Applied Soft Computing Journal 150 (2024) 111094

Contents lists available at ScienceDirect

Applied Soft Computing

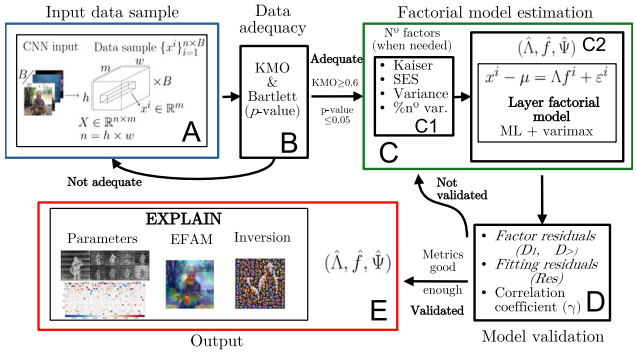
journal homepage: www.elsevier.com/locate/asoc

Layer factor analysis in convolutional neural networks for explainability

Clara I. López-González [a,*], María J. Gómez-Silva [b], Eva Besada-Portas [b], Gonzalo Pajares [c]

[a] Department of Software Engineering and Artificial Intelligence, Complutense University of Madrid, Madrid, 28040, Spain
[b] Department of Computer Architecture and Automation, Complutense University of Madrid, Madrid, 28040, Spain
[c] Institute for Knowledge Technology, Complutense University of Madrid, Madrid, 28040, Spain

GRAPHICAL ABSTRACT



ARTICLE INFO

Keywords:
Deep learning
Explainable artificial intelligence
Statistical modeling
Visual explanation
Feature learning
Attribution map

ABSTRACT

Explanatory methods that focus on the analysis of the features encoded by Convolutional Neural Networks (CNNs) are of great interest, since they help to understand the underlying process hidden behind the black-box nature of these models. However, to explain the knowledge gathered in a given layer, they must decide which of the numerous filters to study, further assuming that each of them corresponds to a single feature. This, coupled with the redundancy of information, makes it difficult to ensure that the relevant characteristics are being analyzed. The above represents an important challenge and defines the aim and scope of our proposal. In this paper we present a novel method, named Explainable Layer Factor Analysis for CNNs (ELFA-CNNs), which models and describes with quality convolutional layers relying on factor analysis. Regarding contributions, ELFA obtains the essential underlying features, together with their correlation with the original filters, providing an accurate and well-founded summary. Through the factorial parameters we gain insights about the information learned, the connections between channels, and the redundancy of the layer, among others. To provide visual explanations in a similarly way to other methods, two additional proposals are made: a) Essential Feature Attribution Maps (EFAM) and b) intrinsic features inversion. The results prove the effectiveness of the developed general methods. They are evaluated in different CNNs (VGG-16, ResNet-50, and DeepLabv3+) on generic datasets (CIFAR-10, imagenette, and CamVid). We demonstrate that convolutional layers adequately fit a factorial model thanks to the new metrics presented for factor and fitting residuals (D_1, $D_>$, and Res, derive from covariance matrices). Moreover, knowledge about the deep image representations and the learning process is acquired, as well as reliable heat maps highlighting regions where essential features are located. This study effectively provides an explainable approach that can be applied to different CNNs and over different datasets.

* Corresponding author.
E-mail addresses: claraisl@ucm.es (C.I. López-González), mgomez77@ucm.es (M.J. Gómez-Silva), ebesada@ucm.es (E. Besada-Portas), pajares@ucm.es (G. Pajares).

https://doi.org/10.1016/j.asoc.2023.111094
Received 20 July 2023; Received in revised form 28 October 2023; Accepted 19 November 2023
Available online 22 November 2023

1. Introduction

Convolutional Neural Networks (CNNs) have been a major breakthrough in the field of image processing, achieving great performance in a wide range of tasks, such as classification [1,2], object detection [3] or semantic segmentation [4]. However, the large number of interconnected nonlinear parts are the cause of both the high performance and the black-box nature of these models, making interpretation and understanding of decision making difficult. In this regard, explanatory strategies have been developed in recent years within the framework of explainable Artificial Intelligence (xAI, [5]).

Feature learning methods have proven to be remarkably useful in deep learning explanations [6–9]. Identifying the features detected by the layers and units of a network helps to understand the latent representation of the input images, the evolution of network knowledge, and to reveal interpretability [10]. Besides, the insights obtained could be used to verify and improve the neural models [11,12].

In order to make the explanations easily understandable to humans, these methods mainly rely on visual representations. Some focus on visualizing filters or channels activation maps, or sets of them [7,11]. Others go back to the input space, finding images that represent the channel encoded feature [6,13]. Attribution-based methods [14], one of the most popular, assign values to each location (either in the input image or in some intermediate activation map) accounting for their contribution to model's prediction. They can be visualized as heat maps that highlight regions which correspond to relevant learned characteristics [9,15–17]. Note that, since each filter gives rise to an output channel, filters and channels will be used interchangeably.

In general, the main problem in using these approaches to explain the features and knowledge gathered in a given convolutional layer, is that either all or some handpicked filters are studied. This makes the process of finding relevant features difficult, due to the high number of existing filters and the redundancy in the information extracted, having to rely on trial and error. Even when the choice is based on certain criteria, e.g. the strongest activations [11], the associated features need not be the only important ones in the learning process. Moreover, each filter is assumed to be associated with a single feature, even though it has been proven incorrect [18].

This motivates our work and defines its scope, where we derive the inherent features underlying a convolutional layer and quantify their influence in each of the channels through a mathematically well-founded explanatory strategy. It entails a substantial improvement over the methods identified in the literature according to their limitations, discussed above and comparatively below and in Section 2, and has a significant impact in xAI.

In particular, we propose a novel statistical xAI approach, named Explainable Layer Factor Analysis for CNNs (ELFA-CNNs), based on Factor Analysis (FA, [19]). This method finds a few essential features, called factors, that are hidden behind the original variables (in our case the channels or filters) and that cause them to correlate. For each convolutional layer, ELFA estimates with a single forward pass a factor model, whose parameters provide not only the latent features but also information about useless filters, layer redundancy, and channels importance. As a result, and referring to the above paragraph, the proposed method avoids having to choose neurons to explain the features learned by the layer, the mathematical model already takes care of finding the underlying characteristics. Besides, they are considerably less numerous than filters and are not associated with individual channels, but their contribution to each of them is known.

Furthermore, visual representations of these essential features, similar to those in the literature, can be defined, which aids human interpretation and helps to gain further insights of the learning process. On the one hand, unobserved activation maps, associated with the essential features, can be obtained and visualized for any input image. By comparing them with channels activation maps, relationships can be discovered, that help to understand decision making. On the other hand, the inversion of intrinsic features up to the input space is conducted. Finally, inspired by the attribution-based approaches, we define Essential Feature Attribution Maps (EFAM), that emphasize areas which contain characteristics important to that layer.

Given that we search explainability, factor analysis is chosen among other standard reduction methods, such as Principal Component Analysis (PCA). The former provides features that explain exactly the common variance between the original variables, and hence the relationships between them, and independent errors that account for the specific variance. Meanwhile, PCA is not able to make this division, giving an imprecise explanation of the contributions and connections between variables.

Regarding the effectiveness of the results, the proposed ELFA-CNNs is applied to the analysis of classification and semantic segmentation networks, proving its generality. Namely VGG-16, ResNet-50, and DeepLabv3+, on generic datasets. By defining some quality metrics, we demonstrate that adequate factorial models can be estimated for the convolutional layers, leading to a summary in essential features that ease the explainability of the network. What is more, the analysis performed on the model's parameters reveals redundant and related filters, as well as network's decisions. The heat maps associated with the underlying features are qualitatively and quantitatively compared with analogous state-of-the-art attribution maps. Accurate and concise visual explanations, which emphasize essential parts, are obtained, verifying the usefulness of our approach.

The main contributions of this paper are the following:

- The design of a novel explanatory method for convolutional neural networks, called ELFA-CNNs, based on a mathematically well-founded finding of the essential underlying features of a layer, by estimating factorial models with a single forward pass.
- We show that convolutional layers can be described by a factorial model of quality, and propose new metrics to establish that validity.
- The summary of a convolutional layer in inherent features, from which those represented in the channels are derived (thus not constraining each filter to one feature), that explain the correlation between channels and provide information about how they are related, their importance, redundancy, and usefulness.
- The integration of our novel strategy into the usual framework of explanation visualization by presenting unobserved activation maps and proposing the Essential Feature Attribution Map (EFAM), which reveals regions with significant information for the layer, and intrinsic features inversion, that describes such features in the input space.
- The explanatory analysis of different CNNs on distinct classification and segmentation datasets, which demonstrates the validity of the proposed general strategy and provides insights into the learning process.

In short, the proposed strategy effectively provides a general explainable approach, that can be applied to different CNNs and over different datasets, and that disentangles and individually investigates the underlying features.

The paper is organized as follows. Section 2 revisits feature learning explanatory methods. Section 3 presents the theoretical background. Section 4 introduces ELFA-CNNs and the general strategy followed. Section 5 addresses the experiments conducted and analyzes the validity of our approach. Section 6 further explores the explainability of convolutional neural networks via the factorial parameters, EFAM, and intrinsic features inversion proposals. Finally, Section 7 draws the conclusions and introduces some future work.

2. Related work

Explanatory methods for convolutional neural networks on image datasets are one of the most proliferating branches of xAI. Identifying

the features detected by CNNs is crucial to understand the representations learned by these deep architectures and how they are involved in decision making. As a result, strategies focused on the analysis of layer-coded features have been developed. These explanatory approaches rely on visual representations of the characteristics, due to their ease of interpretation [20]. As shown in [7], simply visualizing the activations of each layer helps to gain intuition on how the network works. In [6] they generate representative input images of a class that maximize class score. Since not only the last layer is significant, the work in [13] inverts inner layered image representations through an optimization problem, while [11] uses a deconvolutional network to get back to the input pixel space. Moreover, [7] finds images that maximize the activation maps.

Attribution-based methods have been used to highlight image pixels that contribute the most to the layer output. The impact may be positive or negative, as shown in the Layer-wise Relevance Propagation approach (LRP, [15]). The latter finds pixel-wise relevances pointing out contributions to the classification score. When dealing with the last layer, Class Activation Maps (CAM, [21]) identify areas which are relevant to the prediction through a linear weighted combination of activation maps. Nevertheless, the fully-connected layer needs to be replaced by a convolution and global average pooling. This may degrade performance, so gradient-based methods are often used instead, e.g. Grad-CAM [8] which measures importance with positive gradients. Better single and multiple object localization is achieved with Grad-CAM++ [22], which is defined by a weighted instead of a global average of gradients.

While saliency maps [6,16] only calculate image-specific class saliencies, Grad-CAM also detects the regions that activate the most the output of an inner layer. To capture more fine-grained information and improve localization performance, LayerCAM [17] uses weights for each spatial location in the feature map. However, gradients calculation needs back-propagation. Consequently, new techniques try to generalize CAMs, such as Score-CAM [9]. To weight maps, it employs the difference on the output when a binary mask retains only the features of interest in the input. In general, heat maps are followed by a ReLU to capture positive contributions.

On the one hand, visualizing activations alone is somewhat limited in explaining the global behavior of the network. To obtain a general picture, works such us [23,24] summarize the learned features and how they interact to make predictions. On the other hand, these methods assume that each channel specializes in learning one specific characteristic. As a consequence, choices on which filters should be analyzed are necessary. Although all of the previous methods provide valuable information, the multiple filters and the redundant information make it difficult to assure that the important features are selected and studied. As explained above, ELFA-CNNs solves these issues by finding mathematically well-founded essential features underlying the convolutional layer. They also explain the correlation between channels and the redundancy of the layer, providing an accurate summary.

Regarding attribution maps, we adopt a different point of view, computing heat maps which highlight regions relevant to the layer itself. Therefore, the center of attention is not what information is significant when transmitting it to subsequent layers (which influences the output more), but rather what essential (underlying) information is detected and encoded in that layer. The EFAM is calculated using the parameters of the factorial model previously estimated, being computationally inexpensive.

Relying upon statistical theory rather than machine learning [25,26], as done in ELFA, is considered to lead to more verifiable, explainable, and general strategies, as stated in [27]. Although this work also uses factor analysis, it is only applied on the training set to uncover the class representatives. The main explanatory strategy is completely different from ours. It is based on computing mutual information, which in turn is used to determine the probability that a new sample belongs to a given class. Excluding this article, the uses of FA in deep learning are outside the explanatory scope. They cover pre-processing, post-processing [28,29], and dictionary learning [30,31].

All in all, our statistically based approach does not limit each neuron to one feature, guarantees the summary of the layer in the underlying and essential features (explaining the correlation between channels), and detects the input areas to which the latter belong.

3. Mathematical background

This section presents a brief introduction to factor analysis and explains how to proceed in general when applying this method on a data sample. We refer interested readers to [32,33] for additional details. Later on, in Section 4, we will focus on image processing, which is our case.

3.1. Introduction to factor analysis

Factor analysis has its origins in the work of Pearson and Spearman [19]. It is a statistical method whose aim is to explain the correlation between observed variables in terms of a smaller unobserved set. One assumes that the measured variables derive from a few variables that could not be measured, and that the former are linear combinations of this hidden or latent variables, called factors, plus a certain observation-specific perturbation. That is, the observed variables are imperfect expressions (or indicators) of those latent factors. We focus on Explanatory Factor Analysis (EFA), where no prior knowledge about the number or nature of these factors is presumed.

In particular, let $x \in \mathbb{R}^m$ be a vector of random observed variables. The factor analysis model establishes the relation $x - \mu = \Lambda f + \varepsilon$, where $\mu \in \mathbb{R}^m$ is the variables mean and $f \in \mathbb{R}^p$ are the factors, with $p < m$. In order to obtain a representation as useful as possible, factors are supposed uncorrelated and Gaussian distributed $f \sim \mathrm{N}_p(0, I)$, no providing redundant information. Besides, $\varepsilon \in \mathbb{R}^m$ is a vector of unobserved perturbations or errors, uncorrelated with the factors. They are Gaussian distributed with zero mean and covariance equal to $\Psi = \mathrm{diag}(\psi_1, \dots, \psi_m)$, i.e. $\varepsilon \sim \mathrm{N}_m(0, \Psi)$, so they are also uncorrelated between them. Two results are deduced from this. Firstly, factors are independent of x and can be considered as new variables. Secondly, FA is able to divide the covariance into common and specific parts. The former is explained by the latent factors through the loading matrix $\Lambda \in \mathcal{M}_{m,p}$, whose coefficients describe how they influence the measured variables. Meanwhile, the errors collect the effect of all the variables other than the factors that have an effect on x. Indeed, Λ equals the covariance matrix between the factors and the observed variables, or the correlation matrix if x is normalized.

When considering a simple random sample of n elements $\{x^i\}_{i=1}^n \subset \mathbb{R}^m$, the relation for the i-sample of the jth variable is written as

$$x_j^i - \mu_j = \lambda_{j1} f_1^i + \cdots + \lambda_{jp} f_p^i + \varepsilon_j^i, \qquad 1 \le i \le n, \quad 1 \le j \le m, \tag{1}$$

where λ_{jk} are the loading matrix coefficients and $f^i \in \mathbb{R}^p$ are the values of the factors on the ith sample element. Hence, the effect of the latent factors over the jth variable is determined by the jth row of Λ, $(\lambda_{j1}, \dots, \lambda_{jp})$. Note that this is valid for all samples, since it does not depend on i.

Fig. 1 shows the flowchart of the general process, from the time the random sample is considered (box A) until the factor analysis model that governs it is obtained (box E). The dotted boxes should be omitted for the moment, since they are particularities of our neural network framework that will be explained in Section 4. Each of the steps (boxes) is described in detail in the sections that follow, as displayed in the headings.

It should be mentioned that the solution to the parameters (Λ, f, Ψ) of the factorial model does not have a closed form. Instead, they are estimated $(\hat{\Lambda}, \hat{f}, \hat{\Psi})$ via iterative algorithms (which is indicated as box C in Fig. 1), such as the principal factor method or the maximum likelihood approach [32] (box C2). Even though EFA does not presuppose knowledge about the factors, depending on the algorithm, a choice on the number p could be necessary (box C1). Several strategies appear in the literature, and are analyzed in the next section. Moreover, the

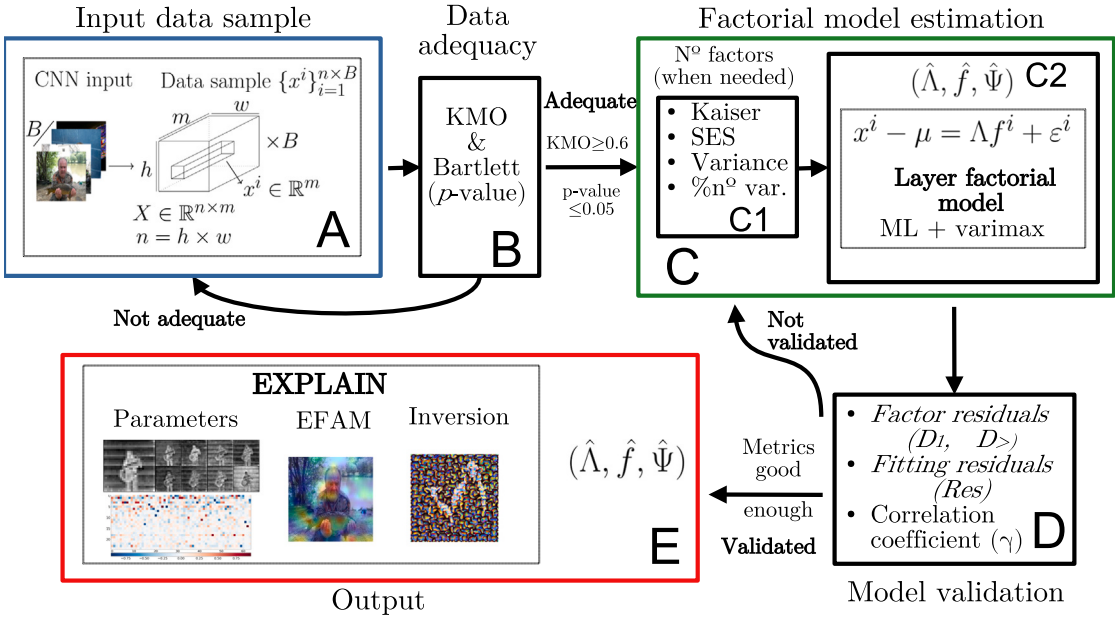


Fig. 1. Flowchart of the ELFA-CNNs strategy. Excluding the dotted boxes would result in the general factor analysis strategy presented in Section 3. Except when the data is not adequate or the model is not validated, the rest of the flow is forward.

model is undetermined, in the sense that any rotation of the factors would lead to an equivalent model. Rotations may be worthwhile in order to obtain more interpretable solutions. Among the existing methods, we center our attention in the varimax strategy [34] (pointed out in box C2, Fig. 1), whose aim is that the factors which affect some variables do not affect the others and vice versa.

3.2. Implementation of explanatory factor analysis

Before assuming a factor analysis model for the data, and even while implementing it, some verifications need to be done. In this section we present the general steps followed, which are exposed in the flowchart of Fig. 1 as already mentioned (excluding dotted boxes, in which the process is customized for the images).

3.2.1. Adequacy of the data (box B, Fig. 1)

Prior to obtaining the factor analysis model, it is necessary to examine the adequacy of the sample and the suitability of the data, as explained in [35]. This can be assessed by the Kaiser–Meyer–Olkin test (KMO, [36]), which examines the partial correlation between variables, being values over 0.6 suitable for EFA. Bartlett's sphericity [37] is used to test the null hypothesis that the correlation matrix of the variables is the identity. A p-value smaller than 0.05 is desired. If the data is not adequate, new samples should be obtained, e.g., by varying the sample size (returning to box A from B).

3.2.2. The number of factors (box C1, Fig. 1)

As mentioned before, the number of factors may need to be determined. Although there are many criteria, they do not always lead to the same or similar results, and often multiple of them are used. This work considers the following ones. Firstly, the Kaiser's method [38], which may be the most used in practice due to its simplicity. It consists in retaining as many factors as eigenvalues are greater than one in the data covariance matrix. Secondly, the popular Cattell's Scree test [39], that entails a visual examination of the graphical representation of the eigenvalues. The number of factors equals the number of points above the breaking point of the graph, which is supposed to separate important from minor or trivial factors. Because of its limitations, quantitative variants have arisen. Among them, we focus on the standard error scree (SES) approach proposed in [40] and based on Cattell's guideline that scree points should fit tightly. Another option is to retain enough factors to account for a certain percentage of the variation, known as the variance criterion. In addition, number of factors equal to the 25%, 50%, and 75% of the number of variables are also considered.

3.2.3. Factorial model fit (box D, Fig. 1)

Once the factor analysis model is estimated (box C, Fig. 1), its suitability should be checked and, if not adequate, the number of factors may be increased and the parameters re-estimated (returning to box C from D). In order to analyze this, we present the following validity metrics, all of them defined through the Frobenius norm.[1] In the flowchart of Fig. 1, the metrics proposed are in italics to distinguish them from those already existing:

1. *Factor residuals*. Note that the covariance matrix of the errors predicted by the model $\hat{\Psi}$ should be diagonal. To evaluate how diagonal it is, we define two measures, D_1 and $D_>$, as
$$D_1 = \frac{\|\text{diag}(\hat{\Psi})\|}{\|\hat{\Psi}\|}, \quad D_> = \frac{\|\text{diag}(\hat{\Psi})\|}{\|\hat{\Psi} - \text{diag}(\hat{\Psi})\|}. \tag{2}$$
The greater the value of $D_>$ the better, while the value of D_1 should be close to 1.
2. *Fitting residuals*. The difference between the sample correlation matrix S' and the one estimated by the model $\hat{S}'$ should be as small as possible. Therefore, we define the measure Res = $\|S' - \hat{S}'\|/(\|S'\| + \|\hat{S}'\|)$. It takes values between 0 and 1, the smaller the better.
3. *Correlation coefficient*. One of the properties of the factor analysis model is that the sample covariance matrix satisfies $S = \Lambda\Lambda^\top + \Psi$. Thus, the variance of the jth variable is given by $\sigma_j^2 = \sum_{k=1}^{p} \lambda_{jk}^2 + \psi_j^2 = h_j^2 + \psi_j^2$. The first term h_j^2, called communality, accounts for the factors influence, while the last one contains the specific error's impact. The squared correlation coefficient is defined as $\gamma_j^2 = h_j^2/\sigma_j^2$, for each $1 \le j \le m$, with values between 0 and 1. For the model to be considered well fitted, it must be close to one, although no particular value is specified in the literature.

[1] Given a matrix A, the Frobenius norm is defined as $\|A\| = (\sum_{i,j} |a_{ij}|^2)^{1/2}$.

Depending on the variables studied and the results obtained, we can be more or less lax with the values of the validity metrics when considering a general factor analysis approach. It should be mentioned that what is important is that the estimated model is useful for our purpose, under certain minimum quality or fit requirements. Thus, for example, in some cases all estimated models may verify $D_1 > 0.9$ (we know that the closer to 1 the better), while for other variables we may consider $D_1 > 0.7$ good enough. However, to derive results from a factorial model with $D_1 < 0.3$ would not be appropriated, and they would not be trustful. The validity metric bounds considered in this work are discussed in Section 5.1.3 and Appendix A.3.

4. Explainable layer factor analysis for CNNs

Our proposal consists in exploiting the factor analysis model for xAI in CNNs. How we apply the previous mathematical concepts on convolutional layers is explained first, and the insights gained by doing this afterwards. Finally, the complete ELFA-CNNs strategy is presented. All this is shown in the flowchart of Fig. 1, highlighting the CNN framework with dotted boxes.

4.1. Layer factor analysis model

Consider a convolutional layer with m filters. Each filter extracts a feature that is encoded in the corresponding output channel or activation map. We propose to treat each of the m output channels as a variable and assume a factor analysis model. That is, we suppose that there exist $p < m$ latent factors, which codify certain features, so that the channels of the convolutional layer can be written as a linear combination of these factors. As a result, the original features can be recovered from this latent features, except for a small error. The factors can be interpreted as unobserved activation maps that encode the main features that summarize the convolutional layer, giving a clearer idea of what the network is learning.

Mathematically, given a CNN and an input image, let C be a convolutional layer and $X \in \mathbb{R}^{h\times w\times m}$ its output, which contains m activation maps with $n = h \times w$ values each. For the sake of clarity, we will call them pixels instead of values hereinafter. It can be written as $X = (x^i)_{i=1}^n$, where $x^i = (x_1^i, \dots, x_m^i)^\top$ and each x_j^i is the value of the ith pixel in the jth channel, where j varies from 1 to m. In analogy with Section 3.1, $x^i \in \mathbb{R}^m$ is a vector of observed variables and $\{x^i\}_{i=1}^n \subset \mathbb{R}^m$ is a sample of n elements. If a batch of B input images is considered, the output of the convolutional layer would lead to $\{x^i\}_{i=1}^N \subset \mathbb{R}^m$, where $N = n \times B$. This proposal for the interpretation of variables and data extraction is shown in box A of Fig. 1. The factor analysis model establishes that the values of the ith pixel along the different channels are given by $x^i - \mu = \Lambda f^i + \varepsilon^i$, that is,

$$\begin{pmatrix} x_1^i - \mu_1 \\ \vdots \\ x_m^i - \mu_m \end{pmatrix} = \begin{pmatrix} \lambda_{11} f_1^i + \cdots + \lambda_{1p} f_p^i \\ \vdots \\ \lambda_{m1} f_1^i + \cdots + \lambda_{mp} f_m^i \end{pmatrix} + \begin{pmatrix} \varepsilon_1^i \\ \vdots \\ \varepsilon_m^i \end{pmatrix}. \tag{3}$$

We observe the following:

1. Each column of the factor loading matrix, $\lambda_k = (\lambda_{1k}, \dots, \lambda_{mk})^\top$ with $1 \le k \le p$, determines the impact of the kth factor or latent feature on the channels.
2. Each row of the factor loading matrix, $\lambda_j = (\lambda_{j1}, \dots, \lambda_{jp})$ with $1 \le j \le m$, shows the influence of the different factors (features) on the jth channel or activation map.

The algorithms used to estimate the solution of the layer factor model are specified in Section 4.3 (box C, Fig. 1). In line with our interpretation of the factors as unobserved activation maps, and given one input image so $N = n$, $f^i = (f_1^i, \dots, f_p^i)$ would be the values of the ith pixel along the p channels. Therefore, by estimating the factors for that input, $\{\hat{f}^i\}_{i=1}^N$, the associated p latent activation maps are found.

4.2. Information obtained (box E, Fig. 1)

The proposed layer factor analysis model of a convolution let us gain insights about the information gathered and how the network learns, as explained in this section. Thus, the output of ELFA-CNNs consists of not only the estimated parameters of the model, but the explanations they provide, which is exhibited in box E of Fig. 1.

4.2.1. Factorial model parameters explainability (parameters, box E, Fig. 1)

Based on the above observations, the estimated factors are identified as the essential features of the layer, and can be visualized as activation maps, as shown in Section 5.2.2. They provide a (visual) summary of the encoded features which is not based on random selection or predefined importance criteria. On the contrary, the mathematical model employed guarantees that the original channels can be (almost) recovered and that their covariance is explained. Furthermore, as noted before, the factor loading matrix Λ determines the relationship between factors and channels. Therefore, from the rows and columns of this matrix, it is possible to analyze how channels are related, the presence of useless filters, channels importance, and layer redundancy. This is further discussed in Sections 5.2.2 and 6.1.

4.2.2. Essential Feature Attribution Maps (EFAM, box E, Fig. 1)

Given a convolutional layer, a heat map uncovering the essential features is obtained thanks to the loading matrix of the estimated factorial model as follows. Consider the latent factors and the associated unobserved activation maps $f_1, \dots, f_p$. For each kth factor, with $1 \le k \le p$, we define its importance or weight β_k as the sum of the absolute values of the elements of the associated column in the loading matrix. That is, $\beta_k = \sum_{j=1}^m |\lambda_{jk}|$, where m is the number of channels of the layer. Note that this can be understood as the influence of the factor on the layer. The greater the weight β_k, the more important the factor is. Then, we compute the weighted sum of these unobserved activation maps, obtaining a heat map $\sum_{k=1}^p \beta_k f_k$ which highlights those regions that correspond to the most important features. Analogously to other attribution methods, ReLU may be applied. The heat map is normalized and bilineal interpolation is used to recover input size. We call it Essential Feature Attribution Map (EFAM or EFAM-R if ReLU is considered) and results are shown in Section 6.2.

4.2.3. Intrinsic features inversion (inversion, box E, Fig. 1)

It is well-known that features become more abstract as we deepen in the network, and the same happens with the intrinsic features estimated. The loading matrix helps to better visualize them by reconstructing an input image in the following way. Assume, without loss of generality, that we study the feature represented by the kth factor. We find an input image whose convolutional output $X' \in \mathbb{R}^{n\times m}$ satisfies $x'^i = (x_1'^i \dots, x_m'^i) = (\lambda_{1k}, \dots, \lambda_{mk})$ for all $i \in \{1, \dots, n\}$, where the right hand side are the coefficients of the kth column of the loading matrix. This implies that the input found activates mainly the kth factor, as seen in (3). Hence, by visualizing this image we can deduce which intrinsic feature is being encoded, as done in Section 6.3. Gradient descent is employed to obtain $X' = \arg\min_X(\|X - \Lambda_k\|_2^2 / \|\Lambda_k\|_2^2)$, where $\Lambda_k \in \mathbb{R}^{n\times m}$ and $\Lambda_k(i, \cdot) = (\lambda_{1k}, \dots, \lambda_{mk})$ for all $i \in \{1, \dots, n\}$.

4.3. General ELFA-CNNs strategy

Consider a convolutional neural network. For each convolutional layer we proceed as explained below and shown in Fig. 1. First, given a batch of size B, a forward pass is performed to recover the output $X \in \mathbb{R}^{B\times n\times m}$ as presented in Section 4.1, i.e. $\{x^i\}_{i=1}^N \subset \mathbb{R}^m$ with $N = n \times B$ and $n = h \times w$ (box A in Fig. 1). This data is normalized, to have zero mean and unitary standard deviation, and its adequacy is evaluated following Section 3.2.1 (box B). If the data is suitable for explanatory factor analysis, we calculate the number of factors (Section 3.2.2, box C1). Then, the parameters of the corresponding

factorial model $(\hat{\Lambda}, \hat{f}, \hat{\Psi})$ are estimated, using the maximum likelihood algorithm and the varimax strategy (box C2). The quality of the model is examined through the metrics introduced in Section 3.2.3 (box D), and discarded if not adequate (see Section 5.1.3 and Appendix A.3). Finally, the results obtained are analyzed towards explainability and interpretability as described in Section 4.2 (box E). This study provides information and visualizations on the learning process.

5. Experiments and results

In this section we validate ELFA-CNNs on different convolutional networks and datasets. The experimental and parameter settings are presented first. Then, we analyze the estimated models and their adequacy. Besides, we examine the factorial model parameters obtained and the information they provide about the learning process. As a result, we demonstrate that convolutional layers can be summarized and explained with a suitable factorial model and few latent features. Section 6 explores this aspect further.

Finally, it is worth noting that all the experiments were conducted within Python[2] and executed on a i7-1165G7 CPU 2.80 GHz with 12 GB of RAM.

5.1. Experimental setting

The following pairs of CNNs and dataset are considered in order to validate our approach: VGG-16 [1] on CIFAR-10 [41], VGG-16 on imagenette [42], ResNet-50 [2] on imagenette, and DeepLabv3+ [43] on CamVid [44]. More detailed settings are given in the next sections.

5.1.1. Datasets

The CIFAR-10 dataset contains 60 000 images of 32 × 32 pixels, divided into 50 000 for training and 10 000 for testing, with 10 different classes. From the training set we randomly pick 10 000 images for validation. Moreover, random horizontal and vertical reflections are used for data augmentation.

The imagenette dataset was build as a subset of ImageNet [45] in order to quickly test algorithms. It consists of resized 320 × 320 pixel images, with 10 different classes, 9469 training images and 3925 validation images.

Finally, the Cambridge-driving labeled Video dataset (CamVid) consists of 701 annotated street-level images of size 720 × 960. We group the 32 classes into 11. Random horizontal reflection and translation of ±10 pixels is done for data augmentation. A random division with 60%, 20% and 20% for training, validation and testing is performed.

5.1.2. Neural networks

Experiments were carried out with VGG-16, which consists of 13 convolutional layers and 2 fully connected layers for classification. In the case of CIFAR-10, a modified version, which includes batch normalization and ReLU layers after each convolution, is employed.

We also use ResNet-50, a Residual Network with 50 layers. Each residual block consists of three convolutional layers and a residual connection. Two types are distinguished: one with a convolutional layer, followed by a batch normalization, and one with the identity.

Finally, it is worth noting that ResNet-50 and VGG-16 in the case of imagenette are models pre-trained on ImageNet [46]. On the contrary, DeepLabv3+ and VGG-16 on CIFAR-10 are trained from scratch. Concerning training options, after trial and error, we choose the sgdm optimizer with momentum of 0.9. In DeepLabv3+ we use 10^{-3} as learning rate, a weight decay of 5×10^{-3}, 30 epochs, mini-batchsize of 8, and validation patience of 4. VGG-16 learning rate is 10^{-2} and drops every 25 epochs by a factor of 3×10^{-3}. The weight decay is 10^{-3}, epochs are 80, and mini-batchsize is 128.

[2] The Factor Analysis class of scikit-learn is employed.

Table 1
Metric values chosen as optimal.

KMO	p-value	D_1	$D_>$	Res	γ_j^2 20th percentile
≥0.6	≤0.05	≥0.7	≥0.9	≤0.05	≥0.8

5.1.3. Factor analysis setting

We follow the ELFA general strategy presented in Section 4.3 with the metric values specified in Table 1. The first two (KMO and p-value), linked to the adequacy of the data, are taken from the literature (as mentioned in Section 3.2.1). For each convolutional layer, we determine a batch size B so that the KMO and Bartlett's tests of the corresponding input data sample satisfy what is shown in the aforementioned table. To properly set B, a thorough study on the possible dependencies of the factorial model on the batch size is conducted in Appendix A. We demonstrate that the number of factors is not affected by B, so there is no need to take them into account when deciding the batch size. Moreover, stability on model's validity metrics is achieved with enough samples. It turns out that the latter is accomplished by setting B to be the smallest number that verifies the data adequacy tests (perhaps by increasing it a bit). The exact values obtained following this reasoning for each CNN and dataset are reported in Table A.7.

Once the input data sample is acquired, we estimate a factorial model for each of the possible number of factors computed (by using the methods in Section 3.2.2) and calculate the quality metrics (Section 3.2.3). Among these models, the one that satisfies the requirements of Table 1 and has the lowest number of factors is chosen. In this case, the fitting thresholds are determined through a consensus between reasonable validity bounds and the metric values obtained in the experiments performed in Appendix A.3, in line with the explanation provided in Section 3.2.3. Fig. A.19 shows that almost all of the several experiments carried out verify these quality thresholds, independently of the layer, CNN or dataset. In addition, they lead to useful results, as will be demonstrated in the following sections.

Given the diversity among the considered CNNs (simple and complex architectures with residual connections) and datasets (classification and segmentation with different grades of detail and textures), we can infer the generalization of all the above reasoning (how to determine the batch size B and establish the model fitting bounds). Even so, for setups with different characteristics, such as character recognition, an adjustment may be needed, always under certain quality requirements as illustrated at the end of Section 3.2.3. Analogously to our approach, this could be done by computing the metrics for different input data and establishing reasonable thresholds based on the outcome. If almost all the values are too low, layer factor analysis approach is not appropriated.

5.2. Results

This section summarizes the factor analysis models obtained for each of the neural networks considered. The values of the fitness parameters of Section 3.2 (KMO, D_1, $D_>$, Res, and γ_j^2) demonstrate that our factorial approach provides suitable models. Besides, an overview of the estimated essential features is displayed using Fig. 2 images, proving the validity of ELFA-CNNs in terms of explainability. Section 6 discusses it in more detailed.

5.2.1. Regarding layer factorial models

For VGG-16, a factor analysis model is estimated for each convolutional layer. The sample data consists of the convolutional output for a batch of images of size B. When trained on CIFAR-10, $B = \lceil 80 \times m/n \rceil$ in the first layer and $B = \lceil 3 \times m/n \rceil$ otherwise, where m is the number of channels or filters and n the number of pixels of the corresponding activation maps. These values B are outlined in Table A.7. As mentioned before, we refer to the Appendix A for a

Table 2
Factor analysis model: factors and metrics for VGG-16 on CIFAR-10. In each layer, the method employed to compute the number of factors is given by: (K) Kaiser, (SES) Standard Error Scree method or (x%) just the x% of the given channels.

Layer	Channels	KMO	Factors	D_1	$D_>$	Res	γ_j^2 20th perc.
conv1	64	0.92	12 (SES)	0.50	0.57	0.0017	0.995
conv2	64	0.79	25 (SES)	0.71	1.01	0.0043	0.969
conv3	128	0.80	60 (SES)	0.70	0.98	0.0031	0.959
conv4	128	0.80	64 (SES)	0.73	1.06	0.0041	0.938
conv5	256	0.82	149 (SES)	0.70	0.98	0.0025	0.953
conv6	256	0.82	128 (50%)	0.73	1.06	0.0055	0.916
conv7	256	0.81	128 (50%)	0.73	1.07	0.0069	0.902
conv8	512	0.89	128 (25%)	0.70	0.96	0.0154	0.837
conv9	512	0.91	343 (SES)	0.74	1.11	0.0022	0.926
conv10	512	0.95	128 (25%)	0.74	1.09	0.0106	0.828
conv11	512	0.99	128 (25%)	0.78	1.24	0.0009	0.959
conv12	512	0.99	126 (SES)	0.75	1.13	0.0003	0.990
conv13	512	0.99	46 (SES)	0.62	0.79	0.0001	0.997

Table 3
Factor analysis model: factors and metrics for VGG-16 on imagenette.

Layer	Channels	KMO	Factors	D_1	$D_>$	Res	γ_j^2 20th perc.
conv1	64	0.87	20 (SES)	0.74	1.09	0.0031	0.969
conv2	64	0.88	25 (SES)	0.84	1.56	0.0055	0.923
conv3	128	0.91	68 (SES)	0.96	3.51	0.0036	0.903
conv4	128	0.79	73 (SES)	0.84	1.58	0.0087	0.802
conv5	256	0.87	128 (50%)	0.84	1.55	0.0072	0.801
conv6	256	0.85	161 (SES)	0.85	1.65	0.0062	0.834
conv7	256	0.82	163 (SES)	0.84	1.55	0.0074	0.809
conv8	512	0.82	370 (SES)	0.77	1.22	0.0034	0.902
conv9	512	0.77	352 (SES)	0.98	4.47	0.0015	0.979
conv10	512	0.72	388 (SES)	0.81	1.40	0.0023	0.938
conv11	512	0.83	380 (SES)	0.76	1.19	0.0035	0.880
conv12	512	0.80	281 (SES)	0.75	1.14	0.0027	0.914

complete comprehension of the choice made based on the number of samples $N = n \times B$, which is selected independently of image size. The number of factors of the best models, together with the metric values obtained, are shown in Table 2. It is worth mentioning that Bartlett's p-value is not included in any table as it is always less than 0.05. We observe that, except for the first layer where metrics D_1 and $D_>$ have too low values, our approach is able to provide suitable factor models, according to the requirements of Table 1. Besides, the number of latent variables of the best fitted models is at maximum half the number of channels, providing a resume in essential features. Note that this number is computed either with the SES criterion or as the 25% or 50% of the channels.

Meanwhile, if VGG-16 on imagenette is considered, all batches equal $B = \lceil 3 \times m/n \rceil$ (Table A.7). The summary of the parameters of the best estimated models are described in Table 3. As before, we obtain well fitted models in all layers. The SES criterion is the predominant one for calculating the number of factors in this case. In general, the number of latent features is a bit greater than previously, slightly surpassing half of the channels in some cases. Despite this, it is a great reduction which represents a summary of the characteristics encoded.

When dealing with residual blocks in ResNet-50 and DeepLabv3+, only their first two convolutional layers are considered. The reason behind this selection is elaborated in Appendix A. It is based on persistently low KMO values and insufficient quality metrics. Therefore, the layer factorial model is not an adequate approach in those cases. For the rest of the layers, a single forward pass for batch of size B provides the convolutional outputs used as sample data to estimate the models. We choose $B = \lceil 10 \times m/n \rceil$ for all layers in DeepLabv3+ and layers conv1, conv 2b11, conv2b22, conv2b32, conv3b12, and conv 3b22 in ResNet-50 (Table A.7). Otherwise, $B = \lceil 3 \times m/n \rceil$. Tables 4 and 5 summarize the parameters obtained for the best estimated models. Except for some layers, where KMO, D_1, or $D_>$ lies slightly under the required threshold established in Table 1, our proposal yields to valid factor analysis models. Moreover, these models are capable of reducing

Table 4
Factor analysis model: factors and metrics for ResNet-50 on imagenette.

Layer	Channels	KMO	Factors	D_1	$D_>$	Res	γ_j^2 20th perc.
conv1	64	0.60	32 (50%)	0.85	1.62	0.0028	0.974
conv2b11	64	0.28	20 (SES)	0.66	0.89	0.0035	0.981
conv2b12	64	0.66	26 (SES)	0.89	0.98	0.0035	0.946
conv2b21	64	0.76	24 (SES)	0.83	1.48	0.0047	0.974
conv2b22	64	0.62	32 (50%)	0.78	1.25	0.0150	0.879
conv2b31	64	0.69	28 (SES)	0.77	1.20	0.0063	0.950
conv2b32	64	0.50	37 (SES)	0.90	2.09	0.0133	0.861
conv3b11	128	0.74	32 (25%)	0.69	0.94	0.0175	0.928
conv3b12	128	0.73	74 (SES)	0.77	1.20	0.0045	0.927
conv3b21	128	0.70	65 (SES)	0.73	1.06	0.0033	0.962
conv3b22	128	0.64	78 (SES)	0.81	1.37	0.0063	0.898
conv3b31	128	0.78	65 (SES)	0.72	1.03	0.0034	0.948
conv3b32	128	0.71	67 (SES)	0.76	1.16	0.0039	0.947
conv3b41	128	0.68	64 (50%)	0.72	1.04	0.0108	0.882
conv3b42	128	0.65	80 (SES)	0.79	1.27	0.0064	0.898
conv4b11	256	0.73	128 (SES)	0.64	0.82	0.0013	0.985
conv4b12	256	0.73	149 (SES)	0.70	0.98	0.0021	0.967
conv4b21	256	0.77	128 (50%)	0.67	0.89	0.0029	0.965
conv4b22	256	0.72	158 (SES)	0.71	1.01	0.0026	0.958
conv4b31	256	0.76	128 (50%)	0.69	0.95	0.0067	0.923
conv4b32	256	0.80	128 (50%)	0.75	1.13	0.0123	0.829
conv4b41	256	0.79	128 (50%)	0.70	0.98	0.0058	0.916
conv4b42	256	0.77	161 (SES)	0.70	0.97	0.0023	0.953
conv4b51	256	0.78	128 (50%)	0.71	1.02	0.0090	0.883
conv4b52	256	0.81	161 (SES)	0.70	0.99	0.0024	0.948
conv4b61	256	0.87	128 (50%)	0.72	1.04	0.0069	0.884
conv4b62	256	0.84	149 (SES)	0.69	0.95	0.0018	0.965
conv5b11	512	0.80	256 (50%)	0.65	0.86	0.0034	0.955
conv5b12	512	0.88	345 (SES)	0.70	0.97	0.0013	0.960
conv5b21	512	0.89	256 (50%)	0.69	0.95	0.0035	0.925
conv5b22	512	0.93	347 (SES)	0.70	0.99	0.0009	0.963
conv5b31	512	0.94	256 (50%)	0.72	1.03	0.0053	0.865
conv5b32	512	0.95	256 (50%)	0.71	1.01	0.0042	0.891

Table 5
Factor analysis model: factors and metrics for DeepLabv3+ on CamVid.

Layer	Channels	KMO	Factors	D_1	$D_>$	Res	γ_j^2 20th perc.
conv1	64	0.68	22 (SES)	0.84	1.52	0.0090	0.964
res2a2a	64	0.83	31 (SES)	0.91	2.17	0.0063	0.932
res2a2b	64	0.78	32 (SES)	0.85	1.61	0.0077	0.861
res2b2a	64	0.70	36 (SES)	0.88	1.83	0.0083	0.851
res2b2b	64	0.59	48 (75%)	0.83	1.47	0.0034	0.854
res3a2a	128	0.59	80 (SES)	0.81	1.40	0.0052	0.936
res3a2b	128	0.74	80 (SES)	0.83	1.48	0.0089	0.807
res3b2a	128	0.70	81 (SES)	0.80	1.35	0.0084	0.826
res3b2b	128	0.79	96 (75%)	0.81	1.37	0.0048	0.822
res4a2a	256	0.79	160 (SES)	0.70	0.98	0.0022	0.953
res4a2b	256	0.83	170 (SES)	0.77	1.19	0.0033	0.894
res4b2a	256	0.87	164 (SES)	0.76	1.17	0.0029	0.912
res4b2b	256	0.91	168 (SES)	0.76	1.18	0.0028	0.901
res5a2a	512	0.90	343 (SES)	0.69	0.96	0.0010	0.969
res5a2b	512	0.93	354 (SES)	0.74	1.10	0.0016	0.928
res5b2a	512	0.95	346 (SES)	0.72	1.04	0.0011	0.949
res5b2b	512	0.82	332 (SES)	0.61	0.77	0.0007	0.985
aspp1	256	0.81	128 (75%)	0.68	0.94	0.0042	0.938
aspp2	256	0.88	155 (SES)	0.76	1.18	0.0033	0.912
aspp3	256	0.91	149 (SES)	0.77	1.20	0.0031	0.906
aspp4	256	0.95	146 (SES)	0.77	1.20	0.0022	0.916
dec1	256	0.93	151 (SES)	0.75	1.13	0.0024	0.920
decUp	256	0.92	156 (SES)	0.77	1.19	0.0025	0.912
dec3	256	0.95	142 (SES)	0.75	1.12	0.0016	0.941
dec4	256	0.97	18 (K)	0.64	0.83	0.0156	0.817

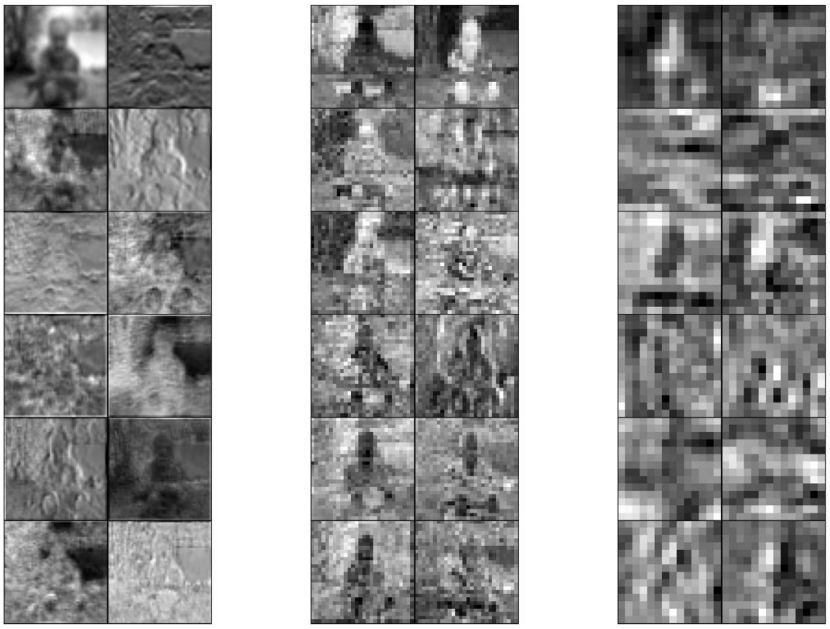
the original channels by (almost) a half, capturing the essential features through the latent factors (mostly obtained with the SES method).

5.2.2. Regarding factorial parameters and explainability

Going back to our interpretation of the estimated factors as unobserved activation maps, we visualize the latter in Fig. 3 for ResNet-50. Note that the intrinsic features of the first layers correspond to edge, color or simple pattern detectors, being consistent with what has been

(a) CIFAR-10 (b) Imagenette (c) CamVid

Fig. 2. Images of each dataset used to show the explanatory results across the article.



(a) Layer *conv2b12* (b) Layer *conv3b31* (c) Layer *conv4b22*

Fig. 3. Visualization of some latent features of ResNet-50 on imagenette.

(a) ELFA-CNNs (b) Random selection (c) Strongest activations (d) Principal components

Fig. 4. Visualizing the "first" 6 activation maps for different criteria (a–d) of the second convolutional layer of VGG-16 on CIFAR-10.

already presented in previous works. They become more abstract as we go deep into the network. In order to better visualize the characteristics encoded in those maps, we refer to Section 6.3, where their inversion to the input space is performed.

Fig. 4 compares the first six features given by ELFA (a) with others in the literature: (b) random selection, (c) strongest activations, and (d) principal components. The ones that present more variety, and better summarize the learning of the layer, are the two based on statistics: factor analysis (a) and PCA (d). The choice of FA over PCA has already been explained. It is based on the ability of FA to separately account for common and specific variance, and its characterization as a modeling rather than a compression method.

The correlation between the essential features and the original ones is provided by the loading matrix Λ, whose transpose for *conv*2 of VGG-16 on CIFAR-10 is represented in Fig. 5 (i.e. rows correspond to factors and columns to channels). Despite Section 6.1 describes the explanatory

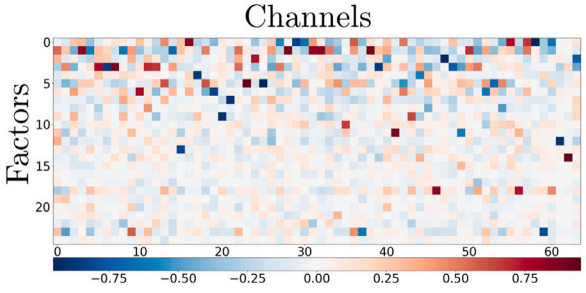
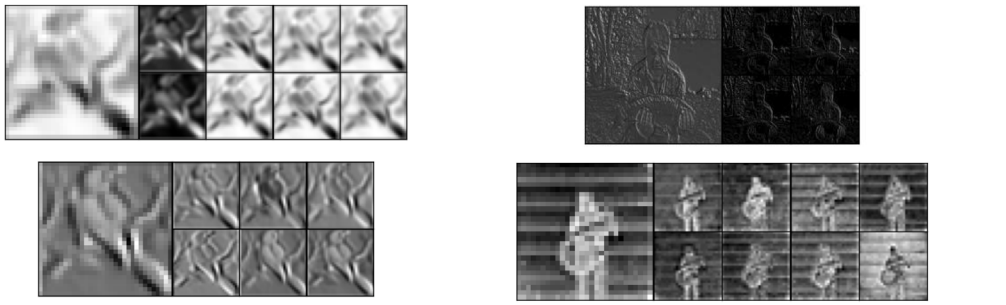
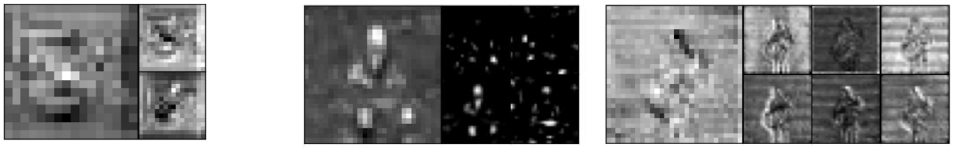


Fig. 5. Transpose loading matrix $\hat{\Lambda}^{\top}$ of the factor analysis model estimated for the second convolutional layer of VGG-16 on CIFAR-10.



(a) VGG-16 on CIFAR-10 (b) VGG-16 (top) and ResNet-50 (bottom) on imagenette

Fig. 6. Each image shows redundant channels (right) that share a latent feature (left).



(a) VGG-16 on CIFAR-10 (cat) (b) VGG-16 (left) and ResNet-50 (right) on imagenette

Fig. 7. Each image shows apparently unrelated filters (right) that share a common latent feature (left).

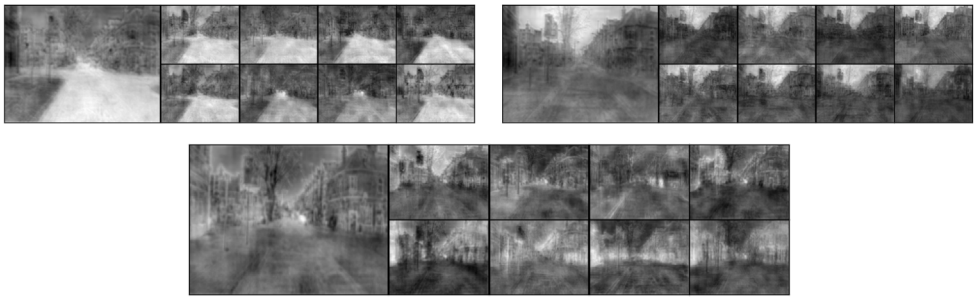
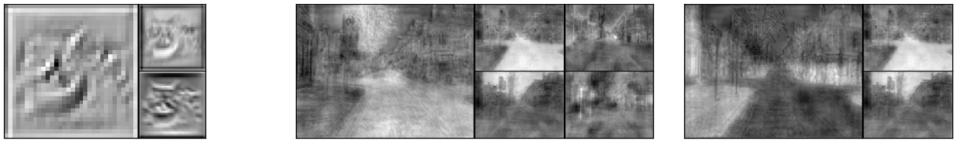


Fig. 8. Relationship between the channels of the last convolutional layer of DeepLabv3+ trained on CamVid for semantic segmentation. Each image shows the latent factor on the left and the channels associated on the right.



(a) CIFAR-10 (b) CamVid

Fig. 9. Each image shows the latent features (right) associated with an activation map (left). The sidewalk (last image) is understood as the opposite of the sky and the road.

Table 6
Attribution quality metrics results. The best values are bold-faced.

	Gradient	LRP	Grad-CAM	Grad-CAM++	Score-CAM	LayerCAM	EFAM-R
Faithfulness	0.0367	0.0876	0.0580	0.0435	0.1666	0.0577	**0.0167**
Sparseness	0.4913	0.6101	**0.7617**	0.4709	0.5850	0.5230	**0.7610**

Fig. 10. Comparison of EFAM (second column), Grad-CAM (third) and LRP (fourth) for VGG-16 on imagenette on the third (top) and last (bottom) convolutional layers.

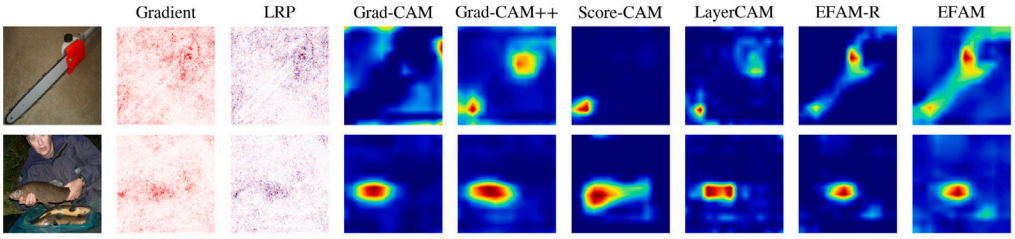


Fig. 11. Visualization results of Gradient (the gradient of the output neuron with respect to the input), LRP [15], Grad-CAM [8], Grad-CAM++ [22], Score-CAM [9], LayerCAM [17], and our proposed EFAM. The last convolution of VGG-16 on imagenette is analyzed. More results are provided in Appendix B.

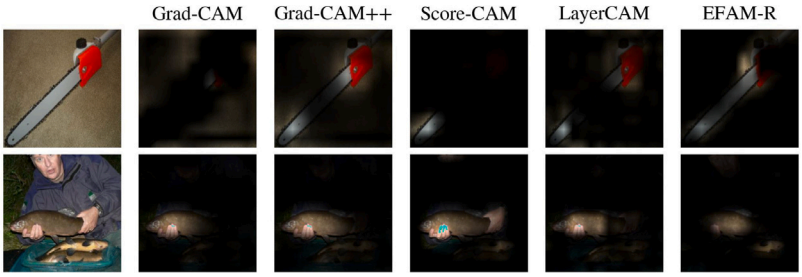


Fig. 12. Masking results associated to Fig. 11. Transparencies correspond to high heat map values. More results are provided in Appendix B.

power of Λ, a quick visual analysis can give us some insights, as we shall now briefly discuss.

It was previously explained (Section 4.1) that each row indicates the influence of a factor on the channels of the layer (more intense color, higher correlation between them). Meanwhile, columns show which of the essential features are hidden behind the corresponding channel (the stronger the color, the more present that feature is). Thus, in the case of Fig. 5, the last intrinsic features (from the tenth row onwards) can virtually be identified with one of the filters, whereas the first ones influence several of them. Channels with the same unique factor having an impact on them can be considered redundant (e.g. the 6–8th columns all correspond to the 4th essential feature). Moreover, we could measure the importance of a channel according to the contributions of the latent features. Hence, useless filters match columns that are uncorrelated with almost any factor (e.g. the second one).

6. Discussion

This section addresses in detail the explainable ability of ELFA-CNNs, which is pointed out by box E in Fig. 1. In this regard, the loading matrix is analyzed, from which EFAM (Section 4.2.2) and intrinsic features inversion are calculated (Section 4.2.3).

6.1. Redundancy and channel relationship

Consider a convolutional layer, with m filters, and the estimated layer factorial model, with p factors. It is known that $\hat{\Lambda} \in \mathcal{M}_{m,p}$ stores information about the relationship between the different filters and their correlation with the latent features. This allows us to quickly obtain much more complete and explanatory visualizations than other methods, as shown below.

As explained in Section 4, the columns, $\lambda_k = (\lambda_{1k}, \dots, \lambda_{mk})^\top$ with $1 \leq k \leq p$, link a certain factor or latent feature with the original channels or characteristics. Each value λ_{jk} indicates the influence of the kth factor on the jth channel. Therefore, by plotting the activation maps associated with the most influenced channels (i.e., for each k we plot the channels with the largest λ_{jk}), we are able to discover connections between different filters, as well as the presence of redundant ones.

Examples of this are shown in Fig. 6 for the different CNNs studied. On the right, a given factor is plotted. On the left, the activation

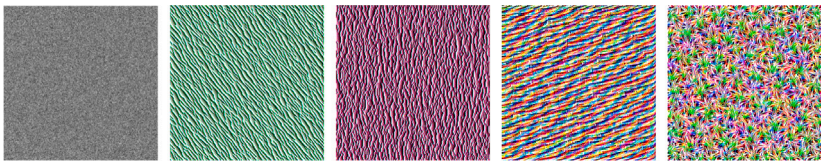
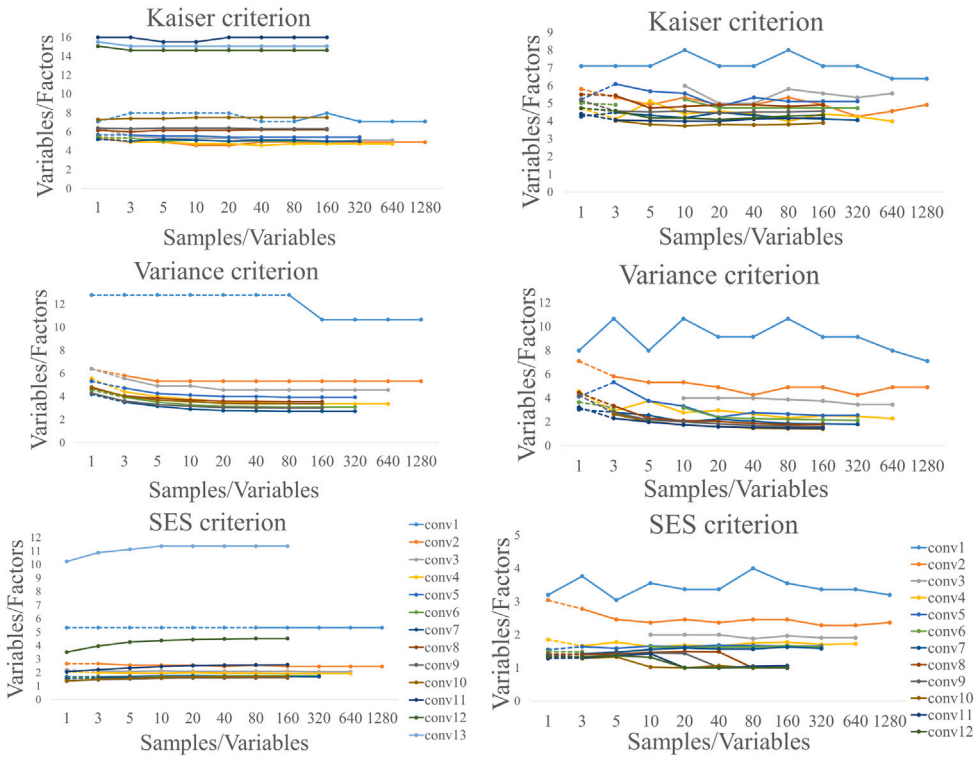


Fig. 13. Reconstructed images from generated random noise (leftmost) for some latent factors of the first convolutional layers of VGG-16 pre-trained on ImageNet. Simple edge detectors are found.

Fig. 14. Reconstructed images for some latent factors of different convolutional layers of VGG-16 pre-trained on ImageNet. As we move from left to right, they correspond to deeper layers of the network. This is reflected in the abstraction of the features. Image by Pete Linforth via Pixabay.



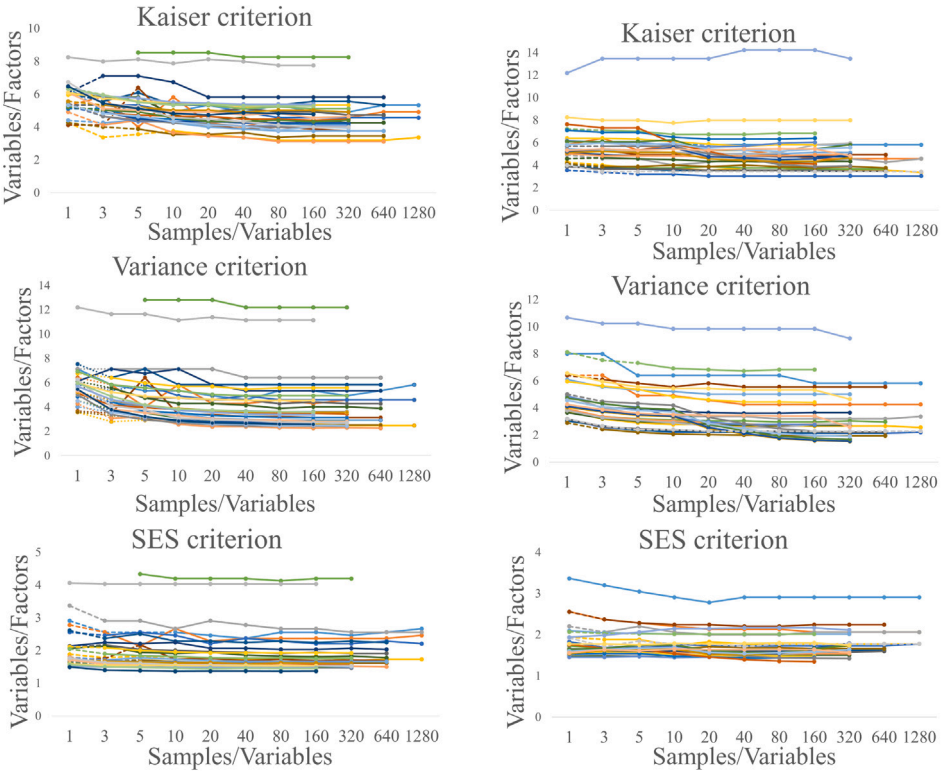
(a) VGG-16 on CIFAR-10 (b) VGG-16 on imagenette

Fig. A.15. Ratio variables to factors versus samples to variables for each convolutional layer of VGG-16. Dotted lines indicate low KMO value.

maps associated with the channels where the influence (positive or negative) of this factor is high. We observe that redundancy is present in different layers across the networks. When the features represented by the activation maps of the channels are identical to that of the latent factor, it is enough to keep one of them, since all of the filters are detecting the same essential feature.

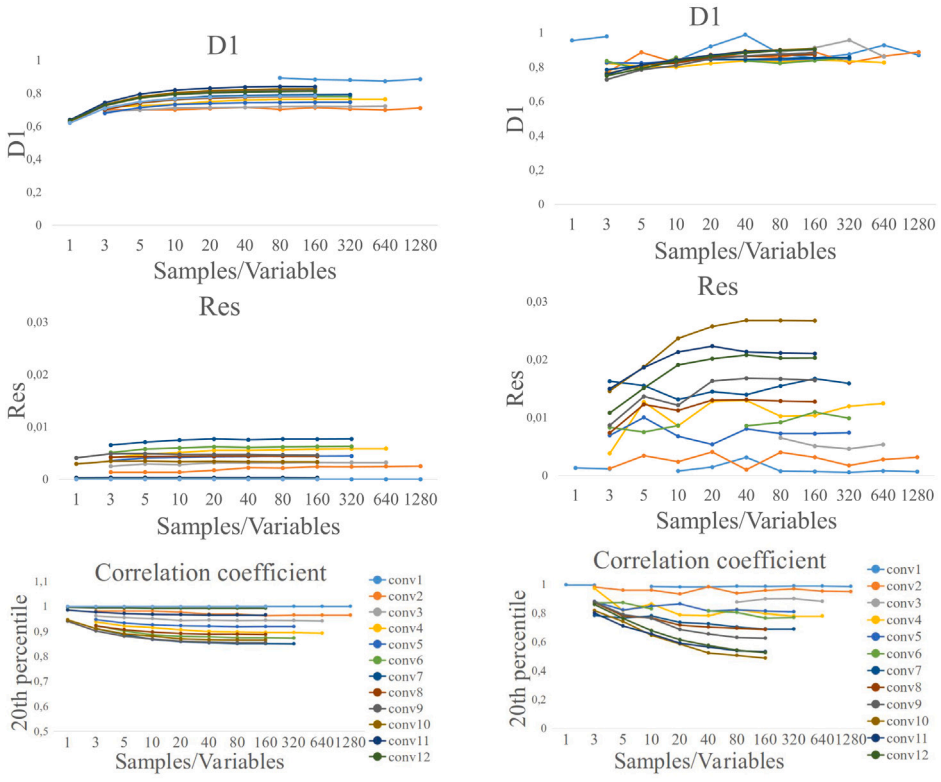
Even though we consider the channels where a given intrinsic feature has high impact, they need not to be identical to it. This would mean that the channels are greatly influenced by other latent factors too. As a result, affinity between channels that are apparently unrelated can be discovered. Examples are visualized in Fig. 7, which has the same structure, i.e., the latent factor on the left and the associated channels on the right. Besides, they allow us to deduce how the network is coding information: the essential feature encoded in the latent factor (left) is being understood and stored by the network as a combination of these channel features (right).

A more interesting analysis is that of the last layer of DeepLabv3+, shown in Fig. 8. This network is trained on CamVid for semantic segmentation. Therefore, the activation maps of this layer are very close to the final segmentation provided by the network. Note that the



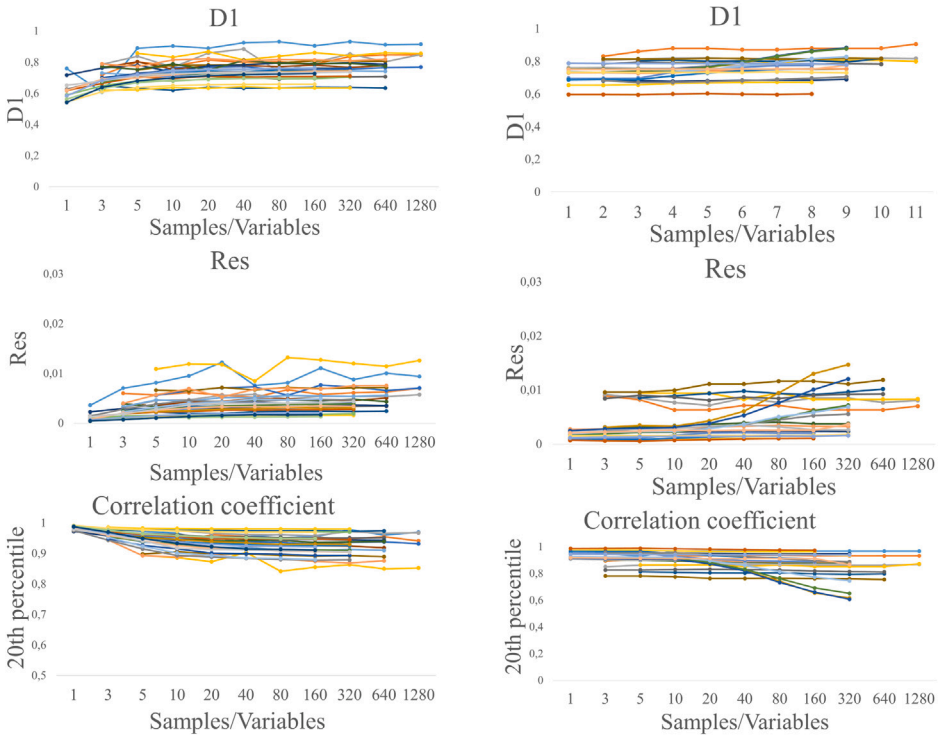
(a) ResNet-50 on imagenette (b) DeepLabv3+ on CamVid

Fig. A.16. Ratio variables to factors versus samples to variables for each convolutional layer. Dotted lines indicate low KMO value.



(a) VGG-16 on CIFAR-10 (b) VGG-16 on imagenette

Fig. A.17. Validation metrics values for each convolutional layer of VGG-16. Factors are computed with SES. Only values obtained with adequate data are plotted.



(a) ResNet-50 on imagenette (b) DeepLabv3+ on CamVid

Fig. A.18. Validation metrics values for each convolutional layer of VGG-16. Factors are computed with SES. Only values obtained with adequate data are plotted.

number of latent factors is almost equal to the number of classes to be segmented (18 vs. 11 in Table 5). Hence, the corresponding unobserved activation maps should almost coincide with this segmentation, as Fig. 8 exhibits. The channels associated with each latent feature either show the segmentation in one of the classes (e.g. the first two images with the road and sky classes) or exhibit activations in different areas whose combination gives rise to the final segmentation. This shows how the network has learned and related the different regions of the image to recognize the desired classes. For example, in the last image of Fig. 8, the detection of the sky, the buildings and others (channels on the right) are combined to get car class segmentation (latent factor on the left).

With respect to the rows of the loading matrix, $\lambda_j = (\lambda_{j1}, \ldots, \lambda_{jp})$ with $1 \leq j \leq m$, they connect a certain channel or filter (j) with the latent essential features. Analogously to the previous case, the impact of those features on the channel is given by the values λ_{jk}, and the most influential can be visualized via the unobserved activation maps associated with the greater lambda values. This is done in Fig. 9 (on the left, the channel, on the right, the essential features that most influence it). The ideal case would be that in which there is only one of these lambda values, since each channel would be identified with a single latent feature. Despite that in general this does not occur, interesting information can be obtained about what characteristics a channel or filter can be summarized or decomposed into. For example, (a) in Fig. 9 shows how an activation map of the first convolutional layer of VGG-16 can be decomposed into a color and an edge detector. The channels of (b), which belong to the last convolution of DeepLabv3+, are decomposed into different segmentation classes, given by the latent features. Hence, we gain insights into how final decisions are made.

6.2. EFAM

In this section we obtain essential feature attribution maps for VGG-16 on imagenette, and compare them with other state-of-the-art approaches. Fig. 10 contrasts the third and last convolutional layers. Together with EFAM, we examine Grad-CAM [8] and LRP.[3] [15]. Our proposal shows to a greater extent the evolution from low-level (edges) to high-level features. Focusing on the last convolution, Figs. 11 and 12 present a qualitative comparison between different explanatory methods.[4] More examples are provided in Appendix B. As opposed to other approaches, we observe that EFAM emphasizes the essential parts of the classified objects. The whole chainsaw is pointed out as well as both fishes, while other methods can only identify one of them and even focus their attention on the hands.

The images shown in Fig. 11, 12, and Appendix B are representative of the set of images analyzed. These characteristics and general behavior have been observed, so that our proposal better highlights the relevant parts of the images that are essential in network's learning.

It is worth mentioning that the other explanatory methods evaluated are designed to emphasize regions that contribute to the final decision and to what extent they do so. However, our proposal draws attention to areas where the main characteristics detected by the given convolutional layer are. Note that these features may not agree with the ones detected by the filters given that, unlike similar methods, we define the heat map with a selection or summary of essential features. Although it is to be expected that these approaches provide similar heat maps, since it is presumed that the areas that most influence the final decision are those where the decisive features are located, they do not have to, depending on how the network has learned.

A complementary quantitative analysis is performed.[5] We evaluate the faithfulness of the computed explanations through pixel flipping [49]. This strategy consists in progressively replacing the input image pixels associated with higher heat map values by black pixels. Thus, the worse the prediction of the perturbed image, the more

[3] Grad-CAM is obtained from scratch while iNNvestigate package [47] is used for LRP.

[4] Gradient and LRP use iNNvestigate, while the others employ tf-keras-vis.

[5] Metrics are obtained with quantus [48].

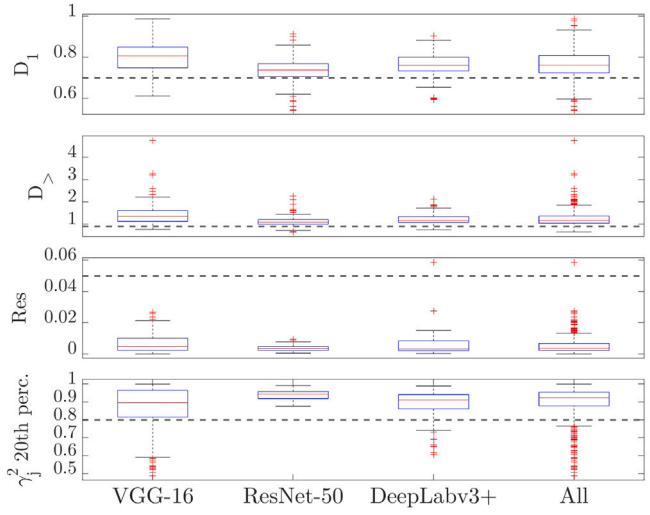


Fig. A.19. Box plots of the validation metrics for each type of CNN. The last column (All) considers all values for all networks. A dotted line indicates the chosen thresholds.

important are the detected features for decision making. The predictive results for a batch of size 100 are plotted against the percentage of flipped pixels, and the mean area under the curve is obtained. Note that we would like to keep this area as small as possible. Moreover, we evaluate the complexity of the method, i.e., whether the explanations provided are concise. In particular, we measure sparseness via the Gini Index [50], being the larger the better. Among all the possible metrics, these two are chosen due to the assumed properties of EFAM: to detect the underlying essential characteristics. Being the essential ones should imply conciseness in explanations, while disturbing the underlying features should cause a quick degradation of performance.

Results are found in Table 6. EFAM sparseness is similar to Grad-CAM while outperforming the rest, which demonstrates its validity compared to a well-studied method and its improvement compared to others. We infer that the features pointed out by ELFA-CNNs provide a concise and adequate summary of the layer's knowledge.

6.3. Inverting intrinsic features

Given a convolutional layer, there exist explainable methods which invert an activation map in order to obtain an input image that provides the same activation [11,13] or that maximizes it [6,7]. This procedure helps to better visualize the features encoded by the convolutional filters, which become more abstract as we go deep into the network, and to understand the learning process. However, convolutional layers have many channels, and therefore activation maps. This, together with the redundancy present in the layers (and exhibited in previous sections), makes the choice of the channel to study difficult and inaccurate. We solve these issues by considering the activation maps estimated from the latent factors of the layer factorial model, since they represent a summary of the most important filters. Following the procedure presented in Section 4.2.3, it is possible to invert the essential features. Input images resulting from applying this inversion proposal on VGG-16 pre-trained on ImageNet are shown in Figs. 13 and 14. They are reconstructed starting with either random noise or a given image, respectively. Results similar to those of previous methods are obtained, proving the consistency of our approach.

7. Conclusion

This paper proposes a novel explanatory method for CNNs, named ELFA-CNNs, which relies in statistics and determines the essential features behind a convolutional layer, as well as their correlation with channels. The strategy is based on factor analysis to estimate layer models, that are validated thanks to the new metrics defined (D_1, $D_>$ and Res). As a result, ELFA provides an accurate and well-founded summary of the features learned, in addition to knowledge about the relationships between channels, useless filters, and the redundancy of the layer, among others. Therefore, the usual choice of specific neurons and their association with individual features presented in other methods is avoided, improving and simplifying explanations.

An analysis of different CNNs following ELFA is performed, gaining insights into the learning process. Further visual explanations are acquired through the proposal of essential feature attribution maps (EFAM) and intrinsic features inversion. Reliable and accurate heat maps are obtained compared to other methods.

In the future, we would like to conduct more investigations on the layer factorial models, studying the number of features retained and comparing between networks, as well as delving into their use in other aspects of learning. Besides, we would like to improve visualizations.

CRediT authorship contribution statement

Clara I. López-González: Conceptualization, Methodology, Software, Investigation, Writing – original draft. **María J. Gómez-Silva:** Validation, Investigation, Writing – review & editing. **Eva Besada-Portas:** Supervision, Funding acquisition, Writing – review & editing. **Gonzalo Pajares:** Supervision, Project administration, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data used in this article is publicly available.

Acknowledgments

The authors acknowledge support from the Research Project IA-GES-BLOOM-CM (Y2020/TCS-6420) of the Synergic program of the Comunidad Autónoma de Madrid, the Research Project SMART-BLOOMS (TED2021-130123B-I00) funded by the Spanish Ministry of Science and Innovation and the European Union NextGeneration, and from the Research Project INSERTION (PID20211-27648OB-C33) of the Knowledge Generation Programs of the Spanish Ministry of Science and Innovation. The first author, Clara I. López-González, is supported by a FPU Ph.D. scholarship from the Spanish Ministry of Universities. The authors thank the anonymous referees for their very valuable comments and suggestions.

Appendix A. Analysis of the number of samples

Given a CNN and a batch of B images, consider the output of a convolutional layer, $x \in \mathbb{R}^{B\times n\times m}$ where m is the number of channels or filters and n the number of pixels. A pending question in the layer factor analysis setting is how many samples $N = n \times B$ are sufficient to obtain a great factorial model. In general, recommendations in the literature are guiding rules of thumb for minimum sample size [51] or ratio of sample size to number of variables, based on communalities, number of factors, or variables per factor [52]. Although larger samples seem better, there exist varied and even contradictory suggestions.

Consequently, in order to determine an appropriate sample size N for our approach, we analyze its relationship with the number of factors computed, the data adequacy, and the quality of the factorial model estimated. The studies are conducted with 11 different sample sizes,

Table A.7
Batch size B chosen for each convolutional layer on each CNN. The number of channels equals m, while n refers to the number of pixels of the activation maps. Sam/var stands for samples/variables ratio.

CNN	Dataset	Layers	Sam/var	B
VGG-16	CIFAR-10	conv1	80	$80 \times m/n$
		The rest	3	$3 \times m/n$
VGG-16	Imagenette	All	3	$3 \times m/n$
ResNet-50	Imagenette	conv1-2b11-2b22-2b32-3b12-3b22	3	$3 \times m/n$
		The rest	10	$10 \times m/n$
DeepLabv3+	CamVid	All	10	$10 \times m/n$

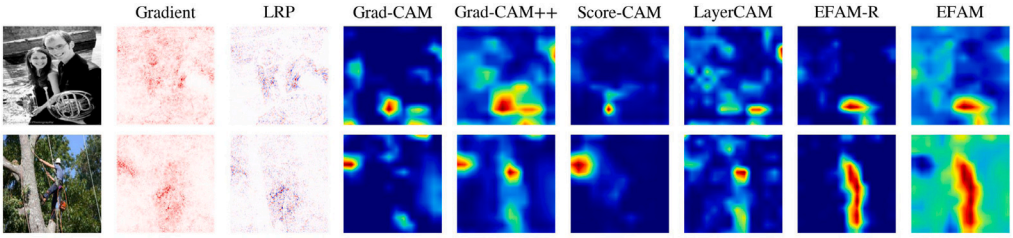


Fig. B.20. Visualization results of Gradients, LRP [15], Grad-CAM [8], Grad-CAM++ [22], Score-CAM [9], LayerCAM [17], and our proposed EFAM on the last convolution.

chosen independent of batch and image size. Namely, $N_s = 80 \cdot 2^s$ with $s = 0, \dots, 10$; taking into account that the number of samples cannot be less than that of variables (in our case m, the channels of the convolutional layer). Note that from the appropriate N_s we get the required batch size as $B = \lceil N_s/n \rceil$.

A.1. Influence on the number of factors

Regarding the number of factors, we examine how it varies with the size of the sample. For this purpose, we study the change in the ratio of number of variables to number of factors (variables/factors) with respect to the ratio of number of samples to number of variables (samples/variables). Firstly, we select one of the methods for the calculation of the number of factors (Section 3.2.2). Then, this number is computed for each data sample, i.e., $\{x^i\}_{i=1}^{N_s}$ output of the convolution for $s = 0, \dots, 10$. By doing so, the ratios variables/factors are obtained, and plotted against the ratios samples/variables (N_s/m). We repeat this procedure for all the convolutional layers and each of the methods mentioned in Section 3.2.2.

Figs. A.15 and A.16 show the outcome for the different convolutional neural networks considered. Dotted lines indicate data samples for which the KMO value was not high enough, according to Table 1 (Bartlett's p-value was always lower than 0.05). We observe that the ratio of variables to factors is almost stable with respect to the ratio of samples to variables. Consequently, the calculation of the number of factors does not depend on the number of samples (even though the methods employed use the data as input), and will result in almost the same number for all samples sizes. Nevertheless, this does not mean that all sizes are valid. At least, it should yield to data adequate for EFA, in terms of KMO and Bartlett tests.

A.2. Influence on data adequacy

The previous analyses help us to investigate the relationship between the sample size and the adequacy of data, since for each convolutional layer in the figures, dotted lines are used to indicate unsuitable data according to KMO.

On the one hand, VGG-16 presents adequate data for all but the first ratio, samples/variables = 1 (Fig. A.15), so any sample size that satisfies samples/variables ≥ 3 can be considered. In the case of CIFAR-10, the first convolutional layer is more sensible ((a), Fig. A.15), and until samples/variables = 80 unsuitable data is obtained. On the other hand, when dealing with complex networks (Fig. A.16), some of the convolutional layers inside the residual blocks exhibit persistent low KMO values, letting us know that for those the layer factorial model may not be an adequate approach. In the case of DeepLabv3+ on CamVid ((b), Fig. A.16), these exceptions correspond to the convolutional layers of the residual connections. In ResNet-50 ((a), Fig. A.16) the problematic layers correspond to the last convolution of the residual blocks and the one on the residual connections. Besides, the quality metrics obtained for these layers on different factorial models are not sufficient. Therefore, we omit them in what follows, not being appropriate a factorial approach. For the rest of the layers, a ratio samples/variables greater than 10 provides adequate data (in some cases even ≥ 3).

A.3. Influence on factorial model quality

Concerning the estimated layer factor model, we study whether for the same number of variables and factors, but varying number of samples, the quality metrics change. Consider a convolutional layer and a number of factors, in our case obtained with the SES criterion. For each sample size N_s ($s = 0, \dots, 10$) that yields adequate data $\{x^i\}_{i=1}^{N_s}$ (in terms of KMO and Bartlett's p-value), we estimate the factorial model and compute the validity tests of Section 3.2.3. The values are displayed against the ratio samples/variables in Figs. A.17 and A.18 for each of the studied networks. After arriving at certain ratio the metric values stabilize, except for some convolutional layers. We conclude that, once a proper sample size is reached, the quality of the model does not vary if we increase the sample. Indeed, in some exceptional cases, it could even worsen. Therefore, it is enough to consider the smallest sample size for which the previous behavior stabilizes. This outcome is combined with the above findings in the next section to come up with an appropriate sample and batch size.

It is worth noting that this study does not seek an optimal factorial model, but rather an analysis of the evolution of the metrics with respect to the number of samples. Even so, in general, the quality values settle above reasonable thresholds for all CNNs and datasets. Fig. A.19 exposes this fact via the box plots of the metrics, with dotted lines indicating the bounds. In particular, Res, that takes values between 0 and 1 being the smaller the better, is under 0.05 (except for some outliers). For D_1 and the 20th percentile of γ_j^2, which should be close to 1, the 25th percentile lies above 0.7 and 0.8 respectively. Lastly, $D_>$ 25th percentile, the higher the value the better, is over 0.9. This allows us to establish the quality thresholds of our approach, displayed in Table 1. Due to the variety in the networks and datasets considered,

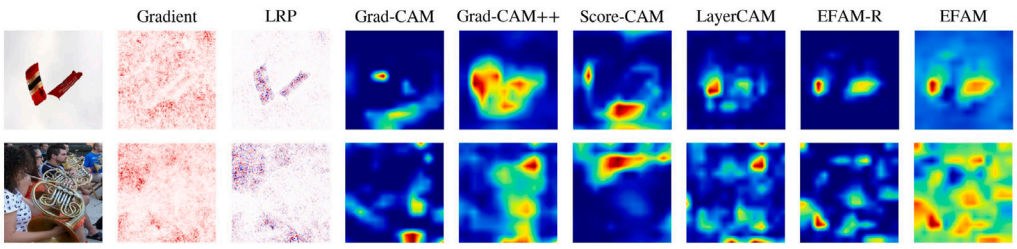


Fig. B.21. Multi-target visualization results on the last convolutional layer of VGG-16.

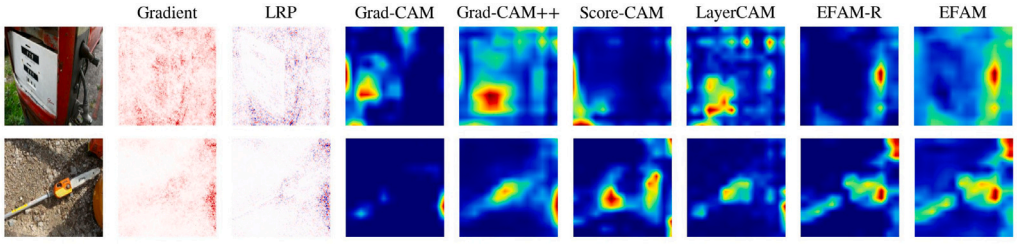


Fig. B.22. Visualization results on the last convolutional layer of VGG-16. Top: while LRP and EFAM focus on the hose, the other methods emphasize the gas pump structure. Bottom: EFAM provides a more accurate and localized heat map.

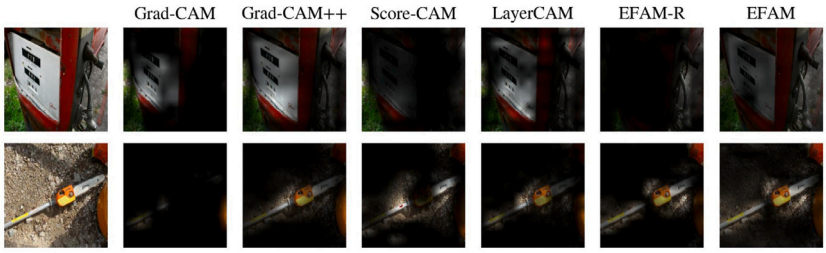


Fig. B.23. Masking results associated with Fig. B.22. Transparencies correspond to high heat map values.

it is possible to infer their generalization for other setups. Perhaps very different scenarios need small modifications, but always satisfying minimum quality requirements.

A.4. Concluding and determining the batch size

The above experiments let us conclude the following regarding layer factor analysis: (a) the number of factors does not depend on the number of samples; (b) when a layer factor analysis approach is appropriate, not too many samples are needed in order to obtain adequate data; (c) in general, the validity metrics of the estimated model stabilize after a sample size (not too big) is reached; and (d) too many samples may be counterproductive. Taking this into account, and combining the ratios samples/variables derived in Appendix A.2 together with the stabilization behavior of the estimated model metrics of Appendix A.3, we deduce the suitable batch size for each layer on each CNN. This is summarized in Table A.7.

Note that samples/variables $= N_s/m = n \times B/m$, where n is the number of pixels of the activation maps and m the number of channels of the layer. Hence, knowing the ratio of number of samples to number of variables, let us denote it by $M =$ samples/variables, the batch size B can be computed as $B = \lceil M \times m/n \rceil$.

Appendix B. Attribution visualization results

For a qualitative comparison between the heat maps of the explanatory methods considered in Section 6.2 on VGG-16, we provide results for some random images, extracted from imagenette, in Figs. B.20, B.21, B.22, and B.23. Note how EFAM better focuses on classified objects and their essential parts, being able to detect multiple targets as opposed to other methods (also shown in Fig. 11).

References

[1] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: International Conference on Learning Representations, 2015, pp. 1–14, arXiv:1409.1556.
[2] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778, http://dx.doi.org/10.1109/CVPR.2016.90.
[3] H. Law, J. Deng, CornerNet: Detecting objects as paired keypoints, Int. J. Comput. Vis. 128 (2020) 642–656, http://dx.doi.org/10.1007/s11263-019-01204-1.
[4] A. Lou, M.H. Loew, CFPNet: Channel-wise feature pyramid for real-time semantic segmentation, in: 2021 IEEE International Conference on Image Processing (ICIP), 2021, pp. 1894–1898, arXiv:2103.12212.
[5] D. Minh, H.X. Wang, Y.F. Li, T.N. Nguyen, Explainable artificial intelligence: a comprehensive review, Artif. Intell. Rev. 55 (5) (2021) 3503–3568, http://dx.doi.org/10.1007/s10462-021-10088-y.
[6] K. Simonyan, A. Vedaldi, A. Zisserman, Deep inside convolutional networks: Visualising image classification models and saliency maps, in: Workshop At International Conference on Learning Representations, 2014, arXiv:1312.6034.
[7] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, H. Lipson, Understanding neural networks through deep visualization, 2015, arXiv:1506.06579.
[8] R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-CAM: Visual explanations from deep networks via gradient-based localization, Int. J. Comput. Vis. 128 (2) (2019) 336–359, http://dx.doi.org/10.1007/s11263-019-01228-7.
[9] H. Wang, Z. Wang, M. Du, F. Yang, Z. Zhang, S. Ding, P. Mardziel, X. Hu, Score-CAM: Score-weighted visual explanations for convolutional neural networks, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE, 2020, http://dx.doi.org/10.1109/cvprw50498.2020.00020.

[10] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, A. Torralba, Object detectors emerge in deep scene CNNs, 2015, arXiv:1412.6856.

[11] M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: Computer Vision – ECCV 2014, Springer International Publishing, Cham, 2014, pp. 818–833, http://dx.doi.org/10.1007/978-3-319-10590-1_53.

[12] L. Weber, S. Lapuschkin, A. Binder, W. Samek, Beyond explaining: Opportunities and challenges of XAI-based model improvement, Inf. Fusion 92 (2023) 154–176, http://dx.doi.org/10.1016/j.inffus.2022.11.013.

[13] A. Mahendran, A. Vedaldi, Understanding deep image representations by inverting them, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 5188–5196, http://dx.doi.org/10.1109/CVPR.2015.7299155.

[14] K. Abhishek, D. Kamath, Attribution-based XAI methods in computer vision: A review, 2022, arXiv:2211.14736v1.

[15] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, W. Samek, On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation, PLoS One 10 (7) (2015) http://dx.doi.org/10.1371/journal.pone.0130140.

[16] L. Hoyer, M. Munoz, P. Katiyar, A. Khoreva, V. Fischer, Grid saliency for context explanations of semantic segmentation, in: Advances in Neural Information Processing Systems, vol. 32, Curran Associates, Inc., 2019.

[17] P.-T. Jiang, C.-B. Zhang, Q. Hou, M.-M. Cheng, Y. Wei, LayerCAM: Exploring hierarchical class activation maps for localization, IEEE Trans. Image Process. 30 (2021) 5875–5888, http://dx.doi.org/10.1109/tip.2021.3089943.

[18] A. Nguyen, J. Yosinski, J. Clune, Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks, in: Visualization for Deep Learning Workshop At International Conference on Machine Learning, 2016, arXiv:1602.03616.

[19] C. Spearman, "General intelligence," objectively determined and measured, Am. J. Psychol. 15 (2) (1904) 201–292, http://dx.doi.org/10.2307/1412107.

[20] F. Hohman, M. Kahng, R. Pienta, D.H. Chau, Visual analytics in deep learning: An interrogative survey for the next frontiers, IEEE Trans. Vis. Comput. Graphics 25 (8) (2019) 2674–2693, http://dx.doi.org/10.1109/TVCG.2018.2843369.

[21] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, A. Torralba, Learning deep features for discriminative localization, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2921–2929, http://dx.doi.org/10.1109/CVPR.2016.319.

[22] A. Chattopadhay, A. Sarkar, P. Howlader, V.N. Balasubramanian, Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks, in: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, 2018, http://dx.doi.org/10.1109/wacv.2018.00097.

[23] S. Carter, Z. Armstrong, L. Schubert, I. Johnson, C. Olah, Activation atlas, Distill 4 (3) (2019) http://dx.doi.org/10.23915/distill.00015.

[24] F. Hohman, H. Park, C. Robinson, D.H. Polo Chau, Summit: Scaling deep learning interpretability by visualizing activation and attribution summarizations, IEEE Trans. Vis. Comput. Graphics 26 (1) (2020) 1096–1106, http://dx.doi.org/10.1109/TVCG.2019.2934659.

[25] M. Ribeiro, S. Singh, C. Guestrin, "why should I trust you?": Explaining the predictions of any classifier, in: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, Association for Computational Linguistics, San Diego, California, 2016, pp. 97–101, http://dx.doi.org/10.18653/v1/N16-3020.

[26] M.R. Zafar, N. Khan, Deterministic local interpretable model-agnostic explanations for stable explainability, Mach. Learn. Knowl. Extract. 3 (3) (2021) 525–541, http://dx.doi.org/10.3390/make3030027.

[27] M. Zolanvari, Z. Yang, K. Khan, R. Jain, N. Meskin, TRUST XAI: Model-agnostic explanations for AI with a case study on IoT security, IEEE Internet Things J. (2022) 1, http://dx.doi.org/10.1109/jiot.2021.3122019.

[28] Y. Wen, Z. Li, Y. Qiao, Latent factor guided convolutional neural networks for age-invariant face recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2016, pp. 4893–4901, http://dx.doi.org/10.1109/CVPR.2016.529.

[29] H. Shi, G. Cao, Y. Zhang, Z. Ge, Y. Liu, D. Yang, F^3 net: Fast Fourier filter network for hyperspectral image classification, IEEE Trans. Instrum. Meas. (2023) 1, http://dx.doi.org/10.1109/tim.2023.3277100.

[30] B. Chen, G. Polatkan, G. Sapiro, D. Dunson, L. Carin, The hierarchical beta process for convolutional factor analysis and deep learning, in: Proceedings of the 28th International Conference on Machine Learning, ICML, 2011, pp. 361–368, http://dx.doi.org/10.5555/3104482.3104528.

[31] A. Stevens, Y. Pu, Y. Sun, G. Spell, L. Carin, Tensor-dictionary learning with deep Kruskal-factor analysis, in: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, in: Proceedings of Machine Learning Research, vol. 54, PMLR, 2017, pp. 121–129, arXiv:1612.02842.

[32] H.H. Harman, Modern Factor Analysis, University of Chicago Press, 1976, URL https://books.google.es/books?id=e-vMN68C3M4C.

[33] D. Barber, Bayesian Reasoning and Machine Learning, Cambridge University Press, 2012.

[34] B. Thompson, Exploratory and Confirmatory Factor Analysis: Understanding Concepts and Applications., American Psychological Association, 2004, http://dx.doi.org/10.1037/10694-000.

[35] H. Taherdoost, S. Sahibuddin, N. Jalaliyoon, Exploratory factor analysis; concepts and theory, in: Advances in Applied and Pure Mathematics, in: Mathematics and Computers in Science and Engineering Series, vol. 27, WSEAS, 2014, pp. 375–382, URL https://hal.science/hal-02557344.

[36] H.F. Kaiser, A second generation little jiffy, Psychometrika 35 (4) (1970) 401–415, http://dx.doi.org/10.1007/bf02291817.

[37] M.S. Bartlett, Tests of significance in factor analysis, Brit. J. Stat. Psychol. 3 (2) (1950) 77–85, http://dx.doi.org/10.1111/j.2044-8317.1950.tb00285.x.

[38] H.F. Kaiser, The application of electronic computers to factor analysis, Educ. Psychol. Meas. 20 (1) (1960) 141–151, http://dx.doi.org/10.1177/001316446002000116.

[39] R.B. Cattell, The scree test for the number of factors, Multivar. Behav. Res. 1 (2) (1966) 245–276, http://dx.doi.org/10.1207/s15327906mbr0102_10.

[40] K.W. Zoski, S. Jurs, An objective counterpart to the visual scree test for factor analysis: The standard error scree, Educ. Psychol. Meas. 56 (3) (1996) 443–451, http://dx.doi.org/10.1177/0013164496056003006.

[41] A. Krizhevsky, Learning Multiple Layers of Features from Tiny Images, Toronto, ON, Canada, 2009.

[42] J. Howard, Imagenette, 2022, URL https://github.com/fastai/imagenette/.

[43] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, Encoder-decoder with atrous separable convolution for semantic image segmentation, in: Proceedings of the European Conference on Computer Vision (ECCV), Springer International Publishing, 2018, pp. 833–851, http://dx.doi.org/10.1007/978-3-030-01234-2_49.

[44] G.J. Brostow, J. Fauqueur, R. Cipolla, Semantic object classes in video: A high-definition ground truth database, Pattern Recognit. Lett. 30 (2) (2009) 88–97, http://dx.doi.org/10.1016/j.patrec.2008.04.005, Video-based Object and Event Analysis.

[45] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, vol. 25, Curran Associates, Inc., 2012, http://dx.doi.org/10.1145/3065386.

[46] Keras applications: Modules, 2023, URL https://www.tensorflow.org/api_docs/python/tf/keras/applications.

[47] M. Alber, S. Lapuschkin, P. Seegerer, M. Hägele, K.T. Schütt, G. Montavon, W. Samek, K.-R. Müller, S. Dähne, P.-J. Kindermans, INNvestigate neural networks!, J. Mach. Learn. Res. 20 (93) (2019) 1–8, URL http://jmlr.org/papers/v20/18-540.html.

[48] A. Hedström, L. Weber, D. Krakowczyk, D. Bareeva, F. Motzkus, W. Samek, S. Lapuschkin, M.M.M. Höhne, Quantus: An explainable AI toolkit for responsible evaluation of neural network explanations and beyond, J. Mach. Learn. Res. 24 (34) (2023) 1–11, URL http://jmlr.org/papers/v24/22-0142.html.

[49] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, W. Samek, On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation, in: O.D. Suarez (Ed.), PLoS One 10 (7) (2015) e0130140, http://dx.doi.org/10.1371/journal.pone.0130140.

[50] P. Chalasani, J. Chen, A.R. Chowdhury, X. Wu, S. Jha, Concise explanations of neural networks using adversarial training, in: Proceedings of the 37th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 119, PMLR, 2020, pp. 1383–1391, URL https://proceedings.mlr.press/v119/chalasani20a.html.

[51] A.L. Comrey, A First Courses in Factor Analysis, New York, Academic Press, 1973.

[52] R.K. Henson, J.K. Roberts, Use of exploratory factor analysis in published research: Common errors and some comment on improved practice, Educ. Psychol. Meas. 66 (3) (2006) 393–416, http://dx.doi.org/10.1177/0013164405282485.