INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Explaining Image Classification by Monitoring Layers activity

Clara Nunes Barrancos

Master in Computer Engineering

Supervisor:
PhD *Luís Miguel Martim Nunes*, *Associate* Professor,
Iscte – Instituto Universitário de Lisboa

Co-Supervisor:
PhD *Alex Marchioni*, *Assistant* Professor,
University of Bologna

September, 2025

TECHNOLOGY
AND ARCHITECTURE

Department of Information Science and Technology

Explaining Image Classification by Monitoring Layers activity

Clara Nunes Barrancos

Master in Computer Engineering

Supervisor:
PhD *Luís Miguel Martim Nunes*, *Associate* Professor,
Iscte – Instituto Universitário de Lisboa

Co-Supervisor:
PhD *Alex Marchioni*, *Assistant* Professor,
University of Bologna

September, 2025

To the reader. If you exist, wow. Respect.

Acknowledgment

I would like to express my sincere gratitude to Lorenzo Capelli and Leandro de Souza Rosa for their invaluable guidance and support throughout this research. Their insights and mentorship played a crucial role in shaping the direction and depth of this work.

I also extend my thanks to the Advanced Research Center on Electronic Systems "Ercole De Castro" (ARCES) at the University of Bologna for providing the academic environment and resources that made this thesis possible, and to Fundação para a Ciência e a Tecnologia, I.P. (FCT), whose partial support through the ISTAR projects UIDB/04466/2020 and UIDP/04466/2020 was instrumental in enabling this work.

A special thank you to my parents and my partner Matteo, whose constant support has been essential throughout my academic journey.

Resumo

A Inteligência Artificial Explicável (XAI) procura tornar os sistemas de IA mais transparentes, permitindo que os utilizadores compreendam e confiem nas suas decisões. A capacidade de identificar previsões não confiáveis é crucial para aplicações em domínios críticos como saúde, finanças e sistemas autónomos.

Uma metodologia recente, **SVD-based Peephole**, propõe uma forma de detetar decisões ambíguas ao analisar as informações que passam por uma camada de rede neural. Esta tese visa melhorar a explicabilidade em IA, extraindo peepholes baseados em SVD das camadas de uma rede neural leve. O método combina redução de dimensionalidade (SVD) e agrupamento não supervisionado para revelar padrões de ativação estruturados na rede.

Investigamos como as representações internas evoluem nas camadas do MobileNetV2, um modelo otimizado para eficiência, e como se relacionam com o processo de decisão. Este trabalho estende técnicas de peephole, antes aplicadas a modelos maiores, adaptando-as à arquitetura compacta e complexa do MobileNetV2.

Os resultados mostram que, mesmo em arquiteturas comprimidas, é possível alcançar explicabilidade significativa. Ao rastrear conceitos relacionados a classes e visualizá-los com conceptogramas, demonstramos como representações abstratas se desenvolvem ao longo das camadas. Estas visualizações não apenas melhoram a compreensão do modelo, mas também fornecem uma forma prática de avaliar a confiabilidade das suas previsões. Assim, peepholes baseados em SVD tornam-se ferramentas explicativas e de diagnóstico, contribuindo para sistemas de IA mais transparentes, confiáveis e responsáveis.

Palavras Chave: *Inteligência Artificial explicável*, *Redes Neuronais*, *Interpretabilidade*, *Confiabilidade*

Abstract

Explainable Artificial Intelligence (XAI) seeks to make AI systems more transparent, enabling users to understand and trust their decisions. The ability to identify untrustworthy predictions is particularly important for deploying AI in critical domains such as healthcare, finance, and autonomous systems. A recently proposed methodology, **SVD-based Peephole**, offers a novel way to detect ambiguous decisions by analyzing information passing through a neural network layer.

This thesis aims to improve explainability in AI by extracting SVD-based peepholes from the layers of a very lightweight neural network. The method combines dimensionality reduction method (SVD) and unsupervised clustering to reveal structured activation patterns within the network. We investigate how internal representations evolve across layers of MobileNetV2 (a model optimized for efficiency) and how they relate to the decision making process. This work extends previous peephole techniques, traditionally applied to larger models, by adapting them to the more compact and complex architecture of MobileNetV2.

Our findings show that even in compressed architectures like MobileNetV2, meaningful explainability can be achieved. By tracing class-related concepts through the network and visualizing them with *conceptograms*, we demonstrate how abstract representations develop across layers. Importantly, these visualizations not only improve explainability but also provide a practical way to assess the trustworthiness of model outputs. This positions SVD-based peepholes as both an explanatory tool and a diagnostic method, contributing to the broader goal of making AI systems more transparent, reliable, and accountable.

KEYWORDS: *Explainable Artificial Intelligence*, *Neural Networks*, *Interpretability*, *Trustworthiness*

Contents

Resumo iii

Abstract v

List of Figures ix

List of Tables xi

Glossary xiii

Introduction 1
Research Questions 2

Chapter 1. Background 3
1.1. Explainability 3
1.2. SVD-based Peephole 4

Chapter 2. Literature Review 11
2.1. Systematic Review 11
2.2. Related Work 11

Chapter 3. Setup 17
3.1. Mobilenet 17
3.2. Training 20
3.3. Layer selection 22

Chapter 4. Experimental Results 31
4.1. Conceptogram 31
4.2. Superconceptogram 43

Conclusion and Future Work 53
4.3. Conclusion 53
4.4. Future Works 54

Bibliography 55

Appendix A. Mobilenet Layer Organization 59

List of Figures

1.1 Toeplitz method: $\overline{W}$ is a reshaped multiple embedding of the weight tensor into a tensor space [10]. 6
1.2 Peephole extraction and clustering scheme 9

3.1 Standart convolution and depthwise seperable convolution [30] 18
3.2 Residual block and inverted residual block [32] 18
3.3 Skip connection in Mobilenet v2 (when *Stride*=1) 19
3.4 Standard vs. grouped convolutions: (a) In a standard convolution, each filter is convolved with all of the input's channels; (b) In a grouped convolution with two groups, half of the filters are applied to each half of the input for a 2× reduction in parameters used. [33] 19
3.5 Architecture of Mobilenet v2 [28] where t: expansion factor in bottleneck blocks; c: Number of output channels for the block; n: Number of times this block is repeated and s: Stride of the first block in this stage . . . 20
3.6 Confusion matrix of MobileNet fine tuned on Cifar-100: The rows represent the true superclasses (truth labels) and the columns represent the predicted superclasses by the model. Each cell shows the number of test samples truly from class i but predicted as class j. 22
3.7 Singular vectors of layer `features.5.conv.0.0` 23
3.8 Singular vectors of layer `features.9.conv.1.0` 23
3.9 Singular vectors of layer `features.5.conv.0.0` (1000 rank) 25
3.10 Singular vectors of layer `features.9.conv.1.0 (1000 rank)` 25
3.11 Empirical posterior heatmap for layer `features.5.conv.0.0` 27
3.12 Empirical posterior heatmap for layer `features.9.conv.1.0` 27
3.13 Class and cluster coverage of each layer in Mobilenet v2 (Using GMM with $numberofclusters = 100$) 29

4.1 Example of conceptogram of class *woman* and the corresponding final outputs of the network 32
4.2 Conceptogram of the selected layers 34
4.3 Conceptogram of all layers of MobileNet 35
4.4 Conceptogram of a correctly predicted label of superclass *trees* 36
4.5 Conceptogram of a incorrectly predicted label of superclass *trees* 37
4.6 Conceptogram of a sample predicted with low confidence 39
4.7 Conceptogram of a correctly predicted label of superclass *flowers* 40

4.8 Conceptogram of a incorrectly predicted label of superclass *flowers* . . . 41
4.9 Conceptogram of a low-confidence prediction of superclass *flowers* . . . 42
4.10 Superconceptogram of superclass *people* using 20 clusters 43
4.11 Superconceptogram of *palm_tree* 45
4.12 Superconceptogram of *oak_tree* 46
4.13 Superconceptogram of *willow_tree* 47
4.14 Superconceptogram of *sunflower* 48
4.15 Superconceptogram of *rose* 49
4.16 Superconceptogram of sunflower (low confidence case) 50

List of Tables

3.1 Selected layers for peephole extraction (organized by Bottleneck) 30

A.1**Inverted Residual Block Configuration**: as in Figure 3.5, Mobilenet has 7 bottlenecks (some of them are repteated several times), a final convolutional layer and a fully conected layer. 59

A.2Inverted residual block: composed by an expansion layer, a depth wise layer and a projection layer. For example: `features.13.conv.1.0` is a depth wise layer in bottleneck 5. 59

Glossary

AI: Artificial Intelligence. 1, 2, 53
AIA: Union's Artificial Intelligence Act. 1, 53

C-XAI: Concept-based explanation methods. 4

GMM: Gaussian Mixture Model. 8, 26

SVD: Singular Value Decomposition. 4–6, 9, 15

XAI: Explainable Artificial Intelligence. 1, 2

Introduction

Artificial Intelligence (AI) has become an integral part of various fields, ranging from healthcare and finance to autonomous systems and cybersecurity. As AI systems are deployed in high-stakes and socially impactful domains, the need for transparency and accountability has become more pressing than ever. The increasing reliance on AI models, particularly deep neural networks, has raised concerns regarding their interpretability and trustworthiness. Despite their impressive performance, many AI models are often deployed as **black-box systems**, offering little insight into how decisions are made. This lack of transparency raises serious concerns around interpretability and trust. [1].

Explainable Artificial Intelligence (XAI) has emerged as a crucial research area aimed at making AI models more transparent and interpretable. The primary goal of XAI is to provide human-understandable explanations for AI-driven decisions, allowing stakeholders to assess model reliability, detect biases, and ensure compliance with ethical and legal standards [2]. In particular, post-hoc explainability methods, which generate explanations after model training without altering its architecture [3], have gained significant attention. These approaches enable the use of high-performing models, including those developed by third parties, without requiring access to their internals. In contrast, methods that modify the model itself often demand retraining or architectural changes, which can be computationally expensive and less broadly applicable.

The growing importance of explainability is also reflected in regulatory initiatives like the Union's Artificial Intelligence Act (AIA), which sets legal standards for transparency, fairness, and accountability in AI systems [4]. Meeting these requirements calls for explainability methods that can offer meaningful insights into a model's internal logic. In this context, post-hoc approaches that analyze the information flow throughout the network, rather than modifying the model itself, are particularly valuable, as they offer both practical applicability and deeper structural interpretability.

Building on this foundation, recent research within post-hoc explainability, has explored methods to detect uncertainty in neural networks by analyzing information flow within specific layers.

A methodology, SVD-based Peephole and Clustering [5], has demonstrated promising results in detecting ambiguous decisions by leveraging Singular Value Decomposition (SVD) to extract low-dimensional representations of layer parameters. However, this existing approach mainly targets a single layer of a relatively simple model, which limits its ability to reveal potential shortcomings and the broader applicability of the method.

This thesis builds upon that foundation by extending the analysis to multiple layers within a lightweight network architecture: MobileNetV2. By integrating information from different stages of input processing, the proposed method provides a comprehensive and efficient analysis of the model's internal dynamics, better aligned with the demands of modern, resource-constrained AI applications.

In summary, this research is motivated by the urgent need for explainability in AI. By refining and expanding existing methodologies, this thesis aspires to advance the field of XAI and support the development of AI models that are not only powerful but also interpretable and reliable.

Research Questions

Building on the *SVD-based Peephole* methodology and aiming to push forward explainable AI, this work is driven by the following essential research questions: To achieve this, the following key research questions guide this work:

(1) **Can we extract meaningful information from such a compact and compressed network?**

Lightweight architectures like MobileNetV2 are designed to minimize redundancy and compress internal representations. This research investigates whether these compact structures still retain enough organization and semantic depth to allow for effective interpretability using the peephole method. In the broader context of XAI, this question is crucial: if highly efficient models can still provide interpretable insights into their decision-making processes, it suggests that explainability does not have to come at the cost of efficiency. Understanding how compressed architectures encode and organize information contributes to advancing XAI by revealing whether interpretability methods remain reliable and meaningful even under extreme model compression.

(2) **How to choose the best layers to extract information from?**

Extracting information from every single layer of a network would require unnecessary amounts of computational power. Different layers encode different types of information. Identifying which layers offer the most interpretable and semantically rich representations is a key step toward effective peephole extraction.

These questions guide the development and refinement of the proposed methodology, helping to clearly define the objectives of this thesis.

CHAPTER 1

Background

Building on the discussion in the previous chapter about the need for explainability in AI and the challenges of interpreting deep neural networks, this chapter introduces key concepts essential for understanding the methods explored in this thesis. It provides an overview of explainability in AI and the Peephole methodology, which serve as the foundation for the proposed approach.

1.1. Explainability

Explainability in artificial intelligence ensures that AI models and their decision-making processes are comprehensible to human users. As AI systems grow in complexity, they often function as black boxes, making it difficult to understand how predictions and classifications are made. Explainability plays a key role in fostering trust, accountability, and fairness in AI applications.

Trustworthiness, Explainability and Interpretability

Trustworthiness, explainability, and interpretability are closely connected in AI, each playing a distinct role in ensuring reliable and transparent decision-making. **Trustworthiness** refers to the degree to which an AI system can be relied upon to make fair, consistent, and robust decisions, fostering user confidence and acceptance. **Interpretability** pertains to how well a model's internal mechanisms can be understood by humans, allowing for an analysis of how and why specific predictions are made. While explainability focuses on making AI decisions comprehensible, interpretability delves deeper into understanding the underlying computational processes [3]. This thesis primarily emphasizes **explainability**, aiming to develop methods that enhance interpretability and provide clearer insights into AI decision-making. However, it's important to note that explainable and interpretable systems tend to be perceived as trustworthy. Therefore, although this work focuses on explainability, it also offers indirect insights into the trustworthiness of the model, particularly in how confidently and transparently it reaches its decisions.

Post-hoc Explainability

Post-hoc explainability refers to methods that provide insights into an AI model's decision-making process after it has been trained. These methods do not modify the model's structure but instead analyze its internal workings. By not altering the original AI model, they preserve its performance while enhancing interpretability [3]. Unlike intrinsically interpretable models, which often sacrifice accuracy for transparency, post-hoc techniques can be applied to highly complex models, making them more practical for

real-world applications.
The Singular Value Decomposition (SVD)-based Peephole approach falls under the category of **Concept-based explanation methods (C-XAI)**, which aim to interpret model behavior through human-understandable concepts. These methods explain a prediction, an entire class, or even the behavior of internal network components by referencing patterns that align with semantically meaningful features. Such approaches focus on understanding how deep neural networks make decisions using identifiable concepts, and typically evaluated based on criteria such such as coherence, relevance to the predicted class, semantic clarity, and faithfulness to the actual computations of the model [6]. The SVD-based Peephole method follows this paradigm by revealing concept-level structures within the network's layers that are both interpretable and grounded in the model's internal mechanisms.

1.2. SVD-based Peephole

The peephole mechanism is a technique used to analyze the internal activations of a neural network, offering insights into how information flows through its layers. By examining activations at different stages, peephole methods help detect regions of uncertainty and ambiguous decision-making. In the context of explainability, peephole analysis allows researchers to gain a more detailed understanding of how and why a model arrives at certain conclusions, facilitating better transparency and interpretability of deep learning models.

Peephole analysis is particularly useful in neural networks where complex representations emerge at different layers. By systematically probing these activations, it is possible to track how information transforms throughout the network. This method can highlight discrepancies between layers and detect inconsistencies that may lead to misclassifications or unreliable predictions.

One approach to implementing peephole analysis involves computing a dimensionality-reduced representation of the neural network's activations using SVD [5] which helps highlight the most influential features that contribute to decision-making.

An advantage of peephole analysis is its ability to provide layer-wise insights without requiring modifications to the model architecture. This makes it a non-intrusive technique for post-hoc explainability, allowing researchers to retrospectively investigate neural network behavior and improve interpretability without compromising model performance.

To provide a more rigorous foundation for the methodology, the concepts SVD (1.2), Corevector (1.2) and Peephole (1.2) will formally defined in the following subsections.

Singular Value Decomposition

As previously discussed, one of the main uses of SVD is dimensionality reduction. SVD decomposes the data matrix into three matrices [7].

$$A = U\Sigma V^T, \tag{1.1}$$

where:

- U: where the columns are the left singular vectors of A (capturing the relationships between the rows of the original matrix)
- Σ: where the diagonal entries are the singular values of A (capturing the importance of each component of the original matrix)
- V^T: the transpose of matrix V, containing the right singular vectors of A (capturing the relationships between the columns of the original matrix)

In a layer of a deep neural network, with input $x \in \mathbb{R}^n$ and output $y \in \mathbb{R}^m$:

$$y = f(Wx + b) \tag{1.2}$$

where:

- $W \in \mathbb{R}^{m \times n}$,
- $b \in \mathbb{R}^m$,
- $f()$ is the elementwise function[8].
- m is the number of outputs in the layer function
- n is the number of inputs per layer

In our case, A is the concatenation of the weight matrix W_j and the bias vector b_j of a specific layer j [5]:

$$A^{(j)} = W^{(j)} \mid b^{(j)} \tag{1.3}$$

By reducing the dimentionality of A, we can significantly reduce the computational demands associated with processing the information that flows through the network's layers.
For fully connected layers, W_j is naturally a matrix, so applying SVD directly to A (which includes W_j and b_j) is straightforward.
However, convolutional layers have weight tensors

$$W \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times k \times k}, \tag{1.4}$$

representing C_{out} output channels, C_{in} input channels, and a kernel size of $k \times k$ (assuming square kernels).
Unlike regular matrices, convolutional layers are defined by collections of $k \times k$ kernels arranged across input and output channels ($C_{\text{out}} \times C_{\text{in}}$). This structure cannot be directly decomposed using standard SVD, which requires a matrix format, making it necessary to first unroll the kernels into a compatible representation.
Therefore the convolutional kernel W is first unrolled into a large linear operator

$$\overline{W} \in \mathbb{R}^{(C_{\text{out}} \cdot n^2) \times (C_{\text{in}} \cdot n^2)}$$

that exactly represents the full convolution operation over the entire input space. This operator $\overline{W}$ encodes how the convolution acts on vectorized inputs [9].

Only once this linear operator $\overline{W}$ is constructed can SVD be applied to study the layer's structural properties. The procedure for constructing W is as follows:

(1) Zero-pad each kernel slice $W_{c,d}$ (the $k \times k$ kernel for output channel c and input channel d) to an $n \times n$ matrix $K_{c,d}$, ensuring convolution "wraps" on a torus (circular padding).

(2) Construct

$$B_{c,d} = \mathrm{circ}(K_{c,d}),$$

an $n^2 \times n^2$ *doubly-block-circulant matrix* whose rows are circular shifts of $K_{c,d}$.

(3) Assemble the block matrix

$$\overline{W} = \begin{bmatrix} B_{1,1} & \cdots & B_{1,C_{\text{in}}} \\ \vdots & \ddots & \vdots \\ B_{C_{\text{out}},1} & \cdots & B_{C_{\text{out}},C_{\text{in}}} \end{bmatrix}.$$

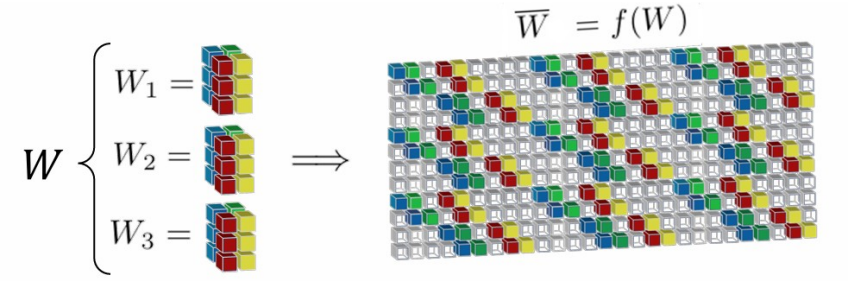

FIGURE 1.1. Toeplitz method: $\overline{W}$ is a reshaped multiple embedding of the weight tensor into a tensor space [10].

Other approaches to enroll convolutional weights for SVD exist, for example, explored in [10]. We chose to adopt the Toeplitz unrolling approach (shown in Figure 1.1) it allows for corevectors of arbitrary size, unlike the method in [10], where the resulting corevectors depend on the activation size of each layer—an undesirable property for the subsequent clustering step.

Corevector

In the Peephole methodology, SVD reduction is applied to A and then the right singular vectors (V^T) are multiplied by the input of that layer. This allows the construction of the corevector: an approximation of the original matrix A with k dimensions[11]:

$$\nu_k(j) = V_k^T h_j \tag{1.5}$$

where:

- V_k^T: is the rank-k approximated V^T, by the selection of the top k singular vectors
- h_j: is the input of layer j

The resulting corevector represents a compressed encoding of the original activation vector h_j, expressed in terms of the most informative patterns identified within the layer. These patterns captured in the top right singular vectors V_k^T are derived from the weight structure of the layer and represent dominant input directions that the layer is most sensitive to. These directions are determined by the layer's learned transformation. They tell us how the layer is structured to act on inputs, rather than patterns from the data distribution itself (so not sample-specific). When we multiply these fixed patterns by the input activation h_j, which is unique to each input sample, the result is a corevector that reflects how that particular sample aligns with the principal patterns of the layer. In essence, for a given sample, the corevector tells us which of these dominant patterns are most "active" or relevant in a specific layer. This means that although a corevector is **sample-specific**, we still capture the **global behavior of the model** by multiplying our input by the right singular vectors.
By working with these corevectors instead of the full high-dimensional activations, we enable an efficient analysis while preserving the essential structure needed for interpretability. These corevectors serve as the foundation for constructing the layer's peepholes, which are then analyzed to reveal how the network organizes and processes information internally.
The extracted corevectors are then analysed to form the samples' peepholes.

Peephole

To construct a peephole for a given layer, we analyze its corevector through a two-step process:

(1) **Unsupervised Clustering of the corevector**
To better understand how the network organizes its internal representations, we apply unsupervised clustering to the corevectors extracted from each layer. As it was said, a corevector represents how the network responds to a specific input at a particular layer. These vectors are high-dimensional and encode features the network has learned to recognize. Importantly, similar inputs tend to produce similar corevectors, meaning that the network "reacts" in comparable ways to related patterns or concepts.
Clustering these corevectors allows us to group together similar internal responses. In essence, we're identifying regions in the activation space where the network consistently expresses a particular kind of response. The idea is that if many vectors are close to each other in this space, they likely reflect a shared underlying feature or concept the network has learned.
Through this process, we can uncover **clusters** of similar activations that may correspond to meaningful visual patterns relevant to the network's decision-making. For example, one cluster might group activations triggered by sharp edges and wheels, while another might correspond to features like white

fur. These clusters give us insight into the types of internal categories the network uses to interpret and classify images [12].
The clustering method chosen was Gaussian Mixture Model (GMM).This method determines the probability of a data point belonging to a cluster. Unlike hard clustering methods, GMM allow data points to belong to multiple clusters with a certain probability, providing a more flexible and nuanced approach to clustering, especially when dealing with overlapping or complex cluster shapes.
Mathematically, the clustering process of corevector v can be defined as the following equation:

$$f(\nu) = \sum_{c=1}^{C} \phi_c \mathcal{N}(\nu \mid \mu_c, \Sigma_c), \tag{1.6}$$

Where:

- ϕ_c represents the probability that a data point belongs to cluster c
- $N(\nu|\mu_c, \Sigma_c)$ the normal distribution of the probability of observing the corevector ν in cluster c
- μ_c and Σ_c the mean and covariance matrix of cluster c, respectively

The result of this step is then a vector of probabilities of belonging to each cluster, for a specific sample (image), which is called the **Membership Probability** (MP) [11].

(2) **Mapping the clusters to classes**
Now we try to understand how the resulting clusters relate to the actual class labels the neural network predicts, given a dataset. We do this by performing an empirical posterior distribution over the training set as the following:

$$\Theta_{c\times s} = [\text{prob}(\Phi(\mathbf{x}) = l \mid \max(f(\nu)) = c)] \mid \; c \in \{0, \ldots, C-1\},\, l \in \{0, \ldots, S-1\} \tag{1.7}$$

Where:

- $f(\nu)$ is the Membership probability
- $\Phi(x)$ is the label assigned to input image x by the NN model Φ
- $max(f(\nu))$ is the most likely cluster associated with the corevector ν at the given layer .
- C the number of clusters
- S the number of classes in the dataset

In simple terms, is this step we try the answer the question "For each cluster, what class does it typically represent?"
Consequently, the result of the empirical posterior a matrix composed of the probability of corevectors belonging to a cluster (rows) to be associated with a class (columns).

Given the Membership probability vector and the empirical posterior matrix, we can form the peephole for each layer of the model by multiplying them:

$$p = f(\nu) * \Theta \tag{1.8}$$

So, the resulting peephole is a weighted average of the cluster-class distributions (Θ), weighted by $f(\nu)$. As a result, we have a sample-specific vector with the size equal to the number of classes of a particular layer. To be more precise, it shows how the distribution of corevectors across clusters maps to a specific probability distribution over a set of concepts (labels). While the function f captures how an input is distributed among clusters, the peephole converts that distribution into class-level probabilities. This allows us to see which concepts are most closely associated with a given input by analyzing how the network organizes its internal representations, revealing how the structure of activations within the network relates to its classification performance. For this reason this "representation" was given the name *peephole* since it allows, by peeking through a "small window", to closely observe the network's decision-making process at a specific layer. Figure 1.2 illustrates the overall scheme of the peephole extraction process

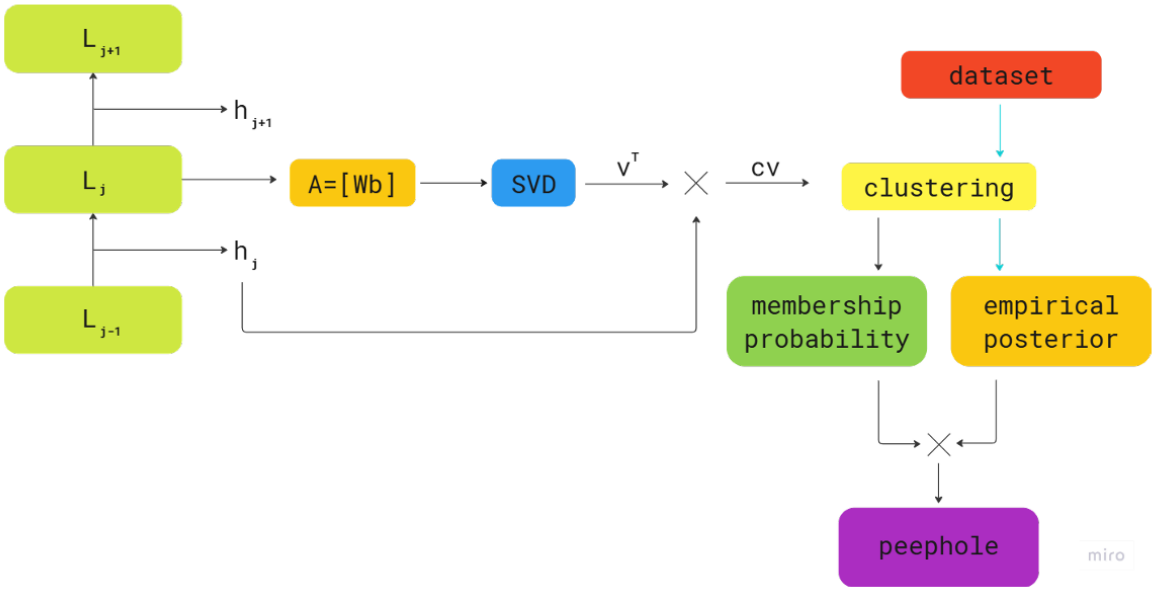

FIGURE 1.2. Peephole extraction and clustering scheme [11]

Now that the fundamental concepts have been established, we can explore existing work related to this methodology. The next chapter reviews relevant literature, examining how different explainability approaches compare and how the SVD-based Peephole method fits within the broader landscape of post-hoc XAI techniques.

CHAPTER 2

Literature Review

This chapter explores the methodology taken for the systematic review and discusses related work to provide a comprehensive foundation for this research.

2.1. Systematic Review

To ensure a systematic and thorough exploration of relevant literature, *b-on* and *Google Scholar* as the primary search platforms. Additionally, specific theses closely aligned with this research, including the foundational work *SVD-based Peephole and Clustering to Enhance Trustworthiness in DNN Classifiers*, were provided by the ARCES department at University of Bologna, where this thesis was being conducted. A total of 36 papers were gathered for the development of this thesis, with some identified through keyword-based searches using terms such as *XAI*, *Explainability*, *Interpretability*, *Trustworthiness*, *Transparency*, and *Deep Neural Networks*. Search queries were constructed by combining these terms (e.g., "*Explainable AI OR XAI AND Trustworthiness OR Confidence OR Interpretability OR Explainability AND Neural Network*") to specifically target post-hoc XAI methods and their applications. Additionally, the snowballing technique was employed, where references from relevant papers led to the discovery of further literature. These resources serve as a fundamental support for the extension of the existing methodology.

2.2. Related Work

There are various approaches to explainability in AI, such as Explaining with Surrogates (which involves building a simpler model that mimics the behavior of a complex model, e.g., LIME [13]) or Explaining with Local Perturbations (which entails modifying small parts of an input to observe how these changes affect the AI's decision, e.g., Prediction Difference Analysis [14]), among others. But we want to focus on post-hoc methods that remain faithful to the model's actual decision-making process (rather than relying on approximations such as surrogate models) and that avoid the potential artifacts introduced by perturbation-based techniques, ensuring that explanations reflect how the original input features influence predictions without modifying the input itself. Considering this, there's some post-hoc methods that are somewhat comparable with the SVD based Peephole:

For example, methods like **Layer-wise Relevance Propagation**[15] (along with DeepSHAP[16], DeepLIFT[17] and others) focus on estimating the importance of input features for a specific decision, making them well-suited for local explanations, that is, they explain why a model made a particular prediction for a given input.[14]. LRP specifically works by redistributing the model's output backward through the network to

determine which input features contributed most to a given decision. This is achieved by propagating relevance scores layer by layer. Its core focus remains on analyzing how input data influences the output rather than interpreting the network's structural properties. This makes it highly effective for local explanations.

However, while LRP and similar methods provide valuable insights into individual predictions, they do not offer a comprehensive view of how the model processes information across layers or how learned representations evolve throughout the network. In contrast, SVD-based Peephole goes beyond analyzing individual feature attributions by examining layer-wise transformations and information compression within the model, allowing for a more holistic understanding of its decision-making process.

There's also **Grad-CAM**[18], which is a technique that generates visual explanations by identifying which parts of an input were most influential in a model's prediction. It does this by analyzing gradient information from a selected layer to determine which areas had the most impact. The result is a heatmap overlaid on the input, providing an intuitive way to understand the model's focus during decision-making [14].

However, while methods like Grad-CAM provides useful insights into how a model processes specific inputs, its explanations remain local, focusing on individual predictions rather than offering a broader understanding of the model's behavior.

While feature attribution and saliency-based methods are valuable for highlighting which input elements influence predictions, they often lack the ability to explain what those patterns represent at a more abstract or semantic level. In image classification there is a growing need to understand not just where the model is focusing, but what kind of high-level concepts it has learned and how those concepts influence its decisions. This brings us to a category of explainability techniques: concept-based methods.

Concept-based post-hoc methods

Peephole can be seen as a concept-based post-hoc method. **Concept-based methods (C-XAI)** are well-suited for image classification tasks among post-hoc explanation techniques because they operate at a higher semantic level, offering explanations in terms of human-understandable concepts rather than individual pixels or gradients. This aligns with how humans interpret visual information and allows for intuitive insights into what the model has learned. Additionally, concept-based methods can capture distributed patterns (e.g., object parts, textures, or abstract features) that are often missed by saliency-based or attribution methods. They also scale well to global explanations, enabling analysis across multiple inputs or classes.

By clustering corevectors, Peephole reveals internal concept-like structures that emerge naturally during training. These structures correspond to patterns the network finds relevant for distinguishing between classes. Through visual tools like conceptograms, the method maps how these concepts evolve across layers and contribute to predictions.

Among concept-based methods, a primary distinction lies in whether they rely on concept-annotated datasets (**supervised**) or operate without such labels (**unsupervised**).

Supervised methods use symbolic concepts (predefined and human-labeled categories) to explain model behavior [6]. An example is **Network Dissection**[19], which aims to understand the role of individual neurons by examining their activations across a dataset and associating them with predefined human-labeled concepts (such as "cat," "dog," or "edge"). This is done by evaluating how strongly a neuron responds to specific features in an input and mapping these responses to predefined visual or semantic concepts.

In contrast, **unsupervised methods** discover latent concepts directly from the model's internal representations, forming a concept basis without human supervision. Both approaches are capable of generating visualizations of the extracted concepts, but their focus differs. Supervised methods aim to link predictions to known, labeled concepts, while unsupervised methods typically explore class-concept relationships and emphasize the discovery and visualization of emergent internal patterns that the network has learned [6].

This makes unsupervised concept-based methods particularly valuable in real-world scenarios where annotated concept datasets are unavailable, impractical to collect, or task-specific concepts are not clearly defined. By extracting concepts directly from the model's learned representations, these methods offer a flexible and scalable approach to interpretability that adapts to the data and the task without prior semantic assumptions.

Moreover, unsupervised methods can reveal emergent internal structures that supervised approaches might overlook, highlighting patterns, intermediate abstractions, or visual features that the network uses internally but are not explicitly labeled. [6] This allows for broader insight into the model's behavior, especially in domains where meaningful explanations may go beyond predefined human categories.

In this context, methods like the SVD-based Peephole offer a way to extract and interpret these latent patterns, without relying on human-labeled concepts reducing the dependence on external supervision.

Several concept-based explanation methods, like ours, use unsupervised approaches tailored to image classification tasks:

- The **Automatic Concept-based Explanations** [20] (ACE) automatically discovers concepts for a specific class without human supervision. It segments input images at multiple resolutions (textures, parts, full objects), embeds these segments in the network's latent space, and clusters them to form candidate concepts. Outliers are removed, and the relevance of each concept is measured using the TCAV score, which estimates the influence of each concept on the class prediction. While ACE captures a variety of patterns, it can produce incoherent or meaningless concepts due to errors in segmentation, clustering, or scoring [6].

- **Invertible Concept-based Explanation** [21] (ICE) enhances ACE by replacing K-means clustering with Non-Negative Matrix Factorization (NMF) for concept extraction. Each NMF dimension reveals a distinct concept, and a model is

trained on top to approximate the model's predictions. The resulting classifier weights indicate the importance of each concept, offering a more interpretable and faithful explanation.

- **Completeness-aware** [22] improves on ACE by focusing on completeness (how sufficient a set of concepts is for predicting a model's outputs). It introduces a completeness score, based on the accuracy achieved when predicting class labels using only concept scores. A concept discovery algorithm is optimized to find maximally complete concepts, while encouraging locality in concept representations (i.e., nearby patches should support similar concepts).
- **Multi-dimensional Concept Discovery** [23] (MCD) extends concept discovery into multiple dimensions by allowing concepts to lie on hyperplanes in the feature space. It uses Sparse Subspace Clustering (SSC) followed by PCA to define concept bases. A model is trained to approximate the model's outputs using these concepts, enabling per-sample completeness scores. MCD provides both concept relevance maps (importance per class) and concept activation maps (visual representation), and demonstrates improved faithfulness and compactness over ACE and ICE.
- **DMA&IMA** [24] focus on identifiability, the ability to provably recover the underlying concepts in data. *DMA* assumes each concept affects disjoint input regions and guarantees recovery even with correlated components. *IMA* relaxes this assumption by requiring only orthogonal concept vectors. While *DMA/IMA* do not generate explanations or visualizations, they evaluate which conceptual structures can theoretically be extracted from neural representations.

The SVD-based Peephole method sets itself apart from these concept-based explanation techniques in both its **objective** and its **approach to interpretability**. While many existing methods focus on discovering human-interpretable concepts to explain individual predictions or class behavior, Peephole emphasizes understanding the internal structure and information flow across layers of a neural network. Its goal is not only to identify which concepts matter, but also to reveal how those concepts emerge, evolve, and contribute to decision-making throughout the model.

Another key difference lies in the fact that Peephole does not rely on an auxiliary neural network or autoencoder to generate explanations. Many modern explanation methods, such as ICE, Completeness-aware, DMA & IMA , MCD, Deep Embedded Clustering[12] and others, use additional models to process activations, discover latent structure, or predict outputs. While these techniques can be powerful, they also introduce a problem of redundancy: explaining a neural network by using another neural network. This not only adds computational and architectural complexity, but also raises a concern: if the explanation depends on another model, then doesn't that model also need to be explained? This creates a potential cycle of explanations that moves further away from the original goal of clarity and transparency.

In contrast, the Peephole method offers a more direct and **mathematically grounded alternative**. It uses Singular Value Decomposition, a well-established linear algebra technique, to reduce high-dimensional activations into compact, structured representations called corevectors. These are then clustered to uncover meaningful patterns, allowing for the creation of interpretable visual tools like conceptograms. Importantly, every step in the Peephole pipeline is transparent, interpretable, and grounded in geometric principles, not learned heuristics. This makes the method not only easier to trust, but also easier to analyze and verify.

By avoiding reliance on neural networks to interpret other neural networks, the SVD-based Peephole approach stays closer to the core values of explainable AI: clarity, simplicity, and interpretability rooted in theory rather than yet another layer of abstraction.

SVD-based Peephole Method

It is essential to discuss the most significant recent methodology, introduced in *SVD-based Peephole and Clustering to Enhance Trustworthiness in DNN Classifiers*[5], which serves as the foundation for this thesis. This approach compares the model's output with the results of a clustering technique, offering a more transparent and intuitive decision-making process. The clustering is applied to information passing through a specific network layer, represented by low-dimensional vectors obtained through SVD applied to the layer's parameters.

Due to its reliance on SVD applied to layer parameters, it allows for a deeper understanding of how the network processes information across different layers, rather than just interpreting individual predictions.

By clustering these low-dimensional representations, the method moves beyond local explanations (which focus solely on specific input-output relationships) and instead provides a global perspective on how decision boundaries are formed within the network. This enables a more comprehensive analysis of the model's internal behavior, making it possible to detect patterns, structural dependencies, and sources of ambiguity that influence multiple predictions rather than just a single classification instance.

This thesis aims to build upon and extend the SVD-based peephole by applying the procedure across multiple layers of a very light neural network. In addition, this work extends the peephole extraction process to convolutional layers, moving beyond the approach used in SVD-based Peephole and Clustering to Enhance Trustworthiness in DNN Classifiers [5], which focused solely on fully connected layers. By integrating information extracted from different stages of input processing, we seek to enhance explainability at each layer, leading to a more comprehensive understanding of the model's internal decision-making process. This extension allows to explore the scope of adaptability and capacity of this method applied in such a compact network.

This multi-layer interpretability across not only fully connected layers approach was explored in *Visual Transformer Conceptogram Through Intermediate Activation Analysis* [11], using the Vision Transformer (ViT) architecture.

While ViT offers a clean and transparent structure that makes it useful for visualizing and explaining interpretability methods, it presents several limitations for the peephole extraction task. First, it is data-hungry and typically requires large-scale pretraining [25]. It is also highly demanding in terms of memory and computational resources [26], making it a poor fit for the peephole framework, which already involves computationally intensive operations such as large matrix manipulations. Given that efficiency is increasingly prioritized in modern AI applications, a lighter and more practical architecture is preferred.

Second, despite its usefulness as a conceptual model, ViT doesn't reflect the architectures most commonly deployed in real-world systems today. Its simplicity, while helpful for early-stage testing and explanation, limits its realism. This is part of what motivated our choice to evaluate the peephole approach on a lightweight yet architecturally complex model that better mirrors current trends in applied deep learning.

Summary

While many explainability techniques exist, they often lack scope, model faithfulness, or a balance between structural and data-driven insights. Local methods like LRP and Grad-CAM explain specific decisions but miss the broader network dynamics, while global approaches like Network Dissection provide structural insights but may disconnect from the data. Autoencoder-based methods add complexity by requiring another neural network, reducing transparency.

The SVD-based Peephole method addresses these limitations by combining data-driven and structural analysis, capturing both information flow and network transformations. Unlike other methods, it provides a layer-wise, global, and mathematically grounded explanation.

This thesis extends the original peephole approach by applying it across multiple layers of a lightweight and structurally complex model: MobileNetV2. Unlike earlier work focused on large models and fully connected layers, this study investigates whether meaningful explanations can be extracted from compact, efficient architectures that better reflect current trends in AI.

CHAPTER 3

Setup

In this Section, we focus on three key aspects essential to setting up for the peephole extraction process. First, we introduce the neural network architecture selected for this methodology as well as its structural design and the efficiency-driven innovations that make it suitable for our thesis. Second, we outline the training process used to prepare the network for analysis, including the choice of optimizer, learning rate schedule, and performance metrics. Finally, we detail the selection of layers for peephole extraction. This involves analyzing the singular value of each layer and evaluating their empirical posterior matrices to assess how well the clustered activations align with class labels.

3.1. Mobilenet

The neural network chosen for the task of peephole extraction was MobileNet v2. Previously this peephole extraction method was tested in networks like VGG-16 and ViT. But we wanted to explore a efficient lightweight network since this peephole task is very computationally heavy (namely the large-scale matrix multiplications) and that efficiency seems to become more prioritized in the industry throughout the years [27] [28]. Lightweight deep learning refers to the development of compact neural networks that maintain high accuracy while requiring fewer computational resources, such as processing power and memory, compared to traditional models [29]. Moreover, VGG-16 and ViT, while useful for illustrating the mechanics of peephole extraction due to their simple straightforward designs, do not perfectly reflect the kinds of architectures commonly used in modern AI applications. MobileNetV2, on the other hand, incorporates a range of nuanced design decisions and architectural complexities, making it a more realistic and challenging benchmark for evaluating the capabilities and limitations of the peephole approach.

Mobilenet efficiency innovations

MobileNet V2 is a convolutional neural network architecture designed for resource-constrained environments. It's a 53-layer deep model that prioritizes efficiency and performance.

Unlike conventional deep networks that stack heavy convolutional blocks, MobileNetV2 builds its performance on a series of smart design choices that minimize computational cost without significantly compromising its accuracy:

- MobileNetV2 relies on depthwise separable convolutions. Instead of using full convolutions across all channels, these are split into two steps: a depthwise convolution (one filter per input channel) followed by a pointwise (1×1) convolution (shown in Figure 3.1)that mixes the outputs across channels. This drastically reduces the number of parameters and operations compared to traditional convolutions [28].

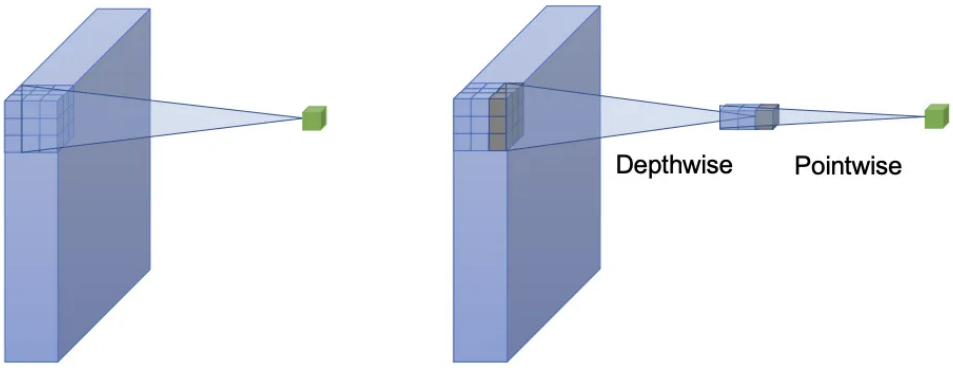

FIGURE 3.1. Standart convolution and depthwise seperable convolution [30]

- MobileNetV2 uses batch normalization after each convolutional step to stabilize training and allow the model to converge faster. Batch normalization works by normalizing the inputs of each layer in a neural network, which helps to stabilize the learning process and allows for the use of higher learning rates [31].
- A Bottleneck Residual Block (Figure 3.2) follows a wide–narrow–wide structure to reduce computation by compressing features with 1×1 convolutions before and after processing. MobileNetV2, on the other hand, uses an Inverted Residual Block with a narrow–wide–narrow design: it first expands the input (**Expansion layer**), processes it with depthwise convolutions to capture spatial patterns efficiently (**Depth-wise layer**), and then then projects it back to a compact form (**Projection layer**). This inverted structure works better in mobile and resource-constrained settings because it avoids information loss during early compression and takes full advantage of lightweight convolutions [28].

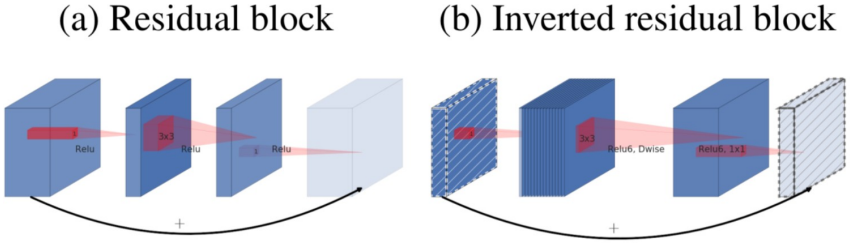

FIGURE 3.2. Residual block and inverted residual block [32]

- Another defining feature of MobileNetV2 is its use of skip connections (Figure 3.3), which allow gradients to propagate more easily during training and help retain low-level features throughout the network [28].

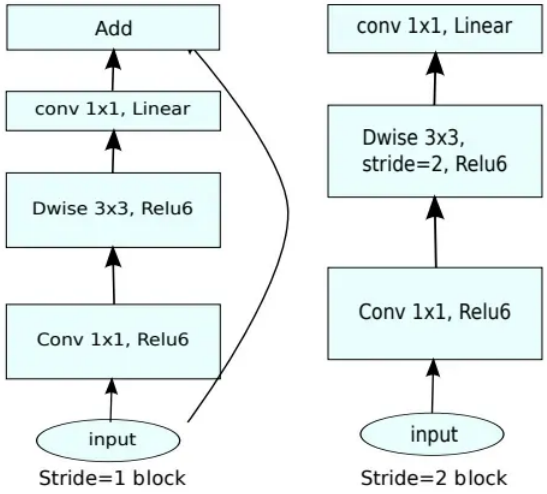

FIGURE 3.3. Skip connection in Mobilenet v2 (when *Stride*=1)

- The use of groups in MobileNet input channels also greatly impacts efficiency. In a grouped convolution, the input channels are split into groups, and each group is convolved independently with its own set of filters (illustrated in Figure 3.4). This reduces both computation and number of parameters, at the expense of connectivity across channels [28].

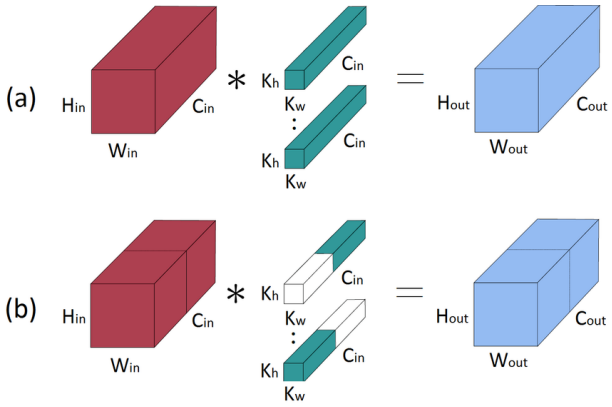

FIGURE 3.4. Standard vs. grouped convolutions: (a) In a standard convolution, each filter is convolved with all of the input's channels; (b) In a grouped convolution with two groups, half of the filters are applied to each half of the input for a 2× reduction in parameters used. [33]

Together, these innovations make MobileNetV2 highly efficient: it achieves competitive accuracy on benchmarks like ImageNet using a fraction of the parameters and compute of larger models.

This unconventional and light weight network makes MobileNetV2 an good candidate for testing the peephole approach introduced in this thesis.

Mobilenet Architecture

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

FIGURE 3.5. Architecture of Mobilenet v2 [28] where t: expansion factor in bottleneck blocks; c: Number of output channels for the block; n: Number of times this block is repeated and s: Stride of the first block in this stage

Like illustrated in Figure 3.5, MobileNetV2 begins with a standard 2D convolution layer that reduces the input image's spatial resolution while expanding the channel depth to 32. The main body of the network is made up of 7 bottleneck blocks, which are the key to its efficiency and flexibility. Each bottleneck consists of three layers: an **expansion layer** that increases the number of channels by a factor t, a **depthwise convolution** that applies spatial filtering independently to each channel, and a **projection layer** that compresses the result back to a lower-dimensional space with c output channels.

Residual connections are used when the input and output dimensions match and the stride (s) is 1, helping to preserve information and stabilize training. For example, a block with $t = 6$, $c = 24$, $n = 2$, and $s = 2$ takes a 16-channel input, expands it to 96 channels, applies a depthwise convolution, and projects it down to 24 channels. After the sequence of bottleneck blocks, a final 1×1 convolution increases the channel count to 1280. This is followed by a global average pooling layer that reduces spatial dimensions to 1×1, and a final 1×1 convolution that outputs k logits corresponding to the target classes.

Having chosen the neural network to perform the peephole extraction, the next step is to choose a dataset and fine-tune it.

3.2. Training

The training process and choice of dataset play a central role in shaping a neural network's internal representations - the activation patterns that arise in response to input images. In the context of the peephole framework, which interprets these patterns through dimensionality reduction (producing **corevectors**) and unsupervised clustering (using Gaussian Mixture Models), the training data effectively defines the kinds of structures

that emerge. At the core of this interpretive process is the **empirical posterior matrix**, which estimates the probability of a class label given a cluster activation in the reduced space. This matrix links unsupervised cluster structure to supervised learning goals, capturing how activation patterns relate to class labels. Its reliability, and by extension the usefulness of the peephole surrogate model, depends on how well the training data represents the task's underlying categories and structure.

The chosen dataset was CIFAR-100 which is a collection of 60,000 32x32 color images, divided into 100 different classes, with 600 images per class. These 100 classes are further organized into 20 broader "superclass" categories. The dataset is commonly used for training and testing machine learning models, especially in the field of computer vision.

For this study, the model was fine-tuned on the 100 classes of CIFAR-100 using the *Adam* optimizer. A *ReduceLROnPlateau* (*rOp*) learning rate scheduler was applied, starting with a learning rate of 0.001, reducing it by a factor of 0.1 if the validation performance plateaued for 5 consecutive epochs. These parameters were selected based on the overall accuracy and the gap between training and validation accuracy, aiming to avoid overfitting while maximizing performance. The best model achieved a Top-1 accuracy of 78.07% and a Top-5 accuracy of 94.75%, meaning the correct class was among the top five predictions nearly 95% of the time.

Figure 3.6 is a confusion matrix representing the classification performance of the MobileNet V2 model fine-tuned on the 100 classes of CIFAR-100 dataset. The matrix has been aggregated over the 20 superclasses of CIFAR-100 providing a high-level view of how well the model distinguishes between broad semantic categories.

We can observe that there's a strong diagonal line indicating an overall correct classification, which represents an accuracy of 87.80%, which is higher than the accuracy obtained when considering all 100 individual classes. For example, we can see that *Vehicles_1*, *trees*, non-*insect invertebrates*, and *reptiles* have strong diagonal dominance, indicating high classification accuracy for these categories. On the other hand, *Large man-made outdoor things* is confused with *large natural outdoor scenes*, likely due to visual similarity in scene-level context. And *Fruit and vegetables* vs. *flowers* and *food containers* shows some confusion which is understandable given texture/color similarities.

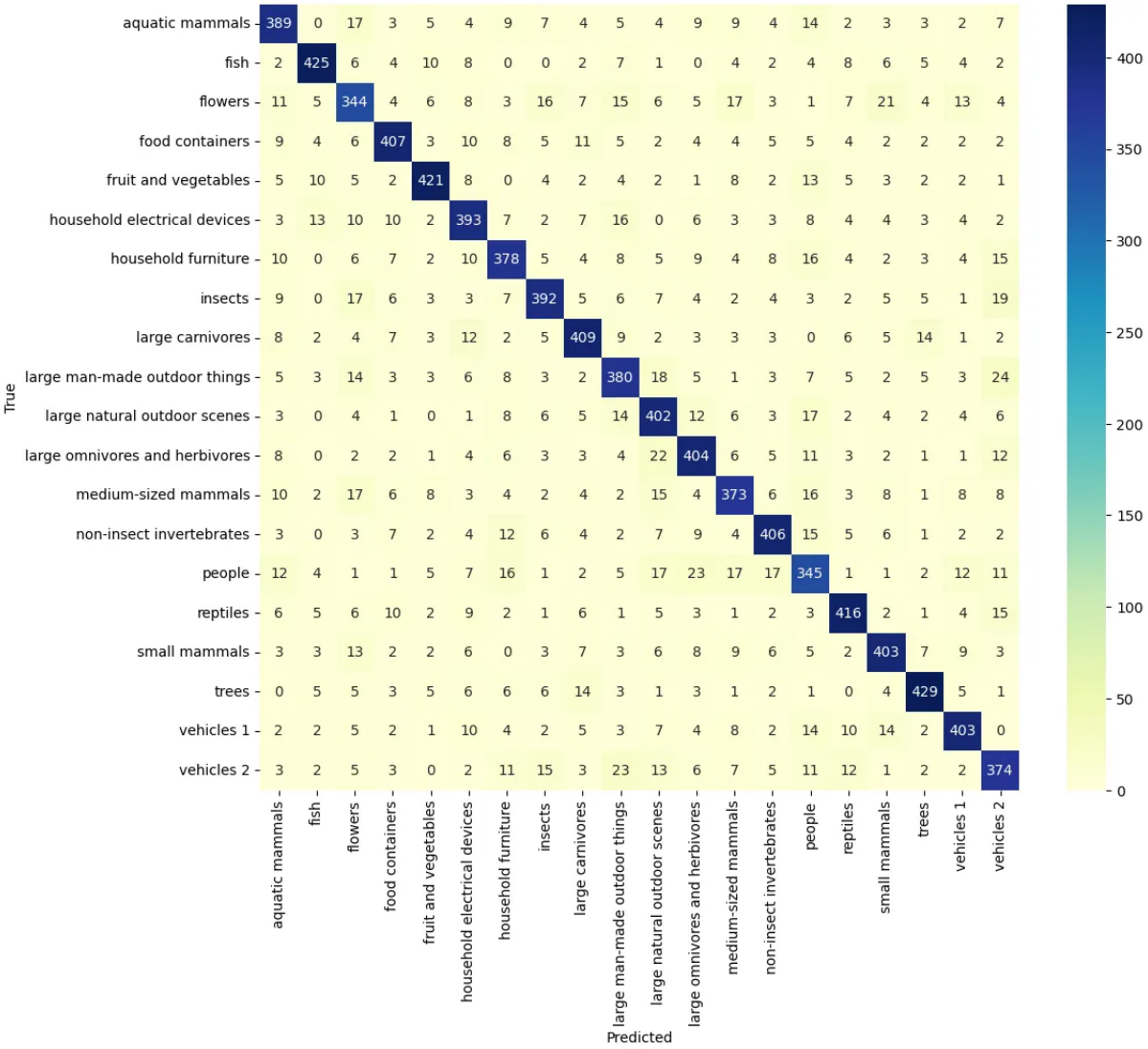

FIGURE 3.6. Confusion matrix of MobileNet fine tuned on Cifar-100: The rows represent the true superclasses (truth labels) and the columns represent the predicted superclasses by the model. Each cell shows the number of test samples truly from class i but predicted as class j.

With the training of the neural network completed, the missing setup step is choose the layers to extract the peepholes.

3.3. Layer selection

This peephole method can be applied in both fully connected layers and convolution layers. But to extract the peepholes of every single layer requires unecessary computational power since MobileNet has 51 convolutional layers and 1 fully connected. To clarify the nomenclature and organization of the layers, we provide a detailed structural breakdown of MobileNet's architecture in Appendix A.

For that reason in this Section we will explore the decision making process of choosing the proper layers to analyse in this thesis.

For this we analyse two properties:

- The singular values: For each layer, we analize the singular values present in the sigma matrix resulted from the SVD decomposition method. These represent "scale" or "strength" of the linear transformation represented by a matrix and reveal important properties about how informative is each layer.
- The empirical posterior: We analize the empirical posterior of each layer which is a matrix composed of the probability of corevectors belonging to a cluster (rows) to be associated with a class (columns). Our objective is to see if the classes are under-represented in our clustering

Singular Values

Singular values are non-negative numbers that represent the "scale" or "strength" of the linear transformation represented by a matrix. Mathematically, they are the square roots of the eigenvalues of $A^T A$, where A is the activation matrix. When ordered and placed along the diagonal of the Sigma matrix in the SVD factorization, they indicate how strongly the transformation represented by the matrix acts along each corresponding direction.

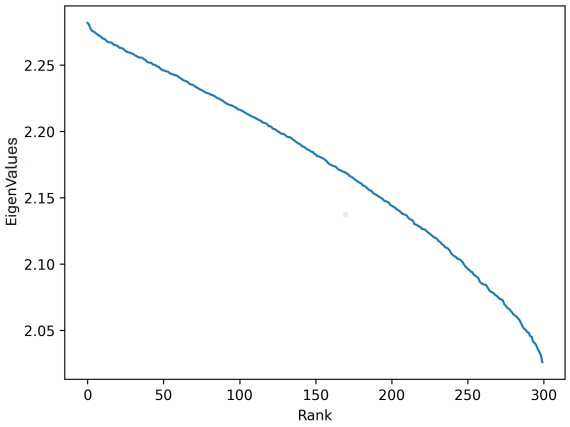

FIGURE 3.7. Singular vectors of layer `features.5.conv.0.0`

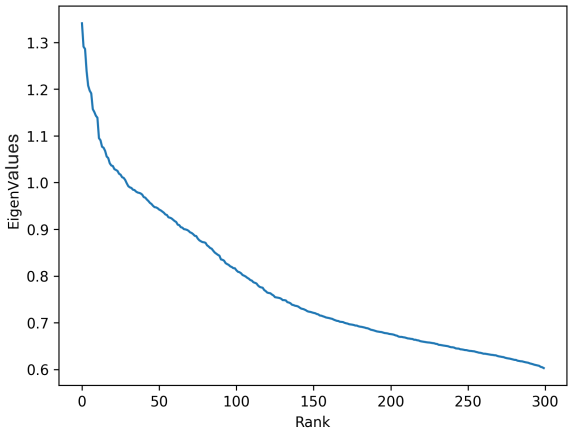

FIGURE 3.8. Singular vectors of layer `features.9.conv.1.0`

The plots 3.7 and 3.8 illustrate the first 300 singular values from the Σ (Sigma) matrix resulting from the Singular Value Decomposition (SVD) of activation matrices from two layers from different bottlenecks in MobileNetV2: Bottleneck 3 and Bottleneck 4. The shape of the singular spectrum reveals fundamental properties about the **structure and informational richness** of the layer's activation. A decaying graph implies that most of the variance is concentrated in a few principal directions. This means better clustering and interpretability, as the data forms compact and structured representations. In contrast,

a slow decay indicates that the data is spread across many dimensions, making it high-dimensional and entangled, which complicates clustering and reduces the clarity of any semantic structure inferred.

In essence, a curved or steeply descending spectrum is more ideal for unsupervised clustering and for building a reliable empirical posterior matrix. A flatter spectrum implies overlapping distributions in the latent space, weakening the ability to map clusters to class behavior.

Looking at the two plots, we observe notable differences in the singular value decay patterns of the layers `features.5.conv.0.0` and `features.9.conv.1.0`, which have direct implications for peephole extraction.

- In the left plot, representing `features.5.conv.0.0`, the singular values decay slowly and almost linearly. There is no sharp drop in the early ranks, and the values decrease gradually across all 300 components. This pattern suggests that the layer's activations are spread out across many dimensions, with no small subset of directions clearly dominating the structure. As a result, the representation is likely high-dimensional and diffuse, making it harder to identify meaningful groupings through clustering. Without strong geometric compression, the corevectors extracted from this layer may be noisy and less aligned with semantic concepts.
- In contrast, the second plot, corresponding to `features.9.conv.1.0`, shows a much steeper initial drop in singular values. The first few components carry most of the variance, and the curve flattens significantly after the early ranks. This indicates that the layer's activations are more compact and organized, with a clear set of dominant directions capturing the majority of the information.

Peephole extraction is fundamentally about revealing structure, so choosing layers with meaningful compression (as shown by curvature) improves our chance of finding explainable patterns.

For this reason, the curvature of the singular values is a indicator of the layer's compatability for peephole analysis. It tells you whether the layer naturally organizes information along dominant semantic axes, which in turn helps the success of clustering and posterior estimation. Thus, analyzing singular value decay is about choosing layers that inherently structure information in a way that's interpretable.

Choosing the top k-singular values

An additional practical consideration involves deciding how many singular values to retain from each layer's decomposition, that is, determining the cutoff index for the Σ matrix (resulting from the SVD reduction method). While the final classification layer naturally limits this to 100 components (equal to the number of classes), other layers produce much larger matrices. For instance, the expansion layer of the last Bottleneck

has dimensions of [47040 × 7840], and extracting all singular values from such a large matrix would be prohibitively expensive in terms of computation and memory.

To balance interpretability with feasibility, we chose to retain the **top 300 singular values** for most layers. This cutoff preserves a substantial portion of the layer's structure while keeping the computation feasable.

To better justify and understand this decision lets observe 2 layers of the last bottleneck with a much bigger cut (1000). This comparison helps illustrate how much representational information is actually concentrated in the top components, and whether our chosen cutoff adequately captures the meaningful structure of the layer

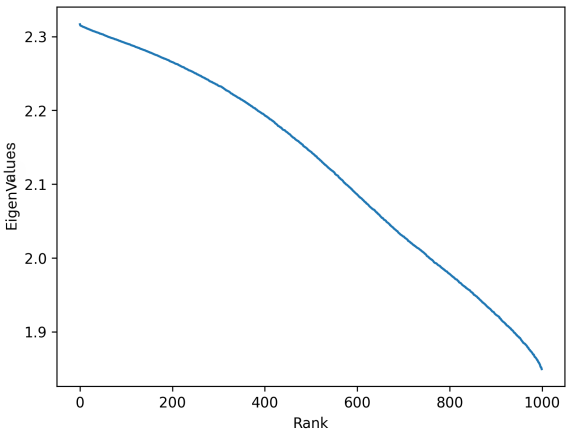

FIGURE 3.9. Singular vectors of layer `features.5.conv.0.0` (1000 rank)

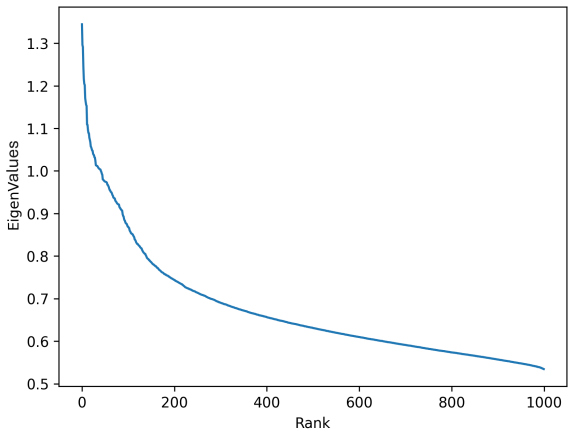

FIGURE 3.10. Singular vectors of layer `features.9.conv.1.0` `(1000 rank)`

From Figures 3.9 and 3.10 we can see that the profile of the singular values only changes slightly with a larger cut-off. Moreover, the Sigma matrix lists singular values in descending order by definition, meaning the largest, most significant values always appear first. As a result, the early portion of the spectrum captures the most informative structure of the data, regardless of where the cutoff is applied. This means that each additional singular value is contributing less and less unique or meaningful information. We're mostly capturing finer details or even noise, specially considering that we're dealing with such low values. So, cutting off at 300 would retain the dominant structure in the data while discarding the tail-end values that contribute marginally.

Empirical Posterior

Another valuable step in evaluating layer suitability for peephole extraction is assessing how well the clustering captures class-relevant information at each layer. To do this, we compute the empirical posterior matrix, which reflects the relationship between the unsupervised clusters formed from the corevectors and the supervised class labels.

The empirical posterior is a probability matrix derived from applying GMM clustering to the corevectors of a given layer. Each row in this matrix corresponds to a cluster, and each column corresponds to a class. The value at position (i, j) represents the probability that a sample in cluster i belongs to class j, essentially showing how frequently class j appears within that cluster. In this way, the empirical posterior gives a direct measure of **how representative or class-specific each cluster is**.

This analysis is performed using a baseline number of clusters equal to the number of classes ($n = 100$).

For the purpose of explainability it is useful to set the number of clusters equal to the number of classes since this one-to-one ratio creates a natural mapping that simplifies interpretation. Ideally, each cluster can be associated with a dominant class, making it easier to track which internal patterns the model uses to distinguish between categories.

By using number of clusters = number of classes, we aim for a direct and interpretable alignment between internal representations (clusters) and output concepts (classes).

While the actual relationship is rarely perfectly one-to-one in practice, using an equal number provides a baseline interpretability framework. It ensures each class has the opportunity to be represented by at least one dominant cluster, making it easier to assess whether the network has developed distinct, class-specific internal representations. That said, this mapping can be adjusted depending on the desired level of abstraction. Using fewer clusters may result in **more general concepts**, where each cluster captures broader visual patterns shared across multiple classes. Alternatively, using more clusters can uncover **finer-grained distinctions** within or between classes, highlighting subtle patterns that the model uses for more nuanced discrimination. The choice of cluster count ultimately depends on whether the goal is to capture global themes or detailed variations in the network's learned representations.

In Figure 3.12 we can see that, for example, the cluster 99 is almost solely class 57 (*lawn_mower*). This means that the clustering is capturing a pattern in the activations of a concept or feature in class 57. We can also see that for example cluster 95 is half class 13 (*bottle*) and class 16 (*bus*), which means that features common in both classes are captured by this cluster. This means that, for our purpose (extracting meaningful, interpretable concepts that closely align with the CIFAR-100 classes), the ideal empirical posterior heatmap would display a clear, structured pattern: one bright (yellow) square per row and one per column. In other words, each cluster should correspond predominantly to a single class, and each class should be primarily captured by one cluster. This one-to-one mapping ensures that the internal representations learned by the network can be clearly linked to specific semantic categories, which is the core goal of concept-based interpretability. This assumption holds when the number of clusters is set equal to the number of classes (e.g., $n = 100$ in CIFAR-100). However, the same framework can be adapted to different levels of abstraction. For example, if we choose $n = 20$ clusters instead, and we're working with the 20 CIFAR superclasses, each cluster would ideally

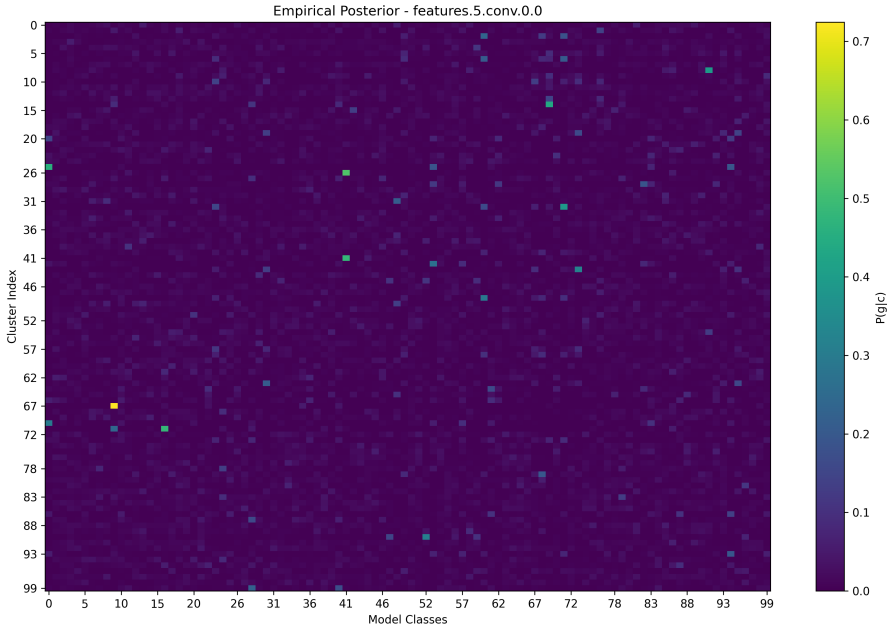

FIGURE 3.11. Empirical posterior heatmap for layer `features.5.conv.0.0`

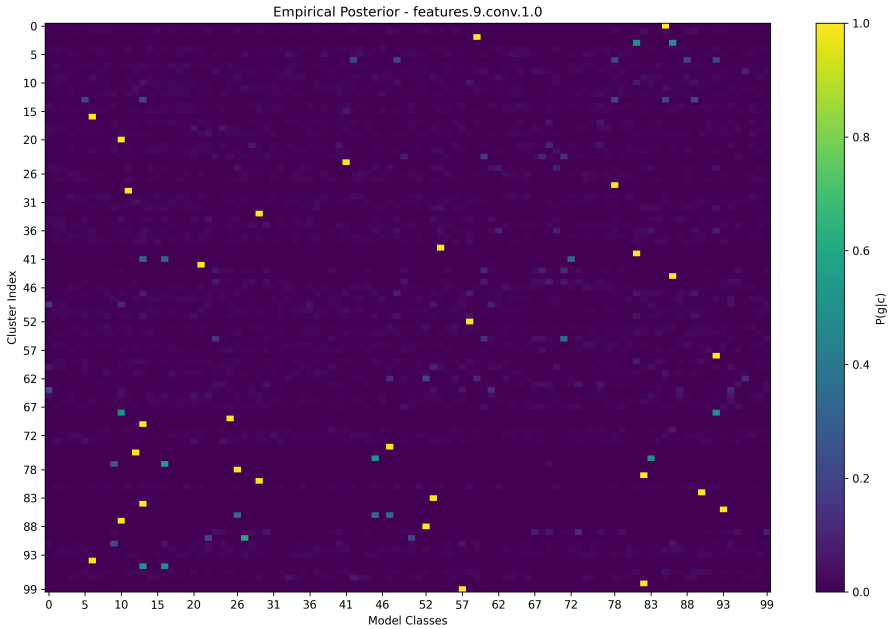

FIGURE 3.12. Empirical posterior heatmap for layer `features.9.conv.1.0`

represent a broader concept grouping around 5 fine-grained classes. In this case, the clusters reflect more general, higher-level concepts rather than specific categories. This flexibility in cluster count allows the method to scale concept granularity depending on the interpretability goal: whether the focus is on precise class-specific patterns or broader semantic groupings.

From the two empirical posterior heatmaps shown, we can clearly observe a difference in how representative each layer is in terms of class-conditional clustering behavior.

- In the heatmap 3.11, corresponding to `features.5.conv.0.0`, the posterior distribution is very diffuse. Most values are low, and there are very few yellow squares (values close to 1). The matrix lacks localized bright regions. This suggests that the activations at this layer do not form well-separated, semantically meaningful representations. The clustering likely groups noisy or overlapping patterns, and the lack of dominant associations between clusters and class labels confirms that this layer is not very useful for peephole extraction or interpretation.
- In contrast, the heatmap 3.12, corresponding to `features.9.conv.1.0`, shows a much more structured and informative pattern. There are several bright, localized regions, indicating strong and confident associations between certain clusters and specific classes. This is a key property we want from layers used in peephole extraction: clear mappings between unsupervised clusters and supervised labels. The presence of well-separated, high-probability regions implies that this layer encodes class-relevant structure in a way that is interpretable. Rather than relying solely on visual inspection of the empirical posterior heatmaps to assess layer quality, we introduce a more objective, quantifiable approach by calculating two key metrics: **class coverage** and **cluster coverage**.

Rather than relying solely on visual inspection of the empirical posterior heatmaps to assess layer quality, we introduce a more objective, quantifiable approach by calculating two key metrics: class coverage and cluster coverage. Class coverage measures the percentage of classes that are meaningfully represented by at least one cluster, while cluster coverage measures the percentage of clusters that are meaningfully associated with at least one class. These metrics are critical for interpretability: if a class is not represented by any cluster, or if a cluster does not correspond to any class, it indicates a gap in the model's internal representation that undermines the goal of extracting meaningful, class-aligned concepts.

To determine whether a cluster or class is "represented," we apply a probability threshold of **0.8**. Specifically, if no entry in a given cluster row of the empirical posterior exceeds 0.8, we consider that cluster **non-representative**. Likewise, if no entry in a given class column reaches the 0.8 threshold, we conclude that the class is **not being represented**. This criterion ensures that only strong, confident associations are counted, providing a clearer and more rigorous basis for selecting layers suitable for peephole analysis.

We can observe interesting things on graph 3.13: On average depth-wise layers are more representative in general than projection and expansion layers. This is not surprising, given their roles in the MobileNetV2 architecture.

Depthwise layers use a 3×3 kernel to slide over each channel, detecting local patterns (edges, textures, etc.) in the H×W plane. Expansion/projection are 1×1 convolutions, they only mix information across channels at a single spatial location and have no "reach" into neighboring pixels.

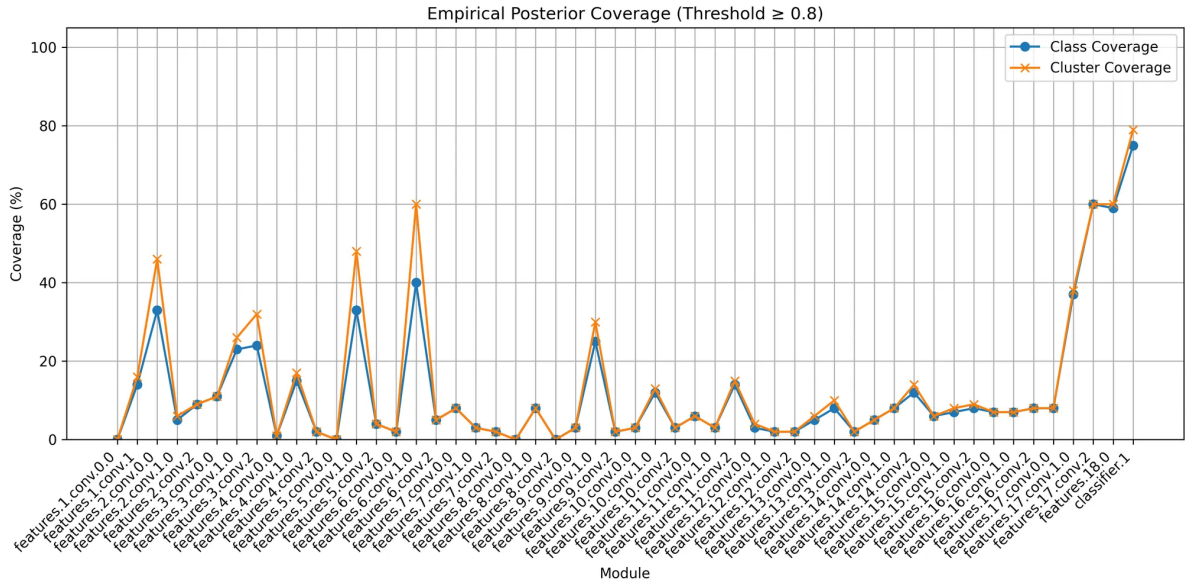

FIGURE 3.13. Class and cluster coverage of each layer in Mobilenet v2 (Using GMM with $number of clusters = 100$)

Moreover, after expanding the channel count, depth-wise conv applies ReLU6 separately within each channel so you get spatial feature extraction plus non-linearity per channel. Expansion layer also has a *ReLU6* after but only acts on the channel-pooled activations at each pixel; projection skips non-linearity entirely to preserve information [28].

This distinct roles of each layer within the bottleneck structure could likely explain why depthwise layers tend to produce more class-relevant representations.

As we can see from Figure 3.13, the layer being close to the classifier does not mean it's necessarily more meaninful. MobileNetV2 is designed for efficiency, not just accuracy which can lead to behaviors that differ from more conventional networks. This is why MobileNetV2 presents a compelling test case for peephole analysis, as it evaluates whether explainability methods can uncover meaningful representations in a model optimized for doing more with less.

We can observe that middle layers are way less representative in comparison to early layers and late layers. A possible explanation is the fact that small adjustments in lower layers cause bigger effects than similar changes in higher layers. That's because deeper layers magnify alterations that happen early in the network, making lower-layer parameters more influential in shaping the final output [34].

On the other hand, the closer you get to the classifier, the more specific and class-discriminative the concepts become [35][36].

This helps possibly explain why the final few layers, particularly the last three or four, are the most representative: they consolidate and refine the high-level features learned throughout the network, aligning them closely with the output classes. These layers are

where abstract patterns are translated into concrete decisions, making them especially valuable for interpretability and concept-based analysis.

Moreover, interstingly, a layer's coverage tends to align with its SVD profile. For instance, the layer `features.5.conv.0.0` displays a singular value spectrum indicative of highly dispersed activations across many dimensions, an undesirable trait for peephole extraction, as discussed in the 3.3 Subsection. Consistent with this, its empirical posterior shows 0% coverage. In contrast, `features.9.conv.1.0`, which exhibited a more favorable and concentrated SVD profile, achieves an average coverage of 27.5% (25% class coverage and 30% cluster coverage), suggesting that its internal representations are more structured and class-relevant.

Based on the coverage percentage of each layer, we exclude the ones with less than 5% average coverage. Leaving us with the layers of Table 3.1:

B1	`features.1.conv.1`
B2	`features.2.conv.0.0, features.2.conv.1.0, features.2.conv.2,` `features.3.conv.0.0, features.3.conv.1.0, features.3.conv.2`
B3	`features.4.conv.1.0, features.5.conv.1.0, features.6.conv.1.0`
B4	`features.7.conv.0.0, features.8.conv.1.0, features.9.conv.1.0,` `features.10.conv.1.0`
B5	`features.11.conv.0.0, features.11.conv.2,` `features.13.conv.0.0, features.13.conv.1.0`
B6	`features.14.conv.1.0, features.14.conv.2,` `features.15.conv.0.0, features.15.conv.1.0,` `features.15.conv.2, features.16.conv.0.0,` `features.16.conv.1.0, features.16.conv.2`
B7	`features.17.conv.0.0, features.17.conv.1.0, features.17.conv.2`
Last Conv	`features.18.0`
Fully connected	`classifier.1`

TABLE 3.1. Selected layers for peephole extraction (organized by Bottleneck)

So we selected 14 depth-wise layers, 8 expansion layers, 7 projection layers, the last convolutional and the fully connected layer.

Now that we decided which layers we will perform the svd-based and the dimension, its time for the next step: forming the peepholes.

CHAPTER 4

Experimental Results

In this Section, we evaluate how conceptograms and superconceptograms behave under different prediction scenarios. Using the CIFAR-100 dataset, we conduct experiments at both the class level (100 classes) and superclass level (20 groups). The goal is to understand how the visual patterns in conceptograms reflect the network's internal decision process across varying conditions.

4.1. Conceptogram

We will now vizualize peepholes by experimenting with conceptograms: a diagram where you can vizualize the concepts of each class that the network is capturing in its layers when a certain sample is fed into the NN.

In our case, as shown in Figure 4.1, the y-axis represents the output classes of CIFAR-100, while the x-axis corresponds to selected layers from MobileNetV2, with each column representing a peephole: a compact representation of a neural network layer, formed by clustering its reduced activation patterns. The colored regions indicate the strength of association between each peephole and each class, based on the empirical posterior probabilities. In simple terms, each rectangle shows how much that layer "lights up" for a certain class. We also highlight the top-5 most activated classes, which are determined by summing the peephole values across all layers for each class; the classes with the highest totals — the most "yellow" rows — reflect the ones most strongly activated by the network, regardless of the final output. On the far right, a bar shows the final output probabilities produced by the neural network's softmax layer. To begin, we present an example that illustrates how the conceptogram reveals the internal decision-making process of the network. The visualization in Figure 4.1 shows how different layers of MobileNetV2 contribute to the classification of an image labeled as *woman*. The network correctly predicts the class *woman* with a high confidence of 95.66%, and the conceptogram helps us trace how that decision is built layer by layer.

From left to right, the conceptogram shows the progression of activations across selected layers, with each column representing a peephole (a compressed encoding of the layer's corevectors). The rows correspond to class concepts, highlighting how strongly each concept is activated at each layer.

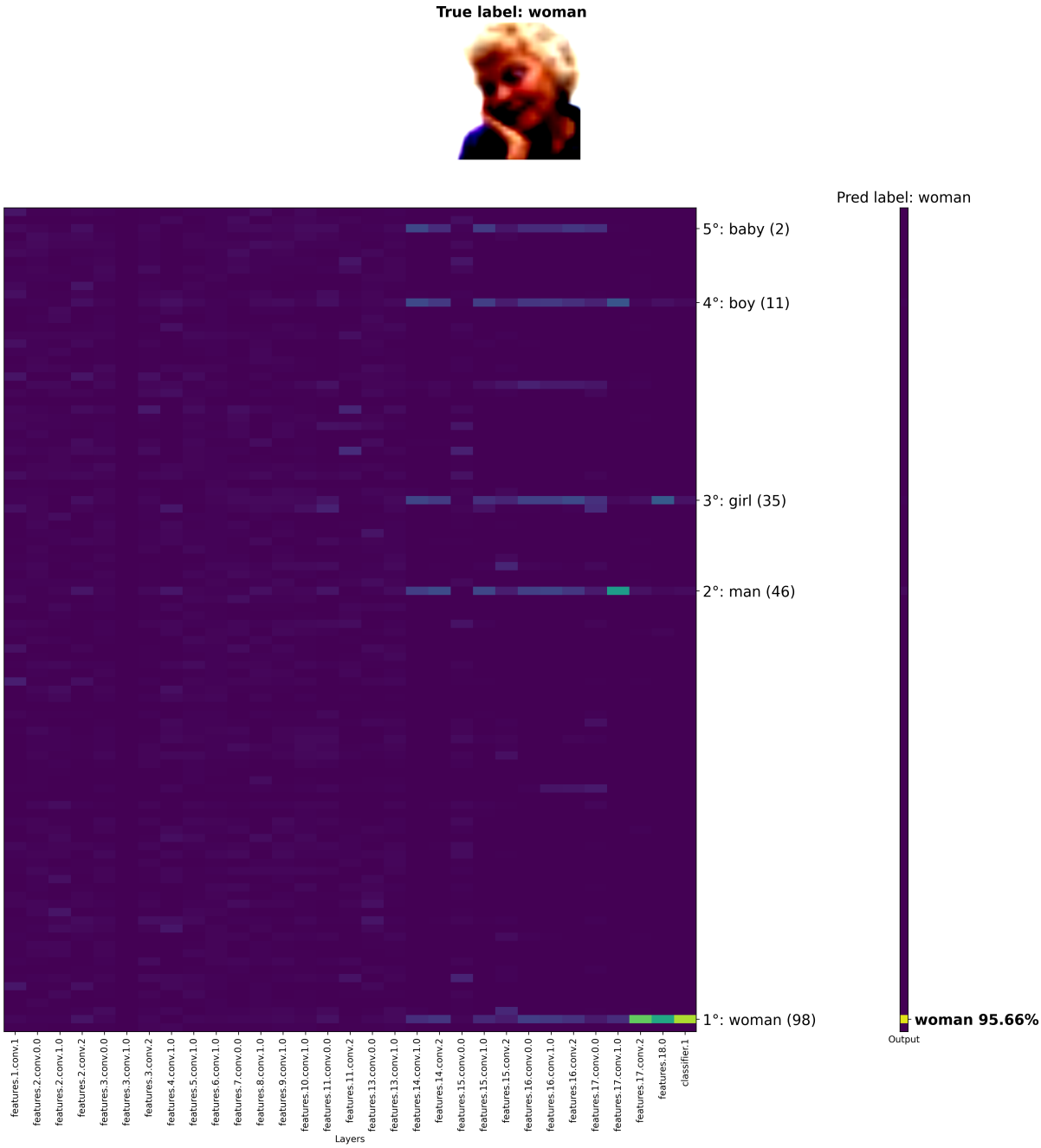

FIGURE 4.1. Example of conceptogram of class *woman* and the corresponding final outputs of the network

One of the most important observations is that early layers in the network exhibit relatively sparse and diffuse activations. These layers do not clearly differentiate the class *woman* from other classes. This aligns with the known behavior of convolutional neural networks: early layers tend to extract general-purpose features like edges and textures, which are not highly class-specific.

As we move deeper into the network, we begin to see stronger and more focused activations, particularly in the final convolutional blocks and the classifier layer. In these late layers, the *woman* class emerges as the dominant concept, with clear and concentrated activations. This suggests that the network refines its understanding of the image progressively, arriving at a class-specific representation only near the output.

Interestingly, some related classes such as *man*, *girl*, *boy*, and *baby*, show moderate activations in certain layers. These activations indicate shared visual features among these classes, likely due to common facial or human characteristics. However, these signals are eventually suppressed or overridden in the final decision, demonstrating that the model is able to resolve the subtle differences that distinguish *woman* from these semantically similar categories.

Overall, this visualization reinforces the notion that class-specific representations are formed late in the network. It also illustrates the utility of SVD-based peepholes in interpreting how individual layers contribute to a prediction. The conceptogram effectively shows that while early layers provide general visual information, the decision-making power of the network lies predominantly in its later stages, where abstract, high-level features are assembled into a confident and correct classification.

Layer Selection and Its Impact on Interpretability

Next, we present two conceptograms: one only with the selected layers and another with all layers of MobileNetV2. The goal is to assess whether removing certain layers (those with low representation) affects the interpretability of the network's decision-making process.

By comparing the two figures, we observe that the conceptogram with the selected layers still captures the decision-making process effectively. The information lost from the omitted layers appears to be minimal, and the overall interpretability remains intact. The observation of this conceptogram and many others suggest that the removal of layers with weak or diffuse representations may have been a sound choice, helping simplify the visualization and preserving efficiency in our analysis.

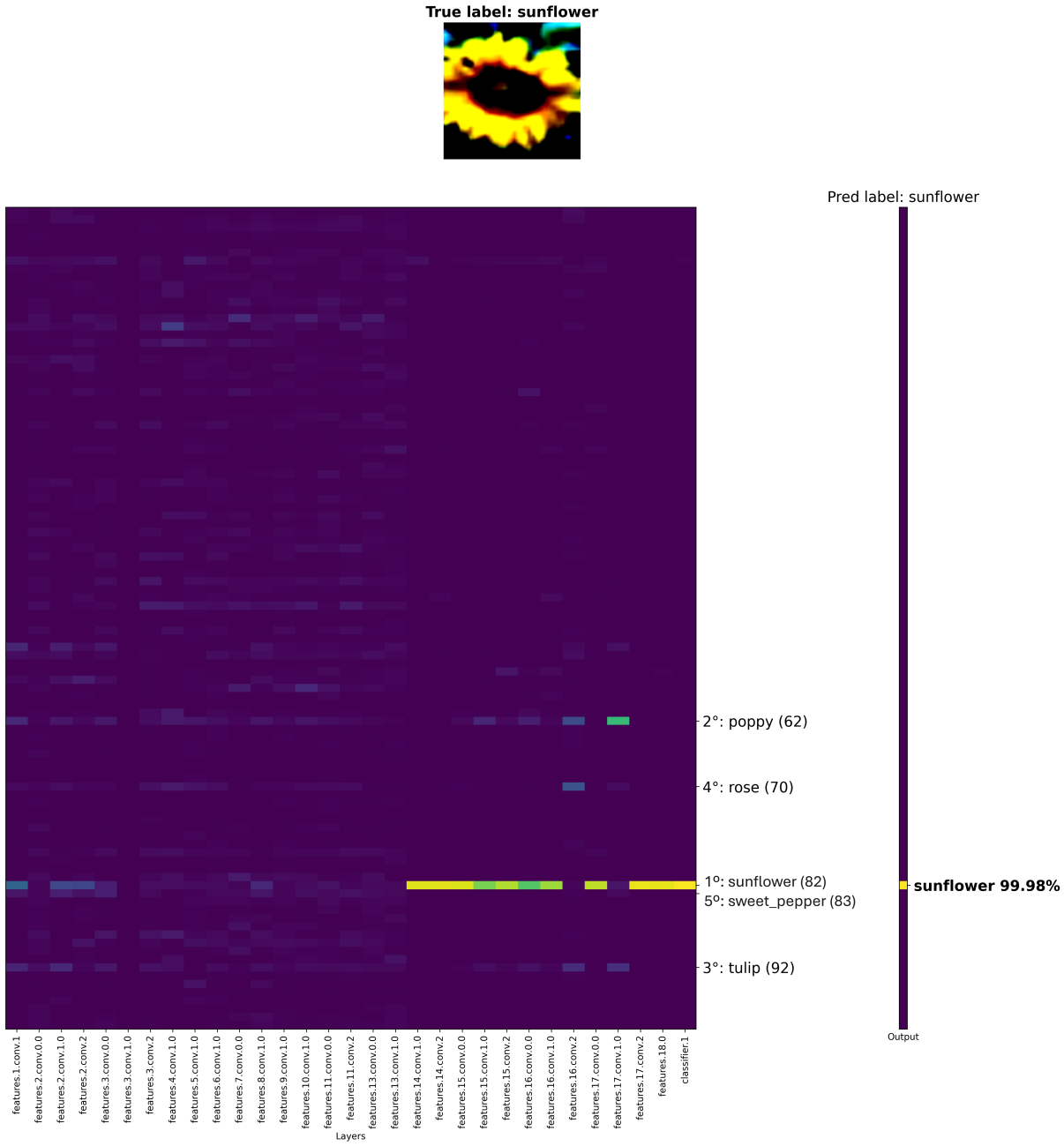

FIGURE 4.2. Conceptogram of the selected layers

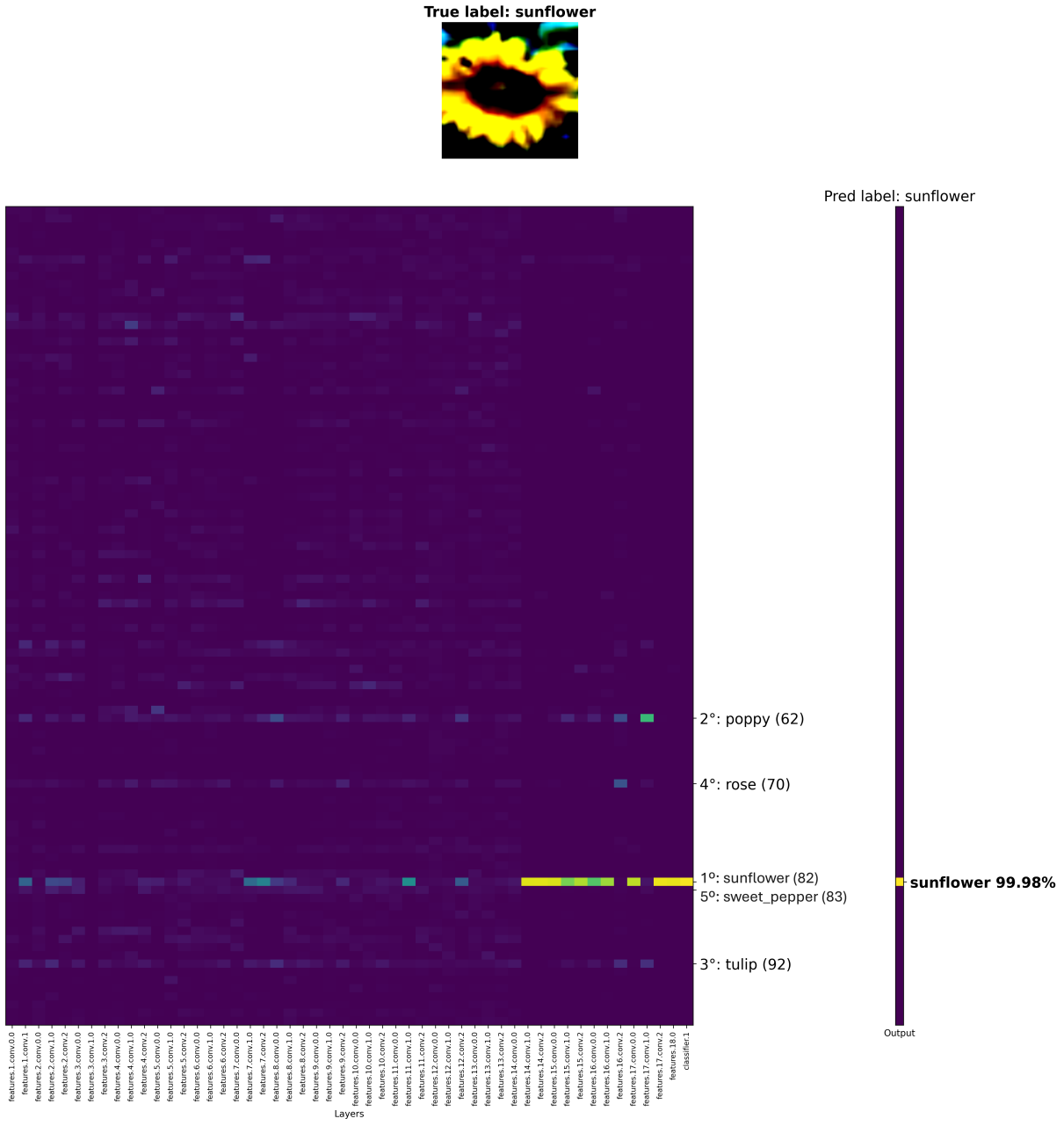

FIGURE 4.3. Conceptogram of all layers of MobileNet

Conceptogram Analysis of High-Accuracy Superclass

In this experiment, we will select the superclass that the trained model predicts most accurately, as indicated by the confusion matrix in Figure 3.6 of the Setup Section. Notably, the superclass *tree* is correctly predicted 429 times out of 500, representing the highest accuracy among all superclasses.

We found it valuable to compare a correctly classified sample with a misclassified one from this class to observe how the network's internal representations differ in each case.

Despite both inputs belonging to the same high-accuracy class (trees), their internal processing paths and final predictions diverge in informative ways.

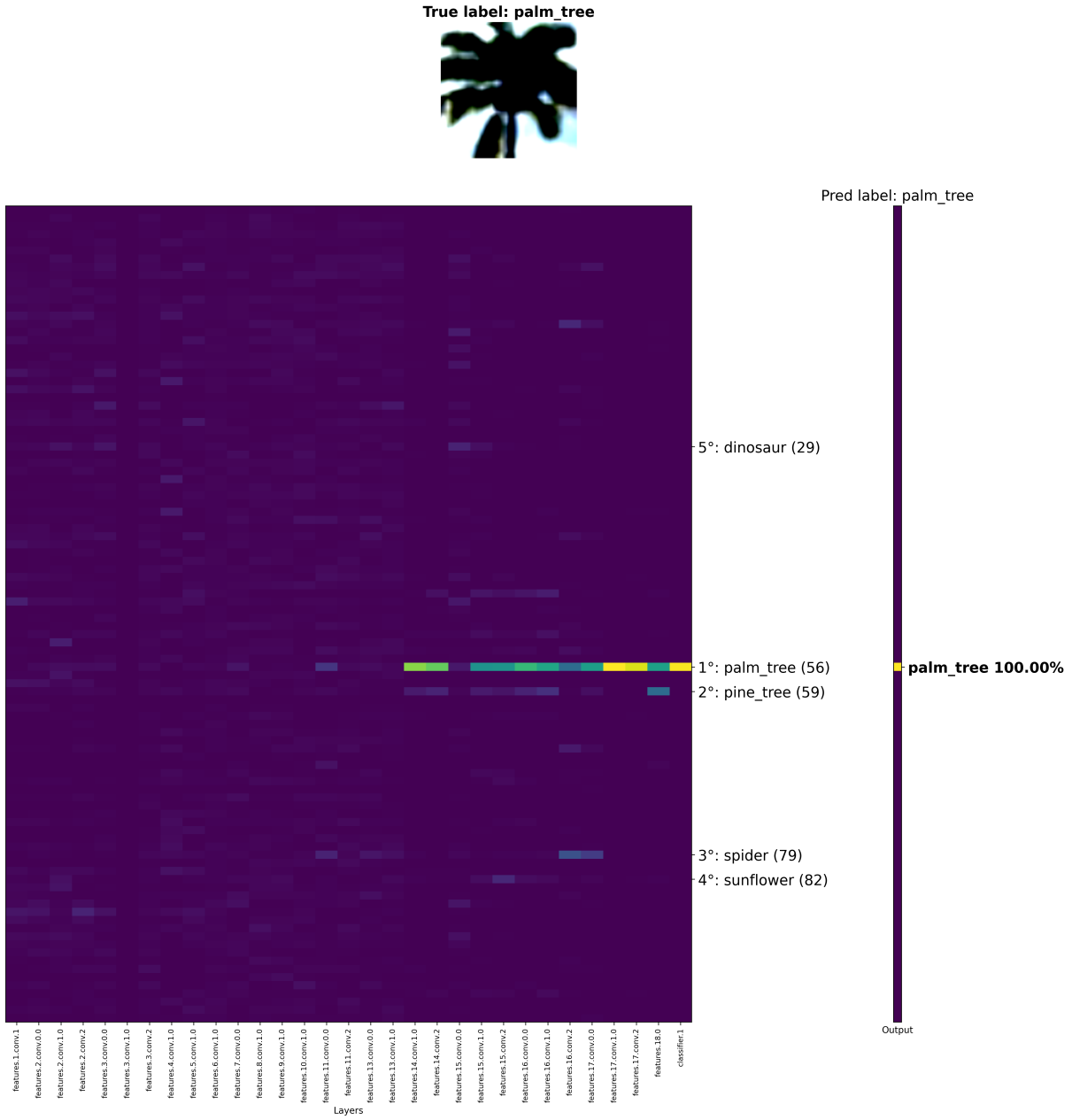

FIGURE 4.4. Conceptogram of a correctly predicted label of superclass *trees*

- In the conceptogram 4.4 (true label: *palm_tree*, predicted label: *palm_tree*), we observe a strong and focused activation for the correct class beginning in the later middle layers and becoming especially pronounced in the final layers. The most dominant class concept (*palm_tree*) is clearly emphasized in one band of layers, which appears to stabilize and solidify the correct prediction. The alignment of this internal representation with the class label is consistent and almost no other class concept competes significantly in terms of activation strength. This conceptogram suggests that the internal features required to recognize a *palm_tree* are clearly formed and resolved in the network.

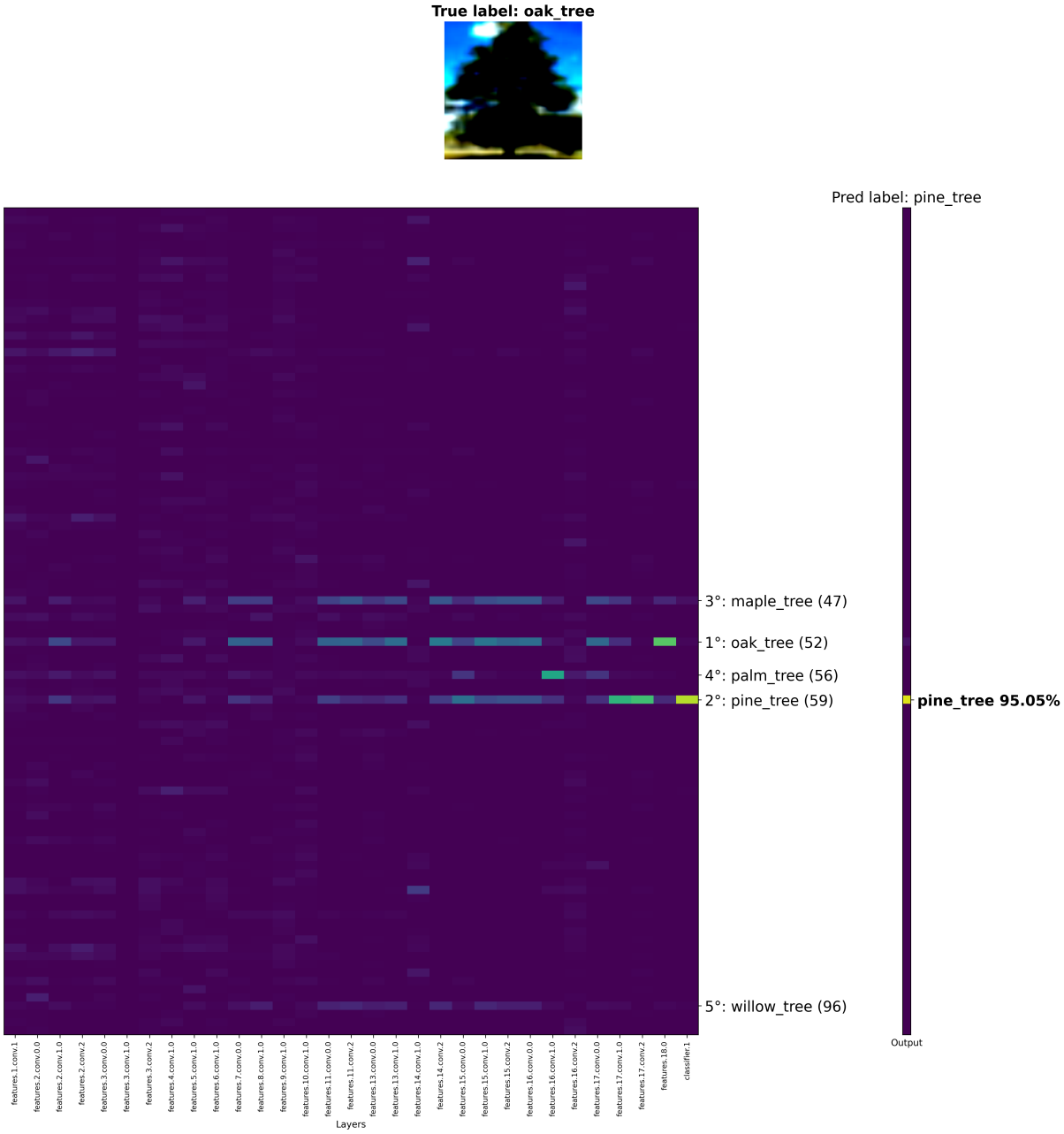

FIGURE 4.5. Conceptogram of a incorrectly predicted label of superclass *trees*

- In contrast, the conceptogram 4.5 (true label: *oak_ tree*, predicted label: *pine_ tree*) shows a more dispersed pattern of activation. The activation patterns are more distributed, possibly indicating less distinctive features in the sample itself, which may have led the network to conflate it with another tree category. Despite *oak_ tree* being ranked **first** among the top 5 concepts based on internal activations suggesting that it is the most prominent class within the layer-wise concept distribution, the network ultimately predicts *pine_ tree* with high confidence (95.05%). This apparent contradiction highlights a subtle but important aspect of how the peephole mechanism interacts with the network's final decision

logic. The conceptogram shows that *oak_ tree* concepts are active across several layers, particularly in the middle stages, yet the *pine_ tree* concept becomes more dominant and sustained in the **later layers**, especially as the output vector is formed. This implies, in this particular case, that the network's final output is disproportionately influenced by late-layer features, which seem to confuse the *oak_ tree* sample for a *pine_ tree* due to possibly overlapping visual characteristics.

From these two examples, we can observe that correct predictions are associated more with localized concept activation (a more distinctive continuous row), while misclassifications often result from overlapping representations that fail to distinguish the target class cleanly (several distinctive non-continuous rows).

As a next experiment we will see how the network behaves in a low-confidence prediction:

- The conceptogram 4.6, which has only 48.29% confidence, shows a much more dispersed pattern compared to the higher confidence conceptograms. The activations are faint and scattered across many layers, and there is no clear cluster of layers where the network consistently detects strong features. Additionally, several other tree-related classes (forest, *maple_ tree* and *pine_ tree*) also have moderate activation levels. This suggests that the network is not confident in distinguishing the specific features that characterize a willow tree. Instead, it activates a broader set of concepts that seem to overlap with other similar categories.

The key conclusion from these visualizations is that strong, localized, and class-specific concept activations correlate with confident and interpretable predictions. Meanwhile, scattered and ambiguous activations across multiple classes reflect uncertainty and semantic overlap, revealing the challenge of fine-grained classification within the same superclass.

4.1.1. Conceptogram Analysis of Low-Accuracy Superclass

As it was previously done, we will analyse 3 different conceptograms: a correctly and incorrectly predicted sample' conceptogram and a low-confidence predicted conceptogram. But this time for the class that the networks get wrong more often. As indicated by the confusion matrix in Figure 3.6 of the Setup section, the superclass "flower" is correctly predicted 344 times out of 500, representing the lowest accuracy among all superclasses.

Once again we can observe a very clear difference between the correctly predicted conceptogram and the incorrectly predicted.

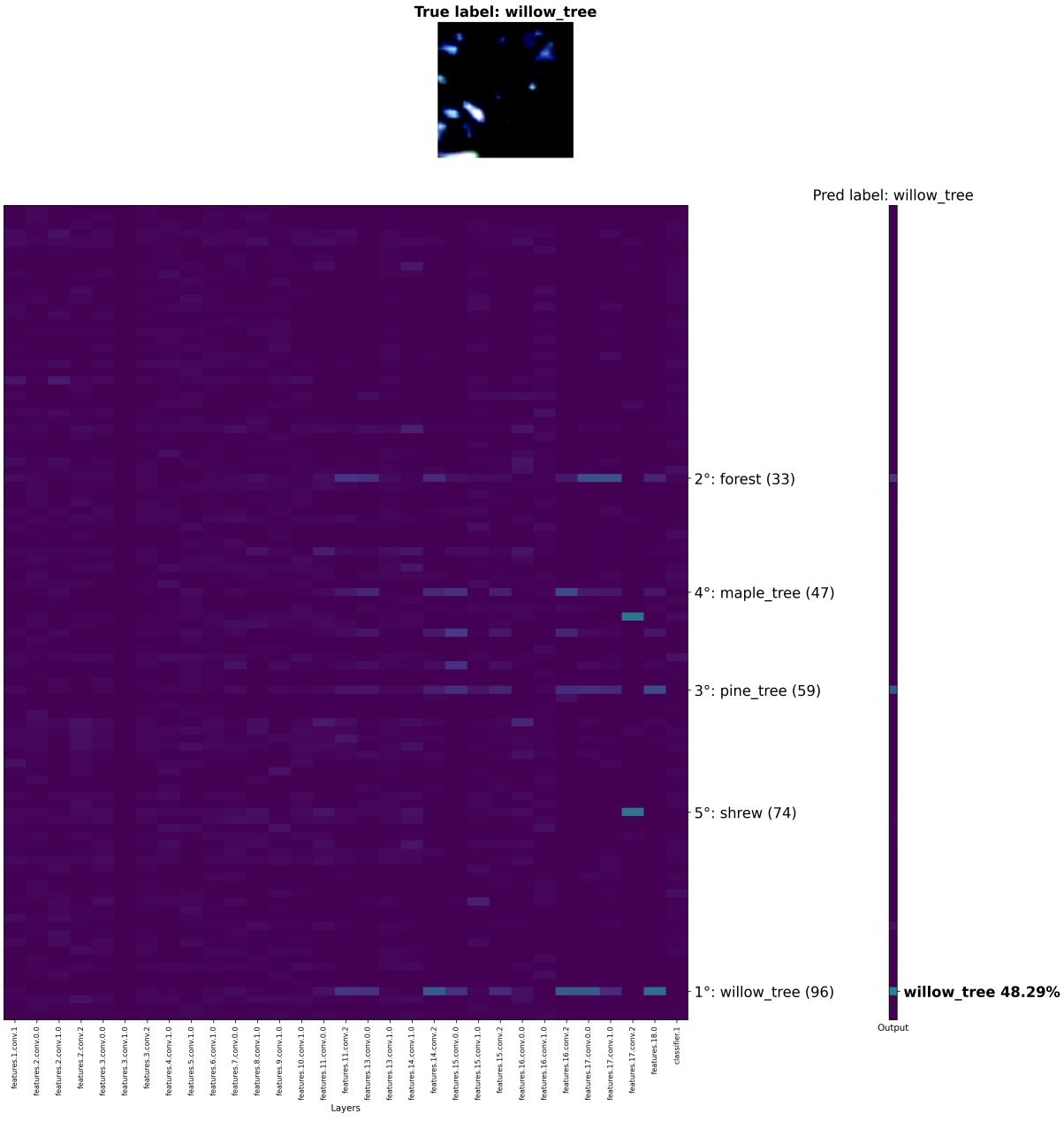

FIGURE 4.6. Conceptogram of a sample predicted with low confidence

- In conceptogram 4.7 (true label: *sunflower*, predicted label: *sunflower*), we see a strong and focused activation for the correct class since very early layers, becoming especially pronounced in the final layers. The internal features required to recognize a sunflower are clearly formed and resolved throughout the network.

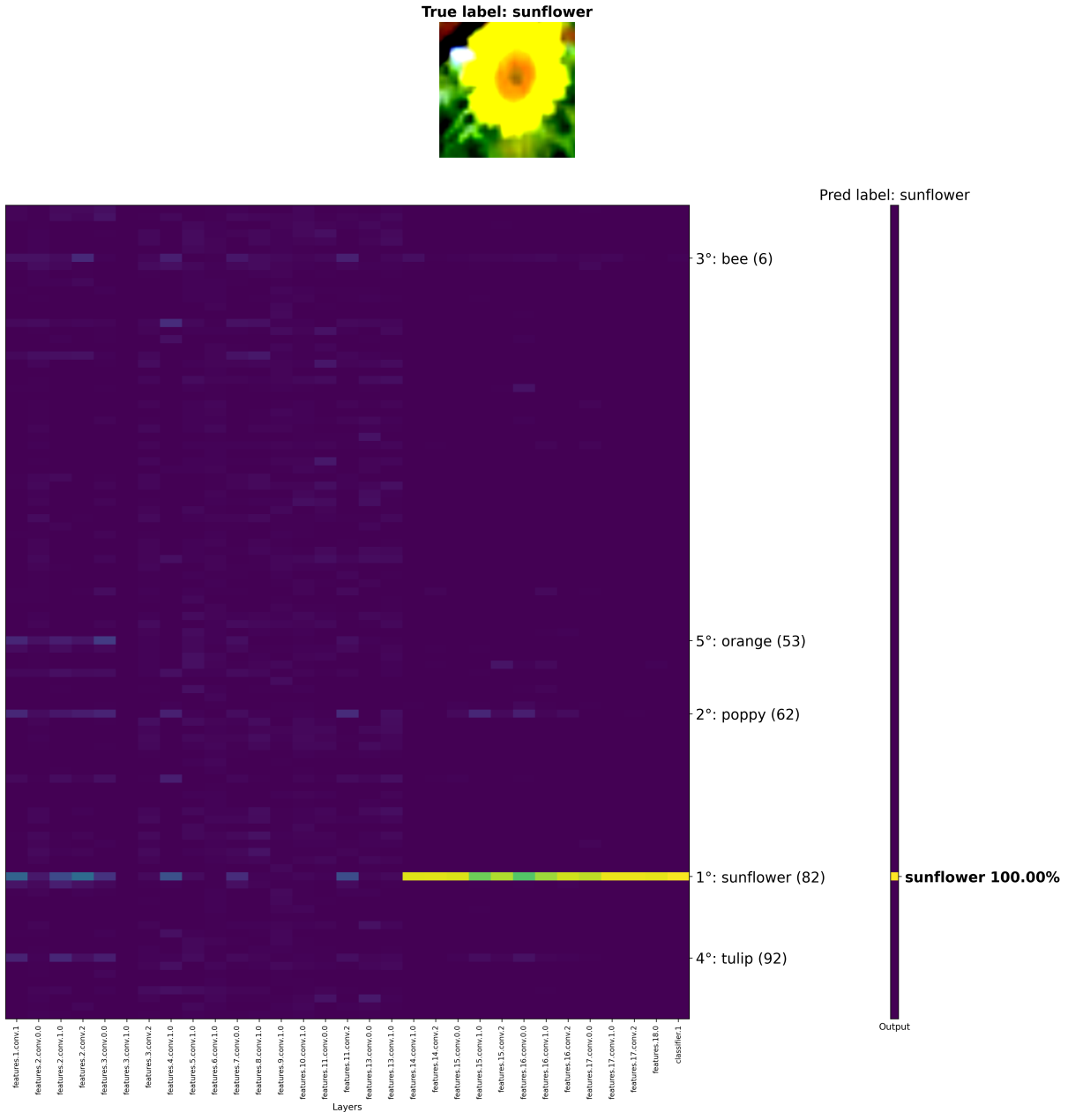

FIGURE 4.7. Conceptogram of a correctly predicted label of superclass *flowers*

- In conceptogram 4.8 (true label: *rose*, predicted label: *orchid*), the activation patterns are more distributed between visually similar classes (where 4/5 correspond to the same superclass). Comparing to the incorrectly predicted *oak_tree*, it seems that the network is catching more similarities in low-level features (for example curves or color pallet) while in the *oak_tree* case it has the rows more highlighted in the mid to late layers suggesting that was confusing with other classes because of more detailed concepts (like the type of leaf).

Now for the low-confidence prediction:

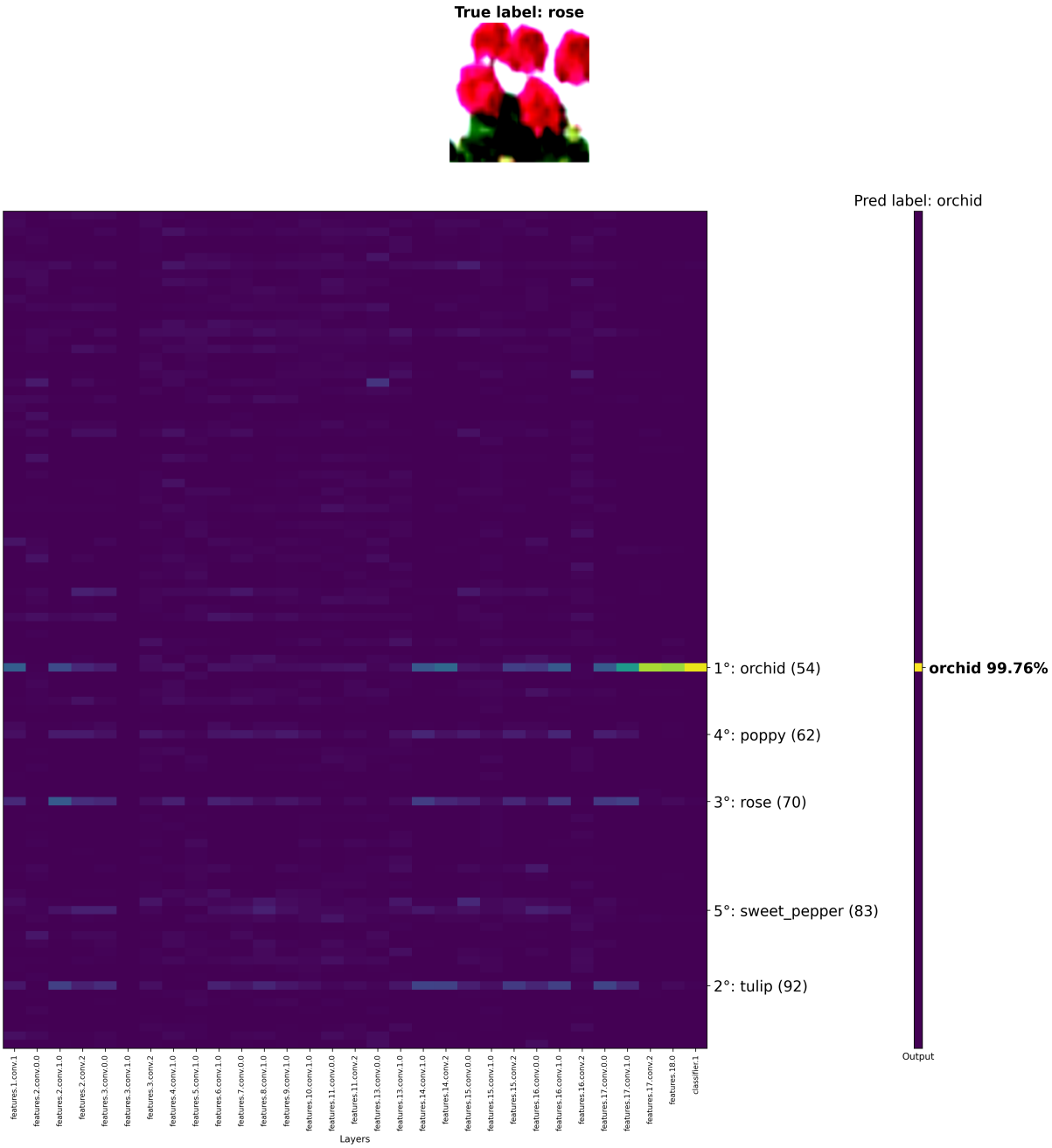

FIGURE 4.8. Conceptogram of a incorrectly predicted label of superclass *flowers*

- The conceptogram in Figure 4.9 (true label: *sunflower*, predicted label: *sunflower*) with only 27.81% confidence, reveals a faint and scattered activation profile. Notably, *sunflower* isn't in the top-1 classes with the strongest concept activations (1º: *lobster*), suggesting that the network is highly uncertain and confused in this decision. In this case, MobileNetV2 appears unreliable, as the internal representation offers little insight into how the model arrived at its prediction, making the decision making process untrustworthy.

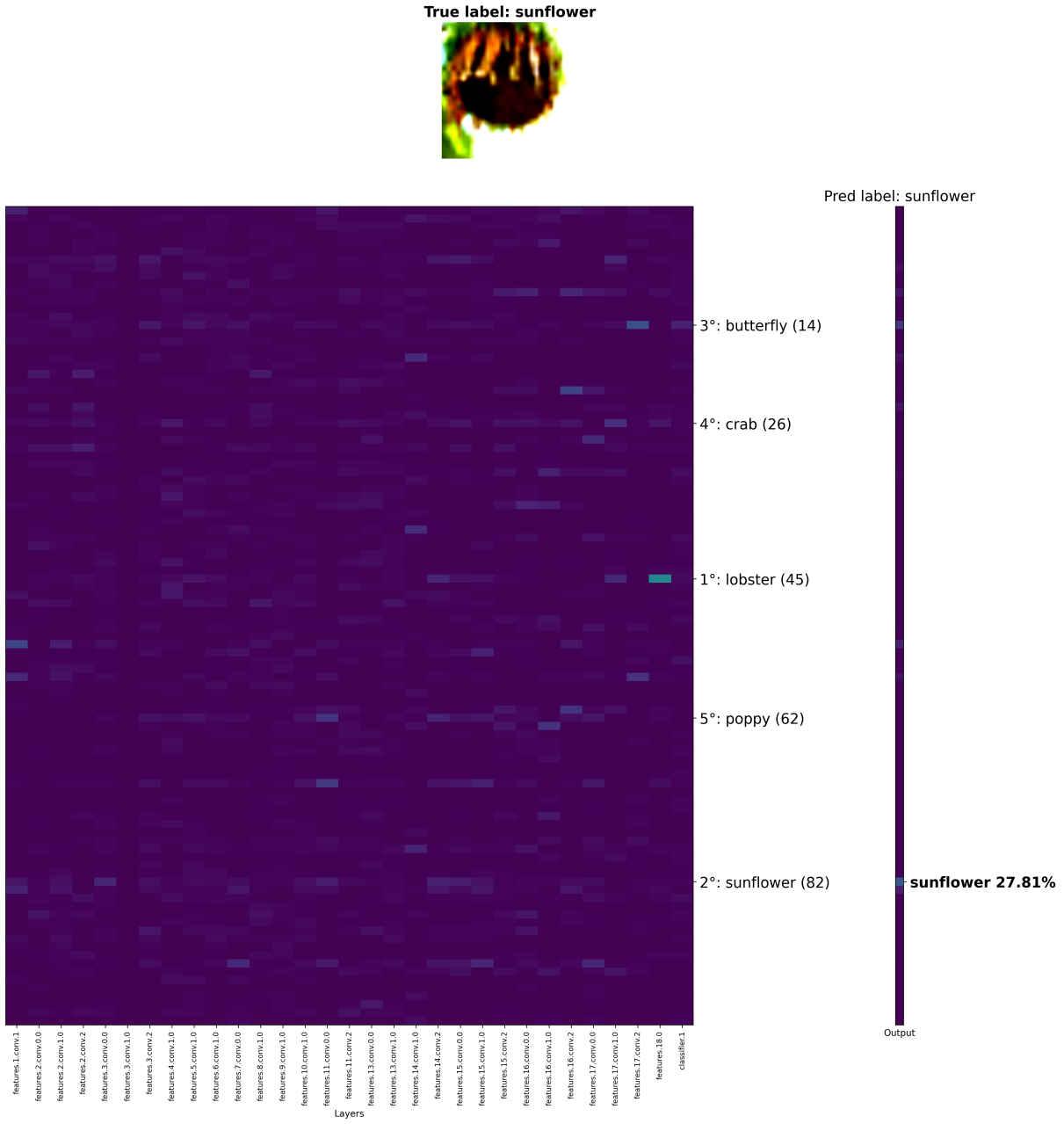

FIGURE 4.9. Conceptogram of a low-confidence prediction of superclass *flowers*

Visualizing the highest and lowest accuracy superclasses allowed us to observe how the network behaves under different conditions. These conceptograms revealed distinct patterns in how information is represented and clarified how internal activations shift across correct, incorrect, and uncertain predictions, highlighting useful behaviors that support deeper interpretability. But what if, instead of analyzing the network across all 100 fine-grained classes, we looked at how it behaves over a broader, more abstract level? In the next chapter, we introduce **superconceptograms**, based on the 20 superclasses of CIFAR-100, to explore how the network organizes higher-level semantic categories.

4.2. Superconceptogram

Building on the conceptogram analysis, we now extend our investigation to superconceptograms, which aggregate activations at the level of CIFAR-100's 20 superclasses. This broader view allows us to explore how the network organizes and represents more general, high-level categories, and whether meaningful patterns still emerge when class-level details are grouped.

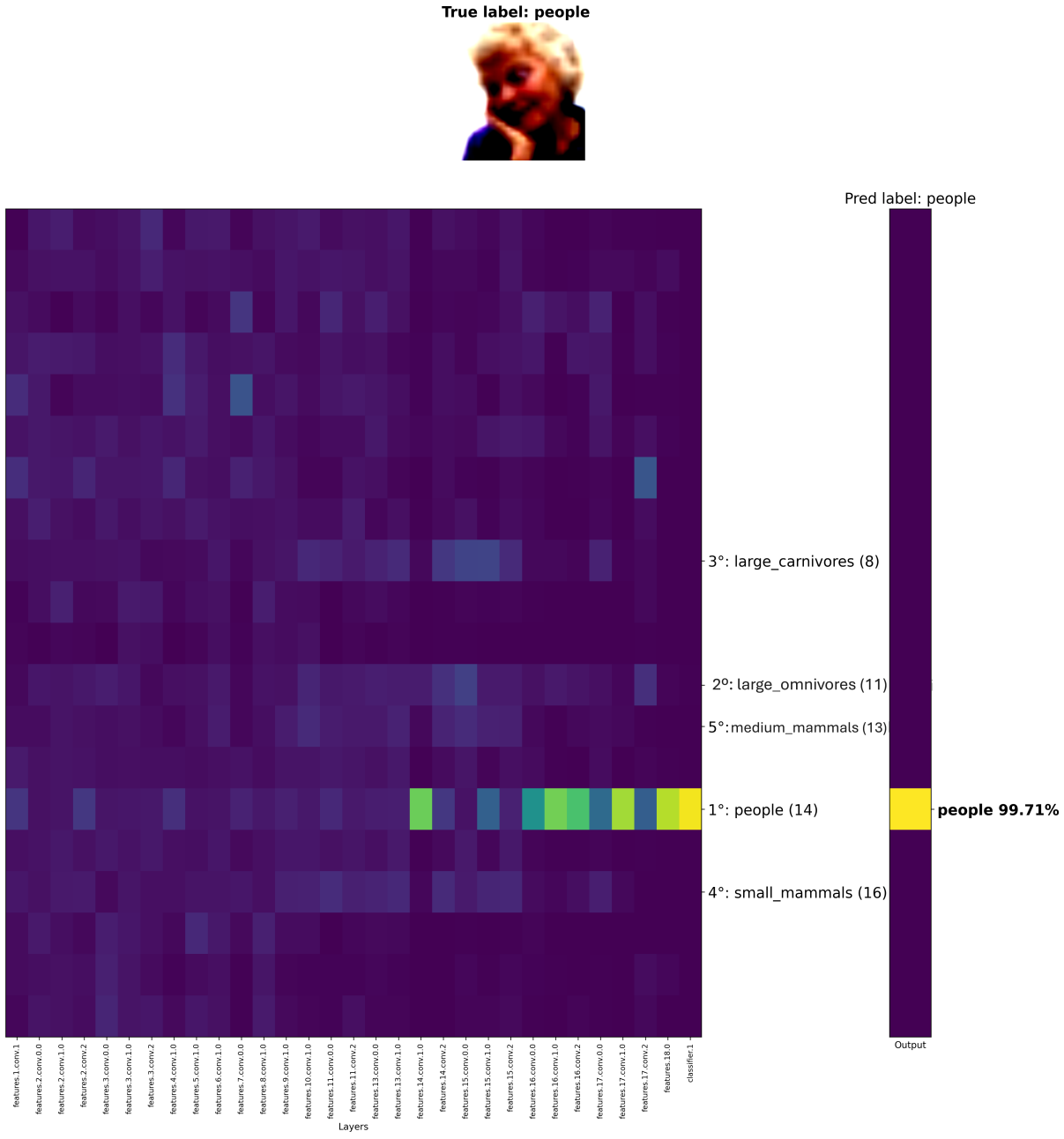

FIGURE 4.10. Superconceptogram of superclass *people* using 20 clusters

- The superconceptogram 4.10 corresponds to peepholes created with 20 superclasses rather than 100 fine-grained categories. This means that during the clustering stage, the number of clusters used to build the empirical posterior was set

to **20** rather than 100. Each cluster is therefore expected to capture broader, more general patterns that align with these superclass categories.
Here, the network assigns the image to *people* with an even higher confidence of 99.71%, compared with conceptogram C. Here we can see a single more distinguishable line in the concept *people*. However, the global internal structure of the activations is visibly more diffuse (more "colorful"). This blurring is likely a reflection of the reduced cluster granularity. Each cluster now represents a broader concept like *people* instead of *woman*, *man*, etc. These broader clusters tend to overlap with more patterns in the data, and as a result, more regions of the activation space may map to them.

The comparison reveals that reducing the number of clusters from 100 to 20 simplifies the concept structure captured by the network. With 100 clusters, the representations are sharper and more class-specific, enabling more refined understanding of the internal decision-making. However, with 20 superclasses, the network seems to generalize more confidently but at the expense of finer distinctions. This demonstrates a clear trade-off between granularity and abstraction in Peephole's interpretability pipeline: more clusters lead to finer semantic resolution, while fewer clusters yield coarser but broader concepts.

4.2.1. Conceptogram Analysis of High-Accuracy Class

In this experiment, we will compare the conceptograms 4.4, 4.5 and 4.6 with their respective superconceptograms and try to observe and interprete the difference of behaviour.

In the case of the palm tree, we observe a distinct difference in activation timing compared to the respective 100-classes conceptogram.

- In the conceptogram 4.4, the concept of *palm_ tree* becomes active relatively early, around `feature.13`. Since *palm _ tree* is a specific and narrow class, the network could be relying on distinctive mid-to-deep features to trigger that concept with a focused and early response. However, in the superconceptogram 4.11, where *palm_ tree* is grouped under the broader superclass *trees*, the activation occurs much later in the network, around `features.17`. In this case, the concept only becomes prominent in the final layers. This delay suggests that the network defers the decision to classify an image as a generic *tree* until deeper layers where more global and abstract features (like overall structure or silhouette) are aggregated. In other words, when the network is asked to recognize a specific object like a palm tree, it could make that decision earlier based on distinctive visual cues. But when asked to classify something as broadly defined as *tree*, it may require more contextual and complete information, which only becomes available in the deeper layers of the network.

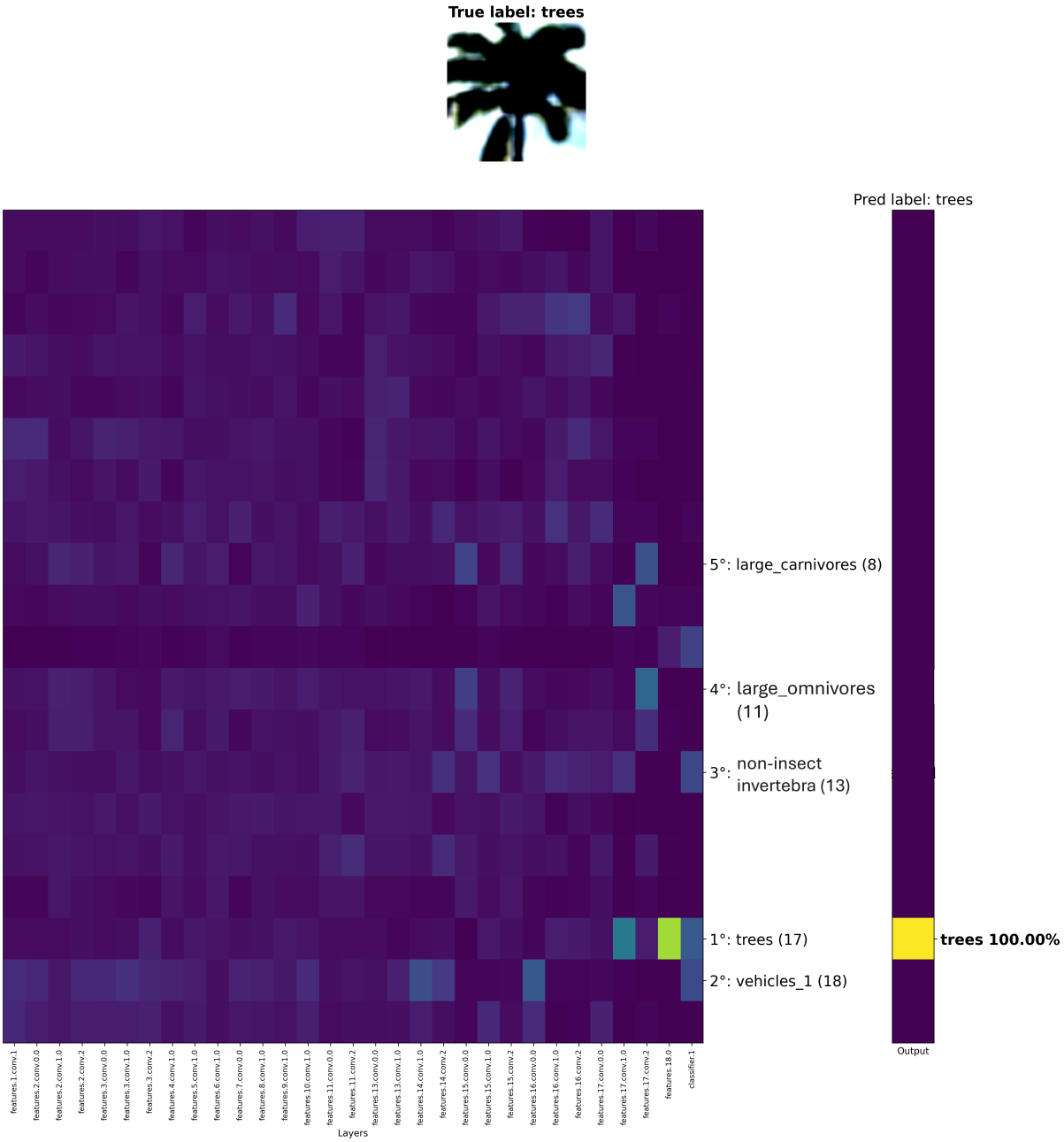

FIGURE 4.11. Superconceptogram of *palm_tree*

- In contrast, for superconceptogram 4.5, the model begins to associate the image with the *trees* superclass at the same time or even earlier in the network, compared with the respective conceptogram. This could be because oak trees have a more archetypal tree shape and overall silhouette that's more aligned with the model's general *tree* features learned during training. As a result, intermediate feature maps already begin aligning with the *trees* superclass before the final layers refine the prediction.

To summarize, the difference in recognition timing is likely due to oak trees resembling the statistical average of *trees* more than palm trees do.

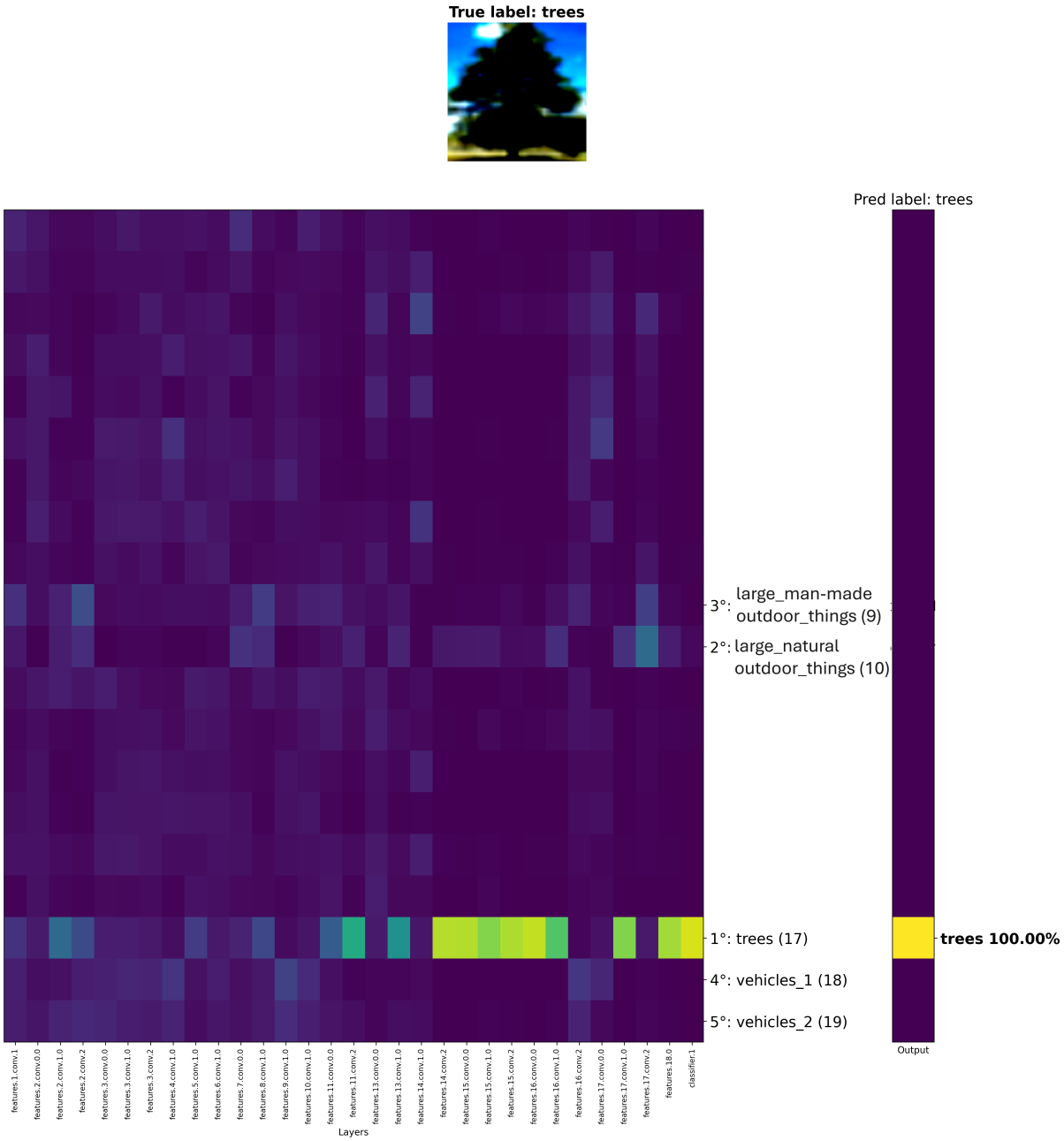

FIGURE 4.12. Superconceptogram of *oak_ tree*

The superconceptogram 4.13 exhibits a strong, coherent activation band across the later layers of the model, with a much higher confidence (75% compared to 48%). In the respective conceptogram, the model struggles to pinpoint subclass-specific patterns, and the heatmap's sparse highlights reflect uncertainty about which detailed characteristics distinguish a willow. In contrast, in the superconceptogram, the network's representation of *trees* remains clear and consistent, suggesting that the model has learned a broadly applicable, high-level concept of what constitutes a tree. This reveal that abstraction into superclasses not only raised model confidence but also helped produce more interpretable, stable representations; detailed subclasses, while potentially valuable for nuanced analysis,

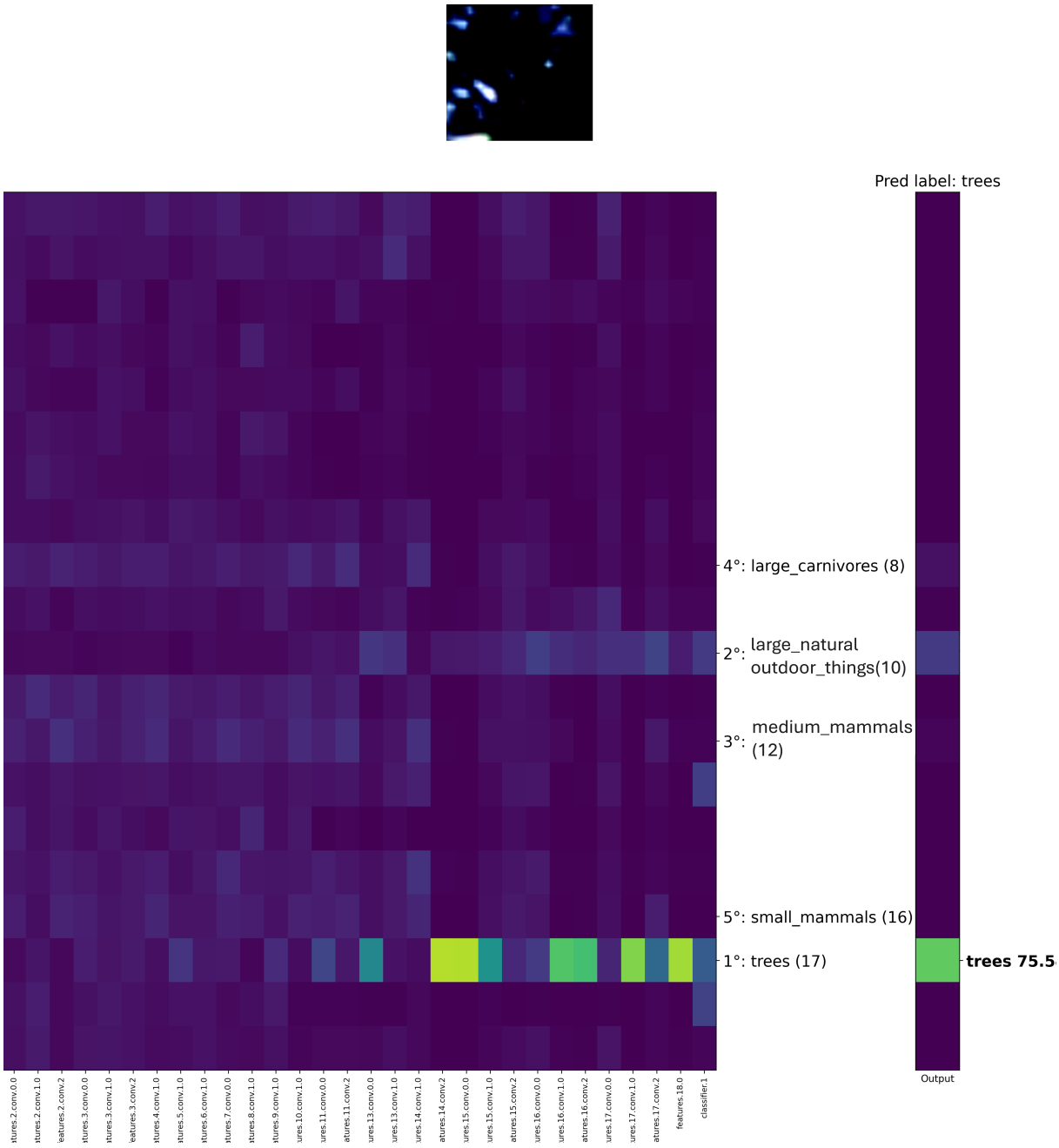

FIGURE 4.13. Superconceptogram of *willow_ tree*

may remain diffuse and under-confident unless the network develops sufficiently specialized feature detectors.

These different experiments contrast highlights how the level of class granularity and different visual features can affects both the timing and specificity of concept activation within the network.

Superconceptogram Analysis of Low-Accuracy Class

Once again, we will compare the conceptograms 4.7, 4.8 and 4.9 with their respective superconceptograms to observe and try interprete the difference of behaviour.

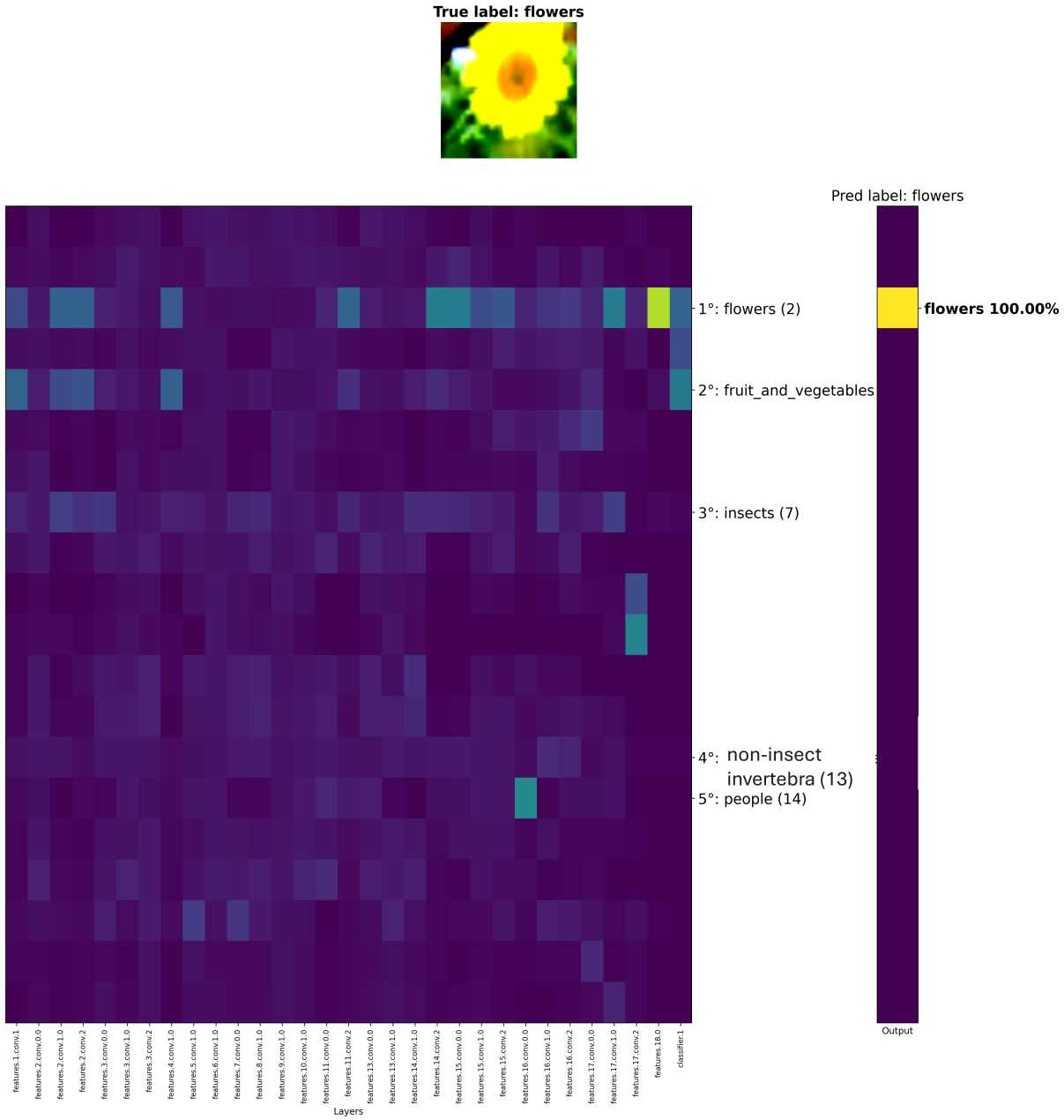

FIGURE 4.14. Superconceptogram of *sunflower*

- In superconceptogram 4.14, we can see a similar behaviour to the *palm_tree* case: Only the latest layers get really bright compared with the conceptogram 4.7 where the network brights up much earlier, around `features.14`. This can indicate that the network also requires more contextual and complete information to classify this image as *flowers*, which only becomes available in the deeper layers, where more global and abstract features are captured.
  Moreover, we can see that creates more colorful "confused" patches in several other classes since collapsing to 20 superclasses probably distributs the model's

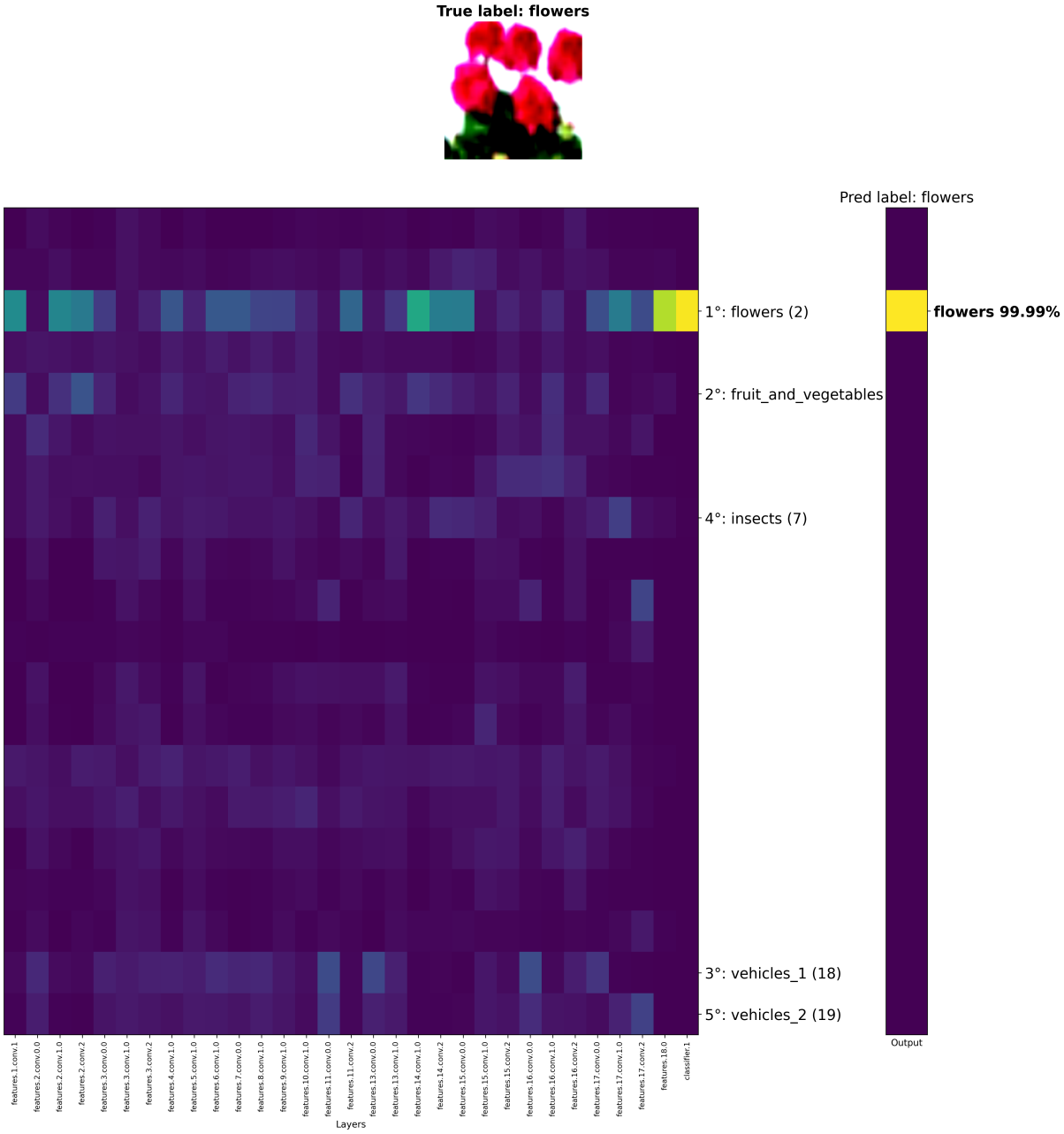

FIGURE 4.15. Superconceptogram of *rose*

attention among a few broad, overlapping concepts, as it was also observed previously in other cases. This could be the result of peepholes sharing its probability mass among fewer overlapping classes.

- In superconceptogram 4.15, the concept *flower* is catched in early-mid layers much more than in the corresponding conceptogram. This could be due to the same reason explained earlier: it relies on general visual features that are shared across many flower types. In the conceptogram 4.8, the network needs to extract precise and detailed patterns to confidently distinguish it from similar classes like *rose* or *poppy*.

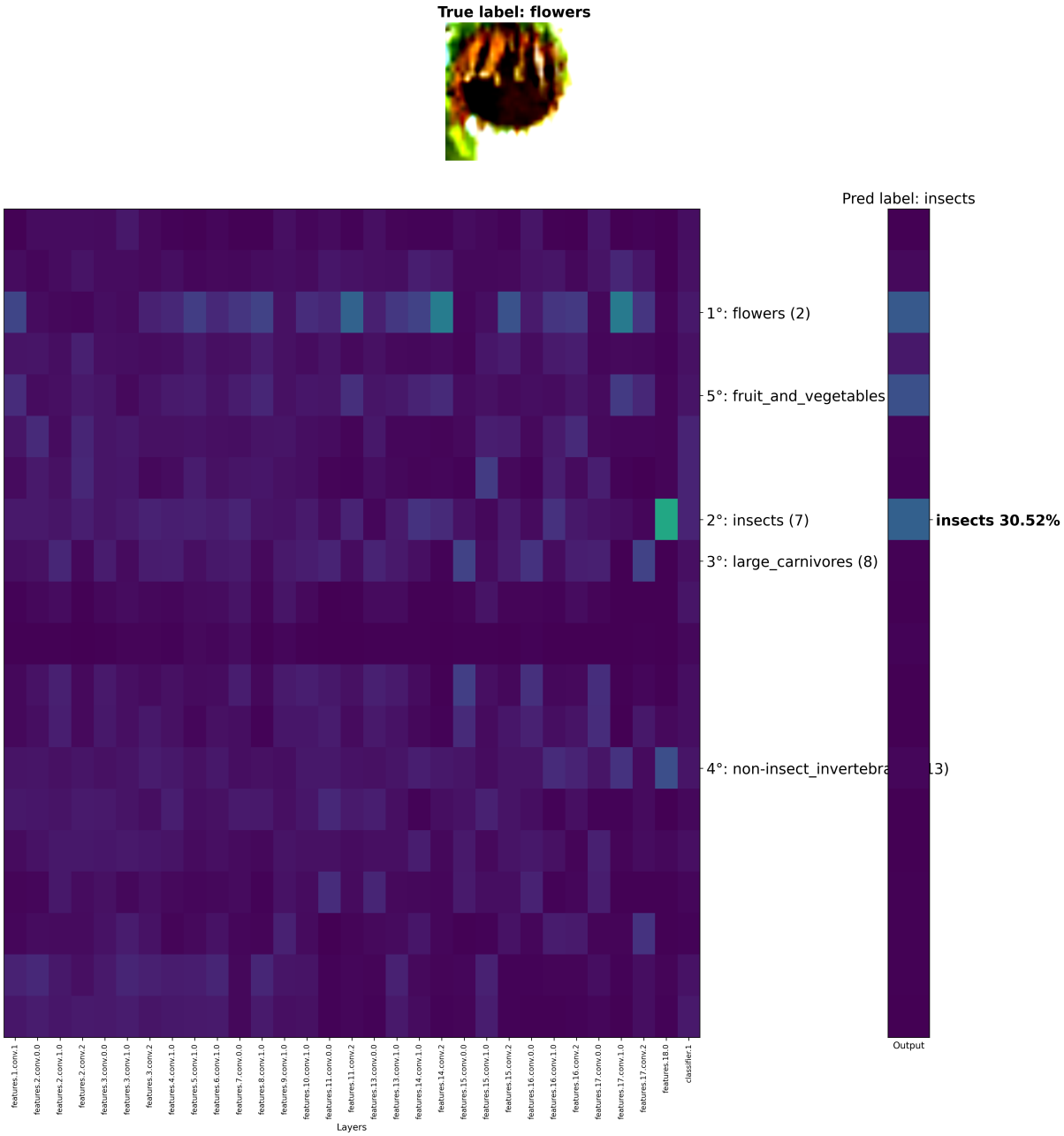

FIGURE 4.16. Superconceptogram of sunflower (low confidence case)

- In superconceptogram 4.16 we can easily observe a more distinct lighter line in row *flowers*. Even tho the network gets the prediction wrong in the end, flowers is the classes with the strongest concept activations, sugesting that the network is capturing much more clear concepts related to flower compared with the respective conceptogram (Figure 4.9). This is another example of how transitioning to 20 broader concepts could help the network express its internal reasoning more clearly and making it more robust. This reinforces the idea that superconcept-level analysis can provide a more forgiving and interpretable view of the model's

internal state, especially in cases where class-specific distinctions are subtle or easily confused.

This simplification from 100 classes to 20 superclasses led to more confident and clearer concept activations, suggesting that peepholes become easier to interpret when aligned with broader, more general categories.

To conclude the experimental results section, the conceptogram visualizations provided valuable insights into the behavior of the network across different scenarios. However, it's important to emphasize that these observations represent possible interpretations, not definitive explanations. In the field of explainable AI, there is an ongoing effort to understand why models behave the way they do, but achieving complete certainty remains out of reach. Explainability is ultimately about uncovering patterns, building intuition, and visualizing the internal workings of a model as clearly as possible, even if we can never fully explain every decision with absolute certainty. This is also why our thesis contributes not only to explainability but to identifying potential untrustworthiness in neural networks. When a model's internal reasoning cannot be clearly understood (when conceptograms fail to reveal how the network went from input to prediction) it raises legitimate concerns about the reliability of its decisions. In such cases, as illustrated in Conceptogram 4.9, the lack of interpretable structure makes it difficult to trust in the model's output. Therefore, conceptograms serve as a tool not just for interpreting neural behavior, but for assessing the trustworthiness of the model itself.

That said, it's important to recognize that the conceptograms shown in this chapter represent only a subset of the network's possible behaviors. Given the wide range of classes, samples, and layer interactions, it's not feasible to explore every individual case. The examples presented here should be understood as illustrative cases and they are not meant to generalize across all scenarios. As such, they offer a starting point for deeper investigation, not a complete map of the network's internal logic.

Conclusion and Future Work

4.3. Conclusion

This thesis set out to explore whether meaningful interpretability could be achieved in compact neural networks, using MobileNetV2 as a test case for the SVD-based Peephole method. We focused on two main research questions: *(1) Can we extract meaningful information from such a compressed architecture?*; *(2) How to choose the best layers to extract information from?*

In response to the first research question, our experiments showed that even in a lightweight model like MobileNetV2, it is possible to extract meaningful, interpretable insights into the network's internal decision-making process. By analyzing the layer-wise flow of class-related concepts, we could trace how abstract representations evolve throughout the network.

These layer-wise compressed informations, or peepholes, were visualized using a tool called *conceptograms*. Conceptograms proved effective in revealing how well each layer captured concepts linked to specific output classes. We found that diffuse, scattered activations were often associated with low-confidence or incorrect predictions, while focused and class-specific conceptograms tended to reflect high-confidence, correct predictions.

Further, we explored aggregating the 100 CIFAR-100 classes into 20 superclasses, creating what is referred as *superconceptograms*. This simplification led to more confident and clearer concept activations, suggesting that peepholes become easier to interpret when aligned with more general categories.

Ultimately, beyond interpretability, conceptograms offer a means of assessing trustworthiness. If a conceptogram fails to reveal any coherent structure (if no meaningful concept can be traced across layers) it signals that the model's decision process is effectively a black box for that input. This lack of transparency undermines trust in the model's output. In this way, conceptograms not only provide insight but serve as a diagnostic tool for identifying untrustworthy or opaque predictions, aligning directly with the goals of Explainable AI. This is particularly relevant for compliance with regulatory frameworks such as the AIA, which emphasizes the need for transparency and accountability in high-risk AI systems. Methods like conceptograms help provide the type of traceability and justification required by such legal standards.

To address the second question, we developed a strategy for layer selection based on the Singular values profile and the empirical posterior matrix, which measures how well each layer's activations correspond to class-relevant structures. This analysis showed that depthwise layers and layers near the classifier or early in the network tended to be the

most representative in MobileNet V2. This likely a product of their unique role in the Inverted Residual Block, compared to expansion and projection layers, which serve more structural functions.

4.4. Future Works

While this thesis demonstrates the usefulness of conceptograms for interpreting neural network behavior, several directions remain open for future exploration and improvement:

- The current evaluation relies heavily on visual inspection of individual conceptograms. While this offers valuable intuition, it is inherently subjective and does not scale well. A promising next step would be to develop **quantitative metrics for scoring or ranking conceptograms** based on properties such as focus, sparsity, class alignment, or consistency across samples. Such metrics would enable more rigorous comparisons across layers, architectures, or datasets.
- The current framework limits itself to class-based concepts, that is, concepts tied to output classes or superclasses. While this provides a structured way to interpret decision-making, it may overlook meaningful internal patterns that are **not necessarly aligned with classes**. Future work could explore expanding the analysis to include more abstract concepts which would allow for a richer and more flexible framework, potentially revealing new insights into what the network is really learning.
- Another important area for future research involves testing the robustness of conceptograms under adversarial conditions. It would be interesting to study how concept activations shift in the presence of **adversarial attacks**, and whether conceptograms can help detect or defend against it.

In summary, future work should aim to move from interpretation to evaluation, broaden the concept space beyond class labels, and investigate how this interpretability framework behaves under adversarial stress, ultimately bringing concept-based explanations closer to practical, trustworthy AI systems.

Bibliography

[1] Gabrielle Ras et al. *Explainable Deep Learning: A Field Guide for the Uninitiated*. 2021. arXiv: 2004.14545 [cs.LG]. URL: https://arxiv.org/abs/2004.14545.

[2] Amina Adadi and Mohammed Berrada. "Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)". In: *IEEE Access* 6 (2018), pp. 52138–52160. DOI: 10.1109/ACCESS.2018.2870052.

[3] Sajid Ali et al. "Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence". In: *Information Fusion* 99 (Nov. 2023). ISSN: 15662535. DOI: 10.1016/j.inffus.2023.101805.

[4] Gabriele Mazzini and Salvatore Scalzo. *The proposal for Artificial Intelligence Act: considerations around some key concepts (*)*. 2021. DOI: //dx.doi.org/10.2139/ssrn.4098809.

[5] Livia Manovi et al. "SVD-based Peephole and Clustering to Enhance Trustworthiness in DNN Classifiers". In: *2024 IEEE 6th International Conference on AI Circuits and Systems, AICAS 2024 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., 2024, pp. 129–133. ISBN: 9798350383638. DOI: 10.1109/AICAS59952.2024.10595919.

[6] Eleonora Poeta et al. *Concept-based Explainable Artificial Intelligence: A Survey*. Dec. 2023. URL: http://arxiv.org/abs/2312.12936.

[7] Michael E. Wall, Andreas Rechtsteiner, and Luis M. Rocha. *Singular Value Decomposition and Principal Component Analysis*. Ed. by Daniel P. Berrar, Werner Dubitzky, and Martin Granzow. Boston, MA, 2003. DOI: 10.1007/0-306-47815-3_5. URL: https://doi.org/10.1007/0-306-47815-3_5.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016. Chap. 6.

[9] Hanie Sedghi, Vineet Gupta, and Philip M. Long. *The Singular Values of Convolutional Layers*. 2019. arXiv: 1805.10408 [cs.LG]. URL: https://arxiv.org/abs/1805.10408.

[10] Brenda Praggastis et al. *The SVD of Convolutional Weights: A CNN Interpretability Framework*. 2022. arXiv: 2208.06894 [cs.CV]. URL: https://arxiv.org/abs/2208.06894.

[11] Leandro De et al. *Visual Transformer Conceptogram Generation Through Intermediate Activation Analysis*. 2023. URL: https://amslaurea.unibo.it/id/eprint/34909/.

[12] Yu-Han Liu and Sercan Ö. Arik. "Explaining Deep Neural Networks using Unsupervised Clustering". In: *CoRR* abs/2007.07477 (2020). arXiv: 2007.07477. URL: https://arxiv.org/abs/2007.07477.

[13] Hung Nguyen et al. "Evaluation of Explainable Artificial Intelligence: SHAP, LIME, and CAM". In: *Proceedings of the 2021 International Conference on Explainable AI*. May 2021. URL: https://www.researchgate.net/publication/362165633_Evaluation_of_Explainable_Artificial_Intelligence_SHAP_LIME_and_CAM.

[14] Wojciech Samek and Klaus Robert Müller. "Towards Explainable Artificial Intelligence". In: vol. 11700 LNCS. Springer Verlag, 2019, pp. 5–22. DOI: 10.1007/978-3-030-28954-6_1.

[15] Grégoire Montavon et al. "Layer-Wise Relevance Propagation: An Overview". In: vol. 11700 LNCS. Springer Verlag, 2019, pp. 193–209. DOI: 10.1007/978-3-030-28954-6_10.

[16] Zeon Trevor Fernando, Jaspreet Singh, and Avishek Anand. "A study on the Interpretability of Neural Retrieval Models using DeepSHAP". In: *CoRR* abs/1907.06484 (2019). arXiv: 1907.06484. URL: http://arxiv.org/abs/1907.06484.

[17] Junbing Li et al. "Deep-LIFT: Deep Label-Specific Feature Learning for Image Annotation". In: *IEEE Transactions on Cybernetics* 52 (8 Aug. 2022), pp. 7732–7741. ISSN: 21682275. DOI: 10.1109/TCYB.2021.3049630.

[18] Ramprasaath R Selvaraju et al. *Grad-CAM: Why did you say that?* 2017. arXiv: 1611.07450 [stat.ML]. URL: https://arxiv.org/abs/1611.07450.

[19] David Bau et al. "Network Dissection: Quantifying Interpretability of Deep Visual Representations". In: *CoRR* abs/1704.05796 (2017). arXiv: 1704.05796. URL: http://arxiv.org/abs/1704.05796.

[20] Amirata Ghorbani et al. *Towards Automatic Concept-based Explanations*. Oct. 2019. URL: http://arxiv.org/abs/1902.03129.

[21] Ruihan Zhang et al. *Invertible Concept-based Explanations for CNN Models with Non-negative Concept Activation Vectors*. 2021. URL: www.aaai.org.

[22] Chih-Kuan Yeh et al. *On Completeness-aware Concept-Based Explanations in Deep Neural Networks*. 2022. arXiv: 1910.07969 [cs.LG]. URL: https://arxiv.org/abs/1910.07969.

[23] Johanna Vielhaben, Stefan Blücher, and Nils Strodthoff. *Multi-dimensional concept discovery (MCD): A unifying framework with completeness guarantees*. June 2023. URL: http://arxiv.org/abs/2301.11911.

[24] Tobias Leemann et al. "When are post-hoc conceptual explanations identifiable?" In: *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*. Ed. by Robin J. Evans and Ilya Shpitser. Vol. 216. Proceedings of Machine Learning Research. PMLR, 2023, pp. 1207–1218. URL: https://proceedings.mlr.press/v216/leemann23a.html.

[25] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014). URL: https://arxiv.org/abs/1409.0473.

[26] Feyza Duman Keles, Pruthuvi Mahesakya Wijewardena, and Chinmay Hegde. *On The Computational Complexity of Self-Attention*. 2022. arXiv: 2209.04881 [cs.LG]. URL: https://arxiv.org/abs/2209.04881.

[27] Yulin Wang et al. *Computation-efficient Deep Learning for Computer Vision: A Survey*. 2023. arXiv: 2308.13998 [cs.CV]. URL: https://arxiv.org/abs/2308.13998.

[28] Mark Sandler et al. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. 2019. arXiv: 1801.04381 [cs.CV]. URL: https://arxiv.org/abs/1801.04381.

[29] Ching Hao Wang et al. "Lightweight Deep Learning: An Overview". In: *IEEE Consumer Electronics Magazine* 13.4 (2024), pp. 51–64. DOI: 10.1109/MCE.2022.3181759.

[30] François Chollet. *Xception: Deep Learning with Depthwise Separable Convolutions*. 2017. arXiv: 1610.02357 [cs.CV]. URL: https://arxiv.org/abs/1610.02357.

[31] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG]. URL: https://arxiv.org/abs/1502.03167.

[32] Shanchen Pang et al. "An artificial intelligent diagnostic system on mobile Android terminals for cholelithiasis by lightweight convolutional neural network". In: *PLOS ONE* 14 (Sept. 2019), e0221720. DOI: 10.1371/journal.pone.0221720.

[33] Perry Gibson et al. *Optimizing Grouped Convolutions on Edge Devices*. 2020. arXiv: 2006.09791 [cs.LG]. URL: https://arxiv.org/abs/2006.09791.

[34] Maithra Raghu et al. *On the Expressive Power of Deep Neural Networks*. June 2017. URL: http://arxiv.org/abs/1606.05336.

[35] Jason Yosinski et al. *How transferable are features in deep neural networks?* 2014. arXiv: 1411.1792 [cs.LG]. URL: https://arxiv.org/abs/1411.1792.

[36] Guillaume Alain and Yoshua Bengio. *Understanding intermediate layers using linear classifier probes*. 2018. arXiv: 1610.01644 [stat.ML]. URL: https://arxiv.org/abs/1610.01644.

APPENDIX A

Mobilenet Layer Organization

For simplicity the following tables demonstrate an organized view of every layer of Mobilenet v2:

Bottleneck	**Repeats**	**Inverted Residual Block name**
B1	1	`features.1`
B2	2	`features.2, features.3`
B3	3	`features.4, features.5, features.6`
B4	4	`features.7, features.8, features.9, features.10`
B5	3	`features.11, features.12, features.13`
B6	3	`features.14, features.15, features.16`
B7	1	`features.17`
conv	-	`features.18.0`
fc	-	`classifier.1`

TABLE A.1. **Inverted Residual Block Configuration**: as in Figure 3.5, Mobilenet has 7 bottlenecks (some of them are repteated several times), a final convolutional layer and a fully conected layer.

Layer Name	**Layer Type**	**Role**
`features.X.conv.0.0`	Conv2d (1×1)	Expansion Conv
`features.X.conv.1.0`	Conv2d (3×3)	Depthwise Conv
`features.X.conv.2`	Conv2d (1×1)	Projection Conv

TABLE A.2. Inverted residual block: composed by an expansion layer, a depth wise layer and a projection layer. For example: `features.13.conv.1.0` is a depth wise layer in bottleneck 5.