PCACE: A Statistical Approach to Ranking Neurons for CNN Interpretability Applied to Air Pollution

Sílvia Casacuberta
Harvard University
scasacubertapuig@college.harvard.edu

Esra Suel
Imperial College London
esra.suel@imperial.ac.uk

Seth Flaxman
Imperial College London
s.flaxman@imperial.ac.uk

ABSTRACT

In this paper we introduce a new problem within the growing literature on interpretability for convolutional neural networks (CNNs), with a special focus on applications to street-level imagery. While previous work has focused on the question of how to visually interpret CNNs, we ask what it is that we care to interpret, that is, which layers and neurons are worth our attention? Due to the vast size of modern deep learning network architectures, automated, quantitative methods are needed to rank the relative importance of neurons so as to provide an answer to this question. We present a new statistical method for ranking the hidden neurons in any convolutional layer of a network. We define importance as the maximal correlation between the activation maps and the class score. We provide different ways in which this method can be used for visualization purposes with MNIST and ImageNet, and show a real-world application of our method to air pollution spatial variation prediction with street-level images. Our ranking algorithm constitutes a mathematical tool that can be used to interpret and visualize deep learning models trained on spatiotemporal data.

KEYWORDS

Convolutional neural networks, interpretability, spatiotemporal prediction

1 INTRODUCTION

In recent years, the use of convolutional neural networks (CNNs) has become widespread due to their success at performing tasks such as image classification or speech recognition. CNNs have achieved outstanding results at the ImageNet challenge and popularized the use of such architectures [25]. Remarkably, these same networks that outperform at the ImageNet challenge are also successful when used for other tasks such as object detection on the PASCAL VOC dataset [28]. Large training datasets, powerful GPUs and the

implementation of regularization techniques, such as dropout, have also helped boost the performance of CNN architectures [27].

While CNNs continue to excel at countless tasks and competitions, their internal procedures remain a mystery and there is very little insight into how these architectures achieve such outstanding results. We effectively treat these neural networks as black boxes, and it is extremely challenging to understand how they operate, due to their complexity and large number of interacting parts. However complex these black boxes might be, it is vital to acquire a deeper understanding of how they work. There are many reasons for encouraging research in this area. From a scientific perspective, understanding the inner mechanisms of models will allow us to improve their results and accuracy. Without any insight on how these black boxes work, their development into better models can only be achieved by trial-and-error. From a social perspective, we should not be allowing a system that applies untransparent and unexplainable algorithms to make decisions that govern health care, banking, or politics. Once the decisions taken by deep neural networks shift from innocuously classifying hand-written digits to deciding whether someone has a particular disease or is eligible for a bank loan, it becomes imperative to advocate for a right to explanation [11].

This paper focuses on the following question: how can we *rank* the hidden units of a convolutional layer in order of importance towards a final classification?[1] While research around understanding the inner mechanisms of CNN architecture advances, visualization methods that are targeted directly towards each neuron (e.g., feature visualization) are unfeasible to apply and analyze to each of the thousands of neurons in the architecture. Therefore, our goal is to provide a method that identifies the neurons that contribute the most to the final classification. On top of it, any other visualization or explainability method that applies to a particular neuron can be studied only on this small set of ranked neurons or combinations of them, hereby making it practical. We discuss this question thoroughly and propose a novel statistical method called *PCACE* that ranks the hidden units of a convolutional layer according to their relevance towards the final (class) score. The algorithm is based on the Alternating Conditional Expectation algorithm by Breiman and Friedman [4], which provides the optimal transformations that allow us to maximize the correlation between the activation maps of the neurons with the final (class) score. While in this paper we use PCACE to produce one ranking per convolutional layer, the algorithm can be applied to obtain a global ranking over the network, given that the PCACE values are normalized.

---

[1] We use the words *unit*, *filter*, *channel*, and *neuron* interchangeably, as there does not appear to be consensus in the literature [17].

We show how to use our statistical method for visualization purposes by using activation maps, CAM, and the activation maximization method. The combination of the statistical algorithm behind PCACE with several visualization methods yields a new procedure to help in the interpretability and explainability of convolutional neural networks. Besides testing our algorithm on the well-known datasets of MNIST and ImageNet [5], we provide a real-world use case to air pollution prediction of street-level images. In the case of ImageNet, we replicate our results with both the ResNet-18 and VGG-16 architectures and compare them. In the case of the street-level images for air pollution spatial variation estimation, we show how our ranking algorithm allows us to rigorously pick the activation maps of the most important neurons for explainability purposes. We further evidentiate how the network trained on spatiotemporal data acts as an object detector (in particular, of trees) despite being trained on a regression task, and how the PCACE algorithm is able to rank the neurons that most detect such trees as those of greatest importance. This sheds light on which features the model has learned towards the final classification, which is crucial towards ensuring an interpretable deployment of deep learning models on satellite images, among other types of data.

2 RELATED WORK ON INTERPRETABILITY AND EXPLAINABILITY OF NEURAL NETWORKS

After the rapid success of deep neural networks, the community has recently started to acknowledge that we have very limited understanding of how these architectures work and how they are able to achieve such remarkable results. Some of the first visualization tools that were proposed were *saliency maps* [22], which indicate the areas of the input image that are discriminative and most important with respect to the given class by using the intensity of the pixels. That is, given an input image, the saliency method ranks its pixels based on their influence on the final class score by using derivatives and backpropagation. A similar idea is the *Grad-CAM method* [21], which employs the gradients of any target concept to produce a localization map that highlights the important regions in the image that predict the concept. Grad-CAM can also be combined with existing pixel-space visualizations (Guided Grad-CAM) to achieve visualizations that are both high-resolution and class-discriminative.

Another way to study the importance of the pixels in the input image is to *perturb the image* by occluding patches of the image and see how the classification score drops [27]. Other authors follow a similar idea by directly deleting some parts of the image in order to find the part of the image that makes the final class score drop the most [8]. A bit differently, Koh and Liang use *influence functions* to gain understanding on which training points are more relevant to the final classification without having to retrain the network [16]. Their method essentially shows how the model parameters change as we upweight a training point by an infinitesimal amount.

In [27], the authors try to understand CNNs backwards by creating a Deconvolutional Network. A *DeConvNet* can be thought as a ConvNet model with the same components but in reverse, so that it maps features to pixels. A similar idea is followed in [6], where image representations are studied by inverting them with an up-convolutional neural network. Other work focuses on developing human-friendly concepts to help understand the machine. In [13], *Concept Activation Vectors* are introduced to provide an interpretation of the internal state of a network in human-friendly concepts, and *Deep Dream* and *Lucid* are Google projects that intend to humanize what the hidden layers and neurons see in the input images [19]. Their paper develops human-like visualizations for understanding what each neuron is focusing on. In the conclusion, the authors point out that one of the issues that still stands out in network interpretability is finding which units are most meaningful for understanding neural net activations, which is what we study in this paper. Other studies also engage in trying to bridge human concepts and neural networks; for example, [28] investigates how transferable are features in deep neural networks by differentiating between general and specific features learnt by the architecture. It was asked in [1] if CNNs learn class hierarchy, and [2] contains a study of the semantic concepts learnt by the units, such as colors, scenes, and textures.

Other papers bring up criticism to some of the methods we just described. In [14], it is argued that saliency methods lack reliability when the explanation is sensitive to factors that do not contribute to the model prediction, and in [15] it is shown that DeConvNets and Guided Backpropagation do not produce the theoretically correct explanations for a linear model, and so even less for a multi-layer network with millions of parameters. Finally, in [9] and [18], the authors propose that neurons do not encode single concepts and that they are in fact multifaceted, with some concepts being encoded by a group of neurons rather than by a sole neuron by itself.

As summarized in this section, most papers in the literature focus on qualitatively studying the specific features and concepts that are being learnt in the network, rather than quantifying the importance of each hidden neuron towards the final class score. Our method bridges this gap by focusing on quantifying the relevance of each neuron, beyond providing qualitative information based on visualization methods. An exception in the literature is the recent paper [3], where the most relevant units for each class are defined by computing which cause the most accuracy loss when removed individually. However, this approach is too computationally costly.

2.1 Class Activation Mapping

It was first observed in [28] that CNNs behave as object detectors even though no location information about the central object is provided. In this paper, we provide evidence supporting this claim. In [29], they combine a simple modification of the global average pooling layer with the class activation mapping (CAM) technique to allow for a classification-trained CNN to localize specific image regions in a single forward-pass. Examples are provided in Sections 5 and 7, where we compare the activation maps of different neurons with the CAM visualization of the same input image.

2.2 Activation Maximization

The activation maximization method [7], instead of highlighting discriminative regions of the input image (as with saliency maps and CAM), synthesizes an artificial image x^* (which we henceforth call *ideal image*) that maximizes the activation of a target neuron [20]:

$$x^* = \text{argmax}_x \, a_{i,l}(\Theta, x), \qquad (1)$$

where Θ denotes the network parameter sets; i.e., the weights and the bias. This is achieved through an iterative process: after initially setting a random image x_0, the gradients with respect to x_0, i.e., $\partial a_{i,l}/\partial x$, are computed with backpropagation. Each pixel of the initial noisy image is changed iteratively to maximize the activation of the neuron, by applying the update

$$x \longleftarrow x + \gamma \cdot \frac{\partial a_{i,l}(\Theta, x)}{\partial x}, \tag{2}$$

where γ denotes the gradient ascent step size. The final image x represents the *preferred* input image for that neuron.
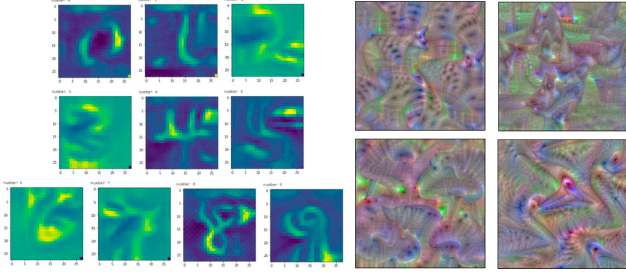

Figure 1: Ideal images synthesized with the activation maximization method for MNIST (left), and the ideal images for the ImageNet classes *Egyptian cat, German shepherd, Pelican,* and *Mushroom* in VGG-16 (clockwise at right).

Note that the activation maximization method allows to both create the ideal image that maximally activates a neuron (Section 6), as well as the ideal image for a particular class (Figure 1). Moreover, the activation maximization method uses the unnormalized final class score immediately before the application of the softmax function, to prevent the values from being squeezed between 0 and 1. We will follow the same principle for our PCACE algorithm. In order to make the ideal image more interpretable, the activation maximization method is normally used with regularization methods, such as ℓ_2 decay or Gaussian blur [20].

3 STATISTICAL METHODS FOR INTERPRETABILITY

3.1 Alternating Conditional Expectation

This section introduces a novel statistical method to rank the hidden units of any neural network in order of importance towards the final class score: PCACE. The name PCACE results from the fusion of PCA (Principal Component Analysis), which we use for dimensionality reduction, and ACE (Alternating Condition Expectation), which we use to compute the maximal correlation coefficient between a dependent variable (the final class score) and multiple independent variables (each entry of the activation matrix of a particular neuron). The units will be ranked on the basis of the strength of their possibly non-linear relationship with the final class score.

The Alternating Conditional Expectation (ACE) algorithm [4] is a non-parametric approach for estimating the transformations that lead to the maximal multiple correlation of a response and a set of independent variables in regression and correlation analysis. The iterative algorithm works by estimating optimal transformations in a multiple regression setup, to find the maximal correlation

between multiple independent variables X_i and a dependent variable Y. These transformations minimize the unexplained variance of a linear relationship between the transformed response variable and the sum of the transformed predictor variables [24]. Moreover, ACE does not require any assumptions on the response or predictor variables and is entirely automatic, in contrast to more recent methods for nonparametric dependence testing (e.g., kernel methods like the Hilbert-Schmidt Independence Criterion [10] require specifying a kernel and its lengthscale). ACE also contrasts with Generalised Additive Models in that it transforms the response variable.

As summarized in [24], the general regression model for p independent variables (predictors) $X_1, X_2, \ldots, X_p$ and a response variable Y is given by

$$Y = \beta_0 + \sum_{i=1}^{p} \beta_i X_i + \epsilon, \tag{3}$$

where $\beta_0, \ldots, \beta_p$ are regression coefficients that are to be estimated, and ϵ is the error term. However, this model is not well-suited for CNN architectures, since the activation functions in the network are *not* linear, and so instead we must use a non-linear regression method. The ACE regression model has the form

$$\Theta(Y) = \alpha + \sum_{i=1}^{p} \phi_i(X_i) + \epsilon, \tag{4}$$

where $\Theta(Y)$ is a function of the response variable Y with zero mean and unit variance, and $\phi_i(X_i)$ are zero-mean functions of the predictors X_i, for $i = 1, \ldots, p$. Then ACE finds the functions that minimize the error variance not explained in the regression, i.e.,

$$\epsilon^2(\Theta, \phi_1, \ldots, \phi_p) = E\left\{\left[\Theta(Y) - \sum_{i=1}^{p} \phi_i(X_i)\right]^2\right\}, \tag{5}$$

with respect to Θ and $\phi_1, \ldots, \phi_p$.

In our case, given a fixed neuron in a convolutional layer in the architecture, the independent variables X_i correspond to the entries of the activation map produced from the convolution between the fixed weights of the neuron and the input image. The predictor variable Y corresponds to the final class score of the input image *before* the application of the softmax function, to avoid the compression of the values into the interval $[0, 1]$. Since ACE will assign a number between -1 and 1 to each hidden unit, it allows to rank all of the hidden units of the network in a reliable, normalized, and clear way. The higher the ACE value is, the stronger is the relationship between that neuron and the score of a particular class.

3.2 Applying ACE to CNN Architectures

We now give a more detailed explanation of how to apply ACE to the CNN setting. Given a trained CNN architecture, we fix one of the convolutional layers, and our goal is to produce a ranking of the m neurons that form that layer.[2] Each of these neurons has a fixed matrices of weights, which has size corresponding to the kernel of the convolutional layer. This matrix of weights produces an activation map whenever an input image is fed into the network

---

[2] As noted in the introduction, PCACE can produce a global ranking of the neurons in the network because the values are normalized. However, in this paper we decide to focus on each layer separately in order to be able to compare the magnitude of the PCACE values across them.

and the neuron is then fired by its activation function. We fix a neuron N_i in this convolutional layer, and we fix a set of input images which will be used to compute the maximal correlation between neuron N_i and the final class score of this set of images.

It can be meaningful to either have this set correspond entirely to training images, entirely to unseen images by the network, or a mix of both. It is also recommended to make this set of pictures be all of the same class, so that we are correlating a neuron with its importance towards the classification of a particular class (for example, in MNIST, we can create a ranking of the neurons for each of the 10 digits separately). However, we note that our algorithm PCACE does not require to be class specific (as we use it in Sections 4 and 5), and can also be applied in regression tasks (as we do in Section 7 for spatiotemporal data). Even for classification tasks, it can be interesting to use a set of input images that consists of a weighted mix of all classes, in order to produce a ranking of the overall most correlated neurons.

Each of these neurons produces an activation map when an input image is fed into the network, which is a matrix of size $k_1 \times k_2$. ACE will then be working with $k_1 \cdot k_2$ independent variables X_i, where each corresponds to one of the entries in the activation maximization map. We favor this method instead of simpler ones (e.g., those consisting of only taking the mean or the maximum value of the $k_1 \cdot k_2$ activation values in order to have just one independent variable), which neglect the complexity and range of values within the activation map. The response variable Y is given by the final correct class score right before applying the softmax function. For each pass of one of the n input images of the set, we store the $k_1 \cdot k_2$ activation maximization values as a column in a matrix M, and the final class score in a vector v.

At this stage, we cannot input M and v directly into the ACE algorithm. Firstly, the dimensions of both the matrix M and the vector v are normally too large for ACE to be able to compute the maximal correlation coefficient in a reasonable computation time. Secondly, the ACE algorithm halts and outputs nan if values in the matrix M are too small due to division by 0. One way to fix this is to standardize each row of the matrix M (i.e., standardize each of the $k_1 \cdot k_2$ predictor variable vectors): center to the mean and component-wise scale to unit variance. However, the issue might still not be resolved if the standard deviation of any of the $k_1 \cdot k_2$ predictor variables is 0.

Both problems can be solved at once by applying the PCA (Principal Component Analysis) algorithm before inputting M and v into ACE. PCA performs a linear mapping of the data in the original matrix to a lower-dimensional space such that the variance of the data in the low-dimensional representation is maximized by using eigenvalue decomposition [12]. After applying PCA, the new matrix M' will have smaller dimensions but will have inherited the maximum possible variance of the original data. Once the dimension of the original matrix M has been reduced, we can now apply ACE in a computationally efficient way. It is important to remark that when using PCA for the purposes of PCACE one needs to reduce the number of rows (the number of predictor variables), and not the number of columns (the number of input images).

We take the absolute value of the final PCACE value, as we are concerned with the magnitude of the correlation. In fact, very few channels in our experiments presented a negative ACE correlation.

3.3 The PCACE Algorithm

PCACE is an algorithm that can be effectively used to rank hidden channels for the following reasons:

(1) It bridges together two effective, reliable, and powerful statistical methods: the Alternating Conditional Expectation (ACE) and the Principal Component Analysis (PCA).
(2) By using PCA on top of ACE, we can significantly reduce the dimensions of the matrix M, therefore making the ranking computation time efficient.
(3) Since the number of predictor variables is significantly reduced, we can increment the number of input images and PCACE will still be computationally efficient.
(4) ACE will not encounter nan problems because PCA standardizes the original data.
(5) The output PCACE values are all between 0 and 1, which yields a standardized method to make comparisons across different neurons and from different layers.

4 AN EXAMPLE WITH MNIST

We first provide an example of applying the PCACE algorithm to the MNIST dataset to produce a ranking of the neurons in the first convolutional layer for each of the 10 digits. The size of the activation matrix of the hidden units of the first convolutional layer in our architecture is 26×26, which means there are 676 predictor variables X_i. For each input image, we record the activation value that each of the elements in the activation matrix achieves, store it, and record the final class score. After feeding m input images into the network, we obtain 676 vectors of length m for the predictor variables, and one vector of length m for the response variable. We use 500 input images of each particular digit from the training dataset. After repeating this procedure with all the neurons in the layer for each of the 10 digits, we obtain the final PCACE rankings.

We observe that many of the channels that PCACE ranks as the most important for each digit are precisely the ones that focus on the surroundings and details of the digit instead of the global shape, which we call *reverse channels*. The reverse channels for MNIST are numerically characterized by having a negative weight mean. They appear to activate at the edges, corners, and surroundings of the digit when the input digit is a real image from the training set (or a mean of the images in the training set), but are nearly dead (i.e., activation close to 0) when the input image is the ideal class digit created with the activation maximization method.

This indicates a potential shortcoming of the activation maximization method: the artificial image produced by the activation maximization does not activate most of the channels that are ranked by the PCACE algorithm as the most important for the correct classification of the digit. These channels do not focus on the main class object (the digit), and therefore are not fired because the pixels that they identify are not maximized by backpropagation. More broadly, we aim to point out that interpretability methods that are gradient-based (such as CAM or saliency maps) do not necessarily correlate well with those that are statistics-based (such as the correlation method of PCACE). As we further develop in Sections 5 and 6, the channels in CNN architectures encode more information than that explicitly related to the class object, and this complexity is oftentimes lost with gradient-based methods.

5 IMAGENET RESULTS

We try our PCACE algorithm with the more complex ImageNet (1,000) dataset, on both the architectures ResNet-18 and VGG-16. With ResNet-18 we are able to use CAM, and thus we can compare the CAM visualizations with the activation maps of the top PCACE channels. While it is not possible to obtain CAM visualizations with VGG-16 [29], the first convolutional layers preserve the size of the input image, which allows for a clearer visualization of the activation maps (112×112 for ResNet-18 as opposed to 224×224 for VGG-16). We apply the PCACE algorithm on all convolutional layers of both architectures for several ImageNet classes, using 300 input images for each class (randomly selected from the ImageNet dataset), and using PCA to reduce by half the size of the matrix that we input to the ACE algorithm. The algorithm yields class-specific rankings of the neurons in each convolutional layer (ranked independently in each layer).

We then plot the activation maps of the top PCACE channels (i.e., those that show the highest correlations) and compare them to the CAM visualization of [29]. Examples are shown in Figure 2. We support the claim in [3, 28] that object detectors emerge in CNNs despite having no supervised training. Not only so, but we observe that the different channels are detectors for different kinds of objects in the input image, and that they are strongly consistent across multiple input images. For example, as shown in Figure 3, channel 6 in the first convolutional layer of ResNet 18 always detects (i.e., has higher activation values) the object immediately next to the class object (which in this case is *Egyptian cat*). Moreover, channel 6 is the top 2 highest-ranked PCACE channel, whereas channel 24 (which consistently focuses on the main class object) ranks last in the PCACE algorithm. The magnitudes of the activations are also consistent in each channel. We visualize different PCACE-ranked channels across the figures in the paper for breadth purposes.
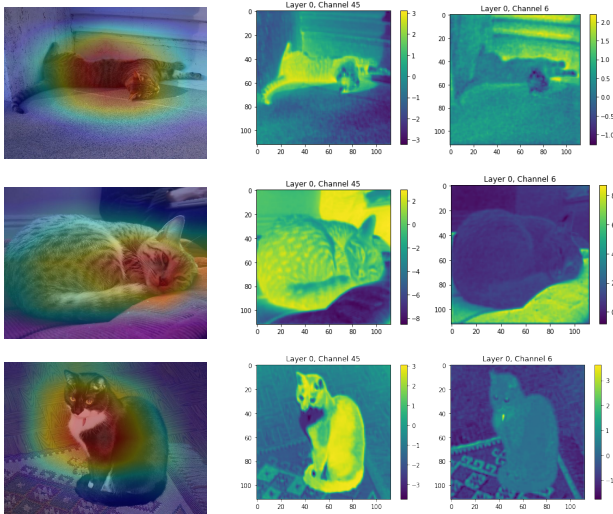
Not only does this show the locative power of CNNs despite being trained on only labels, but we also observe that the most correlated channels tend to be those that do not target the main class object, and focus on different objects in the images or the edges of the shapes in the images. Figure 3 shows how some of the top 5 PCACE channels consistently focus on objects that are not the class object or the edges of the image shape, whereas they tend to not highlight the main class object. There are also instances when this is not the case and some of the top PCACE channels focus on the main class object, but they are scarce.

We repeated the same visualizations with VGG-16 and obtained the same observations, which are shown in Figure 4 (note that VGG-16 does not allow for CAM visualizations, which is why they do not appear in Figure 4). Therefore, these findings are not specific to the ResNet-18 architecture, and we hypothesize that they extend to other architectures as well. These observations are coherent with the findings in [17], where they argue that different neural networks learn the same representations.
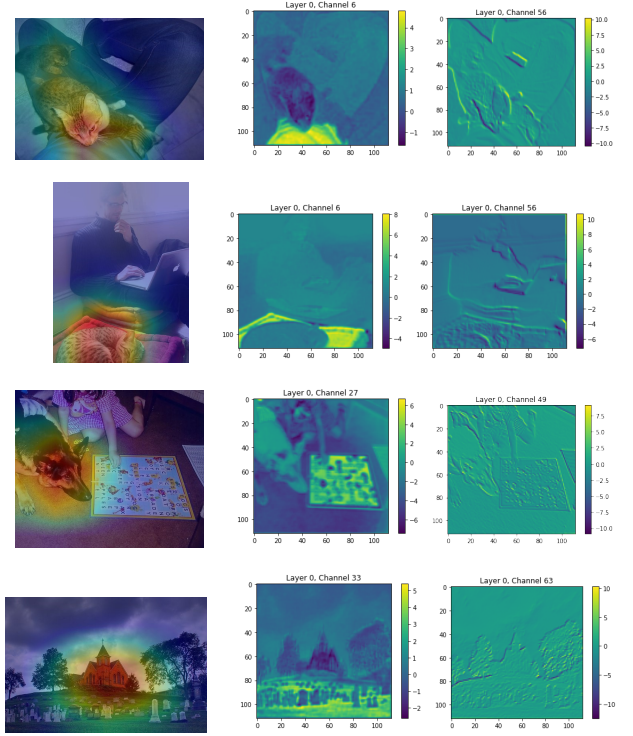

Figure 3: CAM visualization (left) and the activation maps of two of the top 5 PCACE channels (middle, right) for the ImageNet classes *Egyptian cat, German shepherd,* and *Church* in ResNet-18.

Figure 2: CAM visualization (left), activation map of the bottom PCACE channel (middle), and activation map of the second highest PCACE channel (rigth) with ResNet-18 for the ImageNet class *Egyptian cat.*

Finally, Figure 5 shows sorted PCACE values for all the convolutional layers in the VGG-16 architecture for the ImageNet class *Egyptian cat* in order to analyze how the correlation values vary across the different convolutional layers. The PCACE values follow similar trends across the different layers, and they are almost superposed. Deeper layers tend to have lower PCACE values at

the smaller end, but higher correlations on the larger end (e.g., the highest PCACE value in the first Conv layer does not exceed 0.32, whereas the highest PCACE value in Features 28 exceeds 0.65). However, Figure 5 shows that the count distribution of the PCACE values within the same block still differs across layers.

As with MNIST, we find no relevant correlation between PCACE rankings of ImageNet classes, hereby indicating that a global ranking cannot be directly deduced from the class-based PCACE ranking. We repeated the same experiment with the activation maximization method as we did for MNIST: we feed into the network the ideal image generated through activation maximization for each class and record the activations of each channel. We do not find any correlation between PCACE values and the maximum or mean activation of each channel, which again indicates some discrepancy between gradient-based and correlation-based methods.
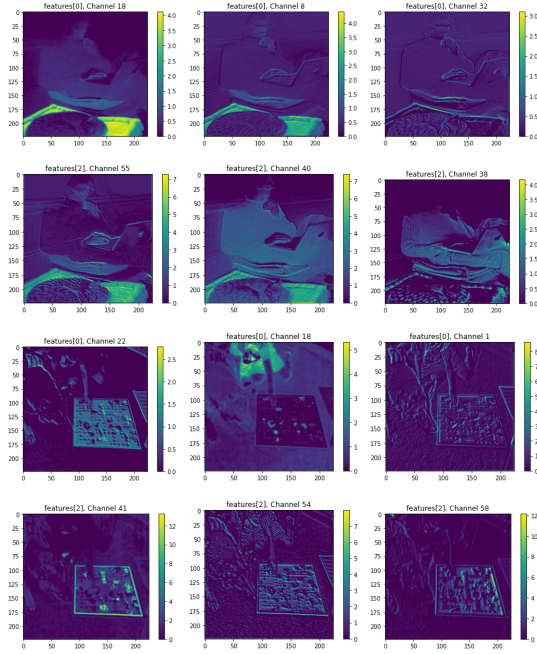

Figure 4: Activation maps of three of the top 5 PCACE channels for *Egyptian cat* and *German shepherd* in VGG-16.

6 FILTER VISUALIZATION WITH ACTIVATION MAXIMIZATION

As explained in Section 5, it becomes harder to visualize activation maps of the deeper convolutional layers as their size reduces. Therefore, we turn to the technique of activation maximization in order to analyze the feature visualizations of the different channels across multiple convolutional layers.

We produce the ideal synthesized image for each channel in all convolutional layers of VGG-16 using the library `tf-keras-vis`.[3] We minimize in the ℓ_2 norm using loss steps of size 50. In Figure 6 we show the filter visualization of the top 5 PCACE channels for the class *Egyptian cat* for each convolutional layer in VGG-16. Note

---

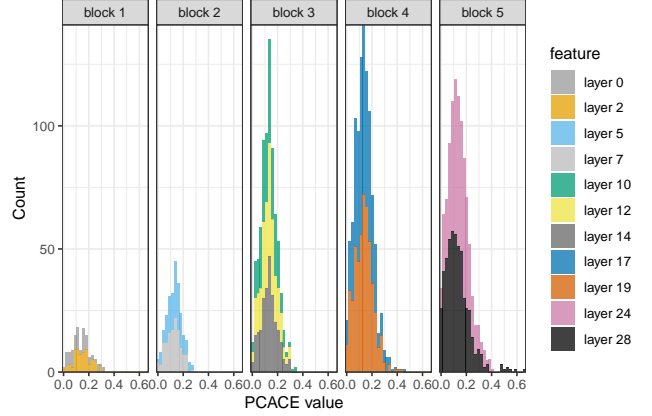[3]https://github.com/keisen/tf-keras-vis.

Figure 5: Histogram of the PCACE values for the successive convolutional layers in VGG-16 computed with the ImageNet class *Egyptian cat*.

that the ideal image of a channel is independent of the class, but the PCACE ranking is not. As in previous situations, showing the feature visualizations for all channels would have required 4,224 images. With the PCACE algorithm, we can meaningfully decide which subset to visualize without this choice being arbitrary. Figure 6 also shows how the channels are sequentially encoding more complex information as we inquire deeper into the network.

7 APPLICATION OF PCACE TO STREET-LEVEL IMAGES FOR AIR POLLUTION DETECTION

In this section, we show how our ranking algorithm yields a rigorous method for interpreting deep learning models trained on spatiotemporal data. Specifically, we study the PCACE algorithm in a real-word application where the aim is to predict air pollution levels from street-level images. We use weights from a model trained by by [23] using a slightly modified ResNet-18 architecture where the outputs are continuous pollutant levels. In this paper, we use the weights trained using a subset of images from the city of London for predicting annual NO_2 levels.

We include this application for two reasons: first, to present the use of PCACE towards explainable deep learning systems trained on spatial and temporal coverage of air quality levels. Secondly, we note that Sections 4 and 5 are centered around *classification tasks*. In this section, the task to predict air pollution from images corresponds to a *regression task*, in which there is only one final score instead of all the class scores. Still, since in this paper we have analyzed PCACE from a class-based perspective, we apply the PCACE algorithm to 300 input images from most polluted areas in the city of London (i.e., top decile). For each of these 300 images, both the true and predicted NO_2 values are above 84 $\mu g/m^3$. This prompts the following question: what are the channels detecting in high-pollution images? In Figure 7 we present some examples of the original input street-level image, its CAM visualization, and the activation maps of the bottom 2 and top 2 PCACE channels.

Our results show that the channels continue to act as object detectors despite having trained the architecture with a regression
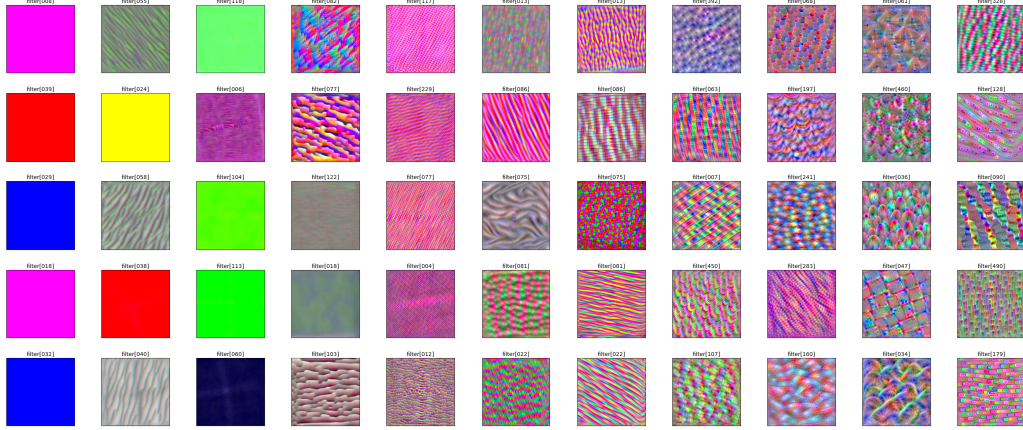
Figure 6: Activation maximization method optimizing the top 5 PCACE channels of all convolutional layers of VGG-16 for the ImageNet class *Egyptian cat* (left to right from first convolutional layers to the deeper ones).
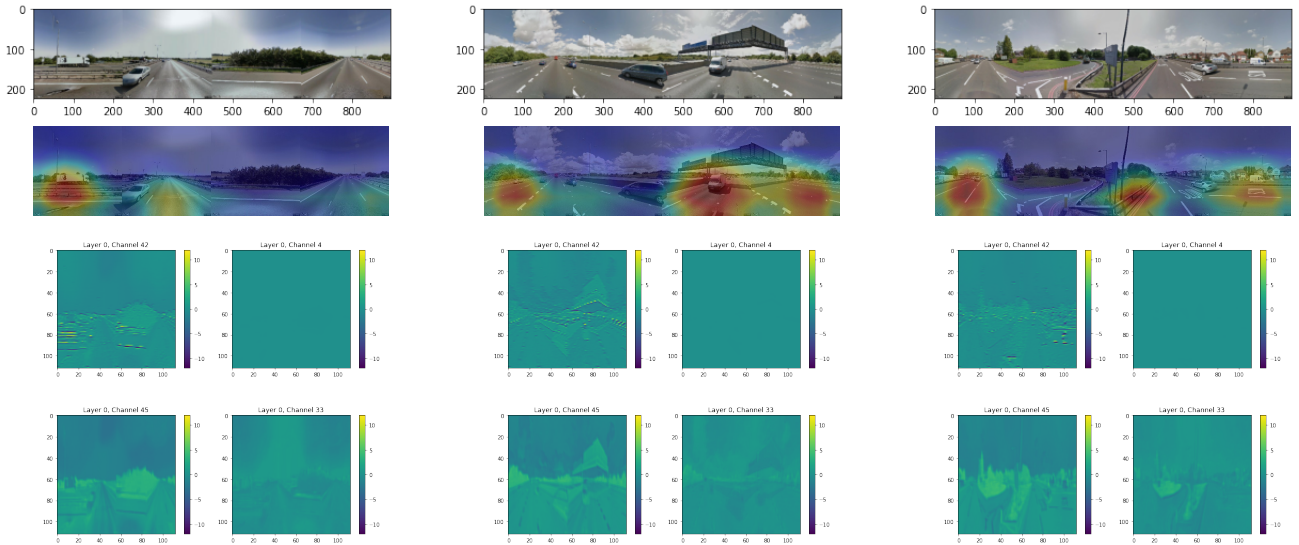

Figure 7: Input image (4 street-level images from a single vantage point covering 360°), CAM visualization, and activation maps of lowest 2 PCACE channels (top) and highest 2 PCACE channels (bottom) with ResNet-18.

task instead of a classification one. Since in this setting we do not have a class object as reference, understanding what the neurons are focusing on becomes a more open-ended question. We find that while some channels continue to focus on the edges of the image, many act as object detectors for buildings and, more remarkably, trees. As it was the case for MNIST and ImageNet, the activation maps of a fixed channel across different input images are consistent. For example, as can be seen in Figure 7, channel 42 consistently activates at the edges of the image whereas channel 45 detects the trees in the input image. Importantly, we remark that this allows to perform computer vision tasks which are very relevant to spatiotemporal data, such as image segmentation, in an unsupervised manner, given that the training has occurred in a regression setting.

In particular, channel 45, which is the second highest ranked PCACE channel, is a surprisingly powerful tree detector, even when

they appear as very small parts of the image, as it is the case of the top right image in Figure 8. More surprisingly, there is no class *tree* in ImageNet 1,000, which provides more evidence for the strong object detection capabilities. Moreover, we observe that the bottom PCACE channels are not doing tree detection, as shown in Figure 7. This further indicates how the PCACE ranking can help in interpreting how the network is encoding the key concepts and identifying certain classes of objects.

When we compare the activation maps of the PCACE channels to the CAM visualizations of the same street-level images of [23], as we exemplify in Figure 7, we observe that the CAM method tends to highlight the pixels that correspond to parts of the road, which likely indicate higher levels of pollution. Since the activation maps act as object detectors, in the case of regression tasks we recommend leveraging CAM methods (which can highlight particular areas
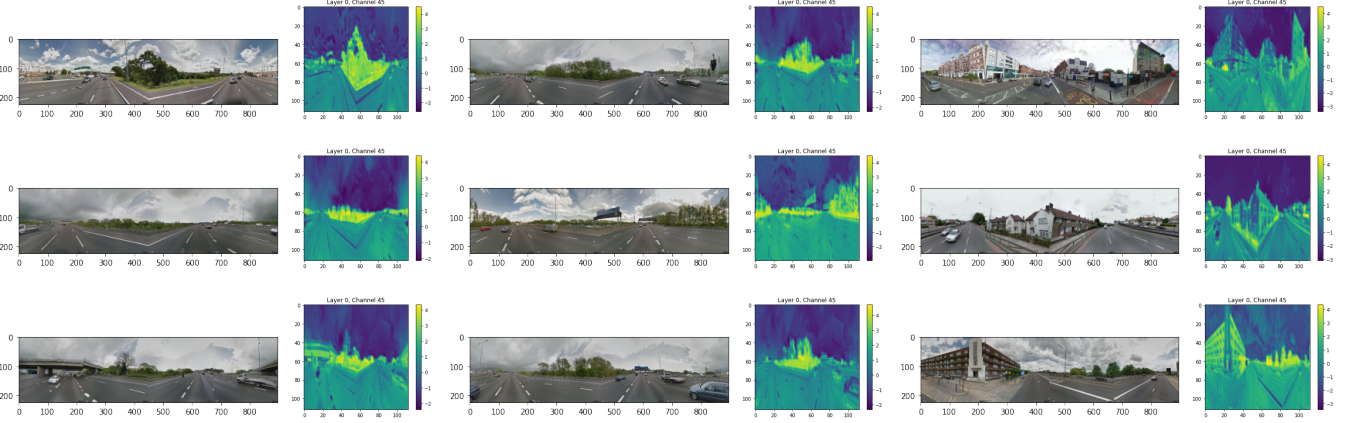
Figure 8: Channel 45 in the first convolutional layer of ResNet-18 trained on ImageNet and fine-tuned with street-level images acts as a powerful tree detector. Channel 45 was the second highest scoring using PCACE, as seen in Fig. 7 (colorbar to scale).

of the image beyond specific objects) with the PCACE activation maps. However, the PCACE method alone is able to rank highest the neurons that do tree object detection, and thus it is sensible to conclude that trees in cities are most predictive for pollutant methods. This indicates the effectiveness of the PCACE ranking in interpreting deep learning models trained on spatiotemporal data.

8 CONCLUSIONS AND DISCUSSION

In this paper we have presented the PCACE algorithm: a new statistical method that combines Principal Component Analysis for dimensionality reduction with the Alternating Conditional Expectation algorithm to find the maximal correlation coefficient between a hidden neuron and the final class score. PCACE yields a rigorous statistical and, most importantly, standardized method which is able to quantify the relevance of each neuron towards classification. Thus, PCACE constitutes a useful tool to analyze deep learning models trained on spatiotemporal data, as shown by the top ranking of tree detector channels on our street-level air pollution imagery.

We have tested our algorithm in two well-known datasets for classification tasks, MNIST and ImageNet. We have also used it in a real-world application to street-level images for detecting air pollution, which shows how we can use PCACE in regression tasks beyond classification ones. We have provided extensive evidence that the channels in CNN architectures act as object detectors, and not only for the main class object in the case of classification tasks, but also for secondary objects in the images as well. Moreover, most of the top PCACE channels tend to be those that detect these secondary objects or corners and edges of the input image. We have shown that the type of object detection ability is consistent for each particular channel independent of the input image (e.g., edges, small set of pixels, secondary object). Moreover, these object detection capabilities are preserved in the case of regression tasks, as we have shown through the strong tree detection properties of the top PCACE channels in the street-level air pollution images.

The PCACE values of each layer can be used in several ways. Firstly, one could study the correlation values directly, as in Figure 5. We showed that the PCACE values across the different convolutional layers in an architecture are coherent and follow the same

trend within each block, with deeper layers acquiring higher correlation values. We believe that the coherence of the PCACE values across layers justifies using the algorithm to compare channels between different layers. We have ranked the neurons within the same convolutional layer, but for future work it would be interesting to rank and compare the PCACE values of all the neurons in the architecture, given the standardized properties of PCACE.

Secondly, we can use the PCACE rankings for visualization purposes. In this paper we have focused on producing the activation maps of the top PCACE channels, as well as the synthesized image produced by the activation maximization method when optimizing for each of the channels separately. The activation maps are meant to be interpreted with respect to an input image, and hence are better for showing the object detection properties of CNNs. On the other hand, the activation maximization method allows us to perform a feature visualization study of the top PCACE channels, as shown in Figure 6. We believe that many other visualization and interpretability methods which are channel-based can be added on top of PCACE with the goal of interpreting CNN models trained on spatiotemporal data. A pressing issue in the current research work on explainability is the infeasibility of visualizing *all* of the neurons in an architecture, which range in the thousands. Previous works needed to make an arbitrary choice of which neurons to illustrate when presenting an interpretability method. The PCACE algorithm now allows to make this choice in a rigorous manner.

Other directions for future work include pruning on top of PCACE to further understand the impact of the top-ranked neurons. It would also be necessary to compare the PCACE rankings to other quantifications of the definition of *importance* of a channel, as recently done in [3]. However, there is a scarcity of work in this direction, and thus a lack of standardized methods to compare to. We hope that our algorithm provides a first step towards a quantifiable notion of explainability in the deep learning field. Beyond street-level images for air pollution, we hope to apply PCACE to other types of sensing imagery in the settings of Earth sciences, urban planning, public safety, and epidemiology, among many others. There is still much to be done to open up these black boxes.

REFERENCES

[1] Bilal Alsallakh, Amin Jourabloo, Mao Ye, Xiaoming Liu, and Liu Ren. 2018. Do convolutional neural networks learn class hierarchy? *IEEE Transactions on Visualization and Computer Graphics*, 24(1):152–162.

[2] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. 2017. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 6541–6549.

[3] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba. 2020. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*, 117(48):30071–78.

[4] Leo Breiman and J. H. Friedman. 1985. Estimating optimal transformations for multiple regression and correlation. *Journal of the American Statistical Association*, 80(391):580–598.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 248–255.

[6] Alexey Dosovitskiy and Thomas Brox. 2015. Inverting visual representations with convolutional networks. arXiv preprint *arXiv:1506.02753*.

[7] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Vincent Pascal. 2019. Visualizing Higher-Layer Features of a Deep Network. Technical report, University of Montreal.

[8] Ruth Fong and Andrea Veldadi. 2017. Interpretable explanations of black boxes by meaningful perturbation. In *ICCV*.

[9] Ruth Fong and Andrea Vedaldi. 2018. Net2Vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. In *Proceedings of CVRP*.

[10] Arthur Gretton, Kenji Fukumizu, Choon Hui Teo, Le Song, Bernhard Schölkopf, and Alexander J. Smola. 2007. A Kernel Statistical Test of Independence. In *Neural Information Processing Systems*.

[11] Bryce Goodman and Seth Flaxman. 2017. European Union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine*, 38(3):50–57.

[12] Ian Jolliffe. 2014. Principal Component Analysis. Springer-Verlag, New York.

[13] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. 2018. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In *ICML*.

[14] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T. Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. 2017. The (un)reliability of saliency methods. arXiv preprint *arXiv:1711.00867*.

[15] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T. Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. 2017. Learning how to explain neural networks: PatternNet and PatternAttribution. *arXiv:1705.05598*.

[16] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *ICML*.

[17] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. 2015. Convergent Learning: Do different neural networks learn the same representations? In *Feature Extraction: Modern Questions and Challenges*, pages 196–212.

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

[19] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. 2018. Feature visualization. In *Distill*.

[20] Zhuwei Qin, Fuxun Yu, Chenchen Liu, and Xiang Chen. 2018. How convolutional neural networks see the world – a survey of convolutional neural network visualization methods. arXiv preprint *arXiv:1804.11191*.

[21] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2016. Grad-CAM: Visual explanations from deep networks via gradient-based localization. arXiv preprint *arXiv:1610.02391*.

[22] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint *arXiv:1409.1556*.

[23] Esra Suel, Meytar Sorek-Hamer, Izabela Moise, Michael von Pohle, Adwait Sahasrabhojanee, Ata Asanjian, Emily Deardorff, Violet Lingenfelter, Nikunj Oza, Majid Ezzati, and Michael Brauer. 2020. Predicting air pollution spatial variation with street-level imagery. In *NeurIPS MLPH Workshop*.

[24] Duolao Wang and Michael Murphy. 2005. Identifying nonlinear relationships in regression using the ACE algorithm. *Journal of Applied Statistics*, 32(3):243–258.

[25] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. 2015. Understanding neural networks through deep visualization. In *ICML*.

[26] Jerrold H. Zar. 1972. Significance testing of the Spearman rank correlation coefficient. *Journal of the American Statistical Association*, 67(339):578-580.

[27] Matthew D. Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *ECCV*.

[28] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. 2015. Object detectors emerge in Deep Scene CNNs. In *ICLR*.

[29] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. 2016. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2921–2929.

9 SUPPLEMENTARY MATERIAL

We provide pseudocode for the PCACE algorithm in Algorithm 1. For reproducibility purposes, the relevant code and data can be found at https://github.com/silviacasac/ranking-CNN-neurons.

Algorithm 1 PCACE Algorithm.

1: *For each channel c in the conv layer l do*:
2:     *For each input image do*:
3:         *Store the activations of channel c into matrix $X_{c,l}$.*
4:         *Store the final class score into vector $Y_{c,l}$.*
5:     *Apply PCA to reduce the dimensionality of matrix $X_{c,l}$.*
6:     *Obtain newly reduced matrix $X'_{c,l} = \{X'_1, X'_2, \ldots, X'_p\}$.*
7:     *Let $\Theta(Y_{c,l}), \phi_1(X'_1), \ldots, \phi_p(X'_p)$ be zero-mean functions.*
8:     *Let $e^2(\Theta, \phi_1, \ldots, \phi_p) = \frac{\mathbb{E}[\Theta(Y_{c,l}) - \sum_{i=1}^{p} \phi_i(X'_i)^2]}{\mathbb{E}[\Theta^2(Y_{c,l})]}$.*
9:     *Let T be the error tolerance.*
10:     *While $e^2 > T$, do:*
11:         *Holding $\phi_1(X'_1), \ldots, \phi_p(X'_p)$ fixed, minimizing e^2 yields*
12:             $\Theta_1(Y_{c,l}) = \mathbb{E}[\sum_{i=1}^{p} \phi_i(X'_i) \mid Y_{c,l}]$.
13:         *Normalize $\Theta_1(Y_{c,l})$ to unit variance.*
14:         *For each k, fix other $\phi_i(X'_i)$ and $\Theta(Y_{c,l})$ minimizing e^2.*
15:         *The solution is $\tilde{\phi}_k = \mathbb{E}[\Theta(Y_{c,l}) - \sum_{i \neq k} \phi_i(X'_i) \mid X'_k]$.*
16:     *Return the absolute value of the Pearson product-moment*
17:     *correlation coefficients.*
18: *Sort the final c values.*