Investigating Neighborhood Generation Methods for Explanations of Obscure Image Classifiers

Riccardo Guidotti[1,2], Anna Monreale[2], and Leonardo Cariaggi[2]

[1] ISTI-CNR, Pisa, Italy, guidotti@isti.cnr.it
[2] University of Pisa, Italy, anna.monreale@di.unipi.it, leonardocariaggi@gmail.com

Abstract. Given the wide use of machine learning approaches based on opaque prediction models, understanding the reasons behind decisions of black box decision systems is nowadays a crucial topic. We address the problem of providing meaningful explanations in the widely-applied image classification tasks. In particular, we explore the impact of changing the neighborhood generation function for a local interpretable model-agnostic explanator by proposing four different variants. All the proposed methods are based on a grid-based segmentation of the images, but each of them proposes a different strategy for generating the neighborhood of the image for which an explanation is required. A deep experimentation shows both improvements and weakness of each proposed approach.

1 Introduction

In the last years, automated decision systems are widely used in all those situations in which classification and prediction tasks are the main concern [20]. All these systems exploit machine learning techniques to extract the relationships between input and output. Input variables can be of any type, as long as it is possible to find a convenient representation for them. For instance, we can represent images by matrices of pixels or by a set of features that correspond to specific areas or patterns of the image [5, 16].

We talk about "black box" classifiers when dealing with classifiers having an opaque, hidden internal structure whose comprehension is not our main concern [12]. The typical example of black box is a neural network, one of the most used machine learning approaches due to its excellent performance. Therefore, the recent interest in explanations derives from the fact that we constantly use decision systems that we cannot understand. How can we prove that an image classifier built to recognize poisonous mushrooms actually focuses on the mushrooms themselves and not on the background?

Another reason for the recent interest in black box explanations is the *General Data Protection Regulation* approved by the European Parliament in May 2018. Besides giving people control over their personal data, it also provides restrictions and guidelines for automated decision-making processes which, for the first time, introduce a right of explanation. This means that an individual has the right to obtain meaningful explanations about the logic involved when automated decision making takes place [13, 18, 27].

In this work we study the reasons that lead classifiers to make certain predictions. A common approach for "opening" black boxes is to focus on the predictions themselves by understanding the predictions *a-posteriori* and comprehend on *what* the black box focused for returning the prediction [8]. A recently established approach consists in generating a neighborhood composed of both similar and different instances from the one to be explained [11, 21]. Then, observing the behavior of the black box on the neighborhood is possible to understand which are the features used for the prediction.

In this work, we propose four variants of the LIME method [21] that enable the explanation of image classifications based on sparse linear approaches. In particular, we design alternative ways for generating the neighborhood of the classified image for which an explanation is required. All the proposed methods use a grid-based approach for the image segmentation instead of the segmentation based on the *quickshift* [26] algorithm typical of LIME. However, each method differs in the strategy adopted for generating a neighborhood of images as perturbation of the image for which an explanation is required. The idea behind our proposals is to obtain neighbors by replacing some parts of the image to be explained with parts of other images and not by simply obscuring the original pixels. We experiment the impact of these approaches to understand which are the informative image regions and the overall quality of the explanations by introducing a systematic approach for the evaluation of explanations. Our evaluation highlights for each proposed approach both improvements and deficiencies.

The rest of this work is organized as follows. In Section 2 we provide an overview of state-of-art methods for explaining predictions in image classification. Section 3 summarizes LIME and provides evidence for some inconsistencies. In Section 4 we present the details of the proposed explanation methods. Section 5 contains a deep experimentation of the proposed approaches. Finally, Section 6 concludes the paper by discussing strengths and weaknesses of the proposed solutions and future research directions.

2 Related Work

In this section we provide an overview of the state-of-the-art for explaining the predictions of a black box image classifier. According to [12], the problem faced in this work is the *outcome explanation problem* that aims at returning a local explanation for an individual instance. The common strategy for the methods solving this problem is to provide a locally interpretable model (i.e. that can be clearly understood by a human). In case of image classification, such interpretable model can be an heatmap, defining the most important regions of the image that contribute to the prediction [21], or a mask, defining the minimal amount of information that, when deleted, causes the prediction to change drastically [8]. Other approaches, instead, aim at providing as explanation a prototypical image that clearly illustrates the treats for the given outcome [15].

Approaches based on saliency masks highlights which are the parts of an image that contribute the most to the prediction. A *saliency mask* is a subset of the record to be explained, i.e. a specific part of an image in our case, that causes the black box to make that specific prediction. The strategy adopted by [29] consists in the generation of attention maps for a CNN (Convolutional Neural Network), which highlight salient regions of an image and localize various categories of objects. Such maps are generated by using the backpropagation scheme. In this setting, each neuron of a convolutional layer can be matched with a specific area of an image. Also in the approaches presented in [23, 32] neuron activations are incorporated in their visual explanations. For this reason, the only family of models that can be explained by these approaches are CNN.

In [8] is presented a model-agnostic framework that defines saliency masks as *meaningful perturbations*. The goal is to study the effect of deleting specific regions from and image and find the smallest *deletion mask* or *artifact* that, when applied, causes the accuracy to drop significantly. Such deletion mask are presented as explanation. A problem with artifacts is that they can look unintuitive and unnatural and this impacts heavily on the quality of the visual explanation. The authors of [8] suggest to not focus on the specific details of the mask and apply a random jitter.

In [6] is developed a detection method performed by single forward pass (rather than iteratively, like in [31]) that produces high quality and sharp masks. The problem of artifacts is solved by cropping the input image rather than masking it: the goal is to find the tightest rectangular crop that contains the entire salient region of the image.

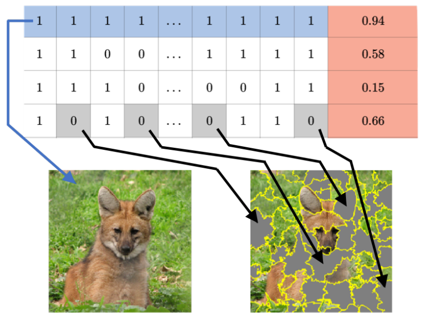
All the aforementioned methods define, with different strategies, an explanation at *pixel level* which identifies "unstructured" relevant areas of an image. The union of these areas may highlight a mixture of features that does not express a well defined concept but a blend of many aspects. In contrast, the aim of [30] is to produce explanation at *object level* that specify clear, distinct and highly interpretable parts of an image. The approach is based on the definition of *interpretable CNNs* by modifying the way features are represented inside filters of convolutional layers.

Our approach shares some properties with [23, 29, 32] with respect to the definition of masks highlighting the salient regions of an image for a certain prediction. However, in these approaches the masks are created by digging into the architecture of specific CNN and they can not be used for other types of black boxes (see [12] for more details). On the other hand, since we propose a set of methods extending LIME, they are black box agnostic by design. Finally, the proposed methods differ both from [21] and [8] because, as detailed in the following, the neighborhood is not generated by just blurring or obfuscating part of the image for which an explanation is required, but rather by replacing some feature of this image with features of other images.

As last remark we point out that, with respect to the way images are represented, our methods differs from traditional matrix of pixels, bags of pixels [14], and bag-of-words [5, 17, 28]. Since we intent to highlight, cut and replace contiguous patches of pixels, our natural choice is to use grids of RGB values obtained by aggregating the pixels in the same cell. Further details are provided in Section 4.

3 Background

In this section we summarize *LIME* [21], its main logic, and we provide evidence for some inconsistencies. LIME is a local *model-agnostic* explainer. It can provide an explanation for individual predictions of any classifier without making any assumptions on its internal structure. LIME's explanations consists in providing feedback that can help to understand which are the relationships between the input and the outcome. In the specific case of image classification, LIME 's explanation is a saliency mask highlighting the areas that the black box looks at when taking its decision.



1	1	1	1	...	1	1	1	1	0.94
1	1	0	0	...	1	1	1	1	0.58
1	1	1	0	...	0	0	1	1	0.15
1	0	1	0	...	0	1	1	0	0.66

Fig. 1. LIME neighborhood generation process. Left: original image x. Right neighbor image $z \in N_x$. Top neighborhood of IDRs weights, first row in blue original image weights, last column in red black box probability of labeling the image as *red fox* (best displayed in colors).

For every image to explain x, LIME builds an interpretable model that mimics the black box in the vicinity of x. To ensure interpretability, LIME builds ad-hoc *interpretable data representations* (IDR) $\hat{x}$ for x by "shattering" x into a set of areas made of contiguous pixels (regardless of their shape). The IDR $\hat{x}$ is therefore defined as an array of bits telling whether or not the single areas are present[3]. In order to build the local interpretable model for x, LIME generates a neighborhood N_x by randomly changing $\hat{x}$. Then, N_x is is provided to the black box for approximating its local behaviour and seeing how it reacts to variations of the input image x. To do that, a certain number m of samples is drawn uniformely at random from the domain of the IDRs in the form of binary vectors $w \in \{0,1\}^k$ where k is the number of features (contiguous patches of pixels[4]) in the image x and each term of w indicates the fact that the corresponding i-th feature is included, if $w_i = 1$, or not[5]. More formally, the neighborhood is defined as $N_x = \{\hat{x} \times w_i | w \in \{0,1\}^k\}$ where w is drawn uniformally at random from $\{0,1\}^k$.

The visual version of an image $z \in N_x$ has a plain color area (e.g. grey, white or black) for the features equals to 0 and the same areas of x for the features equals to 1. Figure 1 shows how a neighborhood image looks like. In the neighborhood image we highlight in yellow all the areas the image has been split into. They are contiguous patches of pixels with an irregular shape. LIME adopts *quickshift* [26] an algorithm based on an approximation of mean-shift [4] for obtaining the contiguous areas.

Keeping the label of the real image as a reference, LIME applies the black box to the images z in N_x obtaining an array $p \in [0,1]^m$ of prediction probabilities (highlighted in red in Figure 1), where m is the number of samples in N_x. Each value is the probability that the black box assigns the original label to a neighborhood image. Then, LIME trains a comprehensible regression model c where the input are the IDRs in N_x and the targets are the corresponding values of p. If a weight of c is *positive*, it means that the

[3] IDRs *do not* necessarily correspond to the features used by the black box in the prediction process. Indeed, such features may be not suitable for being shown as explanation.

[4] In the rest of this work, *feature, patch, area, piece* are used to denote the same concept.

[5] For the neighborhood N_x LIME generates m vectors w uniformly at random, assigning to them a weight that is proportional to their distance from the original image. The distance is used for assigning less importance to noisy images (that are too far away to be considered neighbors) and for focusing on the samples that are close to the original picture.

associated area of the image contributes positively to the prediction of the black box, if the weight is *negative*, then the corresponding feature contributes negatively. Hence, for LIME as well as for the methods we propose, an explanation e is a vector where e_i is the importance of the i-th feature of the image. Finally, LIME returns as explanation the top n features in e (ordering them by their weights, in descending order) and highlights the corresponding areas in the image. Intuitively, those are the areas where the black box looks when making such a prediction. An example of neighborhood images and explanation can be found in Figure 2-*(top left)*.

4 Explanation Methods

In this paper we try to overcome a conceptual problem with the neighborhood generation of LIME. Similarly to [8], LIME does not really generate images with different information, but it only randomly removes some features, i.e., it suppresses the presence of an information rather than modifying it. On the other hand, with respect to tabular data, LIME behaves differently. It generates the neighborhood not by information suppression, but by changing the feature values with other values of the domain: e.g., if the value of the *age* feature for a customer is 24, then LIME might replace it with 18 or 36. Our objective is to understand if an actual modification of the image to generate the neighborhood can lead to any improvement in the explanation itself.

Therefore, the goal of this work is to explore the impact of changing the neighborhood generation of LIME when it tries to provide an explanation for an image classification obtained by a black box. To this end we propose four different variants of the original LIME approach which are described in details in the following.

LIME# adopts a grid-based tessellation function that leads to patches with a regular shape. More formally, given an image x this function returns the vector $\hat{x}$ representing a grid with size $k{=}k_w{\times}k_h$. Any element $\hat{x}_i$ is a cell corresponding to the i-th feature of the original image. Given the IDR $\hat{x}$ and the vector w (introduced in the Section 3), the feature replacement function of LIME# works as in LIME, i.e., it replaces all the pixels in the feature x_i with $w_i{=}0$ with a plain color (e.g. gray, black or white). The *grid size* k is a parameter of LIME$\#$ and has to be specified by the user, for example in Figure 2-*(top right)* is fixed to $8{\times}8$. This figure shows the result of the application of this approach. The explanation illustrated by the green cells is simple and clear.

R-LIME# applies the same grid-based tessellation of LIME#. However, it completely changes the feature replacement function. In particular, it uses an image-based replacement, that substitutes any feature $\hat{x}_i$ having $w_i{=}0$ with the corresponding feature $\hat{z}_i$ of an image z randomly selected among a set of given images I, arbitrary provided by the user. As a consequence R-LIME# takes as input the image to be explained and a set of images I which will be used for the image-based replacement. Such kind of random generation would have been not possible in a straightforward way without the grid-based segmentation providing regular (and hence easily replaceable) features. For each image the algorithm computes the grid-based segmentation and generates the neighbors as described above. Figure 2-*(bottom left)* shows the application of R-LIME#, i.e., some images in the neighborhood and the explanation for a specific image.

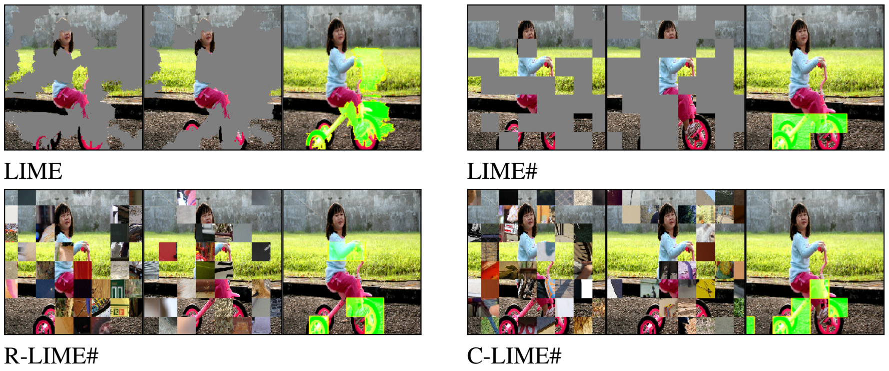


Fig. 2. Examples of neighborhood images generated by LIME, LIME#, R-LIME#, C-LIME#, respectively from left to right, from top to bottom. The first two images are an example of neighbors while the third one highlights the explanation for the label *tricycle* for the four methods.

C-LIME# differs from R-LIME# in the definition of the image-based replacement function. The basic idea of C-LIME# is to use images similar to that one to be explained for its neighborhood generation. Given an image x, and an initial set of images I, C-LIME# opportunely constructs the image pool, to be used for the neighborhood generation of x by selecting only images similar to x itself. To this aim, C-LIME# applies on I a clustering algorithm that returns groups of similar images $\mathcal{C} = \{C_1, \dots C_t\}$. Let f be the function that given an image x assigns it to the closer cluster. Then, for explaining a specific image x, C-LIME# builds the image pool using only images of the cluster $f(x)$. Once the image pool is built, the image-based replacement function works as R-LIME#. Figure 2-*(bottom right)* shows the impact of applying this method for the neighborhood generation. The neighborhood images still contain pieces of other images, but we can notice some interesting details such as wheels of other vehicles.

The clustering of images is computed by applying a process of undersampling of each image that depends on the size of the grid. Given the grid size $k_w \times k_h$ the clustering is performed on images undersampled to $k_w \times k_h$ pixels. The process first applies the grid-based segmentation to each image. Then, before the set of images is passed to the clustering algorithm they are flattened into a monodimensional vector. Clearly, C-LIME# is parametric with respect to the clustering algorithm and its own parameters.

Lastly, we propose **H-LIME#**, an *hybrid* approach that as C-LIME# applies a clustering algorithm on the set of images I (input of the algorithm) for finding groups of visually similar images $\mathcal{C} = \{C_1, \dots C_t\}$. However, the image pool is not composed only by images belonging to the same cluster of the image x to be explained. Instead, it includes the whole set of images, where each image x has associated the information about the cluster label $f(x)$. Thus, the image-based replacement function, given the image x to be explained, substitutes any feature $\hat{x}_i$ having $w_i = 0$ applying the following steps: (*i)* following a uniform probability distribution, it randomly decides if using an image belonging to the closest cluster $f(x)$ or one of the other images[6]; (*ii)* then, it randomly draws an image $\hat{z}$ from the selected group and replaces $\hat{x}_i$ with $\hat{z}_i$.

[6] For the sake of space we do not report experiments varying the probability of selection.

Food	Animals	Clocks	Shops	Vehicles
cheeseburger	timber wolf	analog clock	tobacco shop	motor scooter
hotdog	white wolf	digital clock	barbershop	mountain bike
	red wolf	digital watch	bookshop	unicycle
	coyote	wall clock	toyshop	tricycle
	dingo			bicycle-for-two

Table 1. Selected classes (from ILSVRC2012) for the dataset used in the experiments. For each of the 20 classes, we picked all the 50 corresponding images.

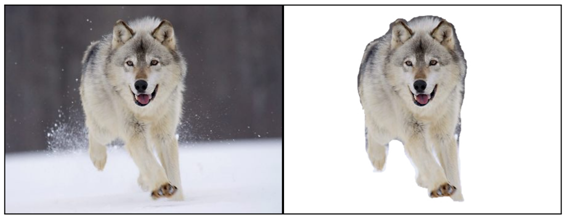


Fig. 3. Original image (left), Reference area (right).

5 Experiments

In this section we evaluate the methods described in Section 4. First, we present the dataset used for the evaluation of the explanations, then we define the measures used for the evaluation and finally, we present the results. The main points we address are the following: (*i*) what is the impact of the neighborhood generation? (*ii*) is it useful to generate neighborhood using visually similar images when replacing features? (*iii*) how well do the explanations approximate the area of the image containing the label? (*iv*) what is the ideal number of features? The experiments[7] were performed on Ubuntu 16.04.5 64 bit, 504 GB RAM, 3.6 GHz Intel Xeon CPU E5-2698 v4.

5.1 Dataset and Black Box

Answering the previous questions requires a dataset of images with the areas responsible for the image classification, so that they can be compared with those of the explanations. In our experiments we use a subset of the images of the *ILSVRC2012* dataset [22]. We selected a subset of 1000 images belonging to 20 classes from the test set, keeping them equally distributed (50 images per class). This set of images represent the images requiring an explanation. Table 1 shows the single classes, grouped by category. We grouped together similar objects to test the model's ability to distinguish between small particulars. In these images we isolated by hand their *reference areas* by selecting the parts that best represent the object specified in the label provided by the black box and filling all the remaining parts with a plain white color. The result of this operation is an enriched dataset that can be used as a reference when evaluating explanations. A sample image from such dataset is shown in Figure 3.

As black box we adopt the *Inception v3* [24]. Inception v3 is a CNN that operates on salient parts of the image. If we consider a picture of a bicycle, we may think of a

[7] Source code and dataset can be found at: `https://github.com/leqo-c/Tesi`

feature as one of its wheels. Hence, the areas of interest identified by the black box fit the type of explanations provided by LIME and by the variants we propose.

5.2 Evaluation Measures

In order to compare the different methods, we define some *quality* measures among explanations. Intuitively, we want to measure the ability of the explanations to cover the reference area: the more the explanation covers the reference area and do not cover not-reference areas, the more it can be trusted. Given the n most important features of an explanation e, namely $e^{(n)}$, and the reference area r for a certain image x, we redefine well-known *precision*, *recall* and *F-measure* as follows:

$$\textit{pre}(e^{(n)}, r) = \frac{|\{e_i^{(n)} \triangleright r | e_i^{(n)} \in e^{(n)}\}|}{n} \quad \textit{rec}(e^{(n)}, r) = \frac{\textit{pixels}(\{e_i^{(n)} \triangleright r | e_i^{(n)} \in e^{(n)}\})}{\textit{pixels}(r)}$$

$$\textit{F-measure}(e^{(n)} r) = (1 + \beta^2) \cdot \frac{\mathrm{pre}(e_n, r) \cdot \mathrm{rec}(e_n, r)}{(\beta^2 \cdot \mathrm{pre}(e_n, r)) + \mathrm{rec}(e_n, r)}$$

where $e_i^{(n)} \triangleright r$ means that a feature among the top n in the saliency mask of the explanation is completely contained in the reference area, and $\textit{pixels}(\cdot)$ returns the number of pixels in a certain area. The precision measures the ability to highlight only the relevant parts in the image, the recall measures the ability to highlight the overall number of pixels of the reference area, while the F-measure with β=0.5 puts more emphasis on precision than recall [3] as we want to focus on quantifying correct explanations [10].

5.3 Assessing Explanation Quality

In this section we analyze the results of the different methods in terms of precision and F-measure. The results are presented by varying the following parameters[8]:

- the grid size k that takes values in the power of 2, i.e., $8{\times}8$, $16{\times}16$, $32{\times}32$, etc[9]
- the number of top n features shown as explanation that takes values in $[1, 100]$.

As clustering algorithm for C-LIME# and H-LIME# we tested an array of approaches, i.e., K-Means [25], DBSCAN [7], Spectral [19]. We evaluated their performance both using internal validation measures (e.g., SSE [25]), and also by checking the cluster purity with respect to the label of the images clustered together. As result, we selected K-Means with 20 clusters as best performer to be used for C-LIME# and H-LIME#.

In Figure 4 and Figure 5 we report the average precision and F-measure, respectively. In these plots we vary the number of shown features n on the x-axis, and we report different lines for different values of the grid size k. Each plot reports the results with one of the proposed methods, from left to right LIME#, R-LIME#, C-LIME#, H-LIME#, and compares them with the original version of LIME.

[8] For the sake of simplicity of exposure and due to length constraints, we analyze both parameters in the same plots and we remand interested readers to the repository for further details.

[9] We do not report results using grid size lower than $8{\times}8$ (i.e., $2{\times}2$, $4{\times}4$, $6{\times}6$) or higher than $32{\times}32$ (i.e., $64{\times}64$, $128{\times}128$) has they have poor performance compared to those reported.

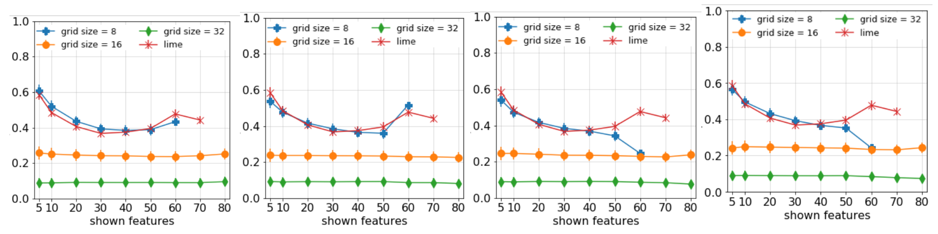


Fig. 4. Precision of LIME#, R-LIME#, C-LIME#, H-LIME# (from left to right) w.r.t. LIME varying the number of shown features, with different lines for different values of the grid size.

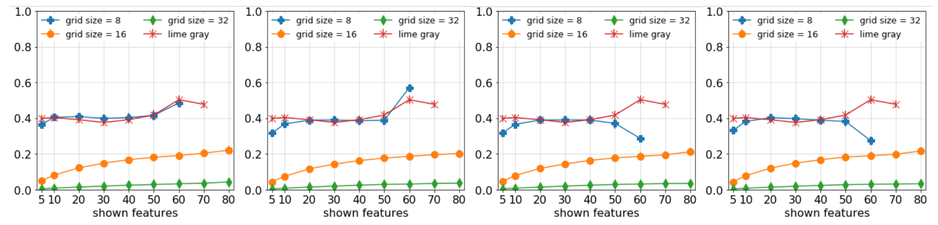


Fig. 5. F-measure of LIME#, R-LIME#, C-LIME#, H-LIME# (from left to right) w.r.t. LIME varying the number of shown features, with different lines for different values of the grid size.

The precision reaches a maximum of about 0.6 using LIME# (Figure 4 leftmost plot) with k=8×8 and considering 5 features in both. For *shown features* ≤ 50 LIME# outperforms LIME. Higher values of *grid size*, have a negative impact on the precision, which stabilizes around low values (about 0.25 and 0.1 respectively). R-LIME#, 2^{nd} plot in Figure 4, has a precision slightly worse than those of LIME#, except when *shown features* $= 60$. C-LIME#, 3^{rd} plot in Figure 4, is outperformed by R-LIME# in terms of precision. This is signaling that generating a neighborhood where the neighbors are created by replacing a part of an image with a part of a similar image (belonging to the same cluster) drops the ability of the explainer in finding which are the important features. Indeed, as we replace a patch with a visually similar patches, the probability of disrupting the prediction is lower, causing the features not to be considered relevant. H-LIME# (rightmost plot in Figure 4) tries to mitigate the weaknesses of C-LIME# by partly using a random replacement similar to R-LIME#. As result H-LIME# outperforms the other methods when *shown features* is in the range $[10, 40]$. Concerning the F-measure shown in Figure 5, we can notice the following aspects. First, with a number of *shown features* n which does not cover nearly the whole image (i.e., $n < 50$), the F-measure reaches a maximum of about 0.42 with LIME# with $n = 15$ and k=8×8 overcoming LIME. Second, *grid size* =8×8 always outperforms the methods with a different setting. Third, for *grid size* $\in \{16\times16, 32\times32\}$ the F-measure rapidly increases for *shown features* lower than 30, while remains almost constant for LIME.

An evident result is that high values of *grid size* inevitably lead to low-quality explainers. This is because the more we reduce the size of the features, the less we are likely to capture meaningful patterns inside an image. On the other hand, lower values noticeably increase the chance of highlighting informative regions. We should however

n	LIME	LIME#	R-LIME#	C-LIME#	H-LIME#
10	.485 ± .260	**.521 ± .256** ↑	.477 ± .260	.473 ± .257	.495 ± .252 ↑
15	.437 ± .243	.469 ± .243 ↑	.443 ± .246 ↑	.440 ± .244 ↑	**.478 ± .239** ↑
20	.406 ± .237	.436 ± .237 ↑	.416 ± .236 ↑	.417 ± .238 ↑	**.438 ± .231** ↑
25	.384 ± .234	.411 ± .231 ↑	.396 ± .232 ↑	.401 ± .234 ↑	**.415 ± .227** ↑
30	.367 ± .232	.393 ± .227 ↑	.384 ± .230 ↑	.384 ± .231 ↑	**.396 ± .224** ↑

Table 2. Precision (mean ± stdev) of the methods evaluated for grid size k=8×8). For each row, the best method is reported in bold, an up-arrow ↑ highlights if a method outperforms LIME.

n	LIME	LIME#	R-LIME#	C-LIME#	H-LIME#
10	.403 ± .181	**.405 ± .161** ↑	.368 ± .163	.365 ± .161	.382 ± .159
15	.398 ± .183	**.411 ± .172** ↑	.386 ± .173	.383 ± .172	.399 ± .170 ↑
20	.391 ± .191	**.409 ± .183** ↑	.389 ± .182	.390 ± .184	.403 ± .180 ↑
25	.384 ± .200	**.404 ± .191** ↑	.388 ± .192 ↑	.392 ± .195 ↑	.401 ± .189 ↑
30	.376 ± .207	**.399 ± .199** ↑	.389 ± .201 ↑	.389 ± .203 ↑	.397 ± .197 ↑

Table 3. F-measure (mean ± stdev) of the methods evaluated for grid size =8×8). For each row, the best method is reported in bold, an up-arrow ↑ highlights if a method outperforms LIME.

keep in mind that overlying extended areas (e.g. *grid size* =4×4) would lead to dispersive and coarse explanations. Moreover, it is worth to analyze some information about the sizes of the superpixels generated by LIME and those of the cells of the grid adopted by the proposed methods . The total number of pixels in an image of the dataset is about 90,000 (300×300). The average number of pixels in a superpixel generated by LIME is 1,405 ± 221. On the other hand, the number of pixels in a cell of the grid depends on the size of the grid. Using grid size k=8×8 we obtain $300/8\times300/8$= 1,406.25. Therefore, a *grid size* k=8×8 on average generates cells that resemble the dimension of the superpixels. This high correspondence of pixels in a feature is a plausible justification for the similar performance got by the proposed methods with respect to LIME.

As a consequence, the *LIME# family of methods is generally more precise than LIME itself. This statement is validated by the deeper analysis we present in the following. Considering that the object of an image, i.e., its reference area, on average covers more than one third of the image itself, we can suppose that for covering the relevant area we need a number of features between 15 and 30. Using *grid size* =8×8, in Table 2 and Table 3 we report the mean and standard deviation of precision and F-measure for the analyzed methods, respectively. We report in bold the best method for each row, an up-arrow ↑ highlights if a method outperforms the original version of LIME. From Table 2 we can notice that the members of the *LIME# family of methods are (nearly) always more precise than LIME, and H-LIME# is the most precise and stable method. On the other hand, Table 3 underlines that *(i)* only LIME# and H-LIME# overtake LIME in terms of F-measure, *(ii)* LIME# as the highest F-measure for the *shown features* observed, *(iii)* H-LIME# is still the most stable approach.

6 Conclusion

We have presented an array of methods based on the re-design of the neighborhood generation process of LIME. We have focused on the concept of interpretability of an explanation returned in terms of saliency mask, and we have tested the impact of using different techniques in the definition of the images' features. In particular, we argued that replacing patches with a solid color is not a natural way for producing a significant perturbed image. As additional contribution we have defined a systematic measure for assessing the quality of an explanation (i.e. what is its level of comprehensibility) by creating an annotated dataset, that contains for each image an ideal reference area, i.e., the part of the image that would be highlighted by an explanation provided by a human.

Our results show that replacing features with similar patterns (rather than random ones) like for C-LIME# prevents the explanation system from individuating relevant areas in the original image. On the other hand, H-LIME# which is based on both random patches and patches similar to the image to be explained, can help in improving the precision of an explanation. Nevertheless, LIME# which simply suppresses the information of a feature, has an overall better accuracy.

The findings of this paper are a solid starting point for a work that extends the methods presented by removing the constraint of the grid tessellation that, while simplifying the replacement process, it anchors all the approaches to behave too similarly to the original LIME method. Indeed, a fascinating research direction would be to adopt other techniques for embedding images into vectors (e.g., by means of image histograms [2], bag of visual words from SIFT key-points [16], or the embedding provided by another neural network [9]), to produce the random perturbations on these vectors and then, to reconstruct the corresponding neighbor images. Another interesting evolution of this work consists in extending the image neighborhood generation process using more sophisticated generation techniques, such as genetic programming [11], or an approach based on the minimum descriptive set [1], rather than drawing instances completely at random. Finally, even though an experimentation on different datasets and neural networks would not probably provide further insights relatively to the goodness of the neighborhood generation processes due to the nature of CNN generally used for classification of images, it would be interesting to check empirically these assumptions.

Acknowledgements. This work is partially supported by the European Community H2020 program under the funding scheme "INFRAIA-1-2014-2015: Research Infrastructures" G.A. 654024 *"SoBigData"*, `http://www.sobigdata.eu`, by the European Unions H2020 program under G.A. 78835, *"Pro-Res"*, `http://prores-project.eu/`, and by the European Unions H2020 program under G.A. 780754, *"Track & Know"*.

References

1. E. Angelino, N. Larus-Stone, D. Alabi, M. Seltzer, and C. Rudin. Learning certifiably optimal rule lists. In *KDD*, pages 35–44. ACM, 2017.
2. O. Chapelle, P. Haffner, and V. N. Vapnik. Support vector machines for histogram-based image classification. *IEEE transactions on Neural Networks*, 10(5):1055–1064, 1999.
3. D. M. Christopher, R. Prabhakar, and S. Hinrich. Introduction to information retrieval. *An Introduction To Information Retrieval*, 151(177):5, 2008.
4. D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:603–619, 2002.
5. G. Csurka et al. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague, 2004.

6. P. Dabkowski and Y. Gal. Real time image saliency for black box classifiers. In *Advances in NIPS 30*, pages 6967–6976. Curran Associates, Inc., 2017.
7. M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.
8. R. Fong and A. Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. *CoRR*, abs/1704.03296, 2017.
9. Y. Gal et al. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027, 2016.
10. R. Guidotti, A. Monreale, M. Nanni, F. Giannotti, and D. Pedreschi. Clustering individual transactional data for masses of users. In *KDD*, pages 195–204. ACM, 2017.
11. R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, and F. Giannotti. Local rule-based explanations of black box decision systems. *arXiv preprint arXiv:1805.10820*, 2018.
12. R. Guidotti, A. Monreale, F. Turini, D. Pedreschi, and F. Giannotti. A survey of methods for explaining black box models. *CoRR*, abs/1802.01933, 2018.
13. R. Guidotti, J. Soldani, D. Neri, A. Brogi, and D. Pedreschi. Helping your docker images to spread based on explainable models. In *ECML-PKDD*. Springer, 2018.
14. T. Jebara. Images as bags of pixels. In *ICCV*, pages 265–272, D.C., USA, 2003. IEEE.
15. B. Kim et al. The bayesian case model: A generative approach for case-based reasoning and prototype classification. In *Advances in NIPS*, pages 1952–1960, 2014.
16. D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision*, volume 2, pages 1150–1157. Ieee, 1999.
17. Z. Lu, L. Wang, and J. Wen. Image classification by visual bag-of-words refinement and reduction. *CoRR*, abs/1501.04292:197–206, 2015.
18. G. Malgieri and G. Comandé. Why a right to legibility of automated decision-making exists in the General Data Protection Regulation. *Int. Data Privacy Law*, 7(4):243–265, 2017.
19. A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.
20. D. Pedreschi, F. Giannotti, R. Guidotti, et al. Open the black box data-driven explanation of black box decision systems. *arXiv preprint arXiv:1806.09936*, 2018.
21. M. T. Ribeiro, S. Singh, and C. Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *KDD*, pages 1135–1144. ACM, 2016.
22. O. Russakovsky et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
23. R. R. Selvaraju et al. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016.
24. C. Szegedy et al. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.
25. P.-N. Tan et al. *Introduction to data mining*. Pearson Education India, 2007.
26. A. Vedaldi et al. Quick shift and kernel methods for mode seeking. In *Computer Vision*, pages 705–718, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
27. S. Wachter et al. Why a right to explanation of automated decision-making does not exist in the general data protection regulation. *Int. Data Privacy Law*, 7(2):76–99, 2017.
28. J. Yang et al. Evaluating bag-of-visual-words representations in scene classification. In *International workshop on multimedia information retrieval*, pages 197–206. ACM, 2007.
29. J. Zhang et al. Top-down neural attention by excitation backprop. *CoRR*, 1608.00507, 2016.
30. Q. Zhang, Y. N. Wu, and S.-C. Zhu. Interpretable convolutional neural networks. *arXiv preprint arXiv:1710.00935*, 2(3):5, 2017.
31. B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. *arXiv preprint arXiv:1412.6856*, 2014.
32. B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. *CoRR*, abs/1512.04150:2921–2929, 2015.