ciu.image: an R package for Explaining Image Classification with Contextual Importance and Utility ⋆

Kary Främling*$^{1,2[0000-0002-8078-5172]}$, Samanta Knapič$^{2[0000-0001-5926-6151]}$, and Avleen Malhi$^{3,2[0]}$

^{1} Dept. of Computing Science, Umeå University, 901 87 Umeå, Sweden
kary.framling@umu.se

^{2} Dept. of Computer Science, Aalto University, 02150 Espoo, Finland
{avleen.malhi,samanta.knapic}@aalto.fi

^{3} Department of Computing and Informatics, Bournemouth University, UK
amalhi@bournemouth.ac.uk

Abstract. Many techniques have been proposed in recent years that attempt to explain results of image classifiers, notably for the case when the classifier is a deep neural network. This paper presents an implementation of the Contextual Importance and Utility method for explaining image classifications. It is an R package that can be used with the most usual image classification models. The paper shows results for typical benchmark images, as well as for a medical data set of gastroenterological images. For comparison, results produced by the LIME method are included. Results show that CIU produces similar or better results than LIME with significantly shorter calculation times. However, the main purpose of this paper is to bring the existence of this package to general knowledge and use, rather than comparing with other explanation methods.

Keywords: Explainable Artificial Intelligence · Contextual Importance and Utility · Image Classification · Deep Neural Network.

1 Introduction

Contextual Importance and Utility (CIU) is a method originally developed by Kary Främling in his PhD thesis [3]. CIU was developed in a context of multiple criteria decision making (MCDM), which is a domain where different mathematical models are used as decision support systems for human decision makers. Possible mathematical models include any kind of AI system, including systems created using machine learning. CIU was designed to provide a mechanism for explaining or justifying the outcome of any such AI system in a uniform way, no matter if it is considered to be a so-called black-box model or not.

⋆ Corresponding Author: Kary.Framling@cs.umu.se.
The work is partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

As in many MCDM methods, CIU makes a difference between *feature importance* and *value utility*. The feature importance expresses to what extent an input feature (or combination of input features) can change the output value. Value utility expresses to what extent the input value contributes towards a high-utility output value. In classification tasks there is usually one output per class and the output value is a class probability value, that can be directly used as the output utility value. Both feature importance and value utility can change depending on the studied instance or context, which is why feature importance is called *Contextual Importance (CI)* and value utility is called *Contextual Utility (CU)*. CI and CU are scalars in the range $[0, 1]$ and are absolute (non-relative) values.

The purpose of this paper is to present an R implementation of CIU for explaining image classification. This 'ciu.image' package is available at https://github.com/KaryFramling/ciu.image. It is a follow-up package to the 'ciu' R package for tabular data available at https://github.com/KaryFramling/ciu [4]. After this Introduction, Section 2 provides implementation details of CIU for image explanation. Section 3 gives software installation and usage instructions. Section 4 shows example results on ImageNet and medical image explanations, followed by Conclusions.

2 Contextual Importance and Utility for Images

The most recent description available about CIU is found in [4]. Only the most relevant parts for explaining image classification with CIU are included here, i.e. the basic definitions of CI and CU. CI expresses how much the output value utility can change when modifying the value(s) of one or several input features $\{i\}$ relative to the total output range:

$$CI_j(\vec{C}, \{i\}) = \frac{Cmax_j(\vec{C}, \{i\}) - Cmin_j(\vec{C}, \{i\})}{absmax_j - absmin_j} \tag{1}$$

CU expresses to what extent the current input feature values $\vec{C}$ are favorable for a high output value utility:

$$CU_j(\vec{C}, \{i\}) = \frac{y_j(\vec{C}) - Cmin_j(\vec{C}, \{i\})}{Cmax_j(\vec{C}, \{i\}) - Cmin_j(\vec{C}, \{i\})} \tag{2}$$

Here, $y_j(\vec{C})$ is the value of output j for the current instance. $Cmin$ and $Cmax$ are the minimal and maximal output values achievable by modifying the value of the given input feature(s) with indices $\{i\}$. $absmin$ and $absmax$ give the minimal and maximal possible values for the output. In the case of explaining image classification (and for classification tasks in general) it is reasonable to use $absmin = 0$ and $absmax = 1$.

'ciu.image' segments images into so-called super-pixels using Simple Linear Iterative Clustering (SLIC) [1] in the same way as the LIME package [6]. Therefore, the input features of CIU actually consist of super-pixel values. The super-pixel values could in principle be 'anything' but what is used in practice is to have only the values present/not-present, where 'not-present' corresponds to setting the super-pixel to transparent. In practice, this leads to a rather trivial implementation of CIU for explaining image classification because calculating CI and CU values of a super-pixel requires exactly two forward-passes of the classification model, i.e. one with the original image and one with the super-pixel(s) of interest set to transparent. For an image with 50 super-pixels, for instance, only 51 forward passes are needed for calculating the CIU values of all super-pixels.

When interpreting the results, the super-pixels with the highest CI values are the ones that contribute the most to the classification result. CU can only take values zero or one in this case, where $CU = 1$ signifies that the contribution is positive and $CU = 0$ signifies that the contribution is negative. As shown in Section 4.1, this approach works relatively well for ImageNet classification. For the gastro-enterological images in Section 4.2, this approach is not sufficient because bleeding in **any** super-pixel will lead to classifying the image as 'bleeding'. However, since CIU can be calculated for any number of input features, an 'inverse' option was introduced, where all other super-pixels except the studied one are set to transparent, which efficiently identifies all the super-pixels with bleeding present.

3 Installation and Use

The package is available at https://github.com/KaryFramling/ciu.image. Installation instructions are also found there. The simplest way to install the package is to first install the 'devtools' package and then install 'ciu.image' with the command `devtools::install_github('KaryFramling/ciu.image')`. The package is loaded with the command `library(ciu.image)`. A `ciu.image` object is created by calling `ciu.image.new(model)` that uses the given predictor model. The optional parameters of `ciu.image.new` and the methods of `ciu.image` objects are explained in the package documentation and reflect the latest updates to the package. Since `ciu.image` is still a research tool, it is expected to evolve over time. Currently, the core `ciu.image` methods are the following:

1. `ciu.superpixels(imgpath, ind.outputs=1, n_superpixels=50, weight=20, n_iter=10, background = "grey", strategy = "straight")`: Return a list with fields `out.names`, `outval`, `CI`, `CU`, `cmin`, `cmax` for the requested number of outputs in `ind.outputs`, where outputs are ordered according to decreasing output value. Only `imgpath` is a compulsory parameter, for the others the default values are often appropriate.
2. `plot.image.explanation(imgpath, ind.outputs=1, threshold=0.02, show_negative=FALSE, n_superpixels=50, weight=20, n_iter=10, background="grey", strategy="straight", ciu.sp.results=NULL, title=NULL)`: Return a list of ggplot objects for the requested number of outputs in `ind.outputs`, where outputs are ordered according to decreasing output value. Most parameters are the same as for `ciu.superpixels`.

The use of 'ciu.image' typically happens as follows:

```
ciu <- ciu.image.new(model, predict_function, output.names)
plist <- ciu$plot.image.explanation(imgpath)
print(plist[[1]])
```

A complete code example is shown in Appendix 1.

4 Results

The experiments were run using Rstudio Version 1.3.1093 on a MacBook Pro, with 2,3 GHz 8-Core Intel Core i9 processor, 16 GB 2667 MHz DDR4 memory, and AMD Radeon Pro 5500M 4 GB graphics card. The LIME R package was used for producing LIME results [6]. For image classification, CIU is entirely deterministic so it always produces the same results, whereas LIME results tend to vary from one run to the other.

4.1 ImageNet classification

Results are shown here for VGG16 and VGG19 models included in the 'keras' package, pre-trained on ImageNet images. The two images used are shown in Figure 1. Appendix 1 shows the complete source code for producing the cat classification results shown in Figure 2, both for CIU and LIME. For CIU, this source code is also included in the online documentation and is accessible by writing `?ciu.image.new` on the R command line. Cat calculation times are 30 sec. for CIU and 5 min. 44 sec. for LIME. Dog playing guitar times are 18 sec. for CIU and 4 min. 44 sec. for LIME.

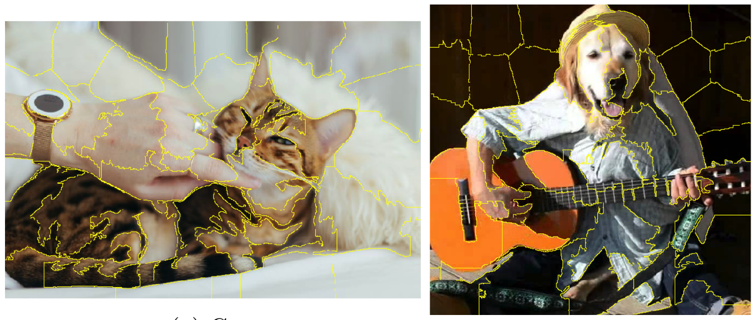
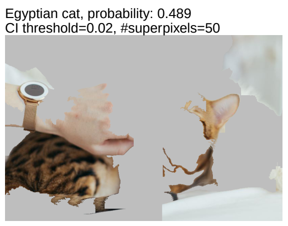
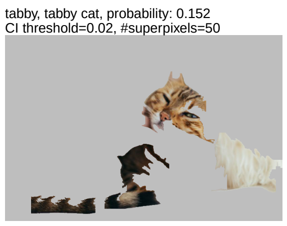

(a) Cat.   (b) Dog playing guitar.

Fig. 1: Original images, with super-pixel borders shown by yellow lines.

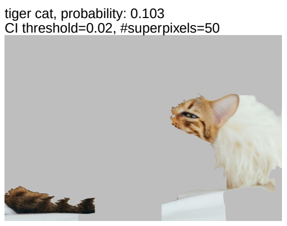
For the cat explanations shown in Figure 2, CIU and LIME explanations are quite similar. However, CIU extracts what makes the difference between Egyptian, Tabby and Tiger cat according to the VGG16 model, whereas LIME


Egyptian cat, probability: 0.489
CI threshold=0.02, #superpixels=50

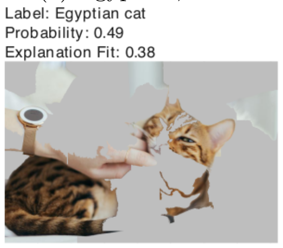
(a) Egyptian, CIU.


tabby, tabby cat, probability: 0.152
CI threshold=0.02, #superpixels=50

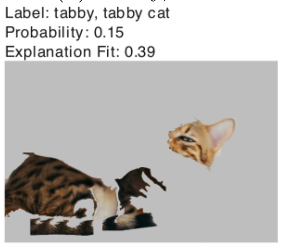
(b) Tabby, CIU.


tiger cat, probability: 0.103
CI threshold=0.02, #superpixels=50

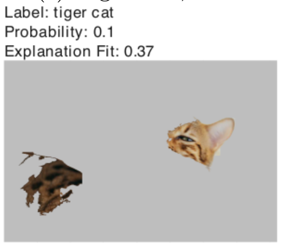
(c) Tiger Cat, CIU.



(d) Egyptian, LIME.



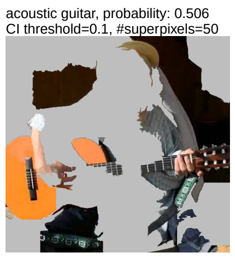
(e) Tabby, LIME.



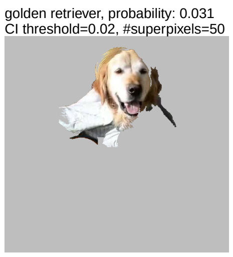
(f) Tiger Cat, LIME.

Fig. 2: Cat image explanation results using LIME and CIU trained with VGG16.

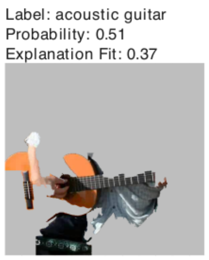
includes the same super-pixels for all three kinds of cat, with a smaller sub-set included for the two lower-probability cat types Tabby and Tiger Cat. For the guitar-playing dog image used in [7], the results are shown in Figure 3. In this case, the interpretation of image explanations tends to be subjective and also depends on how the underlying trained model makes the classification, so it does not make much sense to declare a 'winner'. Furthermore, LIME results tend to change somewhat from one run to the other, whereas CIU results are guaranteed to be identical for every run.


acoustic guitar, probability: 0.506
CI threshold=0.1, #superpixels=50

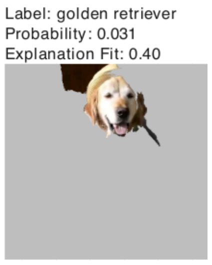
(a) CIU.


golden retriever, probability: 0.031
CI threshold=0.02, #superpixels=50

(b) CIU.



(c) LIME.



(d) LIME.

Fig. 3: Dog playing guitar image explanation results using LIME and CIU trained with VGG19 for 'Acoustic guitar' and 'Golden retriever'.

4.2 Gastro-enterological image explanation

The image data set considered in this case is taken from a Video Capsule Endoscopy (VCE), which is a non-invasive procedure to visualize the entire gastro-enterological tract of a patient. The data set of 3,295 images, retrieved from Coelho[4] [2] was split into 2,941 training and 354 validation images (randomly assigned), and it is a binary classification (bleeding or not). The medical data set was trained using the Convolutional Neural Network (CNN) model from [5], with 50 epochs with batch size of 16 and achieving a validation accuracy of 98.58%.

CIU explanations were generated using the parameter value `strategy="inverse"` to the `plot.image.explanation` method. The threshold value was 0.01 and 50 super-pixels were used. Some CIU results are shown in Figure 4. For non-bleeding images, LIME failed to produce a result with the default settings, as well as for many of the bleeding images. For the bleeding images where LIME gave a result, CIU and LIME results were often quite similar, even though CIU was clearly more precise. LIME's 'explanation fit' tended to be below 0.001, which indicates that the fitted LIME model has low or no explanatory value.

For 'bleeding' images, the parts (super-pixels) identified by CIU were considered relevant and correct by a user panel and also corresponded to the masks of 'correct' answers available for the image set. For 'non-bleeding' images, all super-pixels that belong to the actual image should be included because they are all 'non-bleeding'. However, the black corners of the images are present in all images and therefore do not have any discriminatory effect between 'bleeding' and 'non-bleeding' images, so they have no significance for the classification. CIU indeed filters out those black areas from the explanation, as seen in Figure 4. CIU took less than 7 sec. per image, whereas LIME took about 1 min. 40 sec per image.

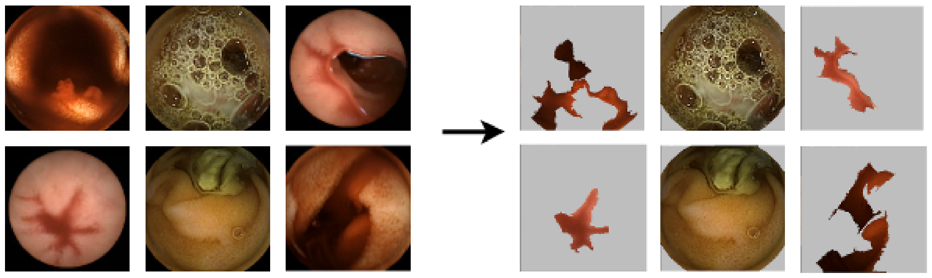


Fig. 4: CIU explanations generated for 'bleeding' (left and right columns) and 'non-bleeding' (middle column) images.

[4] https://rdm.inesctec.pt/dataset/nis-2018-003

5 Conclusions

This CIU implementation for explaining image classification shows that CIU can produce explanations that are at least at comparable level to LIME. For explaining gastro-enterological image classification, CIU manages to produce 'makes-sense' explanations for all images, whereas LIME fails to produce explanations for several images. Moreover, CIU is orders of magnitude faster than LIME, which might in the future be used for exploiting CIU's capability to deal with super-pixel combinations in different ways, rather than only setting one super-pixel transparent, or the opposite. Therefore, CIU's performance can be expected to improve further with future research.

References

1. Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S.: Slic superpixels compared to state-of-the-art superpixel methods. IEEE Transactions on Pattern Analysis and Machine Intelligence **34**(11), 2274–2282 (2012). https://doi.org/10.1109/TPAMI.2012.120
2. Coelho, P., Pereira, A., Salgado, M., Cunha, A.: A deep learning approach for red lesions detection in video capsule endoscopies. In: Digital Image Computing: Techniques and Applications (DICTA). pp. 553–561. Springer (2018)
3. Främling, K.: Modélisation et apprentissage des préférences par réseaux de neurones pour l'aide à la décision multicritère. Phd thesis, INSA de Lyon (Mar 1996), https://tel.archives-ouvertes.fr/tel-00825854
4. Främling, K.: Contextual importance and utility in R: the 'ciu' package. In: Proceedings of 1st Workshop on Explainable Agency in Artificial Intelligence, at 35th AAAI Conference on Artificial Intelligence. pp. 110–114 (2021)
5. Malhi, A.K., Kampik, T., Pannu, H.S., Madhikermi, M., Främling, K.: Explaining machine learning-based classifications of in-vivo gastral images. In: 2019 Digital Image Computing: Techniques and Applications, DICTA 2019, Perth, Australia, December 2-4, 2019. pp. 1–7. IEEE (2019)
6. Pedersen, T.L., Benesty, M.: lime: Local Interpretable Model-Agnostic Explanations (2019), https://CRAN.R-project.org/package=lime, R package version 0.5.1
7. Ribeiro, M.T., Singh, S., Guestrin, C.: "why should I trust you?": Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016. pp. 1135–1144 (2016)

Appendix 1: Source Code for ImageNet cat results.

```
library(keras)
library(lime)
library(magick)
library(ciu.image)
imgpath <- system.file('extdata', 'kitten.jpg',
  package = 'ciu.image')
# load VGG16 image classifier trained on imagenet database
model <- application_vgg16(weights = "imagenet", include_top = TRUE)
# We have to tell how images are prepared and evaluated.
vgg_predict_function <- function(model, imgpath) {
  predict(model, image_prep(imgpath))
}
# Standard preparation for imagenet, VGG16 & VGG19
image_prep <- function(x) {
  arrays <- lapply(x, function(path) {
    img <- image_load(path, target_size = c(224,224))
    x <- image_to_array(img)
    x <- array_reshape(x, c(1, dim(x)))
    x <- imagenet_preprocess_input(x)
  })
  do.call(abind::abind, c(arrays, list(along = 1)))
}
model_labels <- readRDS(system.file('extdata',
'imagenet_labels.rds', package = 'ciu.image'))
ciu <- ciu.image.new(model, vgg_predict_function,
output.names = model_labels)
# Get explanation for three topmost classes.
# Use `threshold` parameter for adjusting CI level to show.
plist <- ciu$plot.image.explanation(imgpath, c(1,2,3))
for ( i in 1:3 ) print(plist[[i]])

# These lines generate corresponding LIME explanations.
explainer <- lime(imgpath, as_classifier(model, model_labels),
image_prep)
explanation <- explain(imgpath, explainer, n_labels = 3,
n_features = 50, n_superpixels=50)
explanation <- as.data.frame(explanation)
p <- plot_image_explanation(explanation, display = 'block',
threshold = 0.01)
print(p)
```