GUARANTEED OPTIMAL COMPOSITIONAL EXPLANATIONS FOR NEURONS

Anonymous authors
Paper under double-blind review

ABSTRACT

While neurons are the basic units of deep neural networks, it is still unclear what they learn and if their knowledge is aligned with that of humans. Compositional explanations aim to answer this question by describing the spatial alignment between neuron activations and concepts through logical rules. These logical descriptions are typically computed via a search over all possible concept combinations. Since computing the spatial alignment over the entire state space is computationally infeasible, the literature commonly adopts beam search to restrict the space. However, beam search cannot provide any theoretical guarantees of optimality, and it remains unclear how close current explanations are to the true optimum. In this theoretical paper, we address this gap by introducing the first framework for computing guaranteed optimal compositional explanations. Specifically, we propose: (i) a decomposition that identifies the factors influencing the spatial alignment, (ii) a heuristic to estimate the alignment at any stage of the search, and (iii) the first algorithm that can compute optimal compositional explanations within a feasible time. Using this framework, we analyze the differences between optimal and non-optimal explanations in the most popular settings for compositional explanations, the computer vision domain and Convolutional Neural Networks. In these settings, we demonstrate that 10–40% of explanations obtained with beam search are suboptimal when overlapping concepts are involved. Finally, we evaluate a beam-search variant guided by our proposed decomposition and heuristic, showing that it matches or improves runtime over prior methods while offering greater flexibility in hyperparameters and computational resources.

1 INTRODUCTION

Compositional explanations (Mu & Andreas, 2020) are a method for expressing the alignment between the locations of a given neuron activation range and the location of concepts, identified by annotations in datasets. The key idea is capturing the interaction between low-level (e.g., colors, textures, shapes) and high-level concepts (e.g., objects, entities) via propositional logic formulas able to express the alignment between these complex relationships and neuron activations.

One of the key problems to achieve this goal is that the full search space encompassing all of the possible combinations between concepts cannot typically be exhaustively explored due to its size. As a result, prior work has relied on beam search to identify the highest spatial alignment within the restricted space(Mu & Andreas, 2020). Although the resulting explanations are valid, it is unclear whether these explanations are, in fact, optimal (i.e., whether they identify the specific combination that expressed the highest absolute spatial alignment in the search space, see Figure 1).While beam search does not guarantee optimality, it might converge to it due to unstudied or unknown properties of the underlying datasets. If not, the explanations produced by beam search may represent only a subset of the alignment structure, offering a partial view of neuron behavior.

The main contribution of this work is to make navigating the state space tractable. To achieve this, we propose a decomposition of the Intersection-over-Union (IoU) metric that reveals a set of fundamental quantities governing alignment quality, and we design a heuristic and a corresponding algorithm that jointly reduce the size of the state space and guide the search process. This approach enables the computation of optimal compositional explanations in feasible time. To the best of our knowledge, this is the first attempt in this research direction. As a first step, we apply our

Figure 1: An example of a case where the optimal algorithm finds a combination of concepts expressing the highest spatial alignment that beam search fails to capture.

method to the computer vision domain and Convolutional Neural Networks, given its prominence in compositional explanation research.

Our contributions are as follows:

- We propose a decomposed Intersection over Union score (dIoU) that identifies fundamental quantities for alignment quality and enables a better characterization of the impact of logical operators on spatial alignment.
- We design a heuristic and a corresponding algorithm that jointly reduce the size of the state space and guide the search process. We show that this algorithm computes guaranteed optimal explanations in feasible time and we analyze the differences between optimal and non-optimal explanations.
- We show that part of our proposed heuristic can be used directly to guide beam search with significant gains in flexibility and competitive or better performance than competitors. Specifically, our variant scales more effectively than competitors with respect to explanation length and beam size, and it is less resource-intensive and easier to parallelize.

In Section 2, we give an introduction to the relevant literature in the background and specify our framework for optimal compositional explanations. In Section 3, we analyze our contribution.

2 Optimal Compositional Explanations

2.1 Background and Related Work

Neuron explanations aim to understand what individual neurons learn during the training process. Different categories of methods have been proposed to decode different behaviors. Among the most popular in computer vision, we can cite the one that generates samples that capture features recognized by a neuron (Erhan et al., 2009; Olah et al., 2017; Nguyen et al., 2016) and the ones that generate textual descriptions that correlate neuron activations and samples associated with a given concept (Hernandez et al., 2022; Oikarinen & Weng, 2023; 2024; Kalibhat et al., 2023; Ahn et al., 2024; Wang et al., 2022) through foundational models.

Differently from them, compositional explanations are a family of neuron explanations that specifically focus on the **spatial alignment** between a neuron activation range and and the location of concepts and express them through propositional logic formulas. The seminal work in this area is Network Dissection (Bau et al., 2017; 2020), which associates each neuron with the single concept that maximizes this alignment. This approach was extended by Mu & Andreas (2020) to associate relationships between multiple concepts, in an attempt to capture a higher degree of polysemantic

behavior (Elhage et al., 2022). Relationships explored in the literature include co-occurrence (Bau et al., 2017), exclusion (Mu & Andreas, 2020), relative position (Harth, 2022), and hierarchy (Massidda & Bacciu, 2023).

Preliminaries and Terminology Let $\mathfrak{L}^1$ be a concept set including properties of interest for a given model and task (e.g., names of colors, objects, categories, shapes). Let $\mathfrak{D} = \{x_1, x_2, ..., x_n\}$ be a dataset, where each input x_i has dimension d and is associated with a set of annotations (possibly empty) identifying the specific locations of concepts in that sample. In this paper, we use the term "location" to refer to the pair (x, j) where x identifies the position of a sample in the probing dataset and j identifies the position of a feature. In vision tasks, these annotations correspond to segmentation masks that group features (i.e., pixels) semantically related, and a "concept" is the semantic label assigned by a human to that annotation. Note that the dataset can differ from the one used to train the model and is assumed to be used solely for neuron analysis. For this reason, $\mathfrak{D}$ is commonly referred to as a "probing dataset". The literature define the *Concept Tensor* $\mathbf{M}^{\mathfrak{L}^1} \in \{0,1\}^{|\mathfrak{L}^1| \times |\mathfrak{D}| \times d}$ as the binary tensor corresponding to the localization of each concept within the dataset samples, where an element in position $[k, x, j]$ is 1 if the annotations associated with the feature (i.e., pixel) in position j of the sample x include the concept k, and 0 otherwise. From the Concept Tensor, we can extract, for each concept k, the *Concept Matrix* $\boldsymbol{M}_k \in \{0,1\}^{|\mathfrak{D}| \times d}$. We use the notation ${}^1\cdot$ to indicate the set of locations equal to 1 in a binary matrix.

Let z be a neuron to be explained in a probed model, and let d be the dimension of its activations. In this paper, following the standard setting in the compositional explanations literature (Mu & Andreas, 2020), z denotes a neuron in a convolutional layer and d is obtained by upscaling its feature map through interpolation. From its activation, we can identify the *Neuron Activation Matrix* $\boldsymbol{N} \in \{0,1\}^{|\mathfrak{D}| \times d}$ as the binary matrix indicating the locations where the neuron fires within the dataset samples, where an element in position $[x, j]$ is 1 if the activation corresponding to position j in sample x lies within the considered activation range, and 0 otherwise.

Let $\mathfrak{L}^n$ be the set of all possible logical formulas of arity at maximum n between concepts in $\mathfrak{L}^1$. Compositional explanations aim to assign to z the logical combination $L \in \mathfrak{L}^n$ of concepts in $\mathfrak{L}^1$ (e.g., ((Cat OR Car) AND White)) that maximizes the alignment between the localization of a given neuron's activation range $[\tau_1, \tau_2]$ and the localization of the concepts within the probing dataset.

Formally, the algorithm identifies the label $L \in \mathfrak{L}^n$ that maximizes the following objective:

$$\arg\max_{L \in \mathfrak{L}^n} IoU(L, \boldsymbol{N}, \mathbf{M}) \tag{1}$$

where the Intersection Over Union (IoU) measures the overlap between label annotations and neuron activations, and it is defined as:

$$IoU(L, \boldsymbol{N}, \mathbf{M}) = \frac{|{}^1\boldsymbol{N} \cap {}^1\theta(\mathbf{M}, L)|}{|{}^1\boldsymbol{N} \cup {}^1\theta(\mathbf{M}, L)|} \tag{2}$$

$|\cdot|$ indicates the cardinality of a set, ${}^1\cdot$ indicates the set of locations equal to 1 in a binary matrix, and $\theta(\mathbf{M}, L)$ is a function that returns the logical combination of the matrices in $\mathbf{M}$ of the concepts involved in the label L. The label L is typically identified through a search algorithm. To reduce the search space, compositional explanations typically make two assumptions: concepts in the explanation are distinct (**Assumption 1**), and concepts are combined incrementally to form labels (e.g., $(((((A \oplus B) \oplus C) \ldots) \oplus Z)$) (**Assumption 2**), where $\oplus$ indicates a logical connective. Even with these assumptions, $\mathfrak{L}^n$ is too large to be explored exhaustively. The total number of combinations is $\sum_{k=1}^{n} n_o^{k-1} \prod_k (|\mathfrak{L}^1| - k)$, where n_o is the number of logic connectives, and each combination requires the comparison of $2d$ values to compute the alignment. In the settings considered by Mu & Andreas (2020), this leads to 2.8×10^{14} operations, rendering both storage and runtime infeasible. To cope with this, prior work adopts vanilla (Mu & Andreas, 2020; Harth, 2022; Massidda & Bacciu, 2023; Makinwa et al., 2022) or informed (La Rosa et al., 2023) beam search with a small beam size. However, beam search does not guarantee optimality, leaving open the question of whether the resulting compositional explanations are the best possible or if better-aligned explanations exist. In the following, we characterize the fundamental quantities that influence alignment and propose both a heuristic and an algorithm that guarantees the optimality of explanations.

2.2 FUNDAMENTAL QUANTITIES

This section introduces our proposed decomposed Intersection over Union score (dIoU) and the corresponding fundamental quantities. Alongside the assumptions introduced earlier, we adopt the following:

Assumption 3. *The logic operators connecting concepts are 00-preserving (i.e., they cannot produce a 1 from two zeros).*

This assumption has been implicitly adopted in prior literature, where the commonly used bitwise OR, AND, and AND NOT operators all satisfy the 00-preserving property. Given the setup described in the previous section, we propose the definitions of the following quantities.

Definition 2.1. We define the set U of *unique elements* of $\mathfrak{D}$ as the set of locations associated with exactly one annotation, and the set C of *common elements* of $\mathfrak{D}$ as the set of locations associated with multiple annotations.

$$U = \{(x, j) \mid \exists! k \in \mathfrak{L}^1 \text{ s.t. } x \in \mathfrak{D}, j \in [d], \mathbf{M}^{\mathfrak{L}^1}[k, x, j] = 1$$
$$C = \{(x, j) \mid \exists k_1, k_2 \in \mathfrak{L}^1 \text{ s.t. } x \in \mathfrak{D}, j \in [d], k_1 \neq k_2, \mathbf{M}^{\mathfrak{L}^1}[k_1, x, j] = 1, \mathbf{M}^{\mathfrak{L}^1}[k_2, x, j] = 1$$

In other words, unique elements are features associated with one and only one concept, while common elements are features associated with multiple concepts.

Definition 2.2. Given a neuron activation matrix $\boldsymbol{N}$, we define the *unique activation* set N^U as the set of locations where the neuron fires on unique elements, and the *common activations* N^C as the set of locations where the neuron fires on common elements.

$$N^U = \{(x, j) \mid (x, j) \in U, \boldsymbol{N}[x, j] = 1\}$$
$$N^C = \{(x, j) \mid (x, j) \in C, \boldsymbol{N}[x, j] = 1\}$$

Definition 2.3. Given a neuron activation matrix $\boldsymbol{N}$, a concept $k \in \mathfrak{L}^1$, and its Concept Matrix $\boldsymbol{M}_k$, we define the *unique intersection* set I^U as the set of unique elements that are annotated with k and where the neuron fires, and the *common intersection* set I^C as the set of common elements in annotated with k where the neuron fires.

$$I^U(k) = \{(x, j) \mid \boldsymbol{M}_k[x, j] = 1, (x, j) \in N^U\}$$
$$I^C(k) = \{(x, j) \mid \boldsymbol{M}_k[x, j] = 1, (x, j) \in N^C\}$$

Definition 2.4. Given a neuron activation matrix $\boldsymbol{N}$, a concept $k \in \mathfrak{L}^1$, and its Concept Matrix $\boldsymbol{M}_k$, we define the *unique extras* set E^U as the set of all unique elements in $\mathfrak{D}$ annotated with the concept k and where the neuron does not fire, and the *common extras* set E^C as the set of all common elements in $\mathfrak{D}$ annotated with the concept k where the neuron does not fire.

$$E^U(k) = \{(x, j) \mid \boldsymbol{M}_k[x, j] = 1, (x, j) \in U, (x, j) \notin N^U\}$$
$$E^C(k) = \{(x, j) \mid \boldsymbol{M}_k[x, j] = 1, (x, j) \in C, (x, j) \notin N^C\}$$

Definition 2.3 and Definition 2.4 can be generalized to the case of label $L \in \mathfrak{L}^{\mathfrak{n}}$. In this case, the binary label matrix $\boldsymbol{M}_L$ is obtained as the result of the bitwise logic operations induced by the logic operators connecting single concepts $k \in L$.

Definition 2.5. Given a binary neuron activation matrix $\boldsymbol{N}$ and a label $L \in \mathfrak{L}^{\mathfrak{n}}$, the decomposed alignment between the annotations associated with L and the neuron activations is defined as:

$$dIoU(\boldsymbol{N}, L, \mathfrak{D}, I^C, I^U, E^U, E^C) = \frac{\sum_{x \in \mathfrak{D}} |I^U(L)_x| + |I^C(L)_x|}{|{}^1\boldsymbol{N}| + \sum_{x \in \mathfrak{D}} |E^U(L)_x| + |E^C(L)_x|} \tag{3}$$

where the subscript indicates the quantity per sample.

Lemma 1. Given a binary neuron activation matrix $\boldsymbol{N}$ and a label $L \in \mathfrak{L}^{\mathfrak{n}}$, the decomposed alignment between the annotations associated with L and the neuron activations is equivalent to the Intersection Over Union if the logical operators involved in L are 00-preserving:

$$\frac{\sum_{x \in \mathfrak{D}} |I^U(L)_x| + |I^C(L)_x|}{|{}^1\boldsymbol{N}| + \sum_{x \in \mathfrak{D}} |E^U(L)_x| + |E^C(L)_x|} = \frac{|{}^1\boldsymbol{N} \cap {}^1\boldsymbol{M}_L|}{|{}^1\boldsymbol{N} \cup {}^1\boldsymbol{M}_L|} \tag{4}$$

Proof. See Section A. □

We visualize all the identified quantities and the $dIoU$ score in Table 3 in the Appendix.

Observation 1 (Impact of Operators on Quantities)**.** Given the above definitions, we can quantify the impact of the OR, AND, and AND NOT bitwise logic operators connecting two concepts k_1 and k_2. The OR operator is 1-preserving (i.e., any 1 in either concept remains 1), and thus it preserves the common elements shared by the two concepts and combines (i.e., sums) the non-shared ones and unique elements. The AND operator is 0-preserving (i.e., any 0 in either concept forces 0) and therefore removes all of the unique elements as well as common elements not shared by both concepts. Finally, the AND NOT operator preserves all of the unique elements but removes the common elements shared by both concepts. These operators will be the ones considered in this paper.

2.3 Heuristic

This section introduces our proposed heuristic. Given a label $L \in \mathfrak{L}^i$ s.t. $i \leq n$, the generic goal of this heuristic is to estimate the label alignment in a faster and less computationally intensive way than directly computing it. Specifically, the proposed heuristic gives an estimation for the following maximum and minimum quantities: (i) given a logic operator $\oplus$, a label $\mathfrak{L}^i$, and a concept k, the estimate alignment of $L \oplus k$, (ii) alignment reachable starting from L and chaining additional concepts not yet included in L, up to a length of n.

2.3.1 Estimate Label Quantities

To facilitate the estimation of the alignment of $L \oplus k$, we introduce the binary *Disjoint Matrix* $D \in \{0,1\}^{|\mathfrak{L}^1|\times|\mathfrak{L}^1|}$, which encodes whether two concepts share any annotation overlap and can be computed once per dataset as a preprocessing step. Specifically, for every pair (k_1, k_2) of concepts:

$$D[k_1,k_2] = \begin{cases} 1, & \text{if } \forall (x,j) \in |\mathfrak{D}| \times d \mid \boldsymbol{M}_{k_1}[x,j] \neq \boldsymbol{M}_{k_2}[x,j], \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

In addition, we introduce two new quantities, *Space for Common Extras* and *Space for Unique Extras*, derived from Definitions 2.1, 2.2 and 2.4, to characterize and constrain the available space for the set of extra elements:

$$SE^C = \{(x,j) \mid (x,j) \in C, \boldsymbol{N}[x,j] = 0\} \qquad SE^U = \{(x,j) \mid (x,j) \in U, \boldsymbol{N}[x,j] = 0\} \tag{6}$$

By Assumption 2, we can separate the label into its left ($L_{\leftarrow}$) and right ($L_{\rightarrow}$) sides. We can observe that unique elements are always disjoint elements and cannot be shared by definition definition 2.1. Therefore, their value can be computed exactly by using Observation 1.

Regarding the common elements, we can use D to check whether the left and right sides of L are disjoint and estimate alignment differently in the two cases.

Disjoint: If the two sides are disjoint, then common elements follow the same principle as the unique elements since the sides do not share elements. Note that for the AND NOT operator, all the quantities converge to the $L_{\leftarrow}$, representing a mathematical equivalence and a degenerate case.[1] To force the algorithm to ignore these duplicate and degenerate cases, we manually set their dIoU to 0.

Overlap: If the two sides overlap, we can estimate the exact value of the unique quantities for OR and AND using the same formulas as in the disjoint case. For the AND NOT operator, by Observation 1, the value equals the quantity of the left side. For the common quantities, we can derive them by combining the definitions in Section 2.2 and Observation 1, obtaining:

[1]For example, "cat AND NOT dog" is always true, and the AND NOT side does not add meaningful information to the explanation.

$$|I^C_{min}(L)_x| = \begin{cases} \max(|I^C_{min}(L_\leftarrow)_x|, |I^C(L_\rightarrow)_x|) & \text{if } \oplus = OR \\ \max(|I^C_{min}(L_\leftarrow)_x| + |I^C(L_\rightarrow)_x| - |N^C_x|, 0) & \text{if } \oplus = AND \\ \max(|I^C_{min}(L_\leftarrow)_x| - |I^C(L_\rightarrow)_x, 0) & \text{if } \oplus = AND\ NOT \end{cases} \tag{7}$$

$$|I^C_{max}(L)_x| = \begin{cases} \min(|I^C_{max}(L_\leftarrow)_x| + |I^C(L_\rightarrow)_x|, |N^C_x|) & \text{if } \oplus = OR \\ \min(|I^C_{max}(L_\leftarrow)_x|, |I^C(L_\rightarrow)_x|) & \text{if } \oplus = AND \\ \min(|I^C_{max}(L_\leftarrow)_x|, |N^C_x| - |I^C(L_\rightarrow)_x|) & \text{if } \oplus = AND\ NOT \end{cases} \tag{8}$$

$$|E^C_{min}(L)_x| = \begin{cases} \max(|E^C_{min}(L_\leftarrow)_x|, |E^C(L_\rightarrow)_x|) & \text{if } \oplus = OR \\ \max(|E^C_{min}(L_\leftarrow)_x| + |E^C(L_\rightarrow)_x| - |SE^C_x|, 0) & \text{if } \oplus = AND \\ \max(|E^C_{min}(L_\leftarrow)_x| - |E^C(L_\rightarrow)_x|, 0) & \text{if } \oplus = AND\ NOT \end{cases} \tag{9}$$

$$|E^C_{max}(L)_x| = \begin{cases} \min(|E^C_{max}(L_\leftarrow)_x| + |E^C(L_\rightarrow)_x|, |SE^C_x|) & \text{if } \oplus = OR \\ \min(|E^C_{max}(L_\leftarrow)_x|, |E^C(L_\rightarrow)_x|) & \text{if } \oplus = AND \\ \min(|E^C_{max}(L_\leftarrow)_x|, |SE^C_x| - |E^C(L_\rightarrow)_x|) & \text{if } \oplus = AND\ NOT \end{cases} \tag{10}$$

Note that, since $L_\rightarrow$ is always an atomic concept (by Assumption 2), its quantities are exact. A detailed derivation is provided in Section E.

Aggregated Computation The above quantities are defined per-sample, requiring $|\mathfrak{D}|$ comparisons for each computation. This can still be costly for large state spaces. To mitigate this, we introduce a lighter aggregated computation, obtained by summing the values per label. For example, $\sum_{x\in\mathfrak{D}} \min(|E^C_{max}(L_\leftarrow)_x|, |SE^C_x| - |E^C(L_\rightarrow)_x|)$ can be transformed in $\min(\sum_{x\in\mathfrak{D}} |E^C_{max}(L_\leftarrow)_x|, \sum_{x\in\mathfrak{D}} |SE^C_x| - \sum_{x\in\mathfrak{D}} |E^C(L_\rightarrow)_x|)$. The trade-off is precision versus efficiency: the sample version is more accurate but computationally intensive, while the aggregated version can be pre-computed once per label, reducing the cost to a single comparison per quantity at the expense of precision (see Section C for details).

2.3.2 Estimate Paths

In this section, we estimate the maximum score achievable by concatenating an arbitrary number of concepts to a label. If the explanation length were unbounded, this maximum would converge to 1, making the heuristic uninformative. However, compositional explanations are inherently bounded, as long explanations would not aid user understanding; in practice, the maximum length is typically small (e.g., 3). This boundedness allows us to produce tighter estimates. Given a label L, our goal is to estimate the numerator and denominator in Definition 2.5 for all possible paths obtained by concatenating additional concepts through logic operators. To this end, we introduce the *maximum and minimum improvement*. For each sample, we extract the $top-n$ and $bottom-n$ values for each quantity and compute cumulative sums into the Top and $Bott$ vectors, starting from the largest or smallest values, respectively. For example, $Top_k(I^C)_x$ represents the cumulative sum of the common intersection values of the k concepts with the highest scores for that quantity. These quantities, combined with Observation 1 and the equations in Section 2.3.1, allow us to compute the maximum and minimum factors for OR, AND, and AND NOT exclusive paths (i.e., paths where only one operator is used from that point onward to concatenate additional concepts):

$$|I_{min}(L)_x| = \begin{cases} \max(|I^C_{min}(L)_x| + |I^U_{min}(L)_x|, Bott_1(I^C)_x + Bott_1(I^U)_x) & \text{OR Path} \\ 0 & \text{AND Path} \\ |I^U_{min}(L)_x| & \text{AND NOT Path} \end{cases} \tag{11}$$

$$|I_{max}(L)_x| = \begin{cases} \min(|I^C_{max}(L)_x| + Top_t(I^C)_x, |N^C_x|) \\ \quad + \min(|I^U_{max}(L)_x| + Top_t(I^U)_x, |N^U_x|) & \text{OR Path} \\ \min(|I^C_{max}(L)_x|, Top_1(I^C)_x) & \text{AND Path} \\ |I^U_{max}(L)_x| + \min(|I^C_{max}(L)_x|, |N^C_x| - Bott_1(I^C)_x) & \text{AND NOT Path} \end{cases} \tag{12}$$

$$\tag{13}$$

$$|Union_{min}(L)_x| = |\boldsymbol{N}_x| + \begin{cases} \max(|E^C_{min}(L)_x| + |E^U_{min}(L)_x|, \\ \quad Bott_1(E^C)_x + Bott_1(E^U)_x) & \text{OR Path} \\ 0 & \text{AND Path} \\ |E^U_{min}(L)_x| & \text{AND NOT Path} \end{cases} \tag{14}$$

$$|Union_{max}(L)_x| = |\boldsymbol{N}_x| + \begin{cases} \min(|E^C_{max}(L)_x| + Top_t(E^C)_x, |SE^C_x|) \\ \quad + \min(|E^U_{max}(L)_x| + Top_t(E^U)_x, |SE^U_x|) & \text{OR Path} \\ \min(|E^C_{max}(L)_x|, Top_1(E^C)_x) & \text{AND Path} \\ |E^C_{max}(L)_x| + \min(|E^U_{max}(L)_x|, \\ \quad |SE^C_x| - Bott_1(E^C)_x) & \text{AND NOT Path} \end{cases} \tag{15}$$

where t denotes the difference between the maximum length and the length of the label L. To estimate the values for paths involving multiple operators, we take the maximum and minimum of each quantity across the operators considered (see Section E.3 for a discussion about explicitly modeling every possible combination). Among the paths, we also include the *final path*, computed by Definition 2.5, to denote the case where the label is not further expanded.

These estimates are finally used to compute the maximum and the minimum dIoU:

$$dIoU_{max} = \frac{\sum_{x \in \mathfrak{D}} |I_{max}(L)_x|}{\sum_{x \in \mathfrak{D}} |Union_{min}(L)_x|} \qquad dIoU_{min} = \frac{\sum_{x \in \mathfrak{D}} |I_{min}(L)_x|}{\sum_{x \in \mathfrak{D}} |Union_{max}(L)_x|} \tag{16}$$

Aggregated Computation As in Section 2.3.1, the aggregated computation estimates the quantities more efficiently by operating on aggregate values (i.e., sums) per label instead of computing them on a per-sample basis. For example, rather than using $|Union^{sample}_{max}(L)_x| = |N_x| + \min(|E^C_{max}(L)_x|, Top_1(E^C)_x)$ the aggregated formulation uses $|Union^{aggr}_{max}(L)| = N + \min(\sum_{x \in \mathfrak{D}} |E^C_{max}(L)_x|, Top^A_1(E^C))$, where Top^A is computed concept-wise.

2.4 Optimal Algorithm

The optimal algorithm is a best-first search guided by our proposed heuristic. We provide a textual overview of the main steps below and a more detailed discussion and pseudocode in Section C.

1. Compute the exact quantities (Section 2.2) for every concept in the dataset.
2. For each concept, compute $dIoU_{max}$ and $dIoU_{min}$ for all possible paths starting from it, using the heuristic aggregated computation.
3. Initialize the frontier with all paths whose estimated $dIoU_{max}$ is greater than the global maximum of the minimum estimates.
4. Iteratively pop nodes from the frontier, starting from those with the highest estimated $dIoU_{max}$.
   (a) If the estimate is aggregated, refine it by computing the sample-based estimate and reinsert the node. Otherwise, proceed to the next step.
   (b) If the node path is not a final one (i.e., can still be extended), expand its label by concatenating every possible concept with every allowed connective. For each new label, compute $dIoU_{max}$ and $dIoU_{min}$ via the heuristic aggregated computation and insert them into the frontier.
   (c) If the node path is a final one, compute its exact IoU. During this step, the algorithm stores the intermediate quantities of the sub-labels composing the label. This information is then backpropagated to the frontier: nodes that share sub-labels with the evaluated node update their estimates using these exact quantities.
   (d) Continue until the frontier is empty. During the process, whenever a new maximum of the $dIoU_{min}$ estimates is found, prune the frontier by removing all nodes whose $dIoU_{max}$ falls below this threshold.

Additionally, to reduce redundant computation, the algorithm incorporates a limited set of logical equivalence rules, which are applied before and during the expansion phase, and maintains a buffer to cache recently explored nodes associated with the same estimated $dIoU_{max}$. We discuss these

Table 1: Average number of visited, expanded, and estimated nodes, along with runtime per unit (in minutes), by the optimal algorithm, a beam search guided by our heuristic, and two alternative beam search algorithms.

Algorithm	Optimal	Visited	Expanded	Estimated	Time (sec)
Low Complexity					
Optimal (our)	✓	1	100	778	0.08
Beam + Our H.	✗	1	11	405	0.09
MMESH Beam	✗	135	14.94	716	10.01
Vanilla Beam	✗	716	14.94	-	2.77
Intermediate Complexity					
Optimal (our)	✓	1	2885	1.76×10^{6}	69.27
Beam + Our H.	✗	1.94	11	24730	9.46
MMESH Beam	✗	41.94	15	37978	37.31
Vanilla Beam	✗	37979	15	-	450
High Complexity					
Optimal (our)	✓	47.36	$3.54\ \times10^{5}$	1.28×10^{8}	5768
Beam + Our H.	✗	5.82	11	28947	139
MMESH Beam	✗	44.98	13.47	$53774\ \pm 2$	106
Vanilla Beam	✗	53775	13.47	-	5929

details and justify the design choices in Section C. Because the maximum $dIoU$ of each path is an overestimation, and since the algorithm is designed to visit all nodes whose $dIoU$ exceeds that of the current best explanation, **the algorithm is guaranteed to return the most aligned explanation**. In other words, the returned explanation is always the optimal one (proof in Section F).

3 Analysis

3.1 Feasibility of Optimality and Heuristic-Guided Beam Search

This section evaluates the feasibility of the proposed optimal algorithm and the effectiveness of the heuristic when used to guide beam search. We consider three scenarios that vary in annotation complexity: **low, moderate, and high**. The low-complexity setting (Cityscapes (Cordts et al., 2016)) includes a small number of concepts (25), all of which are disjoint. The moderate-complexity setting (the extended version of Ade20K (Zhou et al., 2017b)) includes a much larger number of concepts (847) but with no overlapping annotations. Finally, the high complexity setting (Broden (Bau et al., 2017)) involves frequent overlaps combined with a large number of concepts (1198). As reference baselines, we include the MMESH-guided beam search (La Rosa et al., 2023) and the vanilla beam search (Mu & Andreas, 2020). The beam search variant that uses our heuristic (*Beam + Our H.*), replaces MMESH with label-quantity estimation (see Section D for details). For each setting, we report the average number (over 50 units) of visited nodes (i.e., those for which the exact IoU is computed), expanded nodes, estimated nodes, and the computation time per unit (std. dev. can be found in Section B). Following Mu & Andreas (2020), we extract 50 random units of the last convolutional layer in a ResNet (He et al., 2016) model trained on Places365 (Zhou et al., 2017a) and use the highest activations (top 0.005 percentile) as activation ranges.

Table 1 shows that the optimal algorithm consistently finds the optimal solution within feasible runtimes across all scenarios. As expected, informed beam search methods are faster, since they explore a much smaller portion of the state space (i.e., estimated nodes). However, the performance of the optimal algorithm remains comparable to the vanilla beam search of Mu & Andreas (2020), even in the most complex settings. Importantly, the number of expanded states is a small fraction of the overall state space (less than 0.1%) and is significantly smaller than the number of estimated states. This property is crucial for refining heuristic estimates without compromising runtime efficiency (see Section C).

Table 2: Percentage of changed explanations of explanations falling into categories 1, 2, and 3 for different models.

Dataset	Diff	Beam IoU	Optimal IoU	Cat 1 (%)	Cat 2 (%)	Cat 3 (%)
ResNet	8%	0.077	0.083	85	4	11
AlexNet	22%	0.045	0.047	93	3	4
DenseNet	39%	0.039	0.041	73	0	27

More notably, beam search guided by our heuristic outperforms all baselines across all settings in terms of visited states while achieving comparable or better runtimes. Note that both MMESH and our beam variant converge to the same solutions as those found by the extensive search (Vanilla Beam), since they use admissible heuristics. Compared to MMESH, our approach offers several improvements. First, MMESH relies on annotations during beam expansion, which requires annotations to be kept in memory and GPU resources, thus limiting scalability and parallelization. Conversely, our variant uses annotations only when visiting states, allowing them to be loaded from disk on demand. This design, combined with its higher efficiency, removes the need to store annotations in memory and enables easier parallelization in low-resource scenarios. Our beam variant also scales efficiently with changes in hyperparameters. For example, when varying explanation length (3,5,10, and 20) and beam size (5, 10, and 20) in the moderate settings, the runtime of our variant is stable between 0.16 and 0.18 min/unit. By contrast, MMESH slows down progressively with both explanation length [0.62, 1.28, 4.55, > 240] and beam size [0.62, 1.24, > 240], starting from 0.62 for 3-concepts explanations and beam size fixed to 5 to 4.5 min/units (see Table 6 for a table of results). For explanation lengths or beam sizes higher than 10, MMESH becomes infeasible, despite the limited complexity of the considered settings.

3.2 Explanation Analysis

This section addresses the question we originally posed about beam search–based algorithms: are the explanations they compute optimal? In general, the answer is no. Although beam search often identifies valid explanations, our analysis (Table 2) shows that in the High Complexity settings between 10% and 40% of them differ from the optimal ones across several models studied in previous literature (Mu & Andreas, 2020). We classify these differences into three categories: (1) explanations differ in both concepts and IoU, (2) explanations involve the same concepts but differ in how they are connected, resulting in different IoU, and (3) explanations share the same IoU but differ in the way the concepts are connected.

The first category is the most prominent and often involves explanations connected by AND and AND NOT operators, suggesting that beam search struggles to express explanations that describe units specialized in recognizing complex scenarios. This discrepancy does not imply that the explanations produced by beam search are incorrect; they still capture meaningful alignments between the unit and the concepts expressed in the formulas. However, they often fall short of reflecting the highest degree of alignment that the unit actually exhibits. The second category includes cases where the optimal explanations are more precise and more aligned (e.g., from ((table OR sink) AND white-c) (IoU=0.036) to ((white-c AND table) OR sink) (IoU=0.040)). These differences are small and, in this case, we can consider the beam search explanations quite close and faithful to the optimal explanations.

Finally, the third category includes cases where the semantics of the explanations changes. For example, consider the explanation ((ball_pit-s OR flower) AND NOT dining_room-s). At first glance, one might interpret this unit as being specialized in recognizing $ball\_pit$ not located in dining rooms. However, inspecting the dataset reveals that these concepts never co-occur (i.e., they are disjoint), whereas there are instances including both flowers and dining rooms. Thus, part of the explanation is effectively "unverified". In contrast, the optimal algorithm correctly identifies the alignment as ((flower AND NOT dining_room-s) OR ball_pit-s) and exposes a key limitation of beam search: it cannot backtrack on earlier decisions, and the search may compensate for errors by producing explanations that rely on unverified scenarios (further details and examples for 10 units per model are provided in Section G).

3.3 Algorithm Analysis: Insights and Limitations

This section briefly discusses the key design choices behind the optimal algorithm and provides insights into its limitations.

Overall, the beam search guided by our heuristic represents a safe choice when the reduced time for computation is a priority. Conversely, the optimal algorithm represents a first promising step towards the guarantee of optimality on neural explanations and a better choice when the optimality is considered more important than the time of execution. Regarding design choices, Section C includes a detailed discussion of them. In summary, all the quantities and steps proposed in this algorithm are necessary for the search to be tractable in high complexity scenarios. For example, we use aggregated computations to decide which nodes enter the frontier and sample-based ones to parse it, since estimated nodes far outnumber those actually expanded. Importantly, nodes added to the frontier may still be pruned if their estimated IoU falls below the best found so far. Backpropagation and minimum estimation are crucial in this setting, since they reduce redundant nodes and prevent exhaustive exploration of overestimated candidates. By empirically analyzing the algorithm's execution traces, we identified the following insights and areas of improvement for future research:

Convergence Towards Breath-first search We noted that the state space is explored similarly to breadth-first search, since the top vector dominates the search and roughly overestimates the maximum improvement. In theory, this reliance could make it infeasible for long explanations (although shorter explanations are generally preferred). To mitigate this problem, future research could explore vectors that are (1) specific to a label (which is costly) or (2) novel and better representations of the maximum improvement.

Unmeaningful Units Our algorithm could become much slower when either a unit is not interpretable (the IoU < 0.04 (Bau et al., 2020)) or it is unspecialized (using default rules) and the probing dataset is of high complexity. In these extreme cases, the space to be explored is very large since there is no clear alignment and the combinations of concepts are all similar. One possible solution is to run beam search and then refine the units deemed interpretable for optimality. Alternatively, one could automatically switch from the optimal algorithm to beam search once the frontier grows too large, and use the beam search output to initialize and guide the optimal search.

4 Conclusion

This paper presents the first attempt to guarantee optimality in compositional explanations. Specifically, we identified and formalized the fundamental quantities governing spatial alignment, proposed a heuristic to estimate the potential alignment from any label in the search space, and developed an algorithm capable of computing optimal compositional explanations within feasible runtimes. We further demonstrated that our heuristic can also improve existing beam search–based approaches for non-optimal explanations. Since our method does not rely on spatial information, the proposed heuristic is broadly applicable across domains. Moreover, our theoretical contribution may extend beyond neural explanations, as bitwise computations are of interest in areas such as semantic segmentation and communication. Finally, we call for further research on refining this heuristic and on designing new algorithms for computing alternative forms of compositional explanations.

5 Reproducibility Statement

To ensure reproducibility, we will release the code upon acceptance. Additionally, we describe all the assumptions of this work in Section 2.1 and Section 2.2. The full pseudo-code for both the optimal algorithm and the beam search guided by our heuristic is provided in Section C and Section D. Proofs and derivations of the estimations are given in Section 2.2, Section 2.3.1, and Section 2.3.2 and Sections A and E. We detail the rationale behind design choices in Section C and provide example outputs of our algorithm in Section G. These examples can serve as gold references when re-implementing the algorithm. Finally, we describe the datasets and additional setup information in Section B.

REFERENCES

Yong Hyun Ahn, Hyeon Bae Kim, and Seong Tae Kim. Www: a unified framework for explaining what where and why of neural networks by interpretation of neuron concepts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10968–10977, 2024.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Computer Vision and Pattern Recognition*, 2017.

David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*, 2020. ISSN 0027-8424. doi: 10.1073/pnas.1907375117. URL https://www.pnas.org/content/early/2020/08/31/1907375117.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Kirill Bykov, Laura Kopf, Shinichi Nakajima, Marius Kloft, and Marina Höhne. Labeling neural representations with inverse recognition. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 24804–24828. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/4e52bbb99690d1e05c7ef7b4c8b3569a-Paper-Conference.pdf.

Stephen Casper, Tilman Rauker, Anson Ho, and Dylan Hadfield-Menell. Sok: Toward transparent AI: A survey on interpreting the inner structures of deep neural networks. In *First IEEE Conference on Secure and Trustworthy Machine Learning*, 2023. URL https://openreview.net/forum?id=8C5zt-0Utdn.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Maximilian Dreyer, Erblina Purelku, Johanna Vielhaben, Wojciech Samek, and Sebastian Lapuschkin. Pure: Turning polysemantic neurons into pure features by identifying relevant circuits. *arXiv preprint arXiv:2404.06453*, 2024.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. 2009.

Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tcsZt9ZNKD.

Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, oct 2018. doi: 10.1109/dsaa.2018.00018.

Rafael Harth. *Understanding Individual Neurons of ResNet Through Improved Compositional Formulas*, pp. 283–294. Springer International Publishing, 2022. ISBN 9783031092824. doi: 10.1007/978-3-031-09282-4_24.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. IEEE, 2016.

Evan Hernandez, Sarah Schwettmann, David Bau, Teona Bagashvili, Antonio Torralba, and Jacob Andreas. Natural language descriptions of deep features. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=NudBMY-tzDr.

Robin Hesse, Jonas Fischer, Simone Schaub-Meyer, and Stefan Roth. Disentangling polysemantic channels in convolutional neural networks. In *The First Workshop on Mechanistic Interpretability for Vision*, 2025.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Neha Kalibhat, Shweta Bhardwaj, C Bayan Bruss, Hamed Firooz, Maziar Sanjabi, and Soheil Feizi. Identifying interpretable subspaces in image representations. In *International Conference on Machine Learning*, pp. 15623–15638. PMLR, 2023.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Biagio La Rosa, Leilani H. Gilpin, and Roberto Capobianco. Towards a fuller understanding of neurons with clustered compositional explanations. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=51PLYhMFWz.

Sayo M. Makinwa, Biagio La Rosa, and Roberto Capobianco. Detection accuracy for evaluating compositional explanations of units. In *AIxIA 2021 - Advances in Artificial Intelligence*, pp. 550–563. Springer International Publishing, 2022. doi: 10.1007/978-3-031-08421-8_38.

Riccardo Massidda and Davide Bacciu. Knowledge-driven interpretation of convolutional neural networks. In *Machine Learning and Knowledge Discovery in Databases*, pp. 356–371. Springer International Publishing, 2023. doi: 10.1007/978-3-031-26387-3_22.

Jesse Mu and Jacob Andreas. Compositional explanations of neurons. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. *Visualization for Deep Learning workshop, ICML 2016*, February 2016.

Tuomas Oikarinen and Tsui-Wei Weng. CLIP-dissect: Automatic description of neuron representations in deep vision networks. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=iPWiwWHc1V.

Tuomas Oikarinen and Tsui-Wei Weng. Linear explanations for individual neurons. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=WIbntm28cM.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11), nov 2017. doi: 10.23915/distill.00007.

Laura O'Mahony, Vincent Andrearczyk, Henning Müller, and Mara Graziani. Disentangling neuron representations with concept vectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3769–3774, 2023.

Laura O'Mahony, Nikola S. Nikolov, and David J.P. O'Sullivan. Towards utilising a range of neural activations for comprehending representational associations. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2495–2506. IEEE, February 2025. doi: 10.1109/wacv61041.2025.00248.

Anvita A. Srinivas, Tuomas Oikarinen, Divyansh Srivastava, Wei-Hung Weng, and Tsui-Wei Weng. Sand: Enhancing open-set neuron descriptions through spatial awareness. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2993–3002. IEEE, February 2025. doi: 10.1109/wacv61041.2025.00296.

Andong Wang, Wei-Ning Lee, and Xiaojuan Qi. Hint: Hierarchical neuron concept explainer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10254–10264, June 2022.

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017a.

Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017b. doi: 10.1109/cvpr.2017.544.

Table 3: Visualization of identified quantities for a sample x. In pink the unique extras. In yellow the common extras. In green the unique intersection. In cyan the common intersection.

	Vector						$dIoU$
N(x)	1	1	1	0	0	0	
$\boldsymbol{M}_{c_1}[x]$	1	1	0	0	1	1	2/5
$\boldsymbol{M}_{c_2}[x]$	1	1	0	1	0	0	2/4
$\boldsymbol{M}_{c_3}[x]$	1	0	1	0	1	1	2/5

A Proof Equivalence Decomposed Alignment-IoU Score

Proof. Let $\boldsymbol{M}_L$ be the binary label tensor representing the result of the bitwise logic operations induced by the logic operators connecting single concepts $k \in L$. The Intersection over Union (IoU) score is then defined as:

$$IoU(\boldsymbol{N}, L, \mathfrak{D}, \boldsymbol{M}_L) = \frac{|{}^1\boldsymbol{N} \cap {}^1\theta(\mathbf{M}, L)|}{|{}^1\boldsymbol{N} \cup {}^1\theta(\mathbf{M}, L)|} \tag{17}$$

We first observe that $|{}^1\boldsymbol{N} \cap^1 \boldsymbol{M}_L|$ is necessarily equal to $\sum_{x\in\mathfrak{D}} |I^U(L)_x| + |I^C(L)_x|$. Indeed, all elements in $|{}^1\boldsymbol{N} \cap {}^1\boldsymbol{M}_L|$ are associated with both the neuron and the label L. Consider a generic element $j \in {}^1\boldsymbol{N} \cap {}^1\boldsymbol{M}_L$. Since the logic operators are 00-preserving (Assumption 2), this element must be associated with at least one annotation of one of the concepts in L. But if the element is associated with at least a concept, then it is either associated with exactly one, and thus $j \in I^U$, or with multiple concepts, and thus $j \in I^C$. This also holds in the case of concepts connected by the AND NOT operator, because by design this operator must be chained to a positive concept, and AND NOT is 00-preserving.

Regarding the denominator, we can note that

$$|{}^1\boldsymbol{N} \cup^1 \boldsymbol{M}_L| = |{}^1\boldsymbol{N}| + |{}^1\boldsymbol{M}_L| - |{}^1\boldsymbol{N} \cap^1 \boldsymbol{M}_L| \tag{18}$$

Observe that for all $j \in {}^1\boldsymbol{N} \cap {}^1\boldsymbol{M}_L$, the contribution $|{}^1\boldsymbol{M}_L| - |{}^1\boldsymbol{N} \cap {}^1\boldsymbol{M}_L| = 0$, since these elements are counted in both sets. Hence, $|{}^1\boldsymbol{M}_L| - |{}^1\boldsymbol{N} \cap {}^1\boldsymbol{M}_L|$ represents the number of elements that are labeled as 1 in $\boldsymbol{M}_L$ but for which the neuron does not fire. This definition coincides with Definition 2.4. Therefore, we can rewrite the denominator as

$$|{}^1\boldsymbol{N} \cup {}^1\boldsymbol{M}_L| = |{}^1\boldsymbol{N}| + \sum_{x\in\mathfrak{D}} |E(L)_x|. \tag{19}$$

Similarly to the numerator case, because the logic operators are 00-preserving, every element in $E(L)_x$ is either included in $E^U(L)_x$ or in $E^C(L)_x$. Thus,

$$|{}^1\boldsymbol{N} \cup^1 \boldsymbol{M}_L| = |{}^1\boldsymbol{N}| + \sum_{x\in\mathfrak{D}} \left(|E^U(L)_x| + |E^C(L)_x|\right). \tag{20}$$

and thus

$$dIoU(\boldsymbol{N}, L, \mathfrak{D}, I^C, I^U, E^U, E^C) = IoU(\boldsymbol{N}, L, \mathfrak{D}, \mathbf{M}). \tag{21}$$

□

B Complete Results

Table 4 reports the averages and standard deviations of the results presented in Table 1. Table 6 shows differences in scalability between beam search guided by our proposed heuristic and beam search guided by MMESH. Both tables are computed over 50 units randomly extracted from the penultimate layer of a ResNet18 model trained on Places365, consistent with prior work on compositional explanations (Mu & Andreas, 2020; Harth, 2022; Makinwa et al., 2022; La Rosa et al.,

Table 4: Average number of visited, expanded, and estimated nodes, along with runtime per unit (in minutes) and standard deviation. Statistics are computed across three settings and datasets: low complexity (Cityscapes), intermediate complexity (Ade20K Full), and high complexity (Broden). The table reports results for our two proposed algorithms, the beam search powered by MMESH, and the vanilla beam search.

Algorithm	Optimal	Visited	Expanded	Estimated	Time (sec)
Low Complexity					
Optimal (our)	✓	1	100 $\pm$ 34	778 $\pm$ 221	0.08 $\pm$ 0.01
Beam + Our H.	✗	1 $\pm$ 0.30	11 $\pm$ 0.42	405 $\pm$ 25	0.09 $\pm$ 0.01
MMESH Beam	✗	135 $\pm$ 3	14.94 $\pm$ 0.42	716 $\pm$ 17	10.01 $\pm$ 0.30
Vanilla Beam	✗	716 $\pm$17	14.94 $\pm$ 0.42	-	2.77 $\pm$ 0.16
Intermediate Complexity					
Optimal (our)	✓	1	2885 $\pm$ 2583	1.76×10^6	69.27 $\pm$ 40.60
Beam + Our H.	✗	1.94 $\pm$0.24	11	24730 $\pm$ 1090	9.46 $\pm$ 0.38
MMESH Beam	✗	41.94 $\pm$156	15	37978 $\pm$ 1.79	37.31 $\pm$ 2.87
Vanilla Beam	✗	37979 $\pm$ 1.74	15	-	450 $\pm$ 10
High Complexity					
Optimal (our)	✓	47.36 $\pm$ 99.27	3.54 $\times10^5$	1.28×10^8	5768 $\pm$ 1297
Beam + Our H.	✗	5.82 $\pm$ 6.47	11	28947 $\pm$ 2261	139 $\pm$ 23
MMESH Beam	✗	44.98 $\pm$ 69.11	13.47 $\pm$ 4.54	53774 $\pm$ 2	106 $\pm$ 18
Vanilla Beam	✗	53775 $\pm$ 1	13.47 $\pm$ 4.54	-	5929 $\pm$ 548

Table 5: Percentage of changed explanations of explanations falling into categories 1, 2, and 3 for different models along with average IoU and standard deviation of these explanations.

Dataset	Diff	Beam IoU	Optimal IoU	Cat 1 (%)	Cat 2 (%)	Cat 3 (%)
ResNet	8%	0.077 $\pm$ 0.044	0.083 $\pm$ 0.050	85	4	11
AlexNet	22%	0.045 $\pm$ 0.023	0.047 $\pm$ 0.024	93	3	4
DenseNet	39%	0.039 $\pm$ 0.014	0.041 $\pm$ 0.015	73	0	27

2023). We follow Bau et al. (2020) and Mu & Andreas (2020) and use the highest activations corresponding to the top 0.005 percentile across the probing dataset as the activation range and fix the maximum explanation length to 3. As probing datasets, we used Cityscapes (Cordts et al., 2016) (accessible at: `https://www.cityscapes-dataset.com/` under MIT License) for the low complexity settings, Ade20kFull (Zhou et al., 2017b) (accessible via the Detectron2 (Wu et al., 2019) framwork) for the moderate settings, and Broden (Bau et al., 2017) (accessible at `https://github.com/CSAILVision/NetDissect` under MIT license) for the highest complexity settings. All results were computed on a workstation equipped with an NVIDIA GTX 3090 GPU, without parallelization, in order to avoid timing overhead.

C Optimal Algorithm

This sections describe the optimal algorithm we introduced in the main paper. Algorithm 1 shows the pseudocode of our algorithm. The optimal algorithm is a best-first search guided by our proposed heuristic and consists of the following steps:

1. Compute the exact quantities (Section 2.2) for every concept in the dataset (line 5).
2. For each concept, compute $dIoU_{max}$ and $dIoU_{min}$ for all possible paths starting from it, using the heuristic aggregated computation (line 6).
3. Initialize the frontier with all paths (line 7) and keep track of the greatest $dIoU_{min}$ (line 8). The frontier is sorted by the estimated $dIoU_{max}$.

Table 6: Avg. Time across 50 units (min) per hyperparameters on moderate settings.

Value	Our	MMESH
	Explanation Len	
3	0.16	0.62
5	0.16	1.28
10	0.17	4.55
20	0.18	> 240
	Beam Size	
5	0.16	0.62
10	0.17	1.24
20	0.18	> 240

4. Reduce the frontier by removing nodes whose estimated $dIoU_{max}$ is lower than the global maximum of the minimum estimates (line 10).
5. Iteratively pop nodes from the frontier, starting from those with the highest estimated $dIoU$ (lines 11-12).
   (a) If the estimate is aggregated, refine it by computing the sample-based estimate and reinsert the node (lines 13-20). Otherwise, proceed to the next step.
   (b) Apply logical equivalence rules when possible (lines 21-26), recompute the sample-based estimate for the equivalent expression, and reinsert the node if the estimate has changed. Otherwise, proceed to the next step. Currently, we only check distributive properties as logical equivalences. Note that even though the expressions are logically equivalent, their estimates may differ due to overestimation. The goal of this step is to select the form with the smallest overestimation among all possible equivalents.
   (c) Check whether the same node has been recently explored by comparing its maximum IoU with the most recent one. If they match but the node has not yet been explored, add it to memory; if it has already been explored, skip it. Otherwise, clear the memory and initialize the most recent IoU with the node's maximum IoU (lines 27–38).
   (d) If the node is not a final one (i.e., can still be extended), expand its label by concatenating every possible concept with every allowed connective (line 49). For each new label, compute $dIoU_{max}$ and $dIoU_{min}$ via the heuristic aggregated computation and insert them into the frontier (lines 50-51). If a new maximum is found among $dIoU_{min}$ of the new paths, update the global maximum and reduce the frontier (lines 52-55).
   (e) If the node is a final one, compute its exact IoU (line 42). During this step, the algorithm stores the intermediate quantities of the sub-labels composing the label (line 40). This information is then backpropagated to the frontier: nodes that share sub-labels with the evaluated node update their estimates using these exact quantities (line 41). If the IoU is greater than that of the best label found so far, update the best label with the current node (lines 43–45).
   (f) Continue until the frontier is empty (line 11).

In the following, we provide the rationale behind several design choices made during the development of this algorithm.

Memory Mechanism The memory mechanism (lines 27–38) was introduced to account for logical equivalences not handled elsewhere in the algorithm and to prevent redundant computation. We considered several alternatives, but this solution proved to be the least computationally expensive. Without such a mechanism, some logically equivalent rules could be expanded multiple times (lines 49–55), leading to a significantly larger search space. Alternative options would be to check for logical equivalences or the presence of a node directly when adding nodes to the frontier. However, this would make exploration prohibitively expensive: verifying logical equivalence or membership would offset the efficiency gains of the heuristic. In addition, since the frontier is implemented as

a heap, checking whether a node is already present is slower than with a sorted list. Maintaining a sorted frontier would require re-sorting on every insertion, which would be too costly and therefore infeasible.

Concept Quantities A key design choice concerns the handling of concept quantities. We store quantities only for atomic concepts and do not cache them when estimating (lines 6, 14, and 50) or computing the $dIoU$ (line 40). Consequently, the intermediate quantities of labels are recomputed multiple times during the search. This decision was made for two main reasons. First, storing additional quantities increases access time, which is critical because the algorithm frequently accesses these values; this overhead could easily exceed the time required to recompute them from scratch. Second, caching more quantities increases memory usage, potentially limiting the ability to run parallel processes. Given the substantial runtime required for the most complex datasets, preserving the ability to parallelize is essential. Moreover, avoiding extensive caching keeps the approach lightweight in terms of memory and resource requirements. In conclusion, the marginal gains from more precise estimations are outweighed by the overhead, especially given that the algorithm tends to converge toward a breadth-first search (Section 3.3).

Backpropagation This process was introduced to reduce the number of visited states. While it has no (or bad) impact on low and moderate-complexity settings, it significantly reduces runtime in high complexity scenarios, especially when running on CPUs. Specifically, the frontier in the later stages of the search often contains similar explanations that differ only in the last added concept (e.g., (A AND B) OR C) and (A AND B) OR D). These nodes typically appear in the frontier because the left-hand terms provide a rough overestimation relative to the current maximum. When a node is visited, this overestimation is temporarily corrected, and the backpropagation step updates the estimates of all remaining nodes. This correction helps the algorithm to avoid visiting unpromising states. On average, 2000–3000 nodes per unit are updated via backpropagation, highlighting the importance of this mechanism.

Sample vs Aggregated Computation The optimal algorithm leverages both sample-based and aggregated computations for path and label estimations. This design is necessary due to the large state space and the overestimation introduced by the aggregated computation. We explored several alternatives during the development of this work, but this trade-off is the only one that ensures feasibility in high complexity settings. As explained in the main text, sample computation requires $|\mathfrak{D}|$ comparisons per calculation, whereas aggregated computation relies solely on the sum of concept quantities computed at the first step of the algorithm, combining them in a single operation. These two approaches represent a trade-off between precision and efficiency: the sample version is more accurate but computationally intensive, while the aggregated version is faster but less precise. In practice, using only aggregated computation for both node expansion and frontier exploration would yield faster per-node computations but result in a frontier that is orders of magnitude larger than that explored by our combined approach. Conversely, using only sample computation slightly reduces the frontier (by roughly 50,000 nodes per unit in preliminary experiments) but incurs significantly higher computation time per node, effectively nullifying the gains from the smaller frontier.

MinIoU As explained previously, the algorithm computes both the minimum and maximum IoU for each label and path. This increases the time required per estimation, since the maximum and minimum calculations are distinct and rarely share terms, effectively doubling the computation time per node. An alternative design could omit the MinIoU computation and explore only nodes whose maximum IoU exceeds the best visited so far. However, without the MinIoU, it becomes impossible to dynamically reduce the frontier at runtime: the frontier can only grow, and the only way to remove a single node is to fully explore it. This would result in a frontier orders of magnitude larger than the one explored by the proposed design, especially in the early phase of the search, when the MinIoU is updated multiple times and allows significant pruning. Future work could explore adaptive strategies that decide dynamically whether to compute MinIoU based on the search space or external information, potentially improving overall runtime.

Code Optimization In addition to the previously mentioned design choices, we implemented several code optimizations to avoid unnecessary computation of quantities and to handle logical equivalences. For instance, we enforce an order on concept indices when chaining two consecutive

applications of the same operator during node expansion. For example, for a concept with index 10, it can only be chained with concepts having an index greater than 10, since all other combinations will be captured when expanding nodes with lower indices. The same rule applies to consecutive operators of the same type. For example, if we have $(3 \text{ OR } 15)$, we can only chain concepts with indices greater than 15 when applying the OR operator again (e.g., $(3 \text{ OR } 15) \text{ OR } 18$), while we are free to choose any concept for other operators (AND or AND NOT). Other code optimizations involve avoiding the computations of paths when the intersection of the right or left sides is 0 and avoiding the sample computation related to equations involving $Bott_1$ in datasets where this vector is always 0 (see Section E)

C.1 Factors Impacting the Runtime

As discussed in the main text, the primary factor influencing runtime is the degree of neuron alignment: the more aligned a neuron is, the shorter the runtime, since the heuristic can prune the search space more effectively. Beyond alignment, other factors can also affect runtime, including the number and type of concepts and the resulting explanation length.

Explanation length matters because the exploration strategy empirically resembles a breadth-first search, due to the coarse approximation of the maximum possible improvement along a path. As a result, it inherits the typical limitations of breadth-first exploration. The type of concept also plays a role: overly general concepts that are widely distributed across the dataset (e.g., colors) tend to overlap with many concepts and many images, forcing the algorithm to compute expensive approximations for all of them. Similarly, when a neuron responds to a general concept (e.g., bird) but admits many specializations (e.g., different colors), these specializations often have highly similar IoUs, preventing effective pruning and therefore increasing runtime.

The same dynamic arises in extreme cases where a concept covers nearly the entire input across most samples. Fortunately, such broad concepts (e.g., color, shape, or very general categories) are few in number and rarely useful in practice when they are so widely shared within the dataset.

D Beam Search Algorithm

In this section, we provide a high-level description of the beam search guided by our heuristic and present its pseudo-code in Algorithm 2. This algorithm replaces the MMESH heuristic (La Rosa et al., 2023) with the label quantity estimates introduced in Section 2.2, within a standard heuristic-guided beam search framework. The procedure begins by following the initial two steps of the optimal algorithm but keeps only the best b concepts to form the initial beam. It then proceeds iteratively for n steps: at each iteration, the algorithm expands the nodes in the current beam, sorts the resulting state space according to the heuristic, and evaluates candidate nodes by computing their IoU. The top b nodes are then selected to form the next beam. The process terminates when no candidate improves the current best IoU^{max}, or when no nodes remain to be expanded or visited. Differently from the optimal algorithm, this algorithm relies solely on sample computations and does not make use of, or estimate, the paths introduced in Section 2.3.2. Beyond the advantages discussed in Section 3.1, our heuristic does not rely on spatial information and can therefore be applied across multiple domains without requiring modifications to the code or formulation.

E Estimating Quantities

This section presents the rationale and derivation of all estimations computed by our heuristic. We divide the discussion into per-sample estimations and aggregated computations.

E.1 Sample Computation

E.1.1 Label Quantities

In the following, we derive the estimations of the label quantities introduced in the main text for the sample-based computation. For clarity, we group the equations by operator and discuss each

operator separately. Because the quantities for the unique elements are exact and are directly derived from the definition and Observation 1, here we focus on the derivation of the common quantities.

$$|I^C_{min}(L)_x| = \max(|I^C_{min}(L_\leftarrow)_x|, |I^C(L_\rightarrow)_x|) \tag{22}$$
$$|I^C_{max}(L)_x| = \min(|I^C_{max}(L_\leftarrow)_x| + |I^C(L_\rightarrow)_x|, |N^C_x|) \tag{23}$$
$$|E^C_{min}(L)_x| = \max(|E^C_{min}(L_\leftarrow)_x|, |E^C(L_\rightarrow)_x|) \tag{24}$$
$$|E^C_{max}(L)_x| = \min(|E^C_{max}(L_\leftarrow)_x| + |E^C(L_\rightarrow)_x|, |SE^C_x|) \tag{25}$$

Derivation Equations (22) *to* (25) *for the OR operator*: We can start by noting that Equation (22) corresponds to the case where one side is a subset of the other. In this case, the equation gives the maximum cardinality of the intersection between the left and right sides, since the minimum elements are already included in the maximum and the OR is 1-preserving (i.e., the number of ones cannot be lower than before the combination). Conversely, Equation (23) corresponds to the case where the sides are disjoint in their extras, adjusted by the $|N^C_x|$ quantity. Indeed, because the left side is an overestimation, it may happen that the sum exceeds the limits, requiring readjustment using $|N^C_x|$. The estimation of the maximum and minimum extras follows the same reasoning, with the only difference that the common space extra $|SE^C_x|$ is used to adjust the quantity of the maximum extras.

$$|I^C_{min}(L)_x| = \max(|I^C_{min}(L_\leftarrow)_x| + |I^C(L_\rightarrow)_x| - |N^C_x|, 0) \tag{26}$$
$$|I^C_{max}(L)_x| = \min(|I^C_{max}(L_\leftarrow)_x|, |I^C(L_\rightarrow)_x|) \tag{27}$$
$$|E^C_{min}(L)_x| = \max(|E^C_{min}(L_\leftarrow)_x| + |E^C(L_\rightarrow)_x| - |SE^C_x|, 0) \tag{28}$$
$$|E^C_{max}(L)_x| = \min(|E^C_{max}(L_\leftarrow)_x|, |E^C(L_\rightarrow)_x|) \tag{29}$$

Derivation Equations (26) *to* (29) *for the AND operator*: In this case, the AND operator is 0-preserving. Therefore, when computing the minimum, we consider the scenario where a guaranteed overlap (i.e., both sides equal to 1) occurs. This happens when the sum of the two sides exceeds the maximum available space, represented by $|N^C_x|$ and $|SE^C_x|$, respectively. In such cases, the minimum guaranteed overlap is given by the difference between the sum and the cardinality of the available space. In all other cases, the best estimation we can provide is simply 0. The computation of the maximum corresponds to the case of fully overlapping concepts. Since the operator is 0-preserving, the equation in this case selects the minimum of the two sides for each sample.

$$|I^C_{min}(L)_x| = \max(|I^C_{min}(L_\leftarrow)_x| - |I^C(L_\rightarrow)_x|, 0) \tag{30}$$
$$|I^C_{max}(L)_x| = \min(|I^C_{max}(L_\leftarrow)_x|, |N^C_x| - |I^C(L_\rightarrow)_x|) \tag{31}$$
$$|E^C_{min}(L)_x| = \max(|E^C_{min}(L_\leftarrow)_x| - |E^C(L_\rightarrow)_x|, 0) \tag{32}$$
$$|E^C_{max}(L)_x| = \min(|E^C_{max}(L_\leftarrow)_x|, |SE^C_x| - |E^C(L_\rightarrow)_x|) \tag{33}$$

Derivation Equations (30) *to* (33) *for the AND NOT operator*: This operator behaves like the AND operator but combines a negated concept, flipping all the bits in the corresponding vectors. The maximum estimations follow the same reasoning as for the AND operator, except that in this case $|N^C_x| - |I^C(L_\rightarrow)_x|$ and $|SE^C_x| - |E^C(L_\rightarrow)_x|$ represent the bits that were originally 0 and are now 1, corresponding to the common intersection and the common extras of the negated concept. The minimum estimation corresponds to the case where the right side fully overlaps with the left side. In this case, due to the negation, all overlapping bits flip to 0. Since the AND operator is 0-preserving, this results in the loss of all left side elements shared with the right side.

E.1.2 Path Quantities

In the following, we derive the estimations of the path quantities introduced in the main text. For clarity, we group the equations by operator and discuss each operator separately. Note that we assume, for all the paths, that at least 1 concept is added to the label, since the case where no

concepts are added is covered by the "final path" obtained by applying Definition 2.5 to the label quantities. In all the following equations, t denotes the difference between the maximum length and the length of the label L.

$$|I_{min}(L)_x| = \max(|I^C_{min}(L)_x| + |I^U_{min}(L)_x|, Bott_1(I^C)_x + Bott_1(I^U)_x) \tag{34}$$

$$|I_{max}(L)_x| = \min(|I^C_{max}(L)_x| + Top_t(I^C)_x, |N^C_x|) + \min(|I^U_{max}(L)_x| + Top_t(I^U)_x, |N^U_x|) \tag{35}$$

$$|Union_{min}(L)_x| = |\boldsymbol{N}_x| + \max(|E^C_{min}(L)_x| + |E^U_{min}(L)_x|, Bott_1(E^C)_x + Bott_1(E^U)_x) \tag{36}$$

$$\begin{aligned} |Union_{max}(L)_x| = &|\boldsymbol{N}_x| + \min(|E^C_{max}(L)_x| + Top_t(E^C)_x, |SE^C_x|) \\ &+ \min(|E^U_{max}(L)_x| + Top_t(E^U)_x, |SE^U_x|) \end{aligned} \tag{37}$$

Derivation of Equations (34) *to* (37) *for the OR operator*: Equation (34) and Equation (36) follow the same derivation as Equations (22) and (24). In this case, the added concept is represented by $Bott_1(I^C)$, which corresponds to the minimum intersection of any concept in a given sample and reflects the case of fully overlapping concepts. In practice, for most datasets, this quantity is always equal to 0; thus, it reduces to $|I^C_{min}(L)_x| + |I^U_{min}(L)_x|$. The same reasoning applies to $|Union_{min}(L)_x|$ and the extras. Note also that the label quantities can be lower than the bottom values, since they represent combinations of concepts that may further reduce these quantities. Finally, the maximum quantities correspond to the case where the concepts included in L are disjoint from those cumulated in the Top vectors. In this situation, the quantities are simply the sum of neuron activations, maximum quantities, and the Top vectors, adjusted by the available space $|N^U_x|$ and $|SE^U_x|$, respectively.

$$|I_{min}(L)_x| = 0 \tag{38}$$

$$|I_{max}(L)_x| = \min(|I^C_{max}(L)_x|, Top_1(I^C)_x) \tag{39}$$

$$|Union_{min}(L)_x| = |\boldsymbol{N}_x| \tag{40}$$

$$|Union_{max}(L)_x| = |\boldsymbol{N}_x| + \min(|E^C_{max}(L)_x|, Top_1(E^C)_x) \tag{41}$$

Derivation Equations (38) *to* (41) *for the AND operator*: This operator is simpler to derive. Specifically, Equations (26) and (28) corresponds to the case where the label and all the concepts included in the Top vectors are disjoint, and thus all the quantities reduce to 0. Conversely, Equation (40) and Equation (41) represent the case where the label fully overlaps with the quantities stored in Top_1. Therefore, for the AND operator, only the quantities from the smaller concept are preserved per sample. Note that Top_1 is the only applicable Top vector here, since higher indices sum the cardinality of multiple concepts, which would violate the operations defined by the AND operator.

$$|I_{min}(L)_x| = |I^U_{min}(L)_x| \tag{42}$$

$$|I_{max}(L)_x| = |I^U_{max}(L)_x| + \min(|I^C_{max}(L)_x|, |N^C_x| - Bott_1(I^C)_x) \tag{43}$$

$$|Union_{min}(L)_x| = |\boldsymbol{N}_x| + |E^U_{min}(L)_x| \tag{44}$$

$$|Union_{max}(L)_x| = |\boldsymbol{N}_x| + |E^C_{max}(L)_x| + \min(|E^U_{max}(L)_x|, |SE^C_x| - Bott_1(E^C)_x), \tag{45}$$

Derivation Equations (42) *to* (45) *for the AND NOT operator*: Equations (30) and (32) corresponds to the same case as in Equations (26) and (28). However, since the AND NOT operator preserves all unique elements (Observation 1), the minimum intersection corresponds to the minimum unique intersection of the label, and the minimum union includes the minimum unique extras. Conversely, Equations (44) and (45) represents the case where the label fully overlaps with the negated concept, represented by $|N^C_x| - Bott_1(I^C)_x$ and $|SE^C_x| - Bott_1(E^C)_x$, respectively. As previously noted, in practice, for most datasets, $Bott_1$ is always equal to 0. Thus, the equations simplify to $|I^U_{max}(L)_x| + |I^C_{max}(L)_x|$ for the intersection and $|\boldsymbol{N}_x| + |E^C_{max}(L)_x| + |E^U_{max}(L)_x|$ for the union, since by definition $|I^C_{max}(L)_x| < |N^C_x|$ and $|E^U_{max}(L)_x| < |SE^C_x|$ due to the fact that $|N^C_x|$ represents the total neuron common space and $|SE^C_x|$ represents the total available space for common extras.

E.2 AGGREGATED COMPUTATION

This section describes the aggregated computation of the common quantities. To improve readability, we shorten the notation $\sum_{x \in \mathfrak{D}}$ to simply $\sum$, since there are no ambiguities and the summation is used exclusively for this purpose.

E.2.1 LABEL QUANTITIES

In the case of disjoint concepts, we can use the same equation as in the sample-based case described in Section 2.3.1, since these are already aggregated. For the common quantities, the modification consists of pre-computing the aggregate value per label rather than per sample. This requires computing the dataset-wide sum only for atomic concepts, while all higher-arity labels can be derived through arithmetic operations over these sums. Therefore:

$$|I^C_{min}(L)_x| = \begin{cases} \min(\sum |I^C_{min}(L_{\leftarrow})_x|, \sum |I^C(L_{\rightarrow})_x|) & \text{if } \oplus = OR \\ \max(\sum |I^C_{min}(L_{\leftarrow})_x| + \sum |I^C(L_{\rightarrow})_x| - |N^C|, 0) & \text{if } \oplus = AND \\ \max(\sum |I^C_{min}(L_{\leftarrow})_x| - \sum |I^C(L_{\rightarrow})_x|, 0) & \text{if } \oplus = AND\ NOT \end{cases} \tag{46}$$

$$|I^C_{max}(L)_x| = \begin{cases} \min(\sum |I^C_{max}(L_{\leftarrow})_x| + \sum |I^C(L_{\rightarrow})_x|, |N^C|) & \text{if } \oplus = OR \\ \min(\sum |I^C_{max}(L_{\leftarrow})_x|, \sum |I^C(L_{\rightarrow})_x|) & \text{if } \oplus = AND \\ \min(\sum |I^C_{max}(L_{\leftarrow})_x)|, |N^C| - \sum |I^C(L_{\rightarrow})_x)| & \text{if } \oplus = AND\ NOT \end{cases} \tag{47}$$

$$|E^C_{min}(L)_x| = \begin{cases} \max(\sum |E^C_{min}(L_{\leftarrow})_x|, \sum |E^C(L_{\rightarrow})_x|) & \text{if } \oplus = OR \\ \max(\sum |E^C(L_{\leftarrow})_x| + \sum |E^C(L_{\rightarrow})_x| - |SE^C|, 0) & \text{if } \oplus = AND \\ \max(\sum |E^C_{min}(L_{\leftarrow})_x| - \sum |E^C(L_{\rightarrow})_x|, 0) & \text{if } \oplus = AND\ NOT \end{cases} \tag{48}$$

$$|E^C_{max}(L)_x| = \begin{cases} \min(\sum |E^C_{max}(L_{\leftarrow})_x| + \sum |E^C(L_{\rightarrow})_x|, |SE^C|) & \text{if } \oplus = OR \\ \min(\sum |E^C_{max}(L_{\leftarrow})_x|, \sum |E^C(L_{\rightarrow})_x|) & \text{if } \oplus = AND \\ \min(\sum |E^C_{max}(L_{\leftarrow})_x|, |SE^C| - \sum |E^C(L_{\rightarrow})_x|) & \text{if } \oplus = AND\ NOT \end{cases} \tag{49}$$

The derivation of these quantities follows the same rationale as the sample computation, and the only difference is the larger overestimation produced by the aggregation of label-wise quantities.

E.2.2 PATH QUANTITIES

Similarly to the sample-based computation, computing the path $dIoU$ requires estimating the maximum possible improvement. In this case, however, the Top^A and $Bott^A$ vectors are computed concept-wise rather than per sample. Specifically, for each quantity, we sort the values of all individual concepts in the dataset and compute cumulative sums into the Top^A and $Bott^A$ vectors, starting from the largest and smallest values, respectively. Substituting these into the equations of

the sample-based computation, we obtain:

$$I_{min}(L) = \begin{cases} \max(\sum |I^C_{min}(L)_x| + \sum |I^U_{min}(L)_x|, & \\ \quad Bott^A_1(I^C) + Bott^A_1(I^U)) & \text{OR Path} \\ 0 & \text{AND Path} \\ \sum |I^U_{min}(L)_x| & \text{AND NOT Path} \end{cases} \tag{50}$$

$$I_{max}(L) = \begin{cases} \min(\sum |I^C_{max}(L)_x| + Top^A_t(I^C), |N^C|) + & \\ \quad \min(\sum |I^U_{max}(L)_x| + Top^A_t(I^U), |N^U|) & \text{OR Path} \\ \min(\sum |I^C_{max}(L)_x|, Top^A_1(I^C)) & \text{AND Path} \\ \sum |I^U_{max}(L)_x| + \min(\sum |I^C_{max}(L)_x|, & \\ \quad |N^C| - Bott^A_1(I^C)) & \text{AND NOT Path} \end{cases} \tag{51}$$

$$|Union_{min}(L)| = |^1\boldsymbol{N}| + \begin{cases} \max(\sum |E^C_{min}(L)_x| + \sum |E^U_{min}(L)_x|, & \\ \quad Bott^A_1(E^C) + Bott^A_1(E^U)) & \text{OR Path} \\ \max(\sum |E^C_{min}(L)_x| + Bott^A_1(E^C) - |SE^C|, 0) & \text{AND Path} \\ \sum |E^C_{min}(L)_x| & \text{AND NOT Path} \end{cases} \tag{52}$$

$$|Union_{max}(L)| = |^1\boldsymbol{N}| + \begin{cases} \min(\sum |E^C_{max}(L)_x| + Top^A_t(E^C), |SE^C|) & \\ \quad + \min(\sum |E^U_{max}(L)_x| + Top^A_t(E^U), |SE^U|) & \text{OR Path} \\ \min(\sum |E^C_{max}(L)_x|, Top^A_1(E^C)) & \text{AND Path} \\ \sum |E^C_{max}(L)_x| + \min(& \\ \quad \sum |E^U_{max}(L)_x|, |SE^C| - Bott^A_1(E^C)) & \text{AND NOT Path} \end{cases} \tag{53}$$

The derivation of these quantities follows the same rationale as the sample computation case, and the only difference is that these represent a larger overestimation. However, note in this case $|Union_{min}(L)|$ can be greater than 0, unlike in the sample computation where $\forall x \in \mathfrak{D}$, $Bott_1(E^C)_x = 0$, except in the rare degenerate case (not observed in any of the datasets tested in this paper) where a single sample contains all concepts in the dataset.

E.3 Path Combinations

As in the previous section, we shorten here the notation $\sum_{x\in\mathfrak{D}}$ to simply $\sum$. As mentioned in the main text, to estimate values for paths involving multiple operators, we select the maximum and minimum of each quantity across the operators considered. For example, when both OR and AND can appear along a path, we compute:

$$dIoU_{max}(OR, AND)_x = \frac{\max(\sum |I^{OR}_{max}(L)_x|, \sum |I^{AND}_{max}(L)_x|)}{\min(\sum |Union^{OR}_{min}(L)_x|, \sum |Union^{AND}_{min}(L)_x|)} \tag{54}$$

and

$$dIoU_{min}(OR, AND)_x = \frac{\min(\sum |I^{OR}_{min}(L)_x|, \sum |I^{AND}_{min}(L)_x|)}{\max(\sum |Union^{OR}_{max}(L)_x|, \sum |Union^{AND}_{max}(L)_x|)} \tag{55}$$

These expressions simplify to:

$$dIoU_{max}(OR, AND)_x = \frac{\sum |I^{OR}_{max}(L)_x|}{\sum |Union^{AND}_{min}(L)_x|} \tag{56}$$

and

$$dIoU_{min}(OR, AND)_x = \frac{\sum |I^{AND}_{min}(L)_x|}{\sum |Union^{AND}_{max}(L)_x|} \tag{57}$$

We can further rewrite them as:

$$dIoU_{max}(OR, AND)_x = \frac{\min(|I^C_{max}(L)_x| + Top_t(I^C)_x, |N^C_x|) + \min(|I^U_{max}(L)_x| + Top_t(I^U)_x, |N^U_x|)}{|\boldsymbol{N}_x|} \tag{58}$$

and

$$dIoU_{min} = 0 \tag{59}$$

Now, let us consider the case where we explicitly design this combined quantity. As before, we observe that $dIoU_{min} = 0$. For the numerator, however, we can refine the estimation: including an AND operator at any step will, by design, remove all the unique quantities of the current label. Thus:

$$dIoU_{max}(OR, AND) = \frac{\min(|I^C_{max}(L)_x| + Top_t(I^C)_x, |N^C_x|) + Top_t(I^U)_x}{|\boldsymbol{N}_x| + (|E^C_{max}(L)_x| + |Bott_1(E^C)_x| - |SE^C_x|} \tag{60}$$

However, since $|Bott_1(E^C)_x|$ is 0 in most datasets, the denominator reduces to $|\boldsymbol{N}_x|$. Thus, the only practical difference lies in the numerator, where the $|I^U_{max}(L)_x|$ term is missing. However, in general, $Top_t(I^U)_x$ is much larger than $|I^U_{max}(L)_x|$, since it includes the maximum per sample across all concepts in the dataset. This makes the difference between $dIoU^{(OR,AND)}_{max}$ (from explicit design) and our proposed simplification relatively small. Similar observations can be made for all the other combinations of the operators considered in this paper.

From a practical perspective, and given that the optimal algorithm often converges toward breadth-first exploration (Section 3.3), the gain of the refined design is marginal. Conversely, our simplified approach, based on combining the values of exclusive paths of operators, scales more efficiently and facilitates future extensions. Indeed, new logic operators can be incorporated into the optimal algorithm simply by providing their estimates for the exclusive path.

F Proof of Optimality

This section builds upon the derivations introduced in section E to demonstrate that the heuristics employed are admissible, ensuring the algorithm is guaranteed to find the optimal solution and is complete. This section assumes the reader to be familiar with the terminology introduced in Section 2.1.

F.1 Proof of Admissibility of the Label Quantities Heuristic

Let L be the label obtained by concatenating the label $L_{\leftarrow}$ with the concept $L_{\rightarrow}$. This heuristic aims at estimating the IoU score of L. By Equation (4), the IoU score is equivalent to the decomposed $dIoU$ score. To be admissible, the heuristic cannot underestimate the numerator or overestimate the denominator. Thus, it must satisfy the following constraints:

$$\begin{cases} |I^U_{max}(L)_x| + |I^C_{max}(L)_x| \geq |I^U(L)_x| + |I^C(L)_x| & (61) \\ 0 \leq |E^U_{min}(L)_x| + |E^C_{max}(L)_x| \leq |E^U(L)_x| + |E^C(L)_x| & (62) \end{cases}$$

for all $x \in \mathfrak{D}$. Equation 61 ensures that the heuristic returns an optimistic estimate of the intersection (numerator), while Equation 62 ensures a pessimistic estimate of the denominator of eq. (3).

We observe that unique elements are always disjoint and cannot be shared by definition 2.1. Therefore, their value can be computed exactly by using Observation 1, and thus $|I^U_{max}(L)_x| = |I^U(L)_x|$ and $|E^U_{min}(L)_x| = |E^U(L)_x|$.

Therefore, the constraints reduce to:

$$\begin{cases} |I^C_{max}(L)_x| \geq |I^C(L)_x| & (63) \\ |E^C_{min}(L)_x| \leq |E^C(L)_x| & (64) \end{cases}$$

where the lower bound in the second inequality is omitted because cardinalities are non-negative.

These constraints are satisfied by design because of the derivations described in the previous section. Namely, if $L_{\leftarrow}$ and $L_{\rightarrow}$ are disjoint, then they share no common elements, and the algorithm can compute the exact values. Thus $|I^C_{max}(L)_x| = |I^C(L)_x|$ and $|E^C_{min}(L)_x| \leq |E^C(L)_x|$. Note that in this case, the maximum and the minimum of these quantities coincide.

For the AND NOT operator, both the unique and the common elements reduce to those of the left side $L_{\leftarrow}$. Therefore, $dIoU(L) = dIoU(L_{\leftarrow})$, and any path reachable from L is also reachable from $L_{\leftarrow}$ since they are equivalent.

Conversely, if $L_{\leftarrow}$ and $L_{\rightarrow}$ overlap, we analyze each operator $\oplus$.

Case OR operator. For the intersection, recall that

$$|I^C_{max}(L)_x| = \min\left(|I^C_{max}(L_{\leftarrow})_x| + |I^C(L_{\rightarrow})_x|,\ |N^C_x|\right) \tag{65}$$

If the sets overlap, then

$$|I^C(L)_x| = |I^C(L_{\leftarrow})_x| + |I^C(L_{\rightarrow})_x| - |I^C(L_{\leftarrow})_x \cap I^C(L_{\rightarrow})_x| \tag{66}$$

Because they overlap, they share at least one element, and thus

$$|I^C(L_{\leftarrow})_x \cap I^C(L_{\rightarrow})_x| \geq 1. \tag{67}$$

This implies

$$|I^C(L)_x| \leq |I^C(L_{\leftarrow})_x| + |I^C(L_{\rightarrow})_x| \tag{68}$$

Furthermore, by the definition of N^C_x,

$$|I^C(L)_x| \leq |N^C_x| \tag{69}$$

Thus $|I^C_{max}(L)_x| \geq |I^C(L)_x|$.

For the extras, recall that

$$|E^C_{min}(L)_x| = \max(|E^C_{min}(L_{\leftarrow})_x|, |E^C(L_{\rightarrow})_x|) \tag{70}$$

If the sets overlap, then

$$|E^C(L)_x| = |E^C(L_{\leftarrow})_x \cup E^C(L_{\rightarrow})_x| \tag{71}$$

Because the cardinality of a set cannot exceed the cardinality of its union with another set, Equation (64) is satisfied regardless of which element is selected by the max operator.

Case AND operator. In this case, for the intersection,

$$|I^C_{\max}(L)_x| = \min\left(|I^C_{\max}(L_{\leftarrow})_x|,\ |I^C(L_{\rightarrow})_x|\right) \tag{72}$$

The true value is

$$|I^C(L)_x| = |I^C(L_{\leftarrow})_x \cap I^C(L_{\rightarrow})_x| \tag{73}$$

since the AND operator is 0-preserving. Because an intersection cannot exceed either operand, then

$$|I^C_{\max}(L)_x| \geq |I^C(L)_x| \tag{74}$$

Regarding the extras, recall that

$$|E^C_{min}(L)_x| = \max(|E^C_{min}(L_{\leftarrow})_x| + |E^C(L_{\rightarrow})_x| - |SE^C_x|, 0) \tag{75}$$

The zero case is proven by the non-negativity of the cardinality. Conversely, if the max operator selects the first factor, then we can prove it never overestimates the true value by induction by fixing $L_{\leftarrow}$ to an atomic concept. In this case, we have access to the exact computation and thus

$$|E^C_{min}(L)_x| = \max(|E^C(L_{\leftarrow})_x| + |E^C(L_{\rightarrow})_x| - |SE^C_x|, 0) \tag{76}$$

If the sets overlap, then the true value is

$$|E^C(L)_x| = |E^C(L_{\leftarrow})_x \cap E^C(L_{\rightarrow})_x| \tag{77}$$

Recall that by the definition of SE^C, any sum between the cardinality of an extra disjoint set is lower than the cardinality of SE^C. Formally,

$$|SE^C_x| \geq |E^C(L_j)_x| + |E^C(L_k)_x|\ ,\ \forall L_j, L_k \in \mathfrak{L}^1 | E^C(L_j)_x \cap E^C(L_k)_x = \emptyset \tag{78}$$

We rewrite $|E^C(L_\leftarrow)_x|$ and $|E^C(L_\rightarrow)_x|$:

$$|E^C(L_\leftarrow)_x| = |E^C(L_\leftarrow)_x \setminus E^C(L_\rightarrow)_x| + |E^C(L_\leftarrow)_x \cap E^C(L_\rightarrow)_x| \tag{79}$$

$$|E^C(L_\rightarrow)_x| = |E^C(L_\rightarrow)_x \setminus E^C(L_\leftarrow)_x| + |E^C(L_\leftarrow)_x \cap E^C(L_\rightarrow)_x| \tag{80}$$

Since the sets $E^C(L_\leftarrow)_x \setminus E^C(L_\rightarrow)_x$, $E^C(L_\rightarrow)_x \setminus E^C(L_\leftarrow)_x$, and $E^C(L_\leftarrow)_x \cap E^C(L_\rightarrow)_x$ are pairwise disjoint, then:

$$|E^C(L_\leftarrow)_x \setminus E^C(L_\rightarrow)_x| + |E^C(L_\rightarrow)_x \setminus E^C(L_\leftarrow)_x| + |E^C(L_\leftarrow)_x \cap E^C(L_\rightarrow)_x| \leq |SE^C_x| \tag{81}$$

Combining Equation (81) with Equation (76) and the fact that the operator selected the first argument, we obtain

$$\begin{aligned} |E^C(L_\leftarrow)_x| + |E^C(L_\rightarrow)_x| - |SE^C_x| = |E^C(L_\leftarrow)_x \setminus E^C(L_\rightarrow)_x| + |E^C(L_\leftarrow)_x \cap E^C(L_\rightarrow)_x| \\ + |E^C(L_\rightarrow)_x \setminus E^C(L_\leftarrow)_x| + |E^C(L_\leftarrow)_x \cap E^C(L_\rightarrow)_x| - |SE^C_x| \end{aligned} \tag{82}$$

which is smaller than or equal to $|E^C(L_\leftarrow)_x \cap E^C(L_\rightarrow)_x|$ due to Equation (81), thus satisfying Equation (64). Since $|E^C_{min}(L)_x|$ does not overestimate the true value for atomic concepts, and any successive step uses this underestimation to compute $|E^C(L_\leftarrow)_x| + |E^C(L_\rightarrow)_x| - |SE^C_x|$, then Equation (64) holds for any label $L \in \mathfrak{L}^n$.

Case AND NOT operator. In this case, for the intersection,

$$|I^C_{\max}(L)_x| = \min\left(|I^C_{\max}(L_\leftarrow)_x|,\ |N^C_x| - |I^C(L_\rightarrow)_x|\right) \tag{83}$$

The true value is

$$|I^C(L)_x| = |I^C(L_\leftarrow)_x \setminus I^C(L_\rightarrow)_x| = |I^C(L_\leftarrow)_x \cap (N^C_x \setminus I^C(L_\rightarrow)_x)| \tag{84}$$

Since AND NOT flips the right side and is 1-preserving on the left:

- If the minimum selects $|I^C_{\max}(L_\leftarrow)_x|$, then because a set difference cannot exceed its left operand and $I^C_{\max}$ is an overestimation,

$$|I^C_{\max}(L_\leftarrow)_x| \geq |I^C(L_\leftarrow)_x \setminus I^C(L_\rightarrow)_x| \tag{85}$$

- If the minimum selects $|N^C_x| - |I^C(L_\rightarrow)_x|$, then this equals $|N^C_x \setminus I^C(L_\rightarrow)_x|$, and because intersections cannot exceed either operand,

$$|I^C_{\max}(L)_x| \geq |I^C(L)_x| \tag{86}$$

Regarding the extras, recall that

$$|E^C_{min}(L)_x| = \max(|E^C_{min}(L_\leftarrow)_x| - |E^C(L_\rightarrow)_x|, 0) \tag{87}$$

The zero case follows from the non-negativity of cardinality. Conversely, if the max operator selects the first quantity, then we can prove it never overestimates the true value by induction, by fixing $L_\leftarrow$ to an atomic concept, similarly to the AND case. In this case, we have access to the exact computation and thus

$$|E^C_{min}(L)_x| = \max(|E^C(L_\leftarrow)_x| - |E^C(L_\rightarrow)_x|, 0) \tag{88}$$

If the sets overlap, then the true value is

$$|E^C(L)_x| = |E^C(L_\leftarrow)_x \setminus E^C(L_\rightarrow)_x| = |E^C(L_\leftarrow)_x| - |E^C(L_\rightarrow)_x \cap E^C(L_\leftarrow)_x| \tag{89}$$

This is clearly greater than $|E^C(L_\leftarrow)_x| - |E^C(L_\rightarrow)_x|$ and thus eq. (64) is satisfied. Similarly to the AND operator case, since $|E^C_{min}(L)_x|$ does not overestimate the true value for atomic concepts, and any successive step uses this underestimation to compute $|E^C_{min}(L_\leftarrow)_x| - |E^C(L_\rightarrow)_x|$, then Equation (64) holds for any label $L \in \mathfrak{L}^n$.

Thus, for all operators considered in this paper, **the heuristic never underestimates the true common intersection, and admissibility is satisfied.**

Aggregated Computation. The admissibility of the aggregated version follows directly from the observation that the estimation produced by aggregating quantities per label rather than per sample is always larger than the one computed per sample, because it assumes completely disjoint concepts for the OR and AND NOT operators and fully overlapping concepts for the AND operator (i.e., the extreme cases).

F.2 Remaining Minimum Estimations

While not necessary for the proof of admissibility, in this section we prove the minimum estimations for the quantities not discussed in the previous paragraphs, namely $I^C_{min}(L)_x$ and $E^C_{max}(L)_x$. This proof will be useful to establish the admissibility of the Path Heuristic and the completeness of the algorithm. Both follow the same reasoning used for $E^C_{min}(L)_x$ and $I^C_{max}(L)_x$, respectively.

Namely, the minimum estimations for the intersection used by the algorithm are

$$|I^C_{min}(L)_x| = \begin{cases} \max(|I^C_{min}(L_\leftarrow)_x|, |I^C(L_\rightarrow)_x|) & \text{if } \oplus = OR \quad (90) \\ \max(|I^C_{min}(L_\leftarrow)_x| + |I^C(L_\rightarrow)_x| - |N^C_x|, 0) & \text{if } \oplus = AND \quad (91) \\ \max(|I^C_{min}(L_\leftarrow)_x| - |I^C(L_\rightarrow)_x|, 0) & \text{if } \oplus = AND\ NOT \quad (92) \end{cases}$$

The true values are

$$|I^C(L)_x| = \begin{cases} |I^C(L_\leftarrow)_x \cup I^C(L_\rightarrow)_x| & \text{if } \oplus = OR \quad (93) \\ |I^C(L_\leftarrow)_x \cap I^C(L_\rightarrow)_x| & \text{if } \oplus = AND \quad (94) \\ |I^C(L_\leftarrow)_x \setminus I^C(L_\rightarrow)_x| & \text{if } \oplus = AND\ NOT \quad (95) \end{cases}$$

Following the same reasoning used in Section F.1 to prove the underestimation of E^C_{min}, we obtain that $|I^C_{min}(L)_x| \leq |I^C(L)_x|$ holds for the OR operator, because the cardinality of a set cannot exceed that of its union with another set.

Similarly, for the other operators we replace E^C with I^C and SE^C with N^C when needed. For the AND operator, we expand

$$|I^C(L_\leftarrow)_x| + |I^C(L_\rightarrow)_x| = |I^C(L_\leftarrow)_x \setminus I^C(L_\rightarrow)_x| + |I^C(L_\leftarrow)_x \cap I^C(L_\rightarrow)_x| + |I^C(L_\rightarrow)_x \setminus I^C(L_\leftarrow)_x| + |I^C(L_\leftarrow)_x \cap I^C(L_\rightarrow)_x| \quad (96)$$

and using that the three disjoint parts satisfy

$$|I^C(L_\leftarrow)_x \setminus I^C(L_\rightarrow)_x| + |I^C(L_\rightarrow)_x \setminus I^C(L_\leftarrow)_x| + |I^C(L_\leftarrow)_x \cap I^C(L_\rightarrow)_x| \leq |N^C_x| \quad (97)$$

we obtain $|I^C_{min}(L)_x| \leq |I^C(L)_x|$.

For the AND NOT operator, we can rewrite the true value as

$$|I^C(L)_x| = |I^C(L_\leftarrow)_x \setminus I^C(L_\rightarrow)_x| = |I^C(L_\leftarrow)_x| - |I^C(L_\leftarrow)_x \cap I^C(L_\rightarrow)_x| \quad (98)$$

which is greater than $|I^C(L_\leftarrow)_x| - |I^C(L_\rightarrow)_x|$ and thus $|I^C_{min}(L)_x| \leq |I^C(L)_x|$.

Analogously, for E^C the estimations and the true values are

$$|E^C_{max}(L)_x| = \begin{cases} \min(|E^C_{max}(L_\leftarrow)_x| + |E^C(L_\rightarrow)_x|, |SE^C_x|) & \text{if } \oplus = OR \quad (99) \\ \min(|E^C_{max}(L_\leftarrow)_x|, |E^C(L_\rightarrow)_x|) & \text{if } \oplus = AND \quad (100) \\ \min(|E^C_{max}(L_\leftarrow)_x|, |SE^C_x| - |E^C(L_\rightarrow)_x|) & \text{if } \oplus = AND\ NOT \quad (101) \end{cases}$$

and

$$|E^C(L)_x| = \begin{cases} |E^C(L_\leftarrow)_x \cup E^C(L_\rightarrow)_x| & \text{if } \oplus = OR \quad (102) \\ |E^C(L_\leftarrow)_x \cap E^C(L_\rightarrow)_x| & \text{if } \oplus = AND \quad (103) \\ |E^C(L_\leftarrow)_x \setminus E^C(L_\rightarrow)_x| & \text{if } \oplus = AND\ NOT \quad (104) \end{cases}$$

Using the same reasoning as in Section F.1 for I^C_{max}, we see that $|E^C_{max}(L)_x| \geq |E^C(L)_x|$ holds for the AND operator because the cardinality of an intersection cannot exceed that of either operand. For the OR operator, $|E^C_{max}(L)_x| \geq |E^C(L)_x|$ holds because $|E^C(L)_x| \leq |E^C(L_\leftarrow)_x| + |E^C(L_\rightarrow)_x|$ and $|E^C(L)_x| \leq |SE^C_x|$. Finally, for the AND NOT operator the inequality follows because a set difference cannot exceed its left operand when the first term is selected, and because an intersection cannot exceed either operand when the second term is selected.

F.3 Proof of Admissibility of the Path Heuristic

Let L_i be a generic label of length i. This heuristic aims at estimating the maximum achievable IoU score by any label L_j in the state space $\mathfrak{L}^n_{L_i}$, encompassing all labels of length up to n with $i < n$ and $t = n - i$, such that the leftmost part of each label L_j corresponds to L_i (i.e., $L_j = (((L_i \oplus L_k) \dots) \oplus L_l)$).

Let L_* be the label in $\mathfrak{L}^n_{L_i}$ associated with the highest IoU score. To be admissible, the heuristic must satisfy the following constraints:

$$\begin{cases} |\widehat{I^U(L_*)_x}| + |\widehat{I^C(L_*)_x}| \geq |I^U(L_*)_x| + |I^C(L_*)_x| & (105) \\ 0 \leq |\widehat{E^U(L_*)_x}| + |\widehat{E^C(L_*)_x}| \leq |E^U(L_*)_x| + |E^C(L_*)_x| & (106) \end{cases}$$

for all $x \in \mathfrak{D}$. In the contraints and in the following, we use the term "true value" to refer to the exact computation and use the symbol $\widehat{QUANTITY}(L_*)_x$ to express the estimation of the maximum possible value reachable from L_i for the considered quantity. Equation 105 ensures that the heuristic returns an optimistic estimate of the intersection (numerator), while equation 106 guarantees a pessimistic estimate of the denominator of eq. (3).

These constraints are satisfied by design due to the derivations described in the previous section for the Top and Bott vectors, which encode the maximum and minimum possible improvements, respectively. To prove this, let us consider the exclusive paths for each operator considered in this paper.

Case OR Operator. In this case, the heuristic estimation for both the numerator and denominator in eq. (3) is

$$|\widehat{I^U(L_*)_x}| + |\widehat{I^C(L_*)_x}| = \min(|I^C_{max}(L)_x| + Top_t(I^C)_x, |N^C_x|) + \min(|I^U_{max}(L)_x| + Top_t(I^U)_x, |N^U_x|) \quad (107)$$

and

$$|\widehat{E^U(L_*)_x}| + |\widehat{E^C(L_*)_x}| = \max(|E^C_{min}(L)_x| + |E^U_{min}(L)_x|, \; Bott_1(E^C)_x + Bott_1(E^U)_x \quad (108)$$

where we drop the $|N|$ term because it is common to both expressions and does not affect the proof.

Without loss of generality, let us consider the ideal case where there exists a sequence of concepts up to length n, $\{L_j, \dots, L_n\}$, such that each of them fully overlaps with L_i on the extras, and each of them is pairwise disjoint and disjoint from the elements of L_i that already overlap with the neuron activations. Moreover, assume that these concepts have the largest possible number of elements overlapping with the neuron activation. This construction represents the ideal scenario for the OR operator, and no other concept in the dataset can further improve the IoU.

In this ideal case, the true value of the denominator is

$$|E^U(L)_x| + |E^C(L)_x| \quad (109)$$

because all concepts fully overlap with L on the extras.

By comparing the true value with Equation (108), we obtain

$$|\widehat{E^U(L_*)_x}| + |\widehat{E^C(L_*)_x}| \; \leq \; |E^U(L_*)_x| + |E^C(L_*)_x| \quad (110)$$

Indeed, if the max operator selects the first quantity, the inequality holds because $|E^C_{min}(L)_x|$ and $|E^U_{min}(L)_x|$ are underestimations, as proven in Section F.1. If the max operator selects the second quantity, the inequality follows from the definition of the $Bott_1(E^C)_x$ and $Bott_1(E^U)_x$ vectors (Section 2.3.2). These vectors store, for each sample x, the minimum number of extras among all concepts in the dataset (including 0). Thus, for any concept L_k in $\{L_j, \dots, L_n\}$, if L_k has the smallest number of common extras for that sample, then $Bott_1(E^C)_x = |E^C(L_k)_x|$; otherwise,

$$Bott_1(E^C)_x < |E^C(L_k)_x| \quad (111)$$

and thus

$$Bott_1(E^C)_x \leq |E^C(L_*)_x| \tag{112}$$

The same argument applies to unique extras, and therefore Equation (106) is satisfied.

In the numerator case, because all the concepts are pairwise disjoint on the intersection and disjoint with respect to L_i, we can sum their quantities for ease of understanding as a single aggregated factor:

$$|I^C(L_*)_x| = |I^C(L_j)_x| + \cdots + |I^C(L_n)_x| \tag{113}$$

$$|I^U(L_*)_x| = |I^U(L_j)_x| + \cdots + |I^U(L_n)_x| \tag{114}$$

The true value of the numerator in this ideal case is given by

$$|I^C(L_i)_x| + |I^U(L_i)_x| + |I^C(L_*)_x| + |I^U(L_*)_x| \tag{115}$$

By comparing the true value with Equation (107), we can note that when the minimum selects $|N_x^C|$ and $|N_x^U|$, we have

$$\widehat{|I^C(L_*)_x|} + \widehat{|I^U(L_*)_x|} \;\geq\; |I^C(L_*)_x| + |I^U(L_*)_x| \tag{116}$$

by definition, since $\{L_j, \ldots, L_n\}$ are disjoint in the intersection with respect to L_i and their sum cannot exceed the available space for the common and unique intersections.

If the minimum selects the first terms, then as proven in Section F.1, $|I^C_{max}(L_i)_x|$ and $|I^U_{max}(L_i)_x|$ are overestimations, and thus the constraint reduces to proving that

$$Top_t(I^C)_x \geq |I^C(L_*)_x| \tag{117}$$

$$Top_t(I^U)_x \geq |I^U(L_*)_x| \tag{118}$$

This is satisfied by construction of the Top_t vectors. These vectors store, for each sample, the sum of the t concepts in the dataset with the largest number of intersection elements for that sample. Note that t is also the length of L_*. Thus, for any sample $x \in \mathfrak{D}$, if the t concepts with the largest intersections correspond to those in L_*, then $Top_t(I^C)_x = |I^C(L_*)_x|$, and otherwise $Top_t(I^C)_x > |I^C(L_*)_x|$. The same argument applies to the unique intersection, and therefore Equation (105) is satisfied.

Case AND Operator. In this case, the heuristic estimation for both the numerator and denominator in eq. (3) is

$$\widehat{|I^U(L_*)_x|} + \widehat{|I^C(L_*)_x|} = \min(|I^C_{max}(L)_x|, Top_1(I^C)_x) \tag{119}$$

and

$$\widehat{|E^U(L_*)_x|} + \widehat{|E^C(L_*)_x|} = 0 \tag{120}$$

where we drop the $|N|$ term because it is common to both expressions and does not affect the proof. We can note that Equation (106) is satisfied by the non-negativity of the cardinality.

To prove Equation (105), without loss of generality, let us consider the ideal case where there exists a sequence of concepts up to length n, $\{L_j, \ldots, L_n\}$, such that each of them fully overlaps with L_i on the common intersection. This construction represents the ideal scenario for the AND operator since the AND operator cannot increase the intersection, but this construction allows preserving all the already intersecting common elements of L_i.

The true value of the numerator in this ideal case is given by

$$|I^C(L_i)_x| \tag{121}$$

since by definition the AND operator removes all the unique elements and, in this ideal case, all the common intersection of L_i are preserved.

If the minimum selects the first term, then as proven in Section F.1, $|I^C_{max}(L_i)_x|$ and $|I^U_{max}(L_i)_x|$ are overestimations, and thus the constraint reduces to proving that

$$Top_1(I^C)_x \geq |I^C(L_i)_x| \tag{122}$$

$$Top_1(I^U)_x \geq |I^U(L_i)_x| \tag{123}$$

This is satisfied by construction of the Top_1 vectors and the fact in this ideal case the added concepts fully overlap. These vectors store, for each sample, the largest number of intersection elements for that sample among the concepts in $\mathfrak{L}^1$. Thus, since any concept L_k in $\{L_j, \ldots, L_n\}$ is assumed to fully overlap with L_i in the intersection, it either has the largest number of common intersection for that sample and thus $Top_1(I^C)_x = |I^C(L_k)_x|$, or it is not the largest and thus

$$Top_1(I^C)_x > |I^C(L_k)_x| \tag{124}$$

Therefore

$$Top_1(I^C)_x \geq |I^C(L_*)_x| \tag{125}$$

and Equation (106) is satisfied. In the general case, if any of the added concepts L_k does not fully overlap, then the true value becomes

$$|I^C(L_i)_x \cap I^C(L_k)_x| \tag{126}$$

which is satisfied by the fact that the cardinality of an intersection cannot exceed that of either operand when L_k is the concept associated with the largest common intersection in that sample and always in the other cases.

Case AND NOT Operator. In this case, the heuristic estimation for both the numerator and denominator in eq. (3) is

$$\widehat{|I^U(L_*)_x|} + \widehat{|I^C(L_*)_x|} = |I^U_{max}(L)_x| + \min(|I^C_{max}(L)_x|, |N^C_x| - Bott_1(I^C)_x) \tag{127}$$

and

$$\widehat{|E^U(L_*)_x|} + \widehat{|E^C(L_*)_x|} = |E^U_{min}(L)_x| \tag{128}$$

As done previously, let us consider the ideal case where there exists a sequence of concepts up to length n, $\{L_j, \ldots, L_n\}$, such that each of them fully overlaps with L_i on the extras, and each of them is pairwise disjoint and disjoint from the elements of L_i that already overlap with the neuron activations. This construction represents the ideal scenario for the AND NOT operator, and no other concept in the dataset can further improve the IoU. We can note that the negation of these concepts corresponds to the ideal case of the AND operator, since the negation fully overlaps in the intersection and is disjoint in the extras. The difference here is that all the unique elements of L_i are preserved.

Thus, the true value of the numerator in this ideal case is

$$|I^C(L_i)_x| + |I^U(L_i)_x| \tag{129}$$

since the AND NOT operator preserves all unique elements and preserves all common intersections of L_i.

As proven in Section F.1, $|I^U_{max}(L_i)_x|$ is an underestimation, and thus the constraint reduces to proving that

$$\min(|I^C_{max}(L_i)_x|, |N^C_x| - Bott_1(I^C)_x) \geq |I^C(L_i)_x| \tag{130}$$

If the minimum selects the first term, then, as proven in Section F.1, $|I^C_{max}(L_i)_x|$ is an overestimation, and thus Equation (105) is satisfied. If the minimum selects the second term, we must prove that

$$|N^C_x| - Bott_1(I^C)_x \geq |I^C(L_i)_x| \tag{131}$$

Recall that $Bott_1$ stores, for each sample x, the minimum number of common intersections among all concepts in the dataset (including 0). Therefore, the quantity $|N^C_x| - Bott_1(I^C)_x$ represents the maximum number of intersecting elements obtainable from the negation of these concepts. In other words, it represents the maximum intersection space coverable by 0-entries for that sample.

Thus, for any concept L_k in $\{L_j, \ldots, L_n\}$, if L_k has the smallest number of common intersections for that sample, then its negation corresponds to $|N^C_x| - Bott_1(I^C)_x$, and because it is disjoint with respect to L_i, we have

$$|N^C_x| - Bott_1(I^C)_x = |I^C(L_i)_x| \tag{132}$$

Conversely, if L_k does not have the smallest number of common intersections for that sample, then

$$Bott_1(I^C)_x \leq |I^C(L_i)_x| \tag{133}$$

and therefore

$$|N_x^C| - Bott_1(I^C)_x > |I^C(L_i)_x| \tag{134}$$

so Equation (105) is satisfied.

In the general case, if any of the added concepts L_k is not fully disjoint, then the true value becomes

$$|I^C(L_i)_x \setminus I^C(L_k)_x| \tag{135}$$

where L_k is the concept with the largest overlap with L_i. Since $I^C(L_k)_x$ removes at least $Bott_1(I^C)_x$ from $I^C(L_i)_x$, and $|I^C(L_i)_x| \leq |N_x^C|$ by definition, Equation (105) still holds.

Regarding the denominator, since the concepts are disjoint in the extras, the true value is

$$|E^U(L_i)_x| \tag{136}$$

because the AND NOT operator preserves all unique elements. As proven in Section F.1, $|E^U_{min}(L_i)_x|$ is an underestimation, and thus Equation (106) is satisfied.

Aggregated Computation. The admissibility of the aggregated version follows the same proof as in the sample version. The only difference is that the Top and $Bott$ vectors are computed per concept rather than per sample. In this case, they naturally represent overestimations and underestimations since they represent the ideal choice of the concepts to add for each operator at the dataset level (e.g., Top represents the case where the top t concepts in the dataset for the intersection are pairwise disjoint and disjoint with L_i over the full dataset).

F.4 Remaining Minimum Estimations

While not necessary for the proof of admissibility, in this section we prove that the minimum estimations for the path discussed in the previous paragraphs are underestimations. This result will be useful for guaranteeing the completeness of the algorithm.

Namely, the minimum estimations for the intersection used by the algorithm are

$$\widehat{|I_{min}(L*)_x|} \begin{cases} \max(|I^C_{min}(L)_x| + |I^U_{min}(L)_x|, & (137) \\ \quad Bott_1(I^C)_x + Bott_1(I^U)_x) & \text{OR Path} \quad (138) \\ 0 & \text{AND Path} \quad (139) \\ |I^U_{min}(L)_x| & \text{AND NOT Path} \quad (140) \end{cases}$$

We can note that the minimum estimations for the AND and AND NOT paths are underestimations by non-negativity of the cardinality in the AND path and by the property that the AND NOT operator preserves all unique elements.

For the OR path, if the maximum operator selects the first term, then this represents an underestimation since $|I^C_{min}(L)_x|$ and $|I^U_{min}(L)_x|$ are underestimations of $|I^C(L)_x|$ and $|I^U(L)_x|$, as proven in Section F.2, and the OR operator cannot reduce the intersection. If the max operator selects the second term, then $\widehat{|I_{min}(L)_x|} \leq |I(L)_x|$ because OR is 1-preserving and any concept L_k chained to L_i through an OR operator must satisfy

$$|I^C(L_k)_x| + |I^U(L_k)_x| \geq Bott_1(I^C)_x + Bott_1(I^U)_x \tag{141}$$

by definition of the $Bott_1$ vectors.

Regarding the maximum denominator, recall that the algorithm uses the following estimate

$$\widehat{|Union_{max}(L*)_x|} = |\boldsymbol{N}_x| + \begin{cases} \min(|E^C_{max}(L)_x| + Top_t(E^C)_x, |SE^C_x|) & (142) \\ \quad + \min(|E^U_{max}(L)_x| + Top_t(E^U)_x, & (143) \\ \quad |SE^U_x|) & \text{OR Path} \quad (144) \\ \min(|E^C_{max}(L)_x|,\ Top_1(E^C)_x) & \text{AND Path} \quad (145) \\ |E^C_{max}(L)_x| + \min(|E^U_{max}(L)_x|, & (146) \\ \quad |SE^C_x| - Bott_1(E^C)_x) & \text{AND NOT Path} \quad (147) \end{cases}$$

where we can drop $|\boldsymbol{N}_x|$ since it is shared by both the true values and the estimations. These estimates are specular to the ones used for estimating the maximum possible intersection. Therefore, the result follows by applying the same reasoning used for $I^C_{max}(L)_x$ and replacing I^C with E^C, I^U with E^U, and N^C with SE^C.

F.5 Optimality of the algorithm

Because the heuristics used to explore the state space are both admissible (Sections F.1 and F.3), because the quantities used to prune the frontier are guaranteed to be underestimations of the real IoU reachable from that node (Sections F.2 and F.4), and because the algorithm explores exhaustively all the nodes in the search space with an estimated alignment greater than the found solution, the algorithm **is complete and guaranteed to find the combination of concepts** $(L_1 \oplus L_2 \oplus \cdots \oplus L_n)$ **among the concepts in** $\mathfrak{L}^n$ **that captures the highest possible alignment (IoU)**.

G Examples of Explanations Difference

This section presents examples of the differences between explanations computed by beam search and those obtained with the optimal algorithm. To visualize the alignment, we select the samples in the dataset where the explanation holds and where the neuron is active within the considered activation range. We sort these samples by the size of the intersection between the explanation area and the activation area to ease interpretation, and then visualize the top 4 samples, highlighting the activation area in blue. The examples are not cherry-picked; rather, they correspond to the first nine differing explanations for the last convolutional layer of each explained model (ResNet18 (He et al., 2016), AlexNet (Krizhevsky et al., 2012), and DenseNet (Huang et al., 2017)). All the models have been pretrained on the Place365 dataset. Note that due to the selection procedure, samples used for visualization of different explanations or different methods for the same unit may differ, and this is expected. Alternative selection procedures, such as randomly extracting samples where the neuron is active, would produce harder-to-understand visualizations due to the superposition of neurons Elhage et al. (2022); O'Mahony et al. (2023); Dreyer et al. (2024) and variability in the size of activation. We also stress that the visualization is provided only as a reference and contextualization of the highlighted differences in explanations and not as a means of comparison. Despite the good results in the visualization, as discussed in the main text, the **optimality is not related to visualization properties or interpretability** but is related to the optimal (i.e., highest possible) solution found in a search problem and, specifically in this context, to the property of identifying the specific combination of concepts that captures (or expresses) the highest absolute spatial alignment with the neuron's activations.

H Other Neuron and Concept-Based Explanations

While this paper focuses on methods targeting neuron spatial alignment, the literature offers a wide range of explanation approaches that use concepts to derive explanations Casper et al. (2023); Gilpin et al. (2018) or to decode other types of neurons Hesse et al. (2025); Srinivas et al. (2025); Bau et al. (2020); O'Mahony et al. (2025); Bykov et al. (2023) and layer behaviors Gao et al. (2025); Bricken et al. (2023). Our work is inspired by this growing interest in understanding neuron behaviors and using concepts to build explanations. However, the similarities end there: these approaches belong to distinct families of interpretability methods that differ fundamentally from compositional explanations across multiple dimensions, such as their goal (e.g., alignment vs. correlation), scope (e.g., neuron clusters vs. layers), assumptions (e.g., access to internals and availability of masks), representation (e.g., logical vs. statistical explanations), and phase (post-hoc vs. ante-hoc). Because these families are not able to measure the spatial alignment between activations and the locations of concepts, they have traditionally been regarded as complementary rather than competing categories with respect to compositional explanations. Consequently, they are not compared against them, and no established protocols exist for such comparisons. Given these differences, and because our work is theoretical in nature and specifically focuses on guaranteeing the optimality of compositional explanations without altering their established formulation, we leave the exploration of connections and complementarities with other interpretability paradigms for future research.

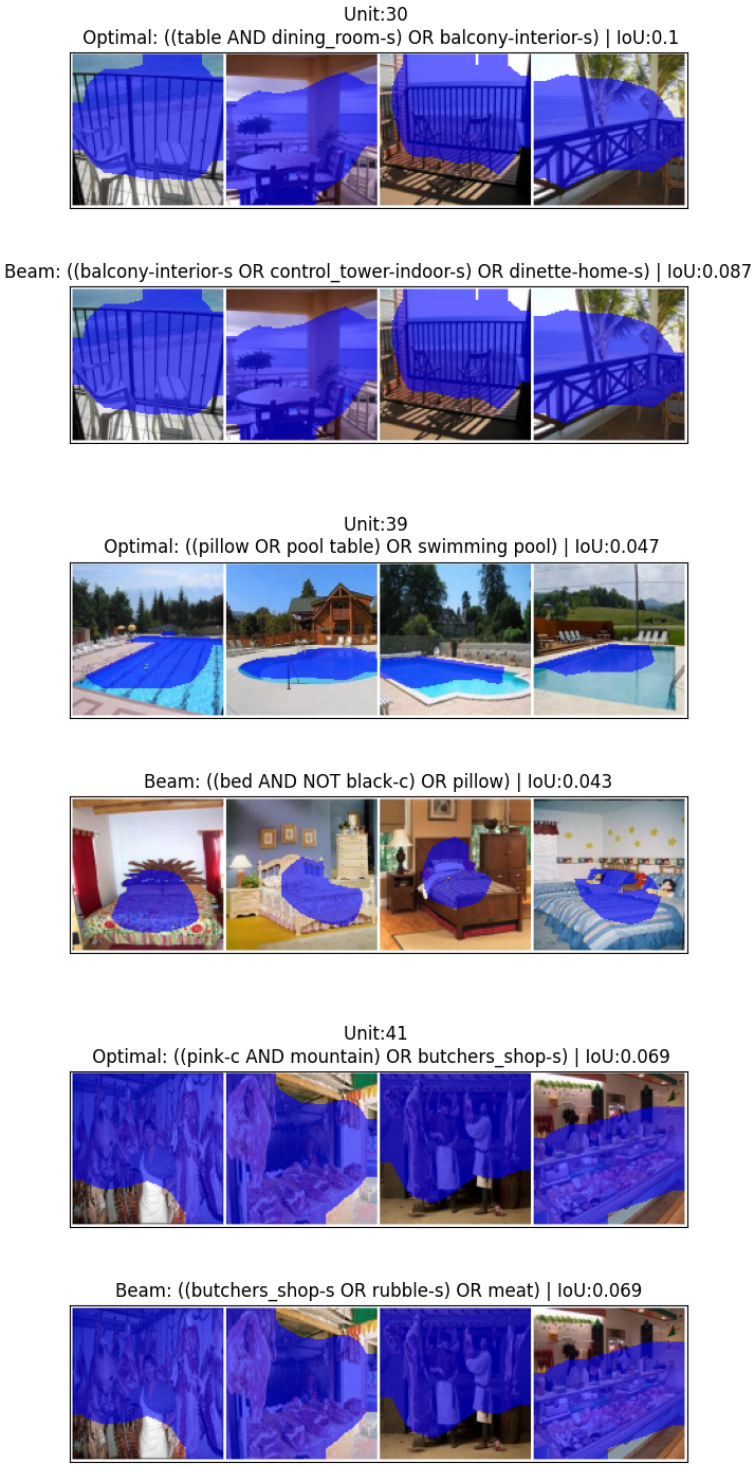

Figure 2: Alignment detected in units of a ResNet18 model by both the optimal and beam-search methods. The blue regions indicate areas of neuron activation within the considered range.

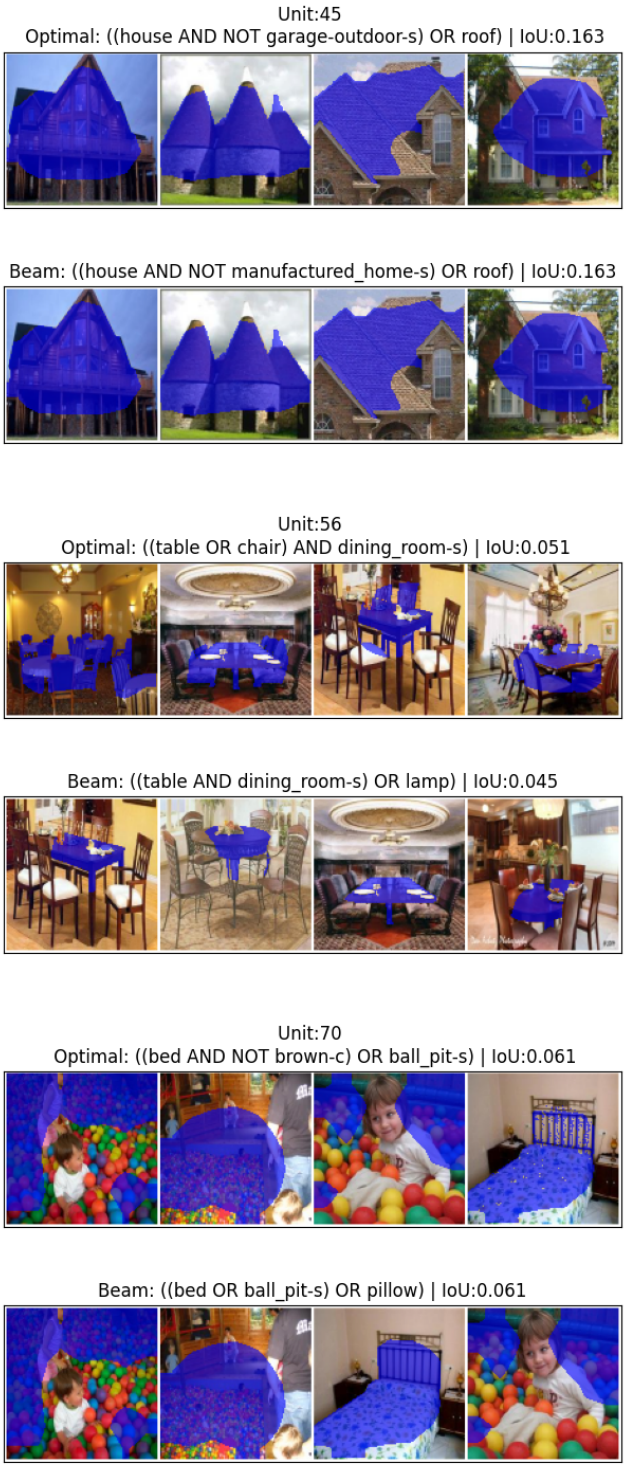

Figure 3: Alignment detected in units of a ResNet18 model by both the optimal and beam-search methods. The blue regions indicate areas of neuron activation within the considered range.

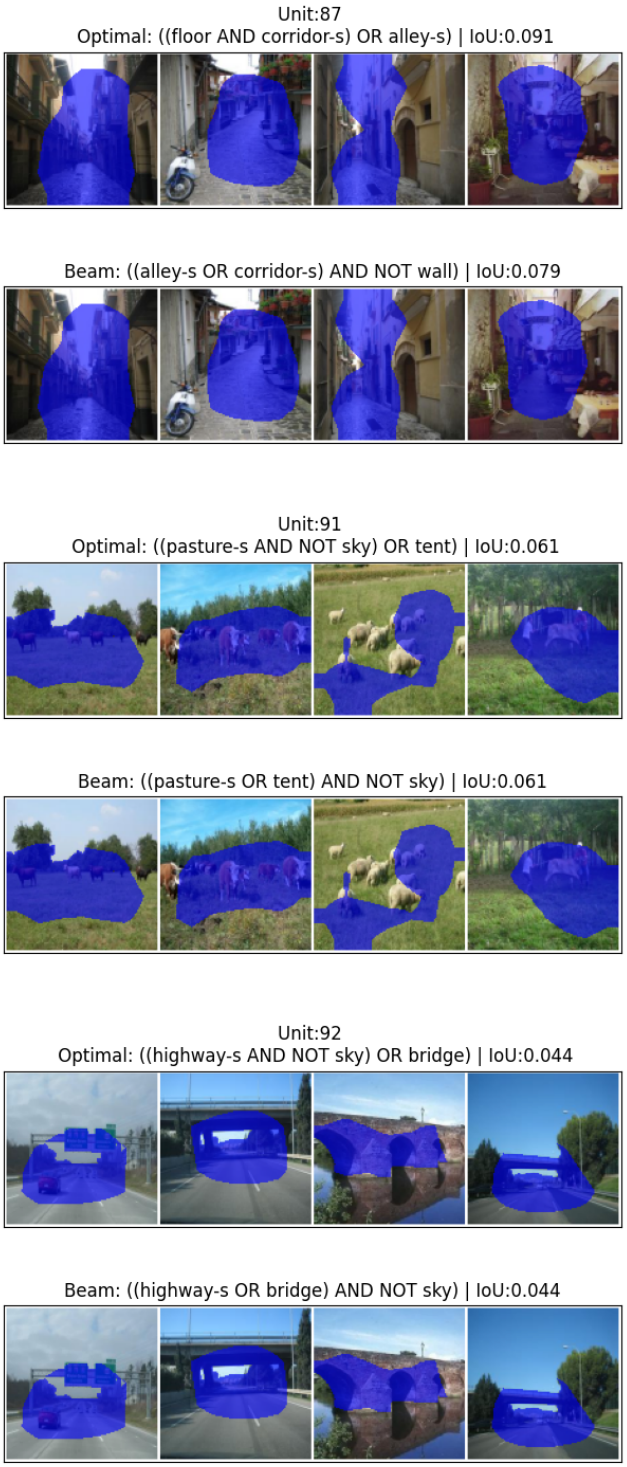

Figure 4: Alignment detected in units of a ResNet18 model by both the optimal and beam-search methods. The blue regions indicate areas of neuron activation within the considered range.

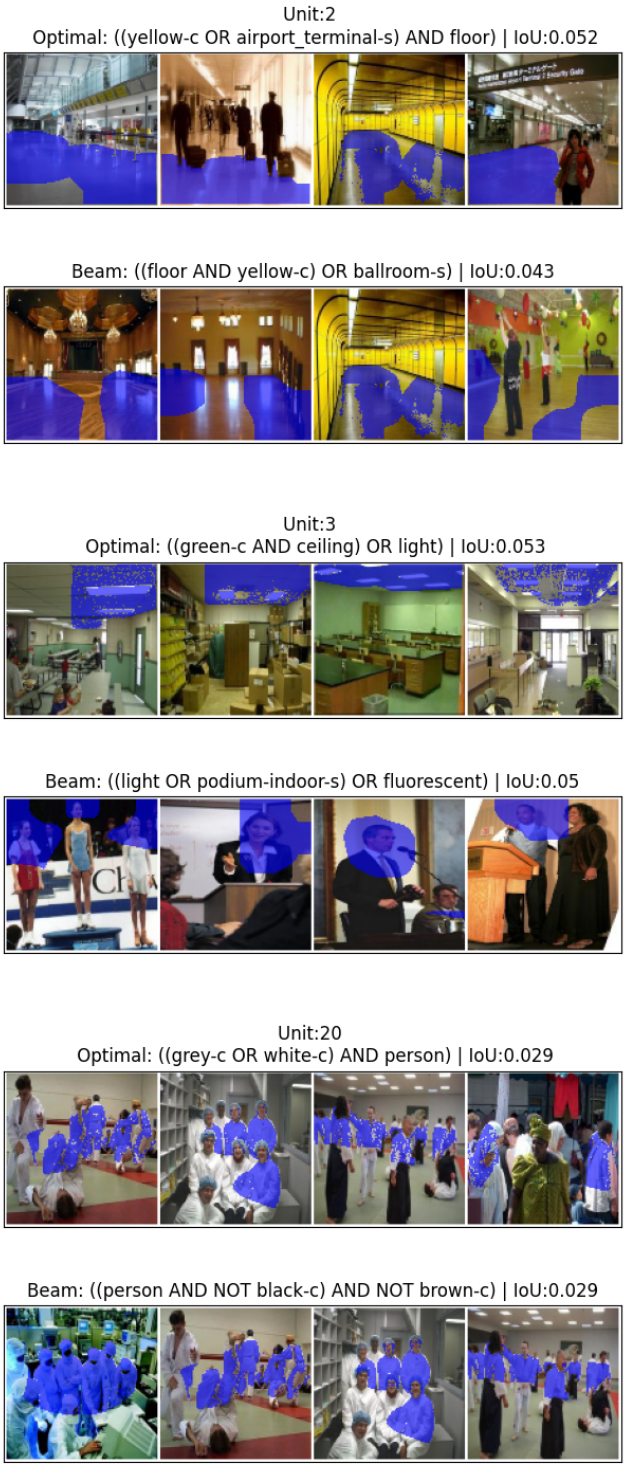

Figure 5: Alignment detected in units of an AlexNet model by both the optimal and beam-search methods. The blue regions indicate areas of neuron activation within the considered range.

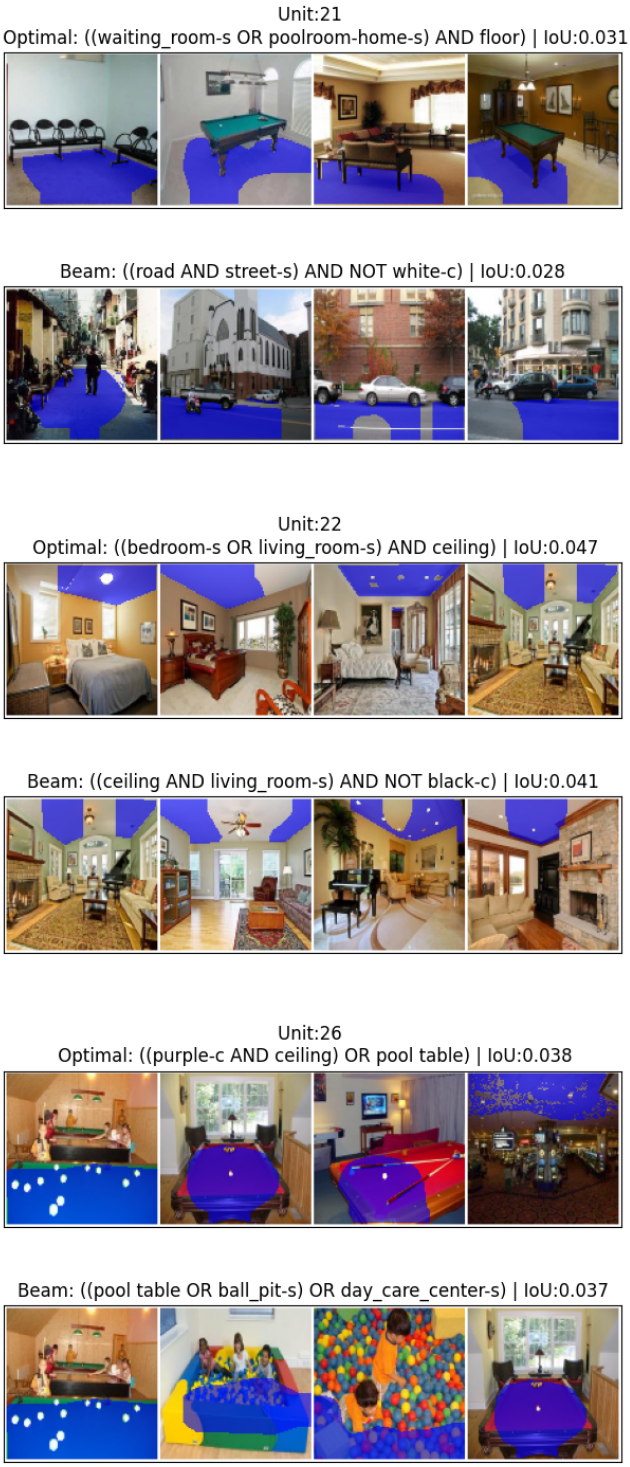

Figure 6: Alignment detected in units of an AlexNet model by both the optimal and beam-search methods. The blue regions indicate areas of neuron activation within the considered range.

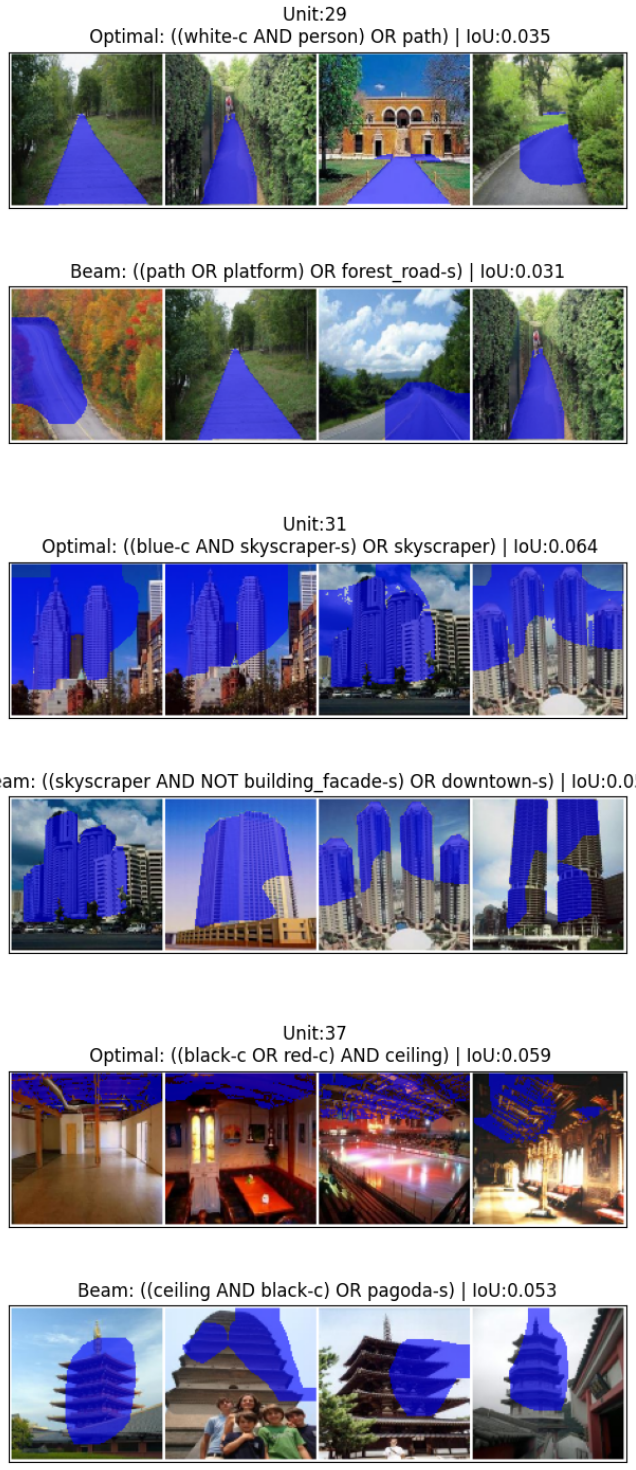

Figure 7: Alignment detected in units of an AlexNet model by both the optimal and beam-search methods. The blue regions indicate areas of neuron activation within the considered range.

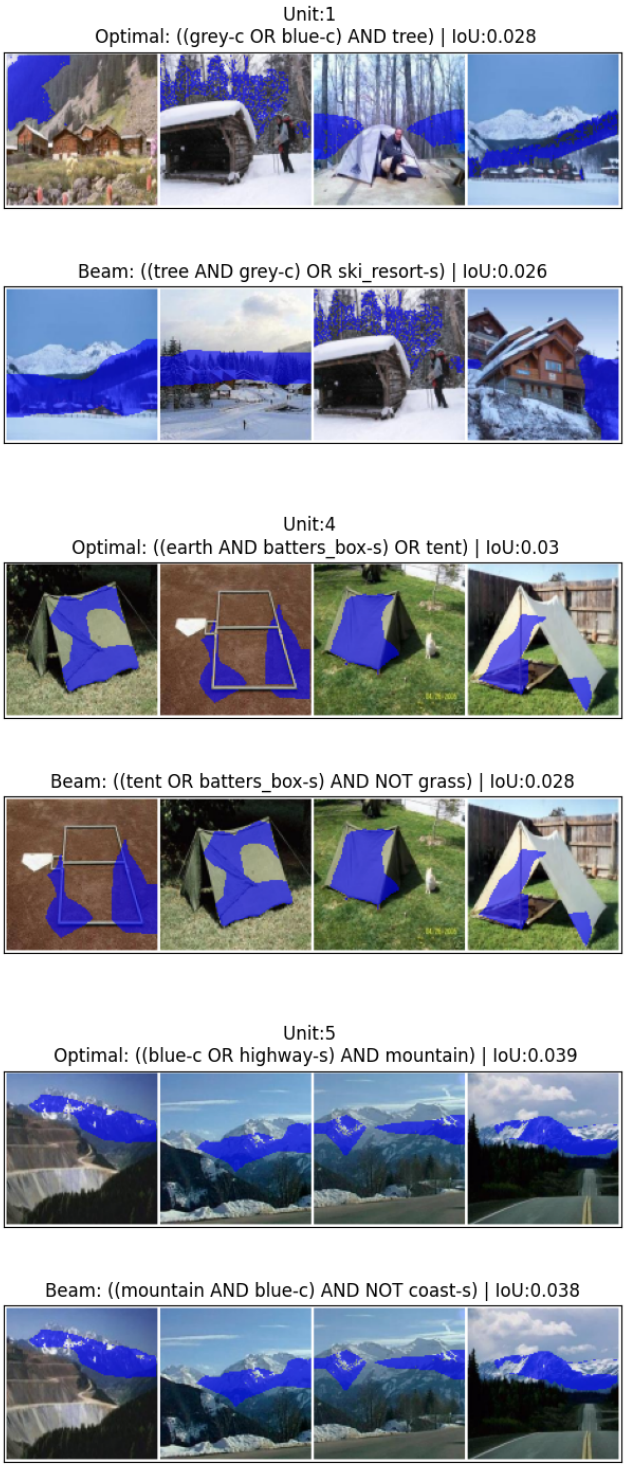

Figure 8: Alignment detected in units of a DenseNet161 model by both the optimal and beam-search methods. The blue regions indicate areas of neuron activation within the considered range.

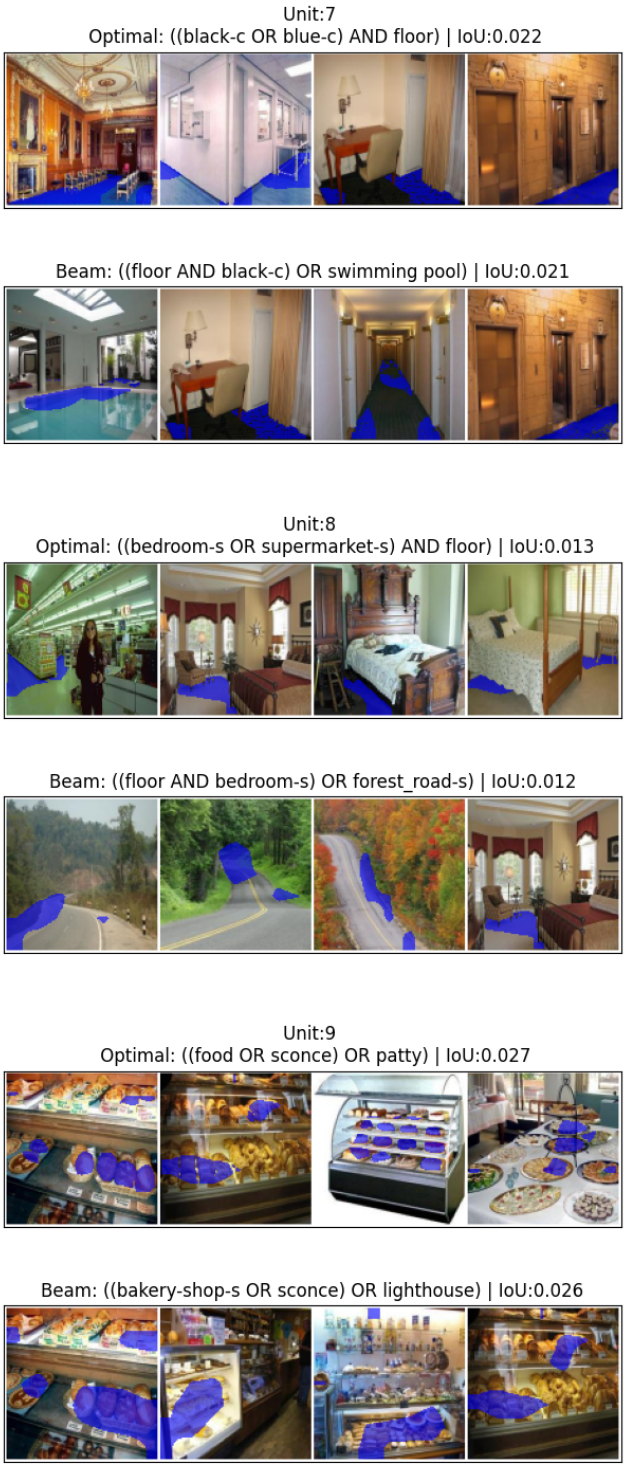

Figure 9: Alignment detected in units of a DenseNet161 model by both the optimal and beam-search methods. The blue regions indicate areas of neuron activation within the considered range.

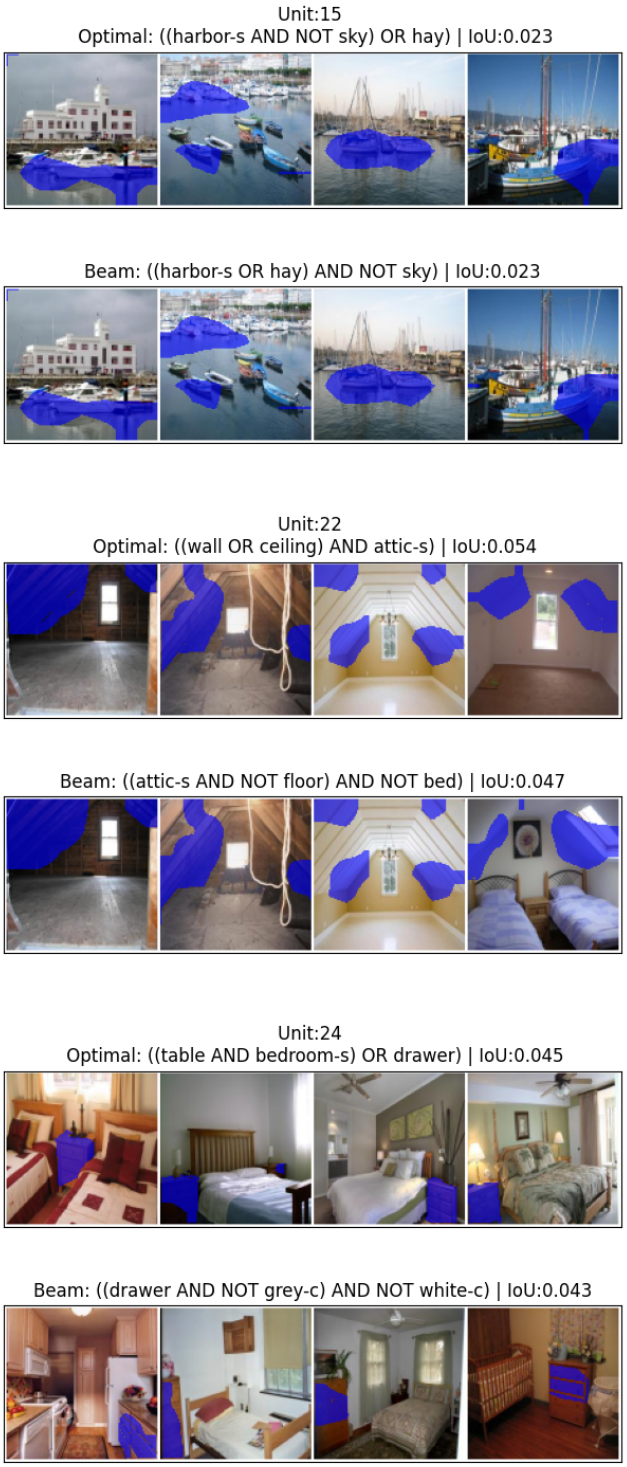

Figure 10: Alignment detected in units of a DenseNet161 model by both the optimal and beam-search methods. The blue regions indicate areas of neuron activation within the considered range.

Conversely, our work is highly related to Network Dissection and Compositional Explanations. Network Dissection Bau et al. (2020) measures the spatial alignment between neurons and a set of individual concepts, identifying the concept that maximizes alignment. Since it focuses on individual concepts, exhaustive search is feasible, and the resulting explanations are optimal within that context. Compositional Explanations extend this approach by searching for combinations of concepts that better express a neuron's alignment. However, they are unable to explore the full state space, which sacrifices optimality. Our work closes this gap between Network Dissection and Compositional Explanations by providing a framework that guarantees optimal compositional explanations. Additionally, the development of our beam variant further improves runtime, narrowing the efficiency gap between Network Dissection and Compositional Explanations.

I LLM USAGE STATEMENT

In this work, large language models were used as a general-purpose assistive tool for polishing the text and grammar checking. Our workflow consisted of providing the LLM with a draft of some paragraphs and requesting it to highlight grammatical errors or suggest improvements. The output from the LLM was then used as a reference to improve the text, whenever the suggestions were deemed correct by the authors. The LLM did not contribute to research ideation or analysis at any stage.

Algorithm 1: Optimal Algorithm

Input: $\mathfrak{L}^1$, $\boldsymbol{N}$, **M**, DisjointMatrix, length
Output: BestLabel,BestIoU

```
Frontier ← empty priority queue
ConceptQuantities, Memory ← empty lists
MinIoU, RecentIoU ← 0
for c_{k,i} in 𝔏^1 do
    ConceptQuantities[c_{k,i}] ← compute_quantities(c_{k,i}, M, N)
    Paths ← estimate_aggregate_paths(ConceptQuantities[c_{k,i}], length, MinIoU)
    Frontier.add(Paths)
    MinIoU ← update_min(Paths, MinIoU)
end
Frontier ← reduce_frontier(Frontier,MinIoU)
while Frontier is not empty do
    Node ← Frontier.pop()
    if Node is aggregate_estimation then
        UpdatedNode ← compute_sample_estimate(Node,MinIoU)
        MinIoU ← update_min(UpdatedNode, MinIoU)
        if UpdatedNode.max_iou > MinIoU then
            Frontier.add(UpdatedNode)
        end
        continue
    end
    UpdatedNode ← apply_logic_equivalences(Node)
    if UpdatedNode.max_iou < Node.max_iou and UpdatedNode.max_iou > MinIoU then
        MinIoU ← update_min(UpdatedNode, MinIoU)
        Frontier.add(UpdatedNode)
        continue
    end
    if Node.iou == RecentIoU then
        if Node in Memory then
            continue
        end
        else
            Memory.add(Node )
        end
    end
    else
        Memory ← empty list
        RecentIoU ← Node.max_iou
    end
    if Node is final then
        TreeQuantities = compute_tree_quantities(Node)
        Frontier ← update_by_tree(Frontier, TreeQuantities)
        IoU = compute_iou(TreeQuantities.get(Node)
        if IoU > BestIoU then
            BestIoU = IoU
            BestLabel = Node.label
        end
    end
    else
        AdditionalNodes ← expand(Node)
        Paths ← estimate_aggregate_paths(AdditionalNodes, Quantities,
          length,MinIoU)
        Frontier.add(Paths)
        if min(Paths) > MinIoU then
            MinIoU ← min(Paths)
            Frontier ← reduce_frontier(Frontier, MinIoU)
        end
    end
end
return BestLabel, BestIoU
```

Algorithm 2: Our Informed Beam Search Algorithm

Input: $\mathfrak{L}^1$, N, $\mathbf{M}$, DisjointMatrix, b, length
Output: BestLabel,BestIoU

```
Beam ← empty list
ConceptsQuantities ← empty list
for c_{k,i} in 𝔏^1 do
    Quantities ← compute_quantities(c_{k,i}, N, M)
    ConceptsQuantities.append(Quantities)
    IoU ← compute_dIoU(Quantities)
    Beam.add(label = c_{k,i}, iou = IoU)
end
sort(Beam)  # Sort by IoU
# Select the best b candidates
Beam ← Beam[:b]
MinIoU ← find_min(Beam)
for 2 to length do
    SearchSpace ← expand_beam(Beam, 𝔏^1)
    Estimations ← estimate_labels_iou(SearchSpace, ConceptQuantities, ,
     DisjointMatrix)
    sort(Estimations)
    for L, EstIoU in Estimations do
        if EstIoU < MinIoU then
            # All the other labels cannot be added to the beam
            break
        end
        Iou ← compute_iou(L, N, M)
        Beam.add(label=L, iou=Iou)
    end
    sort(Beam)
    # Select the best b candidates
    Beam ← Beam[:b]
    # Compute and update info
    MinIoU ← find_min(Beam)
end
BestLabel, BestIoU ← max(Beam)
return BestLabel, BestIoU
```