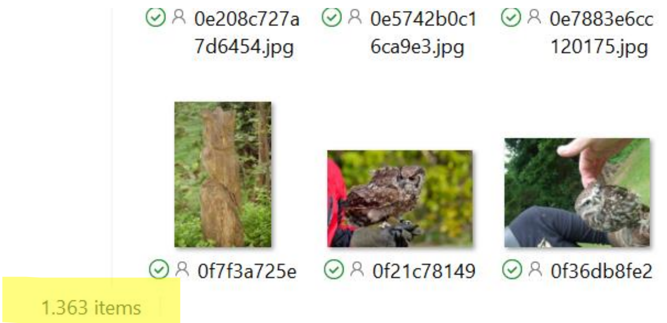
Luca Rijneveld 211271

Contents

Responsible and Explainable AI

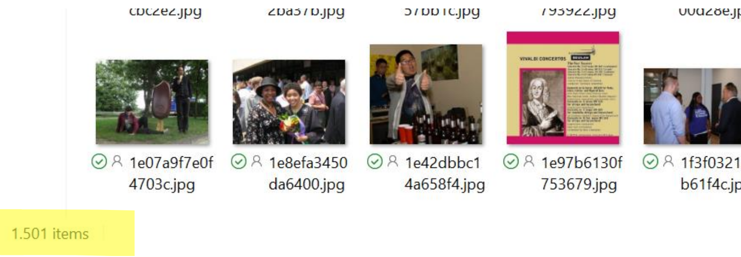
Poor:

My dataset has more images of men then owls. There are also owl images that show a person holding the owl with their face clearly in the picture.



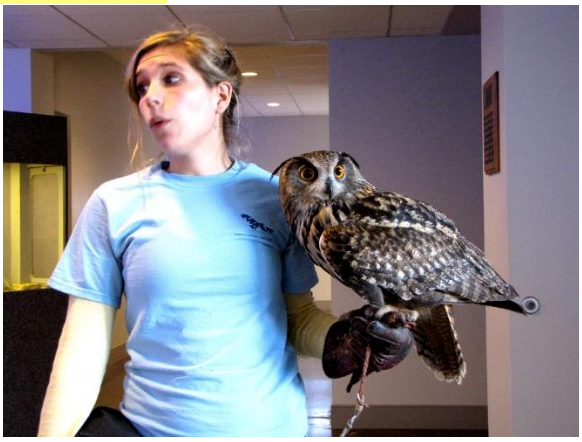
0e208c727a7d6454.jpg
0e5742b0c16ca9e3.jpg
0e7883e6cc120175.jpg
0f7f3a725e
0f21c78149
0f36db8fe2

1.363 items



1e07a9f7e0f4703c.jpg
1e8efa3450da6400.jpg
1e42dbbc14a658f4.jpg
1e97b6130f753679.jpg
1f3f0321b61f4c.jp

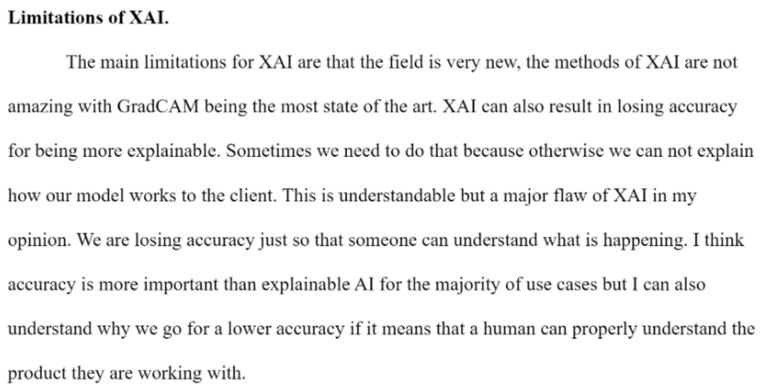
1.501 items



This is a form of bias as the algorithm will have a see a human in the picture of an owl. Something that will lower the model accuracy. One dataset is bigger than the other and this creates a bias because a model is training more on one of the classes compared to the other.

There are also images depicting owls that are not made out in the wild and sometimes they don't even feature a living owl. Art of owls and plushies of owls are also included in the data. They effect the way the model perceives an owl

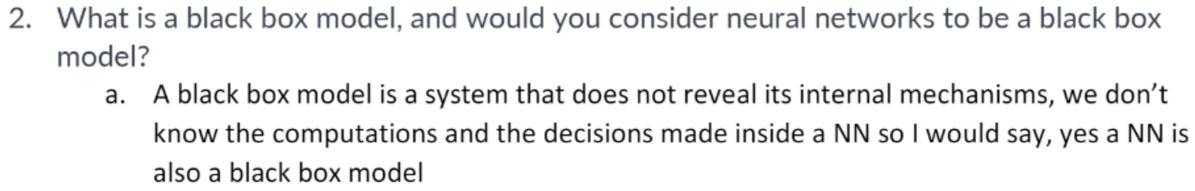
Insufficient:



Limitations of XAI.

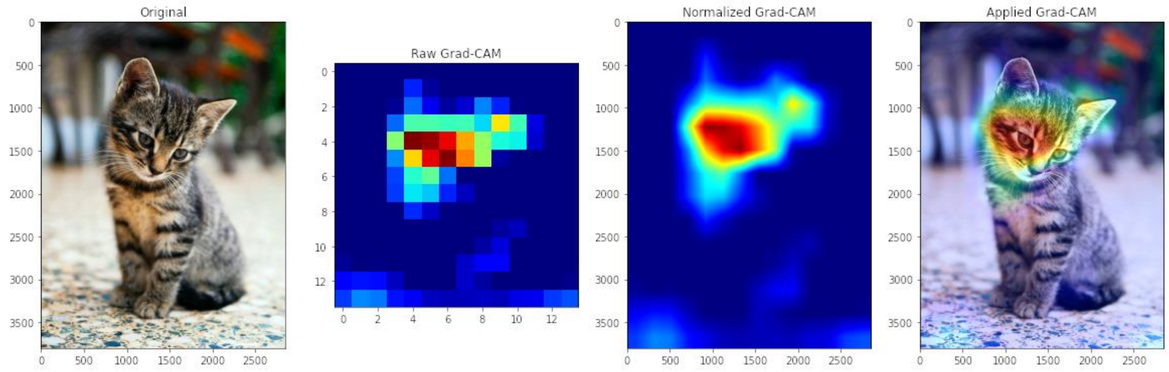
The main limitations for XAI are that the field is very new, the methods of XAI are not amazing with GradCAM being the most state of the art. XAI can also result in losing accuracy for being more explainable. Sometimes we need to do that because otherwise we can not explain how our model works to the client. This is understandable but a major flaw of XAI in my opinion. We are losing accuracy just so that someone can understand what is happening. I think accuracy is more important than explainable AI for the majority of use cases but I can also understand why we go for a lower accuracy if it means that a human can properly understand the product they are working with.

Other limitations applying to XAI is the fact that a NN is a black box model.



2. What is a black box model, and would you consider neural networks to be a black box model?
    a. A black box model is a system that does not reveal its internal mechanisms, we don't know the computations and the decisions made inside a NN so I would say, yes a NN is also a black box model

This means that we have no idea what is happening in terms of computations inside the NN. This is a limitation when it comes to transparency. We also don't know why a model picks certain features in our pictures as key features. Sometimes they might be obvious but, in some cases, a key feature for the AI can be a feature we as humans would not think about.

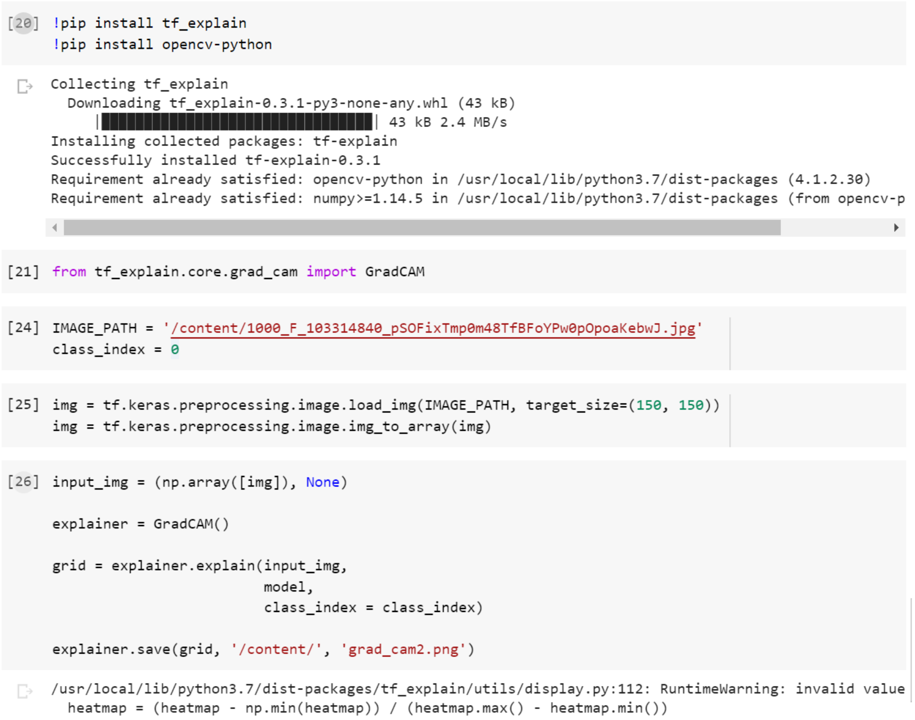


In the example image above for instance we see a cat. The model sees its eye as a very big key feature. The ears of the cat not that much. Humans however would see the pointy cat ears as a big identifying key feature.

Sufficient:

A way to fix some of the bias is making the datasets balanced. Currently there are 1500 images of human men and 1362 images of owls, by removing images from the human folder we make the datasets equal, removing a bit of bias. Having unequal datasets make the model bias, it will be better at identifying a human in a picture compared to an owl. By removing random images of the human dataset (random to avoid including my own bias in the deletion process) or by adding random images from the internet to our owl dataset we can make them equal. Now they both have an equal chance at getting identified by our model.

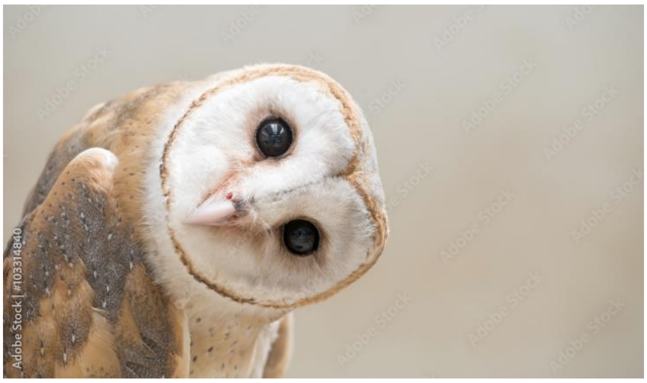
Good:



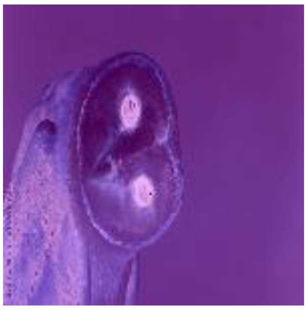
```
[20] !pip install tf_explain
     !pip install opencv-python

     Collecting tf_explain
       Downloading tf_explain-0.3.1-py3-none-any.whl (43 kB)
          |████████████████████████████████| 43 kB 2.4 MB/s
     Installing collected packages: tf-explain
     Successfully installed tf-explain-0.3.1
     Requirement already satisfied: opencv-python in /usr/local/lib/python3.7/dist-packages (4.1.2.30)
     Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.7/dist-packages (from opencv-p

[21] from tf_explain.core.grad_cam import GradCAM

[24] IMAGE_PATH = '/content/1000_F_103314840_pSOFixTmp0m48TfBFoYPw0pOpoaKebwJ.jpg'
     class_index = 0

[25] img = tf.keras.preprocessing.image.load_img(IMAGE_PATH, target_size=(150, 150))
     img = tf.keras.preprocessing.image.img_to_array(img)

[26] input_img = (np.array([img]), None)

     explainer = GradCAM()

     grid = explainer.explain(input_img,
                              model,
                              class_index = class_index)

     explainer.save(grid, '/content/', 'grad_cam2.png')

     /usr/local/lib/python3.7/dist-packages/tf_explain/utils/display.py:112: RuntimeWarning: invalid value
       heatmap = (heatmap - np.min(heatmap)) / (heatmap.max() - heatmap.min())
```

As you can see in my code, I installed tf_explain and implemented GradCAM into my model.

This is the picture I used:



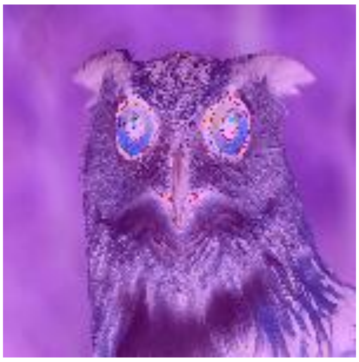
This was the explanation my model gave me:



The main thing you can see is that the eyes light up. Something that I also found interesting is the beak of the owl. It is something that I would see as an important feature, but the model apparently does not. So, a limitation here is that we don't know why the eyes are bright and the beak is not, we don't know why the model sees this as an important and unimportant feature. GradCAM shows me that the model classified my images as an Owl probably because of the eyes in the image.

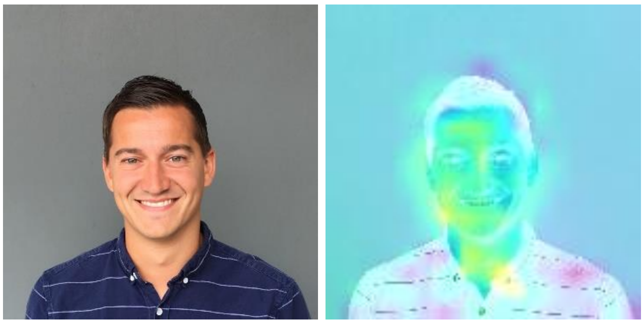
Another owl image:

GradCAM explanation:

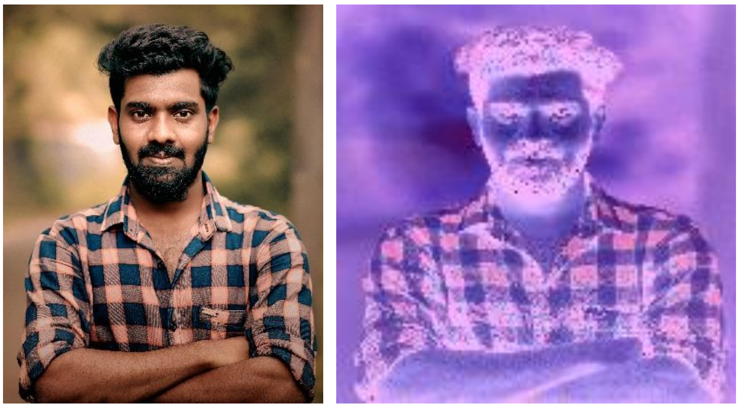


This one highlights the eyes even more, also the mouth a bit.

Now I also have some images of a man



We can see that this image is highlighted a lot, it also shows a very bright background which tells us something about our models' accuracy. I would say that there is some bias in our model regarding background and features of a man. This bias becomes more apparent when we look at our next image of a man.



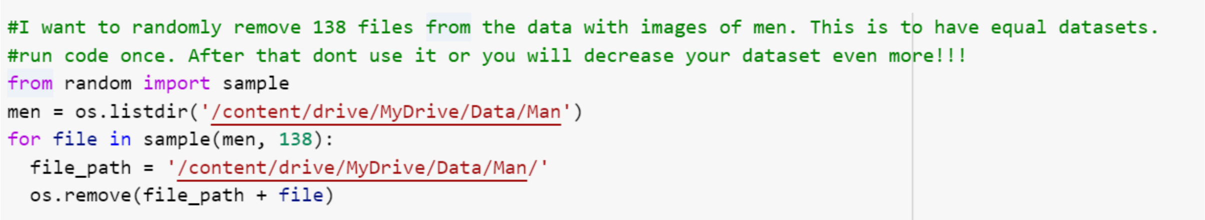
This GradCAM image was shocking to me. The two men in both pictures are in almost the same pose, yet the model had a hard time predicting and finding key features in this image. This could be explained with bias in our dataset.

Excellent:

For the excellent condition I will be applying the fairness method I described in my sufficient condition. I will make both datasets equal by randomly removing images form my image dataset with men.

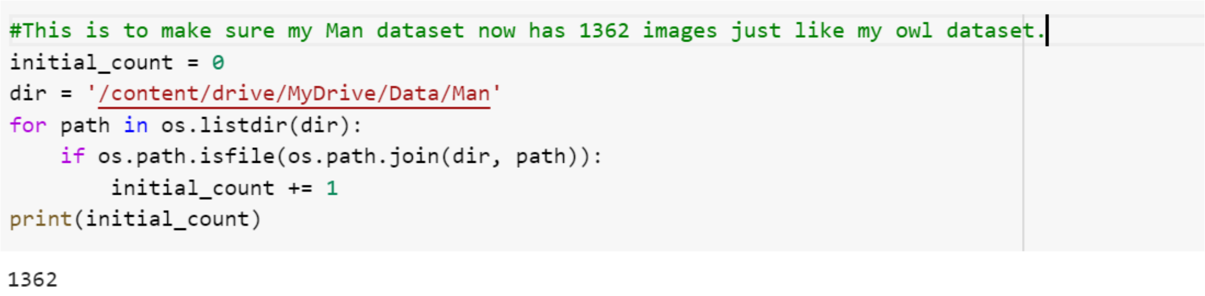
In the code for my model, I included a code block that randomly remove 138 images out of the men dataset. My owls have 1362 images and my men had 1500 images. 1500 – 1362 = 138. So to make the datasets equal I decided to remove 138 images making my men dataset of equal size to the owl dataset

Code:



```
#I want to randomly remove 138 files from the data with images of men. This is to have equal datasets.
#run code once. After that dont use it or you will decrease your dataset even more!!!
from random import sample
men = os.listdir('/content/drive/MyDrive/Data/Man')
for file in sample(men, 138):
  file_path = '/content/drive/MyDrive/Data/Man/'
  os.remove(file_path + file)
```

This code randomly removes the 138 images



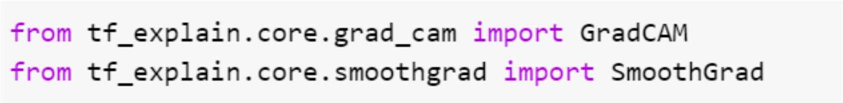
```
#This is to make sure my Man dataset now has 1362 images just like my owl dataset.
initial_count = 0
dir = '/content/drive/MyDrive/Data/Man'
for path in os.listdir(dir):
    if os.path.isfile(os.path.join(dir, path)):
        initial_count += 1
print(initial_count)
```

```
1362
```

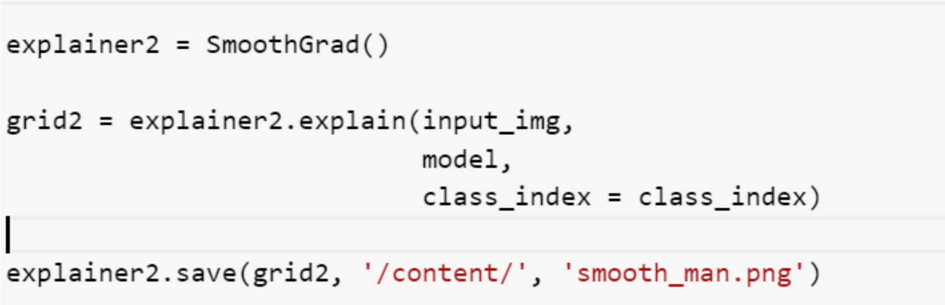
This code prints the number of images in my folder for Men and shows that it counts 1362 images. Same number of images as my Owl dataset (See picture in poor condition)

Important to note is that you should only run this code once, otherwise you will remove another 138 randomly selected images from the men dataset resulting in unequal sets.

I also add another XAI method, I tried to implement SmoothGrad



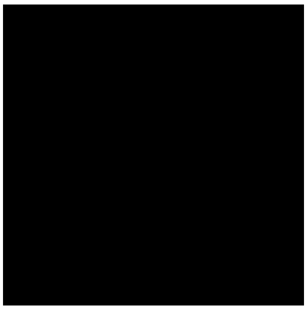
```
from tf_explain.core.grad_cam import GradCAM
from tf_explain.core.smoothgrad import SmoothGrad
```



```
explainer2 = SmoothGrad()

grid2 = explainer2.explain(input_img,
                           model,
                           class_index = class_index)

explainer2.save(grid2, '/content/', 'smooth_man.png')
```

Input image:

SmoothGrad:



As you can see the image is entirely black, so the fact that I get a black image means that SmoothGrad is probably working. But an entirely black image also means that there are no keyfeatures my model can find and there for is probably not that accurate at predicting this image.