Seeing Through Models' Eyes: Decomposing Latent Representations of Convolutional Networks

Bartosz Jerzy Miselis

A Thesis
in
The Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Masters (Computer Science)
Concordia University
Montréal, Québec, Canada

September 2020

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Bartosz Jerzy Miselis**

Entitled: **Seeing Through Models' Eyes: Decomposing Latent Representations of Convolutional Networks**

and submitted in partial fulfillment of the requirements for the degree of

Masters (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Examiner
Dr. Andrew Delong

_____ Examiner
Dr. Sudhir P. Mudur

_____ Supervisor
Dr. Thomas Fevens

_____ Supervisor
Dr. Adam Krzyżak

Approved _____
Dr. Lata Narayanan, Chair of Department

August 26, 2020 _____
Dr. Mourad Debbabi,
Interim Dean, Gina Cody School of Engineering and Computer Science

Abstract

Seeing Through Models' Eyes: Decomposing Latent Representations of Convolutional Networks

Bartosz Jerzy Miselis

"To comprehend inner data representations constructed by convolutional networks" is an ambitious task attracting a constantly growing number of researchers across the globe. The joint effort has led to the broad space of interpretability techniques that roughly fall into two groups: attribution and feature visualization. The former shines the light on "where" the algorithm looks, while the latter reveals "what" the network sees. Combining them into rich interfaces provides new ways to understand convolutional networks and develop deeper intuitions about their complex behavior.

In this thesis, two novel techniques are proposed to advance the analyses in the field of interpretability: Latent Factor Attribution (LFA) and Distilled Class Factors Atlas. LFA identifies distinct concepts in the activation tensor using matrix decomposition and estimates their influence on the classification result. Distilled Class Factors Atlas then aggregates these concepts and presents them in an interactive, exploratory interface that makes it possible to see the entire class of images through the model's eyes by leveraging feature visualization. Both techniques introduced in this thesis extend the holistic view on latent representations of convolutional networks by enabling the collective perspective on the patterns that recur in the activation tensors.

Acknowledgments

This research would not be possible without the tremendous help of several people, that not only supported me mentally throughout the time I was preparing the thesis but also provided the invaluable resources and advice, without which I could not complete the dissertation.

First and foremost, I would like to thank my parents for raising me to become a person I am—open-minded, curious, and devoted to the things I care about. While being in Canada I realized that without their selfless devotion and thousands of hours spent on teaching me about the importance of being a good person, I would never end up being in a place I am right now.

I am truly grateful for having the possibility to spend my life with a wonderful woman who is not only patient and supportive but also exceptionally intelligent. Thank you for all the advice you provided me with. Without your help to organize my work and motivate me when the pool of determination was temporarily empty, I would have not progressed as much as I did.

I would also like to express my gratitude to my supervisors: Dr. Thomas Fevens and Dr. Adam Krzyżak. Your support, both in terms of resources and knowledgeable advice, was the foundation I could build my research ideas on.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

By the Middle Ages, humanity tended to believe that everything that could be discovered had mostly been already brought to light. The scientific revolution of the 1500s, though, undermined the omnipresent confidence by introducing the concept of *blank spots* that began to emerge on redrawn maps, rewritten manuscripts, and in the curious minds of future pioneers.

Surprisingly, certitude is not only the domain of the distant past. In July 1966, Seymour Papert, professor at MIT, set up a summer project to construct "a significant part of a visual system" [51]. The project was assigned to a student Gerald Sussman who was asked to coordinate a group of 10 students to, as it is often phrased, "solve computer vision". It is the year 2020, and researchers still have not found the universal key to all the computer vision tasks that are a trifle to an average homo sapiens individual. The only matter that appears to be certain for now is that humbleness is an essential element of a scientific puzzle.

The 2000s and 2010s have seen a rapid growth of deep learning [32]—a subfield of the broader discipline of machine learning—that became the *de facto* approach to complex, abstract tasks like natural language processing [10], playing games [56], or image processing [63], to name the few. Many junior and senior scientists relish exploring algorithms so effective that they appear to magically solve all the computational challenges humanity has encountered for the last 50 years, not infrequently surpassing human performance (!). As our species have always been characterized by unparalleled curiosity, it is natural at this point to ask the question: how do deep learning algorithms work, and why are they so effective?

An entire field of *interpretability* research emerged to address the aforementioned inquiry. However, as there exist tens or hundreds of challenges being tackled by deep learning, and the number of specific techniques is even larger, it is not feasible to describe everything in a single thesis[1].

[1] In fact, it would not be possible to exhaust that topic in a substantial tome.

Thus, this thesis focuses on the explainability of a single family of algorithms—convolutional neural networks (CNNs)—that was first introduced by Kunihiko Fukushima in 1980 [21], significantly developed and improved by Yann LeCun in the 1990s [33] and brought to the state of wider applicability by Alex Krizhevsky *et al.* in 2012 [30].

The concept of CNNs is elegant and [theoretically] straightforward. Nonetheless, the reality showed that to design models that perform well without having hundreds of millions of parameters, the task needs to be taken away from human researchers and rather tackled by yet another machine learning model (!). Starting from the year 2018, Neural Architecture Search (NAS) [71] became increasingly popular. The technique led to the automatic creation of CNNs that outperformed previous human designs by a significant margin. The understanding debt, to call it such, grew even larger—inner representations of simpler models were already hard to comprehend before NAS.

The author of this thesis strongly believes that it is the right time to start paying off the understanding debt. Doing so will not only provide the answers to the important questions, but will also broaden the depth of our understanding of homo sapiens' visual cortex. As the language gave a specific structure to human thoughts (proving its usefulness throughout the millennia), carefully crafted interpretability interfaces can alter the way the mind thinks about CNNs.

1.1 Contributions of the Thesis

To define the contributions of the thesis, a nomenclature from [49] will be used. The reason for this is the following: *The Building Blocks of Interpretability* (the cited article) is an indisputable inspiration for the main idea introduced in this manuscript. Moreover, it is an exceptional resource that illustrates the value of immersive interfaces that enable researchers to interact with visualizations and, most importantly, provide new ways of thinking about the inner representations constructed by CNNs. The contributions of this manuscript are as follows:

1. Latent Factor Attribution (LFA): new attribution technique that can be used by researchers working on CNN interpretability as a building block in complex visualization interfaces. LFA helps to understand the influence of high-level concepts on the image classification result (for instance, how concepts similar to "mountain" or "lava" (when visualized) impact the CNN classifying the image as "volcano").

2. Activation Explorer, an online web interface that puts the LFA in a proper context, showcasing its usability in concrete scenarios.

3. Distilled Class Factors Atlas, an extension of the Class Activation Atlas [12] technique, based on matrix decomposition (to extract important abstractions from the images) and the LFA (to quantify the importance of the extracted concepts).

4. An online web interface to conveniently interact with the generated atlases.

1.2 Thesis Outline

The thesis consists of five chapters. Chapter 1 describes a brief history of the field of interpretability. Chapter 2 provides background for the techniques used further in the manuscript, establishing the relationships with main related work. Chapter 3 explains the novel Latent Factor Attribution technique, from the motivation, through the implementation details, to the discussion and limitations. Chapter 4 elaborates on the proposed Distilled Class Factors Atlas interface, providing examples of how the ideas introduced in this dissertation can be applied to develop new ways of thinking about the inner data representations constructed by CNNs. Chapter 5 summarizes the entirety of the thesis, presenting a few ideas for future work.

Chapter 2

Background and Related Work

Striving to understand machine learning models is an old concern, not strictly related to neural networks themselves. It has been an area of active research for a very long time—feature importance analysis and input-dependent sensitivity analysis have been used in the field for many years now, both aiming to provide human insight into how the trained model works. In [9], Breiman comments on the interpretability of a few standard methods (decision trees, random forests, SVMs and neural networks), discussing the differences between the two cultures in the use of statistical modeling—either assuming that the data comes from a stochastic model (known data mechanism), or using the algorithmic models instead (unknown data mechanism). Modern deep learning techniques are, in the majority of the cases, the algorithmic models—they achieve superb performance, but solve problems in a way that is not easy to explain using stochastic modeling.

The field of CNN interpretability has been continuously evolving from the year of 2009 when Erhan *et al.* introduced the idea called *activation maximization* [17]. Since then, the multitude of explainability techniques has been gradually uncovering the world of neural networks' latent representations, shedding first light inside the mysterious "black boxes".

This chapter provides a comprehensive overview of the existing CNNs interpretability approaches, with major attention put on its two core branches—feature visualization and attribution. The former focuses on finding ways to effectively visualize features that highly activate defined units of interpretability. The latter, on the other hand, tries to answer the following question: *which parts of the input tensor contributed the most to the algorithm's prediction*? Detailed analyses described further in this chapter cover the vast majority of work done in the field, highlighting the [slightly] underestimated value of the process of combining multiple different building blocks. Hybrid interfaces, e.g. the ones presented in [12, 49], bring remarkable value by putting things into the human scale and enable meaningful explanations to complex hypotheses.

2.1 Holistic Understanding is What Science Needs

Since the AlexNet paper [30], CNNs have become wider [67], deeper [24], or both [63], gradually finding richer, more complex data representations. Nowadays, progress led to a point where the algorithms' behavior can be comprehended by humans only to a very small extent. While for the synthetic problems like MNIST [33] or ImageNet [16] it may not be crucial to provide the exhaustive explanations of the algorithm's predictions, the situation differs dramatically in the fields like finance or healthcare. Here, no one would object that the system deciding whether to grant a loan or make a diagnosis should be deployed only when it is fully understood by its designers. Regulatory bodies seem to have already realized how urgent it is for the AI-based systems to be interpretable. The European Union General Data Protection Regulation (GDPR) is a good example of such. It grants each EU citizen a right to having the decision made by intelligent systems explained, especially if it significantly influences a person's life. Consequently, the algorithms are required to provide human-interpretable outputs. Otherwise, it would be illegal to deploy them in the real world.

The ability of convolutional networks to construct highly abstract representations from lower-level features like curves or color splashes has been vastly unexplored over the years, most of the time leading to superficial comparisons with the human visual cortex. In the short term, this may not sound like the issue, but eventually, the field of computer vision will need the tools that provide a comprehensive interpretation of what happens inside the "black box". Without the right toolbox, researchers will always lack proper intuitions behind algorithms' behavior. However, the skeptics of complete understanding may argue that superficial explanations are more than enough to use deep learning models in practice, the same way as linear approximations are sufficient to control complex dynamic systems. It is not a goal of this thesis to provide a counterargument to that logic. Instead, let the following question be raised:

> **Is not science all about a thorough understanding of the nature of things, with deep intuitions supporting further progress in the field?**

As with many debates, the answer awaits in a reasonable common ground that will make it possible to proceed with future research. The pursuit of the answers has already begun with an initiative called the *Circuits* project [11]. Researchers from the OpenAI Clarity team, together with volunteers from all over the world, are scrutinizing the neurons of CNNs one by one, layer by layer, identifying the ways they interact and influence one another. Their discoveries have already expanded our understanding of CNNs and appear to be a move in the right direction finally.

2.2 Core Data Representation: Activation Tensor

Activation tensor is a core data representation that is extensively referenced across multiple sections of this thesis. In its original form [49], it corresponds to the output of a single layer l of a CNN. To phrase it differently, it is the input tensor $\mathbf{I} \in \mathbb{R}^{h_0 \times w_0 \times d_0}$, transformed by a series of functions $f(\cdot)$ that mapped it into the activation tensor $\mathbf{H}^{(l)} \in \mathbb{R}^{h_l \times w_l \times d_l}$ of layer l. Each tensor belongs to the space $\mathbb{R}^{h_i \times w_i \times d_i}$ (height × width × depth), where $i \in \{0, 1, 2, ..., l, ..., L\}$. Index "0" corresponds to the input layer, l denotes an arbitrary layer, and L is the total number of layers in the model.

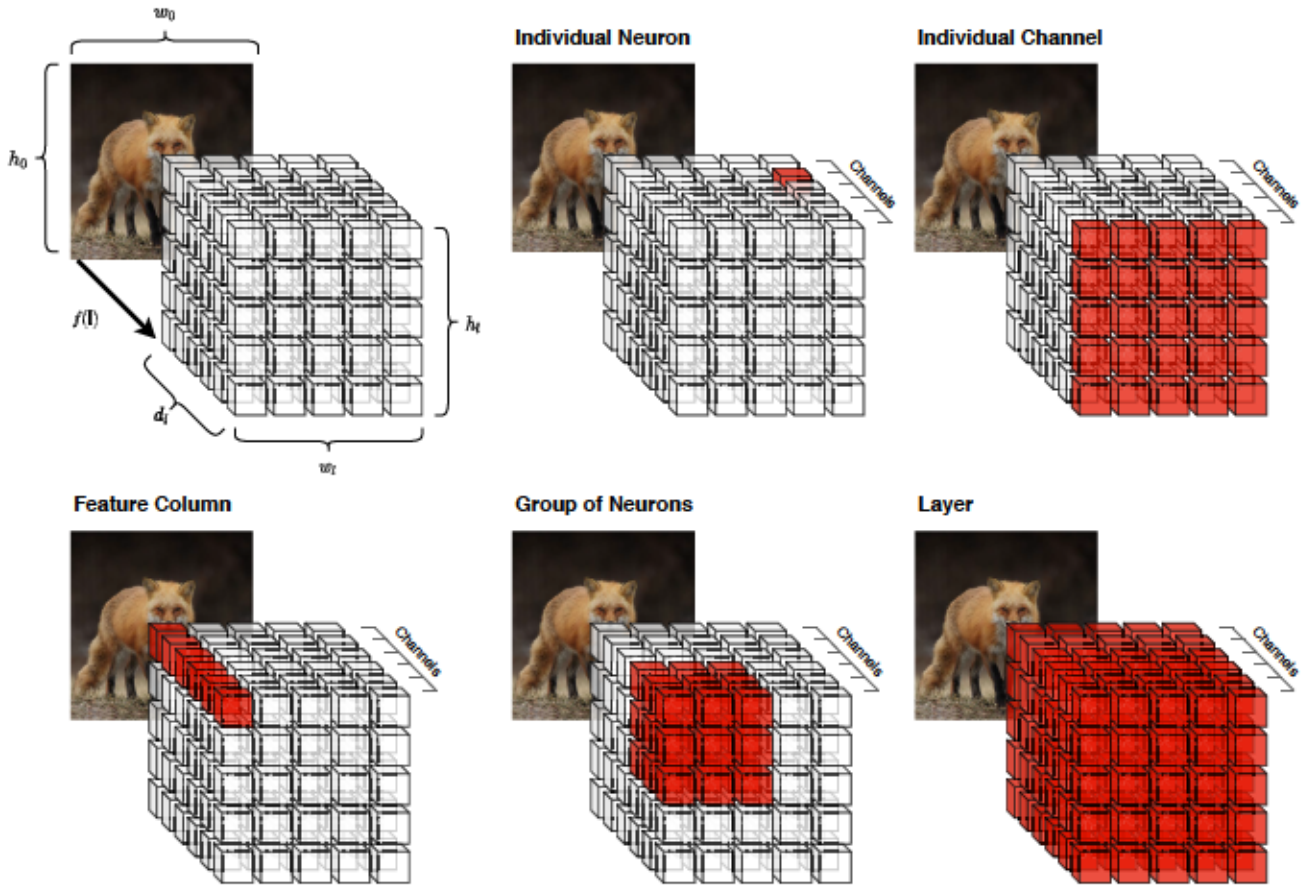


Figure 1: Activation tensor $\mathbf{H}^{(l)}$ of layer l (large white cube) consists of the activations of individual neurons (small white cubes). The figure contains five examples that represent different slicing approaches. They were provided to give a complete view of the basic units of interpretability. The red color in each example indicates the unit of interpretability that is the focus of the given method. Adapted from [49].

To express it in mathematical notation:

$$f(\mathbf{I}) = \mathbf{H}^{(l)}, \quad f : \mathbb{R}^{h_0 \times w_0 \times d_0} \to \mathbb{R}^{h_l \times w_l \times d_l}. \tag{1}$$

The second part of the eq. (1) is the core of CNN computation: it is basically a mapping from one tensor into another. Activation tensor contains a huge amount of information that, when viewed from the appropriate perspective, provides insights into the inner representation of the CNN.

To experiment with the aforementioned perspective, an idea of activation tensor *slicing* has been introduced [49]. More specifically, by extracting arbitrary substructures from the original tensor (e.g. individual neuron, isolated channel, feature column, group of neurons), it is possible to obtain distinct views on the underlying data representation. Every perspective sheds a different light on the processes that occur during computation. Please refer to Figure 1 for further details.

2.3 Attribution

Figure 2: Sample input image with a corresponding heatmap explanation. **Left:** input image (object of class "red fox"). **Right:** attribution map. The brighter the region, the higher its influence on the classification result. Green and red areas denote positive (increase a probability of "red fox" class) and negative (decrease a probability of "red fox" class) influence, respectively.

The following quote from [48] briefly (yet very precisely) describes the essence of attribution:

> *"Attribution studies what part of an [input] example is responsible for the network activating [in] a particular way."*

The current section serves as an overview of the existing interpretability approaches that focus on the signal/relevance flowing through the model. The vast majority of attribution techniques yield heatmaps (often referred to as *saliency maps*). Please refer to Figure 2 for a concrete example of such. The goal of these maps is to visualize the influence of single spatial locations in the activation tensor on the final classification score. The helper question to be answered here is as follows: what would happen to the final classification score if a particular spatial location in the activation tensor responded more/less to the input stimulus?

Kindersman *et al.* [29] separated explanation approaches (technically speaking, what they call the *explanation* is called the *attribution* in this thesis) into three distinct groups: *functions*, *signal approximators*, and *signal decomposers*. In this work, however, based on their views, the aforementioned groups were simplified into three distinct types of explanations that should be easier to comprehend: input-space, feature-space, and non-heatmap explanations.

2.3.1 Input-space Explanations

Input-space explanations analyze how moving along the given direction in the input space affects the classification result. To phrase it differently, input-space explanations show (in the form of a heatmap) how altering individual pixels (brightening/darkening/changing color) changes CNNs' output. Such modifications are, in fact, movements in the input image space, as the input image is essentially a 3D tensor. Modifying some of its values comes down to shifting the tensor slightly towards a particular direction. The shift direction is not random (though it can be if that is what is being studied). Instead, it is calculated using one of a plethora of available methods. The most interesting ones were briefly summarized below.

2.3.1.1 Gradient Family

The first group of input-space explanations is the gradient family. These techniques use a gradient of the output (usually the output neuron associated with the highest predicted class) w.r.t. the input image pixels as the core source of the attribution information. The very first approach of this kind, based on a pure **gradient**, was initially called a saliency map [17, 57]. It illustrates how directly altering input image pixels would affect CNN's predictions. The idea of saliency maps is

elegant and quite powerful, yet in practice, it often struggles with a large amount of noise present in the resulting explanations.

To mitigate the above problem, **SmoothGrad (SG)** [58] and **VarGrad (VG)** [1] methods were proposed. Both SG and VG strongly reduce the noise in the output by computing the saliency map for the given input multiple times and then averaging them (SG) or finding their variance (VG) to obtain a single, smoothened explanation. However, the input is not the same in all iterations—each image has the noise sampled from a normal distribution added to the original pixels. SG and VG work very well in practice and can be combined with other gradient-based methods to significantly improve their quality.

Shrikumar *et al.* [55] were the first ones to propose an idea called **gradient ⊙ input**—an extension of the pure gradient explanation that is essentially the element-wise product of the input image with the computed gradients. The goal of this technique is to reduce visual diffusion and increase the overall sharpness of the saliency maps.

Many researchers believe that the idea of averaging gradients is very powerful. While some methods are based on injecting the noise into the input (as the aforementioned SmoothGrad and VarGrad approaches), other ones alter the exposure of the original tensor from extremely low (completely black input), up to its original, neutral value. Each time the exposure is varied, the gradient of the output w.r.t. the input is computed. Next, all of the obtained saliency maps are averaged to yield the final heatmap. Such a technique is called **Integrated Gradients (IG)** [60].

2.3.1.2 Input Perturbations

Contrary to the gradient family, input perturbations treat the algorithm as a black box system—the only systematically altered factor is the input image itself, nothing else. Each time a change is introduced to the original pixel values, the algorithm's output is recorded to detect any deviations from the baseline. A result of a perturbation method is the mask highlighting the regions that, when covered, yield differences in the prediction. Two perturbation techniques that are most often used are:

- Occluding parts of the image with solid color patches [68].
- Distorting original intensities by introducing strong noise and/or blurring the image [19, 20].

An interesting insight has been provided in [19]—the authors showed that by reformulating the standard perturbation objective it is possible to identify channels in the intermediate layer of a CNN that are salient for the classification of a given image.

2.3.1.3 Relevance Propagation

As defined in the original work on the topic [6], *relevance* is a pixel-wise contribution to the prediction. By introducing a common mathematical notation, Ancona *et al.* [3] showed that **Layer-wise Relevance Propagation (LRP)** [6], together with the ϵ propagation rule—a method called ϵ-LRP—is equivalent to the gradient $\odot$ input, provided that the gradient update rule at each computational node is modified (see [3] for details). The relevance is distributed across the entire network: from the output neuron that activated the most, up to the input itself using a backpropagation variant. In [40], the ϵ-LRP was extended into the **deep Taylor decomposition** approach that decomposes neuron's activation into contributions from its inputs by using a first-order Taylor expansion around x_0, an arbitrarily chosen root point. Yet another technique called **PatternAttribution** [29] solves the problem of choosing x_0 by directly learning it from the data.

2.3.2 Feature-space Explanations

Feature-space explanations, contrary to the input-space ones, do not explicitly show the attribution of individual pixels. Instead, they find more abstract entities—neurons—that activate significantly (either supporting or inhibiting a given class). Neurons' activations are projected back onto the input image space (pixels) to highlight the salient regions that influence the classification score the most. A good example of a feature-space attribution technique can be seen in Figure 2, where the features (neurons activations) from one of the hidden layers of the CNN were overlaid with the input image to indicate the regions supporting (green) or inhibiting (red) the *red fox* class. In this subsection, several most important feature-space explanations were briefly summarized.

2.3.2.1 Deconvolutional Network

Deconvolutional network (Deconvnet) [68] maps chosen neuron's activation "[...] back to the input pixel space, showing what input pattern originally caused a given activation in the feature maps" [68]. The aforementioned *mapping* is exactly the reason why Deconvnet should be seen as a feature-space explanation: it is a projection of a feature map into the input-space, not the tensor from the input-space itself. The procedure to obtain the explanation map is the following:

1. Feed the input image to a CNN to obtain the activation tensor at the chosen layer.
2. Keep only the activations of a single feature map (channel), zeroing out all the other channels of the selected layer's activation tensor.

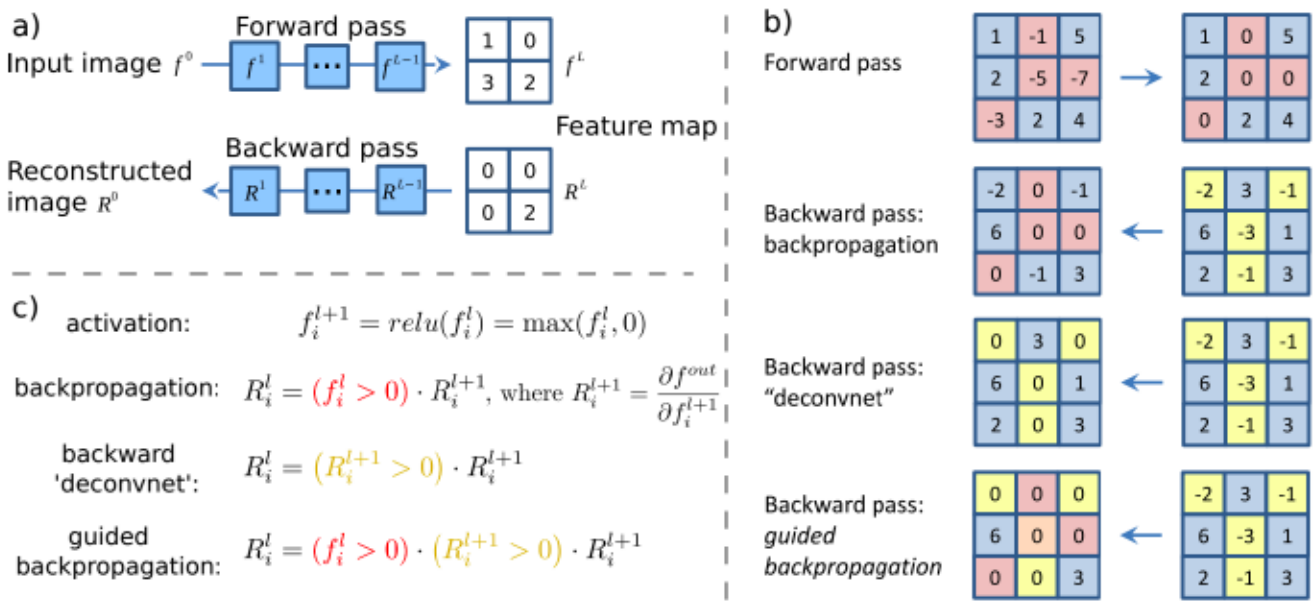


Figure 3: Comparison of the original backpropagation rule with the ones of two feature-space explanations: Deconvnet [68] and guided backpropagation [59]. a) Illustration of the process of zeroing out all but one channel from a feature map and using it for the reconstruction. b) Various backpropagation rules. c) Backpropagation rules from b) formalized. Figure retrieved from [59].

3. Invert (approximately) the computational graph from the layer of interest back to the input image to get an input reconstruction, this time a "compressed" one (because of keeping only the channel of interest in the target layer).

In Figure 3 one can observe the differences between the standard backpropagation rule and the ones of Deconvnet and guided backpropagation (described in the next subsection). Even though these may appear to vary subtly, the explanations they produce can be very different visually and change the explanation from "non-comprehensible" to "interpretable".

2.3.2.2 Guided Backpropagation

Guided backpropagation [59] builds on the idea of Deconvnet [68] by slightly modifying its backward pass: it extends it by adding additional *guidance* signal that unifies the standard backpropagation (all the locations that had negative gradient value in the backpropagation pass are set to zero) and Deconvnet-style backpropagation (all the locations that had a negative unit response in the forward pass are set to zero). Refer to Figure 3 for further details and a numerical example.

2.3.2.3 PatternNet

PatternNet [29] is the equivalent of the gradient computation, yet "during the backward pass the weights of the network are replaced by the informative directions" [29]. These "informative directions" can be obtained by estimating the signal component of the data (overall, data is assumed to contain two components: signal and distractor; the signal is the only component that is correlated with the label). To obtain the final attribution map, the estimated signal is projected back onto the input space—pixels.

2.3.2.4 Class Activation Mapping

Class Activation Mapping (CAM) is an entire family of approaches that vary slightly from one another by merging the original CAM idea [69] with the methods like SmoothGrad [58] or guided backpropagation [59]. This family of techniques was designed specifically for CNNs and, contrary to Deconvnet and guided backpropagation, is class-discriminative, being able to highlight the regions that contribute the most to the specific class (either arbitrary or the one with the highest predicted value). The following quote from [69] provides an intuition on what CAM is:

> *"The class activation map is simply a weighted linear sum of the presence of these [convolutional filters' activations] [...] at different spatial locations. By simply upsampling the class activation map to the size of the input image, we can identify the image regions most relevant to the particular category."*

Grad-CAM [54] generalizes original CAM, enabling the application of this method for a wider set of CNNs (not only fully-convolutional ones) without the need to alter the model architecture. It achieves this by finding a new way to combine feature maps from convolutional filters using the gradient signal. **Grad-CAM++** [13] extends the idea of Grad-CAM, focusing on improving the results in scenarios with multiple object instances or poor object localization. To mitigate these issues, the authors introduced a pixel-wise weighting of the gradients from the output w.r.t. the concrete spatial position in the final convolutional layer. **Smooth Grad-CAM++** [50] adds SmoothGrad [58] to the original Grad-CAM++, further improving the obtained results.

2.3.2.5 Interpretable Basis Decomposition

Another way to think about the attribution is not only to use a CAM-like approach but also to try to understand the underlying "subparts" that contribute to the prediction of a specific class. In [70], authors designed a custom matrix decomposition called Interpretable Basis Decomposition

(IBD)—a hybrid of Non-negative Matrix Factorization (NMF) [34] with positive latent spatial factors and Principal Component Analysis (PCA) [52] with gradual creation of the decomposition basis. The downside of IBD is that the introduced decomposition is tightly bound with the Grad-CAM heatmap computation—it was designed to decompose the activation tensor of the final convolutional layer of the architecture of interest. Hundreds of separate logistic regression classifiers were trained, each to detect a specific concept feature column. The term *concept* is defined here as a learned combination of channels in a single feature column. The aforementioned custom decomposition approach does not perform arbitrary factorization. Instead, it tries to match new feature columns to the concepts learned by the aforementioned linear classifiers and assign the weights to each of them, while minimizing the *residual concept* (the amount of activation tensor that is not covered by any known concepts). The contribution of each concept is the attribution of the concept to the final classification result.

2.3.3 Non-heatmap Explanations

When making an overview of the attribution techniques, one should not forget about these that do not yield visual results that could be displayed as a heatmap. Instead, they may provide numerical measures to quantify the importance of some predefined abstractions in the classification task. Below, the most interesting method has been presented because of its uniqueness in the field.

2.3.3.1 Conceptual Sensitivity

Testing with Concept Activation Vectors (TCAV) [28] was proposed to quantify the concept's importance for a chosen class in each layer of a CNN. Concept activation vector (CAV) is defined as the vector normal to the decision boundary separating target activation tensors (concepts) from the "rest" (all computation takes place in the space of the activation tensors). In [28], the term *concept* means "the entire activation tensor", contrary to the definition from [19], where it denotes "a single feature column" (class-specific combination of channels). In the activation tensor space, CAV points towards a cluster of similar abstractions—either it being "striped", "dotted" or "zigzagged"[1]. Learning a set of CAVs allows the later assessment of conceptual sensitivity for the unseen activation tensors: a directional derivative in the direction of every known CAV is computed to quantify the influence of learned concepts on the classification output.

[1]It is not straightforward to decide on what should be called a concept. While it is relatively easy to agree that textures/colors are good candidates for such, it is not that clear for more abstract terms like the entire objects.

2.3.4 Are Heatmaps Reliable?

Since 2009 [17], the field of interpretability has seen a significant increase in the number of attribution methods available. As these explanations began to be more widespread in real-world settings (e.g. for the interpretation of the lung cancer detection system [4]), they need to be trustworthy. But are saliency maps reliable? This exact question has been asked and preliminarily answered by Adebayo *et al.* in [2]. Some of their results can be observed in Figure 2 of [2]. The conclusions are somewhat appalling—multiple widespread techniques (especially guided backpropagation and guided GradCAM) **cannot** be trusted as they are agnostic to weights and labels randomization[2]. It indicates that such explanations should be used with caution. At the very least, new attribution methods should be rigorously tested using weights randomization to ensure sensitivity to the changes in weights. Such heatmaps could be seen as trustworthy and safe to use.

2.4 Feature Visualization

The following quote from [48] briefly (yet accurately) describes the topic of this section:

> *"Feature Visualization answers questions about what a network—or parts of the network—are looking for by generating examples."*

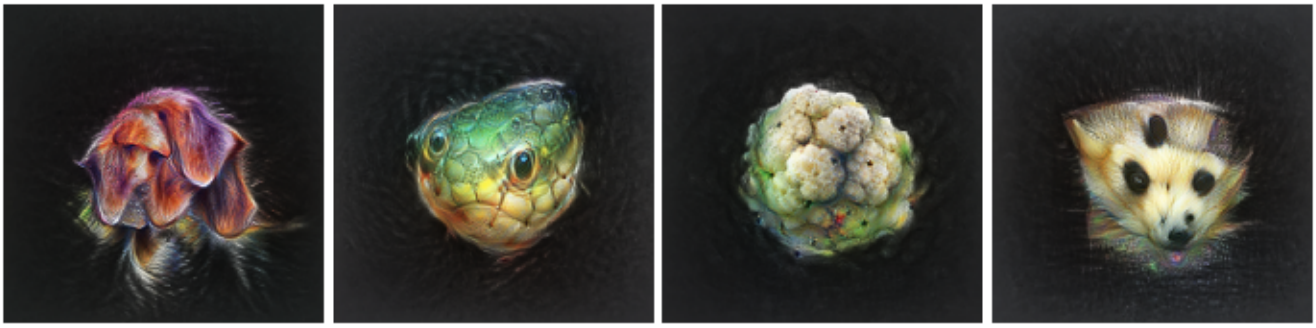


Figure 4: Examples of feature visualization.

Over the years, a branch of the interpretability research that focuses on trying to understand what selected parts of a CNN learn to recognize put a lot of effort into exploring the approach called *activation maximization*, originally introduced by Erhan *et al.* in [17]. The idea is, using gradient ascent, to optimize for the input pattern that activates the chosen unit of interpretability

[2]Heatmaps are very similar even if the model's weights are randomized or the training is conducted using random labels. The expected behavior is that the more the weights are randomized, the less meaningful the heatmap is.

the most (from individual neurons, through single channels, up to the feature columns and the entire groups of neurons). Feature visualization can complement other techniques like investigating dataset examples to understand a given unit of interpretability. A good example was presented in [31], where the activation maximization was used to generate synthetic stimuli and ensure a neuron suspected to detect human faces indeed responded to such concept the most.

Major attention in the field has been put into maximizing the activation of neurons from the final layer of a CNN, corresponding directly to the classes the network learns to classify (see [72] for simple examples). The reason for this particular interest is that by maximizing the activation of a class-specific neuron, the generated image depicts the ideal[3] input that the algorithm would assign to a given category. Generally, this is seen as helpful—researchers often desire to understand what it means for the image to be a "healthy tissue" or a "nodule" from the CNN's perspective, to give an example from a medical domain.

Feature visualization by optimization enables powerful and in-depth exploration of CNNs. However, several aspects need to be taken care of to make it work properly. Some may wonder why to put the effort to search for synthetic stimuli when dataset examples are at hand—they could be searched over to find images or image patches that highly activate a given neuron, thus providing a sufficient explanation. There are two strong arguments for feature visualization:

1. **Flexibility:** with activation maximization, even though it makes the problem much more complex, it is possible to carry out the analyses that would not be possible with just the images from the training data. For instance, experimenting with the interactions between neurons (optimizing jointly for multiple units at the same time) or performing diversity study (random starts or slightly varying image priors lead to uncovering multiple facets of a single neuron) shines entirely new light on what the CNN may have learned to detect [46, 48].

2. **Clarity:** when studying dataset examples that strongly stimulate a given unit of interpretability, it is often challenging to come down to a single explanation for all the images supporting the underlying abstraction. By leveraging the potential of feature visualization, one can look at the problem from additional perspectives. The idea is not necessarily to replace dataset examples but rather augment the available interpretability toolkit. By doing so, one can formulate the explanations that are not only coherent but also reliable.

[3] From a subjective point of view of a CNN.

2.4.1 Regularization

Regularization is an inherent element of every activation maximization task. A wide variety of techniques that guide the optimization towards desired manifold regions[4] have been explored in the past, gradually producing results of higher and higher quality. Feature visualization evolved from the unregularized optimization [17] to the work of Nguyen *et al.* [43] based on generative models priors that yield visually compelling outputs. However, one needs to be aware of the trade-off between the strength of the regularization and the reliability of the synthesized image— strong priors help generate sublime visualizations almost by default, yet may distort the real nature of the latent representations of a CNN and not reflect what the model truly responds to. On one extreme, if no regularization is applied, visualizations (in some sense "pure") often resolve to adversarial examples [62]. On the other end, inducing strong prior by searching over patches extracted from the dataset examples [59, 68] produces photorealistic results, but is only a very rough approximation of the underlying representation that responds to a given stimulus.

A comprehensive review of the core concepts of regularizing feature visualization—frequency penalization, transformation robustness, and learned prior—has been presented in [48] and was briefly summarized in the following subsections.

2.4.1.1 Frequency Penalization

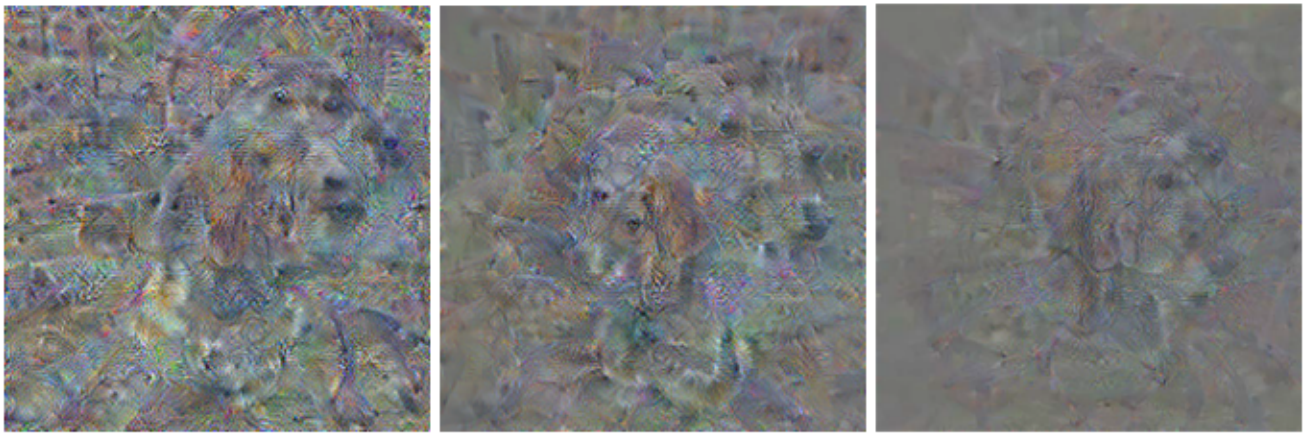


Figure 5: Regularization by frequency penalization using total variance technique. **Left:** unregularized feature visualization of a single neuron. **Middle:** moderate total variance regularization (visible decrease in the high-frequency noise). **Right:** strong total variance regularization (high-frequency noise almost removed from the visualization result).

[4]Regularization not only guides but also restricts the optimizer from entering certain areas of the space.

Unregularized optimization suffers from prominent high-frequency noise. A corner case of such patterns is very similar to the idea of adversarial examples [62]—the optimization reaches an easy, completely unintuitive solution that highly activates a given unit of interpretability (e.g. an individual neuron). To mitigate this issue, multiple frequency penalization methods have been proposed. One of them, called **total variance** [37], directly penalizes the variance between the pixels that are close to one another. Figure 5 contains an example of how such noise can be reduced using the total variance technique. As can be seen in the Figure, the stronger the regularization, the smoother the resulting visualization. It is also possible to regularize high-frequency noise by blurring the image at each optimization step with a Gaussian kernel [45] or a bilateral filter [64].

The aforementioned methods address high-frequency patterns directly in the generated feature visualization. It is, however, possible to mitigate the problem by regularizing the gradients flowing through the model instead. The idea to transform the gradient is called **preconditioning** in the field of optimization and has been deeply explored in several studies [41, 72]. Olah *et al.* [48] introduced a regularization technique that operates in decorrelated color space by performing gradient ascent in a Fourier basis, removing the majority of high-frequency patterns. Performing the activation maximization task in such a basis does not change the local minima. Instead, it alters the steepest directions and reshapes basins of attractions, thus helping the optimization procedure converge more quickly.

2.4.1.2 Transformation Robustness

Figure 6: Regularization using transformation robustness technique. **Left:** no regularization. **Middle:** moderate transformation robustness. **Right:** strong transformation robustness.

The idea of transformation robustness is that before every step of the optimization, the current state of the visualization is randomly scaled, rotated, or cropped. This is supposed to help find the input patterns that not only strongly activate the chosen unit of interpretability but also do so even after applying small geometrical transformations on them—hence the use of the term *transformation robustness*. Figure 6 illustrates the desired effect the transformation robustness has on the quality of the resulting visualization—the stronger the regularization, the higher the quality. On a side note, the transformation robustness was first introduced in the DeepDream project [41], where it helped to generate abstract, high-quality patterns that were not observed in the feature visualization subfield before. Without transformation robustness, it would not be possible to obtain such crisp visualizations.

2.4.1.3 Learned Prior

The natural extension of the aforementioned regularization techniques is to learn the model that captures the real data distribution and enforce it in the feature visualization procedure. It is tempting to assume that the more precisely that model reflects the training data, the better. However, this is not necessarily the case—well-fitted model often produces more realistic visualizations, yet it is no longer clear what part of the result is the true response to the stimuli and what part comes from the learned prior itself. It is important to remember that there exists a trade-off between the reliability of the results and the strength of the prior.

Wei *et al.* [65] introduced a way to regularize the color distribution of the visualization by building the database of patches from training images and minimizing the distance between the patches of the optimization output and those in the database. This can be seen as an approximation of the natural image prior that is often captured by generative models.

Nguyen *et al.* [44] took a different approach and produced realistic visualizations by performing gradient ascent in the latent space of a generator network to produce patterns maximizing the activations of selected neurons of a separate classifier. The extension of their work [43] used additional latent code prior, achieving state-of-the-art both in single sample quality and diversity.

2.4.2 Diversity Study

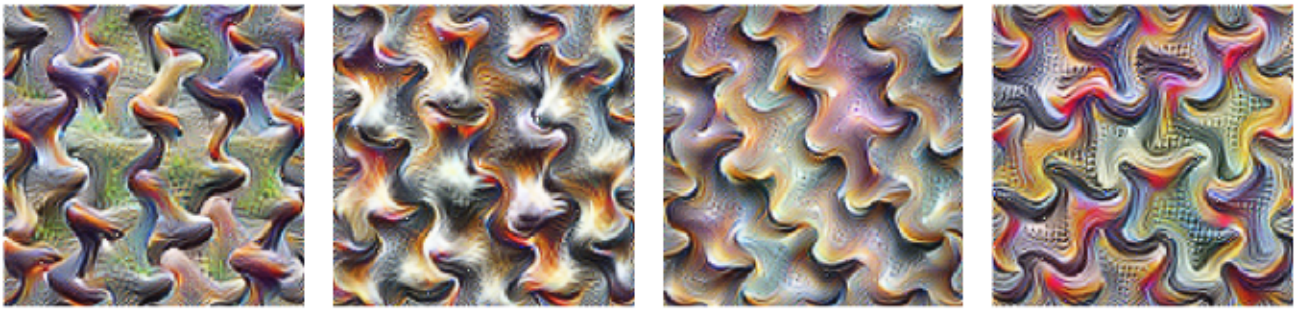


Figure 7: Example of a diversity study. The unit of interpretability under investigation: InceptionV1 model, the output of the mixed4a module, channel 97.

When considering the process of generating input patterns that highly activate a given unit of interpretability, the following question often arises:

> **Is the resulting image a full picture of what has been learned by an individual unit or is it only part of a bigger picture?**

The answer is that a standard feature visualization approach reveals only a single facet of the latent representation. The most straightforward way to mitigate this problem would be to use dataset examples instead of feature visualization. However, as mentioned before, patches from the training data are hard to comprehend when trying to extract common characteristics from, leaving the researcher with a significant grain of uncertainty. Ideally, the feature visualization technique would be used but in a way that would return multiple facets, not just a single one.

Early work by Wei *et al.* [65] explored intra-class variation inside a CNN by 1) noticing that the network can capture two kinds of variations—location and content—and 2) leveraging this insight to alter the optimization process and thus extract multiple facets of the visualized unit.

Nguyen *et al.* [46] approached the problem from a different angle and analyzed how to use dataset examples directly as the optimization priors (concrete image serves as the initialization for the entire procedure) and, as a result, achieve multifaceted feature visualization. Their contributions uncovered various patterns that are used by CNNs to represent classes of interest. Further research [43] explored how class visualization can be used with generative models to yield visually compelling, yet potentially misleading results (see section 2.4.1.3 for details).

Olah *et al.* [48] introduced a much simpler way of revealing multiple facets of a given unit of interpretability: the "diversity term" was to the optimization objective to make the generated

input patterns as diverse as possible. Their idea is very similar to the style transfer [23] and is based on penalizing the cosine similarity between correlation matrices of the activations at a given layer across all the images in a batch. And example of feature visualization with the "diversity term" can be seen in Figure 7.

2.4.3 How to Generate Compelling Visualizations?

If one's interest is to generate high-quality visualizations, it is vital to implement a very concrete toolkit to achieve that goal. This section aims to provide more details on the set of techniques that were used to generate all the feature visualizations in core parts of this thesis—chapter 3 and chapter 4. Table 1 contains a summary of the parameters that are considered essential to reproduce the results presented in this work. The following sections provide further descriptions of some of them.

Table 1: Essential feature visualization parameters used in this work.

Model	InceptionV1 pre-trained on ImageNet
Input value range	[-117.0, 138.0]
Dying ReLU prevention	Pre-ReLU tensors
Transparency	Yes
Optimization steps	3000
Optimizer	Adam with 0.01 learning rate
Parametrization	Fourier basis, $128 \times 128 \times 4$ tensor
Transform robustness	Yes (details in the text)
Objective	Neuron direction with transparency

2.4.3.1 Pre-ReLU Tensors

When optimizing for a given unit of interpretability, one needs to provide the tensor that will be used by the framework (e.g. TensorFlow) in the activation maximization procedure. If that tensor is directly an output of a ReLU activation function, there is a risk that the preceding operation (ReLU) will "die"—no matter the input values, it will always output 0.

The de facto approach to mitigate this problem (if there is no option to change the layer to leaky ReLU [35] or ELU [15]) is to override the gradient computation of the ReLU operation so that it does not get stuck. However, it was not possible to do that in TensorFlow 2.2[5] at the

[5]The framework used to implement the experiments in the thesis.

time of preparing this thesis. Instead, the idea from lucid[6] library to use pre-ReLU tensors in the optimization was followed. Pre-ReLU tensor is the tensor that is the input to the ReLU tensor. Using it for the activation maximization not only prevents the generated image from converging to an all-zero matrix but also [subjectively] improves the quality of the resulting input pattern when compared to the standard feature visualization.

2.4.3.2 Objective

The objective used for feature visualization is a modified variant of the one from [42] that encourages the content to centralize in the middle of the generated image. Let $\mathbf{h}$ denote vector of shape (1, 1, channels) that is a slice of the activation tensor $\mathbf{H}_{\text{height}/2,\text{width}/2,:}$. Let $\mathbf{f}$ denote a vector of the same shape that represents a target linear combination of channels to compare the $\mathbf{h}$ with. It is possible to define the so-called *neuron direction objective* (*neuron* because the $\mathbf{h}$ and $\mathbf{f}$ vectors are compared only at a single spatial location, central by default) as follows:

$$\text{obj}(\mathbf{h}, \mathbf{f}) = \mathbf{h} \cdot \mathbf{f} \cdot \max(0.1, \frac{\mathbf{h} \cdot \mathbf{f}}{\|\mathbf{h}\| \|\mathbf{f}\|}). \tag{2}$$

In this case, a dot product $\mathbf{h} \cdot \mathbf{f}$ yields a scalar value. Note that to avoid optimizing for the negative direction (both dot product and cosine similarity would be negative in this case) and to prevent the process from getting stuck if vectors $\mathbf{h}$ and/or $\mathbf{f}$ were zero, a maximum of 0.1 and a cosine similarity is taken.

2.4.3.3 Transparency

Figure 8: Influence of the transparency factors on the optimization objective. **Left:** standard objective. **Middle:** simple transparency. **Right:** transparency with the "concentrated content".

[6] https://github.com/tensorflow/lucid

This section describes how the transparency with the "concentrated content" assumption was realized. Let I denote the current state of the image that is being optimized in the feature visualization procedure. Its alpha channel can be expressed as I_α. In order to encourage some transparency, one can modify the objective from eq. (2) as follows [42]:

$$\text{obj}_{\text{new}} = \text{obj} \cdot (1 - \text{mean}(I_\alpha)). \tag{3}$$

This would help the transparency develop properly, yet does not encourage the content to be concentrated (middle visualization in Figure 8). For this, an objective should be further extended:

$$\text{obj}_{\text{final}} = \text{obj} \cdot [0.5 \cdot (1 - \text{mean}(I_\alpha)) + (1 - \text{mean}(\tilde{I}_\alpha))], \tag{4}$$

where $\tilde{I}_\alpha$ denotes an alpha channel of a randomly cropped image I. The objective from eq. (4) lets the transparency have higher values around the cropped area (the 0.5 factor before it makes these values influence the objective value to a lesser extent than the ones in the cropped regions). The sample result can be seen in the image on the right-hand side of Figure 8.

2.4.3.4 Transformation Robustness

There was a set of strong transformation robustness functions applied so that the resulting visualizations are free of noise and higher quality. The following operations were applied in sequence: padding with 16 pixels from each side; random scale from a range [0.96, 1.05]; random rotation from range [-5, 5] degrees; random crop so that each dimension is 90% of its original value (0.9·128); alpha blending with the noise standard deviation equal to 0.2.

2.5 Units of Interpretability

As mentioned in section 2.2, activation tensor can be sliced in many different ways, providing multiple views on the underlying abstractions represented by CNNs. The problem with raw activations, though, is that they are neither intuitive nor particularly interpretable—their analyses commonly revolve around quantitative observations like: "n units fired at this spatial location" or "top n units' indices are...". It would be helpful if one could describe the model's behavior in terms more abstract than just bare numbers.

Olah *et al.* [49] introduced the idea of *semantic dictionaries* to specifically address this problem (refer to Figure 9 for an example). Semantic dictionaries leverage feature visualization to represent each channel not as the index in the activation tensor but rather as a synthetic image that depicts

a pattern that stimulates this particular channel the most. Moreover, the qualitative information (visualization) is presented together with the quantitative one (response strength), providing a hybrid interface that sheds new light on CNNs' computation.

Naturally, semantic dictionaries are not only applicable to individual channels but can also be successfully used to scrutinize more complex units of interpretability like feature columns or groups of neurons. In the following sections, one can read about and compare the insights that each of these methods provides. To a greater or lesser extent, feature visualization is presented jointly with attribution in the form of informative interfaces that help develop deeper intuitions about the studied algorithms. Such interfaces scale down the analyses and make them feasible to be conducted by a single human researcher.

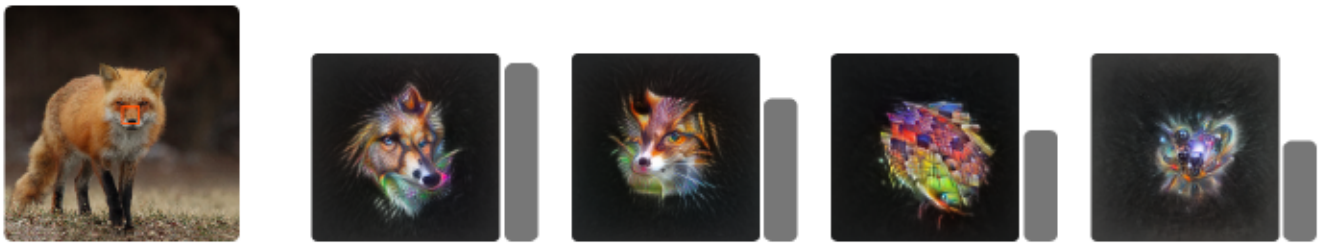


Figure 9: Sample semantic dictionary (model: InceptionV1, layer: mixed4d). Overlaid on the input image one can see a small red square that indicates the current feature column that is analyzed. At this particular location (layer mixed4d has a feature map of shape (14, 14, 528) so a single feature column is of shape (1, 1, 528)), four channels that activated the most are presented on the right-hand side of the figure. The higher the gray vertical bar, the stronger the response to the stimulus. Code used to generate the figure adapted from Colab notebook published in [49].

2.5.1 Individual Pixels

The most basic method of studying attribution is to quantify the influence of individual pixels on the classification result (see Figure 10 for details). This fundamental idea—called saliency map—was first introduced by Simonyan *et al.* in 2013 [57] and has seen a dramatic improvement since then. The multitude of available techniques has been described in section 2.3.

The major disadvantage of treating individual pixels as a unit of interpretability is that it is not a suitable level of abstraction for computer vision analyses. To comprehend the scene, systems need to be able to process more subtle concepts like textures, shapes of specific objects, and complex high-level features like complete entities. Even though not being particularly useful, per-pixel analysis is a very important introduction to the study of the units of interpretability.

Figure 10: Unit of interpretability: individual pixels. **Left:** schematic representation of what is being analyzed in this scenario (marked with red color—pixels of the input image). **Middle**: sample input image of a red fox class. **Right:** saliency map representing the influence of individual pixels on the classification result (white denotes high influence, gray indicates neutral—zero—importance, and black represents a negative (decreasing the probability of a "red fox" class) attribution).

2.5.2 Individual Neurons

The first step to go one level of abstraction deeper than mere pixels is to pick an arbitrary neuron of any channel (in any layer of a CNN) and use it as the optimization objective. As a result, the input pattern that causes that particular unit to fire strongly is generated (refer to Figure 11 for an example). That is exactly what has been mentioned before when introducing the idea of the semantic dictionary—at this point, it is possible to characterize part of a CNN not only by a bare number (activation value) but also by a "visual identifier" that makes it easier to conceptualize what a specific unit responds to.

Another useful aspect of treating neurons as the units of interpretability is that, intuitively, they tend to have more sense than naive pixels—if a neuron can represent a complex concept like the "fox's snout", then maybe it could be seen as a building block of the interpretability toolkit to study CNNs? Unfortunately, the truth appears to be much more compound. An attempt to unravel some of its pieces was made in chapter 3, in the argument on the distributed coding scheme, without rendering a definite answer, though. Nonetheless, the analyses of individual neurons provide new insights on the representational capacity of the CNNs and are surely a step in a plausible research direction, if properly extended.

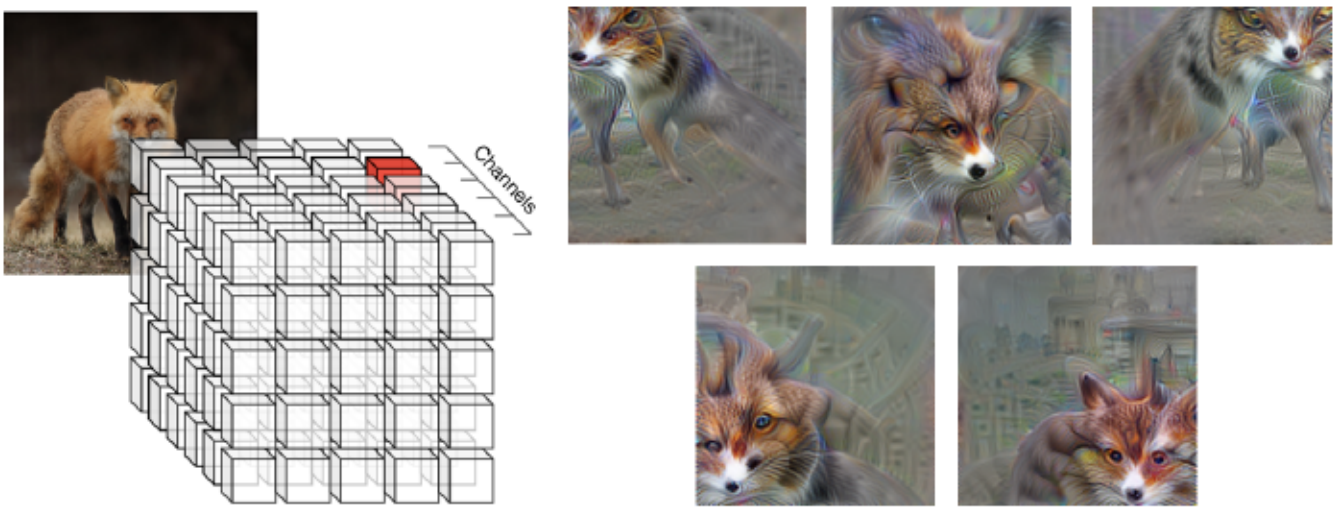


Figure 11: Unit of interpretability: individual neurons. **Left:** schematic representation of what is being analyzed in this scenario (marked with red color—single neuron in a given layer). **Right:** 5 different neurons from the same channel, yet with differing spatial locations (model: InceptionV1, layer: mixed4d, channel: 493). Note the varying location of the core input pattern (fox's snout-like) depends on the neuron that was chosen as the objective for the activation maximization.

2.5.3 Individual Channels

Maximizing the activation of the entire channel often leads to a better understanding of a specific convolutional filter than when looking only at a single neuron—the resulting image contains more potentially useful information (refer to Figure 12 for a concrete example). The undesirable side effect, though, is that visualizing the entire channel can yield the same pattern being scattered across multiple locations of the synthesized input, which can be very misleading. When the objective includes the transparency term, however, the repetitive nature of the output is rarely the case, making the channel a go-to objective over mere neuron.

Even though the majority of the research done in the fields of feature visualization and attribution focuses mostly on the class-level optimization (finding image that would activate given class the most), more fine-grained analysis of individual channels provides deep insights into the latent representation of the data. Major contributions come from [48, 49], yet the idea was also used in [19] to reveal salient, class-specific channels and analyze their attribution to the output.

There is a significant negative aspect of the analysis of CNN's channels that makes it practically unfeasible to accomplish: there are too many convolutional filters to study and visualizing all of them is not suitable for efficient and effective human analysis. Hence the need to automate the process of extracting the important substructures from the activation tensor as much as possible.

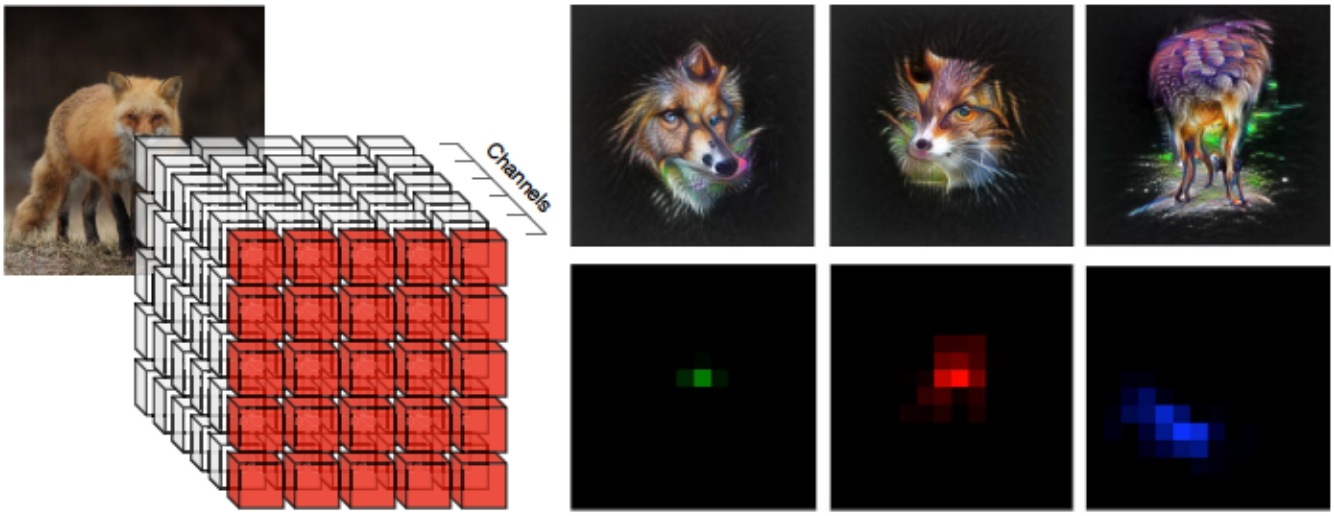


Figure 12: Unit of interpretability: individual channels. **Left:** schematic representation of what is being analyzed in this scenario (marked with red color—single channel of a given layer). **Right:** top 3 channels that activated the most (when applying reduce mean operation on each channel separately) for the red fox image. The top row contains feature visualizations of the channels. The bottom row represents the attribution of the corresponding channel at a particular spatial location of the image. The resolution of the heatmap is not equal to the resolution of the input image—it is equivalent to the spatial dimension of the feature map of the layer of interest (mixed4d layer of the InceptionV1 model) and in this example amounts to (14, 14). Colors do not have a particular meaning here: the brighter the heatmap, the stronger the influence on the final classification result.

2.5.4 Feature Columns

Channels can be combined in multiple different ways: randomly, manually (e.g. by specifying a set of neighboring ones), or automatically by leveraging techniques like matrix decomposition to identify representative combinations that can be used for further analyses of the CNNs. While the first two options do not differ significantly from previously described units of interpretability, the use of automatic methods to extract meaningful linear combinations of filters is a far more nuanced and complex topic.

In this work, the term *concept* is used to denote an individual, representative feature column, understood as a linear combination of channels at a single spatial location of the activation tensor. This definition is contradictory to the one presented by Kim *et al.* in [28]—in their work what they call the "concept" is the entire activation tensor in this thesis' nomenclature.

Linear combinations of channels can theoretically represent highly abstract objects like, to provide a concrete example, something resembling a red fox snout (channels that seem to respond to

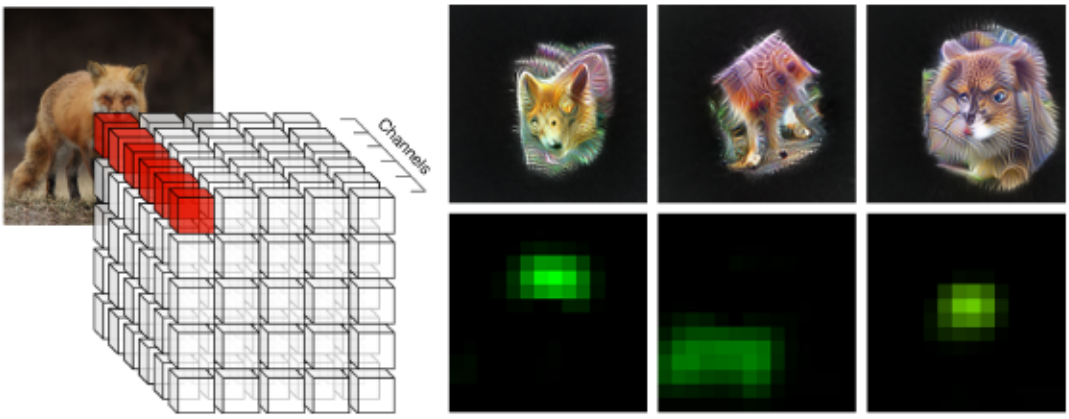


Figure 13: Unit of interpretability: feature columns. **Left:** schematic representation of what is being analyzed in this scenario (marked with red color—single feature column extracted from a given layer). **Right:** 3 examples of feature columns automatically identified using matrix decomposition. The top row contains their feature visualizations, while the bottom row depicts the attribution to the classification result a specific feature column has at a given spatial location. At first glance, the abstractions represented by feature columns do not appear to be more interesting or complex than the ones of individual channels, yet more advanced studies reveal the differences between the two approaches. Feature column is a generalization of the objective of the individual channel—it can reduce itself into a single channel if beneficial for the optimization.

input patterns similar to red fur, ears, and snout are merged together to represent the concept that is astonishingly alike a red fox snout). It has to be highlighted that every time a specific concept is given a name, it is solely the author's subjective interpretation of the generated visualization. Please refer to Figure 13 for an example of feature columns' visualization.

A key example of what insights and intuitions can be gained when conducting the analysis on the feature column level was presented in [19]. Fong *et al.* maximized the dot-product of the channel attribution vector and the activation tensor to unravel a subset of channels that is most meaningful to the class of interest. They managed to automatically identify core concepts for a set of classes and conduct compelling, previously unseen analyses.

Studying feature columns is one of the most underexplored branches of the interpretability discipline (when it comes to feature visualization and attribution methods). Even though learned combinations of channels were explored in some research in the past ("per-instance channel attribution" and "class-specific channels" in [19], a building block of "neuron groups" in [49] and "attribution graphs" in [25]), it is still a new term in the field and needs further contributions.

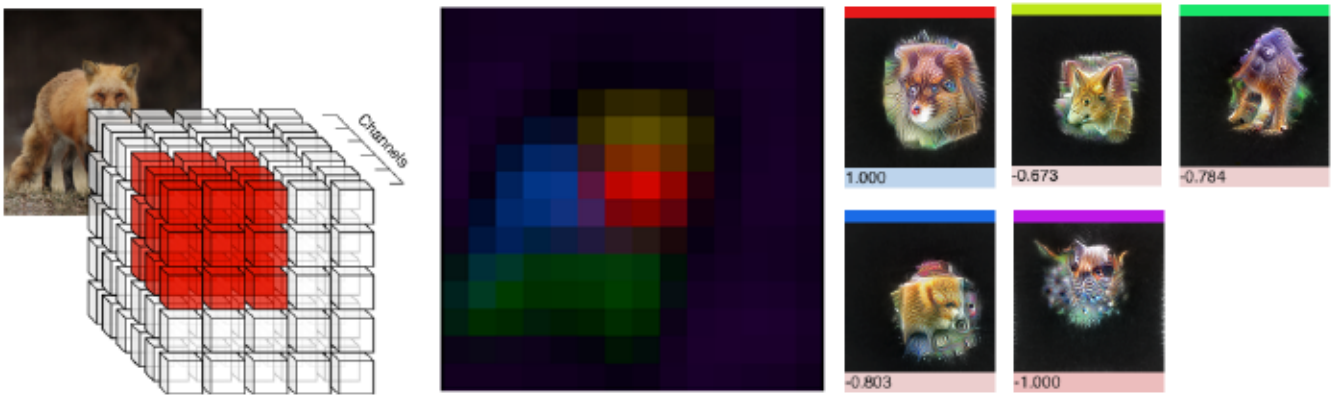


Figure 14: Unit of interpretability: neuron groups. **Left:** schematic representation of what is being analyzed in this scenario (marked with red color—a group of neurons identified by the matrix decomposition technique). **Middle:** 5 distinct groups of neurons from a red fox image (visible in the back of the schematic on the left). Each color denotes a different group and matches the labels of the feature visualizations on the right-hand side of the figure. The intensity of a given spatial location tells how much "present" the feature column associated with the concrete group is in this area of the image. **Right:** all of the neuron groups that were identified in the activation tensor by the matrix decomposition. For each group, a color label, feature visualization, and the mean attribution value were provided. Please note that the groups' attribution has been scaled so that the values range from -1 to 1. The normalization is not essential but it helps to compare the impact of each group. Code used to generate the figure adapted from a Colab notebook from [49].

2.5.5 Groups of Neurons

An idea to optimize the activation of the entire group of neurons is unique to the work of Olah *et al.* [49]. Their motivation was to design an interpretability interface that would be human-scale, contrary to the overwhelming amounts of neurons, channels, or manually-identified feature columns, all coming in the numbers of hundreds or thousands. What is more, neuron groups try to follow the intuition that it should be possible to decompose the activation tensor into multiple meaningful clusters that would be easily comprehensible and that would focus only on the important concepts in the image, disregarding the distracting part of the available information.

Neuron group is an abstraction that integrates the "what" and "where" information. Matrix decomposition is used to find feature columns[7] that represent the most frequent concepts in the activation tensor. For each column, another piece of information—the "where" aspect—is provided

[7]Number of groups to decompose the activation tensor into can be determined either manually or automatically, depending on the technique of choice.

to illustrate how much of a given abstraction is present at a single spatial location of the activation tensor. Coupling feature columns with spatial locations in such a way yields the results that are often much easier to interpret than the visualizations obtained using other techniques. Please refer to Figure 14 for an example). The original approach [49] leverages the NMF decomposition technique to reduce an overwhelming number of neurons into a comprehensible set of groups. Optimal parameters and the obtained visualizations are image-specific, though. It means that matrix decomposition needs to be performed for each input separately, which can be perceived as a major downside of the described method. Nonetheless, the influential explanations that it enables are outstanding, clearly justifying its computational complexity.

Even though it might resemble the aforementioned saliency maps, the information on where and how strongly the concept activates in the image does not represent the attribution. Preliminary research on the neuron groups' influence on the classification result was conducted by Olah *et al.* [49]. Their results, however, were not presented as the attribution maps. Instead, they were published as a single scalar value associated with each group separately, representing its influence on the classification result (negative values denoting a **decrease** in target class's probability, and positive values denoting an **increase** in the target class's probability).

2.6 Interpretability Interfaces

Even though feature visualization and attribution are usually studied in isolation, the research that involved treating them as hybrid interfaces yielded both intriguing and compelling results [12, 48, 49]. Thanks to such novel studies the researchers could develop deeper intuitions about models' behavior. It is truly encouraging to observe and makes one rethink the way the field of interpretability has been using what is already available. It also paves new, exciting ways for further exploration.

2.6.1 Network Dissection

One of the first interfaces that combined the "what" and "where" information is called Network Dissection [8]. It is an alternative approach to feature visualization and attribution that revealed the existence of neurons that respond to high-level abstractions like houses, airplanes, staircases, or human heads (and many more), even though these concepts were not present as separate classes in the training data. These neurons (or rather: channels of the convolutional layers) alone contributed from 37% to 71% of the interpretable abstractions in the final conv5 layer of the AlexNet

architecture [8]. The remaining units did not appear to be comprehensible by human evaluators without any doubts or inconsistencies, suggesting that the subjective impressions' influence was too strong to make reliable claims.

2.6.2 CNN Codes

Andrej Karpathy is one of the most recognizable figures in the research on CNNs. In the CNN Codes experiment, he "took 50,000 ILSVRC 2012 validation images, extracted the 4096-dimensional fc7 CNN [...] features using Caffe and then used Barnes-Hut t-SNE to compute a 2-dimensional embedding that respects the high-dimensional (L2) distances. In other words, t-SNE arranges images that have a similar CNN (fc7) code nearby in the embedding" [27]. To paraphrase his quote, the goal was to cluster high-dimensional vectors and project them down onto the 2D space so that they can be easily visualized. An interesting side effect of the t-SNE manifold projection technique is that the vectors that are to some extent similar end up being close to one another in the 2D space as well. By inspecting the projections, it is possible to visually capture the relations between representations that a CNN builds for individual images and to evaluate whether the images that contain similar concepts are close to one another according to the CNN Codes' metric.

2.6.3 Activation Atlas

Carter *et al.* built on top of the ideas of clustering activation tensors and projecting them onto the 2D space [27, 46]. They went one step further and designed the highly interactive and informative interface called the Activation Atlas [12]. Activation Atlas unifies the previous work done by Karpathy and Nguyen *et al.* in a very elegant way—the core rule is the same as in CNN codes (clustering + projection), yet instead of using samples from the training data to attach visual identifiers to the 2D projections, Carter *et al.* leveraged the potential of feature visualization (see Figure 15 for an example). Additionally, they have identified the UMAP mapping technique [38] to yield more plausible atlases than the ones generated using t-SNE.

Figure 15: Sample activation atlas of the InceptionV1 model, trained on the ImageNet dataset. Image retrieved from [12], licensed under CC BY 4.0.

Chapter 3

Latent Factor Attribution

Finding an equilibrium between low-level (e.g. gradient [57]) and high-level (e.g. TCAV [28]) attribution techniques served as the core motivation for the research presented in this chapter. There is a lot of space for new contributions in the domain of concepts' attribution—according to the author's best knowledge, there are only two significant pieces of work on this topic [49, 70]. To fill the niche and, effectively, expand the space of mid-level interpretability interfaces, a novel Latent Factor Attribution technique is proposed in this thesis.

LFA extends the holistic view on the attribution maps by enabling the collective perspective on the entire groups of neurons' influence on the score assigned to each class. What makes it possible is the fact that, when using LFA, the classification output directly depends on the information on "where" a given concept is present in the activation tensor and "how much" of it is there. The strongest asset of the proposed method is that it enables backpropagation through the computational graph—it is feasible to calculate the gradient of the classification score w.r.t. the concepts identified using matrix decomposition, thus obtaining the attribution of concepts at each spatial location of the generated heatmap. This is not the case for the existing techniques that "disconnect" the computational graph—their attribution estimates are separate from the rest of the calculations and serve as rather indirect heuristics for the factors' importance. Latent Factor Attribution can be used by researchers to have a closer look at their models from a very different perspective—when exploring the latent representations of their CNN or reasoning about how it comes to specific conclusions; they can glimpse into the model's response to a given input (or even the entire class of inputs), thus gradually building deeper intuitions about the algorithm's behavior. Even though concept analysis could theoretically be presented to the non-expert end user in a form similar to a report, at this stage of the research as presented in this thesis, it is rather not feasible (yet may be explored in the future).

In this work, the InceptionV1 network [61] was used (also known as GoogleNet). The model was pre-trained on the ImageNet dataset [16], with the weights retrieved from the Lucid[1] library and available[2] at the following URL: `gs://modelzoo/vision/other_models/InceptionV1.pb`. For the analysis, mixed4a, mixed4b, mixed4c, mixed4d, mixed4e, mixed5a and mixed5b layers were used (table 1 from the original paper on InceptionV1 [61] introduced the following naming convention for the layers: inception(4a), inception(4b), inception(4c), etc. In this work, they are called "mixed4a", "mixed4b" etc., following the nomenclature from [12, 48, 49]). "Mixed" layers are the outputs of the inception modules, concatenated to form a single activation tensor. The analyses presented here begin at mixed4a layer's output so that the underlying representation is complex enough (see [47] for details on earlier layers of the InceptionV1 model). As a final note, the auxiliary classifiers that were originally present in the network were not used in this work.

3.1 Distributed Coding Scheme

For many years now, neuroscientists have been making efforts to better understand how information is encoded in the biological neurons in the visual cortex of the human brain. Two opposing hypotheses arose: sparse coding scheme and distributed coding scheme. The former states that it might be beneficial to have a dedicated unit for each feature to allow better disentanglement of the incoming visual stimuli and, eventually, simplify the downstream processing [5, 7, 18]. As elegant as the sparse coding appears to be, the research suggests that it is not feasible for the visual cortex to dedicate a distinct neuron to each higher-level concept the average human encounters in everyday life [22]—one would quickly run out of neurons in such scenario. Rather, a distributed coding scheme emerges as an alternative hypothesis. Even though it may not seem to be as elegant or simple to analyze as its sparse counterpart, it theoretically enables the neurons to encode exponentially more features than in sparse coding and, as a result, make the entire visual processing feasible [14, 22, 39, 53, 66].

Network Dissection [8] revealed the existence of neurons that respond to high-level abstractions like houses, airplanes, staircases, or human heads (and many more), even though these concepts were not present as a separate class in the training data. These neurons (or rather: channels of the convolutional layers) alone contributed from 37% to 71% of the interpretable abstractions in the final conv5 layer of the AlexNet architecture. The remaining units did not appear to be comprehensible by human evaluators without any doubts or inconsistencies, suggesting that the

[1] https://github.com/tensorflow/lucid

[2] At the time of preparing this manuscript.

subjective impressions' influence was too strong to make reliable claims. May it be true that such an uninterpretable neural population participates in the distributed coding scheme inside a CNN and thus is not explainable when analyzed per-unit? This would, indeed, correspond to the emerging theory in neuroscience that there exists a truly distributed coding strategy in many areas of the brain, including the visual cortex [14, 39]. Such structure deprecates the meaningfulness of a single unit—an entire neural population would have to be analyzed as a whole to be able to draw unbiased, objective conclusions. Such a theory would be an elegant explanation to the problem of "polysemantic neurons" [11] (they do not seem to respond to a single feature, but rather a mixture of unrelated concepts). Individual analysis of such units will not reveal their true nature—they only make sense as a part of the greater structure. To verify the "polysemanticity" claim one would have to manually inspect the activation tensors at every layer of a CNN of interest and identify the latent patterns one by one. Such an approach would not be feasible to accomplish by humans—it would take a lot of time and effort, not to mention the complexity of the analysis of high-dimensional activation tensors.

A recent initiative called Circuits [11] attempts to manually reverse-engineer a single CNN—InceptionV1— to understand how individual neurons interact with one another and, as a result of these interactions, can represent complex abstractions. The Circuits experiment is still in an early stage so it is hard to tell what the outcomes are going to be. Nonetheless, the initial results suggest that the inner representations constructed by CNNs are incredibly rich and, most importantly, comprehensible.

3.2 Vanilla Attribution

In the preceding parts of this thesis, the term "attribution map" has been introduced to denote heatmaps that visually indicate the influence of activation tensor's spatial locations on the classification score. From now on, the attribution map that is generated using standard techniques (like pure gradient [57], gradient $\odot$ input [55] or Integrated Gradients [60]) will be referred to as a *vanilla attribution map*. In this work, all the attribution experiments were based on pure gradient, gradient $\odot$ input, and Integrated Gradients. Such a decision was based on recent work that has taken a closer look at the reliability of attribution methods [2], indicating that the results yielded by the aforementioned techniques can be trusted.

The goal of the vanilla attribution maps is to help understand the influence of single spatial locations in the activation tensor on the final classification score. Such visualizations allow the

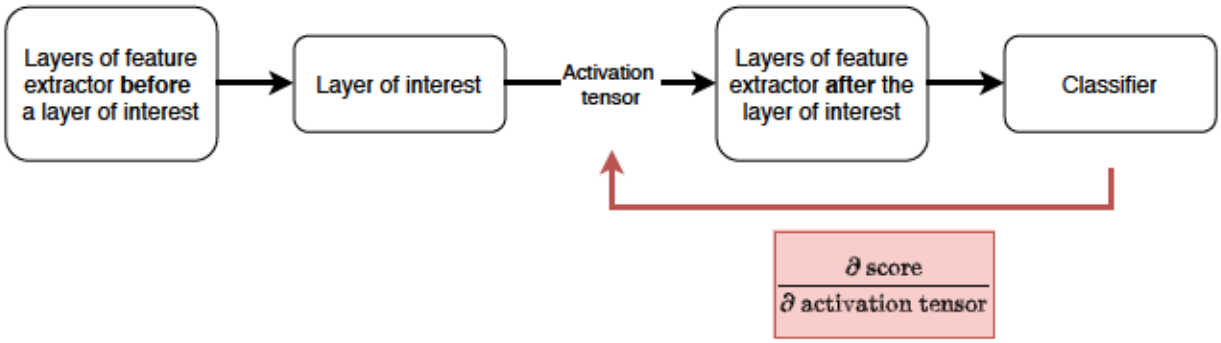


Figure 16: Vanilla attribution computation scheme. The classification score depends directly on the activation tensor from the layer of interest. The direct relationship is what enables calculation of the gradient of the score w.r.t the activation tensor to find its attribution to the model's output.

following question to be answered: what would happen to the final classification score if a particular spatial location in the activation tensor fired more/less? At any layer of the model, it is possible to calculate the gradient of the score (e.g., logit—final classification result before applying softmax) w.r.t. the arbitrary activation tensor to better understand its complex relationship with the prediction. The process of obtaining a 2D saliency map from the 3D activation tensor can be expressed mathematically as follows:

$$\boldsymbol{A} = \sum_{d=1}^{\text{depth}} \mathbf{A}_{:,:,d}, \quad \mathbf{A} = \mathbf{H} \odot \nabla_{\mathbf{H}} \text{logit}_c, \tag{5}$$

where $\boldsymbol{A}$ denotes the resulting 2D vanilla attribution map, $\mathbf{A}$ is the 3D attribution tensor and $\mathbf{H}$ is the 3D activation tensor. Here, tensor $\mathbf{A}$ is the result of the element-wise product of the activation tensor $\mathbf{H}$ and the gradient of the logit of class c w.r.t. that activation tensor. The element-wise product estimates the rate at which an increase in a neuron's response influences the logit tensor [12] and can be seen as the equivalent of the gradient $\odot$ input approach, with the input being the activation tensor. The gradient can be calculated using, for instance, the Integrated Gradients technique. Please note that the term $\nabla_{\mathbf{H}} \text{logit}_c$ in eq. (5) is only a placeholder for any gradient calculation method, i.e., it does not reflect the actual method used to obtain the gradient. Finally, to produce a 2D attribution heatmap $\boldsymbol{A}$ from 3D attribution tensor $\mathbf{A}$, a sum reduction function is applied along the depth axis. The entire vanilla attribution computation that has been described above can be seen in Figure 16.

3.3 Activation Tensor Decomposition

An alternative technique that can help to tackle the problem of uncovering the latent structure of a CNN in an automated (yet less-precise) way is called matrix decomposition. The matrix decomposition idea emerged from the need to discover the latent structure of matrices. By factorizing the activation tensor, one can identify groups of neurons (also known as neural populations) that, in theory, jointly represent complex abstractions in a distributed fashion [49].

Activation tensor $\mathbf{H}$ has shape (height, width, channels). A slice $\mathbf{H}_{x,y,:}$ of shape (1, 1, channels), where $x \in X = \{0, 1, 2, ..., \text{height}\}$ and $y \in Y = \{0, 1, 2, ..., \text{width}\}$, is called a feature column and can be interpreted as a linear combination of channels at a particular spatial location. Thus, one can say that the activation tensor $\mathbf{H}$ is made of height × width feature columns. After flattening, tensor $\mathbf{H}$ becomes a 2D matrix of shape (height × width, channels) (denoted as $\boldsymbol{H}$) that can be factorized to explore whether there exist repetitive patterns in the feature columns (i.e., if there are multiple columns that roughly represent a similar combination of channels). Matrix $\boldsymbol{H}$ of shape (height × width, channels) can be decomposed into two matrices: $\boldsymbol{S}$ that contains **latent spatial factors** and is of shape (height × width, #factors) and $\boldsymbol{F}$ with **latent factors** (distinct combinations of channels) of shape (#factors, channels). Latent factor is a mathematical representation of a concept, discovered automatically by the matrix decomposition. Latent spatial factor, on the other hand, is a mathematical representation of the information on "where" a given concept is present in the activation tensor and "how much" of it is at every spatial location, identified automatically with the matrix decomposition as well. Both latent factors and latent spatial factors can be visualized in order to be better understood: feature visualization is used for the former one (generating a synthetic superstimulus that activates the latent factor the most is equivalent to doing this for a feature column), while the latter can be displayed as heatmaps.

Matrix decomposition technique that is widely used across this thesis is called the Non-negative Matrix Factorization. If finds latent factors and latent spatial factors by iteratively minimizing the following optimization objective:

$$\underset{\boldsymbol{S},\boldsymbol{F}}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{H} - \boldsymbol{S}\boldsymbol{F}\|_F^2, \quad \boldsymbol{S}, \boldsymbol{H} \geq 0. \tag{6}$$

$\|\cdot\|_F^2$ denotes a Frobenius norm and is equal to $\|M\|_F^2 = \sum_{x,y} M_{xy}^2$, with M being an arbitrary matrix. Note that the product $\boldsymbol{S}\boldsymbol{F}$ is called a *reconstruction* and is an approximation of matrix $\boldsymbol{H}$. In the NMF decomposition's multiplicative update technique, $\boldsymbol{S}$ and $\boldsymbol{F}$ are iteratively updated according to the following rules:

$$\boldsymbol{F} \leftarrow \boldsymbol{F} \odot \frac{\boldsymbol{S}^T\boldsymbol{H}}{\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{F}}, \qquad \boldsymbol{S} \leftarrow \boldsymbol{S} \odot \frac{\boldsymbol{H}\boldsymbol{F}^T}{\boldsymbol{S}\boldsymbol{F}\boldsymbol{F}^T}. \tag{7}$$

That is the moment when neurological theories meet machine learning, facilitating the rise of an attractive research avenue. Uncovering combinations of neurons that act together is just part of the story though—it is important to be able to not only discover the existence of abstractions like neuron groups but also to understand what they represent and how they attribute to the visual processing output. Since the linear combination of channels can be seen as a direction in an n-dimensional space, it is possible to generate a synthetic input image that would result in a strong response of the neuron group [49]. Once both latent spatial factors (where the abstraction is located in the activation and how much of it is there) and the abstraction feature vectors themselves are available, the final step is to quantify their attribution.

The commonly accepted approach to quantify the attribution when working with CNNs is realized by generating so-called *attribution maps* that quantify the saliency of input image pixels [57]. This idea is based on a rather questionable assumption that an individual pixel is the right unit to analyze the attribution. Human intuitions are utterly different in this domain: we perceive the underlying objects in the image, not the single RGB intensities at each spatial location. Moreover, input space is just a fraction of the CNNs' representation capacity, hence it would be interesting to produce attribution maps for every layer of the network. With the intermediate layers' attribution maps, though, many new questions arise:

1. Do all the spatial locations in the attribution map indicate the presence/absence of a target class concept or rather something more specific?

2. Is there a way to decompose the activation tensor into subparts that would explain the bigger idea up to a predefined precision?

3. Do the hypothetical subparts appear repetitively across the images of the same class?

4. Is it possible to estimate subparts' attribution to the classification result for each of them separately? How reliable would such a measure be? How could this information be used?

Fortunately, matrix decomposition of the activation tensor makes it possible to answer all of the above questions, filling the niche in the space of latent factors interpretability. With all of the above, one would get a detailed overview of what information is encoded in every layer of the CNN, how it is represented, and what is its influence on the visual processing output.

To obtain the results presented in Figure 17, the activation tensor from the mixed4d layer of the InceptionV1 model was analyzed using the Non-negative Matrix Factorization technique. An alternative matrix decomposition approach—Principal Components Analysis—was also evaluated in this work. Sample results for PCA can be observed in Figure 18. In both Figure 17 and Figure 18,

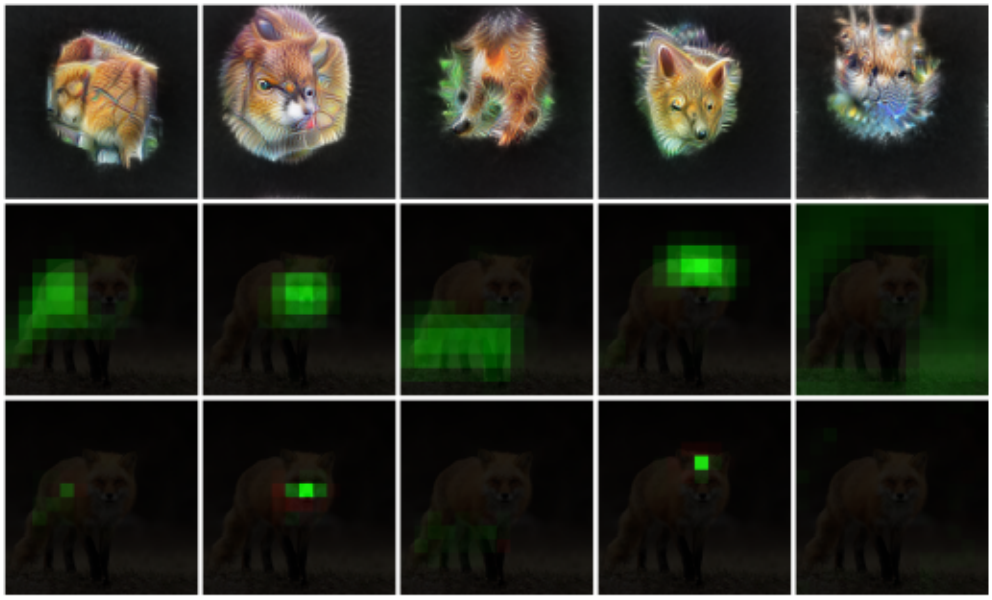


Figure 17: An example of the LFA using NMF matrix decomposition as a base for computation. **First row:** visualizations of distinct latent factors identified using NMF. **Second row:** corresponding latent spatial factors. **Third row:** LFA attribution map. Green and red areas denote positive and negative influence on the predicted class, respectively.

the latent spatial factors and LFA intensities have been globally normalized (each row separately) to highlight the absolute importance of single factors. Even though a per-factor normalization would yield more vivid visualizations, the relative importance information would be lost.

Please note that at this point in the narrative, both examples in this section were provided to visualize a kind of result that LFA yields. The intention is also to help the reader build visual intuitions about the problem being discussed, not to compare the LFA with the existing techniques or to comprehend its inner mechanism.

3.4 Latent Factor Attribution

The goal of LFA is to understand the influence of the most significant latent factors in the factorized activation tensor on the final classification score. The objective is similar to the one of vanilla attribution, yet it focuses on a more abstract unit of interpretability—latent factors. The question to be answered with LFA is the following: what would happen to the final classification score if a particular latent factor was more/less present in the activation tensor? The LFA technique consists of three stages (presented as a schema in Figure 19) that are executed sequentially:

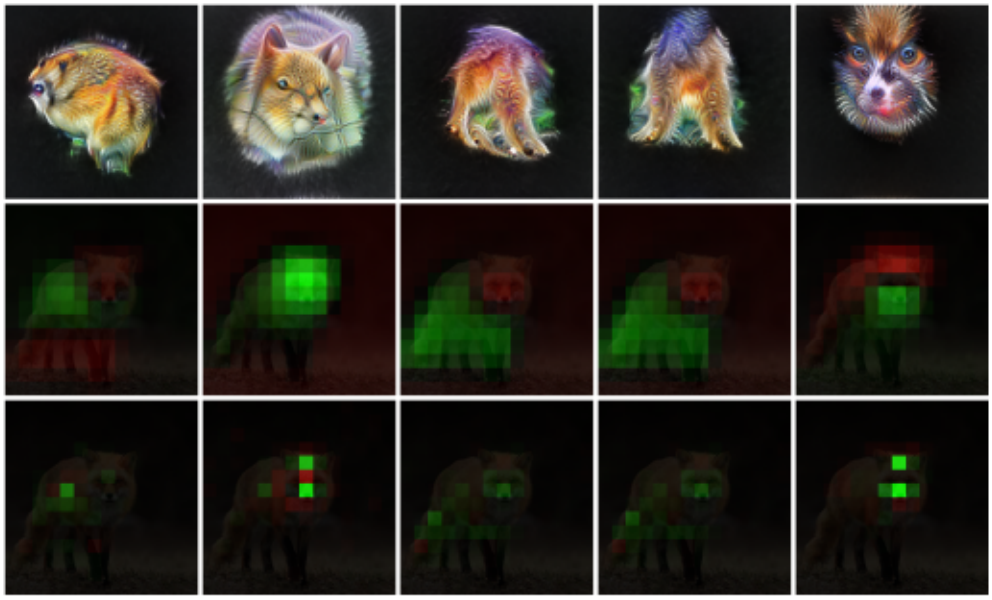


Figure 18: An example of the LFA using PCA matrix decomposition as a base for computation. **First row:** visualizations of distinct latent factors identified using PCA. **Second row:** corresponding latent spatial factors. **Third row:** LFA attribution map. Green and red areas denote positive and negative influence on the predicted class, respectively.

1. **Decompose** The activation tensor $\mathbf{H}$, flattened into an activation matrix $\boldsymbol{H}$ is decomposed into N latent spatial factors $\boldsymbol{S}$ (every **column** in $\boldsymbol{S}$ is a distinct latent spatial factor) and N latent factors $\boldsymbol{F}$ (every **row** in $\boldsymbol{F}$ is a distinct latent factor) using standard matrix decomposition techniques. In this work, NMF and PCA ones were evaluated.

2. **Reconstruct** There are N pairs of factors available in the form (*concept, where it is in the activation tensor*). The core idea of the LFA computation is to reconstruct the original activation matrix $\boldsymbol{H}$ into a compressed activation matrix $\tilde{\boldsymbol{H}}$, but using **only distinct factors** that jointly constitute to the majority of the information encoded in the latent representation. Selecting a subset of factors means selecting only the columns $\boldsymbol{S}^*$ (in case of $\boldsymbol{S}$) and rows $\boldsymbol{F}^*$ (in case of $\boldsymbol{F}$) that represent distinct concepts, discarding the rest. The resulting number of factors to be kept depends on the target value of the reconstruction error and/or a predefined number of factors a researcher would like to obtain and is equal to N^*. $\tilde{\boldsymbol{H}}$ contains a piece of the original information, represented solely by the most significant factors.

3. **Compute the attribution** As can be seen in Figure 19, the classification score depends directly on the distinct factors. That direct relationship is what makes it possible to calculate

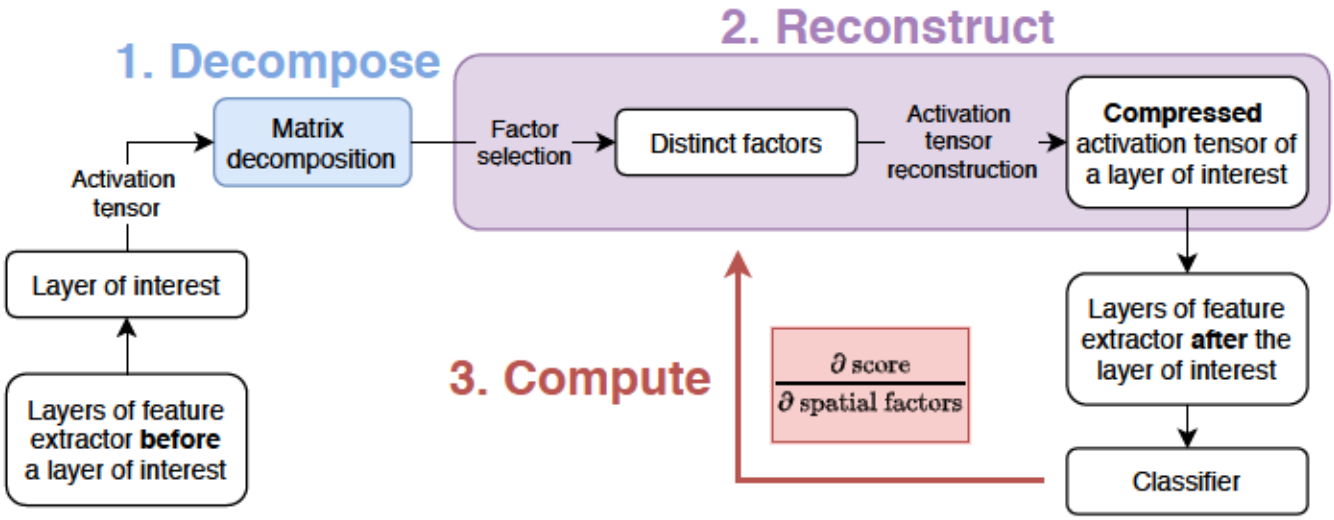


Figure 19: Latent Factor Attribution technique consists of three stages that are executed sequentially: decomposition, filtering and compression, and attribution computation. Since in this setup a score depends directly on the factors, one can backpropagate through the computational graph to obtain the gradient of the score w.r.t the factors and yield an attribution map for each of them.

the gradient of the score logit_c w.r.t. the filtered latent spatial factors and, thus, find their influence (attribution) on the result of the classification:

$$\boldsymbol{A}^{(\boldsymbol{F}^*_{n,:})} = \mathbf{S}^*_{:,:,n} \odot \nabla_{\mathbf{S}^*_{:,:,n}} \text{logit}_c, \tag{8}$$

where $\boldsymbol{A}^{(\boldsymbol{F}^*_{n,:})}$ denotes the attribution of the concept $\boldsymbol{F}^*_{n,:}$, $\mathbf{S}^*$ denotes a matrix $\boldsymbol{S}^*$ reshaped so that its shape is (height, width, #factors), and $n = \{0, 1, 2, ..., N^*\}$ is an index of the arbitrary concept. Here, the element-wise product $\odot$ between the latent spatial factor associated with the n-th concept and the gradient of the score w.r.t. that factor plays the same role as in eq. (5). When estimating the importance of the n-th concept $\boldsymbol{F}^*_{n,:}$, we are not interested in getting the information about how altering the concept itself ($\boldsymbol{F}^*_{n,:}$) would influence the classification score. Rather, what is important is how increasing/decreasing a given concept at spatial locations would influence the output of a CNN. That is exactly the kind of information the $\nabla_{\mathbf{S}^*_{:,:,n}} \text{logit}_c$ term represents. Changes in the $\mathbf{S}^*_{:,:,n}$ affect the reconstructed activation tensor $\tilde{\mathbf{H}}$, thus influencing the classification score.

3.5 Differences with the Existing Techniques

The field of interpretability has already seen the techniques that compute the attribution of the latent factors found by the matrix decomposition of the activation tensors: *The Building Blocks of Interpretability* [49] and *Interpretable Basis Decomposition* (IBD) [70].

In [49], preliminary research on neuron groups was presented. After an activation tensor $\mathbf{H}$ (see eq. (6)) is decomposed into N factors (from which N^* ones are selected based on an arbitrary criterion), neuron groups are to be formed. Let $\hat{\mathbf{S}}^{(n)}$ denote the n-th latent spatial factor $\mathbf{S}^*_{:,:,n}$ broadcast in such a way that $\hat{\mathbf{S}}^{(n)}_{:,:,d} = \mathbf{S}^*_{:,:,n}$, i.e. every arbitrary channel d is equal to the latent spatial factor $\mathbf{S}^*_{:,:,n}$. Also, let $\hat{\mathbf{F}}^{(n)}$ denote the n-th latent factor $\boldsymbol{F}^*_{n,:}$ (concept) broadcast in such a way that $\hat{\mathbf{F}}^{(n)}_{x,y,:} = \boldsymbol{F}^*_{n,:}$, i.e. feature column at every arbitrary spatial location (x, y) of the tensor $\hat{\mathbf{F}}^{(n)}$ is equal to $\boldsymbol{F}^*_{n,:}$. To obtain a neuron group $\mathbf{G}^{(n)}$ associated with the n-th factor, the following formula can be used:

$$\mathbf{G}^{(n)} = \hat{\mathbf{S}}^{(n)} \odot \hat{\mathbf{F}}^{(n)}, \tag{9}$$

where $\odot$ denotes an element-wise multiplication of tensors. To estimate the influence of a neuron group on the classification result, a group tensor $\mathbf{G}^{(n)}$ is multiplied element-wise with the vanilla attribution tensor $\mathbf{A}$ as follows:

$$\mathbf{A}^{(\mathbf{G}^{(n)})} = \mathbf{G}^{(n)} \odot \mathbf{A}, \quad \mathbf{A} = \mathbf{H} \odot \nabla_{\mathbf{H}} \text{logit}_c \tag{10}$$

where $\mathbf{A}^{(\cdot)}$ denotes an attribution of a given group. For example, an attribution of a group related to the n-th concept would be denoted as $\mathbf{A}^{(\mathbf{G}^{(n)})}$. Originally, neuron groups' attribution does not come in a form of 2D heatmaps [49]. Instead of applying a reduction function along the channel dimension, all the elements of the neuron groups' attribution tensor are summed together, yielding a single scalar value for each group tensor, according to the following formula:

$$a^{(n)} = \sum_{x=1}^{\text{height}} \sum_{y=1}^{\text{width}} \sum_{d=1}^{\text{depth}} \mathbf{A}^{(\mathbf{G}^{(n)})}_{x,y,d}. \tag{11}$$

If, in place of global sum reduction, a channel-wise variant was used, a 2D neuron group's attribution heatmap would be computed. A compelling side-effect of such an approach would be that it could be directly compared with LFA maps. That is exactly what was done in this thesis in order to obtain a 2D neuron groups' attribution $\boldsymbol{A}^{(\mathbf{G}^{(n)})}$ and, effectively, juxtapose the two methods:

$$\boldsymbol{A}^{(\mathbf{G}^{(n)})} = \sum_{d=1}^{\text{depth}} \mathbf{A}^{(\mathbf{G}^{(n)})}_{:,:,d}. \tag{12}$$

The core distinction between the LFA and the neuron groups' attribution manifests itself in the way how they handle both latent spatial factors and latent factors. By embedding factor's selection in the main computational graph (by introducing the idea of compressed activation tensor reconstruction), LFA makes it possible to backpropagate through the computational graph up to the latent spatial factors and, as result, directly compute their attribution to the target class. On the other hand, neuron groups' attribution technique does not perform any computation of this kind, instead using the broadcasting trick and element-wise product with the vanilla attribution tensor as the base for the concepts' importance estimation. By comparing both techniques (with neuron groups' attribution in the expanded form) side-by-side one can see that they are, in fact, very different:

$$\boldsymbol{A}^{(\mathbf{G}^{(n)})} = \sum_{d=1}^{\text{depth}} \left(\hat{\mathbf{S}}^{(n)} \odot \hat{\mathbf{F}}^{(n)} \odot \mathbf{H} \odot \nabla_{\mathbf{H}} \text{logit}_c \right)_{:,:,d}, \quad \boldsymbol{A}^{(\boldsymbol{F}^{\star}_{n,:})} = \left(\mathbf{S}^{\star} \odot \nabla_{\mathbf{S}^{\star}} \text{logit}_c \right)_{:,:,n}. \tag{13}$$

LFA attribution $\boldsymbol{A}^{(\boldsymbol{F}^{\star}_{n,:})}$ is much more elegant, concise, and, most importantly, provides a direct estimate for the concepts' importance, contrary to its counterpart.

In Figures 20 to 28, neuron groups' attribution maps have been juxtaposed with the LFA ones. Please note that each factor has been individually normalized so that its values fall into [0, 255] range. This was done to highlight the differences between the two methods (global normalization would yield dimmer attribution maps that would make visual comparison infeasible). One can observe that LFA produces sparser heatmaps that appear to be more coherent with the underlying latent spatial factors than the ones obtained using neuron groups' attribution. LFA attribution maps have an important feature of isolating the per-factor attribution information better than their counterpart—if some region of the vanilla attribution map clearly dominates over the others, its information "leaks" into every groups' attribution map, making it harder to understand them in isolation. A very good example of this phenomenon is easily noticeable in Figure 24. This problem is not visible when computing factors' attribution using the proposed LFA technique.

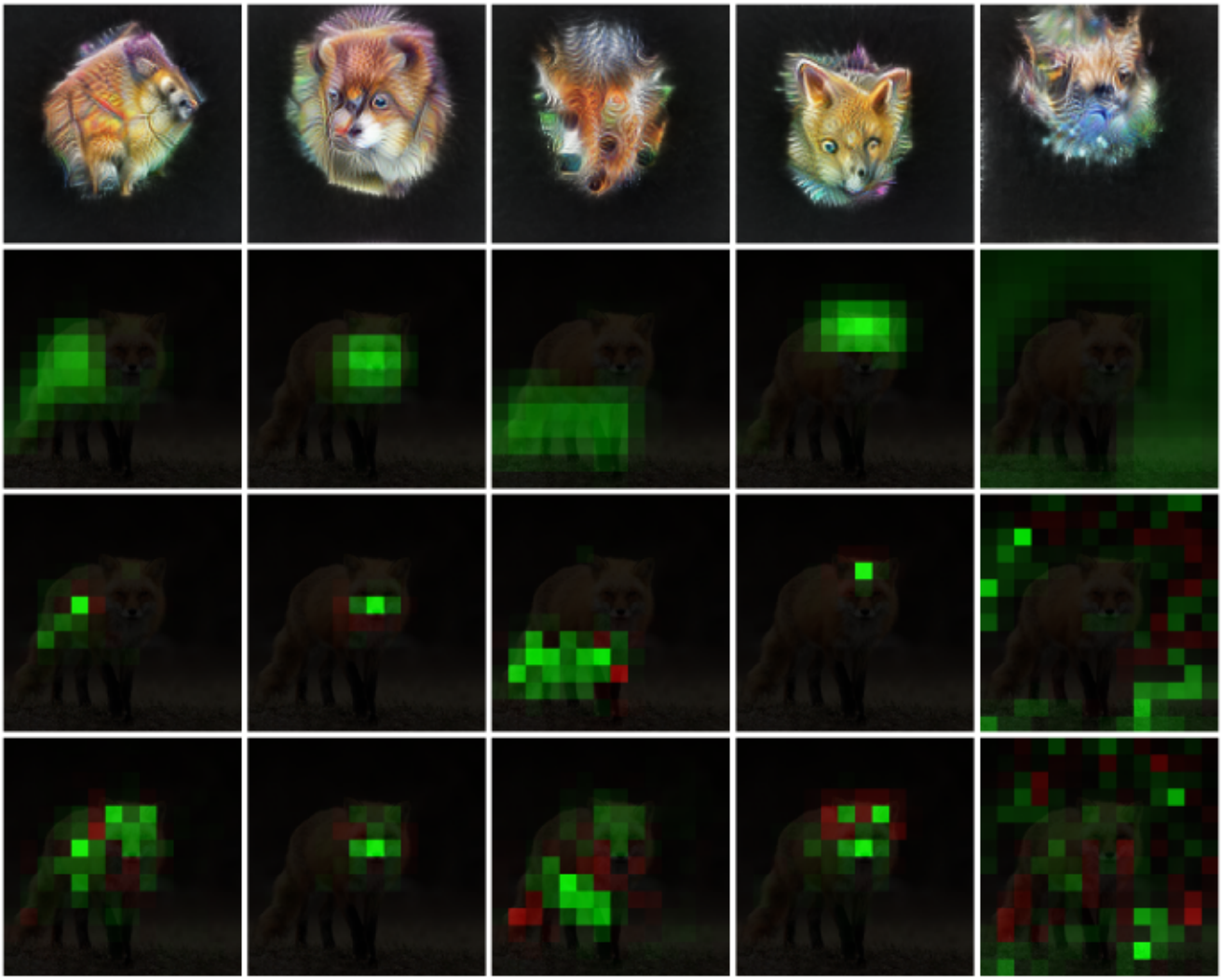


Figure 20: Comparison of LFA and neuron groups' attribution maps for a red fox target class (predicted class: red fox, with a 51.06% confidence).. **First row:** feature visualizations of the distinct latent factors identified using NMF. **Second row:** corresponding latent spatial factors. **Third row:** LFA attribution maps. **Fourth row:** neuron groups' attribution maps. Green and red areas denote positive and negative influence, respectively. Code used to compute the fourth row of the figure adapted from a Colab notebook published in [49].

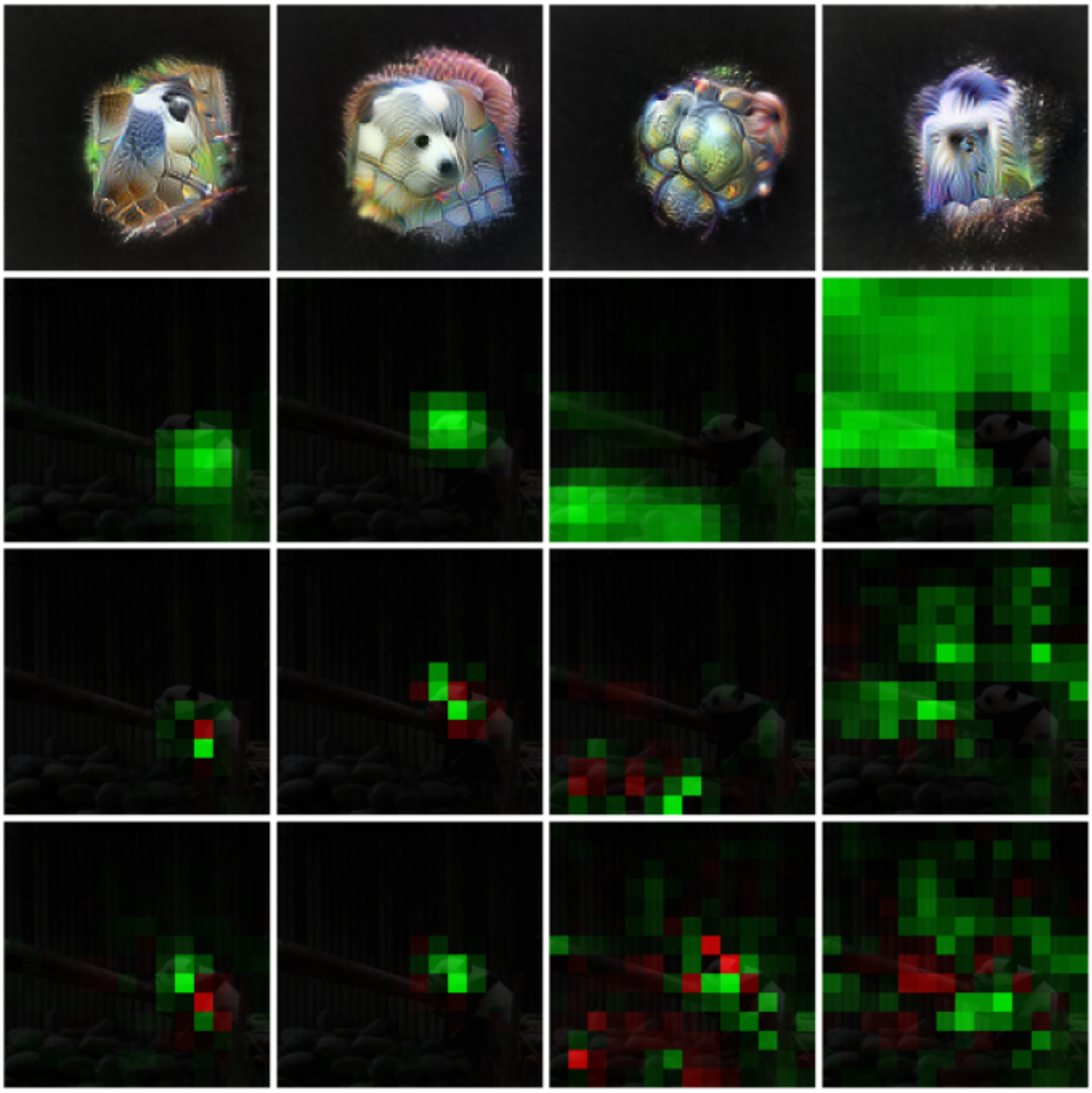


Figure 21: Comparison of LFA and neuron groups' attribution maps for a giant panda target class (predicted class: giant panda, with a 95.55% confidence). **First row:** feature visualizations of the distinct latent factors identified using NMF. **Second row:** corresponding latent spatial factors. **Third row:** LFA attribution maps. **Fourth row:** neuron groups' attribution maps. Green and red areas denote positive and negative influence, respectively. Code used to compute the fourth row of the figure adapted from a Colab notebook published in [49].

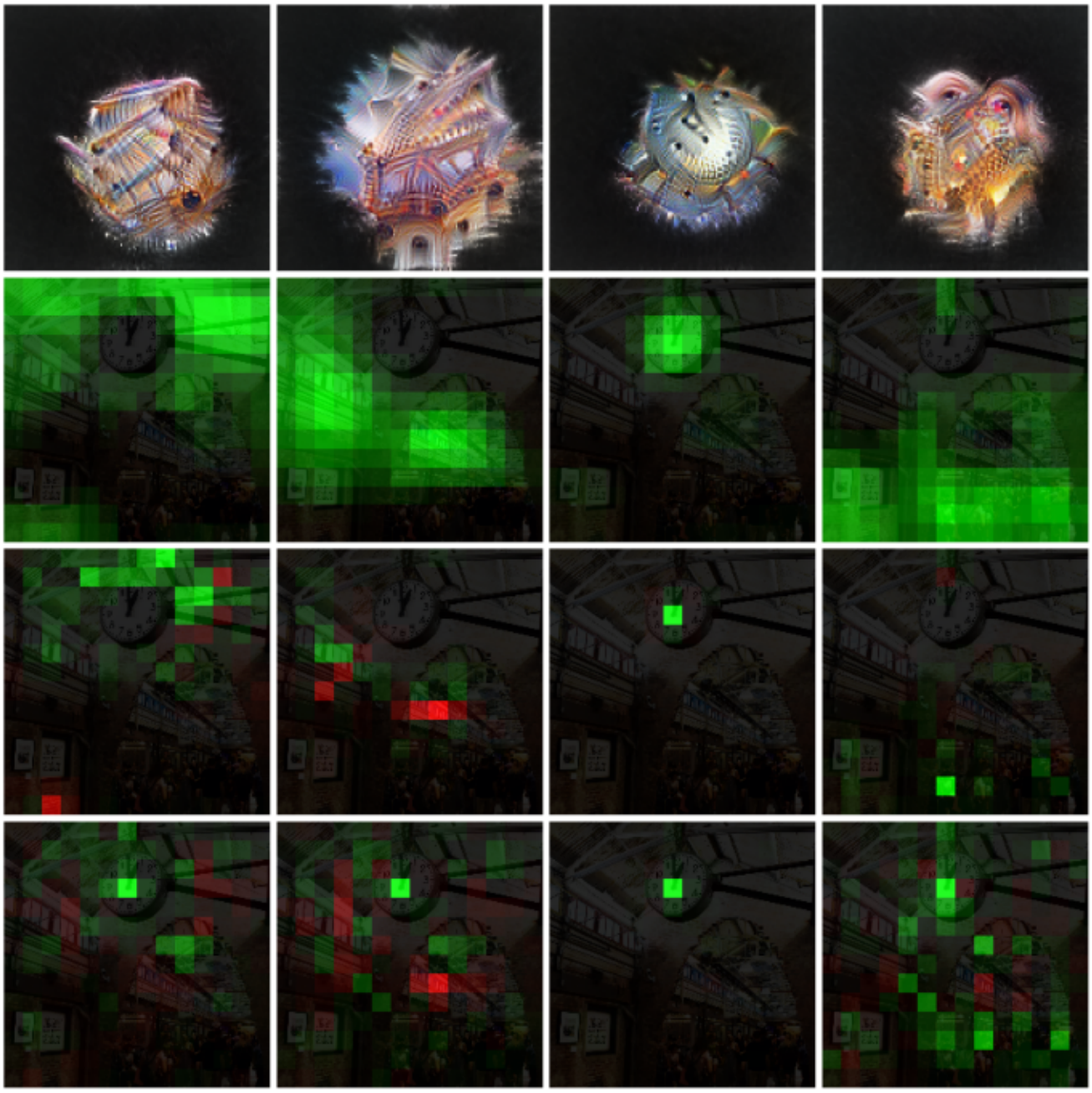


Figure 22: Comparison of LFA and neuron groups' attribution maps for an analog clock target class (predicted class: analog clock, with a 57.87% confidence). **First row:** feature visualizations of the distinct latent factors identified using NMF. **Second row:** corresponding latent spatial factors. **Third row:** LFA attribution maps. **Fourth row:** neuron groups' attribution maps. Green and red areas denote positive and negative influence, respectively. Code used to compute the fourth row of the figure adapted from a Colab notebook published in [49].

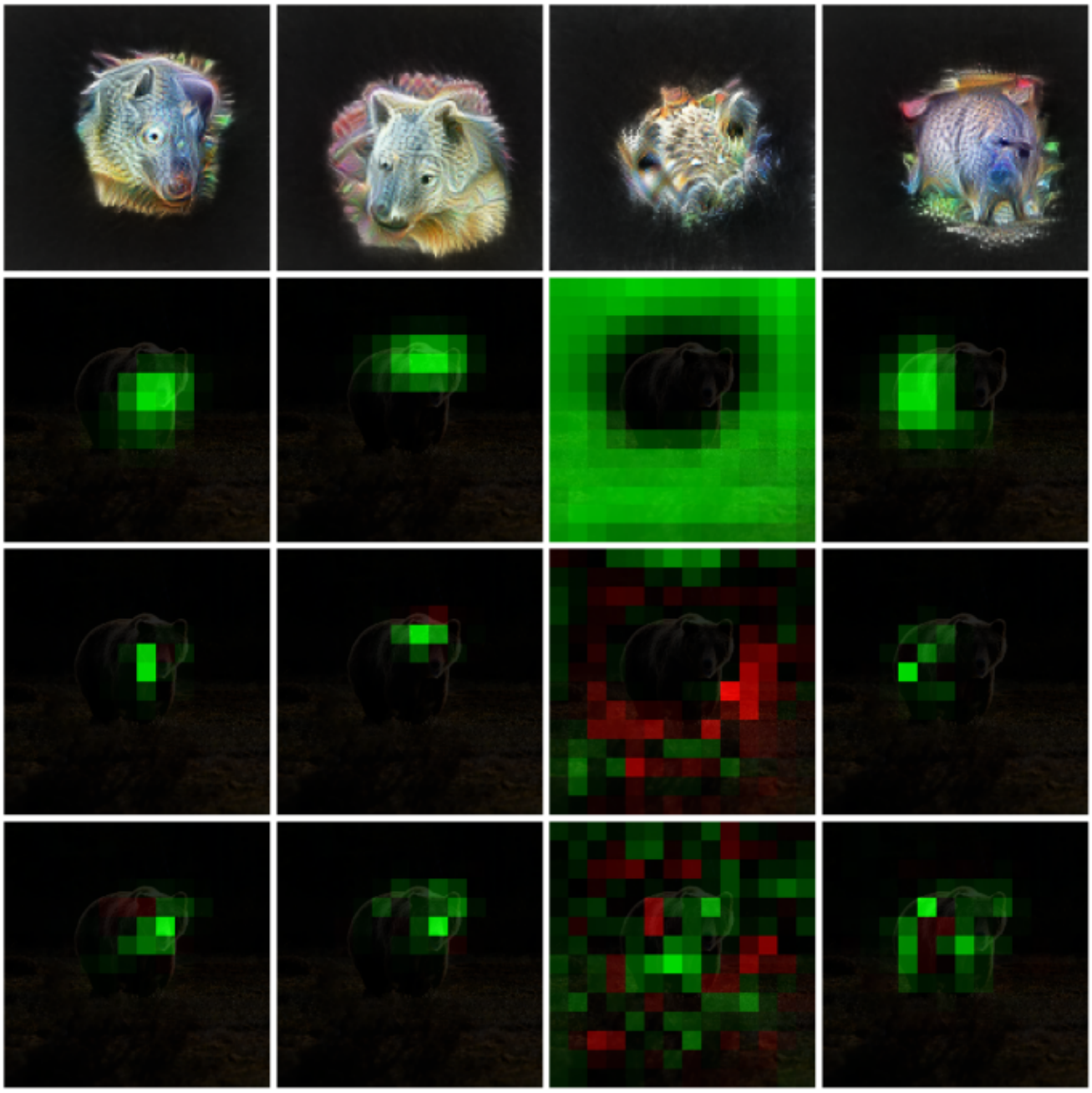


Figure 23: Comparison of LFA and neuron groups' attribution maps for a brown bear target class (predicted class: brown bear, with a 82.98% confidence). **First row:** feature visualizations of the distinct latent factors identified using NMF. **Second row:** corresponding latent spatial factors. **Third row:** LFA attribution maps. **Fourth row:** neuron groups' attribution maps. Green and red areas denote positive and negative influence, respectively. Code used to compute the fourth row of the figure adapted from a Colab notebook published in [49].

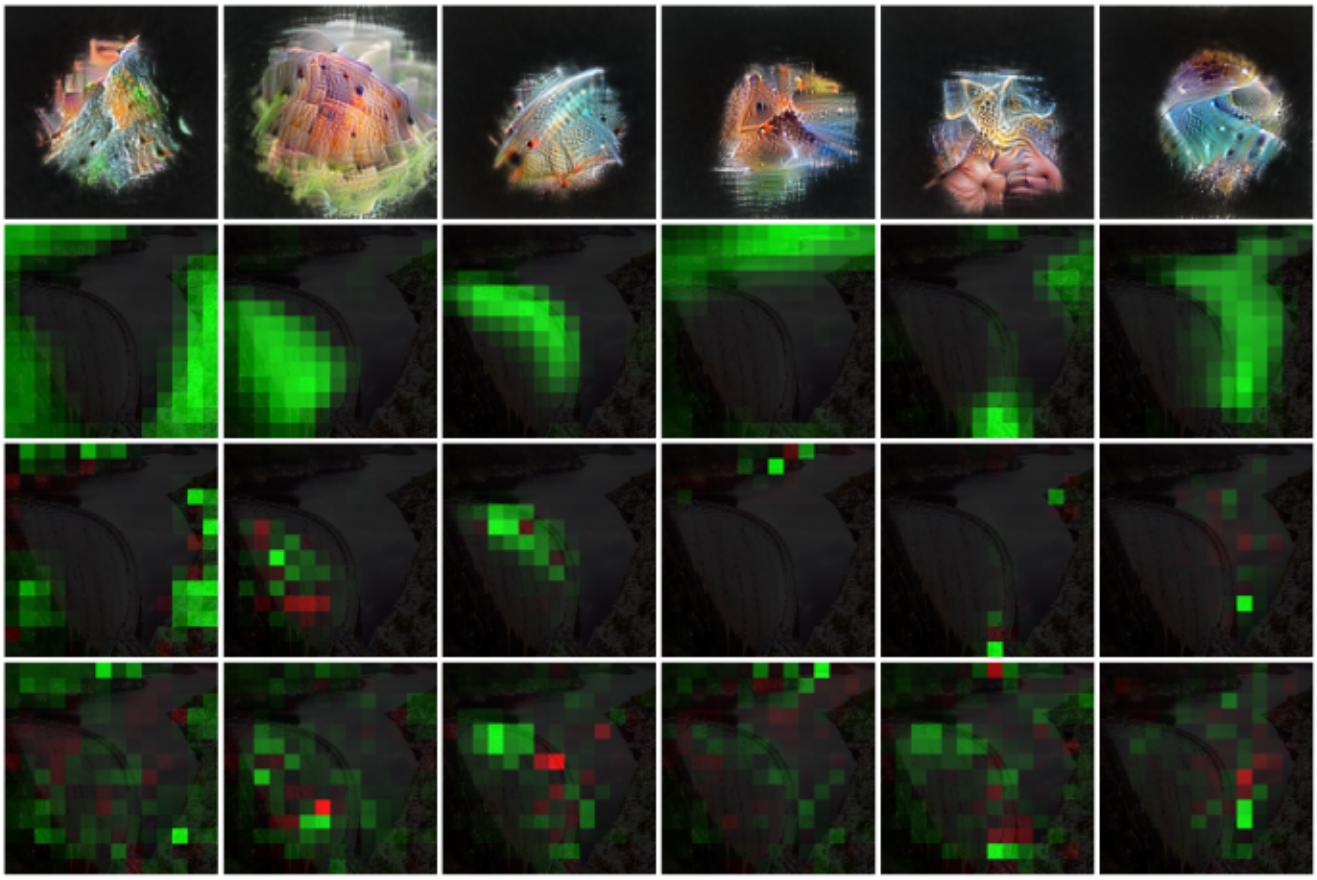


Figure 24: Comparison of LFA and neuron groups' attribution maps for a dam target class (predicted class: dam, with a 84.22% confidence). **First row:** feature visualizations of the distinct latent factors identified using NMF. **Second row:** corresponding latent spatial factors. **Third row:** LFA attribution maps. **Fourth row:** neuron groups' attribution maps. Green and red areas denote positive and negative influence, respectively. Code used to compute the fourth row of the figure adapted from a Colab notebook published in [49].

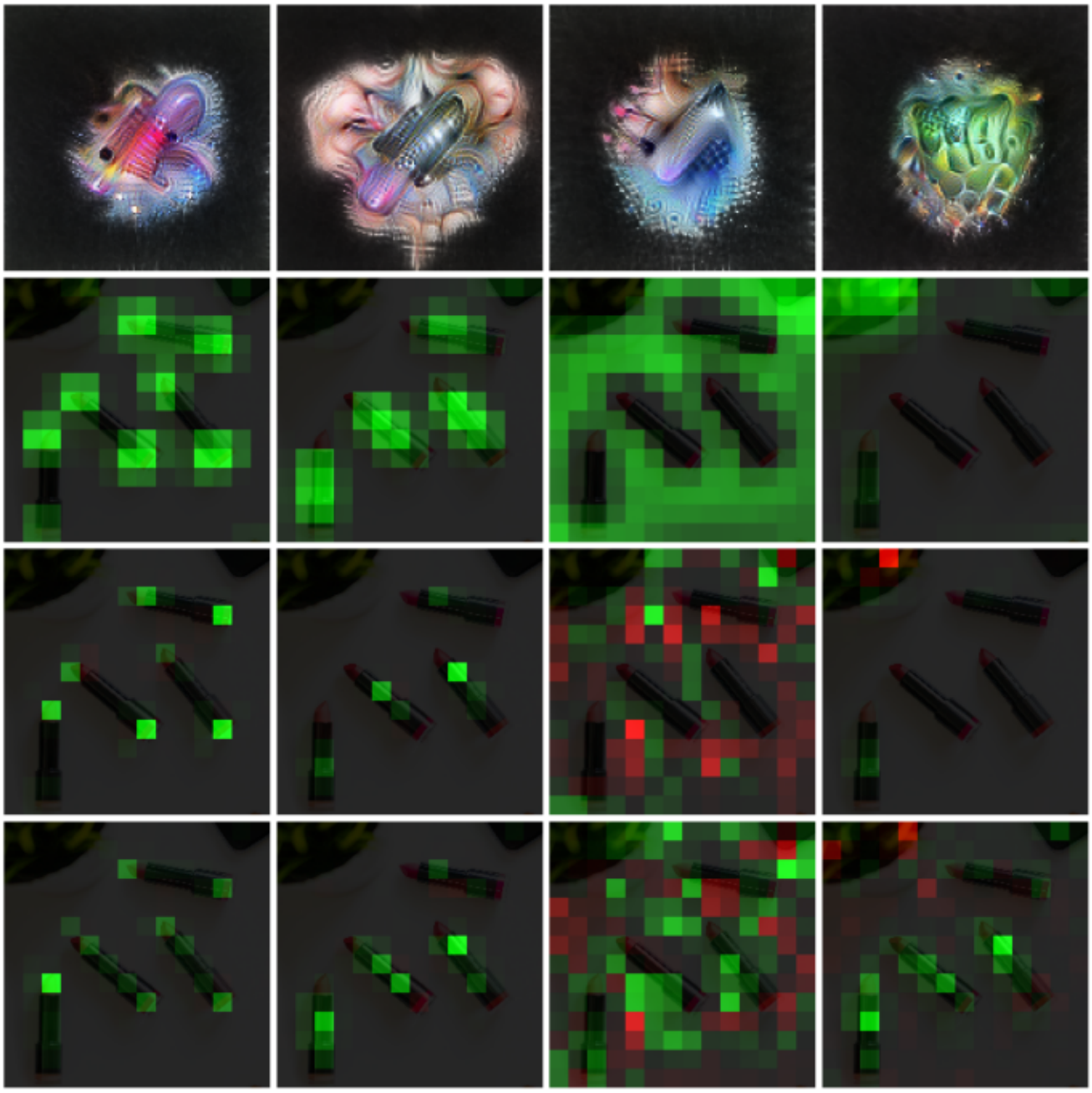


Figure 25: Comparison of LFA and neuron groups' attribution maps for a lipstick target class (predicted class: lipstick, with a 99.61% confidence). **First row:** feature visualizations of the distinct latent factors identified using NMF. **Second row:** corresponding latent spatial factors. **Third row:** LFA attribution maps. **Fourth row:** neuron groups' attribution maps. Green and red areas denote positive and negative influence, respectively. Code used to compute the fourth row of the figure adapted from a Colab notebook published in [49].

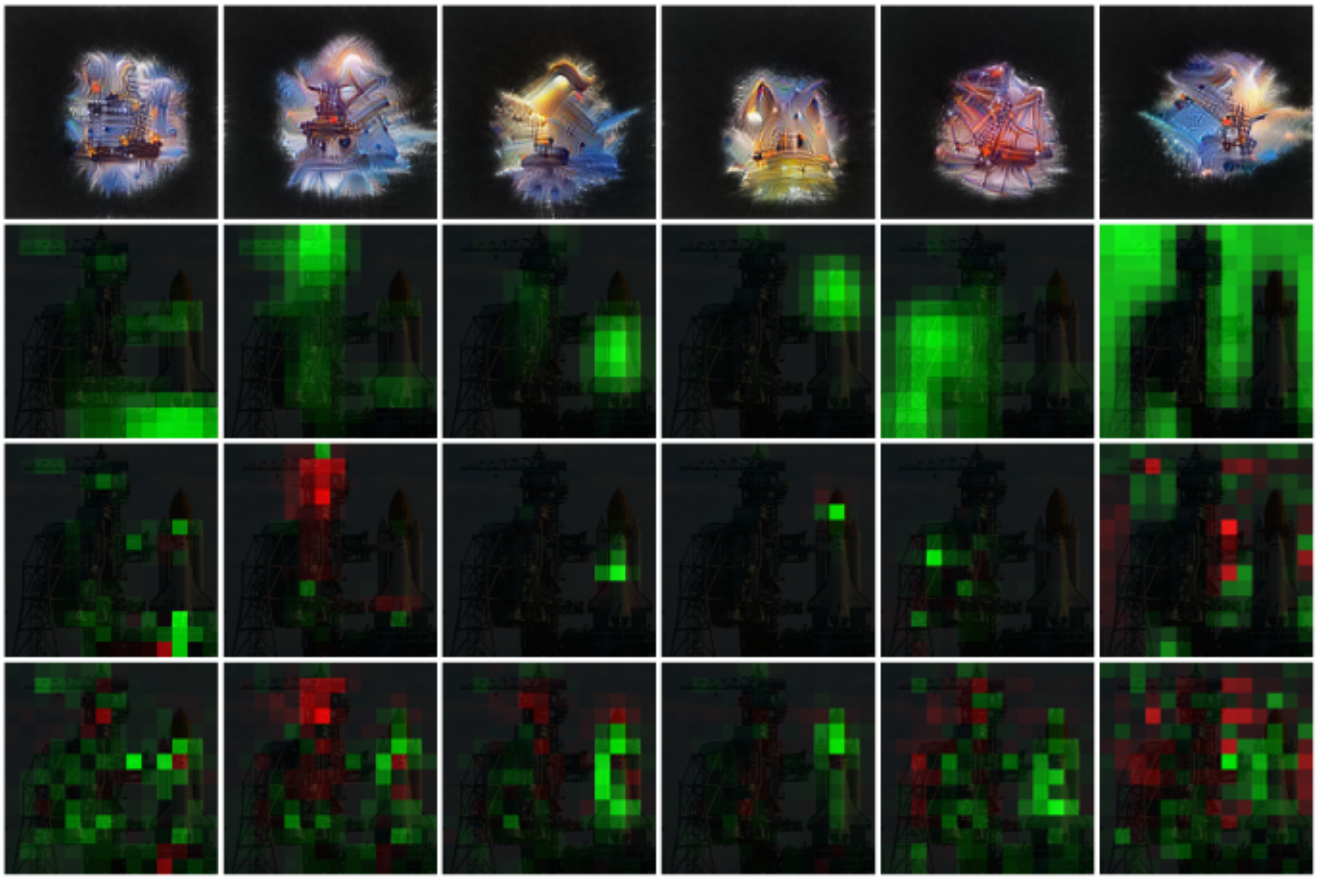


Figure 26: Comparison of LFA and neuron groups' attribution maps for a space shuttle target class (predicted class: space shuttle, with a 99.55% confidence). **First row:** feature visualizations of the distinct latent factors identified using NMF. **Second row:** corresponding latent spatial factors. **Third row:** LFA attribution maps. **Fourth row:** neuron groups' attribution maps. Green and red areas denote positive and negative influence, respectively. Code used to compute the fourth row of the figure adapted from a Colab notebook published in [49].

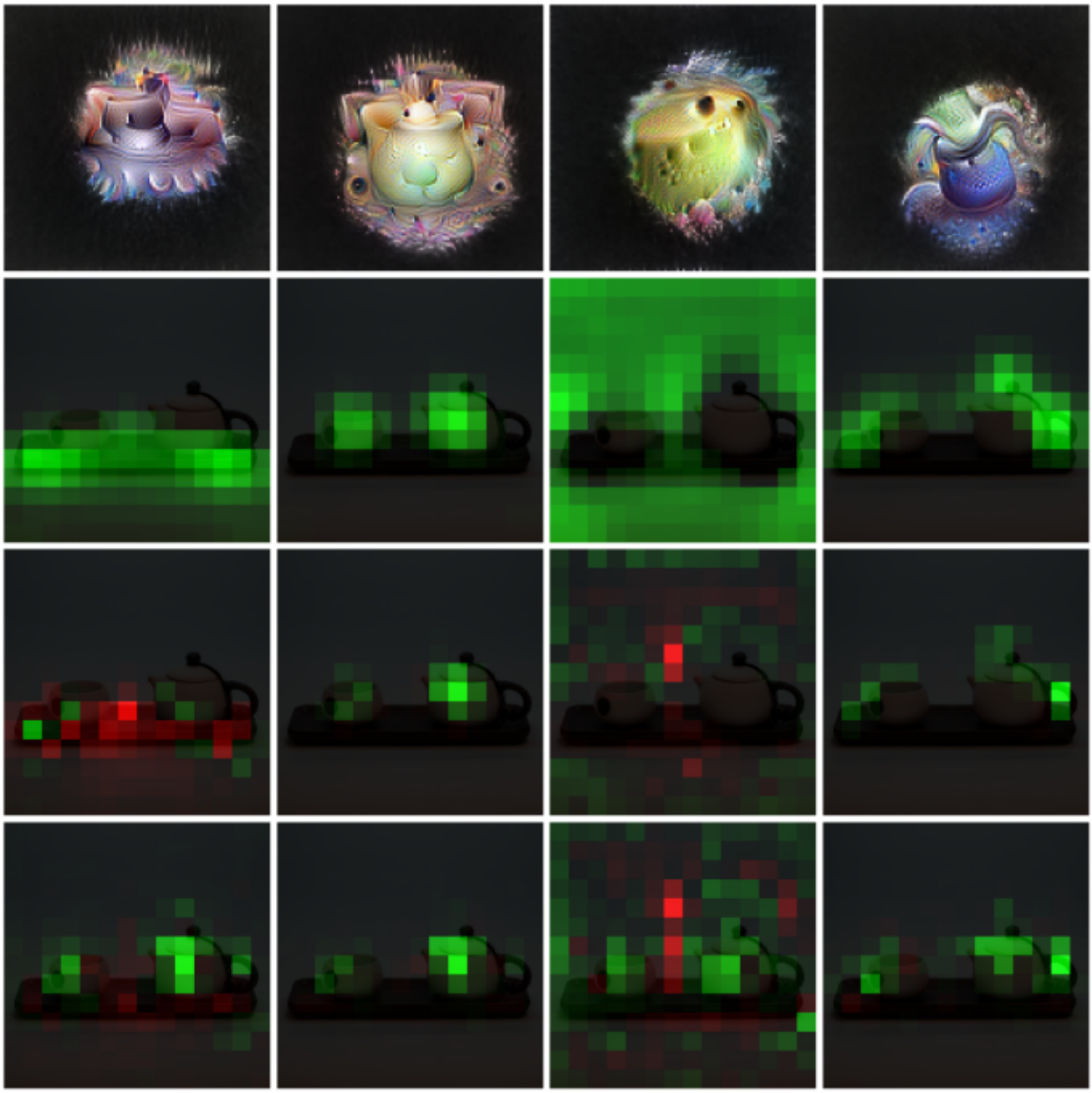


Figure 27: Comparison of LFA and neuron groups' attribution maps for a teapot target class (predicted class: teapot, with a 95.78% confidence). **First row:** feature visualizations of the distinct latent factors identified using NMF. **Second row:** corresponding latent spatial factors. **Third row:** LFA attribution maps. **Fourth row:** neuron groups' attribution maps. Green and red areas denote positive and negative influence, respectively. Code used to compute the fourth row of the figure adapted from a Colab notebook published in [49].

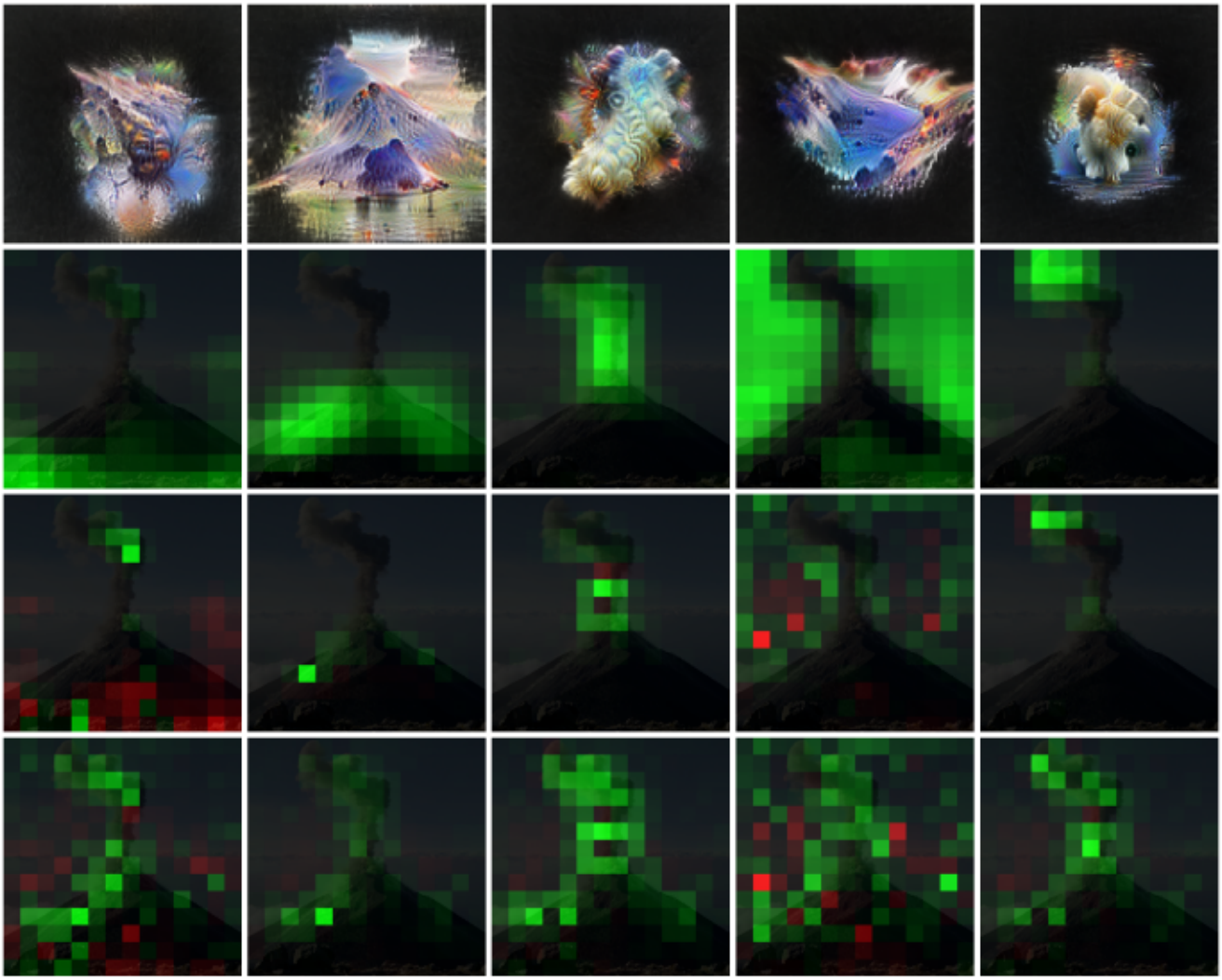


Figure 28: Comparison of LFA and neuron groups' attribution maps for a volcano target class (predicted class: volcano, with a 99.97% confidence). **First row:** feature visualizations of the distinct latent factors identified using NMF. **Second row:** corresponding latent spatial factors. **Third row:** LFA attribution maps. **Fourth row:** neuron groups' attribution maps. Green and red areas denote positive and negative influence, respectively. Code used to compute the fourth row of the figure adapted from a Colab notebook published in [49].

Concept contribution [70] is another technique similar to LFA. At its core lies a custom matrix decomposition—*Interpretable Basis Decomposition*. IBD is a hybrid of NMF (positive latent spatial factors) and PCA (gradual creation of decomposition basis). Here, the actual attribution estimation is very similar to one of the neuron groups: first, latent spatial factors are multiplied element-wise with the corresponding feature vectors to get the equivalent of group tensors. Next,

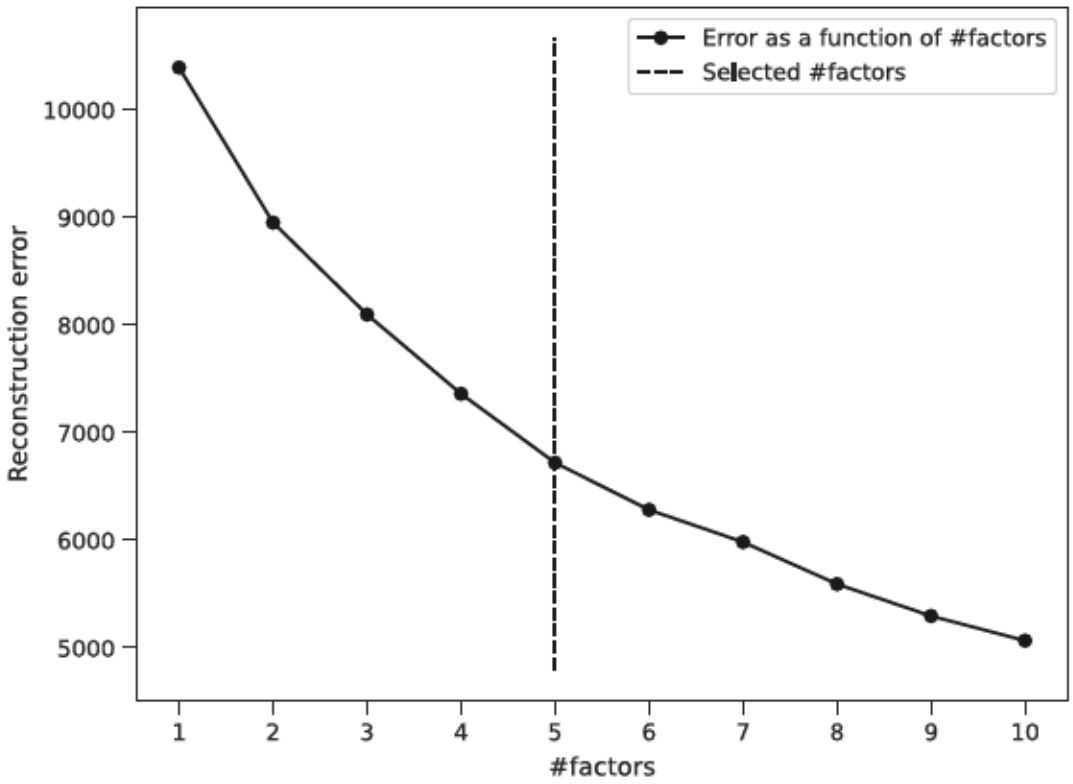


Figure 29: Estimation of the number of factors in the NMF matrix decomposition. Multiple runs with varying number of factors were carried out and the matrix reconstruction error was then plotted. From such, an "elbow" was identified —a point where the function begins to flatten.

"group tensors" are multiplied element-wise with the activation tensor, yielding concept contribution. As IBD is tightly bound with the Grad-CAM heatmap computation (designed to decompose the activation tensor of the final convolutional layer of a CNN of interest), it is not suitable for computing intermediate layers' attribution maps. Consequently, it has been excluded from the visual comparisons presented in the aforementioned web interface.

3.6 Determining the Number of Factors

Having looked at Figure 17 and Figure 18, a question may arise: why are there exactly 5 latent factors presented in each example? Is the number of factors for NMF and PCA always equal? The fact that these numbers are the same for both NMF and PCA is a coincidence—the number of factors is determined individually for each image and depends on the decomposition technique.

For PCA, it is by definition possible to determine the importance of any latent factor—each has the explained variance ratio value assigned to it that indicates how much of total variance in

the decomposed matrix is represented by this particular factor. Explained variance ratio values range from 0 (exclusive—the existence of a factor with zero explained variance would not make sense) to the value of 1 (entire variance explained by a single factor). For the experiments in this thesis, a threshold for the total summed explained variance ratio has been set to be at least 0.7 (all the factors, starting from the most explanatory one, are gradually added to the pool until their total explained variance ratio reaches 0.7).

In the NMF case, the problem of factor selection is non-tractable—there does not exist a straightforward technique to determine how many latent factors should the matrix be "optimally" decomposed to. Instead, a novel, relatively robust estimate was proposed in this thesis. It makes it possible to automatically determine the number of latent factors in the NMF procedure by analyzing how the matrix reconstruction error changes as a function of a number of components used for the decomposition. As an example, to generate the data for Figure 29, NMF decomposition has been run multiple times (gradually increasing the number of components from 1 to 10). In each iteration, a matrix reconstruction error was recorded to eventually determine a reasonable number of factors to use by identifying an "elbow"—curve flattening point.

3.7 Activation Explorer

The author of this dissertation believes that the interpretability field exists for two reasons:

1. It serves as a torch that is used to lighten the dark nooks and crannies of black-box neural networks and thus, to some extent, explain what makes the models produce specific outputs.

2. It helps humans develop new intuitions about the inner data representations that are constructed in these models during their training procedures.

Sometimes it is the bare numbers that researchers need. Sometimes, though, it is an interactive interface. In this work, since LFA was designed with image data in mind and that the experiments yielded the results in a quantity too high to navigate through with tables or simple grids of images, a clear web application was designed. It facilitates data exploration and enables interactive controls over the available parameters. Not only does it make a side-by-side comparison of LFA and neuron groups' attribution possible but, most importantly, it puts LFA in the functional context. The interface is called *Activation Explorer* and is available for the public at the following URL: `https://bmiselis.github.io/explorer.html`.

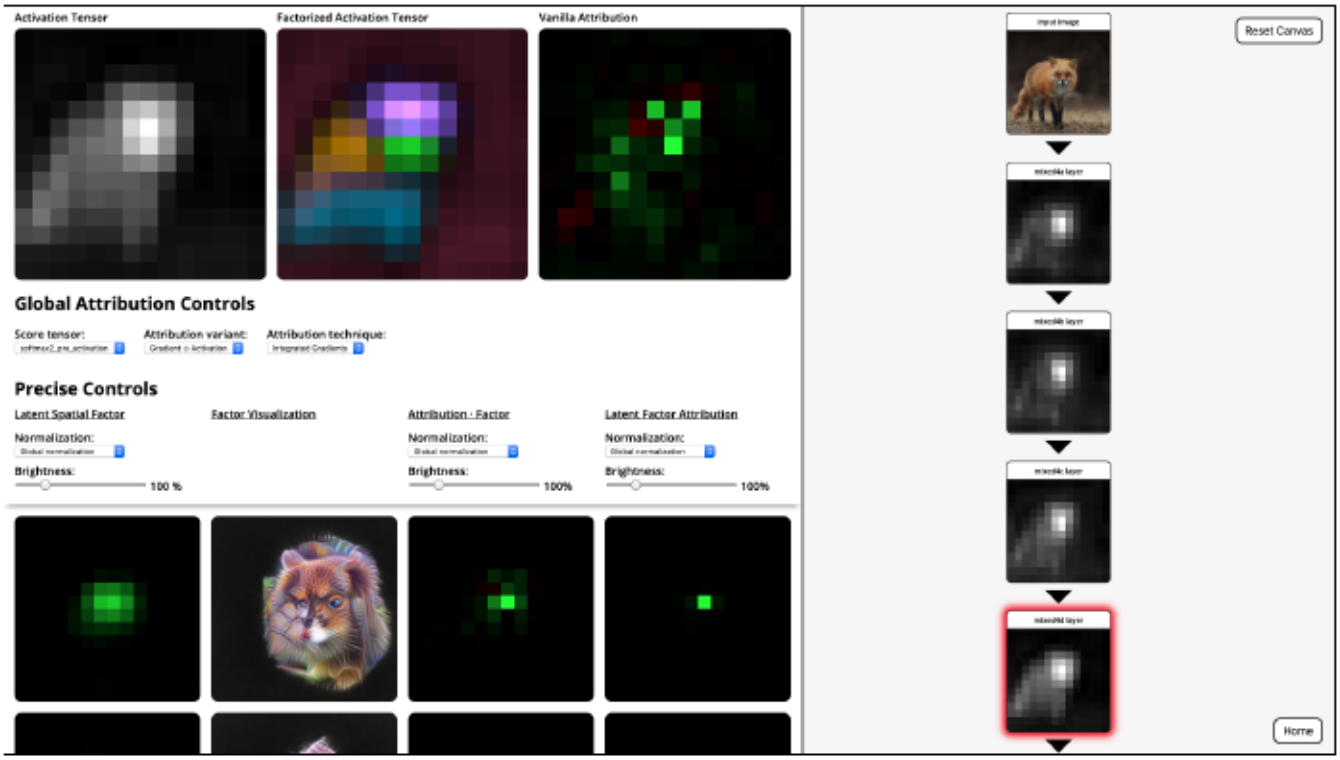


Figure 30: Activation Explorer is a supplementary web interface that was designed to enable convenient overview of the results generated by the experiments presented in this chapter.

An overview of the interface will begin with the default pane (see Figure 31)—a zoomable and draggable panel that allows the user to inspect basic visualizations in greater detail. It presents how the activation tensors evolve in the consecutive inception modules (starting from the mixed4a one; see the current chapter's introduction for justification on this). Each 2D activation map was obtained by applying a sum reduction over the channel dimension of the activation tensor $\mathbf{H}$. Target class and the image sample can be picked by the user after clicking on the input image at the top of the graph. The data for the following classes is available: *red fox*, *giant panda*, *brown bear*, *analog clock*, *dam*, *lipstick*, *space shuttle*, *teapot* and *volcano*. Three sample images were chosen for each class from the , ensuring that the predictions match the ground-truth.

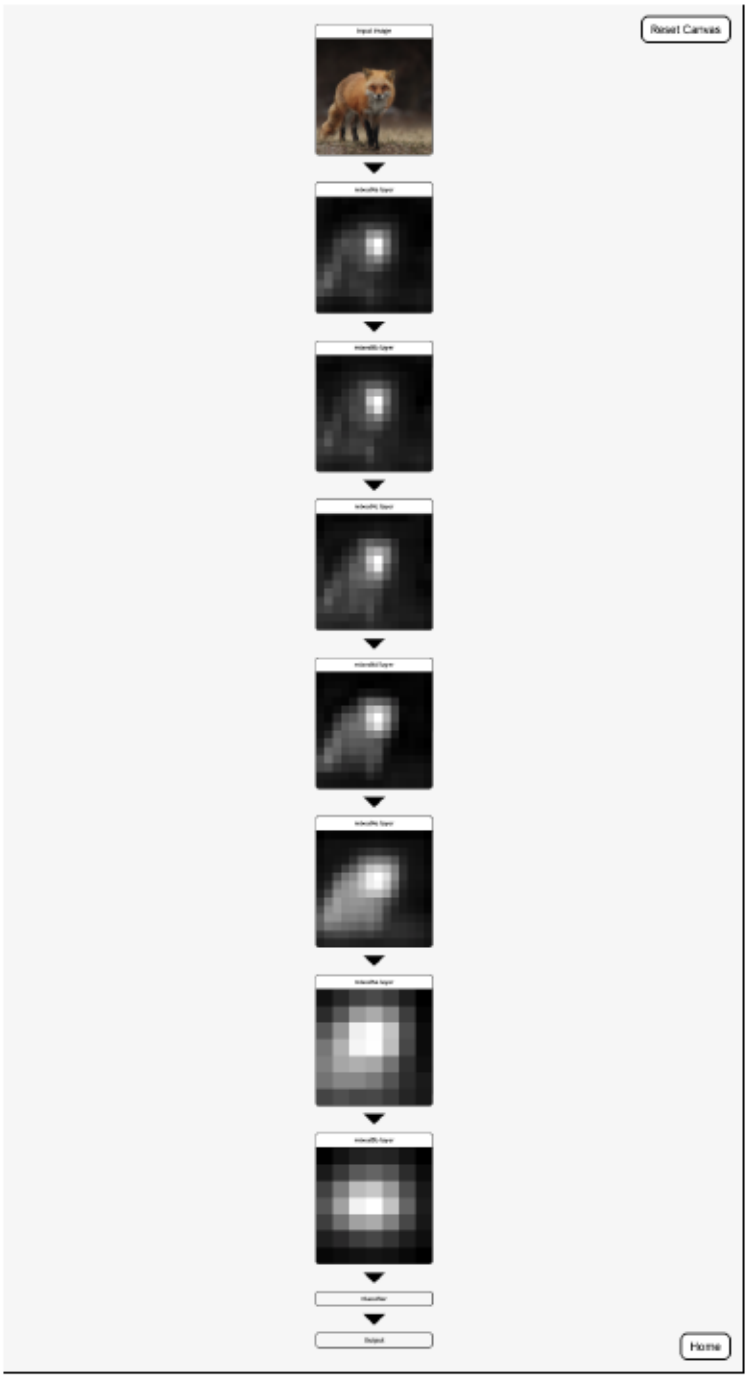


Figure 31: Default view of the Activation Explorer interface. It presents how the activation tensors evolve in the consecutive inception modules (starting from the mixed4a one; see the current chapter's introduction for justification on this). Each 2D activation map was obtained by applying a sum reduction over the channel dimension of the activation tensor **H**. Target class and the image sample can be picked by the user after clicking on the input image at the top of the graph.

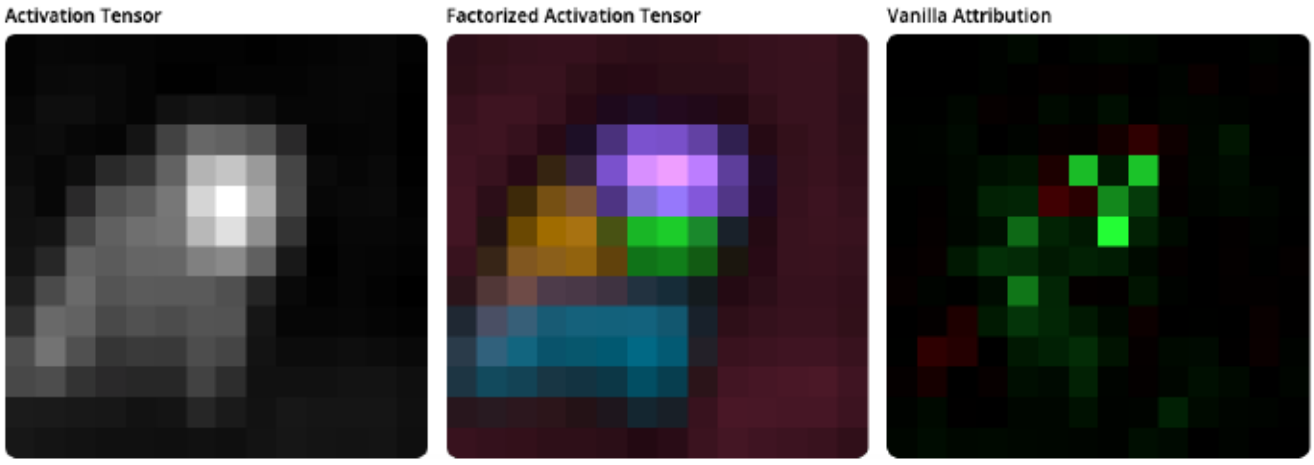


Figure 32: A snapshot of the *Overview* section, extended view panel. **Left:** sum-reduced activation tensor. **Middle:** latent spatial factors from NMF, overlaid. **Right:** vanilla attribution map.

After clicking on an arbitrary 2D grayscale image of any activation tensor, the *extended view* panel is presented to the user (visible in the left-hand side of Figure 30). It is the core element of the interface that lets the user explore how the information flowing through the CNN is represented at the layer of interest. The extended view panel is divided into three sections: *Overview*, *Controls* and a scrollable grid of detailed results of the decomposed activation matrix $\boldsymbol{H}$.

The *Overview* section provides background information, required to comprehend the forthcoming visualizations: sum-reduced activation tensor $\mathbf{H}$, factorized activation tensor (overlaid latent spatial factors $\boldsymbol{S}$) and the vanilla attribution map. Please refer to Figure 32 for an example. It is worth mentioning that the latent spatial factors in the factorized activation tensor image are colored randomly when NMF decomposition is used—all values are by definition positive so there is no need to differentiate them with "green" and "red" to indicate positive and negative areas in latent spatial factors. For PCA, though, the green-red coloring has been used to make it clear which areas depict positive presence and which negative presence of each factor.

After the *Overview*, there is the *Controls* section. It is split into the controls that manipulate global behavior (applicable to all attribution data in the panel) and the controls that enable precise configuration of each column of a grid further down individually:

- *Score tensor* lets the user choose the node in the computational graph that will be used in the attribution calculation. It is either *softmax2_pre_activation* (**before** softmax activation) or *softmax2* (**after** softmax activation).
- *Attribution technique* has two options: gradient or Integrated Gradients.

- *Attribution variant* allows to select either gradient-based or the gradient $\odot$ activation-based attribution calculation.

- *Normalization* gives the user a choice on whether the heatmaps are normalized per-factor (each heatmap is normalized individually so that its minimum value is 0 and maximum is 255) or globally (a minimum value across all heatmaps is 0 and maximum is 255). Please note that if the *global normalization* option is picked, it gives the user a chance to see the absolute importance of each factor, maintaining the relative differences between the individual heatmaps. Per-factor normalization may be helpful when conducting an individual analysis of each factor's importance, yet should not be used when building holistic intuitions.

- *Brightness* is a slider that controls how dim/bright the pixels in the column below it are. The intention is to let the user make the vague areas pop up for a more fine-grained analysis, whenever needed.

The last and yet the most important element of the extended view panel is the grid of distinct visualizations of the data resulting from the matrix decomposition of the activation tensor $\mathbf{H}$. Here, each row is dedicated to a single factor. In the case of PCA, the rows are sorted so that the explained variance ratio of the consecutive factors decreases (hovering over the latent spatial factor reveals the precise value of the explained variance ratio). However, NMF decomposition does not provide any information on the factors' importance, so the rows in this situation are sorted randomly.

The first column contains the latent spatial factors. They are presented to help the user identify what input stimuli the factor is responding to if the factor visualization is ambiguous—information on "where" the concept is located is often helpful in comprehending the underlying abstraction. The second column includes factor visualizations generated using the knowledge acquired from a series of Distill articles [12, 42, 48, 49]. All the parameters that are essential to reproduce the results from this column have been summarized in section 2.4.3. The third and fourth columns contain the neuron groups' attribution maps (denoted as "Factor · Attribution") and the LFA attribution maps, respectively. Here, the user can visually compare both methods.

3.8 Discussion and Limitations

LFA is a mid-level attribution technique that operates on the abstractions produced by the hidden layers of a CNN. It provides a holistic view on the latent representation of the intermediate tensors

in the visual processing of the model, yet does not enable a global view on CNN's behavior in the current form. Consequently, the forthcoming discussion refers to per-image observations. Nonetheless, the observed repetitive patterns were mentioned where possible.

The main question to be answered in this chapter was whether LFA yields the results more insightful than the ones of the existing techniques. From the examples presented in Figures 20 to 28, the initial hypothesis was that it is indeed the case: the LFA produces sparser attribution maps that are more coherent with the underlying latent spatial factors than the ones obtained using neuron groups' attribution—LFA attribution maps appear to be more "concentrated" around the target concept and isolate the per-factor attribution information better than their counterpart. It should not come as a surprise since LFA attribution is tightly bound with the underlying latent factors, much more than the one of neuron groups. The strong correlation of LFA with corresponding latent spatial factors, together with the fact that it reacts well to global normalization is what makes it beneficial for the fine-grained analysis of factors' attribution.

In the *Controls* section of the Activation Explorer interface, multiple parameters enable extensive exploration of the data from the experiments by various intuition-building interactions. The **score** option allows the user to choose between pre-softmax and after-softmax tensor w.r.t. which the gradient is computed. The pre-softmax tensor (known as *logit*) can be seen as a "class score". After applying softmax, the resulting value is rather a probability of a class given the class score (*class posterior*) [48]. Because the class posterior can be maximized by minimizing other class' posteriors (other class' posteriors are in the denominator of the softmax equation), it is beneficial to choose a pre-softmax tensor to concentrate the optimization procedure only on the target class [57] (each score is independent of the others in such a scenario). When interacting with the **attribution variant**, the users are encouraged to begin with the *Gradient* option and then switch to *Gradient* $\odot$ *Activation*, while carefully observing what happens to the attribution maps (both for neuron groups and LFA). What can be observed is that the element-wise multiplication with the activation tensor is equivalent to "having a spotlight" that "shines more light" on the areas of the gradient that influence the class score more, at the same time "casting a shadow" on the insignificant ones. When it comes to the **attribution technique**, the differences in the resulting maps are not that prominent. The main distinction is that the Integrated Gradients technique tends to produce less noisy heatmaps than the vanilla gradient approach.

All of the parameters mentioned in the preceding paragraph have a noticeable impact on the resulting visualizations. However, the dominant changes are introduced by altering the matrix decomposition technique. Primary dissimilarity comes from the fact that PCA yields latent factors that are much harder to comprehend with human intuitions than the ones obtained with NMF—the

PCA's assumption that the concept can be "positive" in some areas of the image and "negative" in the others is unintuitive and impedes the reasoning about the meaning of a particular abstraction. For instance, while a concept of a "snout" is comprehensible, what would a "negative snout" mean?

After examining multiple image samples, the impact of LFA is quickly noticeable: it helps to navigate through the multitude of factors, especially when investigating PCA decomposition in the "mixed4*" layers. The interpretation of the abstractions they represent is especially hard, though, since the underlying concepts are rather low- to mid-level. In this case, NMF yields much more spatially coherent and, most importantly for the sake of interpretability, all-positive maps, being a method of choice for this part of the InceptionV1. In the mixed5b activation tensors, however, PCA tends to robustly identify factors of very high importance (their explained variance ratios are usually above 0.5 for a single factor). It appears to be able to decompose the underlying representation into a concise set of abstractions, perceptibly more compelling than the ones obtained with NMF. It cannot be said for sure whether such limitation of NMF is because the wrong number of factors were picked by the "elbow-locator" method introduced in section 3.6, or whether it is an imperfection of the decomposition method itself.

The impact LFA has on the interpretability of the factors' attribution maps becomes genuinely noticeable in the last two layers of the InceptionV1: mixed5a and mixed5b. The most interesting results can be observed when the gradient $\odot$ activation variant is used—vanilla gradient-based visualizations collapse into a plain-colored square for the mixed5b layer, making their analysis futile in this case. The collapse is the result of backpropagating through the global average pooling layer, located close to the end of the InceptionV1 model—the resulting gradient has the same average value distributed across all spatial locations[3]. That problem does not occur for the mixed5a layer since the computation includes backpropagating through the entire mixed5b layer as well. For both layers, LFA yields results that are less correlated with the vanilla attribution map than the ones obtained with the neuron groups' attribution technique. The reduced correlation is a positive feature, for the vanilla attribution maps tend to be very diffused in mixed5a and mixed5b. Their scattered nature makes neuron groups' attribution even more spread and harder to comprehend.

The final interesting feature of LFA that can be noticed when analyzing the attribution maps from the mixed5b layer (using gradient $\odot$ activation variant) is that each heatmap is merely a latent spatial factor, uniformly scaled by the gradient value. The scaling can be discarded by applying per-factor normalization, though, to reveal an interesting fact: for the mixed5b layer, the LFA maps are equivalent to the corresponding latent spatial factors.

[3]One might notice that in some cases the vanilla attribution map is plain-red, while the neuron groups' attribution maps and the LFA ones are plain-green. This is both fine and expected—multiplying a uniform gradient tensor with the heterogeneous activation tensor has a strong influence on the channel-reduced attribution maps.

Chapter 4

Distilled Class Factors Atlas

Experiments from the previous chapter indicate that latent factors are an interesting unit of interpretability to study CNNs. Not only do they provide higher-level abstractions to think about hidden data representations of the algorithms, but they also reduce the overwhelming amount of available information, scaling it down for feasible human analysis. The natural question that arises while reflecting on this topic is whether matrix decomposition could be applied analogously, but to yield a much more complex interface than just the LFA attribution maps?

Extending the research scope from image-specific to model-specific analyses is a non-trivial objective. Projects like Network Dissection [8], CNN Codes [27] or Activation Atlas [12] provided invaluable insights on how to work with vast, complex manifolds that CNNs construct. These rich, intuitive interfaces were the inspiration for the research presented in this chapter. Here, a second novel interface—Distilled Class Factors Atlas—is introduced. It unifies the idea of Class Activation Atlases [12] with matrix decomposition and LFA to yield a specialized form of the atlas that enables in-depth analyses of concepts that are repetitively used by a CNN to predict a given class. A word *distill* stands for "extracting the essential meaning or most important aspects of". It was used to indicate the process of filtering out the factors whose attribution towards a class of interest is low (or even negative) while keeping only the ones that support the prediction.

The closest work that resembles Distilled Class Factors Atlas is the Class Activation Atlas [12]. In the original method, 1 million randomly selected feature columns[1], collected from all ImageNet images, were filtered out to keep only the ones that attribute the most to the target class. The core difference is that Distilled Class Factors Atlas does not select columns randomly—it utilizes matrix decomposition to identify factors specific to a given class. Moreover, it uses the LFA to estimate concepts' attribution, instead of the gradient $\odot$ input from the preceding research.

[1]The terms *feature column*, *activation vector* and *linear combination of channels* are equivalent.

4.1 Humans Do Not Scale

In principle, CNNs are exceptionally large when compared with other machine learning algorithms. Their number of parameters often exceeds 5-10 million, reaching hundreds of millions in some cases [26]. Even though interpretability researchers typically study structures more abstract than raw weights, they still keep the amount of other potential units of interpretability (neurons, channels, feature columns) on an overwhelmingly high level. As an example, the familiar InceptionV1 CNN has been overviewed in the next paragraph.

InceptionV1 [61] is a 22-layer CNN with 6.7977 million parameters, which makes it a sizable deep neural network (see Figure 34 for a detailed overview of its architecture). The most interesting parts of the model are the inception modules: mixed3a, mixed3b, mixed4a, mixed4b, mixed4c, mixed4d, mixed4e, mixed5a, and mixed5b, described in details in the introduction of the chapter 3. As mentioned previously, the output tensors of these modules are high dimensional. For instance, if the input tensor of the InceptionV1 is of shape (224, 224, 3), then the output of the mixed4d module is of shape (14, 14, 528). 528 channels are a lot, and each of them denotes a unique filter, sensitive to a more or less sophisticated abstraction. Not to mention the fact that for mixed4d output tensor, there are 196 neurons in a single channel, potentially detecting fine-grained variants of the same concept w.r.t. the spatial location.

To analyze all of the channels of the mixed4d module at the same time, a spritemap with feature visualizations can be generated. An example of such can be seen in Figure 33. The first thing that catches the eye of the observer is the number of icons (528) and the variety of concepts they represent. The analysis of spritemaps offers a significant benefit by bringing closer the understanding of the model as-is, conditioned only on its weights, not the concrete input tensor, making it a more general interpretability approach. There are, however, two major issues with it: 1) the amount of data to examine, even for just a single layer, begins to be overwhelmingly high and 2) the icons are presented in a strongly unstructured way (by the order the corresponding convolutional filters appear in a given layer of a CNN). While the first problem is rather indisputable, this is not the case for the second one. After all, how could the abstractions that channels detect be grouped to provide additional insights about behavioral patterns of a CNN?

It turns out there exist techniques that extract additional information from the manifold constructed by a CNN and then use that information to span an intuitive 2D "scaffolding" that clusters the available visualizations in a compelling, explanatory way. Activation Atlas [12] is one of such techniques—that is the main reason it was used in this chapter as a base for designing more specialized interfaces, like the aforementioned Distilled Class Factors Atlas.

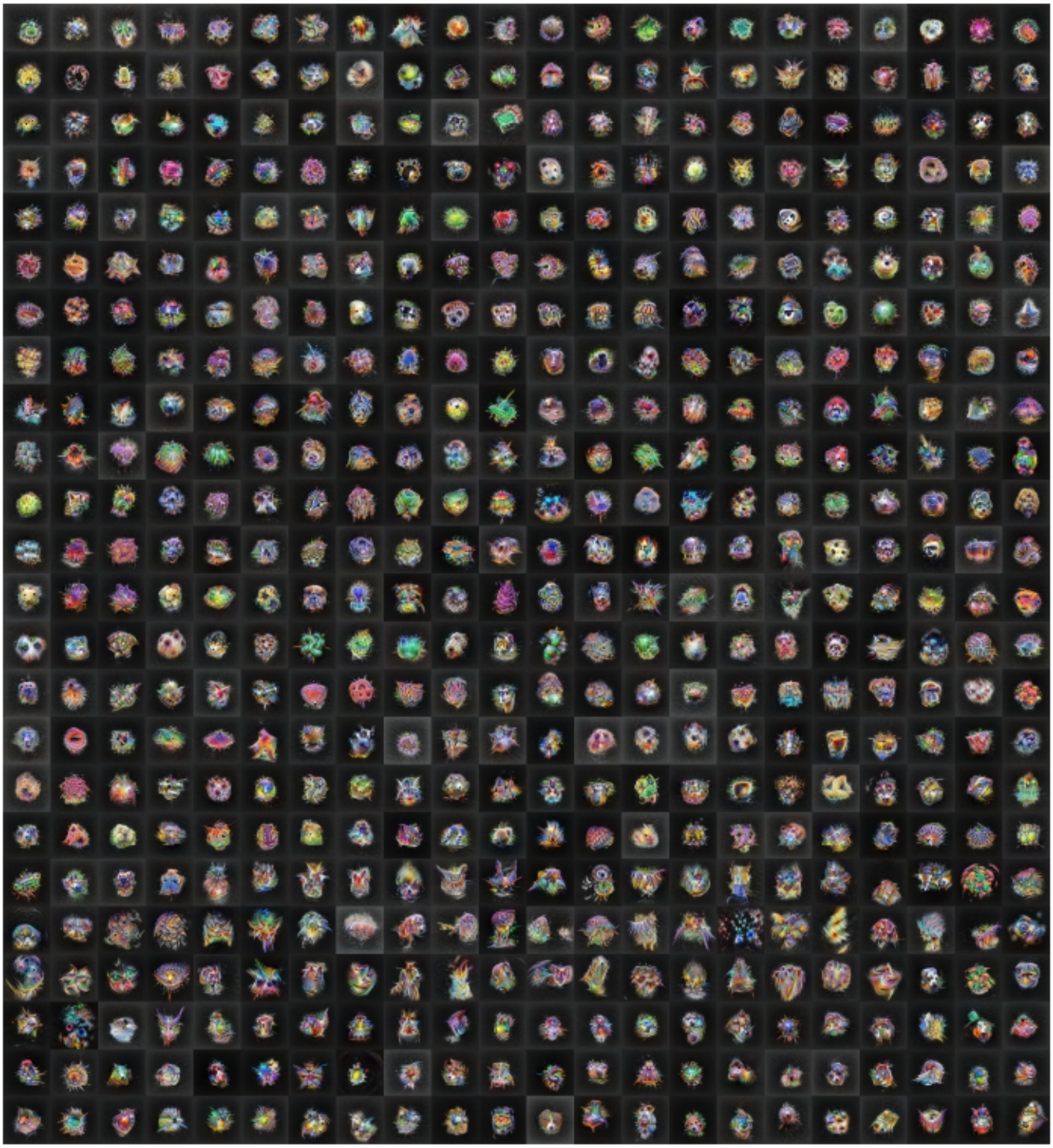


Figure 33: A spritemap with icons from all the channels of the mixed4d layer of the InceptionV1 model. The visualizations are presented in the order the corresponding convolutional filters appear in the layer. Best seen digitally.

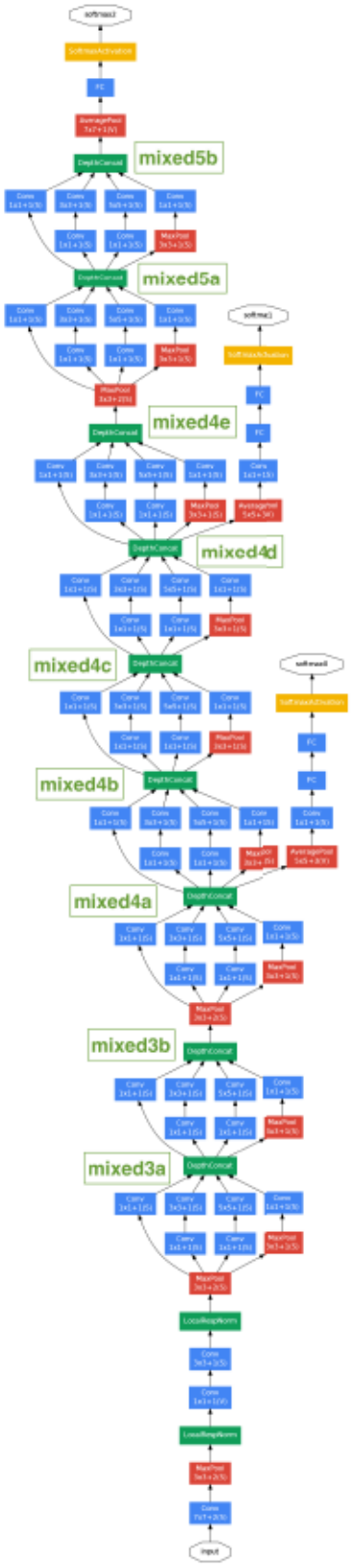


Figure 34: InceptionV1 architecture (best seen digitally). Figure retrieved from [61], with the green-text annotations added by the author of this thesis for clarity.

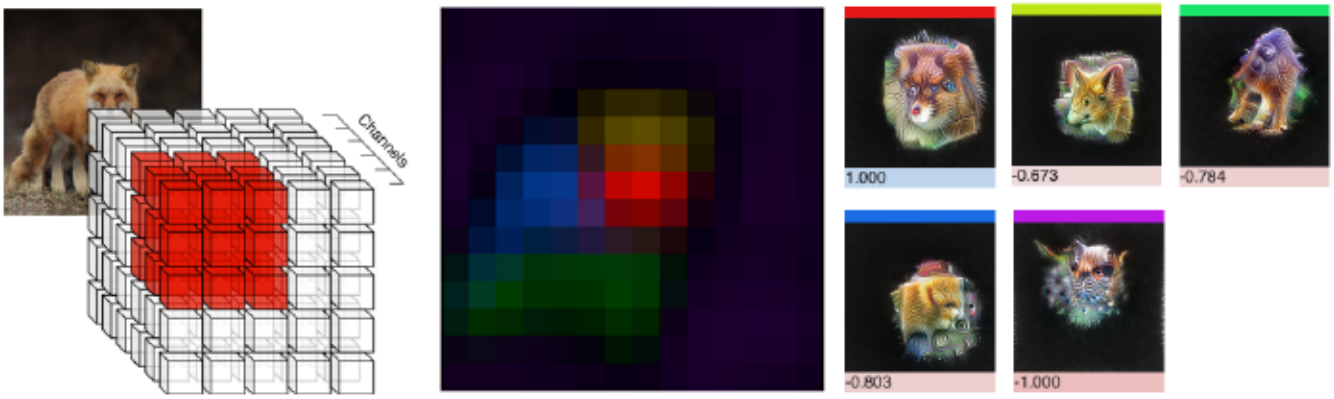


Figure 35: An example of image-specific model analysis technique: neuron groups. Code used to generate the figure adapted from a Colab notebook published in [49].

On the other end of the spectrum lie image-specific techniques. An example from Figure 35 has already been presented in chapter 2. Earlier, it was provided to offer an interesting perspective on the latent representation of a CNN. In this section, however, where the focus is put on the manifold-centric methods, neuron groups are revisited to undermine their applicability as an interface to understand the patterns in the functioning of an algorithm as a whole. To justify this claim, consider the following example: there are 1851 images of a *red fox* class in the ImageNet dataset. For each image, there are, on average, 6–8 reasonable latent factors to decompose the activation tensor to[2]. It would be almost impossible to manually inspect all neuron groups generated for every image for even a single layer, not to mention the fact that there are 22 layers in the model under discussion. Moreover, please keep in mind that there are 1 000 classes (!) in the aforementioned dataset. Roughly speaking, there would be around 160-200 million neuron groups to examine. Such a methodology is a dead-end and requires an alternative solution.

Even though it is usually not possible to draw meaningful conclusions about the general behavior of a given CNN from image-specific analyses, they are still valuable and help researchers develop better intuitions about the model's response for each case separately. The kind of insights they provide is invaluable when, for instance, the objective is to make a diagnosis from a single image (or a small set of related data samples). In such context, extracting case-specific features, together with an in-depth description of what has been identified, is an essential element of the visual processing stack. To summarize: image-specific analyses are useful from the perspective of an end-user, while the model-specific ones are valuable from the researcher's perspective.

[2]This is merely an educated guess, based on the experience from conducting the experiments on multiple layers of the InceptionV1 model.

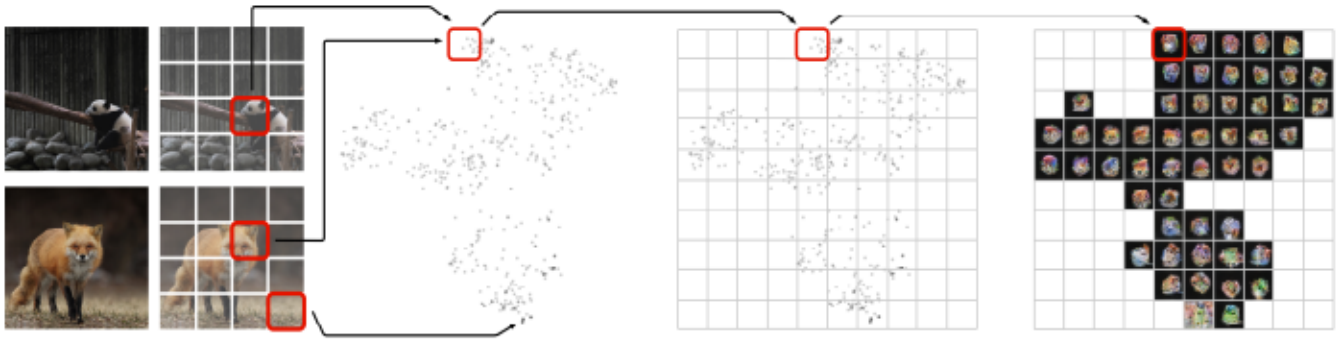


Figure 36: Sequential steps to generate an original Activation Atlas. 1. Feed all the available images to a CNN and collect activation tensors from the layer of interest. 2. Select feature columns for further steps (for every activation tensor it is either all of them, or a single, randomly selected one [12]). 3. Use UMAP or t-SNE to project high dimensional feature columns down to a 2D space. 4. Divide the 2-dimensional layout into a grid of cells. 5. Compute an average feature column for every cell and generate feature visualizations to depict the underlying concepts. Figure and the UMAP projection code adapted from [12] (and associated Colab notebooks).

4.2 Visualizing High Dimensional Space

All types of Activation Atlases presented in this chapter are 2-dimensional. However, the underlying data they work with—feature columns—belongs to a multidimensional space, often with hundreds of dimensions. To be able to visualize high dimensional vectors while minimizing distortions of a topological structure of the original space, projection techniques like Uniform Manifold Approximation and Projection (UMAP) [38] or t-SNE [36] can be used. This section is not to provide detailed mathematical explanations on why the aforementioned dimensionality reduction techniques work, though. Instead, an intuitive perspective on the functioning of UMAP and t-SNE is presented to complete the introduction of the techniques necessary to generate the atlases. Figure 36 provides a schema depicting how the Activation Atlas is obtained.

The process begins with feeding all the available images to the algorithm and collecting activation tensors across all the layers of a CNN. Then, an arbitrary layer is picked, for which the Activation Atlas will be generated (atlas can only represent activations originating from the same space; in the majority of cases that is equivalent to an individual layer of the network). High dimensional feature columns from the layer of interest (for every activation tensor it is either all

of them or a single, randomly selected one [12]), schematically marked with red squares in Figure 36, are projected by UMAP or t-SNE down to a 2D space, transforming a multidimensional activation vector into a 2-dimensional embedding. These projection techniques can preserve part of the local structure of the original space and, consequently, organize the feature columns into a convenient layout. An interesting side-effect of such dimensionality reduction is that feature columns representing similar concepts are projected roughly to the same region of the 2D space. This particular characteristic makes it possible to simplify the analysis from thousands of points to barely tens or hundreds. The idea is to divide a 2-dimensional layout into a grid of cells [12]. All the points that fall into the same cell are averaged together, yielding an average feature column (concept) for that cell. Finally, using feature visualization, there is an icon generated for each cell to provide the visual interpretation of the abstractions in this area of the projected manifold. To summarize: Activation Atlas is just an alternative name for the "grid of icons depicting average, high dimensional feature columns, projected down to a 2D space, with each icon depicting an underlying concept".

4.3 Manifold Sampling

From a theoretical point of view, all possible activations of a given CNN form a multidimensional manifold. Such a topological structure is an interesting object for in-depth studies since it contains details on how the algorithm responds (behaves), given a wide range of inputs. The challenge in examining such a vast and complex construct is the extent of a potential analysis to be conducted. Fortunately, there exists a solution to this problem: if there is too much data to process, a sampling technique proves to be useful. It approximates the original data distribution with its sparser equivalents, reducing the amount of information to be analyzed by a significant factor.

Interestingly, feature visualization can be seen as a technique for sampling the manifold of possible activations of a given CNN. Depending on a chosen unit of interpretability, the scope of the obtained results differs notably. Figure 37 illustrates this idea, originally introduced in [12]. Note that for the presented example, the original 528-dimensional manifold from the mixed4d layer was projected down to 3 dimensions using UMAP to make the illustration possible. Looking at feature visualization from the manifold sampling perspective, the following points can be made:

- Basic units of interpretability like neurons or channels are mere 1-dimensional slices of the space of possible activations. They should be seen as tiny blocks used to construct relevant representations like feature columns, not the main target of the analyses.

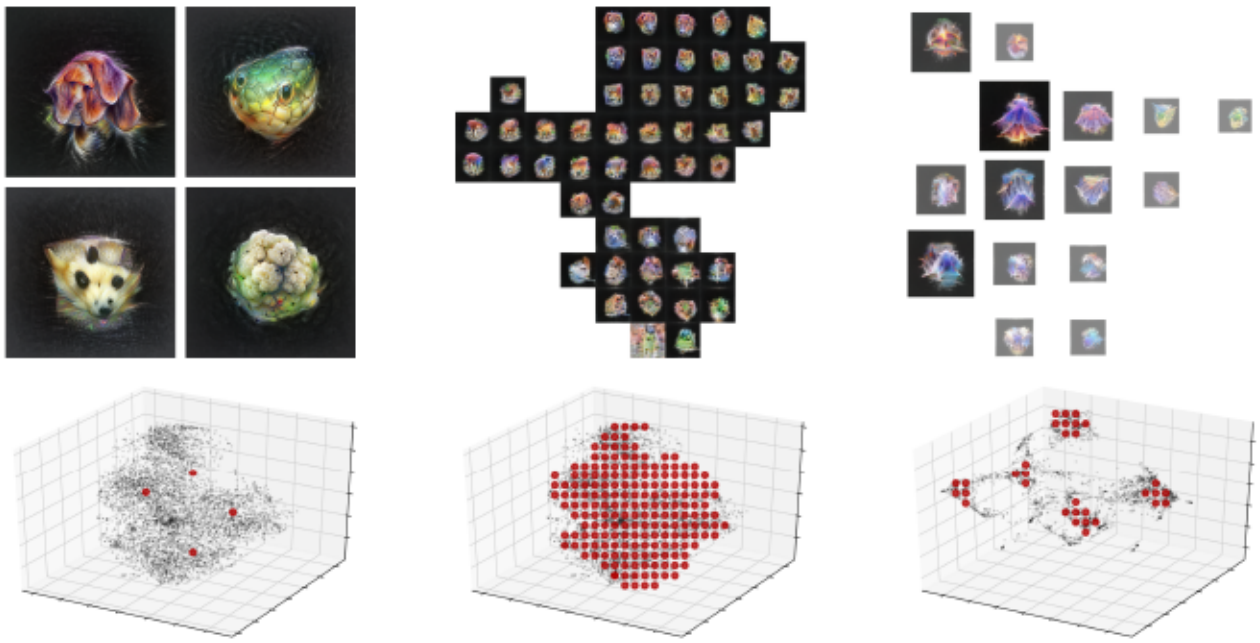


Figure 37: Feature visualization as a technique for sampling the manifold of possible activations of a CNN. Note that in this example, the original 528-dimensional manifold was projected down to 3 dimensions to enable visual overview. **Left:** visualizing basic units of interpretability like neurons or channels result in a scarce sampling of the manifold. **Middle:** Activation Atlas performs dense sampling, providing a holistic perspective on the space of possible activations. **Right:** Distilled Class Factors Atlas, due to the altered set of activation vectors used for the analysis, yields a manifold unlike the one of the original Activation Atlas, yet still sampling it extensively.

- Feature columns (linear combination of channels) are vectors denoting points in the original space of activations. They are a much more meaningful representation to work with when sampling the manifold; however, they still yield sparse results if studied in isolation.
- Activation Atlas performs dense sampling, providing a holistic perspective on the space of possible activations. The main caveat with this technique is that it requires tremendous computational resources to 1) get the activation tensors for over 1 million images from the ImageNet dataset and 2) generate hundreds of feature visualizations.
- Distilled Class Factors Atlas identifies latent clusters of activations in the original space by automatically determining which factors (feature columns) attribute strongly to a classification result. A significant decrease in the number of feature columns to look at (achieved by careful filtering and averaging) scales down the analyses to a level feasible for humans.

When sampling the manifold of possible activations of a CNN, the importance of choosing an appropriate unit of interpretability to work with manifests itself even more strongly than in previous tasks. To obtain a systemic view on the CNN's behavior, it is insufficient to work with simple neurons or channels. Per-image studies of distinct feature columns (as it was the case with neuron groups) are likewise not enough—the amount of data to examine quickly reaches levels infeasible for human analysis. On the other hand, the innovative nature of CNN Codes [27] and Activation Atlas [12] is hard to overestimate—these techniques were exactly what the field of interpretability needed to study a complex nature of CNNs' data representations at scale. A proposed Distilled Class Factors Atlas extends the space of available interfaces by narrowing down the analysis—it yields a class-specific manifold without the need to precompute a full atlas, allowing to draw conclusions about a sizable part of the activation space with limited computational resources.

4.4 Class Activation Atlas

It might be beneficial to isolate feature columns that strongly attribute to a class of interest and use them to construct a "specialized" version of the Activation Atlas, instead of using activation vectors collected from all images available in the entire dataset. Feature columns that either contribute to the class of interest the most or whose logit values have the highest overall magnitudes (even if the top class they support is not the target one) can be used to generate Class Activation Atlases [12].

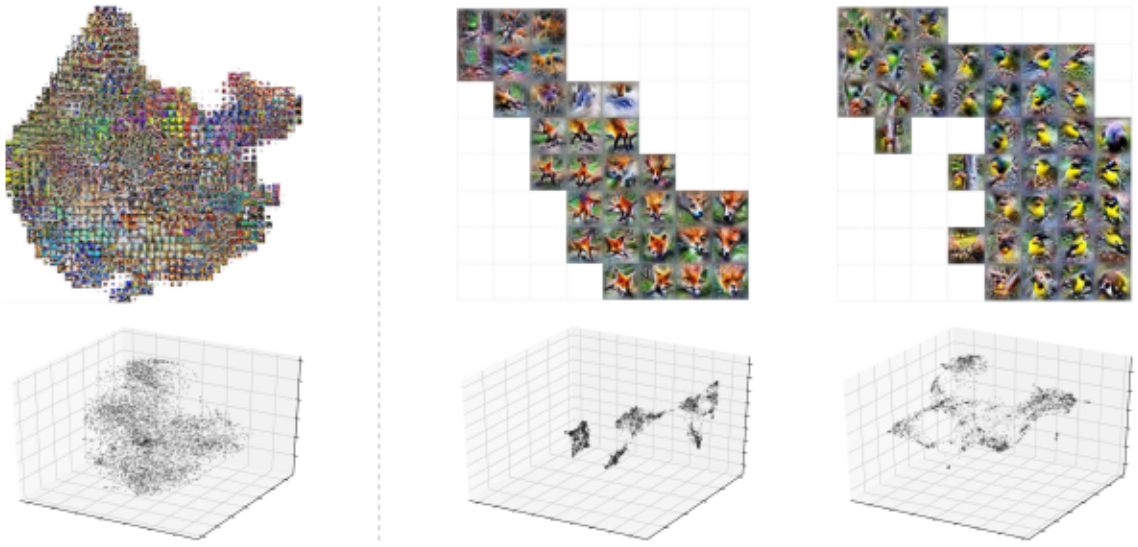


Figure 38: Comparison of standard Activation Atlas and Class Activation Atlas. Note the differences in resulting manifolds. Images of atlases retrieved from [12], licensed under CC BY 4.0.

Class Activation Atlas exposes feature columns that are used by a CNN to process the inputs belonging to the same class. It is as if the CNN Codes methodology was followed, but only for the examples of a single class, and with the results seen through the eyes of the network (thanks to feature visualization). Figure 39 illustrates a sample Class Activation Atlas generated for the inputs belonging to a red fox class, with the activation vectors collected from the mixed5b layer of the InceptionV1 model. As this layer is the very last inception module of the network, its data representation is highly abstract. When visualized, the activation tensors of such complexity are often hard to interpret. Still, one can observe red fox's snouts (bottom-right), fur (center and bottom-right), legs (center), and various types of backgrounds (top-left).

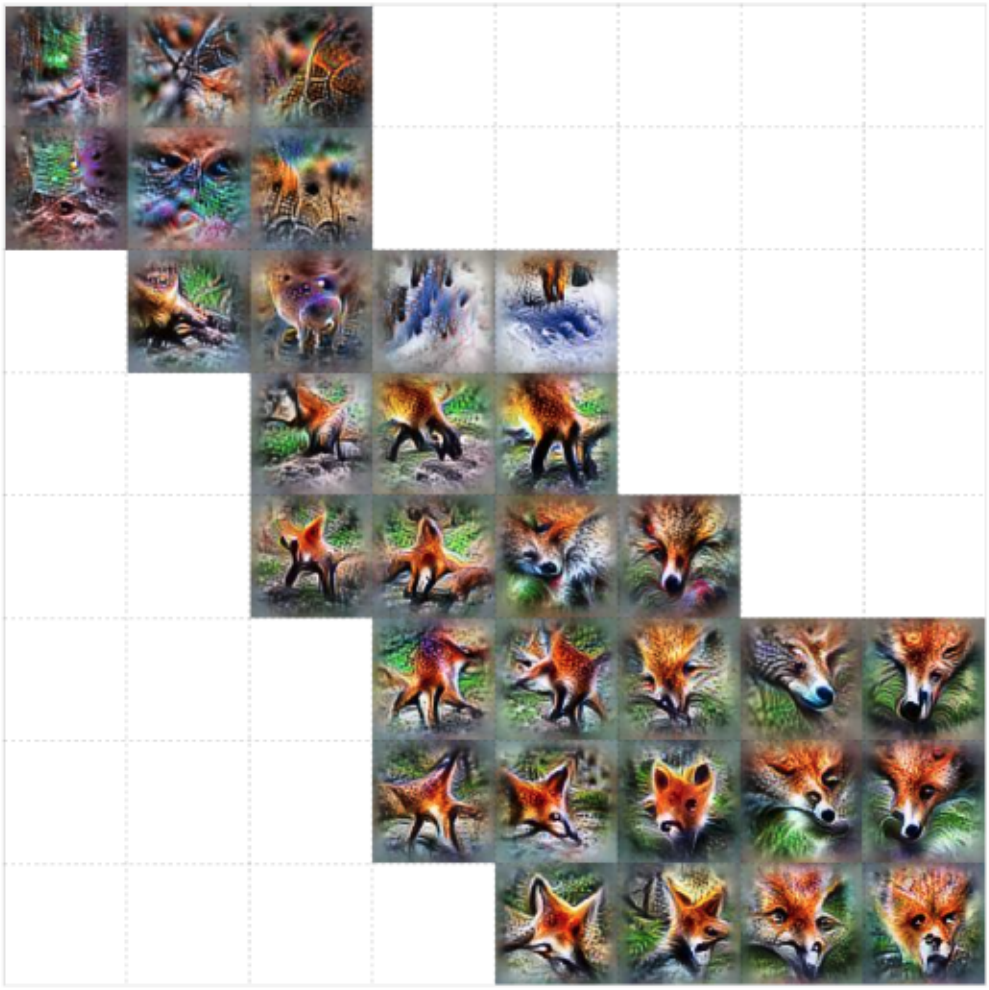


Figure 39: Sample Class Activation Atlas of the aggregated activation vectors from the mixed5b layer of the InceptionV1, collected from the images belonging to a red fox class. In spite of concepts' complexity, it is still possible to observe red fox's snouts (bottom-right), legs (center), and various backgrounds (top-left). Image retrieved from [12], licensed under CC BY 4.0.

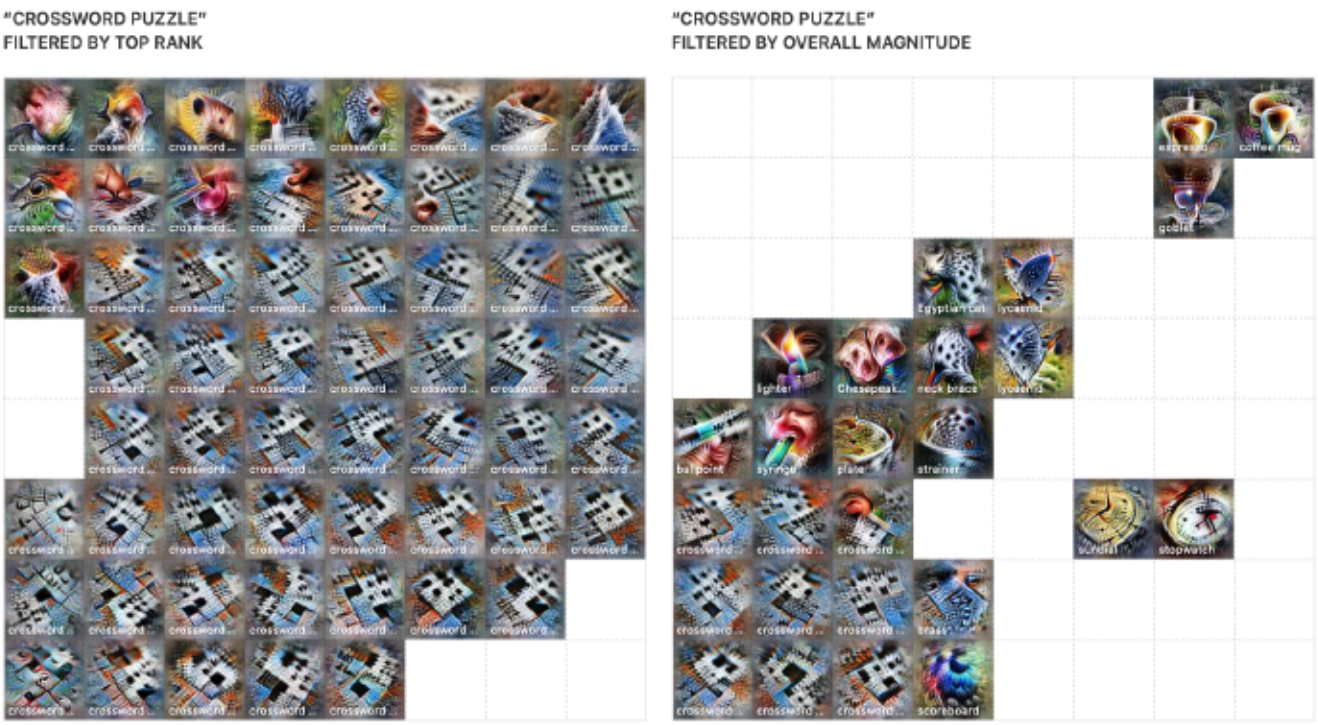


Figure 40: Activation vectors filtering techniques for Class Activation Atlas generation. Each icon is labeled with the top class it supports. Images retrieved from [12], licensed under CC BY 4.0.

In the example from Figure 39, only the activation vectors that attributed the most to the target class were used to generate the Class Activation Atlas. It is, however, possible to apply a different filtering approach: from all available feature columns (no matter what is the top class they contribute to), select the top-n (e.g. top-2000) in terms of the magnitude of their overall attribution to the class of interest and use those to generate a Class Activation Atlas.

Figure 40 illustrates an example of a "crossword puzzle" class to highlight differences that the two filtering techniques may yield. Careful analysis allows several interesting conclusions to be drawn. First, note that selecting activation vectors that attribute the most to the class of interest (left part of the figure) produces much more uniform results, focused specifically on the concepts related to a crossword puzzle. Such filtering approach prevents potentially valuable correlations from being exposed, since these may come from feature columns that are shared among many classes and that do not necessarily contribute to the target class the most. Secondly, please have a closer look at the right part of the aforementioned figure. Here, top-2000 activation vectors with the highest overall magnitude (value) of the attribution score towards the class of interest were selected [12]. Each icon is labeled with the top class it supports. Interestingly, concepts that contribute to classes like "coffee mug", "ballpoint", "plate", and of course "crossword puzzle" were

revealed, shining new light on the family of abstractions that participate in the "crossword puzzle" classification, exposing correlations that appear very natural and intuitive for human readers.

4.5 Distilled Class Factors Atlas

To generate a Class Activation Atlas, it is necessary to precompute activation tensors for all images available in the dataset. Next, a single feature column is randomly selected from each of those tensors. Then, score vectors associated with every feature column (obtained by computing a gradient of the score w.r.t. the original activation tensor) are used to determine how a concrete activation vector influences every possible output class. It is this additional attribution information that makes it possible to filter feature columns that are important to predict a target class the atlas is generated for.

Originally, random sampling of the activation tensors was used to address the problem of an almost intractable amount of data to process (if all feature columns were to be analyzed). Even with just a single randomly selected activation vector in each case, with over 1 million images the strategy was still able to yield meaningful results, reducing the extent of the analysis by at least two orders of magnitude[3]. The study was possible thanks to a high number of diverse examples—there were no significant improvements visible when the number of data points reached over 100 000 (below that number, however, generated atlas were not visually appealing [12]). It follows that, from a statistical point of view, 100 000+ images were sufficient to sample the original manifold of possible activations densely enough so that the resulting atlas was representative of the underlying data distribution. But what if there is not enough data to work with? Is random sampling the only strategy that can be used to extract important information from the activation tensors to better understand the behavior of convolutional networks?

In section 2.5.5, the idea of neuron groups was introduced. It is based on the assumption that there exists a set of higher-level concepts (specific linear combinations of channels) in the image that can be automatically extracted from the activation tensor of an arbitrary layer using matrix decomposition. Matrix decomposition factorizes the activation tensor into a product of two matrices: $\boldsymbol{F}$ (linear combinations of channels ≡ "what" is there in the activation tensor ≡ concepts) and $\boldsymbol{S}$ (latent spatial factors ≡ "where" the concepts are). Please refer to eq. (6) for further details.

The analysis of latent factors with neuron groups scales the problem from hundreds of feature columns to inspect down to just a few (often around 6-8 per image). That aspect, together with

[3]If all feature columns were to be used, there would be tens or hundreds of them for each input image, depending on the layer of interest.

the fact that the extracted concepts are often relatively easy to interpret, is what makes the matrix decomposition a suitable technique to use when the goal is to robustly identify distinct feature columns. One thing to keep in mind is that the resulting concepts do not necessarily exist as-is (in the form obtained during the factorization procedure). Rather, they should be seen as an approximation of the actual feature columns, which represents a group of similar activation vectors.

As compelling as the groups of neurons are, they were not designed to draw comprehensive conclusions about the models' behavior. Instead, they provide per-image insights. There is, however, a captivating use case for matrix decomposition: the way it extracts concepts could replace a random sampling strategy in the process of Activation Atlas generation. That particular idea led to the creation of a novel interpretability interface—Distilled Class Factors Atlas.

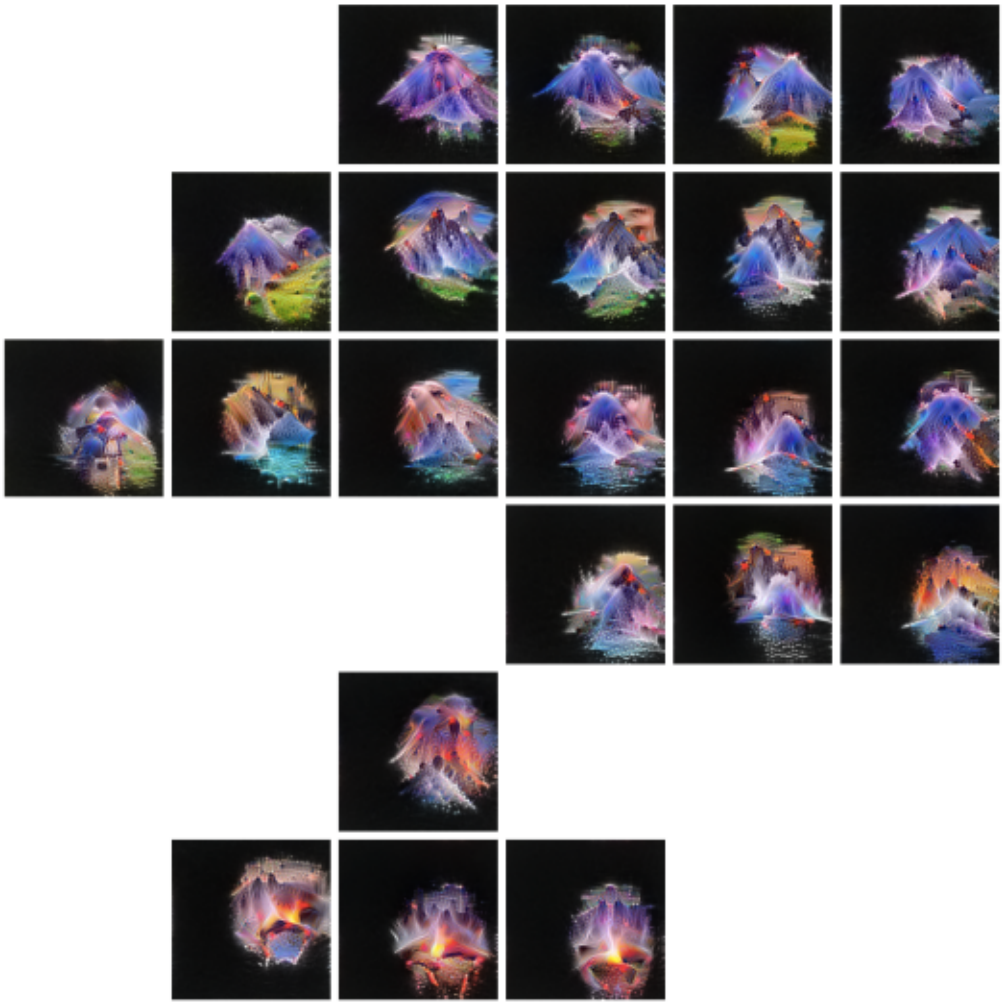


Figure 41: Sample Distilled Class Factors Atlas, generated for a *volcano* class.

Figure 41 depicts a Distilled Class Factors Atlas in a way similar to how Class Activation

Atlases were presented in the previous section—a uniform grid of icons organized into a 2D layout, reflecting a high dimensional manifold projected down to two dimensions. Such a basic type of visualization provides an overview of the latent factors identified in multiple images of a *volcano* class, without any additional information on concepts' influence on the classification result. In Figure 41, feature columns from a mixed5a layer were used. The layer was chosen arbitrarily since the corresponding atlas contains interesting icons that accurately summarize what a volcano concept means: lava (bottom), smoke (bottom-center), landscapes related to mountains, with varying scales and contexts (center and top).

Although the type of result a Distilled Class Factors Atlas technique yields might resemble the one obtained using Class Activation Atlas, the generation process differs considerably. To produce a Class Activation Atlas for a single class of interest, it is required to randomly sample feature columns from all the images available in the dataset. Theoretically, one could sample the activation vectors only from the examples belonging to a target class, yet this could result in a multitude of important concepts being rejected due to the random nature of the selection procedure. Alternatively, all feature columns from such examples could be taken into account; however, this would introduce a lot of meaningless noise that would impede the process of atlas generation. Distilled Class Factors Atlas addresses both of the aforementioned issues by 1) using only the images belonging to a given class and 2) leveraging matrix decomposition to robustly identify a small set of distinct concepts. Consequently, atlas generation is computationally efficient, while ensuring that the important abstractions are not dismissed by any randomness. Distilled Class Factors Atlas is not able, however, to reveal correlations between a class of interest and concepts that are not present in the images belonging to that class. For such a use case, the Class Activation Atlas technique would be a better choice.

An important detail that needs to be taken into consideration when generating any kind of atlas is a grid size. It can have a tremendous influence on the interpretability of a resulting interface. In Figure 42, three examples were provided to illustrate the impact of a grid size parameter. A grid that is too small yields icons that are the result of averaging over too many feature columns, obliterating the complexity of the underlying representation. The just-right size of a grid reveals a rich set of concepts identified by a matrix decomposition. When a grid size is too large, a resulting atlas loses its advantage of aggregating clusters of the existing abstractions, instead, displaying an overwhelming amount of icons to analyze.

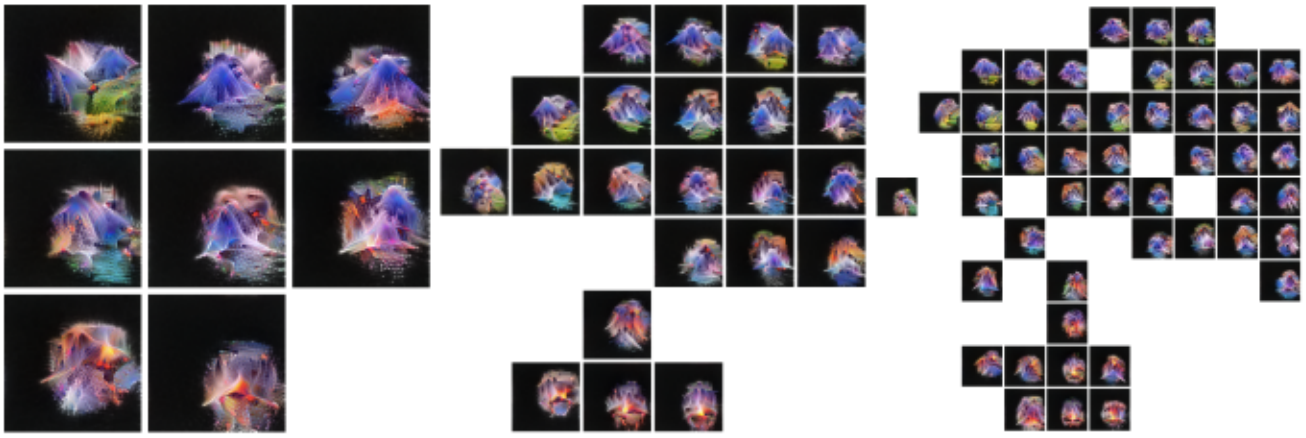


Figure 42: Influence of a grid size on the generated atlas. **Left:** too small grid yields the icons that are the result of averaging over too many feature columns, obliterating the complexity of the underlying representation. **Middle:** just right size of a grid reveals a rich set of concepts identified by a matrix decomposition. **Right:** when a grid size is too large, a resulting atlas loses its advantage of aggregating clusters of the existing abstractions, instead displaying an overwhelming amount of icons to analyze.

4.5.1 Factors Are Not Created Equal

When analyzing icons in the atlases, it would be exceptionally useful to be able to determine how important a given concept is to a classification of a class of interest. In the original work [12], this kind of information for **full** Activation Atlases was provided in two different ways: 1) scaling the icons (the more points in a cell of a grid, the larger the resulting icon) and 2) altering icons' transparency (the stronger the attribution to the class of interest, the less transparent the icon)[4]. While the first one is a basic heuristic that should work reliably only if thousands of data points are available (based on the laws of statistics), the second one estimates the importance of a concept by averaging over the attributions of all feature columns that fell into the same cell in a grid. The attributions are calculated by linearly approximating the computation that occurs in the part of the convolutional network that follows the layer the atlas is being generated for. Such approximation is then used to obtain a single scalar value indicating the influence of a specific concept on a class of interest. To clarify, the aforementioned process can be expressed mathematically as follows:

$$a = \sum_{x=1}^{\text{height}} \sum_{y=1}^{\text{width}} \sum_{d=1}^{\text{depth}} \mathbf{A}_{x,y,d}, \quad \mathbf{A} = \mathbf{H} \odot \nabla_{\mathbf{H}} \text{logit}_c, \tag{14}$$

[4]Neither of these were used in the process of Class Activation Atlases generation, though.

where a is the resulting scalar value denoting the sum-reduced attribution of the concept to the class of interest, $\mathbf{A}$ is the 3D attribution tensor and $\mathbf{H}$ is the 3D activation tensor. Tensor $\mathbf{A}$ is the result of the element-wise product of the activation tensor $\mathbf{H}$ and the gradient of the logit of class c w.r.t. that activation tensor. Equation (14) represents a form of a gradient $\odot$ input attribution technique, with the input being an activation tensor. The linear approximation is suitable when operating on a feature column level; however, it does not directly apply to latent factors. To include the concepts' importance in the atlas, a different approach needs to be taken.

In [49], Olah *et al.* introduced the *neuron groups' attribution* method that can be used to estimate the importance of factors obtained from the activation tensor decomposition. In the previous chapter, an improvement over their technique was introduced, namely: Latent Factor Attribution. Combining LFA with Distilled Class Factors Atlas yields a novel type of visualization—class-specific atlas that includes the well-grounded attribution information. To better understand the value provided by such a hybrid interface, consider the following 2D layout with a basic atlas:

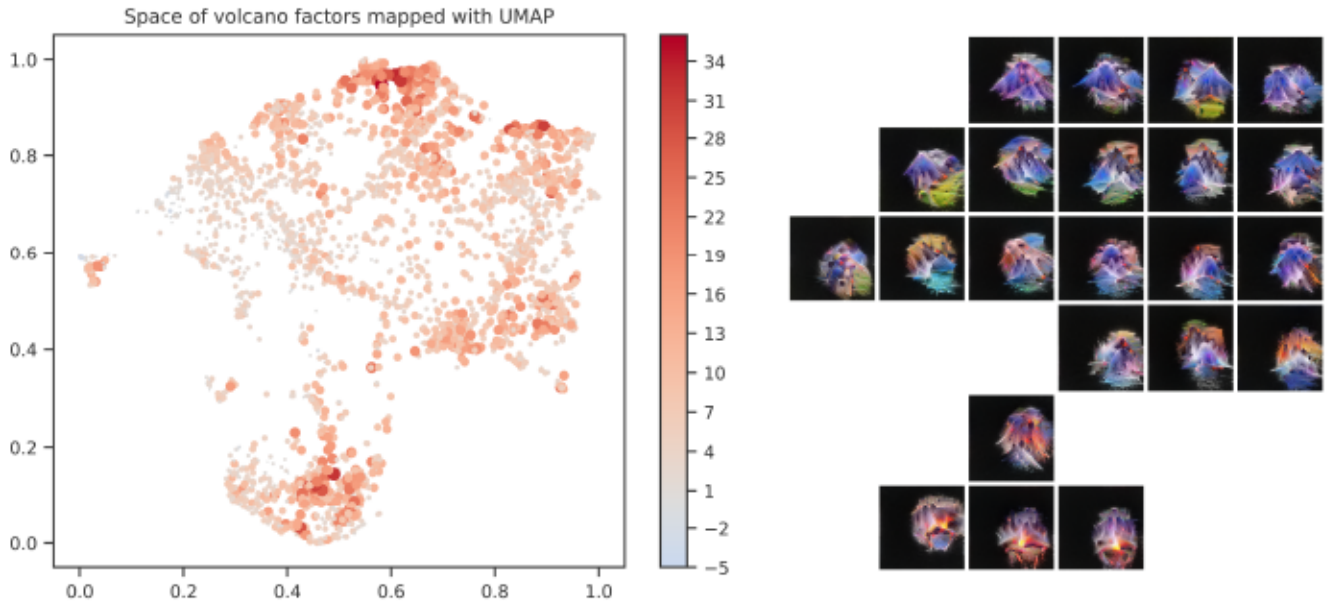


Figure 43: A 2D projection of class-specific latent factors, together with a basic atlas. The size and color of points correspond to the values of sum-reduced LFA attribution tensors.

The layout on the left differs from the ones presented before—this time it is not a uniform set of points, but rather a diverse visualization. Points' sizes and colors depend on a sum-reduced attribution (the former based on an absolute value): gray denotes "neutral to the classification", blue means "inhibitory to the class of interest" and red is "supporting the prediction of a target class". Here, the attribution is computed using the LFA technique. The juxtaposition of the

raw manifold projection with the atlas, as illustrated in Figure 43, makes it possible to observe the clusters of points that specific icons represent and to draw some initial conclusions about their impact on the classification result. Such a comparison, however, is neither efficient nor a particularly effective in assessing factors' importance. Instead, the attribution information can be directly linked with the Distilled Class Factors Atlas by altering icons' size and opacity.

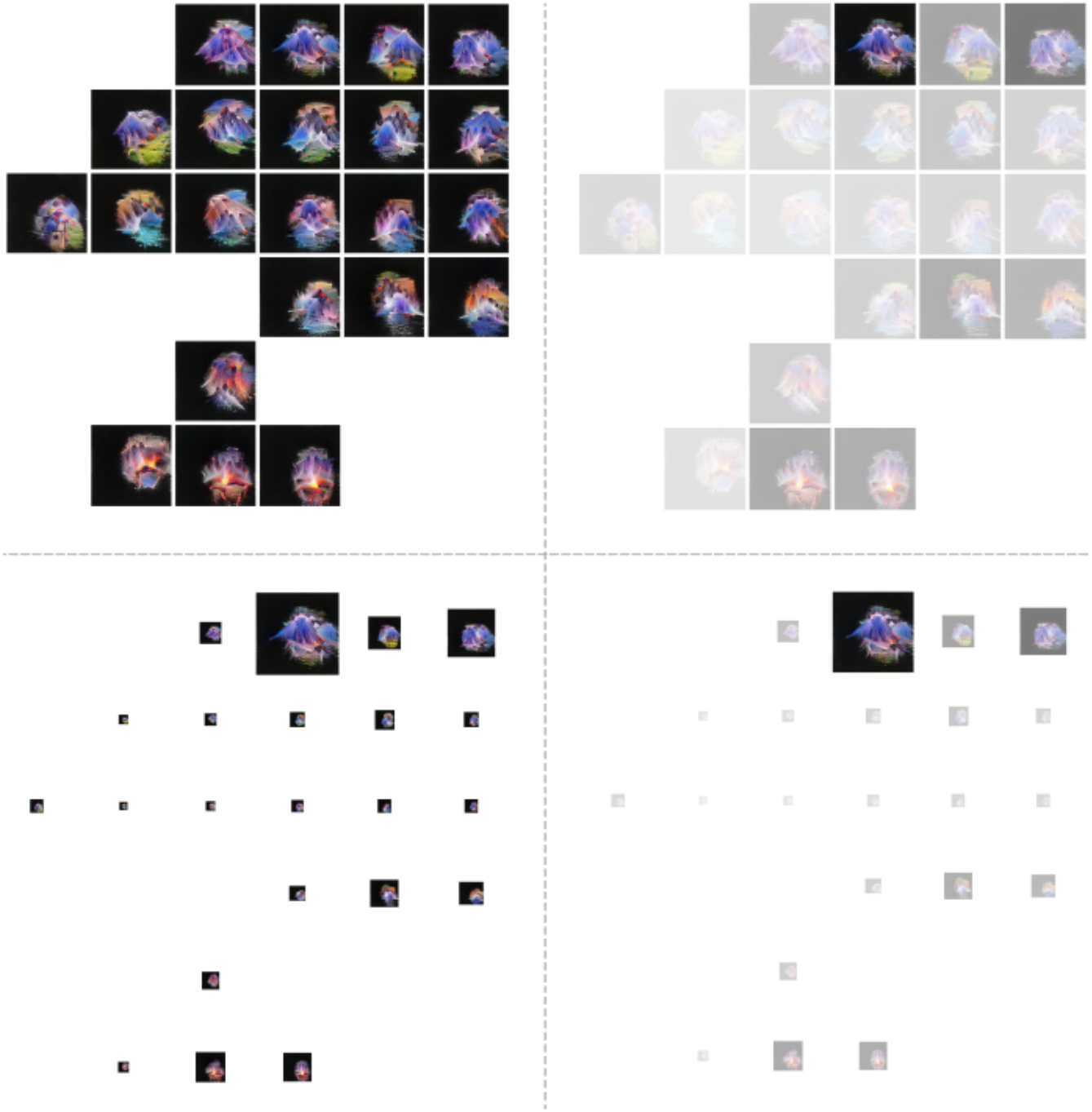


Figure 44: A basic atlas (top-left), compared to the ones with the LFA information incorporated.

In Figure 44 one can clearly see how different the basic atlas (top-left) is from the atlases that have the LFA attribution information incorporated into them. Top-right atlas has the icons' opacity mapped to the scalar attribution: the more opaque the feature visualization is, the higher its importance. Bottom-left atlas, on the other hand, alters icons' scale instead of the opacity, with the same rule being applied here as well: the larger the icon is, the more it influences the class of interest. Bottom-right atlas applies both opacity and scale modifications. The mathematical formula to obtain target values of the parameters to be modified is as follows:

$$m(\boldsymbol{a}_i) = \left(\frac{\exp[\boldsymbol{a}_i - \max(\boldsymbol{a})]}{\sum_j [\exp(\boldsymbol{a}_j - \max(\boldsymbol{a}))]} \right)^s, \tag{15}$$

where $m(\boldsymbol{a}_i)$ is a value between 0 and 1, denoting a scale or opacity modifier (0: "zero size" and/or "fully transparent"; 1: "original scale" and/or "fully opaque"), $\boldsymbol{a}$ is the vector of sum-reduced attribution values (see eq. (14) for details on obtaining a single element, denoted as $\boldsymbol{a}_i$ in the equation above), and s is the smoothing factor that enables fine-grained control over the obtained visualizations (value between 0 and 1, inclusively). Equation (15) is essentially a standard softmax function, modified by subtracting $\max(\boldsymbol{a})$ from the value to be exponentiated, with the result raised to the power equal to a smoothing factor s. The subtraction is used to ensure that neither overflow nor the underflow will occur in the computation. In Figures 45 to 51 one can find examples illustrating how manifold projections and the corresponding atlases for a *volcano* class vary across the layers of the InceptionV1 network.

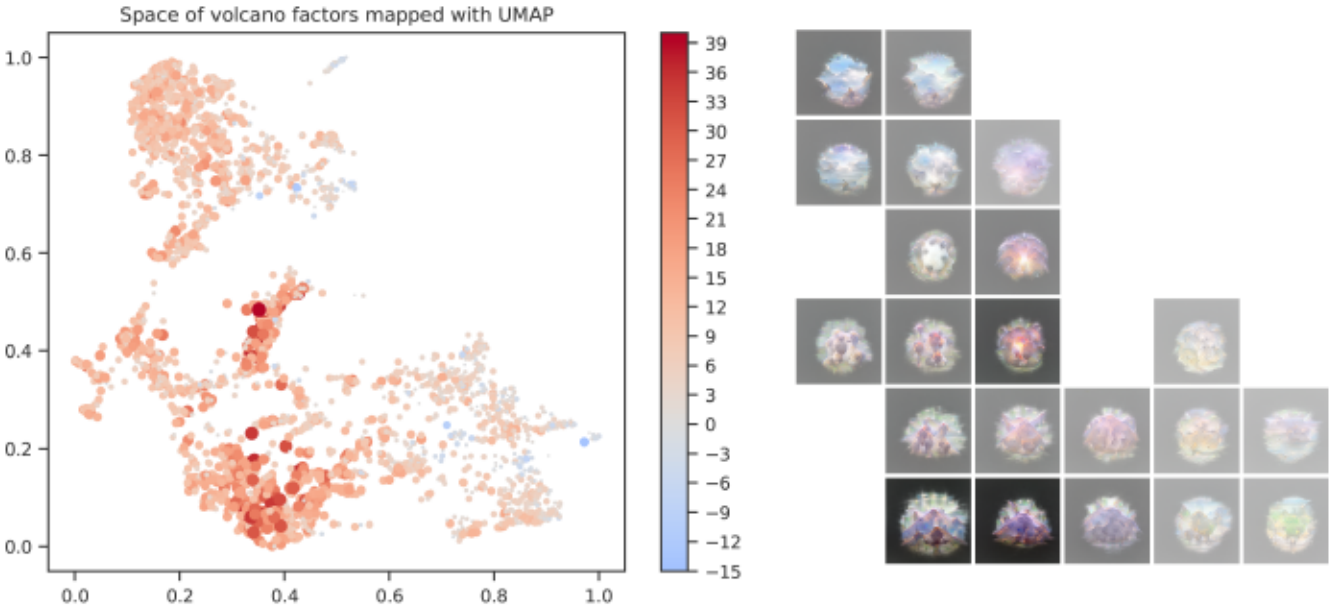


Figure 45: Projected manifold and corresponding atlas for mixed4a layer of the InceptionV1.

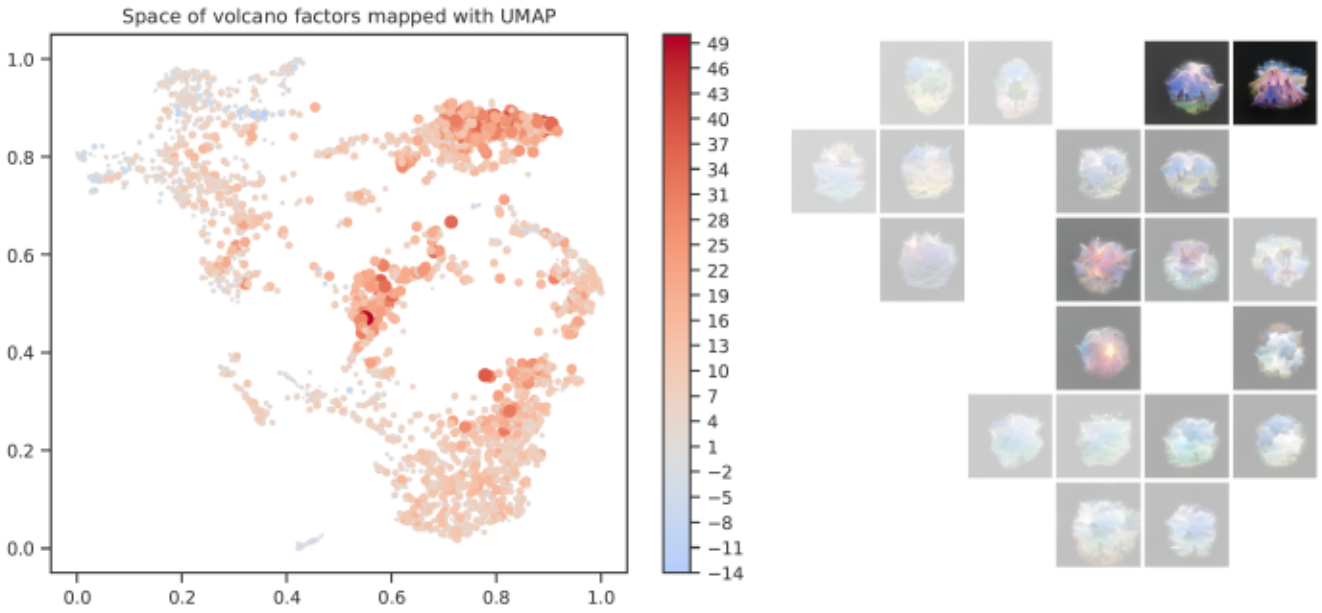


Figure 46: Projected manifold and corresponding atlas for mixed4b layer of the InceptionV1.

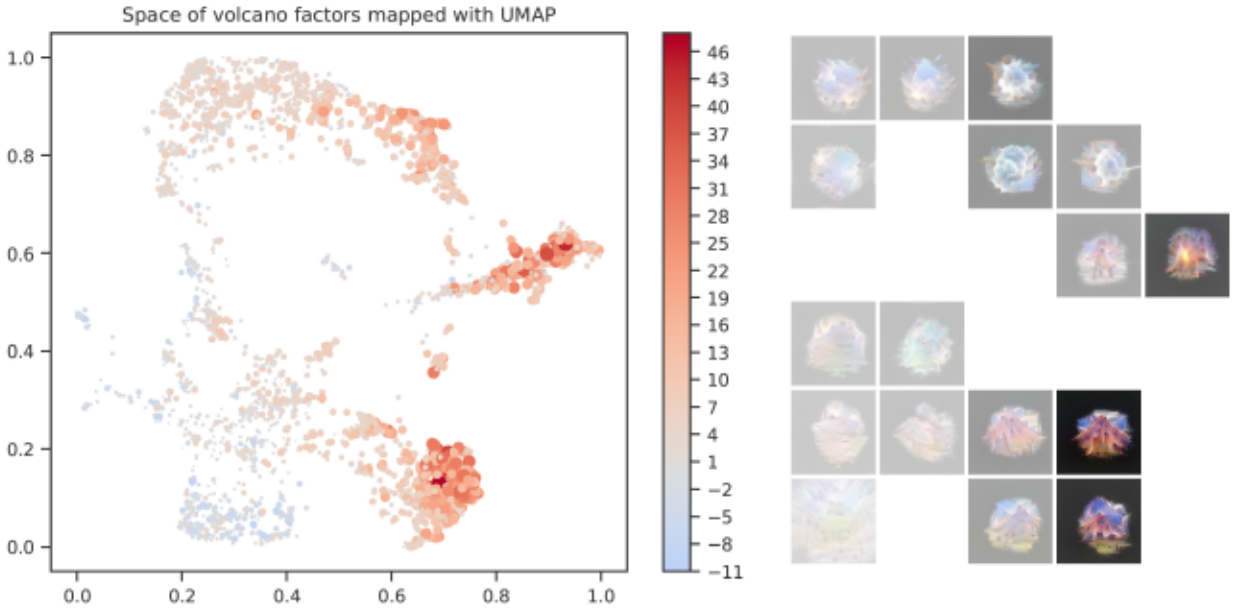


Figure 47: Projected manifold and corresponding atlas for mixed4c layer of the InceptionV1.

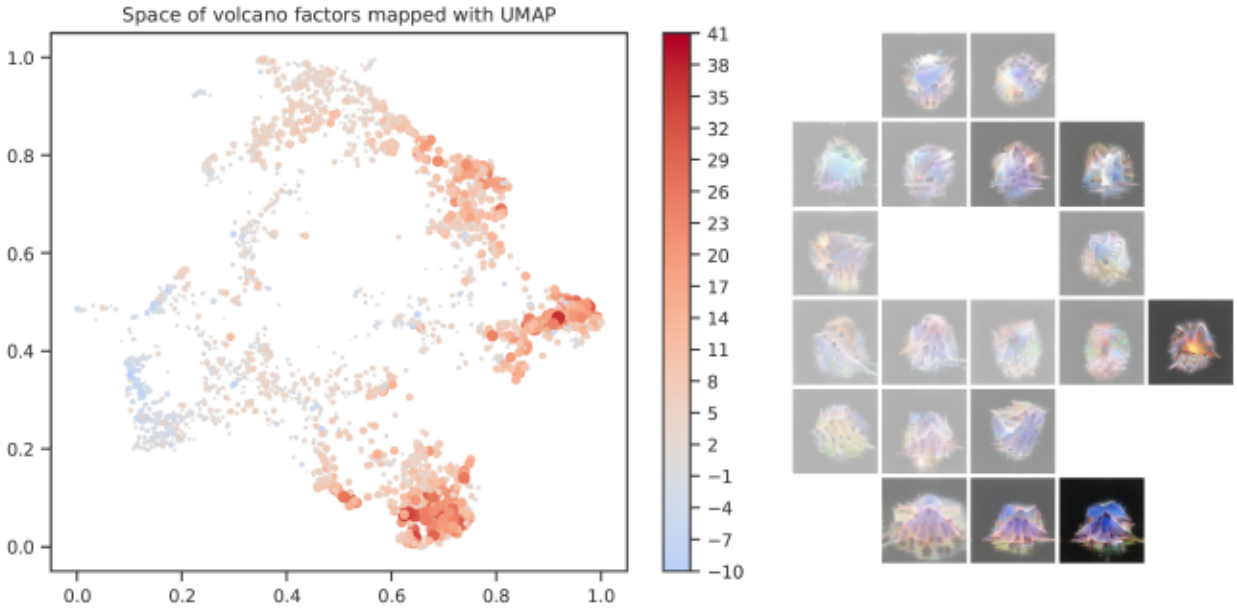


Figure 48: Projected manifold and corresponding atlas for mixed4d layer of the InceptionV1.

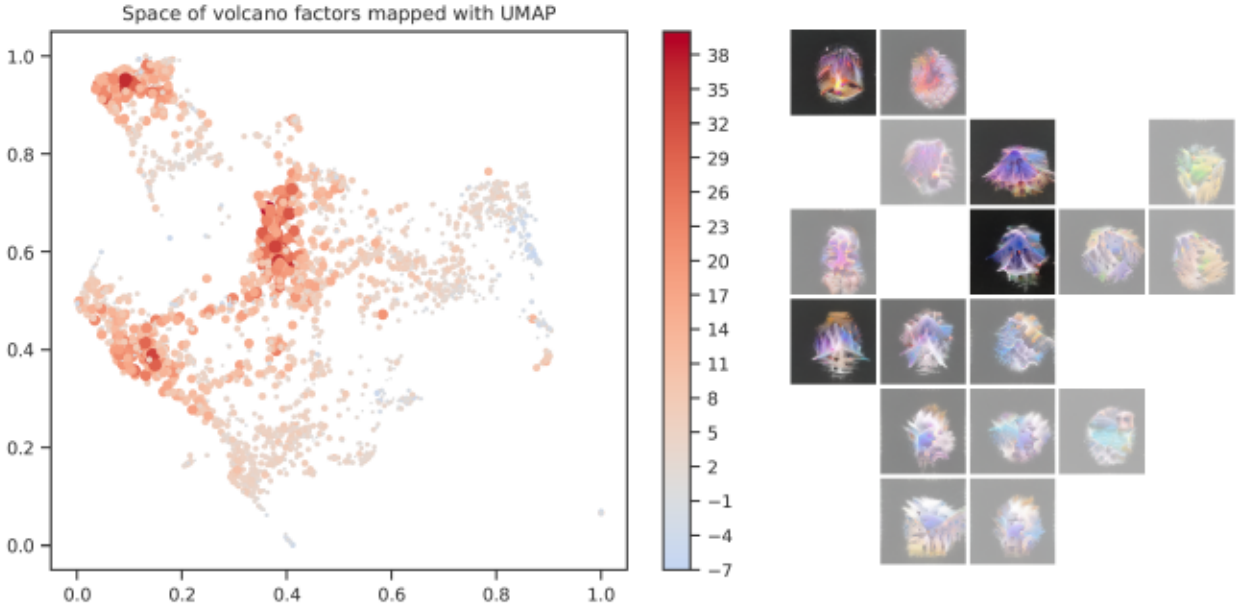


Figure 49: Projected manifold and corresponding atlas for mixed4e layer of the InceptionV1.

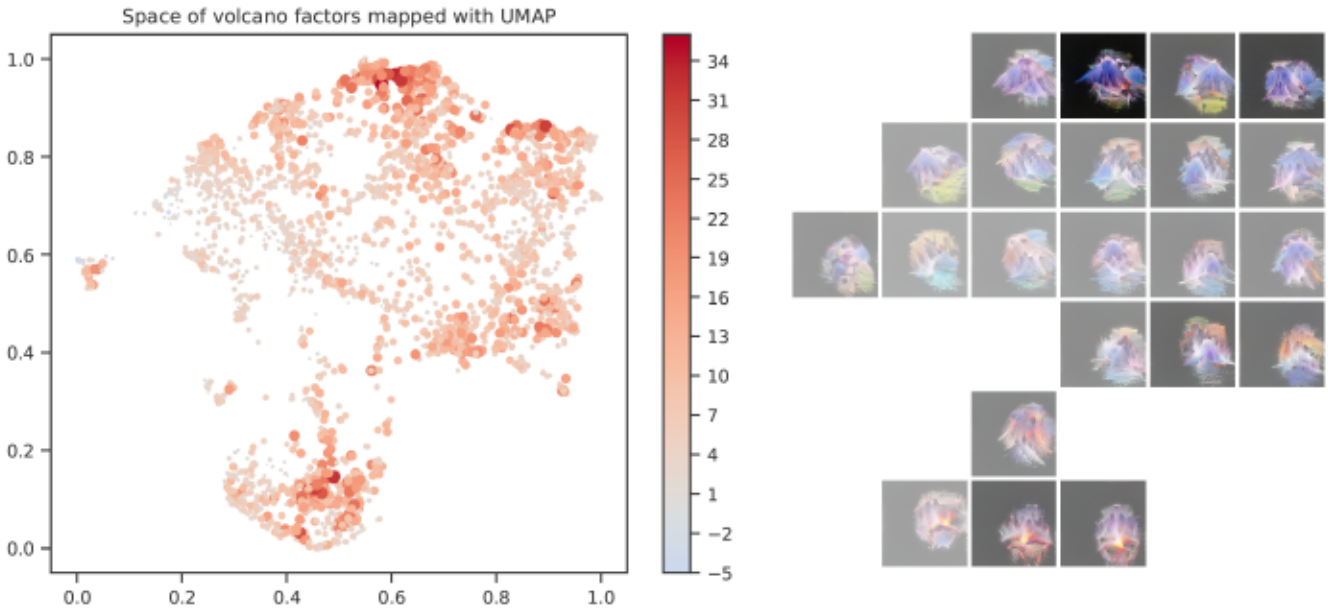


Figure 50: Projected manifold and corresponding atlas for mixed5a layer of the InceptionV1.

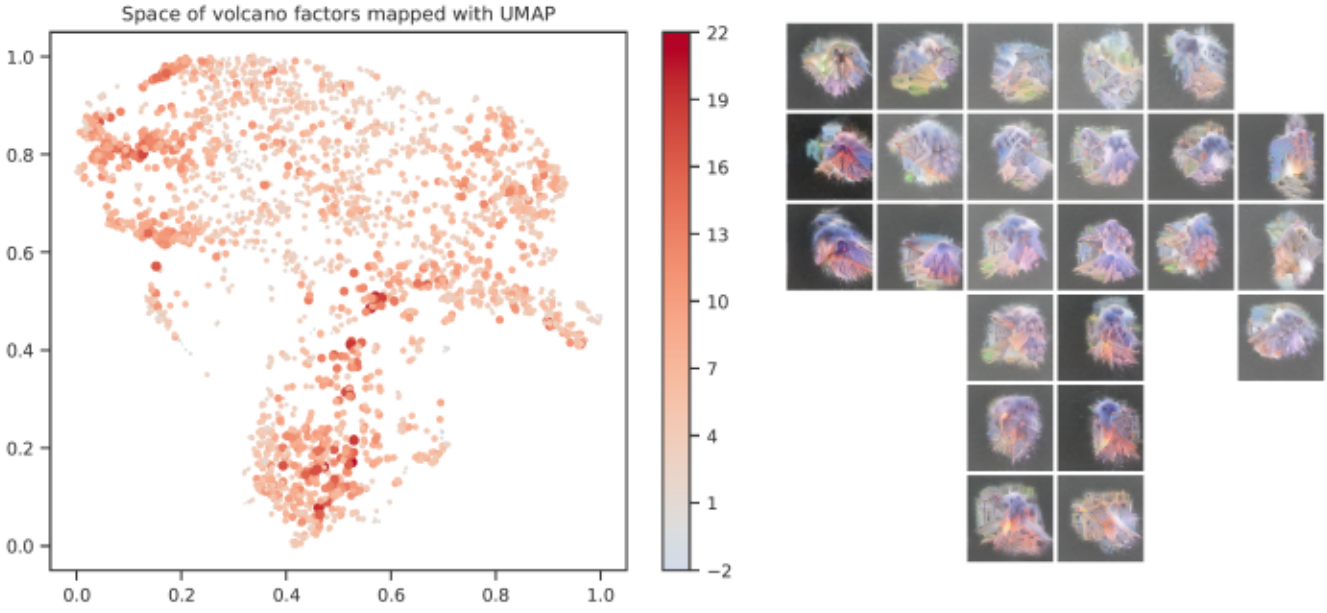


Figure 51: Projected manifold and corresponding atlas for mixed5b layer of the InceptionV1.

4.6 Discussion and Limitations

To study a multitude of Distilled Class Factors Atlases, a dedicated web interface has been created (see Figure 52). The resource is available at `https://bmiselis.github.io/atlases.html`. It is possible to explore any layer of interest (all mixed4* and mixed5* layers are available), together with a class of choice from the following: *grand piano*, *sax*, *scuba diver*, *indian cobra*, *african elephant*, *dalmatian*, *peacock*, *red fox*, *giant panda*, *tarantula*, *zebra*, *dam*, *sea shore*, and *volcano*. These particular classes were selected due to the fact that each of them is very characteristic and easy to interpret. Some represent an instrument or a distinct animal species, while the other depict more abstract concepts like a landscape or a geological structure.

Another aspect that can be controlled in the interface is a grid size, with the values ranging from 3×3 to 10×10. Additionally, users can control smoothing factors for icons' size and transparency independently (transparency = 1 – opacity). All of the aforementioned tunable parameters provide interesting insights into the concepts that attribute to a given class and the way their underlying representations evolve in the consecutive layers of the convolutional network.

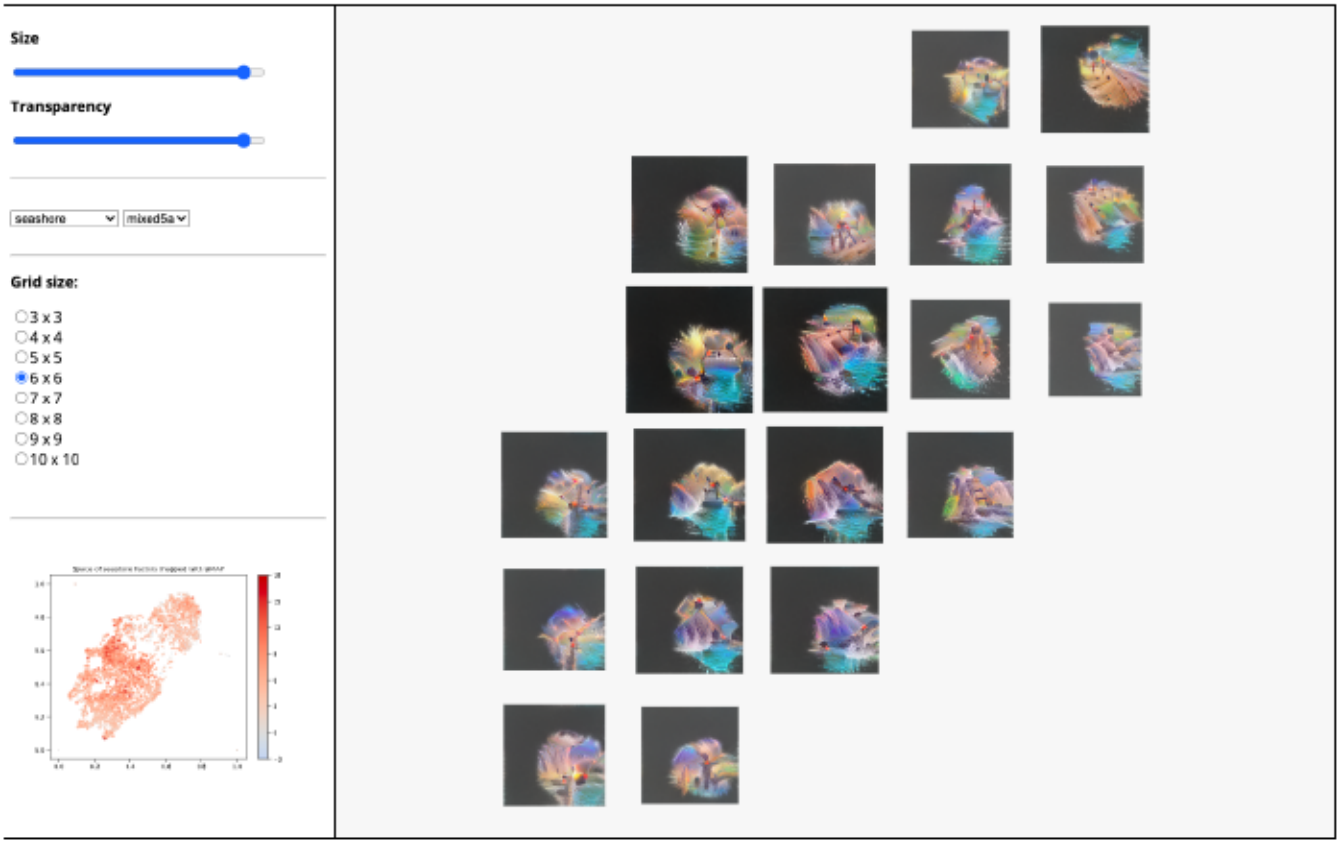


Figure 52: A web interface to explore data from Distilled Class Factors Atlas experiments.

One of the main questions of this chapter is the following: do Distilled Class Factors Atlases help interpret the class-specific behavior of a convolutional network? According to the subjective opinion of the author of the thesis, the answer to this question is: yes, they do. It cannot, however, be proved mathematically in an objective way. Instead, several case-studies were described below to indicate what kind of insights the proposed interface provides and why these can be perceived as valuable in the process of an algorithm's comprehension.

Naturally, each image contains (or at least should contain) an "object" and a "background". The background is the surrounding of the instance, i.e. a context in which it occurs naturally. When extracting latent factors from the activation tensors, a subset of them will inevitably be related to that context. The important question here is whether this additional information is actually used by the network while making a prediction, or is it, rather, discarded and does not participate in the classification. To address this question, three examples are provided in Figures 53 to 55 (with captions containing per-case analysis). For each case, the atlas was generated using factors extracted from a mixed4e layer, with the left visualization being a basic distilled class (without the attribution) and the right one incorporating the concepts' importance.

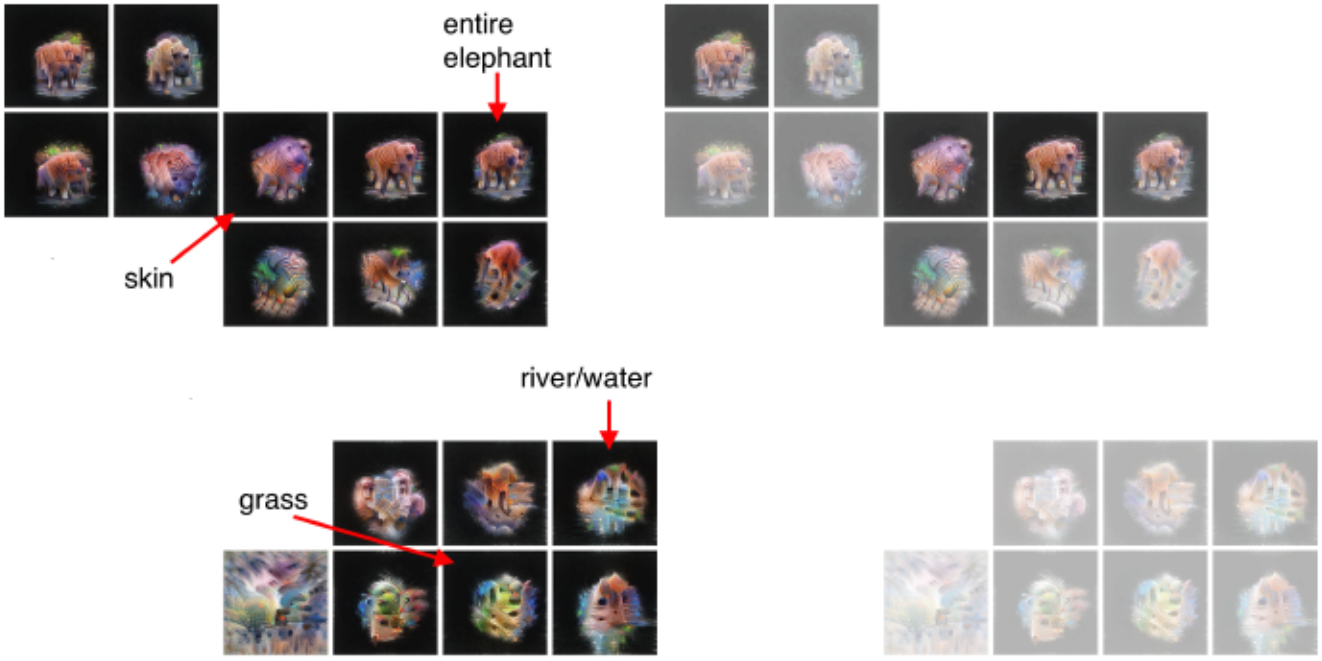


Figure 53: Background's influence on the prediction of the "african elephant" class. The upper cluster contains concepts directly related to the "african elephant" class, while the icons in the bottom depict various kinds of background. Model's behavior matches human intuition—background does not influence the prediction of the "african elephant" class.

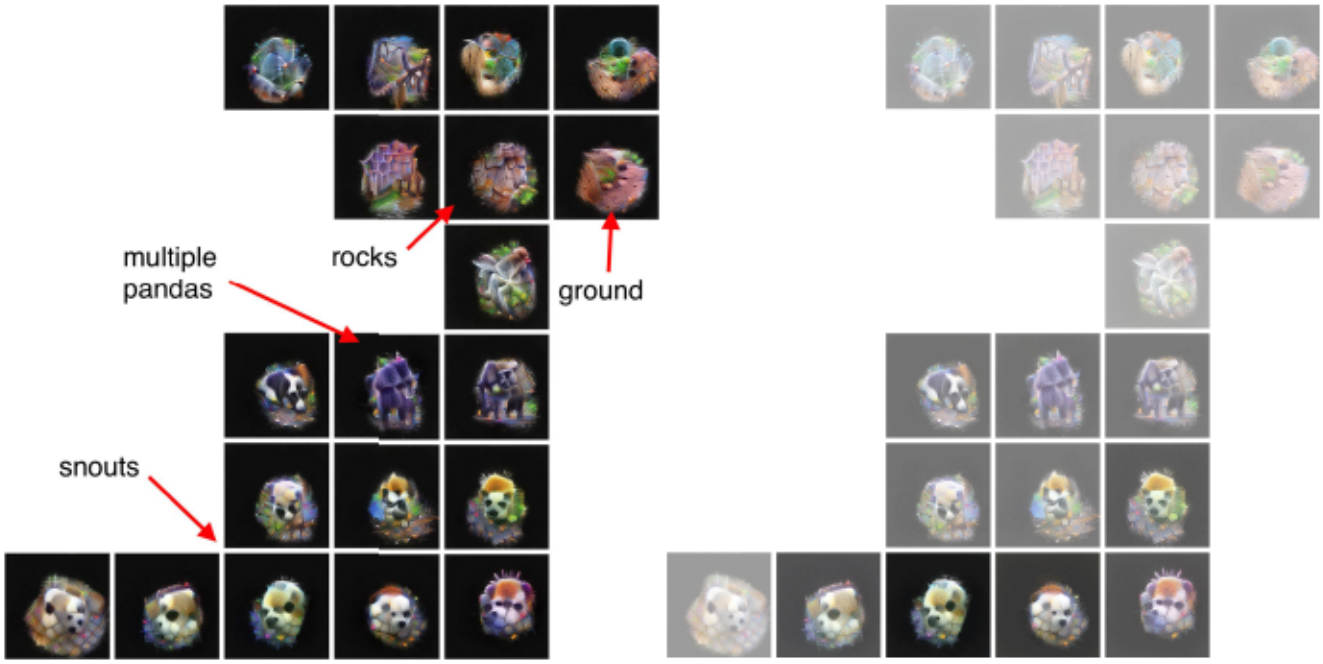


Figure 54: Background's influence on the prediction of the "giant panda" class. Multiple interesting concepts were identified, yet the one depicting the panda snout remains the most important.

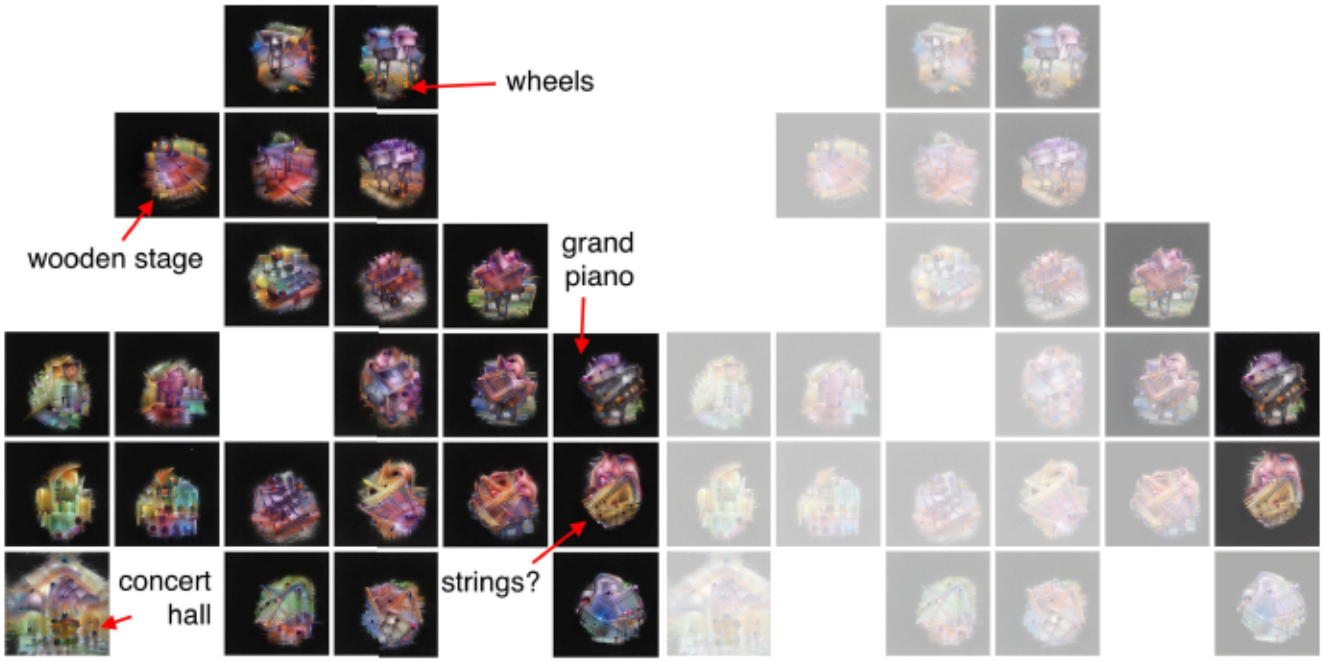
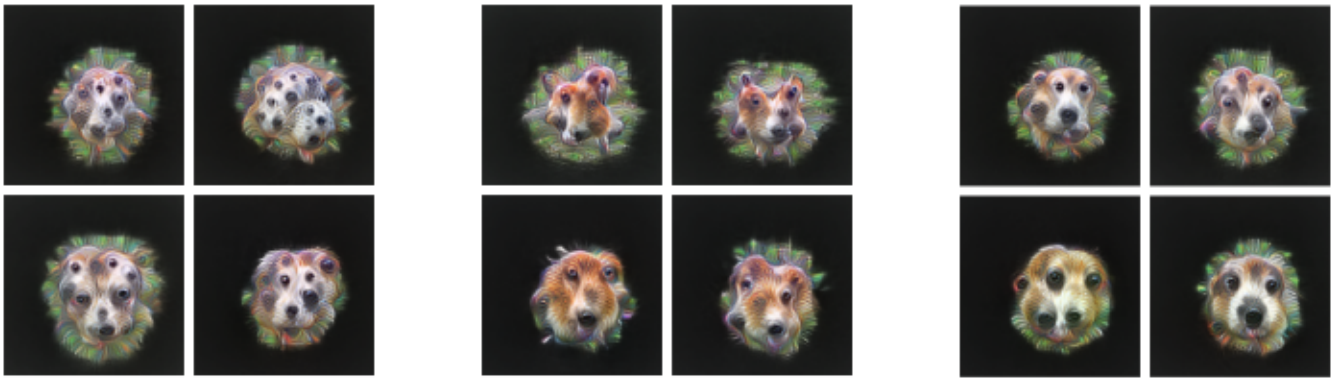


Figure 55: Background's influence on the prediction of the "grand piano" class. Multiple interesting concepts were identified, yet the one depicting the entire instrument and something similar to grand piano's strings remain the most important ones for the classification.

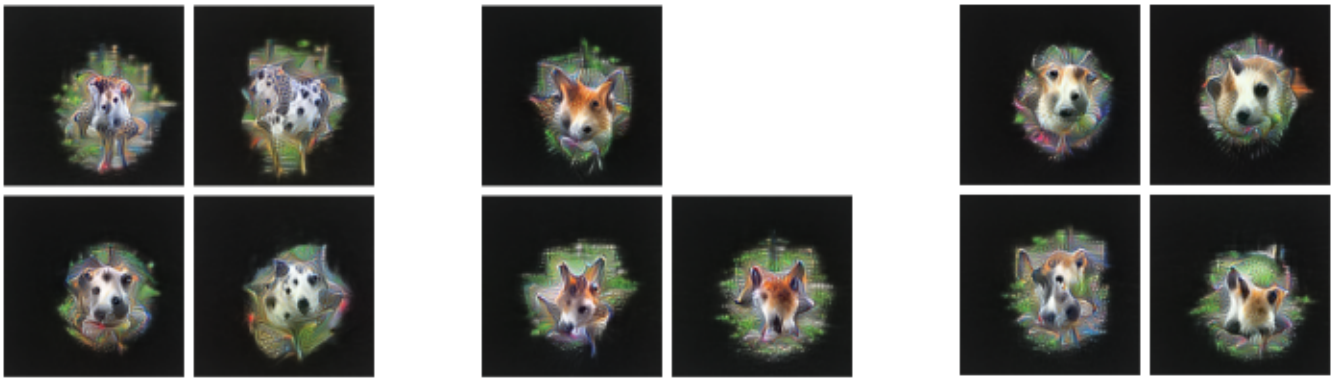
It is very reassuring that the actual behavior of the convolutional network under investigation matches human intuition—both the examples provided in the text and the ones that were not presented in this thesis (due to the volume of the visualization) indicate that class's instances attribute to the classification result the most, not the background or the other vague contextual information. Of course, there are cases for which it is non-trivial to determine what is an instance of the class and what the background is. Good examples would be categories like "volcano", "seashore" or "dam". For these, the boundary between the important concept and "the rest" is rather vague and hard to express even for humans; hence, they were not included in the analysis.

Another insight that the Distilled Class Factors Atlas with the attribution information offers is the following: it is possible to compare concepts that attribute to given classes the most. Such a juxtaposition makes it possible to determine whether there are any visual similarities (or differences) between the underlying abstractions. Intuitively, an expected result would be that if there are classes that are related to one another in some sense, then there should be similar concepts identified by the model while processing them. An example of such classes could be "red fox", "dalmatian" and "giant panda". The animals belonging to these species all have fur, snouts, legs, torsi, eyes, and ears. Is it true that abstractions of these types occur in the layers of the InceptionV1? If so, how many layers does it take for the representation to specialize so that the differences between the concepts are clearly visible? The following examples from mixed4a, mixed4b, and mixed4c layers (Figures 56 to 58) show that there indeed are close similarities between the concepts, provided that the representation is extracted from the early enough part of the model.



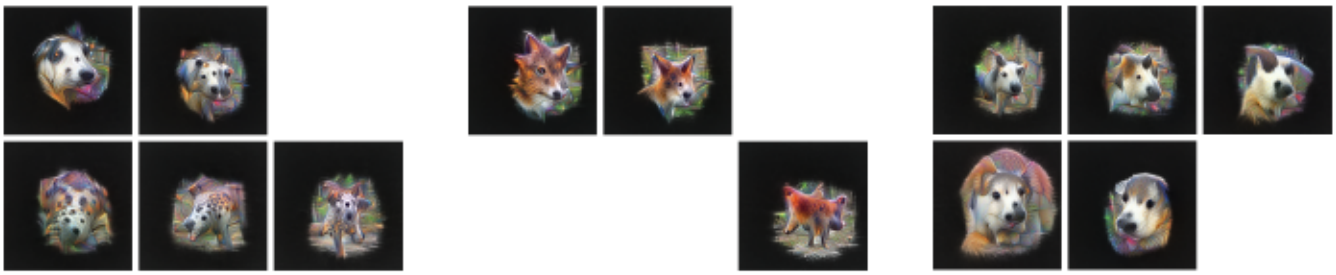
(a) Dalmatian (b) Red fox (c) Giant panda

Figure 56: Comparison of the concepts identified in mixed4a layer that attribute the most to the class of interest. Note how similar the core abstractions are: they all seem to depict proto-snouts with a multitude of eyes and noses. There are, however, subtle differences in textures that correspond to the target classes: "dots" for a dalmatian category, or clear "red fur" for the fox.



(a) Dalmatian (b) Red fox (c) Giant panda

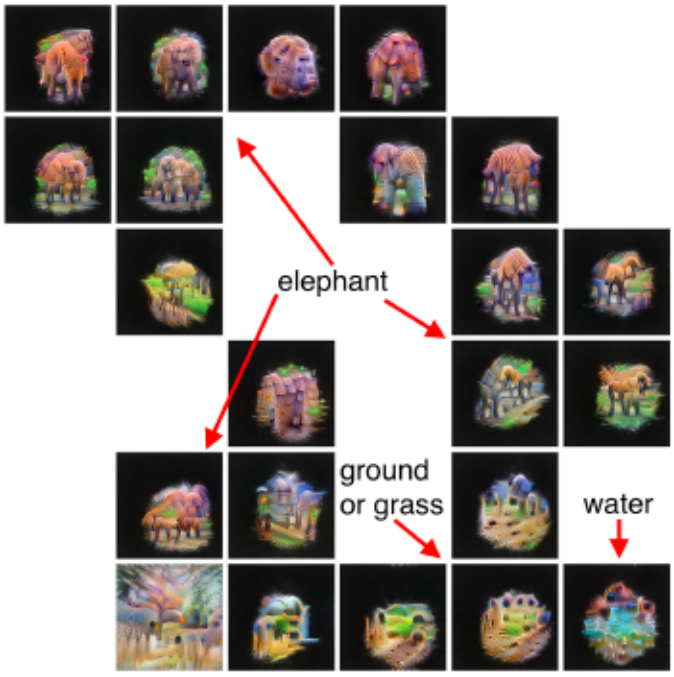
Figure 57: Comparison of the concepts identified in mixed4b layer that attribute the most to the class of interest. Concepts begin to specialize, with clearer distinction between the categories.



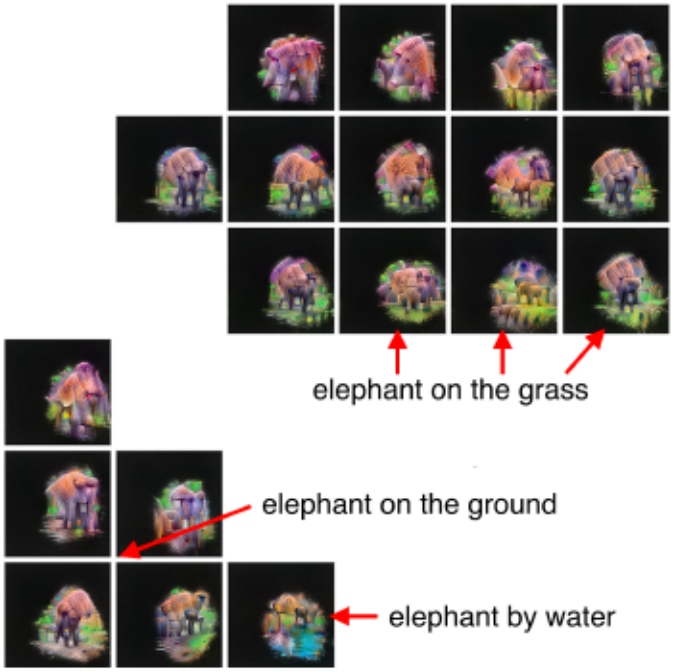
(a) Dalmatian (b) Red fox (c) Giant panda

Figure 58: Comparison of the concepts identified in mixed4c layer that attribute the most to the class of interest. At this point, the abstractions that influence the classification the most (in a given layer) differ significantly between examined categories. Specialized snout detectors appear across all three classes presented in the figure.

The final use case of the Distilled Class Factors Atlas presented in this section relates to quite specific behavior of the network that can be observed between mixed4e and mixed5a layers. A phenomenon that could be called "concept merging" manifests itself when comparing the atlases of the same class generated for mixed4e and mixed5a layers. All of the scenarios were analyzed in detail and put as a caption for Figures 59 to 61 below. However, here is an example of what "concept merging" means: if there are "elephant", "grass/ground" and "water" concepts in the mixed4e layer for the "african elephant" class, then it is interesting to observe "elephant on the grass", "elephant on the ground" and "elephant by water" concepts in the atlas for a mixed5a layer. It is as if the weights of the mixed5a layer assembled the incoming concepts into the more complex ones (if it is true, it would be very similar to the idea of circuits [11]).

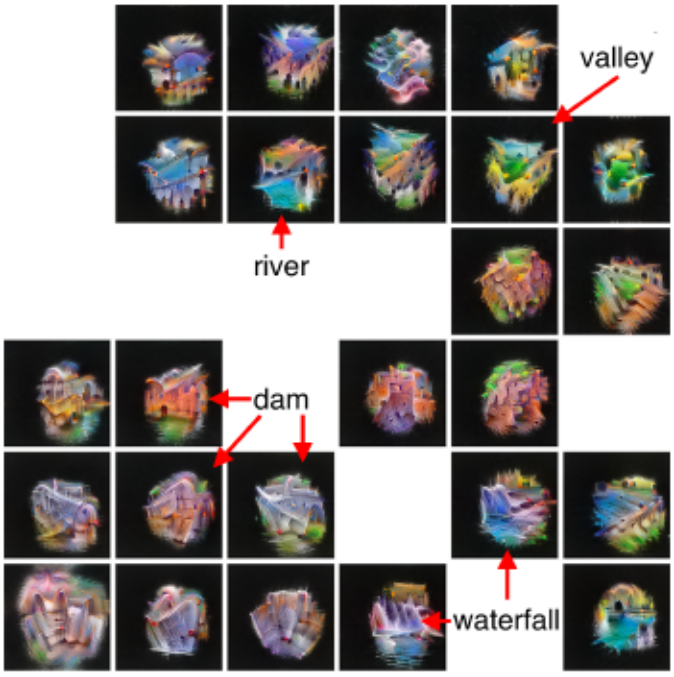


(a) Layer: mixed4e

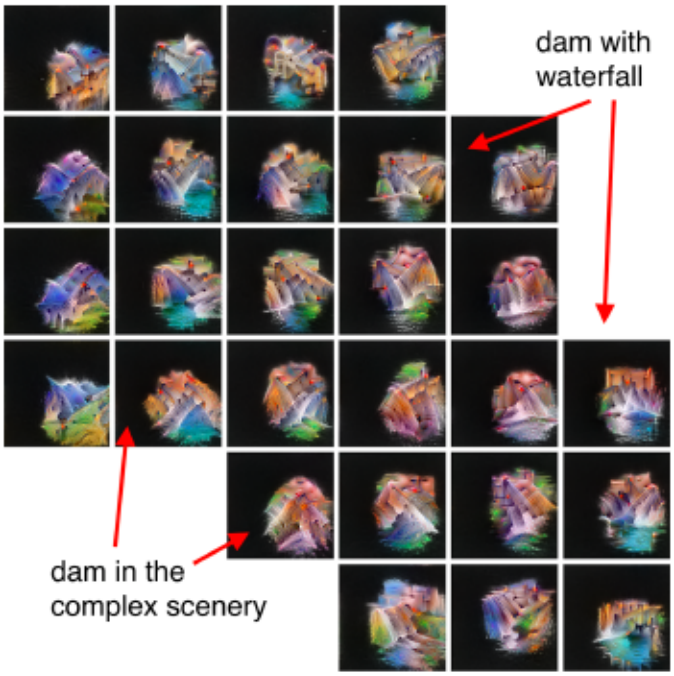


(b) Layer: mixed5a

Figure 59: Concept merging phenomenon for the "african elephant" class. Core concept (elephant) is merged with the more contextual ones (ground, grass and water) in the mixed5a layer, yielding the "elephant on the grass", "elephant on the ground" and "elephant by water" abstractions. Interestingly, the core concept is present across all icons in the atlas of the mixed5a layer.

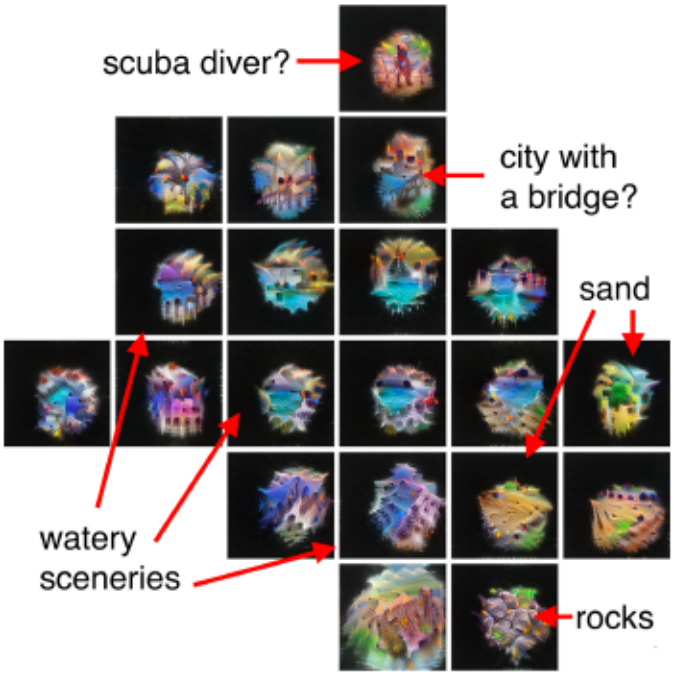


(a) Layer: mixed4e

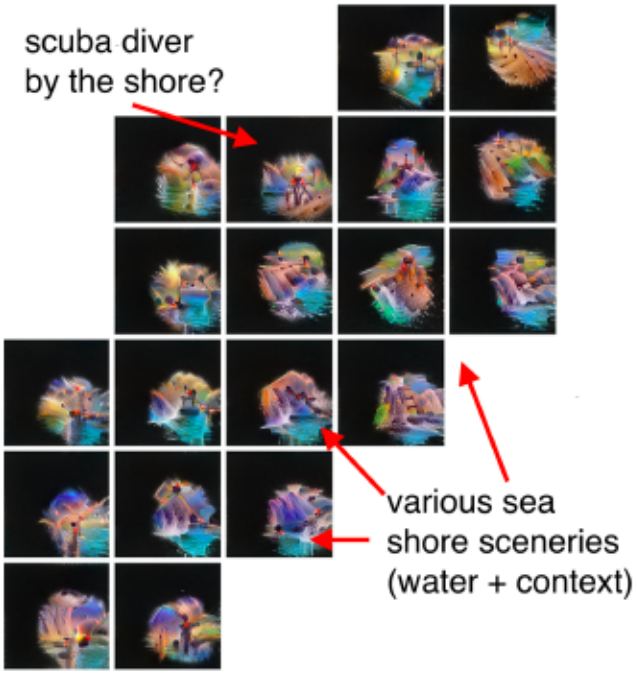


(b) Layer: mixed5a

Figure 60: Concept merging phenomenon for the "dam" class. Although in this case it is harder to assign names to the concepts visible in the mixed4e atlas, one can still notice a river, valley and waterfall abstractions, together with the core concept—dam. Mixed5a layer merged the "dam" abstraction with the remaining ones, producing "dam with a waterfall" and a multitude of others, collectively called "dam in the complex scenery" in the figure.



(a) Layer: mixed4e



(b) Layer: mixed5a

Figure 61: Concept merging phenomenon for the "sea shore" class. In this example, it is quite easy to detect "sand", "rocks" and "watery sceneries". Arguably, there appear to exist "city with a bridge" and "scuba diver" concepts, yet it is not totally clear. After concept merging in mixed5a layer, the resulting atlas contains a wide variety of sea shore sceneries (essentially: water in different contexts). One can spot a visualization resembling "scuba diver by the shore" abstraction as well, although it is quite moot.

Distilled Class Factors Atlas is not flawless. The major limitation of a proposed interface is the fact that has already been mentioned before: it cannot, by design, identify concepts correlated with a class of interest that do not appear in the images belonging to that class. Even though the Distilled Class Factors Atlas deliberately limits the analysis and requires examples to be of the same type, it is important to acknowledge such a constraint. If one's interest is to conduct a broader study, however, Class Activation Atlas is a good candidate and should be used instead.

Another difficulty while working with Distilled Class Factors Atlases is the fact that their generation process is a computationally demanding task. When using a single GeForce GTX 1080 Ti, it takes between 2 to 5 minutes to obtain the atlas, depending on the number of icons. Before a severe code optimization, it took up to 30 minutes (!) to get a single visualization. Parallelizing the procedure by making it possible to optimize for the batch of icons simultaneously led to a significant decrease in the computation, enabling widespread data generation.

Despite providing a new perspective on the latent representations of a CNN, Distilled Class Factors Atlas has a flaw, at least a part of it. The problem is that icons that are there to reveal the visual meaning of data points from a projected manifold often illustrate concepts that are a bit ambiguous in terms of their interpretation, especially for later layers of the network (like mixed5b). In the current form, the interface does not offer any additional details related to feature visualizations that could help identify the true nature of the abstraction. A good example (and a potential extension) would be to display subset images the concepts in a given cell were extracted from, together with the corresponding latent spatial factors. Combining feature visualization with image samples and spatial information would increase the method's interpretability.

Chapter 5

Conclusions and Future Work

In this work, the objective was to uncover the world of inner representations constructed by a single convolutional network—InceptionV1. The choice of a model was based on the conclusions drawn from the preceding research [12, 48, 49]: the InceptionV1 uses concepts that are particularly easy to interpret when visualized with various feature visualization techniques.

To meet this objective, two novel techniques were proposed in the thesis: Latent Factor Attribution and Distilled Class Factors Atlas. Thanks to the insights they provided, they made it possible to see the world through the model's eyes and, to some extent, understand how it comes to the specific outputs given an input image.

Latent Factor Attribution is a mid-level attribution method that allows for quantification of the importance of latent factors identified in the hidden layers' activation tensors by matrix decomposition approaches like NMF and PCA. When using LFA, it is possible for the CNNs' interpretability researchers to develop new intuitions on how visual reasoning occurs in the convolutional networks, especially in terms of what contributes to the complex conclusions that are often regarded by laymen as "magical". Activation tensor decomposition, when coupled with feature visualization and LFA, shines new light on how the information is represented in the entire groups of neurons. Through the inspiring work on distributed coding scheme in neuroscience, the field of machine learning interpretability is extended by LFA—a new building block that, when used in the right context, can yield captivating results, like the ones that were included in a supplementary interactive Activation Explorer interface.

Distilled Class Factors Atlas creates a summary of concepts that are strongly associated with a given class. It shifts the focus to class-wise analyses and helps to form conclusions covering the visual processing of entire categories, not merely individual samples. It extends the idea of Class Activation Atlas [12] to increase the feasibility of computation and reliability of the obtained

results. Here, feature columns are not randomly sampled to construct the atlas anymore. Instead, distinct concepts (latent factors) are extracted from a multitude of images belonging to the same class and are then visualized collectively. Combining the representational strength of Activation Atlases, together with the LFA-based attribution information, yields the interface that helps to keep the researcher's attention only on the regions of the atlas that truly matter for the classification of a class of interest.

There exist potential future research avenues that could either extend the Distilled Class Factors Atlas interface or apply it to the real-world problems. One idea is to leverage the technique to analyze the incorrectly predicted inputs to uncover the reasons for mistakes and, potentially, help fix the model's flaws. A proposed method could be an alternative debugging approach—instead of going through the code of the algorithm, researchers could reason about it on a different level of abstraction by directly inspecting the underlying representations. Potentially, this would provide the kind of information they need to redesign the architecture so that it constructs the concepts they want it to.

Another future goal could be to use the Distilled Class Factors Atlas to better understand the differences between two classes that are strongly related to each other, e.g. to understand in details how a convolutional network decides whether the input image belongs is the "african elephant" or the "indian elephant". Such insights would contribute to the increase in the trust of the model—if it truly attended the meaningful concepts, the "black box" would instantly become less opaque.

Due to limitations in computational resources and time, it was not possible to robustly extend the analysis presented in this thesis to other important convolutional networks. The natural next step would be to apply the proposed techniques on models like ResNet, or the EfficientNet family.

Last but not least, by observing the current initiatives like the *Circuits* project [11], the author of the thesis wonders whether careful decomposition-like approaches could be leveraged to robustly identify the neurons that act in a distributed way across multiple layers, as a result representing interpretable abstractions—circuits.

The contributions from the field of interpretability are truly hard to overestimate. Thanks to thorough and careful analyses, some important questions have already been answered. Novel research teaches humans a lot about visual processing that takes place in convolutional networks. Most importantly, though, one day, it might help us redefine how we perceive the mechanisms that occur in our brains daily, imperceptibly. Computer vision was supposed to be a summer project for a small group of students. It has been over 50 years from that time, and not only does it remain unsolved, but it constantly provides new insights that change the way we think about machine learning algorithms. That is the real beauty of science.

Chapter 6

Bibliography

[1] Julius Adebayo, Justin Gilmer, Ian J. Goodfellow, and Been Kim. Local explanation methods for deep neural networks lack sensitivity to parameter values. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net, 2018.

[2] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian J. Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 9525–9536, 2018.

[3] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[4] Diego Ardila, Atilla P Kiraly, Sujeeth Bharadwaj, Bokyung Choi, Joshua J Reicher, Lily Peng, Daniel Tse, Mozziyar Etemadi, Wenxing Ye, Greg Corrado, et al. End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography. *Nature Medicine*, 25(6):954–961, 2019.

[5] Baktash Babadi and Haim Sompolinsky. Sparseness and expansion in sensory representations. *Neuron*, 83(5):1213 – 1226, 2014.

[6] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7):1–46, 07 2015.

[7] Horace Barlow. Redundancy reduction revisited. *Network: Computation in Neural Systems*, 12(3):241–253, 2001.

[8] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network Dissection: Quantifying interpretability of deep visual representations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 3319–3327. IEEE Computer Society, 2017.

[9] Leo Breiman. Statistical modeling: The two cultures. *Statistical Science*, 16(3):199–231, 2001.

[10] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

[11] Nick Cammarata, Shan Carter, Gabriel Goh, Chris Olah, Michael Petrov, and Ludwig Schubert. Thread: Circuits. *Distill*, 2020. https://distill.pub/2020/circuits.

[12] Shan Carter, Zan Armstrong, Ludwig Schubert, Ian Johnson, and Chris Olah. Activation atlas. *Distill*, 2019. https://distill.pub/2019/activation-atlas.

[13] Aditya Chattopadhyay, Anirban Sarkar, Prantik Howlader, and Vineeth N. Balasubramanian. Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018, Lake Tahoe, NV, USA, March 12-15, 2018*, pages 839–847. IEEE Computer Society, 2018.

[14] Mark M Churchland, John P Cunningham, Matthew T Kaufman, Justin D Foster, Paul Nuyujukian, Stephen I Ryu, and Krishna V Shenoy. Neural population dynamics during reaching. *Nature*, 487(7405):51–56, 2012.

[15] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016*, 2016.

[16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255. IEEE Computer Society, 2009.

[17] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. Technical Report 1341, University of Montreal, 2009. Also presented at the ICML 2009 Workshop on Learning Feature Hierarchies, Montréal, Canada.

[18] David J. Field. What is the goal of sensory coding? *Neural Computation*, 6(4):559–601, 1994.

[19] Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 2950–2958. IEEE, 2019.

[20] Ruth C. Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 3449–3457. IEEE Computer Society, 2017.

[21] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

[22] Stefano Fusi, Earl K Miller, and Mattia Rigotti. Why neurons mix: high dimensionality for higher cognition. *Current Opinion in Neurobiology*, 37:66 – 74, 2016. Neurobiology of cognitive behavior.

[23] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, volume 9908 of *Lecture Notes in Computer Science*, pages 630–645. Springer, 2016.

[25] Fred Hohman, Haekyu Park, Caleb Robinson, and Duen Horng (Polo) Chau. Summit: Scaling deep learning interpretability by visualizing activation and attribution summarizations. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 26(1):1096–1106, 2020.

[26] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Xu Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. GPipe: Efficient training of giant neural networks using pipeline parallelism. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 103–112, 2019.

[27] Andrej Karpathy. t-SNE visualization of CNN codes. `https://cs.stanford.edu/people/karpathy/cnnembed/`, 2014. (Accessed on 11/23/2019).

[28] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie J. Cai, James Wexler, Fernanda B. Viégas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2673–2682. PMLR, 2018.

[29] Pieter-Jan Kindermans, Kristof T. Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. Learning how to explain neural networks: PatternNet and PatternAttribution. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114, 2012.

[31] Quoc V. Le, Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Greg Corrado, Kai Chen, Jeffrey Dean, and Andrew Y. Ng. Building high-level features using large scale unsupervised

learning. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012.

[32] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[33] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[34] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[35] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[36] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[37] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 5188–5196. IEEE Computer Society, 2015.

[38] Leland McInnes, John Healy, and James Melville. UMAP: Uniform Manifold Approximation and Projection for dimension reduction. *ArXiv e-prints*, February 2018.

[39] Ethan M. Meyers, David J. Freedman, Gabriel Kreiman, Earl K. Miller, and Tomaso Poggio. Dynamic population coding of category information in inferior temporal and prefrontal cortex. *Journal of Neurophysiology*, 100(3):1407–1419, 2008. PMID: 18562555.

[40] Grégoire Montavon, Sebastian Bach, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

[41] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. `https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html`, July 2015. (Accessed on 11/09/2019).

[42] Alexander Mordvintsev, Nicola Pezzotti, Ludwig Schubert, and Chris Olah. Differentiable image parameterizations. *Distill*, 2018. https://distill.pub/2018/differentiable-parameterizations.

[43] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 3510–3520. IEEE Computer Society, 2017.

[44] Anh Mai Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3387–3395, 2016.

[45] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 427–436. IEEE Computer Society, 2015.

[46] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. *CoRR*, abs/1602.03616, 2016.

[47] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. An overview of early vision in InceptionV1. *Distill*, 2020. https://distill.pub/2020/circuits/early-vision.

[48] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[49] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. *Distill*, 2018. https://distill.pub/2018/building-blocks.

[50] Daniel Omeiza, Skyler Speakman, Celia Cintas, and Komminist Weldemariam. Smooth grad-CAM++: An enhanced inference level visualization technique for deep convolutional neural network models. *CoRR*, abs/1908.01224, 2019.

[51] Seymour A Papert. The summer vision project. 1966.

[52] Karl Pearson. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[53] Shreya Saxena and John P Cunningham. Towards the neural population doctrine. *Current Opinion in Neurobiology*, 55:103–111, 2019.

[54] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359, 2020.

[55] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3145–3153. PMLR, 2017.

[56] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

[57] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*, 2014.

[58] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. SmoothGrad: removing noise by adding noise. *CoRR*, abs/1706.03825, 2017.

[59] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for simplicity: The all convolutional net. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015.

[60] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference*

on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR, 2017.

[61] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1–9. IEEE Computer Society, 2015.

[62] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[63] Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 2019.

[64] Mike Tyka. Class visualization with bilateral filters. `https://mtyka.github.io/deepdream/2016/02/05/bilateral-class-vis.html`, Feb 2016. (Accessed on 11/09/2019).

[65] Donglai Wei, Bolei Zhou, Antonio Torralba, and William T. Freeman. Understanding intra-class knowledge inside CNN. *CoRR*, abs/1507.02379, 2015.

[66] Rafael Yuste. From the neuron doctrine to neural networks. *Nature Reviews Neuroscience*, 16(8):487–497, 2015.

[67] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Richard C. Wilson, Edwin R. Hancock, and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*. BMVA Press, 2016.

[68] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, volume 8689 of *Lecture Notes in Computer Science*, pages 818–833. Springer, 2014.

[69] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2921–2929. IEEE Computer Society, 2016.

[70] Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. Interpretable basis decomposition for visual explanation. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VIII*, volume 11212 of *Lecture Notes in Computer Science*, pages 122–138. Springer, 2018.

[71] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8697–8710. IEEE Computer Society, 2018.

[72] Audun Øygard. Visualizing GoogLeNet classes. `https://www.auduno.com/2015/07/29/visualizing-googlenet-classes/`, July 2015. (Accessed on 24/06/2020).