Policy-Guided Stepwise Expansion

One class of expansion algorithms makes permanent, stepwise commitments. At each step, the algorithm evaluates the valid boundary (frontier) of the current region R, selects a single node to add, and repeats until the size limit m (number of segments) is reached.

We abstract this implementation of the GrowRegion interface in Algorithm 2. The critical component defining the behavior of the search is the evaluation policy $\pi(v \mid R)$, which estimates the value of adding a specific candidate superpixel v. We define four distinct policies of increasing complexity. Notably, for stochastic policies that utilize rollouts to estimate π, the algorithm tracks and ultimately returns the highest-scoring

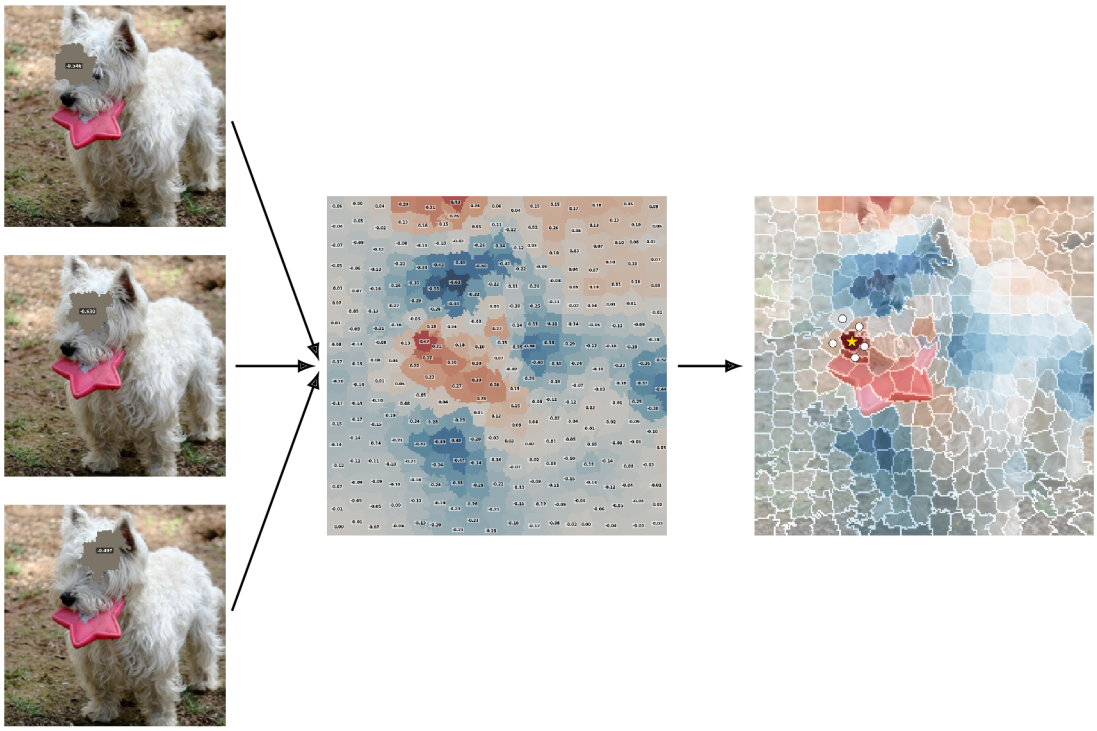


Figure 5.3: Seed selection on an example image of a West Highland Terrier. Left: three sample 1-hop neighborhood clusters with their log-odds scores. Center: the full smoothed score heatmap $\bar{g}_\sigma$ composed of all cluster evaluations; warm colors indicate high scores, cool colors low scores. Right: the resulting superpixel boundaries overlaid on the heatmap, with the highest-scoring superpixel marked by a yellow star and its 1-hop neighborhood by white circles.

complete region encountered during any rollout, rather than strictly returning the final committed path.

Algorithm 2 Policy-Guided Stepwise Region Expansion

Require: Seed v_{seed}, graph G, objective g_σ, size limit m, budget T, evaluation policy π

Ensure: The highest-scoring region R_{best}

```
procedure GrowRegion_Stepwise(v_seed, π)
    R ← {v_seed}
    R_best ← R
    while |R| < m do
        F ← ValidNeighbors(R, G)
        if F is empty then
            break
        end if
        v* ← arg max_{v∈F} π(v | R) /* Stochastic policies update R_best internally */
        R ← R ∪ {v*}
    end while
    return R_best
end procedure
```

Greedy Policy

The strictly greedy policy evaluates only the immediate outcome of the expansion

$$\pi_{\text{greedy}}(v \mid R) = g_\sigma(R \cup \{v\}). \tag{5.3}$$

While efficient, this policy is shortsighted and vulnerable to local optima; it cannot traverse a superpixel with a temporarily low score to reach a highly important feature later.

Lookahead Policy

To mitigate the myopic nature of greedy expansion, the lookahead policy evaluates short sequences of future expansions. Let $\mathcal{S}_d(R \cup \{v\})$ denote the set of all valid connected regions formed by adding up to $d-1$ additional superpixels to the base region $R \cup \{v\}$. The policy assigns the candidate node v the value of the highest-scoring region in this reachable lookahead set

$$\pi_{\text{lookahead}}(v \mid R) = \max_{R' \in \mathcal{S}_d(R \cup \{v\})} g_\sigma(R'). \tag{5.4}$$

While this allows the algorithm to bypass local optima by looking d steps ahead, the size of $\mathcal{S}_d$ scales exponentially with d, making large depths computationally intractable. Note that $d = 1$ recovers the greedy policy, while $d = m - |R|$ would exhaustively evaluate all valid completions.

Potential Policy

To approximate deep lookahead without the exponential cost, stochastic methods utilize random rollouts. For a given budget T, each step is allocated $\lfloor T/m \rfloor$ rollouts. Starting from a frontier node v, a region is completed to size m via a random walk on G, iteratively adding uniformly sampled adjacent vertices.

Because rollouts initiated by one node often overlap with other frontier nodes, a candidate node accumulates evidence from any rollout that passes through it. Let $\mathcal{W}(v)$ denote the set of all rollout regions generated in the current step that contain vertex v. The policy evaluates v by constructing a descending-sorted vector of these rollout scores—its *potential vector*:

$$\pi_{\text{potential}}(v \mid R) = \text{SortDesc}\Big(\{g_\sigma(R') \mid R' \in \mathcal{W}(v)\}\Big). \tag{5.5}$$

The arg max in Algorithm 2 then performs a lexicographic comparison of these vectors, committing to the node most robustly associated with high-scoring outcomes.

Batched UCB Policy

The UCB variant treats the frontier nodes as arms in a multi-armed bandit problem. It distributes the step budget $\lfloor T/m \rfloor$ across frontier nodes by dynamically balancing

exploitation and exploration. An internal sampling loop selects nodes maximizing the acquisition function

$$\mathrm{acq}(v) = \alpha \cdot \max_{v} + (1-\alpha) \cdot \bar{g}(v) + c\sqrt{\frac{\ln n}{n_v}}, \tag{5.6}$$

where $\alpha \in [0,1]$ is a max/mean blend weight adapted from single-player Monte Carlo Tree Search [10], $\max_v$ and $\bar{g}(v)$ are the running maximum and mean rollout scores for node v, n is the total rollouts completed so far in the current step, and n_v is the number of times node v has been selected in the current step. All counts reset at the start of each commit step.

Once the step budget is exhausted, the policy returns the total pull count of the arm, committing to the node the bandit strategy explored most

$$\pi_{\mathrm{UCB}}(v \mid R) = n_v. \tag{5.7}$$

This variant introduces two additional hyperparameters: the exploration constant c and the max/mean blend weight α.

5.3.2 Monte Carlo Tree Search

While the stepwise methods permanently commit to a node and discard alternatives, Monte Carlo Tree Search (MCTS) maintains a persistent global search structure across all iterations. Mapping to the formalism from Section 2.3.3: each node represents a connected region $R \subseteq V$, a valid action corresponds to adding a frontier node $v \in F(R)$ to R, and a terminal state is reached when $|R| = m$ or $F(R) = \emptyset$. The root holds the singleton region $\{v_{seed}\}$, and the rollout reward is $g_\sigma(R_{terminal})$. It continuously refines value estimates to direct the computational budget toward the most promising branches (Algorithm 3). This formulation is a deterministic MDP, so the UCT convergence results from Section 2.3.4 apply in principle; in our setting, however, rewards are unbounded and the objective is global-optimum discovery rather than optimal root-action selection, so the guarantee does not transfer directly and serves as a motivation rather than a formal claim.

Unlike classical decision-making settings where the objective is to identify the best action at the root, our goal is to discover the single highest-scoring region anywhere in the search space. Consequently, the role of the selection policy shifts from optimizing expected outcomes to directing computational effort toward branches likely to contain high-value terminal states.

To achieve this, we replace the standard mean-based UCT estimate with the same max/mean blend utilized in our UCB variant (Section 5.3.1). Following the single-player MCTS formulation proposed by Schadd et al. [10], we define

$$Q_\alpha(\mathrm{ch}) = \alpha \cdot Q_{\max}(\mathrm{ch}) + (1-\alpha) \cdot \overline{Q}(\mathrm{ch}), \tag{5.8}$$

where $Q_{\max}(\text{ch})$ is the maximum reward over rollouts passing through child ch, $\overline{Q}(\text{ch})$ is the corresponding mean, and $\alpha \in [0, 1]$ is the same blend weight as in the UCB variant. Because our log-odds objective is unbounded, these Q-values cannot be directly balanced against a fixed exploration constant c. To resolve this, we adopt the local normalization proposed by Seify and Buro [19]. Let $q_{\min}$ and $q_{\max}$ be the minimum and maximum Q_α among a parent's children. We map each child's score into $[-1, 1]$ via

$$\widetilde{Q}(\text{ch}) = \begin{cases} 2 \cdot \dfrac{Q_\alpha(\text{ch}) - q_{\min}}{q_{\max} - q_{\min}} - 1 & \text{if } q_{\max} > q_{\min}, \\ 1 & \text{otherwise.} \end{cases} \tag{5.9}$$

The fallback value of 1 is optimistic: when all children share the same Q_α value, the exploration term fully governs selection, directing rollouts toward the least-visited child and promoting broad coverage across all branches before revisiting. The selection phase then chooses the child maximizing

$$\text{UCT}(\text{ch}) = \widetilde{Q}(\text{ch}) + c\sqrt{\frac{\ln n(\text{parent})}{n(\text{ch})}}. \tag{5.10}$$

Algorithm 3 Monte Carlo Tree Search Main Loop

Require: Seed v_{seed}, objective g_σ, size limit m, budget T, batch size B, exploration constant c, max/mean weight α

Ensure: The highest-scoring region R_{best} found during search

```
procedure GrowRegion_MCTS(v_seed, g_σ, m, T, B, c, α)
    root ← MCTSNode({v_seed})
    R_best ← {v_seed},    S_best ← −∞
    for i = 1 to ⌊T/B⌋ do
        node, path ← Select(root, m, c, α)
        node, path ← Expand(node, path, m)
        batch ← Simulate(node, m, B)
        rewards ← EvaluateBatch(batch)
        for j = 1 to |batch| do
            if rewards[j] > S_best then
                S_best ← rewards[j],    R_best ← batch[j]
            end if
        end for
        Backpropagate(path, rewards)
    end for
    return R_best
end procedure
```

Each iteration evaluates B rollout regions from the selected leaf as a single GPU forward pass. The SELECT, EXPAND, SIMULATE, and BACKPROPAGATE procedures are detailed in Appendix A.

5.3.3 Monte Carlo Graph Search

Standard MCTS treats different action sequences as distinct paths, even if they lead to the same region. Monte Carlo Graph Search eliminates this redundancy by replacing the search tree with a directed acyclic graph (DAG).

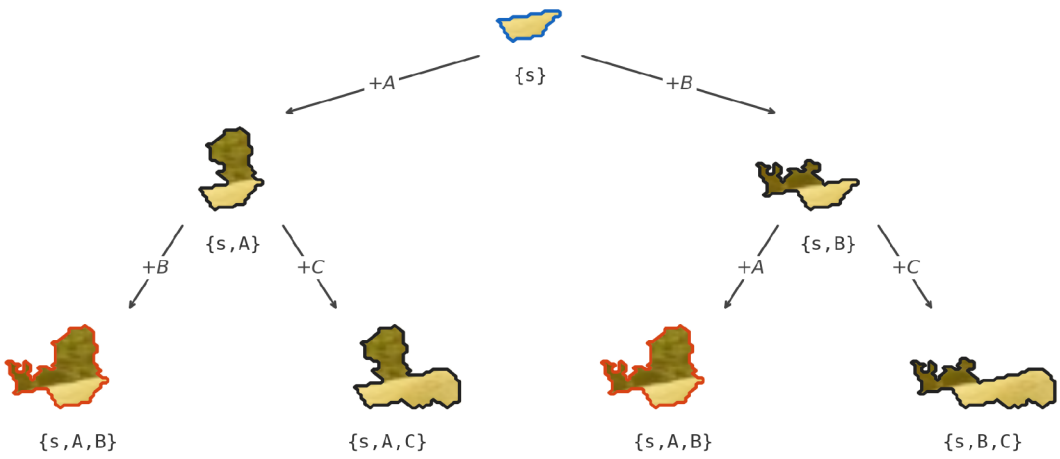
A transposition table maps each region to a single shared node. During expansion, if a candidate child region already exists in the table, the algorithm grafts an edge to the existing node rather than creating a new one. All paths arriving at the same region therefore share one set of accumulated value estimates.

Because a node can be reached from multiple parents, edge statistics (visit counts) are tracked per (parent, action) pair. The Q-value of a node is computed recursively as a weighted combination of its direct rollout rewards and its children's Q-values

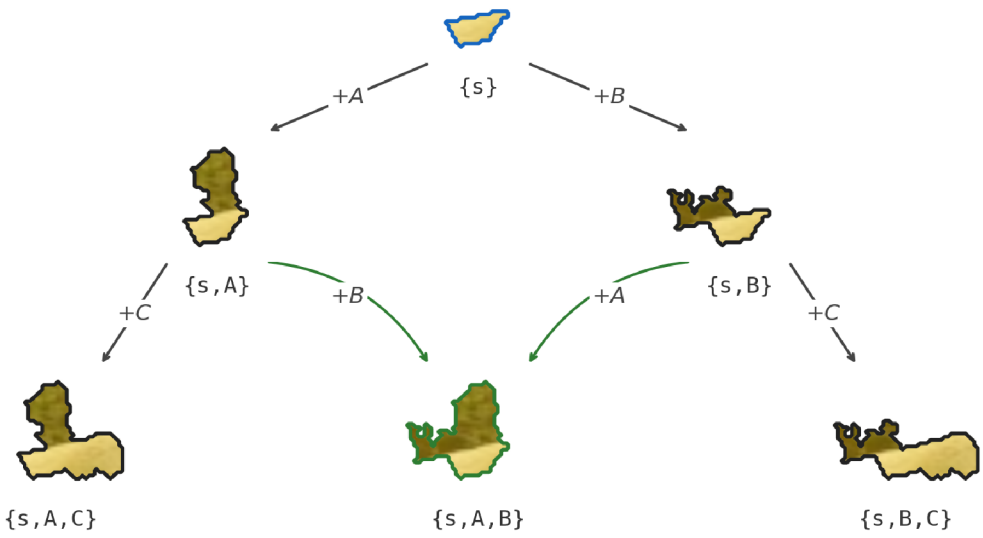
$$Q(n) = \frac{V_n + \sum_a e_a \cdot Q(\text{child}(n,\, a))}{V_n + \sum_a e_a}, \tag{5.11}$$

where V_n is the sum of rewards from rollouts evaluated directly at n, and e_a is the visit count of the outgoing edge for action a. This ensures that a node's value reflects the full subgraph below it, and backup must therefore propagate bottom-up after each update. Figure 5.4 contrasts the MCTS tree with the MCGS DAG.

This style of graph-based Monte Carlo search was notably employed by KataGo [20], where transposition merging proved critical for search efficiency in Go.



(a) MCTS tree: the state $\{s, A, B\}$ is duplicated (highlighted in red) because the order of actions matters in a tree structure.



(b) MCGS DAG: both orderings converge to a single node for $\{s, A, B\}$ (highlighted in green), sharing statistics and saving budget.

Figure 5.4: MCTS tree (top) versus MCGS DAG (bottom). Starting from seed s, adding regions A and B in different orders reaches the same state. MCTS evaluates it twice; MCGS merges them into a single node, eliminating redundant evaluations.

6 Implementation

The framework is named CIAO—an acronym for Contextual Importance Assessment via Obfuscation—reflecting its core mechanism: it quantifies the importance of image areas by obfuscating them and measuring the effect on the model output. For conciseness, this chapter provides a high-level architectural overview; complete technical details are available in the public source code at https://github.com/RationAI/ciao. All experiments presented in this thesis were conducted on the `test/all-algorithms` branch, while ongoing framework development continues on the `master` branch.

6.1 Technology Stack

The framework is implemented in Python and uses `uv` for dependency management. The core libraries include:

- **PyTorch** [21] for all tensor operations, device management, and model inference. Pre-trained classifiers are sourced from `torchvision` [21] and `timm` [22].
- **scikit-image** [23] for the SLIC superpixel segmentation algorithm.
- **Hydra** [24] for hierarchical configuration management, enabling experiment composition from modular YAML files without requiring code changes.
- **MLflow** [25] for experiment tracking, metric logging, and artifact storage.

6.2 System Architecture

The framework uses the Strategy design pattern. The main entry point, `CIAOExplainer`, takes interfaces for three components: `SegmentationFn`, `ExplanationMethodFn`, and `ReplacementFn`. Concrete implementations are composed at runtime by Hydra from YAML configuration files, ensuring that switching algorithms or segmentation methods requires no modifications to the core codebase. The top-level package layout is shown in Figure 6.1.

Adding a new search algorithm simply requires implementing the search logic within the `algorithm/` module and registering a constructor that maps the specific hyperparameters to the standardized `ExplanationMethodFn` type signature.

```
ciao/
├── algorithm/     # Search algorithms (MCTS, MCGS, Lookahead, etc.)
├── data/          # Preprocessing, loaders, and segmentation routines
├── explainer/     # Core pipeline (CIAOExplainer)
├── metrics/       # Saliency and segmentation evaluation metrics
├── model/         # Unified model inference wrapper
├── scoring/       # Surrogate dataset creation and region evaluation
└── visualization/ # Artifact generation and tree/graph rendering
```

Figure 6.1: Simplified high-level directory structure of the CIAO Python package.

6.3 Pipeline Execution

The explanation lifecycle is encapsulated within the `CIAOExplainer.explain()` method, executed in four sequential stages:

1. **Input Preparation:** The image is loaded and preprocessed; subsequently, the `ReplacementFn` constructs the perturbation baseline image I_{rep}.

2. **Graph Construction:** The `SegmentationFn` partitions the image and builds an `ImageGraph`—an adjacency list mapping superpixel indices. This allows the search algorithms to operate efficiently on integer segment IDs rather than dense pixel arrays.

3. **Segment Scoring:** The `create_surrogate_dataset` routine masks each segment individually and evaluates the log-odds drop via batched forward passes through the `ModelPredictor`. This produces the localized per-segment scores required for seed selection.

4. **Region Search:** The configured `ExplanationMethodFn` traverses the `ImageGraph` and returns a `RegionResult`, which contains the optimal found region alongside the search statistics utilized for later evaluation.

6.4 Computational Optimizations

GPU forward passes are the primary computational bottleneck for large batch sizes, taking on the order of milliseconds each. For algorithms relying on stochastic random walks—such as the Potential and UCB policies—graph traversal and sampling overhead can become a secondary bottleneck as the target region size grows. To mitigate this, the framework implements three critical optimizations to minimize the total number of required forward passes:

Batched Evaluation. All model calls are routed through `calculate_region_deltas`, which stacks multiple occluded images into a single tensor, evaluating them in one forward pass. This optimization is applied during both the initial per-segment scoring phase and the rollout evaluations of every search algorithm.

Result Caching and Deduplication. In MCTS and MCGS, a terminal state is evaluated at most once; subsequent visits dynamically reuse the cached reward without triggering a new forward pass. This principle also applies within a single search iteration: the K rollout regions sampled from a non-terminal leaf are deduplicated before GPU evaluation. Each unique region is forwarded exactly once, and the result is broadcast to any duplicate rollouts.

UCB Batching with Virtual Loss. Because a single GPU forward pass incurs roughly the same temporal overhead regardless of batch density, the UCB policy accumulates B arm selections before issuing any GPU call. While a batch is actively being filled, arms that have already been selected receive a virtual increment to their pull count n_v. This conceptually discourages the acquisition function from repeatedly picking the same arm before its true reward is computed [26]. The batch size B is a tunable hyperparameter of the UCB policy; its effect on explanation quality is analyzed in Chapter 7.

6.5 Replacement Image Strategies

Choosing how to fill an occluded area is a non-trivial problem in perturbation-based XAI. While replacing pixels with a constant value is computationally simple, it can inadvertently push inputs far outside the natural training distribution [27]. This forces the model to respond to artificial distribution shifts rather than the actual absence of the masked content.

We implemented three distinct replacement strategies within the framework, shown in Figure 6.2:

- **Solid color:** Fills occluded pixels with a constant value. This can be configured as a user-defined parameter, the image-wide mean color, or a dataset-level mean (e.g., the ImageNet channel mean).
- **Gaussian blur:** Replaces occluded pixels with a heavily blurred version of the original image, preserving local color statistics while effectively destroying fine structural features.
- **Pixel interlacing:** Disrupts spatial structure by swapping alternating rows and columns of the original image data.

While solid colors are straightforward, calculating replacements on a per-segment basis (such as using a segment-wise mean color) can produce severe out-of-distribution artifacts. When a target region covers many segments and each is filled with its own local mean, the image takes on an unnatural pattern that strongly resembles a jigsaw puzzle or a band-aid, as illustrated in Figure 6.3. In such cases, the replacement strategy does not merely occlude content; it introduces a highly salient new pattern. The resulting drop in target-class probability may therefore reflect the classifier's adverse reaction to this artifact rather than the intrinsic importance of the hidden area.

(a) Original (b) Solid color (c) ImageNet mean (d) Gaussian blur (e) Pixel interlacing

Figure 6.2: The three replacement strategies applied to the same image and region (outlined in red). Panels (b) and (c) are both solid-color variants (black color and ImageNet channel mean, respectively). Each strategy produces a different I_{rep}, changing what the classifier sees in the occluded region.

Conversely, a replacement strategy can also fail by being overly cohesive with the original image, effectively camouflaging the occlusion. For example, replacing a black dog with a solid black fill, or applying a Gaussian blur to a background region that is already out of focus, produces an occluded image that is nearly identical to the original. In these instances, the perturbation fails to actually remove the underlying visual feature. The model's confidence will naturally remain unchanged, leading the evaluator to falsely conclude that the region is unimportant, when in reality it was simply never hidden.

(a) Band-aid effect (baby) (b) Jigsaw-puzzle effect (bird) (c) Jigsaw-puzzle effect (penguin)

Figure 6.3: Artifacts produced by hex segmentation with ImageNet mean color replacement at large region sizes. Instead of naturally hiding content, the occluded image resembles a band-aid (a) or a jigsaw puzzle (b, c). The classifier may penalize this unnatural pattern rather than the absence of the original content.

Because no single masking strategy can successfully avoid both out-of-distribution artifacts and accidental camouflage across all domains, the framework explicitly exposes this choice through the `ReplacementFn` interface. This allows any arbitrary function $I_{rep} = f(I)$ to be dynamically injected, including the potential future integration of learned inpainting models [28] capable of producing semantically coherent, distribution-safe fillings.

7 Experiments

This chapter details the empirical evaluation of the proposed framework. We establish optimal algorithm configurations, quantify the stability of the stochastic search policies, internally benchmark the expansion strategies against simpler baselines, and evaluate our selected approach against state-of-the-art XAI methods across multiple datasets.

All code, experiment scripts, and figure generation routines are available at https://github.com/RationAI/ciao on the `test/all-algorithms` branch. The experiments were executed on the e-INFRA CZ cluster; reproducing them requires access to equivalent compute. All hyperparameters, configuration files, and run scripts are provided, though cluster-specific credentials and paths must be adjusted accordingly.

7.1 Experimental Setup

All experiments were executed using an NVIDIA H100 GPU.

7.1.1 Datasets and Models

We utilize the following datasets and models for evaluation:

ImageNet-S [2] is a subset of ImageNet [29] featuring pixel-level segmentation annotations. We evaluate using ResNet50 [30] initialized with ImageNet1K v1 weights.

PatchCamelyon (PCam) [3] is a binary histopathology dataset for lymph node metastasis detection. We evaluate using the pretrained `1aurent/resnet50.tiatoolbox-pcam` model [31].

FunnyBirds [1] is a controlled synthetic benchmark designed for evaluating XAI methods. Each image depicts a procedurally generated bird composed of five swappable parts (beak, eye, wing, tail, and foot) placed over a background with irrelevant distractor objects. Because the 50 bird species are defined by unique combinations of part styles, correct classification depends strictly on part appearance. This provides full ground-truth knowledge of relevant parts, enabling automatic intervention-based evaluation protocols (detailed in Appendix E). We evaluate using the pretrained VGG16 [32] and ViT-B/16 [5] models provided by the dataset authors.

7.1.2 Fixed Parameters

The following parameters are held constant across all experiments:

Segmentation. For ImageNet-S and PCam, we partition each image into a regular hexagonal grid with a circumradius of 4. For FunnyBirds, we use SLIC [7] generating 800 segments with a compactness of 10.0.

Replacement image. Occluded segments are filled using the ImageNet channel mean for ImageNet-S, a Gaussian blur ($\sigma = 5$, kernel 15×15) for PCam, and the designated dataset background color for FunnyBirds.

Target class and evidence direction. For each image, the target class c is fixed as the argmax of the predicted probability on the unoccluded image. We fix the evidence direction to $\sigma = 1$ to search exclusively for positive evidence.

7.1.3 Evaluation Metrics

Throughout this section, the original image I and the replacement image I_{rep} are fixed as described in Section 7.1.2.

Log-odds drop. The direct optimization objective, as formally defined in Section 4.2.

Probability drop. A more interpretable alternative to the log-odds drop, defined as:

$$\Delta p(R) = p_c(I) - p_c(I_R). \tag{7.1}$$

Normalized probability drop. Because raw probability drops naturally vary with baseline confidence and object scale, we normalize the scores within each image. Letting $\Delta p_{\min}$ and $\Delta p_{\max}$ denote the minimum and maximum probability drops observed across all compared methods on a given image, the normalized drop is:

$$\widehat{\Delta p} = \frac{\Delta p - \Delta p_{\min}}{\Delta p_{\max} - \Delta p_{\min}}. \tag{7.2}$$

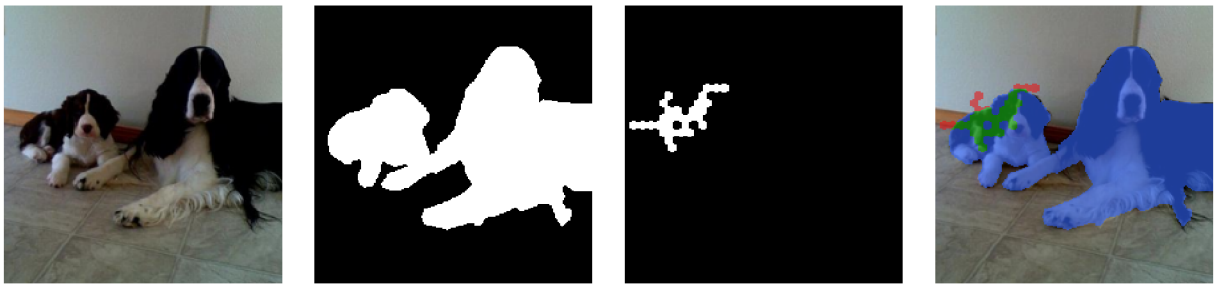
We report the mean of $\widehat{\Delta p}$ averaged across all images.

Mean per-image rank. For each evaluated image, all methods are ranked sequentially by their raw probability drop, with rank 1 assigned to the highest drop. We report the mean rank across all images. By discarding magnitude, this metric remains robust against outlier images where a single method might dominate by an unusually large margin. Lower is better.

Intersection over Union (IoU) with ground-truth segmentation. For datasets containing pixel-level annotations, the IoU is calculated as:

$$\text{IoU} = \frac{|M_R \cap M_{gt}|}{|M_R \cup M_{gt}|}. \tag{7.3}$$

Here, $M_R \in \{0,1\}^{H \times W}$ is the binary mask of the predicted region (Section 4.2), and $M_{gt} \in \{0,1\}^{H \times W}$ is the ground-truth segmentation mask, illustrated in Figure 7.1.



(a) Original image (b) M_{gt} (c) M_R (d) Overlap (IoU = 0.06)

Figure 7.1: IoU between the predicted region mask M_R and the ground-truth segmentation mask M_{gt} on an ImageNet-S example. In panel (d), blue denotes $M_{gt} \setminus M_R$ (GT only), green denotes $M_R \cap M_{gt}$ (intersection), and red denotes $M_R \setminus M_{gt}$ (prediction outside GT).

Note that human-annotated segmentation reflects human perception rather than internal model reasoning. Consequently, this metric does not measure explanatory faithfulness, but acts as a secondary proxy for how well an explanation aligns with semantic boundaries.

Deletion and Insertion AUC. Segments are ranked by their assigned contribution. In the deletion protocol, segments are iteratively removed from the original image (replaced by the baseline) while continuously recording the model probability. In the insertion protocol, the process is reversed: starting from a fully occluded baseline image, segments are iteratively revealed. We report the Area Under the Curve (AUC) for both probability trajectories. A highly faithful explanation yields a lower deletion AUC and a higher insertion AUC.

Because our proposed methods output discrete binary masks lacking internal rank ordering, we follow the protocol established by Fong et al. [33] to generate a continuous saliency map. The search is executed at multiple target sizes m (number of segments). The resulting binary masks are summed together and convolved with a Gaussian filter ($\sigma = 3\%$ of the shorter image dimension), producing the final map evaluated in the AUC metrics. This process is illustrated in Figure 7.2.

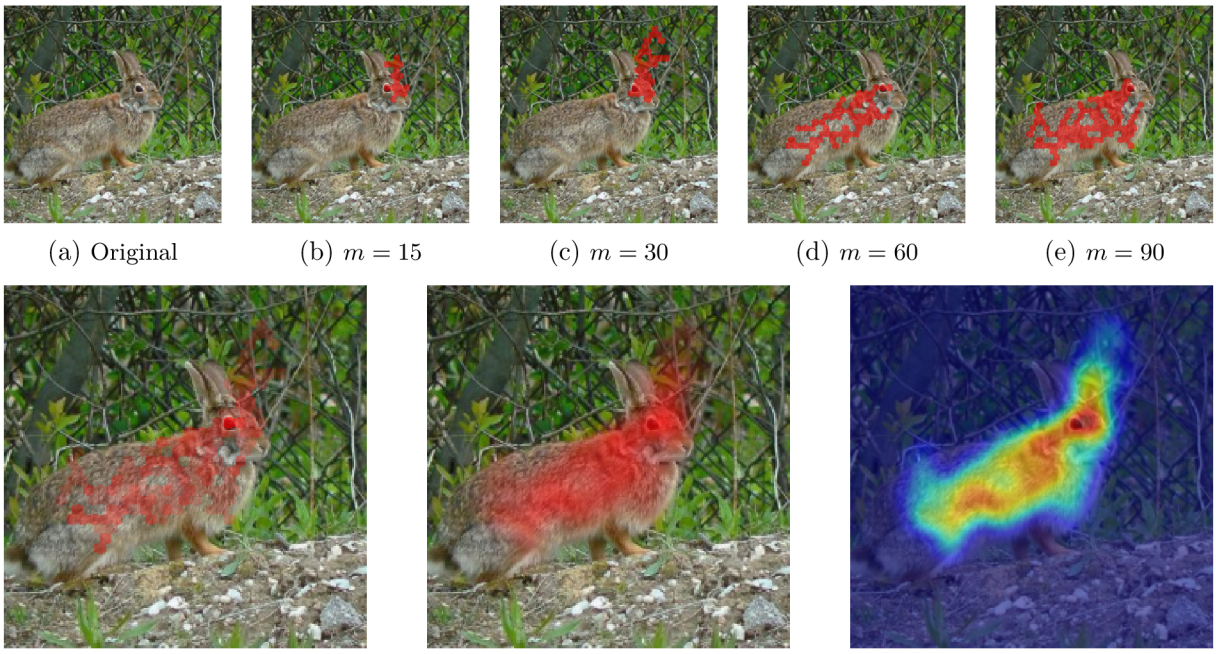
Pointing game. Given a continuous saliency map, the score is 1 if the maximum peak response falls within the ground-truth bounding box of the target object, and 0 otherwise.

The FunnyBirds benchmark employs a dedicated suite of six intervention-based evaluation metrics—Controlled Synthetic Data Check (CSDC), Preservation Check (PC), Deletion Check (DC), Distractibility (D), Single Deletion (SD), and Target Sensitivity (TS)—whose formal definitions are provided in Appendix E.

7.2 Baselines and Competitors

7.2.1 Baselines

The following internal baseline methods are implemented natively within our framework. Because they utilize identical segmentation, replacement, and seed selection logic as our proposed algorithms, they serve to isolate the specific impact of the expansion strategy.



(a) Original (b) $m = 15$ (c) $m = 30$ (d) $m = 60$ (e) $m = 90$

(f) Sum of masks (g) After Gaussian blur (h) Saliency map

Figure 7.2: Construction of a saliency map from multiple binary region masks. The search algorithm executes across several target sizes m (number of segments, top row); the resulting masks are summed (f) and convolved with a Gaussian filter (g) to produce a continuous heatmap (h).

Pure Monte Carlo evaluates the problem purely stochastically. It generates random connected regions of the target size by executing unguided random walks from the starting seed, ultimately returning the highest-scoring region discovered within the allocated evaluation budget.

Greedy executes the greedy policy defined in Section 5.3 with a lookahead depth of $d = 1$, permanently committing to the immediate highest-scoring frontier node at each sequential step.

Beam search on a saliency map operates without further model evaluations by maintaining a beam of b partial regions. When expanding from size k to $k+1$, all possible one-segment extensions of every tracked region are scored using the static, pre-computed 1-hop heatmap $\bar{g}_\sigma$ generated during seed selection (Section 5.2). Only the top b regions are retained for the next step, approximating a search for the highest cumulative static score.

7.2.2 Competitors

We evaluate against three external state-of-the-art XAI methods: LIME [15], Occlusion Sensitivity [16], and Grad-CAM++ [12]. A theoretical overview of these approaches is provided in Chapter 3. Unlike our proposed framework, these external approaches are not constrained to produce spatially cohesive regions. Meaningful Perturbations [14] was initially tested but ultimately excluded from formal benchmarking due to severe optimization artifacts observed in the resulting masks; a detailed analysis of this exclusion is provided in Appendix C.

7.3 Hyperparameter Optimization

The MCGS, MCTS, and UCB algorithms expose three primary hyperparameters: the exploration constant c, the value mixing weight α, and the rollout/batch size. To determine optimal configurations, we performed a grid search on 30 randomly sampled ImageNet-S images with parameters $m = 30$ segments, `numRegions` = 1, and a fixed wall-clock budget of approximately 30 seconds per image. The full optimization data is reported in Appendix B.

Based on the grid search results, we exclude MCTS from further experiments due to its empirical similarity to MCGS. For MCGS, we adopt $c = 1.4$, $\alpha = 0$, `numRollouts` = 64, and an evaluation budget of 100,000. For UCB, we adopt $c = 1.4$, $\alpha = 0.5$, `UCBBatchSize` = 64, and a dynamic step budget of $\lfloor 100{,}000/m \rfloor$, normalizing the total target evaluations to exactly 100,000 to match MCGS. These configurations remain fixed for all subsequent experiments across all target sizes, explicitly prioritizing a fair, uniform comparison over per-size optimization.

7.4 Stability of Stochastic Algorithms

Because MCGS and UCB rely on stochastic random walks, distinct random initializations yield slightly varying output regions even under identical inputs. To quantify this variance, we executed both algorithms 10 times per image on the same 30 ImageNet-S samples utilized in Section 7.3, maintaining the fixed hyperparameter configurations. The empirical stability results are summarized in Table 7.1.

We report two aggregated stability metrics. **Self-IoU** is the mean pairwise IoU calculated over all $\binom{10}{2} = 45$ run pairs per image, measuring the spatial consistency of the algorithm. **Score CV** is the coefficient of variation (σ/μ) of the log-odds drop across the 10 runs, measuring the consistency of the optimization capability.

Table 7.1: Stability of MCGS and UCB on 30 ImageNet-S images, 10 runs each, across three target sizes m (number of segments).

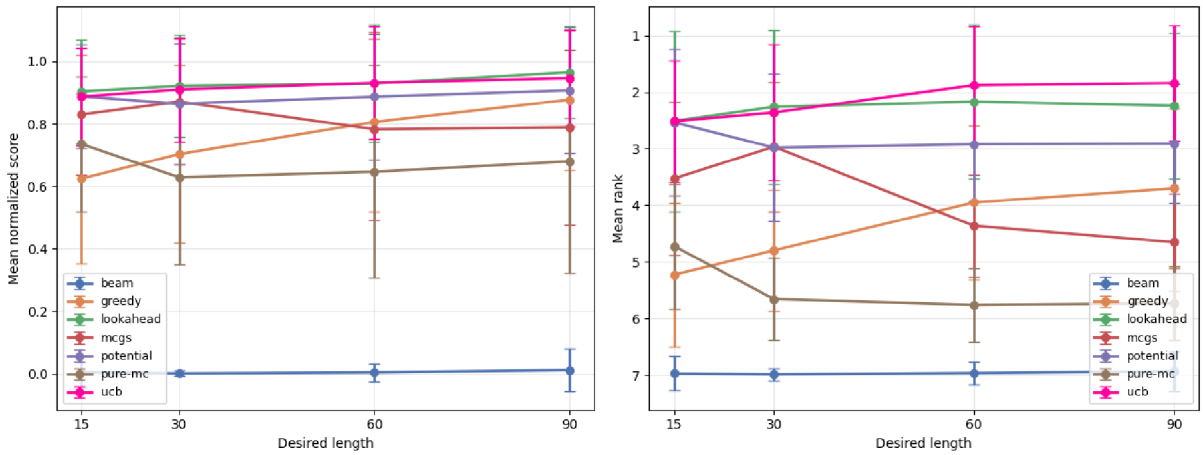
Algorithm	Metric	$m = 15$	$m = 30$	$m = 90$
MCGS	Self-IoU	0.52 ± 0.14	0.49 ± 0.10	0.37 ± 0.05
	CV(log-odds drop)	0.05 ± 0.04	0.07 ± 0.04	0.06 ± 0.06
	CV(prob. drop)	0.07 ± 0.09	0.07 ± 0.10	0.07 ± 0.15
UCB	Self-IoU	0.55 ± 0.13	0.48 ± 0.09	0.35 ± 0.08
	CV(log-odds drop)	0.05 ± 0.03	0.07 ± 0.04	0.10 ± 0.08
	CV(prob. drop)	0.06 ± 0.08	0.08 ± 0.11	0.05 ± 0.16

Both algorithms maintain a Score CV below 10% across nearly all evaluated conditions, demonstrating that the ultimate optimization objective is reached consistently. As expected, Self-IoU decreases as target size m increases, reflecting the exponentially larger valid search space available for large regions. Consequently, run-to-run variance is treated as negligible, and subsequent comparisons in Section 7.5 rely on single-run evaluations.

7.5 Internal Method Comparison

To evaluate the relative efficacy of the proposed expansion strategies, we compare all internal search algorithms against the baselines defined in Section 7.2. This internal evaluation is conducted on 100 randomly sampled ImageNet-S images across four predefined target sizes (number of segments): $m \in \{15, 30, 60, 90\}$. External competitors are omitted from this phase to strictly isolate search algorithm performance.

Table 7.2 reports the mean rank metric, while Figure 7.3 plots both the mean rank and the normalized objective score as a function of the target size. Complete numerical results, including raw probability drops and wall-clock execution times, are provided in Table D.1.



(a) Normalized score

(b) Mean rank

Figure 7.3: Performance of all compared algorithms as a function of target region size m on 100 ImageNet-S images. Lower mean rank is better; higher normalized score is better.

Table 7.2: Mean per-image rank (↓, mean ± std) on 100 ImageNet-S images. Lower is better. Full results in Table D.1.

Algorithm	$m = 15$	$m = 30$	$m = 60$	$m = 90$
Pure Monte Carlo	4.72 ± 1.10	5.66 ± 0.72	5.76 ± 0.65	5.73 ± 0.65
Beam search (saliency)	6.97 ± 0.30	6.98 ± 0.11	6.96 ± 0.20	6.93 ± 0.36
Greedy	5.22 ± 1.27	4.80 ± 1.07	3.95 ± 1.37	3.70 ± 1.40
Lookahead	$\mathbf{2.51 \pm 1.59}$	$\mathbf{2.26 \pm 1.36}$	2.17 ± 1.36	2.24 ± 1.29
Potential	2.54 ± 1.30	2.98 ± 1.30	2.92 ± 1.01	2.91 ± 1.05
MCGS	3.52 ± 1.35	2.96 ± 1.14	4.36 ± 0.89	4.65 ± 0.86
UCB	2.52 ± 1.07	2.36 ± 1.20	$\mathbf{1.88 \pm 1.04}$	$\mathbf{1.84 \pm 1.01}$

7.6 State-of-the-Art Comparison

Based on the internal rank performance observed in Section 7.5, UCB was selected as the primary representative algorithm for the external evaluation phase. We benchmark UCB against the competitors identified in Section 7.2 using 100 ImageNet-S images alongside

a balanced PCam subset containing 50 positive (tumorous) and 50 negative patches. The IoU metric is evaluated at a static region size of $m = 90$. The exact configurations for the external competitors are documented in Appendix C.

Quantitative benchmarking results are reported in Tables 7.3 and 7.4, while visual qualitative comparisons are provided in Figures 7.4 and 7.5. We additionally report performance on the FunnyBirds dataset, utilizing its specific intervention-based protocols in Tables 7.5 and 7.6, with qualitative examples displayed in Figure 7.6.

Table 7.3: Comparison across methods on ImageNet-S (100 images). Ground-truth annotations enable IoU and pointing game. Deletion AUC is lower-is-better; all other metrics are higher-is-better.

Method	Del. AUC ↓	Ins. AUC ↑	IoU ↑	Pointing game[‡] ↑	Time (s) ↓
Ours (UCB)	0.22 ± 0.22	0.50 ± 0.29	0.11 ± 0.11	40/99	$49.92 \pm 1.55^{\dagger}$
LIME	$\mathbf{0.10 \pm 0.09}$	0.49 ± 0.31	0.10 ± 0.09	46/99	4.93 ± 0.17
Occlusion Sensitivity	0.21 ± 0.23	0.43 ± 0.24	0.08 ± 0.09	40/99	2.30 ± 0.09
Grad-CAM++	0.17 ± 0.16	$\mathbf{0.56 \pm 0.27}$	$\mathbf{0.13 \pm 0.13}$	**49/99**	$\mathbf{0.24 \pm 0.23}$

[†] Mean over target sizes m; runtime increases with m, likely due to Python `frozenset` overhead.
[‡] One image was excluded: the classifier's top-1 prediction on the unoccluded image was incorrect, so no ground-truth bounding box for the predicted class exists.

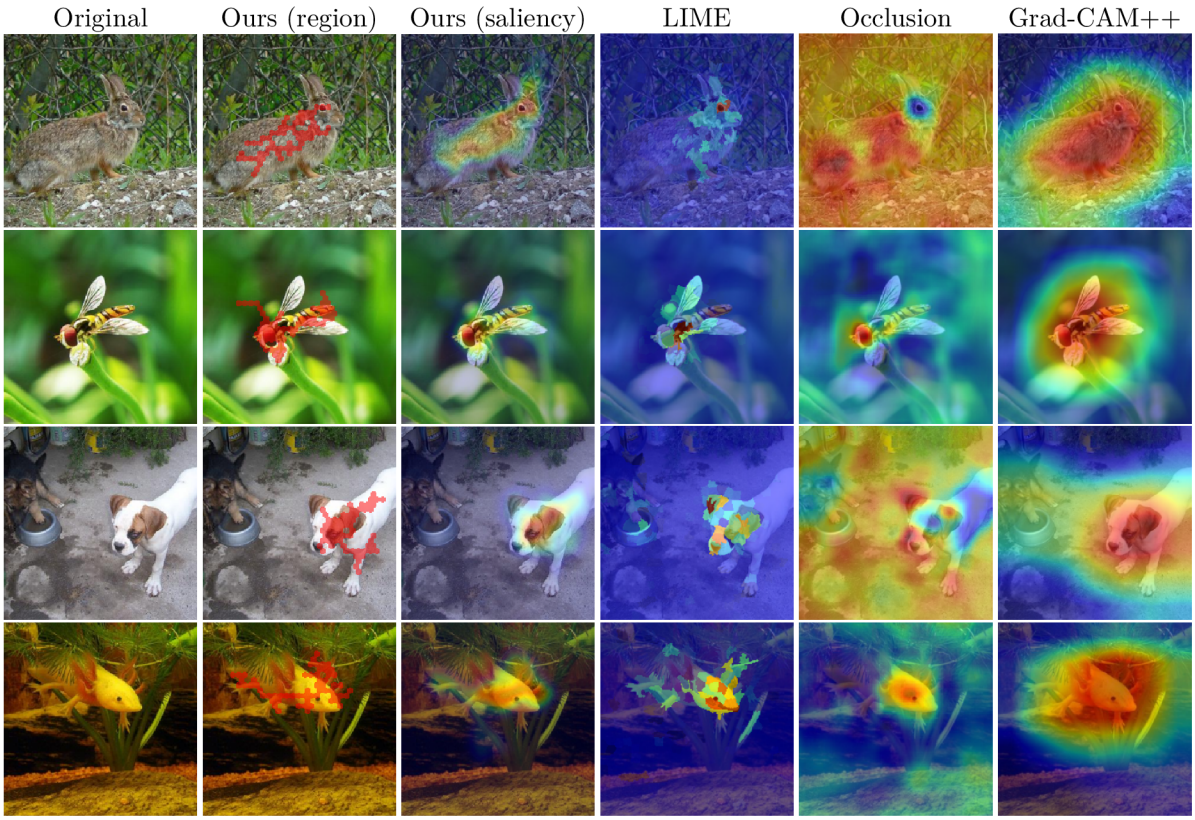


Figure 7.4: Qualitative explanation comparison on ImageNet-S. Each row shows one image; columns correspond to different methods. "Ours (region)" is the binary region mask at $m = 60$ segments; "Ours (saliency)" is the aggregated saliency map.

For FunnyBirds, we utilize UCB with `stepBudget` $= 500$ and $m = 12$ segments, extracting up to 5 disjoint regions. A part is classified as important if its aggregate log-odds

Table 7.4: Comparison across methods on PCam (50 positive + 50 negative patches). Deletion AUC is lower-is-better; Insertion AUC is higher-is-better.

Method	Del. AUC ↓	Ins. AUC ↑
Ours (UCB)	0.58 ± 0.33	0.88 ± 0.17
LIME	$\mathbf{0.44 \pm 0.32}$	$\mathbf{0.92 \pm 0.15}$
Occlusion Sensitivity	0.56 ± 0.33	0.85 ± 0.22
Grad-CAM++	0.69 ± 0.32	0.88 ± 0.17

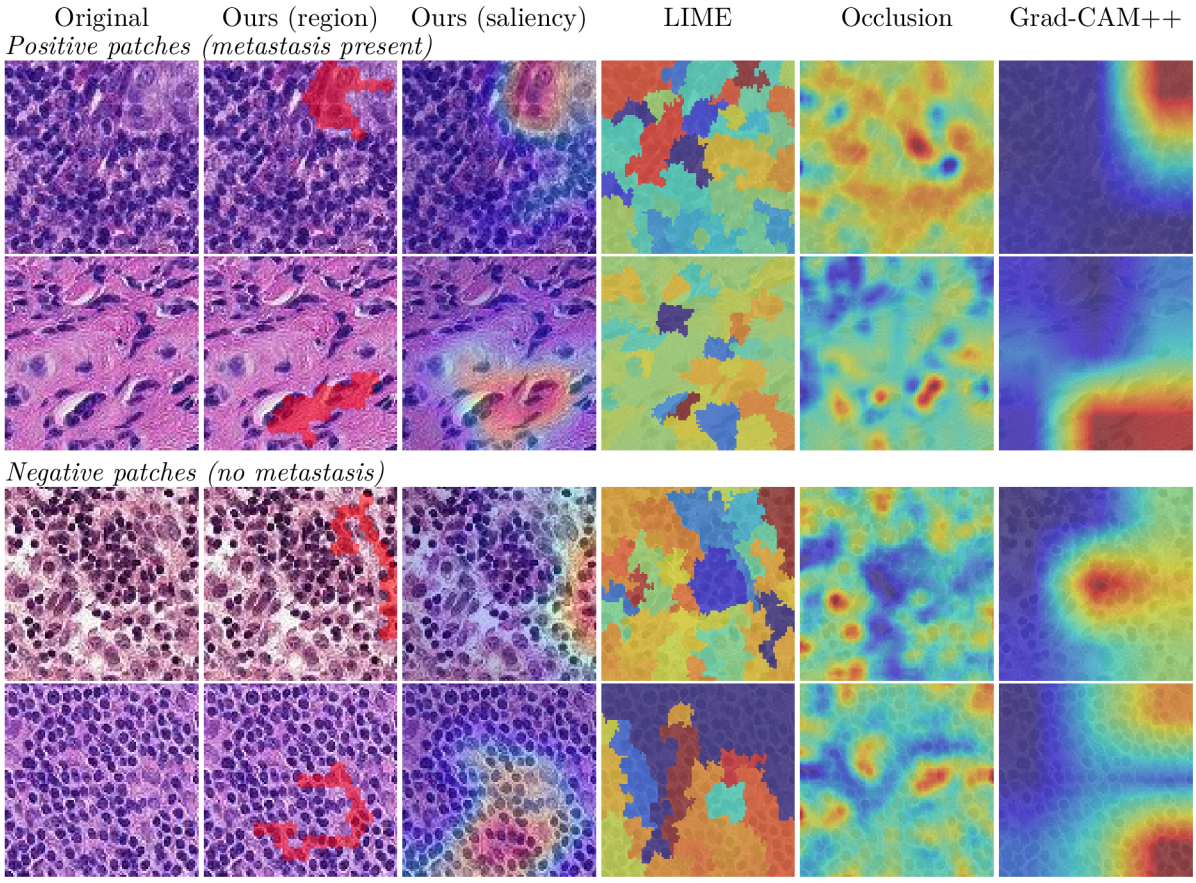


Figure 7.5: Qualitative explanation comparison on PCam patches. Positive patches (top, metastasis present) and negative patches (bottom, no metastasis) are shown separately. "Ours (region)" is the binary region mask at $m = 20$ segments; "Ours (saliency)" is the aggregated saliency map.

drop exceeds threshold τ. We tune τ to maximize CSDC ($\tau = 0.01$ for VGG16, $\tau = 0.016$ for ViT-B/16).

We evaluate UCB natively on random samples (500 for VGG16, 150 for ViT). Reference results for LIME [15], Grad-CAM [13], Integrated Gradients [34], RISE [35], Input×Gradient [36], and Chefer LRP [37] are sourced directly from Hesse et al. [1] on the full test set.

Table 7.5: FunnyBirds evaluation on VGG16. Only our method was evaluated natively. All other results are quoted from Hesse et al. [1]. LIME does not produce a part ranking, so its SD and TS are N/A. Bold marks the best per column among reference methods.

Method	CSDC	PC	DC	D	SD	TS
Ours (UCB)	0.93	0.93	0.92	0.90	0.71	0.83
Grad-CAM	**0.94**	**0.97**	**0.93**	0.87	**0.75**	**0.93**
Integ. Grad.	0.92	0.92	0.92	**0.97**	0.67	0.92
RISE	0.80	0.73	0.70	0.84	0.73	0.83
LIME	0.89	0.88	0.90	0.92	N/A	N/A
IxG	0.79	0.71	0.69	0.94	0.55	0.69

Table 7.6: FunnyBirds evaluation on ViT-B/16. Only our method was evaluated natively. All other results are quoted from Hesse et al. [1]. LIME does not produce a part ranking, so its SD and TS are N/A. Bold marks the best per column among reference methods.

Method	CSDC	PC	DC	D	SD	TS
Ours (UCB)	0.94	0.92	0.91	0.90	0.74	0.87
Grad-CAM	0.75	0.67	0.68	**0.91**	0.70	0.48
Integ. Grad.	0.89	0.86	0.85	0.90	0.65	0.91
RISE	0.79	0.71	0.70	0.83	**0.79**	0.75
LIME	**0.95**	**0.96**	**0.96**	0.85	N/A	N/A
IxG	0.74	0.59	0.60	0.43	0.51	0.67
Chefer LRP	0.91	0.92	0.89	0.90	0.74	**0.95**

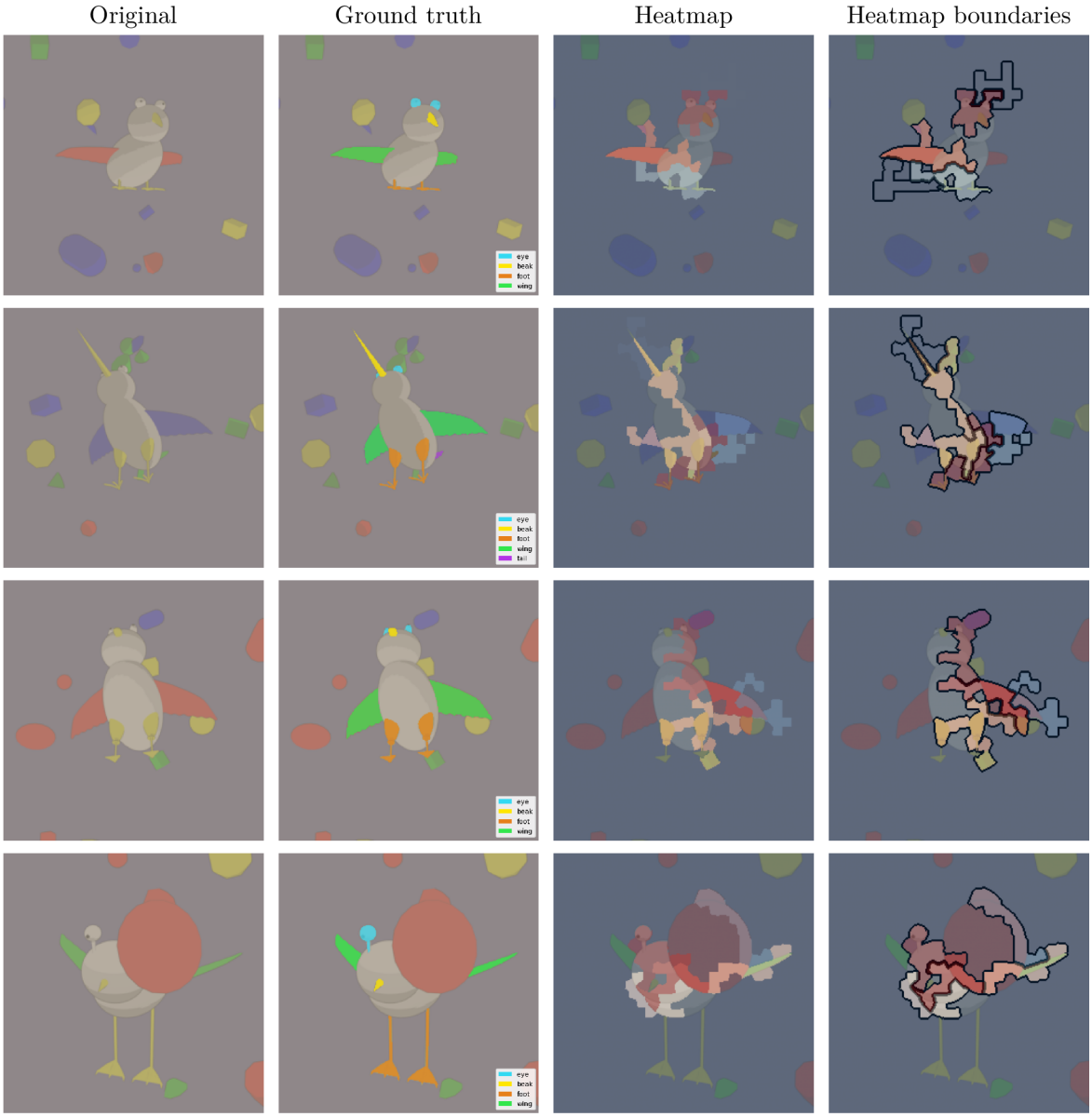


Figure 7.6: Qualitative FunnyBirds examples. Each row shows one bird image; columns show the original image, the ground-truth part annotation, the predicted occlusion heatmap, and the heatmap with region boundaries overlaid.

8 Discussion

8.1 Algorithmic Trade-offs

Lookahead, Potential, and UCB all substantially outperform the random-walk and saliency-guided beam search baselines (Table 7.2). MCGS is competitive at small target sizes but is matched or beaten by the greedy baseline at $m \geq 60$, which confirms that complexity alone does not guarantee better optimization.

Lookahead achieves the highest mean normalized score, but its depth parameter d does not map cleanly to a fixed compute budget—cost grows combinatorially with d, so we had to retune d for each target size to keep runtimes comparable. UCB, conversely, exposes a budget T directly and consistently achieves the best mean rank across almost all target sizes, making it the most practical choice when both quality and budget control matter.

MCGS did not outperform the simpler stepwise methods. We believe the main reason is that standard MCGS enhancements that make the framework effective in practice—Rapid Action Value Estimation (RAVE)/All Moves As First (AMAF) heuristics [38] for faster value propagation and parallel search with virtual loss [26] to avoid redundant exploration—were not implemented in our version. These are natural directions for a stronger MCGS variant. A contributing factor is that MCGS exposes more hyperparameters than the stepwise methods, making it harder to identify an optimal configuration: a fixed grid-search budget covers a smaller fraction of the joint space of c, α, and rollout count.

8.2 Stability of Stochastic Algorithms

MCGS and UCB are stochastic, and the stability experiment in Section 7.4 shows that optimization quality is consistent: score CV stays below 10% in almost all conditions (Table 7.1).

Self-IoU is more nuanced. At $m = 15$, a value of roughly 0.5 means two independent runs share about half of the selected segments on average—a level of agreement that is reasonable given the search space size, but not negligible variation for an XAI method. If a user runs the same algorithm twice on the same image, they may receive noticeably different regions even though both achieve similar scores. The score is stable across runs, but the specific spatial layout is not.

8.3 Comparison with State of the Art

ImageNet-S. Comparing discrete regions against continuous baselines requires the aggregation protocol from Section 7.1.3. Among black-box methods (Table 7.3), our Insertion AUC (0.50) is the highest, edging LIME (0.49) and clearly above Occlusion (0.43). IoU at $m = 90$ (0.11) is comparable to LIME (0.10) and above Occlusion (0.08). Deletion AUC (0.22) and pointing game (40/99) are weaker than LIME (0.10, 46/99)—both rely on the aggregated saliency approximation rather than the native binary output, so the gap can partly reflect the protocol rather than the underlying search quality. Grad-CAM++ leads on Insertion AUC, IoU, and pointing game, but requires gradient access. The structural property that distinguishes our output from all competitors is spatial connectedness: our search enforces a single connected region by construction, whereas LIME and Occlusion produce spatially scattered attributions with no such guarantee.

Qualitatively (Figure 7.4), single-region output is visually clean when the object is compact, but breaks down when no small subregion covers the discriminative area—representative failure cases are shown in Section D.1. We also tested extracting multiple regions per image, but the results were inconsistent: the individual regions often merged into a single big region, and the quantitative metrics showed no improvement over single-region search.

PatchCamelyon (PCam). PCam was included to test the framework on a different domain and image resolution (histopathology vs. natural images). Results (Table 7.4) are broadly comparable to other black-box methods, though LIME leads on both Deletion and Insertion AUC. Interpreting the explanation is harder here than on natural images: there is no salient foreground object, only a subtle textural signal. Hiding part of a histology patch with a mean-color fill has no clear semantic meaning—it is not obvious that an occluded area corresponds to tissue that is "more tumorous." A rigorous evaluation would require expert pathologist review and higher-resolution inputs; these results confirm the framework generalizes beyond natural images but should not be read as a clinical claim.

FunnyBirds. On VGG16 (Table 7.5), our method is the top-performing black-box approach across almost all metrics and competitive with white-box methods. On ViT-B/16 (Table 7.6), completeness and correctness scores (CSDC, PC, DC) are strong, confirming the regions capture the model-relevant features. The spatial deletion score (SD) is slightly lower than the best competitor on both architectures, likely because a single rigid shape sometimes fails to isolate individual procedurally generated bird parts cleanly. Regions occasionally bleed into the background, though this is observed across all other evaluated methods as well. A contributing factor is that $m = 12$ may be too large for this dataset: a single region sometimes spans two adjacent bird parts at once, leaving the remaining regions with negligible optimization scores. Per-dataset tuning of the region size would likely improve part localization. Despite these issues, the quantitative results remain competitive across methods.

8.4 Limitations

Fixed region size. Requiring the region to target m segments (and reaching at most m when the frontier is exhausted) is a coarse constraint. An image of a wide scene may need a large region to capture the discriminative area, while a close-up of a small object warrants a much smaller one. The user must choose m without a principled criterion, and the wrong choice can either truncate the explanation or dilute it with uninformative segments.

Fixed number of regions. The framework grows a single connected region per seed. When multiple seeds are used, the number of output regions is a second hyperparameter that must be set a priori. This choice is equally coarse: too few regions may miss discriminative areas that a single connected component cannot cover, while too many can flood the explanation with redundant content. No principled stopping criterion currently determines when enough regions have been found.

Computational cost. UCB is slower than the state-of-the-art methods (Table 7.3). We fixed a search budget of approximately 30 seconds per target size to prioritize optimization quality over speed and did not investigate smaller budgets. A lighter configuration would reduce runtime substantially, though likely at some cost to score quality; this trade-off remains unexplored.

Replacement artifacts. As demonstrated in Section 6.5, perturbation-based evaluation is inherently sensitive to how occluded pixels are filled. Simple replacements (such as solid colors or segment-wise means) can introduce severe out-of-distribution artifacts, causing the model to penalize the artificial pattern rather than reacting to the missing feature. While our framework supports custom replacement functions, the fundamental XAI problem of finding a universally neutral baseline remains unresolved.

9 Conclusion

9.1 Summary of Contributions

We identified a structural weakness in existing black-box XAI methods: their explanations consist of disconnected patches or scattered pixel attributions that are difficult to interpret as a single coherent reason for a prediction. To address this, we formalized the explanation problem as a combinatorial search over a region adjacency graph, where the output is required to be a connected induced subgraph of fixed size. This constraint is enforced throughout the search rather than applied as a post-processing step.

On top of this formulation, we built an extensible open-source Python library implementing five search strategies—Lookahead, Potential, MCTS, MCGS, and UCB. MCTS was excluded from the final evaluation due to its empirical similarity to MCGS (Section 7.3). The framework supports growing multiple regions and is designed so that new search algorithms can be added by implementing a single interface, making it straightforward to experiment with alternative strategies.

The method is competitive with the state of the art across all three evaluated datasets: it reaches the highest Insertion AUC among black-box competitors on ImageNet-S, is the top-performing black-box approach on FunnyBirds (VGG16), and is broadly comparable to other black-box methods on PCam. All of this holds under the constraint that the output is always a single connected region—something none of the evaluated competitors enforce natively.

9.2 Future Directions

9.2.1 Algorithmic Enhancements

The fixed target size m is the most restrictive hyperparameter of the current framework. A natural extension is adaptive termination: the search stops when the log-odds drop falls below a user-defined confidence threshold, letting the region size emerge from the data rather than being imposed externally. A related direction is adaptive budget allocation—for example, the current uniform split of $\lfloor T/m \rfloor$ evaluations per step could be replaced by a schedule that concentrates budget on early expansions where marginal gain is highest.

Multi-seed initialization is another straightforward direction. Instead of starting from a single seed segment, the search can be launched simultaneously from a grid of candidate seeds, each growing an independent region. The top-k results by optimization score—

possibly overlapping—can then be returned as a ranked set of explanations, giving the user multiple perspectives on the prediction.

9.2.2 Graph and Framework Extensions

MCGS currently uses a standard UCB backpropagation rule. Integrating RAVE/AMAF heuristics, which amortize value estimates across sibling nodes in the transposition graph, could close the gap with local stepwise methods. Parallel tree search with virtual loss is a complementary direction that prevents redundant exploration across concurrent threads.

The pure black-box assumption can also be relaxed. If the classifier is assumed to be Lipschitz-continuous, one can bound the change in output over small perturbations and prune branches of the search tree without evaluating them, enabling algorithms with theoretical convergence guarantees.

9.2.3 Application Domain

The most immediate practical limitation is the out-of-distribution masking artifact. Replacing the masked region with a generative inpainting result—rather than a constant fill color—would produce a replacement image I_{rep} that is plausible to the classifier, making the faithfulness score more reliable.

On the dataset side, PCam patches are a constrained testbed: the images are small and the classification is binary. Applying the framework to whole-slide images in digital pathology, where tissue sections can span gigapixels, would require efficient hierarchical segmentation and graph construction, but the graph-search formulation is well suited to such spatially structured data.

Bibliography

1. HESSE, Robin; SCHAUB-MEYER, Simone; ROTH, Stefan. FunnyBirds: A Synthetic Vision Dataset for a Part-Based Analysis of Explainable AI Methods. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023. Available also from: https://arxiv.org/abs/2308.06248.

2. GAO, Shanghua; LI, Zhong-Yu; YANG, Ming-Hsuan; CHENG, Ming-Ming; HAN, Junwei; TORR, Philip. Large-Scale Unsupervised Semantic Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* 2023, vol. 45, no. 6, pp. 7457–7476. ISSN 0162-8828, ISSN 2160-9292, ISSN 1939-3539. Available from DOI: 10.1109/TPAMI.2022.3218275.

3. VEELING, Bastiaan S.; LINMANS, Jasper; WINKENS, Jim; COHEN, Taco; WELLING, Max. Rotation Equivariant CNNs for Digital Pathology. In: *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*. Springer, 2018. Available from DOI: 10.1007/978-3-030-00934-2_24.

4. KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks. In: *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2012, vol. 25. Available also from: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

5. DOSOVITSKIY, Alexey; BEYER, Lucas; KOLESNIKOV, Alexander; WEISSENBORN, Dirk; ZHAI, Xiaohua; UNTERTHINER, Thomas; DEHGHANI, Mostafa; MINDERER, Matthias; HEIGOLD, Georg; GELLY, Sylvain; USZKOREIT, Jakob; HOULSBY, Neil. An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale. In: *International Conference on Learning Representations*. 2021. Available also from: https://openreview.net/forum?id=YicbFdNTTy.

6. GUO, Chuan; PLEISS, Geoff; SUN, Yu; WEINBERGER, Kilian Q. On Calibration of Modern Neural Networks. In: *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2017. Available also from: https://arxiv.org/abs/1706.04599.

7. ACHANTA, Radhakrishna; SHAJI, Appu; SMITH, Kevin; LUCCHI, Aurelien; FUA, Pascal; SÜSSTRUNK, Sabine. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2012, vol. 34, no. 11, pp. 2274–2282. ISSN 1939-3539. Available from DOI: 10.1109/TPAMI.2012.120.

8. AUER, Peter; CESA-BIANCHI, Nicolò; FISCHER, Paul. Finite-Time Analysis of the Multiarmed Bandit Problem. *Machine Learning*. 2002, vol. 47, no. 2, pp. 235–256. ISSN 1573-0565. Available from DOI: 10.1023/A:1013689704352.

9. KOCSIS, Levente; SZEPESVÁRI, Csaba. Bandit Based Monte-Carlo Planning. In: FÜRNKRANZ, Johannes; SCHEFFER, Tobias; SPILIOPOULOU, Myra (eds.). *Machine Learning: ECML 2006*. Berlin, Heidelberg: Springer, 2006, pp. 282–293. ISBN 978-3-540-46056-5. Available from DOI: 10.1007/11871842_29.

10. SCHADD, Maarten; WINANDS, Mark; HERIK, H.; CHASLOT, Guillaume; UITERWIJK, Jos. Single-Player Monte-Carlo Tree Search. In: *Computers and Games*. Berlin, Heidelberg: Springer, 2008, pp. 1–12. ISBN 978-3-540-87607-6. Available from DOI: 10.1007/978-3-540-87608-3_1.

11. CZECH, Johannes; KORUS, Patrick; KERSTING, Kristian. *Monte-Carlo Graph Search for AlphaZero*. arXiv, 2020. No. arXiv:2012.11045. Available from DOI: 10.48550/arXiv.2012.11045. Accessed: 2026-03-12.

12. CHATTOPADHYAY, Aditya; SARKAR, Anirban; HOWLADER, Prantik; BALASUBRAMANIAN, Vineeth N. Grad-CAM++: Improved Visual Explanations for Deep Convolutional Networks. In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2018, pp. 839–847. Available from DOI: 10.1109/WACV.2018.00097.

13. SELVARAJU, Ramprasaath R.; COGSWELL, Michael; DAS, Abhishek; VEDANTAM, Ramakrishna; PARIKH, Devi; BATRA, Dhruv. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. *Int J Comput Vis*. 2020, vol. 128, no. 2, pp. 336–359. ISSN 0920-5691, ISSN 1573-1405. Available from DOI: 10.1007/s11263-019-01228-7.

14. FONG, Ruth; VEDALDI, Andrea. Interpretable Explanations of Black Boxes by Meaningful Perturbation. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 3449–3457. Available from DOI: 10.1109/ICCV.2017.371.

15. RIBEIRO, Marco Tulio; SINGH, Sameer; GUESTRIN, Carlos. "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*. 2016. Available from DOI: 10.1145/2939672.2939778.

16. ZEILER, Matthew D.; FERGUS, Rob. Visualizing and Understanding Convolutional Networks. In: *Computer Vision – ECCV 2014*. Springer, 2014. Available from DOI: 10.1007/978-3-319-10590-1_53.

17. BUI, Ngoc; NGUYEN, Hieu Trung; NGUYEN, Viet Anh; YING, Rex. Explaining Graph Neural Networks via Structure-aware Interaction Index. In: *Proceedings of the 41st International Conference on Machine Learning*. PMLR, 2024, vol. 235, pp. 329–342. Available from DOI: 10.5555/3692070.3692263.

18. YUAN, Hao; YU, Haiyang; WANG, Jie; LI, Kang; JI, Shuiwang. On Explainability of Graph Neural Networks via Subgraph Explorations. In: *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 2021. Available also from: https://arxiv.org/abs/2102.05152.

19. SEIFY, Arta; BURO, Michael. *Single-Agent Optimization Through Policy Iteration Using Monte-Carlo Tree Search*. arXiv, 2020. No. arXiv:2005.11335. Available from DOI: 10.48550/arXiv.2005.11335. Accessed: 2026-04-13.

20. WU, David J. *Monte-Carlo Graph Search from First Principles*. GitHub, 2024. Available also from: https://github.com/lightvector/KataGo/blob/master/docs/GraphSearch.md. Accessed: 2026-05-06.

21. PASZKE, Adam; GROSS, Sam; MASSA, Francisco; LERER, Adam; BRADBURY, James; CHANAN, Gregory; KILLEEN, Trevor; LIN, Zeming; GIMELSHEIN, Natalia; ANTIGA, Luca; DESMAISON, Alban; KOPF, Andreas; YANG, Edward; DEVITO, Zachary; RAISON, Martin; TEJANI, Alykhan; CHILAMKURTHY, Sasank; STEINER, Benoit; FANG, Lu; BAI, Junjie; CHINTALA, Soumith. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019, vol. 32.

22. WIGHTMAN, Ross. *PyTorch Image Models*. 2026. Available from DOI: 10.5281/zenodo.4414861. Accessed: 2026-05-06.

23. VAN DER WALT, Stéfan; SCHÖNBERGER, Johannes L.; NUNEZ-IGLESIAS, Juan; BOULOGNE, François; WARNER, Joshua D.; YAGER, Neil; GOUILLART, Emmanuelle; YU, Tony; SCIKIT-IMAGE CONTRIBUTORS. Scikit-Image: Image Processing in Python. *PeerJ*. 2014, vol. 2, e453. ISSN 2167-8359. Available from DOI: 10.7717/peerj.453.

24. YADAN, Omry. *Hydra - A framework for elegantly configuring complex applications* [Github]. 2019. Available also from: https://github.com/facebookresearch/hydra. Accessed: 2026-05-06.

25. ZAHARIA, Matei A.; CHEN, Andrew; DAVIDSON, Aaron; GHODSI, Ali; HONG, Sue Ann; KONWINSKI, Andy; MURCHING, Siddharth; NYKODYM, Tomas; OGILVIE, Paul; PARKHE, Mani; XIE, Fen; ZUMAR, Corey. Accelerating the Machine Learning Lifecycle with MLflow. *IEEE Data Eng. Bull.* 2018, vol. 41, pp. 39–45.

26. CHASLOT, Guillaume M. J. -B.; WINANDS, Mark H. M.; VAN DEN HERIK, H. Jaap. Parallel Monte-Carlo Tree Search. In: VAN DEN HERIK, H. Jaap; XU, Xinhe; MA, Zongmin; WINANDS, Mark H. M. (eds.). *Computers and Games*. Berlin, Heidelberg: Springer, 2008, pp. 60–71. ISBN 978-3-540-87608-3. Available from DOI: 10.1007/978-3-540-87608-3_6.

27. HASE, Peter; XIE, Harry; BANSAL, Mohit. The Out-of-Distribution Problem in Explainability and Search Methods for Feature Importance Explanations. In: BEYGELZIMER, A.; DAUPHIN, Y.; LIANG, P.; VAUGHAN, J. Wortman (eds.). *Advances in Neural Information Processing Systems*. 2021. Available also from: https://openreview.net/forum?id=HCrp4pdk2i.

28. AGARWAL, Chirag; NGUYEN, Anh. Explaining Image Classifiers by Removing Input Features Using Generative Models. In: ISHIKAWA, Hiroshi; LIU, Cheng-Lin; PAJDLA, Tomas; SHI, Jianbo (eds.). *Computer Vision – ACCV 2020*. Cham: Springer International Publishing, 2021, pp. 101–118. ISBN 978-3-030-69544-6. Available from DOI: 10.1007/978-3-030-69544-6_7.

29. DENG, Jia; DONG, Wei; SOCHER, Richard; LI, Li-Jia; LI, Kai; FEI-FEI, Li. ImageNet: A Large-Scale Hierarchical Image Database. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. ISSN 1063-6919. Available from DOI: 10.1109/CVPR.2009.5206848.

30. HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. Deep Residual Learning for Image Recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016. Available from DOI: 10.1109/CVPR.2016.90.

31. POCOCK, Johnathan; GRAHAM, Simon; VU, Quoc Dang; JAHANIFAR, Mostafa; DESHPANDE, Srijay; HADJIGEORGHIOU, Giorgos; SHEPHARD, Adam; BASHIR, Raja Muhammad Saad; BILAL, Mohsin; LU, Wenqi; EPSTEIN, David; MINHAS, Fayyaz; RAJPOOT, Nasir M.; RAZA, Shan E. Ahmed. TIAToolbox as an End-to-End Library for Advanced Tissue Image Analytics. *Commun Med*. 2022, vol. 2, no. 1, p. 120. ISSN 2730-664X. Available from DOI: 10.1038/s43856-022-00186-5.

32. SIMONYAN, Karen; ZISSERMAN, Andrew. Very Deep Convolutional Networks for Large-Scale Image Recognition. In: *International Conference on Learning Representations*. 2015. Available also from: https://arxiv.org/abs/1409.1556.

33. FONG, Ruth; PATRICK, Mandela; VEDALDI, Andrea. Understanding Deep Networks via Extremal Perturbations and Smooth Masks. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019. Available from DOI: 10.1109/ICCV.2019.00488.

34. SUNDARARAJAN, Mukund; TALY, Ankur; YAN, Qiqi. Axiomatic Attribution for Deep Networks. In: *Proceedings of the 34th International Conference on Machine Learning*. 2017, pp. 3319–3328. Available also from: https://proceedings.mlr.press/v70/sundararajan17a.html.

35. PETSIUK, Vitali; DAS, Abir; SAENKO, Kate. RISE: Randomized Input Sampling for Explanation of Black-box Models. In: *Proceedings of the British Machine Vision Conference (BMVC)*. 2018. Available also from: https://arxiv.org/abs/1806.07421.

36. SIMONYAN, Karen; VEDALDI, Andrea; ZISSERMAN, Andrew. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In: *International Conference on Learning Representations Workshop*. 2014. Available also from: https://arxiv.org/abs/1312.6034.

37. CHEFER, Hila; GUR, Shir; WOLF, Lior. Transformer Interpretability Beyond Attention Visualization. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 782–791. Available from DOI: 10.1109/CVPR46437.2021.00084.

38. GELLY, Sylvain. Monte-Carlo Tree Search and Rapid Action Value Estimation in Computer Go. *Artif. Intell.* 2011, vol. 175, pp. 1856–1875. Available from DOI: 10.1016/j.artint.2011.03.007.

39. KOKHLIKYAN, Narine; MIGLANI, Vivek; MARTIN, Miguel; WANG, Edward; ALSALLAKH, Bilal; REYNOLDS, Jonathan; MELNIKOV, Alexander; KLIUSHKINA, Natalia; ARAYA, Carlos; YAN, Siqi; REBLITZ-RICHARDSON, Orion. *Captum: A Unified and Generic Model Interpretability Library for PyTorch*. arXiv, 2020. No. arXiv:2009.07896. Available from DOI: 10.48550/arXiv.2009.07896. Accessed: 2026-05-10.

40. GILDENBLAT, Jacob; CONTRIBUTORS. *PyTorch library for CAM methods* [https://github.com/jacobgil/pytorch-grad-cam]. GitHub, 2021. Accessed: 2026-05-10.

A MCTS Implementation Details

This appendix details the four phase procedures of the MCTS algorithm described in Section 5.3.2. Throughout all procedures, the graph $G = (V, E)$, the scoring objective g_σ, and the evaluation context (model p, image I, replacement image I_{rep}, target class c) are available as shared context and are not passed as explicit parameters.

A.1 Phase Algorithms

Algorithm 4 MCTS Phase 1 & 2: Selection and Expansion

function SELECT($node, m, c, \alpha$)
  $path \leftarrow [node]$
  while IsFullyExpanded($node$) **and not** IsTerminal($node, m$) **do**
    let $Q(ch) = \alpha \cdot ch.maxVal + (1 - \alpha) \cdot ch.meanVal$
    let $q_{min} = \min_{ch \in node.children} Q(ch), \quad q_{max} = \max_{ch \in node.children} Q(ch)$
    for $ch \in node.children$ **do**
      if $q_{\max} > q_{\min}$ **then**
        $\widetilde{Q}(ch) \leftarrow 2 \cdot \dfrac{Q(ch) - q_{\min}}{q_{\max} - q_{\min}} - 1$
      else
        $\widetilde{Q}(ch) \leftarrow 1$ /* Optimistic fallback when all children share the same value */
      end if
      $\mathrm{UCT}(ch) \leftarrow \widetilde{Q}(ch) + c\sqrt{\dfrac{\ln(node.visits)}{ch.visits}}$
    end for
    $node \leftarrow \arg\max_{ch \in node.children} \mathrm{UCT}(ch)$
    $path$.append($node$)
  end while
  return $node, path$
end function

function EXPAND($node, path, m$)
  if not IsTerminal($node, m$) **then**
    $node \leftarrow$ ExpandRandomChild($node$)
    $path$.append($node$)
  end if
  return $node, path$
end function

Algorithm 5 MCTS Phase 3 & 4: Simulation and Backup

```
function SIMULATE(node, m, B)
    batch ← []
    for b = 1 to B do
        batch.append(SAMPLECONNECTEDSUPERSET(node.state, m))
    end for
    return UNIQUE(batch) /* Deduplicate before GPU evaluation */
end function

function BACKPROPAGATE(path, rewards)
    k ← |rewards|
    sumRewards ← ∑ rewards,    maxReward ← max(rewards)
    for n ∈ path do
        n.meanVal ← (n.meanVal · n.visits + sumRewards) / (n.visits + k)
        n.visits ← n.visits + k
        n.maxVal ← max(n.maxVal, maxReward)
    end for
end function
```

A.2 Data Structures

MCTSNode: A node in the search tree encapsulating a connected region of superpixels. Each node stores:

- *state*: the connected region $R \subseteq V$ this node represents,
- *children*: a list of child nodes (each an **MCTSNode**), one per frontier vertex already expanded from *state*,
- *visits*: the number of rollout rewards routed through this node,
- *meanVal*: the running average of those rewards,
- *maxVal*: the maximum reward discovered in the subtree rooted at this node.

A.3 Helper Functions

- **IsFullyExpanded(*node*)**: Returns true if every frontier superpixel adjacent to the current state has already been instantiated as a child node.
- **IsTerminal(*node*, *m*)**: Returns true if the state size has reached m or if no further expansion is possible (empty frontier).
- **EvaluateBatch(*batch*)**: Executes a single batched forward pass on the GPU to score a set of regions. The reward for each region is the log-odds score difference of the target class relative to the baseline occlusion.

- **ExpandRandomChild(***node***)**: Identifies all frontier superpixels not yet instantiated as children, selects one uniformly at random, and creates it as a new child node.
- **SampleConnectedSuperset(***state***, m)**: Performs a random walk starting from *state*, appending uniformly sampled valid neighbors one at a time until the region reaches size m or the frontier is exhausted.

B Search Algorithm Configurations

B.1 Search Space

For MCGS and MCTS, we swept the exploration constant $c \in \{0.1, 0.7, 1.4, 3, 5\}$, the value mixing parameter $\alpha \in \{0, 0.5, 1\}$, and the rollout count `numRollouts` $\in \{1, 16, 64\}$. For UCB, the same c and α grid was utilized with a batch size `UCBBatchSize` $\in \{1, 16, 64\}$.

B.2 Evaluation Budget

A purely theoretical sweep would fix the total number of model evaluations (`numEvals`) and vary the rollout or batch size. However, batched GPU evaluation is significantly more efficient than sequential evaluation; processing 64 states in a single forward pass is computationally "cheaper" than 64 individual passes. Fixing `numEvals` would therefore unfairly penalize larger rollout counts by underutilizing GPU parallelism.

Instead, we fixed an empirical wall-clock budget of approximately 30 seconds per image and calibrated the `numEvals` (or step budgets for UCB) to match this practical compute cost. These calibrated budgets for $m = 30$ segments are shown in Table B.1. Note that for UCB, the budget refers to the number of batch-update steps rather than total rollouts.

Table B.1: Calibrated budgets for each rollout or batch size ($m = 30$ segments, `numRegions = 1`).

Algorithm	Budget by rollout/batch size		
	1	16	64
MCGS	10,000	85,000	100,000
MCTS	9,000	83,000	100,000
UCB	300	2,767	3,333

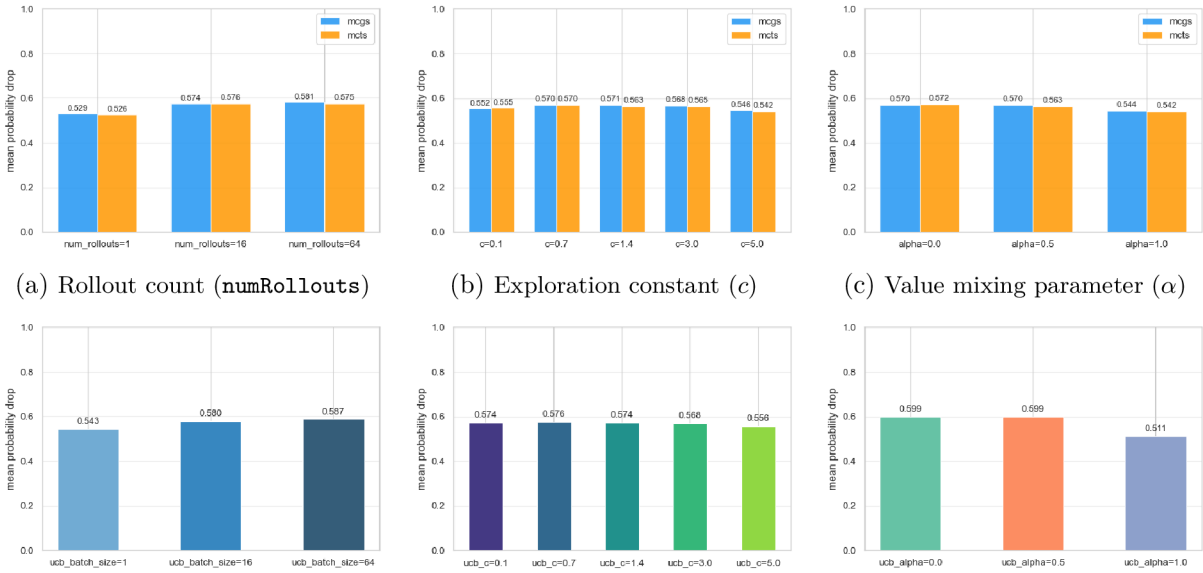
B.3 Effect of Hyperparameters

For MCTS and MCGS, `numRollouts` controls the quality of value estimates per step at the cost of additional model evaluations. The exploration constant c balances exploitation of known high-value configurations against exploration of unvisited parts of the search tree. The mixing parameter α blends the maximum and mean rollout return: $\alpha = 1$ favors outliers while $\alpha = 0$ relies on the mean. For UCB, c and α play the same roles, and `UCBBatchSize` controls how many states are evaluated in a single GPU forward pass.

The grid search results, summarized in Figure B.1, confirm several of these trends. For MCTS and MCGS, `numRollouts = 1` performed noticeably worse, while 64 rollouts

offered only marginal gains over 16. Both extreme values of c ($c = 0.1$ and $c = 5$) performed slightly below the middle range—too greedy or too exploratory, respectively, to make good use of the fixed time budget. Setting $\alpha = 1$ consistently degraded performance; $\alpha = 0$ and $\alpha = 0.5$ were comparable. For UCB, larger batch sizes improved results, though the step from 16 to 64 was again modest. The exploration constant had almost no effect, and $\alpha = 1$ again performed substantially worse than the other values.

Based on these findings, under a low computational budget we recommend keeping c small, using moderate rollout or batch sizes, and avoiding $\alpha = 1$. Under a high computational budget, a larger c is beneficial. Note that the optimal configuration may vary across different segmentation techniques and desired explanation lengths.



(a) Rollout count (`numRollouts`) (b) Exploration constant (c) (c) Value mixing parameter (α)

(d) Batch size (`UCBBatchSize`) (e) Exploration constant (c) (f) Value mixing parameter (α)

Figure B.1: Marginal effect of hyperparameters on mean probability drop. Top row: MCGS and MCTS. Bottom row: UCB.

B.4 Selection of Optimal Configurations

Comparing configurations by raw probability drop is often misleading, as base model confidence varies significantly across images. Similarly, normalization across a single image can be problematic if the performance variance between configurations is negligible; in such cases, small numerical noise is inflated to fill the entire range, distorting the aggregate results.

To ensure a robust selection, we evaluated each configuration using the three metrics defined in Section 7.1.3, ranked them by each metric independently, and selected the configuration with the highest average rank. The top five configurations are listed in Tables B.2 and B.3; the full heatmaps across all metrics are shown in Figures B.2 to B.5 for MCGS and MCTS, and Figures B.6 to B.9 for UCB.

Table B.2: Top five MCGS/MCTS configurations by final average rank.

Algorithm	Rollouts	c	α	Prob. drop	Norm score	Mean rank	Final score
MCGS	64	1.4	0	$\mathbf{0.61 \pm 0.30}$	0.83 ± 0.16	17.78 ± 15.39	**2.00**
MCTS	64	1.4	0	0.61 ± 0.31	0.83 ± 0.19	$\mathbf{17.30 \pm 15.36}$	3.33
MCGS	16	1.4	0	0.60 ± 0.31	$\mathbf{0.83 \pm 0.18}$	18.20 ± 17.18	4.33
MCGS	64	3.0	0.5	0.61 ± 0.31	0.81 ± 0.16	19.47 ± 14.56	5.67
MCTS	16	0.7	0	0.60 ± 0.31	0.82 ± 0.21	19.02 ± 17.02	5.67

Table B.3: Top five UCB configurations by final average rank.

Batch size	c	α	Prob. drop	Norm score	Mean rank	Final score
64	1.4	0.5	$\mathbf{0.63 \pm 0.30}$	$\mathbf{0.88 \pm 0.20}$	10.82 ± 9.55	**1.67**
16	0.7	0.5	0.62 ± 0.30	0.87 ± 0.16	10.27 ± 6.46	2.00
64	5.0	0.5	0.62 ± 0.30	0.87 ± 0.14	$\mathbf{10.13 \pm 6.52}$	3.33
64	0.1	0.5	0.62 ± 0.30	0.87 ± 0.19	10.95 ± 8.36	3.33
16	0.1	0.5	0.62 ± 0.30	0.86 ± 0.19	12.63 ± 8.17	6.33

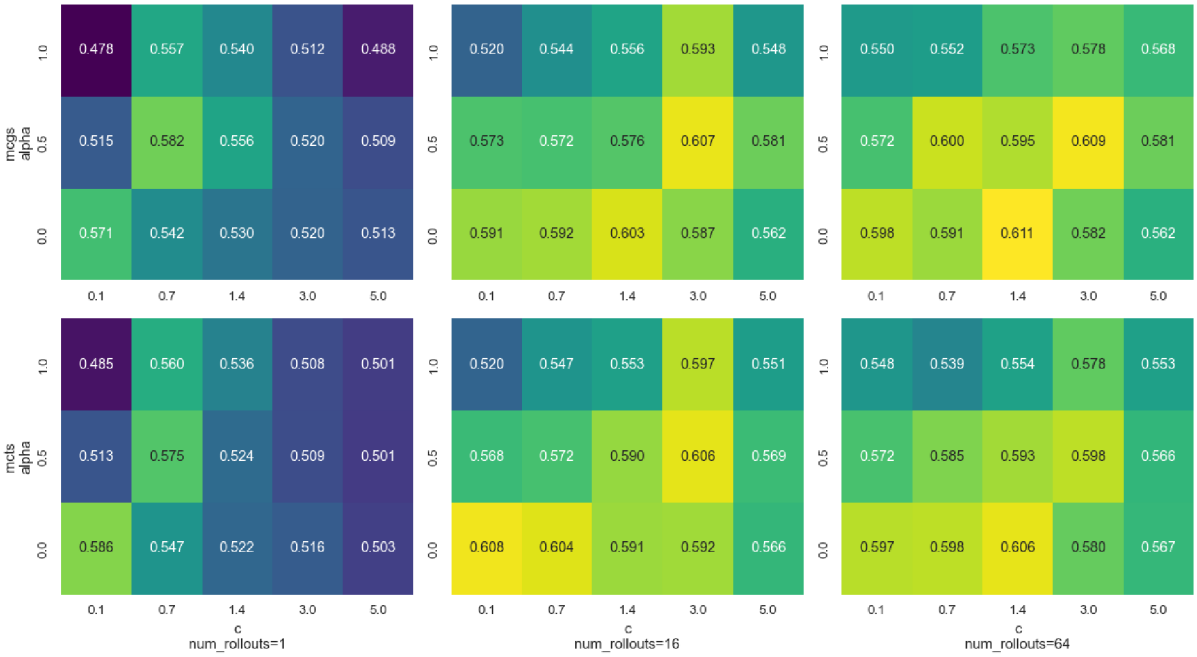


Figure B.2: Mean probability drop for MCGS (top) and MCTS (bottom) as a function of exploration constant c and value mixing parameter α, for rollout counts 1, 16, and 64 (left to right). Higher is better.

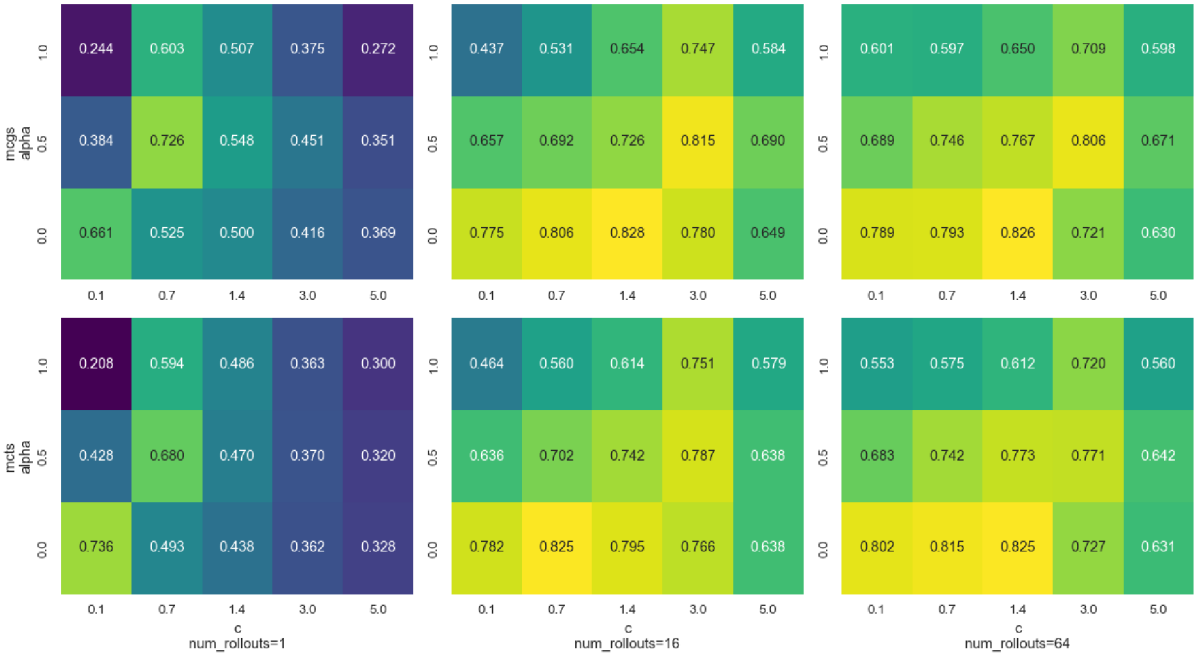


Figure B.3: Normalized score for MCGS (top) and MCTS (bottom) as a function of c and α, for rollout counts 1, 16, and 64 (left to right). Higher is better.

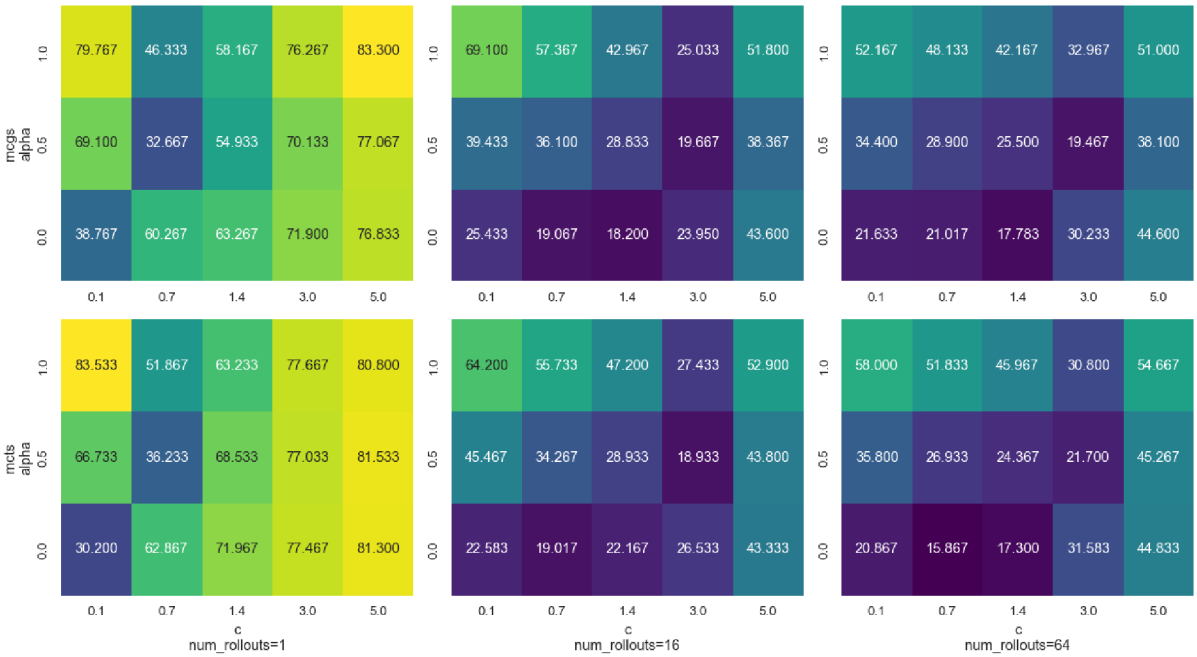


Figure B.4: Mean rank for MCGS (top) and MCTS (bottom) as a function of c and α, for rollout counts 1, 16, and 64 (left to right). Lower is better.

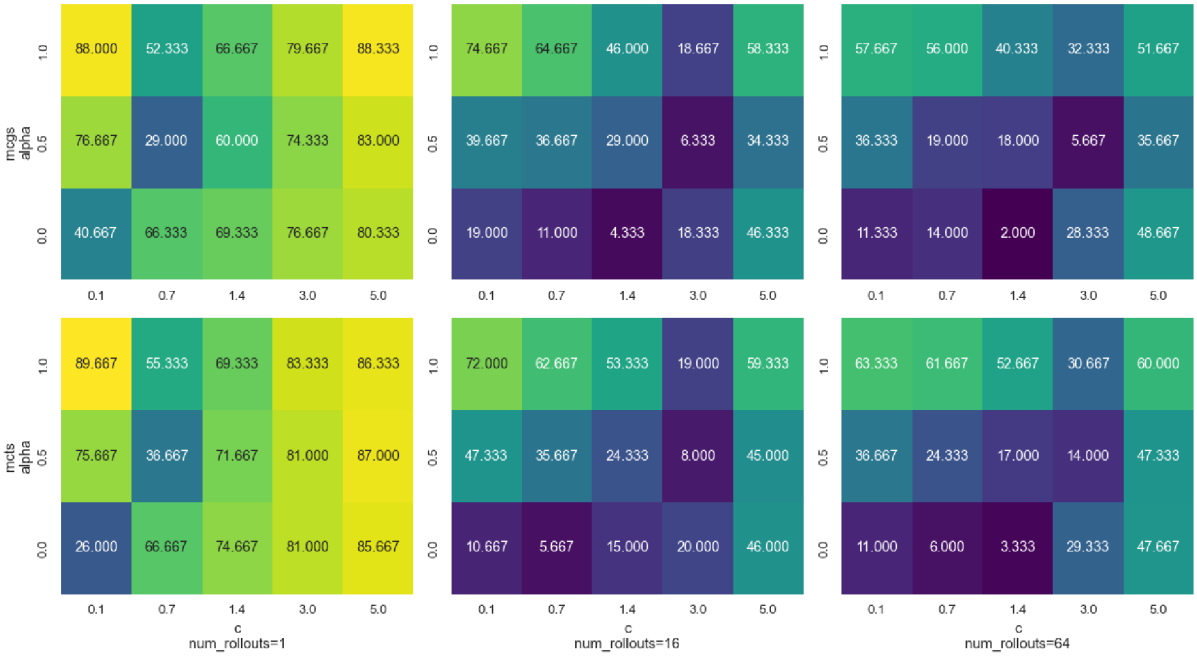


Figure B.5: Final score for MCGS (top) and MCTS (bottom) as a function of c and α, for rollout counts 1, 16, and 64 (left to right). Lower is better.

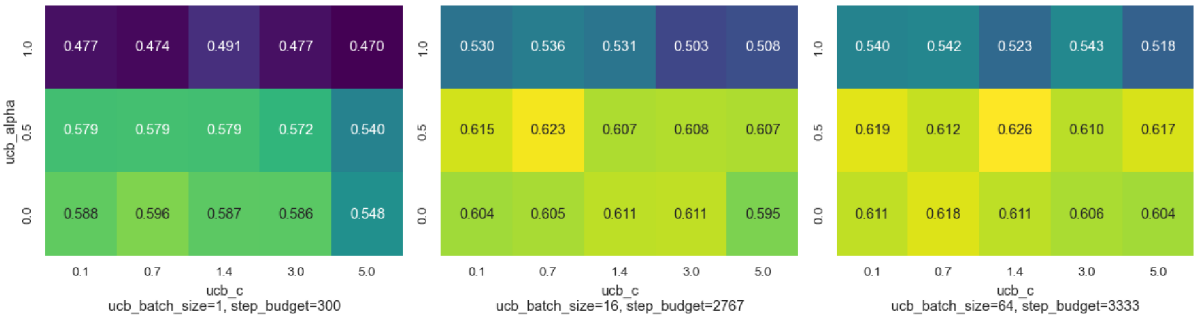


Figure B.6: Mean probability drop for UCB as a function of exploration constant c and value mixing parameter α, for batch sizes 1, 16, and 64 (left to right). Higher is better.

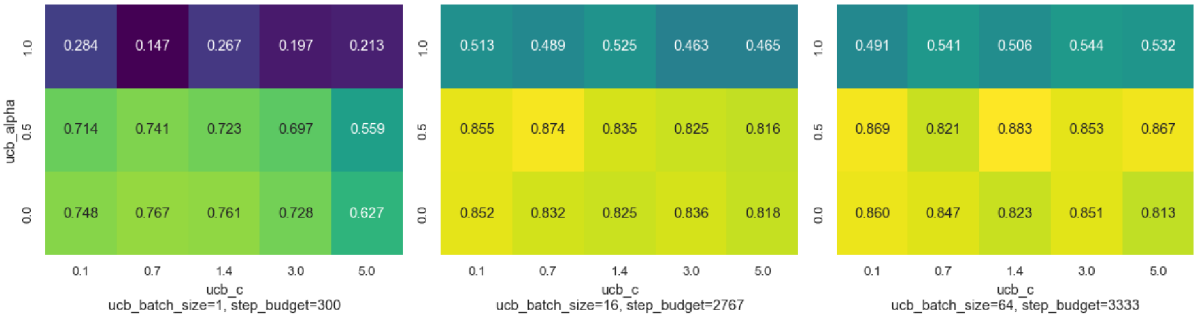


Figure B.7: Normalized score for UCB as a function of c and α, for batch sizes 1, 16, and 64 (left to right). Higher is better.

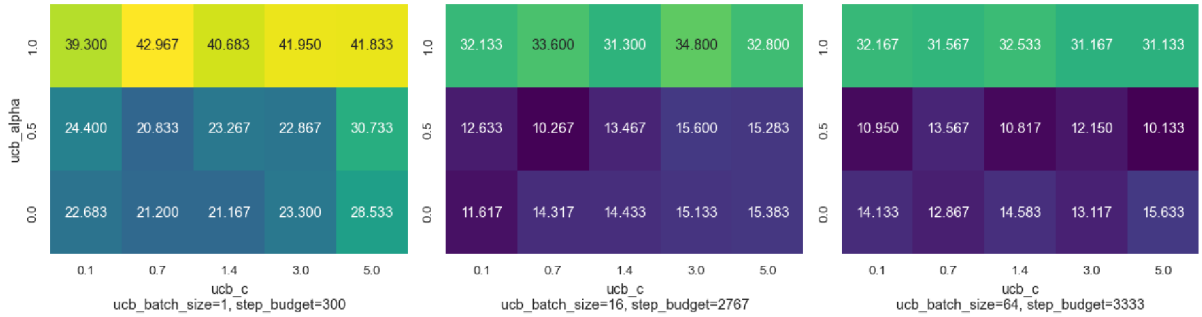


Figure B.8: Mean rank for UCB as a function of c and α, for batch sizes 1, 16, and 64 (left to right). Lower is better.

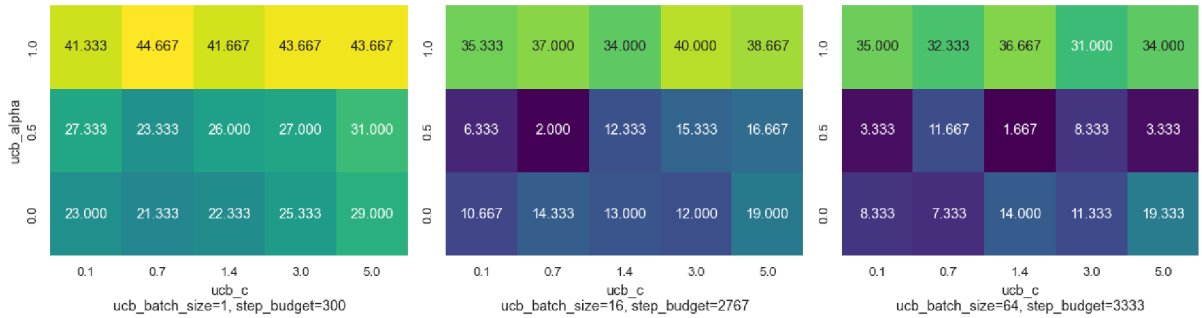


Figure B.9: Final score for UCB as a function of c and α, for batch sizes 1, 16, and 64 (left to right). Lower is better.

Baseline Method Configurations

All baseline methods in Section 7.5 were given a unified budget of 100,000 model evaluations to match MCGS. For methods that operate step-by-step, the per-step budget is $\lfloor 100{,}000/m \rfloor$, where m is the target region size.

Lookahead utilizes a search depth d that varies by target region size m (Table B.4), as smaller regions allow for deeper combinatorial lookahead within the wall-clock budget. **Greedy** is configured as Lookahead with $d = 1$.

Table B.4: Lookahead depth d by target region size m.

m	15	30	60	90
d	4	3	2	2

MCGS and **UCB** utilize the top-ranked configurations from the grid search: MCGS ($c = 1.4, \alpha = 0, \text{Rollouts} = 64$) and UCB ($c = 1.4, \alpha = 0.5, \text{Batch} = 64$).

C Competitor Configurations

This appendix documents the exact configuration used for each competitor in Section 7.6. All methods share the same replacement images as our framework (Section 7.1.2): the ImageNet channel mean for ImageNet-S and Gaussian blur ($\sigma = 5$, kernel 15×15) for PCam.

C.1 LIME

We use the Captum [39] implementation of LIME with SLIC superpixel segmentation as the feature partition. The top-K segments by positive attribution weight are selected as the explanation, where K matches our target size m. The values of SLIC segments and perturbation samples are given in Table C.1.

Table C.1: LIME configuration by dataset.

Parameter	ImageNet-S (224×224)	PCam (96×96)
SLIC segments	600	100
Samples	10,000	5,000

C.2 Occlusion Sensitivity

We use the Captum sliding-window occlusion implementation. Window size and stride are scaled to image resolution to yield a comparably smooth heatmap; the values are given in Table C.2.

Table C.2: Occlusion Sensitivity configuration by dataset.

Parameter	ImageNet-S (224×224)	PCam (96×96)
Window size	16×16 px	8×8 px
Stride	2 px	1 px

C.3 Grad-CAM++

We use the Grad-CAM++ variant from the `pytorch-grad-cam`[40] library, targeting the final convolutional block (`layer4`) of the ResNet50 backbone. Grad-CAM++ requires only a single forward and backward pass and has no evaluation budget parameter.

C.4 Meaningful Perturbations

We implemented Meaningful Perturbations within our framework, following the original gradient-based soft-mask optimization (1,500 iterations, up to 5 minutes wall-clock time). The optimized masks consistently produced artefacts, concentrating on isolated high-frequency texture patches rather than semantically coherent image areas. We were unable to resolve this with standard regularization tuning, so Meaningful Perturbations were excluded from the final comparison.

C.5 IoU Computation for Saliency Maps

LIME, Occlusion Sensitivity, and Grad-CAM++ produce continuous saliency maps rather than binary masks. To compute IoU against the ground-truth pixel annotations (Section 7.1.3), we threshold each saliency map to retain approximately the same pixel area as our method's region at the comparison target size, converting the map to a binary mask before applying the IoU formula.

D Full Internal Comparison Results

This appendix provides the complete, unaggregated numerical results for the internal method comparison discussed in Section 7.5.

While the main text summarizes the mean rank and normalized scores to establish the relative performance of the expansion strategies, Table D.1 details the raw probability drops, log-odds drops, wall-clock execution times, head-to-head win rates, and mean ranks for every evaluated configuration across all target region sizes ($m \in \{15, 30, 60, 90\}$).

The algorithms evaluated include both the basic stepwise baselines (Pure Monte Carlo, Beam Search, Greedy, Lookahead) and the more advanced stochastic search policies (Potential, MCGS, UCB). The results confirm that while simpler methods perform adequately at small target sizes, the advanced combinatorial policies—particularly UCB—maintain superior optimization quality as the search space grows.

Table D.1: Search algorithm comparison on 100 ImageNet-S images (mean ± std). Prob. drop, log-odds drop, norm. score, and wins are higher-is-better; time and mean rank are lower-is-better. The Wins column counts images on which the method's probability drop exceeds the second-best by more than 0.05; the denominator is the number of images with any such margin at that size.

Algorithm	Metric	$m = 15$	$m = 30$	$m = 60$	$m = 90$
Pure Monte Carlo	Prob. drop	0.36 ± 0.27	0.42 ± 0.29	0.48 ± 0.30	0.51 ± 0.31
	Log-odds drop	3.84 ± 1.99	4.53 ± 2.42	5.61 ± 3.38	6.50 ± 4.01
	Norm. score	0.74 ± 0.22	0.63 ± 0.28	0.65 ± 0.34	0.68 ± 0.36
	Time (s)	25.05 ± 0.31	33.11 ± 0.66	75.58 ± 1.71	110.30 ± 4.59
	Wins	2/40	0/30	0/25	0/16
	Mean rank	4.72 ± 1.10	5.66 ± 0.72	5.76 ± 0.65	5.73 ± 0.65
Beam search (saliency)	Prob. drop	0.19 ± 0.20	0.24 ± 0.23	0.28 ± 0.25	0.31 ± 0.28
	Log-odds drop	2.00 ± 1.39	2.39 ± 1.95	2.80 ± 2.43	3.18 ± 2.60
	Norm. score	0.01 ± 0.05	0.00 ± 0.01	0.00 ± 0.03	0.01 ± 0.07
	Time (s)	2.60 ± 0.23	4.83 ± 0.73	8.80 ± 1.70	13.12 ± 2.60
	Wins	0/40	0/30	0/25	0/16
	Mean rank	6.97 ± 0.30	6.98 ± 0.11	6.96 ± 0.20	6.93 ± 0.36
Greedy	Prob. drop	0.34 ± 0.27	0.45 ± 0.30	0.56 ± 0.31	0.63 ± 0.30
	Log-odds drop	3.65 ± 2.08	5.07 ± 2.71	7.40 ± 4.27	9.22 ± 5.27
	Norm. score	0.62 ± 0.27	0.70 ± 0.28	0.81 ± 0.29	0.88 ± 0.23
	Time (s)	$\mathbf{0.57 \pm 0.07}$	$\mathbf{0.71 \pm 0.10}$	$\mathbf{1.27 \pm 0.18}$	$\mathbf{2.50 \pm 0.40}$
	Wins	0/40	0/30	0/25	1/16
	Mean rank	5.22 ± 1.27	4.80 ± 1.07	3.95 ± 1.37	3.70 ± 1.40
Lookahead	Prob. drop	$\mathbf{0.40 \pm 0.28}$	$\mathbf{0.53 \pm 0.30}$	$\mathbf{0.63 \pm 0.30}$	$\mathbf{0.68 \pm 0.30}$
	Log-odds drop	$\mathbf{4.38 \pm 2.34}$	$\mathbf{6.38 \pm 3.20}$	9.02 ± 4.54	11.22 ± 5.46
	Norm. score	$\mathbf{0.90 \pm 0.17}$	$\mathbf{0.92 \pm 0.16}$	0.93 ± 0.19	$\mathbf{0.97 \pm 0.15}$
	Time (s)	24.75 ± 6.45	39.01 ± 11.55	23.67 ± 6.82	82.29 ± 25.36
	Wins	**18/40**	**17/30**	**13/25**	**9/16**
	Mean rank	$\mathbf{2.51 \pm 1.59}$	$\mathbf{2.26 \pm 1.36}$	2.17 ± 1.36	2.24 ± 1.29
Potential	Prob. drop	0.40 ± 0.28	0.51 ± 0.30	0.61 ± 0.29	0.64 ± 0.30
	Log-odds drop	4.35 ± 2.27	5.94 ± 3.16	8.52 ± 4.93	10.71 ± 6.53
	Norm. score	0.89 ± 0.17	0.86 ± 0.19	0.89 ± 0.20	0.91 ± 0.20
	Time (s)	17.94 ± 0.61	26.46 ± 0.48	60.67 ± 1.13	85.03 ± 3.37
	Wins	12/40	2/30	2/25	0/16
	Mean rank	2.54 ± 1.30	2.98 ± 1.30	2.92 ± 1.01	2.91 ± 1.05
MCGS	Prob. drop	0.38 ± 0.27	0.51 ± 0.31	0.55 ± 0.31	0.57 ± 0.31
	Log-odds drop	4.09 ± 2.08	6.11 ± 3.26	7.21 ± 4.40	8.11 ± 4.91
	Norm. score	0.83 ± 0.19	0.87 ± 0.20	0.78 ± 0.29	0.79 ± 0.31
	Time (s)	5.32 ± 0.68	25.82 ± 2.67	72.82 ± 2.18	111.37 ± 4.67
	Wins	4/40	3/30	0/25	0/16
	Mean rank	3.52 ± 1.35	2.96 ± 1.14	4.36 ± 0.89	4.65 ± 0.86
UCB	Prob. drop	0.39 ± 0.27	0.53 ± 0.30	0.63 ± 0.30	0.67 ± 0.30
	Log-odds drop	4.29 ± 2.17	6.28 ± 3.41	$\mathbf{9.78 \pm 5.87}$	$\mathbf{12.90 \pm 7.85}$
	Norm. score	0.89 ± 0.16	0.91 ± 0.17	$\mathbf{0.93 \pm 0.18}$	0.95 ± 0.15
	Time (s)	18.26 ± 0.71	27.57 ± 0.63	63.91 ± 1.88	89.96 ± 4.28
	Wins	4/40	8/30	10/25	6/16
	Mean rank	2.52 ± 1.07	2.36 ± 1.20	$\mathbf{1.88 \pm 1.04}$	$\mathbf{1.84 \pm 1.01}$

D.1 Failure Cases

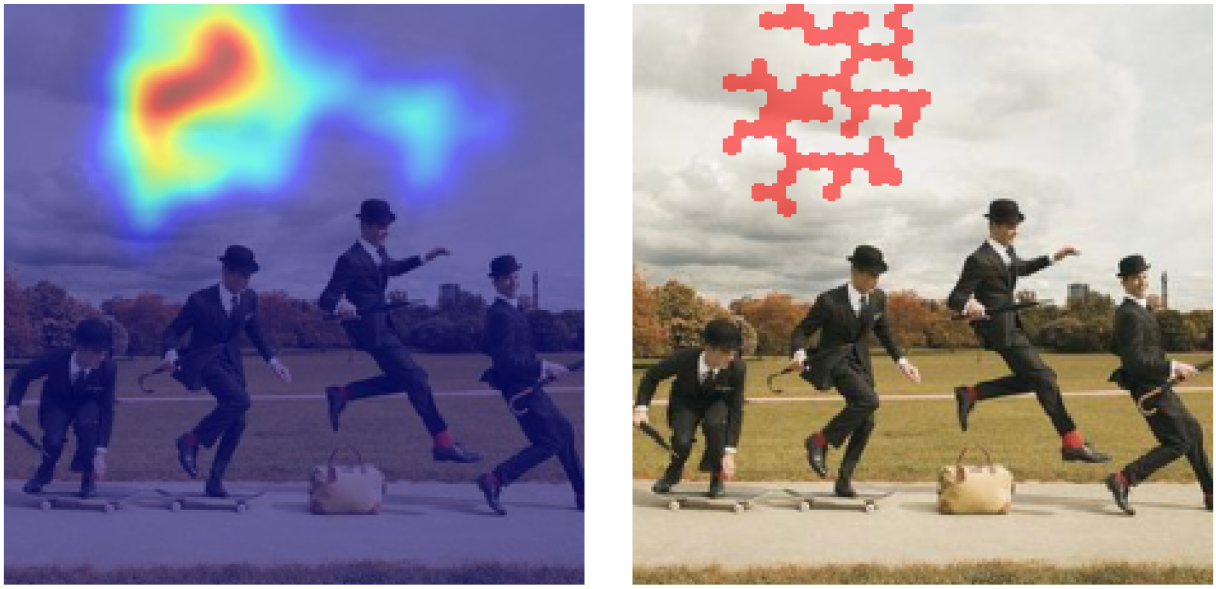
The following figures show representative cases where the identified region does not align with the semantically relevant object area. Each pair shows the saliency map used for seed selection and the found region at $m = 60$.

(a) Saliency map (b) Found region ($m = 60$)

Figure D.1: Class *puck*. The object occupies most of the frame, so even the largest allowed region covers only a corner. The probability drop is modest ($0.9990 \rightarrow 0.9558$) even at $m = 90$, because no small subregion is sufficient to explain a nearly full-frame object.

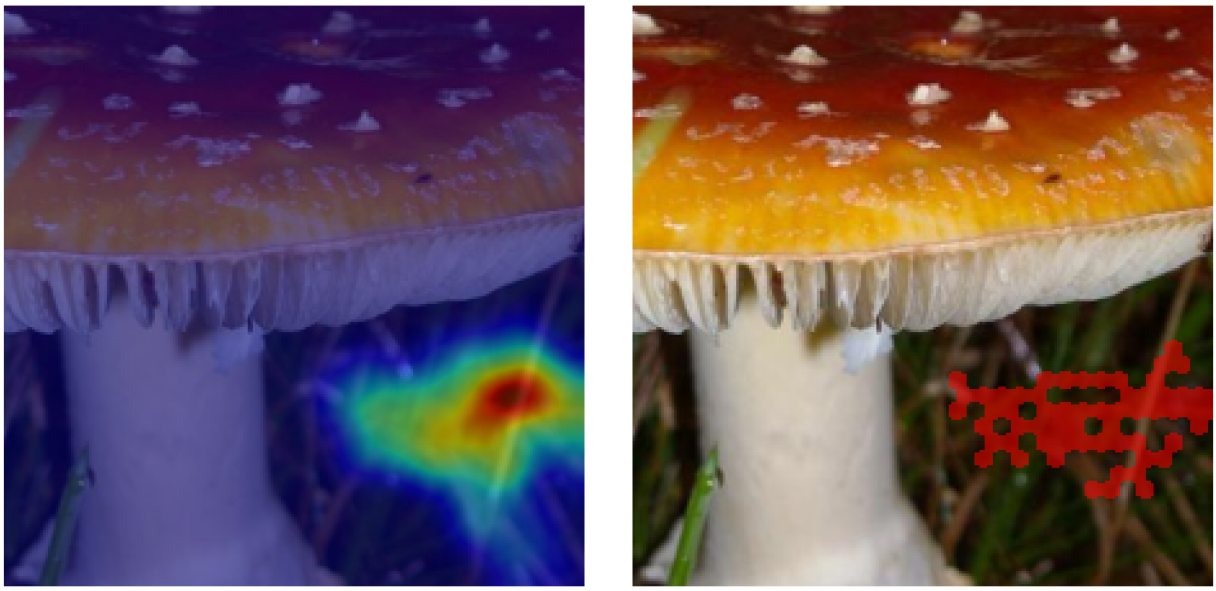
(a) Saliency map (b) Found region ($m = 60$)

Figure D.2: Class *paddlewheel*. The algorithm focuses entirely on the water surface rather than the boat or its wheel—likely because the ripple texture carries more discriminative signal for the classifier than the object itself.



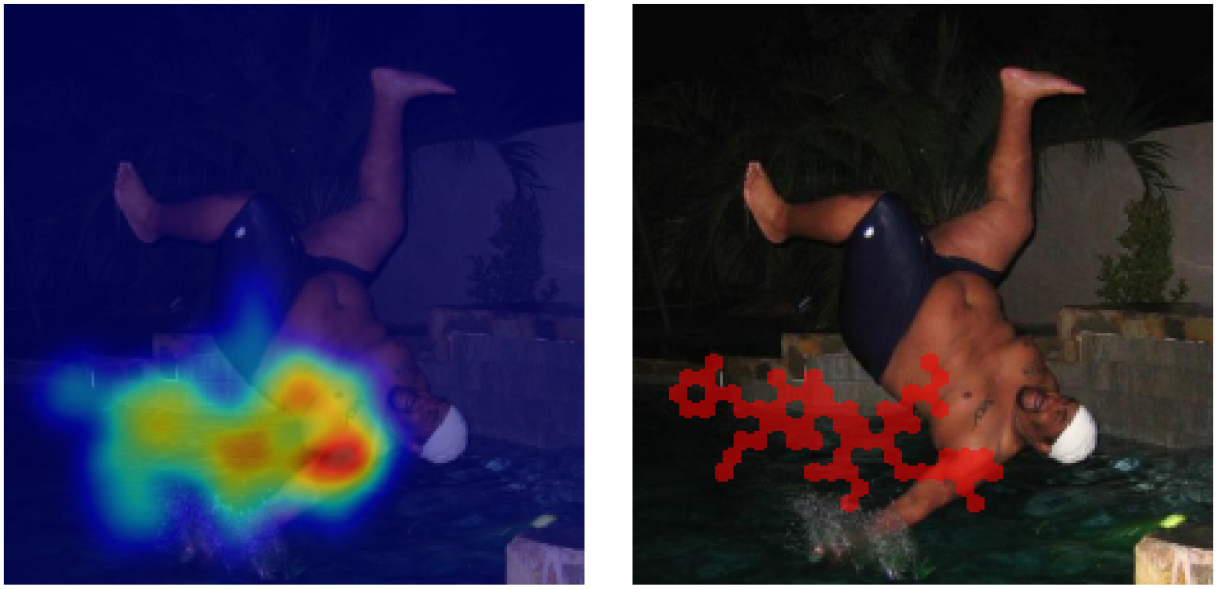
(a) Saliency map (b) Found region ($m = 60$)

Figure D.3: Class *unicycle*. The top predicted class is incorrect—no unicycle is present in the image—and the found region covers sky rather than any object. This reflects a misclassification by the model rather than a failure of the search.



(a) Saliency map (b) Found region ($m = 60$)

Figure D.4: Class *agaric*. The initial scoring mechanism assigns low importance to the mushroom cap superpixels, so no seed is placed there. The search consequently explores the background, where hiding patches produces a larger probability change.



(a) Saliency map (b) Found region ($m = 60$)

Figure D.5: Class *bathing cap*. The found region looks atypical: after the identified area is hidden, the top prediction shifts to *jigsaw puzzle*, suggesting the model attends to a texture pattern rather than the cap's shape or color.

D.2 Region Merging at Small Target Sizes

Figure D.6 shows an example where a small target size causes two separate features to be bridged into one connected region. The red region wraps around the bird's head and partially encloses the smaller teal component, which cannot independently reach size m—the same configuration also illustrates why Ω_m requires $|R| \leq m$, as discussed in Section D.3.

(a) Original image (b) Per-superpixel importance (c) Identified regions

Figure D.6: Region merging at a small target size. At low m, a surrounding region bridges two separate features: the red region wraps around the bird's head and partially encloses a smaller teal component that cannot independently reach size m.

D.3 Why Ω_m Uses $|R| \leq m$ Rather Than $|R| = m$

Figure D.6 illustrates why allowing $|R| \leq m$ is necessary. When a surrounding region is pruned, the remaining connected components may contain fewer than m superpixels, making an exact-size region unreachable. Requiring $|R| = m$ strictly would force the algorithm to skip such components entirely, leaving important enclosed features unexplained.

E FunnyBirds Evaluation Protocols

The FunnyBirds benchmark [1] evaluates XAI methods through a set of automatic, intervention-based protocols. Each protocol queries the model on images with specific parts removed or modified and compares the model's response to the explainer's output. The metrics below are all higher-is-better.

Controlled Synthetic Data Check (CSDC). For each correctly classified image, the explainer identifies a set of important parts P (via thresholding; see Section 7.6). Let $\mathcal{M}$ be the set of minimal sufficient part sets for the predicted class—the smallest subsets of parts whose presence is enough for correct classification. The score for one image is

$$\max_{S \in \mathcal{M}} \frac{|P \cap S|}{|S|}, \tag{E.1}$$

and CSDC is the average over all valid images. It measures whether the explanation covers the causally relevant parts.

Preservation Check (PC). All parts *not* identified as important are removed (replaced by a featureless background). PC is the fraction of images for which the model's predicted class is unchanged after this removal, i.e., the identified parts alone are sufficient to preserve the prediction.

Deletion Check (DC). The identified important parts are removed. DC is the fraction of images for which the model's prediction *changes* after this removal, i.e., the identified parts are necessary for the prediction.

Distractibility (D). Parts (including background objects) whose removal changes the model's confidence by less than 5% are treated as irrelevant. Let Q be the set of irrelevant parts and P the set the explainer marks as important. The score for one image is $1 - |P \cap Q| \,/\, |Q|$, and D is its average. It measures how often the explainer is distracted by irrelevant image elements.

Single Deletion (SD). Each bird part is individually removed, and the drop in model confidence is recorded. SD is the Spearman rank correlation between these confidence drops and the explainer's per-part importance scores, mapped from $[-1, 1]$ to $[0, 1]$ by the transform $r \mapsto 0.5r + 0.5$. It measures whether the explainer correctly ranks parts by their individual importance to the model.

Target Sensitivity (TS). For each image, two alternative classes are found that share disjoint subsets of parts with the true class. The explainer is run separately with each alternative class as target. TS is the fraction of times the explanation correctly assigns higher importance to the parts that overlap with the respective target class, measuring whether explanations are sensitive to the choice of target class.