

# On Pixel-wise Explanations for Non-Linear Classifier Decisions by Layer-wise Relevance Propagation<sup>\*</sup>

Sebastian Bach<sup>1,2,\*,†</sup>, Alexander Binder<sup>2,5,†</sup>, Grégoire Montavon<sup>2</sup>, Frederick Klauschen<sup>3</sup>, Klaus-Robert Müller<sup>2,4,\*</sup>, Wojciech Samek<sup>1,2,\*</sup>

**1** Machine Learning Group, Fraunhofer Heinrich Hertz Institute, Berlin, Germany

**2** Machine Learning Group, Technische Universität Berlin, Berlin, Germany

**3** Charité University Hospital, Berlin, Germany

**4** Department of Brain and Cognitive Engineering, Korea University, Seoul, Korea

**5** Information Systems, Technology and Design Pillar, Singapore University of Technology and Design (SUTD), Singapore

<sup>\*</sup> Email: sebastian.bach@hhi.fraunhofer.de, klaus-robert.mueller@tu-berlin.de, wojciech.samek@hhi.fraunhofer.de

<sup>†</sup> Authors contributed equally

<sup>\*</sup>  **PLOS** ONE to appear

## Abstract

Understanding and interpreting classification decisions of automated image classification systems is of high value in many applications, as it allows to verify the reasoning of the system and provides additional information to the human expert. Although machine learning methods are solving very successfully a plethora of tasks, they have in most cases the disadvantage of acting as a black box, not providing any information about what made them arrive at a particular decision. This work proposes a general solution to the problem of understanding classification decisions by pixel-wise decomposition of non-linear classifiers. We introduce a methodology that allows to visualize the contributions of single pixels to predictions for kernel-based classifiers over Bag of Words features and for multilayered neural networks. These pixel contributions can be visualized as heatmaps and are provided to a human expert who can intuitively not only verify the validity of the classification decision, but also focus further analysis on regions of potential interest. We evaluate our method for classifiers trained on PASCAL VOC 2009 images, synthetic image data containing geometric shapes, the MNIST handwritten digits data set and for the pre-trained ImageNet model available as part of the Caffe open source package.

## Introduction

Classification of images has become a key ingredient in many computer vision applications, e.g. image search [1], robotics [2], medical imaging [3], object detection in radar images [4] or face detection [5]. Two particularly popular approaches, neural networks [6] and Bag of Words (BoW) models [7], are widely used for these tasks and were among the top submissions in competitions on image classification and ranking such as ImageNet [8], Pascal VOC [9] and ImageCLEF [10]. However, like many methods in machine learning, these models often lack a straightforward interpretability of the classifier predictions. In other words the classifier acts as a *black box* and does not provide detailed information about why it reaches a certain classification decision.

This lack of interpretability is due to the non-linearity of the various mappings that process the raw image pixels to its feature representation and from that to the final classifier function. This is a considerable drawback in classification applications, as it hinders the human experts to carefully verify the classification decision. A simple *yes* or *no* answer is sometimes of limited value in applications, where questions like *where* something occurs or *how* it is structured are more relevant than a binary or real-valued one-dimensional assessment of mere presence or absence of a certain structure.

In this work, we aim to close the gap between classification and interpretability both for multilayered

neural networks and Bag of Words (BoW) models over non-linear kernels which are two classes of predictors which enjoy popularity in computer vision. We will consider both types of predictors in a generic sense trying to avoid whenever possible a priori restrictions to specific algorithms or mappings. For the first part, the Bag of Words models will be treated as an aggregation of non-linear mappings over local features in an image which includes a number of popular mapping methods such as Fisher vectors [11], regularized coding [12–14] and soft coding [15], combined with differentiable non-linear kernels and a range of pooling functions including sum- and max-pooling. For the second part, the neural networks (e.g. [6, 16]), we will consider general multilayered network structures with arbitrary continuous neurons and pooling functions based on generalized p-means.

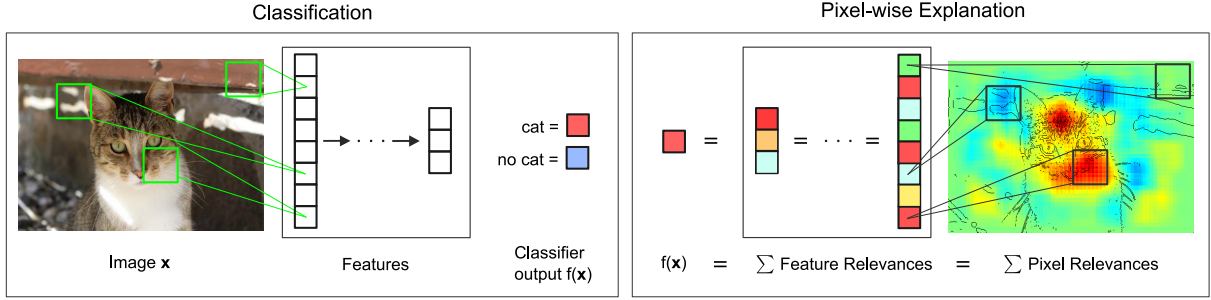
The next Section *Pixel-wise Decomposition as a General Concept* will explain the basic approaches underlying the pixel-wise decomposition of classifiers. In Section *Bag of Words models revisited*, we will give a short recapitulation about Bag of Words features and kernel-based classifiers and summarize related work. *Overview of the decomposition steps* will discuss the decomposition of a kernel-based classifier into sums of scores over small regions of the image, and the projection down to single pixels. Our method then is applied to a number of popular mappings and kernels in Section *Examples for various mappings and kernels*. *Pixel-wise Decomposition for Multilayer Networks* applies both the Taylor-based and layer-wise relevance propagation approaches explained in *Pixel-wise Decomposition as a General Concept* to neural network architectures. The experimental evaluation of our framework will be done in *Experiments* and we conclude the paper with a discussion in *Discussion*.

## Pixel-wise Decomposition as a General Concept

The overall idea of pixel-wise decomposition is to understand the contribution of a single pixel of an image  $x$  to the prediction  $f(x)$  made by a classifier  $f$  in an image classification task. We would like to find out, separately for each image  $x$ , which pixels contribute to what extent to a positive or negative classification result. Furthermore we want to express this extent quantitatively by a measure. We assume that the classifier has real-valued outputs which are thresholded at zero. In such a setup it is a mapping  $f : \mathbb{R}^V \rightarrow \mathbb{R}^1$  such that  $f(x) > 0$  denotes presence of the learned structure. Probabilistic outputs can be treated without loss of generality by subtracting 0.5. We are interested to find out the contribution of each input pixel  $x_{(d)}$  of an input image  $x$  to a particular prediction  $f(x)$ . The important constraint specific to classification consists in finding the differential contribution relative to the state of maximal uncertainty with respect to classification which is then represented by the set of root points  $f(x_0) = 0$ . One possible way is to decompose the prediction  $f(x)$  as a sum of terms of the separate input dimensions  $x_d$  respectively pixels:

$$f(x) \approx \sum_{d=1}^V R_d \quad (1)$$

The qualitative interpretation is that  $R_d < 0$  contributes evidence against the presence of a structure which is to be classified while  $R_d > 0$  contributes evidence for its presence. In terms of subsequent visualization, which however will not be the scope of this paper, the resulting relevances  $R_d$  for each input pixel  $x_{(d)}$  can be mapped to a color space and visualized in that way as a conventional heatmap. One basic constraint will be in the following work that the signs of  $R_d$  should follow above qualitative interpretation, i.e. positive values should denote positive contributions, negative values negative contributions. Fig. 1 depicts the main idea of our method.



**Fig. 1. Visualization of the pixel-wise decomposition process.** In the classification step the image is converted to a feature vector representation and a classifier is applied to assign the image to a given category, e.g., "cat" or "no cat". Note that the computation of the feature vector usually involves the usage of several intermediate representations. Our method decomposes the classification output  $f(x)$  into sums of feature and pixel relevance scores. The final relevances visualize the contributions of single pixels to the prediction. Cat image by pixabay user stinne24.

In this paper we propose a novel concept we denote as *layer-wise relevance propagation* as a general concept for the purpose of achieving a pixel-wise decomposition as in Equation (1). We also discuss an approach based on *Taylor decomposition* which yields an approximation of layer-wise relevance propagation. We will show that for a wide range of non-linear classification architectures, layer-wise relevance propagation can be done without the need to use an approximation by means of Taylor expansion. The methods we propose in this paper do not involve segmentation. They do not require pixel-wise training as learning setup or pixel-wise labeling for the training phase. The setup used here is image-wise classification, in which during training one label is provided for an image as a whole, however, the contribution of this paper is not about classifier training. The proposed methods are built on top of a pre-trained classifier. They are even applicable to an already pre-trained image classifier as shown in Section *Neural Network for 1000 ILSVRC classes*.

## Layer-wise relevance propagation

We will introduce layer-wise relevance propagation as a concept defined by a set of constraints. Any solution satisfying the constraints will be considered to follow the concept of layer-wise relevance propagation. In later sections we will then derive solutions for two particular classifier architectures and evaluate these solutions experimentally for their meaningfulness. Layer-wise relevance propagation in its general form assumes that the classifier can be decomposed into several layers of computation. Such layers can be parts of the feature extraction from the image or parts of a classification algorithm run on the computed features. As shown later, this is possible for Bag of Words features with non-linear SVMs as well as for neural networks.

The first layer are the inputs, the pixels of the image, the last layer is the real-valued prediction output of the classifier  $f$ . The  $l$ -th layer is modeled as a vector  $z = (z_d^{(l)})_{d=1}^{V(l)}$  with dimensionality  $V(l)$ . Layer-wise relevance propagation assumes that we have a Relevance score  $R_d^{(l+1)}$  for each dimension  $z_d^{(l+1)}$  of the vector  $z$  at layer  $l+1$ . The idea is to find a Relevance score  $R_d^{(l)}$  for each dimension  $z_d^{(l)}$  of the vector  $z$  at the next layer  $l$  which is closer to the input layer such that the following equation holds.

$$f(x) = \dots = \sum_{d \in l+1} R_d^{(l+1)} = \sum_{d \in l} R_d^{(l)} = \dots = \sum_d R_d^{(1)} \quad (2)$$

Iterating Equation (2) from the last layer which is the classifier output  $f(x)$  down to the input layer  $x$  consisting of image pixels then yields the desired Equation (1). The Relevance for the input layer will serve as the desired sum decomposition in Equation (1). In the following we will derive further constraints beyond Equations (1) and (2) and motivate them by examples. As we will show now, a decomposition satisfying Equation (2) per se is neither unique, nor it is guaranteed that it yields a meaningful interpretation of the classifier prediction.

We give here a simple counterexample. Suppose we have one layer. The inputs are  $x \in \mathbb{R}^V$ . We use a linear classifier with some arbitrary and dimension-specific feature space mapping  $\phi_d$  and a bias  $b$

$$f(x) = b + \sum_d \alpha_d \phi_d(x_d) \quad (3)$$

Let us define the relevance for the second layer trivially as  $R_1^{(2)} = f(x)$ . Then, one possible layer-wise relevance propagation formula would be to define the relevance  $R^{(1)}$  for the inputs  $x$  as

$$R_d^{(1)} = \begin{cases} f(x) \frac{|\alpha_d \phi_d(x_d)|}{\sum_d |\alpha_d \phi_d(x_d)|} & \text{if } \sum_d |\alpha_d \phi_d(x_d)| \neq 0 \\ \frac{b}{V} & \text{if } \sum_d |\alpha_d \phi_d(x_d)| = 0 \end{cases} \quad (4)$$

This clearly satisfies Equations (1) and (2), however the Relevances  $R^{(1)}(x_d)$  of all input dimensions have the same sign as the prediction  $f(x)$ . In terms of pixel-wise decomposition interpretation, all inputs point towards the presence of a structure if  $f(x) > 0$  and towards the absence of a structure if  $f(x) < 0$ . This is for many classification problems not a realistic interpretation.

Let us discuss a more meaningful way of defining layer-wise relevance propagation. For this example we define

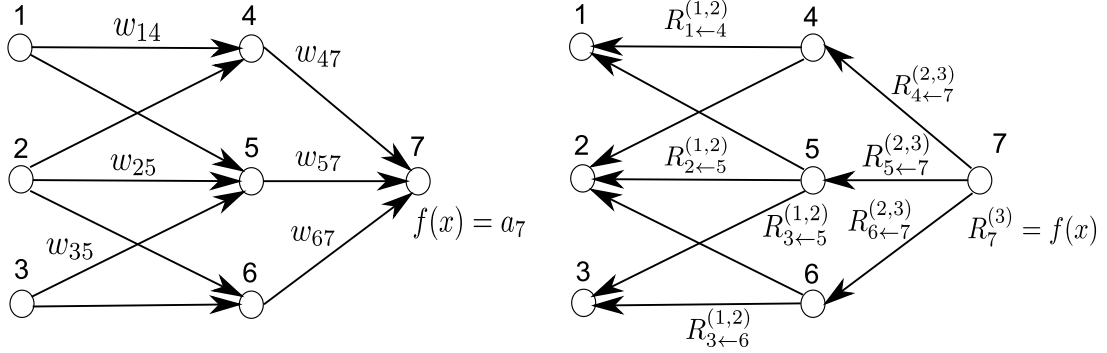
$$R_d^{(1)} = \frac{b}{V} + \alpha_d \phi_d(x_d) \quad (5)$$

Then, the relevance of a feature dimension  $x_d$  depends on the sign of the term in Equation (5). This is for many classification problems a more plausible interpretation. This second example shows that the layer-wise relevance propagation is able to deal with non-linearities such as the feature space mapping  $\phi_d$  to some extent and how an example of layer-wise relevance propagation satisfying Formula (2) may look like in practice. Note that no regularity assumption on the feature space mapping  $\phi_d$  is required here at all, it could be even non-continuous, or non-measurable under the Lebesgue measure. The underlying Formula (2) can be interpreted as a conservation law for the relevance  $R$  in between layers of the feature processing.

The above example gives furthermore an intuition about what relevance  $R$  is, namely, the local contribution to the prediction function  $f(x)$ . In that sense the relevance of the output layer is the prediction itself  $f(x)$ . This first example shows what one could expect as a decomposition for the linear case. The linear case is not a novelty, however, it provides a first intuition.

We give a second, more graphic and non-linear, example. The left panel of Fig. 2 shows a neural network-shaped classifier with neurons and weights  $w_{ij}$  on connections between neurons. Each neuron  $i$  has an output  $a_i$  from an activation function.





**Fig. 2.** Left: A neural network-shaped classifier during prediction time.  $w_{ij}$  are connection weights.  $a_i$  is the activation of neuron  $i$ . Right: The neural network-shaped classifier during layer-wise relevance computation time.  $R_i^{(l)}$  is the relevance of neuron  $i$  which is to be computed. In order to facilitate the computation of  $R_i^{(l)}$  we introduce messages  $R_{i←j}^{(l,l+1)}$ .  $R_{i←j}^{(l,l+1)}$  are messages which need to be computed such that the layer-wise relevance in Equation (2) is conserved. The messages are sent from a neuron  $i$  to its input neurons  $j$  via the connections used for classification, e.g. 2 is an input neuron for neurons 4, 5, 6. Neuron 3 is an input neuron for 5, 6. Neurons 4, 5, 6 are the input for neuron 7.

The top layer consists of one output neuron, indexed by 7. For each neuron  $i$  we would like to compute a relevance  $R_i$ . We initialize the top layer relevance  $R_7^{(3)}$  as the function value, thus  $R_7^{(3)} = f(x)$ . Layer-wise relevance propagation in Equation (2) requires now to hold

$$R_7^{(3)} = R_4^{(2)} + R_5^{(2)} + R_6^{(2)} \quad (6)$$

$$R_4^{(2)} + R_5^{(2)} + R_6^{(2)} = R_1^{(1)} + R_2^{(1)} + R_3^{(1)} \quad (7)$$

We will make two assumptions for this example. Firstly, we express the layer-wise relevance in terms of messages  $R_{i←j}^{(l,l+1)}$  between neurons  $i$  and  $j$  which can be sent along each connection. The messages are, however, directed from a neuron towards its input neurons, in contrast to what happens at prediction time, as shown in the right panel of Fig. 2. Secondly, we define the relevance of any neuron except neuron 7 as the sum of incoming messages:

$$R_i^{(l)} = \sum_{k: i \text{ is input for neuron } k} R_{i←k}^{(l,l+1)} \quad (8)$$

For example  $R_3^{(1)} = R_{3←5}^{(1,2)} + R_{3←6}^{(1,2)}$ . Note that neuron 7 has no incoming messages anyway. Instead its relevance is defined as  $R_7^{(3)} = f(x)$ . In Equation (8) and the following text the terms *input* and *source* have the meaning of being an input to another neuron in the direction as defined during classification time, not during the time of computation of layer-wise relevance propagation. For example in Fig. 2 neurons 1 and 2 are inputs and source for neuron 4, while neuron 6 is the *sink* for neurons 2 and 3. Given the two assumptions encoded in Equation (8), the layer-wise relevance propagation by Equation (2) can be satisfied by the following sufficient condition:

$$R_7^{(3)} = R_{4←7}^{(2,3)} + R_{5←7}^{(2,3)} + R_{6←7}^{(2,3)} \quad (9)$$

$$R_4^{(2)} = R_{1←4}^{(1,2)} + R_{2←4}^{(1,2)} \quad (10)$$

$$R_5^{(2)} = R_{1←5}^{(1,2)} + R_{2←5}^{(1,2)} + R_{3←5}^{(1,2)} \quad (11)$$

$$R_6^{(2)} = R_{2←6}^{(1,2)} + R_{3←6}^{(1,2)} \quad (12)$$

In general, this condition can be expressed as:

$$R_k^{(l+1)} = \sum_{i: i \text{ is input for neuron } k} R_{i \leftarrow k}^{(l, l+1)} \quad (13)$$

The difference between condition (13) and definition (8) is that in the condition (13) the sum runs over the sources at layer  $l$  for a fixed neuron  $k$  at layer  $l+1$ , while in the definition (8) the sum runs over the sinks at layer  $l+1$  for a fixed neuron  $i$  at a layer  $l$ . When using equation (8) to define the relevance of a neuron from its messages, then condition (13) is a sufficient condition in order to ensure that equation (2) holds. Summing over the left hand side in equation (13) yields

$$\begin{aligned} \sum_k R_k^{(l+1)} &= \sum_k \sum_{i: i \text{ is input for neuron } k} R_{i \leftarrow k}^{(l, l+1)} \\ &= \sum_i \sum_{k: i \text{ is input for neuron } k} R_{i \leftarrow k}^{(l, l+1)} \\ &\stackrel{eq.(8)}{=} \sum_i R_i^{(l)} \end{aligned}$$

One can interpret condition (13) by saying that the messages  $R_{i \leftarrow k}^{(l, l+1)}$  are used to distribute the relevance  $R_k^{(l+1)}$  of a neuron  $k$  onto its input neurons at layer  $l$ . Our work in the following sections will be based on this notion and the more strict form of relevance conservation as given by definition (8) and condition (13). We set Equations (8) and (13) as the main constraints defining layer-wise relevance propagation. A solution following this concept is required to define the messages  $R_{i \leftarrow k}^{(l, l+1)}$  according to these equations.

Now we can derive an explicit formula for layer-wise relevance propagation for our example by defining the messages  $R_{i \leftarrow k}^{(l, l+1)}$ . The layer-wise relevance propagation should reflect the messages passed during classification time. We know that during classification time, a neuron  $i$  inputs  $a_i w_{ik}$  to neuron  $k$ , provided that  $i$  has a forward connection to  $k$ . Thus, we can rewrite the left hand sides of Equations (9) and (10) so that it matches the structure of the right hand sides of the same equations by the following

$$R_7^{(3)} = R_7^{(3)} \frac{a_4 w_{47}}{\sum_{i=4,5,6} a_i w_{i7}} + R_7^{(3)} \frac{a_5 w_{57}}{\sum_{i=4,5,6} a_i w_{i7}} + R_7^{(3)} \frac{a_6 w_{67}}{\sum_{i=4,5,6} a_i w_{i7}} \quad (14)$$

$$R_4^{(2)} = R_4^{(2)} \frac{a_1 w_{14}}{\sum_{i=1,2} a_i w_{i4}} + R_4^{(2)} \frac{a_2 w_{24}}{\sum_{i=1,2} a_i w_{i4}} \quad (15)$$

The match of the right hand sides of Equations (9) and (10) against the right hand sides of (14) and (15) can be expressed in general as

$$R_{i \leftarrow k}^{(l, l+1)} = R_k^{(l+1)} \frac{a_i w_{ik}}{\sum_h a_h w_{hk}} \quad (16)$$

While this solution (16) for message terms  $R_{i \leftarrow k}^{(l, l+1)}$  still needs to be adapted such that it is usable when the denominator becomes zero, the example given in Equation (16) gives an idea what a message  $R_{i \leftarrow k}^{(l, l+1)}$  could be, namely the relevance of a sink neuron  $R_k^{(l+1)}$  which has been already computed, weighted proportionally by the input of the neuron  $i$  from the preceding layer  $l$ . This notion holds in an analogous way when we use different classification architectures and replace the notion of a neuron by a dimension of a feature vector at a given layer.

The Formula (16) has a second property: The sign of the relevance sent by message  $R_{i \leftarrow k}^{(l, l+1)}$  becomes inverted if the contribution of a neuron  $a_i w_{ik}$  has different sign then the sum of the contributions from all input neurons, i.e. if the neuron fires against the overall trend for the top neuron from which it inherits

a portion of the relevance. Same as for the example with the linear mapping in Equation (5), an input neuron can inherit positive or negative relevance depending on its input sign. This is a difference to the Equation (4). While this sign switching property can be defined analogously for a range of architectures, we do not add it as a constraint for layer-wise relevance propagation.

One further property is visible here as well. The formula for distribution of relevance is applicable to non-linear and even non-differentiable or non-continuous neuron activations  $a_k$ . An algorithm would start with relevances  $R^{(l+1)}$  of layer  $l+1$  which have been computed already. Then the messages  $R_{i \leftarrow k}^{(l,l+1)}$  would be computed for all elements  $k$  from layer  $l+1$  and elements  $i$  from the preceding layer  $l$  – in a manner such that Equation (13) holds. Then definition (8) would be used to define the relevances  $R^{(l)}$  for all elements of layer  $l$ .

The relevance conservation property can in principle be supplemented by other constraints that further reduce the set of admissible solutions. For example, one could constrain relevance messages  $R_{i \leftarrow k}^{(l,l+1)}$  that result from the redistribution of relevance onto lower-level nodes in a way that is consistent with the contribution of lower-level nodes to the upper layer during the forward pass. If a node  $i$  has a larger weighted activation  $z_{ik} = a_i w_{ik}$ , then, in a qualitative sense, it should also receive a larger fraction of the relevance score  $R_k^{(l+1)}$  of the node  $k$ . In particular, for all nodes  $k$  satisfying  $R_k, \sum_i z_{ik} > 0$ , one can define the constraint  $0 < z_{ik} < z_{i'k} \Rightarrow R_{i \leftarrow k}^{(l,l+1)} \leq R_{i' \leftarrow k}^{(l,l+1)}$ . (The formulas provided in section *Pixel-wise Decomposition for Multilayer Networks* adhere to this ordering constraint.) Nevertheless, nothing is said about the exact relevance values beyond their adherence to the constraint stated above.

To summarize, we have introduced layer-wise relevance propagation in a feed-forward network. In our proposed definition, the total relevance is constrained to be preserved from one layer to another, and the total node relevance must be equal to the sum of all relevance messages incoming to this node and also equal to the sum of all relevance messages that are outgoing to the same node. It is important to note that the definition is *not* given as an algorithm or a solution of an objective with a distinct minimum. Instead, it is given as a set of constraints that the solution should satisfy. Thus, different algorithms with different resulting solutions may be admissible under these constraints.

## Taylor-type decomposition

One alternative approach for achieving a decomposition as in (1) for a general differentiable predictor  $f$  is first order Taylor approximation.

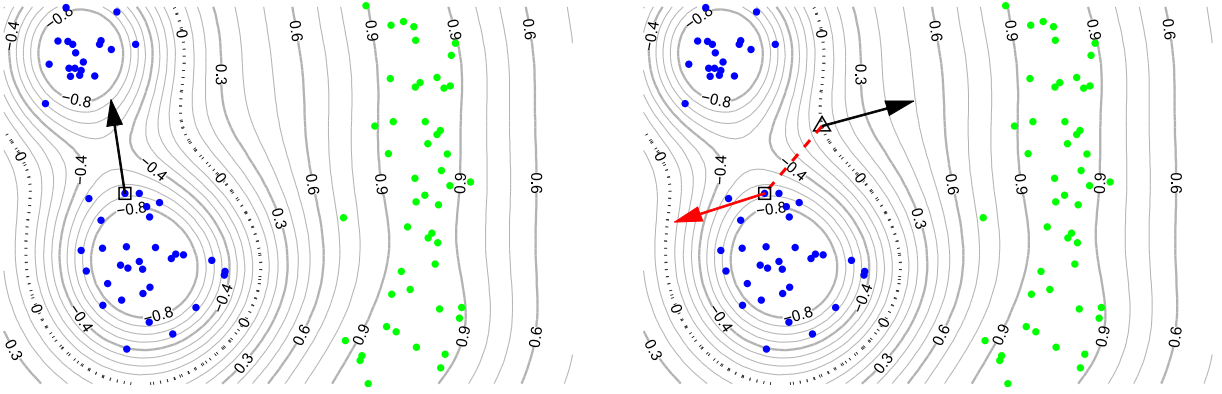
$$\begin{aligned} f(x) &\approx f(x_0) + Df(x_0)[x - x_0] \\ &= f(x_0) + \sum_{d=1}^V \frac{\partial f}{\partial x_{(d)}}(x_0)(x_{(d)} - x_{0(d)}) \end{aligned} \quad (17)$$

The choice of a Taylor base point  $x_0$  is a free parameter in this setup. As said above, in case of classification we are interested to find out the contribution of each pixel relative to the state of maximal uncertainty of the prediction which is given by the set of points  $f(x_0) = 0$ , since  $f(x) > 0$  denotes presence and  $f(x) < 0$  absence of the learned structure. Thus,  $x_0$  should be chosen to be a root of the predictor  $f$ . For the sake of precision of the Taylor approximation of the prediction,  $x_0$  should be chosen to be close to  $x$  under the Euclidean norm in order to minimize the Taylor residuum according to higher order Taylor approximations. In case of multiple existing roots  $x_0$  with minimal norm, they can be averaged or integrated in order to get an average over all these solutions. The above equation simplifies to

$$f(x) \approx \sum_{d=1}^V \frac{\partial f}{\partial x_{(d)}}(x_0)(x_{(d)} - x_{0(d)}) \quad \text{such that } f(x_0) = 0 \quad (18)$$

The pixel-wise decomposition contains a non-linear dependence on the prediction point  $x$  beyond the Taylor series, as a close root point  $x_0$  needs to be found. Thus the whole pixel-wise decomposition is not a linear, but a locally linear algorithm, as the root point  $x_0$  depends on the prediction point  $x$ .

Several works have been using sensitivity maps [17–19] for visualization of classifier predictions which were based on using partial derivatives *at the prediction point*  $x$ . There are two essential differences between sensitivity maps based on derivatives at the prediction point  $x$  and the pixel-wise decomposition approach. Firstly, there is no direct relationship between the function value  $f(x)$  at the prediction point  $x$  and the differential  $Df(x)$  at the same point  $x$ . Secondly, we are interested in explaining the classifier prediction relative to a certain state given by the set of roots of the prediction function  $f(x_0) = 0$ . The differential  $Df(x)$  at the prediction point does not necessarily point to a root which is close under the Euclidean norm. It points to the nearest local optimum which may still have the same sign as the prediction  $f(x)$  and thus be misleading for explaining the difference to the set of root points of the prediction function. Therefore derivatives at the prediction point  $x$  are not useful for achieving our aim. Fig. 3 illustrates the qualitative difference between local gradients (black arrows) and the dimension-wise decomposition of the prediction (red arrow).



**Fig. 3. An exemplary real-valued prediction function for classification with the dashed black line being the decision boundary which separates the blue from the green dots. The blue dots are labeled negatively, the green dots are labeled positively. Left: Local gradient of the classification function at the prediction point. Right: Taylor approximation relative to a root point on the decision boundary. This figure depicts the intuition that a gradient at a prediction point  $x$  – here indicated by a square – does not necessarily point to a close point on the decision boundary. Instead it may point to a local optimum or to a far away point on the decision boundary. In this example the explanation vector from the local gradient at the prediction point  $x$  has a too large contribution in an irrelevant direction. The closest neighbors of the other class can be found at a very different angle. Thus, the local gradient at the prediction point  $x$  may not be a good explanation for the contributions of single dimensions to the function value  $f(x)$ . Local gradients at the prediction point in the left image and the Taylor root point in the right image are indicated by black arrows. The nearest root point  $x_0$  is shown as a triangle on the decision boundary. The red arrow in the right image visualizes the approximation of  $f(x)$  by Taylor expansion around the nearest root point  $x_0$ . The approximation is given as a vector representing the dimension-wise product between  $Df(x_0)$  (the black arrow in the right panel) and  $x - x_0$  (the dashed red line in the right panel) which is equivalent to the diagonal of the outer product between  $Df(x_0)$  and  $x - x_0$ .**

One technical difficulty is to find a root point  $x_0$ . For continuous classifiers we may use unlabeled test data in a sampling approach and perform a line search between the prediction point  $x$  and a set of candidate points  $\{x'\}$  such that their prediction has opposite sign:  $f(x)f(x') < 0$ . It is clear that the line

$l(a) = ax + (1 - a)x'$  must contain a root of  $f$  which can be found by interval intersection. Thus each candidate point  $x'$  yields one root, and one may select a root point which minimizes the Taylor residuum or use an average over a subset of root points with low Taylor residues. A second possible solution would be, for example, to find the root point  $x_0$  that is the nearest to the data point  $x$ , or optimal in some measurable sense. However, while we have presented two types of solutions to finding a root point, in the most general case, the Taylor-type decomposition is best described as a constraint-based approach. In particular, the root point  $x_0$  at which the Taylor decomposition is computed is constrained to satisfy  $f(x_0) = 0$  and to lie not too far (e.g. within a fixed radius) from the actual data point  $x$ . Using this constraint-based definition, the most desirable properties of the Taylor decomposition are preserved, while the remaining specification is deferred to a later point in time.

Note that Taylor-type decomposition, when applied to one layer or a subset of layers, can be seen as an approximate way of relevance propagation when the function is highly non-linear. This holds in particular when it is applied to the output function  $f$  as a function of the preceding layer  $f = f(z_{i-1})$ , as Equation (18) satisfies approximately the propagation Equation (2) when the relevance of the output layer is initialized as the value of prediction function  $f(x)$ . Unlike the Taylor approximation, layer-wise relevance propagation does not require to use a second point besides the input point. The formulas in Sections *Pixel-wise Decomposition for Classifiers over Bag of Words Features* and *Pixel-wise Decomposition for Multilayer Networks* will demonstrate that layer-wise relevance propagation can be implemented for a wide range of architectures without the need to approximate by means of Taylor expansion.

## Related Work

Several works have been dedicated to the topic of explaining neural networks, kernel-based classifiers in general and classifiers over Bag of Words features in particular.

As for neural networks, [20] is dedicated towards analyzing classifier decisions at neurons which is applicable also to the pixel level. It performs a layer-wise inversion down from output layers towards the input pixels for the architecture of convolutional networks [21]. This work is specific to the architecture of convolutional neural networks. Compared to our approach it is based on a different principle. See [22] which establishes an interpretation of the work in [20] as an approximation to partial derivatives with respect to pixels in the input image. In a high-level sense, the work in [20] uses the method from their own predecessor work in [23] which solves optimization problems in order to reconstruct the image input, while our approach attempts to reconstruct the classifier decision. In a more technical sense, the difference between [20] and our approach can be seen by comparing how the responses are projected down towards the inputs. [20] uses rectified linear units to project information from the unfolded maps towards the inputs with the aim to ensure the feature maps to be non-negative. In our approach we use the signed activations of the neurons from the layer below for weighting the relevance quantity from the layer above with the aim of conserving the relevance quantity across layers. The sign of the activations of neurons of the layer below, for which the relevance is to be computed, is used in our work for encoding assumptions about the structure of the neural network. With respect to neural networks, our work is applicable to a wide range of architectures. A different type of analysis for understanding neural networks deals with creating counter-intuitive examples, see for example the work in [24, 25]. For a treatment on how to discriminate structures which influence predictions from correlations with hidden variables such that the hidden variables have an impact on the predictor, see [26].

Another approach which lies between partial derivatives at the input point  $x$  and a full Taylor series around a different point  $x_0$  is presented in [22]. This work uses a different point  $x_0$  than the input point  $x$  for computing the derivative and a remainder bias which both are not specified further but avoids for an unspecified reason to use the full linear weighting term  $x - x_0$  of a Taylor series. Besides deviating in the usage of a Taylor series, the work in [22] does not make use of the layer-wise propagation strategy for neural networks presented in this paper, which unlike a Taylor series does not rely on a local approximation. Explanation of neural network behavior on the level of single neurons is done in [27] and [28]. These

works try to find inputs which maximize the activation of neurons by means of optimization problems which can be solved by gradient ascent. Our approach aims at explaining the decision for a given input rather than finding optimal stimuli for a particular neuron. Note that the approach presented in our paper is different from plotting the activations of neurons during a forward-pass of the input image. The latter is independent from the neural network properties in higher layers whereas in our approach we feed the classification score into the top neurons and use quantities computed by using properties of higher layers to obtain a representation at lower layers. Quantifying the importance of input variables using a neural network model has also been studied in specific areas such as ecological modeling, where [29, 30] surveyed a large ensemble of possible analyses, including, computing partial derivatives, perturbation analysis, weights analysis, and studying the effect of including and removing variables at training time. A different avenue to understanding decisions in neural network is to fit a more interpretable model (e.g. decision tree) to the function learned by the neural network [31], and extract the rules learned by this new model.

With respect to sensitivity of kernel-based classifiers to input dimensions, [32] yields sensitivity maps, which are invariant to the sign of the predictions. [17–19] consider explanation vectors based on local gradients which are sign-sensitive. Both approaches use partial derivatives at the point which is to be predicted. The first works to our knowledge to visualize BoW models are [33] and [34]. The former work finds regions with a large influence for classification for the special case of max pooling, sparse coding and a linear classifier. The latter work obtains an exact decomposition into local feature scores for the case of a histogram intersection kernel and zero-one coding of local features onto visual words.

We differ from the above works on kernel-based classifiers over Bag of Words features in the following sense: Our methodology is applicable to *arbitrary* Bag of Words models including various feature codings such as Fisher vectors and regularized codings and to a broader class of kernels, namely all differentiable or so-called sum decomposable kernels. When relying on Taylor decomposition, our work relies on a Taylor series around a root close to the prediction point rather than partial derivatives at the prediction point itself. The rationale for this choice was justified in the preceding Section *Taylor-type decomposition*. Finally, in contrast to the preceding publications, our work introduces layer-wise relevance propagation for neural networks and Bag of Words features.

## Pixel-wise Decomposition for Classifiers over Bag of Words Features

Despite recent advances in neural networks, Bag of Words models are still popular for image classification tasks. They have excelled in past competitions on visual concept recognition and ranking such as Pascal VOC [35, 36] and ImageCLEF PhotoAnnotation [37]. In our experience Bag of Words Models perform well for tasks with small sample sizes, whereas neural networks are at risk to overfit due to their richer parameter structure.

### Bag of Words models revisited

We will consider here Bag of Words features as an aggregation of non-linear mappings of local features. All Bag of Words models, no matter whether based on hierarchical clustering [38], soft codebook mapping [15, 39–41], regularized local codings [12–14], or Fisher Vectors [11], share a multi-stage procedure in common.

In the first stage local features are computed across small regions in the image. A local feature such as SIFT [42, 43] is in the abstract sense a vector computed from a region of the image, for example capturing information of interest such as shape characteristics or properties of color or texture on a local scale. In a second stage which is performed once during training, representatives in the space of local features are computed, no matter whether they are cluster centroids obtained from k-means clustering,

regions of the space as for clustering trees [38], or centers of distributions as for Fisher vectors [44]. The set of representatives, in the following referred to as *visual words*, serves as a vocabulary in the context of which images can be described as vectors. In the third stage, statistics of the local features are computed relative to those visual words. These statistics are aggregated from all local features  $l$  within an image in order to yield a BoW representation  $x$ , usually done by sum- or max-pooling.

The computation of statistics can be modeled by a mapping function accepting local feature vectors  $l$  as input, which are then projected into the Bag of Words feature space. Let  $m$  be such a mapping function and let  $m_{(d)}$  denote the mapping onto the  $d$ -th dimension of the BoW space. We assume the very generic p-means mapping scheme for local features  $l$  as given in Equation (19).

$$x_{(d)} = \left( M^{-1} \sum_{j=1}^M (m_{(d)}(l_j))^p \right)^{\frac{1}{p}} \quad (19)$$

This contains sum- and max-pooling as the special cases  $p = 1$  and the limit  $p = \infty$ .

Finally, a classifier is applied on top of these features. Our method supports the general class of classifiers based on kernel methods. For brevity we use here an SVM prediction function which results in a prediction function over BoW features  $x_i$ , training data labels  $y_i$ , kernel functions  $k(\cdot, \cdot)$ , and SVM model parameters  $b$  and  $\alpha_i$ .

$$f(x) = b + \sum_{i=1}^S \alpha_i y_i k(x_i, x) \quad (20)$$

This assumption can be extended without loss of generality to approaches using multiple kernel functions such as multiple kernel learning [45–49], structural prediction approaches with tensor product structure between features and labels as in taxonomy-based classifiers [50–52] or boosting-like formulations as in [53]

$$f(x) = b + \sum_{i=1}^S \sum_{u=1}^K \alpha_{i,u} k_u(x_{i(u)}, x_{(u)}). \quad (21)$$

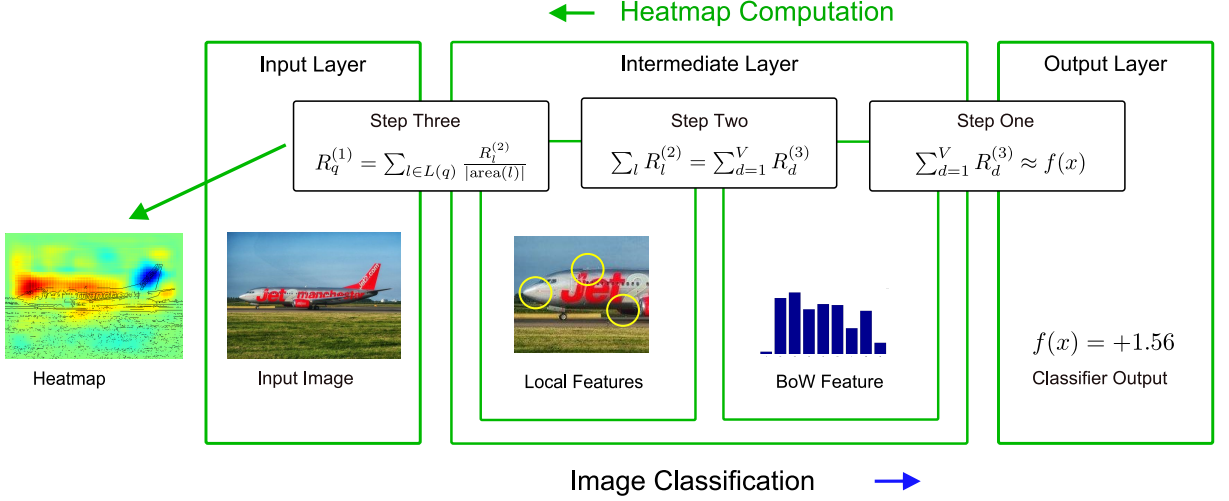
An overview over notations used for the Bag of Words features is given in Table 1.

**Table 1.** Notation Conventions Used in This Section

$f(\cdot)$	the classifier’s prediction function
$x, y$	BoW representation and class label
$x_0, z$	root point of Taylor Expansion, root point candidates
$\alpha, b$	learned model parameters
$k(x_i, x_{i'})$	kernel function
$d, V$	counter and number of BoW-dimensions
$R_d^{(3)}$	(approximate) contribution of BoW-dimension $d$
$R_l^{(2)}$	local feature relevance
$R_q^{(1)}$	pixel-wise decompositions per pixel $q$
$m(l)$	mapping function between local features $l$ and BoW
$l, l'$	local feature descriptors
$Z(x)$	the set of unmapped dimensions of a BoW data point $x$
$\text{area}(l)$	the set of pixel coordinates covered by $l$

## Overview of the decomposition steps

The main contribution of this part is the formulation of a generic framework for retracing the origins of a decision made by the learned kernel-based classifier function for a BoW feature. This is achieved, in a broad sense as visualized in Fig. 4, by following the construction of a BoW representation  $x$  of an image and the evaluation thereof by a classifier function in reverse direction. In this section we will derive a decomposition of a kernel-based classifier prediction into contributions of individual local features and finally single pixels. The proposed approach consists of three consecutive steps.



**Fig. 4. Local and global predictions for input images are obtained by following a series of steps through the classification- and pixel-wise decomposition pipelines.** Each step taken towards the final pixel-wise decomposition has a complementing analogue within the Bag of Words classification pipeline. The calculations used during the pixel-wise decomposition process make use of information extracted by those corresponding analogues. Airplane image in the graphic by Pixabay user tpsdave.

In the first step we will use, depending on the type of kernel, either the Taylor-type decomposition strategy or the layer-wise relevance propagation strategy. In the first step relevance scores  $R_d^{(3)}$  for the third layer of the BoW feature extraction process are obtained, describing the influence of all BoW feature dimensions  $d$  by deconstructing the classifier prediction function  $f(x)$  such that  $\sum_d R_d^{(3)} \approx f(x)$ . In other words, we gain a decomposition into contributions  $R_d^{(3)}$  describing  $f(x)$  as a sum of individual predictions for all dimensions of  $x$ .

In the second step we will apply the layer-wise relevance propagation strategy in order to obtain relevance scores  $R_l^{(2)}$  for the local features  $l$  from the relevance scores  $R_d^{(3)}$ . Layer-wise relevance propagation ensures that  $\sum_l R_l^{(2)} = \sum_d R_d^{(3)} \approx f(x)$  holds.

The third step describes the computation of pixel-wise scores  $R_q^{(1)}$  from local feature relevance scores  $R_l^{(2)}$ , which are then visualized as heatmaps by color-coding.



**Step one: relevance scores  $R_d^{(3)}$  for the third layer of the BoW feature extraction process**

The third layer is the BoW feature itself. In the first step we would like to achieve a decomposition of the classifier prediction  $f(x)$  into relevance scores  $R_d^{(3)}$  for BoW feature dimension  $d$ .

$$f(x) \approx \sum_{d=1}^V R_d^{(3)} \quad (22)$$

The work of [34] has performed this step for the special case of one single histogram intersection kernel. Such a decomposition can be generalized naturally and performed without error for all kernel functions which are sum-decomposable along input dimensions. We define a kernel function  $k$  to be sum-decomposable if there exists kernel functions  $k^{(d)}$  acting on single input feature dimensions such that

$$k(x_i, x_{i'}) = \sum_{(d)} k^{(d)}(x_{i(d)}, x_{i'(d)}) \quad (23)$$

In this case, as with the linear and histogram intersection kernel, we can achieve Formula (22) with equality in the following definition by applying layer-wise relevance propagation which results in Equation (24).

**Def. 1 Relevance scores for sum decomposable kernels**

---

$$R_d^{(3)} = \frac{b}{V} + \sum_{i=1}^S \alpha_i y_i k^{(d)}(x_{i(d)}, x_{(d)}) \quad (24)$$

For the case of a general differentiable kernel we apply the Taylor-type decomposition strategy in order to linearly approximate the dimensional contributions  $R_d^{(3)}$ . The approximated dimensional contributions can be expressed as in Equation (25).

**Def. 2 Relevance scores for differentiable kernels**

---

$$R_d^{(3)} := (x - x_0)_{(d)} \sum_{i=1}^S \alpha_i y_i \frac{\partial k(x_i, \cdot)}{\partial x_{(d)}}(x_0) \quad (25)$$

**Step two: relevance scores  $R_l^{(2)}$  for the second layer of the BoW feature extraction process**

The second layer are the local features extracted from many regions of the image. In the second step we would like to achieve a decomposition of the classifier prediction  $f(x)$  into relevance scores  $R_l^{(2)}$  for the local features  $l$  based on the relevances  $R_d^{(3)}$  from the third layer.

For the sake of clarity, we do for now start with the case of sum-pooled BoW aggregation, to later extend to a more general formulation for p-means pooling from this point on.

As introduced in context of Equation (19)  $m_{(d)}$  denotes the mapping projecting to dimension  $d$  of the BoW space. We define the set of input dimensions  $Z(x)$  which are effectively not reached by the mappings of local features of an image. Applying the layer-wise propagation allows to define the local feature relevance score  $R_l^{(2)}$  as given in Equation (27).

---

**Def. 3 Local feature scores for sum pooling**


---

$$Z(x) = \left\{ d \mid \sum_l m_{(d)}(l) = 0 \right\} \quad (26)$$

$$R_l^{(2)} := \sum_{d \notin Z(x)} R_d^{(3)} \frac{m_{(d)}(l)}{\sum_{l'} m_{(d)}(l')} + \sum_{d \in Z(x)} R_d^{(3)} \frac{1}{|\{l'\}|} \quad (27)$$


---

The coarse structure of definition (27) can be explained as taking the relevances from the layer above and weighting them with the outputs from the layer below. The summations over the mappings  $m_{(d)}(l)$  over the local features  $\{l\}$  in Equation (27) achieve a weighting of the relevance from the third layer proportional to the ratios of the mappings. The second part describes an equal distribution of those relevance scores  $R_d^{(3)}$  which correspond to dimensions of the Bag of Words feature  $x$  which have a value of zero and yet may contribute to the classifier prediction. To see this consider computing the  $\chi^2$ -kernel value to a support vector which has a non-zero value in the same feature dimension. This is necessary to ensure that the propagation Equation (2) holds.

Summing the local feature relevance scores  $R_l^{(2)}$  from Equation (27) yields the Taylor approximation  $R_d^{(3)}$  of the prediction score  $f(x)$  in Equations (24) or (25). This property is the key to our approach. We obtain exact summation to the prediction  $f(x)$  in the case of sum decomposable kernels and usage of Equation (24) in this special case.

$$\sum_l R_l^{(2)} = \sum_{d=1}^V R_d^{(3)} \approx f(x) \quad (28)$$

We would like to point out that this property holds also in the case when mappings  $m_{(d)}$  can become negative as a consequence of the definition used in (26) and (27), as can be seen from the summation given in the appendix. For that reason our approach is also applicable to Fisher vectors [11] and regularized coding approaches [12–14]. Furthermore note that definition (27) has no explicit dependence on the way how the local features are pooled in Equation (19) and this might be inappropriate weighting for max-pooling or general p-means pooling.

We can extend this definition to reflect the usage of p-means pooling

$$M_p(x_1, \dots, x_n) = \left( \frac{1}{n} \sum_{i=1}^n x_i^p \right)^{1/p} \quad (29)$$

which may yield different results in prediction and local decomposition than sum pooling. The extension is well-defined for non-negative mappings  $m_{(d)} \geq 0$  and any value of  $p$  and for arbitrary mappings when combined with a value of  $p$  from the natural numbers.

---

**Def. 4 Local feature scores for p-means pooling**


---

$$Z^{(p)}(x) = \left\{ d \mid \sum_l m_{(d)}^p(l) = 0 \right\} \quad (30)$$

$$R_l^{(2)} := \sum_{d \notin Z^{(p)}(x)} R_d^{(3)} \frac{m_{(d)}^p(l)}{\sum_{l'} m_{(d)}^p(l')} + \sum_{d \in Z^{(p)}(x)} R_d^{(3)} \frac{1}{|\{l'\}|} \quad (31)$$


---

The first quotient in Equation (31) converges to an indicator function for the maximal mapping element in the limit  $p \rightarrow \infty$  which is consistent to max-pooling

$$\frac{m_{(d)}^p(l)}{\sum_{l'} m_{(d)}^p(l')} \rightarrow \mathbb{I}_{\{\arg\max_{l'} m_{(d)}(l')\}}(l) \quad (32)$$

**Step three: relevance scores  $R_q^{(1)}$  for the first layer of the BoW feature extraction process**

The first layer are the pixels of the image. In order to calculate scores for each pixel we make use of information regarding local feature geometry and location known from the local feature extraction phase at the beginning of the image classification pipeline. The pixel score  $R_q^{(1)}$  of an image coordinate  $q$  is calculated as a sum of local feature scores of all local features  $l$  covering  $q$ , weighted by the number of pixels covered by each local feature  $l$ . In terms of layer-wise relevance propagation a local feature is a computation unit which has as much inputs as the number of pixels it is covering. Without assumption of any further structure we distribute the relevance of the local feature equally to all its covered pixels. Layer-wise propagation yields Equation (34).

$$L(q) = \{l \mid q \in \text{area}(l)\} \quad (33)$$

$$R_q^{(1)} = \sum_{l \in L(q)} \frac{R_l^{(2)}}{|\text{area}(l)|} \quad (34)$$

The operator  $\text{area}(l)$  used Equations (33) and (34) returns the set of pixel coordinates covered by a local feature  $l$ . Projecting the obtained scores  $R_q^{(1)}$  to their respective image coordinates yields the pixel-wise decomposition representation  $h$  of the evaluated image.

For visualization in the sense of color coding, the pixel-wise decomposition  $R^{(1)}$  is then normalized as

$$R_q'^{(1)} = \frac{R_q^{(1)}}{\max_{q'} (|R_q'^{(1)}|)} \quad (35)$$

ensuring that  $\forall q : R_q'^{(1)} \in [-1, 1]$ . The normalized pixel-wise decomposition is then color coded by mapping the pixel scores to a color space of choice. In the case of individually classified sub-regions originating from the same image, a global pixel-wise decomposition can be constructed by averaging local pixel-wise decompositions scores.

Note that by choosing above normalization scheme, the assumption is made that at least one class is represented within the image. In case the assumption holds this might lead to prominent local predictions of even weakly projected features which we found suitable for the purpose of detecting class evidence. If this assumption does not hold, then images may display score artifacts dominated by the set of pixels covered by a small subset of local features which would otherwise be considered input noise. A solution for this problem is global normalization which uses a maximum over pixels over a set of images instead of one image. We found that a global normalization scheme can be more appropriate for visualizing the actual decision process of the classifier, as it preserves the relative order of magnitude of local feature scores in between pixel-wise decomposition tiles. Algorithm 1 gives an overview how to compute the pixel-wise decomposition for classifiers based on Bag of Words features and support vector machines.

---

**Algorithm 1** Pixel-wise decomposition for BoW features with SVM classifiers
 

---

**Inputs:**Image  $I$ Local features  $L$ BoW representation  $x$  (and Taylor root point  $x_0$ )

model and mapping parameters

**for**  $d = 1$  **to**  $V$  **do** $R_d^{(3)}$  as in Equations (24) or (25)**end for****for all**  $l \in L$  **do** $R_l^{(2)}$  as in Equations (27) or (31)**end for****for all** pixels  $q \in I$  **do** $R_q^{(1)}$  as in Equation (34)**end for****Output:**  $\forall q : R_q^{(1)}$ **Examples for various mappings and kernels**

In order to illustrate the generality of this framework we give some examples for various methods of mapping local features and kernels.

**Example case: Soft codebook mapping**

A soft codebook mapping like in [15, 40] fits trivially into the framework

$$m_{(d)}(l) = \frac{\exp(-\beta d(l, b_d))}{\sum_{(d)} \exp(-\beta d(l, b_d))} \quad (36)$$

**Example case: Regularized coding**

Considering formulations as in [12, 14]

$$\operatorname{argmin}_{B, C} \sum_j \|l_j - Bc_j\|^2 + Q(c_j) \quad (37)$$

where  $B$  is the codebook and  $Q$  denotes a regularizer on the codebook coefficients such as the  $\ell_1$ -norm  $Q(c) = \sum_j |c(j)|$ , we arrive at

$$m(l_j) := c_j \quad (38)$$

where the  $d$ -th dimension of the so defined mapping  $m_{(d)}$  corresponds to the  $d$ -th basis element in  $B$ .

**Example case: Fisher Vectors**

The Fisher vector [44] can be replicated using the following mapping  $m_{(d)}$  given below. It is expected to be used with a linear kernel, thus we can use the exact formula from Equation (24) for defining the BoW dimension relevance scores  $R_d^{(3)}$ . We stick to the formulation of Fisher vectors as given in [11]. The difference to other mappings is that a BoW feature dimension corresponds to a derivative of a Gaussian mixture center with respect to some parameter and not to a visual word.

Let  $g_k(l)$  be the  $D$ -dimensional Gaussian mixture component with diagonal covariance  $\sigma_k$  and mean  $\mu_k$ . We define the softmax of the GMM mixture parameters  $\alpha_k$  as

$$w_k = \frac{\exp(\alpha_k)}{\sum_{j=1}^K \exp(\alpha_j)} \quad (39)$$

Let the soft assignment of local feature  $l$  to Gaussian  $k$  be given as

$$\gamma[l](k) = \frac{w_k u_k(l)}{\sum_{j=1}^K w_j u_j(l)} \quad (40)$$

Let  $k$  denote the index of a Gaussian mixture component, then the mapping contains three cases depending on whether we consider the derivative for the mixture parameter  $\alpha_k$  stored in dimension  $d = (2D + 1)(k - 1) + 1$ , derivatives for the Gaussian mean parameter  $\mu_k$  stored in dimensions  $d = (2D + 1)(k - 1) + 1 + r, r \in [1, D]$ , or finally, derivatives for the Gaussian variance parameter  $\sigma_k$  stored in dimensions  $d = (2D + 1)(k - 1) + 1 + D + r, r \in [1, D]$

We assume one kernel without loss of generality,  $V_1 = (2D + 1)K$  where  $K$  is the number of Gaussian mixture components and  $D$  is the local feature dimension.

$$m_{(d)}(l) = \begin{cases} \frac{1}{\sqrt{w_k}}(\gamma[l](k) - w_k) & \text{if } d = (2D + 1)(k - 1) + 1 \\ \frac{1}{\sqrt{w_k}}\gamma[l](k)\frac{l_r - \mu_k}{\sigma_k} & \text{if } d = (2D + 1)(k - 1) + 1 + r, r \in [1, D] \\ \frac{1}{\sqrt{w_k}}\gamma[l](k)\frac{1}{\sqrt{2}}\left(\frac{(l_r - \mu_k)^2}{\sigma_k^2} - 1\right) & \text{if } d = (2D + 1)(k - 1) + 1 + D + r, r \in [1, D] \end{cases} \quad (41)$$

This definition can be extended in a straightforward manner to use power normalization and  $\ell_2$  normalization from [11]. The difference to the soft mapping case from Section *Example case: Soft codebook mapping* is that a the  $d$ -th dimension of the mapping  $m_{(d)}$  corresponds not to a visual word but to a derivative with respect to a Gaussian mixture component.

#### Example case: Histogram intersection kernel

The histogram intersection kernel applies to the exact decomposition formula in Equation (24). It is defined as

$$k_{\text{HI}}(x_i, x_{i'}) = \sum_d \min(x_{i(d)}, x_{i'(d)}) \quad (42)$$

Plugging the kernel into (24) yields

$$R_d^{(3)} = \frac{b}{V} + \sum_{i=1}^S \alpha_i y_i \min(x_{i(d)}, x_{(d)}) \quad (43)$$

#### Example case: $\chi^2$ kernel

The  $\chi^2$  kernel function has been applied with good performance in various image classification tasks [41, 54, 55]. The  $\chi^2$  kernel is defined as

$$k_{\chi^2}(x_i, x_{i'}) = \exp\left(-\sigma \sum_d \frac{(x_{i(d)} - x_{i'(d)})^2}{(x_{i(d)} + x_{i'(d)})}\right) \quad (44)$$

To avoid divisions by zero in case of  $x_{i(d)} = x_{i'(d)} = 0$  we adhere to the convention that  $\frac{0}{0} = 0$  for the kernel function itself, as well as its derivative

$$\frac{\partial k_{\chi^2}(x_i, x_{i'})}{\partial x_{i'(d)}} = k_{\chi^2}(x_i, x_{i'}) \sigma \left( \frac{4 (x_{i(d)})^2}{(x_{i(d)} + x_{i'(d)})^2} - 1 \right) \quad (45)$$

When plugged into Equation (25), the dimensional contributions of the  $\chi^2$  kernel are then obtained as

$$R_d^{(3)} := (x - x_0)_{(d)} \cdot \sum_{i=1}^S \alpha_i y_i k_{\chi^2}(x_i, x_0) \sigma \left( \frac{4 (x_{i(d)})^2}{(x_{i(d)} + x_{0(d)})^2} - 1 \right) \quad (46)$$

### Example case: Gaussian-RBF kernel

The Gaussian-RBF kernel function is widely used in different communities and is one of the most prominent, if not the most prominent non-linear kernel function. The kernel function is defined as

$$k_{\text{Gauss}}(x_i, x_{i'}) = \exp \left( -\frac{\|x_i - x_{i'}\|_2^2}{\sigma} \right) \quad (47)$$

Deriving the kernel function w.r.t  $x_{i'(d)}$  and plugging it into Equation (25) yields

$$\frac{\partial k_{\text{Gauss}}(x_i, x_{i'})}{\partial x_{i'(d)}} = k_{\text{Gauss}}(x_i, x_{i'}) \left( \frac{2x_{i(d)} - 2x_{i'(d)}}{\sigma} \right) \quad (48)$$

and consequently

$$R_d^{(3)} := (x - x_0)_{(d)} \cdot \sum_{i=1}^S \alpha_i y_i k_{\text{Gauss}}(x_i, x_0) \left( \frac{2x_{i(d)} - 2x_{0(d)}}{\sigma} \right) \quad (49)$$

## Pixel-wise Decomposition for Multilayer Networks

Multilayer networks are commonly built as a set of interconnected neurons organized in a layer-wise manner. They define a mathematical function when combined to each other, that maps the first layer neurons (input) to the last layer neurons (output). We denote each neuron by  $x_i$  where  $i$  is an index for the neuron. By convention, we associate different indices for each layer of the network. We denote by “ $\sum_i$ ” the summation over all neurons of a given layer, and by “ $\sum_j$ ” the summation over all neurons of another layer. We denote by  $x_{(d)}$  the neurons corresponding to the pixel activations (i.e. with which we would like to obtain a decomposition of the classification decision). A common mapping from one layer to the next one consists of a linear projection followed by a non-linear function:

$$z_{ij} = x_i w_{ij}, \quad (50)$$

$$z_j = \sum_i z_{ij} + b_j, \quad (51)$$

$$x_j = g(z_j), \quad (52)$$

where  $w_{ij}$  is a weight connecting the neuron  $x_i$  to neuron  $x_j$ ,  $b_j$  is a bias term, and  $g$  is a non-linear activation function. Multilayer networks stack several of these layers, each of them, composed of a large number of neurons. Common non-linear functions are the hyperbolic tangent  $g(t) = \tanh(t)$  or the rectification function  $g(t) = \max(0, t)$ . This formulation of neural network is general enough to encompass a wide range of architectures such as the simple multilayer perceptron [56] or convolutional neural networks [57], where convolution and sum-pooling are linear operations.

## Taylor-type decomposition

Denoting by  $f: \mathbb{R}^M \mapsto \mathbb{R}^N$  the vector-valued multivariate function implementing the mapping between input and output of the network, a first possible explanation of the classification decision  $x \mapsto f(x)$  can be obtained by Taylor expansion at a near root point  $x_0$  of the decision function  $f$ :

$$R_d^{(1)} = (x - x_0)_{(d)} \cdot \frac{\partial f}{\partial x_{(d)}}(x_0) \quad (53)$$

The derivative  $\partial f(x)/\partial x_{(d)}$  required for pixel-wise decomposition can be computed efficiently by reusing the network topology using the backpropagation algorithm [56]. In particular, having backpropagated the derivatives up to a certain layer  $j$ , we can compute the derivative of the previous layer  $i$  using the chain rule:

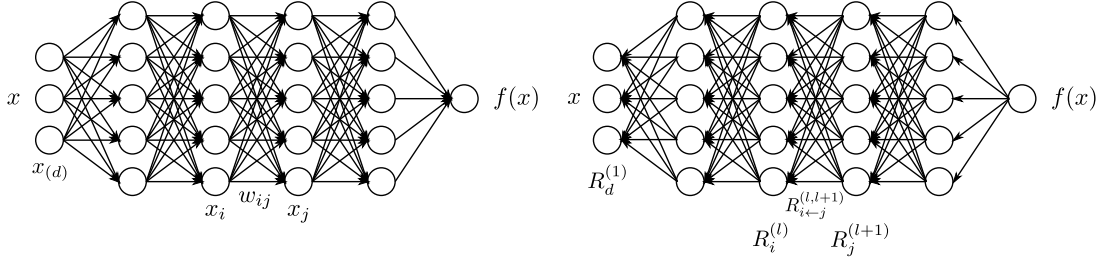
$$\frac{\partial f}{\partial x_i} = \sum_j \frac{\partial f}{\partial x_j} \cdot \frac{\partial x_j}{\partial x_i} = \sum_j \frac{\partial f}{\partial x_j} \cdot w_{ij} \cdot g'(z_j). \quad (54)$$

A requirement of the Taylor-based decomposition is to find roots  $x_0$  (i.e. points on the classification boundary) that support a local explanation of the classification decision for  $x$ . These roots can be found by local search in the neighborhood of  $x$ . However, as noted in [24], this can lead to points of the input space that are perceptually equivalent to the original sample  $x$  and whose choice as a root would produce non-informative pixel-wise decompositions.

Alternatively, root points can be found by line search on the segment defined by  $x$  and its closest neighbor of a different class. This solution is problematic when the data manifold is sparsely populated, as it is the case for natural images. In such case, it is likely that following a straight line between  $x$  and its nearest neighbor will strongly depart from the data manifold and produce roots  $x_0$  with similarly poor pixel-wise decompositions.

## Layer-wise relevance backpropagation

As an alternative to Taylor-type decomposition, it is possible to compute relevances at each layer in a backward pass, that is, express relevances  $R_i^{(l)}$  as a function of upper-layer relevances  $R_j^{(l+1)}$ , and backpropagating relevances until we reach the input (pixels). Fig. 5 depicts a graphic example.



**Fig. 5. Multilayer neural network annotated with the different variables and indices describing neurons and weight connections. Left: forward pass. Right: backward pass.**

The method works as follows: Knowing the relevance of a certain neuron  $R_j^{(l+1)}$  for the classification decision  $f(x)$ , one would like to obtain a decomposition of such relevance in terms of messages sent to neurons of the previous layers. We call these messages  $R_{i←j}$ . In particular, as expressed by Equations (8) and (13), the conservation property

$$\sum_i R_{i←j}^{(l,l+1)} = R_j^{(l+1)} \quad (55)$$

must hold. In the case of a linear network  $f(x) = \sum_i z_{ij}$  where the relevance  $R_j = f(x)$ , such decomposition is immediately given by  $R_{i \leftarrow j} = z_{ij}$ . However, in the general case, the neuron activation  $x_j$  is a non-linear function of  $z_j$ . Nevertheless, for the hyperbolic tangent and the rectifying function – two simple monotonically increasing functions satisfying  $g(0) = 0$ , – the pre-activations  $z_{ij}$  still provide a sensible way to measure the relative contribution of each neuron  $x_i$  to  $R_j$ . A first possible choice of relevance decomposition is based on the ratio of local and global pre-activations and is given by:

$$R_{i \leftarrow j}^{(l, l+1)} = \frac{z_{ij}}{z_j} \cdot R_j^{(l+1)} \quad (56)$$

These relevances  $R_{i \leftarrow j}$  are easily shown to approximate the conservation properties of Equation (2), in particular:

$$\sum_i R_{i \leftarrow j}^{(l, l+1)} = R_j^{(l+1)} \cdot \left(1 - \frac{b_j}{z_j}\right) \quad (57)$$

where the multiplier accounts for the relevance that is absorbed (or injected) by the bias term. If necessary, the residual bias relevance can be redistributed onto each neuron  $x_i$ .

A drawback of the propagation rule of Equation (56) is that for small values  $z_j$ , relevances  $R_{i \leftarrow j}$  can take unbounded values. Unboundedness can be overcome by introducing a predefined stabilizer  $\varepsilon \geq 0$ :

$$R_{i \leftarrow j}^{(l, l+1)} = \begin{cases} \frac{z_{ij}}{z_j + \varepsilon} \cdot R_j^{(l+1)} & z_j \geq 0 \\ \frac{z_{ij}}{z_j - \varepsilon} \cdot R_j^{(l+1)} & z_j < 0 \end{cases} \quad (58)$$

The conservation law then becomes

$$\sum_i R_{i \leftarrow j}^{(l, l+1)} = \begin{cases} R_j^{(l+1)} \cdot \left(1 - \frac{b_j + \varepsilon}{z_j + \varepsilon}\right) & z_j \geq 0 \\ R_j^{(l+1)} \cdot \left(1 - \frac{b_j - \varepsilon}{z_j - \varepsilon}\right) & z_j < 0 \end{cases} \quad (59)$$

where we can observe that some further relevance is absorbed by the stabilizer. In particular, relevance is fully absorbed if the stabilizer  $\varepsilon$  becomes very large.

An alternative stabilizing method that does not leak relevance consists of treating negative and positive pre-activations separately. Let  $z_j^+ = \sum_i z_{ij}^+ + b_j^+$  and  $z_j^- = \sum_i z_{ij}^- + b_j^-$  where “−” and “+” denote the negative and positive part of  $z_{ij}$  and  $b_j$ . Relevance propagation is now defined as

$$R_{i \leftarrow j}^{(l, l+1)} = R_j^{(l+1)} \cdot \left( \alpha \cdot \frac{z_{ij}^+}{z_j^+} + \beta \cdot \frac{z_{ij}^-}{z_j^-} \right) \quad (60)$$

where  $\alpha + \beta = 1$ . For example, for  $\alpha, \beta = 1/2$ , the conservation law becomes:

$$\sum_i R_{i \leftarrow j}^{(l, l+1)} = R_j^{(l+1)} \cdot \left( 1 - \frac{b_j^+}{2z_j^+} - \frac{b_j^-}{2z_j^-} \right) \quad (61)$$

which has similar form to Equation (57). This alternate propagation method also allows to control manually the importance of positive and negative evidence, by choosing different factors  $\alpha$  and  $\beta$ .

Once a rule for relevance propagation has been selected, the overall relevance of each neuron in the lower layer is determined by summing up the relevance coming from all upper-layer neurons in consistence with Equations (8) and (13):

$$R_i^{(l)} = \sum_j R_{i \leftarrow j}^{(l, l+1)} \quad (62)$$



The relevance is backpropagated from one layer to another until it reaches the input pixels  $x_{(d)}$ , and where relevances  $R_d^{(1)}$  provide the desired pixel-wise decomposition of the decision  $f(x)$ . The complete layer-wise relevance propagation procedure for neural networks is summarized in Algorithm 2.

---

**Algorithm 2** Pixel-wise decomposition for neural networks

---

**Input:**  $R^{(L)} = f(x)$   
**for**  $l \in \{L - 1, \dots, 1\}$  **do**  
     $R_{i \leftarrow j}^{(l, l+1)}$  as in Equations (58) or (60)  
     $R_i^{(l)} = \sum_j R_{i \leftarrow j}^{(l, l+1)}$   
**end for**  
**Output:**  $\forall d : R_d^{(1)}$

---

Above formulas (58) and (60) are directly applicable to layers which satisfy a certain structure. Suppose we have a neuron activation  $x_j$  from one layer which is modeled as a function of inputs from activations  $x_i$  from the preceding layer. Then layer-wise relevance propagation is directly applicable if there exists a function  $g_j$  and functions  $h_{ij}$  such that

$$x_j = g_j \left( \sum_i h_{ij}(x_i) \right) \quad (63)$$

In such a general case, the weighting terms  $z_{ij} = x_i w_{ij}$  from Equation (50) have to be replaced accordingly by a function of  $h_{ij}(x_i)$ . A key observation is that relevance propagation is invariant against the choice of function  $g_j$  for computing relevances for the inputs  $x_i$  *conditioned on keeping the value of relevance  $R_j$  for  $x_j$  fixed*. This observation allows to deal with non-linear activation functions  $g_j$ . The function  $g_j$  does however exert influence on computing the relevance  $R_i$  by its influence on the relevance  $R_j$  for  $x_j$ . This can be explained by the fact, that the choice of  $g_j$  determines the value of  $x_j$  and thus also relevance  $R_j$  for  $x_j$  which gets assigned by the weights in the layer above.

We remark again, that even max pooling fits into this structure as a limit of generalized means, see Equation (32) for example. For structures with a higher degree of non-linearity, such as local renormalization [58, 59], Taylor approximation applied to neuron activation  $x_j$  can be used again to achieve an approximation for the structure as given in Equation (63).

Finally, it can be seen from the formulas established in this section that layer-wise relevance propagation is different from a Taylor series or partial derivatives. Unlike Taylor series, it does not require a second point other than the input image. Layer-wise application of the Taylor series can be interpreted as a generic way to achieve an approximate version of layer-wise relevance propagation. Similarly, in contrast to any methods relying on derivatives, differentiability or smoothness properties of neuron activations are not a necessary requirement for being able to define formulas which satisfy layer-wise relevance propagation. In that sense it is a more general principle.

## Experiments

For Bag of Words features we show two experiments, one on an artificial but easily interpretable data set and one for on Pascal VOC images which have a high compositional complexity. For the artificial data set we apply the Taylor-type strategy for the top layer, for Pascal VOC images we apply the strategy for sum-decomposable kernels for the top layer. In both cases these strategies are combined with our definitions for arbitrary mappings for the lower layers.

For neural networks we show results also on two data sets, two sets of results on MNIST which are easy to interpret, and a second set of experiments in which we rely on a 15 layers already trained network

provided as part of the Caffe open source package [60], which predicts the 1000 categories from the ILSVRC challenge. On one side, by the experiments on MNIST digits, we intend to show that we can uncover details specific to the training phase. On the other side, the results for the pre-trained network from the Caffe toolbox demonstrate, that the method works with a deep neural network out of the box and does not rely on possible tricks during the training phase.

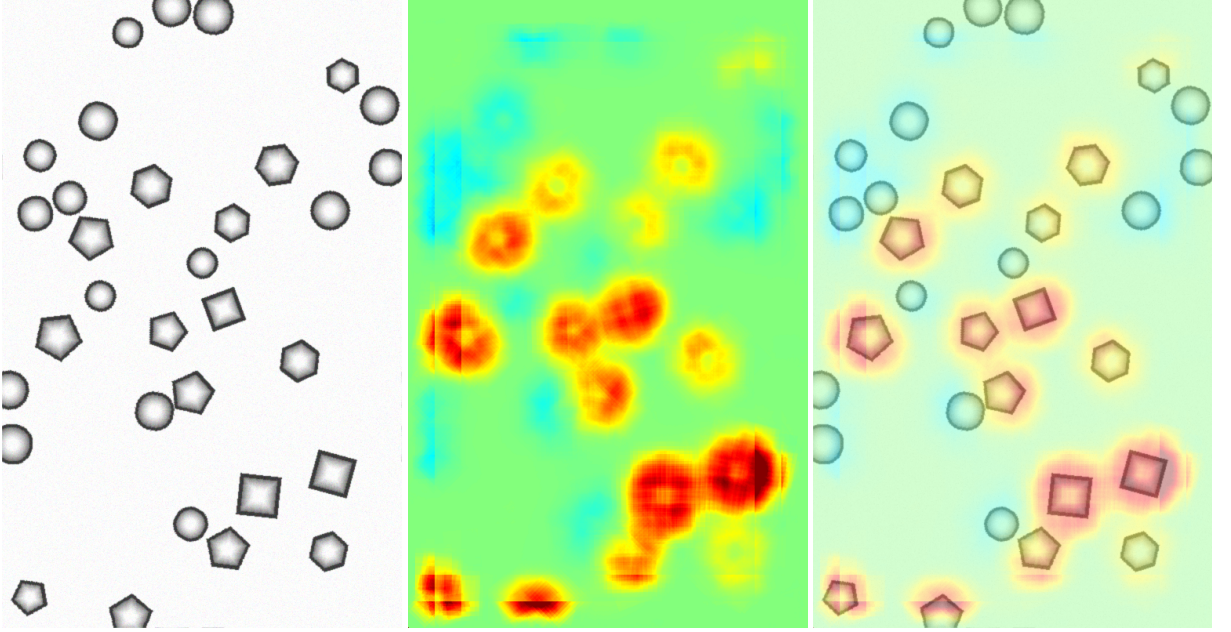
### Bag of Words features for polygons versus circles

An example of a pixel-wise decomposition for synthetic data is given in Fig. 6. The training images were image tiles of size  $102 \times 102$  pixels. An image was labeled positive if it contained at least one polygon independent of the presence of circles, and labeled negative if it contained no shape or circles only. The BoW features have been computed over standard SIFT features on gray scale brightness values on a dense grid with scales 2.0 and 3.0 resulting in one BoW feature for each scale. The SIFT feature masks have been rotated relatively to the main pixel gradient direction at their respective anchor locations. One  $\chi^2$ -kernel as in Equation (44) was used for each BoW feature in conjunction with multiple kernel learning. Therefore the Taylor-type decomposition was used for computing the third layer relevances  $R_d^{(3)}$  by Equation (46). The 128-dimensional BoW features are computed using a sum-pooled rank-mapping paradigm

$$x_{(d)} = M^{-1} \sum_{j=1}^M m_{(d)}(l_j)$$

$$m_{(d)}(l) = \begin{cases} p^{-\text{rk}_d(l)} & ; \text{rk}_d(l) \leq n \\ 0 & ; \text{else} \end{cases} \quad (64)$$

followed by an  $\ell_1$ -normalization of  $x$ . Similar as in [41] we set  $p = 2$  and  $n = 4$ . The expression  $\text{rk}_d(l)$  in Formula (64) describes the rank of the BoW prototype representing dimension  $d$  in the BoW feature space among ascendingly ordered Euclidean distances between  $l$  and all BoW prototypes. Thus Formula (27) based on layer-wise relevance propagation was used to obtain predictions  $R_l^{(2)}$ . In order to compute the visualization in Fig. 6, the test picture was divided into tiles of sizes  $102 \times 102$  with a grid step of 34 pixels. For each image tile one pixel-wise decomposition was computed. The overlapping pixel-wise decompositions were averaged, the resulting heatmap was normalized to the range  $[-1, +1]$  by dividing it by the maximal absolute pixel value.



**Fig. 6. Pixel-wise decomposition for Bag of Words features over  $\chi^2$ -kernels using the Taylor-type decomposition for the third layer and the layer-wise relevance propagation for the subsequent layers.** Left: The original image. Middle: Pixel-wise prediction. Right:

Superposition of the original image and the pixel-wise prediction. The decompositions were computed on tiles of size  $102 \times 102$  and having a regular offset of 34 pixels. The decompositions from the overlapping tiles were averaged. In the heatmap, based on linearly mapping the interval  $[-1, +1]$  to the jet color map available in many visualization packages, green corresponds to scores close to zero, yellow and red to positive scores and blue color to negative scores. See text for interpretation.

The averaged classifier decomposition in Fig. 6 shows firstly that the pixel-wise decomposition achieved by combining both decomposition strategies yields plausible results. The classifier making predictions on tiles was trained to discriminate polygons versus circles. The heatmap visualization of the pixel-wise decomposition is consistent with that result: Pixels close to polygons achieve positive pixel-wise scores, visible by the yellow to red colors. Pixels close to circles are mapped in light blue colors corresponding to negative pixel-wise scores. Pixels far away from shapes are in green which encodes scores close to zero.

This observation implies that local features over polygons are provided on average with positive second layer scores  $R_l^{(2)} \gg 0$  and local features over circles receive on average weakly negative second layer scores  $R_l^{(2)} < 0$ . Furthermore regions which are far from shapes result in scores close to zero which are colored in green in the heatmap from pixel-wise decomposition of the predictions and in the overlay of the image and the decomposition. We remark that the shapes have arbitrary positions in the image tiles of the training set and that position information was never used during training or BoW feature extractions. Thus, the pixel-wise decomposition is able to detect structures on a smaller scale than the classifier was trained on although position information was never provided during training or used in the classification algorithm which maps an image tile to a real-valued classifier prediction. The resolution of the smaller scale is limited by the scale of the SIFT features used here, the scale 3.0 results in a SIFT feature with side length of  $4 \cdot 3 \cdot 3 = 36$  pixels. This explains also why pixel scores outside of a shape but nearby to a shape receive the color which corresponds to the label induced by the neighboring shape.

Finally we remark that the rank-mapping is a discontinuous weighting scheme for BoW feature dimensions, yet the layer-wise propagation yields reasonable explanations.

## Bag of Words features for the Pascal VOC2009 data set

We have calculated pixel-wise predictions for images from the evaluation set of the Pascal VOC 2009 image classification challenge. The BoW representations of the training and test part of the data set have been computed over whole images based on local features extracted from a dense regular grid and fixed rotation. Standard SIFT features and stacks of 9-dimensional quantile features measuring values from 0.1 to 0.9 of the data over color intensities as in [41] have been used at local feature scale 3.0. The quantile features are split into two regions describing halves of a circle, yielding one 9-dimensional quantile over color intensities each, which are then concatenated. The SIFT feature descriptors have been computed once over a gray scale version of the image, stacked with feature descriptors computed over a combination of opposing color channels  $\frac{\text{red}+\text{green}-2\cdot\text{blue}+2}{4}$ , resulting in a  $128 + 128 = 256$  dimensional local feature vector. For SIFT as well as the color intensity quantiles, the same procedure has been applied to the red, green and blue color channels of the images separately, resulting in  $128 \cdot 3 = 384$  and  $(2 \cdot 9) \cdot 3 = 54$  dimensions for the local features respectively. SIFT and quantile features were mapped onto 1024 dimensional and 512 dimensional BoW spaces respectively, using the ranked mapping scheme in Formula (64) with  $p = 2.4$  and  $n = 8$ . Classifiers were trained for detecting the presence of the target object category regardless of the presence of other classes, using multiple kernel learning on three histogram intersection kernels, one for each computed local feature configuration. Consequently, the layer-wise relevance propagation Formulas (24) and (27) were used to compute local predictions for each evaluation image as a whole. Pixel-wise predictions for the classes representing buses, persons, airplanes and cats are presented in Figs. 7, 8, 9 and 10 respectively. An interpretation which has been drawn from the depicted examples and other examples which are omitted here is given in the figure captions.



**Fig. 7. Pixel-wise decomposition for Bag of Words features over a histogram intersection kernel using the layer-wise relevance propagation for all subsequent layers and rank-mapping for mapping local features.** Each triplet of images shows – from left to right – the original image, the pixel-wise predictions superimposed with prominent edges from the input image and the original image superimposed with binarized pixel-wise predictions. The decompositions were computed on the whole image. Images twice by Pixabay users tpsdave, and by Pixabay users sirocumo and Pixeleye.

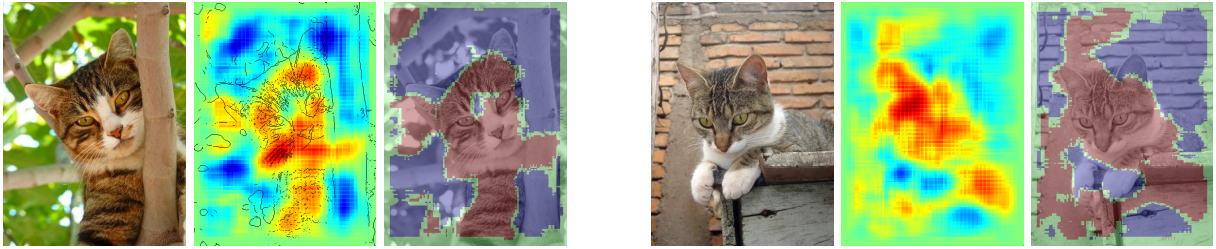


**Fig. 8. Pixel-wise decomposition for Bag of Words features over a histogram intersection kernel using the layer-wise relevance propagation for all subsequent layers and rank-mapping for mapping local features.** Each triplet of images shows – from left to right – the original image, the pixel-wise predictions superimposed with prominent edges from the input image and the original image superimposed with binarized pixel-wise predictions. The decompositions were computed on the whole image. Faces below the hairline but also hands yield high scores, see the woman in the third picture which turns away her face from the camera as an example that hair alone is not relevant. Images by Pelagio Palagi, Wikimedia users Rorschach, Frankie Fouganthin and Flickr user Le vent dans les dunes.



**Fig. 9. Pixel-wise decomposition for Bag of Words features over a histogram intersection kernel using the layer-wise relevance propagation for all subsequent layers and rank-mapping for mapping local features.** Each triplet of images shows – from left to right – the original image, the pixel-wise predictions superimposed with prominent edges from the input image and the original image superimposed with binarized pixel-wise predictions. The decompositions were computed on the whole image. Notably the tail of a plane receives negative scores consistently. Blue sky context seems to contribute to classification which has been conjectured already in the PASCAL VOC workshops [35] and which was observed also on other images not shown here, see the the second picture for comparison against the other three images which have more blueish sky. Images from Pixabay users Holgi, nguyentuanhung, rhodes8043 and tpsdave.





**Fig. 10. Pixel-wise decomposition for Bag of Words features over a histogram intersection kernel using the layer-wise relevance propagation for all subsequent layers and rank-mapping for mapping local features.** Each triplet of images shows – from left to right – the original image, the pixel-wise predictions superimposed with prominent edges from the input image and the original image superimposed with binarized pixel-wise predictions. The decompositions were computed on the whole image. Positive responses seem to exist for certain fur texture patterns, see also the false responses on the wood and the plaster in the second example which both have similar texture and color to a cat’s fur. Images by Pixabay users LoggaWiggler and Holcan.

## Neural Networks for MNIST digits

We would like to investigate the capacity of relevance propagation to find evidence for classification of MNIST handwritten digits. A particular advantage of this data set over many-class image data sets is that it is easy for humans to interpret both positive and negative evidence, because of the small number of classes. For example, evidence for the handwritten digit “1” comes from the presence of a vertical bar on the pixel grid, but also from the absence of horizontal bar starting from the top of the vertical bar, which would make it a “7”. Also, the data set is relatively simple and the relation between the training algorithm and the resulting pixel-wise relevances can be analyzed.

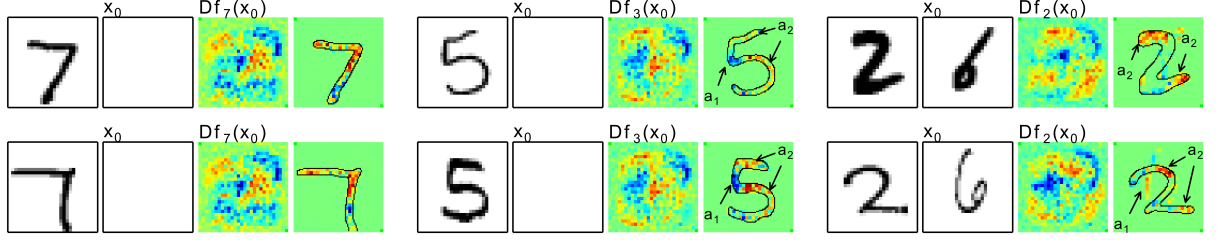
We perform three experiments for MNIST data, as we would like to demonstrate that the method is able to uncover properties specific to the way of training. One experiment is done with a smaller network which is trained without translations of digits for the sake of allowing a direct comparison of the pixel-wise decomposition results to class-densities for each pixel of digits and seeing the impact of artifacts in the training set. Two further experiments are done on a larger network which has been trained without artifacts and with translated versions of digits and more training iterations for the sake of a better response to digits. The latter two experiments intend to show the impact of non-digit pixels as positive and negative evidence for a class of digits.

### MNIST experiments I

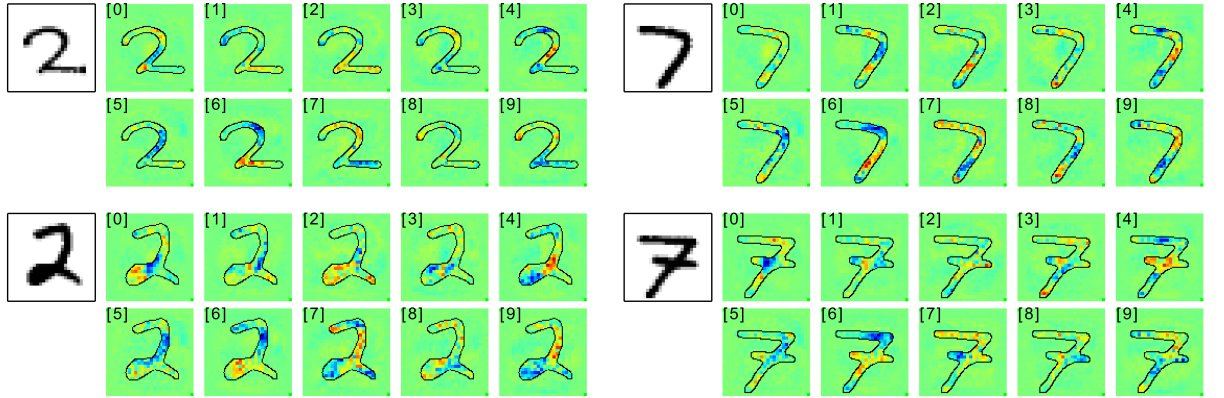
The first set of experiments is done on a fully-connected neural network trained in the most common way: Input data is normalized so that the sum of pixels is on average zero, and the variance of pixel values is on average one. This setting implies that only black pixels yield strong inputs whereas white pixels fire only due to mean subtraction. The absence of translation invariance during training allows to uncover correlations of the pixel-wise decomposition to pixel-wise training densities and, as we will see in the experiments, allow to uncover artifacts in the data that may harm generalization.

Examples for pixel-wise decompositions for the first type of neural networks are given in Figs. 11, 12 and 13. Multilayer neural networks were trained on the MNIST [57] data of handwritten digits and solve the posed ten-class problem with a prediction accuracy of 98.25% on the MNIST test set. Our network consists of three linear sum-pooling layers with a bias-inputs, followed by an activation or normalization step each. The first linear layer accepts the  $28 \times 28$  pixel large images as a 784 dimensional input vector and produces a 400-dimensional tanh-activated output vector. The second layer projects those 400 inputs

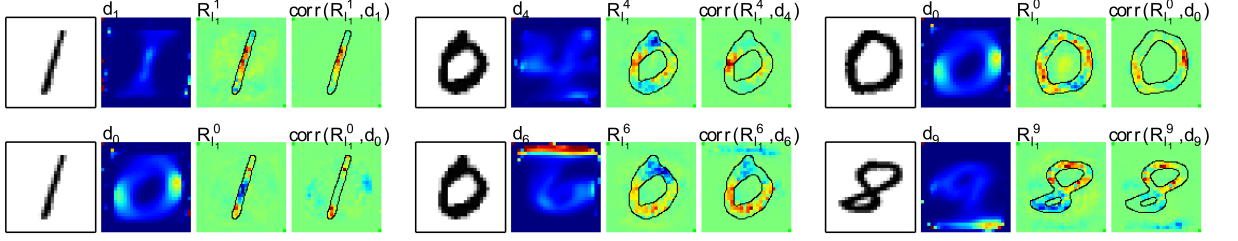
to equally many tanh-activated outputs. The last layer then transforms the 400-dimensional space to a 10-dimensional output space followed by a softmax layer for activation in order to produce output probabilities for each class. The network was trained using a standard error back-propagation algorithm using batches of 25 randomly drawn training samples with an added Gaussian noise layer per training iteration. The above prediction accuracy was achieved after terminating the training procedure after 50 000 iterations.



**Fig. 11. Taylor-approximated pixel-wise predictions for a multilayer neural network trained and tested on the MNIST data set.** Each group of four horizontally aligned panels shows – from left to right – the input digit, the Taylor root point  $x_0$ , the gradient of the prediction function  $f$  at  $x_0$  of a specific digit class indicated by the subscript next to  $f$  and the approximated pixel-wise contributions for  $x$ .



**Fig. 12. Pixel-wise decompositions for a multilayer neural network trained and tested on MNIST digits, using layer-wise relevance propagation as in Formula (56).** Each group shows the decomposition of the prediction for the classifier of a specific digit indicated in parentheses.



**Fig. 13.** Each quadruple shows: on the leftmost the input digit; on the middle left the class specific pixel-wise density ratios  $d_k$  (Equation (65)) for the digit class  $k$  for which the pixel-wise decomposition is computed; on the middle right the pixel-wise decomposition  $R^{(1)}$  for that digit and the digit class  $k$ ; on the rightmost the correlation between  $d_k$  and the pixel-wise decomposition  $R^{(1)}$ . When considering a digit from class  $i$  and a pixel-wise decomposition from class  $k \neq i$ , it is observable that the pixel-wise decomposition shows frequently highly positive activations on pixels of the digit from class  $i$  which have high relative density  $d_k$  for the digit class  $k \neq i$ .

Fig. 11 shows pixel-wise relevances resulting from Taylor-type decomposition as defined in Equations (18) and (53). The left column of the figure shows approximated pixel-wise relevances of the input digits, both times a 7, relative to a blank pixel tile. Relevances are calculated for the classifier output producing the prediction score for the class of digits 7. Evidently the upper part of the digit and most prominently the horizontal bar at the top are important characteristics for the predictor speaking for the digit's class. The middle column of Fig. 11 shows pixel-wise relevances for inputs from digit class 5 relative to a blank tile as interpreted by the classifier output reserved for the class of digit 3. As for digit class 7, the gradient at the root point  $x_0$  shows high positive prediction weights in areas corresponding to typical characteristics of the target class. We observe strong positive responses in areas shared by both digit classes 5 and 3 (indicated by annotation  $a_2$ ) and negative pixel-wise responses where the upper left vertical bar of the input digits is located, which is usually not present in a digit 3 ( $a_1$ ). The rightmost column of Fig. 11 indicates distinguishing parts of the input digits from class 2 relative to their counter parts from digit class 6 chosen by minimal euclidean distance. High pixel scores highlight characteristics of the prediction point indicating membership in class 2, most notable the upper arc and the tail of the input digits 2, as marked with the annotation  $a_2$ . Notably in the lower right image of the figure the absence of the leftmost vertical arc of the digit 6 (annotation  $a_1$ ) in the prediction point causes positive local predictions for class 2.

Pixel-wise predictions obtained via the layer-wise relevance propagation Formula (56) were calculated based on the output of the last linear layer without taking the succeeding softmax normalization layer into account. We can see pixel-wise decompositions for exemplary digits in Fig. 12. Notably, for the pixel-wise decomposition of the digit 2, we see highly positive responses in lower parts of the digits for the classifiers for digits 2 and 6, however for the classifier for digit 6, the prediction score is negative, because the high positive responses in the lower part are suppressed by high negative responses in the upper part where the digit 2 has an arc which is not present in the digit 6.

Correlating the pixel-wise decompositions for classifier for digit  $k$  with the relative density of the pixels of the digits  $k$  in the training set demonstrates the plausibility of the decomposition.

$$d_k(p) = \frac{\sum_{N \in \text{digits}(k)} N(p)}{\sum_{l=0}^9 \sum_{N \in \text{digits}(l)} N(p)} \quad (65)$$

This can be seen in Fig. 13. When considering the decomposition for a classifier for digit  $k$  and using a digit  $i \neq k$  from the wrong class as input, then we observe often that the decomposition shows high positive activations on those pixels of digit  $i$  which have high relative density  $d_k(p)$  (Equation (65))



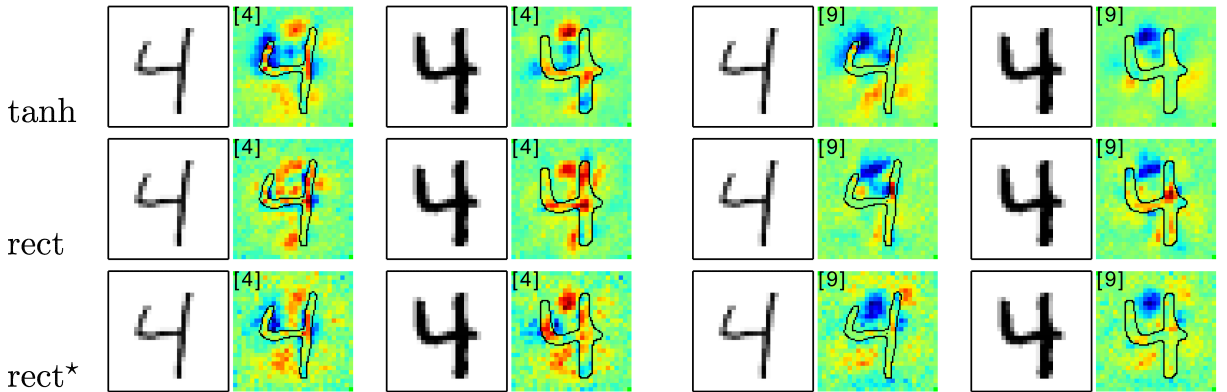
within the training set for digits of type  $k$ . Even when we use the classifier for the wrong class, it uncovers evidence on those pixels which overlap regions with high density of positively labeled digits of its training set.

## MNIST experiments II

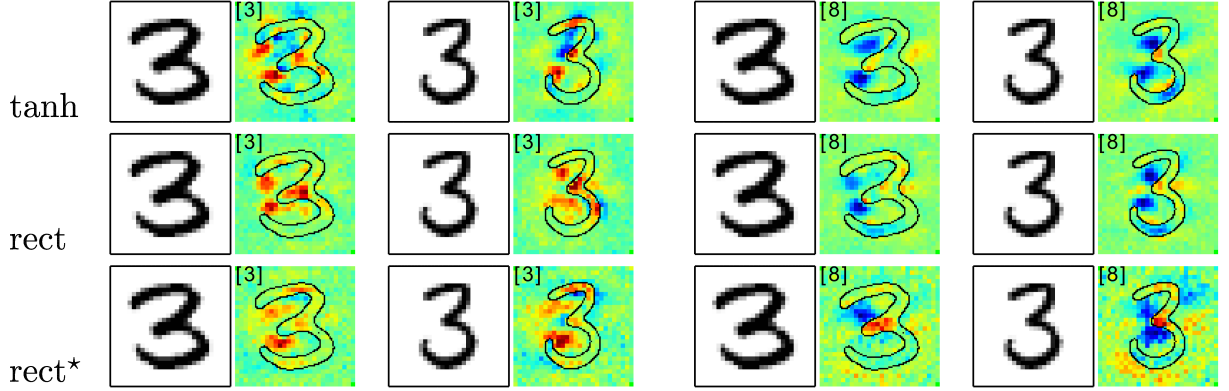
In this set of experiments, we train larger neural networks on the MNIST data set augmented translated copies of the digits. Neural networks are composed of three hidden layers of 1296 units each, where weight connections between layers are initialized at random. Neural networks are trained by backpropagation using stochastic gradient descent with mini batches of size 25 and using a softmax objective [6]. In order to force the neural network to make decisions based on the shape of digits rather than their absolute position on the pixel grid, we augment the training data with pixel-wise translations of  $\pm 4$  pixels (both vertical and horizontal) and add 10% salt-and-pepper noise to the inputs. Also, in order to favor the construction of evidence on both black and white pixels, MNIST digits are re-scaled to take pixel values between  $-1$  (white pixel) and  $1$  (black pixel).

We consider two types of non-linearities: (1) rectified linear units and (2) hyperbolic tangent sigmoids. These non-linearities are some of the most commonly used in neural networks and are plotted in Fig. 16. Also, in order to study explanations on different level of detail, we consider (1) a network that has been trained for 1 000 000 iterations until it reaches 99.2% accuracy and (2) a network trained for only 10 000 iterations, at which point it reaches approximately 97.0% accuracy.

Fig. 14 and 15 are two case studies of explanations produced by the neural networks we have trained. The number in brackets on the top-left corner of the pixel-wise decomposition indicates the class with respect to which evidence is measured. Similar explanations are produced by a network trained with rectified linear units. It is interesting to note that layer-wise relevance propagation, which is not making use of the gradient information – and therefore agnostic to the type of neurons in the network, – still produces similar results across different types of neurons. Explanations learned by the rectifying network are arguably more global, because the rectified linear units are prevented from saturating on both sides and therefore more linear. Explanations produced by a rectifying network trained for a shorter time (rect\*) are clearly more global, as evidenced by the positive and negative parts of the pixel-wise decompositions spreading over larger pixel areas.



**Fig. 14. Evidence for a handwritten digit being a “4” or a “9”.** Strong positive evidence for “4” is allocated to the top part of the image for keeping it blank. If trying to interpret these digits as “9”, the open top-part of the image is perceived as negative evidence for this class, because a “9” would rather have a top-dash closing the upper loop of the “4”. Explanations are consistent across a variety of neural networks and samples.



**Fig. 15. Evidence for a handwritten digit being a “3” or a “8”.** Classifying as “3” is supported by the middle horizontal stroke featured in this digit and the absence of vertical connections on the left of the image. Evidence for being a “8” feature again the middle horizontal stroke, however, the absence of connections on the left side of the digit constitutes negative evidence. Explanations are again stable for various models and samples.

In order to compute the pixel-wise decompositions, we used Equation (56), which does not use numerical stabilizers. Use of numerical stabilizers causes relevance to leak from one layer to another, in particular, when the relevance of a particular handwritten digit is measured for another class. Fig. 27 shows examples of pixel-wise decompositions for 16 randomly drawn digits sorted according to the digit class.

### **Pixel flipping experiments using the rect-long model from Section *MNIST experiments II***

In this section we intend to make a semi-quantitative analysis of the pixel-wise decompositions. The basic idea is to compute a decomposition of a digit for a digit class and then flip pixels with highly positive, highly negative scores or pixels with scores close to zero and then to evaluate the impact of these flips onto the prediction scores. The advantage of demonstrating this on MNIST data is that a vast majority of pixels either has very high (black) or very low (white) values, with very few pixels having values in between. For results on photographic scenes one may need to resort to building masks which are specific to each category, for example, a gray box may be good for masking a flower, but it will not mask effectively a gray road or fur of a Koala bear.

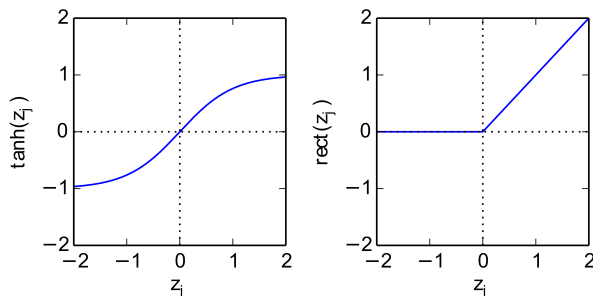
In general: If we *flip* a pixel, we do  $flipped = pixel \cdot (-1)$ . In further experiments we will consider the digits three and four as inputs. The test set contains 983 fours and 1011 threes, over which the mean prediction is calculated.

One apparent result from the preceding section is that we can observe non-digit pixels with highly positive pixel scores. In Fig. 17 we evaluate the impact of flipping non-digit pixels with highly positive pixel scores on the resulting classifier prediction. We observe in Fig. 17 that the classifier prediction for the true digit class, namely three and four, decreases fast when the non-digit pixels with high pixel scores are flipped. This shows, firstly, that having non-digit pixels with high positive pixel scores makes sense for a correct classification. We see also, that flipping the highest scoring pixels for a digit three turns it into a digit eight, which is consistent to the decompositions seen in Fig. 15.

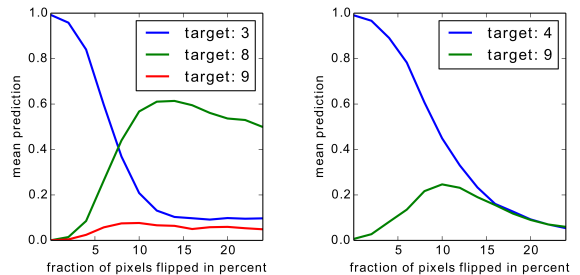
Secondly, it shows that measuring the quality of a pixel-wise decomposition by an object segmentation mask is not always a good idea. When seeing the digit as an object, having non-object pixels with high scores can make sense in the case of geometric constraints for objects which are to be recognized, as we have in our case. For this reason we deliberately did not choose object segmentation masks for the digits

as a basis for evaluation in the sense that digit-pixels should have highly positive scores, and non-digit pixels should have zero or negative scores.

For the same reason, namely the possibility of the presence of geometric constraints, pixel-wise decomposition is not always a good weak segmentation in contrast to the convincing results in the experiments of [22] on images from the ILSVRC data set. Another reason for a possible divergence between pixel-wise decomposition and segmentation is the possible influence of context in a scene.



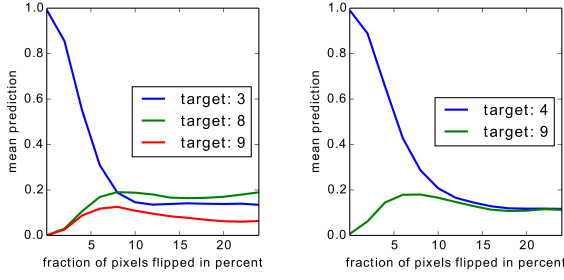
**Fig. 16.** Example of non-linear activation functions  $g$  used in multilayer neural networks.



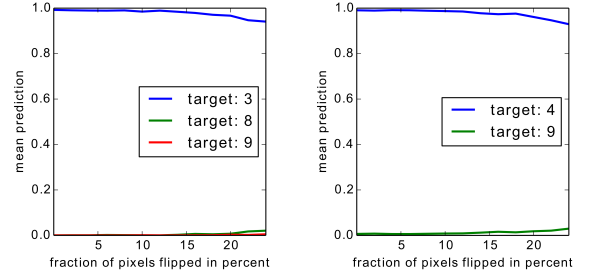
**Fig. 17.** Flipping of high-scoring non-digit pixels. Pixels with highest positive scores are flipped first. The pixel-wise decomposition was computed for the true digit class, three (left) and four (right).

Once we have established the reason why we do not use segmentation masks for evaluating the quality of pixel-wise prediction we can observe the effects of flipping the highest scoring pixels, independently of whether they are a digit or non-digit pixel. We can see from Fig. 18 that the prediction of the correct digit class decreases sharply in this case as well. This decrease can be compared to the case when we flip those pixels whose absolute value of the decomposition score  $s$  is closest to zero by sorting the pixels according to  $1 - |s|$ . We see in Fig. 19 that the classifier prediction for the true class decreases only very slowly, thus neutrally scored pixels are less relevant for the prediction. The comparison between Fig. 18 and 19 shows that the decomposition score is able to identify those pixels which play little role in the classification of a digit and those pixels which are important for identifying a digit.

Thus, the pixel-wise decomposition is not only intuitively appealing to a human but also makes sense for the representation used in classifier to make its decision. Using digits for demonstrating such a statement has a mild bias towards our method because for a geometric-driven task like digits we can expect that firstly the problem can be learned well by a classifier so that the resulting pixel-wise decompositions are very informative, and secondly what has been learned on digits might be more similar between humans and algorithms compared to complex natural scene recognition tasks. See, however, the experiments in the Section *Neural Network for 1000 ILSVRC classes* on the categories from ILSVRC challenge for results on an object recognition tasks on photographic images which also yield results plausible to a human. In general one can expect that a classifier with poor recognition ability will yield also pixel-wise decompositions with less informative scores.

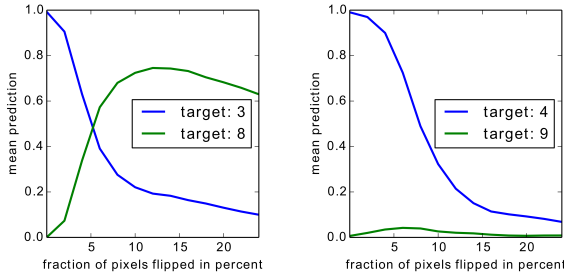


**Fig. 18. Flipping of digit and non-digit pixels with positive responses.** Pixels with highest positive scores are flipped first. The pixel-wise decomposition was computed for the true digit classes three (left) and four (right).

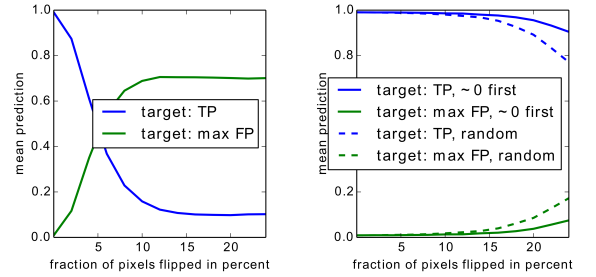


**Fig. 19. Flipping of pixels with pixel-wise decomposition score close to zero.** Pixels with absolute value closest to zero are flipped first. Digit and non-digit pixels may be flipped. Pixel-wise decomposition have been computed for the true digit classes three (left) and four (right).

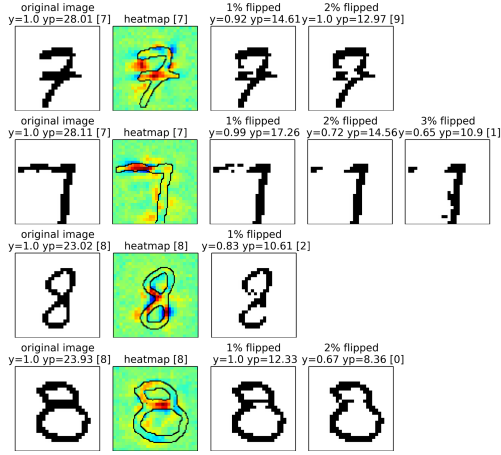
Finally, we evaluate the influence of pixels with negative scores. For this, we take a digit, compute the pixel-wise decomposition for a wrong class, then flip the pixels which are marked negatively when trying to predict the wrong class. Results are shown in Fig. 20. We observe for both example cases that the prediction for the true class declines sharply as we tweak the digits towards the wrong class. For digit 9 as target and digits 4 as inputs this flipping results in a noisy blob on top of a stick, such that the result is not cleanly predictable as a nine. However one can observe still a moderate rise of the prediction of the 4 as a 9. For digit 8 as target and digits 3 as inputs we can see, that for some intermediate percentage of flips, the 3 will be identified as an 8. In summary, negatively scored pixels are able to represent what is contradictory to classifying a digit to belong to a particular class.



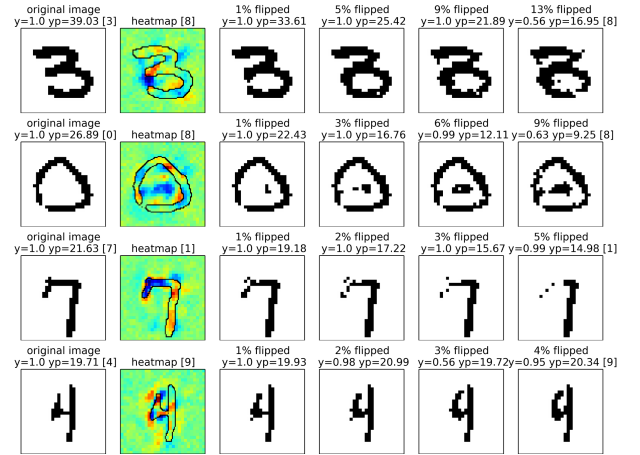
**Fig. 20. Flipping of pixels with negative responses,** due to a pixel-wise decomposition for prediction targets 8 (for digits 3 on the left) and 9 (for digits 4 on the right). Pixels with lowest negative scores are flipped first.



**Fig. 21. Flipping of pixels for digit and non-digit pixels, compared for each modified digit for the true class against the maximal prediction of all wrong classes.** Left: Pixels with highest positive scores are flipped first. Right: Flipping of neutrally predicted pixels, i.e. pixels with absolute value closest to zero are flipped first (solid lines), and flipping of randomly picked pixels (dashed lines). Results are averaged over digits from all digit classes in contrast to using only digit classes 3 and 4 in the preceding figures.



**Fig. 22. Examples of images with an increasing amount of flipped pixels and the corresponding predictions of the classifier.** Here pixels are flipped away from the class label given in parentheses above the heatmap. Pixels were flipped in steps of 1% of all pixels until the predicted class label changed. The plots show the output of the softmax function  $y$  and the output score of the preceding linear layer  $yp$ . The pixels were sorted before flipping in *decreasing* order of the pixel-wise score, i.e. highest scoring pixels were flipped first. In this panel the heatmap was computed for the classifier which produced the highest score, i.e. for the predicted class label. The originally predicted label is given on the leftmost image in parentheses, the predicted label after the switch of the prediction is given in the rightmost image.



**Fig. 23. Examples of images with an increasing amount of flipped pixels and the corresponding predictions of the classifier.** Here pixels are flipped towards the class label given in parentheses above the heatmap. Pixels were flipped in steps of 1% of all pixels until the predicted class label changed. The plots show the output of the softmax function  $y$  and the output score of the preceding linear layer  $yp$ . The pixels were sorted before flipping in *increasing* order of the pixel-wise score, i.e. lowest scoring pixels were flipped first. In this panel the heatmap was computed for a classifier which did not produce the highest score, i.e. for a random false class label. The originally predicted label is given on the leftmost image in parentheses, the predicted label after the switch of the prediction is given in the rightmost image.

The results in Figs. 17, 18 and 19 are shown for the true class against one or more particular wrong classes. The results hold also when we show the behavior of the classifier predictions under flipping of the highest scoring and the most zero-like pixels when we compared for each digit against the maximum of predictions over all wrong classes. Furthermore we compute the average over all 10 digits, not only digits 3 and 4 as true classes as shown in Figs. 17, 18 and 19. This result is shown in Fig. 21. On average over all digits, flipping the highest scoring pixels at first results in a fast decline of the prediction for the true class, and at some point another class is predicted. Flipping the pixels at first with scores close to zero results in a much slower decline of the prediction for the true class. This result demonstrates a quantifiable plausibility of the pixel-wise decomposition by layer-wise relevance prediction. In order to visualize the process of pixel flipping, Figs. 22 and 23 provide examples of a digit, its heatmap and the resulting images with increasing amounts of flipped pixels. For each of the resulting images the softmax output  $y$  and the linear layer output  $yp$  are shown. The first difference between Fig. 22 and 23 is that in the former the pixels were flipped according to the heatmap for the highest scoring class, and in the latter the pixels were flipped according to the heatmap for a random class which did not yield the highest score. The second difference is that in Fig. 22 the highest scoring pixels were flipped first, which implies that evidence for

the predicted class label is removed. As a consequence, the original prediction is gradually lost without giving a specific false class label as target. The digit in Fig. 22 changes towards that class label which the classifier and the heatmapping implicitly perceive as the nearest neighbor. With respect to the second difference, in Fig. 23 the lowest scoring pixels for the random false class label were flipped first, which implies that evidence against the false class label is removed. As a consequence, the digit is modified towards the specific false class label which is denoted in parentheses above the heatmap. In Fig. 22 each of the digits 7 and 8 results into different class labels after flipping. We can see that destroying evidence for a given predicted class may result into differing final class labels. The nearest neighbor of a digit is not constant within the digit class, but depends on the style of the digit itself. Furthermore we can see that pixel-flipping sometimes removes and sometimes adds black pixels, depending on where the highest pixel scores are located. In Fig. 23 we can see results of flipping towards a specified false class label. In both cases the resulting digits are plausible to the human observer. Note also small details, e.g. the 4 in Fig. 23 is rounded at its lower left end during flipping, while the classifier regards the 7 in Fig. 23 to be too short for being a 1 as seen on the blue spot above the vertical bar of the digit. Finally, Fig. 27 provides examples of the pixel-wise decomposition for 16 randomly drawn digits and the predictors for all 10 digit classes.

## Neural Network for 1000 ILSVRC classes

We use here the pre-trained neural network which is provided by the Caffe open source package [60]. Fig. 24 shows pixel-wise decompositions for example images from categories of the ILSVRC data set. Results are shown on images from Wikimedia Commons rather than the ILSVRC images in order to avoid license problems when showing images. Unsurprisingly the evidence is not as sharp as for the MNIST data set because the underlying classification problem is much more diverse with a higher complexity of classes and images to be processed. We can see for the rooster high scores at his cockscomb and parts of his feathers. The black widow spider has high evidence in her head section which is surprising because it is not so prominent to humans but is consistent among other images of black widow spiders as well. Further high evidence lies in parts of black widow legs, and, as expected, the characteristic red spots of a black widow. For the tabby cat we see evidence in his fur but also in the road surface under the cat which has similar color and texture to a cat's fur. The espresso cup has high responses on the cup's edges, the grip, some evidence in the espresso liquid, however only little response for the edges on the accompanying sweets which has the wrong color, gray, for an espresso. Note that a good classifier for a concept does not necessarily imply that an object can be localized very well by its pixel-wise decomposition. Since the classifier processes an image globally, it can perceive evidence away from the actual object, for example by taking into account typical background of objects, the context of an object or global statistics which are correlated to an object. In that sense, for example, bottles can serve as evidence for a dining table or sky can be a clue for photos with airplanes on the ground or photos of outside views of churches.

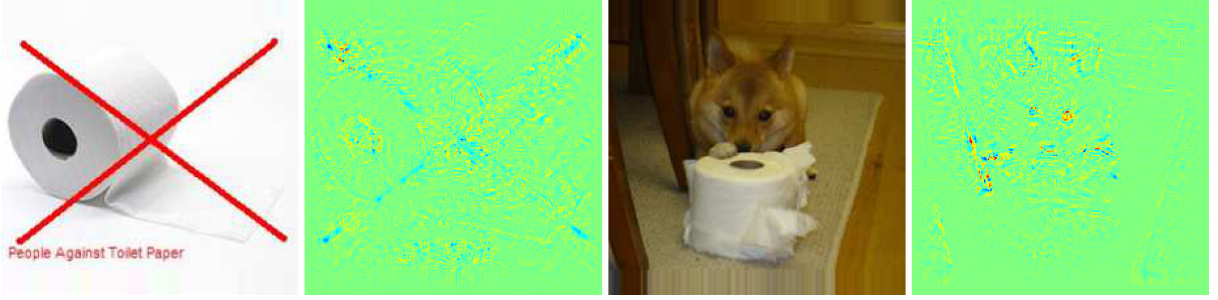
Fig. 25 shows examples where the pixel-wise decomposition does not yield discriminative results. For the toilet paper, positive prediction scores were already low in both examples. This corresponds to our expectation that a less discriminative classifier results also in less convincing pixel-wise decompositions.

Comparing the different shown methods in Figs. 24 and 26, one can see that the method (58) with a small value of the stabilizer  $\epsilon$  produces results with the finest granularity and also with the highest amount of negative evidence. When increasing the values of the stabilizer  $\epsilon$ , the method becomes less granular and shows less negative evidence. This result was also observed when comparing other small and large values of the stabilizer  $\epsilon$ . The decrease of negative pixel-wise scores and of granularity can be both explained under some assumptions. Firstly, note that compared to equation (58) with stabilizer equal to 0 the Equation (58) with stabilizer  $\epsilon$  differs by a multiplicative factor of  $\frac{|z_j|}{|z_j| + \epsilon}$  which goes to zero when  $|z_j|$  goes to zero. Thus dampening by this factor increases when the absolute value of inputs  $|z_j|$  decreases. Secondly note, that in equation (58) the sign of the relevance  $R_j$  of input neuron  $j$  will





**Fig. 24.** The pixel-wise decompositions for examples images of the neural net pre-trained on ILSVRC data set images and provided by the Caffe open source package [60]. Second column shows decompositions computed by Formula (58) with stabilizers  $\varepsilon = 0.01$ , the third column with stabilizers  $\varepsilon = 100$ , the fourth column was computed by Formula (60) using  $\alpha = +2, \beta = -1$ . The artifacts at the edges of the images are caused by filling the image with locally constant values which comes from the requirement to input square sub-parts of images into the neural net. Pictures in order of appearance from Wikimedia Commons by authors Jens Nietschmann, Shenrich91, Sandstein, Jörg Hempel.

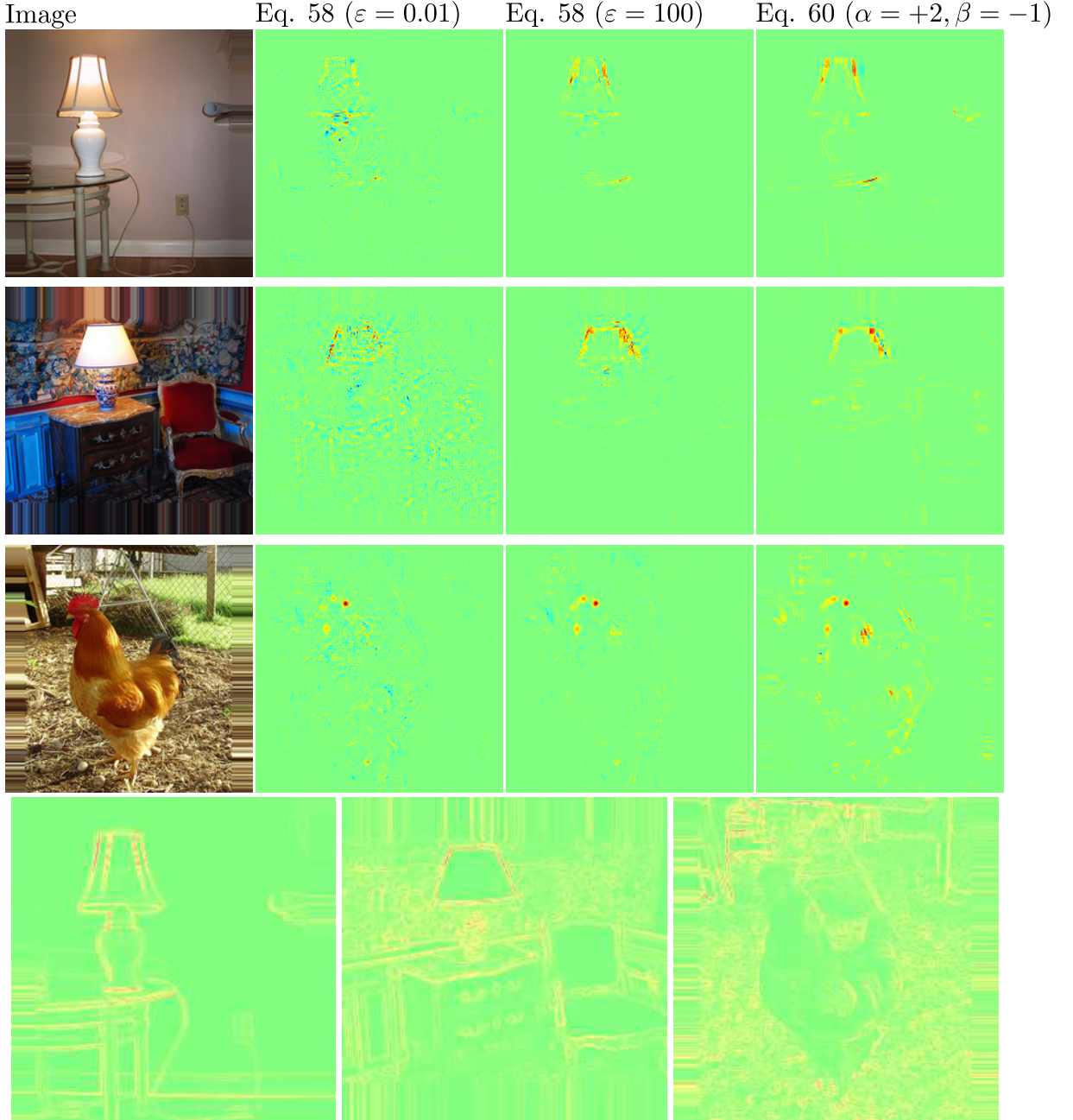


**Fig. 25. Failure examples for the pixel-wise decomposition.** Left and Right: Failures to recognize toilet paper. The decompositions computed by Formula (58) with stabilizers  $\varepsilon = 0.01$ . The neural net is the pre-trained one on ILSVRC data from the Caffe package [60]. The computing methods for each column are the same as in Fig. 24. Pictures in order of appearance from Wikimedia Commons by authors Robinhood of the Burger World and Taro the Shiba Inu.

be flipped in the messages  $R_{i \leftarrow j}^{(l, l+1)}$  for either the positive or the negative inputs  $z_{ij}$  (depending on the sign of  $\frac{z_{ij}}{z_j}$ ). When we assume that the neurons at layer  $l + 1$  with small absolute value of inputs  $|z_j|$  are responsible for generating a high proportion of all negative messages  $R_{i \leftarrow j}^{(l, l+1)} < 0$ , then the reduction of negative pixel-wise scores by increasing stabilizer  $\epsilon$  can be explained. Such an assumption can be reasonable because a small absolute value of inputs can sometimes be explained by the sum of positive inputs being close to the sum negative inputs, and noting that either the positive or the negative sets will produce negative messages  $R_{i \leftarrow j}^{(l, l+1)} < 0$ . Therefore, in case of small inputs, sometimes a large fraction of the relevance  $R_j$  can be turned into negative messages. On the other side, one can expect that in a neural network which predicts a positive score  $f(x) > 0$ , a majority of network inputs is positive, and a majority of neuron relevances  $R_j$  is positive, since their sum within a layer equals  $f(x) > 0$ . In particular one may expect from that, that the majority of neurons with large inputs  $z_j$  have positive relevances  $R_j > 0$ , and the messages for neurons with large inputs  $z_j$  are mostly positive. In conclusion, positive scores are less affected by dampening when  $f(x) > 0$ . Canceling out negative scores by a large value of  $\epsilon$  in intermediate layers leads also to more positive scores observed in the final heatmaps, and thus also to less visual granularity. We emphasize that this is a qualitative argument which we validated by checking a small number images rather than a universally holding claim. As for properties of Equation (60), we note that a choice of  $\beta < 0$  such that  $\alpha = 1 - \beta$  fixes the ratio of negative to positive messages to  $\frac{-\beta}{1-\beta}$  for each neuron. In particular, the sum of the negative messages is upper bounded by the sum of the positive messages. For rectified linear neurons this is a reasonable assumption because they fire only when their input is positive. In conclusion, our choice  $\beta = -1$ , yielding a ratio of 1 : 2 towards positive messages explains why the heatmaps for the method (60) in Fig. 24 and 26 are dominantly positive.

It is known from [21] that lower layers of deep networks which are very similar to the one from [60] used here may act like learned texture or gradient detectors. Fig. 26 shows that the pixel-wise decomposition is, however, not equivalent to simply assigning scores to dominant edges. The original images from Fig. 26 have many regions with strong gradients, however the pixel-wise decomposition assigns notable scores only to a small fraction of pixels with large gradients. Note for example that the highly textured and visually prominent wallpaper in the middle example with the table lamp receives only small scores. The same holds for the artifacts at the edges of the images which come from filling the image with locally constant values which is a compromise in order to make these images to be a square.





**Fig. 26. The pixel-wise decomposition is different from an edge or texture detector.** Only a subset of strong edges and textures receive high scores. Panels show the original image on the left, and the decomposition on the right. The decompositions were computed twice for the classes table lamp and once for the class rooster. The neural net is the pre-trained one on ILSVRC data from the Caffe package [60]. The computing methods for each column are the same as in Fig. 24. The last row shows the gradient norms normalized to lie in  $[0, 1]$  mapped by the same color scheme as for the heatmaps. Pictures in order of appearance from Wikimedia Commons by authors Wtshymanski, Serge Ninanne and Immanuel Clio.

## Discussion

Nonlinear learning machines are ubiquitous in modern problem solving. While highly successful in e.g. hard classification, regression or ranking problems, their non-linearity so far has prevented these models to additionally explain and thus contribute to a better understanding about the nature of the solved problem. Making, say, a nonlinear classification decision for one particular novel data point transparent to the user is essentially orthogonal to the standard task of optimizing for an excellent and well generalizing classifier. We have introduced a tool set for deconstructing a nonlinear decision and thus fostering transparency for the user.

In particular we have introduced the general concept of decomposition of a nonlinear image classification decision in terms of pixels. In other words, for a well-classified image, a heatmap can be produced that highlights pixels that are responsible for the predicted class membership. Note that this is possible without need of segmented training images. We consider heatmapping as an important part of the interpretation of nonlinear learning machines and its applicability goes far beyond what has been exemplarily presented in this work: it ranges from the interpretation of bio-medical images to the practical validation of a trained models for image classification.

Practically, we have proposed two different approaches to pixel-wise decomposition: The first one, Taylor-type decomposition, seeks to linearly approximate the class scoring function locally by performing a Taylor decomposition of it near a neutral data point without class membership, where the contribution of each dimension (i.e. pixel) can easily be identified. The second one, coined layer-wise relevance propagation, applies a propagation rule that distributes class relevance found at a given layer onto the previous layer. The layer-wise propagation rule was applied iteratively from the output back to the input, thus, forming another possible pixel-wise decomposition. This inherits the favorable scaling properties of backpropagation.

Notably, these two methods were not defined as a particular solution to the heatmapping problem, but instead as a set of constraints that the heatmapping procedure must fulfill in order to be admissible. For instance, the exact choice of Taylor reference point was not specified beyond the constraints that it should be a root and that it should be located close to the actual data point. Similarly, the layer-wise relevance propagation has been defined with the sole restriction that the propagation rule conserves class relevance on a layer and node basis. Thus, the further specification of relevance propagation rule is deferred to the appreciation of the user, or as a future work, and may either be model-specific, problem-specific, or subject to a particular practical or computational requirement.

Specific instances of the pixel-wise decomposition procedure satisfying the constraints mentioned above have been proposed and analyzed. In particular, our work has covered a set of non-linear learning algorithms for image classification, including kernel classifiers over Bag of Words pooled features, and feed-forward multilayer neural networks. Both models are popular choices for image classification or analysis.

Our experiments show that applying Taylor-type decomposition, layer-wise relevance propagation or a combination of both on these non-linear models produces highly informed heatmaps that reflect in many aspects the sophistication of the learned classifier. In particular, we have demonstrated that the same relevance propagation rule may, for different images, react to a variety of image features within the bounds modeling capacity. For example, in the case of the ImageNet convolutional network, we have shown that the heatmapping procedure finds class-relevant features that can be large areas of a particular color, localized features, image gradients, or more structured visual features such as edges, corners, contours, or object parts.

An important aspect of the proposed heatmapping procedure lies in the fact that it does not require to modify the learning algorithm, or to learn an additional model for heatmapping. Instead, it can be directly and transparently applied to any (pre-)trained Bag of Words model or neural network when applicable. This desirable property is demonstrated in this paper by the heatmapping of images classified by the third-party GPU-trained ImageNet neural network. In particular, our heatmapping procedure

was applied to this network without any further training or retraining. Thus, heatmaps for the ImageNet network could be quickly produced using a modest CPU.

While we have proposed in this paper several instances of pixel-wise decomposition and demonstrated their excellent performance in practice, the set of possible relevance propagation methods, and their mathematical properties, will certainly have to be further explored. A first aspect that needs to be investigated is the greediness of the layer-wise relevance propagation procedure, in the sense that it is computed one layer at a time, and its potential impact on the quality of heatmaps: While computationally advantageous, some of the backpropagated relevance might encounter a dead-end in the lower layers and be distributed randomly. Another open question relates to the heuristic nature of the proposed instances of relevance propagation, in particular, whether the distributed relevance messages being proportional to the weighted neuron activations can be analytically justified.

Finally, it is not clear how to evaluate the quality of a heatmap beyond simple visual assessment. In this paper we have proposed as a starting point a pixel-flipping method that allows to discriminate between two heatmapping methods that may otherwise look of similar quality to the human. Finding quantitative properties that are desirable for these heatmaps, or further constraints on the heatmapping procedure is therefore of nature to complement the human assessment. Future work will need to explore the many domain- and data-specific degrees of freedom in the heatmapping process in order to ultimately propose a universal metric for quantification. It is our firm belief, that heatmapping will be an important ingredient of future knowledge discovery and exploratory analysis and understanding of complex data in the sciences and industry, even beyond the presented field of image analysis.

## Acknowledgments

This work was supported in part by the Federal Ministry of Economics and Technology of Germany (BMWi) under the project THESEUS, grant 01MQ07018, by the German Ministry for Education and Research as Berlin Big Data Center BBDC, funding mark 01IS14013A and by DFG. KRM thanks for partial funding by the National Research Foundation of Korea funded by the Ministry of Education, Science, and Technology in the BK21 program. Correspondence should be addressed to SB, KRM and WS.

## Contributions

Conceived the theoretical framework: AB, GM, SB, WS. Conceived and designed the experiments: SB, AB, GM, WS, FK, KRM. Performed the experiments: SB, AB, GM. Wrote the manuscript: AB, SB, WS, GM, KRM, FK. Revised the manuscript: KRM, WS, GM, AB, SB. Figure design: WS, SB, GM, AB, FK. Wrote code for MNIST: SB, GM. Wrote code for ILSVRC: AB, Wrote code for BoW: SB, AB. Image search and generation: AB, SB, GM. Final drafting: all equally. The used images can be downloaded at [http://doc.ml.tu-berlin.de/layerwise\\_relevance\\_propagation/](http://doc.ml.tu-berlin.de/layerwise_relevance_propagation/) (Accessed 2015 Apr 01).

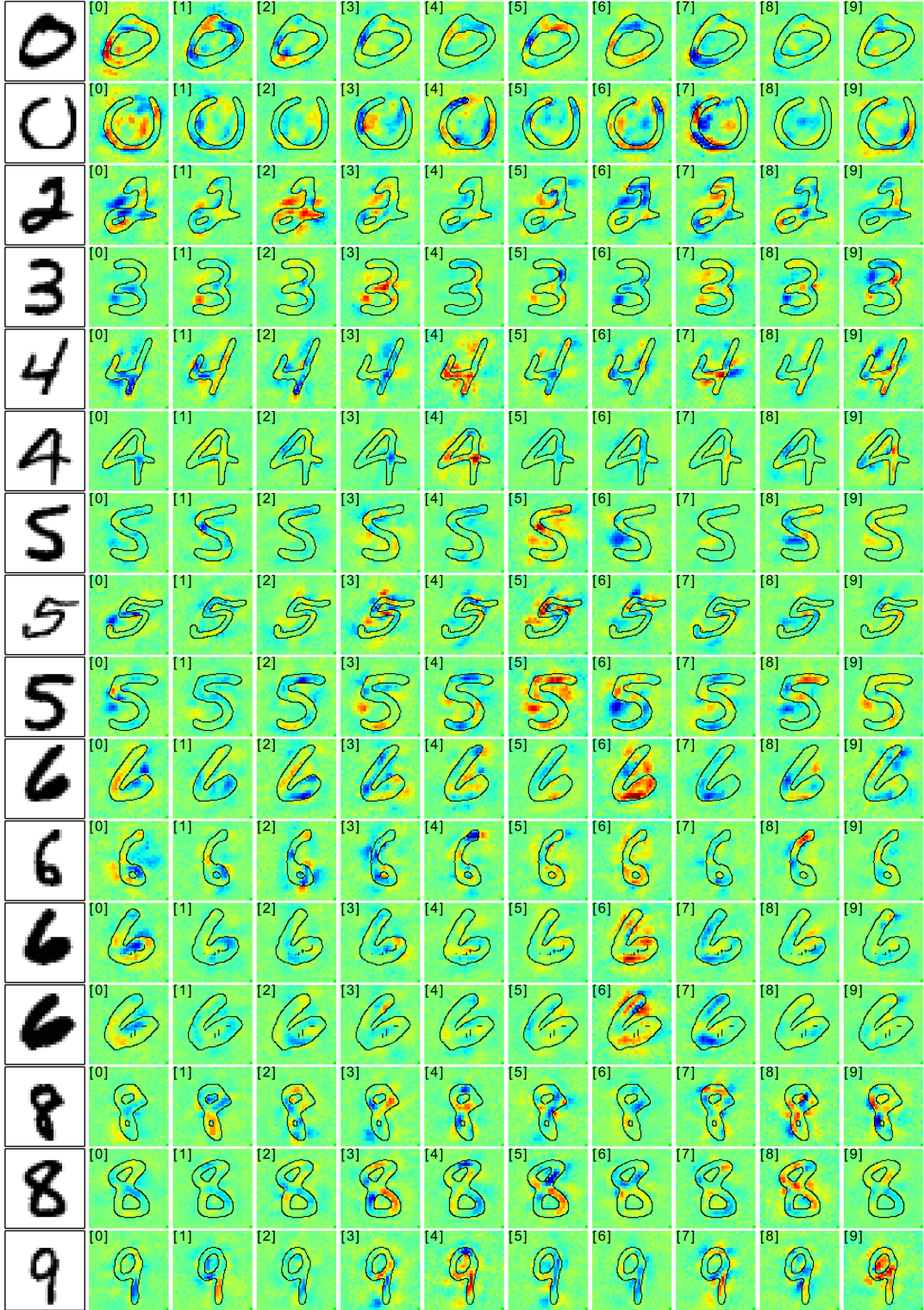


Fig. 27. Pixel-wise decompositions for all classes for 16 randomly drawn digits from the MNIST test set. Results are obtained using the relevance propagation Formula (56) with the rectifying network trained for 1 000 000 iterations from Section *MNIST experiments II* .

## References

1. Fei-Fei L., Perona P. A Bayesian Hierarchical Model for Learning Natural Scene Categories. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA. IEEE Computer Society; 2005. p. 524–531. Available from: <http://dx.doi.org/10.1109/CVPR.2005.16>.
2. Dahlkamp H, Kaehler A, Stavens D, Thrun S, Bradski GR. Self-supervised Monocular Road Detection in Desert Terrain. In: Robotics: Science and Systems; 2006. .
3. Walker R, Jackway P, Lovell B, Longstaff D. Classification Of Cervical Cell Nuclei Using Morphological Segmentation And Textural Feature Extraction. In: Australian New Zealand Conference on Intelligent Information Systems; 1994. .
4. Hänsch R, Hellwich O. Object Recognition from Polarimetric SAR Images. In: Soergel U, editor. Radar Remote Sensing of Urban Areas. vol. 15 of Remote Sensing and Digital Image Processing. Springer Netherlands; 2010. p. 109–131. Available from: [http://dx.doi.org/10.1007/978-90-481-3751-0\\_5](http://dx.doi.org/10.1007/978-90-481-3751-0_5).
5. Viola PA, Jones MJ. Rapid Object Detection using a Boosted Cascade of Simple Features. In: CVPR (1); 2001. p. 511–518.
6. Bishop CM. Neural Networks for Pattern Recognition. New York, NY, USA: Oxford University Press, Inc.; 1995.
7. Csurka G, Dance CR, Fan L, Willamowski J, Bray C. Visual categorization with bags of keypoints. In: In Workshop on Statistical Learning in Computer Vision, ECCV; 2004. p. 1–22. Accessed 2015 Apr 01. Available from: [http://www.xrce.xerox.com/Research-Development/Publications/2004-0105/\(language\)/eng-GB](http://www.xrce.xerox.com/Research-Development/Publications/2004-0105/(language)/eng-GB).
8. Deng J, Berg A, Satheesh S, Su H, Khosla A, Fei-Fei L. The ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012);. Accessed 2015 Apr 01. Available from: <http://www.image-net.org/challenges/LSVRC/2012/>.
9. Everingham M, Van Gool L, Williams CKI, Winn J, Zisserman A. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results;.
10. Müller H, Clough P, Deselaers T, Caputo B. ImageCLEF: Experimental Evaluation in Visual Information Retrieval. vol. 32 of The Information Retrieval Series. Springer; 2010.
11. Perronnin F, Sánchez J, Mensink T. Improving the Fisher Kernel for Large-Scale Image Classification. In: Daniilidis K, Maragos P, Paragios N, editors. ECCV (4). vol. 6314 of Lecture Notes in Computer Science. Springer; 2010. p. 143–156.
12. Yang J, Yu K, Gong Y, Huang TS. Linear spatial pyramid matching using sparse coding for image classification. In: CVPR. IEEE; 2009. p. 1794–1801.
13. Yu K, Zhang T, Gong Y. Nonlinear Learning using Local Coordinate Coding. In: Bengio Y, Schuurmans D, Lafferty JD, Williams CKI, Culotta A, editors. NIPS. Curran Associates, Inc.; 2009. p. 2223–2231.
14. Wang J, Yang J, Yu K, Lv F, Huang TS, Gong Y. Locality-constrained Linear Coding for image classification. In: CVPR. IEEE; 2010. p. 3360–3367.



15. van Gemert J, Veenman CJ, Smeulders AWM, Geusebroek JM. Visual Word Ambiguity. *IEEE Trans Pattern Anal Mach Intell.* 2010;32(7):1271–1283.
16. Montavon G, Orr GB, Müller KR, editors. *Neural Networks: Tricks of the Trade, Reloaded.* vol. 7700 of *Lecture Notes in Computer Science (LNCS)*. 2nd ed. Springer; 2012.
17. Baehrens D, Schroeter T, Harmeling S, Kawanabe M, Hansen K, Müller KR. How to Explain Individual Classification Decisions. *Journal of Machine Learning Research.* 2010;11:1803–1831.
18. Rasmussen PM, Schmah T, Madsen KH, Lund TE, Yourganov G, Strother SC, et al. Visualization of Nonlinear Classification Models in Neuroimaging - Signed Sensitivity Maps. In: Huffel SV, Correia CMBA, Fred ALN, Gamboa H, editors. *BIOSIGNALS*. SciTePress; 2012. p. 254–263.
19. Hansen K, Baehrens D, Schroeter T, Rupp M, Müller KR. Visual Interpretation of Kernel-Based Prediction Models. *Molecular Informatics.* 2011;30(9):817–826.
20. Zeiler MD, Fergus R. Visualizing and Understanding Convolutional Networks. *CoRR.* 2013;abs/1311.2901.
21. Krizhevsky A, Sutskever I, Hinton GE. ImageNet Classification with Deep Convolutional Neural Networks. In: Bartlett PL, Pereira FCN, Burges CJC, Bottou L, Weinberger KQ, editors. *NIPS*; 2012. p. 1106–1114.
22. Simonyan K, Vedaldi A, Zisserman A. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *CoRR.* 2013;abs/1312.6034.
23. Zeiler MD, Taylor GW, Fergus R. Adaptive deconvolutional networks for mid and high level feature learning. In: *ICCV*; 2011. p. 2018–2025.
24. Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow IJ, et al. Intriguing properties of neural networks. *CoRR.* 2013;abs/1312.6199.
25. Goodfellow IJ, Shlens J, Szegedy C. Explaining and Harnessing Adversarial Examples. *CoRR.* 2014;abs/1412.6572. Available from: <http://arxiv.org/abs/1412.6572>.
26. Chalupka K, Perona P, Eberhardt F. Visual Causal Feature Learning. *CoRR.* 2014;abs/1412.2309. Available from: <http://arxiv.org/abs/1412.2309>.
27. Erhan D, Bengio Y, Courville A, Vincent P. Visualizing Higher-Layer Features of a Deep Network. *University of Montreal*; 2009. 1341.
28. Le QV. Building high-level features using large scale unsupervised learning. In: *ICASSP*; 2013. p. 8595–8598.
29. Gevrey M, Dimopoulos I, Lek S. Review and comparison of methods to study the contribution of variables in artificial neural network models. *Ecological Modelling.* 2003;160(3):249–264.
30. Olden JD, Joy MK, Death RG. An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data. *Ecological Modelling.* 2004;178(3–4):389–397.
31. Setiono R, Liu H. Understanding Neural Networks via Rule Extraction. In: *IJCAI*. Morgan Kaufmann; 1995. p. 480–487.
32. Rasmussen PM, Madsen KH, Lund TE, Hansen LK. Visualization of nonlinear kernel models in neuroimaging by sensitivity maps. *NeuroImage.* 2011;55(3):1120–1131.

33. Liu L, Wang L. What has my classifier learned? Visualizing the classification rules of bag-of-feature model by support region detection. In: CVPR. IEEE; 2012. p. 3586–3593.
34. Uijlings JRR, Smeulders AWM, Scha RJH. The Visual Extent of an Object - Suppose We Know the Object Locations. *International Journal of Computer Vision*. 2012;96(1):46–63.
35. Everingham M, Van Gool L, Williams CKI, Winn J, Zisserman A. The PASCAL Visual Object Classes Challenge 2009, Part2 – Classification Task;.
36. Everingham M, Van Gool L, Williams CKI, Winn J, Zisserman A. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results;.
37. Nowak S, Nagel K, Liebetrau J. The CLEF 2011 Photo Annotation and Concept-based Retrieval Tasks. In: CLEF 2011 Labs and Workshop, Notebook Papers, 19-22 September 2011, Amsterdam, The Netherlands; 2011. Accessed 2015 Apr 01. Available from: <http://ceur-ws.org/Vol-1177/CLEF2011wn-ImageCLEF-NowakEt2011.pdf>.
38. Moosmann F, Nowak E, Jurie F. Randomized Clustering Forests for Image Classification. *IEEE Trans Pattern Anal Mach Intell*. 2008;30(9):1632–1646.
39. van Gemert J, Geusebroek JM, Veenman CJ, Smeulders AWM. Kernel Codebooks for Scene Categorization. In: ECCV (3); 2008. p. 696–709.
40. Liu L, Wang L, Liu X. In defense of soft-assignment coding. In: Metaxas DN, Quan L, Sanfeliu A, Gool LJ, editors. ICCV. IEEE; 2011. p. 2486–2493.
41. Binder A, Samek W, Müller KR, Kawanabe M. Enhanced representation and multi-task learning for image annotation. *Computer Vision and Image Understanding*. 2013;117(5):466–478.
42. Lowe DG. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*. 2004;60(2):91–110.
43. van de Sande KEA, Gevers T, Snoek CGM. Evaluating Color Descriptors for Object and Scene Recognition. *IEEE Trans Pattern Anal Mach Intell*. 2010;32(9):1582–1596.
44. Sánchez J, Perronnin F, Mensink T, Verbeek JJ. Image Classification with the Fisher Vector: Theory and Practice. *International Journal of Computer Vision*. 2013;105(3):222–245.
45. Lanckriet GRG, Cristianini N, Bartlett PL, Ghaoui LE, Jordan MI. Learning the Kernel Matrix with Semidefinite Programming. *Journal of Machine Learning Research*. 2004;5:27–72.
46. Bach FR, Lanckriet GRG, Jordan MI. Multiple kernel learning, conic duality, and the SMO algorithm. In: ICML; 2004. .
47. Kloft M, Brefeld U, Sonnenburg S, Laskov P, Müller KR, Zien A. Efficient and Accurate Lp-norm Multiple Kernel Learning. *Advances in Neural Information Processing Systems*. 2009;22(22):997–1005.
48. Kloft M, Brefeld U, Sonnenburg S, Zien A.  $l_p$ -Norm Multiple Kernel Learning. *Journal of Machine Learning Research*. 2011;12:953–997.
49. Binder A, Nakajima S, Kloft M, Müller C, Samek W, Brefeld U, et al. Insights from Classifying Visual Concepts with Multiple Kernel Learning. *PLoS ONE*. 2012;7(8).
50. Hwang SJ, Grauman K, Sha F. Semantic Kernel Forests from Multiple Taxonomies. In: NIPS; 2012. p. 1727–1735.

51. Binder A, Müller KR, Kawanabe M. On Taxonomies for Multi-class Image Categorization. *International Journal of Computer Vision*. 2012;99(3):281–301.
52. Blaschko MB, Zaremba W, Gretton A. Taxonomic Prediction with Tree-Structured Covariances. In: *ECML/PKDD (2)*; 2013. p. 304–319.
53. Cao L, Luo J, Liang F, Huang TS. Heterogeneous feature machines for visual recognition. In: *ICCV*. IEEE; 2009. p. 1095–1102.
54. Zhang J, Marszalek M, Lazebnik S, Schmid C. Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study. *International Journal of Computer Vision*. 2007;73(2):213–238. Available from: <http://dx.doi.org/10.1007/s11263-006-9794-4>.
55. Vedaldi A, Gulshan V, Varma M, Zisserman A. Multiple kernels for object detection. In: *ICCV*; 2009. p. 606–613.
56. Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. *Nature*. 1986 Oct;323:533–536.
57. LeCun Y, Cortes C. The MNIST database of handwritten digits; 1998. Accessed 2015 Apr 01. Available from: <http://yann.lecun.com/exdb/mnist/>.
58. Pinto N, Cox DD, DiCarlo JJ. Why is Real-World Visual Object Recognition Hard? *PLoS Comput Biol*. 2008 1;4(1):27.
59. LeCun Y, Kavukcuoglu K, Farabet C. Convolutional networks and applications in vision. In: *ISCAS*. IEEE; 2010. p. 253–256.
60. Jia Y. Caffe: An Open Source Convolutional Architecture for Fast Feature Embedding; 2013. Accessed 2015 Apr 01. Available from: <http://caffe.berkeleyvision.org/>.