

MASTER

Towards the Explanation of Bayesian Neural Networks

Bykov, Kirill

Award date:
2020

Awarding institution:
Technische Universität Berlin

[Link to publication](#)

Disclaimer

This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

TECHNISCHE UNIVERSITÄT BERLIN

MASTER OF SCIENCE THESIS

Towards the Explanation of Bayesian Neural Networks

Author:
Kirill BYKOV

Supervisor:
Prof. Dr. Klaus-Robert Müller

*A thesis submitted in fulfillment of the requirements
for the degree of Master of Science*

in the

Machine Learning Group
Institute of Software Engineering and Theoretical Computer Science

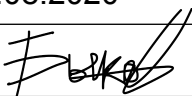
August 12, 2020

Declaration of Authorship

I, Kirill BYKOV, hereby declare that this thesis titled, "Towards the Explanation of Bayesian Neural Networks" submitted is my own, unaided work, completed without any unpermitted external help. Only the sources and resources listed were used.

Berlin, Germany

Date: 12.08.2020

Signature:  _____

“If I knew the meaning of life, do you think I would be wasting my time here?”[62]

GPT-3

TECHNISCHE UNIVERSITÄT BERLIN

Abstract

Fakultät IV Elektrotechnik und Informatik

Institute of Software Engineering and Theoretical Computer Science

Master of Science

Towards the Explanation of Bayesian Neural Networks

by Kirill BYKOV

Explainable AI (XAI) aims to provide interpretations for predictions made by learning machines, such as Deep Neural Networks, in order to make machines more transparent for the user and furthermore trustworthy for applications, e.g. in safety-critical areas. So far, however, no methods for explaining the decision-making process of *Bayesian Neural Networks* have been conceived. In contrast with standard MAP-trained Neural Networks, within the Bayesian framework weights follow a distribution that extends standard single explanation scores and heatmaps to distributions thereof. In the following work, a new framework is proposed, that allows to convert *any* arbitrary explanation method for Neural Networks into an explanation method for Bayesian Neural Networks. Furthermore, the proposed method is even applicable in a non-bayesian scenario and enables the researches to carve out uncertainties associated with a model explanation and subsequently gauge the appropriate level of explanation confidence for a user. The effectiveness and usefulness of our approach are demonstrated extensively in various experiments, both qualitatively and quantitatively.

Acknowledgements

I would first like to thank my thesis advisor Dr. Marina M.-C. Höhne for all the help and guidance throughout this research project. This work was possible mainly because of Marina's great ideas, continuous support, motivation, and organizational skills. Her immense knowledge and connections in the scientific community helped me all the time of the research and I could not imagine a better advisor and mentor for my Master's thesis.

I would also like to thank Dr. Shinichi Nakajima and Prof. Dr. Marius Kloft for their invaluable help throughout the project. I was very lucky to work with two great experts in the Machine Learning community and constantly learn from them. I am very grateful for their insightful comments and encouragement that steered this work in the right direction. Their contribution and guidance allowed me to widen this research from various perspectives.

I would like to thank Dr. Sebastian Lapuschkin for his suggestions on XAI in general. I would like to specially thank Luis Augusto Weber Mercado for designing and producing the overview graphic and Matthias Kirchler for fruitful discussions.

Finally, I want to express my very profound gratitude to Prof. Dr. Klaus-Robert Müller for his support and advice towards this Master's thesis.

Kirill BYKOV

Contents

Declaration of Authorship	iii
Abstract	vii
Acknowledgements	ix
1 Introduction	1
2 Background	5
2.1 Deep Neural Networks	5
2.1.1 Artificial neuron	6
2.1.2 Artificial Neural Networks	6
2.1.3 Convolutional Neural Networks	8
2.2 Explaining Deep Neural Networks	9
2.2.1 Global explanation methods	9
2.2.2 Local explanation methods	11
2.2.3 Layer-wise Relevance Propagation (LRP)	13
2.3 Bayesian Neural Networks	15
2.3.1 Variational Inference	17
2.3.2 Laplace approximation	18
3 Explaining Bayesian Neural Networks	21
3.1 Distribution of relevance maps	21
3.2 Mean LRP	22
3.3 B-LRP	22
3.3.1 B-LRP +	24
3.4 Bayesian Strategies Clustering (BSC)	24
3.5 Explaining non-Bayesian Neural Networks with Bayesian principles	26
4 Experiments	27
4.1 Methodology	27
4.1.1 Quantative evaluation metric	27
Pixel-flipping	27
Bayesian pixel-flipping	28
Pixel perturbation polices	28
4.1.2 Visualization parameters	29
4.2 Bayes by Backprop: MNIST	30
4.2.1 Visual Inspection	30
4.2.2 Quantitative evaluation	30
4.2.3 Bayesian Strategies Clustering	32
4.3 MC Dropout: Imagenet	33
4.3.1 Visual inspection	34
4.3.2 Quantitative evaluation	34

4.3.3	Bayesian Strategies Clustering	36
4.4	Confirming the Clever Hans Effect with B-LRP	37
4.5	Noisy KFAC: Application of B-LRP to other explanation methods	39
5	Concluding discussion	43
5.1	Bayesian Explanation Pipeline	43
5.2	Future work	44
5.3	Conclusion	45
A	Bayes by Backprop: MNIST	47
B	MC Dropout: Imagenet	51
C	Noisy KFAC: Application of B-LRP to other explanation methods	53
	Bibliography	55

List of Figures

1.1	Exponential growth in publications related to a XAI	2
2.1	Illustrations of the natural neural architecture and artificial neuron . . .	6
2.2	Illustration of ANN with five layers	7
2.3	Illustration of LeNet-5 architecture	9
2.4	Illustration of several input samples that maximise activation of network's output neurons	10
2.5	Illustration of possible different objectives in AM algorithm	10
2.6	Comparison between LIME and SHAP explanation methods	12
2.7	Comparison between model-aware explanation methods	13
2.8	Illustration of LRP redistribution procedure	14
2.9	Illustration of LRP Composite method	15
2.10	Comparison between point estimate and bayesian neural network . . .	16
2.11	Illustration of the Dropout Procedure	18
3.1	Illustration of relevance maps sampled from a posterior distribution . .	22
3.2	Overview of the proposed B-LRP procedure	23
3.3	Illustration for B-LRP method explanations for VGG-16 image classification	23
3.4	Illustration for B-LRP+ method explanations for VGG-16 image classification	24
4.1	Illustration of 3 different perturbation policies, used to evaluate performance of explainability methods in our work	29
4.2	Demonstration of different colormaps	30
4.3	Visualisation of B-LRP method for MNIST example	31
4.4	Visualisation of B-LRP+ method for MNIST example	31
4.5	Bayesian pixel-flipping performance comparison on MNIST	32
4.6	Illustration of B-LRP and BSC explanations for MNIST example	33
4.7	Eigenvalues distribution in BSC MNIST example	34
4.8	t-SNE visualisation of relevance maps in BSC MNIST example	34
4.9	Visualisation of B-LRP method for Imagenet example	35
4.10	Visualisation of B-LRP+ method for Imagenet example	35
4.11	Pixel-flipping performance comparison on Imagenet	36
4.12	Illustration of BSC method on Imagenet: example 1	38
4.13	Illustration of BSC method on Imagenet: example 2	38
4.14	Illustration of Clever Hans effect with B-LRP, ex.1	39
4.15	Illustration of Clever Hans effect with B-LRP, ex.2	39
4.16	Application of B-LRP for different explainability methods on LeNet Bayesian Network for class "deer"	40
4.17	Application of B-LRP for different explainability methods on LeNet Bayesian Network for class "bird"	40

4.18	Application of B-LRP for different explainability methods on LeNet Bayesian Network for class "horse"	41
4.19	Illustration of the invariant behaviour of GB explanations toward percentile operation	41
5.1	BNNs Explanation pipeline illustration	43
A.1	Illustration of B-LRP explanations for LeNet, Bayes by Backprop, MNIST	47
A.2	Illustration of B-LRP+ explanations for LeNet, Bayes by Backprop, MNIST.	48
A.3	Illustration of random samples of relevance maps from Cluster 1 in BSC example, MNIST	49
A.4	Illustration of random samples of relevance maps from Cluster 2 in BSC example, MNIST	50
B.1	Illustration of B-LRP explanations for VGG-16, Imagenet	51
C.1	Application of B-LRP for different explainability methods for class "bird"	53
C.2	Application of B-LRP for different explainability methods for class "plane"	53

List of Tables

4.1	Bayesian pixel-flipping AUC scores for MNIST	32
4.2	Pixel-flipping AUC scores for Imagenet	37

List of Abbreviations

AI	Artificial Intelligence
AM	Activation Maximisation
AUC	Area Under the Curve
BDL	Bayesian Deep Learning
BNN	Bayesian Neural Network
B-LRP	Bayesian - Layer-wise Relevance Propagation
BSC	Bayesian Strategies Clustering
CNN	Convolutional Neural Network
CV	Computer Vision
DL	Deep Learning
DNN	Deep Neural Network
GB	Guided Backprop
IG	Integrated Gradients
LRP	Layer-wise Relevance Propagation
MLE	Maximum Likelihood Estimation
MSE	Mean Squared Error
MLP	Multi- Layer Perceptron
NLP	Natural Language Processing
RL	Reinforcement Learning
SC	Spectral Clustering
XAI	Explainable Artificial Intelligence

Chapter 1

Introduction

Artificial Intelligence (AI) remains to be one of the most important technological breakthroughs over the past decade. In 2017, Andrew Ng, one of the world's most famous and influential computer scientists, called AI the new electricity, stating that AI will disrupt all industries the same way the invention of electrical power transformed everything a century ago [8]. From new and emerging technology, AI-powered systems became an integral part of contemporary society and, usage of such technologies is already decisive for companies in diverse areas.

Current advances in AI are tied to developments in the field of Deep Learning (DL). Deep neural networks (DNNs) can learn highly complex, non-linear predictors. Over the last years, DNNs have achieved remarkable results in many applications in different fields, such as computer vision (CV), natural language processing (NLP), and reinforcement learning (RL). However, while DNNs learn powerful representations and achieve state-of-the-art results, in contrast to linear learning machines they are unable to directly reveal their prediction strategy to the user.

Given the powerful but often opaque nature, DNNs are often considered as a 'black-box': effectiveness of these systems in the real world is highly restricted by the machine's current inability to explain their decisions and actions to a regular human user. In various areas of applications, such as safety-critical areas, transparency and insight are mandatory, thus, despite showing great performance in the test environment, DNNs could not be used.

Addressing the concerns among the research community about the necessity to understand the decision-making process of powerful, yet opaque, systems, the field of Explainable AI (XAI) has emerged, establishing techniques to explain predictions made by nonlinear learning machines. This field aims to produce explainable AI models — models that achieve high performance on the assigned tasks and enable humans to understand the decision-making behavior of these machines.

In 2018, global research and advisory firm Gartner has identified XAI in the list of the most promising technologies in the field of AI [94]. The rising trend of contributions to the field of XAI could be observed in Figure 1.1. Current advancements may be associated with a general shift in the Machine Learning community towards the *Responsible Artificial Intelligence* — methodology for implementation of AI-powered systems for real-life applications with fairness, model explainability, and accountability at its core [10]. This tendency shares its rationale with an increasing number of cases related to unethical usage of AI. In particular, hidden discrimination has been, and, unfortunately, remains one of the biggest and unresolved problems in the field of AI. For example, in 2018, Amazon found out that AI that was used for scoring potential candidates for recruitment purposes had a bias against women: the system taught itself that male candidates were more preferable [5]. Another famous example of "machine bias" is assessment software known as COMPAS: it was being

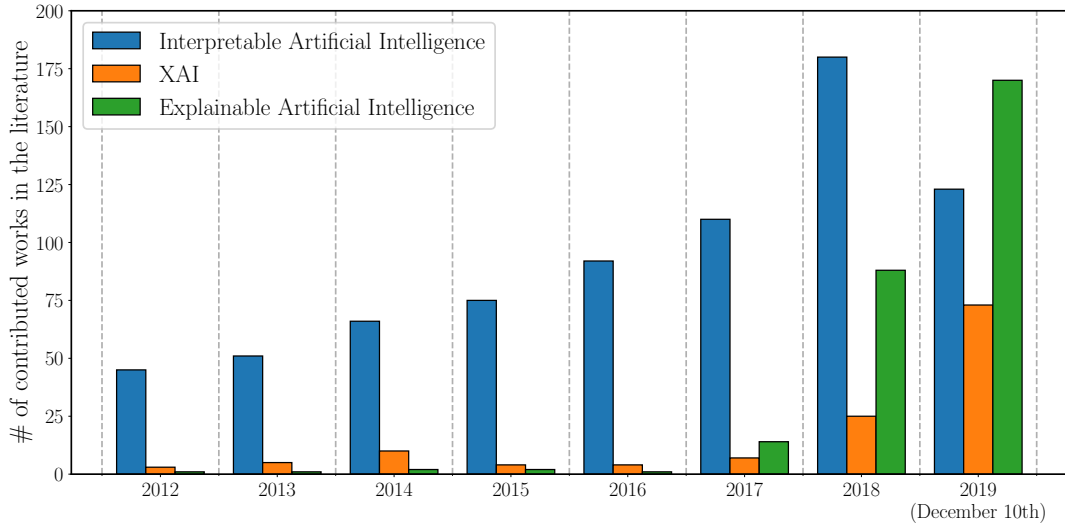


FIGURE 1.1: Growth of the number of publications related to XAI and other related fields during the last years, retrieved from Scopus database [10].

used to forecast which criminals are most likely to re-offend and was proved to have a racial bias against black defendants [37].

Recent advances in the field of XAI introduced a collection of novel methods with the aim of explaining DNNs behavior. However, those methods are specific to a Neural Networks that are trained in a maximum likelihood estimation (MLE) setting. Furthermore, despite the growing interest towards Explainable AI, there has not been found an appropriate way to interpret the Bayesian Deep Neural Networks — a popular class of models that unifies the probabilistic framework with deep learning.

In this work, we develop a novel pipeline for explaining the decision-making process of Bayesian Deep Neural Networks. The described approach can be applied to any explanation method for Neural Networks — be it model-agnostic or model-aware — and to any (approximate) inference procedure of BNNs. Moreover, in our work we discuss how the proposed method can be applied for the creation of better explanations for a non-bayesian model throughout *bayesianization* procedure.

The main contributions of this paper are as follows:

- We propose a new methodology that can leverage any existing local explanation method for neural networks to an explanation method for BNNs.
- We study our approach in detail for a particular explanation method—layer-wise relevance propagation (LRP) [11]—thus proposing the first concrete explanation method of BNNs—called B-LRP, and its enhancement — B-LRP+.
- B-LRP and B-LRP+ provide us with a novel manner of explanation since it outputs a distribution, which can be exploited in interesting ways:
 1. By considering percentiles of the explanation distribution, we can instantiate more *cautious* or *risky* explanations than standard LRP. Thereby the choice of the percentile governs the risk.
 2. We can visually describe areas of certainty and uncertainty of explanations within any example (e.g., image).

3. B-LRP reveals the varying importance of multiple prediction strategies used by the learner.
- Although showcased here for LRP, our proposed methodology for explaining neural networks under uncertainty can in principle be applied to any explanation method for neural networks.
 - We propose a Bayesian Strategies Clustering (BSC) method — a semi-automated technique for discovering primal strategies in local behavior of Bayesian Neural Networks

The validity of the above findings is studied and demonstrated in various experiments. Qualitative and quantitative experiments nicely underline the usefulness of the B-LRP and B-LRP+ methods, which we additionally provide as an open-source PyTorch implementation¹.

Thesis structure

This work is structured as follows: Chapter 2 introduces the basic concepts of Deep Neural Networks, Explainability algorithms, and Bayesian Deep Learning. Chapter 3 is dedicated towards the description of proposed methods: *Mean LRP*, *B-LRP* and *BSC*. In Chapter 4 we demonstrate the usefulness of proposed methods on several examples for different BNNs. Finally, in Chapter 5 we conclude our work and formulate the main insights that were obtained throughout the research.

¹<https://github.com/lapalap/B-LRP>

Chapter 2

Background

In this section, basic concepts of Artificial Neural Networks (ANNs), Deep Neural Networks (DNNs), and, in particular, Convolutional Neural Networks (CNNs) are described. We discuss the main approaches for interpreting the decision-making process of DNNs with a brief introduction to the most popular methods. The Layer-wise Relevance Propagation (LRP) algorithm is explained in detail, being a foundation of the proposed methods in Chapter 3. Finally, we describe the theory behind Bayesian Neural Networks (BNNs) and their differences to Neural Networks trained in a standard fashion.

2.1 Deep Neural Networks

Deep Neural Networks (DNNs) have arguably achieved great success in the last years and remain to be responsible for the majority of recent advances in the field of AI. Driven by both, the exponential growth in available data and computational resources, DNNs won numerous contests in pattern recognition and machine learning, achieving state-of-the-art status in different tasks, such as Computer Vision (CV) [114, 102, 100], Natural Language Processing (NLP) [19, 91, 79, 117] and Reinforcement Learning (RL) [18, 95]. Although current Deep Learning research is far away from achieving an Artificial General Intelligence [27], there are already domains where DNNs surpass human performance, like game-playing or image recognition [31, 104, 58].

The history of Neural Network research goes back to the middle of the last century, to the works of McCulloch and Pitts [64], who created an initial computational model for neural networks, followed by the invention of a novel learning mechanism by Hebb [32], that later became known as *Hebbian learning*. Some researchers [87] claim that the ideas of Neural Networks can be traced back to the literature even earlier since early supervised NNs can be considered as variants of linear regression methods, that were described by mathematicians at the beginning of the 19-th century [54, 24, 25]. The initial wave of excitement came after the invention of Rosenblatt's perceptron algorithm in 1958 [82], which was seen as a fundamental cornerstone in the field of Neural Networks research. The next great achievement in the field of NNs was the invention of a Backpropagation algorithm, which was first described in the work of Linnainmaa in 1970 [56] and later popularised in a work of Rumelhart and Hinton in 1986 [83]. This discovery opened a way for practical training of multi-layer networks, greatly increasing the scope of the possible application of Neural Networks and their performance. Unfortunately, lack of computational power and training data was a bottleneck for the development of NNs at that time and the interest of the research community in Neural Networks has shifted to other methods, such as SVMs [16]. In 2012, after the first convolutional neural network

(CNN) AlexNet [48] won the annual Imagenet challenge [17], neural networks received comprehensive community recognition. Advances in computing power of modern machines, in particular the development of fast GPUs, coupled with the availability of large amounts of data made NN overwhelmingly popular across researchers and engineers. This marked the new milestone in AI research and sparked the so-called 'AI-revolution' which continues to this day.

2.1.1 Artificial neuron

The original idea of Artificial Neural Networks (ANNs) was to mimic biological learning mechanisms of a brain. Human nervous system contains cells, *neurons*, that are connected to one another with the use of *axons* and *dendrites*. Human brain learns new information by changing the strength between *synapses* — connections between axons and dendrites [3]. ANNs simulate this mechanism with own computational units — *artificial neurons*. Figure 2.1 illustrates the similarities between natural and artificial neurons. Each artificial neuron makes a particular computation based on other units it is connected to, that described in two steps: *pre-activation* and *activation*. Let m be the number of input signals $x = [x_1, x_2, \dots, x_m]$

1. **Pre-activation:** in the first step we compute the product between the inputs x and the weights w and adding the bias term b :

$$g(x) = \sum_{i=1}^m x_i w_i + b.$$

2. **Activation:** the output of artificial neuron is computed by passing the pre-activation through nonlinear activation function σ :

$$f(x) = \sigma(g(x)).$$

On practice popular activation functions are Sigmoid function, Rectified linear unit (ReLU) function and Hyperbolic tangent function [110].

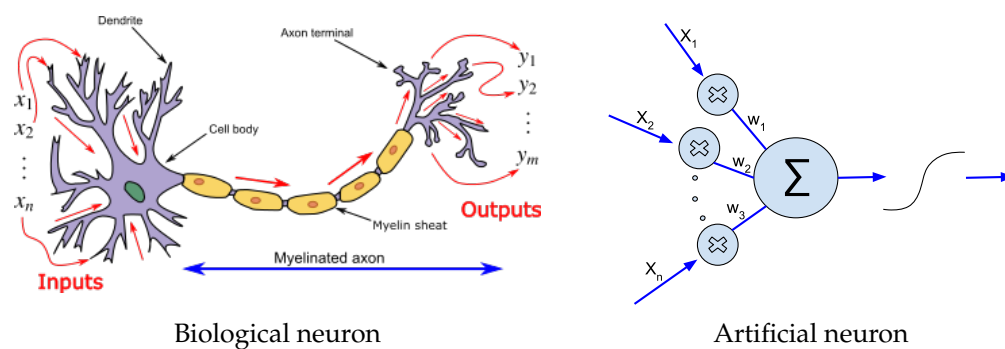


FIGURE 2.1: Comparison between natural neural architecture and artificial neuron: LEFT: schematic illustration of the biological neuron [111], RIGHT: diagram of a McCulloch-Pitts artificial neuron model [112].

2.1.2 Artificial Neural Networks

The neurons in ANNs are usually organized in *layers*: in each layer, neurons take as input values only from the immediately preceding layer and output values to

neurons in the following layer. The first layer that receives the original data as an input is called *input layer* and the layer that produces the final result is called *output layer*. All layers between input and output layers are called *hidden layers*, as the result of the performed computations is not visible to the user. A *Deep Neural Network* (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers.

Different patterns of connections between layers are possible: for example in Fully Connected (FC) layers, each neuron in one layer connecting to every neuron in the next layer. Other popular types include pooling [14], where a cluster of neurons outputs to a single neuron in the next layer. In this work, we will work with a *feed-forward* NN, where successive layers feed into one another in the forward direction from input to output. In other words in feed-forward NN connections of all neurons form a directed acyclic graph. Figure 2.2 illustrates feed-forward fully-connected NN.

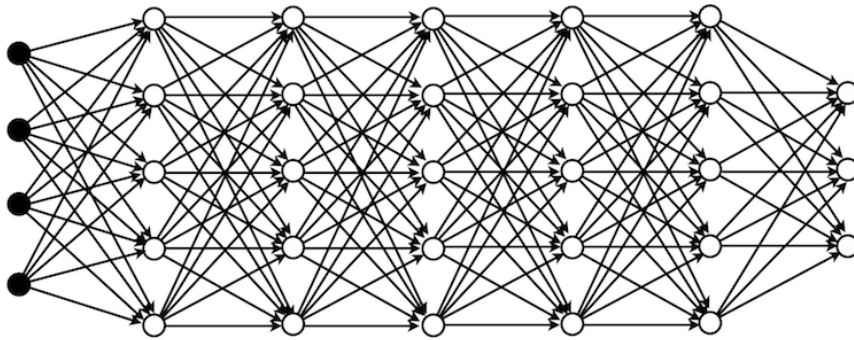


FIGURE 2.2: Schematic illustration of ANN with with five hidden layers. Black nodes illustrate the input, each white node represents an artificial neuron. Arrows are the parameters (weights) indicating the strength of connection between neurons [65].

ANN could be viewed as a function, that maps a point from a (high-dimensional) input domain \mathcal{X} to an output domain \mathcal{Y} :

$$f : \mathcal{X} \times \mathcal{W} \rightarrow \mathcal{Y},$$

where \mathcal{W} is a subspace of parameters for the particular network architecture. Usually, \mathcal{X} corresponds to the space of d dimensional real-valued vectors \mathbb{R}^d . In image classification setting, the network produces real-valued output $Y \in \mathbb{R}^k$, where k is the number of classes, and the resulting vector represents the "scores" of input belonging to each of the k classes. Often, these scores are further normalized (e.g., with a SoftMax function), and the final decision is made by verifying whether the output is above a certain threshold or larger than the output of other functions representing the remaining classes. The function output can be interpreted as the amount of evidence that the model allocates to each class for the current sample [86].

ANNs are constructed with a building blocks of layers: let $L > 0$ be the number of layers in the network, where zeroth layer corresponds to an input and L -th layer is the output of the network. We denote $\sigma^{(1)}, \sigma^{(2)}, \dots, \sigma^{(L)}$ to activation functions in different layers. Let $\mathbf{x}^{(l)}$ denote the output l -th layer, and so $\mathbf{x}^{(0)} = \mathbf{x}$.

The prediction rule could be represented as follows:

$$\begin{aligned}
\mathbf{x}^{(1)} &= \sigma^{(1)} \left(W^{(0)} \mathbf{x} + \mathbf{b}^{(0)} \right), \\
\mathbf{x}^{(2)} &= \sigma^{(2)} \left(W^{(1)} \mathbf{x}^{(1)} + \mathbf{b}^{(1)} \right), \\
&\dots \\
\mathbf{x}^{(L)} &= \sigma^{(L)} \left(W^{(L-1)} \mathbf{x}^{(L-1)} + \mathbf{b}^{(L-1)} \right), \\
f(\mathbf{x}, W) &= W^{(L)} \mathbf{x}^{(L-1)} + \mathbf{b}^{(L)}.
\end{aligned}$$

In the standard (frequentist) setting of NN learning, a MLE or MAP estimate of the parameters is found through the minimisation of a non-convex loss function $\mathcal{L}(x, y)$ w.r.t. network weights. Minimization of this function is performed through backpropagation [83], where the output of the model is computed for the current parameter settings, using the chain rule partial derivatives w.r.t parameters are found and then used to update each parameter by gradient descent:

$$W \leftarrow W - \alpha \frac{\partial \mathcal{L}(x, y)}{\partial W}.$$

2.1.3 Convolutional Neural Networks

Convolutional Neural Network (CNN) is a particular type of a Deep Learning model, most commonly used for Computer Vision tasks. The architecture of these models was inspired by the works of Hubel and Wiesel's about receptive fields in the visual cortex of the cat's brain [38]. Their work showed that specific parts of the visual field of the animal activate particular neurons in the brain: individual neurons respond to stimuli only in a special region of the visual field known as the Receptive Field. Such fields overlap with each other, covering the whole visual field of an animal. Inspired by their work, in 1980 Fukushima et al. [22] introduces two special layers that became founding blocks of CNN: *convolutional* layer and *pooling* layer.

- **Convolutional layer:** the motivation for this layer is to extract features from input by preserving the spatial relationship between the pixels. A rectangular receptive field with a weight matrix of fixed size, a *filter*, slides over the input (image), and for each position produces an element-wise multiplication between filter and values, that lie in the receptive field. The results of the multiplication are summed and saved to the output matrix.
- **Pooling layer:** (or downsampling layer) is used to reduce the dimension of the feature maps, previously computed by convolutional layers. Such a unit usually calculates aggregates over the sliding window (patch) of fixed size, such as average, as was originally proposed by Fukushima [22], or taking a maximum value over the window [107].

Fukushima's ideas were later generalized in the LeNet-5, one of the earliest CNN for handwritten and machine-printed character recognition [52]. Figure 2.3 illustrates LeNet-5 architecture. The network was trained by a backpropagation algorithm and was applied for identifying handwritten zip code numbers [52]. Modern CNN architectures for image classification incorporate the ideas from the LeNet-5, utilizing the convolutions, pooling, and fully-connection structure.

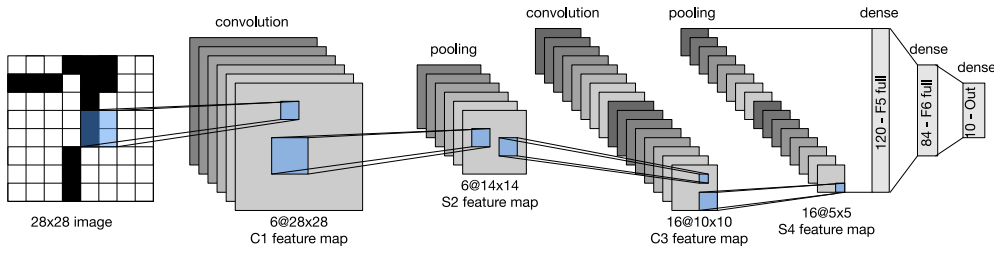


FIGURE 2.3: Visualisation of LeNet 5 CNN architecture applied for a handwritten digits classification task [15].

2.2 Explaining Deep Neural Networks

Deep Neural Networks perform very well at representation learning — they are able to map features from (usually) high-dimensional input to an output [28, 67]. Yet, as modern Neural Networks become more and more complex, it is very hard to explain what particular features have been learned and what particular attributes influenced the decision-making process of a learning machine.

In general, XAI methods could be divided into two main categories: *global* and *local* explanation methods.

2.2.1 Global explanation methods

Global explanation methods interpret the decision-making process of DNNs across a population: they can highlight and describe the inner mechanisms of complex learning machines, that thereby helps to increase their transparency. One popular global explainability method is Activation Maximisation (AM) [21].

In a CNN, convolutional layers learn specific filters so that activation is maximized when a similar pattern is found in the input. The intuition behind the AM method is quite simple: we try to explain a hidden unit by looking for a particular input that maximizes its activation. This input is not searched for in the original dataset, instead, this task is viewed as an optimization problem and we try to artificially generate a pattern, which a particular hidden unit reflects with the highest activation.

Let $h_{ij}(x, W)$ be the activation of a given unit i from a given layer j in the network; where x is an input sample and W represents all parameters (weights and biases) of a trained neural network, that we aim to explain. In AM method we are looking for a particular simulated sample x^* , such as:

$$x^* = \arg \max_{x \text{ s.t. } \|x\|=\rho} h_{ij}(x, W).$$

For this non-convex optimization problem, the local minimum could be found using simple gradient-based methods.



FIGURE 2.4: Input samples that maximize activation of output neurons in the CaffeNet deep neural network, which has learned to classify different types of ImageNet images [75].

More recent research, such as [74, 75], continues with improving the qualitative state of the art of Activation Maximization. Figure 2.4 illustrates several of synthesized input. AM algorithm allows users to identify the ‘prototypical’ cases for the output quantity and aim for a general description of the Machine Learning model.

Global explanations with AM algorithm are not bounded to be used only for explaining the output of the last layer [76]. An important question that arises is: what exactly do we want to explain? For individual features, we can maximize the activation of one neuron, to understand, what was learned by a particular unit. However, it is also possible to explain the layer as a whole (DeepDream algorithm [72]), class logits, or the probability of a class. Figure 2.5 illustrates different approaches for AM algorithm for different objectives.

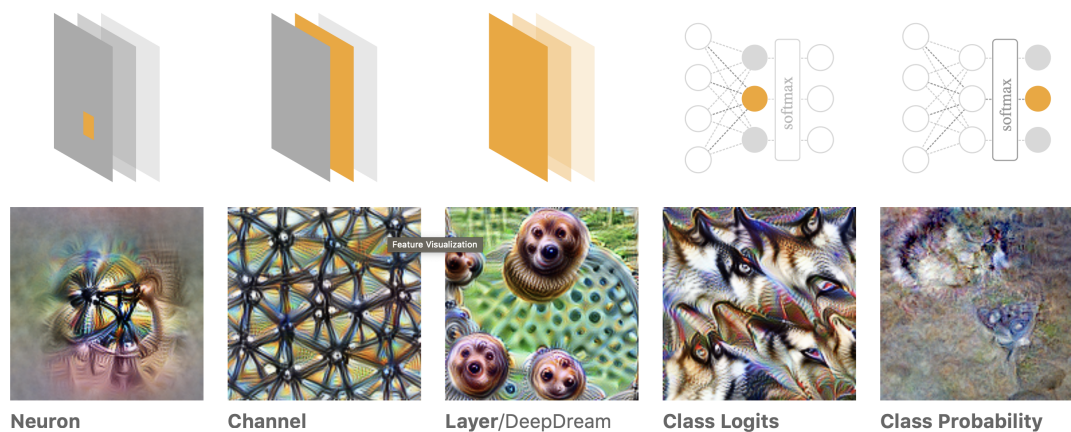


FIGURE 2.5: Examples of different objectives being maximized in AM algorithm [76]. Depending on the goal of the user, that is to explain a particular neuron, layer, or the prediction itself, different objective functions are be maximized.

2.2.2 Local explanation methods

Global explanation methods are useful for knowledge discovery, however, they do not provide explanations given a particular data example. Usually, researchers and practitioners are interested in *local* explanations: given a particular input x , they expect from an algorithm to explain the decision-making process on a particular example. In general, local explanation methods attribute relevances to each of d features of input x to visualize and emphasize which components of input are the most important regarding a given prediction. As a result, explanation methods usually create relevance maps (also known as saliency maps) for each datapoint [11, 103, 88, 70].

To ease future description of various local explainability methods we define a relevance attribution function.

Definition 1 (Relevance Attribution) *A function $R(\cdot, \cdot)$ that maps an input vector $x \in \mathbb{R}^d$ and a vector of network's parameters $W \in \mathcal{W}$ to a real-valued vector of relevances $R(x, W) \in \mathbb{R}^d$ is called a relevance attribution function.*

Existing methods for local explanation can be categorised into *model-agnostic* and *model-aware* methods.

Model-agnostic methods

Model-agnostic methods are based on the idea of separating the interpretations from the machine learning model. Usually, these methods explain the black-box model by detecting changes in the predictions of the learning machine, while perturbing the input in a specific way. The main advantage over model-aware methods is that model-agnostic methods do not depend on any specific architecture of the model, which makes them flexible and easy to use.

Several popular model-agnostic methods include:

- **LIME**

Lime stands for Local Interpretable Model-Agnostic Explanations [80]. This method aims to create an explanation for a particular input by approximating the model *locally* with an interpretable one, such as linear model with regularisation [20], decision tree [84] or random forest [55]. To train a new learner, the original input is perturbed around its neighborhood, and changes in model behavior are recorded. Perturbed data points are weighted by their distance to an original input, and a new, interpretable model is trained on new data and associated with its predictions, thus creating a *locally faithful* approximation of the original model — a model, that can predict how the original learner behaves in the vicinity of the input.

- **SHAP**

SHAP is short for SHapley Additive exPlanations: a game-theoretic approach for machine learning explanations [59]. The method is based on the idea that a prediction of a learning machine can be explained by assuming that each feature from the input is a "player" where the prediction is the payout [69]. To distribute relevances ("payout") between input features, Shapley values, a method from coalitional game theory, is used [90].

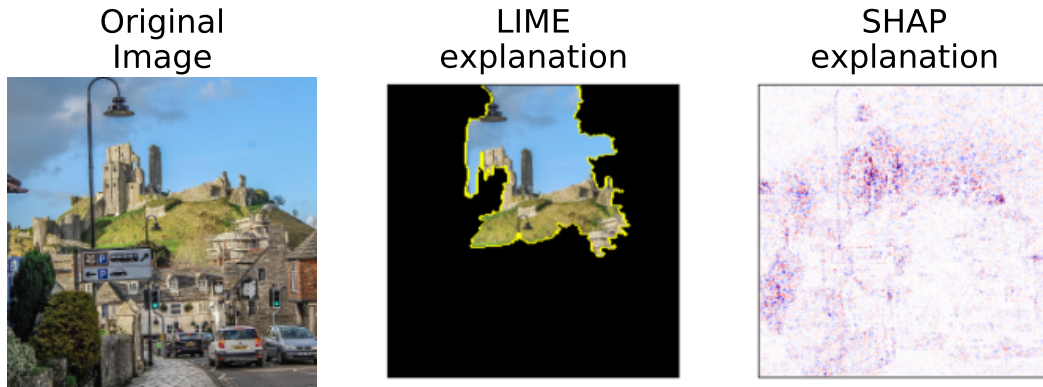


FIGURE 2.6: Comparison between LIME and SHAP explanation methods of image classification predictions made by VGG-16 Neural Network [96] pretrained on Imagenet dataset. From LEFT to RIGHT: original image, LIME explanation for class "castle", SHAP explanation of same class. LIME explanation is usually represented by a mask applied to the original input, that highlights what part of the image influenced the prediction the most. SHAP method creates standard relevance map, that attributes positive scores (red) to the features contributing to the prediction, and negative scores (blue) that were considered as evidence against the prediction.

Model-aware methods

In contrast with model-agnostic methods, model-aware (also known as model-specific) explainability algorithms use the underlying structure of the learner and explicitly work with computational graphs. While these methods require to be specifically crafted for different types of learning machines, model-aware methods usually use less computational resources and produce more accurate explanations.

Popular model-aware methods are:

- **Input \odot Gradient**

This method is considered to be a baseline approach for computing the relevance maps: it multiplies an input x with the gradient with respect to input, denoted $x \odot \frac{\partial f}{\partial x}$ [40]. This method addresses the "gradient saturation" problem in explanations, and reduce visual diffusion [2, 93].

- **Integrated gradients**

Integrated Gradients (IG) is an axiomatic local explanation algorithm that also addresses the "gradient saturation" problem. It assigns the relevance score to each feature by approximation of the integral of gradients of the model's output with respect to a scaled version of the input. [99]. Relevance attribution function, in this case, can be defined as

$$R(x, W) = (x - \bar{x}) \int_0^1 \frac{\partial f(x + \alpha(x - \bar{x}))}{\partial x} d\alpha,$$

where \bar{x} is a *baseline* that represents the absence of a feature in the input.

- **Guided Backprop**

Guided Backprop (GB) attributes relevance using *guided backpropagation*: the gradient of the target output with respect to the input, but negative gradient entries are set to zero while backpropagated through the ReLU functions [97].

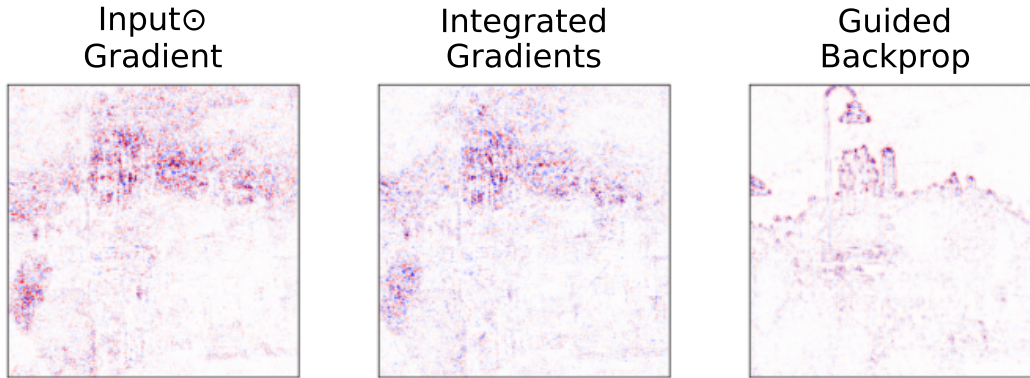


FIGURE 2.7: Comparison between model-aware explanation methods of image classification predictions made by a VGG Neural Network [96] on the same "castle" image as in Figure 2.6. From LEFT to RIGHT: Input \odot Gradient explanation, Integrated Gradients and Guided Backprop.

2.2.3 Layer-wise Relevance Propagation (LRP)

In this work, we focus on a particular explanation method — *Layer-wise Relevance Propagation* (LRP). LRP is a computationally cheap and simple approach that is based on the idea of backpropagating the relevance from the output through the layers of the network.

Layer-wise Relevance Propagation (LRP) [11] is a model-aware explanation technique that can be applied for feed-forward neural networks and it can be used for different types of inputs, such as images, videos, or text [7, 9]. Intuitively, the core idea underlying the LRP algorithm is that it uses the network weights and the neural activations created by the forward-pass to propagate the output back through the network until the input layer is reached. This propagation procedure is subject to a conservation rule — analogous to Kirchoff's conservation laws in electrical circuits [71] — on each step of backpropagation of the relevances from the output layer toward the input, the sum of relevances should remain the same. In each layer, LRP redistributes the relevance of the output node to the input node, proportionally to the contribution of each input neuron. Figure 2.8 illustrates LRP redistribution procedure.

The backpropagation of relevances from the output layer to the input features can be done in several ways. In the following, we present four different backpropagation rules that could be found in the LRP literature. Let the feed-forward neural network consist of $L > 0$ consecutive layers, where 0 is the input and layer L -th layer is a network's output. i and j be the neurons at layers l and $l + 1$, respectfully.

- **LRP-0 [11]**

The standard LRP-0 rule distributes the relevance in proportion to the contributions of each input to the neuron activation. Backpropagation of a relevance

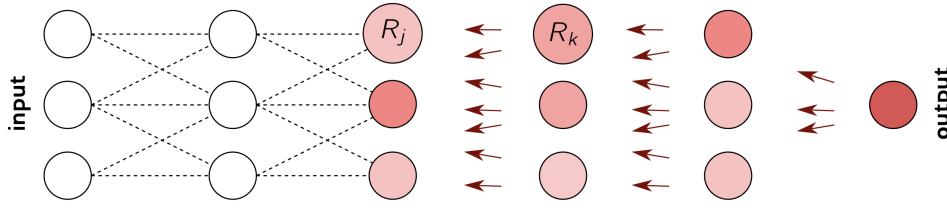


FIGURE 2.8: Illustration of the relevance redistribution procedure in LRP algorithm [71]. Redistribution procedure resembles the Kirchhoff's first law: each neuron redistributes to the lower levels value of relevance, that it received from the higher levels.

$R_i^{(l)}$ for the l -th layer from the output layer towards the input is achieved by recursively applying the following rule:

$$R_i^{(l)} = \sum_j \frac{z_{ij}^{(l+1)}}{\sum_{i'} z_{i'j}^{(l+1)}} R_j^{(l+1)},$$

where $z_{ij}^{(l+1)} = x_i^{(l)} w_{ij}^{(l+1)}$ is the activation of the $(l+1)$ -th layer computed in the forward pass and $\{w_{ij}^{(l+1)}\}$ are the learned weight parameters of the network between l -th and $(l+1)$ -th layers.

While the motivation behind this rule is quite simple and intuitive, it can be shown [93, 6] that explanations, produced by this method are genuinely equivalent to Input \odot Gradient. However, this explainability method can suffer from noisy gradients, that can occur during backpropagation.

- **LRP- ε** [11]

An enhancement of the basic LRP-0 rule consists of adding a small positive number $\varepsilon > 0 \in \mathbb{R}$ in the denominator:

$$R_i^{(l)} = \sum_j \frac{z_{ij}^{(l+1)}}{\varepsilon + \sum_{i'} z_{i'j}^{(l+1)}} R_j^{(l+1)}.$$

The idea behind the augmented procedure is that for small values $\sum_{i'} z_{i'j}^{(l+1)}$, relevances can take unbounded values. Thus, this unboundedness can be overcome by introducing a predefined stabilizer $\varepsilon > 0$.

- **LRP- γ** [71]

LRP- γ rule is another improvement to the standard LRP-0 rule. The main idea is to favor the effect of contributions made by positive weights over the contribution of the negative ones. The parameter $\gamma \geq 0$ is set to enhance positive contributions:

$$R_i^{(l)} = \sum_j \frac{x_i^{(l)} (w_{ij}^{(l+1)} + \gamma w_{ij}^{(l+1)+})}{\sum_{i'} x_{i'}^{(l)} (w_{i'j}^{(l+1)} + \gamma w_{i'j}^{(l+1)+})} R_j^{(l+1)}.$$

- **LRP-CMP [71]**

In our work, we use the best practice LRP rule, namely LRP-CMP (Composite) rule, which was recently published by Kohlbrenner et al. [45, 86] and LRP- ϵ rule [11]. LRP-CMP uses a combination of different basic LRP rules, i.e., LRP-0 rule, LRP- ϵ rule, and LRP- γ [11] for different layer types of the deep neural network and thus acts as an enhanced combination of those. The particular implementation of the LRP-CMP rule for the VGG-16 Neural Network and comparison with other LRP rules is illustrated in Figure 2.9.

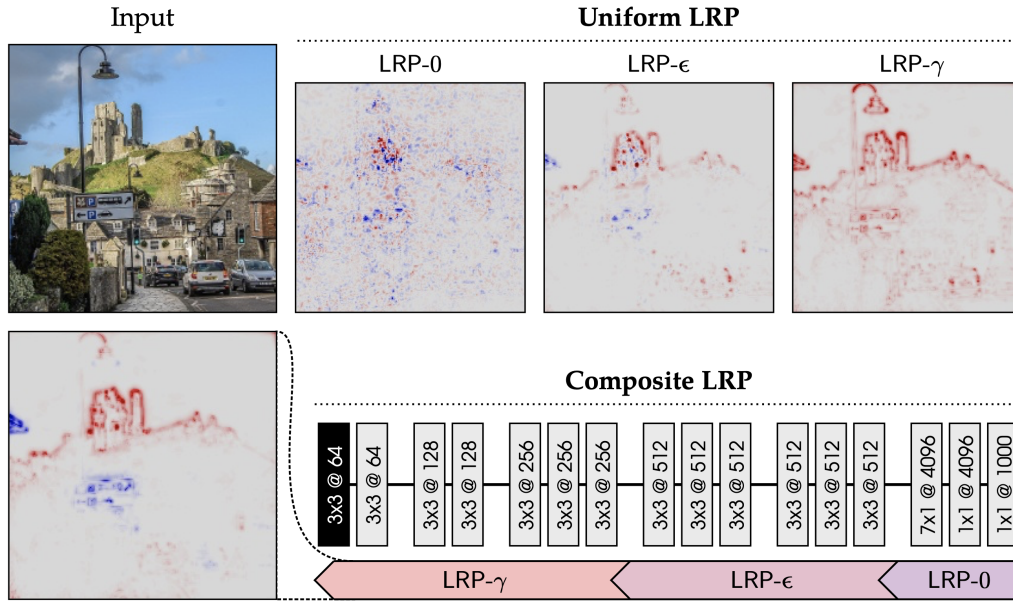


FIGURE 2.9: Illustration of LRP CMP rule: Top row: original image ("castle"), LRP-0 explanation, LRP- ϵ explanation, LRP- γ explanation, Bottom row: left — LRP CMP explanation of class "castle", right — illustration for which layers what LRP rules were used [71].

2.3 Bayesian Neural Networks

From a statistical perspective, DNNs are usually trained using the Maximum Likelihood principle, aiming to find a set of parameters (weights) that maximizes regularized maximum likelihood:

$$\hat{W} = \operatorname{argmax}_W \log p(W | \mathcal{D}_{\text{tr}}). \quad (2.1)$$

This procedure is sometimes called *maximum a-posteriori (MAP) optimization*, as it involves maximizing a posterior [113]. The most commonly used loss functions and regularizers fit into this framework, such as categorical cross-entropy for classification tasks or MSE for regression. Although this procedure is efficient since networks learn only a fixed set of weights, vanilla networks suffer from the inability of specifying uncertainties on the learned weights and subsequently on the prediction. Predictions produced by NN have biases which, in the case of MAP, cannot be trivially distinguished from being due to the nature of data or due to the model [13]. In contrast, Bayesian neural networks (BNNs) estimate the posterior *distribution* of weights, and thus, provide uncertainty information on the prediction which

can provide probabilistic guarantees on predictions. Figure 2.10 illustrates the difference between standard and Bayesian Neural Networks. Particularly, in critical real-world applications of deep learning—for instance, medicine [30, 35, 44] and autonomous driving [46, 109]—where predictions are to be highly precise and wrong predictions could easily be fatal, the availability of uncertainties of predictions can be of fundamental advantage.

[26, 106] describe the history and origins of BNNs. The study of BNNs dates back to the 1990s, mainly to the work of Tishby et. al [101]. It was shown that by specifying the prior distribution of weights of a neural network, by using the Bayes Rule the approximate posterior can be found, however, no practical means for inference were provided. Other notable works [33, 61] contributed in many ways to a drastic development of the field of Bayesian Neural Networks. Another wave of interest from a scientific community came after the so-called AI-revolution of 2012: with the popularity of the Deep Learning field of Bayesian Deep Learning has emerged roughly in 2014 [106]. Nowadays, collinearly with the progress in the field of DL, Bayesian Neural Networks continue to gain more and more popularity among practitioners across a wide spectrum of different fields.

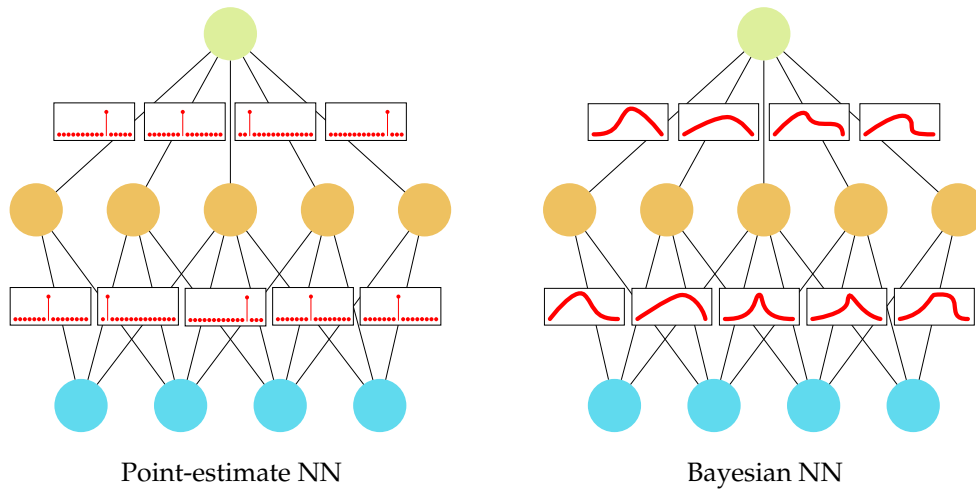


FIGURE 2.10: Standard point estimate NN (LEFT) and BNN with learned probability distribution over the weights (RIGHT) [39].

Let $f(\cdot; W) : \mathbb{R}^d \rightarrow \mathbb{R}^k$ be a feed-forward neural network with the weight parameter $W \subset \mathcal{W}$. Given a training dataset $\mathcal{D}_{\text{tr}} = \{x_n, y_n\}_{n=1}^N$, Bayesian learning (approximately) learns the posterior distribution

$$p(W|\mathcal{D}_{\text{tr}}) = \frac{p(\mathcal{D}_{\text{tr}}|W)p(W)}{\int_{\mathcal{W}} p(\mathcal{D}_{\text{tr}}|W)p(W)dW}, \quad (2.2)$$

where $p(W)$ is the prior distribution of the weight parameter. After training, the output for a given test sample x is predicted by the distribution:

$$p(y|x, \mathcal{D}_{\text{tr}}) = \int_{\mathcal{W}} p(y|f(x; W))p(W|\mathcal{D}_{\text{tr}})dW. \quad (2.3)$$

Since the denominator of the posterior, shown in Eq. (2.2), is intractable for neural networks, many approximation methods have been proposed, e.g., Laplace approximation [81], variational inference [29, 77], MC dropout [23], variational dropout [41, 68], and MCMC sampling [108]. Such approximation methods support efficient

sampling from the approximate posterior, which allows us to obtain output samples from the predictive distribution, given in Eq. (2.3), for uncertainty evaluation along with prediction for a test sample x . Classical non-bayesian training procedure could be also seen as performing approximate Bayesian inference, using the approximate posterior $p(W|\mathcal{D}_{\text{tr}}) \approx \delta(W = \hat{W})$, where δ is the Dirac delta function.

In the following sections, we will briefly describe popular posterior approximation methods that were used in the following work.

2.3.1 Variational Inference

Modern approaches to develop efficient Bayesian inference algorithms mostly perform *variational inference*: given the training data \mathcal{D}_{tr} , the variational inference aims on approximating the posterior distribution of the parameters $p(W|\mathcal{D}_{\text{tr}})$ with a variation distribution, $q(W|\mathcal{D}_{\text{tr}}, \phi)$ [78], that is restricted to belong to a family of distributions of simpler form (for example, family of Gaussian distributions). Parameters ϕ are optimized in order to minimize predefined dissimilarity measure between $p(W|\mathcal{D}_{\text{tr}})$ and $q(W|\mathcal{D}_{\text{tr}}, \phi)$. Usually, Kullback-Leibler divergence [49] is used as a dissimilarity function between distributions.

$$\text{KL}(q \parallel p) = \int q(W | \mathcal{D}_{\text{tr}}, \phi) \log \frac{q(W | \mathcal{D}_{\text{tr}}, \phi)}{p(W | \mathcal{D}_{\text{tr}})} dW.$$

Unfortunately, $p(W | \mathcal{D}_{\text{tr}})$ is usually not tractable. To counter that problem, evidence lower bound (ELBO) is introduced:

$$\text{ELBO}(\phi) = \int q(W | \mathcal{D}_{\text{tr}}, \phi) \log \frac{p(W) \prod_{i=1}^N p(y_i | f(x_i; W))}{q(W | \mathcal{D}_{\text{tr}}, \phi)} dW$$

It could be shown [116] that log evidence $p(D)$ could be viewed as:

$$\log p(D) = \text{ELBO}(\phi) + \text{KL}(q \parallel p)$$

As the $\log p(D)$ is fixed with respect to variational distribution $q(W|\mathcal{D}_{\text{tr}}, \phi)$, maximization of evidence lower bound (ELBO) is equivalent to minimization of $\text{KL}(q \parallel p)$. By appropriate choice of variational distribution, ELBO becomes tractable to compute. The resulting maximization problem $\text{ELBO}(\phi) \rightarrow \max_{\phi}$ can be solved using stochastic gradient descent.

Popular variational inference methods for training BNNs include:

- **Bayes by Backprop** [12]

One of the first major breakthroughs in the field of BNN was *Bayes by Backprop* algorithm [12]. The method is a variational inference method that utilises the *Local Reparametrisation Trick* [42]. As the name implies, a variational distribution that maximizes ELBO is learned by a backpropagation. Weights uncertainties that were afforded by Bayes by Backprop trained networks were used successfully in different applications [57, 36]. Throughout this work, we will be using the Bayes by Backprop trained CNNs, described in [92].

- **MC Dropout** [23]

Originally, the Dropout method [98] was proposed as a regularisation technique that counters the problem of over-fitting in NNs by randomly dropping hidden units with pre-defined probability p during the training phase. This

approach restricts hidden units to co-adapt too much: dropout procedure prevents any neuron in the NN to excessively rely on the output of some other particular neuron, driving it to connect to the population behavior of its inputs. In a testing phase, the whole network is used for a prediction, thus giving a deterministic result. Figure 2.11 is illustrating the dropout procedure. This approach was shown to effectively reduce over-fitting, improve the generalization of the learning machine, and yield remarkable improvements in different tasks such as ImageNet classification [34].

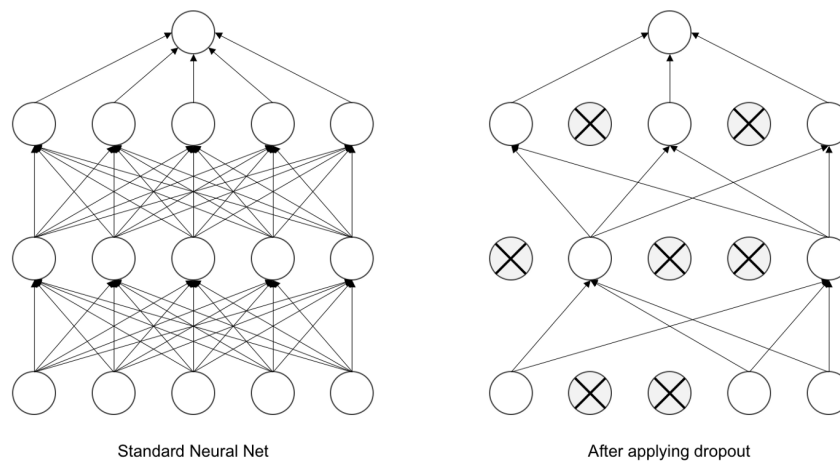


FIGURE 2.11: Illustration of a dropout procedure: LEFT: standard NN, RIGHT: NN with applied dropout [1].

The dropout process can be interpreted as multiplicative noise on the parameter. Therefore, it was shown [23] that dropout training can be seen as variational inference with the variational distribution restricted to two-component mixture distributions. MC dropout can be performed simply by turning on the dropout procedure in the test phase and taking the output random samples as the prediction from networks with the weight parameter sampled from the approximate posterior.

- **Noisy KFAC [118]**

Noisy KFAC is another variational inference method for BNNs. It is based on the idea of using natural gradient ascend [4] with adaptive weight noise to maximize evidence lower bound (ELBO). For natural gradient computation, it is necessary to obtain Fisher matrix F , and as modern neural networks may contain millions of parameters, storing the exact matrix is impractical. Authors use the Kronecker-Factored Approximate Curvature (K-FAC) [63] as an efficient way for approximating the Fisher matrix, used to perform efficient approximate for the natural gradient.

2.3.2 Laplace approximation

Standard NNs have gained popularity and wide-spread adoption by performing MAP inference that only yields point estimates for the mode of the posterior distribution. However, the Laplace approximation allows to locally approximate the posterior by taking the second-order Taylor expansion around the mode:

$$\log p(W | \mathcal{D}_{\text{tr}}) \approx \log p(W | \mathcal{D}_{\text{tr}}) - \frac{1}{2}(W - \hat{W})^T \bar{H}(W - \hat{W}),$$

where $\bar{H} = \mathbb{E}(H)$ i.e. average Hessian of a negative log posterior. Assuming that \bar{H} is positive semi-definitive, it was shown [61, 81] posterior distribution over the weights could be approximated as follows:

$$p(W | \mathcal{D}_{\text{tr}}) \sim N(\hat{W}, \bar{H}^{-1}). \quad (2.4)$$

Thus, by computing the average Hessian we are able to approximate the posterior around the mode with a Normal distribution. However, it is unfeasible to compute or invert the Hessian matrix w.r.t. all of the weights altogether, so different approximations methods were proposed, such as diagonal and tri-diagonal approximations. The recent developments in second-order optimization showed that Kronecker-factored Approximate Curvature (K-FAC) [63] can be used effectively to approximate the Hessian, even for complex DNN with hundreds of thousands of parameters [81]. Approximation of Hessian does not require re-training the network, thus by performing the Laplace approximation it is possible to transform any MAP trained NN into the BNN.

Chapter 3

Explaining Bayesian Neural Networks

Bayesian Neural Networks is an important class of learning machines that combine the neural network design with stochastic modeling. The main advantage of BNNs over the standard DNNs lies in their ability to generate the distribution of the parameters that it has learned from the data and also to produce probabilistic guarantees on its predictions. Yet, to our knowledge, no method explains the decision-making process of such machines. In this chapter we introduce several novel methods for local explanations of the behavior of BNNs for the image classification problem: we start with the simple and intuitive method called *Mean LRP* and continue with more advanced methods like *B-LRP* and its enhancement *B-LRP+*. Finally, we describe the semi-autonomous method for finding different major strategies in the decision-making process — Bayesian Strategies Clustering (BSC) method.

3.1 Distribution of relevance maps

Previously, we defined the relevance map of an input x by $R(x; W) \in \mathbb{R}^d$. The relevance depends on the learned parameter value W , and for a fixed input x , $R(x, W)$ is a deterministic mapping. In BNN weights are not fixed and follow a posterior distribution, therefore, we can naturally consider the distribution of the relevance maps induced by the BNN

Given the posterior distribution of $W \sim p(W|\mathcal{D}_{tr})$, we can define the distribution of relevance as

$$p(R|x, \mathcal{D}_{tr}) = \int R(x, W)p(W|\mathcal{D}_{tr})dW. \quad (3.1)$$

Relevance samples can be obtained by computing the relevance for posterior parameter samples:

$$R(x; W) \sim p(R|x, \mathcal{D}_{tr}) \quad \text{if} \quad W \sim p(W|\mathcal{D}_{tr}). \quad (3.2)$$

Under the assumption of the correctness of the chosen explainability mapping R (i.e chosen explainability algorithm indeed explains the prediction of the neural network), the decision-making process of a Bayesian Neural Network could be explained by just visually looking over some number of samples from the relevance distribution. Figure 3.1 illustrates several relevance maps, sampled from the posterior distribution. Unfortunately, in real-world applications of BNNs, it is just not practical to look over a huge number of relevance heatmaps just per one particular

data point. Thus, aggregation methods over the samples should be introduced to ease the explanation of BNNs.

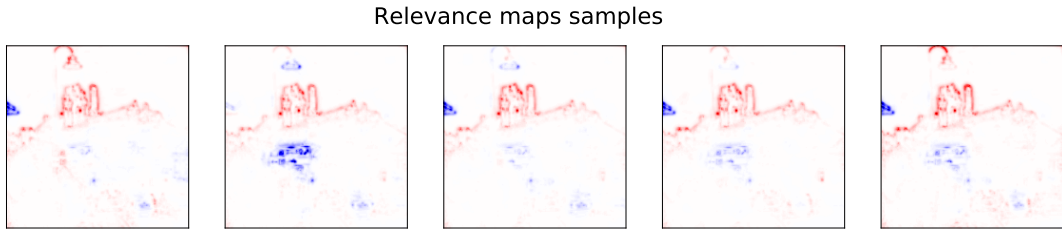


FIGURE 3.1: Several examples of LRP-CMP relevance maps that were sampled from the posterior distribution of VGG-16 CNN with MC Dropout, explaining the "castle" image. We can observe that castle itself was attributed positive relevance on each sample, however, it is noticeable that the relevance of a lamppost in the top-middle part of the image changes significantly from one sample to another.

3.2 Mean LRP

The method called *Mean LRP* is a simple and straightforward approach for explaining the BNNs: the Mean LRP method computes a relevance map for the network with weights set to the mean of the posterior distribution over its parameters:

$$\text{MeanLRP}(x; W) = R(x; \bar{W}). \quad (3.3)$$

We use the Mean LRP method as a baseline for the comparison of the efficiency of the other proposed methods.

3.3 B-LRP

Limitations of the Mean LRP is quite evident — this method does not use any information about the posterior distribution of weights. The next step would be an aggregation over the relevance maps distribution. Our proposal, which we call *Bayesian LRP* (B-LRP; illustrated in Figure 3.2), is to treat the relevance of a BNN as a random variable that follows Eq.(3.1), and use it to explain the network, with uncertainty information taken into account. Let $\{R_m\}_{m=1}^M$ be samples from the relevance distribution, obtained by Eq.(3.2). Then we define our B-LRP as follows:

$$\text{B-LRP}_\alpha(x; W) = \mathcal{P}_\alpha(\{R_m\}), \quad (3.4)$$

where $\mathcal{P}_\alpha(\{R_m\})$ is an operator computing the entry-wise (pixel-wise) percentile from the set $\{R_m\}$ of random samples.

B-LRP reveals the features where the explanation is uncertain: for small values of α , for instance, $\alpha = 5$ the fact that a feature has a positive relevance on B-LRP explanation suggests that this attribute has positive relevance in $1 - \alpha = 95$ percent of all computed relevance maps — this means there is strong evidence that this feature is relevant to the models' decision. Similarly, with high values of α we can find most

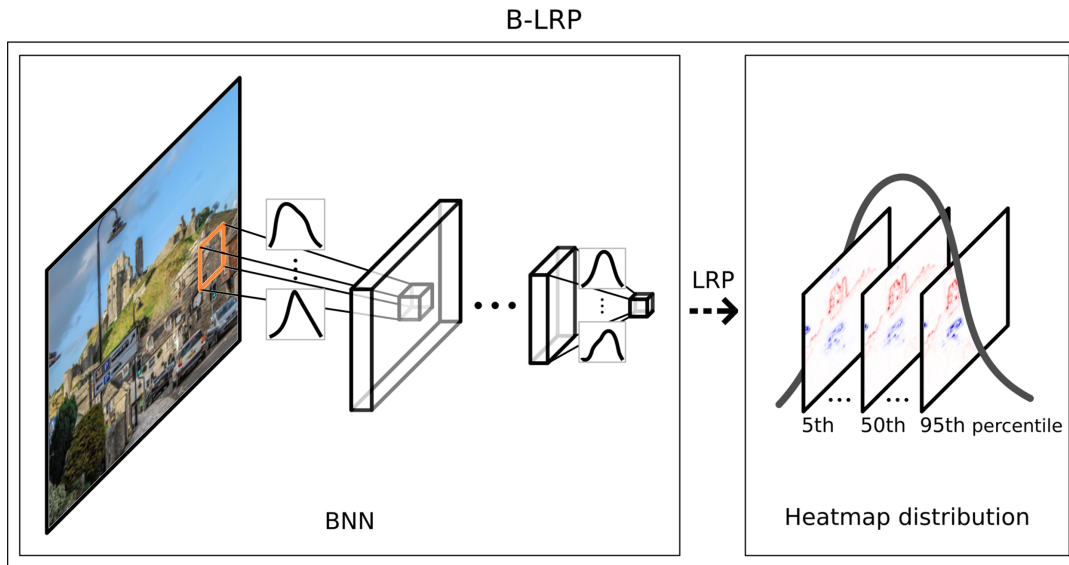


FIGURE 3.2: Overview of the proposed B-LRP procedure. For a given input, standard LRP generates from a trained neural network an explanation as a heatmap. Our B-LRP considers a Bayesian neural network (shown to the left), which induces a distribution over neural networks. Subsequently, applying LRP evokes a distribution of heatmaps (shown to the right). The variation in this distribution informs us about the (un)certainty in explanations. B-LRP considers a percentile of the heatmap distribution, leading to more cautious or risky explanations, depending on the chosen percentile.

certain features with negative relevances. Figure 3.3 illustrates B-LRP explanations for the "castle" image for VGG-16 CNN.

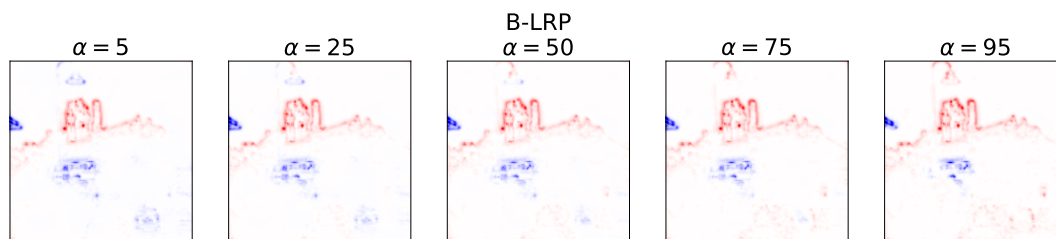


FIGURE 3.3: B-LRP explanations for different parameter α for "castle" image. We can observe from B-LRP, $\alpha = 5$ that in 95% of explanations castle on the image was attributed positive relevance, in contrast to other objects like the lamppost or the car in the bottom-left part of the image. On the other hand, B-LRP, $\alpha = 95$ demonstrates that at least 5% of explanations view the lamppost and a car as contributing towards the prediction.

A particular choice of the parameter α used in the B-LRP method should be guided by the application scenario: small values (0 – 25) of α are more favorable, for example, in use-cases when the main goal is to analyze *where* explanations are certain about positive relevances of features, while large values (75 – 100) are more suitable for the cases when it is important to understand if the particular feature was attributed with a positive relevance in at least some (small) subset of explanations.

In practice, the use of several different α parameters could help to better understand the decision-making process of a BNN. For a general practitioner, we propose to use the following set of alpha's values: [5, 25, 50, 75, 95] — by visualizing the following percentiles of a marginalized distribution, it is possible to detect the features, in which the explanation algorithm is certain about their impact on the prediction.

3.3.1 B-LRP +

B-LRP+ is the enhancement to a B-LRP method, that only visualizes features which impact to a prediction is certain. B-LRP+ with parameter $\alpha \in [0, 50]$ is defined as follows:

$$\text{B-LRP+}_\alpha(x; W) = \mathbb{1}_{\mathcal{P}_\alpha(\{R_m\}) > 0} \odot \mathcal{P}_\alpha(\{R_m\}) + \mathbb{1}_{\mathcal{P}_{100-\alpha}(\{R_m\}) < 0} \odot \mathcal{P}_{100-\alpha}(\{R_m\}),$$

where $\mathcal{P}_\alpha(\{R_m\})$ is an operator computing the entry-wise (pixel-wise) percentile from the set $\{R_m\}$ of random samples and \odot is the element-wise product.

The formula for B-LRP+ is quite intuitive: we visualize only features with "relevance certainty" equal to a $(100 - \alpha)$ — features that in $(100 - \alpha)$ percent of the samples showed the same contribution: positive or negative, towards the prediction. In the first term, we take features with only positive relevance from α percentile: that means that the explainability algorithm is certain that these attributes have a positive impact on the prediction in $(100 - \alpha)$ percent of the samples. The second term in the formula is following a similar idea, but for negative relevances. In the end, we achieve that only features, in which the explainability algorithm is confident in $(100 - \alpha)$ percent of the cases, are visualized. Figure 3.4 illustrates B-LRP+ visualizations for different parameters α .

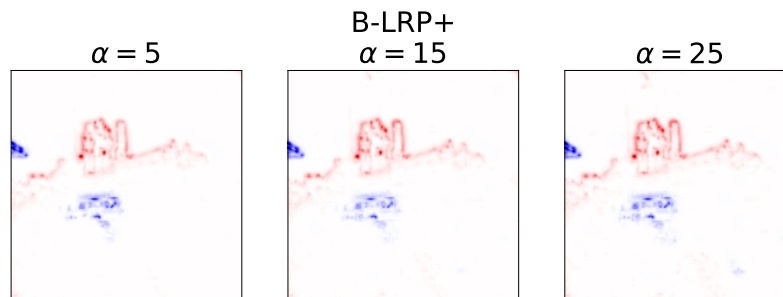


FIGURE 3.4: B-LRP+ explanations for "castle" image for VGG-16 network with MC Dropout. For B-LRP+ $\alpha = 5$ we can see that lamppost is deleted from an explanation — explanation consist only with attributes that 95% of their distribution in maintaining the same sign.

3.4 Bayesian Strategies Clustering (BSC)

The limitation both of B-LRP and B-LRP+ methods is that produced explanations are aggregations over the marginalized distributions of individual features — that means that both methods discard the information about correlation and interconnection between relevances in one saliency map. Thus, that leads to the fact that resulting explanations might not correspond to any of the existing strategies.

We propose a method called *Bayesian Strategies Clustering* (BSC) that aims to explain the decision-making of a BNN by investigating the "prime" strategies (relevance maps) of a learning machine. To decompose the behavior of BNN into groups of inter-similar saliency maps we perform clustering of the sampled relevance maps. While our method does not restrict a user in choosing an algorithm for clustering, we propose to use the SpRAy (**S**pectral **R**elevance **A**nalysis) clustering method. This method was initially introduced in [51] to solve a similar problem — process-wide spectrum of learned decision behaviors and detect unexpected or undesirable ones in a semi-automated manner. After the main clusters are found, the decision-making process could be explained by a collection of feature-wise averages over relevance maps in each cluster. Moreover, the number of saliency maps in each cluster (normalized by the number of sampled relevance maps) could identify the "strength" of each strategy.

Originally, the SpRAy method was used to investigate the typical traits in the decision-making process over the large collection of relevance maps, that were obtained from different data points from the source dataset. This approach resembles our initial problem: in contrast to an original setting of a SpRAy method, we want to exhibit and understand typical as well as atypical behavior in the BNN decision-making process. Similarly, we have a collection of relevance maps that we want to cluster in several groups, and the only difference from the initial setting is that all of these maps are obtained from one data point, instead of whole the dataset.

For image classification problem BSC employs SpRAy algorithm as follows [51]:

1. Relevance maps sampling

A collection $\{R_i\}_{i=1}^N$ of relevance maps is sampled from the posterior distribution.

2. Preprocessing of the relevance maps.

All relevance maps are normalized using the MinMax normalization procedure. If needed, all relevance maps should be made uniform in shape and size, for future clustering.

To speed up the clustering process and to produce more robust results we propose to use downsampling methods on the collection of samples. While the user is free to choose any dimensionality reduction method, we propose to use the Average Pooling method in order to achieve visually different strategies in different clusters.

3. Spectral Cluster (SC) analysis on pre-processed relevance maps.

Pre-processed relevance maps are clustered by Spectral Clustering method. For affinity matrix, necessary for SC method, matrix based on k-nearest-neighborhood relationships is used. In more detail, affinity matrix $M = (m_{ij})_{i,j=1,\dots,N'}$ measuring the similarity $m_{ij} \geq 0$ between all N samples R_i and R_j of a source dataset is constructed in a following way:

$$m_{ij} = \begin{cases} 1 & \text{if } R_i \text{ is among the } k \text{ nearest neighbors of } R_j \\ 0 & \text{else} \end{cases}$$

Since this rule is asymmetric, symmetric affinity is created matrix M by taking $m_{ij} = \max(m_{ij}, m_{ji})$. Authors of the original paper highlight that similar clustering results were obtained using the Euclidean distance, only with only small differences in eigenvalue spectra.

Laplacian L is computed from M as follows :

$$d_i = \sum_j m_{ij}.$$

$$D = \text{diag} [d_1, d_2, \dots, d_N].$$

$$L = D - M.$$

Matrix D is a diagonal matrix, which entities describe the measure of connectivity of a particular sample i with D being a diagonal matrix with entries d_{ii} describing the degree (of connectivity) of a sample i [51].

4. Identification of interesting clusters by eigengap analysis

By performing an eigenvalue decomposition on the Laplacian L , eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_N$ are obtained. The number of eigenvalues $\lambda_i = 0$ identifies the number of (completely) disjoint clusters within the analyzed set of data.

The final step of SpRAy is label assignment, which can then be performed using an (arbitrary) clustering method: in our work we use k-means clustering on the N eigenvectors. The number of clusters can be obtained by *eigengap* analysis [105]: it can be identified by eigenvalues close to zero as opposed to exactly zero, followed by an eigengap — rapid increase in the difference between two eigenvalues in the sequence $|\lambda_{i+1} - \lambda_i|$ [51].

Once clustering has been performed, "prime" strategies can be identified as the pixel-wise mean over each of the clusters. Optionally, clusters and average cluster strategies could be visualized by t-SNE [60] to compute a two-dimensional embedding of the relevance maps. As a measure of the importance of each strategy, the number of samples in the corresponding cluster could be taken.

3.5 Explaining non-Bayesian Neural Networks with Bayesian principles

Naturally, proposed methods can be applied only to the Bayesian Neural Networks. However, some approximate Bayesian learning can be performed as post-processing applied to pre-trained non-Bayesian networks, such as MC Dropout (Chapter 2.3.1) or Laplace approximation (Chapter 2.3.2), which can broaden the applicability of Bayesian LRP. These post-processing procedures for "Bayesianizing" non-Bayesian learning machines do not only broaden the applicability in terms of models but also offer another use of Bayesian LRP.

Specifically, we can view standard LRP as a statistic method that behaves similarly to the median and mean. From this view, one can assess the reliability/uncertainty of standard LRP by using Bayesian LRP. In this manner, BNNs can make the uncertainty of explanation apparent for any existing explanation method and even for non-Bayesian learning machines. We study the application of proposed method for a non-bayesian network in more detail in Chapter 4.3.

Chapter 4

Experiments

This chapter is devoted to a practical examination of the proposed explainability methods for BNNs in an image classification setting. We demonstrate the usefulness of our proposed B-LRP and B-LRP+ methods along with the BSC method on different datasets and models. For quantitative evaluation of the performance of the methods, we introduce a Bayesian pixel-flipping method — a simple enhancement of the standard pixel-flipping technique, adjusted for a Bayesian setting. We discuss different pixel perturbation policies that were employed along with the specifics of the visualization of saliency maps.

4.1 Methodology

4.1.1 Quantative evaluation metric

Pixel-flipping

Usually, for a quantitative evaluation of local explanation methods in a standard non-bayesian setting the pixel-flipping method is employed [85] — a practical technique to quantify the goodness of an explanation with respect to a specific decision of a trained model. Given the relevance map (e.g., in the form of heatmap), at first, the pixels of the input image x are ranked in descending order of their relevance scores, i.e., pixels with higher relevance scores are ranked first. Then, the original pixel values are iteratively perturbed (e.g., set to zero or replaced with random values), according to the ranking. Namely, the pixel with k -th highest relevance score is perturbed at the k -th iteration. The prediction score (output of the trained model for the perturbed input) is evaluated and recorded at each step. Thus, the pixel-flipping algorithm yields the decaying curve, which illustrates the performance of the explanation on the particular image: the steeper the prediction score drops the better is relevance map in terms of explaining the decision-making process of NN.

Typically, pixel-flipping curves are computed over a set of images and the resulting curve is obtained by averaging. It is later visualized as a function of k , where k corresponds to the percentage of flipped pixels. For a quantitative measure to compare algorithms, one can take Area Under the Curve (AUC) metric: the smaller AUC is, the better algorithm is in explaining.

The pixel-flipping procedure is summarized in Algorithm 1.

Algorithm 1 Pixel-Flipping [85]

Require: x – input image, R – relevance map, f – trained model.**Ensure:** $scores$ – the sequence of decaying prediction scores.

```

scores  $\leftarrow []$ 
for  $p$  in argsort(- $R$ ): do
    Perturb the pixel  $x_p$ .
    scores.append( $f(x, W)$ ).
end for
return scores

```

Bayesian pixel-flipping

In contrast with standard NNs, Bayesian Neural Networks induce the distribution of predictions, rather than a point estimate. Thus, adjustment to a pixel-flipping algorithm is needed. In the Bayesian case, we evaluate changes in the mean of the prediction distribution – each time pixel is perturbed, we sample the posterior and average over T prediction scores, where T is a parameter of the method.

Algorithm 2 Bayesian Pixel-Flipping

Require: x – input image, R – relevance map, f – trained model, T – number of samples for averaging the prediction.**Ensure:** $scores$ – the sequence of decaying prediction scores.

```

scores  $\leftarrow []$ 
for  $p$  in argsort(- $R$ ): do
    Perturb the pixel  $x_p$ .
    Initialise score  $\leftarrow 0$ .
    for  $i = 1$  to  $T$ : do
        Sample  $W_i \sim p(W | \mathcal{D}_{tr})$ .
        score  $\leftarrow score + \frac{f(x, W_i)}{T}$ .
    end for
    scores.append(score).
end for
return scores

```

Pixel perturbation polices

An important part of the pixel-flipping algorithm is the pixel perturbation procedure. By perturbing pixels we aim to delete information from the image, however, an ideal method deletes information without introducing spurious structures. Moreover, it neither should disrupt image statistics nor move the corrupted image far away from the data manifold [85].

To conduct a correct quantitative evaluation of proposed methods we employ 3 different methods, each with own motivation:

- **Random choice:** pixel is replaced by a randomly chosen neighbor from the original image. The neighborhood is a rectangular window with a predetermined width of l .

This type of perturbation policy sustains original distribution in RGB values while destroying local patterns.

- **Gaussian blur:** original image is blurred and the pixel is replaced with a blurred pixel from the original image. For blurring, Gaussian blur is used [89]. This perturbation policy modifies the values of the pixel, yet does not introduce visual artifacts.
- **Combination of random choice and blurring:** again, the original image is blurred and the pixel is replaced with a randomly chosen blurred pixel from the neighborhood in the original image. This method combines both described above methods and modifies the pixel's values while destroying local patterns.

Figure 4.1 illustrates perturbation policies that were used during the quantitative evaluation of the performance of explainability methods.



FIGURE 4.1: Images, illustrating different perturbation policies after pixel-flipping of 25% of pixels based on LRP CMP explanation. From LEFT to RIGHT: **(a)** — random choice policy, **(b)** — Gaussian blur policy, **(c)** — composite policy.

4.1.2 Visualization parameters

For visualization, each relevance map is normalised using the MinMax transformation [11], which maps positive relevances to $[0, 1]$ and negative relevances $[-1, 0]$. Namely, the positive relevances are divided by the maximal positive relevance over the pixels, and the negative relevances are divided by the absolute value of the minimal negative relevance over the pixels.

Normalized relevance maps are visualized using 'seismic' colormap¹, which attributes red tones to pixels with positive relevances and blue tones to the pixels with negative relevances.

Quite often researchers are not interested in particular relevances of attributes, but the order of attributes in terms of importance to a prediction. For example, the pixel-flipping algorithm uses only the ranking of features by descending relevances. In our work, we use a specific type of visualization — *rank map*. On a rank map, only k percent of pixels with the highest relevance are visualized.

Figure 4.2 illustrates 2 colormaps, used for the visual examples.

¹<https://matplotlib.org/3.1.0/tutorials/colors/colormaps.html>

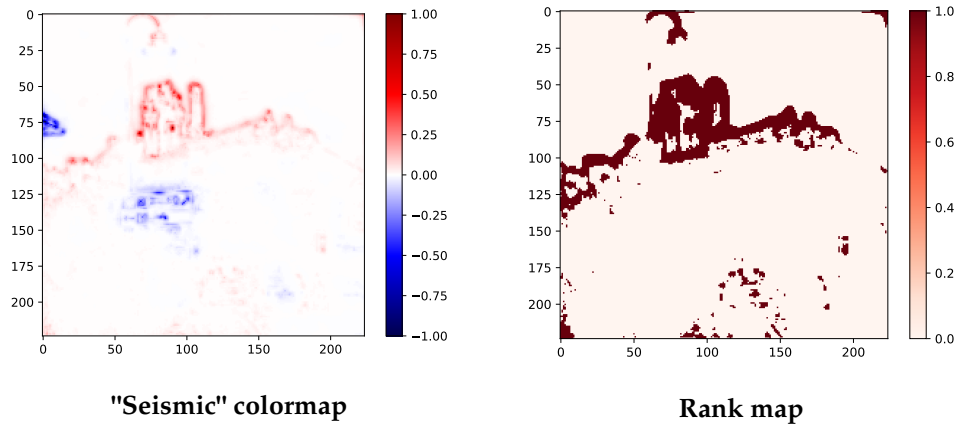


FIGURE 4.2: LRP CMP visualisation of castle image, LEFT — standard "seismic" relevance map, RIGHT — rank map with top 10% of pixels with highest relevance.

4.2 Bayes by Backprop: MNIST

In our first experiment, we will examine the performance of proposed explanation methods with Bayesian Neural Network trained by Bayes by Backpropagation [92]. Standard LeNet neural network architecture [53] was used for training on MNIST handwritten digit dataset, using the choice of parameters from [92]². For explanations, LRP- ϵ rule ($\epsilon = 10^{-9}$) was used as a relevance attribution function.

4.2.1 Visual Inspection

The B-LRP method allows us to access statistics of the relevance map's distribution. Figure 4.3 illustrates explanations, produced by B-LRP for 3 different images, with 1000 relevance maps sampled from the posterior per each image. On this figure, we observe that for $\alpha = 95$ positive relevances are attributed across almost the whole digit, in comparison with the Mean LRP. With $\alpha = 5$, we can notice that positive relevances are attributed to a smaller fraction of pixel — those are the features that BNN considers to be contributing towards the true class in 95% of all cases.

B-LRP+ aims to visualize parts of the input, the contribution of which is certain for the model. Figure 4.4 compares explanations produced with different parameter α . Positive relevances at B-LRP+ $\alpha = 5$ illustrate regions, that contributed positively to a true class in 95% of all samples from the posterior.

More illustrations, both for B-LRP and B-LRP+ could be found in the Appendix A.

4.2.2 Quantitative evaluation

Evaluation of the performance of each described method is done by conducting the Bayesian pixel-flipping on 500 images, randomly sampled from the MNIST test-set. The model score on each step of Bayesian pixel-flipping is evaluated $T = 100$ times. Parameters for perturbation methods were chosen as follows: we employ 2 Random Choice policies with different window size: $l = [7, 13]$ for methods one and two

²Code for training the BNN could be found at <https://github.com/kumar-shridhar/PyTorch-BayesianCNN>

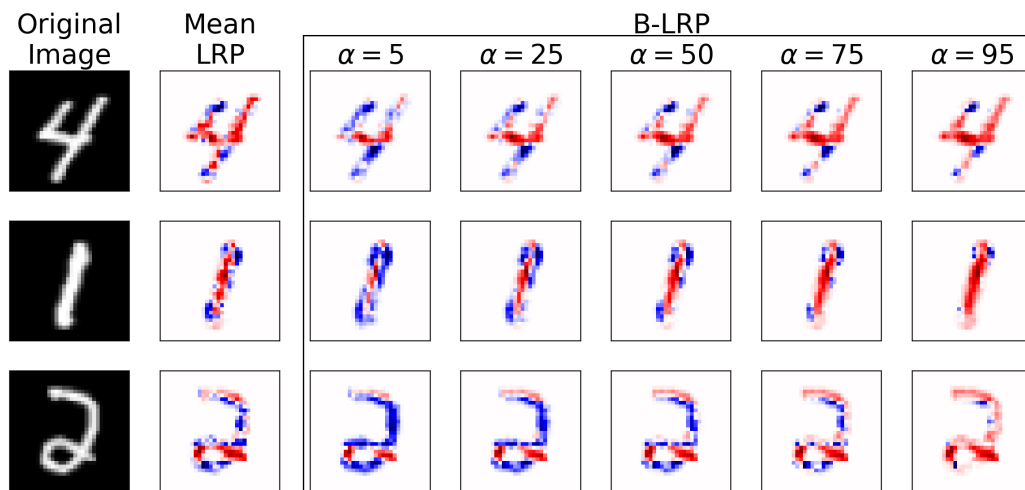


FIGURE 4.3: Comparison between Mean LRP method (Left) and different parameters for B-LRP (Right).

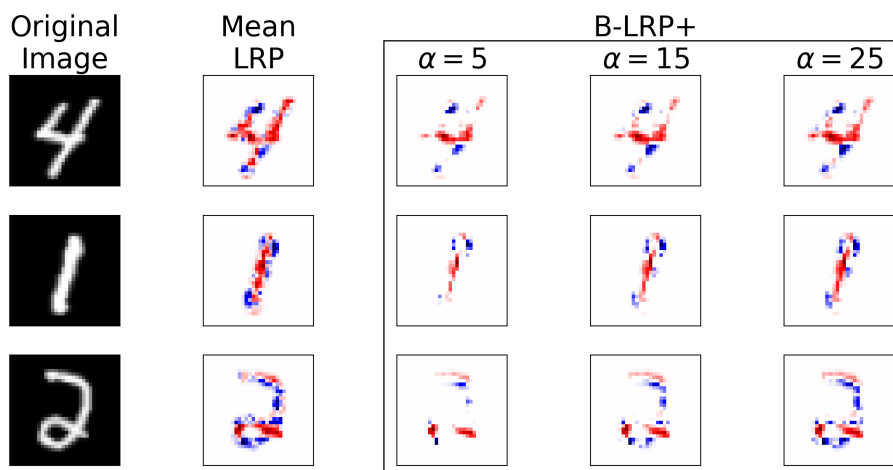


FIGURE 4.4: Comparison between Mean LRP method (Left) and different parameters for B-LRP+ (Right)

respectfully, Gaussian blur with $\sigma = 4$ and Composite method that combines the Random Choice method one and Gaussian blur.

Figure 4.5 illustrates average pixel-flipping curves for all 4 methods for the first 25% of pixels. As expected in all methods, the explainability algorithms perform better than random relevance map. Visually, we can assess that the B-LRP with $\alpha = [75, 95]$ are outperforming other methods. This is expected behavior, as from Figure 4.3 we observed, that for B-LRP $\alpha = 95$, positive relevance covers almost the whole digit. Visual evaluation of the performance of explainability methods is confirmed by the AUC performance, the results for which could be found in Table 4.1.

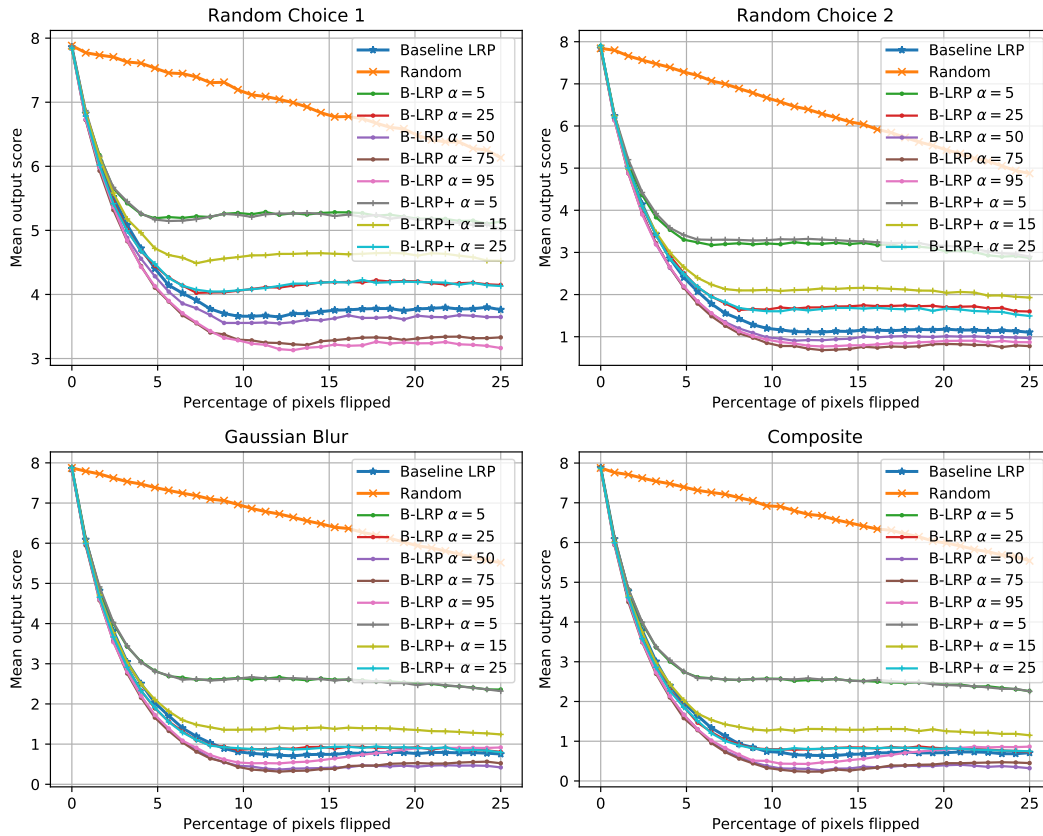


FIGURE 4.5: Bayesian pixel-flipping performance comparison between different B-LRP, B-LRP+ and Baseline LRP (Mean LRP) on MNIST.

Method	Random choice 1	Random choice 2	Gaussian blur	Composite
<i>Random</i>	0.8213	0.7924	0.8451	0.8446
<i>Mean LRP</i>	0.2183	0.1686	0.1612	0.1608
<i>B-LRP</i> $\alpha = 5$	0.4731	0.3872	0.3482	0.3442
<i>B-LRP</i> $\alpha = 25$	0.2792	0.2218	0.1667	0.1655
<i>B-LRP</i> $\alpha = 50$	0.1923	0.1438	0.1202	0.1191
<i>B-LRP</i> $\alpha = 75$	0.1389	0.1234	0.1199	0.1182
<i>B-LRP</i> $\alpha = 95$	0.1292	0.1317	0.1454	0.1446
<i>B-LRP +</i> $\alpha = 5$	0.4691	0.3978	0.3479	0.345
<i>B-LRP +</i> $\alpha = 15$	0.3622	0.2651	0.2163	0.2128
<i>B-LRP +</i> $\alpha = 25$	0.2796	0.2163	0.1664	0.1644

TABLE 4.1: AUC scores for different explainability methods (rows) in MNIST experiment for 4 different perturbation policies. Lower is better.

4.2.3 Bayesian Strategies Clustering

We demonstrate the usefulness of the described BSC (Chapter 3.4) method on a particular example, describing all the steps that need to be performed to obtain the final explanation. Figure 4.6 illustrates the resulting explanation of BNN by the BSC

method.

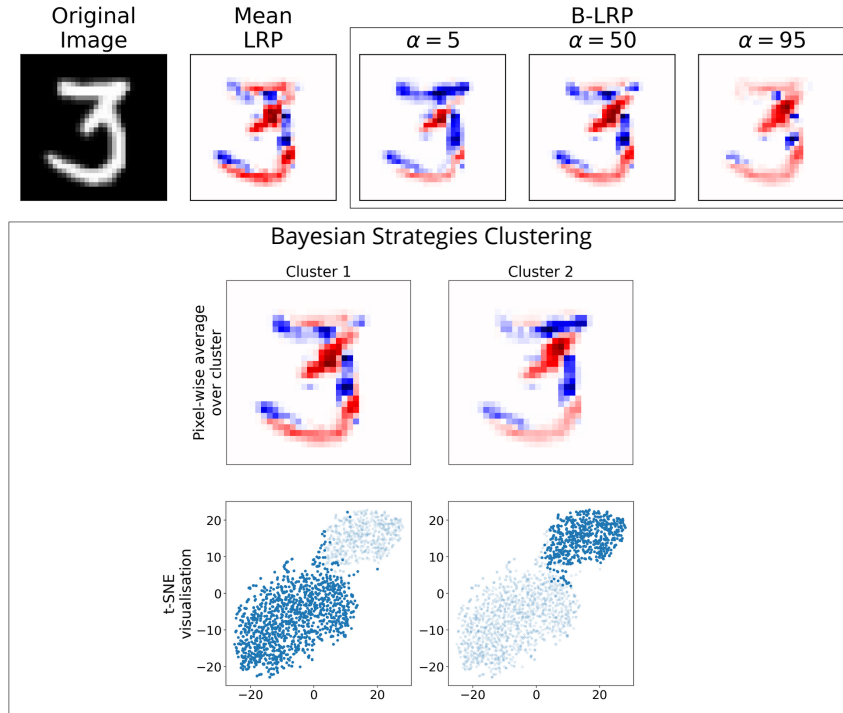


FIGURE 4.6: Illustration of explanations, computed by B-LRP and BSC methods. First row corresponds to the B-LRP explanations of the image, in the latter part of the image are "prime" strategies — pixel-wise average of relevance maps inside of clusters, coupled with a t-SNE visualisation of relevance maps distribution with highlighted (blue) cluster.

To perform Bayesian Strategies Clustering, $N = 2000$ relevance maps were sampled from the posterior distribution. As described in Chapter (Chapter 3.4), collection of relevance samples $\{R_i\}_{i=1}^N$ was down-sampled with Average Pool operation with kernel size of 2. For affinity matrix, parameter k in k -nearest Neighbour algorithm was chosen be 100.

The number of clusters is determined by eigengap analysis (Chapter 3.4). Figure 4.7 illustrates the distribution of 50 smallest eigenvalues: we observe that there is an eigengap between the second and third eigenvalue, thus the number of clusters was set to 2. Figure 4.8 demonstrates t-SNE two-dimensional visualization of the collection of downsampled relevance maps.

From Figure 4.6 we can observe the differences in the "prime" strategies: the differences in relevance attribution are clearly visible at the top of the digit. Illustrations of the relevance maps from different clusters could be found in the Appendix A, in Figures A.3 and A.3, for Cluster 1 and 2, respectfully.

4.3 MC Dropout: Imagenet

Next, we demonstrate the usefulness of the proposed methods for explaining the decision making in a non-bayesian case. In this experiment, widely used VGG16 network [96] pre-trained on the Imagenet dataset is employed with MC Dropout,

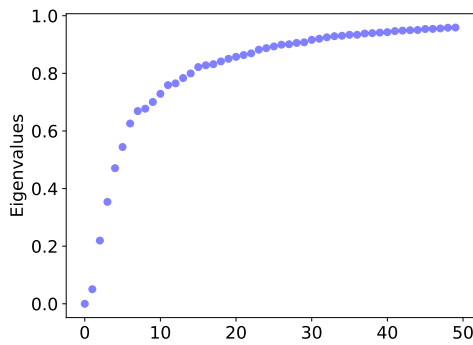


FIGURE 4.7: Distribution over the 50 smallest eigenvalues.

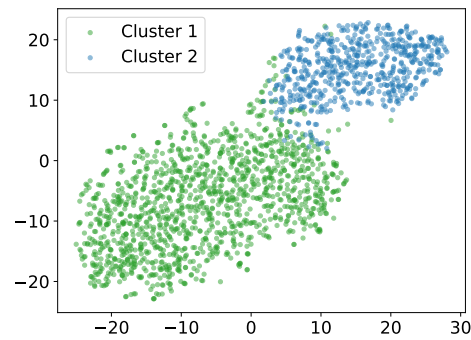


FIGURE 4.8: t-SNE visualisation of down-sized relevance maps.

and LRP-CMP (Chapter 2.2.3) explainability method is used as relevance attribution function.

For the evaluation of the performance of the proposed methods, a small subset of Imagenet, consisting of 5 classes: "castle", "lemon", "llama", "wine" and "tiger cat", 1000 random images per each class were downloaded ³.

4.3.1 Visual inspection

Figure 4.9 illustrates the result of the B-LRP procedure. Similar to the previous experiments, from B-LRP $\alpha = 5$, we can extract the main features, that contributed towards true prediction: for example, for the first "llama" image we can observe that 95% of all strategies that were sampled using the information about the nose and ears of the animal to make a correct prediction. In the case of B-LRP $\alpha = 95$, we can observe what features were used at least in 5% of the samples — we observe that for each of three images, the network "looked" at all main features of the class. We notice that in $\alpha = 95$ the whole head of the llama was attributed with positive relevance, in contrast, to Mean LRP and B-LRP with lower α . More visualizations for B-LRP method could be found in the Appendix B.

Figure 4.10 demonstrates B-LRP+ explanations for the same images. We observe the same traits but in different visualizations: parts, where BNN is uncertain — features from the image that have not been consistently attributed to being positive or negative, are deleted from an explanation. For example, we see that for the llama image, the network is certain about negative relevances from the bottom-center part of the image and positive relevance from the nose and ears of the animal.

4.3.2 Quantitative evaluation

In this experiment we employ a standard pixel-flipping method for evaluation of the performance of explainability methods — thus, we can compare the performance of the proposed methods for the non-bayesian case. In case of MC Dropout, the Mean LRP method is just equivalent to the standard explanation for the network, due to the

³To download a subset of Imagenet dataset, following library was used: <https://github.com/mf1024/ImageNet-Datasets-Downloader>.

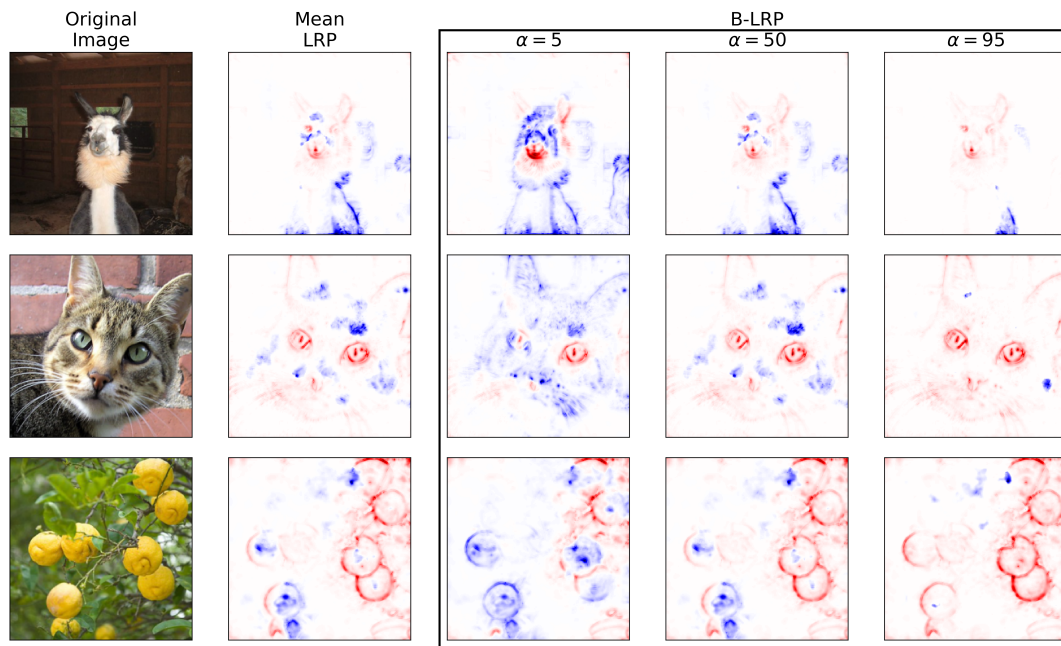


FIGURE 4.9: Comparison between different B-LRP methods (Right) and Mean LRP (Left) for 3 different random images from Imagenet dataset. From top to bottom: "llama", "tiger cat", "lemons". We can observe that in contrast to the Mean LRP, B-LRP with $\alpha = 95$ attributes positive relevance over the whole area, where the main features of the true class are distributed: llama's head, cat's head and all the lemons.

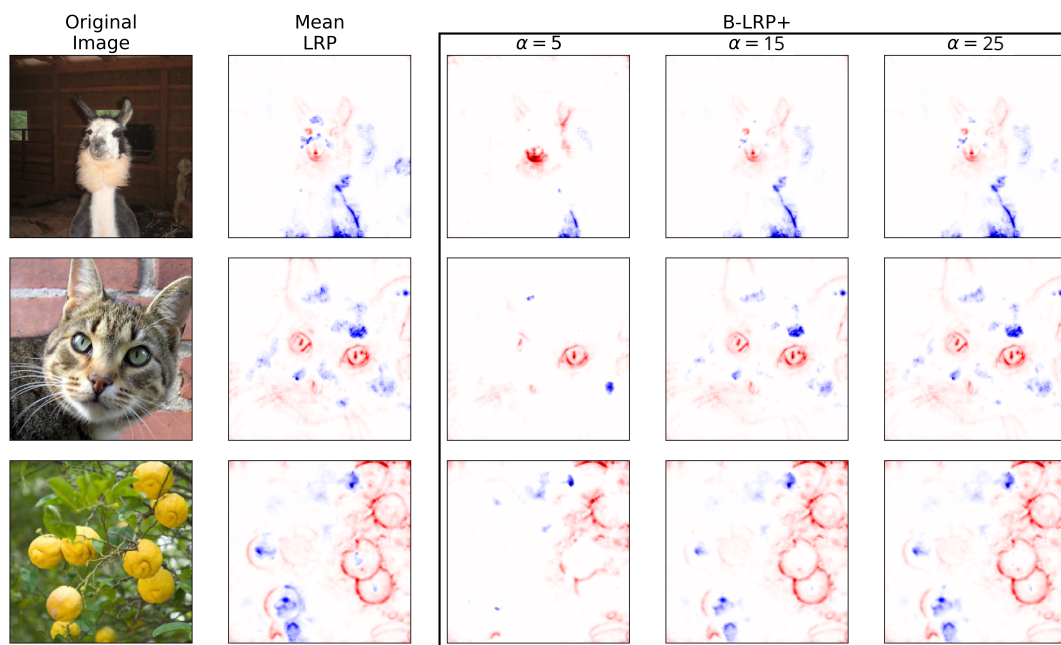


FIGURE 4.10: Comparison between different parameters for B-LRP+ methods (RIGHT) and Mean LRP (LEFT) for 3 different random images from Imagenet dataset.

fact that the weights in dropout trained networks are the mean of the posterior for

MC Dropout [23].

Figure 4.11 illustrates the performance of each explainability method in pixel-flipping evaluation for 100 randomly sampled images with 50% perturbed pixels. Same perturbation policies as in the previous example were used. For all pixel perturbation policies, we observe similar behavior for all methods for the first 20% of pixels. However, after we crossing the 20 percent threshold, we witness that B-LRP with higher α performs better than others, in particular for $\alpha = 95$. This is supported by an AUC results from Table 4.2: best performance is shown by the B-LRP $\alpha = 95$ and $\alpha = 75$ methods.

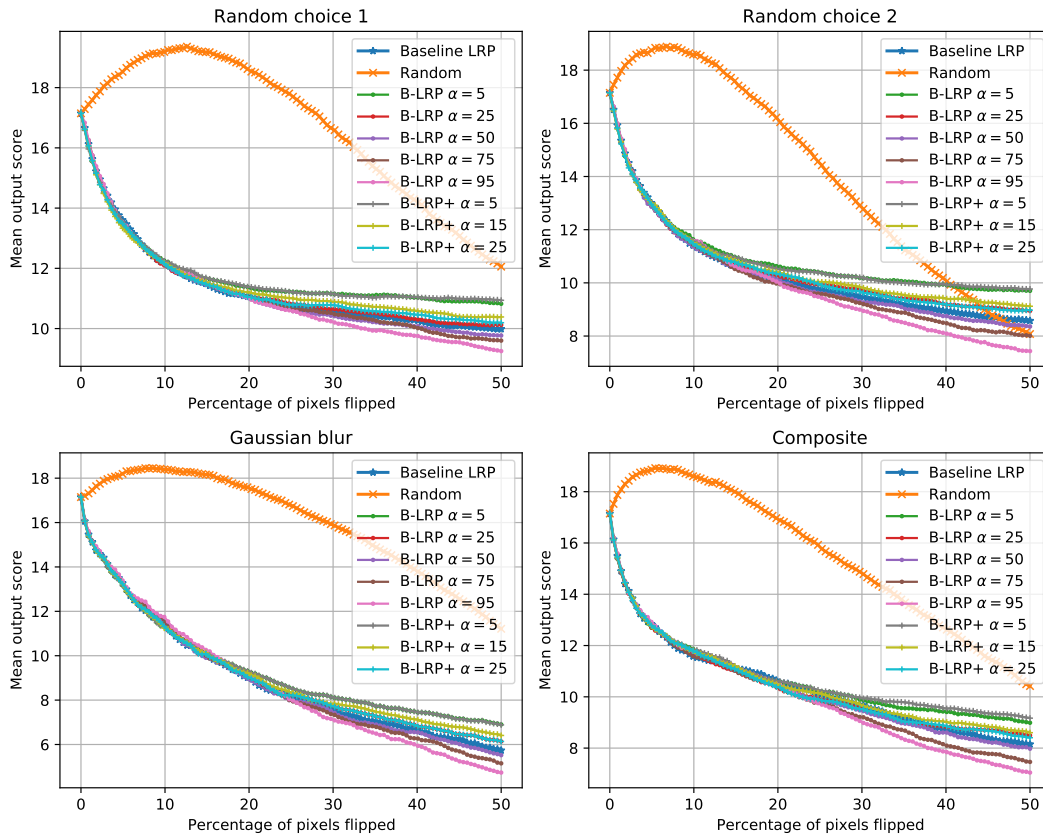


FIGURE 4.11: Pixel-flipping comparison of various methods with 4 different pixel perturbation policies.

4.3.3 Bayesian Strategies Clustering

In the same fashion as in the previous example, we demonstrate the effectiveness of the BSC method on two particular examples. For this experiment, we omit clustering details and concentrate only on the obtained results and their analysis. For both examples, Average Pooling with kernel size of 16 was used as pre-processing for the relevance maps and parameter k for computing the affinity matrix was set to 100.

Figure 4.12 illustrates B-LRP and BSC explanations for the "lemon" class. From B-LRP we can already observe that some of the relevance maps highlight lemon in the bottom-left corner of the image differently in terms of its contribution towards the true prediction: this area has negative relevance in B-LRP $\alpha = 5$, close to zero relevance for $\alpha = 50$ and slight positive relevance at $\alpha = 95$. BSC method finds 6

Method	Random choice 1	Random choice 2	Gaussian blur	Composite
<i>Random</i>	0.9478	0.6938	0.913	0.8391
<i>Mean LRP</i>	0.2637	0.2996	0.3419	0.3322
<i>B-LRP</i> $\alpha = 5$	0.3176	0.359	0.3763	0.3604
<i>B-LRP</i> $\alpha = 25$	0.2672	0.3153	0.351	0.3292
<i>B-LRP</i> $\alpha = 50$	0.2483	0.2874	0.3345	0.3201
<i>B-LRP</i> $\alpha = 75$	0.2499	0.2758	0.3298	0.3004
<i>B-LRP</i> $\alpha = 95$	0.2341	0.2598	0.3228	0.2921
<i>B-LRP</i> + $\alpha = 5$	0.3196	0.3566	0.3761	0.3684
<i>B-LRP</i> + $\alpha = 15$	0.2851	0.3279	0.361	0.3428
<i>B-LRP</i> + $\alpha = 25$	0.2733	0.3156	0.351	0.3307

TABLE 4.2: AUC scores for different explainability methods (rows) in Imagenet experiment for 4 different perturbation policies. Lower is better.

different clusters of strategies for this image. From the mean pixel-wise strategy for each cluster, we observe two strategies clusters that stand out: cluster 4 and cluster 5. Strategies found in cluster 4 all attribute negative relevances toward the lemon in the bottom-right part of the image, while strategies from cluster 5 share the common trait of attributing negative relevances near the biggest lemon.

Figure 4.13 illustrates another example of B-LRP and BSC explanations. We, again, observe the same situation where the lemon in the bottom-left part of the image is not recognized by some samples. With BSC we can identify the cluster of relevance maps (and the corresponding weights of the network) that consider this lemon to have a negative relevance regarding the true prediction of the class "lemon".

4.4 Confirming the Clever Hans Effect with B-LRP

In the following experiment, we revisit the work of Lapuschkin et al. [50, 51] on the *clever Hans* effect. A *clever Hans* strategy denotes a problematic solution strategy that provides the right answer for the wrong reason: the classic example being the one of the horse Hans, which was able to correctly provide answers to simple computation questions while actually not doing math but rather reading its master⁴. A modern machine-learning example is an artifact or a watermark in the data that happens to be present in one class, i.e., there is a random artifactual correlation that the model systematically and erroneously harvests [50, 51].

We conduct a similar experiment training⁵ a VGG16 DNN on the Pascal VOC 2007 challenge dataset experiment as in [50, 51]. The B-LRP explanations with respect to the class *horse* are illustrated in Figures 4.14 and 4.15. We indeed observe the fact that the watermark in the bottom left corner of the image occurs with a high relevance on both images in the 5-th percentile explanation. In other words, 95% of the samples of relevance maps consider this feature to highly contribute to the class "horse". Given this finding, we can now say for sure that the *clever Hans* is really clever, in the sense that the classification is based on the information from the

⁴https://en.wikipedia.org/wiki/Clever_Hans

⁵Code for training could be found at <https://github.com/lapalap/B-LRP>.

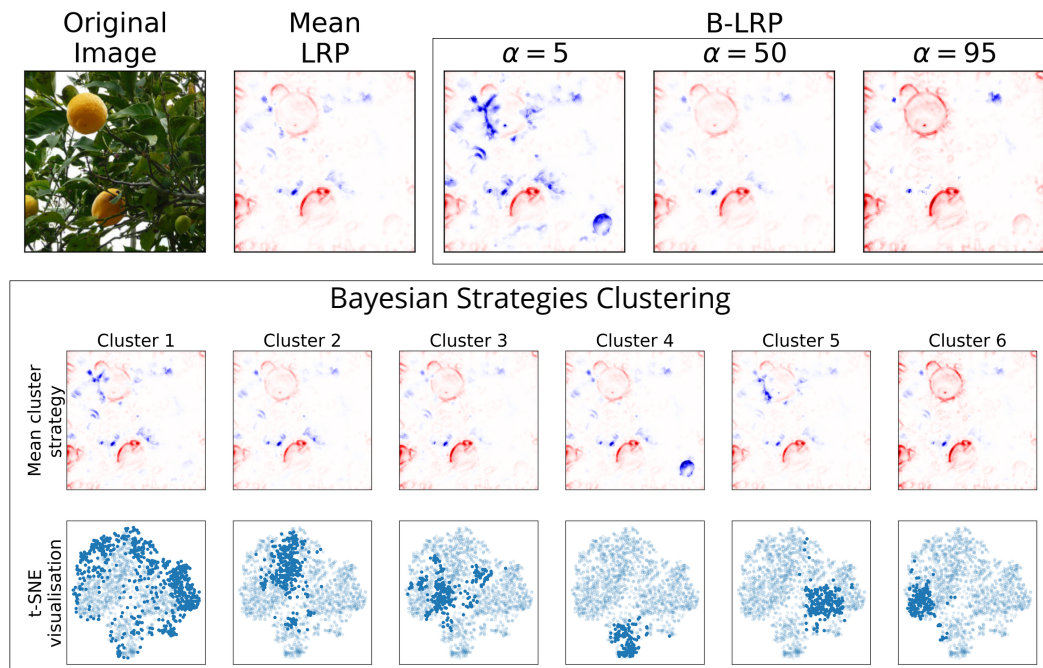


FIGURE 4.12: Illustration of B-LRP (TOP) and BSC (BOTTOM) explanations for class "lemon". Description of BSC explanations (BOTTOM): each column represents a cluster that was found with SpRAY. First row visualizes the pixel-wise average relevance map in each cluster, the second row highlight (blue) strategies inside of each cluster in the t-SNE two-dimensional visualization of the sampled relevance maps.

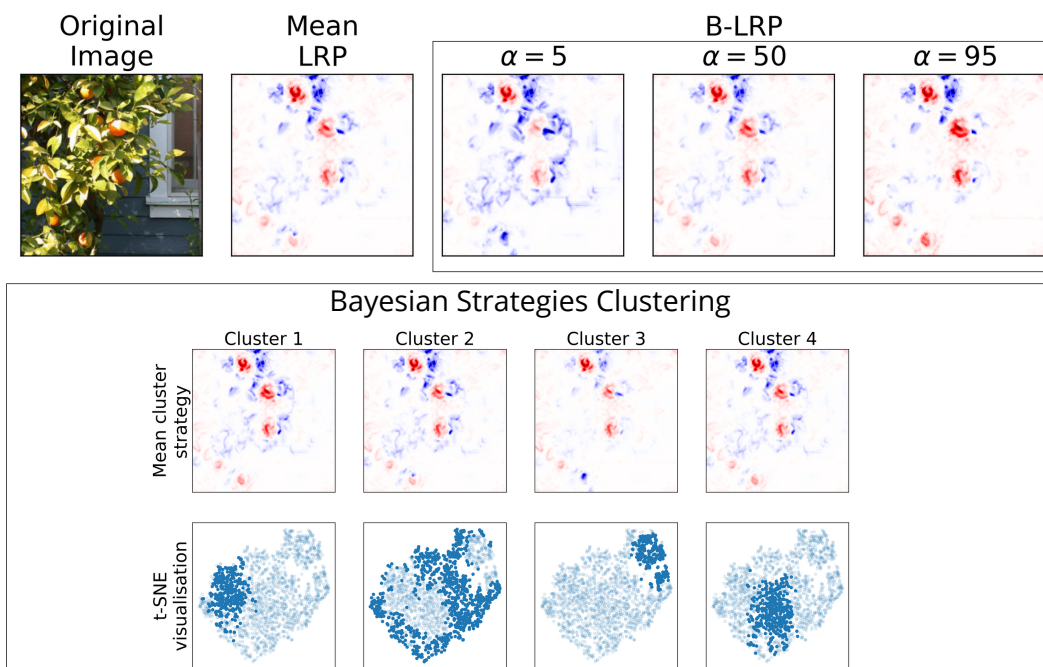


FIGURE 4.13: Illustration of B-LRP (TOP) and BSC (BOTTOM) explanations for class "lemon".

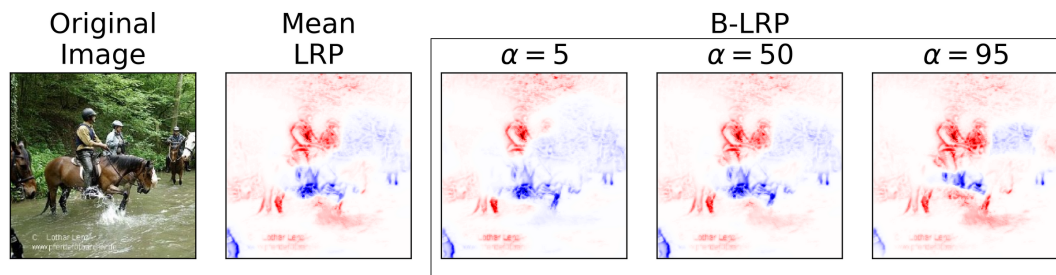


FIGURE 4.14: B-LRP explanations with respect to a "horse" class. B-LRP helps the user to distinguish between random artifacts on the explanations and systematic behaviour of a learning machine.

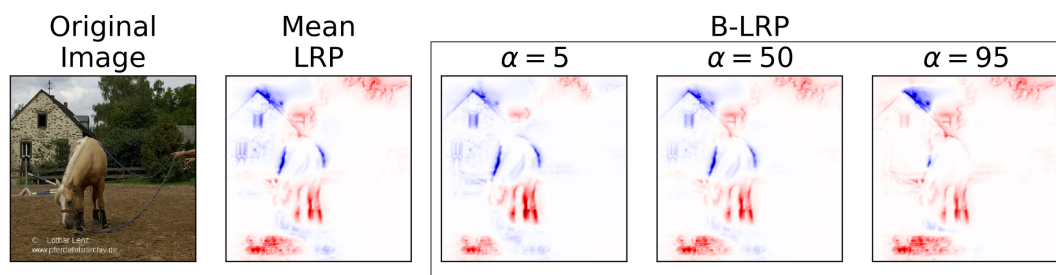


FIGURE 4.15: B-LRP explanations with respect to a "horse" class. An interesting observation is that for B-LRP $\alpha = 5$ learning machine is over-fitting not on the whole watermark, but only on the left part.

learned artifact. With B-LRP we can distinguish between systematic behavior and just an explainability artifact.

Hence, B-LRP enables us to confirm the clever Hans effect in our network, in the sense that the classifier draws information from artifacts that exist in the training (and also in validation and test) data set.

4.5 Noisy KFAC: Application of B-LRP to other explanation methods

This section is devoted to an investigation, how the proposed B-LRP method could be applied for other local explainability methods. For this experiment, we employ VGG-like CNN⁶ trained on CIFAR-10 dataset [47] by the Noisy KFAC variational inference method [118]⁷.

In this experiment, we visually assess the performance of local explainability methods, introduced in Chapter 2.2.2. For visualization, we employ a *rank map* — to compare different algorithms and their performance we will visualize only the top 20% percent of pixels for each explanation. We compare *baseline* methods — explainability methods that employed mean of the posterior of the parameter distribution, with pixel-wise percentiles, for percentile $\alpha = [75, 95]$. Percentiles for each method are obtained by sampling each explainability method for 500 times for each image.

⁶Exact architecture of the NN and parameters used for the training procedure could be found at <https://github.com/lapalap/B-LRP>.

⁷Code for training could be found at <https://github.com/team-approx-Bayes/dl-with-bayes>

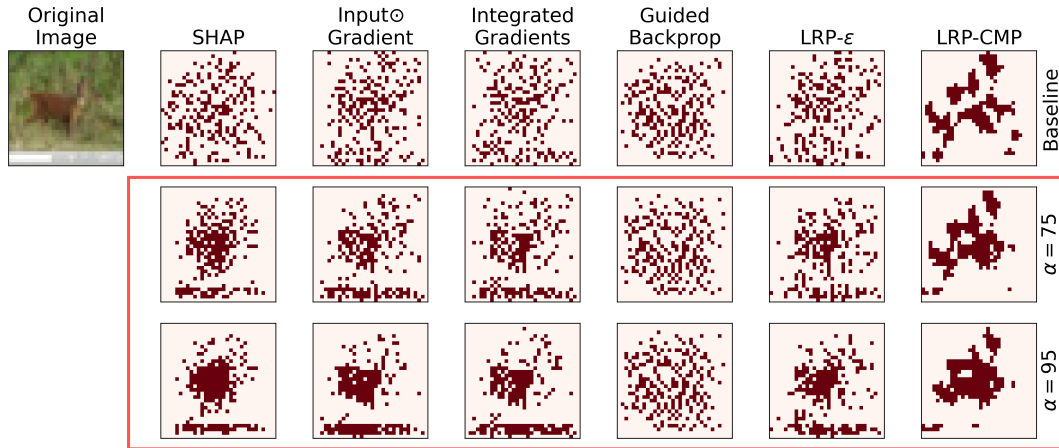


FIGURE 4.16: Application of B-LRP method for different interpretability methods for class "deer". First row corresponds to a *baseline* explanations, Second and Third rows (framed in red) correspond to a pixel-wise percentile with parameter α equal to $[75, 95]$, respectively. While top 20% of most important pixels in baseline explanations for each method distributed sparsely, for percentiles we see how most important pixels becoming dense around the deer.

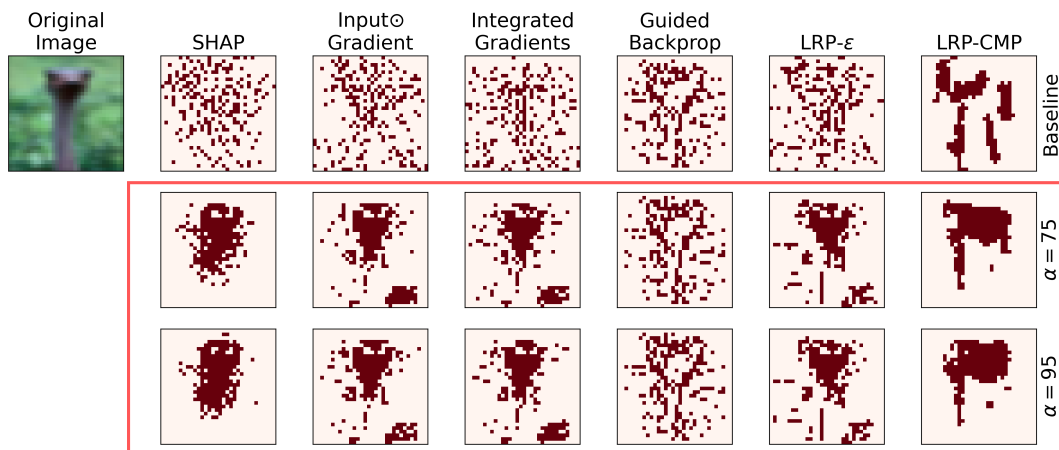


FIGURE 4.17: Application of B-LRP method for different interpretability methods for class "bird" (ostrich). Again, same as in Figure 4.16 we observe that higher percentiles for each method correspond to the fact that the top 20% of most important pixels are covering the bird head on the image, while the baseline methods distribute the most important pixels almost uniformly over the image.

Figures 4.16, 4.17 and 4.18 demonstrate the interesting behavior of how the usage of percentiles affects the most important pixels on the produced explanations. We observe, that for *baseline* methods — methods that explain the mean of the posterior distribution, 20% of the most important pixels are distributed rather uniformly across the image. On the other hand, after sampling the relevance maps from the posterior and performing a pixel-wise percentile operation, we observe that most important pixels cover the area of the original object. This again proves that percentile operations by the B-LRP methods apply to any local explainability method

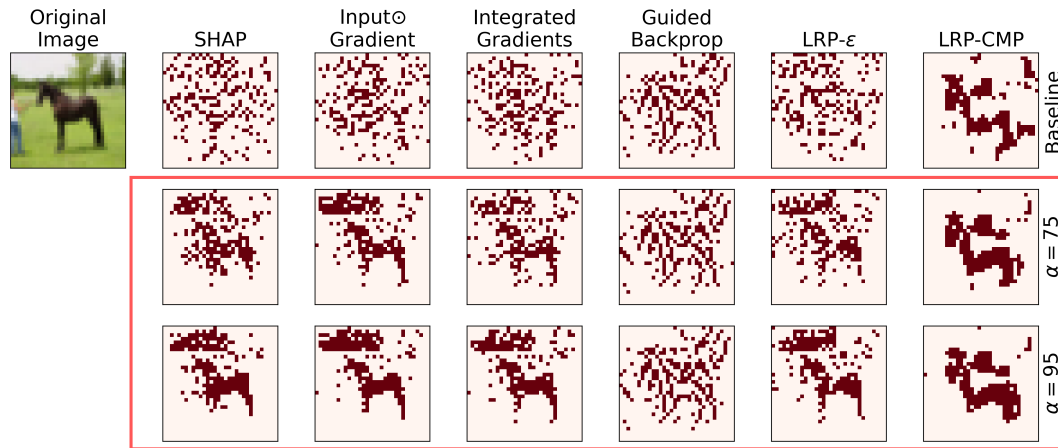


FIGURE 4.18: Application of B-LRP method for different interpretability methods for class "horse". Visually we observe the difference in most important pixels for each method before and after taking the high ($\alpha = [75, 95]$) percentile from pixel-wise relevance distribution

for producing better explanations for BNNs. More illustrations could be found in Appendix C.

Another noticeable fact is that for Guided Backprop (GB) method [97] most important pixels do not change by using a percentile operation. In other words, explanations that are computed by the GB algorithm for each sample of weights from the posterior do not differ from each other. Figure 4.19 illustrates (in a "seismic" colormap, Chapter 4.1.2) that the GB algorithm is not changing its explanations after applying the percentile operator. The main reason for this behavior is described in [2] — resulting explanations of GB method do not change after all the weights of a trained network are randomly sampled.

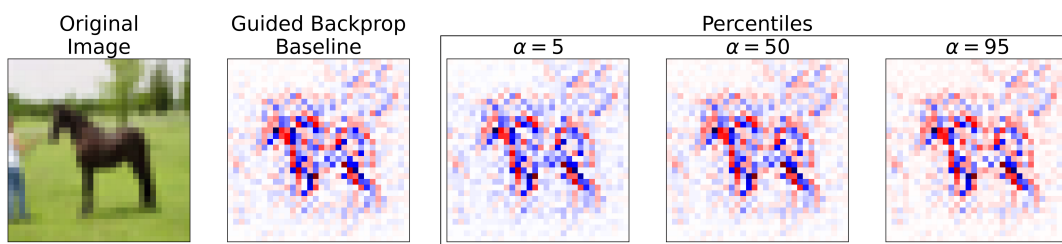


FIGURE 4.19: Demonstration of the invariant behaviour of Guided Backprop towards the percentile operation. From LEFT to RIGHT: Original image of a "horse", Baseline GB explanation, 5-th, 50-th and 95-th pixel-wise percentile of GB explanation distribution, respectively.

Chapter 5

Concluding discussion

In this final chapter, we will discuss the main advantages, as well as limitations of the proposed explanation methods, namely *Mean LRP*, *B-LRP*, and *BSC*. We will discuss the best-practice for explaining the decision-making process of BNNs, using these methods. In the end, we will analyze several ideas that naturally follow from this work that can be explored in the future.

5.1 Bayesian Explanation Pipeline

In our work, we proposed a novel framework for explaining and interpreting the decision-making process of Bayesian Neural Networks. Our approach includes 3 main steps, illustrated at Figure 5.1.

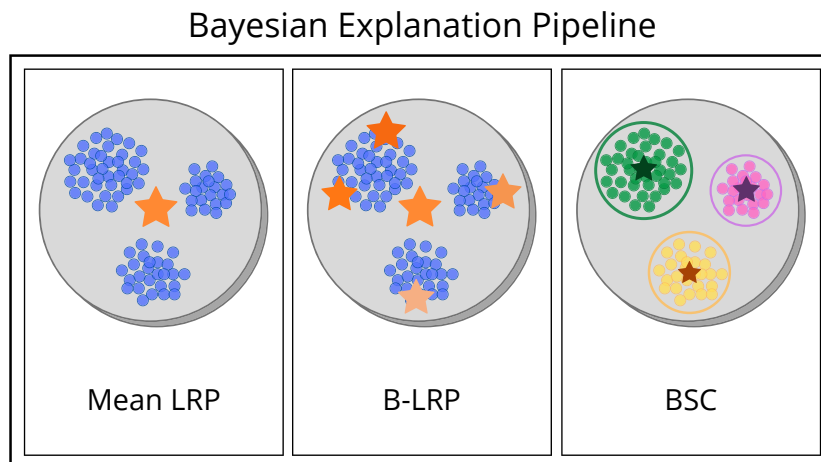


FIGURE 5.1: Schematic illustration of the BNNs Explanation pipeline: each figure illustrates the schematic t-SNE visualisation of relevance maps, sampled from the posterior distribution with stars illustrating the results of explainability procedure.

- **Mean LRP**

The Mean LRP is the baseline method for the explanation of the decision-making process of BNN and might be considered as a starting point for investigation in the model behavior. Being the simplest of all described explanation methods, it does not use any information about the posterior distribution of the parameters of a Network and just creates an explanation for a mean of the posterior.

While for some cases, for example for BNN with small variance over the weights, this type of explanation might be sufficient, unfortunately, it might not work for any BNN. For example, recently introduced Variance Networks [73] have a zero-mean posterior, thus Mean LRP won't work in this situation.

- **B-LRP**

Introduced in this paper B-LRP method is a new way of explaining the decision making process of BNNs. B-LRP allows to not only inspect the most relevant pixels for a decision but also their (un)certainties – a formidable starting point for obtaining novel insight into the behavior of Bayesian learning models.

In our work, we showed that with a high parameter α , B-LRP is explaining the behavior of the network better, in comparison with the Mean LRP. By choosing between different parameters, user can understand the rationale behind the network: with small parameters of α we can observe what features that are considered to be contributing towards the prediction are common for all sampled strategies, while with high parameter α we can understand all collection of features, that at least small fraction of strategies considers to have positive relevance. By choosing parameter α users can instantiate between more cautious or risky explanations, depending on the objective in mind.

Trivially, the computational complexity of B-LRP is linear to the number of posterior samples, and even 100 samples turned out to be sufficient for stably assessing the explanation uncertainty on a coarse grain in our experiments. However, more fine-grained percentiles (e.g., 1st percentile and lower) would require more samples to be drawn.

- **Bayesian Strategies Clustering**

BSC method produces an alternative explanation to the decision-making process of BNNs, in comparison with other proposed methods. It is based on the clusterization of saliency maps, that were sampled from the posterior distribution. By choosing preferable parameters it is possible to decompose the decision-making process into a small set of 'prime' strategies, with each strategy being assigned the score of importance — the number of relevance maps in each cluster. This method allows users to see what particular traits are shared between the strategies. On the other hand, the BSC method can be applied to identify anomalies in the BNNs strategies: for example, this could be used for investigation of the Clever Hans effect in the models or hidden bias in the decision-making process of BNN.

In comparison with B-LRP, this method is more computationally complex, mainly due to the clusterization procedure. While the suggested Spectral Clustering method is able to produce high-quality clusterings for the small-sized collections of relevance maps, its applications for large collections is limited by its computational complexity of $O(n^3)$ [115]. One interesting line of future research would be to use different clusterization algorithms as well as different dimensionality reduction techniques for pre-processing relevance maps before clustering.

5.2 Future work

Several ideas follow as a logical continuation of this research:

- **Practical application of BSC method**

We illustrated several examples of explanations done with the BSC method. While our experiments serve as proof for the usefulness of the described method, it would be interesting to see the performance of the BSC method in real-life applications. For instance, as BNN is vastly used in medical applications, it would be compelling to how the proposed method can determine "prime" strategies in such applications as MRI cancer detection or X-Ray pneumonia classification. In the same way, as was demonstrated in our work, BSC might help to find some differences between strategies or detect over-fitted / degenerate strategies.

- **Usage of different clustering algorithm in BSC**

Another possible research direction is to analyze different clustering methods for the relevance distribution. The SpRAY method is based on a Spectral Clustering, that suffers from such issues like computational complexity for big datasets. It might be interesting to research to compare different clustering algorithms and how they perform to identify 'prime' strategies.

Another fascinating line of research is to try to cluster relevance maps with modern DL clustering techniques [66, 43]: clustering would be performed not the relevance maps themselves but on a semantic representation thereof. This way strategies would be grouped not by some pixel-wise distance metric, but by a distance in a semantic sense: for example, relevance maps would be clustered by how they allocate relevance to specific objects in the image.

5.3 Conclusion

In this work we proposed a novel pipeline for explaining model behavior, that is based on three different methods: *Mean LRP*, *B-LRP*, and *BSC*. All of the proposed methods make up an informative explanation framework that allows users to understand the decision-making behavior of the BNN in detail. Moreover, this framework is not limited by Bayesian Networks nor by a particular explanation method: we demonstrate the application of this method in a case of a non-bayesian network as well as its application to other local explanation methods.

We believe that this work closes the gap in XAI related to the explanation of Bayesian Neural Networks. Moreover, the novel possibility to quantify uncertainty in explanations and to be able to set the appropriate risk level in an application will be helpful in practice. Gaining a better understanding of trained Neural Networks is beneficial to users aiming for safe, verifiable, and trustworthy AI.

Appendix A

Bayes by Backprop: MNIST

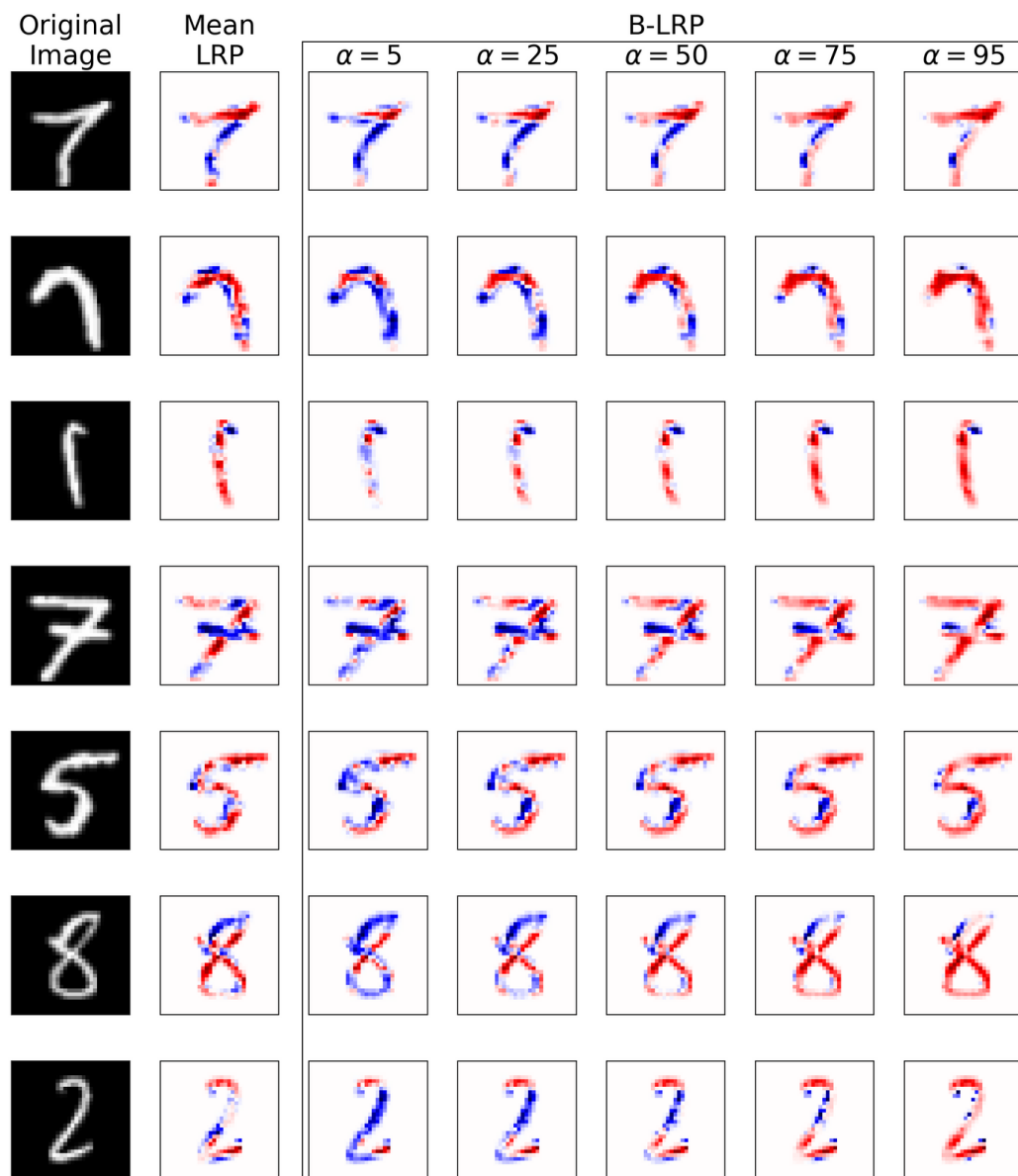


FIGURE A.1: Illustration of B-LRP explanations for LeNet, Bayes by Backprop, MNIST.

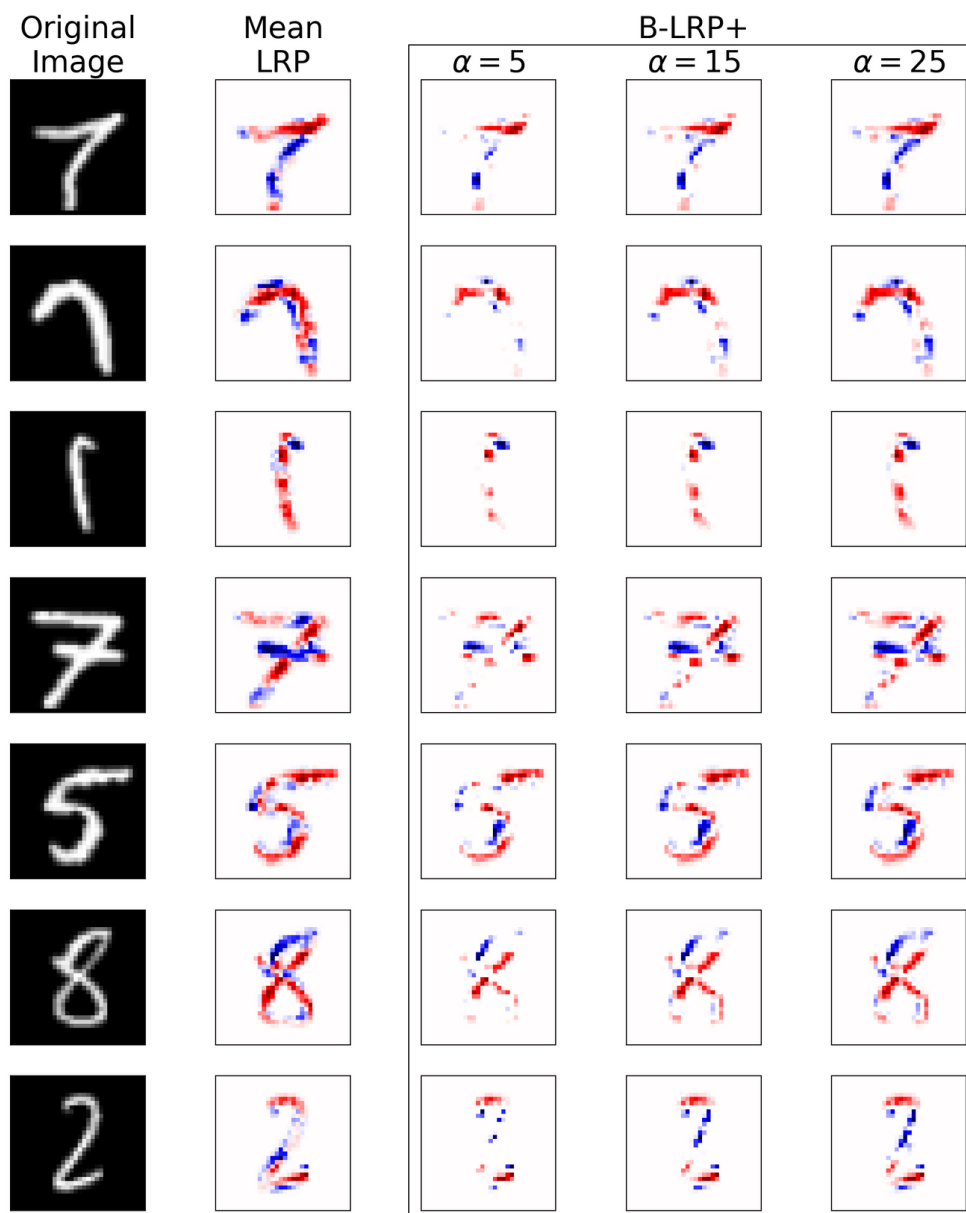


FIGURE A.2: Illustration of B-LRP+ explanations for LeNet, Bayes by Backprop, MNIST.

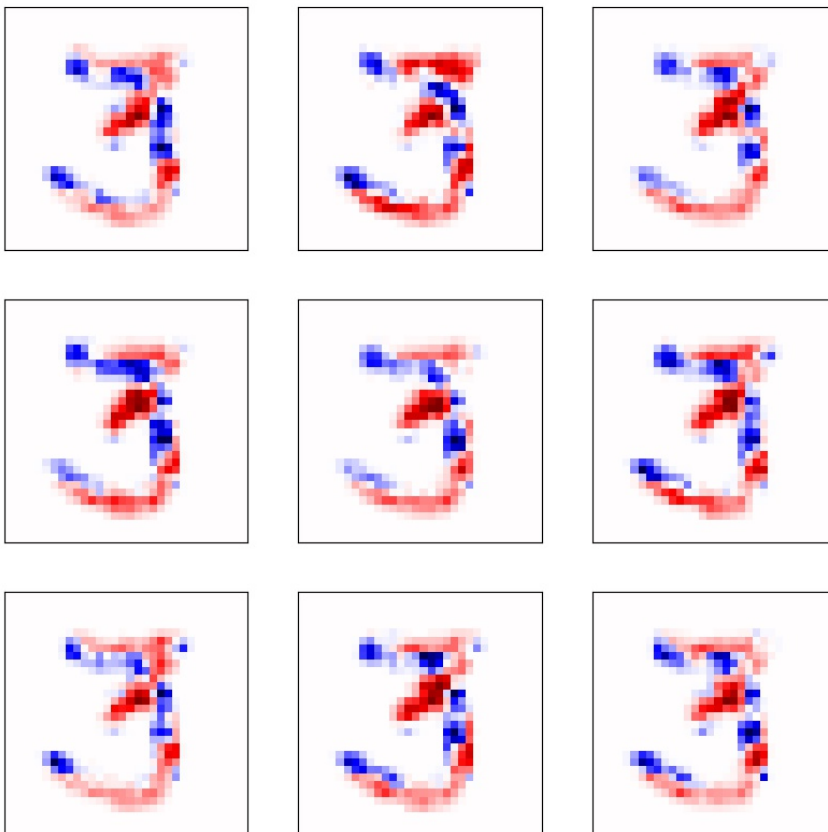


FIGURE A.3: Illustration of random samples of relevance maps from Cluster 1 in BSC example, MNIST.

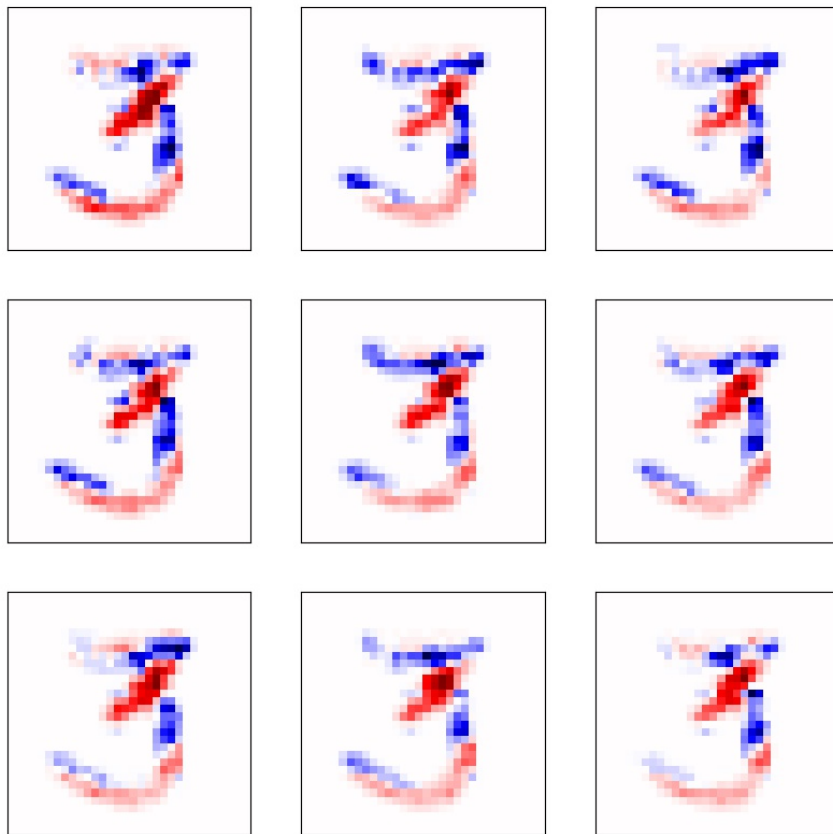


FIGURE A.4: Illustration of random samples of relevance maps from Cluster 2 in BSC example, MNIST.

Appendix B

MC Dropout: Imagenet



FIGURE B.1: Illustration of B-LRP explanations for VGG-16, Imagenet.

Appendix C

Noisy KFAC: Application of B-LRP to other explanation methods

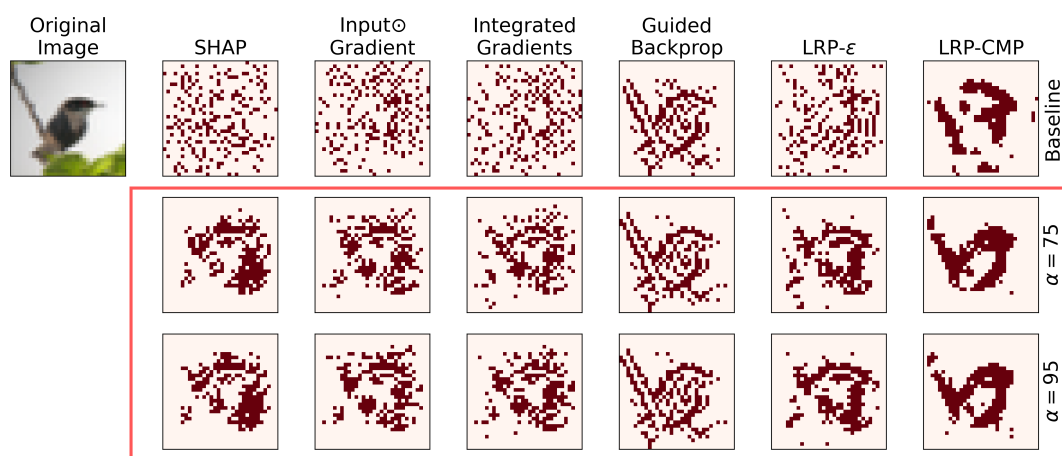


FIGURE C.1: Application of B-LRP for different explainability methods for class "bird".

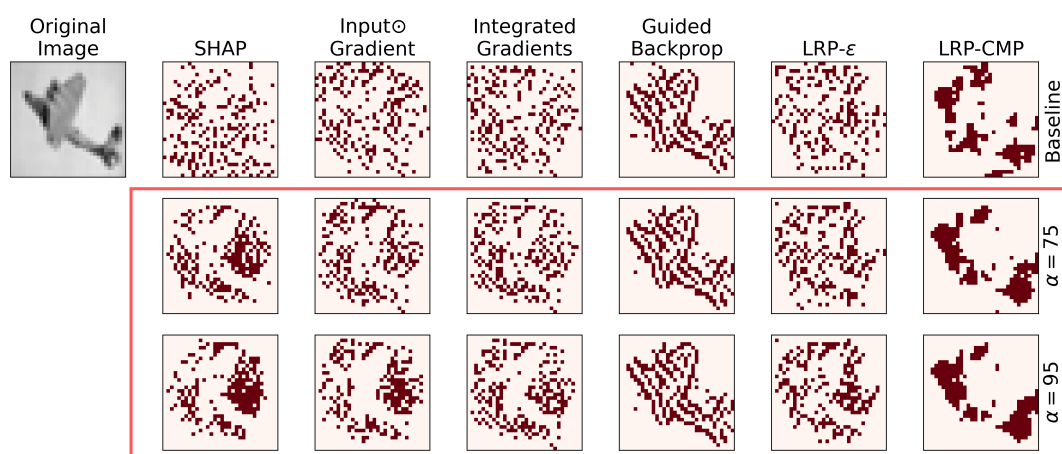


FIGURE C.2: Application of B-LRP for different explainability methods for class "plane".

Bibliography

- [1] Aashay Sachdeva. *Experimentation with Variational Dropout -Do Subnetworks exist inside a Neural Network?* [Online; accessed 29-July-2020]. 2020. URL: <https://medium.com/@aashay96/experimentation-with-variational-dropout-do-subnetworks-exist-inside-a-neural-network-e482cbbea7dd>.
- [2] Julius Adebayo et al. "Sanity checks for saliency maps". In: *Advances in Neural Information Processing Systems*. 2018, pp. 9505–9515.
- [3] Charu C. Aggarwal. "An Introduction to Neural Networks". In: *Neural Networks and Deep Learning: A Textbook*. Cham: Springer International Publishing, 2018, pp. 1–52. ISBN: 978-3-319-94463-0. DOI: [10.1007/978-3-319-94463-0_1](https://doi.org/10.1007/978-3-319-94463-0_1). URL: https://doi.org/10.1007/978-3-319-94463-0_1.
- [4] Shun-ichi Amari. "Neural learning in structured parameter spaces-natural Riemannian gradient". In: *Advances in neural information processing systems*. 1997, pp. 127–133.
- [5] *Amazon scraps secret AI recruiting tool that showed bias against women*. <https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G>.
- [6] Marco Ancona et al. "Towards better understanding of gradient-based attribution methods for deep neural networks". In: *arXiv preprint arXiv:1711.06104* (2017).
- [7] Christopher J Anders et al. "Understanding patch-based learning of video data by explaining predictions". In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019, pp. 297–309.
- [8] Andrew Ng: *Why AI Is the New Electricity*. <https://www.gsb.stanford.edu/insights/andrew-ng-why-ai-new-electricity>.
- [9] Leila Arras et al. "'What is relevant in a text document?': An interpretable machine learning approach". In: *PloS one* 12.8 (2017), e0181142.
- [10] Alejandro Barredo Arrieta et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI". In: *Information Fusion* 58 (2020), pp. 82–115.
- [11] Sebastian Bach et al. "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation". In: *PloS one* 10.7 (2015).
- [12] Charles Blundell et al. "Weight uncertainty in neural networks". In: *arXiv preprint arXiv:1505.05424* (2015).
- [13] Tom Charnock, Laurence Perreault-Levasseur, and François Lanusse. "Bayesian Neural Networks". In: *arXiv preprint arXiv:2006.01490* (2020).
- [14] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. "Multi-column deep neural networks for image classification". In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 3642–3649.

- [15] *Convolutional Neural Networks (LeNet)*. https://d2l.ai/chapter_convolutional-neural-networks/lenet.html. Accessed: 2020-07-23.
- [16] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.
- [17] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.
- [18] Adrien Ecoffet et al. "Go-explore: a new approach for hard-exploration problems". In: *arXiv preprint arXiv:1901.10995* (2019).
- [19] Sergey Edunov et al. "Understanding back-translation at scale". In: *arXiv preprint arXiv:1808.09381* (2018).
- [20] Bradley Efron et al. "Least angle regression". In: *The Annals of statistics* 32.2 (2004), pp. 407–499.
- [21] Dumitru Erhan et al. "Visualizing higher-layer features of a deep network". In: *University of Montreal* 1341.3 (2009), p. 1.
- [22] Kunihiko Fukushima and Sei Miyake. "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition". In: *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285.
- [23] Y. Gal and Z. Ghahramani. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: *Proceedings of ICML*. 2016.
- [24] Carl Friedrich Gauss. *Theoria motus corporum coelestium in sectionibus conicis solem ambientium*. Vol. 7. Perthes et Besser, 1809.
- [25] CF Gauss. "Theoria combinationis observationum erroribus minimis obnoxiae: pars prior.[Translated (1995) by GW Stewart as Theory of the Combination of Observations Least Subject to Error. SIAM, Philadelphia.]" In: *Gill, P., Murray, W., Saunders, M., Tomlin, T. and Wright* (1821).
- [26] Ethan Goan and Clinton Fookes. "Bayesian Neural Networks: An Introduction and Survey". In: *Case Studies in Applied Bayesian Data Science*. Springer, 2020, pp. 45–87.
- [27] Ben Goertzel and Cassio Pennachin. *Artificial general intelligence*. Vol. 2. Springer, 2007.
- [28] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- [29] A. Graves. "Practical variational inference for neural networks". In: *Advances in NIPS*. 2011.
- [30] Miriam Hägele et al. "Resolving challenges in deep learning-based analyses of histopathological images using explanation methods". In: *Scientific reports* 10.1 (2020), pp. 1–12.
- [31] Kaiming He et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.
- [32] Donald Olding Hebb. *The organization of behavior: a neuropsychological theory*. J. Wiley; Chapman & Hall, 1949.
- [33] Geoffrey E Hinton and Drew Van Camp. "Keeping the neural networks simple by minimizing the description length of the weights". In: *Proceedings of the sixth annual conference on Computational learning theory*. 1993, pp. 5–13.

- [34] Geoffrey E Hinton et al. "Improving neural networks by preventing co-adaptation of feature detectors". In: *arXiv preprint arXiv:1207.0580* (2012).
- [35] Andreas Holzinger et al. "Causability and explainability of artificial intelligence in medicine". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 9.4 (2019), e1312.
- [36] Rein Houthoofd et al. "Vime: Variational information maximizing exploration". In: *Advances in Neural Information Processing Systems*. 2016, pp. 1109–1117.
- [37] *How We Analyzed the COMPAS Recidivism Algorithm*. <https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm>.
- [38] David H Hubel and Torsten N Wiesel. "Receptive fields of single neurones in the cat's striate cortex". In: *The Journal of physiology* 148.3 (1959), p. 574.
- [39] Laurent Valentin Jospin et al. "Hands-on Bayesian Neural Networks—a Tutorial for Deep Learning Users". In: *arXiv preprint arXiv:2007.06823* (2020).
- [40] Pieter-Jan Kindermans et al. "Investigating the influence of noise and distractors on the interpretation of neural networks". In: *arXiv preprint arXiv:1611.07270* (2016).
- [41] D. P. Kingma, T. Salimans, and M. Welling. "Variational Dropout and the Local Reparameterization Trick". In: *Advances in NIPS*. 2015.
- [42] Durk P Kingma, Tim Salimans, and Max Welling. "Variational dropout and the local reparameterization trick". In: *Advances in neural information processing systems*. 2015, pp. 2575–2583.
- [43] B Ravi Kiran, Dilip Mathew Thomas, and Ranjith Parakkal. "An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos". In: *Journal of Imaging* 4.2 (2018), p. 36.
- [44] Frederick Klauschen et al. "Scoring of tumor-infiltrating lymphocytes: From visual estimation to machine learning". In: *Seminars in cancer biology*. Vol. 52. Elsevier. 2018, pp. 151–157.
- [45] Maximilian Kohlbrenner et al. "Towards best practice in explaining neural network decisions with LRP". In: *arXiv preprint arXiv:1910.09840* (2019).
- [46] Jeamin Koo et al. "Why did my car just do that? Explaining semi-autonomous driving actions to improve driver understanding, trust, and performance". In: *International Journal on Interactive Design and Manufacturing (IJIDeM)* 9.4 (2015), pp. 269–275.
- [47] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).
- [48] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.
- [49] Solomon Kullback and Richard A Leibler. "On information and sufficiency". In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.
- [50] Sebastian Lapuschkin et al. "Analyzing classifiers: Fisher vectors and deep neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2912–2920.
- [51] Sebastian Lapuschkin et al. "Unmasking clever hans predictors and assessing what machines really learn". In: *Nature communications* 10 (2019), p. 1096.

- [52] Yann LeCun et al. "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4 (1989), pp. 541–551.
- [53] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.
- [54] Adrien Marie Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes*. F. Didot, 1805.
- [55] Andy Liaw, Matthew Wiener, et al. "Classification and regression by randomForest". In: *R news* 2.3 (2002), pp. 18–22.
- [56] Seppo Linnainmaa. "The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors". In: *Master's Thesis (in Finnish), Univ. Helsinki* (1970), pp. 6–7.
- [57] Zachary Lipton et al. "Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.
- [58] Xiaodong Liu et al. "Improving multi-task deep neural networks via knowledge distillation for natural language understanding". In: *arXiv preprint arXiv:1904.09482* (2019).
- [59] Scott M Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Advances in neural information processing systems*. 2017, pp. 4765–4774.
- [60] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE". In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.
- [61] David JC MacKay. "A practical Bayesian framework for backpropagation networks". In: *Neural computation* 4.3 (1992), pp. 448–472.
- [62] Mario Klingemann. *GPT-3 Quote*. [Online; accessed 29-July-2020]. 2020. URL: <https://twitter.com/quasimondo/status/1286749705695887360?s=21>.
- [63] James Martens and Roger Grosse. "Optimizing neural networks with kronecker-factored approximate curvature". In: *International conference on machine learning*. 2015, pp. 2408–2417.
- [64] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.
- [65] Bernhard Mehlig. "Artificial neural networks". In: *arXiv preprint arXiv:1901.05639* (2019).
- [66] Erxue Min et al. "A survey of clustering with deep learning: From the perspective of network architecture". In: *IEEE Access* 6 (2018), pp. 39501–39514.
- [67] Riccardo Miotto et al. "Deep learning for healthcare: review, opportunities and challenges". In: *Briefings in bioinformatics* 19.6 (2018), pp. 1236–1246.
- [68] D. Molchanov, A. Ashukha, and D. Vetrov. "Variational Dropout Sparsifies Deep Neural Networks". In: *Proceedings of ICML*. 2017.
- [69] Christoph Molnar. *Interpretable Machine Learning*. Lulu. com, 2020.
- [70] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. "Methods for interpreting and understanding deep neural networks". In: *Digital Signal Processing* 73 (2018), pp. 1–15.

- [71] Grégoire Montavon et al. "Layer-wise relevance propagation: an overview". In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019, pp. 193–209.
- [72] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. "Deepdream-a code example for visualizing neural networks". In: *Google Research 2.5* (2015).
- [73] Kirill Neklyudov et al. "Variance networks: When expectation does not meet your expectations". In: *arXiv preprint arXiv:1803.03764* (2018).
- [74] Anh Nguyen, Jason Yosinski, and Jeff Clune. "Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks". In: *arXiv preprint arXiv:1602.03616* (2016).
- [75] Anh Nguyen et al. "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks". In: *Advances in neural information processing systems*. 2016, pp. 3387–3395.
- [76] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. *Feature visualization: How neural networks build up their understanding of images*. Distill. 2018.
- [77] K. Osawa et al. "Practical Deep Learning with Bayesian Principles". In: *Advances in NeurIPS*. 2019.
- [78] Nicholas G Polson, Vadim Sokolov, et al. "Deep learning: A Bayesian perspective". In: *Bayesian Analysis* 12.4 (2017), pp. 1275–1304.
- [79] Colin Raffel et al. "Exploring the limits of transfer learning with a unified text-to-text transformer". In: *arXiv preprint arXiv:1910.10683* (2019).
- [80] Marco Tulio Ribeiro, Sameer Singh, and Carlos 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.
- [81] Hippolyt Ritter, Aleksandar Botev, and David Barber. "A scalable laplace approximation for neural networks". In: *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings*. Vol. 6. International Conference on Representation Learning. 2018.
- [82] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.
- [83] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.
- [84] S Rasoul Safavian and David Landgrebe. "A survey of decision tree classifier methodology". In: *IEEE transactions on systems, man, and cybernetics* 21.3 (1991), pp. 660–674.
- [85] Wojciech Samek et al. "Evaluating the visualization of what a deep neural network has learned". In: *IEEE transactions on neural networks and learning systems* 28.11 (2016), pp. 2660–2673.
- [86] Wojciech Samek et al. "Toward Interpretable Machine Learning: Transparent Deep Neural Networks and Beyond". In: *arXiv preprint arXiv:2003.07631* (2020).
- [87] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural networks* 61 (2015), pp. 85–117.

- [88] Ramprasaath R Selvaraju et al. "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.
- [89] LG Shapiro and GC Stockman. *Computer Vision, March 2000*. 2000.
- [90] Lloyd S Shapley. "Notes on the n-Person Game—II: The Value of an n-Person Game". In: (1951).
- [91] Mohammad Shoeybi et al. "Megatron-Lm: Training multi-billion parameter language models using gpu model parallelism". In: *arXiv preprint arXiv:1909.08053* (2019).
- [92] Kumar Shridhar, Felix Laumann, and Marcus Liwicki. "A comprehensive guide to bayesian convolutional neural network with variational inference". In: *arXiv preprint arXiv:1901.02731* (2019).
- [93] Avanti Shrikumar et al. "Not just a black box: Learning important features through propagating activation differences". In: *arXiv preprint arXiv:1605.01713* (2016).
- [94] Svetlana Sicular and Kenneth Brant. "Hype cycle for artificial intelligence, 2018". In: *Gartner (July 24, 2018)*. <<https://www.gartner.com/doc/3883863/hype-cycle-artificial-intelligence> (2018).
- [95] David Silver et al. "Mastering chess and shogi by self-play with a general reinforcement learning algorithm". In: *arXiv preprint arXiv:1712.01815* (2017).
- [96] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).
- [97] Jost Tobias Springenberg et al. "Striving for simplicity: The all convolutional net". In: *arXiv preprint arXiv:1412.6806* (2014).
- [98] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.
- [99] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks". In: *arXiv preprint arXiv:1703.01365* (2017).
- [100] Christian Szegedy et al. "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.
- [101] Naftali Tishby, Esther Levin, and Sara A Solla. "Consistent inference of probabilities in layered networks: Predictions and generalization". In: *International Joint Conference on Neural Networks*. Vol. 2. 1989, pp. 403–409.
- [102] Hugo Touvron et al. "Fixing the train-test resolution discrepancy: FixEfficientNet". In: *arXiv preprint arXiv:2003.08237* (2020).
- [103] Marina M-C Vidovic et al. "Feature importance measure for non-linear learning algorithms". In: *arXiv preprint arXiv:1611.07567* (2016).
- [104] Oriol Vinyals et al. "Starcraft ii: A new challenge for reinforcement learning". In: *arXiv preprint arXiv:1708.04782* (2017).
- [105] Ulrike Von Luxburg. "A tutorial on spectral clustering". In: *Statistics and computing* 17.4 (2007), pp. 395–416.
- [106] Hao Wang and Dit-Yan Yeung. "Towards bayesian deep learning: A survey". In: *arXiv preprint arXiv:1604.01662* (2016).

- [107] John J Weng, Narendra Ahuja, and Thomas S Huang. "Learning recognition and segmentation of 3-D objects from 2-D images". In: *1993 (4th) International Conference on Computer Vision*. IEEE. 1993, pp. 121–128.
- [108] F. Wenzel et al. "How Good is the Bayes Posterior in Deep Neural Networks Really?" In: *arXiv:2002.02405* (2020).
- [109] Gesa Wiegand et al. "I Drive-You Trust: Explaining Driving Behavior Of Autonomous Cars". In: *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pp. 1–6.
- [110] Wikipedia contributors. *Activation functions* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 23-July-2020]. 2020. URL: https://en.wikipedia.org/wiki/Activation_function.
- [111] Wikipedia contributors. *Artificial neuron* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 29-July-2020]. 2020. URL: https://en.wikipedia.org/wiki/Artificial_neuron.
- [112] Wikipedia contributors. *McCulloch-Pitts artificial neuron* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 29-July-2020]. 2020. URL: https://en.wikipedia.org/wiki/File:Artificial_Neuron.svg.
- [113] Andrew Gordon Wilson. "The case for Bayesian deep learning". In: *arXiv preprint arXiv:2001.10995* (2020).
- [114] Qizhe Xie et al. "Self-training with noisy student improves imagenet classification". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10687–10698.
- [115] Donghui Yan, Ling Huang, and Michael I Jordan. "Fast approximate spectral clustering". In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2009, pp. 907–916.
- [116] Xitong Yang. *Understanding the variational lower bound*. 2017.
- [117] Zhilin Yang et al. "Xlnet: Generalized autoregressive pretraining for language understanding". In: *Advances in neural information processing systems*. 2019, pp. 5753–5763.
- [118] Guodong Zhang et al. "Noisy natural gradient as variational inference". In: *International Conference on Machine Learning*. 2018, pp. 5852–5861.