THE COMPLEXITY OF UNDERSTANDING: FORMALIZING CONCEPT DISCOVERY IN MECHANISTIC INTERPRETABILITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Mechanistic Interpretability (MI) is an emerging subfield of eXplainable Artificial Intelligence (XAI), which aims to reverse-engineer the internal representations of black-box machine-learning models (like neural networks) into human-understandable concepts. Despite increasing attention and promising results, the literature still lacks a formalization of the most basic and fundamental MI concept of "feature", leading to limited rigor and inconsistencies in the general adoption of MI, as well as a continued reliance on human judgment for its evaluation. In this work, we fill this gap by devising, for the first time, a formal definition of the notion of mechanistic feature, grounded in the core principles of the MI literature. We formulate a novel optimization problem to soundly identify mechanistic features according to our definition, and theoretically characterize it inf terms of **NP**-completeness, inapproximability, and mixed-integer nonlinear programming (MINLP) formulation. To face the inherent computational complexity of the problem, we design two well-founded heuristics: one consisting in solving a relaxed yet easier- and faster-to-solve MINLP formulation of our problem, and the other relying on a combined local- and beam-search strategy. We extensively evaluate our methods on numerous datasets and testing scenarios. The obtained results demonstrate their high effectiveness and superiority over a variety of baselines.

1 INTRODUCTION

Since its introduction, *eXplainable Artificial Intelligence* (XAI) (Gunning & Aha, 2019; Molnar, 2020) has designed ever evolving tools to provide human-interpretable understanding of black-box machine learning models, such as neural networks (NNs). The XAI landscape is vast and heterogeneous, comprising many different paradigms, approaches, and methods (Miller, 2019). A prominent subfield of XAI that has recently emerged is *Mechanistic Interpretability* (MI) (Bereska & Gavves, 2024; Räuker et al., 2023; Somvanshi et al., 2026; Zhang & Nanda, 2024), which draws inspiration from neuroscience and whose aim consists in reverse-engineering the internal representations of black-box machine- learning models into human-interpretable terms. MI focuses on understanding *why* and *how* internal computational units of models (e.g., neurons, activation pathways) determine certain outputs. MI represents a radical shift with respect to earlier approaches to XAI, such as post-hoc explainability, which simplistically describes *what* models do and, as such, typically suffers from limited expression power and reliability (Saeed & Omlin, 2023; Samek et al., 2021).

Among the key MI notions is the one of "*feature*",[1] which the MI literature (informally) refers to as a fundamental and irreducible unit of computation learned by a NN. The notion of feature is a cornerstone in MI: it is the most basic and fundamental MI notion which is at the basis of other more advanced MI concepts (e.g., "*circuits"*) and of the entire MI paradigm in general (Olah et al., 2020).

Motivation. Although the MI notion of feature has been informally described from an intuitive and empirical point of view (Bereska & Gavves, 2024; Cammarata et al., 2020; Elhage et al., 2022; Zeiler & Fergus, 2014), it lacks a rigorous and principled definition. This has led to repeatedly "reinvent the wheel" in the activities of characterizing and studying features, as well as in the design

[1]Throughout this work, the term "*feature*" refers to the corresponding core MI notion and not to the common machine-learning concept of individual measurable property or characteristic of a data point.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



Figure 1: Overview of the proposed framework. Given a matrix $V_{L,D}$, containing the activation value of each neuron of layer L of a NN trained model on each sample of dataset D used to probe the model, and parameters t and ϵ (cf. Problem 1), the KMF solver extracts K blocks of neurons and samples (i.e., submatrices of $V_{L,D}$) satisfying the definition of mechanistic feature (Definition 1) and maximizing the cumulative block size (left panel). Features are shown alongside their representative samples and synthesized visualizations, offering a qualitative reading of the underlying concepts (middle panel). Each block F_k induces a direction $\varphi(F_k)$ in $\mathbb{R}^{|L|}$; ϵ-orthogonality bounds their pairwise cosine similarity, ensuring diversity among features (right panel).

of algorithms for their identification. This has also resulted in heavy reliance on human judgment for evaluation (Bereska & Gavves, 2024), leaving the MI core assumptions hard to validate or refute.

Contributions. In this work, we fill this gap by devising a principled definition of feature adhering to the MI core principles. Let $V_{L,D}$ be the matrix collecting the activation value of each neuron of layer L of a trained NN on each sample of dataset D used to probe the NN. In MI, features are broadly acknowledged to correspond to geometric directions in the activation space identified by $V_{L,D}$ and satisfying the following requirements: (***R1***) *Neuron relevance*, which operationalizes feature visualization results and constrains the (absolute) magnitude of the activation above which a neuron is deemed relevant to the feature (Cammarata et al., 2020; Wu et al., 2024; Zeiler & Fergus, 2014); (***R2***) *Irreducibility* of features as fundamental units of computation, which is expressed essentially by requiring features not to be contained into one another (Bereska & Gavves, 2024); (***R3***) *Quasi-orthogonality* (Bereska & Gavves, 2024; Elhage et al., 2022), which imposes diversity among features by identifying them as ideally representing basis vectors of a geometric space and requiring that their geometric directions are pairwise orthogonal up to a small enough tolerance (Johnson-Lindenstrauss lemma (Larsen & Nelson, 2017)).

As a first contribution, we introduce a formal definition of feature represented as a block (i.e., submatrix) of $V_{L,D}$ and translating the informal requirements ***R1***–***R3*** into formal mathematical statements. We term the proposed definition simply *mechanistic feature* (MF). Building on this definition, we formulate the novel K-MECHANISTIC FEATURES (KMF) optimization problem: find K MFs that maximize their resulting cumulative size. Maximizing the cumulative MF size soundly expresses the desirable natural objective that the identified MFs cover as many neurons and samples as possible. We theoretically characterize KMF by proving that it is **NP**-complete and hard to approximate, as well as providing a mixed-integer nonlinear programming (MINLP) formulation of it. These theoretical results unveil the intrinsic difficulty of the task of MF discovery. To handle the inherent computational complexity burden of KMF, we devise two well-founded heuristics that scale to realistic problem instances. The first one consists in solving a relaxed yet easier- and faster-to-solve MINLP formulation of KMF. The second one adopts a combined local- and beam-search strategy.

Figure 1 illustrates our contributions. To the best of our knowledge, this is the first work on the formalization and theoretical characterization of MI concepts and their algorithmic identification.

Summary and roadmap. To summarize, our main contributions are as follows:

- We fill an important gap in the MI literature (§2) by providing the first formal definition of the key MI concept of "feature", in accordance with the core MI principles (§3.1).
- We formalize a novel optimization problem, K-MECHANISTIC FEATURES (KMF), to properly identify features according to the proposed definition and theoretically characterize it in terms of **NP**-completeness, inapproximability, and MINLP formulation (§3.2).
- To cope with the intrinsic computational hardness of KMF, we design two well-founded heuristics which are capable of handling realistic problem instances (§4).
- We extensively test the proposed methods on a variety of datasets and scenarios (§5), demonstrating their effectiveness and superior performance over several baselines (§6).

2 Related Work

Mechanistic Interpretability (MI) principles. MI has recently emerged as a prominent direction within the XAI landscape (Bereska & Gavves, 2024; Räuker et al., 2023; Somvanshi et al., 2026; Zhang & Nanda, 2024). The most basic MI notion – which other more complex MI concepts (e.g., "*circuits*" (Olah et al., 2020; Elhage et al., 2021)) and the entire MI paradigm rely on – is the one of "*feature*". Features are described as directions in the activation space of a NN layer (Bereska & Gavves, 2024). Neurons are often recognized as polysemantic (Olah et al., 2020; Bricken et al., 2023a), suggesting that the features exceed them ("*superposition*" property). Also, features are acknowledged to correspond to linear combinations of neurons, forming a basis of a geometric space under a degree of orthogonality (Larsen & Nelson, 2017) ("*linear representation*" assumption).

MI feature discovery has been traditionally tackled via *Principal Component Analysis* (*PCA*) and *Sparse Autoencoders* (*SAE*). *PCA* yields orthogonal directions via a change of basis of the activation space (Abdi & Williams, 2010). This fulfills the linear representation assumption, but not superposition, as it allows at most one component per neuron (Olah et al., 2020). *SAE* learn a sparse dictionary of directions in a higher-dimensional space (Han et al., 2018; Shi et al., 2020). *SAE* performance strongly depends on training and their evaluation mostly relies on ground-truth features (Karvonen et al., 2024). Further approaches fall into the *Causal Abstraction* (CA) framework (Pîslar et al., 2025; Geiger et al., 2022; 2025). *Boundless Distributed Alignment Search* (*BDAS*) searches for feature subspaces via gradient descent, offering a causal rather than correlational criterion (Grant et al., 2026). However, it needs a causal model specified in advance, often unavailable in realistic settings.

All the above methods do not define or characterize features formally: methods are designed directly, trying to capture high-level yet informal intuitions that derive intrinsically from the adopted method itself. Another important downside of those methods is that they adhere to MI core principles only in part. Conversely, we provide for the first time a rigorous mathematical pipeline for MI feature discovery: a formal definition of feature, an optimization problem to find features, a theoretical characterization and algorithms for our problem. Also, our pipeline fulfills the core MI principles. Nevertheless, we involve existing approaches to MI feature discovery in our experiments (§5–6).

3 Problem Definition

We are given a trained NN model M, a target layer L of M, and a dataset D of samples used to probe M, with $|L| = l$, $|D| = d$. $V_{L,D} \in \mathbb{R}^{l \times d}$ represents the activation matrix of L on D in M, i.e., the matrix whose cells $V_{L,D}[i, j]$ contain the activation of neuron i on sample j in M.

3.1 Mechanistic feature

We represent a feature through its empirical footprint in $V_{L,D}$, namely a block (i.e., a submatrix), i.e., a pair $\langle N_L, S_D \rangle$, where $N_L \subseteq L$ and $S_D \subseteq D$. A block identifies the subset of neurons of L carrying the feature representation and the subset of samples of D on which the feature is expressed. Neurons and samples in a block need not to be contiguous, although rows and columns of $V_{L,D}$ can be permuted to achieve this. Importantly, the feature itself is not the block, but the direction in $\mathbb{R}^l$ identified by it, defined as the mean of the columns of $V_{L,D}$ on the samples spanned by the block (formal definition provided later, in Definition 1). It is on this direction that the conditions of Definition 1 below are meant to hold. For instance, a metal feature may be carried by a handful of neurons responding to metal body panels, and be active precisely on the subset of samples depicting cars. Such neurons, though, need not be exclusive to cars, and may be exhibited by samples depicting other vehicles too, e.g., trucks, boats, airplanes, consistent with the shared angular, metal-like structure observed across these features (as visible from the results in §6, Fig. 2 and Appendix A.13).

Next, we state our formal definition of mechanistic feature. Note that we provide a general collective definition of $K \geq 1$ features due to the pairwise diversity requirement (**R3** in §1). The definition of a single feature in isolation easily derives from the general collective definition as a special case for $K = 1$ and with condition **C3** necessarily holding.

Definition 1 (Mechanistic feature (MF))**.** *Given an activation matrix $V_{L,D}$, we say that $K \in \mathbb{N}$ submatrices (blocks) F_1,..., F_K of $V_{L,D}$, with $F_k = V^k_{N,S} = \langle N^k_L, S^k_D \rangle$, $\forall k = 1,..., K$,* represent K mechanistic features *if:*

C1. $|v_{i,j}| \geq t, \forall v_{i,j} \in V^k_{N,S}, \forall\ k = 1,..., K$ ***(thresholding)***

C2. $\nexists\ F_{k'} \subseteq F_k, \forall k, k' = 1,..., K, k \neq k'$ ***(maximality)***

C3. $\cos\big(\varphi(F_{k'}), \varphi(F_k)\big) \in [-\epsilon, \epsilon], \forall k, k' = 1,..., K, k \neq k'$ ***(ϵ-orthogonality)***

where $t \in \mathbb{R}^+$ is the thresholding value of a neuron to be considered relevant for a feature, $\epsilon \in [0, 1]$ is the tolerance bounding the pairwise cosine similarity between feature directions, and $\cos(\cdot, \cdot)$ is the cosine similarity between two vectors. The feature represented by F_k is the direction $\varphi(F_k) = \mu\big(V^k_{L,S}\big) \in \mathbb{R}^l$, where $\mu(\cdot)$ is the row-wise means vector of a matrix: $\mu\big(V^k_{L,S}\big)[i] = (\sum_{j \in S^k_D} V_{L,D}[i, j])/|S^k_D|, \forall\ i = 1,..., l$.

Definition 1 is a faithful translation in formal terms of the core MI assumptions discussed in §1 (**R1**–**R3**). **C1** expresses **R1** (neuron relevance): we fix a magnitude $t \in \mathbb{R}^+$ below which a neuron is not considered relevant to a feature on a given sample. We take entries in absolute value, since activation strength depends on a neuron's magnitude, not on its sign. **C2** reflects **R2** (feature irreducibility): to be recognized as an irreducible fundamental unit of computation, a feature cannot be contained into any other, thus it must be maximal. **C3** captures **R3** (quasi-orthogonality). Following the superposition and linear representation assumptions (Bricken et al., 2023a; Larsen & Nelson, 2017; Olah et al., 2020) (cf. §2), features should ideally correspond to basis vectors of a high-dimensional space. Since this space's dimensionality typically exceeds the number of neurons l in the target layer L, features cannot be expressed as perfectly orthogonal in $\mathbb{R}^l$. Yet, according to the Johnson-Lindenstrauss lemma (Larsen & Nelson, 2017), an exponential number of almost orthogonal vectors can still be found within $\mathbb{R}^l$, and these are the vectors that define the feature space, with a tolerance ϵ of noise. We compute this directions as the mean activation of every neuron in L, restricted to the samples in the block, rather than restricting it to the block's own neurons. Restricting the direction to the block's own neurons, rather than to all l neurons in L, would prevent ϵ-orthogonality from being meaningfully evaluated in $\mathbb{R}^l$. As an example, a wheel feature and a metal feature, represented by two blocks on car samples, must be compared within the same l-dimensional space in which their directions are defined, not merely on the handful of neurons each block was built from.

We further remark that the blocks underlying features are neither exhaustive nor exclusive, as they do not necessarily cover $V_{L,D}$ entirely nor explain every neuron's behavior, and can freely overlap with one another. This is deliberate: overlap on neurons reflects the polysemantic nature of neurons, while overlap on samples is because a sample can activate multiple features (e.g., a car picture activating both a wheel feature and a metal feature). Moreover, as shown in Appendix A.1 in more detail, maximality (**C2**) is not subsumed by ϵ-orthogonality (**C3**), thus both **C2** and **C3** are needed.

3.2 The K-Mechanistic Features (KMF) problem

To identify MFs in a principled way, we tackle the following combinatorial-optimization problem:

Problem 1 (K-Mechanistic Features (KMF))**.** *Given input $(V_{L,D}, t, \epsilon, K)$, find K MFs $\mathscr{F}^{L,D} = \{F_1,..., F_K\}$, defined according to Def. 1 with parameters K, t, ϵ, which maximize $f(\mathscr{F}^{L,D}) = \sum_{k=1}^{K} |V^k_{N,S}| = |N^k_L| \cdot |S^k_D|$.*

KMF simply seeks to find K MFs that maximize their resulting cumulative size, indicating that the identifies MFs should span as many neurons and samples as possible. This is a natural objective: as the linear representation assumption (§3.1) implies that a feature is exhibited by multiple neurons and samples, it is desirable to identify the largest possible set of neurons and samples of any feature.

Theoretical characterization of KMF starts with noticing that, being features non-exhaustive and non-exclusive, we have $(2^{l+d})^K$ possible solutions. Not all of them satisfy Definition 1, however we cannot make any assumptions, as the valid sets depend on the specific input values in $V_{L,D}$. Still, the search space can be a-priori reduced through pruning and symmetry-breaking considerations (Appendix A.2, although it remains asymptotically exponential.

That said, we show next that KMF is $\mathbf{NP}$-complete. For the sake of brevity, here we provide proof sketch only. A complete proof is reported in Appendix A.3.

Theorem 1. *Problem 1 (KMF) is $\mathbf{NP}$-complete.*

Proof sketch. Membership in **NP** follows since, given a candidate solution, all conditions in Definition 1 can be verified in time polynomial in K, l, and d (Appendix A.3). For **NP**-hardness, we reduce from the MAX-EDGE BICLIQUE (MBC) problem, itself **NP**-complete (Peeters, 2003). Given a bipartite graph $G = (A \cup B, E)$, we build its adjacency matrix $\mathscr{A}$ and treat it as an instance of KMF with $K = 1$ and $0 < t < 1$. With a single feature, maximality (**C2**) and ϵ-orthogonality (**C3**) of Def. 1 trivially hold, thus the problem reduces to finding the largest all-above-threshold submatrix of $\mathscr{A}$, which corresponds exactly to a biclique in G. Consequently, a KMF solution with $|V^1_{N,S}| \geq W$ exists if and only if G contains a biclique with at least W edges (Appendix A.3).

Since MBC can be reduced in polynomial time to KMF, then the latter inherits MBC inapproximability results (Ambühl et al., 2011; Manurangsi, 2018). Let be $n = \max\{l, d\}$, then:

Corollary 1. *If the Small Set Expansion Hypothesis (Raghavendra & Steurer, 2010) holds, then no polynomial-time algorithm can approximate* KMF *within a factor of* $n^{1-\varepsilon}$ *of the optimum, for any* $\varepsilon > 0$*, unless* $\mathbf{NP} \subseteq \mathbf{BPP}$ *(Zachos & Heller, 1986).*

We also provide a mixed-integer nonlinear programming (MINLP) formulation of KMF. The output of KMF is a set of K blocks $\{(N^k, S^k)\}_{k=1}^K$. To express this, we use binary decision variables $x^k_{ij} \in \{0, 1\}$, where $x^k_{ij} = 1$ if $V_{L,D}[i, j]$ is part of block k, 0 otherwise. We also introduce support binary variables $n^k_i = 1$ iff $\sum_{j=1}^d x^k_{ij} \geq 1$ and $s^k_j = 1$ iff $\sum_{i=1}^l x^k_{ij} \geq 1$ (0 otherwise). The ultimate MINLP formulation of KMF is as follows:

$$\text{Maximize } \sum_{k=1}^K \sum_{i=1}^l \sum_{j=1}^d x^k_{ij} \tag{1}$$

$$\text{subject to } x^k_{ij} t \leq |v_{ij}|,\ \forall i = 1,...,l, j = 1,...,d, k = 1,...,K \tag{2}$$

$$\sum_{i=1}^l \max(0, n^{k_1}_i - n^{k_2}_i) + \sum_{j=1}^d \max(0, s^{k_1}_j - s^{k_2}_j) \geq 1,\ \forall k_1, k_2 = 1,...,K, k_1 \neq k_2 \tag{3}$$

$$o_{k_1 k_2} = \frac{\sum_{i=1}^l \varphi^{k_1}_i \varphi^{k_2}_i}{\sqrt{\sum_{i=1}^l (\varphi^{k_1}_i)^2} \sqrt{\sum_{i=1}^l (\varphi^{k_2}_i)^2}} \in [-\epsilon, \epsilon] \quad \begin{matrix} k_1, k_2 = 1,...,K \\ k_1 \neq k_2 \end{matrix}, \quad \varphi^k_i = \frac{\sum_{j=1}^d s^k_j v_{ij}}{\sum_{j=1}^d s^k_j} \tag{4}$$

$$\sum_{i=1}^l \sum_{j=1}^d x^k_{ij} \geq 1,\ \forall k = 1,...,K \tag{5}$$

$$x^k_{ij} \in \{0, 1\},\ \forall i = 1,...,l, j = 1,...,d, k = 1,...,K \tag{6}$$

(2), (3), and (4) express **C1**, **C2**, and **C3** in Definition 1, respectively. (3) is presented here in a nonlinear form for the sake of readability, but it can be reformulated as a set of linear constraints (Appendix A.5). (4) relies on the average direction vector φ^k whose inner product introduces an intrinsic nonlinearity. Also, (4) can be split into two constraints, one for each sign of ϵ. (5) constrains the output to correspond to K features. The MINLP formulation can be used to solve KMF exactly by simply adopting standard solvers. Clearly this is affordable only for small problem instances. Furthermore, MINLP paves the way for future theoretical and algorithmic research on KMF.

4 ALGORITHMS

MINLP-relaxed algorithm. The first algorithm we devise for KMF consists in solving an alternative MINLP formulation that relaxes the exact one defined in (1)–(6) and, as such, is easier- yet faster-to-solve. It is well-known that the main source of hardness in a MINLP formulation lies on the hard nonlinear constraints. For this reason, the proposed relaxed MINLP formulation of KMF consists in softening the only nonlinear hard constraint of (1)–(6), i.e., (4), by moving it as part of the objective function. Specifically, we define this soft constraint as the sum of the pairwise squared cosine similarities between each pair of features, normalized in $[0, 1]$. This way, the identification of features far from orthogonality is penalized. Ultimately, the objective function of the MINLP relaxation thus becomes as follows:

$$\max \sum_{k=1}^K w_k - \lambda \sum_{k_1=1}^K \sum_{k_2=k_1+1}^K o_{k_1 k_2} \tag{7}$$

where $w_k = \left(\sum_{i=1}^l \sum_{j=1}^d x^k_{ij}\right) / |(V_{L,D})_t| \in [0, 1]$, $|(V_{L,D})_t|$ is the number of cells of $V_{L,D}$ whose value is $\geq t$, and λ is a parameter controlling the importance of the ϵ-orthogonality-derived term. Since $w_k, o_{k_1 k_2} \in [0, 1]$, then the upper bound of $\sum_{k=1}^K w_k$ is K, while the upper bound of $\sum_{k_1=1}^K \sum_{k_2=k_1+1}^K o_{k_1 k_2}$ is $\frac{K(K-1)}{2}$. Thus, λ can be soundly set to $\frac{2}{(K-1)}$, so as to have the two terms roughly comparable. A more detailed analysis of (7) is reported in Appendix A.4.

Algorithm 1 MINLP-relaxed

Input: $V_{L,D} \in \mathbb{R}^{l \times d}$, $t \in \mathbb{R}^+$, $K \in \mathbb{N}$
Output: K MFs (Def. 1) $\mathscr{F}^{L,D}$ represented by $\mathcal{F} = \{F_1, ..., F_K\}$, with F_k submatrix of $V_{L,D}$, $\forall k = 1,..., K$
1: Define MINLP formulation in (8) with input $\langle V_{L,D}, t, K \rangle$
2: $\mathcal{F} \leftarrow$ solve (8) with a solver

Algorithm 2 L&B-search

Input: $V_{L,D} \in \mathbb{R}^{l \times d}$, $t \in \mathbb{R}^+$, $p \in (0, 1]$, $K, m, \rho \in \mathbb{N}$ ($p = 0.25$, $\rho = 1\,000$ by default)
Output: K MFs (Def. 1) $\mathscr{F}^{L,D}$ represented by $\mathcal{F} = \{F_1, ..., F_K\}$, with F_k submatrix of $V_{L,D}$, $\forall k = 1,..., K$
1: $\mathcal{B} \leftarrow$ blocks corresponding to the rows of $V_{L,D}$ cleared of all the cells whose value is $< t$; $\mathcal{C} \leftarrow \mathcal{B}$
2: **repeat**
3: $\quad \mathcal{C}' \leftarrow$ pairwise merge blocks in $\mathcal{C}$; $\mathcal{C} \leftarrow \{$top-$\eta = \min(\lfloor p|\mathcal{C}'| \rfloor, \rho)$ blocks from $\mathcal{C}'$ w.r.t. $\phi(\cdot)$ score$\}$; $\mathcal{B} \leftarrow \mathcal{B} \cup \mathcal{C}$
4: **until** $\mathcal{B}$ does not change w.r.t. the previous iteration
5: Remove from $\mathcal{B}$ non-maximal blocks and blocks corresponding to near-duplicate directions
6: $\widehat{\mathcal{B}} \leftarrow \{$top-$m$ blocks from $\mathcal{B}$ w.r.t. $\psi(\cdot)$ pre-selection score$\}$; $\mathcal{F} \leftarrow$ solve (9) on input $\widehat{\mathcal{B}}$ with a solver

The ultimate relaxed MINLP formulation of KMF is as follows:

$$\text{Maximize } (7) \text{ subject to } (2), (3), (5), (6) \tag{8}$$

while the proposed algorithm based on it, termed MINLP-relaxed, is outlined in Algorithm 1.

L&B-search. Although MINLP-relaxed can handle moderate-size problem instances, still it might be impractical for larger instances. For this reason, we devise a faster algorithm, which we term L&B-search and high-level describe in Algorithm 2. More details are in Appendix A.6.

In a nutshell, L&B-search selects promising blocks via local search. It proceeds iteratively, adopting beam search to retain the best blocks from the previous iteration and use them to identify further promising blocks in the current iteration. The adoption of local search and beam search motivates the algorithm name. All the blocks from the various iterations undergo a further filtering step of blocks that violate the problem constraints or are unpromising (details below). The resulting blocks are used to get the output K blocks via solving an auxiliary optimization problem (details below).

More specifically, L&B-search starts from blocks corresponding to the rows of $V_{L,D}$ cleared of all the cells whose value is $< t$. Then, the following pre-selection steps are iteratively executed. *(1) Composing score.* $\forall \langle B_i, B_j \rangle \mid B_i \circ B_j \neq \emptyset$ and $\min(B_n) \geq t$, we compute the composing score $\phi(\langle B_i, B_j \rangle) = (\lambda_{size} w_n) + (\lambda_{div} o_{ij})$, where $0 \leq \lambda_{size}, \lambda_{div} \leq 1$ and $\lambda_{size} + \lambda_{div} = 1$ weigh the new block's size against the orthogonality between its two components. Unlike (7), the terms are summed, since we want to merge blocks that both contribute to a larger structure and are highly similar, so that the final blocks are diverse enough. Block composition is defined as $B_n = B_i \circ B_j = \langle N^i \circ_N N^j, S^i \circ_S S^j \rangle$, with $\circ_N, \circ_S \in \{\cup, \cap\}$. *(2) Sorting and categorization.* Pairs are sorted in non-increasing order by $\phi(\cdot)$, retaining only the top-$\eta = \min(\lfloor p|\mathcal{C}| \rfloor, \rho)$ for the next iteration, with parameters $p \in (0, 1]$ and $\rho \in \mathbb{N}$ ($p = 0.25, \rho = 1\,000$ by default). ρ controls the growth of the search space. *(3) Composition and filtering*: Selected pairs are composed under (i) $\circ_N = \cup, \circ_S = \cap$, (ii) $\circ_N = \cap, \circ_S = \cup$, or (iii) all combinations of $\circ_N$ and $\circ_S$. Blocks then pass three stages: *Closure* adds as many samples as possible while respecting thresholding; *Maximality* removes from $\mathcal{B}$ any non-maximal block; and *DirectionFiltering* orders blocks by size and discards any whose direction has cosine similarity $\geq \tau$ with an already-visited block. We set $\tau = 0.95$, as this step only removes near-duplicate features, most filtering being done in earlier stages.

All the blocks $\mathcal{B}$ from the various iterations undergo a final filtering. The top-m blocks $\widehat{\mathcal{B}}$ according to score $\psi(B_i) = \frac{w_i}{(\sum_{j \neq i} o_{ij}/N - 1)}$ are retained As ϵ-orthogonality is a pairwise quantity, each block's contribution is evaluated by averaging its similarity to all others. We use a ratio-based rather than subtractive score, as averaging over multiple comparisons can attenuate meaningful differences. Eventually, the m blocks $\widehat{\mathcal{B}}$ resulting from this filtering form the input for the following auxiliary optimization problem, whose solution (through a solver) gives the ultimate output K blocks:

$$\text{Maximize } \sum_{i=1}^{m-1} x_i w_i - \lambda \sum_{i=1}^{m} \sum_{j=i+1}^{m} x_i x_j o_{ij} \text{ subject to } \sum_{i=1}^{m} x_i = K, x_i \in \{0, 1\}, \forall i = 1,..., m \tag{9}$$

where $x_i = 1$ if $B_i \in \widehat{\mathcal{B}}$ is selected, 0 otherwise, and w_i and o_{ij} are defined as in (7) and precomputed after pre-selection. As for conditions **C1**–**C3** in Def. 1, **C1** and **C2** are enforced during pre-selection, while **C3** is incorporated into the objective (softening it, like in MINLP-relaxed). In Appendix A.7 we analyze the computational complexity of L&B-search and how to set m in practice.

5 EXPERIMENTAL SETUP

NN models, datasets, activation matrix. We involve five model-dataset combinations: *VGG16* (Simonyan & Zisserman, 2015) and *ViT* (Dosovitskiy et al., 2021) models on both *Cifar10* and *Cifar100* (Krizhevsky et al., 2009) datasets, and *Llama* 3.2 1B (Grattafiori et al., 2024) model on *SST5* (Socher et al., 2013) dataset. We consider the last two layers of *VGG16* and *ViT* and the last two layers of an MLP head fine-tuned on *SST5* with *Llama* 3.2 1B. The activation matrix $V_{L,D}$ is obtained probing the dataset D selecting $d' = 50$ test samples per class $c \in C$, as assigned by the NN. The last layer offers a controlled setting: its neurons encode class information, so an ideal method should recover $|C|$ blocks, one per neuron with d' samples each. The second-last layer is instead a realistic setting to assess efficiency and compare with baselines. We cover two activation regimes: ReLU on *VGG16* and *Llama* and linear on *ViT*. More details on these settings are in Appendix A.8.

Baselines include *PCA*, *SAE*, and *BDAS* (§2). We also employ *Spectral Co-clustering* (SCC) (Von Luxburg, 2007), as a representative of the co-clustering paradigm (Battaglia et al., 2024). Resorting to co-clustering is motivated as the detection of blocks of $V_{L,D}$ satisfying the condition of neuron relevance in our MF definition roughly resembles a co-clustering objective. However, co-clustering does not consider all the conditions of our MF definition, thus it can be considered only as a baseline, not a well-founded method for MF identification.

Testing scenarios. *Last* and *second-last layer*: On the last layer, we run both our MINLP-relaxed and L&B-search algorithms, along with the baselines. On the second-last layer, as expected, MINLP-relaxed could not run in reasonable time. However, this layer mainly serves the purpose of testing L&B-search on a more complex scenario and evaluating its scalability. *Varying* K: we tested L&B-search on the second-last layer with varying K. We adapted *PCA* and *SAE* outputs to our feature representation following prior work (Gorton, 2024; Huben et al., 2024), but with a different thresholding strategy than top-K (Appendix A.10). Hyperparameter tuning is in Appendix A.9.

Evaluation scores we use are both internal and external. The former include the objective function ($f(V_{L,D})$), the mean feature dot product ($\mathscr{O}$), and the mean block size ($\mathscr{W}$). Since baselines may find $K' < K$ blocks (Appendix A.10), we report the number K' of blocks actually found by them in the results. As $ub(f(V_{L,D}))$ scales with K (Appendix A.4), objective values obtained with different K are not comparable; hence, we report $f(V_{L,D})$ normalized by K (or K'). We also involve clustering-based measures, i.e., Purity ($\mathscr{C}_P$), Entropy ($\mathscr{C}_E$), and Consistency ($\mathscr{C}_C$), which capture a different notion than MI, as they measure alignment with ground-truth/predicted classes while MFs do not necessarily correspond to classes. Clustering-based scores do not necessarily reflect feature quality, but we find them useful, complementary indicators.Finally, we report Ablation ($\mathscr{I}^S_{abl}$) and Gaussian ($\mathscr{I}^{S'}_{\sigma}$) IIA scores (Geiger et al., 2022). They need a high-level causal model pre-aligned with the NN, and assume that intervening on a single variable suffices to flip the prediction. Neither holds in our context, so we adjust the metrics accordingly, as detailed in Appendix A.11. We additionally report Loss Recovered ($\mathscr{L}$) Bricken et al. (2023b); Karvonen et al. (2024), a reconstruction-fidelity score borrowed from the SAE literature, adapted to our setting as detailed in Appendix A.12.

6 RESULTS

From now on, $\mathcal{M}$ denotes MINLP-relaxed, while $\mathcal{L}$ denotes L&B-search. Threshold t is set to the $(1-1/|C|)$-quantile on the last layer (where each sample mostly activates one of the $|C|$ neurons) and the 90th percentile on the second-last layer of the non-zero entries of $V_{L,D}$ (Appendix A.8). Table 1 reports the results on the last layer with $K = 5$ and on the second-last layer with $K = 10$. The choice of K is for a fair comparison to the baselines, as they may output $K' \neq K$ features and to allow the execution of $\mathcal{M}$. Table 2 reports $\mathcal{L}$ results on the second-last layers with varying K.

Last layer. On *Cifar10* and *SST5*, $\mathcal{M}$ and $\mathcal{L}$ achieve identical results, while $\mathcal{L}$ requires substantially less execution time. On *Cifar100*, $\mathcal{M}$ is omitted since its execution time becomes prohibitive, whereas $\mathcal{L}$ remains at most 0.12s. Overall, $\mathcal{L}$ generally outperforms the baselines or ties with their best results, demonstrating the effectiveness of how we tackle KMF. *SAE* and *BDAS* do not always capture all K features, highlighting their limits in finding actual MFs in the activation space. Across all models, $\mathscr{C}_C$ remains above 96%, while $\mathscr{C}_P$ is lower for the less accurate models, namely *VGG16* on *Cifar100* and *Llama*. This is consistent with the last layer encoding class information, where the quality of the predictions affects the alignment between the discovered blocks and the ground truth.

Table 1: Results on last and second-last layers on *VGG16*, *ViT* (*Cifar10* and *Cifar100*), and *Llama* (*SST5*). Numbers under brackets aside the baselines correspond to K'. Boldface/underlined for best/second-best results.

			Time	$f(V_{L,D})$	$\mathscr{W}$	$\mathscr{O}$	$\mathscr{C}_P$	$\mathscr{C}_C$	$\mathscr{C}_E$	$\mathscr{I}^S_{abl}$	$\mathscr{I}^{S'}_{\sigma}$	$\mathscr{L}$
Last layer (K = 5)	VGG16 (*cifar10*)	$\mathcal{M}$	130.65s	**0.11**	**56.8** (± 6.08)	0.01 (± 0.02)	77.11%	96.83%	0.35	-	-	-
		$\mathcal{L}$	0.02s	**0.11**	**56.8** (± 6.08)	0.01 (± 0.02)	77.11%	96.83%	0.35	-	-	-
		PCA(5)	0.01s	-0.02	40.2 (± 5.71)	0.1 (± 0.3)	**89.55%**	**100%**	**0.17**	-	-	-
		SAE(4)	**0.0s**	-0.42	39 (± 8.92)	0.5 (± 0.5)	84.62%	98.72%	0.25	-	-	-
		SCC (5)	2.26s	0.1	54.4 (± 7.89)	8.84 (± $< 10^{-2}$)	75.74%	97.43%	0.37	-	-	-
		BDAS(5)	15.66s	0.08	38.8 (± 16.87)	**0.0** (± 0.0)	80.41%	98.97%	0.29	-	-	-
	VGG16 (*cifar100*)	$\mathcal{M}$	-	-	-	-	-	-	-	-	-	-
		$\mathcal{L}$	0.09s	**0.01**	59.4 (± 3.38)	**0.0** (± 0.0)	61.62%	96.63%	0.34	-	-	-
		PCA(5)	0.02s	-0.59	**62.6** (± 2.8)	0.6 (± 0.49)	55.41%	97.45%	0.4	-	-	-
		SAE(5)	**0.01s**	-0.09	49.4 (± 2.33)	0.1 (± 0.3)	80.24%	**99.6%**	0.18	-	-	-
		SCC (5)	3s	**0.01**	59.2 (± 3.82)	6.38 (± 1.92)	58.78%	97.3%	0.35	-	-	-
		BDAS(5)	67.08s	**0.01**	49.4 (± 5)	**0.0** (± 0.0)	**80.65%**	97.18%	**0.14**	-	-	-
	ViT (*cifar10*)	$\mathcal{M}$	114.21s	**0.11**	**55.2** (± 3.66)	$< 10^{-2}$ (± 0.01)	78.99%	97.83%	0.34	-	-	-
		$\mathcal{L}$	0.02s	**0.11**	**55.2** (± 3.66)	$< 10^{-2}$ (± 0.01)	78.99%	97.83%	0.34	-	-	-
		PCA(5)	**0.01s**	0.06	31.6 (± 6.05)	$< 10^{-2}$ (± $< 10^{-2}$)	**94.94%**	**100%**	**0.09**	-	-	-
		SAE(4)	**0.01s**	0.08	39.25 (± 1.48)	$< 10^{-2}$ (± $< 10^{-2}$)	94.9%	98.09%	0.1	-	-	-
		SCC (5)	0.05s	0.1	54.6 (± 3.14)	$< 10^{-2}$ (± $< 10^{-2}$)	78.75%	98.9%	0.34	-	-	-
		BDAS(5)	30.93s	0.09	46.4 (± 3.14)	**-0.01** (± 0.0)	82.76%	**100%**	0.28	-	-	-
	ViT (*cifar100*)	$\mathcal{M}$	-	-	-	-	-	-	-	-	-	-
		$\mathcal{L}$	0.12s	**0.01**	**54** (± 0.0)	**0.0** (± 0.0)	85.4%	97.81%	0.14	-	-	-
		PCA(5)	0.02s	-0.59	**54** (± 0.0)	0.6 (± 0.49)	88%	98.18%	0.12	-	-	-
		SAE(5)	**0.01s**	**0.01**	48.2 (± 1.47)	**0.0** (± 0.0)	**92.98%**	**100%**	**0.07**	-	-	-
		SCC (5)	0.16s	**0.01**	53.8 (± 2.93)	**0.0** (± 0.0)	75.71%	96.43%	0.19	-	-	-
		BDAS(5)	386.89s	**0.01**	51.2 (± 2.48)	**0.0** (± 0.0)	80.23%	99.22%	0.15	-	-	-
	Llama (*SST5*)	$\mathcal{M}$	37.65s	**0.2**	**25** (± 11.98)	**0.0** (± 0.0)	58.4%	**100%**	0.61	-	-	-
		$\mathcal{L}$	0.02s	**0.2**	**25** (± 11.98)	**0.0** (± 0.0)	58.4%	**100%**	0.61	-	-	-
		PCA(5)	**0.01s**	-0.02	9.6 (± 4.72)	0.1 (± 0.3)	62.5%	**100%**	**0.54**	-	-	-
		SAE(5)	**0.01s**	-0.01	24.2 (± 12.02)	0.2 (± 0.4)	**65.29%**	**100%**	0.57	-	-	-
		SCC (5)	0.04s	**0.2**	**25** (± 11.98)	**0.0** (± 0.0)	58.4%	**100%**	0.61	-	-	-
		BDAS(4)	8.07s	0.11	13.25 (± 9.42)	**0.0** (± 0.0)	58.49%	**100%**	0.63	-	-	-
Second-last layer (K = 10)	VGG16 (*Cifar10*)	$\mathcal{L}$	65.94s	**0.03**	**199.3** (± 202.12)	0.11 (± 0.07)	**100%**	**100%**	**0.0**	0.49 (± 0.08)	**-0.63** (± 0.07)	0.99
		PCA(10)	0.03s	-0.02	193.4 (± 143.01)	0.17 (± 0.19)	99.19%	**100%**	0.01	**0.5** (± 0.09)	-0.62 (± 0.08)	0.98
		SAE(10)	**0.01s**	-0.11	57.8 (± 30.76)	0.22 (± 0.27)	98.47%	99.24%	0.02	0.47 (± 0.09)	-0.52 (± 0.13)	0.96
		SCC (9)	2.44s	0.02	181.56 (± 203.33)	**0.1** (± 0.07)	**100%**	**100%**	**0.0**	0.48 (± 0.1)	-0.61 (± 0.1)	0.97
		BDAS(10)	30.58s	-0.03	27.5 (± 22.75)	0.14 (± 0.15)	97.94%	**100%**	0.03	0.46 (± 0.11)	-0.53 (± 0.08)	**1.0**
	VGG16 (*Cifar100*)	$\mathcal{L}$	37.06s	-0.16	370.7 (± 110.18)	0.39 (± 0.11)	25.02%	25.4%	0.43	0.77 (± 0.06)	-0.41 (± 0.03)	0.32
		PCA(10)	0.03s	-0.22	404 (± 106.96)	0.44 (± 0.19)	39.76%	40.45%	0.29	**0.84** (± 0.05)	-0.43 (± 0.03)	0.48
		SAE(10)	**0.01s**	-0.23	200.2 (± 75.64)	0.47 (± 0.14)	25.92%	25.79%	0.41	0.8 (± 0.08)	-0.4 (± 0.03)	0.4
		SCC (10)	0.18s	-0.08	**411** (± 168.33)	0.25 (± 0.11)	44.89%	46.05%	0.27	0.82 (± 0.05)	**-0.44** (± 0.03)	0.5
		BDAS(10)	121.44s	**-0.07**	194.3 (± 91.65)	**0.22** (± 0.14)	**90.61%**	**91.08%**	**0.05**	0.75 (± 0.09)	-0.42 (± 0.05)	**0.78**
	ViT (*Cifar10*)	$\mathcal{L}$	136.47s	**-0.03**	50.3 (± 31.58)	**0.11** (± 0.15)	60%	62.5%	0.44	0.01 (± 0.06)	-0.05 (± 0.04)	0.72
		PCA(10)	0.03s	-0.19	67.8 (± 33.86)	0.35 (± 0.28)	61.69%	65.17%	0.43	**0.05** (± 0.07)	-0.08 (± 0.03)	0.67
		SAE(10)	**0.01s**	-0.28	35.1 (± 11.4)	0.48 (± 0.21)	39.78%	40.86%	0.65	-0.07 (± 0.08)	**-0.13** (± 0.03)	0.47
		SCC (10)	2.78s	-0.13	**77.3** (± 74.33)	0.27 (± 0.26)	56.25%	60.29%	0.45	0.01 (± 0.07)	-0.09 (± 0.03)	0.7
		BDAS(10)	61.14s	-0.09	35 (± 16.37)	0.23 (± 0.21)	**80.32%**	**88.75%**	**0.28**	-0.05 (± 0.1)	-0.12 (± 0.03)	**0.91**
	ViT (*Cifar100*)	$\mathcal{L}$	77s	$> -10^{-2}$	179.8 (± 27.57)	**0.03** (± 0.04)	**54.65%**	**56.89%**	**0.31**	-0.01 (± 0.07)	-0.15 (± 0.02)	**0.62**
		PCA(10)	0.02s	-0.18	**198.7** (± 37.49)	0.37 (± 0.2)	19.59%	20.97%	0.61	-0.04 (± 0.16)	-0.17 (± 0.02)	0.28
		SAE(10)	**0.01s**	-0.06	135.2 (± 30.7)	0.19 (± 0.15)	22.11%	22.91%	0.62	-0.02 (± 0.21)	-0.18 (± 0.02)	0.27
		SCC (10)	2.47s	-0.03	198.2 (± 35.36)	0.14 (± 0.13)	21.03%	22.33%	0.59	**0.04** (± 0.14)	-0.16 (± 0.01)	0.23
		BDAS(10)	886.19s	-0.04	108 (± 16.64)	0.16 (± 0.11)	32.28%	33.61%	0.48	-0.09 (± 0.16)	**-0.19** (± 0.02)	0.36
	Llama (*SST5*)	$\mathcal{L}$	54.29s	**-0.01**	32.3 (± 73.35)	**0.1** (± 0.13)	68.42%	**73.68%**	**0.38**	0.16 (± 0.06)	-0.08 (± 0.03)	**1.0**
		PCA(10)	0.01s	-0.3	43.9 (± 42.61)	0.42 (± 0.37)	58.76%	69.07%	0.52	0.21 (± 0.05)	-0.11 (± 0.02)	0.79
		SAE(10)	**0.0s**	-0.4	40 (± 11.1)	0.43 (± 0.48)	**72.92%**	73.61%	0.41	0.23 (± 0.08)	-0.1 (± 0.03)	0.9
		SCC (6)	0.05s	-0.11	**65.33** (± 66.53)	0.26 (± 0.27)	48.11%	70.75%	0.7	**0.25** (± 0.07)	**-0.13** (± 0.03)	0.65
		BDAS(8)	6.17s	-0.22	9.25 (± 5.97)	0.37 (± 0.3)	54.41%	61.76%	0.62	0.24 (± 0.09)	-0.12 (± 0.04)	0.95

Second-last layer. In the second-last layer, $\mathcal{L}$ obtains excellent results in objective function and orthogonality. On *VGG16* (*Cifar10*), *ViT* (*Cifar100*), and *Llama*, it also obtains high clustering scores. Orthogonality and clustering largely go together here: *VGG16* (*Cifar100*), whose blocks are the least orthogonal of the layer, also attains the lowest clustering scores, while on the other models the most orthogonal directions come with high clustering ones. The exception is *ViT* (*Cifar10*), where the most orthogonal directions do not translate into the highest clustering scores, suggesting features that reach a finer granularity than the classes, unlike *ViT* (*Cifar100*), whose supervised pretraining on ImageNet may aligns its representation with classes (Appendix A.8). On intervention scores and Loss Recovered, $\mathcal{L}$ ranks first or second on at least one of them on four models out of five, achieving the strongest Loss Recovered on *ViT* (*Cifar100*) and *Llama*, and is second only to *BDAS* in how often it ranks among the top two, despite not directly optimizing for these scores as *BDAS* does. Like the last layer, *BDAS* is not always capable of retrieving all K features, nor is *SCC*. The runtime of $\mathcal{L}$ remains manageable for large $V_{L,D}$ (in the order of a few minutes).

Varying K. Execution time shows no systematic growth with K, highlighting $\mathcal{L}$'s capability to handle both large inputs and different numbers of features. As K grows, the ϵ-orthogonality score across more blocks becomes increasingly harder to be kept low by any solver, thus the objective function value $f(V_{L,D})$ tends to decrease with K. This structural property of $f(V_{L,D})$ (Appendix A.4) means it should not be compared across values of K, though it remains useful as a benchmark against other methods at a fixed K. As $\mathscr{O}$ increases with K, $\mathcal{L}$ finds progressively larger blocks that a stricter orthogonality bound would have ruled out. The exception is *VGG16* (*Cifar10*) and *Llama*, where

Table 2: Results of $\mathcal{L}$ on the second-last layer with varying K. Boldface/underlined for best/second-best results.

	K	Time	$f(V_{L,D})$	$\mathscr{W}$	$\mathscr{O}$	$\mathscr{C}_P$	$\mathscr{C}_C$	$\mathscr{C}_E$	$\mathscr{I}^S_{abl}$	$\mathscr{I}^{S'}_s$	$\mathscr{L}$
VGG16 (*Cifar10*)	10	65.94s	**0.03**	**199.3** (± 202.12)	**0.11** (± 0.07)	**100%**	**100%**	**0.0**	**0.49** (± 0.08)	**-0.63** (± 0.07)	0.99
	20	21.16s	-0.05	121.1 (± 165.61)	0.19 (± 0.2)	82.33%	82.76%	0.15	0.44 (± 0.09)	-0.58 (± 0.09)	0.98
	30	**20.88s**	-0.09	98.07 (± 141.97)	0.24 (± 0.22)	70.57%	71.49%	0.22	0.43 (± 0.11)	-0.56 (± 0.1)	0.94
	40	21.17s	-0.11	83.47 (± 126.5)	0.28 (± 0.23)	68.71%	69.54%	0.25	0.46 (± 0.11)	-0.57 (± 0.11)	**1.0**
	50	21.28s	-0.14	75.44 (± 114.71)	0.32 (± 0.24)	60.41%	61.27%	0.33	0.45 (± 0.11)	-0.55 (± 0.12)	0.93
VGG16 (*Cifar100*)	10	37.06s	**-0.16**	370.7 (± 110.18)	**0.39** (± 0.11)	**25.02%**	**25.4%**	**0.43**	0.77 (± 0.06)	-0.41 (± 0.03)	0.32
	20	34.84s	-0.2	370.65 (± 87.36)	0.42 (± 0.16)	24.94%	25.21%	**0.43**	0.79 (± 0.07)	-0.43 (± 0.03)	0.52
	30	34.58s	-0.22	376.03 (± 80.02)	0.45 (± 0.17)	23.67%	24.08%	0.45	0.77 (± 0.06)	**-0.44** (± 0.03)	0.61
	40	**33.7s**	-0.24	398.4 (± 105.06)	0.46 (± 0.17)	23.11%	23.47%	0.46	**0.8** (± 0.07)	-0.42 (± 0.03)	0.63
	50	35.25s	-0.25	**406.04** (± 101.94)	0.48 (± 0.17)	22.8%	23.19%	0.46	0.78 (± 0.07)	-0.41 (± 0.03)	**0.65**
ViT (*Cifar10*)	10	136.47s	**-0.03**	50.3 (± 31.58)	**0.11** (± 0.15)	60%	62.5%	**0.44**	**0.01** (± 0.06)	-0.05 (± 0.04)	0.72
	20	**49.76s**	-0.09	61.85 (± 34.05)	0.2 (± 0.23)	**61.7%**	**68.42%**	0.45	**0.01** (± 0.07)	-0.07 (± 0.04)	0.72
	30	50.79s	-0.13	71.2 (± 32.63)	0.25 (± 0.27)	56.04%	61.76%	0.5	**0.01** (± 0.07)	-0.06 (± 0.04)	0.73
	40	50.89s	-0.15	79.9 (± 42.25)	0.29 (± 0.28)	53.19%	58.34%	0.52	**0.01** (± 0.07)	-0.08 (± 0.04)	0.74
	50	51.59s	-0.18	**84.52** (± 39.72)	0.33 (± 0.28)	51.11%	54.98%	0.55	**0.01** (± 0.07)	**-0.09** (± 0.04)	**0.77**
ViT (*Cifar100*)	10	77s	$> -10^{-2}$	179.8 (± 27.57)	**0.03** (± 0.04)	**54.65%**	**56.89%**	**0.31**	-0.01 (± 0.07)	-0.15 (± 0.02)	**0.62**
	20	153.07s	-0.01	192.7 (± 39.72)	0.06 (± 0.07)	46.29%	48.21%	0.39	-0.02 (± 0.08)	-0.14 (± 0.02)	0.56
	30	29.32s	-0.02	212.7 (± 55.72)	0.09 (± 0.1)	38.25%	40.31%	0.47	-0.01 (± 0.09)	-0.16 (± 0.02)	0.5
	40	32.13s	-0.03	211.28 (± 52.73)	0.1 (± 0.13)	38.33%	40.31%	0.46	**0.01** (± 0.1)	-0.17 (± 0.02)	0.51
	50	**27.93s**	-0.04	**224.52** (± 66.26)	0.11 (± 0.14)	34.43%	36.16%	0.49	$< 10^{-2}$ (± 0.1)	**-0.18** (± 0.02)	0.47
Llama (SST5)	10	54.29s	**-0.01**	**32.3** (± 73.35)	**0.1** (± 0.13)	68.42%	73.68%	**0.38**	0.16 (± 0.06)	-0.08 (± 0.03)	**1.0**
	20	**20.88s**	-0.07	21.9 (± 52.99)	0.17 (± 0.21)	**69.09%**	**74.55%**	**0.38**	0.17 (± 0.07)	-0.09 (± 0.04)	0.92
	30	22.09s	-0.11	13.03 (± 7.8)	0.24 (± 0.25)	55.68%	65.91%	0.56	0.18 (± 0.08)	-0.1 (± 0.04)	0.28
	40	20.93s	-0.15	20.32 (± 38.4)	0.28 (± 0.28)	59.49%	70.25%	0.5	0.19 (± 0.08)	-0.11 (± 0.04)	0.73
	50	21.28s	-0.18	21.78 (± 35.05)	0.32 (± 0.29)	51.79%	59.28%	0.63	**0.2** (± 0.07)	**-0.12** (± 0.03)	0.52

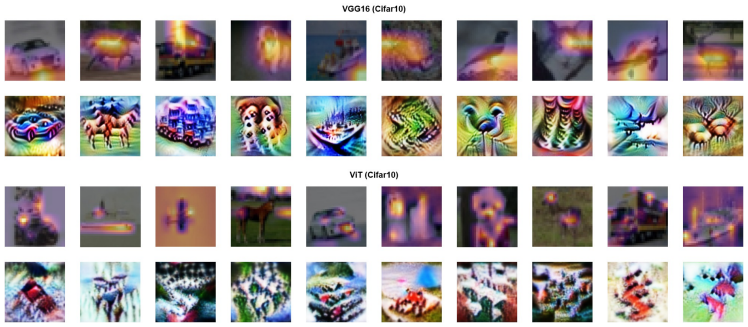


Figure 2: Grad-CAM on the sample most cosine-aligned with each feature's direction, alongside the corresponding synthesized canvas, for *VGG16* and *ViT* (*Cifar10*), second-last layer, L&B-search, $K = 10$.

blocks shrink with K instead: *VGG16* (*Cifar10*) and *Llama* may have no structure over the class level at second-last layer, hence extra blocks can only be smaller fragments of existing ones. *ViT* faces no such constraint, consistently with the finer-than-class granularity already noted above, and its blocks grow with K throughout. Clustering scores generally worsen with K across models, as blocks progressively stop aligning with classes, which is expected once K exceeds their number. Intervention scores and Loss Recovered generally improve instead. The exception is *VGG16* (*Cifar10*), where intervention scores are instead highest at $K = 10$, while on *ViT* (*Cifar10*) they remain highest at higher K throughout. This is consistent with prior evidence that neurons tend to become more class-specific in the final layers (Bau et al., 2017; Zeiler & Fergus, 2014).

Anecdotal evidence. Figure 2 shows a representative example: on *VGG16*, canvases and Grad-CAM largely converge on whole class objects, whereas on *ViT* Grad-CAM instead highlights finer boundary details consistent with more granular features (e.g., the boundary between the ship and the sea, or between the horse and the surrounding grass). More details on this are in Appendix A.13.

7 Conclusion

We advance the state of the art in Mechanistic Interpretability (MI) by devising the first rigorous mathematical pipeline for identifying mechanistic features (MFs): a formal definition of MF, an optimization problem to find MFs in a principled way, a theoretical characterization and algorithms for such a problem. Our algorithms are experimentally shown to be effective and superior to a variety of baselines on numerous datasets and testing scenarios. Future work include extending our theoretical framework to other MI concepts (e.g., circuits), investigating how to properly determine the number of output MFs, and experimenting with further types of NN model and more layers.

AI USE STATEMENT

In this work we used generative AI tools to improve the readability and clarity of the writing and to assist with minor coding tasks such as debugging, documentation and formatting. We *did not* use generative AI tools to develop theoretical models or conceptual frameworks, formulate or prove mathematical claims, design the research methodology, implement the core methods, propose or refine hypotheses or interpret results. All such contributions are the authors' own. Generation of synthetic datasets, assistance with translation, dataset cleaning or reformatting and qualitative or thematic data analysis are not applicable to this work. All AI assisted text and code were reviewed and verified by the authors for accuracy, correctness and originality. We take full responsibility for the final content of this work, including text and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies the internal representations of neural networks and does not involve human subjects, personal or sensitive data. All experiments are run on publicly available datasets (*Cifar10* and *Cifar100* (Krizhevsky et al., 2009), *SST5* (Socher et al., 2013)) and on models that are either trained from scratch by the authors or fine-tuned from publicly released checkpoints, and no new data is released. We see no direct harmful application of the proposed formulation or of the methods that solve it. We note, however, that our hardness results bound what any method for feature identification can guarantee: since the problem is NP-complete and hard to approximate, the features returned in practice are approximations, and interpretations built on them should be treated accordingly, particularly when interpretability is used as evidence about the safety of a model.

REPRODUCIBILITY STATEMENT

All the code needed to reproduce the results of this work is available at the anonymous repository (https://anonymous.4open.science/r/COU-MI-BA39), together with the scripts used to build the activation matrices, to run the proposed methods and the baselines, and to compute every score reported in the tables. The code is fully documented, and the repository includes a README file providing an overview of the code structure and highlighting the most important details. Experiments were run on Intel Core Ultra 7 155U, 32GB RAM and NVIDIA RTX 3050 4GB (laptop). The implementation uses Python 3.8.19, PyTorch 2.4.1, Gurobi 11.0.3 for solving KMF, and scikit-learn for *SCC* and *PCA*. In the repository is included a *.yml* file so that is possible to recreate the described environment using conda. Appendix A.8 describes the models, the way the probing dataset D is sampled, the construction of the activation matrix and the choice of the threshold t, and reports the test accuracy of every model together with the statistics of every activation matrix, so that the inputs of our methods can be reconstructed. Appendix A.9 reports the hyperparameter tuning of the greedy heuristic, listing both the grids explored and the values selected for each model. Appendix A.10 details how the outputs of *PCA*, *SAE* and *BDAS* are adapted to our feature representation, and Appendices A.11 and A.12 detail the adjustments made to the intervention scores and to Loss Recovered, which are applied identically to every method. On the theoretical side, Appendix A.3 reports the complete proof of NP-completeness and inapproximability, Appendix A.5 the linearization of the Maximality constraint, and Appendix A.4 the analysis of the objective function. Appendices A.6 and A.7 report the detailed pseudocode of the greedy heuristic and the role of its parameters. All experiments are run with a fixed random seed, which is reported in the released code.

REFERENCES

Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

Christoph Ambühl, Monaldo Mastrolilli, and Ola Svensson. Inapproximability results for maximum edge biclique, minimum linear arrangement, and sparsest cut. *SIAM Journal on Computing*, 40 (2):567–596, 2011.

Elena Battaglia, Federico Peiretti, and Ruggero Gaetano Pensa. Co-clustering: A survey of the main methods, recent trends, and open problems. *ACM Computing Surveys*, 57(2):1–33, 2024.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 3319–3327, 2017. doi: 10.1109/CVPR.2017.354. URL https://doi.org/10.1109/CVPR.2017.354.

Leonard Bereska and Stratis Gavves. Mechanistic interpretability for AI safety - A review. *Transactions on Machine Learning Research (TMLR)*, 2024. URL https://openreview.net/forum?id=ePUVetPKu6.

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*, 2023a. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2(5):6, 2023b.

Nick Cammarata, Gabriel Goh, Shan Carter, Ludwig Schubert, Michael Petrov, and Chris Olah. Curve detectors. *Distill*, 2020. doi: 10.23915/distill.00024.003.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, 2009.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proc. of Int. Conf. on Learning Representations (ICLR)*, 2021.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

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. URL https://transformer-circuits.pub/2022/toy_model/index.html.

Atticus Geiger, Zhengxuan Wu, Hanson Lu, Josh Rozner, Elisa Kreiss, Thomas Icard, Noah Goodman, and Christopher Potts. Inducing causal structure for interpretable neural networks. In *Proc. of Int. Conf. on Machine Learning (ICML)*, pp. 7324–7338, 2022.

Atticus Geiger, Duligur Ibeling, Amir Zur, Maheep Chaudhary, Sonakshi Chauhan, Jing Huang, Aryaman Arora, Zhengxuan Wu, Noah Goodman, Christopher Potts, et al. Causal abstraction: A theoretical foundation for mechanistic interpretability. *Journal of Machine Learning Research (JMLR)*, 26(83):1–64, 2025.

Liv Gorton. The missing curve detectors of inceptionv1: Applying sparse autoencoders to inceptionv1 early vision. In *ICML 2024 Workshop on Mechanistic Interpretability*, 2024.

Satchel Grant, Simon Jerome Han, Alexa Tartaglini, and Christopher Potts. Addressing divergent representations from causal interventions on neural networks. In *Proc. of Int. Conf. on Learning Representations (ICLR)*, pp. 11330–11356, 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024. doi: 10.48550/ARXIV.2407.21783. URL https://doi.org/10.48550/arXiv.2407.21783.

David Gunning and David Aha. Darpa's explainable artificial intelligence (xai) program. *AI magazine*, 40(2):44–58, 2019.

Tao Han, Kuangrong Hao, Yongsheng Ding, and Xuesong Tang. A sparse autoencoder compressed sensing method for acquiring the pressure array information of clothing. *Neurocomputing*, 275: 1500–1510, 2018.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *Proc. of Int. Conf. on Learning Representations (ICLR)*, 2024.

Adam Karvonen, Benjamin Wright, Can Rager, Rico Angell, Jannik Brinkmann, Logan Smith, Claudio Mayrink Verdun, David Bau, and Samuel Marks. Measuring progress in dictionary learning for language model interpretability with board game models. In *Proc. of Conf. on Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/9736acf007760cc2b47948ae3cf06274-Abstract-Conference.html.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images.(2009), 2009.

Kasper Green Larsen and Jelani Nelson. Optimality of the johnson-lindenstrauss lemma. In *Proc. of IEEE Symp. on Foundations of Computer Science (FOCS)*, pp. 633–638, 2017.

Pasin Manurangsi. Inapproximability of maximum biclique problems, minimum k-cut and densest at-least-k-subgraph from the small set expansion hypothesis. *Algorithms*, 11(1):10, 2018.

Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.

Christoph Molnar. *Interpretable machine learning*. Lulu.com, 2020.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001.

René Peeters. The maximum edge biclique problem is np-complete. *Discrete Applied Mathematics*, 131(3):651–654, 2003.

Theodora-Mara Pîslar, Sara Magliacane, and Atticus Geiger. Combining causal models for more accurate abstractions of neural networks. In *Proc. of Conf. on Causal Learning and Reasoning (CLeaR)*, pp. 114–138, 2025.

Prasad Raghavendra and David Steurer. Graph expansion and the unique games conjecture. In *Proc. of ACM Symp. on Theory of Computing (STOC)*, pp. 755–764, 2010.

Tilman Räuker, Anson Ho, Stephen Casper, and Dylan Hadfield-Menell. Toward transparent ai: A survey on interpreting the inner structures of deep neural networks. In *Proc. Conf. on Secure and Trustworthy Machine Learning (SaTML)*, pp. 464–483, 2023.

Waddah Saeed and Christian Omlin. Explainable AI (XAI): A systematic meta-survey of current challenges and future opportunities. *Knowledge-based systems*, 263:110273, 2023.

Wojciech Samek, Grégoire Montavon, Sebastian Lapuschkin, Christopher J Anders, and Klaus-Robert Müller. Explaining deep neural networks and beyond: A review of methods and applications. *Proceedings of the IEEE*, 109(3):247–278, 2021.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proc. of IEEE Int. Conf. on Computer Vision (ICCV)*, pp. 618–626, 2017.

Peiming Shi, Xiaoci Guo, Dongying Han, and Rongrong Fu. A sparse auto-encoder method based on compressed sensing and wavelet packet energy entropy for rolling bearing intelligent fault diagnosis. *Journal of Mechanical Science and Technology*, 34(4):1445–1458, 2020.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. of Int. Conf. on Learning Representations (ICLR)*, 2015.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1631–1642, 2013.

Shriyank Somvanshi, Md. Monzurul Islam, Amir Rafe, Anannya Ghosh Tusti, Arka Chakraborty, Anika Baitullah, Tausif Islam Chowdhury, Nawaf Alnawmasi, Anandi K. Dutta, and Subasish Das. Bridging the black box: A survey on mechanistic interpretability in AI. *ACM Computing Surveys*, 58(8):210:1–210:35, 2026. doi: 10.1145/3787104. URL `https://doi.org/10.1145/3787104`.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Proc. of Int. Conf. on Machine Learning (ICML)*, pp. 10347–10357, 2021.

Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. Reft: Representation finetuning for language models. *Advances in Neural Information Processing Systems*, 37:63908–63962, 2024.

Stathis Zachos and Hans Heller. A decisive characterization of bpp. *Information and Control*, 69 (1-3):125–135, 1986.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Proc. of European Conf. on Computer Vision (ECCV)*, pp. 818–833. Springer, 2014.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *Proc. of Int. Conf. on Learning Representations (ICLR)*, 2024.

A Appendix

A.1 Example: Need for Maximality

We illustrate with a small hand-crafted instance why ϵ-orthogonality alone does not prevent one feature from being a strict subset of another, and why Maximality is needed as an independent condition.

Consider a layer L with $l = 3$ neurons n_1, n_2, n_3 and a dataset D with $d = 3$ samples s_1, s_2, s_3, with activation matrix

$$V_{L,D} = \begin{array}{c|ccc} & s_1 & s_2 & s_3 \\ \hline n_1 & 0.6 & 0.6 & 0.6 \\ n_2 & 0.8 & 0.8 & 0.8 \\ n_3 & 10.0 & -5.0 & -5.0 \end{array}$$

and threshold $t = 0.5$. Consider the two candidate blocks $F = \langle\{n_1, n_2\}, \{s_1, s_2, s_3\}\rangle$ and $F' = \langle\{n_1\}, \{s_1\}\rangle$, with $F' \subseteq F$, both satisfying Thresholding since every entry they use has magnitude at least t. Their directions, computed as the mean over all $l = 3$ neurons restricted to their own samples, are $\varphi(F) = (0.6, 0.8, 0)$ and $\varphi(F') = (0.6, 0.8, 10.0)$, giving $\cos\big(\varphi(F), \varphi(F')\big) = 1/\sqrt{101} \approx$

0.0995. For any $\epsilon \geq 0.0995$, e.g. $\epsilon = 0.1$, the pair (F, F') therefore satisfies ϵ-orthogonality despite F' being strictly contained in F.

This happens because neuron n_3, though outside both blocks' own neuron sets, still enters both directions, since φ is always computed over all neurons in L: its activation differs sharply between s_1 and $\{s_2, s_3\}$, and it is precisely this difference that pulls $\varphi(F')$ away from $\varphi(F)$ enough to satisfy ϵ-orthogonality. Without Maximality, a solver could then return $\{F, F'\}$ as a valid set of $K = 2$ features, even though F' is nothing more than a fragment of F restricted to one neuron and one sample, and not an independent, irreducible concept. Maximality rules out exactly this configuration, by discarding any block that is a subset of another block in the same solution.

A.2 Search Space Reduction of the KMF problem

Before discussing the proposed algorithms for solving the KMF problem, it is useful to analyze how the search space can be reduced without affecting the problem formulation. First, depending on the reference layer L, the sampled dataset D, and the threshold value t, not all rows and columns of $V_{L,D}$ are relevant. In particular, any row i such that $V_{L,D}[i, j] < t \ \forall j \in 1, \ldots, d$ can be safely removed, since it cannot belong to any feasible feature. Similarly, any column j satisfying $V_{L,D}[i, j] < t \ \forall i \in 1, \ldots, l$ can also be discarded. After this pruning step, the search space reduces from $(2^{l+d})^K$ to $(2^{l'+d'})^K$, where $l' < l$ and $d' < d$.

A second reduction comes from the observation that the ordering of the extracted features is irrelevant. For example, when $K = 3$, the solutions $\mathscr{F}^{L,D} = \{F_1, F_2, F_3\}$ and $\mathscr{F}^{L,D} = \{F_3, F_2, F_1\}$ represent the same solution. Therefore, imposing an ordering on the returned features removes equivalent permutations from the search space.

In the proposed methods, features are sorted according to the minimum value contained in each block, denoted by y_k. More precisely, the ordering constraint $y_1 \leq y_2 \leq \cdots \leq y_K$ is enforced. Under the assumption that all minimum values are distinct, this symmetry-breaking condition reduces the search space by a factor of $K!$, since each unordered set of features admits exactly $K!$ equivalent permutations. If repeated minimum values occur, the reduction factor becomes smaller. Specifically, if there exist multiplicities $m_1, \ldots, m_r$, the reduction factor becomes $\frac{K!}{m_1! \cdots m_r!}$.

Defining V_{max} as the absolute maximum value in $V_{L,D}$, ordering constraints can be inserted in Section 4 formulation as follows:

$$y_k \leq |v_{ij}| + V_{max}(1 - x_{ij}^k) \qquad \begin{matrix} i = 1, \ldots, l \\ j = 1, \ldots, d \\ k = 1, \ldots, K \end{matrix} \tag{10}$$

$$y_k \geq y_{k+1} \qquad k = 1, \ldots, K - 1 \tag{11}$$

Overall, these two observations reduce the search space to at most $\frac{(2^{l'+d'})^K}{K!}$. Although the resulting search space remains exponential in several parameters, these reductions substantially simplify the exploration process and enable the proposed methods to handle more complex instances more efficiently.

A.3 Proof: NP-completness of KMF problem

We now prove the KMF NP-completeness via reduction from the *Max-Edge bilclique Problem* (MBC), which is NP-complete (Peeters, 2003; Ambühl et al., 2011; Manurangsi, 2018). To establish the hardness of our optimization problem, we consider its associated decision version. Given a fixed parameter $W \in \mathbb{N}$, the decision problem asks whether there exists a set of features $\mathscr{F}^{L,D} = \{F_i\}_{i=1}^K$ such that, for all $i = 1, \ldots, K$, the corresponding block satisfies $|V_{N,S}^i| = |N_L^i| \cdot |S_D^i| \geq W$.

Theorem 2. *The* K-Mechanistic Features *problem is NP-complete.*

Proof. We divide the proof in two steps:

1. KMF is NP: Given an instance of the KMF, a candidate solution can be verified in polynomial time. It is possible to verify that features have cardinality greater or equal to W in $O(K \cdot max(|F_K|))$, while for the Definition 1:
   - **Thresholding**: $O(K \cdot max(|F_K|))$, by checking all entries of each block;
   - **Maximality**: $O(K \cdot (K-1) \cdot (max(|N_L^{k'}|, |N_L^k|) + max(|S_D^{k'}|, |S_D^k|))$, by comparing neuron sets and, when necessary, sample sets for each pair of blocks $(V_{N,S}^k, V_{N,S}^{k'})$;
   - **ϵ-orthogonality**: $O(K \cdot (K-1) \cdot 2(l \cdot max(|S_D^{k'}|, |S_D^k|)))$. by computing, for each pair of blocks $(V_{N,S}^k, V_{N,S}^{k'})$, the feature directions over respective samples and over L.
2. the MAX-EDGE BICLIQUE (MBC) can be reduced in polynomial time to the KMF problem:

   Definition 2. MAX-EDGE BICLIQUE (MBC): *Given a bipartite graph $G = (A \cup B, E)$ and $M \in \mathbb{N}$, does G contain a biclique with at least M edges?*

   The NP-completness of MBC is proved in (Peeters, 2003). We rewrite MBC in KMF in polynomial time rewriting G as a adjacency matrix $\mathscr{A}$ ($|A| \times |B|$). Without loss of generality, let be ($\mathscr{A}$, $0 < t < 1$, $\epsilon \geq 0$, $K = 1$) the input of KMF and $W = M$ and suppose there exists a method $\mathscr{M}$ which decide a solution of KMF in polynomial time. Then, any solution verified by $\mathscr{M}$ will be composed of only F_1 and it will respect Maximality and ϵ-orthogonality (since $K = 1$). If we give the adjacency matrix obtained from G to $\mathscr{M}$, since $0 < t < 1$, a cell i, j will be in $F_1 \Leftrightarrow (i, j) \in E$. Furthermore, since all elements in F_1 must be greater or equal than t, all nodes in A induced by F_1 are connected to all nodes in B induced by F_1. In other terms, the graph induced by the block F_1 is a biclique. As the block is composed of at least $W = M$ cells:
   - if solution of KMF is yes, then the one of MBC is yes;
   - if solution of KMF is no, then the one of MBC is no.

   Since we can reduct the MBC in KMF, and MBC is NP-complete, then KMF is NP-complete. □

A.4 OBJECTIVE FUNCTION ANALYSIS OF THE MINLP FORMULATION

This appendix analyzes the behavior of the objective function as the parameter K increases. This analysis is necessary in order to correctly interpret and compare the obtained solutions.

In general, if $ub(\cdot)$ and $lb(\cdot)$ denote the upper and lower bounds of their arguments, the objective function satisfies $ub(f(V_{L,D})) = ub(\mathscr{W}) - lb(\mathscr{O})$ and $lb(f(V_{L,D})) = lb(\mathscr{W}) - ub(\mathscr{O})$, where $\mathscr{W} = \sum_{k=1}^{K} w_k$ and $\mathscr{O} = \lambda \sum_{k_1=1}^{K} \sum_{k_2=k_1+1}^{K} o_{k_1 k_2}$. Since w_k, $o_{k_1 k_2} \in [0, 1] \ \forall \ k, k_1, k_2 \in [1, K], k1 < k2$, if follows that $ub(\mathscr{W}) = K$, $ub(\mathscr{O}) = \lambda \cdot \frac{K(K-1)}{2}$ and $lb(\mathscr{W}) = lb(\mathscr{O}) = 0$. In our experiments we set $\lambda = \frac{2}{(k-1)}$, so $ub(\mathscr{O}) = K$. Therefore, $ub(f(V_{L,D})) = K$, while $lb(f(V_{L,D})) = -K$.

From this analysis follows that $\lim_{K \to \infty} ub(f(V_{L,D})) = +\infty$ suggesting that larger values of K always allow for better solutions. This bound, however, is rarely attained in practice, and the actual objective value obtained by an optimal solution of KMF can behave very differently as K grows. We illustrate this with the following example.

Example 1. *Let be $\mathscr{F} = \{F_1, ..., F_K\}$ the solution of a given instance of the KMF problem such that:*

- $w_{k_1} = w_{k_2} \ \forall \ k_1 \neq k_2; k_1, k_2 \in [1, K]$
- $o_{k_1 k_2} = 0 \ \forall \ k_1, k_2 \in [1, K], k_1 < k_2$
- *each row and each column in $V_{L,D}$ are selected by exactly one block in $\mathscr{F}$*

In this scenario, the non-normalized size of each block is exactly $\frac{ld}{K^2}$. Consequently, the contribution of $\mathscr{W}$ is equal to $K \cdot \frac{ld}{K^2} = \frac{ld}{K}$, which tends to 0 as K increase. On the other hand, the lower bound of remains unchanged, since $o_{k_1 k_2}$ is equal to 0 for every pair of blocks. Consequently, $ub(f(V_{L,D})) = \frac{ld}{K}$ and therefore $\lim_{K \to \infty} ub(f(V_{L,D})) = 0$.

In other words, when the solution structure is preserved, $f(V_{L,D})$ asymptotically decrease to 0 as K increase. Intuitively, this happen because we assumed to have blocks with the highest size such

that $o_{k_1 k 2_2} = 0 \; \forall \; k_1, k_2 \in [1, K], k_1 < k_2$. Thus, increasing K necessarily reduces the size of each block, since otherwise the the orthogonality among blocks would increase.

While the upper bound $ub(f(V_{L,D}))$ grows without bound as K increases, the previous example shows that the actual objective value attained by a solution can behave very differently, tending to 0 instead in optimistic scenario. Nevertheless, it still represents an optimistic case, one in which the size of each block is maximized while keeping $\mathscr{O}$ equal to its lower bound independently of K. Hence, an improvement in the objective function could only occur if the gain induced by larger block sizes exceeded the additional penalty introduced by $\mathscr{O}$ as a consequence of increased overlap among blocks. Under the MI core assumption, however, this situation is unlikely to happen in realistic settings, since feature should be ϵ-orthogonal. This observation suggests that any optimization method which aims to find features of a NN w.r.t. MI core assumption, naturally tend to produce smaller objective values as K increases. Consequently, objective values obtained with different values of K should not be directly compared (yet, the objective function remain a valid benchmark among baselinse with fixed K).

Because of this behavior, it remains unclear which stopping condition should be adopted to determine the actual number of $\mathscr{K}$ features learned by the NN. To the best of our knowledge, no existing method can exactly identify $\mathscr{K}$, making this limitation consistent with the current literature. As discussed in Section 6, candidate criterion could be MI evaluation scores. Since the primary goal of this work is to introduce a new theoretical framework and scalable methods for solving the KMF problem, we leave the investigation of appropriate stopping criteria for future research.

A.5 Linearization of the Maximality constraint

Constraint (3) imposes, for every ordered pair $k_1 \neq k_2$, that $F_{k_1} \not\subseteq F_{k_2}$. Since blocks are submatrices of $V_{L,D}$, $F_{k_1} \subseteq F_{k_2}$ holds if and only if $N_L^{k_1} \subseteq N_L^{k_2}$ and $S_D^{k_1} \subseteq S_D^{k_2}$, so it suffices that F_{k_1} contains at least one neuron or one sample not selected by F_{k_2}. This is exactly what (3) counts, since, being n_i^k and s_j^k binary, $\max(0, n_i^{k_1} - n_i^{k_2})$ equals 1 if and only if $n_i^{k_1} = 1$ and $n_i^{k_2} = 0$, and analogously for samples. Note that (3) also rules out two identical blocks, since $F_{k_1} = F_{k_2}$ implies $F_{k_1} \subseteq F_{k_2}$. The only non-linearity lies in the $\max(0, n_i^{k_1} - n_i^{k_2})$, which, on binary arguments, is equivalent to the product $n_i^{k_1}(1 - n_i^{k_2})$, and can therefore be linearized through the standard reformulation of the product of binary variables (analogously for $\max(0, s_j^{k_1} - s_j^{k_2})$). We thus introduce the support variables $u_i^{k_1 k_2}, v_j^{k_1 k_2} \in [0, 1]$, and replace (3) with the following constraints:

$$u_i^{k_1 k_2} \leq n_i^{k_1}, \;\; u_i^{k_1 k_2} \leq 1 - n_i^{k_2} \;\; \begin{array}{l} i = 1, \ldots, l \\ k_1, k_2 = 1, \ldots, K, \; k_1 \neq k_2 \end{array} \tag{12}$$

$$v_j^{k_1 k_2} \leq s_j^{k_1}, \;\; v_j^{k_1 k_2} \leq 1 - s_j^{k_2} \;\; \begin{array}{l} j = 1, \ldots, d \\ k_1, k_2 = 1, \ldots, K, \; k_1 \neq k_2 \end{array} \tag{13}$$

$$\sum_{i=1}^{l} u_i^{k_1 k_2} + \sum_{j=1}^{d} v_j^{k_1 k_2} \geq 1 \;\; k_1, k_2 = 1, \ldots, K, \; k_1 \neq k_2 \tag{14}$$

Overall, the linearization adds $K(K-1)(l+d)$ continuous variables and $K(K-1)\big(2(l+d)+1\big)$ linear constraints, without introducing any additional integer variable.

A.6 L&B-search Algorithm

In the following, we report a detailed pseudocode of the L&B-search algorithm, presented in Section 4.

Algorithm 3 L&B-search

Input: $V_{L,D} \in \mathbb{R}^{l \times d}$, $t \in \mathbb{R}^+$, $p \in (0, 1]$, $K, m, \rho \in \mathbb{N}$ ($p = 0.25$, $\rho = 1\,000$ by default)
Output: K MFs (Def. 1) $\mathscr{F}^{L,D}$ represented by $\mathcal{F} = \{F_1, ..., F_K\}$, with F_k submatrix of $V_{L,D}$, $\forall k = 1,..., K$

$\mathcal{B} \leftarrow \emptyset$
for $i \in |L| = l$ **do**
  $B_i \leftarrow \{V_{L,D}[i, j] \mid |V_{L,D}[i, j]| \geq t \; \forall j = 1, ..., |D|\}$
  $\mathcal{B} \leftarrow \mathcal{B} \cup \{B_i\}$
end for
repeat
  $\mathcal{C} \leftarrow \emptyset$
  for all $(B_i, B_j) \in \mathcal{B} \times \mathcal{B}$ s.t. $i < j$ **do**
    for all $(\circ_N, \circ_S)$ **do**
      $B_n \leftarrow B_i \circ B_j$
      if $B_n \neq \emptyset$ **and** $\min(B) \geq t$ **then**
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{(B_n, \phi(\langle B_i, B_j \rangle))\}$
      end if
    end for
  end for
  Sort $\mathcal{C}$ by ϕ in non-increasing order
  $\mathcal{C}' \leftarrow \{B_i \in \mathcal{C} \mid i \leq \eta\}$, where $\eta = \min(p \cdot |\mathcal{C}|, \rho)$
  $\mathcal{B} \leftarrow \mathcal{B} \cup \mathcal{C}'$
until no block is added to $\mathcal{B}$
$\mathcal{B} \leftarrow$ Closure($\mathcal{B}$)
$\mathcal{B} \leftarrow$ Maximality($\mathcal{B}$)
$\mathcal{B} \leftarrow$ DirectionFiltering($\mathcal{B}$)
for each block $B_i \in \mathcal{B}$ **do**
  Compute block score $\psi(B_i)$
end for
Sort $\mathcal{B}$ by ψ in non-increasing order
$\widehat{\mathcal{B}} \leftarrow$ top-m blocks of $\mathcal{B}$
$\mathcal{F} \leftarrow$ selected K blocks from $\widehat{\mathcal{B}}$ (gurobi)

A.7 Computational Complexity of L&B-search and the Role of the m Parameter

The final optimization stage of L&B-search requires selecting K blocks from the b candidates identified by the first-stage pre-selection (by $\phi(\cdot)$), yielding a search space of $\binom{b}{K}$ possible subsets. To reduce this search space, the b blocks are sorted in non-increasing order according to the score $\psi(\cdot)$, and only the top m blocks are retained, reducing the number of subsets to $\binom{m}{K}$. Note that, if one want to annhilate execution of this final step, one could set $b = K$. In this way, the search space becomes $\binom{K}{K} = 1$, and thus the solution found by any solver is the same as the one identified at the end of the previous step.

A.8 Models, Datasets, and Activation Matrices

We consider five model-dataset combinations, namely *VGG16* and *ViT* on both *Cifar10* and *Cifar100* (Krizhevsky et al., 2009), and *Llama* 3.2 1B (Grattafiori et al., 2024) on *SST5* (Socher et al., 2013). For each of them, we analyze the last layer, whose neurons correspond to the classes of the task, and the second-last layer, which feeds it. *VGG16* follows the architecture of Simonyan & Zisserman (2015), adapted to the 32×32 images of *Cifar10* and *Cifar100*, which are smaller than the 224×224 ImageNet inputs it was designed for. The convolutional feature extractor is therefore reduced to four blocks, for a total of ten convolutional layers, and the fully connected layers are shrunk accordingly, mapping the resulting 2048-dimensional representation to 512 and then to 256 neurons, rather than to 4096 as in the original architecture. The second-last layer thus has $l = 256$ neurons, while the last layer has one neuron per class, namely $l = 10$ on *Cifar10* and $l = 100$ on *Cifar100*, and both models are trained from scratch. *ViT* follows Dosovitskiy et al. (2021) with 12 encoder layers, 3 attention heads and an embedding size of 192, i.e. the smallest standard configuration, adopted by DeiT-Tiny (Touvron et al., 2021), which keeps the model proportionate to the

size of the Cifar datasets. On *Cifar10*, *ViT* is trained from scratch with a patch size of 4, splitting each image into an 8×8 grid of patches. On *Cifar100*, *ViT* is instead fine-tuned from DeiT-Tiny pretrained on ImageNet-1k (Deng et al., 2009), since ViTs are known to require large amounts of data when trained from scratch (Dosovitskiy et al., 2021), and images are upscaled to 224×224 to match the patch size of 16 of the pretrained model. In both cases, the classification head consists of a LayerNorm applied to the CLS token, followed by a linear classifier, so that the second-last layer has $l = 192$ neurons, while the last layer contains the class logits, with $l = 10$ and $l = 100$ neurons on *Cifar10* and *Cifar100* respectively. Unlike the other models, *ViT* applies no ReLU after this LayerNorm, so its second-last layer activations are signed rather than non-negative. The five combinations thus cover both regimes, and Definition 1 applies to both unchanged, since Thresholding constrains magnitudes and φ keeps the sign of the activations. Finally, *Llama* is fine-tuned on *SST5*, adding an MLP head at the top of it. Such MLP head is made of a hidden layer of 256 neurons with ReLU activation, followed by a linear classifier with one neuron per sentiment class. The size of the hidden layer is chosen to keep the second-last layer comparable with those of *VGG16* and *ViT*, so that $l = 256$ on the second-last layer and $l = 5$ on the last layer. The resulting test accuracies are 76.72% and 70.86% for *VGG16* on *Cifar10* and *Cifar100*, 79.14% and 84.64% for *ViT*, and 55.11% for *Llama* on *SST5*.

For each model, the probing dataset D is built by drawing at random, with a fixed seed, $d' = 50$ test samples for each class $c \in C$, so that $d = d'|C|$, namely $d = 500$ on *Cifar10*, $d = 5000$ on *Cifar100*, and $d = 250$ on *SST5*. The same D is used for both last and second-last layers of a model. The resulting matrix is normalized, so that all its entries lie in $[-1, 1]$, and in $[0, 1]$ on the layers whose activations are non-negative, and the threshold t used in Section 6 is computed as a quantile of the magnitudes of its non-zero entries. On the second-last layer, we use the 90th percentile. On the last layer, instead, each neuron corresponds to a class, and each sample is expected to activate mostly the neuron of its predicted class, i.e. one entry out of $|C|$ in each column, so we use the $(1 - 1/|C|)$-quantile, which coincides with the 90th percentile on *Cifar10*. A fixed 90th percentile would instead leave on average ten entries per column above t on *Cifar100*, most of them close to zero, and at most half of the samples with their predicted class above t on *SST5*. Table 3 reports, for each activation matrix, its size, the mean and standard deviation of its entries, the threshold t, and the percentages of entries above t and equal to 0. On the last layer, no entry is null, since softmax outputs are strictly positive, and the mean is close to $1/|C|$, since each column is a probability distribution over $|C|$ classes. Consistently, a fraction $1/|C|$ of the entries lies above t by construction, i.e. one entry per column on average. On the second-last layer, between 46% and 65% of the entries are null on *VGG16* and *Llama*, whose analysed layer is followed by a ReLU, while none is null on *ViT*, whose second-last layer is a LayerNorm and whose entries are therefore signed and almost evenly split between the two signs. On the former, one tenth of the non-zero entries lies above t by construction, so the percentage of entries above t drops accordingly, ranging between 3.54% and 5.36%, whereas on the latter no entry is null and that percentage is exactly 10%. The mean close to 0 reported for *ViT* follows from the same fact, since its positive and negative entries cancel out, while t is computed on magnitudes and stays comparable to that of the other models.

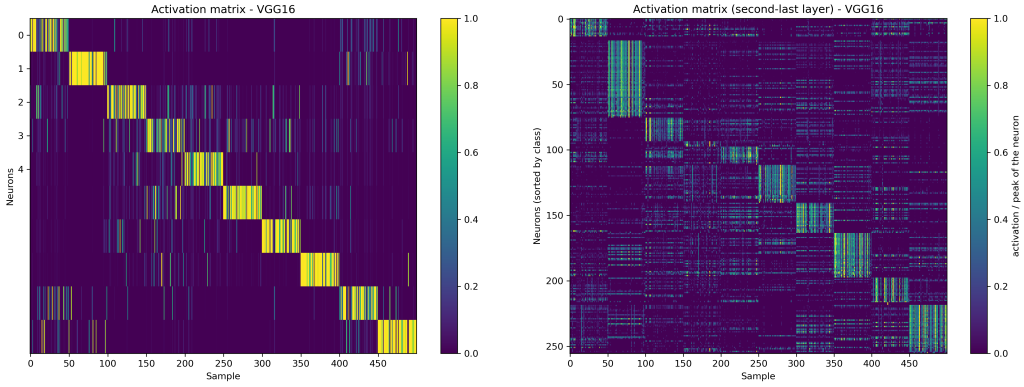
Figures 3 to 7 show the activation matrices of every model, with samples grouped by class. On the second-last layer, neurons are sorted by the class on which they activate the most. On the last layer, a block diagonal pattern is clearly visible for every model, as expected by construction, since each neuron corresponds to a class and samples are grouped by class, so that off-diagonal entries only correspond to misclassified or uncertain samples. On the second-last layer this pattern is less evident, but still present, since neurons no longer correspond to classes and can respond to samples of different ones. In particular, on *Cifar10* the pattern is more visible on *VGG16* than on *ViT*, in line with the results discussed in Section 6, where *VGG16* (*Cifar10*) is the only model whose intervention scores are highest at $K = 10$, consistently with its neurons being more class-specific (see also Figure 2). A similar behavior can also be observed on *Cifar100*, though less markedly.

A.9 HYPERPARAMETER TUNING

Prior to final experiments, which results are reported in Section 6, we conducted a tuning of L&B-search hyperparameters. Specifically, we tuned, in order:

Table 3: Statistics of the activation matrices on the last and second-last layer. t is the $(1 - 1/|C|)$-quantile of the non-zero entries on the last layer, and their 90th percentile on the second-last layer.

	Model	Dataset	l	d	Mean	Std	t	$\% \geq t$	$\% = 0$
Last layer	*VGG16*	*Cifar10*	10	500	0.1000	0.2564	0.3575	10.02%	0.00%
	ViT	*Cifar10*	10	500	0.1046	0.2232	0.3528	10.00%	0.00%
	VGG16	*Cifar100*	100	5000	0.0100	0.0893	0.3498	1.00%	0.00%
	ViT	*Cifar100*	100	5000	0.0103	0.0841	0.2706	1.00%	0.00%
	Llama	*SST5*	5	250	0.2023	0.2640	0.4311	20.00%	0.00%
Second-last layer	*VGG16*	*Cifar10*	256	500	0.0286	0.0666	0.1951	3.54%	64.62%
	ViT	*Cifar10*	192	500	0.0002	0.1681	0.2702	10.00%	0.00%
	VGG16	*Cifar100*	256	5000	0.0575	0.0898	0.2410	5.36%	46.43%
	ViT	*Cifar100*	192	5000	-0.0015	0.1907	0.3129	10.00%	0.00%
	Llama	*SST5*	256	250	0.0765	0.1379	0.4261	3.83%	61.70%



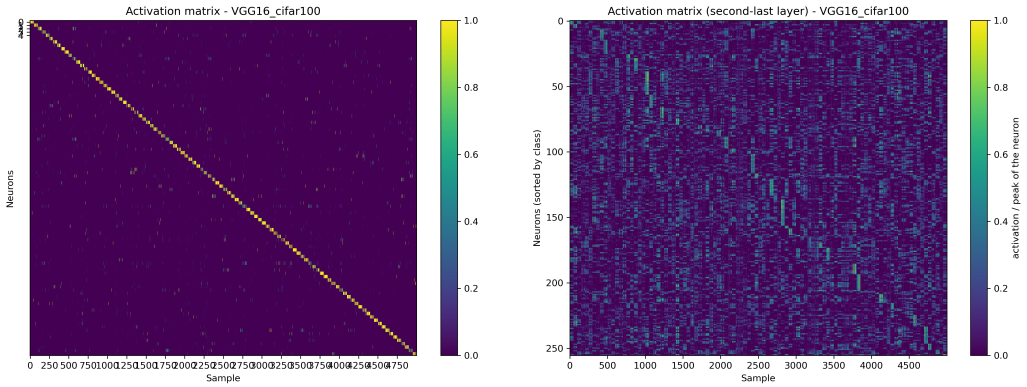
(a) Last layer

(b) Second-last layer

Figure 3: Activation matrices of *VGG16* on *Cifar10*.

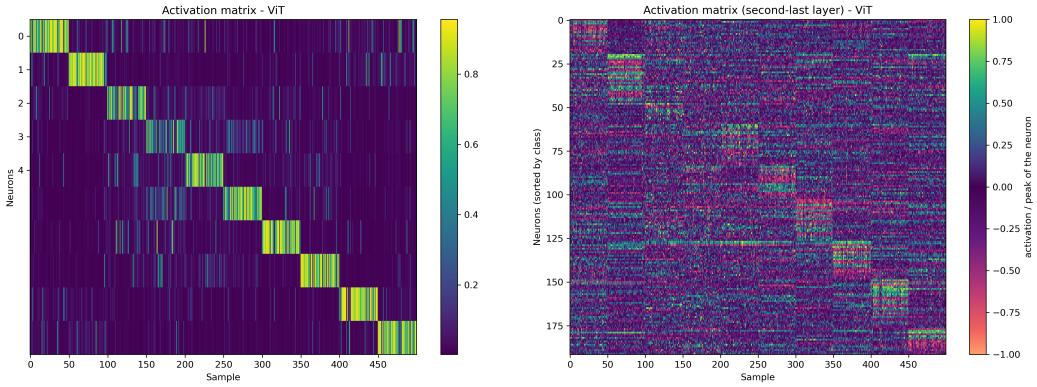
- **Composition method**: as described in Section **??**, the blocks in L&B-search first-stage pre-selection are composed accordingly to $\circ_N$ and $\circ_S$, which both can assume the role of union and intersection. In our tuning we have considered:
  - **NUSI**: $\circ_N = \cup$, $\circ_S = \cap$.
  - **NISU**: $\circ_N = \cap$, $\circ_S = \cup$.
  - **NSALL**: all combinations of $\circ_N, \circ_S \in \{\cup, \cap\}$
- **Solver pre-selection hyperparameter** (m): after the first-stage pre-selection only the first m blocks are retained, accordingly with the score $\psi(\cdot)$ described in Section **??**. Note that this parameter may have an effect on execution time, as described in Appendix A.7. We considered $m = \{70, 80, 90, 100\}$, after fixing the composition method.
- **Composition score regulator terms**: the composing score computed during the first-stage pre-selection is also determined by regulator terms λ_{size} and λ_{div}. In this final stage, after fixing the composition method and m, we tune such terms, considering the combinations $(\lambda_{size}, \lambda_{div}) = \{(0.001, 0.999), (0.01, 0.99), (0.1, 0.9), (0.2, 0.8), \ldots, (0.8, 0.2), (0.9, 0.1), (0.99, 0.01), (0.999, 0.001)\}$.

During the tuning of hyperparameters, those that have not yet been tuned are set to their default values, namely $m = 70$ and $\lambda_{size} = \lambda_{div} = 0.5$. The tuning is performed on both the last and second-last layer; only the second-last layer results are reported, however, since this tuning had no effect on last-layer experiments. Furthermore, in the following tables, only the objective function values are highlighted, as this is the parameter that drove the tuning of the hyperparameters. As described in Section 5, and as visible in tables, the result of the tuning for the second-last layer is the following:



(a) Last layer (b) Second-last layer

Figure 4: Activation matrices of *VGG16* on *Cifar100*.



(a) Last layer (b) Second-last layer

Figure 5: Activation matrices of *ViT* on *Cifar10*.

- ***VGG16*** **(*Cifar10*)**: composition method = (NSALL), $m = 100$, $\lambda_{size} = 0.9$, $\lambda_{div} = 0.1$
- ***VGG16*** **(*Cifar100*)**: composition method = (NUSI), $m = 100$, $\lambda_{size} = \lambda_{div} = 0.5$
- ***ViT*** **(*Cifar10*)**: composition method = (NSALL), $m = 100$, $\lambda_{size} = \lambda_{div} = 0.5$
- ***ViT*** **(*Cifar100*)**: composition method = (NUSI), $m = 100$, $\lambda_{size} = 0.999$, $\lambda_{div} = 0.001$
- ***Llama*** **(*SST5*)**: composition method = (NSALL), $m = 100$, $\lambda_{size} = \lambda_{div} = 0.5$.

For the last layer we used composition method = (NSALL), $m = 100$, and $\lambda_{size} = \lambda_{div} = 0.5$ for all models.

A.9.1 Results on composition method

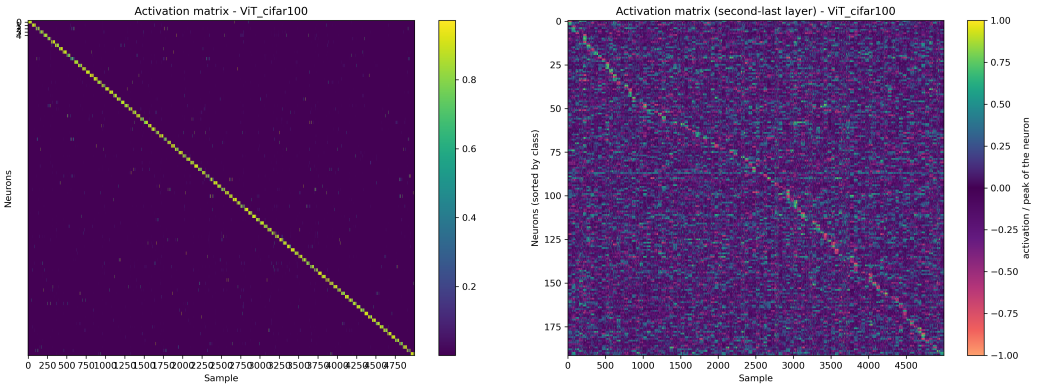
In Table 4, we report the results of tuning of the composition method.

A.9.2 Results on solver pre-selection hyperparameter

In Table 5 are reported the results of tuning of the solver pre-slection hyperparameter m.

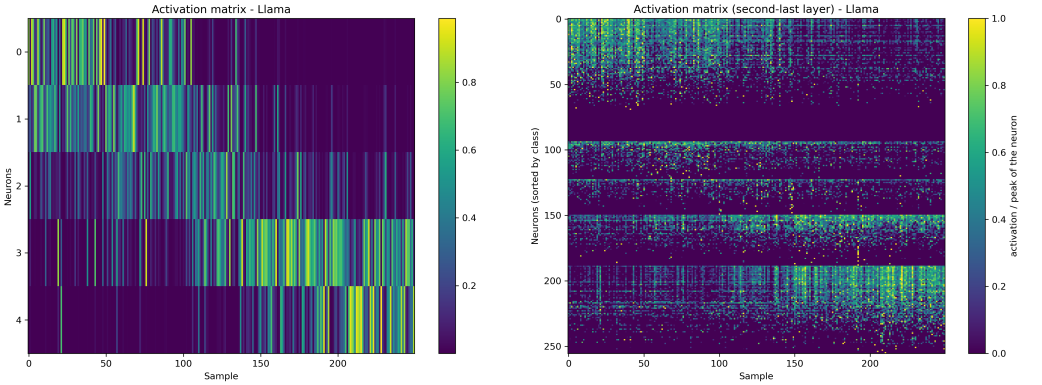
A.9.3 Results on composition score regulator terms

In Tables 6, 7, 8, 9, and 10, are reported the results of tuning of the composition score regulator terms λ_{div} and λ_{size}. Because of the size of this last table, we split it in three tables, one for each model.



(a) Last layer (b) Second-last layer

Figure 6: Activation matrices of *ViT* on *Cifar100*.



(a) Last layer (b) Second-last layer

Figure 7: Activation matrices of *Llama* on *SST5*.

Table 4: $\mathscr{H}$ results on VGG16 (*Cifar10* and *Cifar100*), ViT (*Cifar10* and *Cifar100*) and Llama (*SST5*) second-last layer for the three composition methods.

	$\circ_N, \circ_S$	Time	$f(V_{L,D})$	μ_{size}	μ_{dot}	$\mathscr{C}_P$	$\mathscr{C}_C$	$\mathscr{C}_E$	$\mathscr{I}^S_{abl}$	$\mathscr{I}^{S'}_{\sigma}$	$\mathscr{L}$
VGG16 (*Cifar10*)	NUSI	16.87s	**0.03**	198.8 ($\pm$201.56)	0.11 ($\pm$0.07)	100%	100%	0.0	0.49 ($\pm$0.08)	-0.63 ($\pm$0.07)	1.0
	NISU	0.11s	<u>0.02</u>	190.5 ($\pm$197.76)	0.11 ($\pm$0.07)	100%	100%	0.0	0.5 ($\pm$0.09)	-0.64 ($\pm$0.08)	1.0
	NSALL	34.18s	**0.03**	198.8 ($\pm$201.56)	0.11 ($\pm$0.07)	100%	100%	0.0	0.49 ($\pm$0.08)	-0.63 ($\pm$0.07)	1.0
VGG16 (*Cifar100*)	NUSI	36.07s	**-0.16**	413.7 ($\pm$132.91)	0.4 ($\pm$0.1)	22.8%	23.14%	0.46	0.77 ($\pm$0.07)	-0.41 ($\pm$0.03)	0.24
	NISU	1.18s	-0.31	302 ($\pm$19.05)	0.55 ($\pm$0.12)	12.81%	13.01%	0.62	0.82 ($\pm$0.1)	-0.39 ($\pm$0.05)	0.36
	NSALL	83.27s	<u>-0.17</u>	386.3 ($\pm$92.41)	0.4 ($\pm$0.13)	23.2%	23.53%	0.45	0.79 ($\pm$0.06)	-0.41 ($\pm$0.04)	0.33
ViT (*Cifar10*)	NUSI	14.07s	<u>-0.04</u>	65.5 ($\pm$45.03)	0.13 ($\pm$0.17)	69.92%	75.61%	0.33	0.03 ($\pm$0.06)	-0.04 ($\pm$0.03)	0.82
	NISU	0.15s	-0.08	41.1 ($\pm$37.08)	0.22 ($\pm$0.19)	43.9%	50%	0.65	0.01 ($\pm$0.06)	-0.09 ($\pm$0.04)	0.59
	NSALL	42.23s	**-0.03**	54.2 ($\pm$31.23)	0.11 ($\pm$0.16)	62.5%	65.62%	0.42	0.01 ($\pm$0.05)	-0.05 ($\pm$0.04)	0.73
ViT (*Cifar100*)	NUSI	66.65s	**-0.01**	193.8 ($\pm$79.5)	0.06 ($\pm$0.06)	38.1%	40.09%	0.49	0.04 ($\pm$0.11)	-0.15 ($\pm$0.03)	0.45
	NISU	1.34s	-0.12	446.1 ($\pm$89.91)	0.35 ($\pm$0.06)	9.73%	10.09%	0.78	-0.01 ($\pm$0.15)	-0.17 ($\pm$0.01)	0.13
	NSALL	75.68s	<u>-0.02</u>	192.2 ($\pm$80.63)	0.07 ($\pm$0.06)	37.38%	39.42%	0.5	0.02 ($\pm$0.12)	-0.15 ($\pm$0.03)	0.44
Llama (*SST5*)	NUSI	12.7s	**-0.01**	32.3 ($\pm$73.35)	0.1 ($\pm$0.13)	68.42%	73.68%	0.38	0.16 ($\pm$0.06)	-0.09 ($\pm$0.03)	1.0
	NISU	0.3s	<u>-0.02</u>	7.5 ($\pm$4.13)	0.09 ($\pm$0.12)	84.62%	92.31%	0.2	0.17 ($\pm$0.09)	-0.09 ($\pm$0.03)	1.08
	NSALL	23.18s	**-0.01**	32.3 ($\pm$73.35)	0.1 ($\pm$0.13)	68.42%	73.68%	0.38	0.16 ($\pm$0.06)	-0.09 ($\pm$0.03)	1.0

Table 5: $\mathscr{H}$ results on VGG16 (*Cifar10* and *Cifar100*), ViT (*Cifar10* and *Cifar100*) and Llama (*SST5*) second-last layer for different values of the solver pre-selection parameter m.

	m	Time	$f(V_{L,D})$	μ_{size}	μ_{dot}	$\mathscr{C}_P$	$\mathscr{C}_C$	$\mathscr{C}_E$	$\mathscr{I}^S_{abl}$	$\mathscr{I}^{S'}_{\sigma}$	$\mathscr{L}$
VGG16 (*Cifar10*)	70	34.18s	**0.03**	198.8 (± 201.56)	0.11 (± 0.07)	100%	100%	0.0	0.49 (± 0.08)	-0.63 (± 0.07)	1.0
	80	21.54s	**0.03**	198.8 (± 201.56)	0.11 (± 0.07)	100%	100%	0.0	0.49 (± 0.08)	-0.63 (± 0.07)	1.0
	90	21.42s	**0.03**	198.8 (± 201.56)	0.11 (± 0.07)	100%	100%	0.0	0.49 (± 0.08)	-0.63 (± 0.06)	1.0
	100	21.27s	**0.03**	198.8 (± 201.56)	0.11 (± 0.07)	100%	100%	0.0	0.49 (± 0.08)	-0.63 (± 0.07)	1.0
VGG16 (*Cifar100*)	70	36.07s	-0.17	413.7 (± 132.91)	0.4 (± 0.1)	22.8%	23.14%	0.46	0.77 (± 0.07)	-0.41 (± 0.03)	0.24
	80	36.31s	-0.17	413.7 (± 132.91)	0.4 (± 0.1)	22.8%	23.14%	0.46	0.77 (± 0.07)	-0.41 (± 0.03)	0.24
	90	36.91s	**-0.16**	370.7 (± 110.18)	0.39 (± 0.11)	25.02%	25.4%	0.43	0.77 (± 0.06)	-0.4 (± 0.03)	0.32
	100	37.06s	**-0.16**	370.7 (± 110.18)	0.39 (± 0.11)	25.02%	25.4%	0.43	0.77 (± 0.06)	-0.4 (± 0.03)	0.32
ViT (*Cifar10*)	70	42.23s	-0.05	54.2 (± 31.23)	0.11 (± 0.16)	62.5%	65.62%	0.42	0.01 (± 0.05)	-0.05 (± 0.04)	0.73
	80	42.49s	-0.04	50.3 (± 31.58)	0.11 (± 0.15)	59.26%	62.96%	0.44	0.02 (± 0.06)	-0.05 (± 0.04)	0.71
	90	119.67s	-0.04	50.3 (± 31.58)	0.11 (± 0.15)	59.26%	62.96%	0.44	0.02 (± 0.06)	-0.05 (± 0.04)	0.71
	100	136.47s	**-0.03**	50.3 (± 31.58)	0.11 (± 0.15)	60%	62.5%	0.44	0.01 (± 0.06)	-0.05 (± 0.04)	0.72
ViT (*Cifar100*)	70	66.65s	-0.02	193.8 (± 79.5)	0.06 (± 0.06)	38.1%	40.09%	0.49	0.04 (± 0.11)	-0.15 (± 0.03)	0.45
	80	35.15s	-0.02	193.8 (± 79.5)	0.06 (± 0.06)	38.1%	40.09%	0.49	0.04 (± 0.11)	-0.15 (± 0.03)	0.45
	90	60.68s	-0.02	193.8 (± 79.5)	0.06 (± 0.06)	38.1%	40.09%	0.49	0.04 (± 0.11)	-0.15 (± 0.03)	0.45
	100	45.45s	**-0.01**	199.4 (± 79.88)	0.06 (± 0.06)	35.34%	36.84%	0.49	0.02 (± 0.1)	-0.15 (± 0.03)	0.42
Llama (*SST5*)	70	23.18s	**-0.01**	32.3 (± 73.35)	0.1 (± 0.13)	68.42%	73.68%	0.38	0.16 (± 0.06)	-0.09 (± 0.03)	1.0
	80	58.68s	**-0.01**	32.3 (± 73.35)	0.1 (± 0.13)	68.42%	73.68%	0.38	0.16 (± 0.06)	-0.08 (± 0.03)	1.0
	90	53.36s	**-0.01**	32.3 (± 73.35)	0.1 (± 0.13)	68.42%	73.68%	0.38	0.16 (± 0.06)	-0.08 (± 0.03)	1.0
	100	54.29s	**-0.01**	32.3 (± 73.35)	0.1 (± 0.13)	68.42%	73.68%	0.38	0.16 (± 0.06)	-0.08 (± 0.03)	1.0

Table 6: $\mathscr{H}$ results on VGG16 (*Cifar10*) second-last layer for different values of the composition score regulator terms λ_{size} and λ_{div}.

$(\lambda_{size}, \lambda_{div})$	Time	$f(V_{L,D})$	μ_{size}	μ_{dot}	$\mathscr{C}_P$	$\mathscr{C}_C$	$\mathscr{C}_E$	$\mathscr{I}^S_{abl}$	$\mathscr{I}^{S'}_{\sigma}$	$\mathscr{L}$
(0.001, 0.999)	17.46s	0.01	198.8 (± 201.56)	0.11 (± 0.07)	100%	100%	0.0	0.49 (± 0.08)	-0.63 (± 0.07)	1.0
(0.01, 0.99)	18.33s	0.01	198.8 (± 201.56)	0.11 (± 0.07)	100%	100%	0.0	0.49 (± 0.08)	-0.63 (± 0.06)	1.0
(0.1, 0.9)	23.29s	0.01	198.8 (± 201.56)	0.11 (± 0.07)	100%	100%	0.0	0.49 (± 0.08)	-0.63 (± 0.07)	1.0
(0.2, 0.8)	21.97s	0.01	198.8 (± 201.56)	0.11 (± 0.07)	100%	100%	0.0	0.49 (± 0.08)	-0.63 (± 0.07)	1.0
(0.3, 0.7)	25.98s	0.01	198.8 (± 201.56)	0.11 (± 0.07)	100%	100%	0.0	0.49 (± 0.08)	-0.62 (± 0.07)	1.0
(0.4, 0.6)	25.25s	0.01	198.8 (± 201.56)	0.11 (± 0.07)	100%	100%	0.0	0.49 (± 0.08)	-0.63 (± 0.07)	1.0
(0.5, 0.5)	21.27s	0.01	198.8 (± 201.56)	0.11 (± 0.07)	100%	100%	0.0	0.49 (± 0.08)	-0.63 (± 0.07)	1.0
(0.6, 0.4)	22.83s	0.01	198.8 (± 201.56)	0.11 (± 0.07)	100%	100%	0.0	0.49 (± 0.08)	-0.63 (± 0.07)	1.0
(0.7, 0.3)	24.66s	0.02	199.5 (± 202.08)	0.11 (± 0.07)	100%	100%	0.0	0.49 (± 0.08)	-0.63 (± 0.07)	1.0
(0.8, 0.2)	64.81s	0.02	199.5 (± 202.08)	0.11 (± 0.07)	100%	100%	0.0	0.49 (± 0.08)	-0.63 (± 0.06)	1.0
(0.9, 0.1)	65.94s	**0.03**	199.3 (± 202.12)	0.11 (± 0.07)	100%	100%	0.0	0.49 (± 0.08)	-0.63 (± 0.07)	1.0
(0.99, 0.01)	77.84s	**0.03**	197.1 (± 201.89)	0.11 (± 0.07)	100%	100%	0.0	0.5 (± 0.09)	-0.63 (± 0.07)	1.0
(0.999, 0.001)	74.21s	**0.03**	197.1 (± 201.89)	0.11 (± 0.07)	100%	100%	0.0	0.5 (± 0.09)	-0.64 (± 0.07)	1.0

A.10 Adapting PCA, SAE, and BDAS Outputs to the Proposed MF Definition

Following the methodology proposed in prior work (Gorton, 2024; Huben et al., 2024), but adopting a thresholding strategy different from the top-K approach, we adapted the *PCA* and *SAE* baselines as described in the rest of the appendix.

In the case of *PCA*, features correspond to principal components. Let $Q_{90}(\cdot)$ denotes the 90-percentile of a distribution, W_k the distribution of neurons weight defining the kth PC, and S_k the distribution of kth component activation along samples. Neuron i is assigned to feature k if $w_{ik} \geq Q_{90}(W_k)$ and sample j is assigned to feature k if $s_{jk} \geq Q_{90}(S_k)$. For *SAE*, latent directions are treated as features. Samples S^k_D are selected using the same criterion as above, while neuron i is included in feature k if there exists a sample $j \in S^k_D$ such that: $r^j_{ik} \geq Q_{90}(R^j_k)$, where R^j_k is the distribution of neurons contributions to direction k when reconstructing the sample j.

Table 7: $\mathscr{H}$ results on VGG16 (*cifar100*) second-last layer for different values of the composition score regulator terms λ_{size} and λ_{div}.

$(\lambda_{size}, \lambda_{div})$	Time	$f(V_{L,D})$	μ_{size}	μ_{dot}	$\mathscr{C}_P$	$\mathscr{C}_C$	$\mathscr{C}_E$	$\mathscr{I}^S_{abl}$	$\mathscr{I}^{S'}_{\sigma}$	$\mathscr{L}$
(0.001, 0.999)	35.4s	**-0.16**	370.7 (±110.18)	0.39 (±0.11)	25.02%	25.4%	0.43	0.77 (±0.06)	-0.4 (±0.03)	0.32
(0.01, 0.99)	36.4s	**-0.16**	370.7 (±110.18)	0.39 (±0.11)	25.02%	25.4%	0.43	0.77 (±0.06)	-0.4 (±0.03)	0.32
(0.1, 0.9)	37.23s	**-0.16**	370.7 (±110.18)	0.39 (±0.11)	25.02%	25.4%	0.43	0.77 (±0.06)	-0.4 (±0.03)	0.32
(0.2, 0.8)	37.13s	**-0.16**	370.7 (±110.18)	0.39 (±0.11)	25.02%	25.4%	0.43	0.77 (±0.06)	-0.4 (±0.03)	0.32
(0.3, 0.7)	37.35s	**-0.16**	370.7 (±110.18)	0.39 (±0.11)	25.02%	25.4%	0.43	0.77 (±0.06)	-0.4 (±0.03)	0.32
(0.4, 0.6)	36.24s	**-0.16**	370.7 (±110.18)	0.39 (±0.11)	25.02%	25.4%	0.43	0.77 (±0.06)	-0.4 (±0.03)	0.32
(0.5, 0.5)	37.06s	**-0.16**	370.7 (±110.18)	0.39 (±0.11)	25.02%	25.4%	0.43	0.77 (±0.06)	-0.4 (±0.03)	0.32
(0.6, 0.4)	36.56s	**-0.16**	370.7 (±110.18)	0.39 (±0.11)	25.02%	25.4%	0.43	0.77 (±0.06)	-0.4 (±0.03)	0.32
(0.7, 0.3)	36.53s	**-0.16**	370.7 (±110.18)	0.39 (±0.11)	25.02%	25.4%	0.43	0.77 (±0.06)	-0.4 (±0.03)	0.32
(0.8, 0.2)	36.42s	**-0.16**	370.7 (±110.18)	0.39 (±0.11)	25.02%	25.4%	0.43	0.77 (±0.06)	-0.4 (±0.03)	0.32
(0.9, 0.1)	36.69s	**-0.16**	370.7 (±110.18)	0.39 (±0.11)	25.02%	25.4%	0.43	0.77 (±0.06)	-0.4 (±0.03)	0.32
(0.99, 0.01)	37.22s	**-0.16**	370.7 (±110.18)	0.39 (±0.11)	25.02%	25.4%	0.43	0.77 (±0.06)	-0.4 (±0.03)	0.32
(0.999, 0.001)	36.69s	**-0.16**	370.7 (±110.18)	0.39 (±0.11)	25.02%	25.4%	0.43	0.77 (±0.06)	-0.4 (±0.03)	0.32

Table 8: $\mathscr{H}$ results on ViT (*cifar10*) second-last layer for different values of the composition score regulator terms λ_{size} and λ_{div}.

$(\lambda_{size}, \lambda_{div})$	Time	$f(V_{L,D})$	μ_{size}	μ_{dot}	$\mathscr{C}_P$	$\mathscr{C}_C$	$\mathscr{C}_E$	$\mathscr{I}^S_{abl}$	$\mathscr{I}^{S'}_{\sigma}$	$\mathscr{L}$
(0.001, 0.999)	33.59s	-0.04	50.3 (±31.58)	0.11 (±0.15)	59.26%	62.96%	0.44	0.02 (±0.06)	-0.05 (±0.04)	0.71
(0.01, 0.99)	35.27s	-0.04	50.3 (±31.58)	0.11 (±0.15)	59.26%	62.96%	0.44	0.02 (±0.06)	-0.05 (±0.04)	0.71
(0.1, 0.9)	37.8s	**-0.03**	50.3 (±31.58)	0.11 (±0.15)	60%	62.5%	0.44	0.01 (±0.06)	-0.05 (±0.04)	0.72
(0.2, 0.8)	38.15s	-0.04	50.3 (±31.58)	0.11 (±0.15)	59.26%	62.96%	0.44	0.02 (±0.06)	-0.05 (±0.04)	0.71
(0.3, 0.7)	41.74s	-0.05	45.1 (±22.51)	0.11 (±0.16)	71.15%	73.08%	0.34	0.02 (±0.06)	-0.05 (±0.04)	0.79
(0.4, 0.6)	46.22s	**-0.03**	50.3 (±31.58)	0.11 (±0.15)	60%	62.5%	0.44	0.01 (±0.06)	-0.05 (±0.04)	0.72
(0.5, 0.5)	136.47s	**-0.03**	50.3 (±31.58)	0.11 (±0.15)	60%	62.5%	0.44	0.01 (±0.06)	-0.05 (±0.04)	0.72
(0.6, 0.4)	43.43s	**-0.03**	50.3 (±31.58)	0.11 (±0.15)	60%	62.5%	0.44	0.01 (±0.06)	-0.05 (±0.04)	0.72
(0.7, 0.3)	45.1s	-0.07	45.1 (±22.51)	0.11 (±0.16)	71.74%	75%	0.32	0.04 (±0.06)	-0.05 (±0.04)	0.79
(0.8, 0.2)	52.37s	-0.06	46.3 (±23.57)	0.11 (±0.16)	70.65%	75%	0.34	0.02 (±0.06)	-0.05 (±0.04)	0.78
(0.9, 0.1)	54.84s	-0.08	55.4 (±31.04)	0.14 (±0.19)	63.52%	69.81%	0.41	0.05 (±0.06)	-0.07 (±0.05)	0.75
(0.99, 0.01)	71.08s	-0.09	64 (±28.3)	0.18 (±0.18)	72.77%	78.12%	0.37	0.03 (±0.06)	-0.07 (±0.05)	0.81
(0.999, 0.001)	77.65s	-0.1	64 (±30.52)	0.22 (±0.18)	64.26%	68.95%	0.48	0.02 (±0.05)	-0.07 (±0.04)	0.73

Table 9: $\mathscr{H}$ results on ViT (*cifar100*) second-last layer for different values of the composition score regulator terms λ_{size} and λ_{div}.

$(\lambda_{size}, \lambda_{div})$	Time	$f(V_{L,D})$	μ_{size}	μ_{dot}	$\mathscr{C}_P$	$\mathscr{C}_C$	$\mathscr{C}_E$	$\mathscr{I}^S_{abl}$	$\mathscr{I}^{S'}_{\sigma}$	$\mathscr{L}$
(0.001, 0.999)	42.53s	-0.01	199.4 (±79.88)	0.06 (±0.06)	35.34%	36.84%	0.49	0.02 (±0.1)	-0.15 (±0.03)	0.42
(0.01, 0.99)	45.86s	-0.01	199.4 (±79.88)	0.06 (±0.06)	35.34%	36.84%	0.49	0.02 (±0.1)	-0.15 (±0.03)	0.42
(0.1, 0.9)	46.8s	-0.01	199.4 (±79.88)	0.06 (±0.06)	35.34%	36.84%	0.49	0.02 (±0.1)	-0.15 (±0.03)	0.42
(0.2, 0.8)	46.54s	-0.01	199.4 (±79.88)	0.06 (±0.06)	35.34%	36.84%	0.49	0.02 (±0.1)	-0.15 (±0.03)	0.42
(0.3, 0.7)	48.12s	-0.01	199.4 (±79.88)	0.06 (±0.06)	35.34%	36.84%	0.49	0.02 (±0.1)	-0.15 (±0.03)	0.42
(0.4, 0.6)	49.9s	-0.01	199.4 (±79.88)	0.06 (±0.06)	35.34%	36.84%	0.49	0.02 (±0.1)	-0.15 (±0.03)	0.42
(0.5, 0.5)	45.45s	-0.01	199.4 (±79.88)	0.06 (±0.06)	35.34%	36.84%	0.49	0.02 (±0.1)	-0.15 (±0.03)	0.42
(0.6, 0.4)	49.18s	-0.01	199.4 (±79.88)	0.06 (±0.06)	35.34%	36.84%	0.49	0.02 (±0.1)	-0.15 (±0.03)	0.42
(0.7, 0.3)	54.6s	-0.01	199.4 (±79.88)	0.06 (±0.06)	35.34%	36.84%	0.49	0.02 (±0.1)	-0.15 (±0.03)	0.42
(0.8, 0.2)	54.07s	-0.02	193.8 (±79.5)	0.06 (±0.06)	38.1%	40.09%	0.49	0.04 (±0.11)	-0.15 (±0.03)	0.45
(0.9, 0.1)	58.25s	-0.03	192.4 (±57.41)	0.07 (±0.06)	38.23%	40.17%	0.49	0.06 (±0.1)	-0.16 (±0.02)	0.46
(0.99, 0.01)	69.63s	-0.02	211.4 (±40.15)	0.07 (±0.05)	38.26%	40.51%	0.48	0.04 (±0.09)	-0.15 (±0.01)	0.46
(0.999, 0.001)	77s	$> -\mathbf{10^{-2}}$	179.8 (±27.57)	0.03 (±0.04)	54.65%	56.89%	0.31	-0.01 (±0.07)	-0.15 (±0.02)	0.62

For ***BDAS***, the high-level causal variables required by the method are derived, in an unsupervised manner, from the sample partition returned by ***SCC*** on the same layer: given the kth co-cluster $\langle N_k^{SCC}, S_k^{SCC} \rangle$, we define $Z_k = [x \in S_k^{SCC}]$, and set the target class y_k to the majority class predicted over S_k^{SCC}. Adapted ***BDAS*** returns a learned rotation R of the activation space together with a subset M_k of its rotated dimensions for each variable k; let R_{M_k} denote the sub-matrix of R restricted to the rows in M_k. Neuron i is assigned to feature k if $l_{ik} \geq Q_{90}(L_k)$, where $l_{ik} = \|R_{M_k}[:, i]\|_2$ is its overall contribution to the dimensions in M_k, and sample j is assigned to feature k if $e_{jk} \geq Q_{90}(E_k)$, where $e_{jk} = \|R_{M_k}\, a(x_j)\|_2 / \|a(x_j)\|_2$ is the fraction of its activation

Table 10: $\mathscr{H}$ results on Llama (*SST5*) second-last layer for different values of the composition score regulator terms λ_{size} and λ_{div}.

$(\lambda_{size}, \lambda_{div})$	Time	$f(V_{L,D})$	μ_{size}	μ_{dot}	$\mathscr{C}_P$	$\mathscr{C}_C$	$\mathscr{C}_E$	$\mathscr{I}^S_{abl}$	$\mathscr{I}^{S'}_{\sigma}$	$\mathscr{L}$
(0.001, 0.999)	10.25s	-0.02	21.4 (±40.74)	0.1 (±0.13)	82.61%	86.96%	0.23	0.16 (±0.06)	-0.09 (±0.03)	1.16
(0.01, 0.99)	10.87s	-0.02	21.4 (±40.74)	0.1 (±0.13)	82.61%	86.96%	0.23	0.16 (±0.06)	-0.09 (±0.03)	1.16
(0.1, 0.9)	19.01s	**-0.01**	32.3 (±73.35)	0.1 (±0.13)	68.42%	73.68%	0.38	0.16 (±0.06)	-0.08 (±0.03)	1.0
(0.2, 0.8)	23.91s	**-0.01**	32.3 (±73.35)	0.1 (±0.13)	68.42%	73.68%	0.38	0.16 (±0.06)	-0.09 (±0.03)	1.0
(0.3, 0.7)	23.44s	**-0.01**	32.3 (±73.35)	0.1 (±0.13)	68.42%	73.68%	0.38	0.16 (±0.06)	-0.09 (±0.03)	1.0
(0.4, 0.6)	21.4s	**-0.01**	32.3 (±73.35)	0.1 (±0.13)	68.42%	73.68%	0.38	0.16 (±0.06)	-0.09 (±0.03)	1.0
(0.5, 0.5)	54.29s	**-0.01**	32.3 (±73.35)	0.1 (±0.13)	68.42%	73.68%	0.38	0.16 (±0.06)	-0.08 (±0.03)	1.0
(0.6, 0.4)	18.18s	**-0.01**	32.3 (±73.35)	0.1 (±0.13)	68.42%	73.68%	0.38	0.16 (±0.06)	-0.08 (±0.03)	1.0
(0.7, 0.3)	15.05s	**-0.01**	32.3 (±73.35)	0.1 (±0.13)	68.42%	73.68%	0.38	0.16 (±0.06)	-0.08 (±0.03)	1.0
(0.8, 0.2)	13.77s	**-0.01**	32.3 (±73.35)	0.1 (±0.13)	68.42%	73.68%	0.38	0.16 (±0.06)	-0.08 (±0.03)	1.0
(0.9, 0.1)	11.98s	**-0.01**	32.3 (±73.35)	0.1 (±0.13)	68.42%	73.68%	0.38	0.16 (±0.06)	-0.09 (±0.03)	1.0
(0.99, 0.01)	10.76s	**-0.01**	32.3 (±73.35)	0.1 (±0.13)	68.42%	73.68%	0.38	0.16 (±0.06)	-0.08 (±0.03)	1.0
(0.999, 0.001)	10.66s	**-0.01**	32.3 (±73.35)	0.1 (±0.13)	68.42%	73.68%	0.38	0.16 (±0.06)	-0.09 (±0.03)	1.0

energy falling in M_k. Note that the normalisation is necessary since, unlike neurons, samples are not otherwise on a comparable scale.

Even after these adaptations, the blocks produced by baselines do not generally satisfy Thresholding. To address this issue and ensure consistency with Definition 1 we perform an additional post-processing step. Specifically, for each block returned by these methods, we extract the largest sub-block which entries are all greater than or equal to t. This task can be formulated as an MBP problem and solved through an ILP solver. Since the input blocks are typically of moderate size, this additional step has a negligible impact on the overall execution time, even if MBP is an NP-complete problem.

It may nevertheless happen that a block returned by one of the baseline methods does not contain any entry $\geq t$. In such cases, the corresponding feature is discarded from the final solution, which is consequently treated as a solution with a reduced number of features $K' < K$.

A.11 Intervention-based Evaluation via IIA

In literature, the IIA score (Geiger et al., 2022) is defined as the fraction of interventions that induce a specific change in the network's classification. Its standard formulation presupposes a high-level causal model whose variables are aligned with the network in advance, and it assumes that intervening on one aligned variable suffices to flip the prediction. Neither holds here: no ground-truth causal model is available, and Definition 1 makes features non-exclusive, so several features may be active on the same sample. Two adjustments follow.

First, the reference class must be derived rather than assumed. For a given block, we consider the weights connecting its neurons to each output neuron and compute, for each output neuron, the average incoming weight from the block. The output neuron l_c with the highest average identifies the reference class c. Second, since intervening on a single feature does not necessarily change the predicted class, the fraction of label changes would be close to zero for every method and would not discriminate among them. It may, however, still induce a variation of the corresponding logit from l_c to l'_c, so we define IIA as $l_c - l'_c$.

We perform two types of interventions: (i) ablation, which sets the neurons of the block to zero, and (ii) Gaussian intervention, which adds Gaussian noise to the same neurons. To account for the fact that features are defined in terms of absolute activations, the Gaussian noise is shifted so that its minimum value is zero. Ablation is applied to samples belonging to the block (S), as it aims to evaluate the importance of the feature for those samples; accordingly, we expect IIA > 0. Conversely, Gaussian noise is injected when inferring samples outside the block (S') and originally classified as $c' \neq c$, in order to assess whether activating the feature can influence the network's prediction; in this case, we expect IIA < 0.

A third adjustment concerns comparability. As reported in Section 6, block sizes differ by more than an order of magnitude across methods, and exhibit high variance within each of them. Interventions on larger blocks affect a larger portion of the network and induce stronger effects regardless of

whether such blocks correspond to meaningful features, whereas blocks with few neurons may yield weaker absolute effects even when semantically coherent. We therefore normalize the score of each block by the number of neurons it contains, and report "*IIA per neuron*". Note that all three adjustments are applied identically to every method, including the baselines, and none of them depends on how blocks are produced.

A.12 Adapting Loss Recovered

Loss recovered is a standard reconstruction-fidelity metric for sparse dictionary methods Bricken et al. (2023b); Karvonen et al. (2024), quantifying how much of a network's task performance survives when a layer's activations are replaced by a reconstruction built from the discovered directions, relative to the floor obtained by zero-ablating the layer entirely. Its standard formulation presupposes an encoder that outputs the feature activations from which the reconstruction is built, learned jointly with the dictionary.

Since in the proposed framework no encoder is learned, reconstruction coefficients are estimated post-hoc. For every sample j we reconstruct its activation as a non-negative combination of the directions of the blocks whose S_k contains j:

$$\hat{x}_j \;=\; \sum_{k\,:\,j\in S_k} c_k\,\varphi(F_k), \qquad c_k \geq 0, \tag{15}$$

with c_k fitted by non-negative least squares against the true activation. This plays the same role an SAE's encoder output would, gated by block membership instead of a learned sparsity pattern.

since blocks are non-exhaustive, a sample covered by no block would be reconstructed as the zero vector, coinciding by construction with the zero-ablation baseline; including such samples would measure coverage rather than the fidelity of the directions found. We therefore restrict the evaluation to $D_{\text{cov}} = \bigcup_{k=1}^{K} S_k^D$, the subset of D covered by at least one block, and define Loss recovered as

$$\mathscr{L} \;=\; \frac{\mathrm{CE}(\mathbf{0}) - \mathrm{CE}(\hat{V}_{L,D_{\text{cov}}})}{\mathrm{CE}(\mathbf{0}) - \mathrm{CE}(V_{L,D_{\text{cov}}})}, \tag{16}$$

where $\mathrm{CE}(\cdot)$ denotes the mean cross-entropy loss on D_{cov} when the argument is substituted for the layer's activation matrix, $V_{L,D_{\text{cov}}}$ is the true activation matrix restricted to D_{cov}, $\mathbf{0}$ its zero-ablation, and $\hat{V}_{L,D_{\text{cov}}}$ its reconstruction. The number of evaluated samples cancels between numerator and denominator, both being means over that same subset, so the score stays on a comparable $[0, 1]$-centered scale regardless of how much of D a method covers.

As with IIA, both adjustments are applied identically to every method, including the baselines, and neither depends on how the blocks were produced.

A.13 Qualitative feature visualization via activation maximization

In this appendix, we complement the results in Section 6 with a qualitative post-hoc procedure which synthesizes inputs maximizing the response of N_k over the whole input space. We restrict such analysis to the vision models (*VGG16*, *ViT*), since the feature visualization method is designed for images input. The procedure to synthesize such images is as follows:

- **Objective**: We read the pre-activation $z_i(x)$ (avoiding softmax saturation and dead-ReLU gradients) and standardize each neuron against D : $u_(x) = {}^{(z_i(x)\,-\,b_i)}/_{s_i}$, with $b_i = Q_{50}(\{z_i(x)\ \forall\ x \in D\})$ and $s_i = \max\Big(Q_{99.5}\big(\{z_i(x)\ \forall\ x \in D\}\big) - b_i,\ \eta\Big)$, where η acts as a numerical floor on s_i. Therefore $u_i = 1$ reads as "as active as its 99.5^{th} percentile on real data". We maximize

$$g(x, F_k) = \sum_{i\in N_k} w_i \sigma_c\left(u_i(x)\right) - \beta \sum_{i\notin N_k} \frac{[u_i(x)]_+}{|N_k^c|} \tag{17}$$

  where $\sigma_c(u) = c \cdot \tanh({}^{u}/_{c})$ is a soft cap ($c = 3$) that lets the gradient keep flowing once a neuron is already strongly active, so that optimization is not wasted on the already-satisfied

neurons of N_k at the expense of the others; β ($= 0.25$) weighs the penalty on neurons outside the block, which keeps distinct blocks from converging to a common appearance; and $[\cdot]_+ = \max(\cdot, 0)$ denotes the positive part, so that only off-block neurons active above their own baseline are penalized.

- **Parametrization and optimization**: Following Olah et al. (2017), the canvas is parametrized in the Fourier domain ($1/f^{1.3}$ scaling, decorrelated color) and optimized under random rotation/scale/translation, averaged over several transformed views per step. A scale factor > 1 acts as a random crop and is what produce the repeated, texture-like, appearance of the syntheses. Since *Cifar10* and *Cifar100* are a 32×32 sized image dataset, optimizing the canvas directly at that resolution leaves little room for legible structure to emerge. We sidestep this by optimizing a larger 160×160 canvas and mapping it down to the network's native 32×32 input with an anti-aliased resize at every step, forcing the canvas to express only the structures that survive at the target resolution.
- **Faithfulness**: Since g is evaluated on synthetic input, we additionally compute $u_{\text{real}} = \max_{x \in S_k} g(x; F_k)$ on real exemplars for every block; a synthesis with $u_{\text{synth}} < u_{\text{real}}$ should not be read as evidence of what the block represents. Note that g never sees S_k during synthesis, so this is a check on optimization success, not a term being optimized for.

Figures 8 and 9 report the results of the feature visualization procedure on *VGG16* and *ViT* models results using L&B-search on second-last layer, with $K = 10$, on *Cifar10*. For each feature we roporte the synthesized canvas, the sample of S_k whose activation vector is the most cosine-aligned with $\varphi(F_k)$ (as defined in Definition 1, and the Grad-CAM (Selvaraju et al. (2017)) of the block's score on that sample, localizing which region of a real image the block actually responds to. As discussed below, the synthesized canvas, the real exemplar, and the Grad-CAM map tend to agree with one another. This agreement is an independent, qualitative signal that the blocks our method finds correspond to coherent concepts, rather than being artifacts of the optimization procedure.

As visible in Figure 8, on *Cifar10* there is a visible agreement between synthesized canvas, block image and Grad-CAM result. For instance, feature 1's canvas is a car front/side end, and Grad-CAM lands squarely on the car body. The same pattern appears in features 2, 6, and 10. Feature 4 goes as far as synthesizing a small grid of eyes and snouts, and the Grad-CAM map lands right on the dog's face. A similar result appears in feature 10. A coarser pattern also holds across the grid: wherever the exemplar is a vehicle (features 1, 3, 5, 9) the synthesis leans on sharp, angular structure; wherever it is an animal (2, 4, 6, 7, 8, 10) it turns soft and rounded, echoing fur and body contours. Finally, as discussed in Section 6, results on *VGG16* (*Cifar10*) second-last layer align with class identity, likely because *VGG16*, being a simpler model than *ViT*, exhibits this alignment already at the second-last layer, consistent with prior evidence that convolutional units near the classifier tend to encode class-specific concepts (Bau et al., 2017; Zeiler & Fergus, 2014).

For *ViT*(*Cifar10*), as visible in Figure 9, the correspondence is less immediate: being a more complex model, *ViT* appears to refine features beyond class identity, focusing on finer-grained details rather than whole objects, as also discussed in Section 6. Nevertheless, the signal is there. Feature 9 provides a clear example: the synthesized canvas emphasizes a transition between a light region and a darker background, echoed in the exemplar by the boundary between the lower part of the ship and the surrounding water, where Grad-CAM also concentrates. Feature 4 shows a related pattern: the synthesized canvas captures a predominantly green structure reminiscent of the grassy background, while Grad-CAM concentrates on the boundary between the horse and the surrounding grass rather than on the animal itself.

Figures 10 and 11 report the same analysis on *Cifar100*, where $K = 10$ is far below $|C| = 100$, so each block necessarily spans several classes. On *VGG16*, Grad-CAM keeps landing on the whole object of the exemplar, while the synthesized canvas reflects the mixture of classes covered by the block rather than the class of that single sample, e.g. a canvas of cans for a block whose best-aligned sample is a motorcycle, or of chimpanzees for one whose best-aligned sample is a streetcar. This is consistent with the large and class-heterogeneous blocks reported in Section 6, where *VGG16* (*Cifar100*) has $\mu_{size} = 370.7$ and $\mathscr{C}_P = 25.02\%$. On *ViT*, the boundary-level reading observed on *Cifar10* carries over, although the correspondence is less evident than in *VGG16*. Feature 8 provides one example, where Grad-CAM highlights localized triangular structures on the motorcycle that are also reflected in the synthesized canvas, while Feature 7 shows a similar pattern around the building's edges, loosely echoed by the line-like structures in the synthesis. This weaker correspondence is

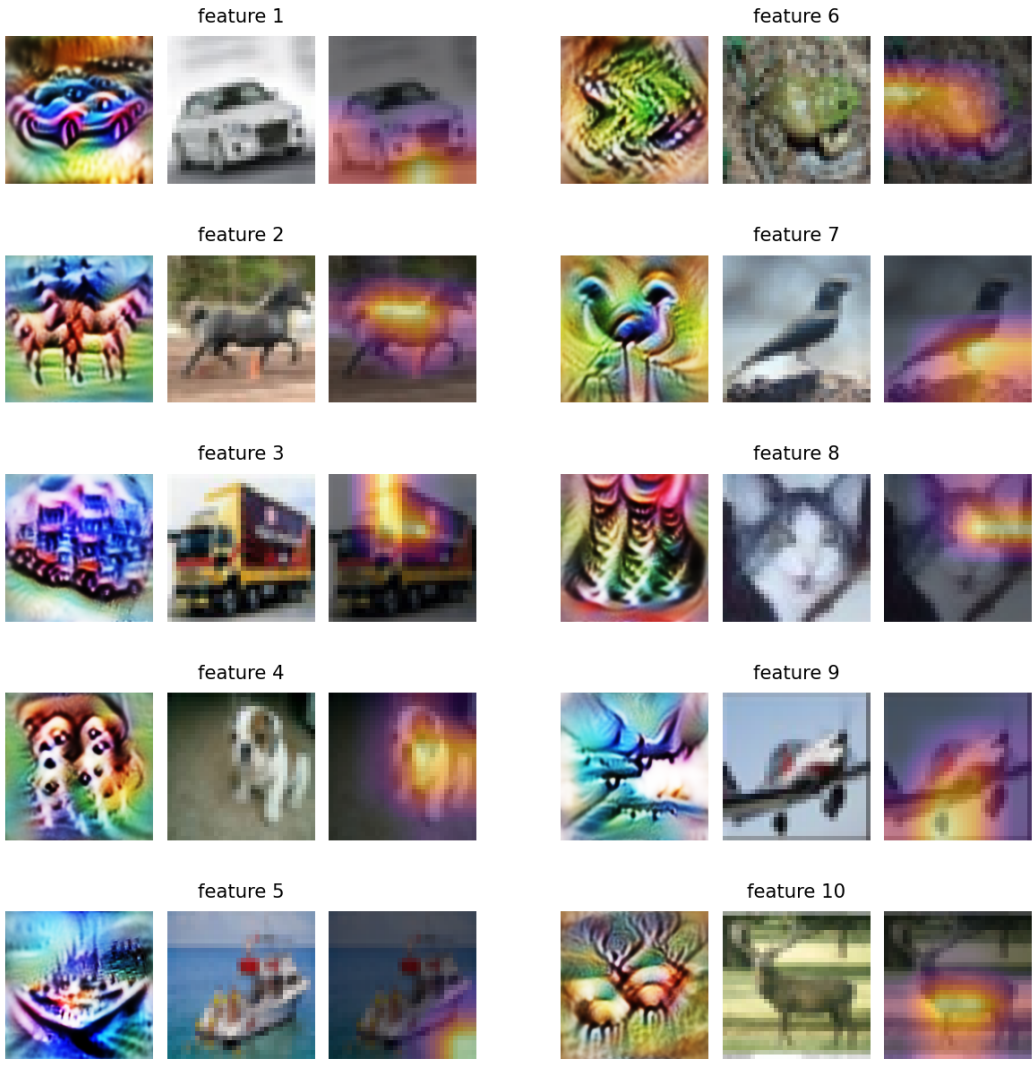


Figure 8: *VGG16* feature visualization result on second-last layer, *Cifar10*, L&B-search, $K = 10$. For each feature is reported the synthesized canvas, best-aligned real sample of S_k, and Grad-CAM on that sample.

expected here, since with K far below $|C|$ each block spans several classes, so the synthesis has no single concept to converge to.

Finally, in Figures 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, and 27 are reported feature visualization procedure of *PCA*, *SAE*, *SCC*, and *BDAS* on both *VGG16* and *ViT* models on second-last layer, on *Cifar10* and *Cifar100*, with $K = 10$. As visible, results on *VGG16* remain similar across methods, further supporting the alignment between features and classes in this model, even though baselines fail to capture all classes and often produce near-duplicate features, a behaviour that becomes more evident on *Cifar100*, where K is far below $|C|$. On *ViT*, instead, the features found tend to differ across methods, showing that baselines, despite their limitations (e.g., recovering fewer features than K, among others discussed throughout this work), are still capable of surfacing insight into the model's learned features. The boundary-level reading of *ViT* discussed above is however recovered by every method, which suggests it reflects the model rather than the procedure used to identify the blocks.

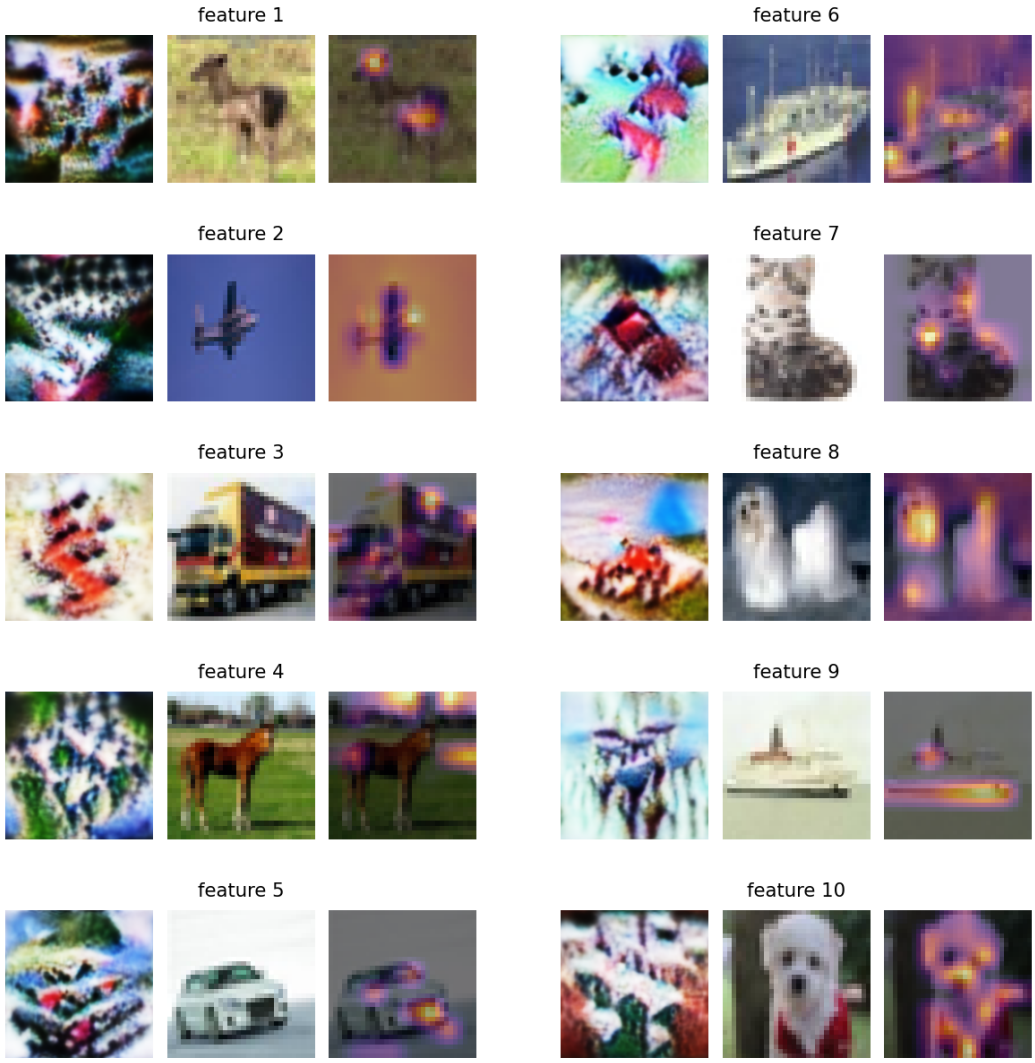


Figure 9: *ViT* feature visualization result on second-last layer, *Cifar10*, L&B-search, $K = 10$. For each feature is reported the synthesized canvas, best-aligned real sample of S_k, and Grad-CAM on that sample.

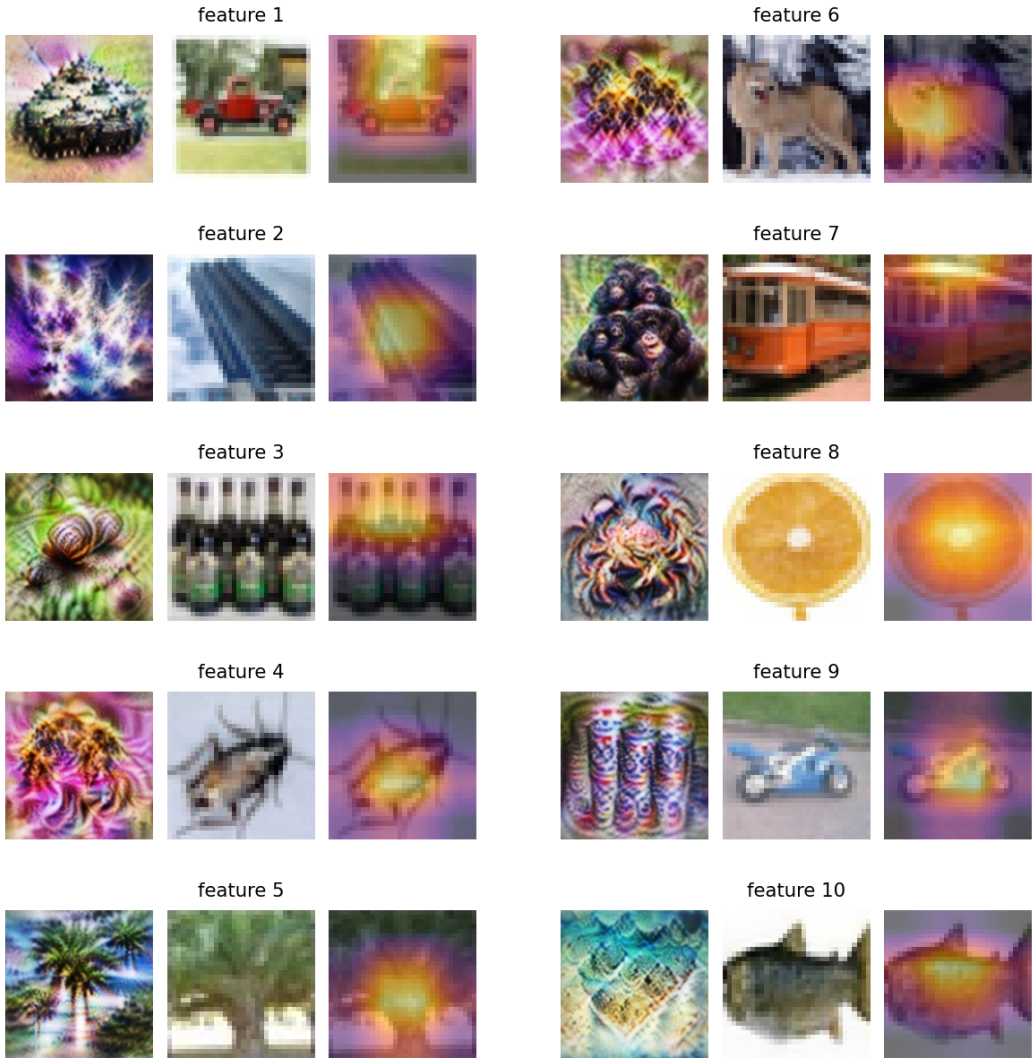


Figure 10: *VGG16* feature visualization result on second-last layer, *Cifar100*, L&B-search, $K = 10$. For each feature is reported the synthesized canvas, best-aligned real sample of S_k, and Grad-CAM on that sample.

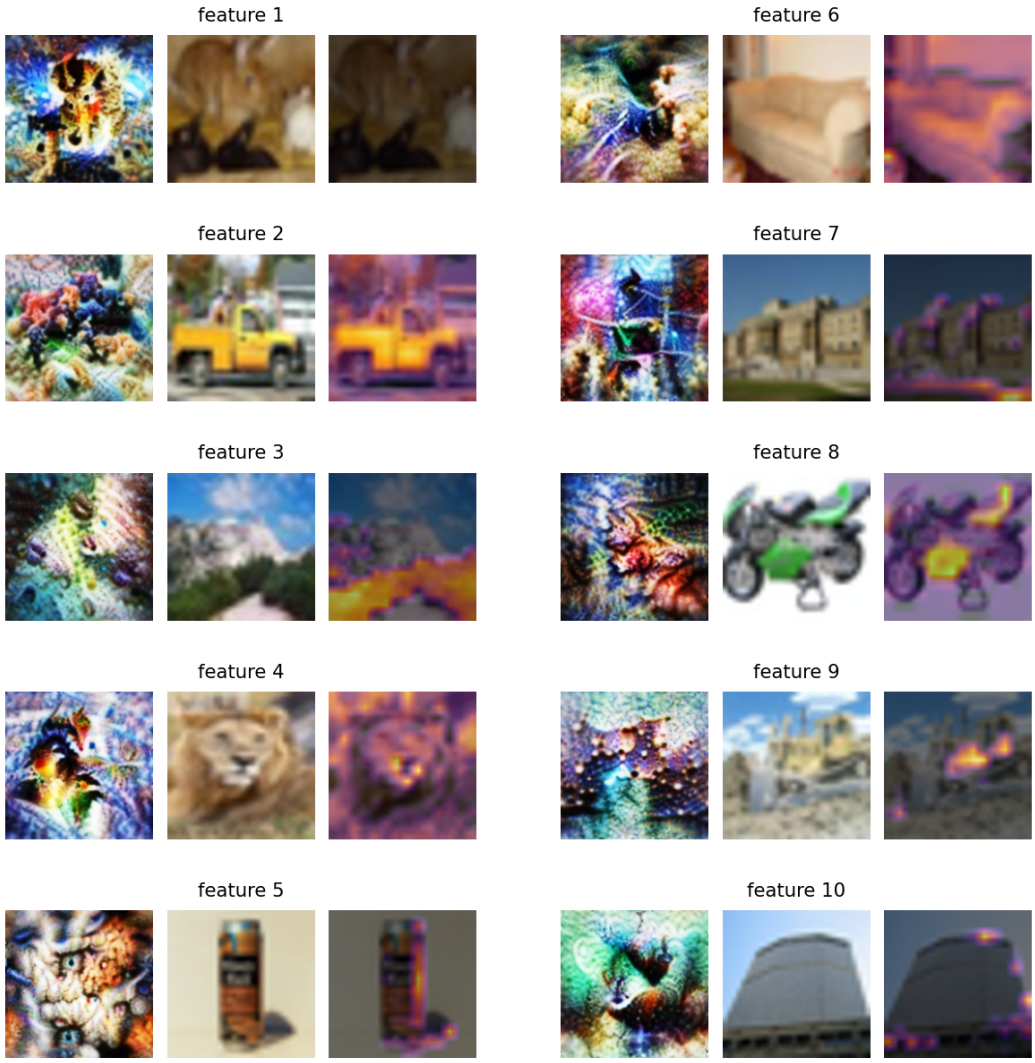


Figure 11: *ViT* feature visualization result on second-last layer, *Cifar100*, L&B-search, $K = 10$. For each feature is reported the synthesized canvas, best-aligned real sample of S_k, and Grad-CAM on that sample.

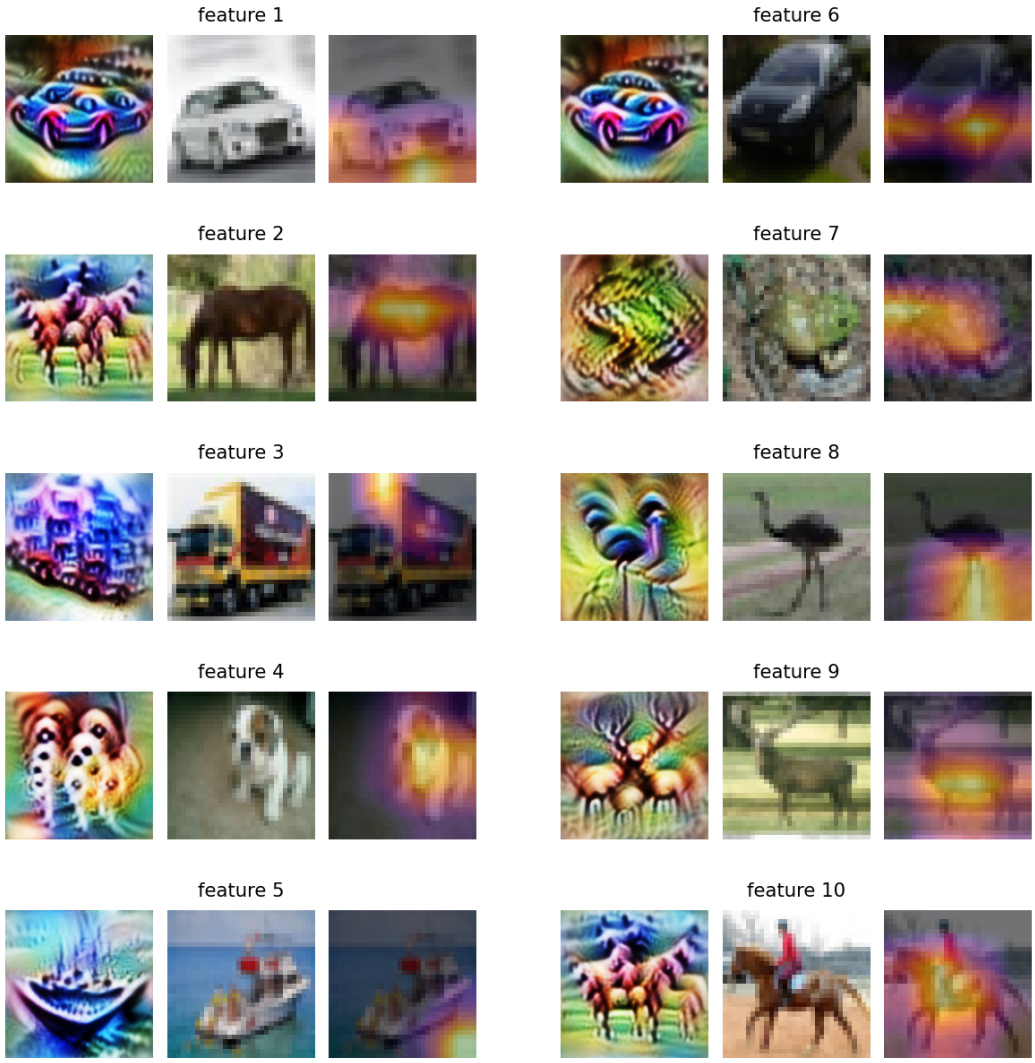


Figure 12: *VGG16* feature visualization result on second-last layer, *Cifar10*, *PCA*, $K = 10$. For each feature is reported the synthesized canvas, best-aligned real sample of S_k, and Grad-CAM on that sample.

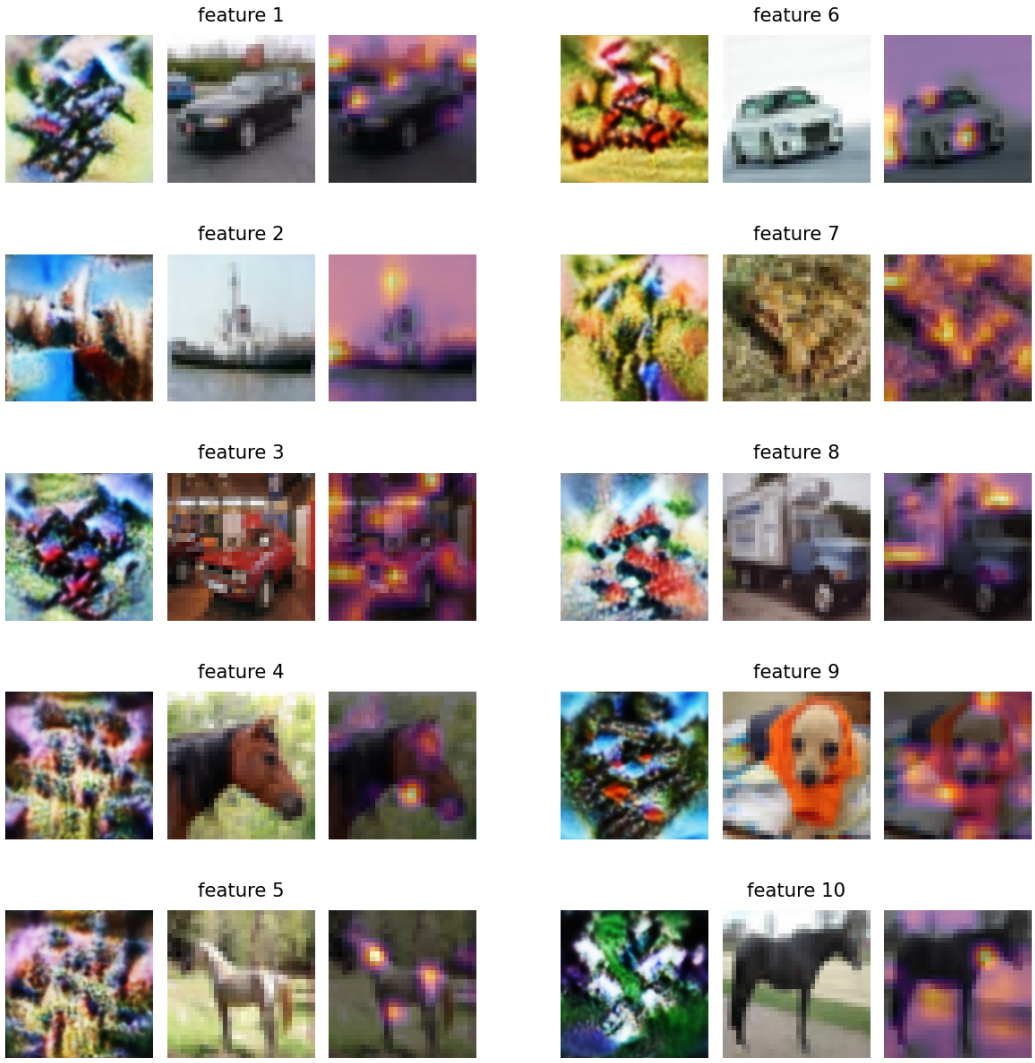


Figure 13: *ViT* feature visualization result on second-last layer, *Cifar10*, *PCA*, $K = 10$. For each feature is reported the synthesized canvas, best-aligned real sample of S_k, and Grad-CAM on that sample.

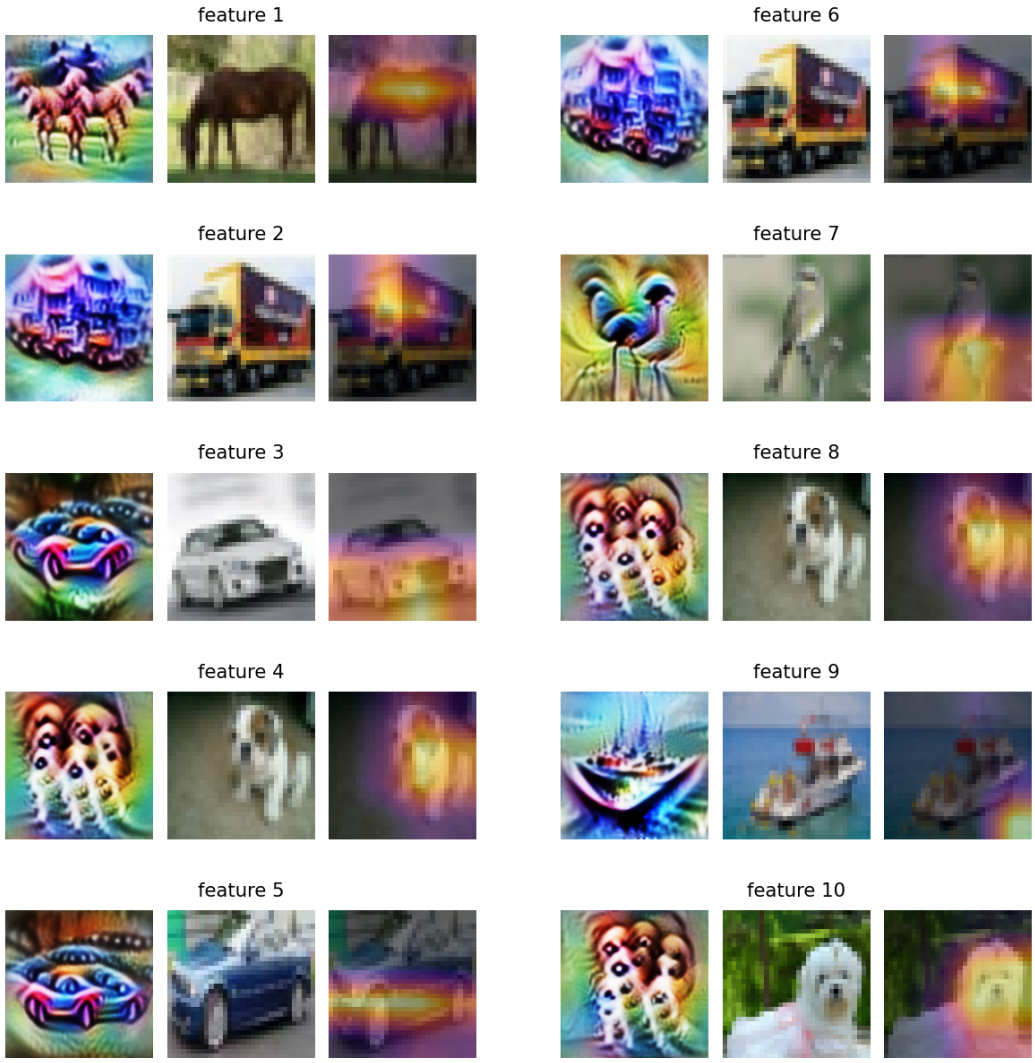


Figure 14: *VGG16* feature visualization result on second-last layer, *Cifar10*, *SAE*, $K = 10$. For each feature is reported the synthesized canvas, best-aligned real sample of S_k, and Grad-CAM on that sample.

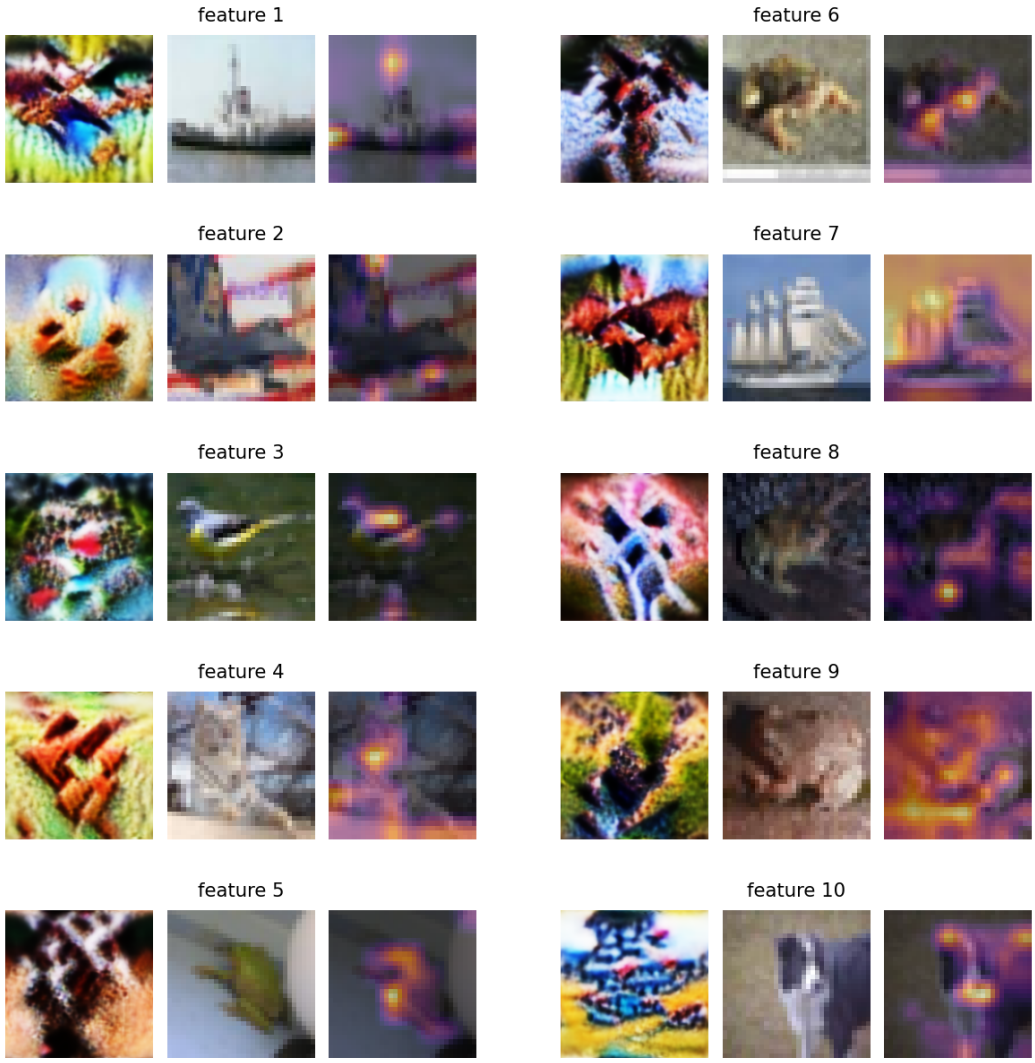


Figure 15: *ViT* feature visualization result on second-last layer, *Cifar10*, *SAE*, $K = 10$. For each feature is reported the synthesized canvas, best-aligned real sample of S_k, and Grad-CAM on that sample.

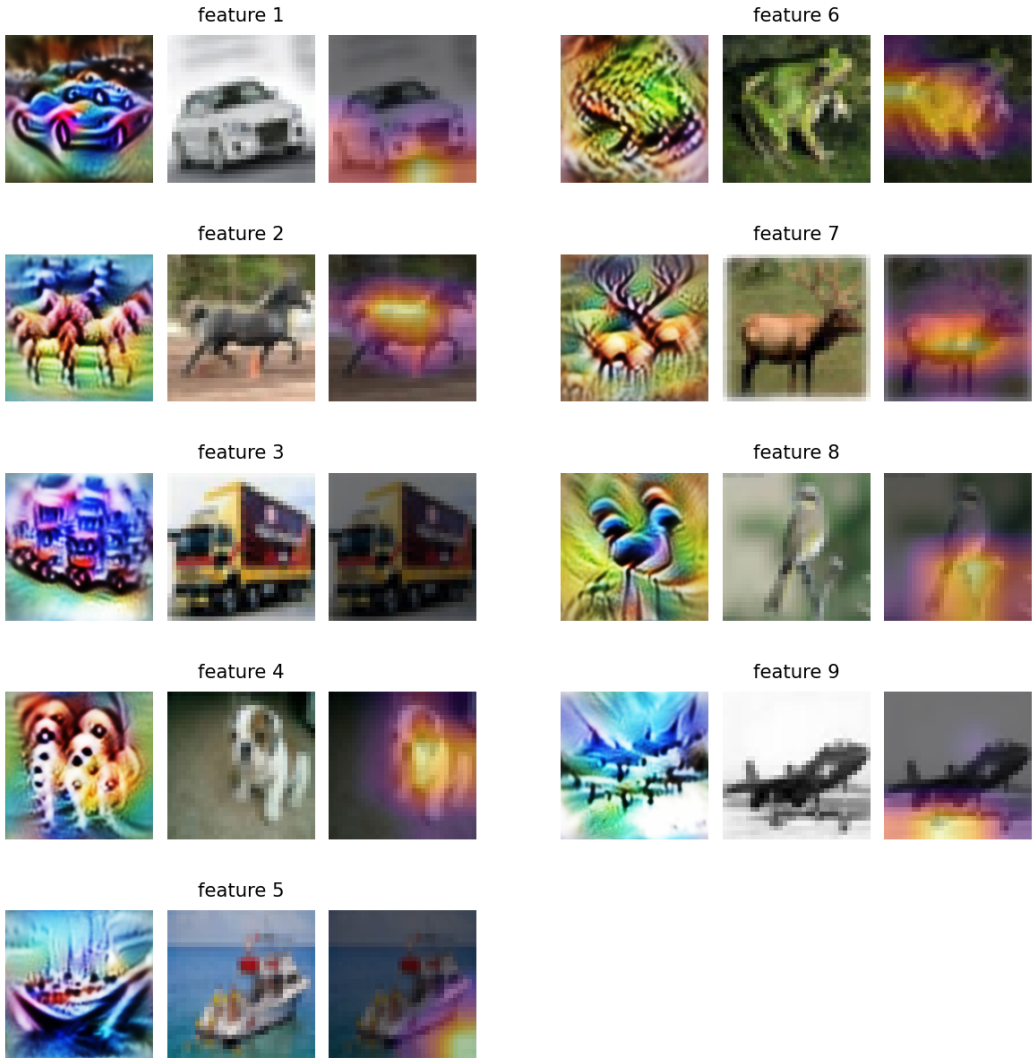


Figure 16: *VGG16* feature visualization result on second-last layer, *Cifar10*, *SCC*, $K = 10$. For each feature is reported the synthesized canvas, best-aligned real sample of S_k, and Grad-CAM on that sample.

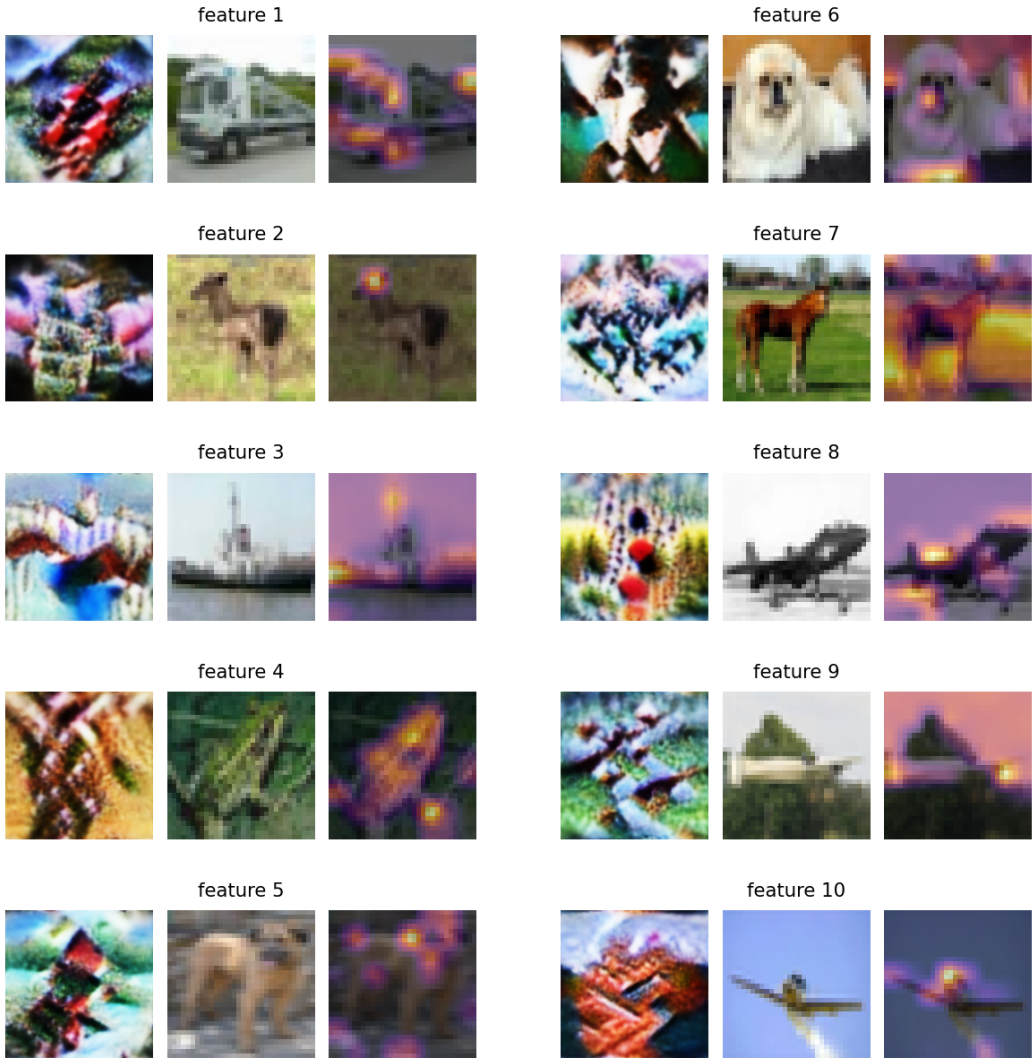


Figure 17: *ViT* feature visualization result on second-last layer, *Cifar10*, *SCC*, $K = 10$. For each feature is reported the synthesized canvas, best-aligned real sample of S_k, and Grad-CAM on that sample.

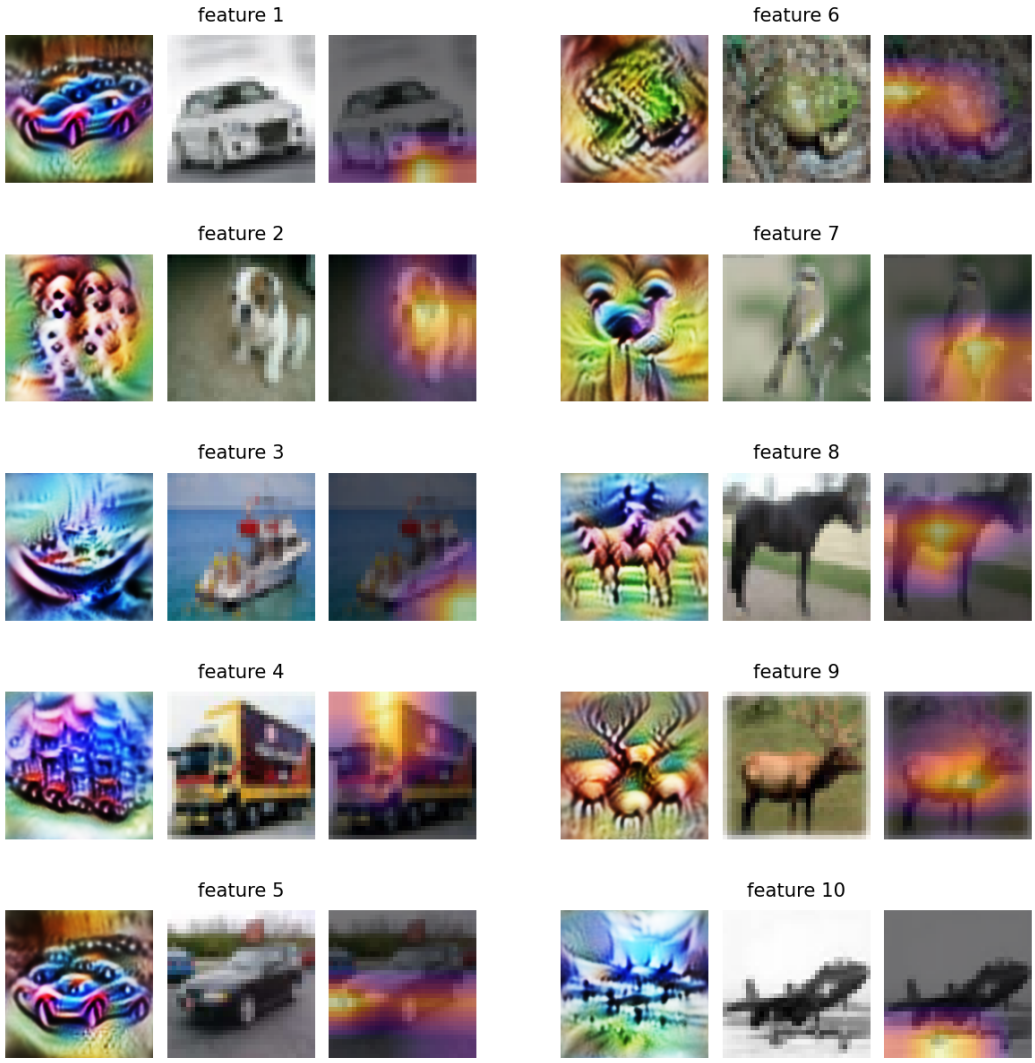


Figure 18: *VGG16* feature visualization result on second-last layer, *Cifar10*, *BDAS*, $K = 10$. For each feature is reported the synthesized canvas, best-aligned real sample of S_k, and Grad-CAM on that sample.

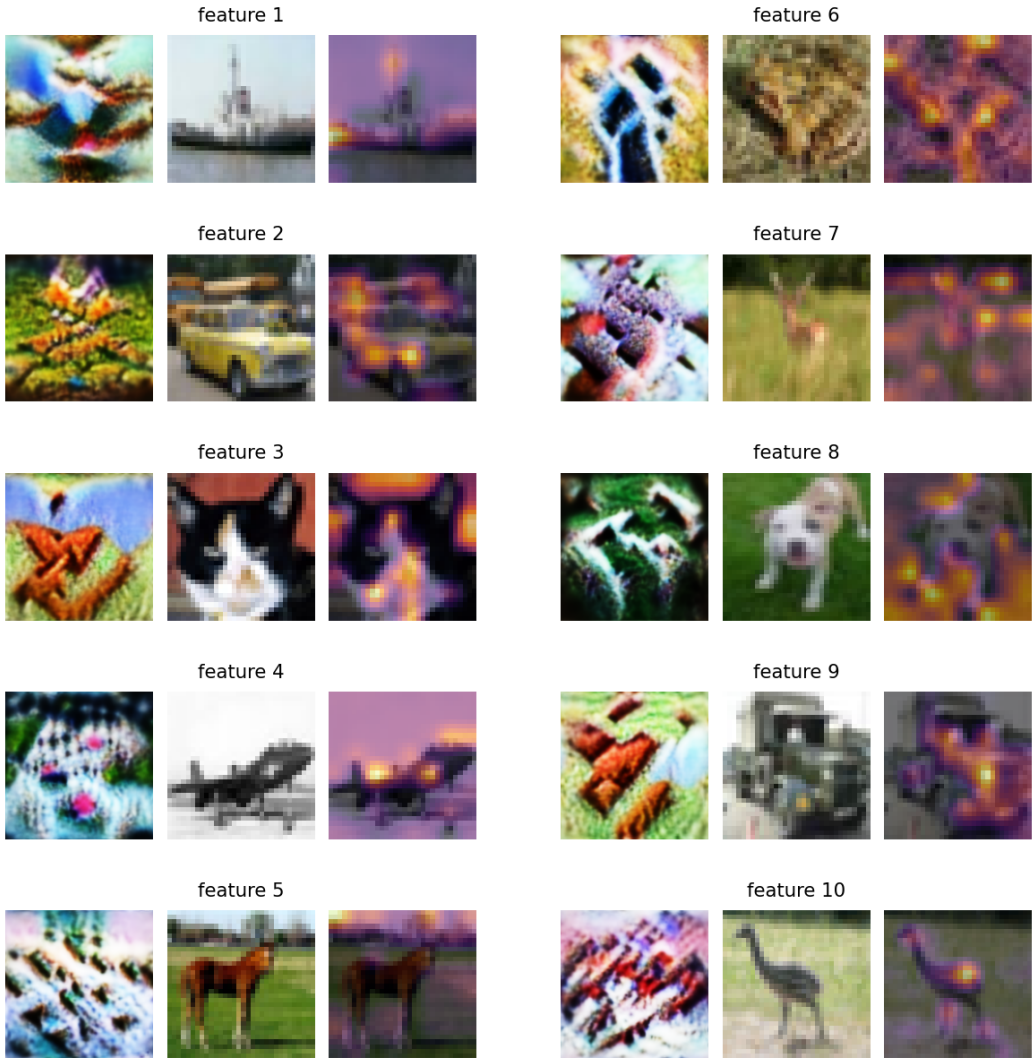


Figure 19: *ViT* feature visualization result on second-last layer, *Cifar10*, *BDAS*, $K = 10$. For each feature is reported the synthesized canvas, best-aligned real sample of S_k, and Grad-CAM on that sample.

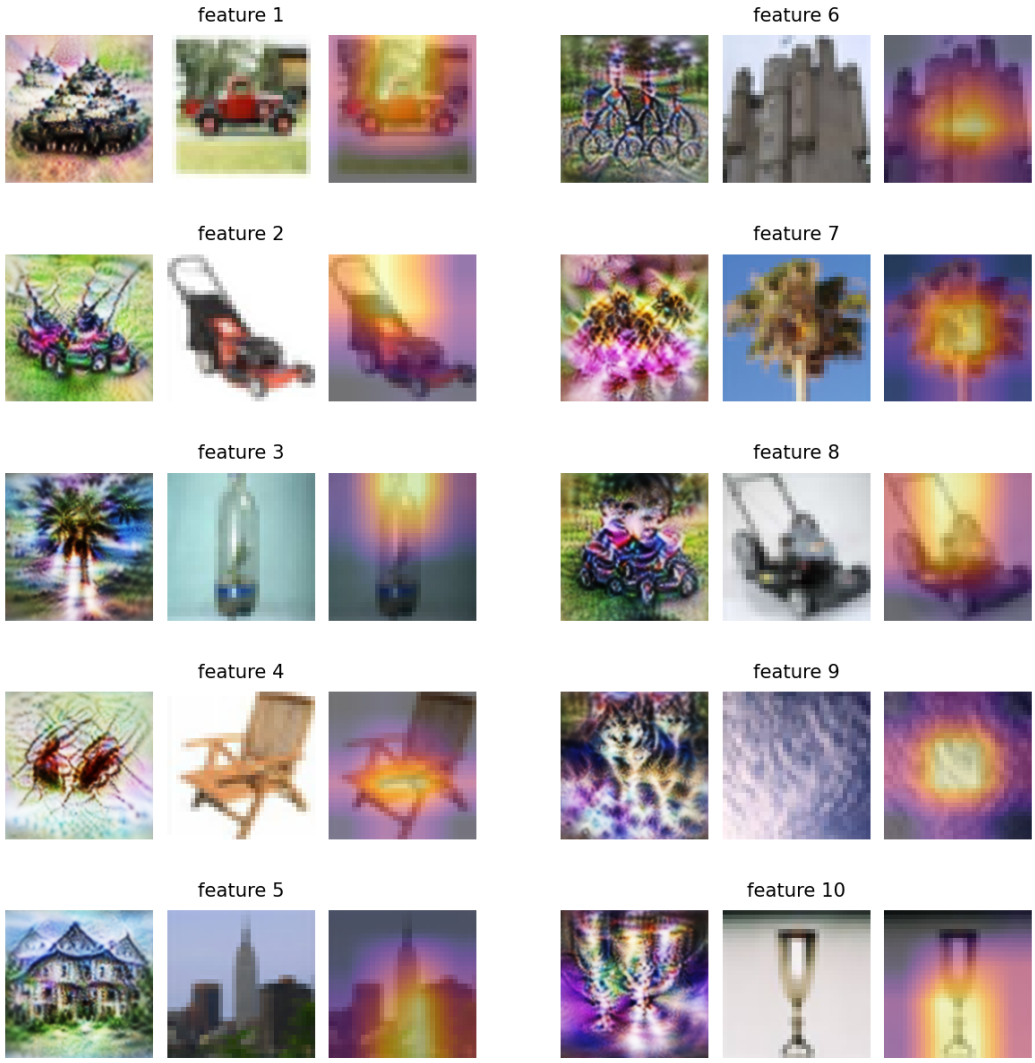


Figure 20: *VGG16* feature visualization result on second-last layer, *Cifar100*, *PCA*, $K = 10$. For each feature is reported the synthesized canvas, best-aligned real sample of S_k, and Grad-CAM on that sample.

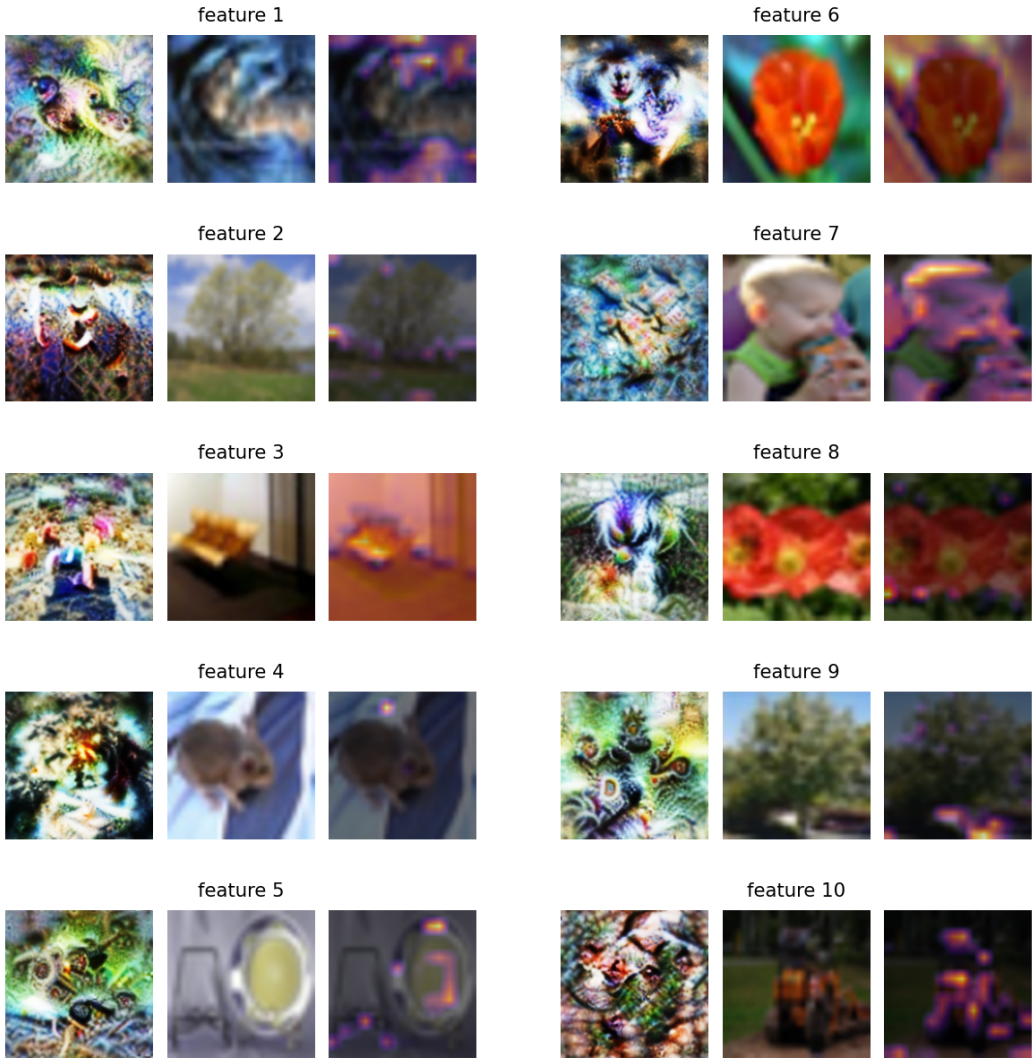


Figure 21: *VGG16* feature visualization result on second-last layer, *Cifar100*, *PCA*, $K = 10$. For each feature is reported the synthesized canvas, best-aligned real sample of S_k, and Grad-CAM on that sample.

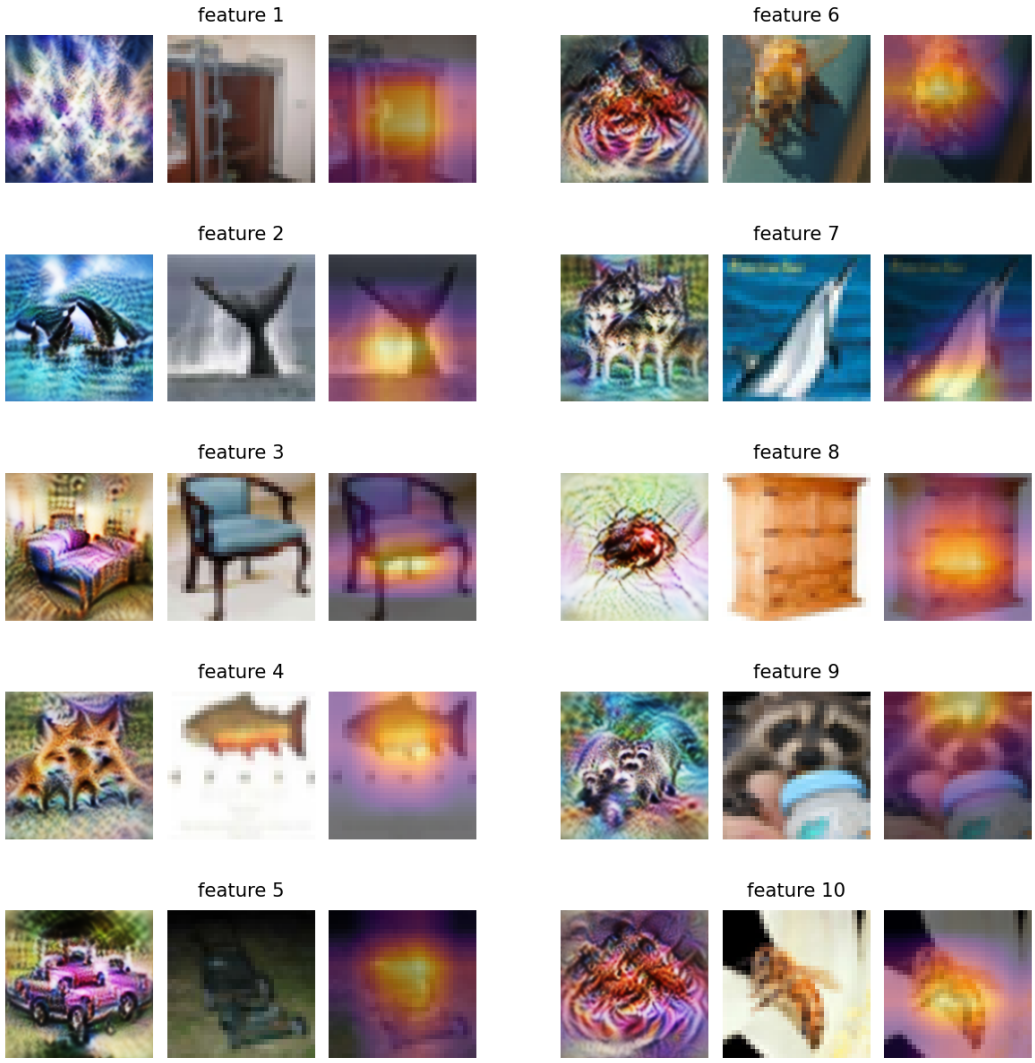


Figure 22: *VGG16* feature visualization result on second-last layer, *Cifar100*, *SAE*, $K = 10$. For each feature is reported the synthesized canvas, best-aligned real sample of S_k, and Grad-CAM on that sample.

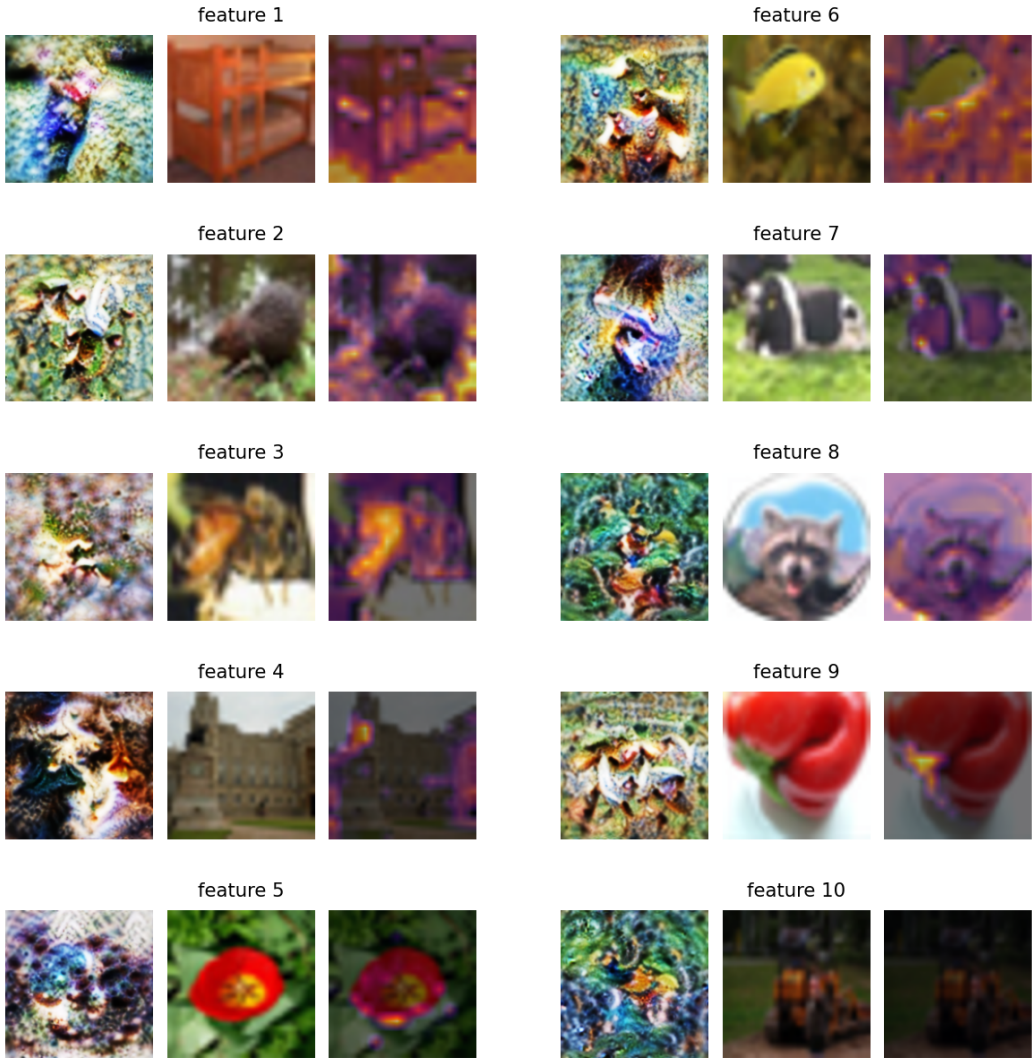


Figure 23: *VGG16* feature visualization result on second-last layer, *Cifar100*, *SAE*, $K = 10$. For each feature is reported the synthesized canvas, best-aligned real sample of S_k, and Grad-CAM on that sample.

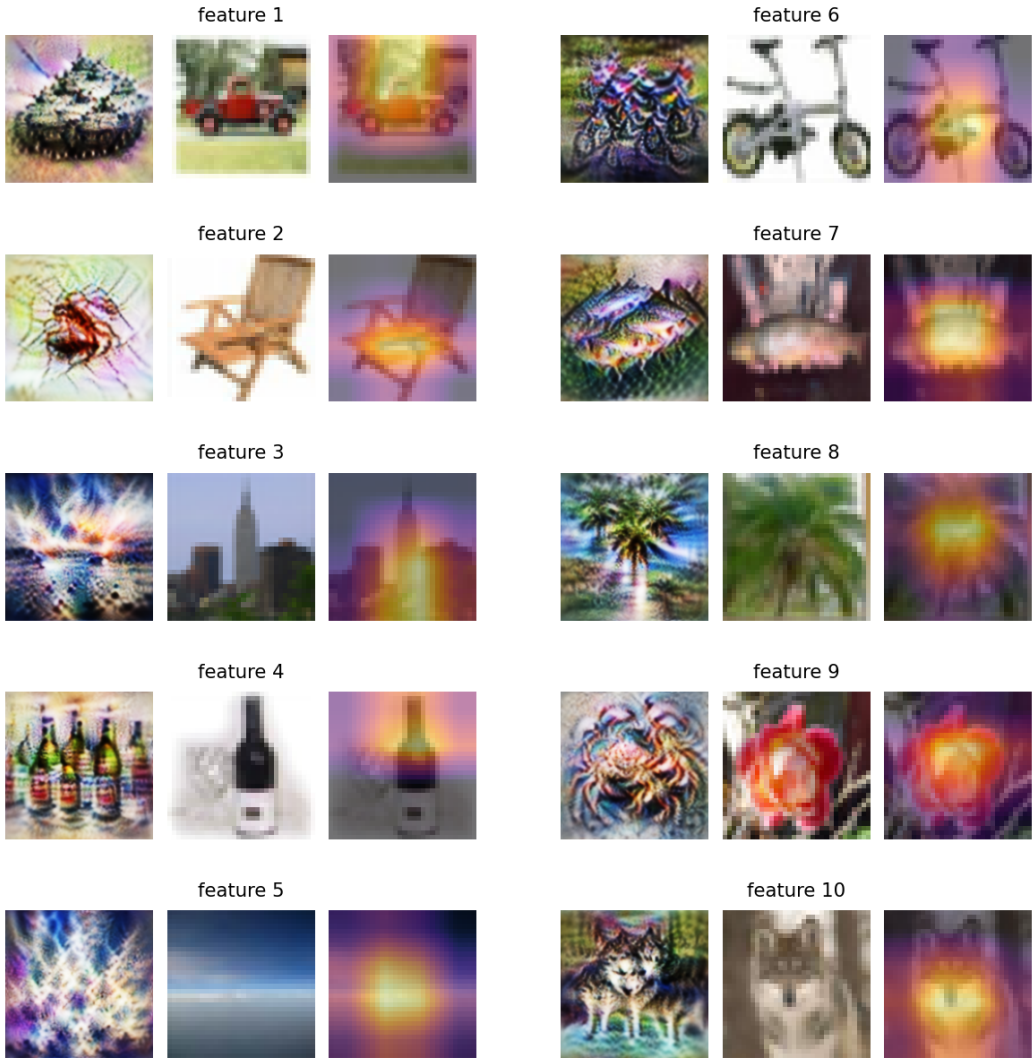


Figure 24: *VGG16* feature visualization result on second-last layer, *Cifar100*, *SCC*, $K = 10$. For each feature is reported the synthesized canvas, best-aligned real sample of S_k, and Grad-CAM on that sample.

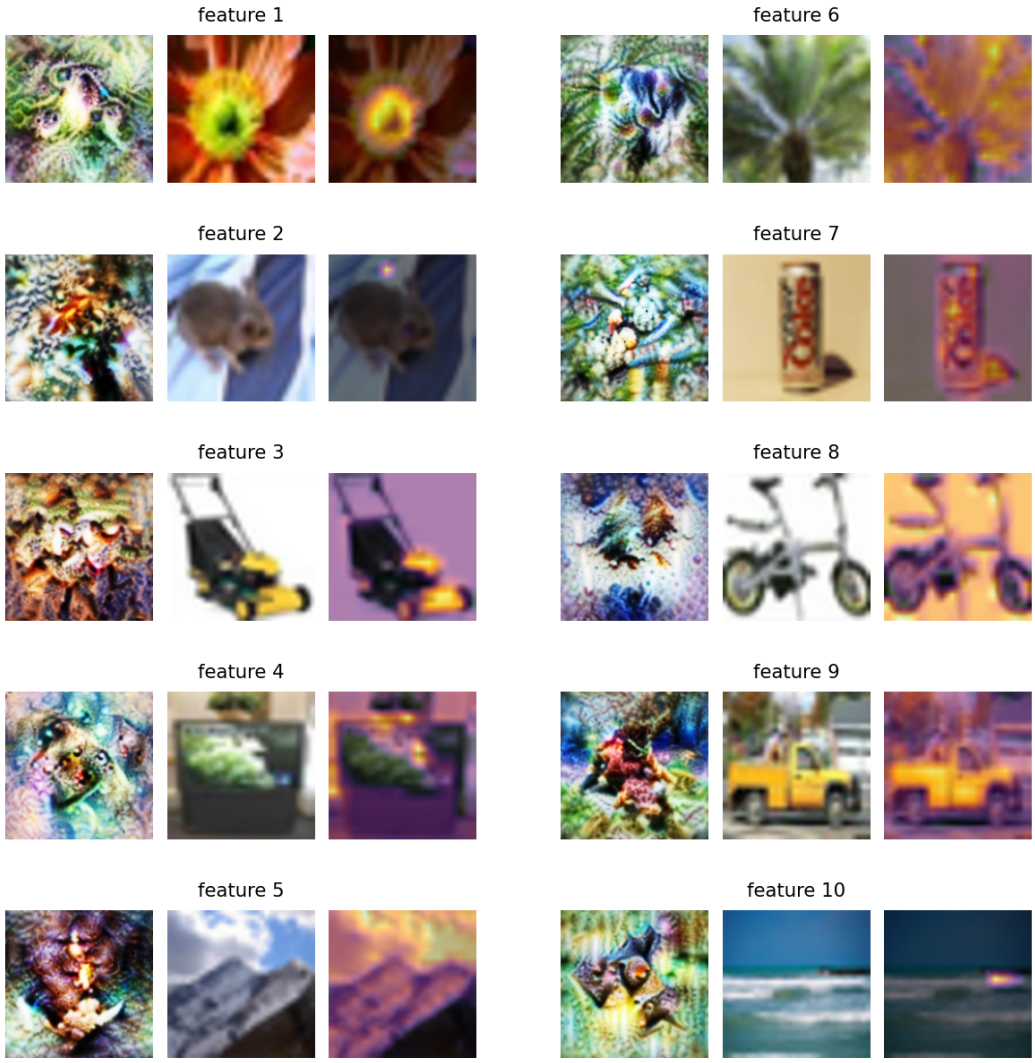


Figure 25: *VGG16* feature visualization result on second-last layer, *Cifar100*, *SCC*, $K = 10$. For each feature is reported the synthesized canvas, best-aligned real sample of S_k, and Grad-CAM on that sample.

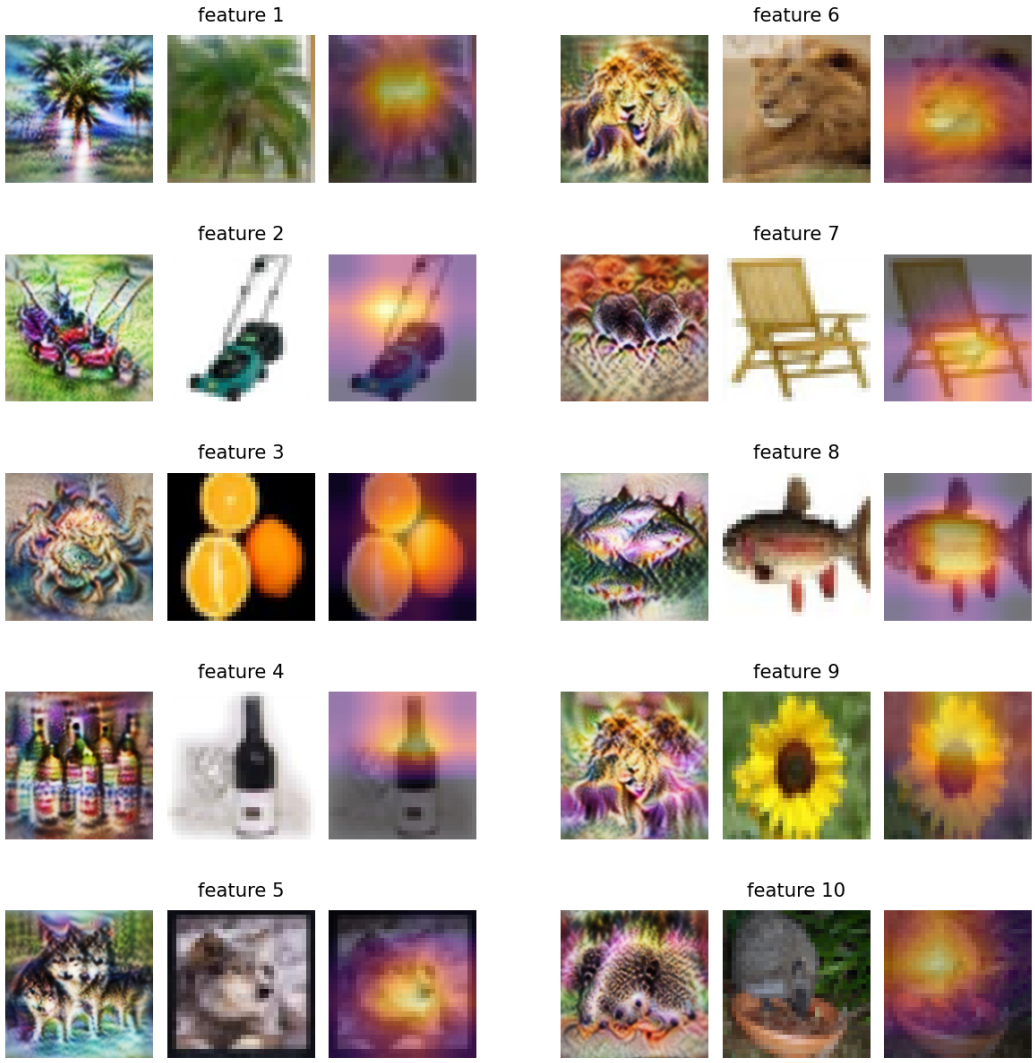


Figure 26: *VGG16* feature visualization result on second-last layer, *Cifar100*, *BDAS*, $K = 10$. For each feature is reported the synthesized canvas, best-aligned real sample of S_k, and Grad-CAM on that sample.

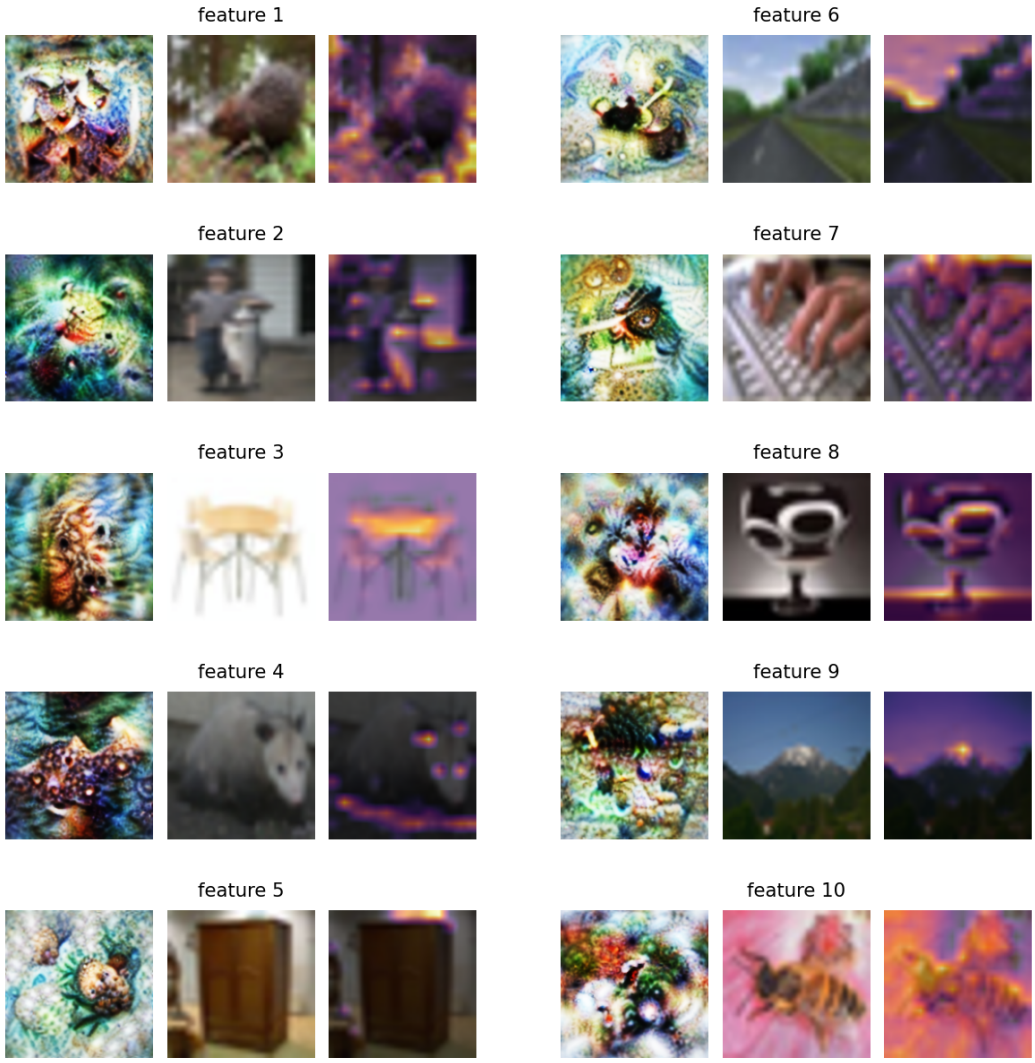


Figure 27: *VGG16* feature visualization result on second-last layer, *Cifar100*, *BDAS*, $K = 10$. For each feature is reported the synthesized canvas, best-aligned real sample of S_k, and Grad-CAM on that sample.