**Ex-LIME – An advanced version of LIME**

1. **INTRODUCTION**
2. Deep learning has performed miracles
3. CNNs are now being used more and more in real-life mission critical applications
4. But the reluctance in using this is lack of trust
5. This can be countered via explanations given to the predictions
6. There have been various attempts to generate explanation. For images, this translates to the one or more important patches in an image which plays the most important role towards the specific prediction
7. There have been various kinds of approaches, some uses generative approaches, some uses gradient-based approaches, and some uses perturbation-based approaches.
8. LIME has been one of these which uses a perturbation-based approach, and it has gained popularity due to its simplicity and non-invasive approach
9. Although LIME performs pretty well on images, there has been some shortcomings:
   1. The validation of the result is a manual process, which means that whether the explanation generated is good or not, depends on the feedback of a human being. This manual step restricts it from being deployed in automated large scale explanation generation system.
   2. Because there are multiple competing systems, the necessity of evaluation of the performances of different systems is also becoming more and more important.
   3. Another shortcoming of LIME is that, for generating explanation for different objects in a multi-object image, LIME needs to be run multiple times, once for each object, with different initializations.
10. In this paper, we propose several add-ons and post processing of the result of LIME for better understanding of the explanation, and also proposed a modified version of LIME named Ex-LIME. These are the details:
    1. An evaluation metric of the explanation generated by LIME or any other method
    2. An automated method to ensure a minimum quality of the explanation
    3. We propose a very simple segmentation scheme of the image, by simply creating a mesh, and treat each cell as a super-pixel of the image. We’ll show that the performance is comparable with other super-pixel method, and it’s obviously much more computationally efficient and manageable, too. Moreover, in this scheme we propose a method that will definitely generate a good explanation, so we don’t have to find out the minimum number of super-pixels.
    4. Finally, we propose a modification in the LIME process, and call this Ex-LIME (Extended LIME) where while generating the distance, instead of considering the top predicted class a single target class, we consider top-n classes according to the prediction. This gives us better results especially for multi-object scenario.
11. **A Survey of existing explanation generation systems**
12. **A brief explanation of LIME**
13. **Ex-LIME**
    1. **Evaluation Metric**:

To evaluate the performance of an XAI platform, without introducing the human intervention step, we introduced one feedback loop, very similar to the classical feedback control systems. The scheme is outlined in the following block-diagram.

**FEEDBACK LOOP**

**Super-Pixelation**

**Input**

**Image**

**Pixelated**

**Image**

**Classifier (CNN, etc)**

**Reference**

**Score Matrix**

**Perturbation Generator**

**Perturbed Images**

**Classifier (CNN, etc)**

**Distance Matrix**

**Linear Explanation Generation via Linear Regression**

Figure-1: Block diagram depicting the major computational steps of LIME and the extra step added by us in Ex-LIME

Let us assume that the original image ***I*** was segmented in ***n*** super-pixels **}**.

Among these super-pixels, let us assume that ***m*** super-pixels belong to the explanation set **.**

We generate a new image by perturbing all super-pixels except the explaining ones, i.e., a super-pixel is madeto 0 for some k where ; otherwise, its value is unchanged. This means that we feed the image back to the classifier which contains only the explaining patches, the rest of the image being blackened out, for example, Figure2A shows the perturbed image of a cat (original image is shown in Figure2), where all the super-pixels other than the explaining ones are blackened, before feeding it back to the classifier.

Then we record the classification scores of the classifier on this new image and evaluate the similarity with the classification scores on the original image. We expect that, if the explanation is a good one, i.e., if the classifier is using the identified patches to reach its classification decision, then the classification score on this perturbed image must be very similar to the classification result on the full image.

To measure the similarity between two classification results, we have designed a new metric called EE-Score (Explanation Effectiveness score), which we describe next:

Let us assume that our training data has ***Q*** classes.

A classification score generated by the classifier, on an unperturbed image ***I***, is a vector

And, the classification score generated by the same classifier, on the perturbed image ***IP*** is **.**

Then the EE-Score of this explanation is defined as

A couple of points about this score we would like to mention here:

* It is very easy to verify that the value of ***ees*** always lies in the interval **[0, 1]** (*Please refer to Appendix-1 for a formal proof*). Absolute dissimilarity will make the score 0, whereas absolute similarity (i.e., when two score vectors are identical) will make the score 1. This range makes it vary suitable to be used for comparison or benchmarking purpose.
* The term indicates that the difference between original and predicted scores gets more importance for the classes, where the original score is higher, i.e., the more important classes, whereas, for classes where the original score was very low, there higher differences are also neglected, because they do not matter much, for an explanation generation system.
* It is also very much evident from the expression that a better explanation means an ***ees*** value closer to 1. Please note here that by better explanation we mean that the scores computed for all the target classes are close, especially the ones with higher values.
  1. **Explanation Quality Assurance**:

Referring to LIME, we have seen that at the last step of LIME, the super-pixels are ordered in descending order of their weights (generated by the linear classifier) and top ***p*** super-pixels are selected as the constituents of the explanation system. The value of this ***p*** is manually chosen. The choice of a fixed value of p may not going to solve our problem, because of the following reasons:

* In case of a complicated image or a zoomed in image, the value of p will be naturally higher whereas for a simple zoomed out image, the value of p would naturally be on the lower side.
* On the other hand, if the number of super-pixels, generated from the original image, is on the higher side, so will be the value of p whereas a lower number of super-pixels (p) will be enough for a good explanation, if the total number of super-pixels generated from the original image is smaller.

We propose the following algorithm to select an optimum value of p automatically

*Algorithm: Find\_Appropriate\_Super-Pixel\_Count*(Algorithm to find out an optimum value of the number of super-pixels necessary to create an explanation which generatesa value of ees above a threshold)

*Input T 🡪 The minimum acceptable value of* ees *required for the system*

*Input I 🡪 The image for which we are trying to generate the explanation*

*Input C 🡪 The classifier whose decisions we are trying to explain*

*Output p 🡪 The optimum number of super-pixels*

*X := classification score vector generated by the classifier C on I*

*Run LIME*

*Rank the super-pixels according to their weights, in descending order*

*p := 1*

*ee := 0*

*while (ee < T) do*

*generate image I\_p by perturbing all super-pixels except top p*

*Y := classification score vector generated by the classifier C on I*

*ee\_new := EE\_SCORE(X, Y)*

*If (ee\_new > T)*

*Break;*

*Else*

*p := p+1*

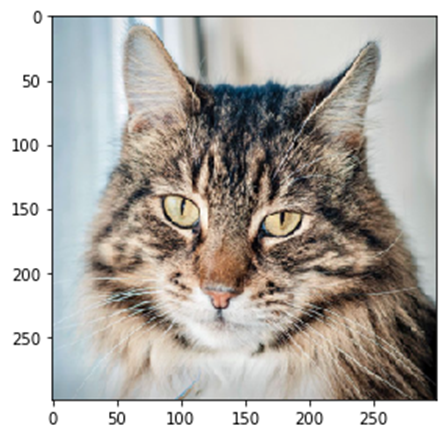
*end while*

*output 🡪 p*

*{sp\_1, sp\_2, …, sp\_p} constitutes the explanation that meets the quality requirement*

***An Example***: We discuss here the utility of the metric and the algorithm by two images.

* The classifier we have chosen is the Inception Net V3 [ref]
* We have taken the trained version of this network, trained on ImageNet [ref]
* The first test image we have chosen as a single image of a cat, as shown in Figure 2.
* The top 5 classes and their corresponding scores generated by InceptionNetv3 are depicted in Table 1.



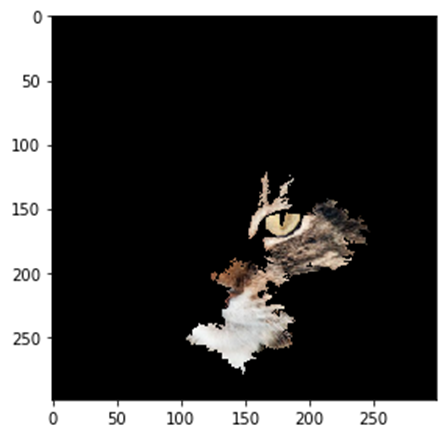
*Table 1: Classification score generated by InceptionNetV3 on the image of the cat shown in Figure 2.*

|  |  |
| --- | --- |
| **Class Name** | **Score** |
| tabby | 0.85062546 |
| tiger\_cat | 0.042112485 |
| Egyptian\_cat | 0.002530468 |
| Swab | 0.000632918 |
| Lynx | 0.000606151 |

Figure2: Image of a cat

* After running LIME (with number of perturbations = 500, and Linear Regression as the linear explanation generation system), and picking up the top 5 super-pixels (5 being the default choice) we get the explanation as shown in Figure 2A.
* Now, when this perturbed image is fed back into the InceptionNetv3 model, the top 5 classes and their corresponding classification score is shown in Table 1A.

*Table 1A: Classification score generated by InceptionNetV3 on the image of the cat shown in Figure 2.*



|  |  |
| --- | --- |
| **Class Name** | **Score** |
| bustard | 0.37 |
| prairie\_chicken | 0.16 |
| pot | 0.04 |
| wood\_rabbit | 0.03 |
| hare | 0.03 |

Figure 2A: Perturbed image of a cat with only the explaining patch unperturbed

* The explanation effectiveness score ees, computed using equation (i), yields a value of 0.467
* Now when we ran our algorithm *Find\_Appropriate\_Super-Pixel\_Count* on this result, the ees values we got are depicted in Figure 3.

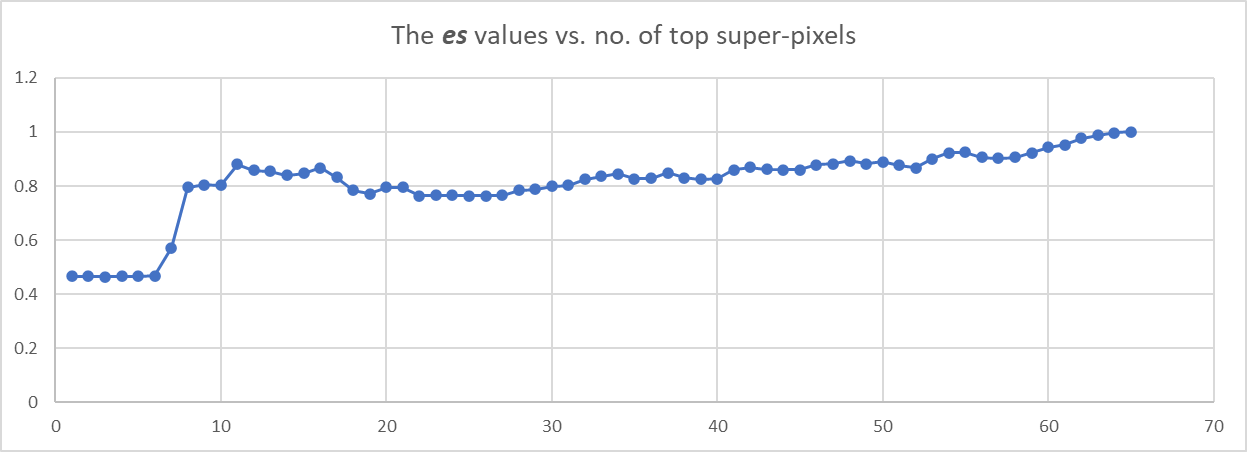
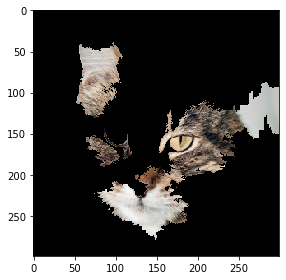


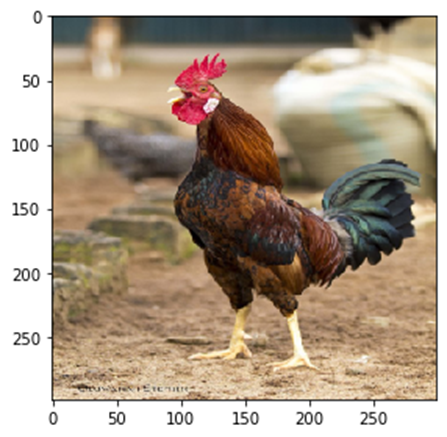
Figure 3: The values of ees, generated by classifying the perturbed images of the cat shown in Figure 1. It starts with only the top super-pixel un-perturbed, and gradually un-perturbs the super-pixels, one-by-one, according to their importance generated by LIME.

* This is clearly visible that with top 11 super-pixels, the explanation effectiveness is much better compared to the explanation with top 5 super-pixels.
* We have tried with top 11 super-pixels. The perturbed image looks like this:



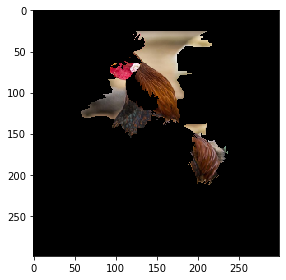
* The perturbed image when fed back into InceptionNetv3, the top-5 prediction scores are:
  + tabby - 0.81
  + Egyptian\_cat - 0.05
  + tiger\_cat - 0.01
  + pot - 0.004
  + feather\_boa - 0.004

***Second Example***: The second image we have taken is an image of a cock. The original image looks like this:

The original top 5 classes by InceptionNetV3 is:

1. cock - 0.95
2. hen - 0.007
3. partridge - 0.002
4. yellow\_lady's\_slipper - 0.058
5. cello - 0.0003

Top-5 super-pixels selected by LIME are shown below:

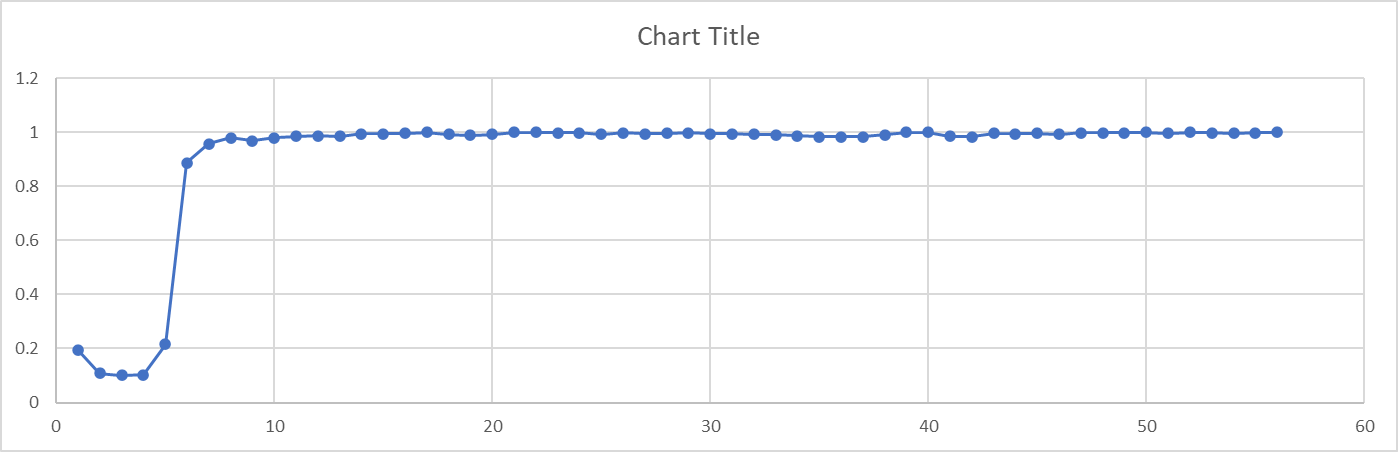


The perturbed image, with only the selected super-pixels unperturbed, when fed back into the classifier, generated the following top-5 classification:

* cock - 0.125
* kite - 0.118
* sulphur-crested\_cockatoo - 0.013
* bee\_eater - 0.058
* coucal - 0.054

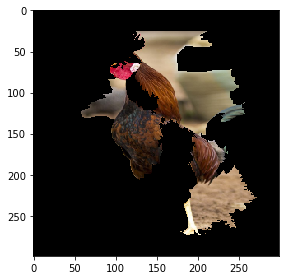
The ***es*** score for this selection is 0.216

Now, if we apply our algorithm, the graph of the ***es*** value is the following:



It is evident from the chart is that for 8 super-pixels the ***es*** value is pretty high 0.98

And when the perturbed image with only top-8 super-pixels unperturbed, is fed back into the classifier, the result generated by the classifier is the following:

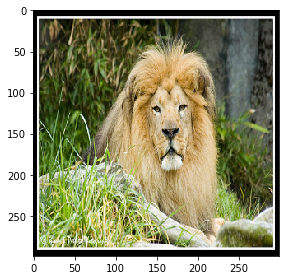
1. cock - 0.972
2. hen - 0.006
3. partridge - 0.001
4. prairie\_chicken - 0.0005
5. goldfish - 0.0002
   1. **Meshified Segmentation**:

So far, for creating the super-pixels, we had been using the quickshift segmentation technique. Now we show that, instead of going into that kind of a complex algorithm, we can very well use simple mesh techniques. We simply divide the image in an array of square grids, 8 x 8 or 10 x 10 or of any arbitrary size. Then each cell in this mesh will be considered as a super-pixel.

As we discussed in the previous section, that finding out the number of super-pixels as an explanation, which will give a permissible value of ***es*** involves some computation. Instead of that, in the meshified super-pixel scenario, without going through the iterative process, we can very easily pick up an explanation, which gives a good ***es*** score.

The process for doing that is explained below:

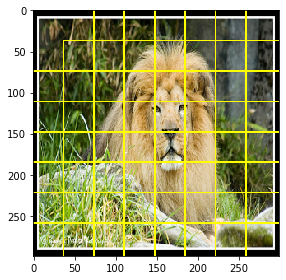
* 1. *Rank the super-pixels by LIME, as usual*
  2. *Pick up the top-3 (or top-5)*
  3. *Select all the neighbours of these top super-pixels*
  4. *This set is the explanation we are looking for*

Let us demonstrate this by an example:

The image we are working with is an image of a lion taken from the imagenet repository (n02129165\_1029.jpg)

The initial predictions by InceptionNetV3 are

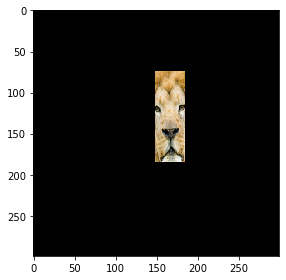
* + 1. lion – 0.955
    2. stopwatch – 0.0007
    3. cheetah – 0.0007
    4. chow – 0.0007
    5. bell\_pepper – 0.0004

The meshified super-pixels (8 x 8) looks like this.

The top 5 super-pixel’s and their coefficient values are

[28 36 20 30 44]

[0.47447904 0.40045162 0.11719285 0.09238523 0.08447626]

Without looking at the picture, just looking at the coefficient values, we can pick up the top-3 as the core explanation system. The top-3 super-pixels are this:

When this image is fed into InceptionNetV3, the predictions are:

lion – 0.553

chow – 0.2708

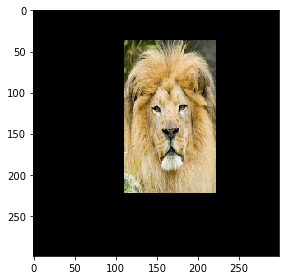
teddy – 0.0441

cheetah – 0.006

norfolk\_terrier – 0.0042

And the ***es*** score is 0.616.

Now, when we extend this explanation to the neighbouring super-pixels also, then the explanation becomes this:

This generates the ***es*** score as 0.926,

and the predictions are:

lion – 0.877

chow – 0.018

stopwatch – 0.002

collie – 0.002

jigsaw\_puzzle – 0.001

* 1. **Extended LIME (Ex-LIME)**:

So far, as we have discussed that in LIME, after generating the perturbed image, the images are fed to the classifier and the prediction score for the top class is recorded (Please note that instead of the top class, it might be the top 2nd class, or top nth class, in general).

Our proposed modification suggests that instead of the top class, we’ll consider top-m classes (m can be 2, 3 or any other number), for a better explanation. The intuition behind this modification is this, that

1. Identification of an object, generally, is not an isolated task, but it’s kind of a hierarchical decision. So while identifying a tiger, we probably identify it as a cat, then a tiger, while there might be some resemblance with a cheetah or a leopard as well. So, while matching the scores, only matching the top score might not be good enough, rather matching the scores of the top 2-3 classes should give us better result.
2. In case, there are multiple different types of objects are present in the image, then the predicted score for the top 2nd or 3rd class might be giving an indication of that. So, if we match the top score only, we are focussing only on the most prominent object but missing out the others.

Keeping these in mind, we are considering top 3 scores to start with. So now, the perturbed image remains a row matrix of 1’s and 0’s with ***p*** columns, where ***p*** is the number of super-pixels, but the predicted score for the desired classes is now 3 instead of 1, i.e., ***S1, S2, S3*** becomes the target value. So, if we have generated ***n*** number of perturbed images, each of which is of size 1 x p, The dimension of the weight matrix becomes ***p x 3***, and the general equation looks like the following:

To optimize the weights, we create a single layer Neural Network, with ***p*** input nodes, and ***3*** output nodes as shown in the diagram below:

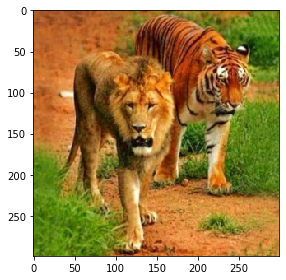
The first part of the overall error is calculated as:

The second part of the error adds a regularization for the above expression, given by:

And the final error is calculated as . The second part in the error expression will ensure that for a particular super-pixel, either its weights are all zero or all are non-zero, because for a particular value of j, high and will give us a confusing information about the overall importance of that super-pixel.

Then the weights are updated using Gradient Descent. Finally, the weights of a particular row are combined by using the expression , and the super-pixels are ranked according to the values of these ’s.

Using this scheme, we have got good result. Let us demonstrate this on the following image where we have a lion and a tiger:

The initial prediction by InceptionNetV3 gives the following top-5 classes along with their score:

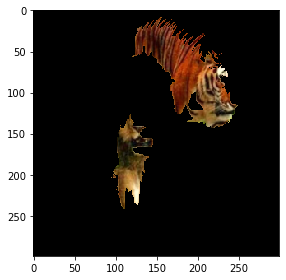
tiger: 0.491

lion: 0.133

tiger\_cat: 0.028

dhole: 0.009

chow: 0.005

Ordinary LIME generates the top 5 super-pixels as shown in the figure and this perturbed image, when fed back into the classifier, generates the following prediction:

tiger: 0.811

tiger\_cat: 0.063

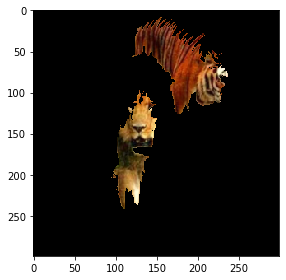
red\_wolf: 0.003

dhole: 0.003

screw: 0.002

And the ***es*** value is 0.82

The plot of super-pixels vs ***es*** values is shown below:

Whereas, when we ran Ex-LIME with number of classes to match as 3, then the top 5 super-pixels were this:

and the chart of es vs unperturbed super-pixels is given below. Evidently, what we can see here is that, with lesser number of super pixels the ***es*** value is approaching to 1. We further compared the result with total number of pixels, instead of super-pixels, and then also we find the same trend, which has been demonstrated below.