Explaining CNN Classifications Using Small Patches

Jean-Marc Boutay[1,*,†], Quentin Leblanc[1], Damian Boquete[1], Deniz Köprülü[1], Ludovic Pfeiffer[1] and Guido Bologna[1,†]

[1]Department of Computer Science, University of Applied Sciences and Arts of Western Switzerland, Rue de la Prairie 4, 1202 Geneva, Switzerland

Abstract

Convolutional neural networks (CNNs) have achieved remarkable success in image classification tasks. However, their decision-making processes remain difficult to interpret, limiting their adoption in sensitive domains. We present here a novel explainability method that generates global explanations in the form of propositional rules, combining both pixel values and probabilities associated with sub-image patches. Our approach integrates a multi-layer perceptron trained on image patches, a CNN trained on patch probabilities, and a global rule extraction technique. The key idea is to highlight the most relevant image regions that the model uses for its predictions while maintaining high classification performance. We apply this method to three image classification problems: MNIST, CIFAR-10, and FER2013. The generated rulesets capture meaningful patterns in the data and provide accurate and faithful explanations. Although the rules generalize well on simpler datasets like MNIST, both simple and complex image classification problems, such as CIFAR-10, result in large rulesets, with the size increasing further as visual variability grows. Our method achieves competitive performance with standard CNNs while adding a rule-based explanation that highlights the inference process.

Keywords

Model Explanation, Rule Extraction, Convolutional Neural Networks, Image Patch Explanation, AI Trustworthiness

1. Introduction

The explainability of deep neural networks (DNNs) remains an open research problem. In the XAI domain, popular techniques applied to DNNs include LIME [1], SHAP [2], and GradCAM [3]. Some object recognition techniques generate heatmaps, e.g. highlighting important regions that contribute to the classification. Reviews of XAI methods have been presented in [4, 5].

A problem with heatmaps provided by GradCAM and similar methods is that they can highlight the same area for different classes, and they do not clarify the inference process. This work proposes a novel explainability method that uses propositional rules with antecedents representing small patches of an image to explain classification decisions. Furthermore, each patch represented in a rule is associated with a probability. For example, a rule could be: "If a patch of a certain size at a given position contributes to a given class with a certain probability, then a certain class of objects is present".

Very few works have tried to generate propositional rules from DNNs. In previous work, our rule extraction algorithm was applied to multi-layer perceptrons (MLPs), support vector machines (SVMs), ensembles of MLPs, and convolutional neural networks [6, 7, 8, 9]. The key idea behind our algorithm is to determine axis-parallel discriminatory hyperplanes using a greedy algorithm that, at each iteration, tries to maximize fidelity. The main contribution of this work is the patches obtained by applying the rule extraction algorithm to a second-trained model, which associates a probability with each patch of each image. We illustrate our method with three benchmark problems. Finally, we present several examples of rules that explain CNN classifications. The following parts present related work, the rule extraction algorithm, the methodologies employed in this study, the experiments, and the conclusion.

ANSyA 2025: 1st International Workshop on Advanced Neuro-Symbolic Applications, co-located with ECAI 2025.

*Corresponding author.

†These authors contributed equally.

jean-marc.boutay@hesge.ch (J. Boutay)

0009-0000-4138-204X (J. Boutay); 0000-0002-6070-3459 (G. Bologna)

2. Related Work

Explainability is a feature lacking in many machine learning models. Examples of explainable models include propositional rules, linear and logistic regression, single decision trees, and, to a certain extent, nearest-neighbor classifiers. Artificial neural networks and model ensembles, such as random forests, are inherently inexplicable. However, a number of methods have made it possible to approximate these opaque models with interpretable ones. For instance, a large number of techniques have been proposed to approximate MLPs or SVMs with propositional rules [10, 11].

Common explainability techniques used to understand model decisions in connectionist models are feature relevance methods and rule-based explanations. Feature relevance methods, such as Shapley values [2] and LIME (Local Interpretable Model-Agnostic Explanations) [1], assess the contribution of each input feature to a specific prediction. Shapley values, which are based on cooperative game theory, have desirable properties such as efficiency and symmetry. However, they are computationally complex and assume feature independence. LIME, on the other hand, constructs a local surrogate model (linear or tree-based) around the instance to be explained using perturbed samples weighted by proximity.

Rule-based explanations present decisions in a logical way, making them easier to understand. Early taxonomies classify rule extraction methods as pedagogical, decompositional, or eclectic [12]. Pedagogical approaches, such as decision trees, learn input-output relations without using model weights, whereas decompositional methods analyze the weights and often encounter exponential complexity. Eclectic methods combine both.

Unlike visual explanations such as heatmaps, which highlight general regions (e.g. in image classification), logical rules provide an explanation and a prediction of the original model's behavior, offering greater interpretability. For instance, in the task of classifying numerals within images, heatmaps typically highlight pixel regions corresponding to the digit, yet this emphasis is uniform across all classes and fails to distinguish between them. We argue that a robust explanation should incorporate both descriptive and predictive elements, allowing the original model to be substituted.

The use of propositional rules to explain deep learning models remains largely unexplored in the existing literature due to the inherent complexity of the problem. Townsend et al. presented ERIC (Extracting Relations Inferred from Convolutions) [13]. This method assumes that each kernel is linked to a specific concept. The output of each kernel is quantized as a binary value, which allows rules to be extracted that link the binarized kernels to each other. This technique involves a phase in which the kernels are labeled using manually established symbols. In future work, the authors proposed the automation of symbol annotation, with particular emphasis on leveraging established approaches to facilitate the mapping of convolutional kernels or receptive fields to their corresponding semantic constructs.

The method introduced by Padalkar et al. is similar to the previous one [14]. The authors consider that each kernel in the last convolutional layer can be associated with multiple concepts rather than just one. Therefore, after binarising the kernel activations, they introduce a method that enables them to automatically label the concepts. However, the entities present in the images must be annotated in advance. For example, the 'bedroom' class is characterized by the presence of one or more beds, which must be annotated in the images in the dataset.

Unlike the previous two approaches, this work extracts propositional rules regardless of any prior knowledge of the classification problems. It is worth noting that this covers the vast majority of datasets. Consequently, the relevant regions related to the rule antecedents, indicated by small squares in the images, are visualized to help users understand the classification strategy.

3. Methods and Models

3.1. The use of Axis-Parallel Hyperplanes

Let us describe a general MLP model and denote $x^{(0)}$ as a vector for the input layer. For layer $l+1 \ \ (l \geq 0)$, the activation values $x^{(l+1)}$ of the neurons are

$$x^{(l+1)} = F(W^{(l)}x^{(l)} + b^{(l)}). \tag{1}$$

$W^{(l)}$ is a matrix of weight parameters between two successive layers l and $l+1$; $b^{(l)}$ is a vector called the bias and $\sigma(x)$ is a sigmoid activation function:

$$\sigma(x) = \frac{1}{1+\exp(-x)}. \tag{2}$$

When $W^{(0)}$ is a diagonal matrix and the activation function in the first hidden layer is a step function $t(x)$ given below, we obtain axis-parallel hyperplanes; one for each input neuron.

$$t(x) = \begin{cases} 1 & \text{if } x > 0; \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

To efficiently train an MLP with axis-parallel hyperplanes, we replace the step function with its generalization, which is a staircase activation function. Each step of this function creates an axis-parallel hyperplane [8].

The creation of axis-parallel hyperplanes can be applied to CNNs. In this case, adding a special layer after the input layer is sufficient [9]. It plays the role of quantization via a diagonal matrix of weights (see eq. 1) and a staircase activation function. We denote this layer as QIL (Quantized Interpretable Layer). Furthermore, the QIL can simply act as a normalization layer, which means that it is frozen during training. In fact, QIL weight values depend on the averages and standard deviations of the training data for each input neuron [7, 9].

3.2. Our Local and Global Rule Extraction Algorithms

Fidelity refers to the degree to which the extracted rules mimic the behavior of a model. This is a measure of the accuracy with which the rules represent the decision-making process of a neural network. Specifically, with s samples in a training set and s' samples for which the classifications of the rules match the classifications of the model, the fidelity is $\frac{s'}{s}$.

Our local rule extraction algorithm strongly uses fidelity. During rule construction, at each iteration, it determines the hyperplane that involves the highest increase of fidelity. Its computational complexity is linear with respect to the product of the following: the dimensionality of the classification problem, the number of training samples, the maximum number of antecedents per rule, and the number of steps in the staircase activation function [9].

The execution time of our local rule extraction technique can be accelerated by considering two dropout parameters: p and q. Essentially, p determines at each step the proportion of input variables that will not be taken into account. The parameter q is similar, but concerning excluded hyperplanes.

Our global rule extraction algorithm generates a set of rules for a training set of size s. It corresponds to a covering technique that calls our local rule extraction algorithm s times. Therefore, it first generates s rules and then uses a heuristic to select a subset of the rule base that covers all s samples. Then, a simple heuristic consists of ranking the rules in descending order according to the number of samples covered, and then selecting the rules in descending order until all the samples are covered.

3.3. Methodologies for Obtaining Rules Involving Patches

Our goal is to obtain explanations on image samples classification using patches reflecting exactly the model's behavior. To do so, we implemented two different yet similar methodologies. The diagram of the first one is illustrated in Figure 1.

In this method, we first split an image into sub-image patches of size $N \times M \times C$, with N the height, M the width, and C the number of channels. The goal is to get localized explanations of the image. We start at the top left of the image to get the first patch; then we slide horizontally and vertically with a stride S to scan the whole image and get all patches. Usually, we choose a patch size of 7×7 and a stride size of 1. If the image has size $H \times W \times C$ (height, width, channels), we obtain $\left(\left\lfloor \frac{H-N}{s} \right\rfloor + 1\right) \times \left(\left\lfloor \frac{W-M}{s} \right\rfloor + 1\right)$ patches per image. We build a dataset composed of all these patches with their corresponding location in the image, and we train an MLP on it. Then, for each patch, we extract its classification probability for each class. We construct another dataset by concatenating these probabilities with the original image samples. The shape of each data in the new dataset is $\left(\left\lfloor \frac{H-N}{s} \right\rfloor + 1\right) \times \left(\left\lfloor \frac{W-M}{s} \right\rfloor + 1\right) + (H \times W \times C)$.

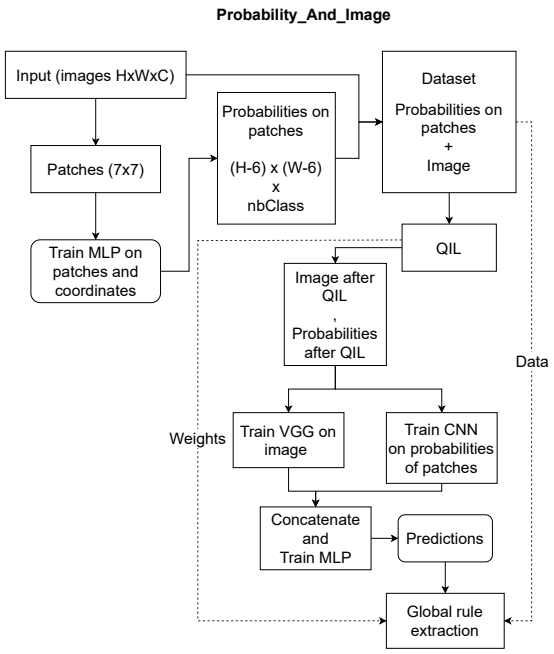

Figure 1: Diagram of the first methodology. This uses the probabilities of train patches and original image pixels. Training of the two models is achieved separately.

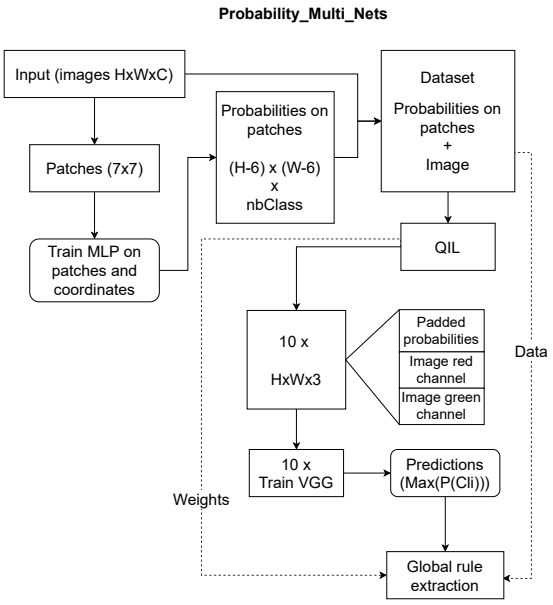

Figure 2: Diagram of the second methodology. This uses the probabilities of train patches and original image pixels. The model is trained jointly on a combined matrix.

We train it on a custom model composed of a QIL (cf. Sect. 2.1) and two separate networks: a VGG-16 and a custom CNN processing respectively the image and the patch probabilities, reunited at the end through a final MLP. The predictions of this model are used to obtain rules explaining the model with our global rule extraction algorithm described above in Sect. 3.2. A rule can contain two types of antecedents. These are image pixels and the probability of a patch for a specific class. The image pixels come from the original image.

The second methodology is similar to the first and gives the same kind of rules. Its diagram is shown in Figure 2. The first steps of the pipeline are the same; we train patches on an MLP and construct the same dataset as before. The second model is different. The data also passes through a QIL, but then we train different VGGs, one for each class in the dataset. Each VGG is responsible for distinguishing one class from all the others and outputs the probability that each sample belongs to that class. The VGG responsible for class i takes input data of shape $H \times W \times 3$. This data is composed of the probabilities of the patches for class i that have been padded to match the size of the image, along with the red and green channels of the original image. The prediction for each sample is the maximum among all class prediction scores. The rules are then computed in the same way.

4. Experiments

4.1. Datasets

We applied our method to three different datasets: MNIST [15], CIFAR-10 [16], and FER2013 [17]. The MNIST dataset is a collection of 28×28 handwritten digits of classes 0-9 and is a common benchmark for image classification. It is made up of tiny black-and-white images. CIFAR-10 is another classification benchmark with colorful $32 \times 32 \times 3$ images of ten different classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. It is known to be a more difficult problem than MNIST. Finally, we used the FER2013 dataset. It is a set of black and white 48×48 images that represent facial expressions. There are seven classes: angry, disgust, fear, happy, sad, surprise, and neutral. To simplify the difficulty of the classification, we have chosen to merge all the non-happy classes into one new class, resulting in two classes: happy and non-happy. This results in an imbalanced dataset, with 33.4% of samples belonging to the happy class. Table 1 shows for each dataset the number of training and testing samples, the size of the images, and the number of classes.

4.2. Metrics

We evaluated our models and rulesets on the datasets using several different metrics. We defined an activated rule for a sample as a rule whose conditions (or antecedents) are satisfied by that specific sample. A correct rule is defined as an activated rule that is faithful to the model's prediction, and a wrong rule as a non-faithful activated rule. The fidelity of a ruleset generated from a training set is 100%. It is essential to note that if no rules are activated for a specific sample, the ruleset will agree by default to the model's prediction. The statistics we consider are the following:

- The training and testing accuracy of the first MLP model;
- The training and testing accuracy of the second model using VGG-16;
- The number of rules in the ruleset;
- The mean number of antecedents per rule;
- The mean covering size per rule (the mean number of train samples covered by a rule);

Table 1
Datasets used in the experiments.

Dataset	#Train samples	#Test samples	Image size	#Classes
MNIST	60000	10000	$28 \times 28 \times 1$	10
CIFAR-10	50000	10000	$32 \times 32 \times 3$	10
FER2013	28709	7178	$48 \times 48 \times 1$	2

- The global fidelity of the ruleset on the test set (how accurately the ruleset reflects the behavior of the model);
- The global accuracy of the ruleset on the test set (the percentage of correct predictions from the ruleset);
- The test accuracy when the ruleset and the model agree (the rate of correct predictions for samples for which the ruleset and the model give the same prediction);
- The test accuracy when the activated rules and the model agree (the percentage of correct predictions for samples for which their activated rules and the model give the same prediction);
- The explainability rate (the rate of test samples for which the activated rules are all correct, or all agree on the same class);
- The default rule rate (the rate of test samples for which no rule in the ruleset is activated);
- The mean number of correct activated rules per train sample;
- The mean number of wrong activated rules per train sample.

4.3. Results

4.3.1. Model architectures and training settings

We detail the architecture of the models and the parameter settings used in our experiments. We have chosen the same configuration for MNIST and CIFAR-10. We trained the first model with patches of size 7×7 and a stride of size 1. We used 100 steps in the staircase activation function in the QIL and a dropout of 0.95% of input variables and hyperplanes during rule extraction. We applied the first methodology described in Sect. 3.3. The three datasets were all learned by the same MLP model on the patches. The model splits into two branches. The first is dedicated to learning each patch and has two dense layers of sizes 128 and 64. The second has only one dense layer of size 8 and focuses on the localization of the patch in the original image, characterized by the coordinate of its uppermost point on the left. The two branches merge and pass through a dense layer of size 64, and through an output layer with softmax, whose size matches the number of target classes. It was trained for 60 epochs with a categorical cross-entropy loss and optimized using the Adam algorithm with a learning rate of 1×10^{-3}. During the second training, the model splits also in two branches. The images are resized to 224×224, passed through a VGG-16 and a dense 256-dimensional layer with a dropout of 30% and batch normalization. On the other side, the probabilities of patches are resized to twice in height and width, passed through three 3×3 convolutional layers of sizes 64, 128, and 256, and a dense layer of size 256. They use batch normalization and a LeakyReLU activation function. A max-pooling of size 2 and an L2-regularization of 5×10^{-4} are used for convolutions, and a final dropout of 40% is applied. The two branches merge and pass through three dense layers of sizes 256, 128, and 64, ending with an output using the softmax activation function. It was trained for 80 epochs, with categorical cross-entropy loss, and optimized with Adam with a learning rate of 1×10^{-5}.

For FER2013, we used the same staircase activation function for the QIL element, but the patch size is 8×8 with a stride of 1, and the dropout is 0.9% of the input variables and hyperplanes. The second methodology has been used. The second training consists of an application of a VGG-16 for each class during 80 epochs, as described in Sect. 3.3.

For each dataset, Table 2 shows the number of training and testing patches and the input and output shape of the first MLP model. Table 3 shows the number of training and testing samples in the second model using VGG-16, and the input and output shape of these samples.

4.3.2. Classification performance

The results presented in this paper were only computed on a single execution and do not represent a mean across several runs, nor do they contain a variance. This is mainly due to the high computational cost of the entire pipeline, even when using a GPU. However, based on our past experiments, we are confident that these results would remain stable across executions, with minimal variance, and are therefore representative of the model's expected behavior.

The accuracies of the first MLP model trained on patches are shown in Table 4. The results of MNIST and CIFAR-10 are not that high because it is difficult to classify between 10 classes with only patches of size 7×7. In the case of FER2013, the performance is strongly related to the rate of non-happy samples, which is 66.6%. The model will almost always predict "non-happy".

The performance of the second model, represented in Table 5, is much better. This is mainly because when we process one sample, we consider the original image sample and all its patches together. All three problems perform well, exceeding 97% in training accuracy. The test accuracies are slightly lower for CIFAR-10 and FER2013, but stay above 92%. These results can be compared with the results shown in Table 6, which reports the mean and standard deviation over 10 runs of training a VGG-16 for 80 epochs on the original images. The performance of our method is really close to these, showing that adding the probability of patches did not really affect the performance.

4.3.3. Statistical analysis of rulesets

Table 7 shows the statistics of the ruleset for each dataset. As CIFAR-10 is a more difficult problem, more rules are required to cover every training sample. The object of interest in the image, for example, a cat, can appear in many different orientations, under various perspectives, scales, and positions. That is the main reason why there are so many rules in the generated ruleset. On the contrary, the faces

Table 2
Size of datasets in the first MLP model.

Dataset	#Train patches	#Test patches	Input size	#Classes
MNIST	29040000	4840000	$7 \times 7 \times 1$	10
CIFAR-10	33800000	6760000	$7 \times 7 \times 3$	10
FER2013	48259829	12066218	$8 \times 8 \times 1$	2

Table 3
Size of datasets in the second model using VGG-16.

Dataset	#Train samples	#Test samples	Input size	#Classes
MNIST	60000	10000	5624	10
CIFAR-10	50000	10000	9832	10
FER2013	28709	7178	5666	2

Table 4
Accuracy of first MLP model.

Dataset	Train accuracy	Test accuracy
MNIST	50.85	51.23
CIFAR-10	33.75	33.47
FER2013	74.84	75.29

Table 5
Accuracy of second model with VGG-16.

Dataset	Train accuracy	Test accuracy
MNIST	99.998	99.61
CIFAR-10	99.97	92.65
FER2013	97.38	92.46

Table 6
Accuracy of VGG-16 trained with the original images, based on ten trials.

Dataset	Train accuracy(std)	Test accuracy(std)
MNIST	$99.94(3.5 * 10^{-4})$	$99.56(6 * 10^{-4})$
CIFAR-10	$99.93(9.1 * 10^{-4})$	$92.87(4.5 * 10^{-3})$
FER2013	$96.47(1.2 * 10^{-2})$	$92.89(2.8 * 10^{-3})$

from FER2013 and the digits from MNIST are very often centered in the image, with similar size and position.

The number of antecedents per rule is nearly the same for each dataset, between three and four. This is fewer than what we could have expected. On average, only three or four conditions are required in a rule to cover samples that have been classified into the same category by the model.

MNIST requires fewer conditions to obtain rules that are faithful to the model, and these rules cover many more training samples, meaning that they capture more common patterns in the training set. They also generalize well, achieving a fidelity of 99.05% and a rule accuracy of 98.82% on the test set. Furthermore, the CIFAR-10 rules are less effective when applied to the test set. This results in a significant decrease in both fidelity and accuracy, which reflects the difficulty in finding reliable and representative rules.

A noteworthy observation is the increase in test accuracy when considering only samples for which the rules and the model agree on the prediction. It increases even further when considering only activated rules (in this case, uncovered samples in the test set are not taken into account). This

Table 7
Statistics of explanatory rules.

Statistics	MNIST	CIFAR-10	FER2013
#Rules	1825	22233	7277
Mean #Antecedents per rule	3.18	3.98	3.75
Mean covering size per rule	473.05	6.22	23.38
Fidelity	99.05	76.55	91.49
Rule accuracy	98.82	72.79	86.21
Test accuracy when rules and model agree	99.70	93.95	93.0
Test accuracy when activated rules and model agree	99.77	95.61	93.63
Explainability rate	97.81	69.12	83.99
Default rule rate	2.13	24.42	16.01
Mean number of correct activated rules per sample	14.45	1.96	5.75
Mean number of wrong activated rules per sample	0.06	0.94	0.73

shows the global relevance of the ruleset, which sometimes outperforms the model. We notice that many samples activate several rules, which means that there are various ways to explain the decision of one sample. Sometimes, one can obtain a rule explaining the test sample that does not reflect the model's decision. It is rare for MNIST but common for the two other datasets. We should prioritize a rule faithful to the model, but it is interesting to analyze these "wrong" rules that maybe predict the truth.

Note that a rule is found for approximately 97.8% of the MNIST test samples, 69% for CIFAR-10, and 84% for FER2013. When no rule is found or when they do not agree on the same prediction, our local rule extraction algorithm would be performed, but this is outside of the scope of this work.

4.3.4. Computational cost

We now look at the computational cost. We need to train all patches, then all probabilities and images, and finally execute the global rule extraction algorithm. We may use GPUs to train the models, but our rule generation is not yet implemented for GPU usage. However, it is parallelized for multiple CPUs. In Table 8, we show the execution times when using one GPU for training and 48 CPUs for rule generation.

Table 8
Execution time (in seconds) on each dataset. This is for the MLP training on patches, the CNN training, and the rule generation. The MLP is trained on one GPU, and the rules are generated using 48 CPUs.

Dataset	MLP time	CNN time	Rule Extr. time
MNIST	75528s(~21h)	16158s(~4.5h)	29820s(~8h)
CIFAR-10	96413s(~27h)	14251s(~4h)	188563s(~52h)
FER2013	62660s(~17h)	16141s(~4.5h)	23931s(~7h)

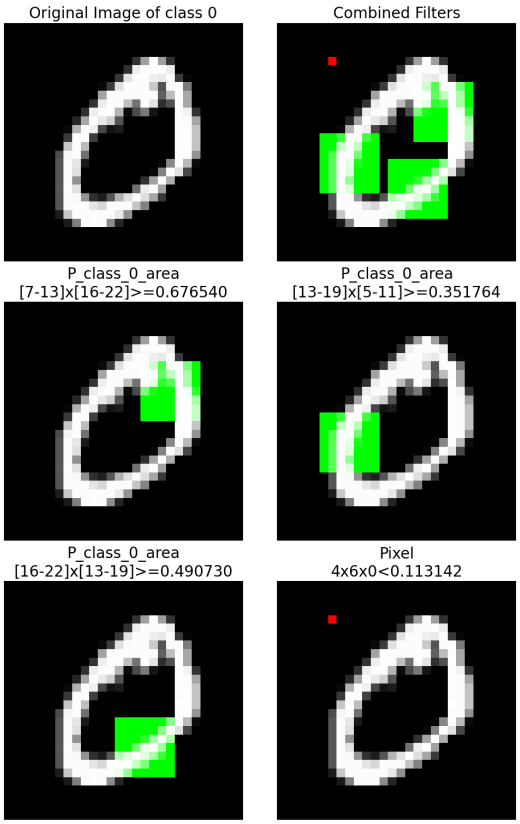

Figure 3: A rule of class 0 activated by a train sample of the MNIST dataset.

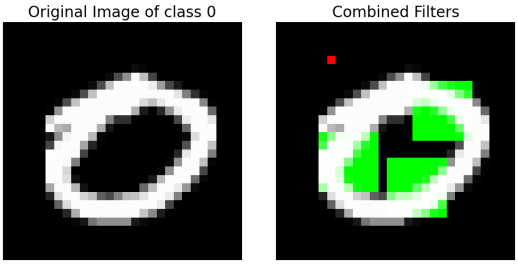

Figure 4: A rule of class 0 activated by another train sample (same rule represented in Fig. 3).

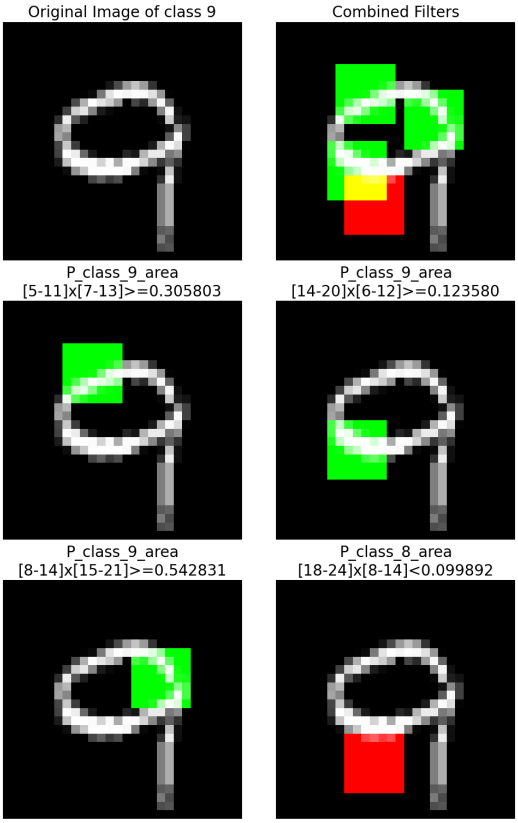

Figure 5: A rule of class 9 activated by a train sample of the MNIST dataset.

4.4. Rules Visualization

Let us now visualize some of the rules that we have obtained. Each figure represents a sample and one of the rules that covers it. It is made up of the original image, an image that contains all the antecedents, and one image for each antecedent. There are two types of antecedents: the probability of a patch for a specific class, represented by a patch on the image, or a pixel value, represented by a single pixel. They are green if the condition inequality is $\geq$ and red if it is $<$. The title of the sub-image describes exactly the antecedent.

4.4.1. Handwritten digits - MNIST

Let us start by looking at some of the rules that we generated for the MNIST dataset. Figure 3 represents a rule of class 0 for a specific sample. This rule has four antecedents. The three patches in the rule are probability conditions for class 0 in these areas. The probability needed goes above 67.65% for the first condition. The last antecedent is a red pixel, which means that this specific pixel value has to be less than 0.11 in the image. If the four conditions of the rule are met by a sample, it will be classified as class 0 by this rule. This specific rule covers 3362 training examples, which means that it is highly representative of the digit 0. This rule has perfect fidelity and accuracy on the train and test sets. Figure 4 shows another sample covered by this rule, which means that the rule covers different shapes of zeros.

In Figures 5 and 6, we see a sample of class 9 covered by two different rules. The two rules have, again, perfect accuracy and fidelity. As before, they cover many samples. We observe that a single sample activates several different rules in the generated ruleset. Both rules have patches around the loop of digit 9. The green ones are related to class 9, and the red one is the probability of another class. For the first one represented in Figure 5, it is interesting to notice that without the red patch, an 8 could fit perfectly the three green patches. That is the reason why the rule needs to eliminate class 8 with a red patch at the bottom, where an 8 will differ from a 9. In Figure 6, it is almost the same, but

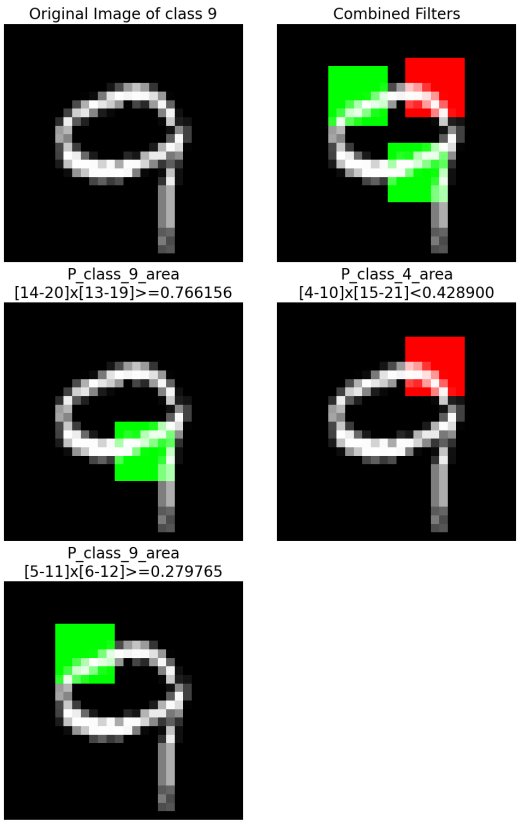

Figure 6: Another rule of class 9, based on the same train sample as in Fig. 5.

Figure 7: A rule of class truck activated by a train sample of the CIFAR-10 dataset.

with class 4 for the red patch. Digit 4 usually does not have a signal on the top right of the image, but the other two green patches could be part of a 4. Therefore, the rule needs to eliminate it with this red patch, asking for a low probability of a 4 there.

4.4.2. Colorful images - CIFAR-10

Figures 7 and 8 show a sample of a CIFAR-10 truck that activates two different rules. It is worth noting that one of them uses many pixels to classify the sample, while the other does not use any. As shown in Figure 7, the patches in the first rule are related to the truck. All antecedents are placed in important spots: the sky, the road, the truck itself, and the wheels (or underneath the truck). They are the principal features of a truck image that can differentiate it from other classes.

The second rule (Figure 8) does not have a sky patch, but the highest patch contains a white area that could be

Figure 8: Another rule of class truck activated by the same train sample as in Fig. 7.

interpreted as the sky. The penultimate patch is a probability for the class frog, likely due to the brown, gray, and blue colors present in the front of the truck. The last one has a probability for the class ship. As an image of a ship usually contains a transition between the water and the ship, the transition between the truck and its wheels can look similar. It is interesting to see how the model can use the other classes to predict an image when it has been trained with patches first. This rule covers only seven train samples and a unique test sample. It is perfectly accurate and faithful on training samples, but even though it is accurate on the covered test sample, it is not faithful, which means that the ruleset predicts better than the model on this specific sample.

Another interesting class is the horse. Figure 9 illustrates a typical rule that we obtain. We observe that patches are located on the head and legs of the animal. There is also a patch eliminating the frog class, as the principal colors of the image, the grass, sky, and the brown of the horse, are often present in a frog image.

4.4.3. Facial expressions - FER2013

Figure 10 shows an example of the Happy class from the FER2013 dataset. There is a pixel in the corner of the eye, one patch in the center of the mouth, one on the left side, and one on the right cheek. Those spots are important for predicting a smile. The different facial folds reveal the person's expression, and that is where the model is looking. An important point to note is that if we use smaller patches during the first model training, the rules will consist exclusively of pixels and will not use patches. This is probably

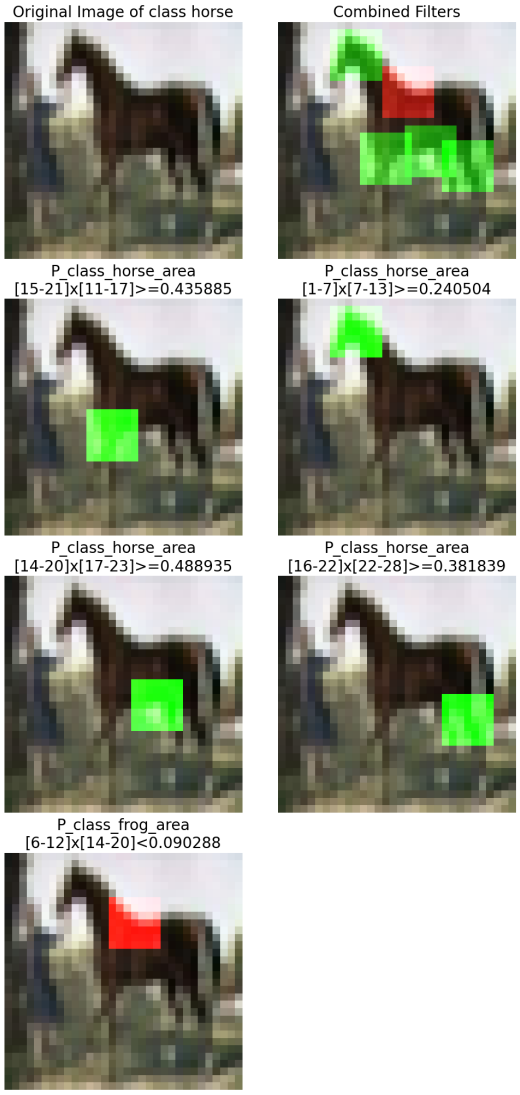

Figure 9: A rule of class horse activated by a train sample of the CIFAR-10 dataset.

because classifying small patches is complex and they do not contain enough information. Consequently, patches measuring 8×8 were used in this experiment.

5. Discussion and Conclusion

We presented two new methodologies explaining the deep neural networks used for image classification. We obtained explanatory rules in the form of patches and pixels of interest by applying a global rule extraction technique. We provided several visualizations to illustrate and explain the rules and image samples. We did it on three different datasets, showing possible results for different use cases. We are not aware of any other work that generates global rules from CNNs with conditions involving patches and pixels. As we used a VGG-16 model with minimal modification (we added the QIL element), our models exhibit very good predictive accuracy. Another strength of our method is that we can always find an explanatory rule to describe a sample because we can activate our local rule extraction algorithm if no rule in an extracted ruleset is applicable.

Our goal was to highlight the most important areas of the image sample that the model uses to predict. In order to obtain good accuracy and fidelity, we needed to train two different models and explain the second model with our global rule extraction technique. The complexity of this method increases because of this, but the results are convincing.

The number of rules obtained in the ruleset is very high for each dataset, especially for Cifar-10. This is mainly because the rules need to cover the entire training set and each rule must have perfect fidelity. The Cifar-10 dataset contains ten different classes, with objects appearing in various locations within the image, at different scales and orientations. It is harder to find rules that correspond to many images.

The highlighted patches and pixels in the rules provide valuable information on the areas on which the model focuses and the links it establishes between the different classes. Often, a rule will look like this: If there is a strong probability for a class to be at some place on the image, and a good probability for another class to be somewhere else, but yet another class should not be at that place, then the rule predicts a certain class. This shows how the model uses the patch predictions to predict the sample.

We plan to try new methodologies and models to see how far performance can be increased and how much we can improve the relevance of the explanation. We wanted to test our method on benchmark problems. We will work on speeding up the whole process to apply it to larger images.

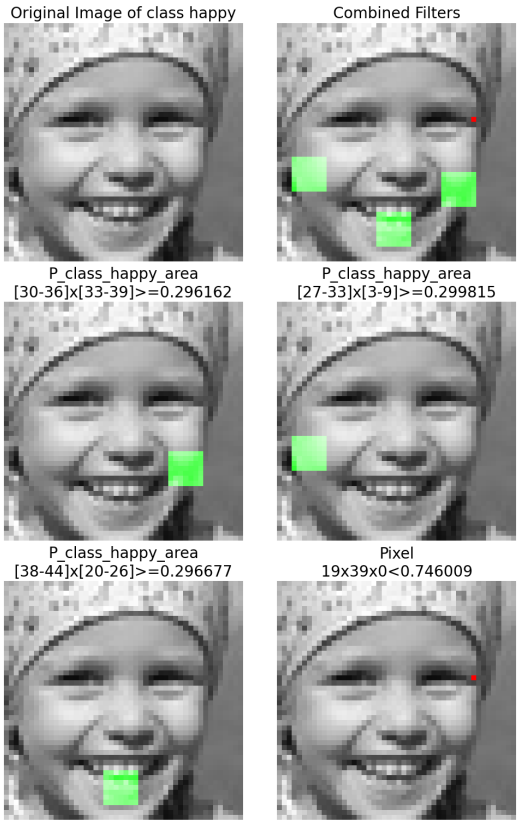

Figure 10: A rule of class happy activated by a training sample of the FER2013 dataset.

Acknowledgments

This work was conducted in the context of the Horizon

Europe project PRE-ACT (Prediction of Radiotherapy side effects using explainable AI for patient communication and treatment modification), and it has received funding through the European Commission Horizon Europe Program (Grant Agreement number: 101057746). In addition, this work was supported by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number 2200058.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT, Grammarly in order to: Grammar and spelling check, paraphrase, and reword. After using these tools, the authors reviewed and edited the content as needed and assume full responsibility for the content of the publication.

References

[1] M. T. Ribeiro, S. Singh, C. Guestrin, "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, 2016, pp. 1135–1144. doi:10.1145/2939672.2939778.

[2] H. Chen, S. Lundberg, S.-I. Lee, Explaining models by propagating shapley values of local components, Explainable AI in Healthcare and Medicine: Building a Culture of Transparency and Accountability (2021) 261–270. doi:10.1007/978-3-030-53352-6_24.

[3] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-cam: Visual explanations from deep networks via gradient-based localization, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 618–626. doi:10.1109/ICCV.2017.74.

[4] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, D. Pedreschi, A survey of methods for explaining black box models, ACM computing surveys (CSUR) 51 (2018) 1–42. doi:10.1145/3236009.

[5] L. V. Haar, T. Elvira, O. Ochoa, An analysis of explainability methods for convolutional neural networks, Engineering Applications of Artificial Intelligence 117 (2023) 105606. doi:10.1016/j.engappai.2022.105606.

[6] G. Bologna, C. Pellegrini, Constraining the mlp power of expression to facilitate symbolic rule extraction, in: 1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227), volume 1, IEEE, 1998, pp. 146–151. doi:10.1109/IJCNN.1998.682252.

[7] G. Bologna, Y. Hayashi, A rule extraction study from svm on sentiment analysis, Big Data and Cognitive Computing 2 (2018) 6. doi:10.3390/bdcc2010006.

[8] G. Bologna, A rule extraction technique applied to ensembles of neural networks, random forests, and gradient-boosted trees, Algorithms 14 (2021) 339. doi:10.3390/a14120339.

[9] G. Bologna, J.-M. Boutay, D. Boquete, Q. Leblanc, D. Köprülü, L. Pfeiffer, Fidex and fidexglo: From local explanations to global explanations of deep models, Algorithms 18 (2025). URL: https://www.mdpi.com/1999-4893/18/3/120. doi:10.3390/a18030120.

[10] F. Sabbatini, Four decades of symbolic knowledge extraction from sub-symbolic predictors. a survey, ACM Computing Surveys 58 (2025) 1–36. doi:10.1145/37490.

[11] J. Diederich, Rule extraction from support vector machines: An introduction, in: Rule extraction from support vector machines, Springer, 2008, pp. 3–31.

[12] R. Andrews, J. Diederich, A. B. Tickle, Survey and critique of techniques for extracting rules from trained artificial neural networks, Knowledge-based systems 8 (1995) 373–389. doi:10.1016/0950-7051(96)81920-4.

[13] J. Townsend, T. Kasioumis, H. Inakoshi, Eric: Extracting relations inferred from convolutions, in: Proceedings of the Asian Conference on Computer Vision, Springer, Cham, 2020, pp. 206–222. doi:10.1007/978-3-030-69535-4_13.

[14] P. Padalkar, H. Wang, G. Gupta, Nesyfold: a framework for interpretable image classification, in: Proceedings of the AAAI Conference On Artificial Intelligence, volume 38, 2024, pp. 4378–4387. doi:10.1609/aaai.v38i5.2823.

[15] L. Deng, The mnist database of handwritten digit images for machine learning research, IEEE Signal Processing Magazine 29 (2012) 141–142.

[16] A. Krizhevsky, Learning multiple layers of features from tiny images, Technical Report TR-2009, University of Toronto, 2009.

[17] D. Erhan, I. Goodfellow, W. Cukierski, Y. Bengio, Challenges in representation learning: Facial expression recognition challenge, https://www.kaggle.com/competitions/challenges-in-representation-learning-facial-expression-recognition-challenge, 2013. Kaggle competition, Accessed: 2025-06-24.

A. Online Resources

Our algorithms can be found on our public Github via

- GitHub