Practical 4 - Trustworthy AI: Interpretability, Explainability and Uncertainty

Team number: 21
Team members: Emil Alizada, Sofiene Boutaj, Felix Hsieh, Daniil Morozov

Task 1 - Interpretability and Explainability

1a: *Using CAM-based methods In this task you are provided with a general template for the use of CAM-based methods for common imaging datasets. The task requires you to replicate these methods and analyse your results on the MedMNIST dataset (you are free to select any task from this collection of datasets). You do not get evaluated on the completeness of your implementation, but rather on the quality of your analysis. Here is the to the tutorial: CAM-Explanations*
You can use the CAM implementation provided in PyTorch-CAM or use one of the alternatives such as Captum. Experiment with different CAM-based methods on the given datasets in order to identify the best one. Here the definition of the "best one" is subjective: You are expected to use between 3 and 5 methods, record the images produced along with the metrics associated with the model (e.g. class-specific accuracy). Your results should be recorded in a table which describes how you concluded that a specific method outperforms the rest (e.g. performance, similarity to how you would explain the image etc.) Be creative with your choice of metrics.
The expected deliverables are: Short (under 1 page) report on which datasets and models you considered; a table with ranking of explanation methods; figures demonstrating ROIs of a few exemplary images.
N.B. Addition of more datasets is highly encouraged and can contribute towards a more objective evaluation of the explanation method.

For the first part of the assignment, we chose the pneumonia MNIST dataset, which contains images of two classes: 0 - normal, and 1 - pneumonia. We divided the original data into train, validation, and a test set consisting of 4708, 524, and 624 images respectively. The example of each class is presented in figure 1.

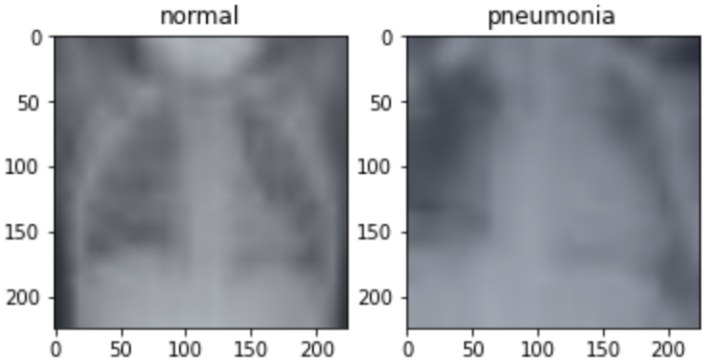

Fig.1 - Samples of each class

For analyzing the dataset resnet-18 model was chosen. Before training the model we implemented the preprocessing, which included resizing the image from 28x28 to 224x224,

which is the original image size for the aforementioned model, normalization, and expansion of the number of channels to 3. We trained our model for 15 epochs. The final accuracy on the validation set was equal to 97.46.

For the model's explainability evaluation we used 5 different methods and 4 metrics from PyTorch-CAM. The results are presented in table 1.

Table 1 - Methods evaluation table

	GradCAM	GradCAM++	EigenGradCAM	AblationCAM	RandomCAM
MultImage	-0.6716 rise: 0 drop: 1	-0.6887 rise: 0 drop: 1	-0.6716 rise: 0 drop: 1	-0.6032 rise: 0 drop: 1	-0.6059 rise: 0 drop: 1
ROAD MORF Average	-1.0276	-1.0276	-1.0276	-1.0276	-1.0276
ROAD LERF Average	0.2886	0.2836	0.0310	-0.0575	0.2164
ROAD combined Average	0.00041	0.00199	0.00042	0.00162	0.00063

Let's discuss the results presented in the table in more detail. Each number represents the change in confidence for the prediction of pneumonia class. As an argument for every method, we passed one image of each class. As the "normal" image is uninformative for the analysis, only scores for the "pneumonia" image are presented. The image was randomly sampled from the validation set.

In order to choose the best explanation method let's compare the metrics:

1. MultImage: The method extracts the CAM and multiplies it with an input image which leads to several patterns deleting. It measures the confidence change between the prediction on the original image and the modified one. We can see that each method shows a decrease in the model's confidence. This drop can be explained by the fact that probably some less important parts of the image, which still influence classification, got blurred due to the aforementioned multiplication. AblationCAM shows the lowest drop in confidence while GradCAM++ - has the highest.

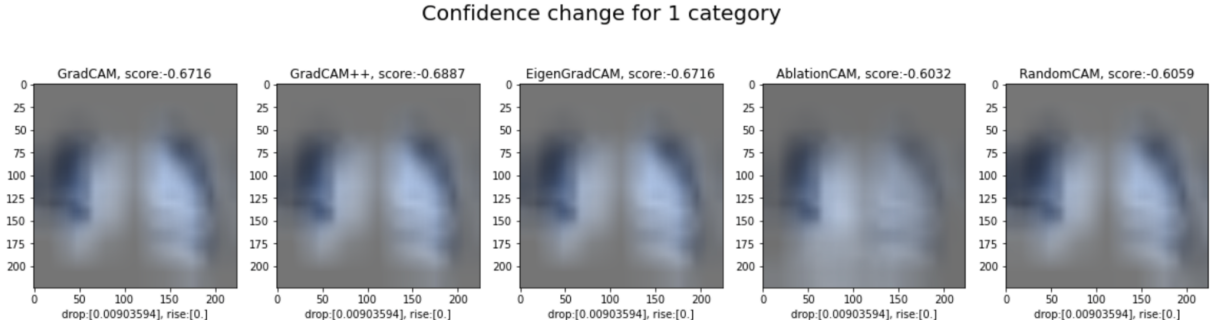

Fig.2 - MultImage visualization

2. ROAD MORF average: Removes the most relevant pixel first and replaces them with the weighted average of its neighbors. We did it for different percentiles [20, 40, 60, 80] of CAM and then took the average. We observe that each method shows a decrease in confidence as zones of the image, relevant to accurate pneumonia prediction, were blurred.

3. ROAD LERF: Removes the least relevant pixel first and replaces them with the weighted average of its neighbors. The same percentiles were used for calculating the average score. This metric is positive for every method, except AblationCAM. As the goal is to increase the ROAD LERF score, it plays against choosing this method as the best.

4. ROAD combined: This metric equals (MORF - LERF) / 2. It is convenient for assessing explanatory power, as it combines the two previous metrics and produces only one score. The highest scores were achieved with GradCAM++ and AblationCAM.

Based on the presented scores the best method for the model's explainability evaluation is GradCAM++. Though the first metric's score is low in comparison to AblationCAM, each of the chosen methods shows a confidence decrease, which theoretically should be the opposite. ROAD MORF is negative and the same for every method, so we couldn't consider this metric for method evaluation. ROAD LERF for GradCAM++ is the second after standard GradCAM. ROAD combined is also the highest for GradCAM++. Moreover, figure 3 shows smooth and meaningful CAM for it.

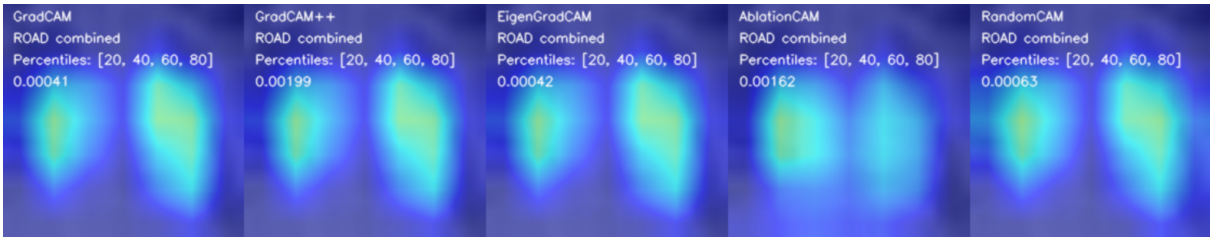

Fig.3 - ROAD combined score visualization

1b: *Analysing the feature selection*
As training progresses, your model is (hopefully) learning more representative features that can explain its predictions. Analyse the predictions made by your model at the start of training and compare to the ones made later in the training process. Do you notice a substantial difference between the features that are selected for explanations? At what point do you think you can stop the training because the features look "good enough"?
For this part we expect a short comment (1-2 paragraphs is enough) on the relative change in explanation quality associated with your selected learning task. You are allowed to re-use the previous part to simplify the implementation.

For explainability evaluation during training we used the GradCAM method. As the training starts, in the normal images we can see that after epoch 5, even though the accuracy increases while validation loss decreases, there isn't much difference in the visual feature representation - explainability stabilizes. Analysis of CAM of normal image is uninformative due to absence of certain

patterns to focus on, so let's have a look at pneumonia images. Similarly in images with pneumonia, after epoch 8, the same situation happens. We can see that the algorithm was able to differentiate spatial features that help to distinguish between normal and pneumonia images. We can notice that in pneumonia images the algorithm highlighted left and right parts, which contain specific patterns related to the aforementioned disease. As we can see in the following image, characteristics of pneumonia are noticeable within the sections of the lungs of the patient, therefore, we would expect the algorithm to highlight lungs as the relevant section for disease identification.

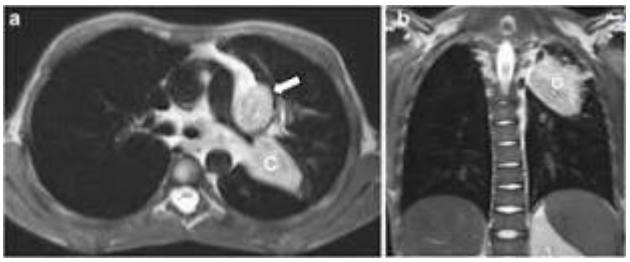

Fig.4 - MRI of pneumonia patient

In conclusion, it's possible to stop training the net after the 8th epoch, as the CAM doesn't show any changes.

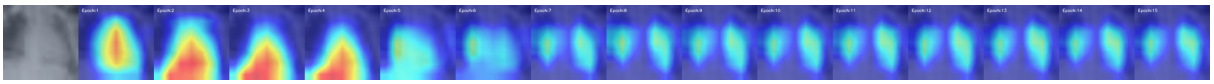

Fig.5 - CAM on pneumonia image

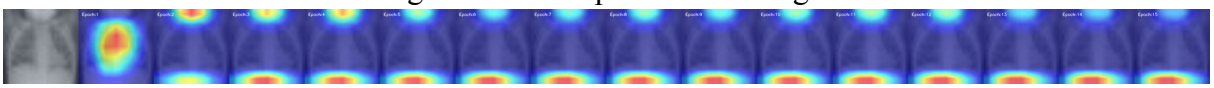

Fig.6 - CAM on normal image

Task 2 - Using MC Dropout to estimate uncertainty:

This is a part of the practical exercise on uncertainty. Your task is to go through the tutorial and fill in the missing code and answer the questions.

Q1: We just transformed the labels using Lambda function for defining the DataClass objects. Any guess why we did that?

target_transform requests a function for transforming the targets (labels) of the data. The Lambda function extracts the label of a sample from a single value array ([y] to y). Lambda is used because it is an easy way to define a function.

An alternative implementation would be writing a method:

```
def test(y):
 return y[0]

train_set = DataClass(split='train', transform=data_transform, target_transform=test, download=True)
```

Q4: Is there a catch? Please write your thoughts on why we do not see a distribution and whether we could fix it somehow.

The batchsize is 3 and for each position in the batch the classification is different because the sample at the batchposition is a different image. (3 different outcomes)

Possible reason: the model that should use dropout is still in eval mode, so it wont use dropout and the result is deterministically the same for the same input sample. The for loop for range(sample) also doesn't iterate over the images but always forwards the same sample to the model.

Fix: change the model mode to train to use dropout and receive different results for the same input image

***For the code sections tasks there are python comments that explain the code lines**

References

https://link.springer.com/article/10.1007/s00247-017-3865-2