Explainable Artificial Intelligence, or **XAI** for short, is a set of tools that helps us understand and interpret complicated **"black box"** machine and deep learning models and their predictions. Today I would like to show you a sneak peek of my newest package called **sauron**, which allows you to explain decisions of Convolutional Neural Networks.

What exactly does CNN see?

Let's start with the basic. We're gonna need a model, test images for which we want to generate explanations and image preprocessing function (if needed).

```
library(sauron)

input_imgs_paths <- list.files(system.file("extdata", "images", package
= "sauron"), full.names = TRUE)
model <- application_xception()
preprocessing_function <- xception_preprocess_input
```

There's a ton of different methods to explain CNNs, but for now with `sauron` you have access to 6 **gradient based** ones. You can check full list using `sauron_available_methods` function:

```
sauron_available_methods
# # A tibble: 6 x 2
#   method name
#
# 1 V      Vanilla gradient
# 2 GI     Gradient x Input
# 3 SG     SmoothGrad
# 4 SGI    SmoothGrad x Input
# 5 IG     Integrated Gradients
# 6 GB     Guided Backpropagation
```

Package is still in development so I won't talk about the theory of those methods today. I will leave it for another post (or more probably multiple posts 🙂).

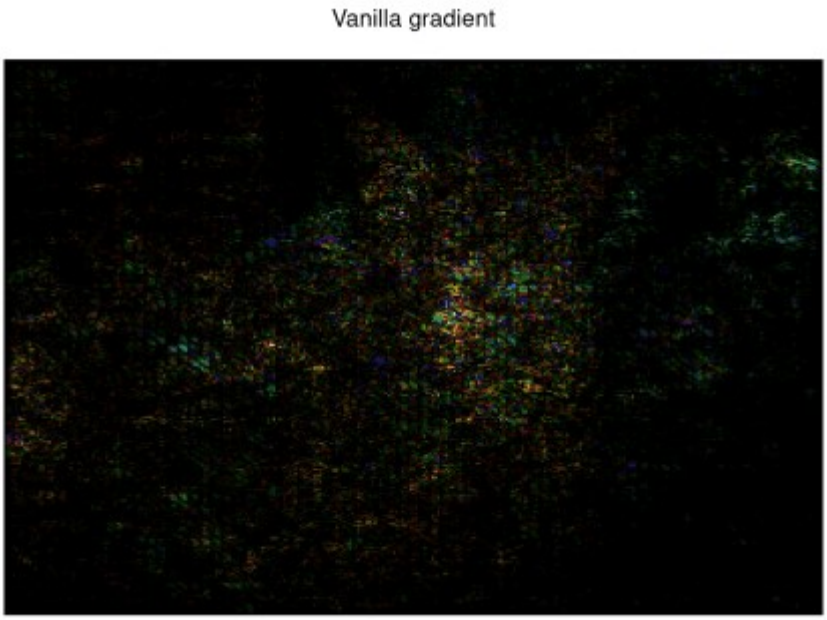
To generate any set of explanations simply use `generate_explanations` function. Beside, images paths, model and optional preprocessing function you have to pass class indexes for which explanation should be made (`NULL` means select class with highest probability for this image), some method specific arguments and if you want to generate grayscale o RGB explanation maps.

```
explanations <- generate_explanations(
  model,
  input_imgs_paths,
  preprocessing_function,
  class_index = NULL,
  methods = sauron_available_methods$method,
  num_samples = 5, # SmoothGrad samples
  noise_sd = 0.1, # SmoothGrad noise standard divination
  steps = 10, # Integrated Gradients steps
  grayscale = FALSE)
```

Now we can plot our results:

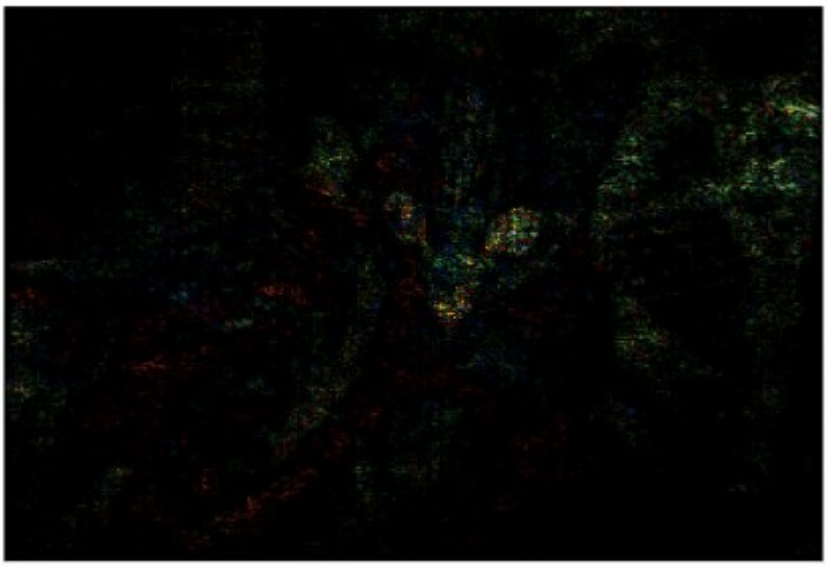
```
plot_explanations(explanations, FALSE)
# $Input
```


```
#
# $V
```




```
#
# $GI
```

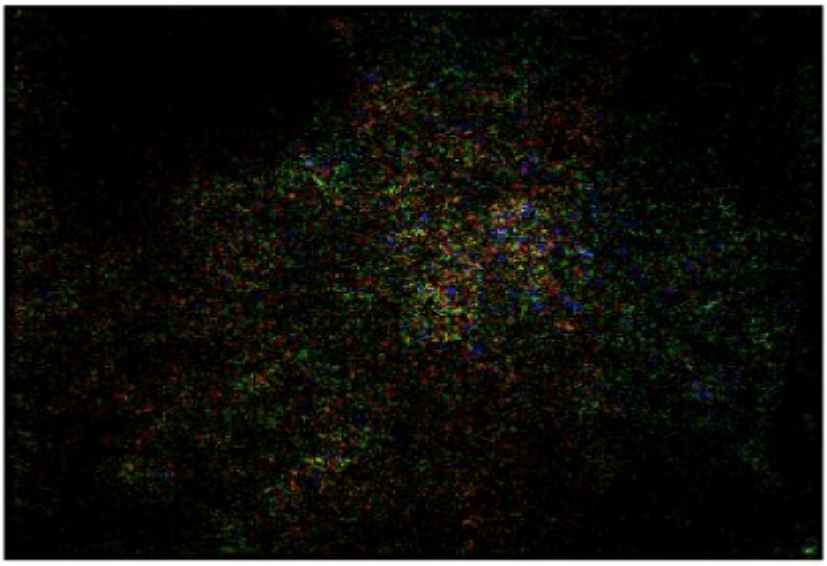
Gradient x Input



#

$SG

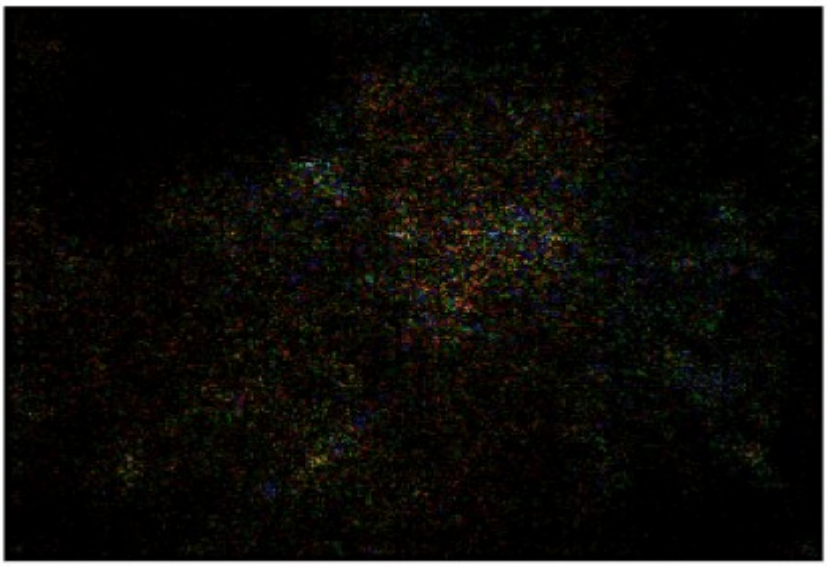
SmoothGrad



#

$SGI

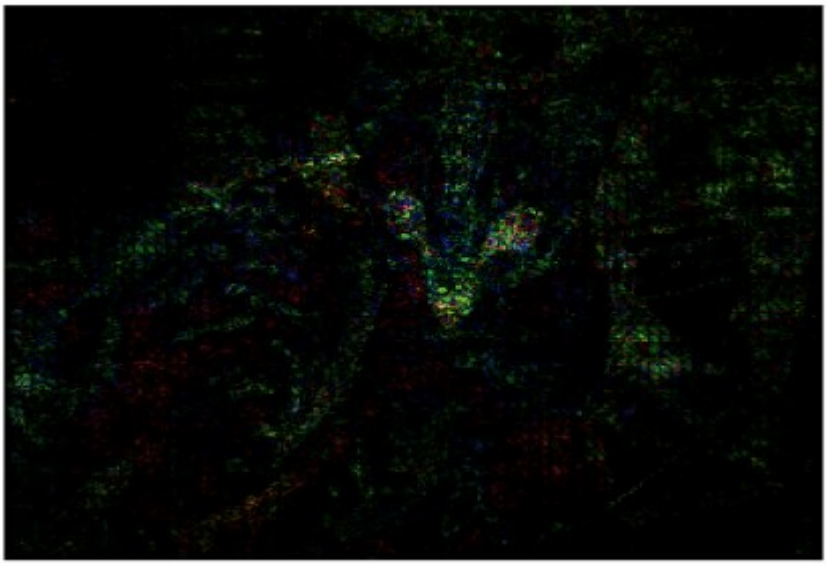
SmoothGrad x Input



#

$IG

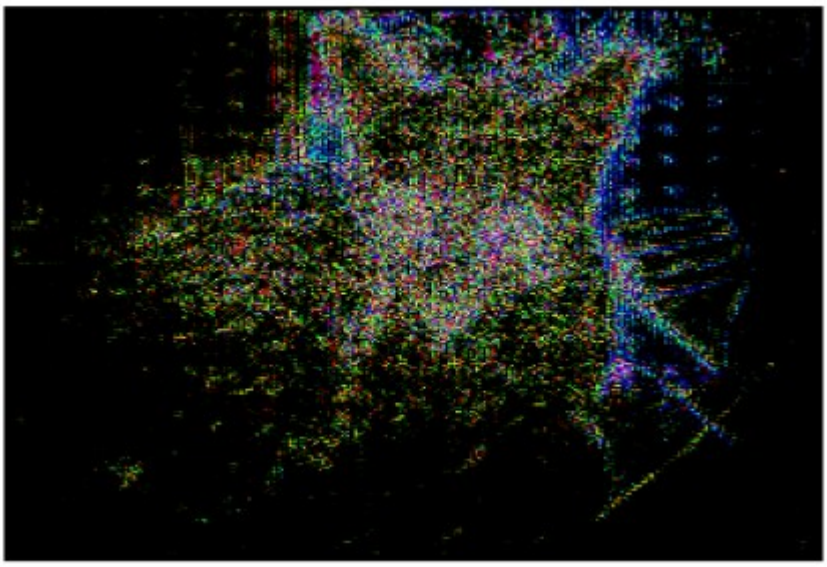
Integrated Gradients



#

$GB

Guided Backpropagation

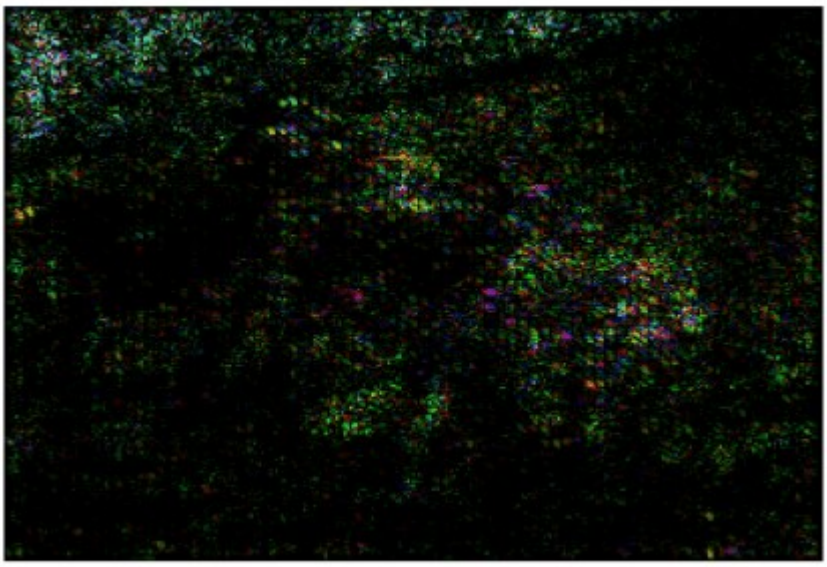


```
#
# $Input
```

Input

```
#
# $V
```

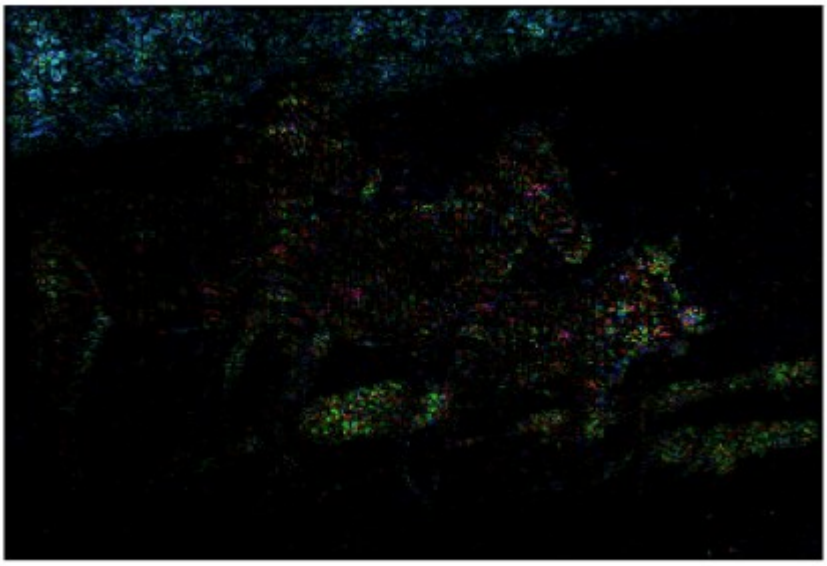
Vanilla gradient



#

$GI

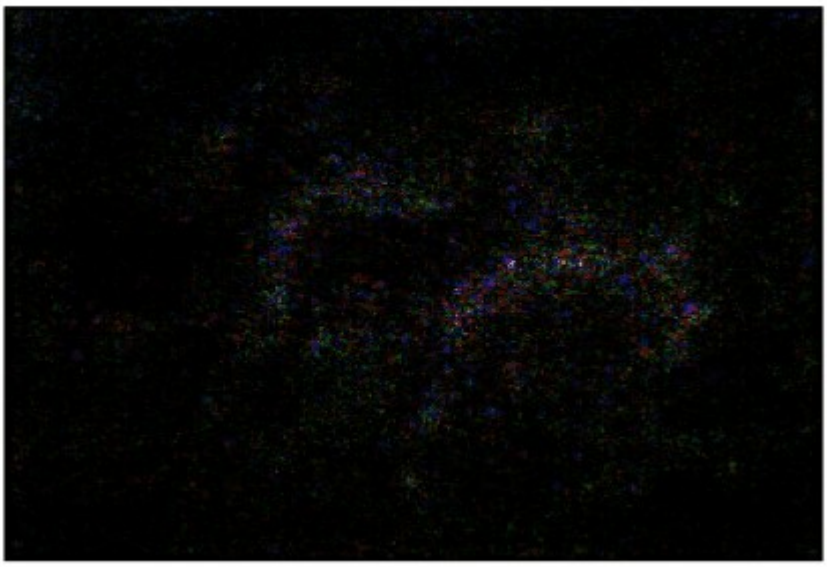
Gradient x Input



#

$SG

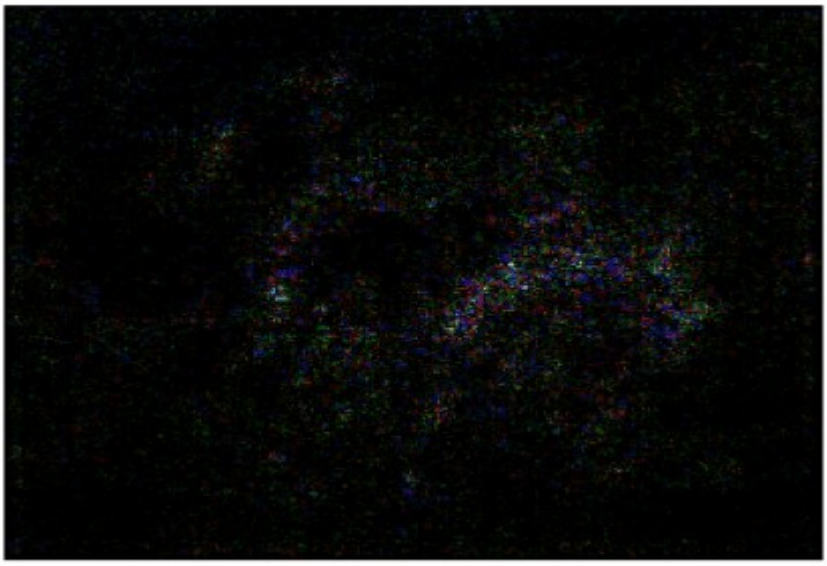
SmoothGrad



#

$SGI

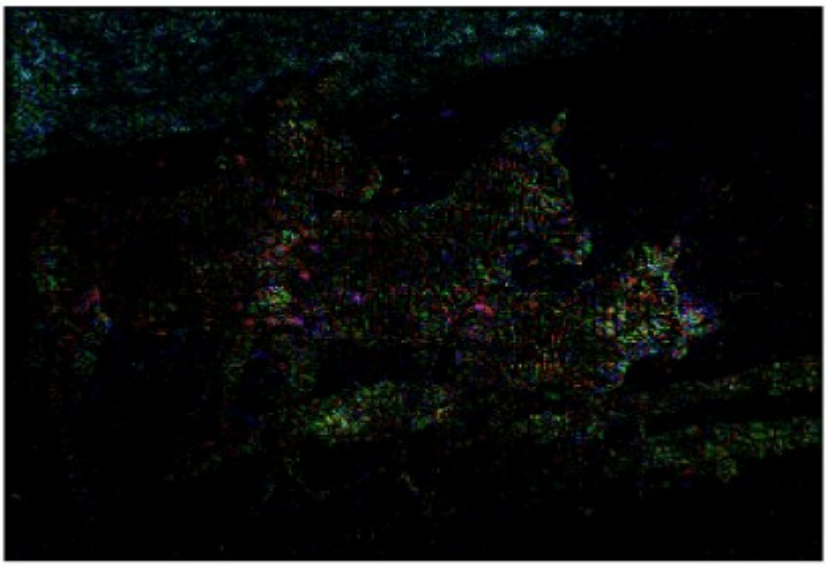
SmoothGrad x Input



#

$IG

Integrated Gradients



#

$GB

Guided Backpropagation