The Academic College of Tel Aviv-Yaffo School of Computer Science

XAI

Explainable Artificial Intelligence

Image Classification Explainability Research

Or Izchak

Under the supervision of Prof. Moti Schneider

Apr 28, 2022

Acknowledgments

I would like to express my profound gratitude to my supervisor, Prof. Moti Schneider, of the Computer Science School at The Academic College of Tel Aviv-Yaffo.

Prof. Moti Schneider provided valuable encouragement, a positive attitude, and meaningful guidance during the research process. The completion of this thesis would not have been possible without his dedication, constant availability, and professional support as I refined my research.

Finally, I want to take this opportunity to express my sincere gratitude to my family, for providing me unfailing support and continuous encouragement during my years of study and writing this thesis.

Thank you

Contents

1. Chapter 1 - Introduction 4
   1.1 Problem Definition 4
   1.1.1 Overview 4
   1.1.2 What is XAI? 5
   1.1.3 What is the problem? 6
1.2 Literature Survey 7
   1.2.1 Global Interpretability 7
   1.2.2 Local Interpretability 8
   1.2.3 Current Community Tools 9
1.3 Solution – Short Description – ModuleX 12
   1.4 Next Chapters 13
   2. Chapter 2 - XAI System 14
   2.1 ModuleX Description 14
   2.3 ModuleX Architecture and Data 15
   2.4 Experiments steps 16
3 Chapter 3 - Case Study 17
   3.3 Results – Accuracy and Loss 17
   3.4 Results - Images 18
   3.5 Explain - Local Interpretability 19
   3.6 Explain - Global Interpretability 20
   3.6 Explain - Mistakes 21
4. Chapter 4 – Conclusion 22
4.1 Summary 22
4.2 Future Research 23
5. Chapter 5 - References 24

1. Chapter 1 - Introduction

1.1 Problem Definition

1.1.1 Overview

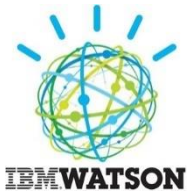
In the era of data science, artificial intelligence is making impossible feats possible. Driverless cars, IBM Watson's question-answering system, cancer detection, electronic trading, etc. are all made possible through the advanced decision-making ability of artificial intelligence.

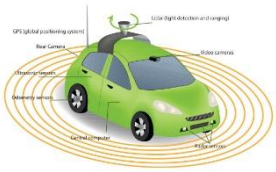
The **deep layers of neural networks** have a magical ability to recreate the human mind and its functionalities. When humans make decisions, they can explain their thought process behind them. They can explain the rationale, whether it is driven by observation, intuition, experience, or logical thinking ability.

Basic ML algorithms like decision trees can be explained by following the tree path which led to the decision. But, when it comes to complex AI algorithms, the deep layers are often incomprehensible by human intuition and are quite opaque.

Data scientists may have trouble explaining why their algorithm gave a specific decision and the layman end-user may not simply trust the machine's predictions without contextual proof and reasoning.

This paper will show a new algorithm for explaining the image classification of a deep neural network.





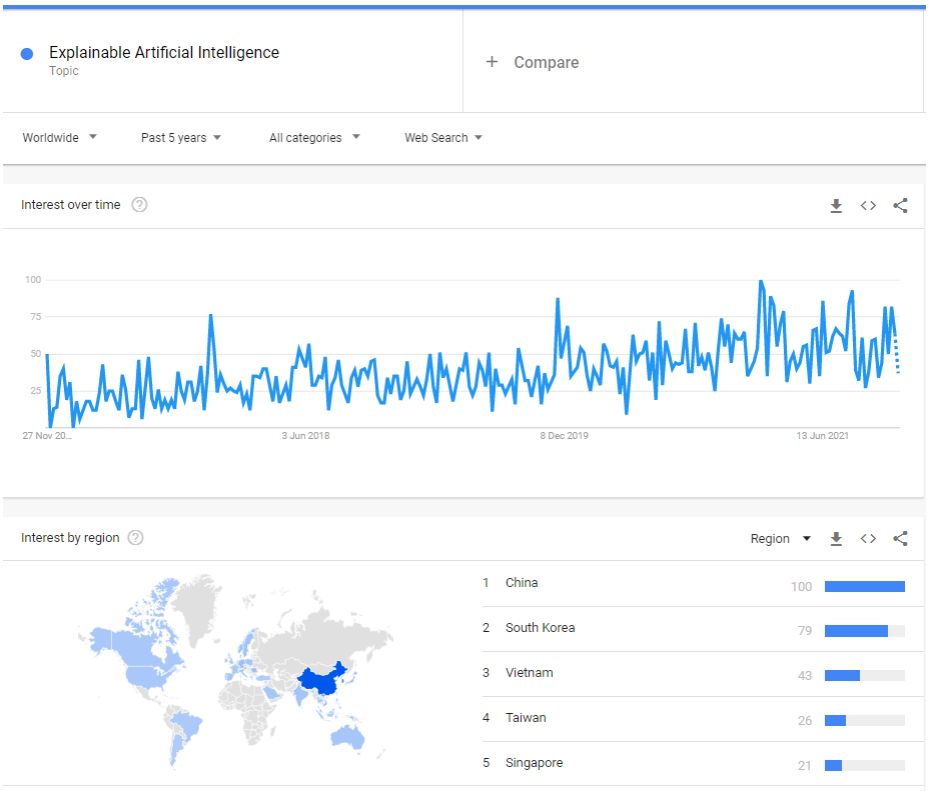
1.1.2 What is XAI?

Explainable AI (XAI) refers to methods and techniques in the application of artificial intelligence technology (AI) such that the results of the solution can be understood by humans. It contrasts with the concept of the "black box" in machine learning where even their designers cannot explain why the AI arrived at a specific decision. XAI may be an implementation of the social right to explanation. XAI is relevant even if there are no legal rights or regulatory requirements, for example, XAI can improve the user experience of a product or service by helping end users trust that the AI is making good decisions.

The technical challenge of explaining AI decisions is sometimes known as the **interpretability problem**. Another consideration is infobesity (overload of information), thus, full transparency may not be always possible or even required. However, simplifying at the cost of misleading users to increase trust or hide undesirable attributes of the system should be avoided by allowing a tradeoff between interpretability and completeness of an explanation.

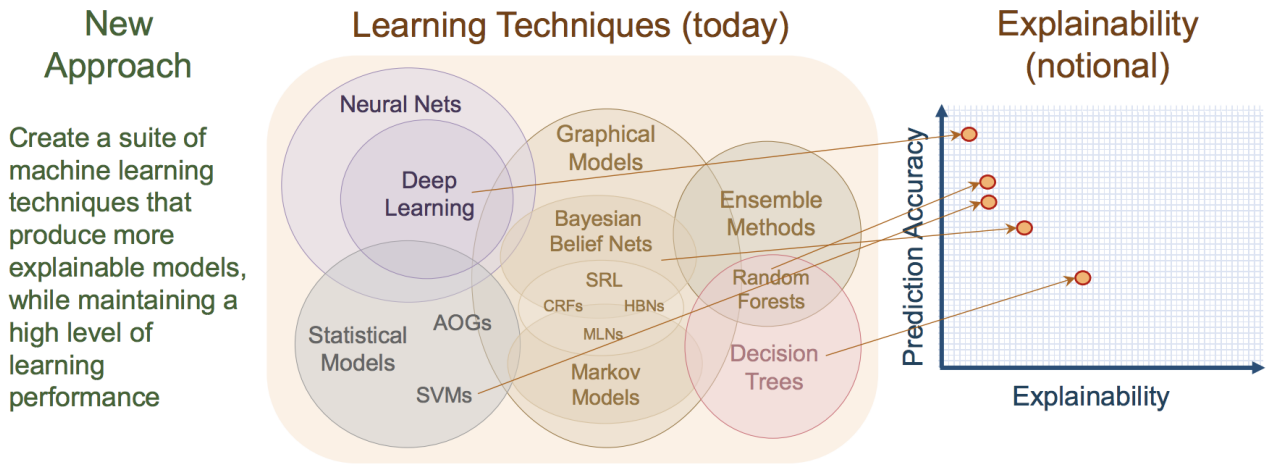
AI systems optimize behavior to satisfy a mathematically specified goal system chosen by the system designers, such as the command "maximize the accuracy of assessing how positive film reviews are in the test dataset". The AI may learn useful general rules from the test set, such as "reviews containing the word 'horrible' are likely to be negative". However, it may also learn inappropriate rules, such as "reviews containing 'Daniel Day-Lewis' are usually positive"; such rules may be undesirable if they are deemed likely to fail to generalize outside the test set, or if people consider the rule to be "cheating" or "unfair". A human can audit rules in an XAI to get an idea of how likely the system is to generalize to future real-world data outside the test set.

The field of XAI started several years ago. By looking at google trends at 'Explainable AI' we can see that it started to be in the interest of people in 2015 and it is in growth.



1.1.3 What is the problem?

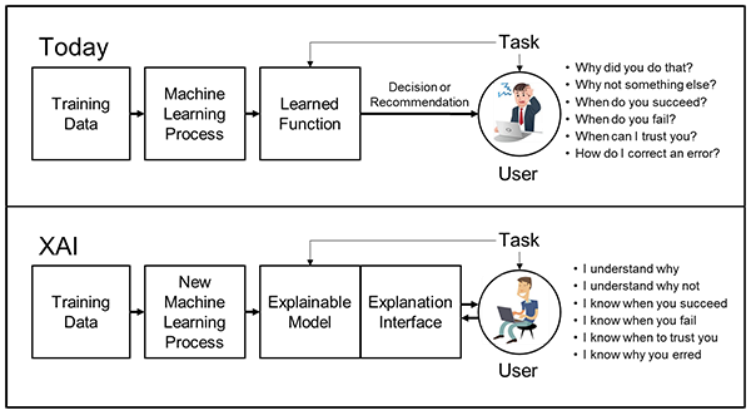
The more complex the model, gives high the Accuracy and less Explainability, as shown below.



There are a lot of from this kind of problem that has a huge sensitive impact for the people who are using these algorithms that involves human lives.
Examples:

- **Cancer prediction** - The algorithm decides whether a roentgen image represents cancer or not and reports that to the doctor. Then the doctor would like to know why the algorithm decided that (base on which parts of the image).
- **Autonomous cars** – The algorithm needs to decide whether to stop the car or not if it recognized a human in front of the car to avoid accidents.
- **Police** - The algorithm needs to decide whether a person is more likely to commit a crime or an algorithm that matches an identikit to suspects.
- **Battlefield** – The algorithm needs to decide whether to shoot on the enemy or an algorithm that tells the soldier what the safest way is to walk etc...

This leads the industry towards solutions to explain these magical models.



1.2 Literature Survey

There are two types of XAI approaches Global and Local interpretability.

1.2.1 Global Interpretability

How does the trained model make predictions?

You could describe a model as interpretable if you can comprehend the entire model at once (Lipton 20167). To explain the global model output, you need the trained model, knowledge of the algorithm, and the data. This level of interpretability is about understanding how the model makes decisions, based on a holistic view of its features and each of the learned components such as weights, other parameters, and structures.

Which features are important and what kind of interactions between them take place?

Global model interpretability helps to understand the distribution of your target outcome based on the features. Global model interpretability is very difficult to achieve in practice.

Any model that exceeds a handful of parameters or weights is unlikely to fit into the short-term memory of the average human. I argue that you cannot imagine a linear model with 5 features, because it would mean drawing the estimated hyperplane mentally in a 5-dimensional space.

Any feature space with more than 3 dimensions is simply inconceivable for humans.
Usually, when people try to comprehend a model, they consider only parts of it, such as the weights in linear models.

How do parts of the model affect predictions?

A Naive Bayes model with many hundreds of features would be too big for me and you to keep in our working memory. And even if we manage to memorize all the weights, we would not be able to quickly make predictions for new data points.

In addition, you need to have the joint distribution of all features in your head to estimate the importance of each feature and how the features affect the predictions on average. An impossible task. But you can easily understand a single weight.

While global model interpretability is usually out of reach, there is a good chance of understanding at least some models on a modular level. Not all models are interpretable at a parameter level.

For linear models, the interpretable parts are the weights, for trees it would be the splits (selected features plus cut-off points) and leaf node predictions.

Linear models, for example, look as if they could be perfectly interpreted on a modular level, but the interpretation of a single weight is interlocked with all other weights. The interpretation of a single weight always comes with the footnote that the other input features remain at the same value, which is not the case with many real applications.

A linear model that predicts the value of a house, that considers both the size of the house and the number of rooms, can have a negative weight for the room feature. It can happen because there is already the highly correlated house size feature.

In a market where people prefer larger rooms, a house with fewer rooms could be worth more than a house with more rooms if both have the same size.

The weights only make sense in the context of the other features in the model. But the weights in a linear model can still be interpreted better than the weights of a deep neural network.

1.2.2 Local Interpretability

Why did the model make a certain prediction for an instance?

You can zoom in on a single instance and examine what the model predicts for this input and explain why.

If you look at an individual prediction, the behavior of the otherwise complex model might behave more pleasantly. Locally, the prediction might only depend linearly or monotonically on some features, rather than having a complex dependence on them.

For example, the value of a house may depend nonlinearly on its size. But if you are looking at only one particular 100 square meters house, there is a possibility that for that data subset, your model prediction depends linearly on the size. You can find this out by simulating how the predicted price changes when you increase or decrease the size by 10 square meters.

Local explanations can therefore be more accurate than global explanations.

Why did the model make specific predictions for a group of instances?

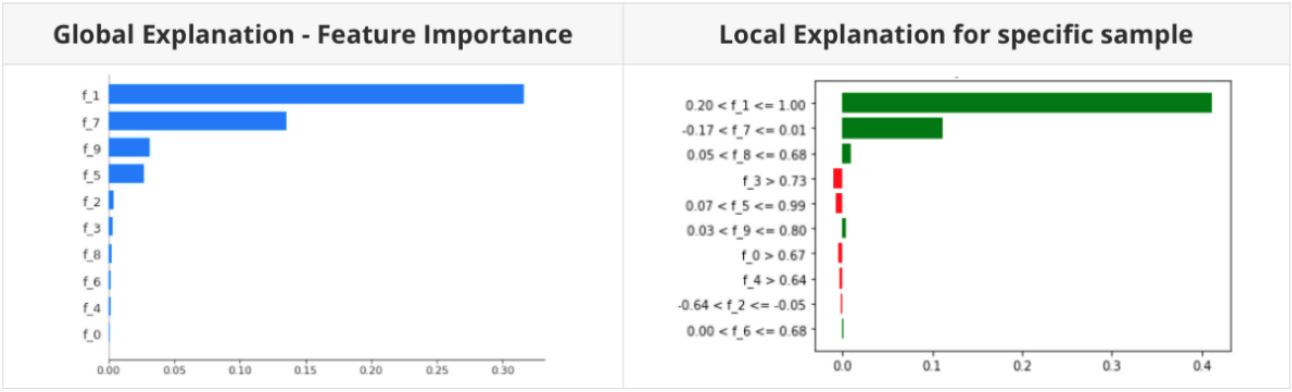
Model predictions for multiple instances can be explained either with global model interpretation methods (on a modular level) or with explanations of individual instances.

The global methods can be applied by taking the group of instances, treating them as if the group were the complete dataset, and using the global methods with this subset.

The individual explanation methods can be used on each instance and then listed or aggregated for the entire group.

1.2.3 Global vs Local Interpretability

The global methods can be applied by taking the group of instances, treating them as if the group were the complete dataset, and using the global methods with this subset.
The individual explanation methods can be used on each instance and then listed or aggregated for the entire group.



Examples of global and local explanations.

1.2.4 Current Community Tools

There are few libraries developed by the community that can help interpret black-box models of deep neural networks. Here are few popular examples:

1. **LIME** – (local interpretable model-agnostic explanations) - for local interpretability only, very popular library.

   Local surrogate models are interpretable models that are used to explain individual predictions of black-box machine learning models. Local interpretable model-agnostic explanations (LIME) is a paper in which the authors propose a concrete implementation of local surrogate models. Surrogate models are trained to approximate the predictions of the underlying black-box model. Instead of training a global surrogate model, LIME focuses on training local surrogate models to explain individual predictions.

   The idea is quite intuitive. First, forget about the training data and imagine you only have the black-box model where you can input data points and get the predictions of the model. You can probe the box as often as you want. Your goal is to understand why the machine learning model made a certain prediction.

   LIME tests what happens to the predictions when you give variations of your data into the machine learning model. LIME generates a new dataset consisting of permuted samples and the corresponding predictions of the black-box model. On this new dataset LIME then trains an interpretable model, which is weighted by the proximity of the sampled instances to the instance of interest.

   The interpretable model can be, for example, Lasso or a decision tree. The learned model should be a good approximation of the machine learning model predictions locally, but it does not have to be a good global approximation.

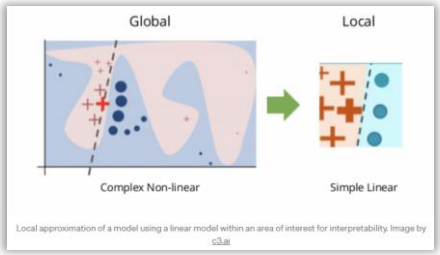
   This kind of accuracy is also called local fidelity.

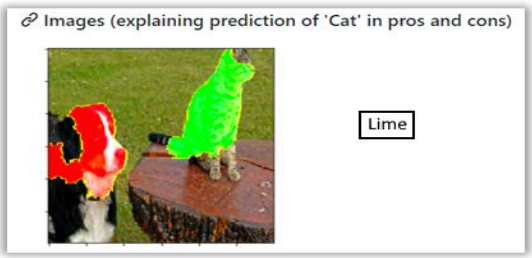
   LIME works for Tabular Data, Text, Images.

   LIME for images works differently than LIME for tabular data and text. Intuitively, it would not make much sense to perturb individual pixels, since more than one pixel contributes to one class. Randomly changing individual pixels would probably not change the predictions by much.

   Therefore, variations of the images are created by segmenting the image into "Super Pixels" and turning them off or on. “Super Pixels” are interconnected pixels with similar colors and can be turned off by replacing each pixel with a user-defined color such as gray.

   The user can also specify a probability for turning off a “Super Pixels” in each permutation.





2. **SHAP** - for local and global interpretability.

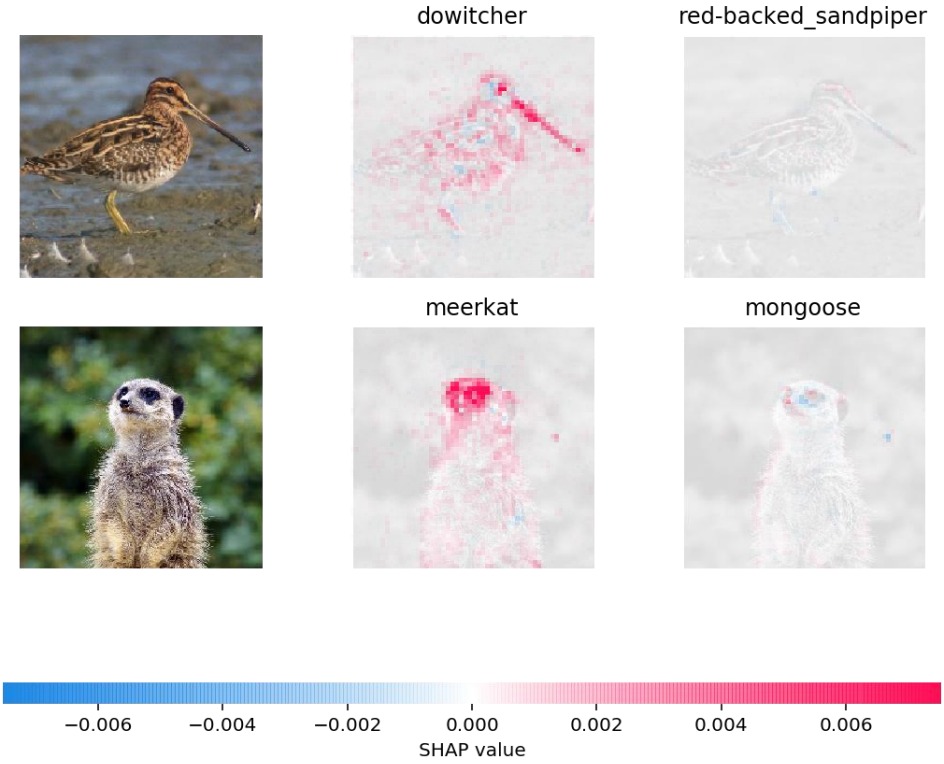
A prediction can be explained by assuming that each feature value of the instance is a "player" in a game where the prediction is the payout. Shapley values, a method from coalitional game theory, tells us how to fairly distribute the "payout" among the features.

The effect of each feature is the weight of the feature times the feature value.
This only works because of the linearity of the model. For more complex models, we need a different solution.

Another solution comes from cooperative game theory: The Shapley value, coined by Shapley (1953), is a method for assigning payouts to players depending on their contribution to the total payout. Players cooperate in a coalition and receive a certain profit from this cooperation.

The "game" is the prediction task for a single instance of the dataset. The "gain" is the actual prediction for this instance minus the average prediction for all instances. The "players" are the feature values of the instance that collaborate to receive the gain (= predict a certain value).

The Shapley value is the average marginal contribution of a feature value across all possible coalitions.
The computation time increases exponentially with the number of features. One solution to keep the computation time manageable is to compute contributions for only a few samples of the possible coalitions.

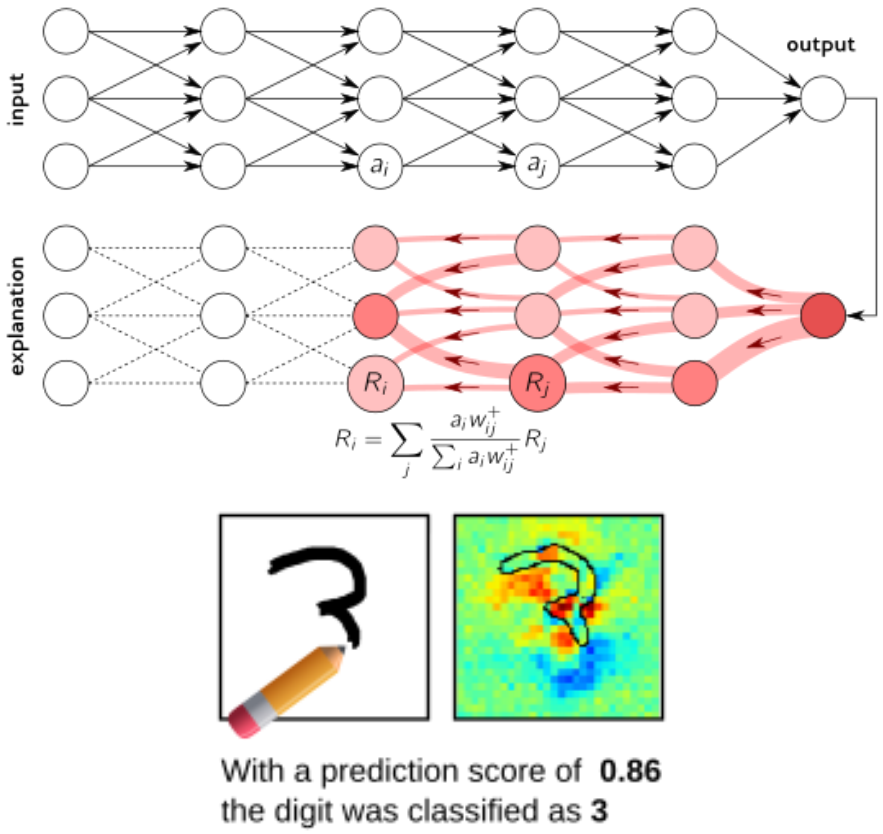


3. **Heatmapping** – for local interpretability

The project studies techniques to decompose the prediction in terms of contributions of individual input variables such that the produced decomposition (i.e. explanation) can be visualized in the same way as the input data.

It is based on the Layer-wise Relevance Propagation (LRP) technique by Bach et al. (2015).

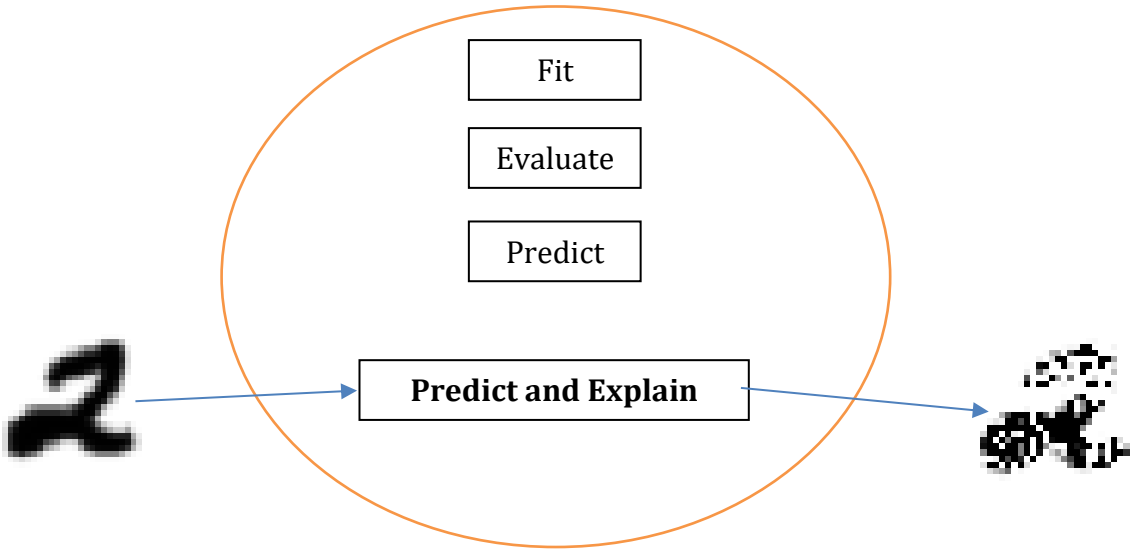
Layer-wise Relevance Propagation (LRP) is a method that identifies important pixels by running a backward pass in the neural network. The backward pass is a conservative relevance redistribution procedure, where neurons that contribute the most to the higher - layer receive the most relevance from it. The LRP procedure is shown graphically in the figure below.



1.3 Solution – Short Description – ModuleX

ModuleX – this module contains the developed components:

- Fit – Train the neural network.
- Evaluate – Test the neural network on test images.
- Predict – Takes a single image and uses the network to predict its label.
- Predict and Explain – Takes a single image, predicts it, and generates the explained image.



Our approach is to look inside the network and monitor the **inner hidden neurons** and look at their outputs values and by that tell which pixels of the image are important for the predicted label**,** by calculating the importance for each pixel individually.

First, we will build a neural network for image classification and train it on specific handwritten characters.
Second, we will predict a single test image.
Third, we will explain the predicted label by calculating for each original pixel how much it contributes to the predicted label, and visualize the pixels according to their contribution value, to get the explained strongest pixels.

We would like to use the inner hidden layers activations to calculate for each input pixel a **'confident'** number which infers to the **'importance' of** the pixel corresponding to the predicted and highlight that pixel according to that value. (hopefully, the new highlighted image will highlight the 'strongest' pixels more clearly)

This module is very similar to **HeatMapping** logic with some exceptions.

In chapter 2 we will introduce and a new formula for calculating the backpropagation step to generate the explained image.

1.4 Next Chapters

- **Chapter 2 - XAI System**
  - 2.1 ModuleX Description
  - 2.2 ModuleX Architecture and Data
  - 2.3 Experiments Steps
- **Chapter 3 - Case study**
  - 3.1 Results – Accuracy and Loss
  - 3.2 Results - Image
  - 3.3 Explain - Local Interpretability
  - 3.4 Explain - Global Interpretability
  - 3.5 Explain - Mistakes
- **Chapter 4 - Conclusion**
  - 4.2 Summary
  - 4.3 Future Research
- **Chapter 5 - References**

2. Chapter 2 - XAI System

2.1 ModuleX Description

The way we calculate the **'importance'** value is for each layer I (I from N-1 to Zero)

H – Importance, **X** – Inputs, **B** – Bias, **W** – Weights, **I –** layer index, **N** – Number of layers

Confident (How important the pixel is) Based on next layer importance

Current Importance

$$H_i = X_i * [\,(H_{i+1} - B_i) * W_i\,]$$

Current inputs | Next Importance | Current biases | Current weights

$H_n =$ Network outputs (the actual final prediction – in our case we have two labels)

$H_0 =$ The explained results - the calculated highlighted image pixels

- The idea is that when we go back through the network, we want to know how much contributes each input (output of the previous layer) to the output value of the current layer.
- For that, we calculate a 'Confident' value and, multiply it with the input.
- The 'Confidence' is based on the next layer importance minus bias (when training we added the bias so now, we subtract it)
- Every input has K weights (which are the ones that in charge of how important the output should be when training), so multiply it by the weights gives us how important was the input. **This is the 'Confident'.**
- We multiply the inputs with the confidence (for each one)
- We do this until we reached the first layer, then we have our explained image.

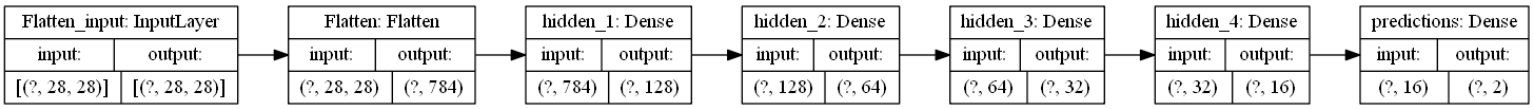
Why is it better? ModuleX VS Lime, SHAP, and HeatMapping

As we saw, **Lime, SHAP and HeatMapping**, models can give nice image explanations and highlight the strongest pixels, so what is the difference to our ModuleX?

- **Lime and SHAP** – do not based directly on the inner hidden layers activations, which is not what we wanted to investigate. (Each has a big other theory for its logic, require high performance).
  Both are generating new input images based on the final error gap. (Which makes more impact on the original image). We are doing it one time and not generating new images each calculation

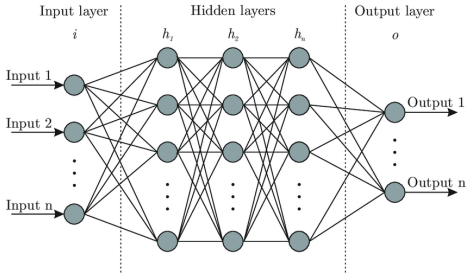
- **HeatMapping** – our new logic reminds **LRP** of' backpropagation'.
  1. The HeatMapping formula doesn't use the Bias, ModuleX considers the bias.
  2. The HeatMapping formula uses an average calculation, ModuleX calculates the importance of each input individually to other inputs, so it doesn't have manipulation on the data.
  3. This formula calculates the importance of each input individually to other inputs, plus the 'Confident' of each input is depends on the trained weights.

2.3 ModuleX Architecture and Data



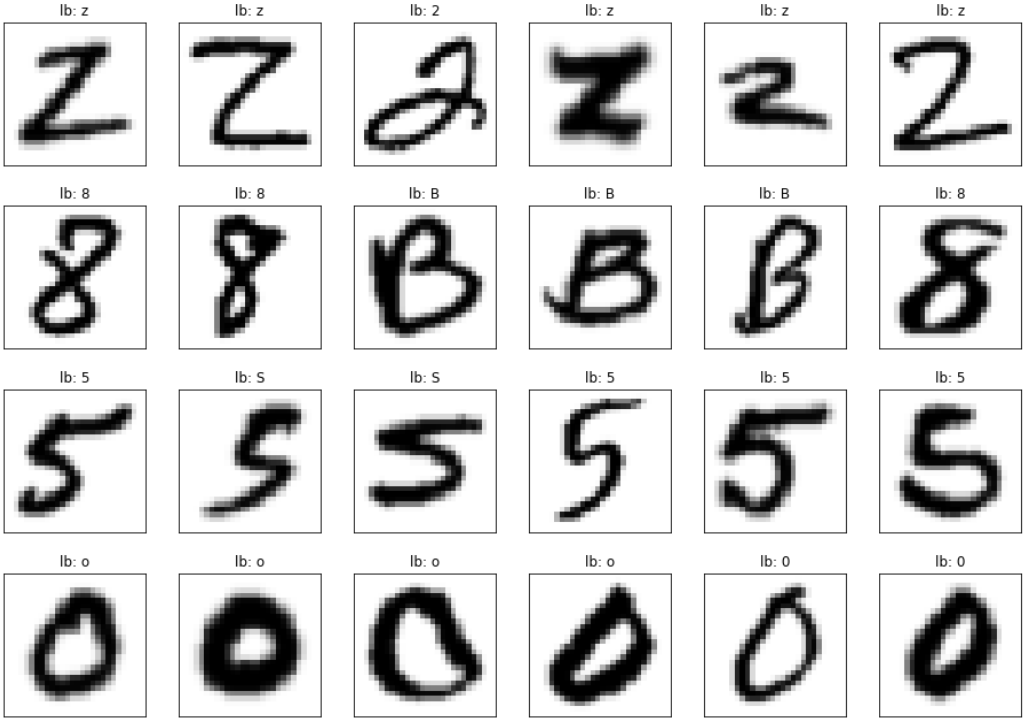
Our network consists of 6 layers:

1. Input layer of 28 x 28 neurons represents the image pixels.
2. Flatten layer of 784 neurons.
3. Hidden layer of 128 neurons (relu)
4. Hidden layer of 64 neurons (relu)
5. Hidden layer of 32 neurons (relu)
6. Hidden layer of 16 neurons (relu)
7. Output layer of 2 neurons - labels (softamx).



Our **EMNIST** data consists of 8 type of 28 x 28 grey level images representing handwritten **'2' and 'Z', '5' and 'S', '8' and 'B', '0' and 'o'**.

Below are some examples, for each label we have 2400 images for training and 200 images for testing.



2.4 Experiments steps

We will focus on the 4 simple basic image classification problems for two types of handwritten characters **'2' vs 'Z'**, **'8' vs 'B', '5' vs 'S'**, **'0' vs 'o'**.

These are very similar and hard to distinguish even for humans, so it will be nice to give explanations for the difference between these two for each network model.

- For each problem, we will build and run a single instance of our network, to train on the two types of images, and track the accuracy of each test data. (to see how accurate the model knows to distinguish between those two types of images)
- To be more accurate in tracking accuracy and loss, we will write its values on each **batch** instead of on each **epoch**.
- We will track and explain our test data images, and visualize the explained images (using **TensorBoard**)

Step 1 – Local interpretability - Monitor neurons output for a single image
We will try to tell why the label was selected for that single image.

- Visualize an input image and highlight the most contributing pixels.
- We expect to see the strongest highlighted pixels look like the input image (more or less).
- Will expect to see patterns that are closed to the input image patterns.

Step 2 – Global interpretability - Monitor neurons outputs for 'K' images
We will try to tell why the label was selected for that single image.

- We will calculate an average of k=400 test data explained images (Local interpretability) of each label.
- Visualize the average image.
- Will the average image look like most of the input images?
- Out of K=400 explained images of '2', Will we the average image look like '2'?

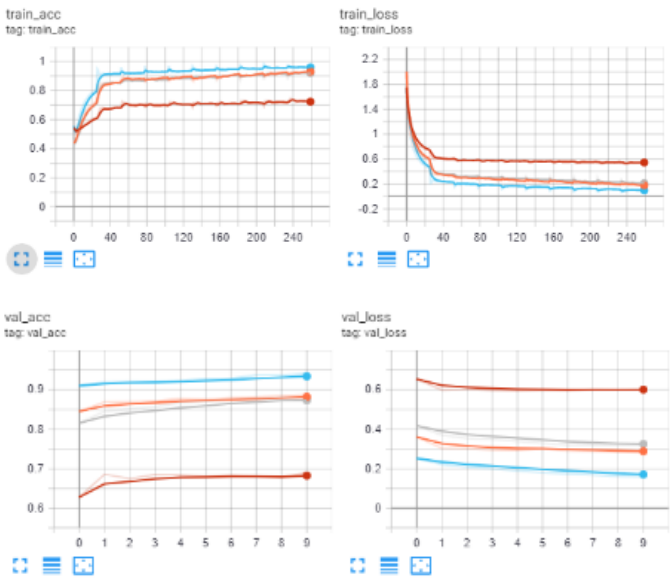
3 Chapter 3 - Case Study

3.3 Results – Accuracy and Loss

How accurate our models?

We've trained 4 models on and here are the accuracy results:

	Train accuracy	Train loss	Validation accuracy	Validation loss	**Test accuracy**	**Test loss**
'8' vs 'B'	0.92	0.09	0.92	0.15	**0.94**	**0.15**
'2' vs 'Z'	0.87	0.17	0.88	0.28	**0.89**	**0.26**
'5' vs 'S'	0.86	0.20	0.86	0.31	**0.86**	**0.30**
'0' vs 'o'	0.69	0.59	0.67	0.60	**0.70**	**0.58**



'8' vs 'B' – this model reached the highest accuracy (0.94) and lowest loss (0.15).
we can understand the model finds this problem as quite an 'easy problem.
Can we give a much more detailed explanation and visualization for that?

'2' vs 'Z', **'5' vs 'S'** – these models have very similar results, and reached also high accuracy (0.89, 0.86) and low loss (0.26, 0.30), but they are a bit worse than '8' vs 'B'.
we can understand that model finds this problem a bit 'harder' problem than '8' vs 'B', but still with good performance.
What could be the reason for that?

'0' vs 'o' – this low accuracy (0.7) and high loss (0.58) means the model finds it very hard to distinguish between '0' and 'o', this makes sense since this mission is very hard to even for a human.
Can we tell which parts (pixels) in the images are the diff between '0' and 'o'?

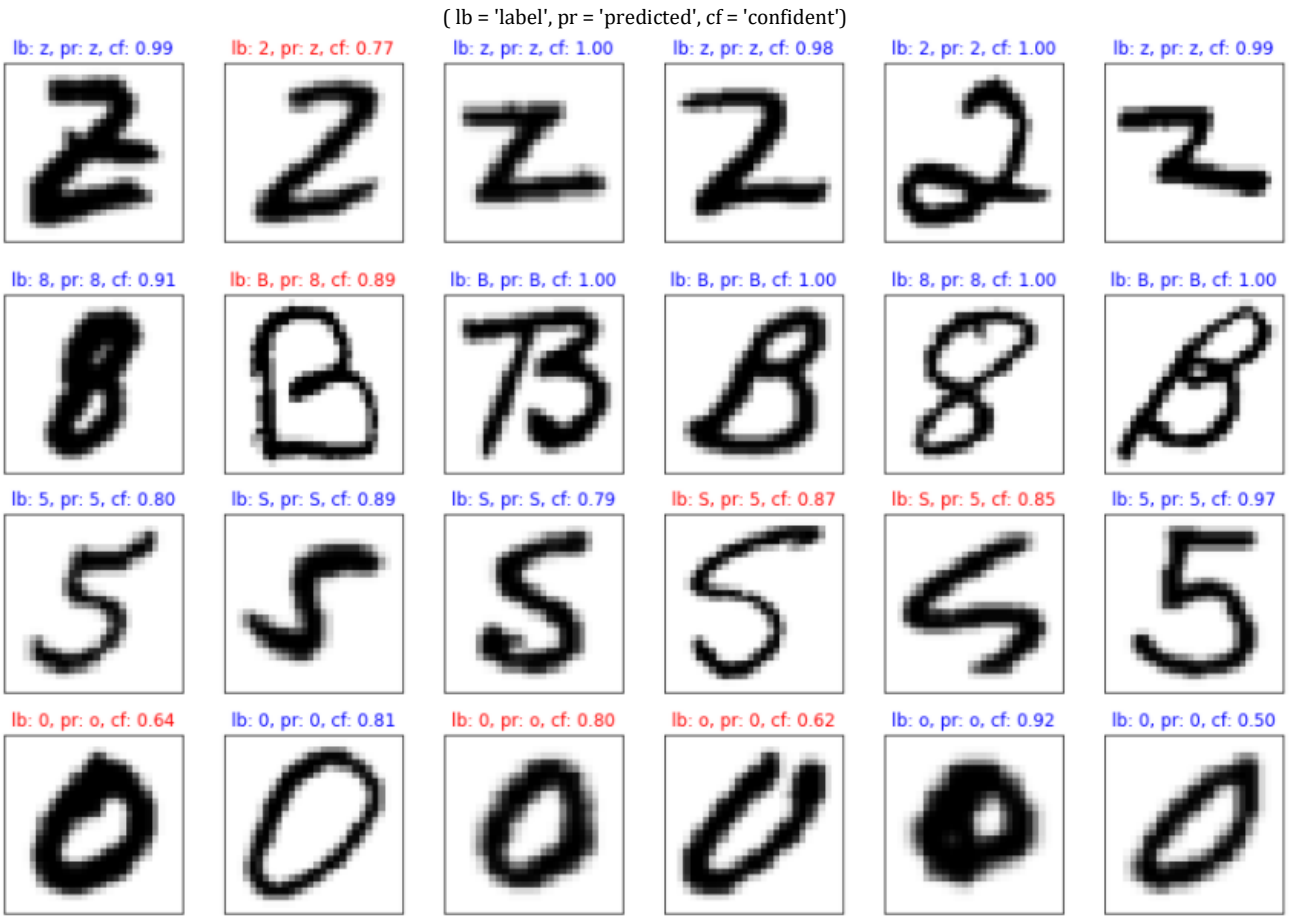
3.4 Results - Images

Looking at some images and use our model to make a prediction.

Here are predicted some samples from our test data, we can see some are correct (blue) and some are mistakes (red).

Can we give a detailed explanation for correct decisions? (this can lead to much trustable model)
Can we give a detailed explanation for mistakes?

We will try to answer and explain these results by using **Local** and **Global** interpretability.



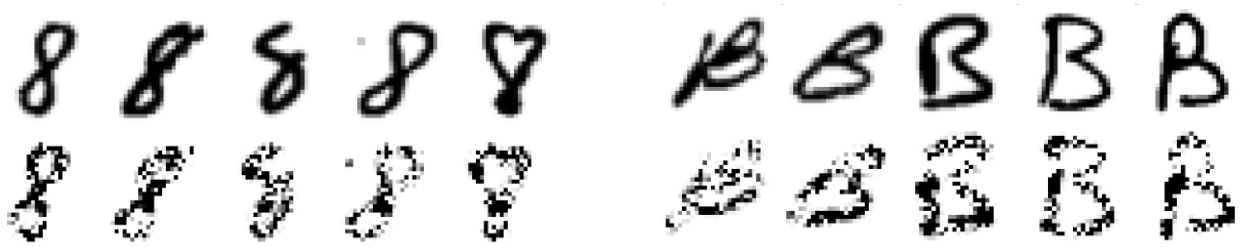
3.5 Explain - Local Interpretability

We will try to explain some test images by looking at the outputs of the hidden layer. From each image, we will generate a new image representing the most contributes pixels. (3.1)

Local interpretability - samples of correct model decisions:

- We can see that almost all the explained images remind the original images.
- By looking at the explained strong image pixels, we can tell the right character, which makes the prediction model much more trustable for us.
- Although we cannot tell so much the difference between **'0' vs 'o'** and **'5' vs 'S'**, the explained images look almost the same, this is not so helpful.

8 vs B



2 vs Z

5 vs S

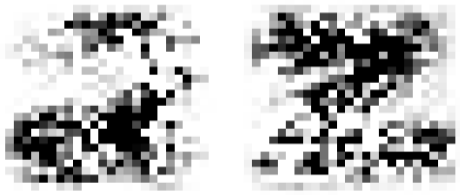
0 vs o

3.6 Explain - Global Interpretability

We will calculate an **average** of 400 test data explained images of each label. This visualization can tell us what the model thinks 'globally' respecting both images.

We can see the most important difference between the two images. Seems like the important differences are stronger in each image. We will look at the strongest pixel areas in each image and see the difference.

2 vs Z



'2' – top center curve, bottom left "circle", bottom center area.
'Z' – top area, top right "triangle", bottom right area.

We can see nicely the diff between '2' and 'Z' in average, by looking at those strong pixels areas.

8 vs B

'8' – top curve, center area, small bottom center curve.
'B' – center left area, bottom area and curves.

The diff between '8' and 'B' in average, is not clear enough but can see the strong pixels in each image are making the different between '8' and 'B'

5 vs S

'5' – top left area, bottom center and left curve.
'S' – bottom right area, bottom area and curves.

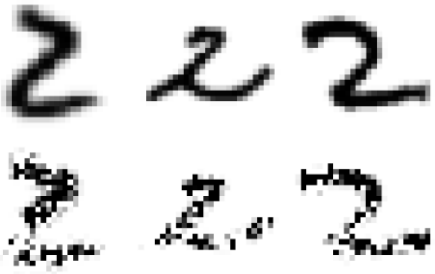
The diff between '5' and 'S' in average, is not clear enough. we can see the strong pixels in each image are not really distinguish between '5' and 's'.

0 vs o

We know this problem is hard and hence the shapes looks almost identical.
'0' – top and bottom curves.
'o' – all "circle" curves are strong.
We know that zero is usually more vertical elliptic than the letter 'o', and that could be the reason.
We can understand now why we got low accuracy for this model. (both explained images are almost identical, with only very small diff
)

3.6 Explain - Mistakes

Now, that we know the 'global' Explainability of our models (We know how it 'thinks' globally), we can tell more details on its mistakes.
Let's look at some images our models **failed** to recognize the true label.



Label = '2' and predicted = 'Z'

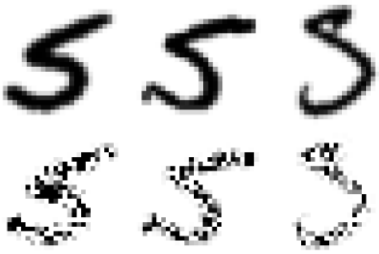
Clearly the first image looks like 'Z' (clear for human), and its explained image, reminds our global explanation image for 'Z' (top area, top right "triangle", bottom right area).

All three images don't have the (bottom left "circle"), that we know our model recognize '2'.

Label = '8' and predicted = 'B'

We saw that the 'center left area' are corresponding to 'B', which exists in the second and third images here, what can explain why the model predict 'B' for those images.

Other than we the explained images are not helping us a lot.



Label = 'S' and predicted = '5'

We know the model finds it hard to tell the difference between '5' and 's' since its shape are almost the same, and hence we can't tell exactly why it was mistakes on those images.

Label = 'o' and predicted = '0' (zero)

These are images labeled as letter 'o', but we can see those are more vertical elliptic (and not complete "circle" like the letter 'o'), so that might be the reason the model recognized them as '0' (zero).

We know our model does not recognized the different with high confidence, and also our explained model failed to explain that.

4. Chapter 4 – Conclusion

4.1 Summary

In this paper, we presented a new technique for constructing a **Local** and **Global** explainability for a deep neural network image classifier. We ran this technique on 4 different hard problems (even for humans). In each model, we showed and compared its accuracy. For each model, we calculated and visualized explained images to tell why a certain label was predicted for a given image **(Local)** and visualize an average of the explained image to tell how the model recognizes images of the same label **(Global)**.

The algorithm is trivially (calculate importance for each pixel in the image), which makes more sense when looking at the end of the explained images, also it has a better performance comparing to other algorithms we showed in this paper.

Out of all the models we investigated, the **'8' vs 'B'** achieved the best results for accuracy and loss, the best-explained model was **'2' vs 'Z'**. By looking at both local and global explained images, we could tell much better the difference between those images.

We basically can say that model **'8' vs 'B'** makes correct predictions in most cases, however, its explainability on those predictions is not quite clear (which makes it less trustable). On the other hand, model **'2' vs 'Z'** is a bit less accurate and can do more mistakes, but its explainability (locally and globally) can help us recognize the data by looking at its explainability. Model **'5' vs 'S'** also achieves high accuracy like the last one but gives non-clear explainability, which makes the model less trustable for us.

Furthermore, we saw that local explainability for **'0' vs 'o'** is not so helpful for us, but by looking at its global explainability we can see the small difference the model finds, which is very important.

The important conclusion we introduced here is: **"Explainability is not just <u>important</u> for deep neural networks models, but more like a <u>must</u>, since without explainability even if your model gets high accuracy, you can't tell if it works well."**

4.2 Future Research

What needs to be done to improve our system (ModuleX)?

1. **Add Explainable confidence number to model output**

   The Module output is an explain image with the strongest pixels, maybe it is possible to add conf number (0…1) for the Explainability output.

2. **Textual explanation**

   The Module output is an explain image with the strongest pixels, however, it will much more usable and trustable if it can give us a textual explanation like: 'This image is 2 with the confidence of 98%, and recognized these most important group of pixels as the top edge of character 2', and so on…

   This way we can help visualize the explained image and its explanation together, which may help to understand hot improve ModuleX.

3. **Convolutional network**

   We chose specific and simple network architecture; all layers are fully connected. However, other architectures should be investigated, Convolutional networks, for example, we know they get better accuracy result, it can be useful to check its explainability.

4. **Expand ModuleX to use all MNIST images**

   On this paper we introduce 4 different experiments: **'2' vs 'Z'**, **'8' vs 'B', '5' vs 'S'**, **'0' vs 'o'**.
Improving ModuleX to experiment with one single network to train on all handwritten digits, and then check the explain mechanism, to see how the model explains the difference between many images.

5. **Check RGB images**

   Furthermore, the RGB image classification problem should be investigated with this new technique, in the end, RGB images are real-world problems.

We are confident that this research will continue by the community, so we can have more explainable and trustable deep neural network models.

6. Chapter 5 - References

[1] The mythos of model interpretability - Link

[2] Deep Neural Networks Help to Explain Living Brains - Link

[3] Peeking Inside the Black-Box - Link

[4] Interpretable Machine Learning Book - Link

[5] Lime - Link1, Link2

[6] SHAP - Link1, Link2

[7] EMNIST - Link

[8] HeatMapping - Link

[9] On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation - Bach et al. (2015). - Link

[10] LRP - Bach et al. (2015).

[11] LRP toolbox on GitHub - Link

[12] TensorBoard - Link