Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Master's Thesis

Uncertainty in Explanation methods for Neural Networks

Mihir Mulye

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fullfilment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

Prof. Dr. Paul G. Plöger
Prof. Dr. Matias Valdenegro Toro

July 2022

I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work.

Date	Mihir Mulye

Abstract

With the increase in the use of neural networks in safety critical applications such as autonomous driving, medical diagnosis, and healthcare, the number of methods developed to explain the network output has also increased. These explanation methods are now being coupled with human expertise to understand the workings, to debug and optimize the performance and sometimes even extend the use of neural networks. As explanation methods gradually become indispensable, it is critical to analyse the uncertainty in the output of these explanation methods themselves. This thesis attempts to address this issue.

A review of the existing research shows that the most intuitive approach to solve this task is to combine the uncertainty estimation and the explanation methods to generate an explanation distribution. However, the majority of such works generally either focus on a single combination of uncertainty estimation and explanation method or propose a solution applicable to a specific task, say classification or regression. This thesis proposes and implements a pipeline which can test multiple combinations of uncertainty estimation and explanation methods and is able to generate explanation distribution for two different types of tasks.

From the literature review, four uncertainty estimation methods: Deep Ensemble, Monte Carlo (MC) Dropout, MC DropConnect, and Flipout are selected for combining with two explanation methods: Integrated Gradients (IG) and Guided Backpropagation (GBP). Eight combinations are identified and used to generate explanation distributions on two image classification datasets: CIFAR-10 and FER+. To visualize the uncertainty in these distributions, a modified *coefficient of variation* is proposed and computed. It is observed that lower values of coefficient of variation indicate higher certainty in the importance of identified feature and vice versa. As a part of this thesis, existing sanity checks have been modified and the explanation methods have been subjected to the same. For evaluation, the pixel deletion/insertion metrics are used and it is found that combinations having GBP as an explanation method generally perform better as compared to those having IG. The best performing deletion metrics are 0.123 (GBP with MC Dropout) and 0.139 (IG with MC Dropout). Similarly, the best performing insertion metrics are 0.931 (GBP with Flipout) and 0.929 (IG with Flipout). The proposed pipeline is also tested on California Housing dataset by using Local Interpretable Model-agnostic Explanation (LIME) rather than IG in the combinations used in the earlier task. The robustness of the pipeline is verified as it is able to generate explanation distributions for a regression task successfully as well. To the best of our knowledge, this is a novel use of GBP on a tabular dataset. As compared to other uncertainty estimation methods, Flipout generally tend to yield low variance which translates to high certainty of the identified features. Also, it is observed that the combinations having LIME as an explanation method are computationally expensive as compared to the ones using GBP.

This thesis is a step towards systematically investigating the possible combinations of uncertainty estimation and explanation methods to ascertain the uncertainty in the explanation of neural networks.

Acknowledgements

I would like to take this opportunity to appreciate the efforts of people who have supported, guided and helped me in this undertaking. I express my sincere gratitude to my supervisors Prof. Dr. Paul G. Plöger and Prof. Dr. Matias Valdenegro Toro for their valuable guidance and timely inputs. I received continued encouragement and support from them throughout the duration of the thesis project. I would like to thank my mother and my brother for providing me with this opportunity to pursue Masters and keeping me motivated throughout this period. I would also like to thank Ashay Mulye and Archana Alva for reviewing the drafts of the report numerous times and providing me with constructive feedback. This research work was possible in part, because of numerous stimulating discussions found on the GitHub and other open source platforms which resulted in the betterment of my understanding as well. This thesis has truly been an opportunity to learn and grow both professionally and personally. Finally, this project would not have been possible without the help and support from a lot of people to whom I am thankful.

Contents

List of Abbreviations

AUC	Area under Curve
BLRP	Bayesian Layer-wise Relevant Propagation
BNN	Bayesian Neural Network
CAM	Class Activation Map
CIFAR	Canadian Institute for Advanced Research
CNN	Convolutional Neural Network
CV	Coefficient of Variation
ELBO	Evidence Lower Bound
EX	Explanation method
FER	Facial Expression Recognition
GAP	Global Average Pooling
GBP	Guided BackPropagation
GDPR	General Data Protection Regulation
GPU	Graphics Processing Unit
IG	Integrated Gradients
KL	Kullback-Leibler divergence
LIME	Local Interpretable Model-agnostic Explanation
LRP	Layer-wise Relevance Propagation
MAE	Mean Absolute Error
MC	Monte Carlo

MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
NLL	Negative Log Likelihood
PASCAL-VOC	Pattern Analysis Statistical Modelling and Computational Learning-Visual Object Classes
ReLU	Rectified Linear Unit
RGB	Red Green Blue
RISE	Randomized Input Sampling for Explanation
RMSProp	Root Mean Squared Propagation
SGD	Stochastic Gradient Descent
SHAP	SHapley Additive ExPlanations
SSIM	Structural Similarity Index Measure
TPU	Tensor Processing Unit
UQ	Uncertainty Quantification/Estimation
VGG	Visual Geometry Group
XAI	Explainable Artificial Intelligence

List of Figures

List of Tables

1

Introduction

Learning machines such as Convolutional Neural Networks (CNN) have achieved remarkable improvement in performances in recent years. This has been majorly attributed to growth in available data and increased availability of computational resources such as Graphics Processing Units (GPU) and Tensor Processing Units (TPU) needed for training these networks. The improvement in the performances of these networks can be gauged by the fact that in some areas of application, such neural networks are performing better than human performance threshold [19]. This has invariably led to the application of these neural networks to variety of fields such as autonomous driving [28], [94], [98], medical diagnosis [30], [31], [36], [92], healthcare [10], video processing [8], time series forecasting [79] and so on.

Many of the applications where CNNs are now being used, have a low error tolerance threshold such as the example of autonomous driving and medical diagnosis. With their use in safety critical settings, not only is it essential to understand "why" a network predicts a particular output, but also a measure of the uncertainties associated with the explanation of the network output becomes important [18]. The objective of finding the explanation for the network output is to identify and understand the underlying prediction mechanism of the network. An important advantage of doing so could be the ability to troubleshoot and if possible, improve the network performance and reliability. In the last few years, this field of study has gained attention of the scientific community and has subsequently given rise to the field of Explainable Artificial Intelligence (XAI) [12], [29], [57], [59], [61]. Within this field, researchers have proposed methods to explain the predictions of a neural network by utilising the network architectural information [50], [60], [96], [101], by calculating the gradients [75], [97], and propagation of the aforementioned gradients according to certain predefined rules [13], [37], [51]. Several other explanation methods build upon the concepts discussed in the works listed above and have been discussed in [21], [44], [55], [62], [69], [73], [89].

In contrast, fewer studies have been conducted to ascertain the uncertainty associated with the explanation of a neural network. As the application possibilities and our reliance on neural networks increases, not only does it become imperative to ensure that the network predictions are correct, but also that there is a logical explanation to the network output and that logic itself is reliable. This field essentially focuses on analysing the quality of explanations generated from the XAI approaches. In this thesis, we focus on exploring this area of research, the available approaches, identifying their limitations and investigating the possibilities of computing uncertainty in explanation output for a given neural network. The subsequent sections discuss the motivation, outline the problem statement and list the

relevant research questions that are addressed by this thesis.

1.1 Motivation

It is important to analyse the uncertainty associated with the explanation methods that are applied to neural networks. This claim is supported by the following discussions that highlight the importance of explanation methods and the subsequent reasons as to why the uncertainty associated with the explanation of a neural network needs to be taken into consideration:

- **Ensuring Safety:** Real world application with critical safety requirements such as autonomous driving and medical imaging require correct and robust predictions along with explanations and insights describing the prediction mechanism of the network [18], [57]. This claim is supported in the works of Bykov et al. [18] who also state that failing to consider the possible uncertainty in the explanation of the network output could result in an unjustified trust in such explanations. As the importance of explanations as a measure to establish credibility of a network output in critical scenario increases, it is essential to ensure the correctness and reliability of these explanations themselves [72]. Explanations that contain significant uncertainty could otherwise be detrimental to the network's effectiveness [99] and might result in catastrophic failures.
- **Identifying Clever Hans systems:** Explanations could be instrumental in identifying Clever Hans[1] predictors. These are predictors that provide the correct output for "incorrect" reasons. An example of Clever Hans predictor is discussed in Lapuschkin et al. [41]. This research applies explanation methods to the winning approach/method of the PASCAL-VOC [22] and claims that the winning method exhibits anomalous behaviour, in that it yields correct classification output for "incorrect" reasons. The approach being investigated detects trains by the presence of rail tracks, boats by the presence of water and horses by the presence of watermarks on the image. From this discussion, it is evident that explanation methods could potentially be used as an analysis tool to identify the shortcomings of a neural network. Hence, it is critical to ensure the quality of the output of these explanation methods themselves [57].
- **Drawing Novel Insights:** Samek et al. [57] highlight the importance of explanation of the output of a neural network by discussing the case of AlphaGo [66]. In a game between AlphaGo and a human player, AlphaGo made a move that was classified by a Go expert as "not a human move" in a post game analysis. The explanation to the move was not available but it proved pivotal to the outcome of the match. Using this as an example, [57] argue that with the explanation methods, it would be possible to explore novel strategies and uncover patterns while solving a task, that might go unnoticed by humans. This could be valuable in applications like drug development, healthcare, material sciences and therefore, ascertaining the uncertainty associated with the explanation methods could prove useful in determining the applicability and usefulness of that particular insight.
- **Usability:** It is important to evaluate the quality of explanation in situations where they influence the processing strategy in a learning system [58]. This can be better understood with the help of an

[1] `https://en.wikipedia.org/wiki/Clever_Hans`

example. For instance, the decision to execute resource intensive algorithms on relevant regions of input image highlighted by heatmap (essentially the explanation method output) can be taken after analysing the quality of the heatmap in question. The critical decision to execute the computation intensive process only on the regions highlighted by the heatmap should be made only if the heatmap itself satisfies the quality requirements. Hence, it is necessary to analyse the quality of the output of the explanation method.

- **Impact on Legislation:** The European General Data Protection Regulation (GDPR)[2] has a "right to explanation" that emphasizes the importance of transparency in the decision making process of learning machines [12], [57]. This enables the user to obtain an explanation of the decision made by an algorithm/learning machine. Though the exact details of its implementation are yet to be codified, it can be concluded that the explanation methods will play a critical role in legal compliance in the future. This in turn underlines the need of high quality explanations and that can be ensured by computing the associated uncertainties.

1.2 Problem Statement

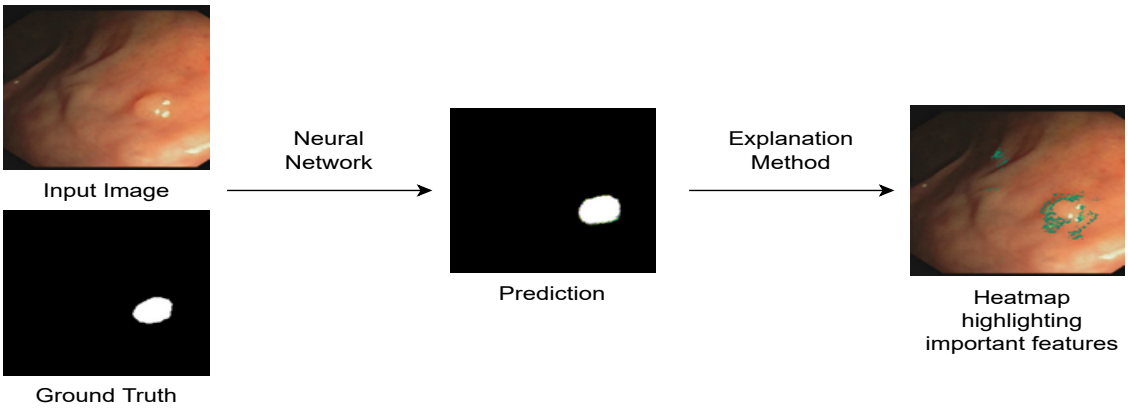
The question of "why" a network produces a specific output has been exhaustively researched in numerous scientific works [18]. However, not many research papers have addressed the question "how much" can these explanations be trusted [18], [19], [100]. This question concerns itself with the quality of explanation of the neural network. Naturally, the lower the uncertainty associated with a particular explanation of the network, higher the confidence in the explanation and hence better the quality of the explanation. As has already been discussed in Section 1.1, the uncertainty related to an explanation for a given neural network plays a vital role for several applications and hence this direction of research must be systematically investigated.

The state of the art approaches attempt to solve the problem by combining uncertainty methods developed for neural networks with explanation methods. A noteworthy approach for analysing the uncertainty in explanation of a neural network has been shown in Figure 1.1b. This research work [93] proposes an approach by combining Monte Carlo (MC) Dropout (an uncertainty estimation method) with GBP (an explanation method) to ascertain the uncertainty in the importance of input features. This approach highlights the features which the model considers important while making a particular decision (in this case, segmentation) along with whether the model is certain regarding the importance of said features.

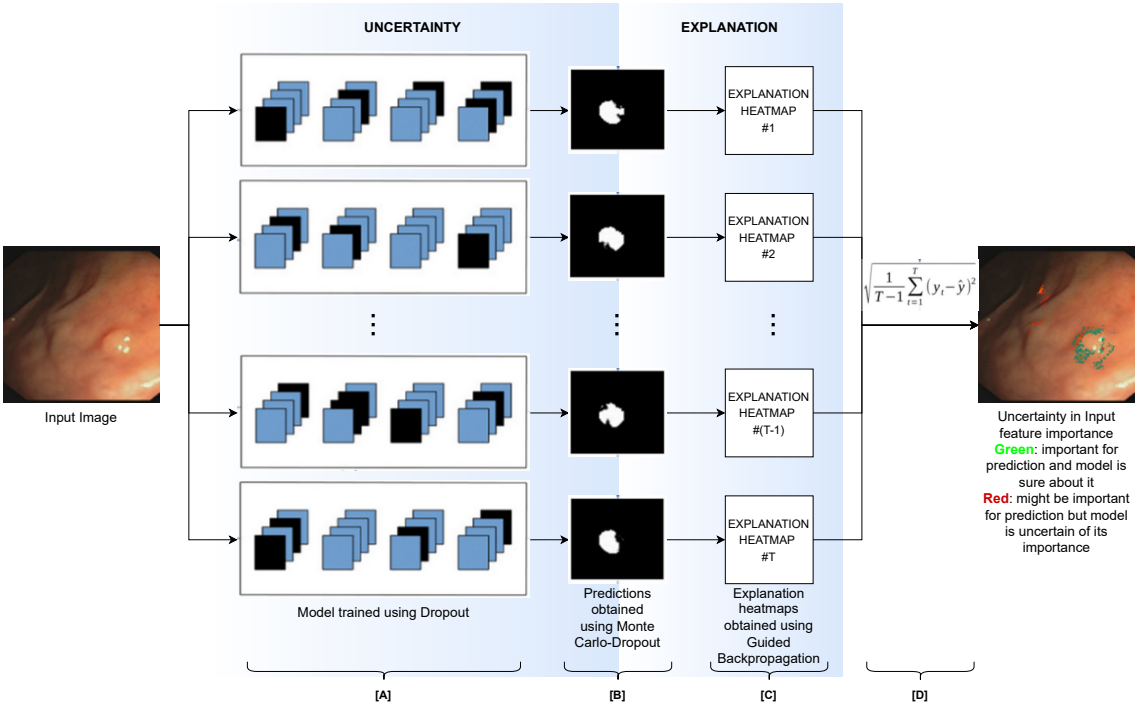
A shortcoming of this approach however is that it only explores one combination of uncertainty estimation method and explanation method. Additional researches conducted in [18] and [19] utilise the Layer-wise Relevance Propagation (LRP) [13] explanation method in order to ascertain the uncertainty in explanation. However, on analysing the aforementioned studies and other contemporary research works conducted in this field, it can be posited that:

- Combinations of uncertainty estimation specifically with gradient-based explanation methods for neural networks have not been exhaustively explored.

[2] `https://gdpr-info.eu/`



(a) The neural network (segmentation model) produces a prediction for the given input image and the explanation method subsequently highlights the region (green pixels) that are considered important by the model for making the prediction.



(b) Working of the Monte Carlo (MC) Guided Backpropagation approach [93] for the context of semantic segmentation. This approach uses the combination of MC Dropout [23] and Guided Backpropagation (GBP)[75] to obtain the uncertainty associated with the explanation output of the neural network. The explanation/heatmaps generated by the different instances of the model are combined using their standard deviation. This results in the (un)certainty in the explanation of the model. This is depicted using the red and green pixels which indicate that these pixels might be important for the prediction. Furthermore, red pixels are those where the uncertainty associated with feature importance is high and green pixels are those where the uncertainty associated with feature importance is low. The labels [A], [B], [C] and [D] depict different components of the pipeline and are used as a reference in the description of Figure 1.2. Image adapted from [93].

Figure 1.1: (a) A standard overview of how explanations/heatmaps are generated (b) The approach discussed in [93] to incorporate the underlying uncertainty in the explanation of the network output.

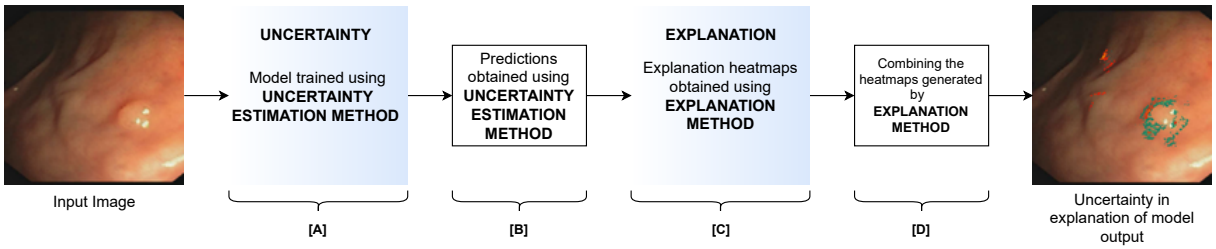


Figure 1.2: Overview of the proposed approach to explore the underlying uncertainty associated with the explanation of a neural network output. The labels [A], [B], [C] and [D] depict the same components in the pipeline shown in Figure 1.1b

- The explanation methods used in combination with uncertainty methods either rely on certain predefined conditions/constraints or are computationally intensive.
- The approaches proposed to analyse the uncertainty in explanation of a neural network primarily focus on visual tasks such as classification, segmentation etc.

1.2.1 Thesis Objective

This thesis addresses the aforementioned points by:

- Exploring combinations of uncertainty estimation methods and gradient-based explanation methods to ascertain the uncertainty in the explanation of a neural network.
- Applying the uncertainty in explanation of a neural network approach to non visual dataset, possibly a regression dataset to test the robustness.
- Evaluate the performance of identified combinations of uncertainty method and explanation methods.

Figure 1.2 depicts the proposed approach that could be used to solve the problem at hand. Multiple combinations of uncertainty estimation approaches and explanation methods can be tested by substituting the uncertainty estimation method in component [A] and explanation method in component [C] of the pipeline. This can be represented in the form of Equation 1.1.

$$\{Uncertainty\ Method\}_i \times \{Explanation\ Method\}_j = \{Combination\ id\}_k \tag{1.1}$$

where i, j and k denote the count of uncertainty methods, explanation methods and their corresponding combinations respectively. By applying uncertainty methods on the input image, we can generate a distribution of predictions (represented by component [B]). Explanation methods can be applied on the distribution of predictions which then generates a distribution of explanation/heatmaps (represented by component [C]). These heatmaps could be merged by suitable combination operation (represented by [D]) to yield a heatmap that represents the uncertainty in the explanation of the network output.

1.2.2 Research Questions

In order to systematically explore the aforementioned research direction, we have formulated certain Research Questions (RQs) that can be helpful to solve the problem at hand.

RQ1: Could additional combinations of uncertainty estimation methods and explanation methods be implemented to ascertain the uncertainty associated with the explanation of neural networks?

RQ2: How can the explanations/heatmaps generated by approaches discussed in RQ1 be combined/processed to compute a condensed representation of uncertainty in explanation?

RQ3: What are the possible approaches to analyse the concise explanation heatmap generated by the methods discussed in RQ2?

RQ4: Which combination of uncertainty estimation method and explanation method identified in RQ1 is the best to analyse the uncertainty associated with the explanation method output for a neural network?

1.3 Structure

This report consists of seven chapters (including the present chapter) with each dedicated to dealing with different aspects of the research. Chapter 1 introduced the topic, discussed the motivation, problem statement, relevant research questions and objective of the thesis. Also, an outline of the thesis report has been provided. In Chapter 2, a brief background of uncertainty in context of neural networks along with an overview of the methods developed to compute uncertainties is provided. A section is dedicated to cover the introduction of explainability and explanation methods. This serves as a foundation for the concepts discussed in subsequent chapters. In Chapter 3, a literature review of the methods developed to ascertain the uncertainty in explanation methods for neural networks has been provided. Along with the description and analysis of the said methods, an overview of different explanation approaches developed in the field of XAI have also been discussed. Chapter 4 describes the proposed pipeline to analyse uncertainty associated with explanation. Chapter 5 discusses the details of the datasets used, architecture of the neural networks trained, specifications of the network training and the implementation details of the experiments conducted in this thesis. The results obtained from the experiments conducted have been discussed in Chapter 6. Along with it, the analysis of these results and observations have also been provided in this chapter. Finally, Chapter 7 concludes the report by enumerating the contributions of this research project, conclusions and insights obtained and discussing possible research directions for the future.

2

Theoretical Background

The previous chapter provided an introduction to the topic of ascertaining uncertainty in explanation methods for a neural network, discussed the motivation and the problem statement of this thesis and outlined the research questions pertaining to it. This chapter discusses uncertainty and the methods to compute uncertainty in the purview of neural networks. Also, an introduction to explainability and explanation methods has been provided along with a discussion of the properties and categorization of the explanation methods. The contents of this chapter serve as a knowledge base for the text in the subsequent chapters that deal with the methods developed to ascertain the uncertainty in the outcome of an explanation method when applied to a neural network.

2.1 Uncertainty in neural networks

Uncertainty can be understood as the lack of confidence or surety in the state of a system. This definition is applicable to neural networks as well. The authors of [34] propose that the uncertainty in context of neural networks can be broadly divided in two categories namely:

- Aleatoric uncertainty: The uncertainty that arises from noise in the training data/distribution and measurement process. This can not be reduced by acquiring additional data as this uncertainty represents inherent randomness associated with real world data [32], [47].

- Epistemic uncertainty: The uncertainty stemming from the lack of sufficient data for the model to train. This type of uncertainty is interpreted as the uncertainty in the model itself and can be reduced by using additional data [32], [47].

Figure 2.1 highlights the distinction between the two types of uncertainties for a basic model with one input (x-axis) and one target (y-axis) dimension. The vertical spread amongst the black crosses ($\mathbf{X}$) for a given input value can be considered as the aleatoric uncertainty whereas missing data points along the input dimension (x-axis) contribute to the epistemic uncertainty. It must be noted that the discussion of different types of uncertainties is provided as an introduction to the concept and that for the scope of this thesis and for the subsequent discussions conducted in this report, the term “uncertainty” will refer to the epistemic uncertainty.

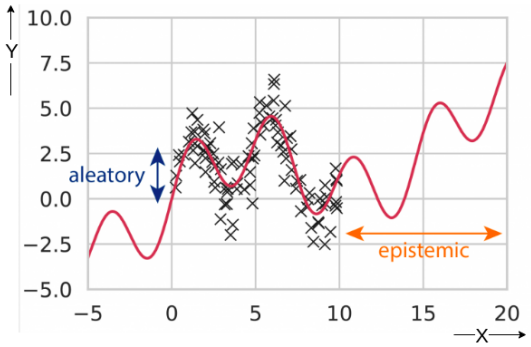


Figure 2.1: Illustration depicting the difference between aleatoric and epistemic uncertainty. Image adapted from [67].

2.2 Uncertainty Estimation methods for neural networks

This section introduces the basics of Bayesian Neural Networks (BNN), how they differ from the standard neural networks and how they can be used to estimate the uncertainties in neural networks. Additionally, non-Bayesian approaches have also been discussed that could potentially be used to obtain uncertainty estimates.

2.2.1 Bayesian Methods and their approximations

Though neural networks have been performing increasingly well in recent years, they are unable to provide the uncertainty information of their output [5], [83]. This inability can be attributed to the fact that the parameters (namely the weights and the biases) of a standard neural networks are point estimates and hence the prediction obtained from such networks are singular points as well and have no uncertainty component to them. In order to obtain predictions with uncertainty estimates, a new type of networks namely Bayesian Neural Networks (BNNs) have been proposed. In such networks, the parameters are modeled as probability distributions rather than as point estimates. As the parameters of these network are probabilistic distributions, the prediction of such network output is a distribution as well. The difference between a standard neural network and Bayesian Neural Networks can be seen in Figure 2.2.

In Bayesian learning, the conditional distribution of weights given the training data $p(w|D)$ (also known as posterior) is given as Equation 2.1[1]:

$$p(w|D) = \frac{p(D|w)p(w)}{\int_w p(D|w')p(w')dw'} \tag{2.1}$$

where $D = \{x_i, y_i\}_{i=N}$ is the labeled training set with N examples, w represents parameters (weights and biases), $p(w)$ is the prior and $p(D|w)$ is the likelihood. The objective in Bayesian learning is to maximise the posterior during the training phase. For inference, having the $p(w|D)$ is beneficial as using

[1]This equation is taken from [3]

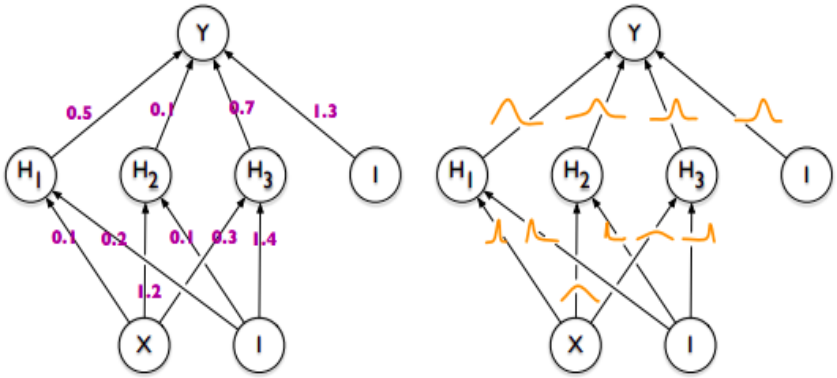


Figure 2.2: Parameters in a standard neural network are point estimates (left) and in a Bayesian Neural Network (BNN) (right) are probability distributions. Image taken from [16].

this, prediction (y) can be calculated as a probability distribution for a given input (x). Equation 2.2[2] can be thought of as an ensemble of numerous models where an individual model is assigned importance according to the calculated posterior that results in the output being a probability distribution.

$$p(y|x, D) = \int p(y|x, w)p(w|D)dw \tag{2.2}$$

Though Equation 2.1 yields the exact inference, for all practical purposes it is expensive to calculate due to the denominator where an integration over all the weights in a network is required. As the number of weights in a typical neural network is very large, this term is typically intractable.

To solve this problem, approaches have been proposed to approximate BNNs in order to make them easier to train [48]. Building upon the concepts discussed in Variational Inference [27] as well as Bayesian Backpropagation [17], authors of [16] propose to model the true posterior ($p(w|x)$) with an approximate distribution ($q_\phi(w)$) that is parameterized by ϕ (the μ (mean) and σ^2 (variance) of the distribution). The approximate posterior is compared to the true posterior using the Kullback-Leibler (KL) divergence [39] as follows:

$$KL(q_\phi(w)||p(w|x)) = \int q_\phi(w) log \frac{q_\phi(w)}{p(w|x)} dw \tag{2.3}$$

From Equation 2.3, it can be observed that the true posterior term will be required for the computation of KL divergence however, this quantity is not available. This problem can be circumvented, by the following mathematical restructuring[3]:

$$KL(q_\phi(w)||p(w|x)) = \int q_\phi(w) log \frac{q_\phi(w)}{p(w, x)} dw + \int q_\phi(w) log\ p(x) dw \tag{2.4}$$

[2]This equation is taken from [3]
[3]This derivation is taken from [87]

Rearranging the terms in Equation 2.4:

$$log\ p(x) = -\int q_\phi(w) log \frac{q_\phi(w)}{p(w,x)} dw + KL(q_\phi(w)||p(w|x)) \tag{2.5}$$

The first term in the right side of Equation 2.5 is defined as the Evidence Lower Bound (ELBO) and according to [48], minimising the KL divergence is equivalent to minimizing the negative ELBO (or maximising the positive ELBO). Further analysis on the first term in Equation 2.5 results in:

$$L(q_\phi(w)) = \int q_\phi(w) log\ \frac{p(x|w)p(w)}{q_\phi(w)} dw \tag{2.6}$$

Splitting the argument of log as per the rules and rewriting:

$$L(q_\phi(w)) = \int q_\phi(w) log\ p(x|w) dw - \int q_\phi(w) log\ \frac{q_\phi(w)}{p(w)} dw \tag{2.7}$$

Rewriting Equation 2.7 as:

$$L(q_\phi(w)) = E_{q_{\phi(w)}} log(p(x|w)) - KL(q_\phi(w)||p(w)) \tag{2.8}$$

where $L(q_\phi(w))$ denotes the ELBO.

Two prominent works namely Monte Carlo Dropout [23] and Monte Carlo DropConnect [48] analyse the relation between optimizing for ELBO in context of a BNN and minimizing the loss for a neural network with dropout/dropconnect. The distinction between the dropout and dropconnect can be seen in Figure 2.3:

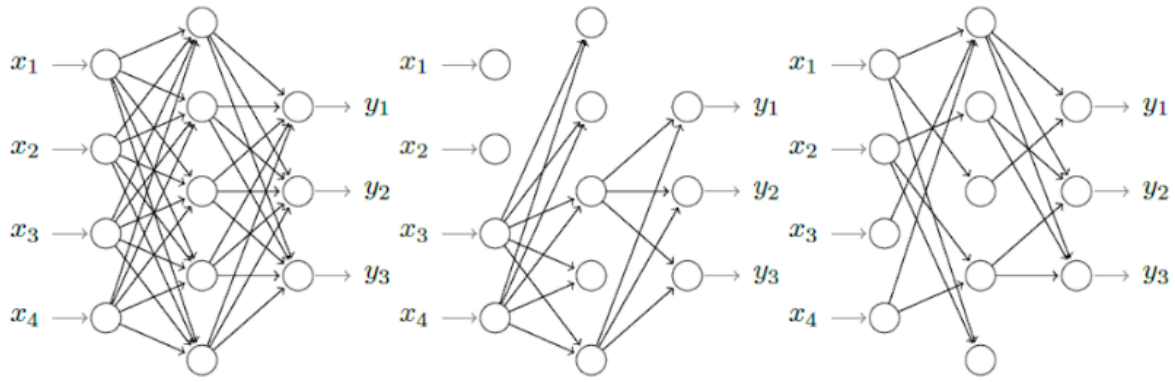


Figure 2.3: Schematic for (i) standard neural network (ii) network with Dropout and (iii) network with DropConnect. Image taken from [74].

These works ([23], [48]) incorporate the concepts of Bernoulli distribution, Variational Inference and Monte Carlo sampling (as the name suggests) in conjunction. The following analysis for the aforementioned mathematical equivalence has been taken from [48]. For a fully connected neural network with L layers, the activation vector of i^{th} layer (having K_i units) is provided as:

$$a_i = \sigma(w_i.x) \tag{2.9}$$

where $\sigma(.)$ is the non-linear activation function, x is a K_{i-1} dimensional input and w_i is the weight matrix. On application of dropout to the output of layer under consideration, the activation of individual neuron is multiplied by a sample from a Bernoulli distribution. This operation can be written as:

$$a_{i\ [dropout]} = \sigma(z_i \odot (w_i.x)) \tag{2.10}$$

where z_i is a sample binary vector of K_i dimensions drawn from $z_i^{(k)} \sim Bernoulli(p_i)$ for $k = [1, 2, \ldots, K_i]$, p_i as the probability of keeping the activation and $\odot$ represents the Hadamard product. Similarly, the layer output can be defined when dropconnect is applied as:

$$a_{i\ [dropconnect]} = \sigma((z_i \odot w_i)x) \tag{2.11}$$

where z_i now represents a binary matrix of the same shape as w_i. In this case, the Bernoulli dropping is applied to each weight directly rather than on each output unit. Assume a BNN having L layers with weights defined as $w = \{w_i\}_{i=1}^{L}$. This network is optimized using variational inference by approximating the distribution $q(w_i|\Theta_i)$ for each layer i as:

$$w_i = \Theta_i \odot z_i \tag{2.12}$$

where Θ_i is the matrix of parameters to be optimized and z_i is a binary matrix that follows a Bernoulli distribution. With this, rewriting the first term of Equation 2.8 as a summation for all samples and replacing the integral with Monte Carlo estimates:

$$\sum_{n=1}^{N} \int -q_\phi(w) log\ p(y_n|x_n, w) dw \tag{2.13a}$$

$$\frac{1}{N} \sum_{n=1}^{N} -log\ p(y_n|x_n, \hat{w}_n) \tag{2.13b}$$

where N denotes the number of samples, $\hat{w}_n$ denotes the Bernoulli distribution realization which is equivalent to the application of dropout and dropconnect to the network. The summation of log probabilities over all samples yields the loss (I_{NN}) and is defined as:

$$I_{NN}(y_n, \hat{y}(x_n, \hat{w}_n)) = -log\ p(y_n|x_n, \hat{w}_n) \tag{2.14}$$

where $\hat{y}(x_n, \hat{w}_n)$ corresponds to the output of the BNN.

In the discussions in [23], the authors show that the KL divergence term is equivalent to $\sum_{i=1}^{L} ||\Theta_i||_2^2$. Combining these, the ELBO can be formulated as:

$$L_{MC} = \frac{1}{N}\sum_{n=1}^{N} I_{NN}(y_n, \hat{y_n}) + \lambda \sum_{i=1}^{L} ||\Theta_i||_2^2 \tag{2.15}$$

Equation 2.15 is identical to the objective function optimized when training a network using $L2$ weight regularization and dropout/dropconnect. For the inference phase, the dropconnect/dropout layer is kept "on" to keep the Bernoulli distribution over the weights active and each forward pass yields a Monte Carlo sample from the posterior distribution. These samples can then be used to compute the uncertainties. Equation 2.16 shows an adapted form of Equation 2.2 for prediction phase in this setting where the true posterior has been replaced with approximate posterior and the integral has been approximated with Monte Carlo integration as:

$$p(y|x, D) \approx \int p(y|x, w) q_\phi(w) dw \tag{2.16a}$$

$$\approx \frac{1}{T}\sum_{t=1}^{T} p(y|x, \hat{w}_t) = p_{MC}(y|x) \tag{2.16b}$$

where $\hat{w}_t \sim q_\phi(w)$, y and x are the unknown label and the input data respectively.

A problem while training variational BNN is the lack of variation in the sampling of kernels in a batch [84]. The weights in these networks can be written as:

$$w = \mu + \sigma \odot \epsilon \tag{2.17}$$

where μ denotes the mean, σ denotes the standard deviation and ϵ denotes a sample from $N \sim (0, 1)$. For any input x, the output distribution is given as $y \sim N(\mu, \sigma)$. The problem lies in the sampling procedure, in that the same sample ϵ is used for all the weights throughout the network in each forward pass. This is necessitated by the prohibitive computational cost of using a unique sample for each forward pass and subsequently results in the gradients to be correlated and thereby preventing the reduction in variance during the training [78], [84].

Authors of Flipout [91] propose to solve the problem of correlating gradients by generating per sample perturbations. The idea is to add independence to the samples of weight in a single batch. The fundamental concept of flipout can be understood in Figure 2.4:

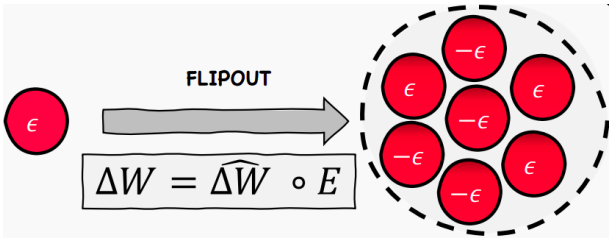


Figure 2.4: Illustration depicting the underlying principle of Flipout. Image taken from [78].

where E is the random sign matrix multiplied to the weight samples ($\Delta\hat{W}$) to yield the perturbed weights (ΔW). Here, the process depicted in Figure 2.4 is repeated for the weights in every mini-batch and

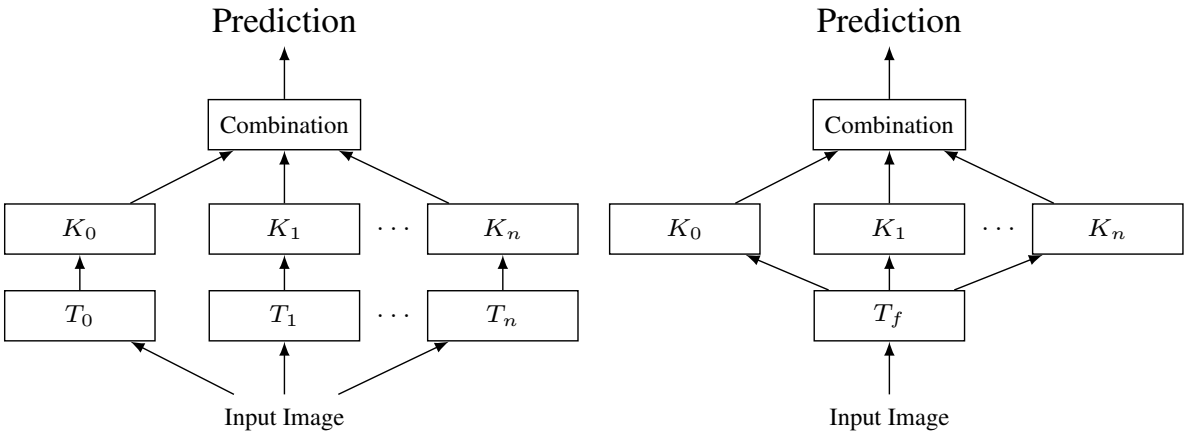
in this manner stochasticity is introduced in the weights of the model. As per the discussions conducted in [84], the equation in Figure 2.4 can be written as:

$$\Delta W_n = \Delta \hat{W} r_n s_n^T \tag{2.18}$$

where r_n and s_n denote the independent samples drawn from a Rademacher distribution ({-1, 1}) and ΔW_n is the amount of perturbation to be added to the kernel.

2.2.2 Non-Bayesian methods

Deep Ensembles [40] is a noteworthy non-Bayesian approximation method. Since the weights are no longer modeled as probability distribution, the method belongs to the non-Bayesian category. The underlying idea is to train a number of networks having the same architecture but with randomly initialized weights. This results in trained models with slightly different final weights owing to random weight initialization and subsequently different optimization trajectories. The prediction results obtained from these trained models can then be combined by averaging them [40]. Though this approach is simple in terms of implementation, it suffers from high computation and memory requirements due to multiple training runs [5]. This drawback is addressed to some extent by the Deep Sub-Ensembles [82]. In this approach, a part of the network (see K in Figure 2.5) is trained multiple times from random initialization. By doing this, the overload of training the entire network multiple times from random initialization is reduced. The difference between Deep Ensembles and Deep Sub-Ensembles is depicted in Figure 2.5.



(a) Deep Ensembles

(b) Deep Sub-Ensembles

Figure 2.5: Difference between Deep Ensembles and Deep Sub-Ensembles for n components. The sub-branches K are trained multiple times in Deep Sub-Ensembles. Image taken from [82].

Table 2.1 provides a summary of the uncertainty estimation methods developed for neural networks and has been compiled from the analysis of discussions conducted in [5], [24], [84].

Method	Category	Notes
Variational Inference [27]	Approximate Bayesian	Approximate the true posterior distribution by a known functional distribution. The parameters of the assumed distribution can be estimated by minimizing the KL divergence between true distribution and approximate distribution.
Bayes by Backpropagation [16]	Approximate Bayesian	Approximates the weights in the model as Gaussian distributions with parameters μ (mean) and σ^2 (variance). These are learned using Monte Carlo gradient descent and the inference is conducted by taking samples of the weights and then applying the forward pass on the input [84].
Monte Carlo Dropout [23]	Approximate Bayesian	Enables Dropout [76] during test (inference) time as well such that the model becomes stochastic. This is achieved by setting the activations in a layer to zero with a predefined probability. This modified model produces samples from the Bayesian posterior distribution for each forward pass which can be used to compute the μ (mean) and σ^2 (variance) of the prediction distribution.
Monte Carlo DropConnect [48]	Approximate Bayesian	Enables DropConnect [88] during test (inference) time. This is achieved by randomly dropping the weights to zero instead of activations (as is done in Dropout). The probability of dropping weights is a tunable hyperparameter. The resulting model assumes stochastic behaviour and produces samples that can be used to calculate uncertainties.
Flipout [91]	Approximate Bayesian	Improves upon the Bayes by Backprop approach by reducing the variance in the prediction distribution. This is achieved by ensuring that during sampling, each sample in a batch receives a different kernel [84].
Deep Ensembles [40]	Non-Bayesian	Trains multiple copies of same network with different random weight initialization for every copy. The outputs from these multiple copies are then combined which usually performs better than an individual model.

Method	Category	Notes
Sub-Ensembles [82]	Non-Bayesian	Divides the network architecture into 2 sub-parts, namely Trunk (T) network and Task (K) network. Multiple instances of K are trained by fixing the weights of T to reduce the computational costs and memory requirements.

Table 2.1: An overview of the uncertainty estimation methods in neural networks.

2.3 Explanations and Explainability

Explanations refer to statements, justifications or reasons that clarify an action or a belief. In context of neural networks, *Explainability* is defined as the ability to provide an answer or a reason (in other words "explanation") for the network output in a human understandable format [15]. Interestingly, another term namely *Interpretability* has also been discussed in some of the researches in the domain of XAI. Research works such as [25] and [56] argue that "interpretability" and "explainability" should be treated separately and attempt to dissociate them. However, others such as [15] and [61] use these terms interchangeably. In this thesis, we do not differentiate between the two terms and use only the term "explainability". Many methods have been proposed that are used to generate explanations and are applicable to neural networks. These methods are referred to as *Explanation methods* in this report.

As described earlier, the explanations are generated in a format that is understandable by humans and the format differs depending on the input modality being explained. For instance, in context of a neural network that uses tabular data, the explanation could be in the form of a bar diagram that highlights the importance of individual input feature while making a particular decision. The height of the bars in such a diagram could provide an intuition of how much importance has been assigned to individual feature by the network while calculating the prediction. Additionally, the explanation for inputs in the form of images can be heatmaps or saliency maps. Such heatmaps highlight the region/pixels in the input image that the model considered to be important while making a decision. In both these cases, these explanations make it convenient for a human to understand which aspects (features/pixels) were responsible and how much weight was assigned (height of the feature bar or intensity of the heatmap pixel) to each of them to reach a particular decision. As has been discussed earlier, these explanations could be used in a variety of scenarios for troubleshooting, optimizing and also designing networks for critical applications.

2.3.1 Properties of Explanation methods

The properties of an explanation method can be formulated as per the discussions conducted in [49]. These are as follows:

- Expressive Power:
  This property is associated with the format of explanation that a method can generate. For instance,

methods could generate explanations of different formats ranging from feature plots and heatmaps to natural language or decision trees and so on.

- Translucency:
  It describes the degree of reliance of explanation method on the internal aspects of a network like the weights and structure. Highly translucent methods are those that are intrinsically understandable. Conversely, methods that only rely on manipulating inputs and observing the predictions to generate explanations have low translucency. Such methods are generally more portable as they treat the model as a black box and can be robustly applied in different settings.

- Portability:
  It is the range of models to which the explanation method can be applied. Higher portability means that such a method is applicable to a variety of models/networks.

- Algorithmic Complexity:
  This property relates to the computational complexity of the explanation method. It is important to consider the bottlenecks in the process of generating explanations.

A quantitative measure of the properties listed above is not formalized yet (as claimed by [49]) however, these properties in their current capacity could be used for approximate qualitative analysis and to decide which explanation methods to implement.

2.3.2 Categorization of Explanation methods

With a number of research works focussing on XAI in recent times, it has become necessary to group the methods in distinct categories for systematic analysis. A few research works and surveys such as [6], [9], [12], [43], [49], [50], [59], [61], [70], [80] offer a comprehensive analysis of these methods and their categorization. The following categorization of explanation methods has been compiled from the aforementioned works:

- Global vs Local methods:
  Global explanation methods are those that are applicable on the entire model and focus on its overall knowledge and the general behaviour. A potential drawback of global methods could be that these might not provide substantial explanation information regarding the feature importance for predictions near the decision boundary for say, a classification task [61]. Local methods generate explanations for individual prediction of the network and thus could prove useful while working with a particular subset of the input sample space.

- Intrinsic vs Post-Hoc methods:
  If explainability of the model is attained by constraining the model complexity (and thereby making it inherently easy to understand), such approaches are referred to as Intrinsic approaches. However, if the approach explains an already trained model, such type of methods are called Post-Hoc methods.

A salient feature of post-hoc methods is that these can be applied to intrinsically explainable model as well [49].

- Model-Agnostic vs Model-Specific methods:
  This distinction is made on the basis of the degree of dependence of explanation method on the structure of the model being explained. Model-Agnostic approaches are those that do not require the model to have a certain architecture. Conversely, if an explanation method necessitates a change in the network structure or only works on a particular type of networks, then such a method is called Model-Specific. Generally, model-agnostic methods work in a post-hoc manner and can not access the internal aspects of model such as its parameters and/or structural information [49]. Also, model-agnostic approaches treat the network to be explained as a black-box.

3

State of the Art

The previous chapter discussed the background of uncertainty and the methods to ascertain the uncertainty in neural networks. The basics of explainability and explanation methods were also covered. This chapter focuses on the methods developed to explain the output of a neural network (Explainable Artificial Intelligence (XAI) methods). Subsequently, those methods that help in ascertaining the uncertainty in explanation methods of a neural network are discussed. For the sake of clarity, the text has been divided into subsections discussing each topic sequentially.

3.1 Explanation methods for neural networks

The subsequent text discusses the working of the methods developed primarily to explain the output of neural networks. The majority of the explanation methods discussed in this thesis depend on generation of explanation using gradient computation of one or the other form as these methods are fast and efficient, make use of networks structural information, and use the core of learning principle of neural networks (backpropagation) that is, gradients to compute the explanations [65].

Gradients

Simonyan et al. [69] proposed this method to generate the saliency maps for a given input image. In this method, the gradient of the predicted class score is calculated with respect to the input image pixels. The method affords the flexibility to either visualize the absolute values of the gradients or the positive and negative contributions separately. This method however, suffers from a problem of gradient saturation because of the way the gradients are propagated back through the ReLU. The resulting heatmap using this method might not correctly represent the important regions of the image.

$$activation => \ f_i^{l+1} = relu(f_i^l) = max(f_i^l, 0) \tag{3.1a}$$

$$backpropagation/gradients => \ R_i^l = (f_i^l > 0).R_i^{l+1}, where\ R_i^{l+1} = \frac{\partial f^{out}}{\partial f_i^{l+1}} \tag{3.1b}$$

where f represents the feature map produced by the network layers, i denotes the feature map id, f^{out} refers to the output activation, R is the intermediate result obtained during backpropagation at any layer

l. Equation 3.1 has been taken from [75] and using this, it can be understood that only those gradient values will be allowed to backpropagate through ReLU where the inputs were positive during the forward pass.

Deconvolutional Network

A modification of the Gradients approach is proposed by Zeiler et al. [97]. This is achieved by reversing the operations like pooling, filtering and activation layers in a neural network while backpropagating to the input. The reverse of convolution is calculated by transposing the kernel used during the forward pass. This operation is called as Deconvolution. For reversing the pooling operation, "switches" are used that store the position of maximum activation in the kernel during the forward pass. Additionally, the gradient backpropagation through ReLU is slightly modified. In this approach, negative value of gradients are set to 0 when backpropagating via ReLU. This is equivalent to the application of ReLU on the gradient signal that is propagating backwards.

$$deconvolution => \; R_i^l = (R_i^{l+1} > 0).R_i^{l+1} \tag{3.2}$$

where the symbols hold the same meaning as in Equation 3.1. Equation 3.2 has been taken from [75] and demonstrates the principle of deconvolution approach, in that it essentially applies ReLU to the gradient signal being backpropagated.

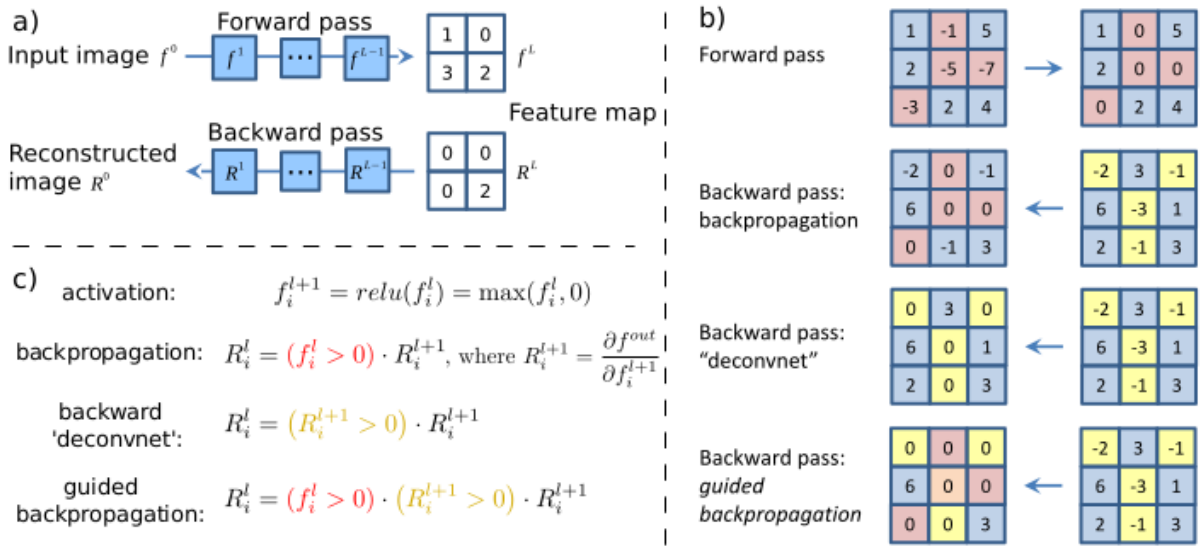


Figure 3.1: Illustration highlighting the difference between gradients, deconvolution and guided backpropagation approaches. (a) shows the forward and the backward pass, (b) shows the modifications of the backward pass for the 3 approaches and (c) shows the difference between the 3 approaches in their mathematical formulation. Image taken from [75].

Guided Backpropagation (GBP)

Springenberg et al. [75] proposed to utilise the ideas of previous two approaches and came up with Guided Backpropagation (GBP). In this method, the gradient backpropagation is modified such that the gradient is latched to 0 either if the input to that ReLU during forward pass is negative or the gradient being backpropagated itself is negative. Equation 3.3 provides a mathematical perspective of the approach:

$$guided\ backpropagation => \ R_i^l = (f_i^l > 0).(R_i^{l+1} > 0).R_i^{l+1} \tag{3.3}$$

where the symbols hold the same meaning as in Equation 3.1. Equation 3.3 has been taken from [75] and demonstrates that GBP uses the concepts from [69] and [97]. As an additional insight, Figure 3.1 provides a consolidated overview of three approaches discussed till now. A few advantages of GBP are that it is straightforward to implement, generates less noisy heatmaps as compared to Gradients and Deconvolutional network and this method is not computationally extensive.

Integrated Gradients (IG)

This method generates explanations by computing the integration of gradient of output with respect to the input features along a linear path from a baseline image to the input image. The path consists of interpolated images with the baseline image serving as a starting point of the linear path and the input image as the final point. The baseline image is often selected to be an all-black image [77]. Equation 3.4 demonstrates the mathematical formulation of Integrated Gradients and has been taken from [77].

$$\text{Integrated\_Gradients}_i(x) ::= (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial F\left(x' + \alpha \times (x - x')\right)}{\partial x_i} d\alpha \tag{3.4a}$$

$$\text{Integrated\_Gradients}_i^{approx}(x) ::= (x_i - x_i') \times \sum_{k=1}^{m} \frac{\partial F\left(x' + \frac{k}{m} \times (x - x')\right)}{\partial x_i} \times \frac{1}{m} \tag{3.4b}$$

where, x denotes the input image
x' is the baseline image
$(x - x')$ denotes the distance of original image from baseline
α is the factor of interpolation that determines the amount of feature perturbation
i is the feature (pixel in case of images)
m is the number of steps in Riemann approximation of the integral
$F()$ is the function mapping the input to the output, and
$\frac{\partial F()}{\partial x_i}$ gradient of model output with respect to each input features x_i
$k = [1, 2, \ldots, m]$.

It should be noted that Equation 3.4b is the Riemann approximation of the integral of Equation 3.4a and is comparatively computationally efficient. IG is applicable to additional scenarios such as question classification, neural machine translations and feature analysis of chemical systems [77].

SmoothGrad

Smilkov et al. [73] proposed an approach that sharpens the saliency maps generated by other gradient-based explanation methods. It should be noted that this is not a standalone explanation approach. The underlying principle is to take an input image, create corrupted versions of this image with increasing amount of noise added to the original image and then compute the average of the saliency maps generated for the corrupted copies. Computing the average of multiple maps smooths the spurious heatmaps generated by the explanation method. Equation 3.5 has been taken from [73] and mathematically presents the working of SmoothGrad:

$$\hat{M}_c(x) = \frac{1}{n}\sum_{1}^{n} M_c\left(x + \mathcal{N}\left(0, \sigma^2\right)\right) \tag{3.5}$$

where x is the input, n is the number of corrupted image copies, σ is the standard deviation of the distribution from where noise is sampled, M_c denotes the saliency map operation on individual samples and $\hat{M}_c$ is the average saliency maps of all samples and is also the output of SmoothGrad approach.

Layer-wise Relevance Propagation (LRP)

Bach et al. [13] proposed an approach called the Layer-wise Relevance Propagation (LRP) to explain the decision of a network. LRP operates by propagating the prediction back to the input to calculate the relevance of input features. This is achieved using specifically designed local propagation rules [51]. The underlying principle is that each neuron redistributes a proportionate amount of information to a lower layer neuron that it received from the higher layer.

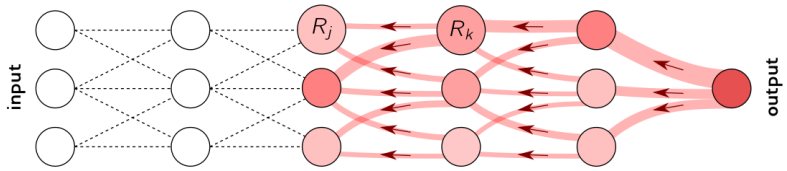


Figure 3.2: Working of Layer-wise Relevance Propagation (LRP) approach. The arrows depict the flow of relevance from the output back to the input. Image taken from [51].

These rules (originally proposed by [13]) have been explained by Montavon et al. [51] and are as follows:

$$R_j = \sum_k \frac{a_j w_{jk}}{\Sigma_{0,j} a_j w_{jk}} R_k \tag{3.6a}$$

$$R_j = \sum_k \frac{a_j w_{jk}}{\epsilon + \Sigma_{0,j} a_j w_{jk}} R_k \tag{3.6b}$$

$$R_j = \sum_k \frac{a_j (w_{jk} + \gamma w^+_{jk})}{\Sigma_{0,j} a_j (w_{jk} + \gamma w^+_{jk})} R_k \tag{3.6c}$$

where w is the weight between two neurons j and k of any successive layers, R denotes the relevance score,

a denotes the activation of neurons, ϵ controls the sparsity in the output explanation map (the larger the value of ϵ, the sparser the output explanation map), and γ is the parameter that controls the amount of positive contributions. Equation 3.6a is the basic LRP rule whereas Equation 3.6b and Equation 3.6c demonstrate the LRP-ϵ and LRP-γ rules respectively. The purpose of these variations of basic rule is to help tune the explanations as required. Interestingly, when the value of ϵ and γ is 0, the special rules converge to the basic LRP rule.

Class Activation Mapping (CAM)

Zhou et al. [101] proposed a method to make use of Global Average Pooling (GAP) [42] layer in CNNs to generate Class Activation Maps (CAM). A CAM for a specific class highlights the discriminative image regions that the network uses to identify that particular class in a given input [101]. The process of computing CAM for a given input image is demonstrated in Figure 3.3.

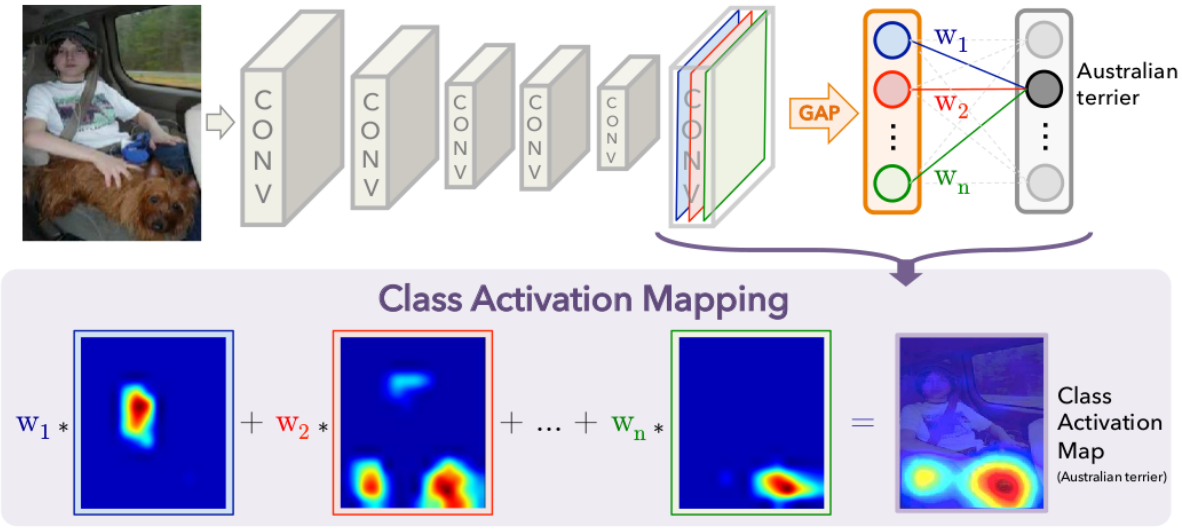


Figure 3.3: Working of the Class Activation Maps (CAM) approach. The desired class is highlighted appropriately in the explanation/heatmap shown in the figure. Image taken from [101].

The highlighted region in the CAM of the input image shows the pixels that are important for the network while making the classification decision, which in Figure 3.3 is an Australian Terrier. A limitation of CAM approach is that it is applicable only to those network architectures where the feature maps of the last convolutional layer are passed through a GAP layer that ultimately precedes the final softmax layer [62].

Gradient-weighted Class Activation Map (Grad-CAM)

Gradient-weighted Class Activation Map (Grad-CAM) is a generalization of CAM and has been proposed by Selvaraju et al. [62]. The underlying principle in Grad-CAM is identical to that of CAM, in that

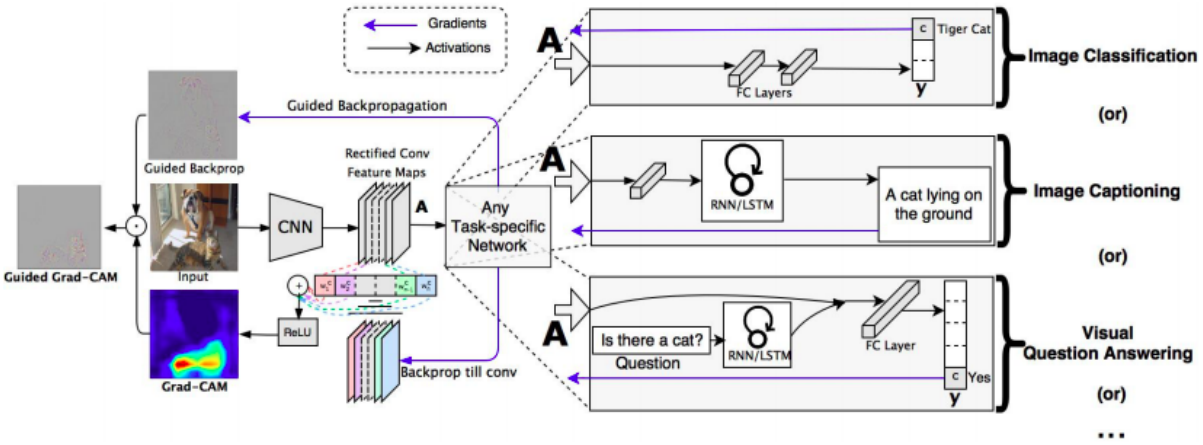


Figure 3.4: Overview of the Gradient-CAM and Guided Grad-CAM method. Image taken from [62].

the spatial information that is preserved in the convolution layer can be exploited to understand which regions in the input image were responsible for the network prediction. In this approach, the feature maps of the convolution layer are weighted using "alpha" values that are calculated on the basis of gradients. Equation 3.7 has been taken from [62].

$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}} \tag{3.7a}$$

$$L^c_{\text{Grad-CAM}} = \text{ReLU}\underbrace{\left(\sum_k \alpha_k^c A^k\right)}_{\text{linear combination}} \tag{3.7b}$$

where y^c is the class score before applying softmax for class c, α is the weighting factor for the feature maps, k denotes the index of the feature map, A is the feature map, i and j denote the dimensions (width and height respectively) of the feature map, and Z is the product of i and j for a given feature map.

Equation 3.7a shows the computation of α values. These α values are used in Equation 3.7b to weight the feature maps of the final convolution layer. The ReLU function is applied on this weighted sum to obtain the final heatmap. The authors of [62] also combine Grad-CAM with Guided Backpropagation (GBP) to yield Guided Gradient-Class Activation Map (Guided Grad-CAM). In this method, the heatmaps obtained from Grad-CAM are multiplied element-wise with the heatmaps obtained from GBP to yield a less noisy explanation output. The working of both Grad-CAM and Guided Grad-CAM can be seen in Figure 3.4.

Local Interpretable Model-agnostic Explanation (LIME)

As the name suggests, LIME is a model-agnostic, local explanation method. LIME is applicable to variety of input modalities such as image, text, tabular data. The idea is that the complex learning model can be substituted with a simple surrogate model in a local neighborhood. LIME then uses this simplified model to explain a particular prediction in that neighborhood. The workflow of LIME can be summarized in the following steps:

- Identify an instance for which the explanations are to be generated for a complex model.
- Sample points in the neighborhood of this instance and obtain the predictions (using the original model) for the sampled points. Essentially, this steps corresponds to generating a new dataset in the neighborhood by perturbing the selected instance.
- Assign weights to the perturbed points depending on the proximity to the original instance.
- Train a weighted, surrogate model (that is easily explainable) on the dataset obtained after previous step.
- Explain the original prediction using the surrogate model.

The logic behind training a surrogate model on the perturbed dataset is to ensure that this simple model will replicate the behaviour of the complex model in the proximity of instance selected to be explained. As this is ensured, the explanation of the surrogate model can be used as an explanation of the complex model in that particular neighborhood.

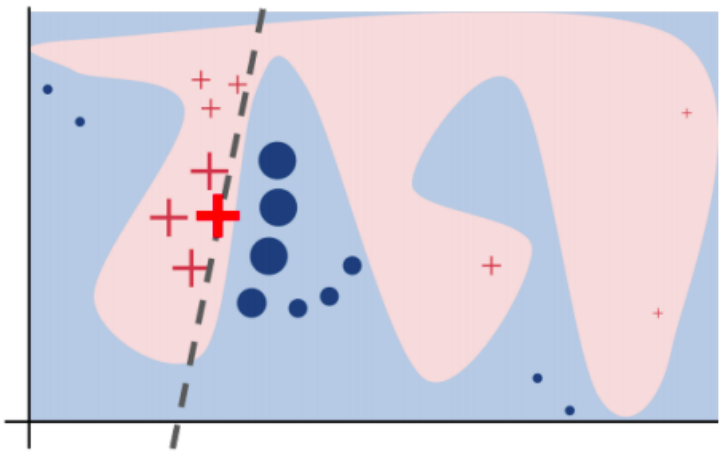


Figure 3.5: Underlying principle of Local Interpretable Model-agnostic Explanation (LIME). The dashed line denotes the simple surrogate model in the neighborhood of the input instance to be explained. The actual decision boundary of the complex model is the interface of pink/blue regions. Image taken from [55].

Figure 3.5 is taken from [55] and the actual decision boundary (of the complex classifier) is represented by the pink/blue interface. The decision boundary of the surrogate model is the dashed line around

which the data points are classified. A drawback of this approach is that the performance depends on a number of factors such as the stability of the perturbation sampling in the neighborhood of the selected instance and the selection of kernel function to compute the weights for perturbed samples. Also, LIME requires training an additional model which could be a computation intensive step in order to generate explanations. Despite these shortcomings, [49] claims it to be a promising method for explaining models.

Shapley values

The idea of Shapley values [63] originated in cooperative game theory and is used to compute the contribution made by input features to the final prediction. Equation 3.8 has been taken from [4] and demonstrates the concept mathematically.

$$\phi_m(v) = \frac{1}{p} \sum_{S} \frac{[v(S \cup \{m\}) - v(S)]}{\binom{p-1}{k(S)}}, \quad m \in [1, 2, \ldots, p] \tag{3.8}$$

where $\phi_m(v)$ denotes the Shapley value (or the contribution to the model prediction) for a given feature member m, p denotes the number of features, S denotes all subsets that can be constructed from $T = \{1, 2, \ldots, p\}$ excluding the feature m, $k(S)$ is the size of S, $v(S)$ is the prediction obtained using S and $v(S \cup \{m\})$ is the prediction obtained after m joins S. For computation of Shapley value of a given feature f, the process is to select all the features randomly except for f, compute the predictions using these different subsets. The next step is to compute the predictions by including the feature f in these subsets. The Shapley value is then the average of the difference of prediction between the subsets having f and not having f. Shapley value can also be understood as the average marginal contribution of that particular feature to the model prediction when considering all the possible feature combinations [4]. A disadvantage of using Shapley values is that they are slow as the computation times increase exponentially with the addition of features and the explanations generated using them are easy to be misinterpreted. An interesting point to note is that on a global level, Shapley values help in examining the prediction of a model and provide an estimate of feature contribution. On a local scale, they allow us to analyse the model's limitations in generalizing to novel data and the possible reasons for failure.

SHapley Additive ExPlanations (SHAP)

Lundberg et al. [44] developed a model-agnostic approach that proposes alternatives to estimating the Shapley values. Two variants of SHAP are KernelSHAP and TreeSHAP. In KernelSHAP, the concepts from LIME and Shapley values are utilized. KernelSHAP makes use of a weighted linear model in which the weights are proportional to those being used in Shapley value estimation. TreeSHAP on the other hand utilises conditional expectation to compute the feature contributions. TreeSHAP is useful only for tree-based machine learning models hence, is model-specific. SHAP can also be used to obtain global interpretations by aggregating Shapley values for all data points [4].

Table 3.1 provides an overview of the explanation methods discussed till now and has been created using the discussions in [20], [43], [70], [80]. The terms *post-hoc*, *local* and *agnostic* used in this table

are the same as described in the previous chapter in Section 2.3.2. The important characteristics and drawbacks of these methods have been listed in the table for an overview as well.

Methods	Post-hoc	Local	Agnostic	Notes
Gradients [69]	✓	✓	✗	Computes gradient of output target neuron with respect to the input. Generates noisy explanations.
Gradients × Input [64]	✓	✓	✗	Pixel-wise product of input with Gradients. Generates less noisy explanation as compared to Gradients.
DeconvNet [97]	✓	✓	✗	Applies ReLU to the gradient computation during backpropagation. Useful for networks with ReLU activations.
GBP [75]	✓	✓	✗	Applies ReLU to both gradient computation and forward pass. Layers in the network should have ReLU activation.
IG [77]	✓	✓	✗	Computes the integration of gradient by approximating local gradients for an input image along a linear path traced from a predefined baseline image. Requires interpolation between baseline and input image.
SmoothGrad [73]	✓	✓	✗	Computes the average saliency map generated by an explanation method for a given input and its corrupted copies. Performance depends on the number of corrupted copies generated and the characteristics of the added noise. Not a standalone explanation method and requires saliency maps as input from another explanation method.
LRP [13]	✓	✓	✗	Propagates the score of output neuron through each layer back to the input based on predefined set of rules.
CAM [101]	✓	✓	✗	Maps the predicted class score back to preceding convolutional layers to generate activation maps. Applicable only to network architectures with GAP layer.

Methods	Post-hoc	Local	Agnostic	Notes
Grad-CAM [62]	✓	✓	✗	Computes the gradient of a target neuron with respect to the feature map activation of preceding convolution layer. Network should contain differentiable layers to allow gradient computation.
Guided Grad-CAM [62]	✓	✓	✗	Computes explanations by combining GBP and Grad-CAM. Network architecture should have layers that are differentiable to allow gradient computation and requires heatmaps as inputs from GBP and GradCAM to generate heatmaps of its own.
LIME [55]	✓	✓	✓	Generates explanations by training a inherently understandable surrogate model in the proximity of individual input sample (for which the explanation is desired). Performance depends on the definition of proximity, stability of perturbation sampling and the choice of surrogate model.
Shapley values [63]	✓	✓	✓	Computes the average marginal contribution of a given feature to the model prediction by considering all the possible feature combinations.
SHAP [44]	✓	✓	✓	Proposes alternatives for computing Shapley values namely KernelSHAP and TreeSHAP. KernelSHAP is a combination of LIME and Shapley values. TreeSHAP works with tree based machine learning models.

Table 3.1: A summary of the explanation methods for neural networks that have discussed in the text. The ticks (✓) indicate that the particular method satisfies the property specified in the column header.

3.2 Uncertainty in Explanation

The previous section provides an overview of the methods that explain the output of a neural network. This section focuses on those approaches that have been developed primarily to ascertain the uncertainty

in the explanation of a neural network. In contrast to the abundance of explanation methods, relatively fewer approaches have been proposed that deal with the uncertainty associated with explanation methods for neural networks as per the claims made in [18], [19], [72] and [99].

Reliable Post-hoc explanations

Slack et al. [72] propose a framework that generates local explanations along with uncertainty estimates. The underlying idea is to extend the existing LIME and KernelSHAP to obtain their respective Bayesian versions that are able to generate confidence intervals. The explanation is modeled as a summation of the feature importance along with an error term that captures the dissimilarity between the generated explanation and the local decision boundary of the model to be explained. This is mathematically presented in [72] as:

$$y|z, \phi, \epsilon \sim \phi^T z + \epsilon \quad \epsilon \sim N(0, \frac{\sigma^2}{\pi_x(z)}) \tag{3.9}$$

where y is the prediction for the perturbed inputs, ϵ denotes the error term, z denotes the perturbations in the neighborhood of input instance (x), ϕ denotes the feature importance and π denotes the proximity function between x and z. Using this formulation ensures that the perturbations close to the input instance x are modeled more accurately as compared to the ones that are farther away. The authors claim that this process ensures that 2 sources of uncertainty in the local explanations are captured namely, feature importance uncertainty and error uncertainty. The feature importance uncertainty is modeled in the ϕ and the error uncertainty is captured in the ϵ. A limitation of this approach is that it is applicable only for explanation methods such as LIME and KernelSHAP, and can not be extended for gradient-based explanation methods.

Understanding uncertainty in LIME

Zhang et al. [99] proposed an approach that attempts to ascertain the sources of uncertainty in the explanations generated using LIME. This work identifies 3 sources of uncertainty namely, uncertainty introduced during selection of parameters for LIME (such as sampling proximity and sample size), variance in explanation across different input points and variance in explanation of a single data point owing to sampling. For analysing the variation introduced due to selection of parameters, the changes in explanation for different parameter settings are compared. The variation in explanation across different data points is ascertained by assessing whether the features identified by explanations for these points are informative or not. In order to test the variation in explanation for a single input instance, explanations are generated multiple times for a single point and the top few features are observed for their cumulative selection probability. This approach is implemented on a total of 3 datasets to test its robustness. A drawback of this research work is that the aspect of uncertainty is only explored in context of LIME and hence would not be applicable to other post-hoc explanation methods discussed in previous sections.

Quantifying uncertainties in Explaining Neural Networks

In this research work by Bykov et al. [18], a framework has been proposed that can model the uncertainties in the explanations of a neural network. This approach, referred to as Bayesian Layer-wise Relevance Propagation (BLRP) generates a distribution of explanations which can then be used to realize the uncertainty in the input feature importance. Figure 3.6 demonstrates the general working of the BLRP approach. With this approach, the different percentiles of the explanation distribution can be sampled and visualized. Having this functionality affords the user freedom to analyse cautious/conservative

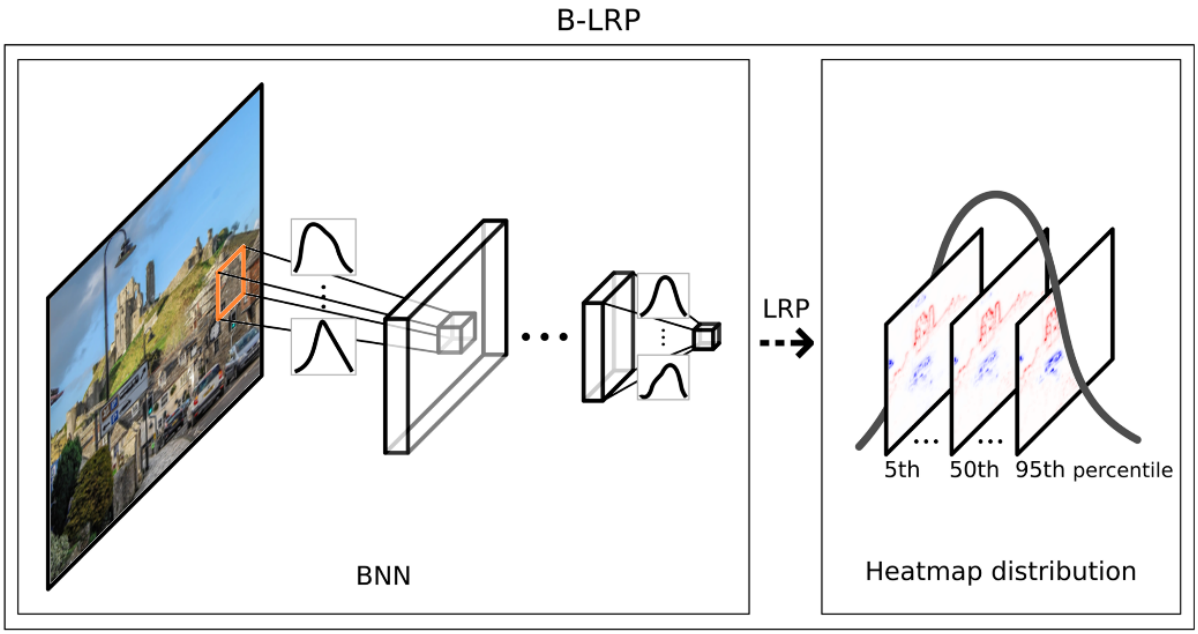


Figure 3.6: Overview of the Bayesian Layer-wise Relevance Propagation (BLRP) approach. Image taken from [18].

explanations (those highlighting feature importance with less uncertainty) or relatively risky explanations (those highlighting feature importance with more uncertainty) depending on the field of application. A limitation of this method is that it discusses ascertaining uncertainty in explanations only for LRP with Monte Carlo Dropout approach.

Explaining BNN

Bykov et al. [19] proposed an approach that generates an uncertainty estimate along with the explanation for a particular model decision. In order to do so, the uncertainty information generated by Bayesian Neural Networks (BNN) is harvested to yield a distribution of explanation. The core principle is to approximate the BNN by sampling from its posterior weight distribution and generating multiple model predictions. Using these predictions, a distribution of explanation can be generated that can be used to analyse the underlying uncertainties. Also, a series of aggregation strategies have been discussed in this

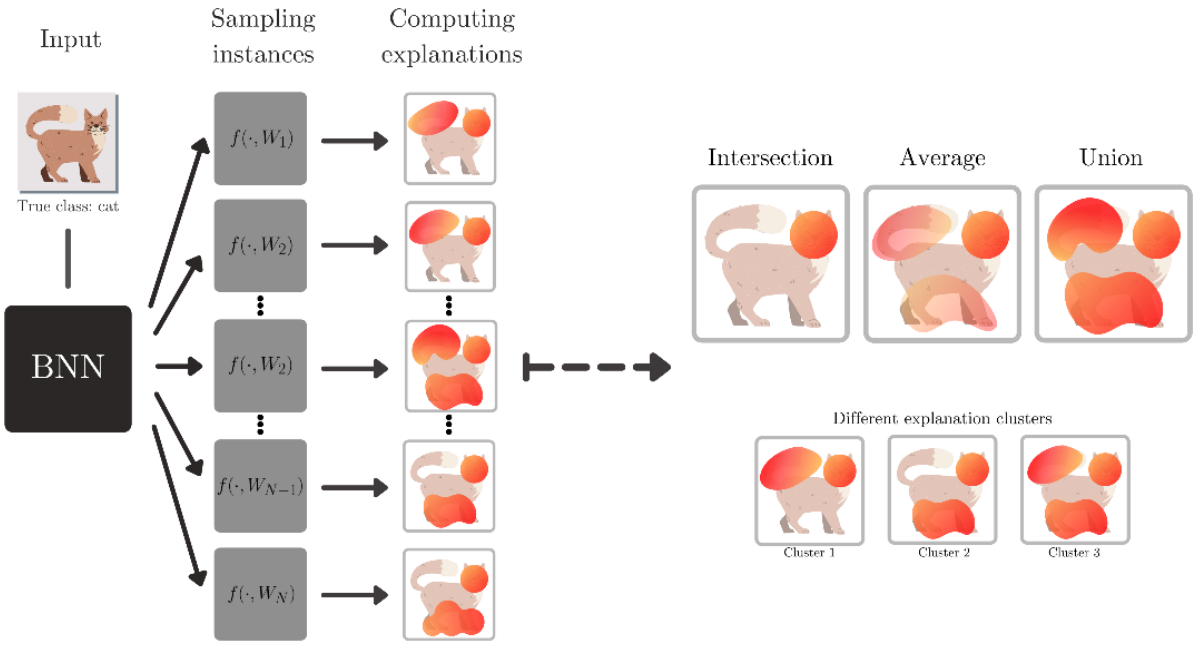


Figure 3.7: Working of the pipeline described in Explaining Bayesian Neural Network (BNN) approach. Explanation combination strategies such as intersection, average and union have been discussed in this research work. Image taken from [19] .

research that are used to map the explanation distribution to concise heatmaps. Figure 3.7 provides an overview of these strategies. The *union* strategy provides a global overview of the learned features whereas the *intersection* explanations provide a relatively conservative explanation. This approach however, only focuses on the combination of LRP with Monte Carlo Dropout and Deep Ensembles to model the uncertainty associated with the explanation. As is already discussed in Chapter 2, additional uncertainty estimation methods have been developed (other than Monte Carlo Dropout and Deep Ensembles) which could be combined with explanation methods to model the uncertainty in explanations.

Monte Carlo Guided Backpropagation

Wickstrøm et al. [93] proposed an approach that uses Monte Carlo Dropout [23] in conjunction with Guided Backpropagation (GBP) [75] in order to ascertain the uncertainties in the importance of input features. A brief introduction to this method has been provided in Chapter 1. The key idea in this approach is to extract sample predictions using MC Dropout and apply the explanation methods (in this case, GBP) to those predictions in order to generate a distribution of explanation heatmaps. From this distribution of explanations, the standard deviation heatmap is calculated to determine the confidence in importance of highlighted features.

The authors in this paper utilise color scheme to highlight the uncertainty in the importance of the identified features. This color scheme is depicted in Figure 3.8c where the red pixels mean that the

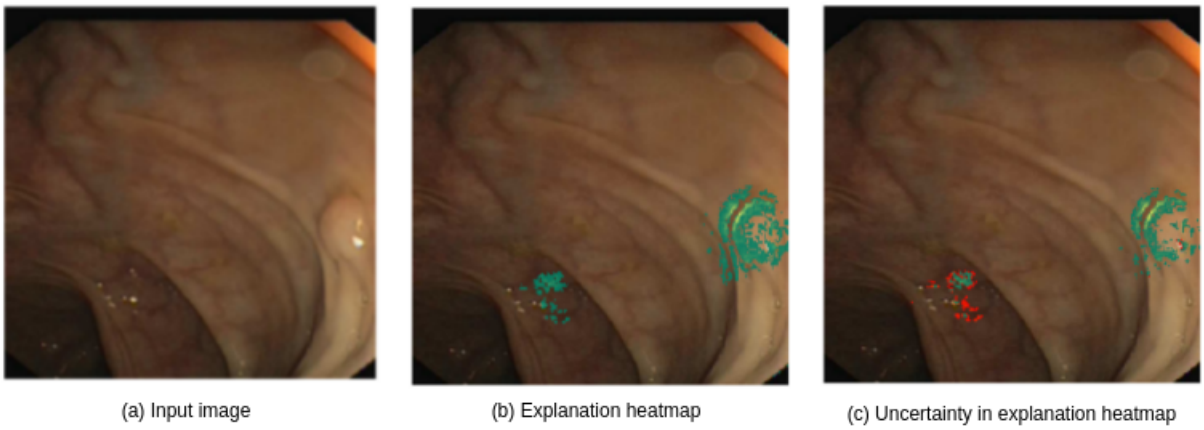


Figure 3.8: Visualizations of uncertainty in importance of input features [93]. The green pixels in (b) denote the features responsible for the model prediction. The green pixels in (c) denotes those features/pixels for which the model is highly certain regarding their importance whereas, the red pixels in (c) highlight those image regions for which the model is highly uncertain regarding their importance in the model prediction process.

uncertainty associated with the importance of these pixels is high. Conversely, the green pixels show the region where the uncertainty associated with the importance of input feature is low. It can be seen from Figure 3.8b and Figure 3.8c that the smaller contour of pixels has been erroneously attributed to be important to the model prediction as the uncertainty associated with its importance is high (and hence denoted in red in (c)). It should be noted that the threshold for determining the acceptable uncertainty in feature importance has not been disclosed in the paper.

From the discussions conducted in this chapter till now, it can be concluded that one of the potential solution to model the uncertainty in the explanation for neural networks is to combine the uncertainty estimation methods with explanation methods. Table 3.2 serves as a lookup for identifying the combinations of uncertainty estimation methods and explanation methods that have implemented and the corresponding research in which they have been discussed. The rows map to the uncertainty estimation method and the columns refer to the explanation method used in the research. The cell value in the table refers to the research paper itself. From this analysis, we can systematically identify which combinations have been explored and which potential combinations can be investigated.

Table 3.3 provides an overview of the approaches that have been discussed in this section. The columns *Explanation method* and *Uncertainty method* list the methods explored in conjunction by the paper.

3.3 Summary

From Table 3.2 and Table 3.3, it can be deduced that of the methods focusing on uncertainty aspect in explanation of neural network, most of the methods are paired with explanation methods such as LIME and SHAP. As we have discussed previously, LIME and SHAP are still under development, suffer from computational complexity, and are prone to misinterpretations. These explanation methods also

Explanation Method → Uncertainty Estimation Method ↓	**LIME**	**SHAP**	**LRP**	**GBP**
Bayesian Weighted Least Square	[72]	[72]	-	-
Repeated generation of explanation to analyse the variation	[99]	-	-	-
Deep Ensembles	-	-	[19]	-
Monte Carlo Dropout	-	-	[18], [19]	[93]

Table 3.2: A summary of the combinations of uncertainty estimation and explanation methods implemented by contemporary research works. The value in the cells refer to the paper discussing that particular combination. Hyphens indicate the combinations that have yet not been explored/investigated.

do not use gradients that are critical aspect of neural networks. Besides not many of the research works discussed above have attempted to analyse the uncertainty in explanation for visual and non-visual data simultaneously. Hence, in this thesis we propose to explore additional combinations of uncertainty estimation and explanation methods that could be used to ascertain the uncertainty in explanation of neural networks.

Paper	Explanation Method	Uncertainty Estimation Method	Notes
Reliable Post-hoc explanations [72]	LIME [55], SHAP [44]	Bayesian weighted least squares	Modifies the kernel of standard LIME and SHAP such that the generated explanations are able to model the uncertainty in the feature importances.
Understanding uncertainty in LIME [99]	LIME	Repeated generation of explanation to analyse the variation in the feature importance map	Investigates the sources of uncertainty in LIME namely, variation in explanation of different inputs, variation in explanation of a single input, randomness introduced during sampling procedure and randomness owing to changing the sampling proximity.
Quantifying uncertainties in Explaining Neural Networks [18]	LRP [13]	Monte Carlo Dropout [23]	Proposes a variant of LRP that explains BNNs by yielding a distribution of explanations. This provides the freedom to select cautious or risky explanations as opposed to the standard LRP.
Explaining BNN [19]	LRP	Ensembles [40], Monte Carlo Dropout	Translates the uncertainty information of the prediction of a BNN into the uncertainty in importance of the input feature. Also, introduces heatmap combinations strategies to concisely represent the distribution of explanation.
Uncertainty and Interpretability in CNN for semantic segmentation of colorectal polyps [93]	GBP [75]	Monte Carlo Dropout	Estimates uncertainty in input feature importance by combining Monte Carlo Dropout with GBP. Generates a heatmap highlighting the regions of (un)certainty in input feature importance.

Table 3.3: An overview of the methods that analyse the uncertainty in the explanation as discussed in the text.

4

Methodology

The previous chapter provided a literature review of the methods developed to analyse uncertainty in the explanation of neural network and the concepts described there would serve as a basis for the subsequent discussions. It was concluded that a potential solution to model uncertainty in explanation is to combine the uncertainty estimation methods with explanation methods. This chapter discusses the methodology adopted to achieve the same in this thesis. An overview of this methodology is already presented in Figure 1.2 and its working will be discussed in detail after incorporating the findings of previous chapters. We begin with analysing the methods (both uncertainty estimation and explanation) that could potentially be used to generate combinations.

4.1 Selection of Methods

4.1.1 Uncertainty Estimation methods

Amongst the methods discussed in Chapter 2, we use a total of 4 uncertainty estimation approaches to combine with explanation methods in this thesis. The uncertainty estimation methods selected for the implementation and the reasons supporting their selection are as follows:

- Monte Carlo Dropout (Approximate Bayesian)
  This method is considered by majority of the works (analysed in literature review) in order to estimate the uncertainty in context of neural networks. It has also been combined with Guided Backpropagation (GBP) [75] in [93] to ascertain the uncertainty in explanation of a neural network. Hence, selecting this method can serve as a reference.

- Monte Carlo DropConnect (Approximate Bayesian)
  According to the the authors of Monte Carlo DropConnect [48], this method performs better as compared to Monte Carlo Dropout. This serves as a natural motivation to test the possibility of combining this approach with explanation methods.

- Flipout (Approximate Bayesian)
  This method is efficient, in that it samples the pseudo-independent weight perturbation and thus reduces the correlation of the gradients inside a mini-batch [45], [84], [91]. Also, as per the claims

of [78], this method is an improvement over the Bayes by Backprop [16] approach. Hence, it is imperative to test this method as an alternative to the already selected approximate Bayesian methods.

- Deep Ensembles (Non-Bayesian)
  As previously discussed, Deep Ensembles belong to the non-Bayesian category of uncertainty approaches. By selecting this method, we can test the possibilities of combining non-Bayesian uncertainty estimation approaches with explanation methods for neural networks as well.

4.1.2 Explanation methods

From the explanation methods discussed in the last chapter, we select a total of 3 methods to combine with the uncertainty estimation methods in this thesis. Following are the selected explanation methods along with the justification for their selection:

- Integrated Gradients (IG)
  With the concept of baseline and interpolated images, this method utilises the information produced by incremental addition of features to identify relevant pixels in the original image. Also, this method does not require any modification to the network architecture. Hence, this could be implemented for testing the proposed approach.

- Guided Backpropagation (GBP)
  This method is a viable candidate for implementation as it generates explanation heatmaps that are relatively less noisy when compared to heatmaps from other methods such as CAM, Gradients, DeconvNets. Also, the algorithmic complexity of this approach is low and it is straightforward to implement.

- Local Interpretable Model-Agnostic Explanations (LIME)
  From the properties of explanation method discussed in Section 2.3.1, LIME exhibits high portability as this is model-agnostic, hence applicable to a variety of models. Along with this, LIME has a high expressive power as this method can be used to generate explanations for a wide variety of input modalities such as images, text, structured data. Hence, this method could be used to test the robustness of our proposed approach.

4.2 Proposed Approach

4.2.1 Description of Pipeline

The discussions concluded in the previous chapter and further analysis of the published research work provided insights into the combinations that have already been explored. This provided an inspiration to investigate additional such combinations. From the methods selected in Section 4.1.1 and Section 4.1.2, we can identify these additional combinations in order to systematically model the uncertainty

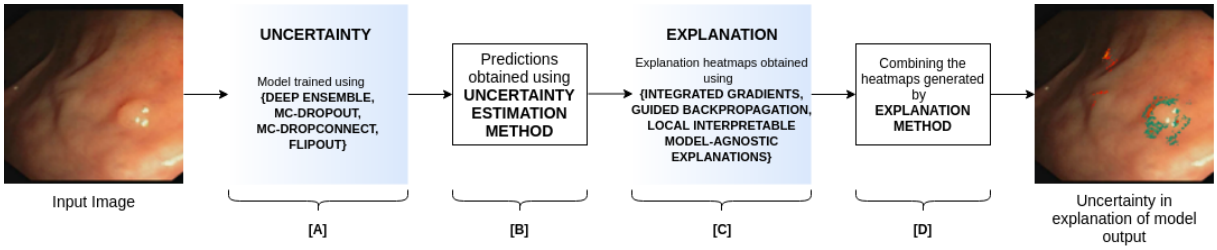


Figure 4.1: Overview of the proposed approach described in Figure 1.2 along with the findings incorporated from the literature analysis. In this image, the uncertainty estimation ([A]) and explanation methods ([C]) cells have been updated after an analysis of the previous research works conducted in Chapter 2 and Chapter 3.

in explanation. The justification for the selection of these methods has been provided in the respective sections. The methods identified for further analysis are presented as follows:

- Uncertainty estimation methods:
  {Deep Ensembles, Monte Carlo Dropout, Monte Carlo DropConnect, Flipout}

- Explanation methods:
  {Integrated Gradients, Guided Backpropagation, Local Interpretable Model-Agnostic Explanations}

Figure 4.1 represents detailed version of the Figure 1.2 (Figure 1.2 is a schematic representation of the proposed pipeline). The aforementioned details have been incorporated in the cells [A] and [C]. After analysing the contemporary research works, we identified additional uncertainty estimation methods and explanation methods that could possibly be used to generate new combinations. The subsequent text outlines the working of the pipeline depicted in Figure 4.1:

Convention used for the description of pipeline

In order to describe the working of the pipeline, following convention has been adopted:
I: Input
M: Stochastic model
$M^{'} = \{m_1, m_2, \ldots, m_n\}$: set of n model instances generated by sampling from M
$P = \{p_1, p_2, \ldots, p_n\}$: distribution of n predictions generated by passing the input through the model instances belonging to $M^{'}$
$E = \{e_1, e_2, \ldots, e_n\}$: distribution of n explanations generated by application of explanation method on predictions belonging to P
Note: In case of Deep Ensembles, the $M^{'}$ will contain n ensemble components. These will be equivalent to the n model instances (belonging to $M^{'}$) generated by sampling the stochastic model (M) obtained using Monte Carlo Dropout, Monte Carlo DropConnect and Flipout.

Stage A: Training a neural network using uncertainty estimation method

In this stage, a neural network is trained using an uncertainty estimation method. It is in this component that the stochasticity is introduced in the pipeline. As can be seen in Figure 4.1[A], a number of uncertainty estimation methods are available. The output of this stage is a stochastic model (M), whose weights can be sampled during inference to generate different instances of the model. In case of Deep Ensembles, the output of this stage will be a set of n trained model instances $\{m_1, m_2, \ldots, m_n\}$ where n denotes the number of components in the ensemble.

Stage B: Generating a distribution of prediction using the stochastic model

Using the model instances sampled from stochastic model (M) generated in the previous stage, a total of n predictions are obtained for a given input (I). This is done by forward propagating the input n times through M. As M is stochastic, each forward pass results in a different instance of the model and for n forward passes, the set of models that are generated can be represented as $M^{'} = \{m_1, m_2, \ldots, m_n\}$ and the predictions can be represented as $P = \{p_1, p_2, \ldots, p_n\}$. In case of Deep Ensembles, each component yields one prediction hence, for n components we obtain n predictions. From this stage on, Deep Ensembles do not require special attention as the next steps process the prediction distribution which has already been obtained. The working of this stage can be mathematically represented as:

$$M(I) \xrightarrow[\text{passes}]{\text{n forward}} M^{'}(I) = \{m_1(I), m_2(I), \ldots, m_n(I)\} = \underbrace{\{p_1, p_2, \ldots, p_n\}}_{\text{P}} \tag{4.1}$$

Stage C: Generating a distribution of explanation using the distribution of prediction

In this stage, the distribution over prediction is used to generate a distribution over explanation. This is achieved by applying an explanation method from Figure 4.1[C] to P. This operation translates to application of the explanation method to each prediction in P producing one explanation (e_i) per prediction (p_i). The resulting distribution (E) can be used to model the uncertainty in the explanation. Equation 4.2 presents the working of this stage:

$$\mathit{explanation\_method}(\underbrace{\{p_1, p_2, \ldots, p_n\}}_{\text{P}}) = \underbrace{\{e_1, e_2, \ldots, e_n\}}_{\text{E}} \tag{4.2}$$

Stage D: Creating concise representations of distribution of explanations

From the distribution of explanation, concise representations are generated in order to capture the essence of uncertainty. For any mathematical distribution, the mean (μ) and standard deviation (σ) can be computed to analyse its characteristics. These two quantities are computed for the explanation distribution (E) in this stage. In addition to these, coefficient of variation (CV) of E is also computed. The mathematical representation of these quantities for a distribution denoted by $E = \{e_1, e_2, \ldots, e_n\}$ is given as:

- **Explanation Mean**: This provides insights into the features/pixels where the different explanations within the distribution "agree" upon. This could be useful to know which features were most commonly considered by the model in making the prediction. The explanation mean is computed as follows:

$$explanation\ \mu = \frac{e_1 + e_2 + \ldots + e_{n-1} + e_n}{n} \tag{4.3}$$

- **Explanation Standard Deviation**: This provides information about the features/pixels where the different explanations of the distribution "disagree". Those features/pixels will have higher standard deviation where explanations vary drastically. The explanation standard deviation can be computed as:

$$explanation\ \sigma = \sqrt{\frac{\sum_{i=1}^{n}(e_i - \mu)^2}{n}} \tag{4.4}$$

- **Explanation Coefficient of Variation (CV)**: It is defined as the ratio of standard deviation (σ) to mean (μ) of the distribution. It provides a measure of dispersion/spread of the distribution under consideration and could potentially be used as an auxiliary to analyse mean and standard deviation explanations simultaneously. The coefficient of variation (CV) is given as:

$$explanation\ CV = \frac{\sigma}{\mu} \tag{4.5}$$

4.2.2 Identification of Combinations

In the subsequent text, the possible combinations of the identified uncertainty estimation and explanation methods are listed. Rewriting the Equation 1.1 by incorporating the knowledge from previous discussions as:

$$\underbrace{\left\{\begin{array}{c} Deep\ Ensembles \\ Monte\ Carlo\ Dropout \\ Monte\ Carlo\ DropConnect \\ Flipout \end{array}\right\}}_{\text{Uncertainty methods}} \times \underbrace{\left\{\begin{array}{c} IG \\ GBP \\ LIME \end{array}\right\}^{\top}}_{\text{Explanation methods}} \tag{4.6}$$

Table 4.1 is obtained by expanding the matrix product in Equation 4.6 and lists all possible combinations that can be tested from the selected methods. As will be discussed in detail in the next chapter, these possible combinations will be tested on two different tasks namely image classification and numerical regression. The details of implementation and the corresponding experiments that are conducted in this thesis have been discussed next.

UQ	**EX**
Deep Ensembles	IG
Deep Ensembles	GBP
Deep Ensembles	LIME
MC Dropout	IG
MC Dropout	GBP
MC Dropout	LIME
MC DropConnect	IG
MC DropConnect	GBP
MC DropConnect	LIME
Flipout	IG
Flipout	GBP
Flipout	LIME

Table 4.1: An overview of the possible combination of uncertainty estimation and explanation methods from the search space defined in Equation 4.6. Here, UQ stands for uncertainty quantification/estimation methods and EX stands for explanation methods.

5

Experimental Setup

The previous chapter discussed the proposed approach to ascertain the uncertainty in the explanation of neural networks developed for this thesis. In this chapter, the dataset specifications, the network architectures, and the experiments conducted using the proposed approach have been presented. These discussions have been divided into sections for systematic analysis.

5.1 Datasets

In this thesis, we test the proposed pipeline on an image classification and a numerical regression task. The rationale to test on two tasks is to verify the robustness of the pipeline. In this section, we introduce and discuss the specifications of the datasets used for the image classification and numerical regression task.

Canadian Institute for Advanced Research (CIFAR-10)

The CIFAR-10 [38] dataset comprises of 60000 RGB images of size 32×32×3 belonging to 10 classes and is used for the task of image classification. The train set contains 50000 images and the test set contains

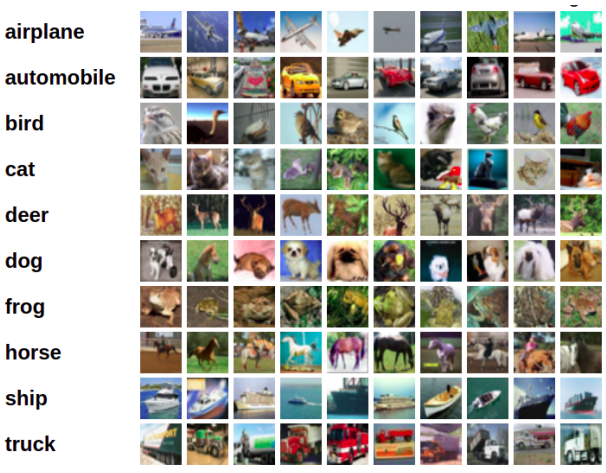


Figure 5.1: Sample images from the CIFAR-10 dataset. Image taken from [38].

10000 images. A custom validation set is created from these 10000 test images by randomly sampling 9500 images and using the remaining 500 images as the custom test set. A salient feature of the images in this dataset is that the classes are mutually exclusive. Figure 5.1 provides an overview of the classes in the CIFAR-10 dataset along with a sample collection of images.

Facial Expression Recognition Plus (FER+)

Facial Expression Recognition Plus (FER+) [14] is an enhanced version of the Facial Expression Recognition (FER) [26] dataset. As the name suggests, this dataset contains images of facial expressions and the objective is to classify the given facial expression into one of the 8 classes. In the FER dataset, a single label is assigned to the image whereas in the FER+ dataset, the final label is determined by conducting a majority voting of 10 crowd source labels per image. This difference can be observed in Figure 5.2 that contains sample images and the corresponding labels from both these datasets. From this figure, it can be seen that this is a difficult dataset to work with [46] and according to [26], humans only have an accuracy of 65±5 % on this dataset. This is natural as facial expressions are a subjective quantity to classify [86]. It can be understood by a closer analysis of the first image in Figure 5.2, where a single annotator labeled the expression as that of "surprise" whereas a consensus of 10 annotators agreed upon the expression as being that of "happiness". The dataset contains a total of 28559 train images, 3579 validation images and 3573 test images. The size of the gray-scale images in this dataset is 48×48×1 pixels. Using this dataset could provide interesting insights into the uncertainty analysis of explanations and hence this dataset is selected.

Figure 5.2: Images in the FER [26] and FER+ [14] dataset. The labels on top (in the individual image caption) belong to FER and those in the bottom correspond to FER+. The labels of FER+ are obtained after majority voting of the crowd sourcing labels for individual images. Image taken from [26].

California Housing Dataset

In order to test the robustness of the proposed combinations of uncertainty estimation and explanation methods explored in this thesis, a regression task has also been investigated. The dataset of choice for

this particular task is the California Housing dataset [53]. The objective is to predict the housing price given a set of features for a particular house. The input is an eight dimension vector for any instance and the output is a scalar which is the price of the house. The dataset is split such that train set has 12750 instances, the validation set has 4250 instances and the test set has 3000 instances. Table 5.1 provides the names of the input features along with their description and the range of possible input values for the dataset.

Column (Feature)	**Description**	**Range**
longitude	how far west a house is	(-180° to 180°)
latitude	how far north a house is	(-90° to 90°)
housingMedianAge	median house age within a block	[1.0, 52.0]
totalRooms	total number of rooms in a block	[2.0, 37937.0]
totalBedrooms	total number of bedrooms in a block	[1.0, 6445.0]
population	total number of people living in a block	[3.0, 35682.0]
households	total number of households (group of people living in a home unit) for a block	[1.0, 6082.0]
medianIncome	median income for households in a block of houses (in tens of thousands of US$)	[0.5, 15.0]
medianHouseValue	median house value for households within a block (in US$)	[14999.0, 500001.0]

Table 5.1: Description of the input features in the California Housing dataset taken from [1]. A range of possible feature values have been provided for reference as well.

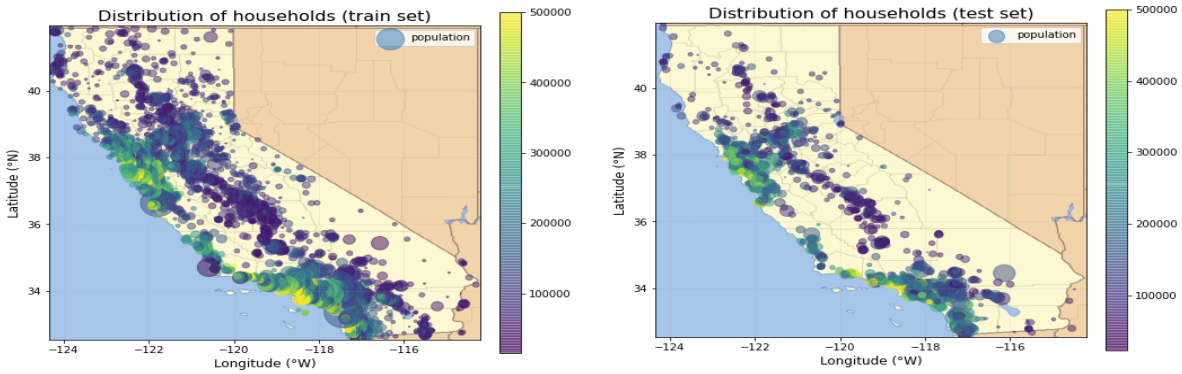
Table 5.1 provides insights into the feature description of the dataset. Figure 5.3 depicts a sample of inputs from the dataset that contains the feature values in a tabular format.

longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
-114.310000	34.190000	15.000000	5612.000000	1283.000000	1015.000000	472.000000	1.493600	66900.000000
-114.470000	34.400000	19.000000	7650.000000	1901.000000	1129.000000	463.000000	1.820000	80100.000000
-114.560000	33.690000	17.000000	720.000000	174.000000	333.000000	117.000000	1.650900	85700.000000
-114.570000	33.640000	14.000000	1501.000000	337.000000	515.000000	226.000000	3.191700	73400.000000
-114.570000	33.570000	20.000000	1454.000000	326.000000	624.000000	262.000000	1.925000	65500.000000

Figure 5.3: Samples of feature input values taken from California Housing [53] dataset.

In order to have a pictorial representation of the data, three features are plotted on a map of the state of California. Figure 5.4 provides a visualization of the population distribution in the state of California as a variation of the longitude and latitude for both the train and test set. It can be seen from these visualizations that the population centers are generally nearer to the coast as compared to the inland

regions. The image depicted in Figure 5.4 has been reproduced using code from GitHub[1]. Many such visualizations can be generated by combining other features however, we restrict ourselves as our objective is to analyse uncertainty in the explanation.



(a) Train set

(b) Test set

Figure 5.4: Visualizing variation of population against the latitude and the longitude.

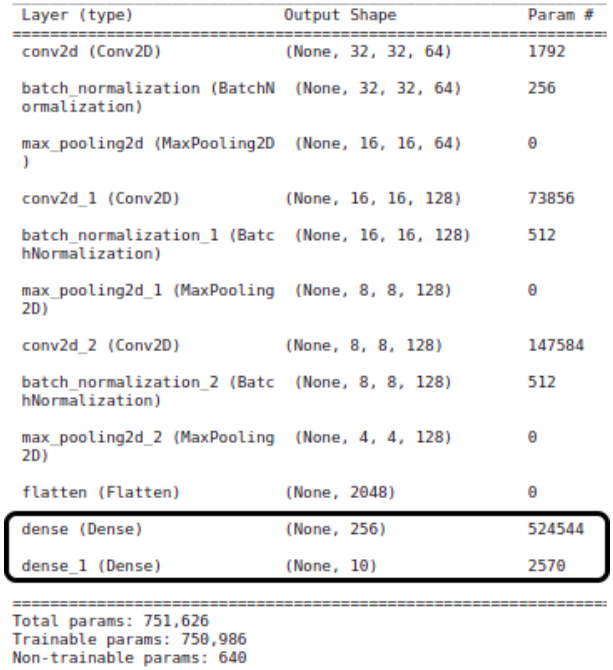
5.2 Network architectures

To test the proposed pipeline on two tasks, we work with two neural network architectures. This choice has been made based on the computational demands of the tasks to be solved. Image classification tasks usually benefit from the use of convolutional neural networks whereas numerical regression tasks can be solved using simple perceptron networks. Hence, for the image classification task on CIFAR-10 and FER+ dataset, we select the miniVGG (a variant of VGG [68]) network used by Toro et al. [85] to train on FER+. For the task of regression on California Housing dataset, a Multi-Layer Perceptron (MLP) is used. The architecture of the MLP is designed following insights obtained after conducting a hyperparameter search. The details of the hyperparameter search are provided in Appendix A.

5.2.1 Architecture of MiniVGG

In this section, we discuss the structures of the neural networks trained in this thesis. First, we analyse the neural network used for image classification task. The architecture of miniVGG is depicted in Figure 5.5. For training this network on CIFAR-10 and FER+, the final dense layer should have 10 and 8 units respectively as these are the total number of different classes available in these datasets. The network consists of 3 basic blocks (each containing a *convolution*, *batch normalization* and *pooling* layer) followed by *flatten* and *dense* layers. As stated earlier, this network has been used by Toro et al. [85] in their work on the FER+ dataset. The code to incorporate the uncertainty in the training of neural networks has been taken from [81].

[1] https://github.com/sonarsushant/California-House-Price-Prediction/blob/master/EDA%20and%20Data%20Cleaning.ipynb



Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 64)	1792
batch_normalization (BatchNormalization)	(None, 32, 32, 64)	256
max_pooling2d (MaxPooling2D)	(None, 16, 16, 64)	0
conv2d_1 (Conv2D)	(None, 16, 16, 128)	73856
batch_normalization_1 (BatchNormalization)	(None, 16, 16, 128)	512
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 128)	0
conv2d_2 (Conv2D)	(None, 8, 8, 128)	147584
batch_normalization_2 (BatchNormalization)	(None, 8, 8, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 128)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 256)	524544
dense_1 (Dense)	(None, 10)	2570

Total params: 751,626
Trainable params: 750,986
Non-trainable params: 640

Figure 5.5: Architecture of the miniVGG network inspired from [85]. The modifications to this network for incorporating uncertainty are applied to the part highlighted by the black box.

Modifying miniVGG to incorporate uncertainties

Next, we discuss the process of incorporating the uncertainty in the neural network according to our proposed approach. It should be noted in Figure 5.5 that only the layers encapsulated in the box are modified depending on the uncertainty estimation method being implemented and the rest of the network is left unchanged.

- **Deep Ensemble:** As discussed previously, Deep Ensembles require training multiple copies of the model with random initialization. Hence, no structural changes are required to incorporate uncertainty using Deep Ensembles and only multiple copies of this network (each with random initialization) need to be trained on the dataset. Figure 5.6a depicts the same structure of the network as shown in Figure 5.5.
- **MC Dropout:** In order to modify the network for this case, the *stochastic_dropout* layer has been inserted after the *flatten* and *dense* layers. These changes have been depicted in Figure 5.6b.
- **MC DropConnect:** In this case, the final *dense* layers have been replaced with *dropconnect_dense* layers in order to induce uncertainties in the prediction. These changes can be seen in Figure 5.6c.
- **Flipout:** For implementing the modification in this case, the last *dense* layers after the *flatten* have been replaced with *flipout_dense* layers. These modifications are shown in Figure 5.6d.

flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 256)	524544
dense_1 (Dense)	(None, 10)	2570

(a) Deep Ensemble

flatten (Flatten)	(None, 2048)	0
stochastic_dropout (StochasticDropout)	(None, 2048)	0
dense (Dense)	(None, 256)	524544
stochastic_dropout_1 (StochasticDropout)	(None, 256)	0
dense_1 (Dense)	(None, 10)	2570

(b) MC Dropout

flatten (Flatten)	(None, 2048)	0
drop_connect_dense (DropConnectDense)	(None, 256)	256
drop_connect_dense_1 (DropConnectDense)	(None, 10)	10

(c) MC DropConnect

flatten (Flatten)	(None, 2048)	0
flipout_dense (FlipoutDense)	(None, 256)	1049088
flipout_dense_1 (FlipoutDense)	(None, 10)	5140

(d) Flipout

Figure 5.6: Changes made to the base miniVGG architecture to incorporate uncertainties.

5.2.2 Architecture of MLP

For training the numerical regression task, we use a Multi-Layer Perceptron (MLP). It is a fully connected neural network consisting of a number of dense layers. As this is a regression task, the final layer is made up of a single neuron. The architecture of the MLP is obtained by conducting a hyperparameter search as discussed earlier. The details of the hyperparameter search are discussed in Appendix A. It should be noted that the result of the hyperparameter search provides a guideline to build the model.

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 8)	72
dense_1 (Dense)	(None, 8)	72
dense_2 (Dense)	(None, 8)	72
dense_3 (Dense)	(None, 1)	9

Total params: 225
Trainable params: 225
Non-trainable params: 0

Figure 5.7: Architecture of the MLP network. This structure is designed using the findings of a hyperparameter search. The details of the hyperparameter search are provided in Appendix A.

Modifying MLP to incorporate uncertainties

As discussed previously for the case of miniVGG, we modify the basic MLP shown in Figure 5.7. These modifications are shown in Figure 5.8.

- **Deep Ensemble:** Similar to the case of miniVGG, here the structure of the MLP remains the same. This structure is shown in Figure 5.8a. In order to induce a distribution of prediction, we

train multiple copies of this model from random initialization.

- **MC Dropout:** Figure 5.8b depicts the modifications required to induce uncertainty in the MLP using MC Dropout. The *stochastic_dropout* layers are added between the *dense* layers which help the model in adopting stochastic behaviour.

- **MC DropConnect:** In this case, the last *dense* layer of the model is replaced with *dropconnect_dense* layer. This modification helps the model in generating stochastic output. These changes are depicted in Figure 5.8c.

- **Flipout:** The final *dense* layer is replaced in order to modify the network. The replacement layer is aptly named as the *flipout_dense* and the change is shown in Figure 5.8d.

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 8)	72
dense_1 (Dense)	(None, 8)	72
dense_2 (Dense)	(None, 8)	72
dense_3 (Dense)	(None, 1)	9

Total params: 225
Trainable params: 225
Non-trainable params: 0

(a) Deep Ensemble

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 8)	72
dense_1 (Dense)	(None, 8)	72
stochastic_dropout (StochasticDropout)	(None, 8)	0
dense_2 (Dense)	(None, 8)	72
stochastic_dropout_1 (StochasticDropout)	(None, 8)	0
dense_3 (Dense)	(None, 1)	9

Total params: 225
Trainable params: 225
Non-trainable params: 0

(b) MC Dropout

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 8)	72
dense_1 (Dense)	(None, 8)	72
dense_2 (Dense)	(None, 8)	72
drop_connect_dense (DropConnectDense)	(None, 1)	9

Total params: 225
Trainable params: 225
Non-trainable params: 0

(c) MC DropConnect

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 8)	72
dense_1 (Dense)	(None, 8)	72
dense_2 (Dense)	(None, 8)	72
flipout_dense (FlipoutDense)	(None, 1)	18

Total params: 234
Trainable params: 234
Non-trainable params: 0

(d) Flipout

Figure 5.8: Changes made to the base MLP architecture to incorporate uncertainties.

The training configurations of the network modifications discussed above are provided in Appendix B along with the performance plots and hyperparameter settings. During inference phase, passing an input multiple times through these networks yields slightly different outputs (as the networks assume stochastic behaviour and each forward pass corresponds to sampling the model weights from a posterior distribution) which can then be used to create a distribution of prediction.

5.3 Experiments

This section provides the details of the experiments conducted in this thesis. The text has been divided into subsections for a systematic analysis. It should be noted that the concepts discussed in Chapter 4 would be used in the subsequent discussions.

5.3.1 Experiment 1: Uncertainty of Explanation

In this experiment, we aim to achieve the primary goal of this thesis. We attempt to combine the uncertainty estimation methods with the explanation methods in order to analyse the uncertainty in the explanation of neural networks. To achieve this objective, the pipeline proposed in Chapter 4 is implemented. The task we choose for this experiment is that of image classification. The datasets of choice for this task are CIFAR-10 and FER+. Table 5.2 lists the combinations that we test in Experiment 1 for the task of image classification:

Dataset	UQ	EX
CIFAR-10	Deep Ensembles	IG
CIFAR-10	Deep Ensemble	GBP
CIFAR-10	MC Dropout	IG
CIFAR-10	MC Dropout	GBP
CIFAR-10	MC DropConnect	IG
CIFAR-10	MC DropConnect	GBP
CIFAR-10	Flipout	IG
CIFAR-10	Flipout	GBP

(a) Combinations tested on CIFAR-10.

Dataset	UQ	EX
FER+	Deep Ensembles	IG
FER+	Deep Ensemble	GBP
FER+	MC Dropout	IG
FER+	MC Dropout	GBP
FER+	MC DropConnect	IG
FER+	MC DropConnect	GBP
FER+	Flipout	IG
FER+	Flipout	GBP

(b) Combinations tested on FER+.

Table 5.2: An overview of the combinations of uncertainty estimation and explanation methods tested for the image classification task.

It should be noted that Table 5.2 contains the dataset names and all the possible combinations of the selected uncertainty estimation and the explanation methods. With Table 5.2, now we have a clearer understanding of the combinations and datasets on which to test the proposed approach. From the selected explanation methods, LIME has not been used for image classification task as it is computationally expensive as compared to IG and GBP. The expected output of this experiment is a concise representation of the explanation distribution in terms of the mean and the standard deviation. An auxiliary, namely the coefficient of variation has also been visualized in this experiment in order to analyse the mean and the standard deviation heatmap simultaneously. The details of the model trainings and the corresponding configurations conducted in this experiment have been discussed in Appendix B. For generating the explanations using Integrated Gradients (IG), the number of interpolation steps is set to $n = 50$ as primarily

the authors of [77] suggest that the number of steps should lie between 20 and 300 and additionally $n = 50$ is used by the sample implementation provided in the TensorFlow tutorials [2].

Post-processing the generated explanation for visualization

As the explanation is generated by computing the gradient of predicted class score with respect to the input image, the resulting values can be positive, negative and have any magnitude. In order to visualize these in the form of an explanation (heatmap), the values are processed to be in the range of $[0, 1]$. The range of values in a particular explanation can be referred to from the color bar adjacent to it.

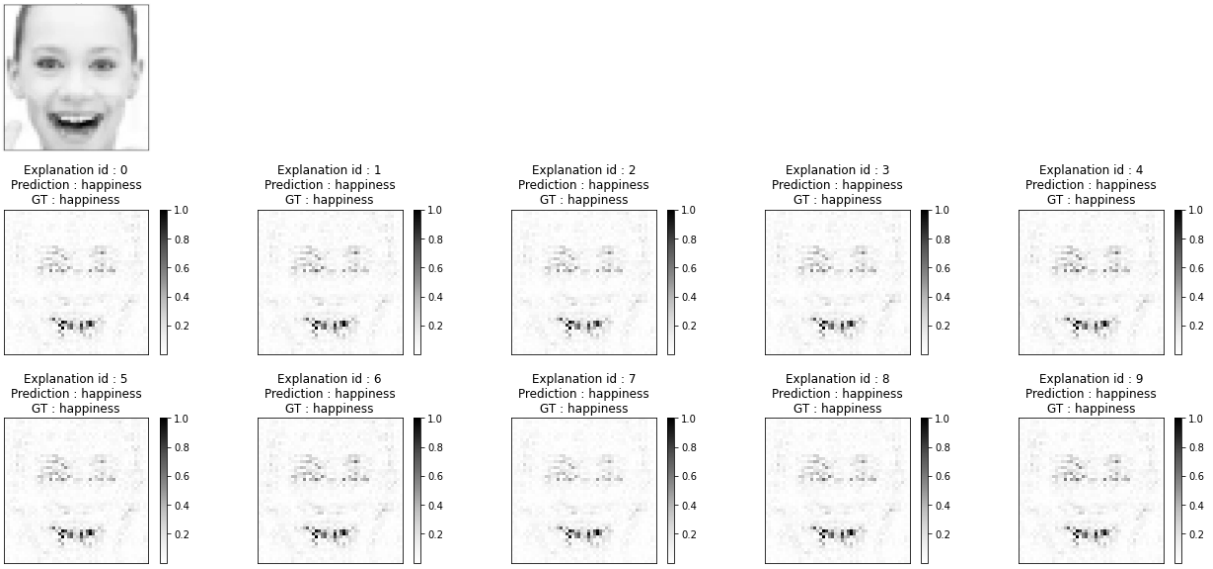


Figure 5.9: Visualizing an input image (first row) and its corresponding explanation distribution (second and third row). The individual instance titles provide information about the instance id, the ground truth label and the network prediction.

Generating concise representation of distribution of explanation

As we have a distribution of explanation, we now focus on generating a concise representation of the same. For this purpose, the formula for computation of the mean, the standard deviation and the coefficient of variation (CV) of the distribution have been provided in Chapter 4. In the subsequent discussion, we propose a modification to the formula of coefficient of variation. Equation 5.1 represents this modification and is an adapted form of Equation 4.5:

$$explanation\ CV = \frac{\sigma + \epsilon}{\mu + \epsilon} \tag{5.1}$$

where σ and μ are the standard deviation and the mean of the explanation distribution respectively. ϵ is a stabilizer term added to avoid the division by zero anomaly and to tune the quality of visualizations.

The effect of changing ϵ is discussed in the subsequent text.

A description of the sample output of this experiment has been provided in this section. The first row of Figure 5.9 depicts the input image under consideration for this discussion. The corresponding explanation distribution is depicted in the subsequent 2 rows of Figure 5.9. It consists of a total of 10 explanation instances each highlighting different pixels responsible for the prediction decision. From this distribution of heatmaps, a concise representation as discussed previously is computed by calculating the mean and the standard deviation of the said distribution.

The concise representation of the explanation distribution is provided in Figure 5.10. It should be noted that mean of the explanation distribution would highlight those pixels where all the explanation instances converge. This can be also understood as the regions in input image where all the explanations have agreement. This is clearly seen in first heatmap of Figure 5.10 where the mouth of the person has been appropriately highlighted as all the explanation instances focus on that particular region. Conversely, the standard deviation highlights those regions in the input image where the explanation instances diverge. This is depicted in second heatmap in Figure 5.10. This highlights the region near the eyebrows of the person as the constituent explanation instances have variations in that region. As will be seen in the next chapter, the result of this experiment will be the mean, the standard deviation and the coefficient of variation visualization of the explanation distribution.

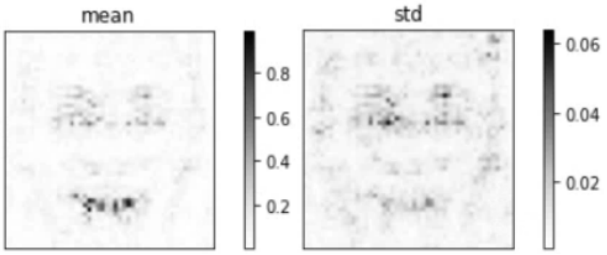


Figure 5.10: The mean and standard deviation representation of the explanation distribution depicted in Figure 5.9.

The necessity for modifications in coefficient of variation is highlighted from Figure 5.10. Many pixels of the mean and the standard deviation visualization have a value of 0, this might cause the coefficient of variation formula provided in Equation 4.5 to fail. This observation justifies the addition of stabilizer terms to both the numerator and the denominator. Figure 5.11 depicts the effect of different ϵ values on the coefficient of variation visualization. As the value of ϵ increases, the noise in the heatmap decreases. However, from Figure 5.11, it can be observed that for higher values of ϵ the improvement saturates. For this reason, we select a conservative value of $\epsilon = 1$ for depicting explanation CV.

5.3.2 Experiment 2: Pixel Deletion/Insertion

In this experiment, we attempt to quantify the quality of the heatmaps generated in previous experiment. For this purpose, we utilise two evaluation metrics namely the *pixel deletion* and *pixel insertion* proposed

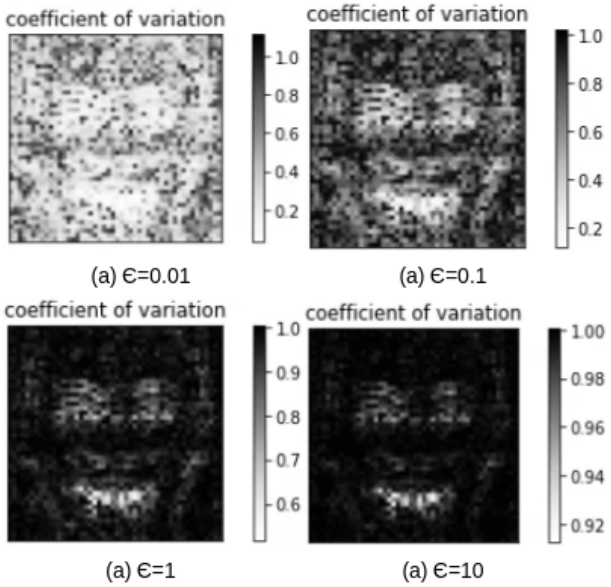


Figure 5.11: Different visualization of coefficient of variation representation obtained by changing the value of ϵ. The changes in the visualization saturate after ϵ reaches a value.

by [54] in their work. To compute the pixel deletion metric, the drop in the class score is measured as pixels are deleted from the original image. The deletion of pixels is based on their importance in the explanation heatmap of the input image. The most important pixels are deleted first and the class score is observed. The previous step is iterated till all the pixels in the input image are deleted. This results in a class score curve that is a function of the fraction of pixels deleted. The pixel deletion metric is then the area under the class score curve obtained in the previous step. A lower Area under the Curve (AUC) corresponds to a good explanation as deleting the highly relevant pixels should diminish the class score significantly in the early steps of the pixel deletion process. Conversely, to compute pixel insertion, the increase in the class score is measured as pixels are inserted in the original image. The process of adding pixels is similar to the one adopted while computing the deletion metric. To compute the pixel insertion metric the most important pixel are added first. However, in this case, a higher AUC corresponds to a good explanation as inserting the most important pixel should improve the class score significantly in the early steps of the process.

Several methods exist to delete pixels from an image such as setting the pixel value to zero, a constant gray value or cropping out the region under consideration. The authors of [54] argue that in order to obtain the best results, setting the pixels to be deleted with a constant value is the best solution. Similar to deletion of pixels from an image, many methods have been developed for pixel insertion. These range from sequentially inserting the desired pixel in a gray/blank image, or starting with a blurred copy of the original image and gradually denoising the blurred image region based on the pixels to be inserted. The authors suggest using a blurred image as a starting point to compute the pixel insertion metric. For the purpose of implementation of these metrics in this thesis, we adopt the design decisions recommended by

the authors of [54]. The following text exhibits the implementation of the aforementioned metrics on a sample image.

Figure 5.12 depicts the input image for which the deletion and insertion metric have been computed. Along with the input, the heatmap generated by GBP approach and the overlay of this explanation on the input image has also been provided. From this figure, it can be concluded that the pixels present on the face of the dog are identified as being highly relevant to the model prediction and subsequently these would be deleted/inserted first while computing the deletion and insertion metrics.

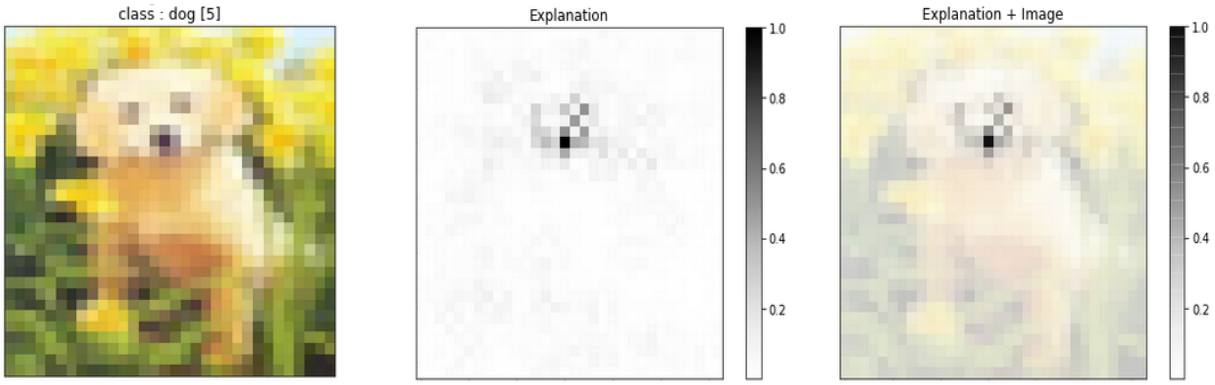


Figure 5.12: Example image to demonstrate the working of pixel deletion and insertion. The explanation of this image and a visualization of this explanation overlapped on the input image is also shown.

Computing the pixel deletion metric

As described earlier, the pixels are deleted based on their importance in the heatmap in order to compute this metric. We start with the original image and delete the important pixels in subsequent steps till all the pixels are removed from the image. This process is depicted in Figure 5.13 where the amount of deleted pixels from the original image increases from left to right. It should be noted that only a few intermediate steps are depicted here for illustration. The corresponding class score is provided along with the amount of pixels deleted. It is observed that the class score decreases significantly when the most relevant 25% pixels of original image are deleted. A plot of the class score as a function of the fraction of deleted pixels is visualized as well. This plot can be used to compute the AUC and is shown in Figure 5.15a. From this figure, the drop in the class score can be seen when the most relevant pixels (based on the explanation) are deleted from the image. The AUC for the plot is also provided for reference.

Computing the pixel insertion metric

To compute this metric, the pixels are inserted in a blurred image depending on their importance in the explanation. We start with a blurred copy of the original image and add the important pixels in subsequent steps. The noise added to blur the image is calculated by multiplying the standard deviation and adding the mean of the dataset from which the image is selected. The process of adding pixels is

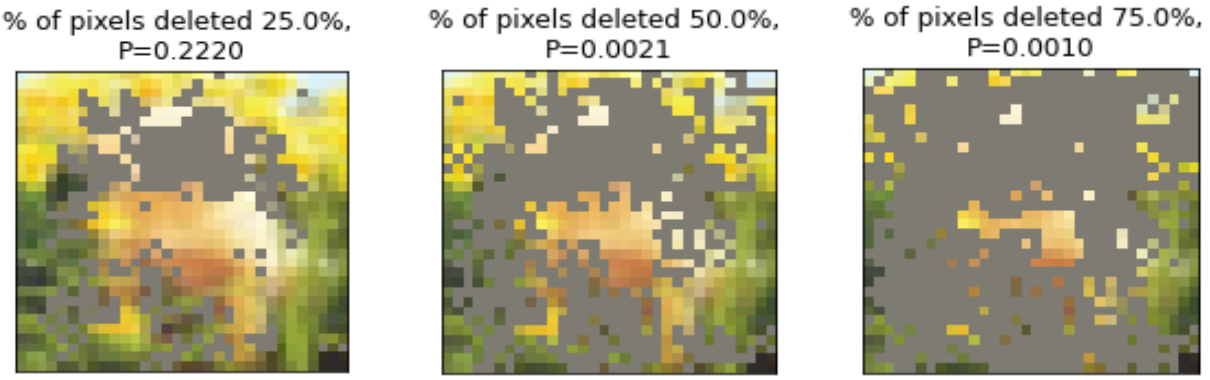


Figure 5.13: Stages of pixel deletion. The amount of pixel deleted increases from left to right.

depicted in Figure 5.14 where the amount of pixels inserted in the original (blurred) image increases from left to right. Additionally, it can be observed that the class score improves with increasing amount of inserted pixels. The class score as a function of the fraction of inserted pixels can be seen in Figure 5.15b The class score is initially low as the image is completely blurred. It gradually increases as the relevant pixels are subsequently added to the image.

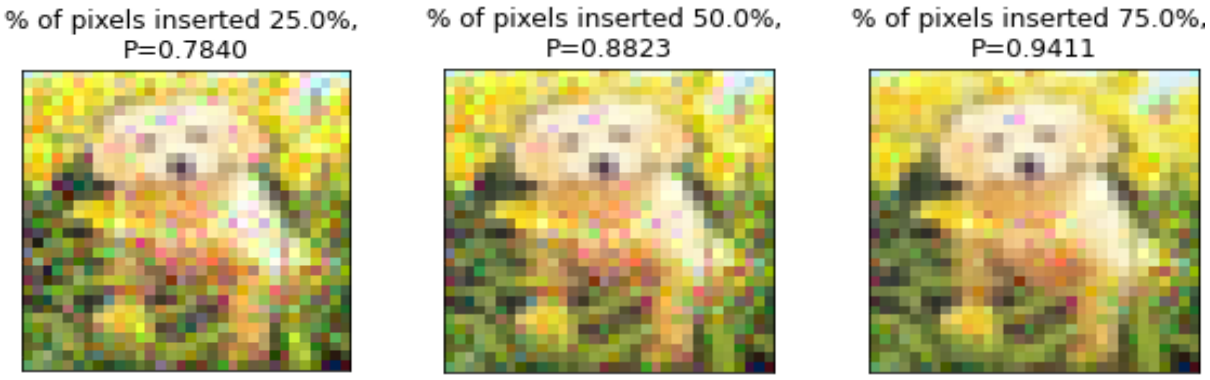
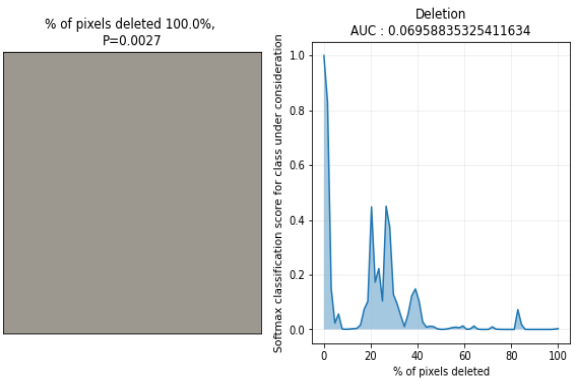
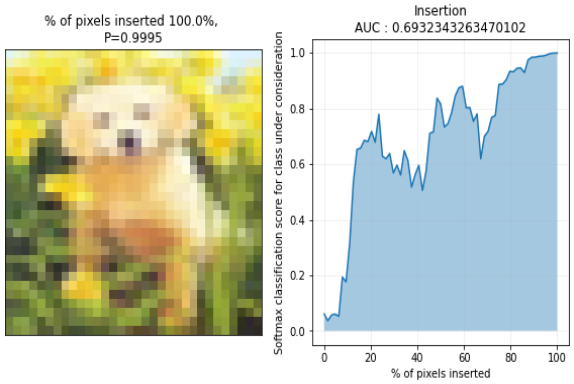


Figure 5.14: Stages of pixel insertion. The amount of pixel inserted increases from left to right.

Similar to the example discussed above, in this experiment the deletion and insertion curves are plotted and the AUC is computed for the mean and standard deviation of explanation distribution obtained in Experiment 1. Here, the order of removal or insertion of pixel depends on the importance of pixel in the mean explanation representation and the standard deviation explanation representation as opposed to a single explanation heatmap. Additionally, it should be noted that instead of analysing the insertion and deletion metric for a single image as has been done in the example discussed above, we conduct the analysis on a batch of images belonging to the same class. This means that each class of the dataset will have a deletion and insertion curve along with the corresponding AUC for every combination of uncertainty estimation and explanation method implemented. This should help in quantifying the performance of the implemented combination on a class-wise basis. Furthermore, this experiment is conducted on both the



(a) Deletion plot along with its AUC.



(b) Insertion plot along with its AUC.

Figure 5.15: Plots generated by application of pixel deletion and insertion on example image depicted in Figure 5.12.

image datasets and the results have been discussed in the next chapter.

5.3.3 Experiment 3: Sanity Checks for the Explanation Methods

A few research works namely [7], [35], [52], [71] proposed authenticating the working of explanation methods. These works claimed that many explanation methods do not "explain" the network prediction and rather perform simple edge recovery on the input image. These research works claim that such methods present the edge recovered image as an explanation of the model decision for a particular input image. The authors of [7] proposed sanity checks to verify if a given explanation method is working as intended. Furthermore, [7] and [35] claim that Guided Backpropagation [75] is among the explanation methods that fail these tests (namely the *weight randomization* test and the *data randomization* test). Another research conducted by [95] claims that the findings of [7] regarding the working of GBP could be incorrect owing to the methodology adopted during the testing. As explanation methods are a core component of our pipeline in this thesis, we decide to test the working of the selected explanation methods by applying the sanity checks ourselves. In the subsequent text, we describe the working of these sanity checks.

Weight randomization test

In this particular test, the authors of [7] suggest to test the reliance of the generated explanation on the weights of the model. The objective is to verify if the explanation changes on perturbing the model weights. If the explanation methods are indeed "explaining" the model decisions, by gradually increasing the amount of perturbation in the model weights, the explanation for a given input image should also be affected. In other words, if the explanation for a given input image does not change even after the model weights are perturbed, the explanation method does not utilise the network information at all and is simply using the edge information to generate the explanation as claimed by [7].

In order to implement this test in our work, we adopt the strategy suggested by the authors to perturb/randomize the weights of our trained model in a layer-by-layer manner. We conduct this test for both the mean and the standard deviation visualization of explanation and begin with a completely unperturbed model (weight randomization=0%). For an undisturbed model, we generate the explanation by applying the explanation method under test to a given input image. We then randomize the weights of a single layer of the network and generate the explanation again. We repeat these steps until we have a completely randomized model (weight randomization = 100%). Here, the value of weight randomization indicates the percent of layers that have perturbed weights. Next, we plot all the explanations for the different weight randomization steps and analyse the differences. In order to quantify the differences between the various explanations generated by the weight randomization steps, we compute the Structural Similarity Index (SSIM). A detailed discussion of this metric and its use in relation to this sanity test is provided at a later stage.

Data randomization test

Another test proposed by [7] to verify the working of explanation methods is that of the data randomization. This test ensures that the explanation methods makes use of the data and its corresponding label while generating explanations and is not just producing the explanation heatmaps by identifying the edges in the input image. This test works by intentionally training the model on shuffled image-label pairs and then observing the performance of explanation methods with this modified configuration. The explanation generated with this new configuration is compared against the explanation generated by the correctly trained model. If both these explanations do not differ, the authors conclude that the explanation method erroneously attributes the edges of input image as the explanation for model decision.

We implement this test in this thesis as well. For this purpose, we train a model correctly and store the explanation (both the mean and the standard deviation visualizations) generated by the explanation method for this input. In this context, training a model correctly means that the labels match the image and are not shuffled. Next, we shuffle the image-label mapping such that the labels no longer match the associated image. Using this shuffled dataset, we train a structurally identical model as used previously from scratch. We then apply the explanation method for a given input image with this configuration and obtain the explanations. In order to observe the differences, we visualize both these explanations simultaneously. If the explanations are different, then the explanation method passes this sanity check. To quantify these differences, we make use of SSIM.

Effect of epochs

This test is not originally proposed in the works of [7]. However, it is designed to serve the same objective as the previous checks. In this test, the idea is to analyse the explanations generated by the explanation method during the model training. In other words, the model is saved at different epochs during the training and these saved copies are then used to generate the explanation for a given input image. This is equivalent to recording the evolution of explanation as the model training progresses. For initial epochs, the explanation should look different than the explanation for later epochs. If this is the case, we can

safely conclude that the explanation method utilises the model training when generating visualizations. We implement this test and similar to the previous tests, we utilise the SSIM to quantify the difference between the heatmaps for the earliest and the latest epoch. This experiment is conducted on both the mean and the standard deviation visualization of the explanation distributions. An advantage of this check is that it does not require additional training of the model as was required in the data randomization test and also eliminates the need for additional processing as was the case in weight randomization test.

Structural Similarity Index (SSIM)

In order to quantify the similarity/dissimilarity of the explanations generated during the sanity checks, we use the Structural Similarity Index (SSIM) metric. This metric lies in the range [0, 1] with a value of 1 indicating two identical images and 0 indicating no structural similarity between the two images. Wang et al. [90] proposed this mathematical concept and it is calculated using three comparison measurements: luminance, contrast and structure. The following mathematical analysis and equations have been taken from [90]:

$$l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \tag{5.2a}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \tag{5.2b}$$

$$s(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3} \tag{5.2c}$$

where x and y are the two images for which the similarity measure is to be computed
$l(x,y)$ is the luminance comparison function
$c(x,y)$ is the contrast comparison function
$s(x,y)$ is the structure comparison function
μ_x is the average of x
μ_y is the average of y
σ_x^2 is the variance of x
σ_y^2 is the variance of y
σ_{xy} is the covariance of x and y
c_1, c_2 and c_3 are values added to stabilize the division terms.
Combining the three comparison functions, we get SSIM as:

$$SSIM(x,y) = l(x,y)^\alpha . c(x,y)^\beta . s(x,y)^\gamma \tag{5.3}$$

where α, β and γ are the weights for the respective comparison measurement. Substituting the functions expressions, setting the weights (α, β and γ) as 1 and using $c_3 = c_2/2$, we get Equation 5.4:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{5.4}$$

It should be noted that authors propose computing the SSIM on a local scale and then averaging these local values in a sliding window fashion. The justification to do so is provided by claiming that the distortions and the statistical features of the image might be space variant. Applying this step modifies the formula as:

$$SSIM_{mean}(X, Y) = \frac{1}{M} \sum_{j=1}^{M} SSIM(x_j, y_j) \tag{5.5}$$

where X and Y are the images for which the similarity is to be computed, x_j and y_j are the j^{th} local windows in the original images and M is the total number of local windows. In the context of this thesis, SSIM is computed using the function *tf.image.ssim* available in TensorFlow. The use of this metric on individual sanity check has been discussed in the next chapter.

5.3.4 Experiment 4: Variation in Uncertainty of Explanation for FER+

This experiment is a modification of Experiment 1 and it should be noted that this experiment is only conducted on the FER+ dataset. In Experiment 1, the explanation is calculated by computing the gradient of network prediction with respect to the input image. This is helpful in the case when the dataset is easy and the network predictions are correct. This approach however, could have problems in cases where the network prediction is often not matching the ground truth (as is the case of FER+ dataset). The modification introduced in this experiment is in the gradient computation step. As opposed to the gradient computation described earlier, we calculate the gradient of ground truth class score with respect to the input image in this experiment. This difference can be further clarified with the help of an example depicted in Figure 5.16.

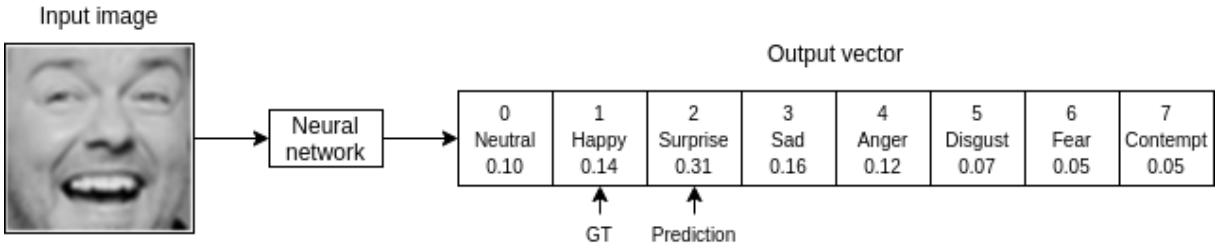


Figure 5.16: Illustration used to describe the differences between Experiment 1 and Experiment 4.

In Figure 5.16, an input image, a neural network and the output vector of the network have been depicted. In this particular example, the ground truth label is "happy" whereas the network predicts the expression as that of "surprise". In Experiment 1, we compute the gradient for explanation as given by Equation 5.6:

$$gradient = \frac{\partial(Output_{Prediction})}{\partial(Input)} \tag{5.6}$$

This implementation helps in visualizing the reason "why" the model predicted what it predicted. Additional insights can also be gained by tuning this gradient computation. The modified gradient

computation is as given in Equation 5.7:

$$gradient = \frac{\partial(Output_{Ground\ Truth})}{\partial(Input)} \tag{5.7}$$

By using this equation, we aim to visualize the features/image regions that the network should focus on in order for its prediction to match the ground truth. Equation 5.6 and Equation 5.7 converge when the network prediction matches the ground truth for a given input image. Experiment 4 has only been conducted on FER+ dataset as the predictions on this dataset often do not match the ground truth. This can be attributed to the fact that this is a difficult dataset to train the networks on.

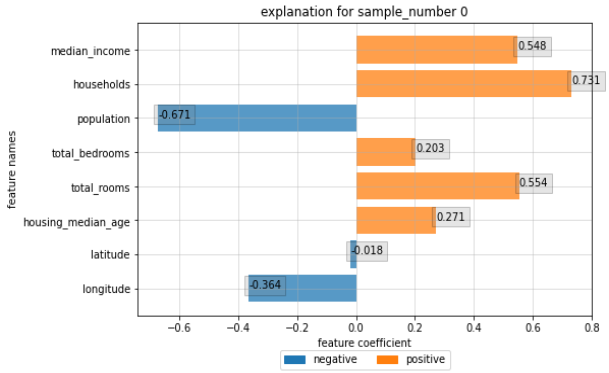
5.3.5 Experiment 5: Uncertainty of Explanation for Numerical Regression

The objective of this experiment is identical to that of Experiment 1, that is to combine the uncertainty estimation and explanation methods. However, the task of choice here is numerical regression instead of image classification. The reason for selecting numerical regression as use case allows to test the robustness of applicability of the proposed approach. Another difference is in the combinations of the uncertainty estimation and explanation method implemented. An overview of the combinations implemented in this experiment have been provided in Table 5.3.

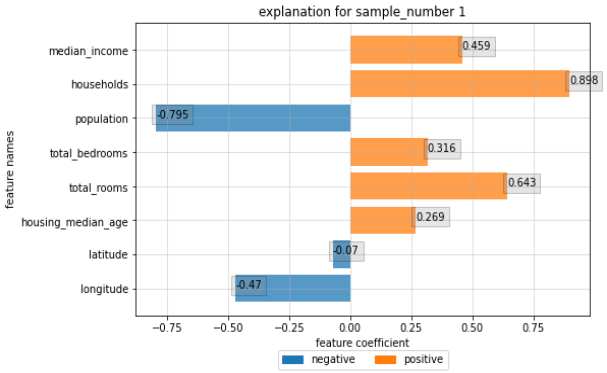
Dataset	UQ	EX
California Housing	Deep Ensembles	LIME
California Housing	Deep Ensembles	GBP
California Housing	MC Dropout	LIME
California Housing	MC Dropout	GBP
California Housing	MC DropConnect	LIME
California Housing	MC DropConnect	GBP
California Housing	Flipout	LIME
California Housing	Flipout	GBP

Table 5.3: An overview of the combinations of uncertainty estimation and explanation methods tested for the numeric regression task.

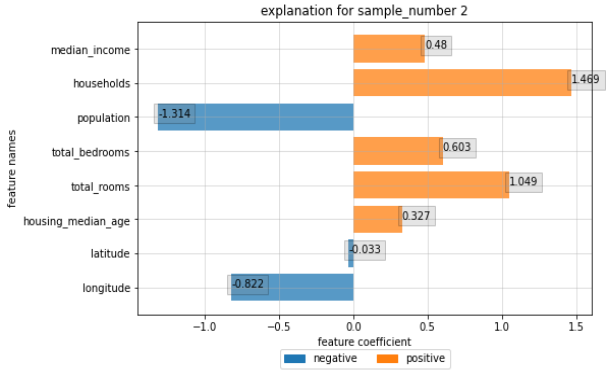
For this experiment, we select LIME as one of the explanation methods as it can be applied to tabular datasets. Additionally, we select GBP for this experiment as it has not been used to generate explanations for tabular data to the best of our knowledge. The rest of the pipeline to combine the uncertainty estimation and explanation method is identical to Experiment 1. As the task to be solved in this experiment is different, the format of explanation will naturally be different as well. In this experiment, we visualize the explanations as horizontal bar charts as opposed to heatmaps of Experiment 1. Figure 5.17 depicts the distribution of explanation having 10 instances for a given set of input features. Figure 5.18 provides the concise representation of this distribution.



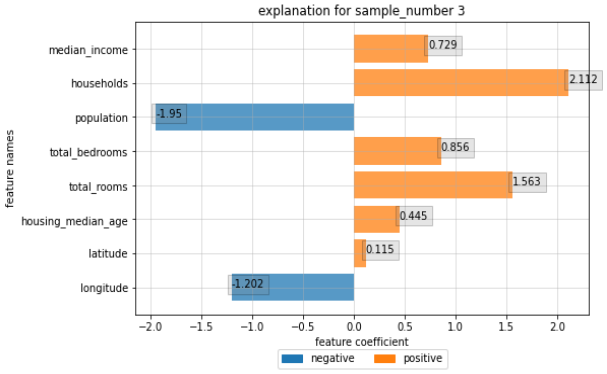
(a) Explanation id: 0



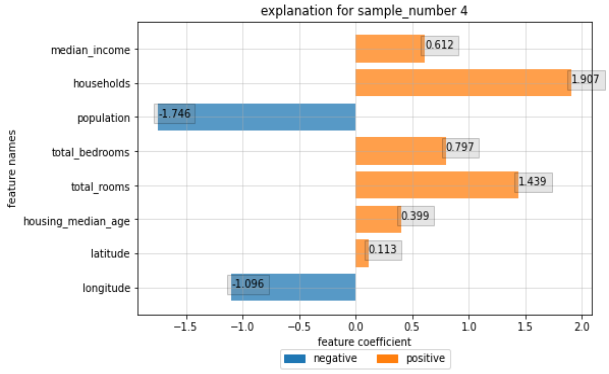
(b) Explanation id: 1



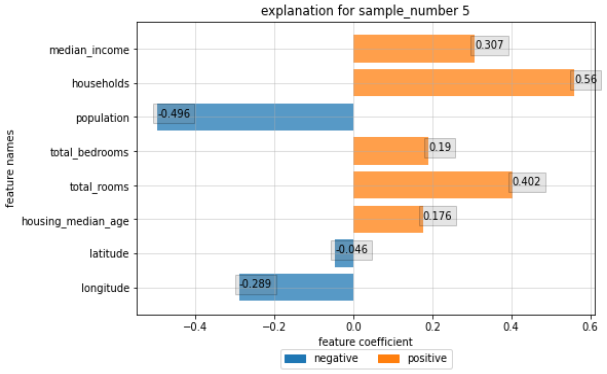
(c) Explanation id: 2



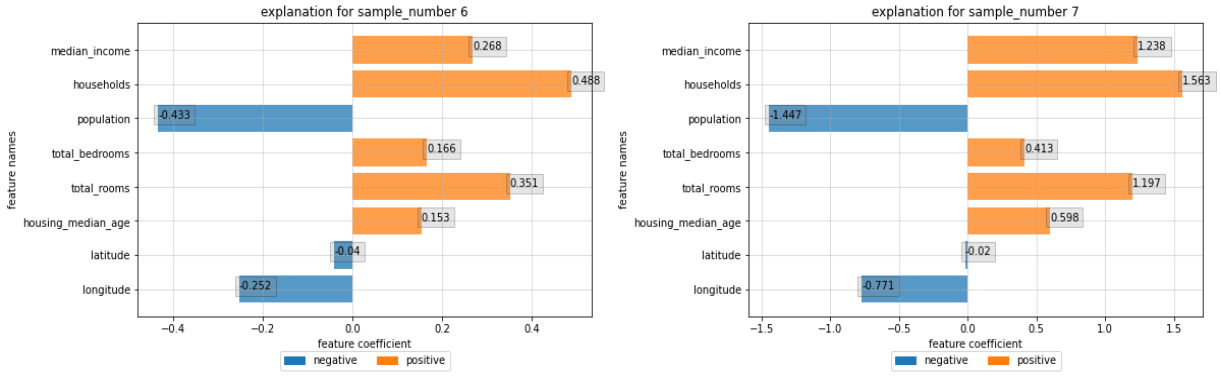
(d) Explanation id: 3



(e) Explanation id: 4

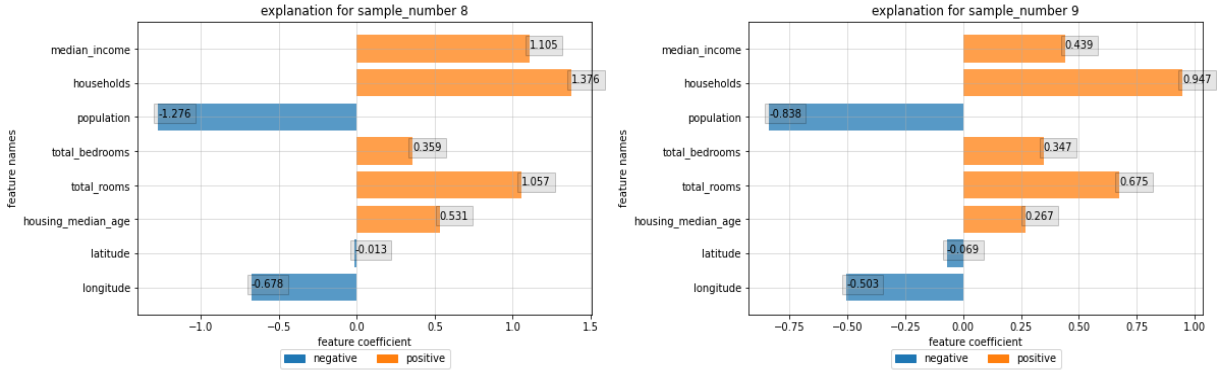


(f) Explanation id: 5



(g) Explanation id: 6

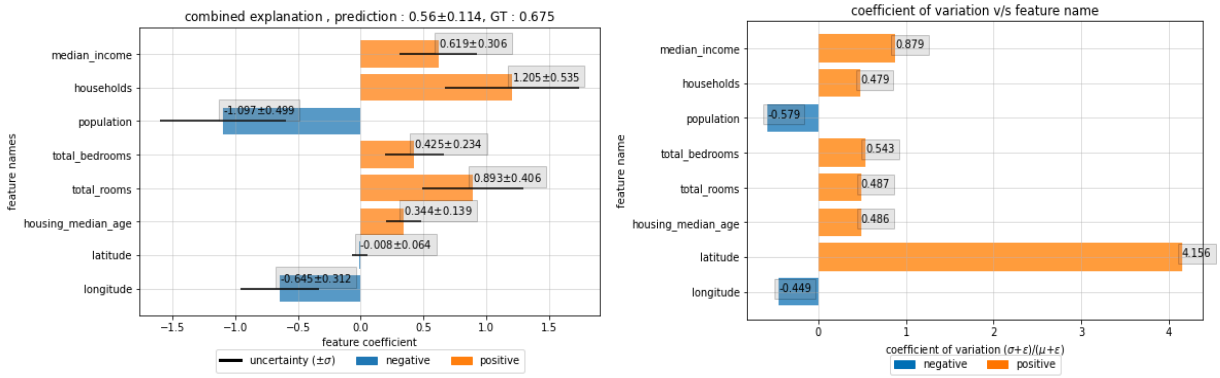
(h) Explanation id: 7



(i) Explanation id: 8

(j) Explanation id: 9

Figure 5.17: Distribution of explanation for an input belonging to California Housing dataset. The values of input feature are: [longitude: 0.190, latitude: 0.556, housing median age: 1, total rooms: 0.103, total bedrooms: 0.166, population: 0.053, households: 0.159, median income: 0.169].



(a) Combined mean and standard deviation representation

(b) Coefficient of variation representation

Figure 5.18: Concise representations of the explanation distribution generated in Figure 5.17.

6

Results

The previous chapter discussed the implementation of the proposed approach and also provided the details of the experiments conducted in this thesis along with a brief description of the datasets used. In this chapter, we discuss the results and analyse the findings of these experiments.

6.1 Experiment 1: Uncertainty of Explanation

In this experiment, the combinations of uncertainty estimation methods with explanation methods are tested in context of image classification task. The results of this experiment are the mean and the standard deviation visualization of the generated explanation distributions. As an auxiliary, the coefficient of variation of the explanation distribution has also been visualized. Particularly those heatmap visualizations are provided in this section from which insights can be drawn. As this experiment is conducted on two image datasets, namely CIFAR-10 and FER+, the results and the findings will be presented and discussed with respect to these datasets. The results have been presented in Figure 6.1 - Figure 6.8.

The visualizations are provided in the form of rows and columns for analysis. The input image is provided at the first row and first column. The rows correspond to the different uncertainty estimation component in the combination. The columns can be symmetrically analysed along their length. The first three columns correspond to the concise representations (the mean (μ), the standard deviation (σ) and the coefficient of variation (CV)) obtained using IG and the last three columns correspond to those obtained using GBP. For instance, if we refer to the 4^{th} row and the last column, we obtain the coefficient of variation of the explanation distribution generated using a combination of GBP and Flipout. In subsequent discussions, the concise representations will be denoted by their identifier namely μ, σ, and CV. It should be noted that a lower value of CV or a lower σ or a higher μ could indicate that the certainty of relevance of that particular pixel in decision making is high. The logic supporting this statement is: A higher mean for a particular pixel would indicate a higher agreement amongst the explanation instances and by extension boosts the certainty of that feature being relevant to the model decision. This logic can be applied to the analyse the standard deviation and the coefficient of variaion representations as well.

CIFAR-10

A few interesting visualizations from this dataset have been depicted from Figure 6.1 - Figure 6.4. We discuss each example individually and attempt to extract meaningful insights.

- In Figure 6.1, the input image is that of a truck. From this figure, we can see that the IG focuses on the object of interest as well as some pixels in the background. Additionally, the explanations of the MC DropConnect and the Flipout configuration tend to have sparser pixel population as compared to those generated from Deep Ensembles and MC Dropout. From the CV visualization of IG, it can be deduced that the pixels on body of the truck cab have been correctly highlighted. This is also the case for CV representations of GBP. Also for GBP, the activation of background pixels in the explanation representation is not as prominent as for IG. Finally, the values of σ representation for IG are relatively high when compared to the values of σ representation for GBP. This could mean that the explanations generated by IG are more uncertain as compared to the explanations generated by GBP.

- Figure 6.2 depicts the results for an image of an airplane with a frontal view. From this figure, it is evident that the explanation representations generated by IG are noisy for all the uncertainty estimation methods. These explanations fail to highlight the features of object of interest. Additionally, the values in σ representation are also significantly high meaning that the amount of uncertainty in the generated explanation is high as well. This impacts the CV visualization. On the other hand, the explanations generated by GBP are not so noisy. We can visually segregate the components of the aircraft from the explanations clearly. The σ values are not as high (as in the case of IG) and this results in clear CV visualizations. We can distinguish the features of the aircraft clearly in the CV representations of GBP. This means that the uncertainty component in the explanations generated by GBP is low.

- Figure 6.3 provides the results for an image of a deer. Interestingly, in this case the IG explanations almost always highlight the torso of the deer. Unlike the last example, the explanations are relatively clear and impart meaningful insights. From the CV representations of the IG, we can deduce that the body of the deer is consistently being identified as a highly certain and relevant feature. For the case of GBP, the upper thigh of the hind legs along with the back of the deer have been identified as being the relevant feature. In this exampls, the CV representation of the GBP highlights the silhouette of the animal. There are a few pixels in the background that have erroneously been identified as being relevant.

- Figure 6.4 shows the results for the image of a dog. The explanations generated using the IG are noisy and do not highlight any particular feature. This could possibly be attributed to the malfunction of the IG approach. However, GBP explanations are able to highlight the face and the front paws of the dog correctly. In this example, the explanation representations of MC DropConnect and Flipout are sparser as compared to those generated by Deep Ensembles and MC Dropout. It can be seen in the CV visualization generated by GBP (for all uncertainty methods), the uncertainty associated with the relevant features is low as compared to IG.

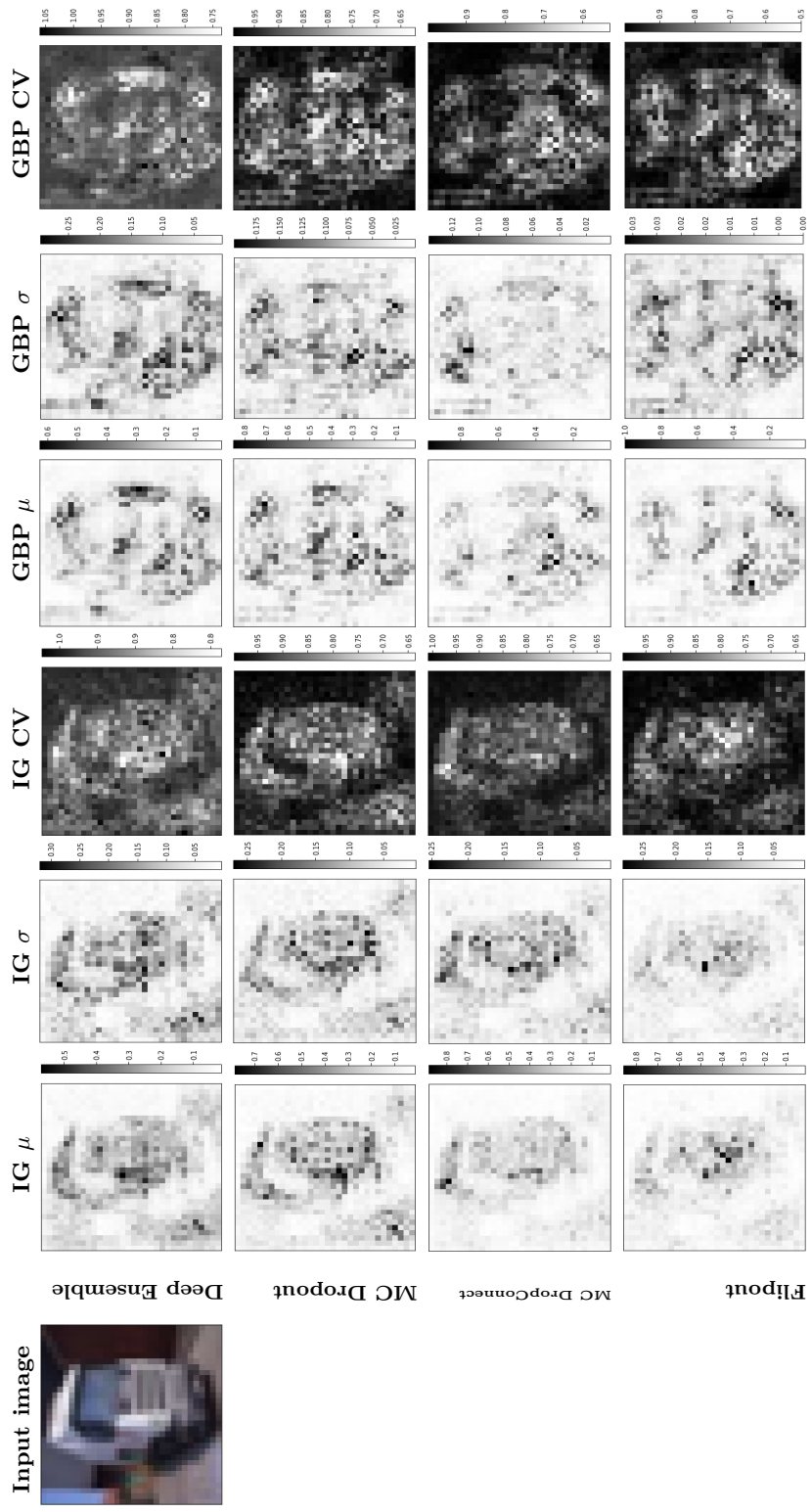


Figure 6.1: Input image of a truck taken from the CIFAR-10. The columns represent the concise representations generated by using the two explanation methods. The rows correspond to different uncertainty estimation approaches.

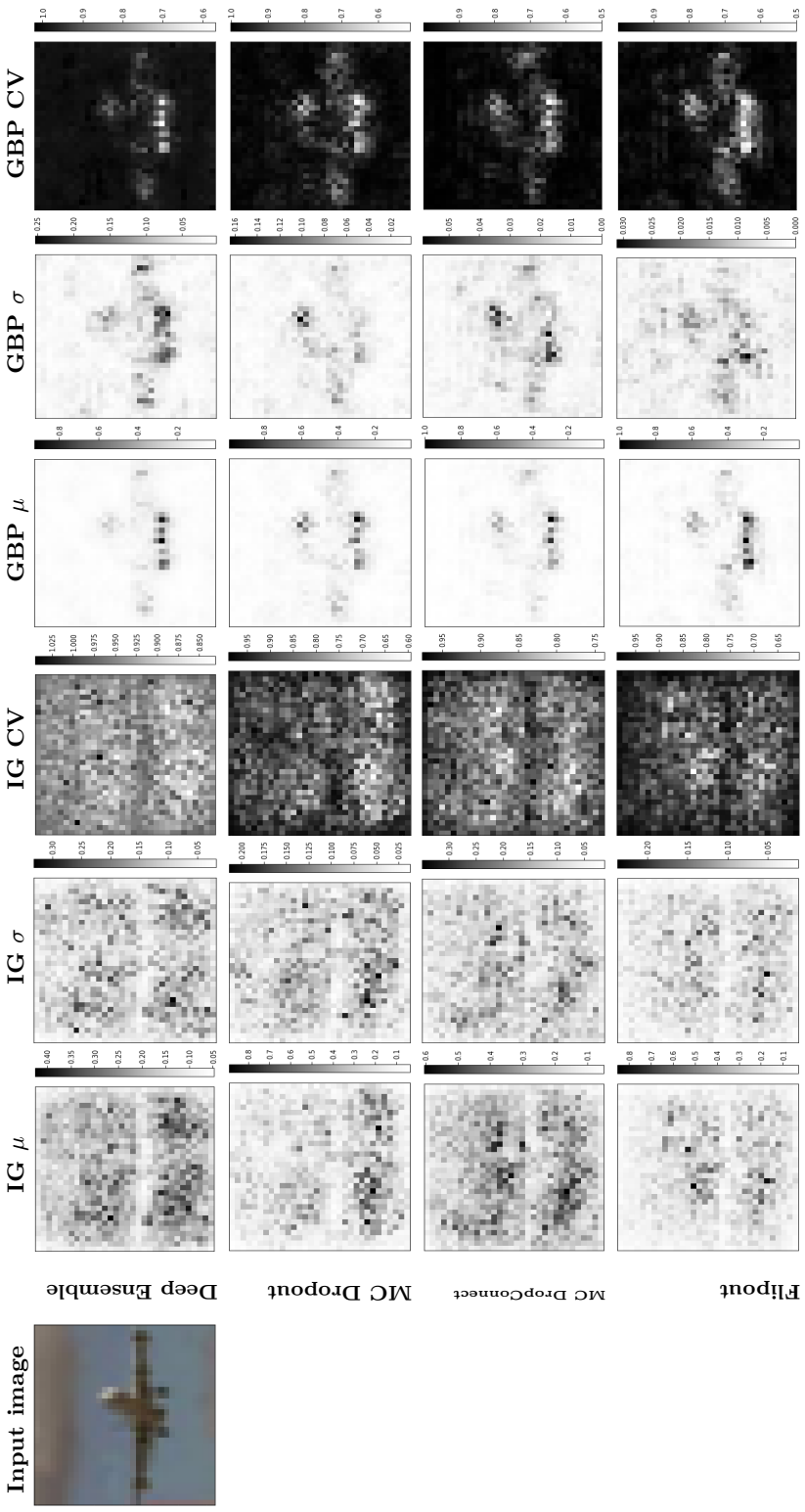


Figure 6.2: Input image of a plane taken from the CIFAR-10. The columns represent the concise representations generated by using the two explanation methods. The rows correspond to different uncertainty estimation approaches.

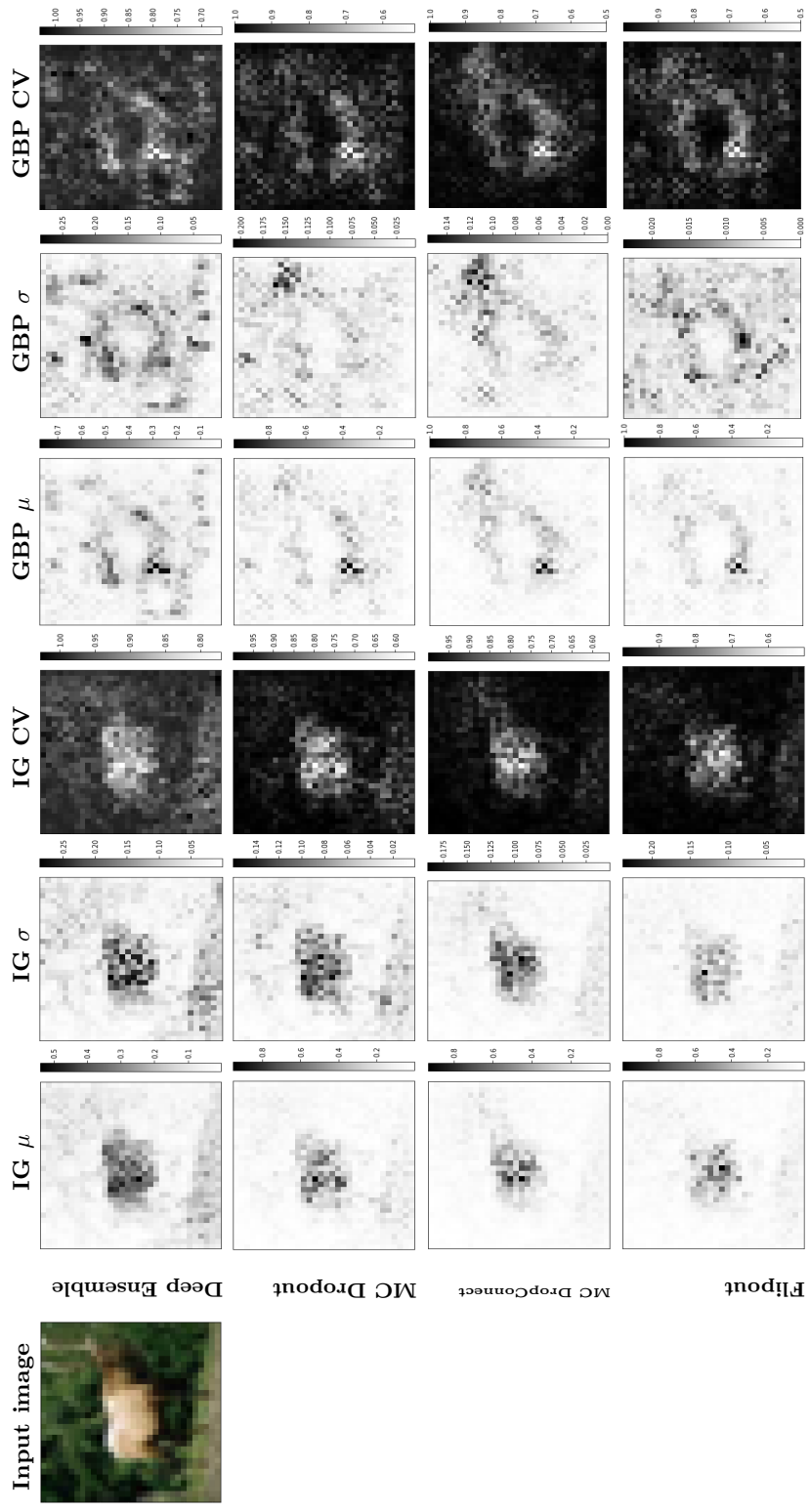


Figure 6.3: Input image of a deer taken from the CIFAR-10. The columns represent the concise representations generated by using the two explanation methods. The rows correspond to different uncertainty estimation approaches.

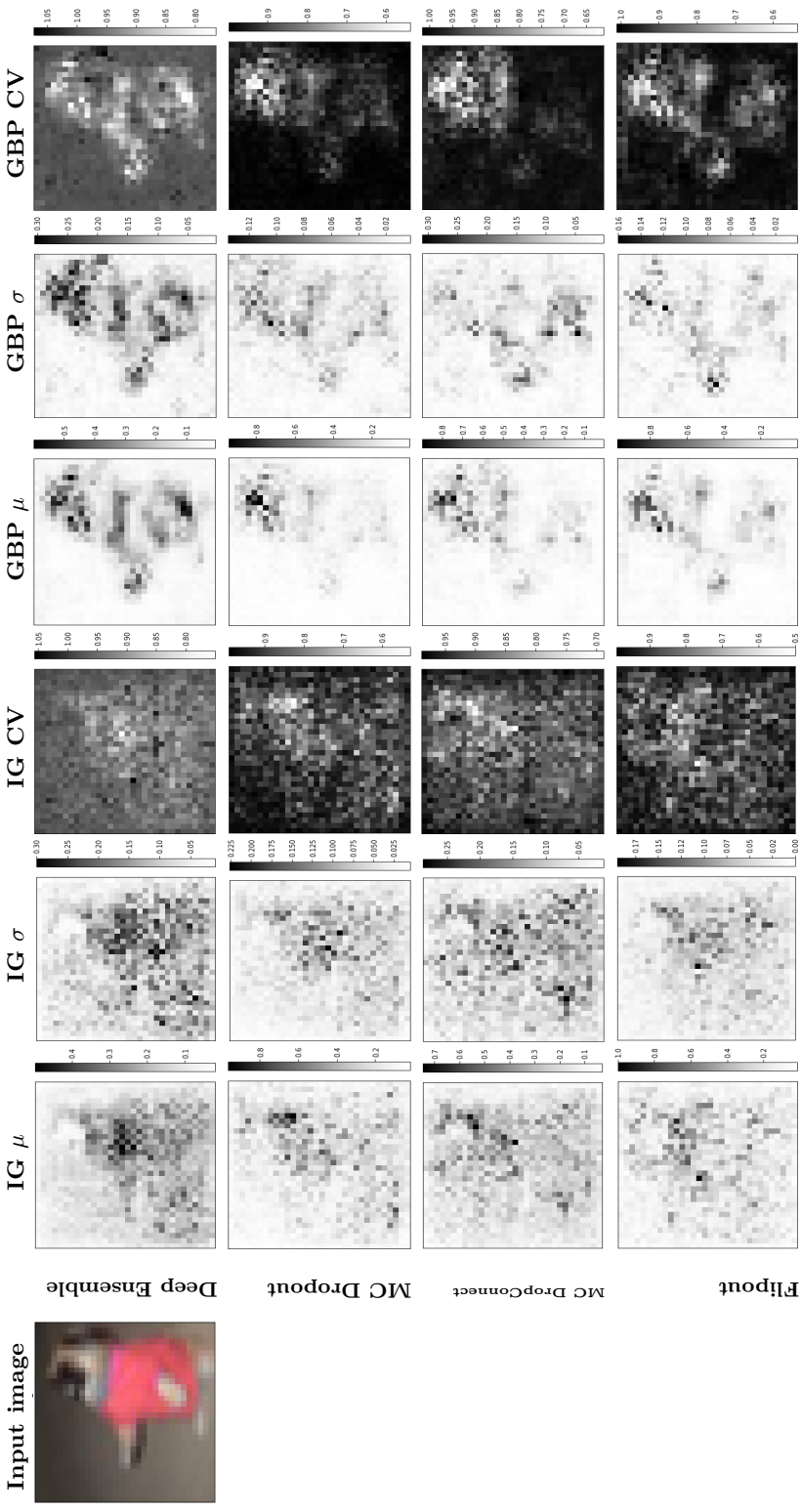


Figure 6.4: Input image of a dog taken from the CIFAR-10. The columns represent the concise representations generated by using the two explanation methods. The rows correspond to different uncertainty estimation approaches.

FER+

Figure 6.5 - Figure 6.8 depict the results taken from FER+ dataset for the analysis.

- Figure 6.5 depicts a happy person and the various explanation representations alongside. It is evident that the IG highlights entire swathes of the face of the person as a highly relevant feature. Additionally, the explanation representations generated by IG corresponding to the MC DropConnect and Flipout are sparse as compared to those of Deep Ensembles and MC Dropout. Interestingly, GBP is able to distinguish amongst different facial features as can be seen in its explanation representations. It can segregate between the nose, the lips and even the eyes of the person properly. Similar to the case with IG, the explanation representations of GBP generated by MC DropConnect and Flipout are sparse as compared to their counterparts from Deep Ensembles and MC Dropout. The values of the standard deviation representations for the GBP are low as compared to their IG equivalents. This could indicate that the explanations generated by GBP are more certain as compared to those by IG.

- Figure 6.6 depicts the surprised expression of a child along with its explanation representations. Similar to the previous example, IG highlights the entire facial region as an important feature and the explanation representations of MC DropConnect and Flipout (for IG) are sparser as compared to those of Deep Ensembles and MC Dropout. For the case of GBP, the difference amongst the mean representation of various uncertainty estimation approaches is not as pronounced. For GBP, the different features are distinguishable (as is the case in the previous example). We can identify the nostrils, the eyes, and the lips on the child's face. The values in the standard deviation representation of GBP are less than or almost equal to their counterpart from IG. This indicates that the confidence in the relevance of features highlighted by GBP explanations is more than or equal to the confidence in the IG equivalents. Also, the CV visualizations for the GBP are clearer than for the IG.

- In Figure 6.7, an angry person and the corresponding explanation representations have been depicted. For the case of IG, the pixels on the hand of the person have also been highlighted in all the representations barring Flipout. As is with the examples till now, the explanation representations for IG generated by MC DropConnect and Flipout are sparser as compared to the other uncertainty estimation approach. Additionally, comparing the values of standard deviation representation for IG and GBP, we can deduce (from the same rationale applied in the previous example) that the uncertainty in the explanations generated by GBP is lower as compared to the ones generated by IG.

- Figure 6.8 depicts a person with a neutral expression along with the explanation representations. The representations of IG do not highlight any particular facial feature. Conversely, the representations for GBP clearly illustrate the facial features including the eyes, the nose, the lips and the chin of the person. As is with previous examples, the representations generated by IG for MC DropConnect and Flipout are sparser as compared to Deep Ensembles and MC Dropout. Moreover, this phenomenon is not as evident for the case of GBP as the density of pixels appears almost identical.

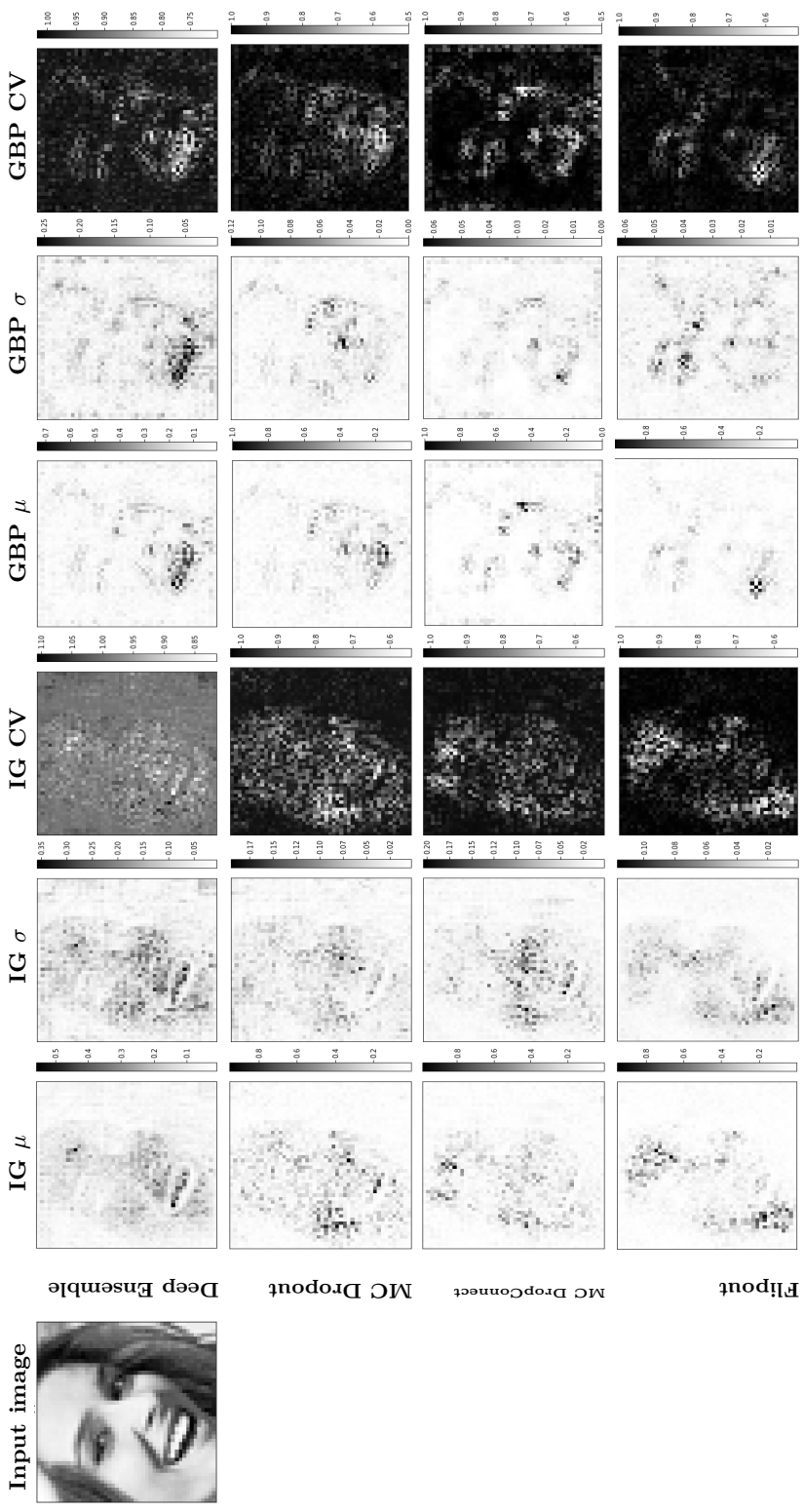


Figure 6.5: Input image of a happy person taken from the FER+. The columns represent the concise representations generated by using the two explanation methods. The rows correspond to different uncertainty estimation approaches.

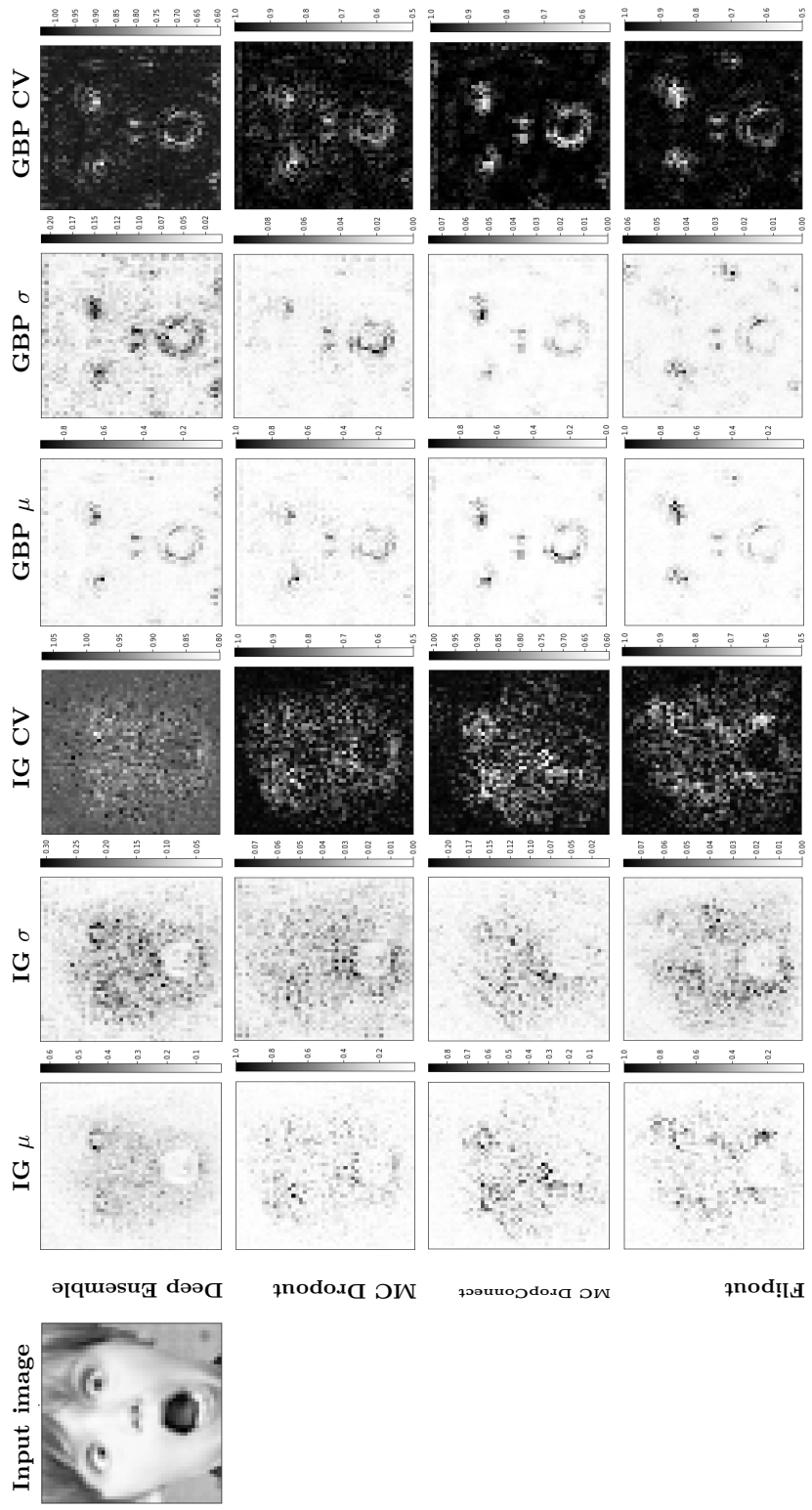


Figure 6.6: Input image of a surprised child taken from the FER+. The columns represent the concise representations generated by using the two explanation methods. The rows correspond to different uncertainty estimation approaches.

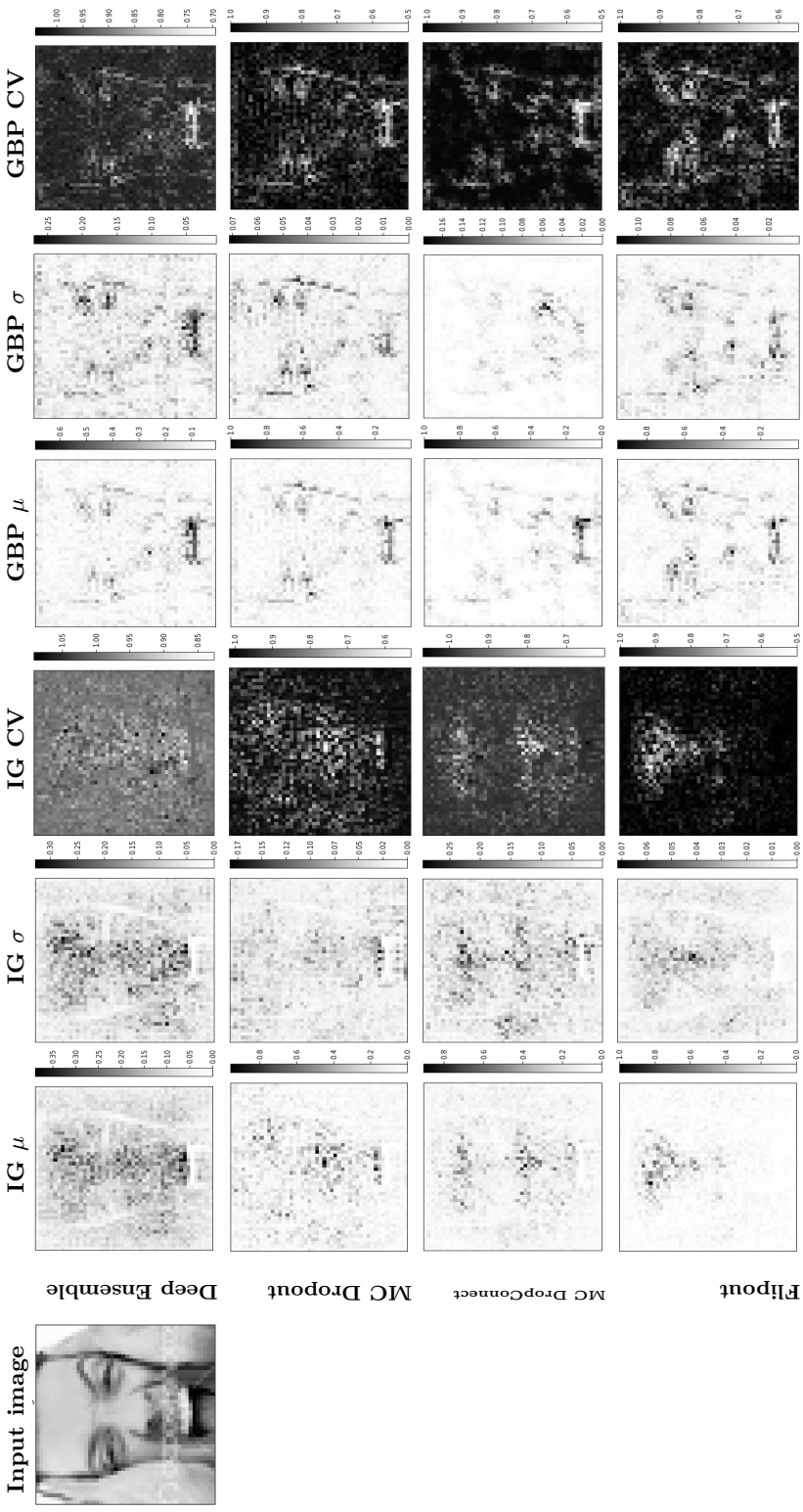


Figure 6.7: Input image of an angry person taken from the FER+. The columns represent the concise representations generated by using the two explanation methods. The rows correspond to different uncertainty estimation approaches.

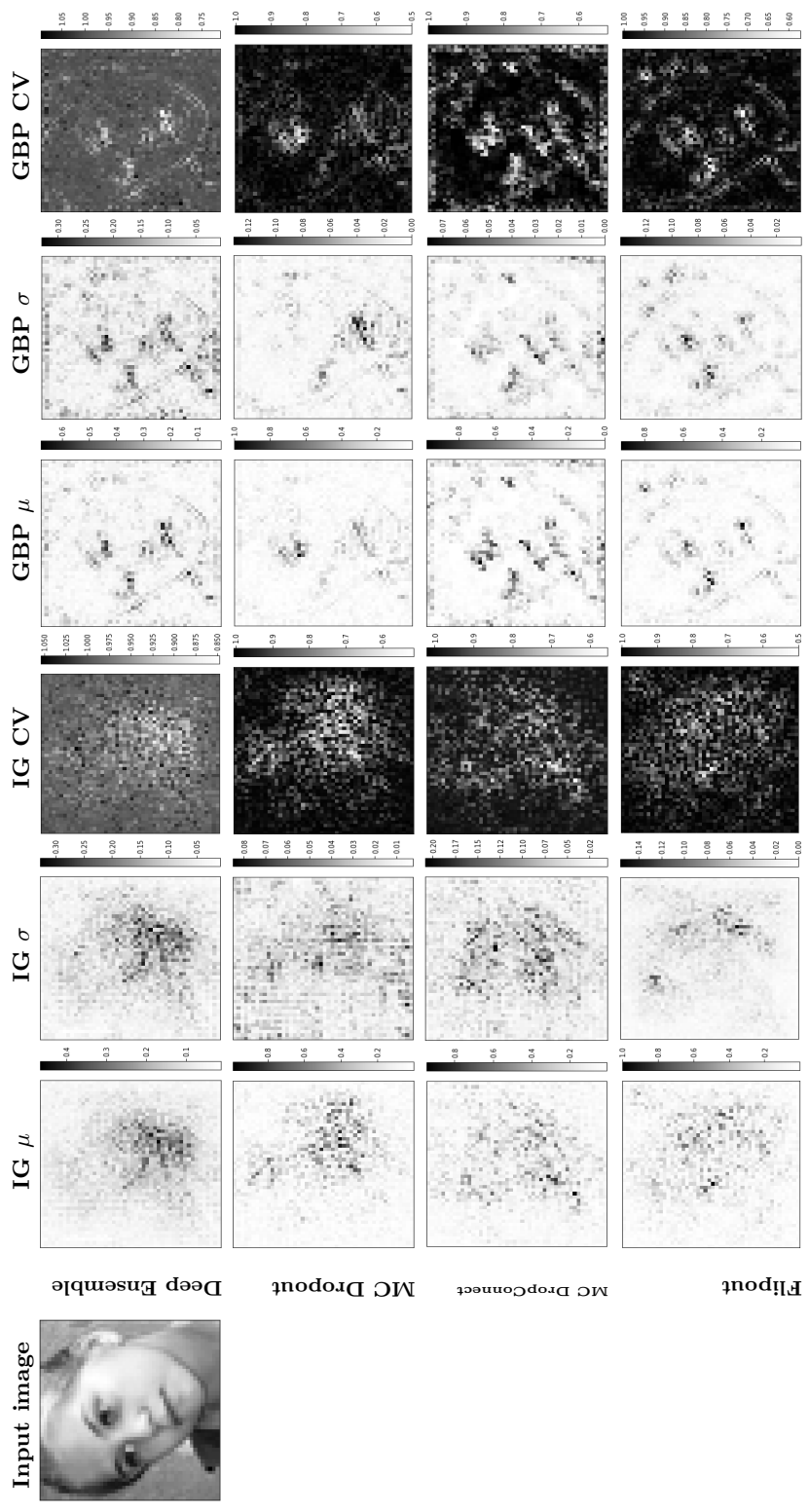


Figure 6.8: Input image of a person with neutral expression taken from the FER+. The columns represent the concise representations generated by using the two explanation methods. The rows correspond to different uncertainty estimation approaches.

Summarizing the results of Experiment 1

In the following text, we summarize the findings of this experiment.

- Explanations generated by IG usually tend to be spread apart on the object of interest and do not clearly highlight any particular features. This has been observed in the examples discussed above for both the datasets. This could be a result of the averaging used in the computation of IG. As many interpolated images contribute to the final explanation, it might be possible that they could end up "spreading" the relevance across the available pixel space rather than building it up at a selected location. This might not be an ideal scenario as a noisy explanation might not be of much help.
- Conversely, explanations generated by GBP usually tend to highlight individual features. This phenomenon has been observed in both the datasets. This is a useful finding as explanations that highlight individual features could be used to troubleshoot/debug networks among other applications.
- From the discussions of the results conducted above, it can be seen that the values of the mean, the standard deviation and the coefficient of variation representation can govern the amount of uncertainty in the generated explanation. A lower mean or a higher standard deviation or a higher coefficient of variation could indicate a higher amount of uncertainty in the explanation distribution.
- The explanation representation for IG for the MC DropConnect and Flipout are generally sparse when compared those generated by Deep Ensemble and MC Dropout. This observation is evident from majority of the examples provided in the previous section. No such repetitive pattern is observed with the GBP.

6.2 Experiment 2: Pixel Deletion/Insertion

In the following section, we present the results obtained from the pixel deletion/insertion test. This approach is proposed by [54]. The working of this experiment along with an example has already been discussed in the previous chapter. For the ease of analysis, we split this evaluation in subsections, each discussing a particular configuration. For two datasets (CIFAR-10, FER+), two explanation approaches (IG, GBP), two metrics (deletion and insertion) and two explanation representation (mean and standard deviation) we obtain a total of 16 set of values. These 16 set of values have been computed over as batches of 100 images representing all the classes (10 for CIFAR-10 and 8 for FER+) in the dataset. It should be noted that the resolution (measure of granularity) of this metric depends on the number of pixels deleted/inserted in each step. For a small number of pixels deleted/inserted per step, the deletion/insertion curves will be smooth and the value of these metrics will be close to the true value. However, the computation time goes up as the number of inferences needed also increase. For large number of pixels deleted/inserted per step, the deletion/insertion curves will be rough and the value of these metrics will be approximation of the true value. We select a number that affords a trade-off between the quality of the metrics and the time for computation. Next, we analyse each table individually to gather insights. During this analysis, a key point to remember is that a lower AUC is better for deletion metric and a higher AUC is better for insertion metric. The deletion and insertion plots for this experiment have

been provided in Appendix C. In Table 6.1 - Table 6.16 following convention has adopted: **DE**=Deep Ensemble, **MCDO**=Monte Carlo Dropout, **MCDC**=Monte Carlo DropConnect, and **F**=Flipout.

Table 6.1 and Table 6.2 present the deletion metrics for the mean and the standard deviation explanation representations respectively generated by applying GBP on a batch of CIFAR-10 images.

CIFAR-10 GBP: Deletion metric for mean representation

The lowest/best AUC in this table is 0.142 for the *dog* class when using the MC Dropout. The highest/worst AUC is 0.684 for the *plane* class when using Deep Ensemble.

UQ	Plane	Auto	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
DE	**0.684**	0.531	0.445	0.439	0.484	0.343	0.663	0.538	0.475	0.621
MCDO	0.493	0.380	0.375	0.208	0.274	**0.142**	0.371	0.229	0.298	0.278
MCDC	0.542	0.283	0.258	0.254	0.216	0.236	0.367	0.227	0.226	0.277
F	0.400	0.344	0.345	0.286	0.189	0.187	0.358	0.218	0.184	0.286

Table 6.1: Class-wise deletion AUC for mean representations of explanation distributions obtained using CIFAR-10 and GBP.

CIFAR-10 GBP: Deletion metric for standard deviation representation

The lowest/best AUC is 0.143 for the *dog* class when using MC Dropout whereas the highest/worst AUC is 0.700 for the *frog* class when using Deep Ensemble.

UQ	Plane	Auto	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
DE	0.647	0.503	0.423	0.414	0.472	0.313	**0.700**	0.509	0.427	0.600
MCDO	0.469	0.380	0.372	0.227	0.290	**0.143**	0.396	0.230	0.306	0.278
MCDC	0.477	0.268	0.264	0.262	0.266	0.218	0.416	0.217	0.225	0.295
F	0.341	0.319	0.371	0.285	0.212	0.186	0.421	0.209	0.163	0.271

Table 6.2: Class-wise deletion AUC for standard deviation representations of explanation distributions obtained using CIFAR-10 and GBP.

Table 6.3 and Table 6.4 provide the insertion metrics for the mean and the standard deviation explanation representations respectively obtained by applying GBP on CIFAR-10 images.

CIFAR-10 GBP: Insertion metric for mean representation

The lowest/worst AUC is 0.284 for the *dog* class with the Deep Ensemble. The highest/best AUC is 0.931 for the *frog* class when using the Flipout.

UQ	Plane	Auto	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
DE	0.333	0.610	0.474	0.413	0.489	**0.284**	0.883	0.513	0.620	0.780
MCDO	0.382	0.674	0.593	0.410	0.539	0.300	0.859	0.447	0.657	0.717
MCDC	0.371	0.718	0.478	0.464	0.660	0.424	0.846	0.525	0.706	0.719
F	0.420	0.730	0.583	0.503	0.506	0.301	**0.931**	0.563	0.597	0.653

Table 6.3: Class-wise insertion AUC for mean representations of explanation distributions obtained using CIFAR-10 and GBP.

CIFAR-10 GBP: Insertion metric for standard deviation representation

The lowest/worst AUC is 0.273 for the *dog* class when using MC Dropout. The highest/best AUC is 0.891 for the *frog* class with Flipout.

UQ	Plane	Auto	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
DE	0.323	0.610	0.481	0.384	0.470	0.308	0.875	0.518	0.613	0.797
MCDO	0.381	0.656	0.533	0.452	0.551	**0.273**	0.846	0.453	0.650	0.679
MCDC	0.349	0.699	0.448	0.435	0.668	0.399	0.855	0.505	0.726	0.699
F	0.442	0.688	0.597	0.472	0.512	0.277	**0.891**	0.561	0.602	0.639

Table 6.4: Class-wise insertion AUC for standard deviation representations of explanation distributions obtained using CIFAR-10 and GBP.

Table 6.5 and Table 6.6 show the deletion metrics for the mean and the standard deviation explanation representations respectively obtained by implementing IG on CIFAR-10 images.

CIFAR-10 IG: Deletion metric for mean representation

The lowest/best AUC in this table is 0.145 for the *dog* class when using MC Dropout. The highest/worst AUC is 0.756 for the *frog* class obtained with Deep Ensembles.

UQ	Plane	Auto	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
DE	0.543	0.700	0.469	0.436	0.532	0.357	**0.756**	0.611	0.604	0.690
MCDO	0.428	0.476	0.353	0.261	0.321	**0.145**	0.482	0.320	0.386	0.341
MCDC	0.475	0.473	0.233	0.302	0.320	0.278	0.420	0.328	0.327	0.358
F	0.282	0.440	0.404	0.336	0.261	0.198	0.450	0.288	0.205	0.378

Table 6.5: Class-wise deletion AUC for mean representations of explanation distributions obtained using CIFAR-10 and IG.

CIFAR-10 IG: Deletion metric for standard deviation representation

The lowest/best AUC is 0.139 for the *dog* clas obtained with MC Dropout. The highest/worst AUC is 0.800 for the *frog* class with Deep Ensemble.

UQ	Plane	Auto	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
DE	0.455	0.640	0.438	0.425	0.497	0.326	**0.800**	0.526	0.560	0.620
MCDO	0.380	0.458	0.360	0.262	0.333	**0.139**	0.485	0.283	0.396	0.329
MCDC	0.369	0.450	0.234	0.294	0.324	0.232	0.460	0.301	0.301	0.321
F	0.250	0.434	0.418	0.340	0.255	0.187	0.477	0.263	0.190	0.324

Table 6.6: Class-wise deletion AUC for standard deviation representations of explanation distributions obtained using CIFAR-10 and IG.

Table 6.7 and Table 6.8 show the insertion metrics for the mean and the standard deviation representations respectively generated by applying IG on batch of CIFAR-10 images.

CIFAR-10 IG: Insertion metric for mean representation

The lowest/worst AUC is 0.342 for the *dog* class with Deep Ensemble. The highest/best AUC is 0.899 for the *frog* class with Flipout.

UQ	Plane	Auto	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
DE	0.432	0.580	0.476	0.400	0.504	**0.342**	0.880	0.595	0.563	0.812
MCDO	0.473	0.710	0.571	0.432	0.605	0.385	0.830	0.545	0.540	0.743
MCDC	0.457	0.650	0.491	0.447	0.633	0.457	0.846	0.596	0.641	0.723
F	0.492	0.666	0.613	0.452	0.577	0.410	**0.899**	0.647	0.572	0.691

Table 6.7: Class-wise insertion AUC for mean representations of explanation distributions obtained using CIFAR-10 and IG.

CIFAR-10 IG: Insertion metric for standard deviation representation

The lowest/worst AUC is 0.322 for the *dog* class when using Deep Ensemble. The highest/best AUC is 0.929 for the *frog* class when using Flipout.

From the results presented in Table 6.1 - Table 6.8, we observe that the lowest/best AUC for deletion is 0.139 for *dog* class (standard deviation representations generated using MC Dropout and IG). The highest/worst AUC for deletion is 0.800 for *frog* class (standard deviation representations obtained using Deep Ensemble and IG). Similarly, the lowest/worst AUC for insertion is 0.273 for *dog* class (standard deviation representation obtained using MC Dropout and GBP) and the highest/best AUC is 0.931 for

UQ	Plane	Auto	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
DE	0.411	0.590	0.490	0.389	0.482	**0.322**	0.872	0.570	0.566	0.802
MCDO	0.452	0.698	0.550	0.413	0.625	0.388	0.819	0.541	0.568	0.750
MCDC	0.451	0.640	0.490	0.445	0.631	0.445	0.868	0.587	0.636	0.727
F	0.473	0.690	0.616	0.464	0.557	0.385	**0.929**	0.581	0.538	0.690

Table 6.8: Class-wise insertion AUC for standard deviation representations of explanation distributions obtained using CIFAR-10 and IG.

frog class (mean representation obtained using Flipout and GBP). A noteworthy observation is that the *dog* class performs well for the pixel deletion but is the worst performer for pixel insertion whereas the *frog* class behaves exactly the other way round. In the next set of tables, we analyse the deletion/insertion metrics for the FER+ dataset.

Table 6.9 and Table 6.10 provide an overview of the deletion metrics for the mean and the standard deviation representation respectively obtained by applying GBP on a batch of FER+ images.

FER+ GBP: Deletion metric for mean representation

The lowest/best AUC is 0.123 for the *fear* class when using MC Dropout whereas the highest/worst AUC is 0.477 for the *contempt* class when using MC Dropout.

UQ	Contempt	Sadness	Neutral	Surprise	Anger	Happy	Fear	Disgust
DE	0.310	0.246	0.314	0.275	0.295	0.374	0.231	0.202
MCDO	**0.477**	0.324	0.460	0.176	0.322	0.193	**0.123**	0.244
MCDC	0.296	0.258	0.284	0.223	0.243	0.345	0.164	0.166
F	0.275	0.257	0.300	0.224	0.238	0.207	0.249	0.191

Table 6.9: Class-wise deletion AUC for mean representations of explanation distributions obtained using FER+ and GBP.

FER+ GBP: Deletion metric for standard deviation representation

The lowest/best AUC is 0.173 for the *fear* class with MC Dropout. The highest/worst AUC is 0.416 for the *neutral* class when using MC Dropout.

Table 6.11 and Table 6.12 show the insertion metrics for the mean and the standard deviation representation respectively when applying GBP on the FER+ images.

UQ	Contempt	Sadness	Neutral	Surprise	Anger	Happy	Fear	Disgust
DE	0.299	0.245	0.303	0.271	0.290	0.363	0.231	0.198
MCDO	0.324	0.302	**0.416**	0.187	0.306	0.196	**0.173**	0.220
MCDC	0.299	0.259	0.284	0.234	0.242	0.363	0.178	0.236
F	0.288	0.272	0.308	0.238	0.234	0.229	0.245	0.213

Table 6.10: Class-wise deletion AUC for standard deviation representations of explanation distributions obtained using FER+ and GBP.

FER+ GBP: Insertion metric for mean representation

The lowest/worst AUC is 0.163 for the *fear* class when using MC DropConnect. The highest/best AUC is 0.501 for the *neutral* class when using MC Dropout.

UQ	Contempt	Sadness	Neutral	Surprise	Anger	Happy	Fear	Disgust
DE	0.264	0.249	0.280	0.324	0.323	0.395	0.279	0.326
MCDO	0.454	0.369	**0.501**	0.395	0.459	0.407	0.288	0.391
MCDC	0.226	0.232	0.235	0.216	0.216	0.278	**0.163**	0.194
F	0.303	0.310	0.320	0.463	0.259	0.325	0.377	0.327

Table 6.11: Class-wise insertion AUC for mean representations of explanation distributions obtained using FER+ and GBP.

FER+ GBP: Insertion metric for standard deviation representation

The lowest/worst AUC is 0.164 for the *fear* class when using MC DropConnect. The highest/best AUC is 0.487 for the *neutral* class when using MC Dropout.

UQ	Contempt	Sadness	Neutral	Surprise	Anger	Happy	Fear	Disgust
DE	0.247	0.251	0.287	0.327	0.322	0.396	0.279	0.332
MCDO	0.463	0.355	**0.487**	0.410	0.481	0.418	0.299	0.360
MCDC	0.219	0.229	0.235	0.217	0.217	0.282	**0.164**	0.193
F	0.293	0.313	0.321	0.456	0.261	0.318	0.392	0.284

Table 6.12: Class-wise insertion AUC for standard deviation representations of explanation distributions obtained using FER+ and GBP.

Table 6.13 and Table 6.14 present the deletion metrics for the mean and the standard deviation representation respectively generated by applying IG on a batch of FER+ images.

FER+ IG: Deletion metric for mean representation

The lowest/best AUC is 0.193 for the *happy* class when using Flipout. The highest/worst AUC is 0.416 for the *neutral* class when using the MC Dropout.

UQ	Contempt	Sadness	Neutral	Surprise	Anger	Happy	Fear	Disgust
DE	0.305	0.267	0.329	0.360	0.337	0.343	0.332	0.326
MCDO	0.358	0.290	**0.416**	0.247	0.363	0.214	0.196	0.292
MCDC	0.234	0.246	0.275	0.232	0.218	0.283	0.224	0.205
F	0.236	0.291	0.320	0.305	0.248	**0.193**	0.289	0.311

Table 6.13: Class-wise deletion AUC for mean representations of explanation distributions obtained using FER+ and IG.

FER+ IG: Deletion metric for standard deviation representation

The lowest/best AUC is 0.187 for the *disgust* class when using MC DropConnect whereas the highest/worst AUC is 0.391 for the *neutral* when using MC Dropout.

UQ	Contempt	Sadness	Neutral	Surprise	Anger	Happy	Fear	Disgust
DE	0.290	0.261	0.316	0.342	0.335	0.356	0.318	0.321
MCDO	0.369	0.281	**0.391**	0.257	0.366	0.241	0.214	0.322
MCDC	0.231	0.245	0.278	0.230	0.215	0.280	0.232	**0.187**
F	0.328	0.322	0.352	0.335	0.266	0.201	0.338	0.337

Table 6.14: Class-wise deletion AUC for standard deviation representations of explanation distributions obtained using FER+ and IG.

Table 6.15 and Table 6.16 depicts the insertion metrics for the mean and the standard deviation explanation representations respectively generated by applying IG on FER+ images.

FER+ IG: Insertion metric for mean representation

The lowest/worst AUC is 0.270 for the *anger* class generated using MC DropConnect. The highest/best AUC is 0.485 for the *surprise* class when using Flipout.

UQ	Contempt	Sadness	Neutral	Surprise	Anger	Happy	Fear	Disgust
DE	0.315	0.286	0.350	0.391	0.346	0.424	0.327	0.318
MCDO	0.452	0.354	0.483	0.427	0.448	0.452	0.337	0.420
MCDC	0.315	0.300	0.312	0.330	**0.270**	0.387	0.279	0.282
F	0.328	0.360	0.398	**0.485**	0.295	0.454	0.377	0.323

Table 6.15: Class-wise insertion AUC for mean representations of explanation distributions obtained using FER+ and IG.

FER+ IG: Insertion metric for standard deviation representation

The lowest/worst AUC is 0.274 obtained for the *fear* class when using MC DropConnect. The higest/best AUC is 0.487 for *neutral* class when using MC Dropout.

UQ	Contempt	Sadness	Neutral	Surprise	Anger	Happy	Fear	Disgust
DE	0.299	0.279	0.339	0.385	0.343	0.423	0.323	0.321
MCDO	0.478	0.351	**0.487**	0.434	0.457	0.463	0.349	0.402
MCDC	0.304	0.302	0.314	0.328	0.276	0.384	**0.274**	0.299
F	0.366	0.360	0.403	0.481	0.298	0.485	0.376	0.334

Table 6.16: Class-wise insertion AUC for standard deviation representations of explanation distributions obtained using FER+ and IG.

From the results shown in Table 6.9 - Table 6.16, we observe that the lowest/best AUC for deletion is 0.123 for the *fear* expression class (mean representation obtained using MC Dropout and GBP) whereas the highest/worst AUC for deletion is 0.477 for *contempt* expression class (mean representation obtained using MC Dropout and GBP). In a similar manner, the lowest/worst AUC for insertion is 0.163 for *fear* class (mean representation obtained using MC DropConnect and GBP) and the highest/best AUC for insertion is 0.501 for the *neutral* class (mean representation obtained using MC Dropout and GBP). The *fear* expression class has the best performance for pixel deletion; however, fares the worst for pixel insertion.

From the analysis conducted above, it can be inferred that the pixel deletion/insertion metric indicates poor quality of explanation representation on the FER+ dataset. We claim this by observing the range of values of the deletion and insertion AUC. For instance, the lowest deletion AUC for CIFAR-10 is 0.139 and the highest insertion AUC for this dataset is 0.931. Conversely, the lowest and the highest AUC value for the FER+ dataset are 0.123 and 0.501 respectively. This is an expected behaviour as the quality of explanation is limited by the quality of prediction and we already know that FER+ is a difficult dataset to train on.

Summarizing the results of Experiment 2

In the discussion of results of Experiment 2, we identify the best and worst performing class based on AUC along with the representation and combination used to produce the value. We do this for both the datasets. The key takeaway from the analysis conducted in this section is that applying the pixel deletion/insertion on the mean and the standard deviation representation of explanation distribution essentially quantifies the quality of "agreement" and "disagreement" amongst the constituent explanation instances respectively.

6.3 Experiment 3: Sanity Checks for the Explanation Methods

In this section, we present the findings of the Experiment 3 conducted in this thesis. The objective of this experiment is to conduct the sanity checks on the explanation methods selected for implementation. The underlying principles of the sanity checks have been provided in the previous chapter. Each subsection in the following text discusses the results of one sanity check. As we have used two explanation methods for image classification task, namely IG and GBP, we need to apply these checks to both these approaches. Additionally, the dataset and the uncertainty estimation configuration used for these checks are CIFAR-10 and MC Dropout respectively. The rationale behind selecting CIFAR-10 is that the feature space representing this dataset and the class coverage is diverse resulting in straightforward analysis of the resulting explanation heatmaps. Additionally, any uncertainty estimation can be used in these tests and the selection of MC Dropout is arbitrary. Following are the explanation configurations that are used as sample in these sanity checks:

- CIFAR-10, MC Dropout, IG, mean
- CIFAR-10, MC Dropout, IG, standard deviation
- CIFAR-10, MC Dropout, GBP, mean
- CIFAR-10, MC Dropout, GBP, standard deviation

The changes in above listed explanations are analysed before and after application of sanity checks. Based on the analysis results and criteria of the sanity check, we conclude if the explanation method under consideration passes/fails that particular sanity check.

Weight randomization test

As discussed in the previous chapter, the objective of this sanity check is to observe the changes in the explanation (for a given input image) when the model weights are randomized progressively. The variation in the heatmaps by sequential perturbation of weights has been presented in Figure 6.9 for IG and Figure 6.10 for GBP. It should be noted, the amount of weight randomization increases from 0% to 100% as we go from left to right. The first columns of these figures depict an undisturbed model (weight randomization=0%) and the final columns depict a completely randomized model (weight randomization=100%). The first row in these figures depicts the input image under consideration along

with the edge image. The edge image is obtained by application of *canny* edge detector functionality of *OpenCV* library. As expected, the edge image highlights the edges and the contrasting boundary regions of the input image. The second, third and final row of these figures depict changes in the mean, the standard deviation and the coefficient of variation representation of the explanation distribution respectively as the weight randomization increases. From the Figure 6.9 and Figure 6.10, following observations can be made:

- **Mean:** As the weight randomization increases in the model, the maximum of the mean visualization falls for both IG and GBP. This can be observed from the maxima of the color bar adjacent to the heatmaps. This could be attributed to the increasing "disagreement" among the constituent explanation instances as the weight randomization increases. This is reinforced by the fact that the mean visualization of explanation distribution is gradually dispersed as the weight randomization increases and the explanation does not highlight the relevant pixels clearly.

- **Standard deviation:** Interestingly, the maximum of the standard deviation goes up only for GBP. Again, this could be attributed to the increasing "disagreement" among the constituent explanation instances. As the weight randomization increases, the location of important pixels is spread across the entire image and the heatmap no longer focuses on the class relevant pixels (in this case, dog). The increment in the maximum of standard deviation visualization is not observed for IG.

- **Coefficient of variation:** For the case of coefficient of variation visualization of the explanation distribution, it can be observed that as the weight randomization increases, the pixels highlighting the important features spread across the entire image. In other words, the explanation heatmap corresponding to the coefficient of variation becomes more noisy with increasing weight randomization. The effect is more pronounced for the case of GBP where we can see from Figure 6.10 that for the increasing weight randomization, the explanation becomes more and more noisy. This is relatively less visible for the case of IG (Figure 6.9).

As discussed in previous chapter, we compute the Structural Similarity Index Measure (SSIM) to quantify the deterioration of the explanation heatmap with progressive weight randomization. Here, we describe how this quantity is used to determine if a given explanation method passes/fails the weight randomization sanity check. We assume the heatmap generated by undisturbed model to be the *baseline*. Next, we compute the SSIM of explanation maps generated by subsequent weight randomization steps with the baseline heatmap. Each weight randomization step will generate a value for SSIM. We plot this value for the mean and the standard deviation visualization for both the explanation methods. Figure 6.11a and Figure 6.11b depict the variation of SSIM for the mean and standard deviation respectively of the explanation distribution for both the explanation methods. Ideally, this should be a monotonically decreasing function as subsequent weight randomization should continuously deteriorate the similarity of the generated explanation with the baseline. The value of SSIM progressively decreases as the weight randomization increases for both the visualizations (mean and standard deviation) and for both the explanation methods. This signifies that with the model weights being perturbed progressively, the generated explanation also changes (becomes less similar to the baseline). This observation demonstrates

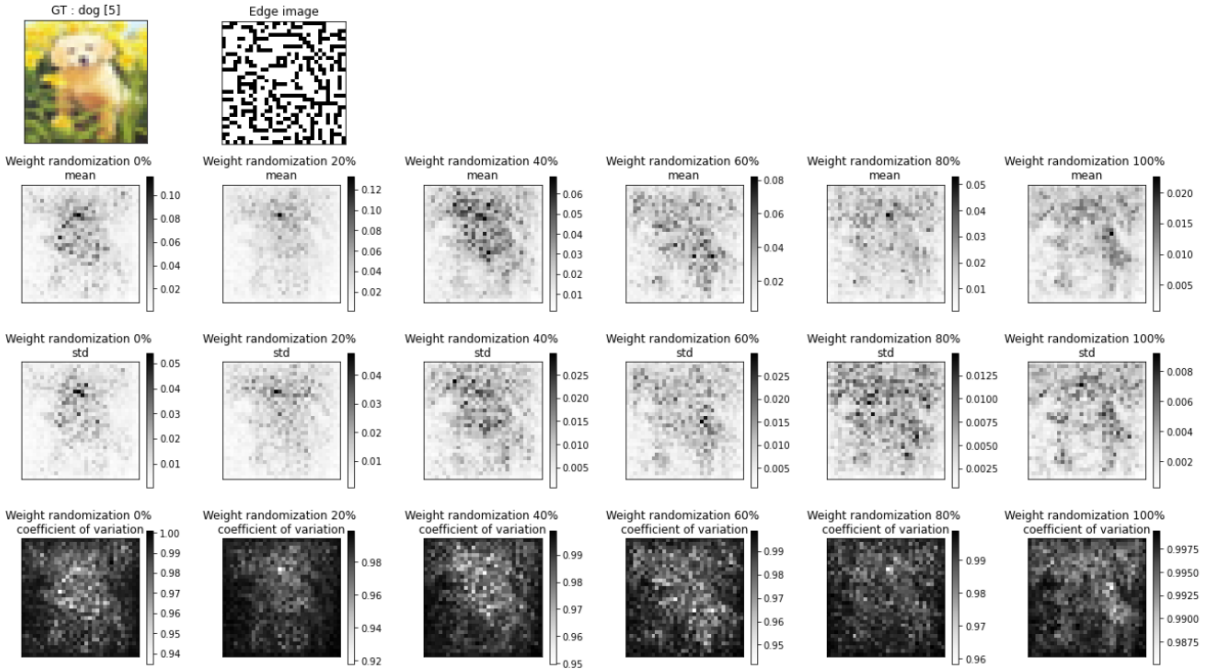


Figure 6.9: Visualizing the effect of weight randomization on the concise explanation representations obtained using IG. The input and the corresponding edge image are provided for reference.

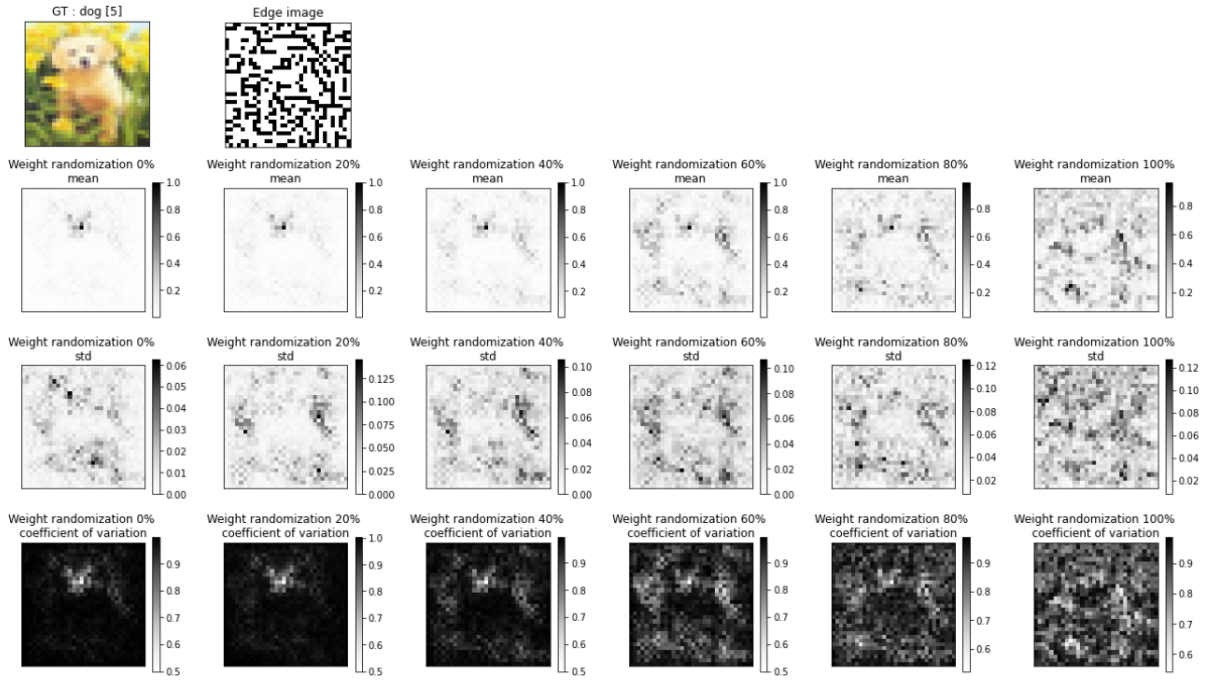
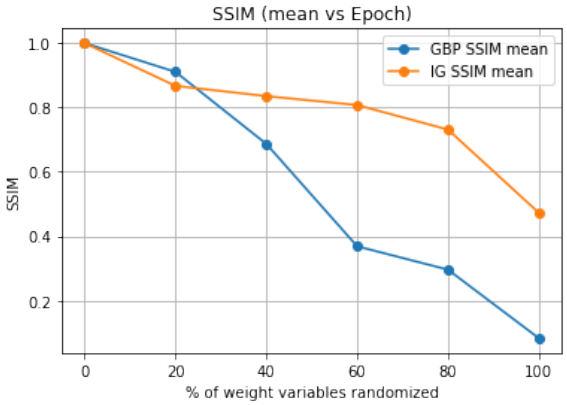
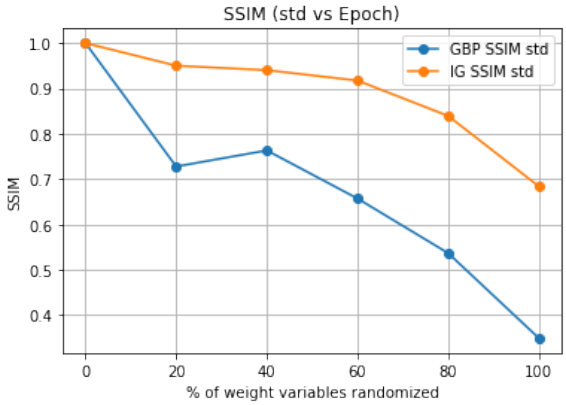


Figure 6.10: Visualizing the effect of weight randomization on the concise explanation representations obtained using GBP. The input and the corresponding edge image are provided for reference.



(a) SSIM for the mean representation.



(b) SSIM for the standard deviation representation.

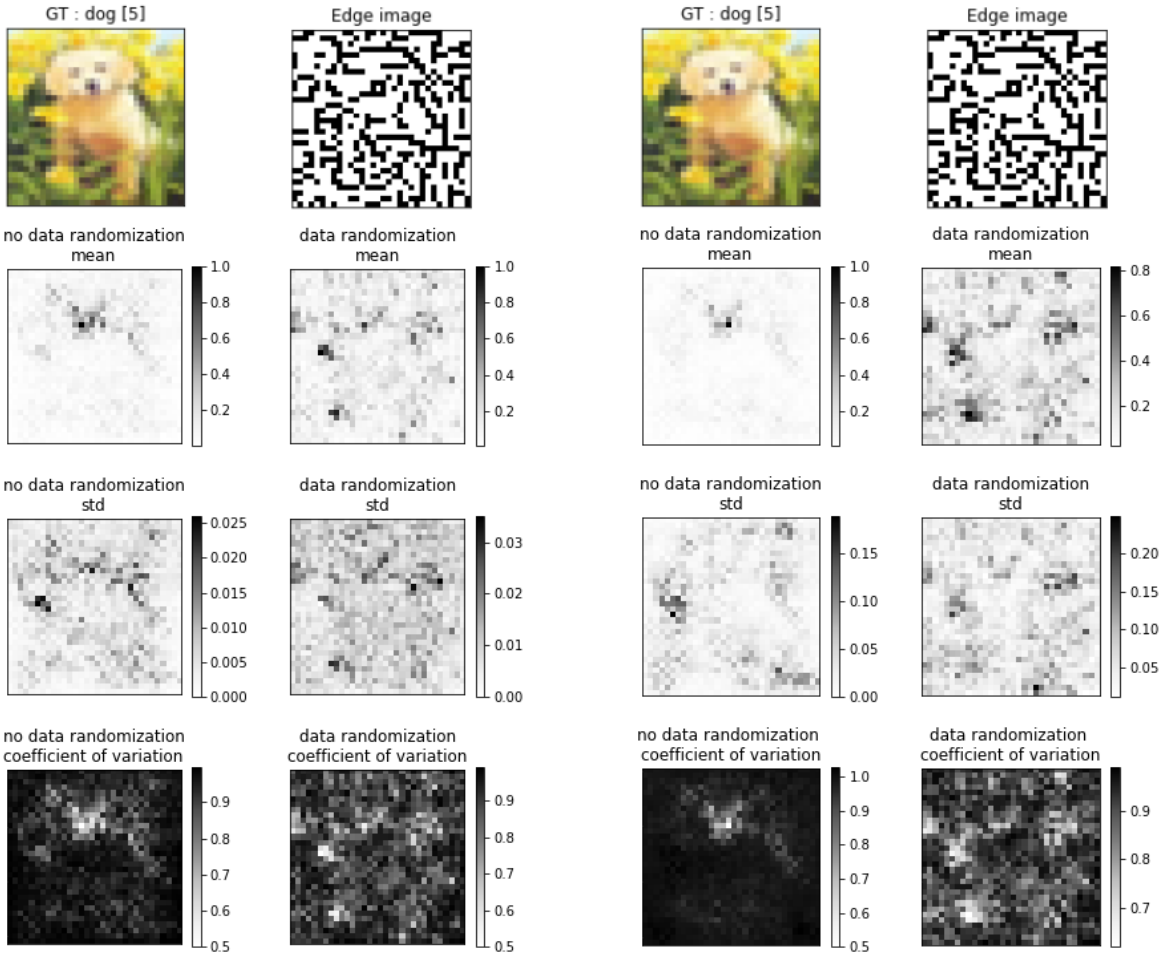
Figure 6.11: The variation in the SSIM values for the weight randomization sanity check. These values are computed between the explanation representation generated with no weight randomization and with incremental weight randomization. As the amount of weight randomization increases, the similarity between the explanation representation decreases.

that the explanation methods do *not* simply extract the edges and present them as the explanation. If that were the case, the subsequent explanations should not have changed significantly with increased weight perturbation. This proves that both the explanation methods *pass* the weight randomization test in our context.

Data randomization test

From our previous discussions, we know that the objective of this check is to ensure that the explanation method utilises the image-label mapping in order to generate explanations. This means that on disturbing the image-label mapping, the explanation for a given image must change as well. If it does not, then the explanation method only utilises the information present in the image and disregards the label to generate the feature relevance map. Figure 6.12a and Figure 6.12b depict the findings of this sanity check for IG and GBP respectively. The first row of both these figures depicts the input image and the edge extracted image. The second, third and the final row depict the mean, the standard deviation and the coefficient of variation visualization of the explanation distribution. The explanations depicted in the left column (second row onward) are generated with a model trained with correct labels. The ones in the second column (second row onward) are generated using a model trained on shuffled labels. We analyse the changes induced due to the shuffling of image-label mapping in each of the aforementioned explanation representation individually:

- **Mean:** With the properly trained model, both the IG and GBP approach generate sensible explanations. The face of the dog is highlighted correctly. As the image-label mapping is shuffled, the resulting explanations are incoherent and provide no insights about the relevant feature. A possible reason for this could be that on shuffling the train dataset, the model is no longer learning



(a) Data randomization check for IG.

(b) Data randomization check for GBP.

Figure 6.12: Visualizing the effect of data randomization on the concise explanation representations. The input and the corresponding edge image are provided for reference.

Explanation Method	Mean	Standard Deviation
IG	0.401	0.894
GBP	0.252	0.510

Table 6.17: The SSIM values for the data randomization sanity check. These values are computed between the explanation representation with no data randomization and with data randomization. The explanation representations are the mean and the standard deviation of the explanation distributions.

relevant features and is basically misplacing the pixel importance.

- **Standard deviation:** With the introduction of data randomization, the maximum value of the standard deviation visualization goes up for both the IG and GBP. This could be attributed to increased amount of "disagreement" in the constituent explanation instances. As the instances disagree, the maximum value of standard deviation visualization goes up.
- **Coefficient of variation:** Prior to randomizing the train dataset, the coefficient of variation visualization is concentrated and highlights only the relevant image regions. As the data randomization is introduced, the coefficient of variation representation also scatters. This effect is more pronounced in the case of GBP as can be seen in the last row of Figure 6.12b.

We now attempt to quantify the differences between the baseline and the explanation heatmaps introduced by the randomization of image-label mapping. For this purpose, we assume the explanations generated by the no data randomization case to be the *baseline*. Next, we compute the SSIM between the baseline and the explanations generated by introduction of data randomization to the process. If the explanations are not affected by the data randomization, the value of SSIM will be close to 1 (identical inputs). This would mean that the explanation method is not using the image-label mapping rather using only the information from the image. The explanation method thus would not pass the sanity check. Using the approach discussed above, we compute the SSIM and present it in a tabular format. Table 6.17 provides these values of SSIM. The difference between the baseline and the explanation from the data randomization is evident from the value of the SSIM. Hence, the both the explanation methods *pass* this sanity check.

Effects of epochs

In the previous chapter, we describe the underlying principle of this sanity check. The objective is to observe the changes in the explanation heatmap (for a given input image) as the model training progresses. Ideally for later epochs, the explanation should highlight relevant features and should be different from the explanation of earlier epochs. Figure 6.13 and Figure 6.14 depict the variation in the explanation for IG and GBP respectively as the model training progresses. As can be seen from the figures, the epochs increase as we go from left to right with the first columns showing the explanation at epoch=1 and the final columns depicting explanation at the best performing epoch. Similar to weight randomization sanity

check, the first row depicts the input image and the corresponding edge extracted image. The second, third and the fourth row depict the evolution of the mean, the standard deviation and the coefficient of variation of the explanation distribution respectively as the model training progresses. Following observations can be made from Figure 6.13 and Figure 6.14:

- **Mean:** As the model training progresses, the maximum of the mean visualization for GBP increases. This could be due to the reason that as the model training progresses, the generated explanations begin to focus on the relevant features rather than paying attention to irrelevant features. The increment in the maximum of the mean visualization is not observed in the case of IG. A possible reason for this could be that the explanations generated for IG for this particular image are not accumulating on the relevant features (class "dog" in this example), hence not boosting the maximum value for the mean visualization.

- **Standard deviation:** The maximum of the standard deviation visualization falls for the case of GBP. This could be attributed to the fact that more explanation instances from the distribution agree upon common features and drive down the standard deviation visualization maximum as the training progresses. However, this is not the case for IG as the standard deviation visualization maximum actually increases as the training progresses. Again this could be attributed possibly to the fact that IG might not be focussing on relevant features in the constituent explanation instances and as a result driving up the standard deviation visualization maximum.

- **Coefficient of variation:** For the case of GBP, the variation in explanation can be seen quite clearly. As the model training progresses, the coefficient of variation visualization of explanation distribution focuses on the face of the dog. This is not quite the case for IG. The coefficient of variation visualization highlights the pixel on the body of the dog and the evolution of explanation is not as apparent across epochs.

We apply the SSIM as before to quantify the similarity/dissimilarity between the *baseline* and the subsequent explanations. In this case, we assign the baseline to be the explanation obtained using the best performing epoch. We then compute the SSIM of explanation maps generated across epochs with the baseline heatmap. Next, we plot the SSIM obtained for the mean and the standard deviation visualization for both the explanation methods on two different plots. Figure 6.15a and Figure 6.15b depict the variation of SSIM across epochs. For an early epoch, the SSIM between the explanation and the baseline is low. This is expected as the network is not yet fully trained and the explanation method is still focussing on less relevant features. Hence, the difference between the baseline and the explanation generated for an early epoch is stark. As the training progresses, the generated explanations become more and more similar to the baseline (the network learns the relevant features). In an ideal situation, the SSIM for this setting should be a monotonically increasing function. From Figure 6.15a and Figure 6.15b, we can observe that these curves are indeed increasing functions. Since the value of SSIM increases for both the explanation method and both the visualizations (mean and standard deviation), we can conclude that both the explanation methods indeed *pass* this sanity check.

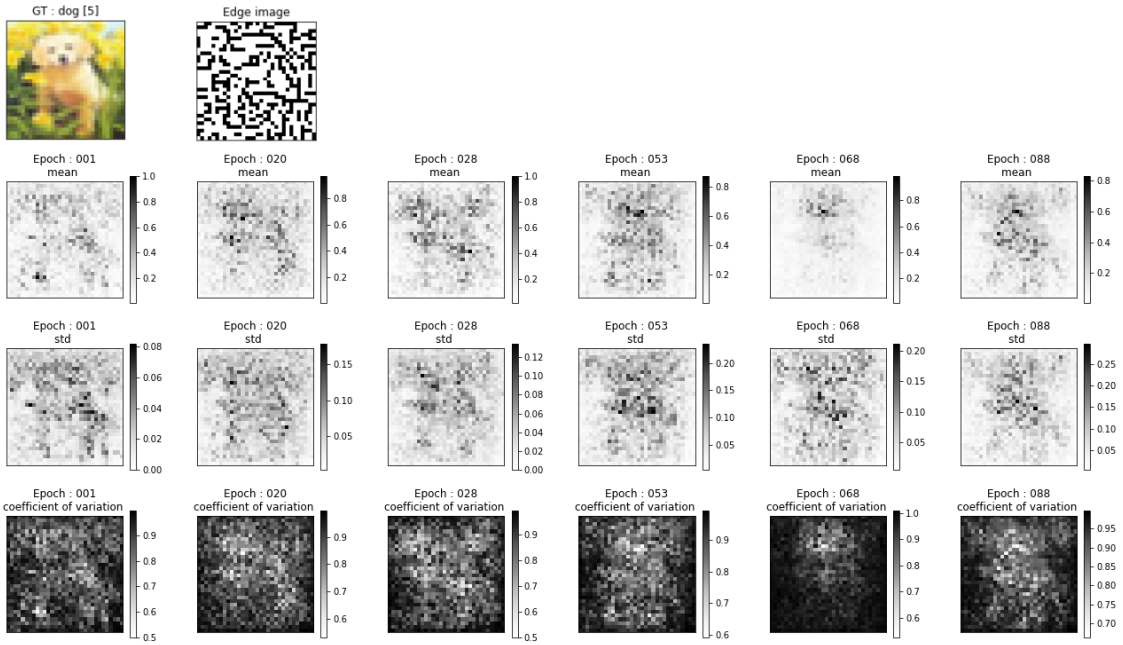


Figure 6.13: Visualizing the effect of epochs on the concise explanation representations obtained using IG. The input and the corresponding edge image are provided for reference.

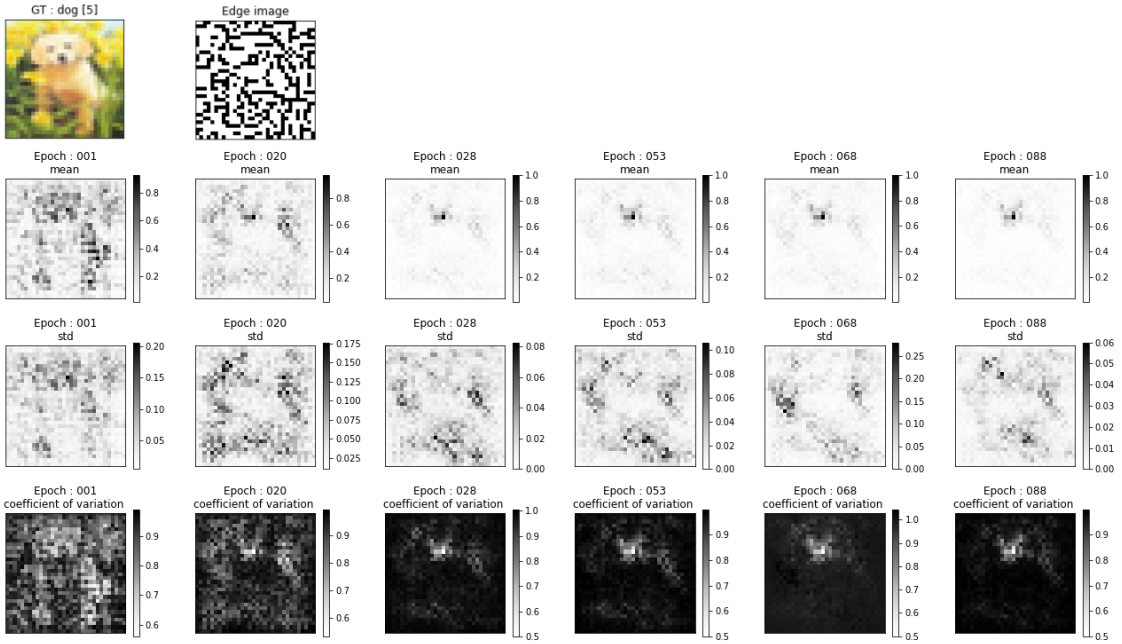
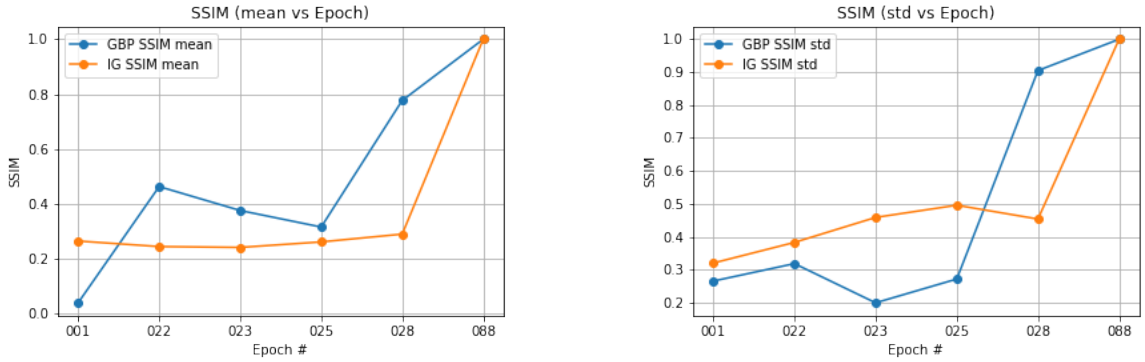


Figure 6.14: Visualizing the effect of epochs on the concise explanation representations obtained using GBP. The input and the corresponding edge image are provided for reference.



(a) SSIM for the mean representation.

(b) SSIM for the standard deviation representation.

Figure 6.15: The variation in the SSIM values for the effect of epoch sanity check. These values are computed between the explanation representation generated at the latest epoch and at selected earlier epochs from the model training. As the epoch value decreases, the similarity between the explanation representation also decreases.

Summarizing the results of Experiment 3

From the discussion of the results and the ensuing analysis, we conclude that in context of this thesis, both the explanation methods (IG and GBP) *pass* the sanity checks namely (i) the weight randomization check, (ii) the data randomization check, and (iii) effect of epoch check, described in the previous chapter. Hence, their use in generation of an explanation distribution is justified.

6.4 Experiment 4: Variation in Uncertainty of Explanation for FER+

In this section, we present the results of the Experiment 4. In the previous chapter, we discuss the details and objective of this experiment. The results are presented in Figure 6.16 - Figure 6.19 and the format of presentation is similar to Experiment 1. The key difference between both these experiments has been described in Figure 5.16. Building upon discussion conducted previously, we expect the output of this experiment to differ from the output of Experiment 1 only when the network prediction does *not* match the ground truth. Next, we analyse the results depicted in individual figure provided above:

- From Figure 6.16 it can be seen that the explanation representations are almost identical to the ones depicted in Figure 6.5. This happens as the network prediction matches the ground truth resulting in the explanation instances (from which these representations are computed) to be almost identical. A slight variation in the pixel values of these representation can be clearly observed. This is to be expected and can be attributed to the stochastic nature of the networks we are working with.

- As is with the previous example, the representations of Figure 6.17 appear almost identical to that shown in Figure 6.6. The network predicts the expression correctly and hence the explanation representation generated from the instances is almost similar.

- For the input image shown in Figure 6.18, a few of the model prediction for the case of Flipout do not agree with the ground truth. Applying the formula described in this experiment for this case results in explanation instances with pixel value as *nan*. In order to generate a meaningful visualization, we specifically drop those explanation instances with *nan* and compute the concise representation. A significant difference between Figure 6.7 and Figure 6.18 can be observed for the case of standard deviation representation generated by GBP and MC DropConnect.
- In case of Figure 6.19, differences can be observed for the standard deviation representation generated by the following combinations: GBP+MC Dropout and GBP+MC DropConnect. The pixel values are slightly different for the equivalent explanation representation for the aforementioned combinations. As with the previous example, this can be attributed to the stochastic nature of the network. In other words, even if the predicted class matches for two stochastic networks, the underlying score can differ which results in slightly different explanation instances.

Summarizing the results of Experiment 4

The key takeaways from the analysis of results conducted in the previous section are:

(a) It is possible that entirely different regions/pixel patches are highlighted as being relevant in both the representations (Experiment 1 and Experiment 4). The reason supporting this claim is the different formula are used for computation of the explanation in case the prediction does *not* match the ground truth. An extreme case of this claim explained in the description of Figure 6.18, where using the Experiment 4 formula to compute the explanation yields heatmaps with *nan* values.

(b) The values of the mean, the standard deviation and the coefficient of variation representations might differ across equivalent explanation representation even when the prediction matches the ground truth. This means that pixels at same spatial orientation in the explanation representation (for Experiment 1 and Experiment 4) have slightly different relevance. The reason behind this is that as the models are stochastic, even using the same mathematical formula (as the formula for Experiment 1 and Experiment 4 converge when the prediction matches the ground truth) to compute explanation might result in slightly different outcomes as the prediction score might vary from model instance to model instance.

6.5 Experiment 5: Uncertainty of Explanation for Numerical Regression

In this experiment, we aim to test the robustness of the pipeline (implemented in Experiment 1) for a numerical regression task. Hence, the output of this task would be concise representation of explanations as well. Figure 6.20 - Figure 6.23 depict the concise representations obtained during this experiment for two example inputs. We analyse the result of each input example individually. First, we describe the format of these figures in order to facilitate the analysis of the findings. In all the figures listed above, the first column corresponds to the representations obtained using GBP and the second column corresponds to those obtained using LIME. Similarly, the rows correspond to the uncertainty estimation method

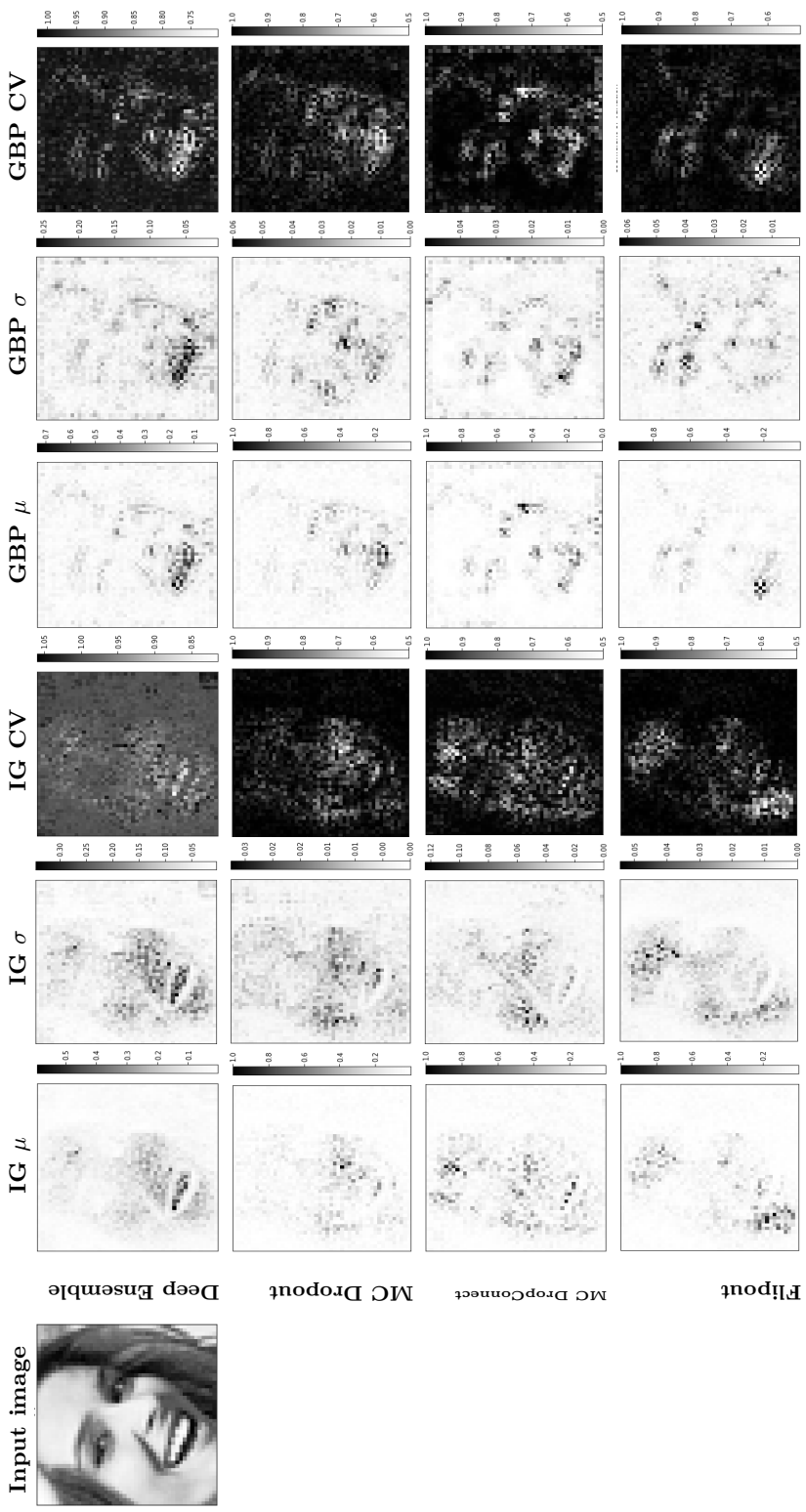


Figure 6.16: Input image of a happy person taken from the FER+. The columns represent the concise representations generated by using the two explanation methods. The rows correspond to different uncertainty estimation approaches.

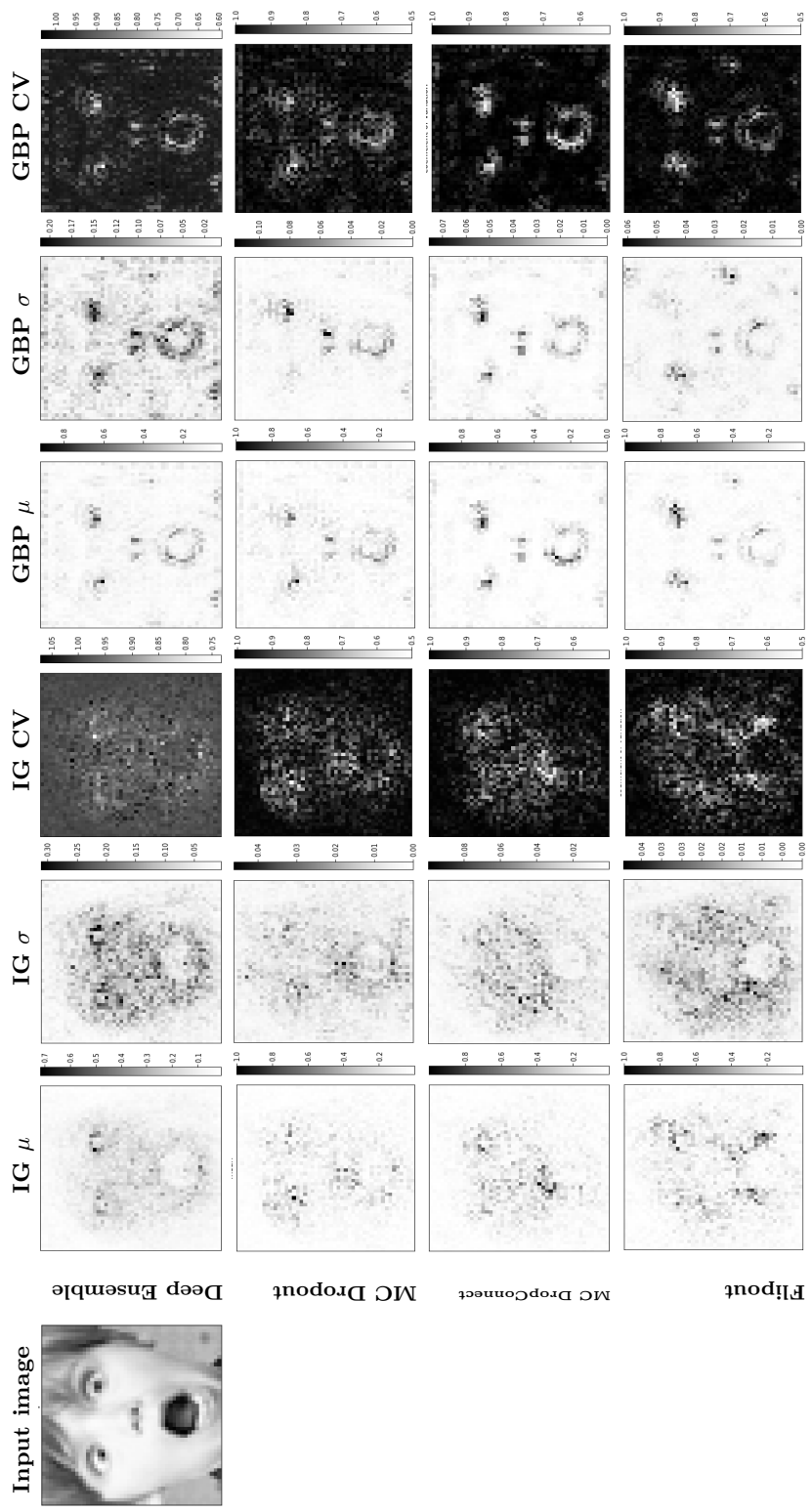


Figure 6.17: Input image of a surprised child taken from the FER+. The columns represent the concise representations generated by using the two explanation methods. The rows correspond to different uncertainty estimation approaches.

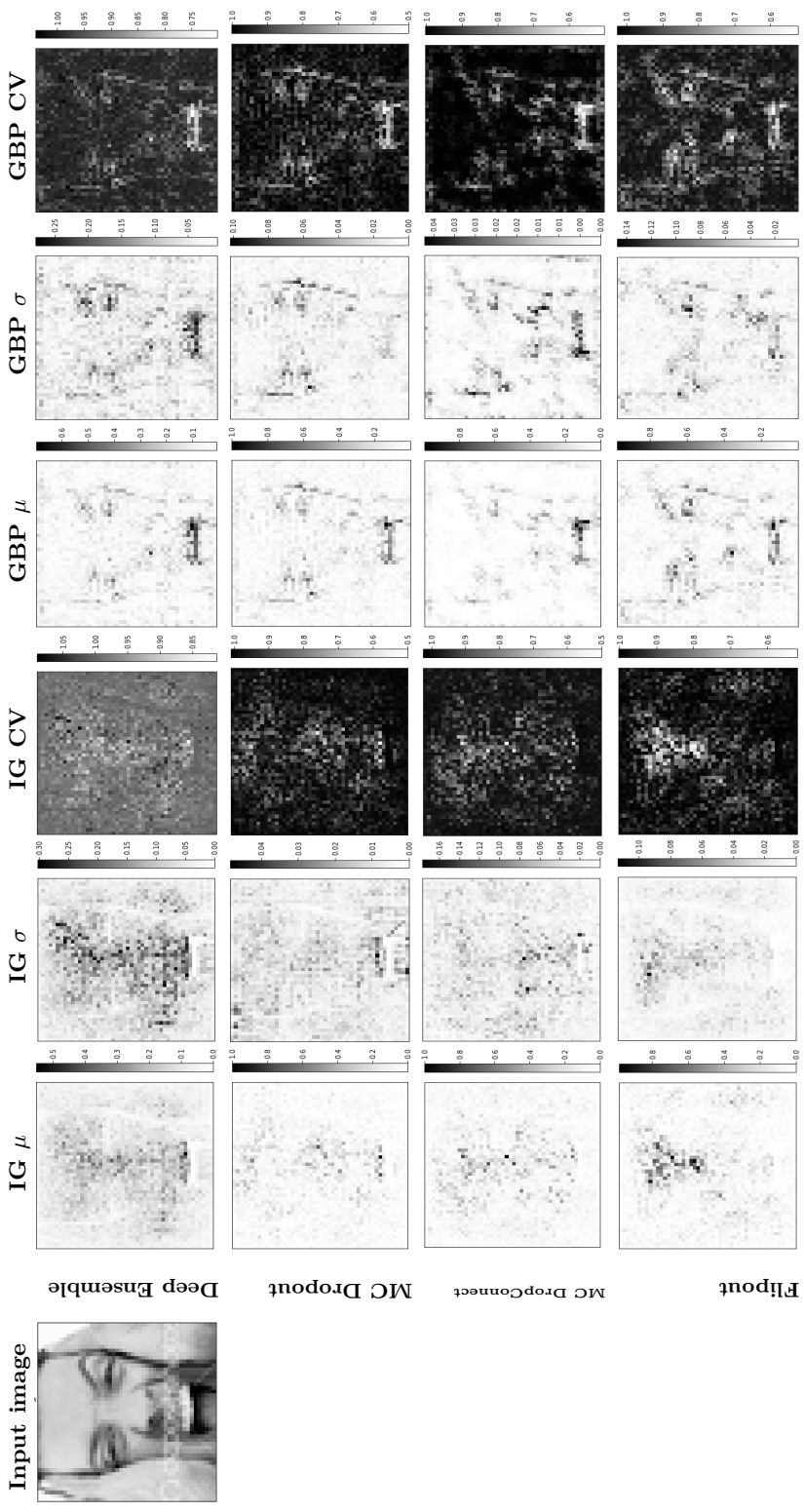


Figure 6.18: Input image of an angry person taken from the FER+. The columns represent the concise representations generated by using the two explanation methods. The rows correspond to different uncertainty estimation approaches.

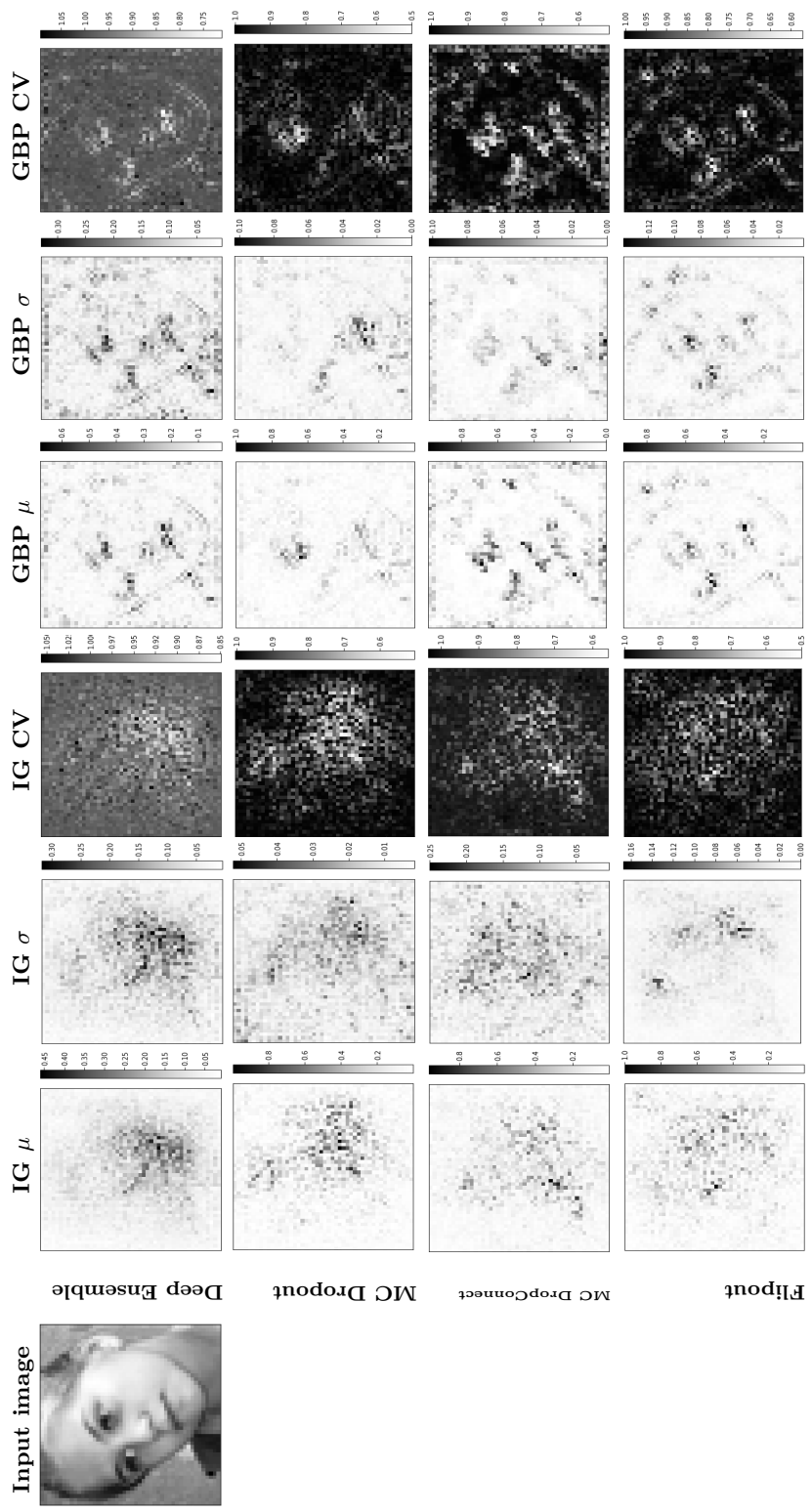


Figure 6.19: Input image of a person with neutral expression taken from the FER+. The columns represent the concise representations generated by using the two explanation methods. The rows correspond to different uncertainty estimation approaches.

with the following order being observed row-wise: Deep Ensemble (DE), MC Dropout (MCDO), MC DropConnect (MCDC) and, Flipout (F). For instance, the mean and the standard deviation explanation representation in the 2^{nd} row and 1^{st} column of Figure 6.20 is generated using the combination GBP + MC Dropout. In contrast to the results of Experiment 1, here we depict both the mean and standard deviation representation in a single diagram.

- The concise representations for the first example are presented in Figure 6.20 (the mean and standard deviation representation) and Figure 6.21 (the coefficient of variation representation). In Figure 6.20, the length of the individual bar signifies the mean relevance of that particular feature and the half-length of the error bar depicts the standard deviation of the relevance. Positive relevance signifies that these features have a positive coefficient in the surrogate model that is used to fit the train data sampled in the neighborhood of the input to be explained. A similar logic applies to the negative relevance as well. An interesting quantity to analyse is the amount of uncertainty in the relevance values for different combinations. To do this, we make use of the coefficient of variation plot provided in Figure 6.21. A low value of this quantity (or high value of mean or low value of standard deviation) indicates high amount of agreement amongst the constituent explanation instances and thus lower uncertainty. It can be seen that generally the coefficient of variation values are low for the case of LIME as compared to GBP for all combinations except for the one with Flipout. Also, the coefficient of variation values are high for the combinations having Deep Ensemble when compared to other combinations.
- For the second example, the concise visualization have been presented in Figure 6.22 (the mean and the standard deviation) and Figure 6.23 (the coefficient of variation). The format of explanation is the same as in previous example. In this case, the coefficient of variation values for GBP + Flipout are low (barring feature *total_rooms*). It should be noted that in this example, the coefficient of variation values for GBP are lower as compared to LIME except for the combination with MC Dropout. For the coefficient of variation visualization of LIME in this particular example, one feature has been assigned unusually high relevance for all the uncertainty estimation method except Flipout. This is not the case of GBP, which assigns comparable relevance values to the features for all combinations except Flipout.

From the explanation representations of the examples above, it can been observed that for the same feature, both explanation methods are assigning contrasting relevance score. For instance, if we analyse the *latitude* feature from the second example for the MC Dropout, the relevance assigned by GBP is -0.075±0.086 whereas by LIME is 0.114±0.008. This is difficult to justify, a plausible reason for such phenomenon could be (*i*) stochasticity in the network affects the prediction that in turn affects the explanation instances, (*ii*) hyperparameters of LIME (choice of surrogate, neighborhood for sampling and specification of sampling LIME) contributes towards the different explanation as well.

Summarizing the results of Experiment 5

The key insights obtained from this experiments have been listed below:

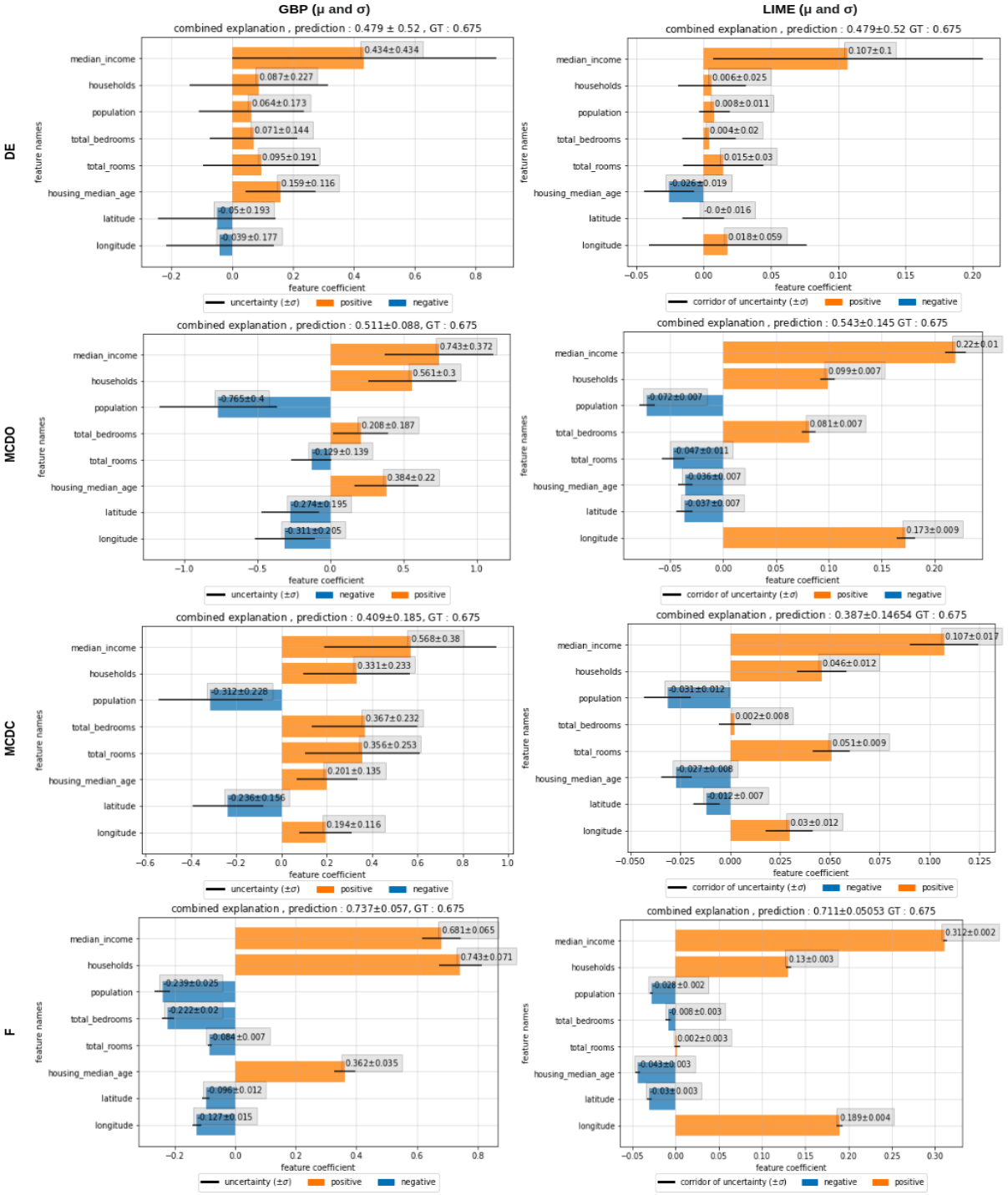


Figure 6.20: The mean and the standard deviation explanation representation for an input belonging to the California Housing dataset. The columns of the figure correspond to the explanation method and the rows map to the uncertainty estimation methods. The value of the input features are: [longitude: 0.219, latitude: 0.513, housing median age: 0.509, total rooms: 0.127, total bedrooms: 0.121, population: 0.128, households: 0.122, median income: 0.421].

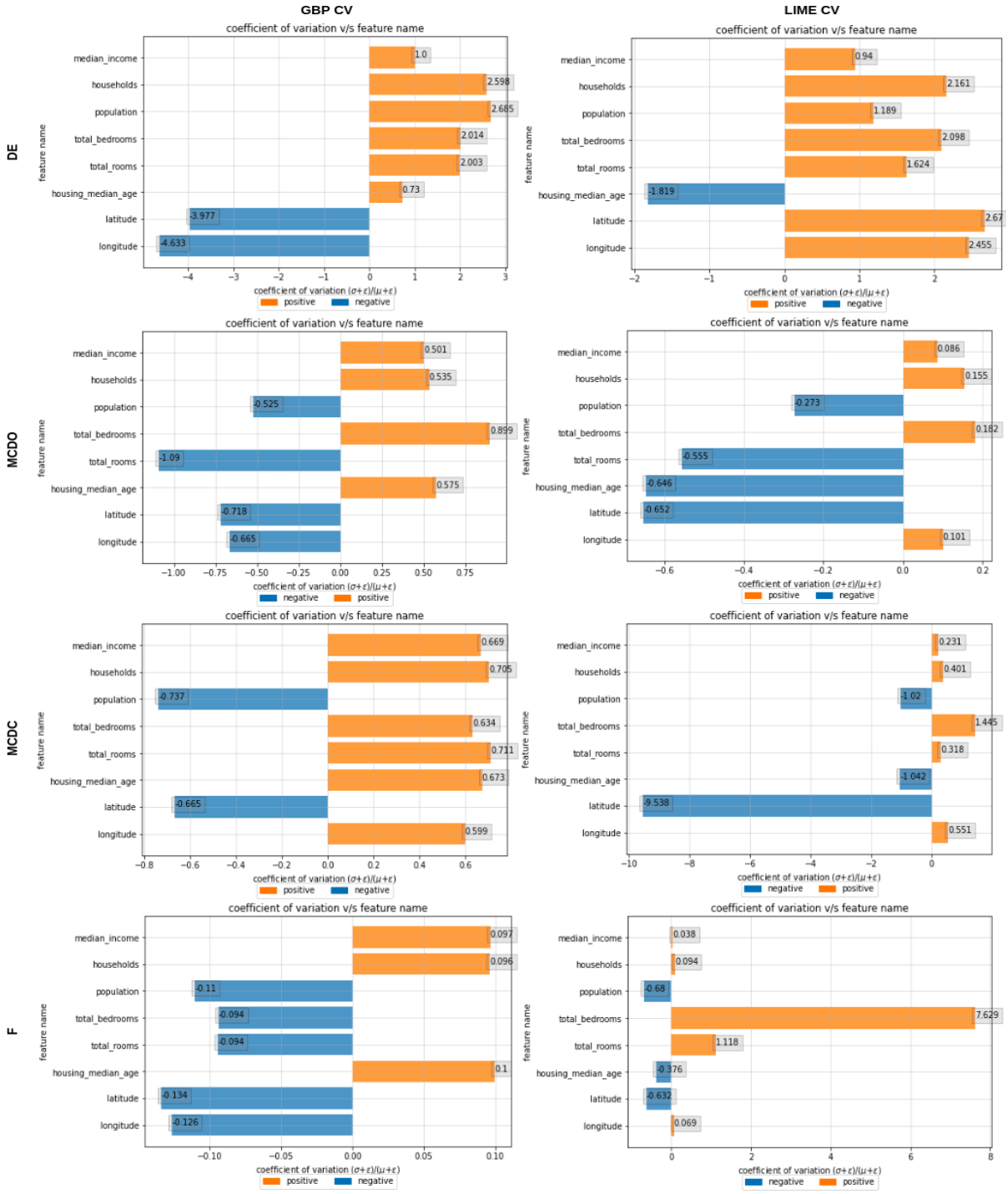


Figure 6.21: The coefficient of variation explanation representation for the input analysed in Figure 6.20. Similar to the previous figure, the columns and the rows of this figure map to explanation and uncertainty estimation methods respectively.

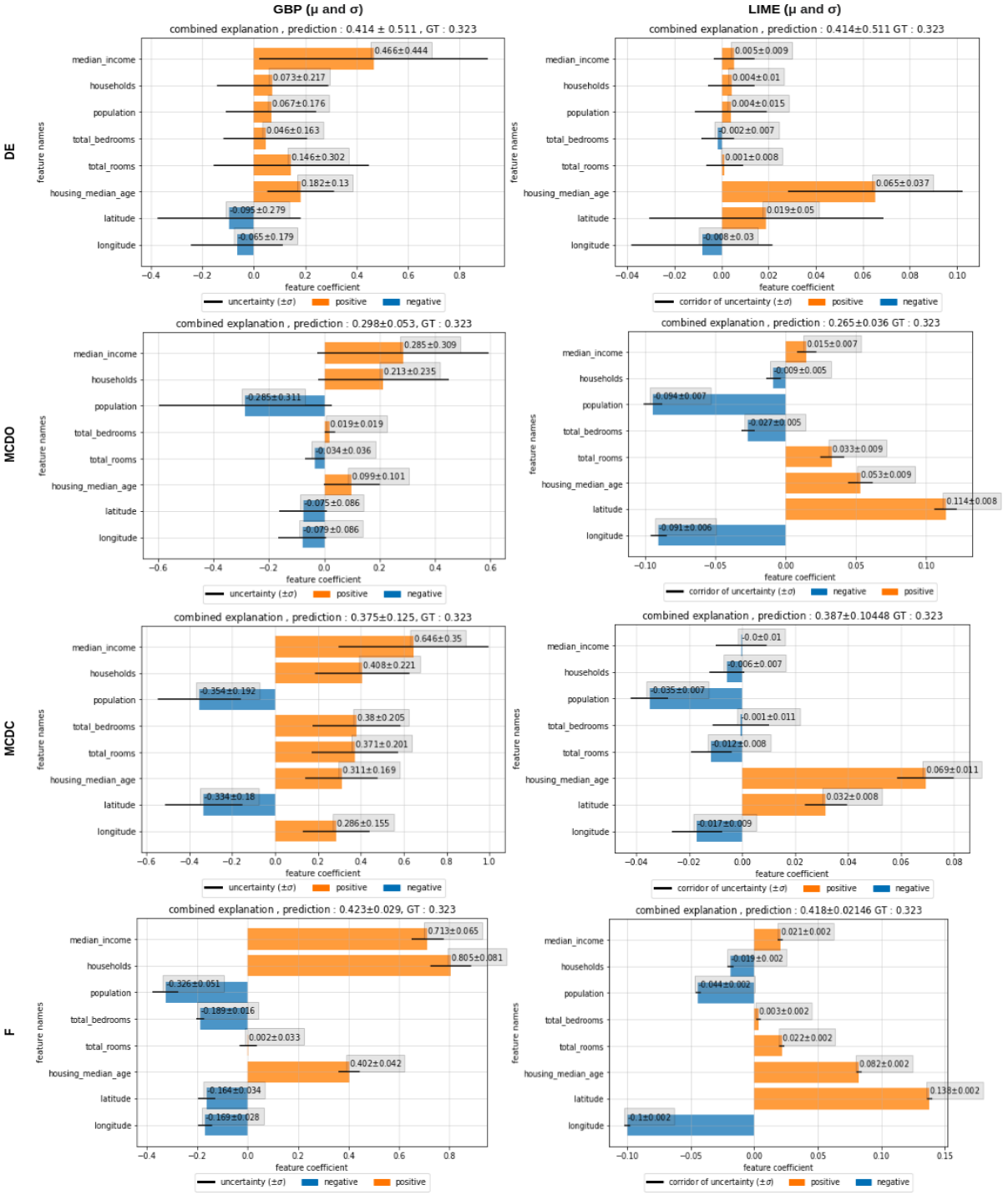


Figure 6.22: The mean and the standard deviation explanation representation for an input belonging to the California Housing dataset. The columns of the figure correspond to the explanation method and the rows map to the uncertainty estimation methods. The value of the input features are: [longitude: 0.606, latitude: 0.181, housing median age: 0.823, total rooms: 0.049, total bedrooms: 0.056, population: 0.067, households: 0.055, median income: 0.213].

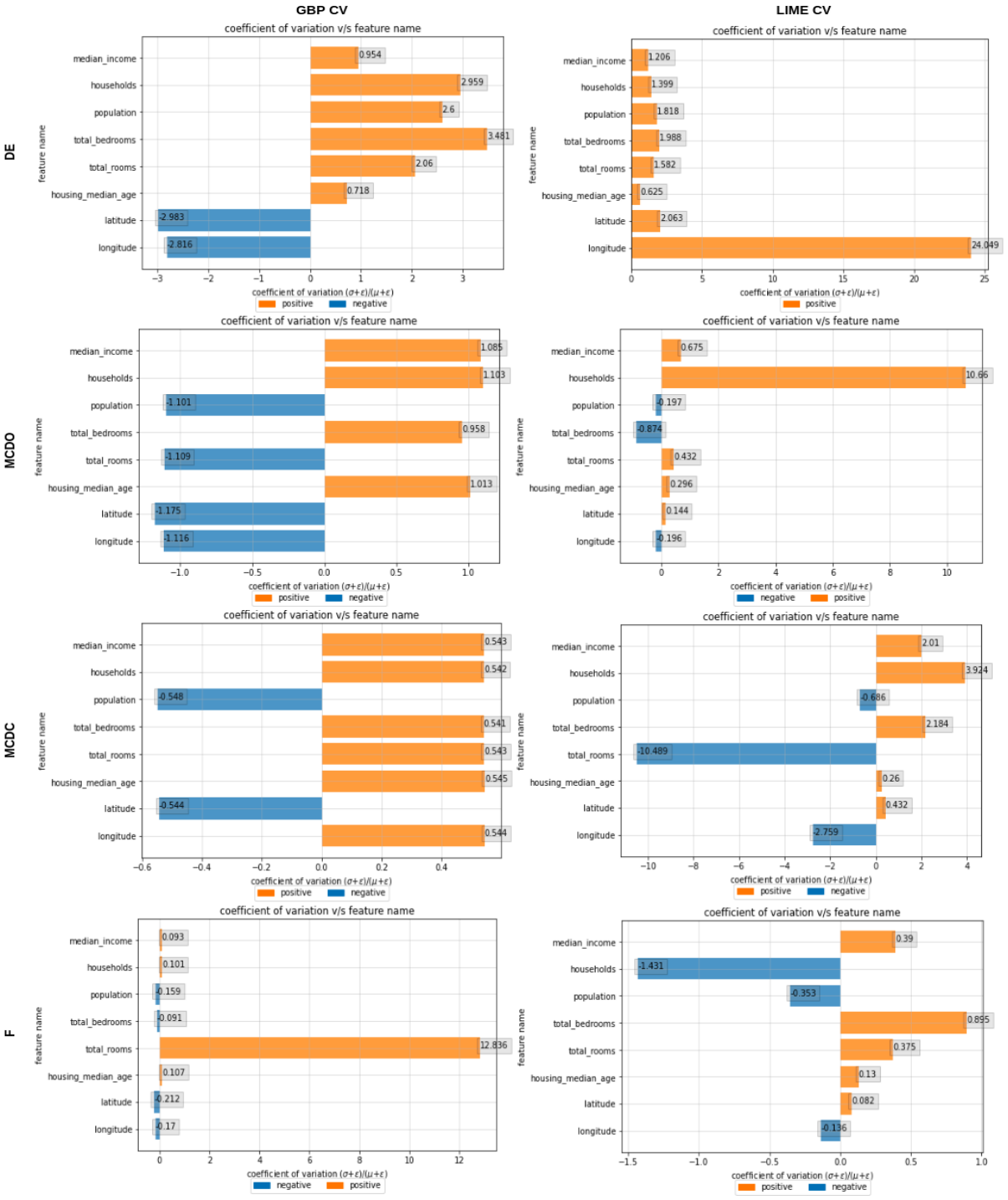


Figure 6.23: The coefficient of variation explanation representation for the input analysed in Figure 6.22. Similar to the previous figure, the columns and the rows map to explanation and uncertainty estimation methods respectively.

- The uncertainty associated with the explanation for the combinations with Flipout are generally low as compared to other combinations. This is inferred from the coefficient of variation values plotted in the analysis. A possible reason for this observation could be that Flipout generate low output variance.
- A few instances where the same feature has been assigned very different relevance score by both the explanation methods for the same uncertainty estimation method. Difficult to determine which relevance score is correct owing to lack of ground truth or evaluation method for the task at hand.
- During the implementation process, we observed that LIME has many settings that could result in an unstable explanation such as:
  - definition of neighborhood around a given input (to be explained) from which to create the perturbation dataset.
  - definition of surrogate model that is used to explain the complicated model in a defined neighborhood
  - number of points to be sampled around a data point to be explained.

6.6 Summary

In this section, we provide an overview of the discussion conducted in this chapter. The summary of the discussions in each of the previous section is as follows:

- Experiment 1: We present the results obtained from different combinations of uncertainty estimation and explanation methods. We analyse four examples from each dataset along with their concise explanation representation. It is observed that the mean representations capture the agreement amongst the constituent explanation instances whereas the standard deviation representations provide insights into the disagreement amongst the explanation instances. The coefficient of variation representation allows us to analyse the mean and the standard deviation representations simultaneously and can also be used to estimate the uncertainty in the feature importance.
- Experiment 2: We attempt to quantify the quality of the mean and the standard deviation representation generated from Experiment 1. The pixel deletion/insertion metric is used to identify the best and worst performing class for the implemented combinations. It is found that combinations having GBP as their explanation component generally perform better than combinations having IG.
- Experiment 3: We discuss the results of applying the sanity checks to IG and GBP. Using the SSIM values to quantify the similarity in explanation representations, it is found that both these methods pass all the three sanity checks.
- Experiment 4: We present the results and the subsequent observations of Experiment 4 wherein we modify the gradient computation step for the FER+ dataset. We observe that for cases when the network output does not match the ground truth, the modified explanation generation approach can offer additional insights about the features/regions where the model should have assigned relevance.

- Experiment 5: The results of applying the proposed pipeline to a numerical regression task have been discussed along with insights drawn from the analysis. These results show the robustness of the proposed pipeline.

7

Conclusions

In the previous chapter, we presented the results of the experiment and its subsequent analysis. In the present chapter, we conclude this thesis by summarising the findings and the contributions of this research, providing the answers to the research questions posed in the earlier chapters, listing the lessons learnt and outlining the scope of the future work.

7.1 Revisiting the Research Questions

In this section, we answer the Research Questions (RQ) formulated in Chapter 1. The answers have been obtained over the course of this project and have been presented here.

RQ1: Could additional combinations of uncertainty estimation methods and explanation methods be implemented to ascertain the uncertainty associated with the explanation of neural networks?

A1: Yes, additional combinations of uncertainty estimation methods and explanation methods can be implemented. This can be observed from Table 7.1. This table highlights the already existing combinations (cells with citations) along with the additional combinations identified and implemented (cells having values R and C) in this thesis. This table can also be expanded by incorporating other uncertainty estimation and explanation methods to identify more combinations in the future.

RQ2: How can the explanations/heatmaps generated by approaches discussed in RQ1 be combined/processed to compute a condensed representation of uncertainty in explanation?

A2: We identify the *mean* and the *standard deviation* as concise representation of the explanation distribution. These could be used to identify the features on which the model is strongly focussing and those features as well where the model is paying attention weakly. Additionally, the *coefficient of variation* (a ratio of the standard deviation and the mean) can also be used as an auxiliary representation. The features/regions where coefficient of variation is low, indicates low standard deviation (or high mean) and vice versa. For instance, a feature/region with low coefficient of variation indicates strong agreement between different explanation instances of the distribution. Conversely, high coefficient of variation for a particular feature/region would indicate strong disagreement among different explanation instances. This quantity could also be used to gauge the amount of

uncertainty in the explanation, the higher the coefficient of variation, the higher the uncertainty in the explanation.

RQ3: What are the possible approaches to analyse the concise explanation heatmap generated by the methods discussed in RQ2?

A3: Owing to the lack of reasonable ground truth, it is difficult to analyse the quality of generated explanations. Researchers have worked to address this problem and proposed a few solutions. In this thesis, we utilise the pixel deletion and pixel insertion metric (proposed in [54]) to analyse the concise explanation representations. The underlying principle is to perturb (either by adding relevant pixels or by removing relevant pixels) and observe the changes in the prediction score. The details of this approach and the modifications made to suit this metric to our use have been discussed in Chapter 4. Another possible approach identified from the literature survey is that of human evaluation. This approach involves conducting a survey and recording the responses for a predefined set of questions based on inputs and their generated explanations.

RQ4: Which combination of uncertainty estimation method and explanation method identified in RQ1 is the best to analyse the uncertainty associated with the explanation method output for a neural network?

A4: From visual analysis of explanation representations for the image datasets, we conjecture that any combination of uncertainty estimation method and explanation method having GBP can produce useful explanation distributions. A supporting observation for this statement is that generally GBP tends to yield less noisier visualization as compared to IG. This helps in distinguishing the certain features from those that are less certain. Furthermore, from the pixel deletion/insertion analysis conducted in Experiment 2, we quantify the quality of explanation representations generated by different combinations of uncertainty estimation and explanation methods. A possible answer from this analysis for the best observed combination is that of MC Dropout with GBP as this combination achieves the lowest AUC (0.123) for pixel deletion amongst all the combinations discussed in Table 6.1 - Table 6.16. Another combination could be that of Flipout with GBP as this has the highest AUC (0.931) for pixel insertion amongst all the combinations discussed in Table 6.1 - Table 6.16. These observations also serve to reinforce the claim that any combination with GBP as its explanation component could be used to produce meaningful explanation distributions.

7.2 Contributions

In the subsequent text, we summarize the contributions made by this research work. For the purpose of simplicity, the contributions are presented in the form of a list as:

- We conduct a systematic literature review of the approaches developed to analyse the uncertainty in the explanation of a neural network. In doing so, we also analyse research works that solely discuss the methods that deal with uncertainty estimation in neural networks and research papers that propose explanation approaches for neural networks.

Explanation Method → Uncertainty Estimation Method ↓	**LIME**	**SHAP**	**LRP**	**GBP**	**IG**
Bayesian Weighted Least Square	[72]	[72]	-	-	-
Repeated generation of explanation to analyse the variation	[99]	-	-	-	-
Deep Ensembles	R	-	[19]	C, R	C
Monte Carlo Dropout	R	-	[18], [19]	[93], C, R	C
Monte Carlo DropConnect	R	-	-	C, R	C
Flipout	R	-	-	C, R	C

Table 7.1: A summary of the combinations of uncertainty estimation and explanation methods implemented by contemporary research works along with the combinations implemented in this thesis. The combinations explored in this thesis have been depicted by cells having C and R. The C highlights the combinations implemented for image classification task and R shows those implemented for numerical regression task.

- Identify and implement new combinations of uncertainty estimation and explanation methods. These combinations have been highlighted in Table 7.1. The additional combinations are implemented using a pipeline described in Chapter 4.

- Originally, the sanity checks have been applied to individual explanation instances. In this work, we modify the implementation of these checks such that they can be used to analyse distributions of explanations. Particularly, we customize the sanity checks such that the mean and the standard deviation of the explanation distribution are subjected to the verification (as opposed to individual explanation instance).

- We customize the pixel deletion and insertion metric such that it can be used to quantify the performance of explanation distribution. Originally, these metrics have been used to quantify the quality of a single explanation instance. In this thesis, we use these metrics to quantify the quality of the mean and the standard deviation of the explanation distribution.

- Tested an alternative of generating explanations by modifying the gradient computation step. We apply this approach on FER+ dataset and compare the generated explanation representations.

- We apply GBP to explain tabular data for the task of numerical regression in this thesis. Usually, GBP is applied exclusively to explain the inputs in the form of images.

7.3 Lessons learnt

In this section, we discuss the lessons learnt, observations made and the insights obtained during the course of this project.

- It is possible to obtain uncertainty in explanation. To do so, we utilise a distribution of prediction to induce a distribution of explanation. From this explanation distribution, we compute the mean and the standard deviation that provides an estimate of uncertainty in the explanations.

- FER+ is a difficult dataset to train upon. This could be attributed to the inherent difficulty encountered during the data labeling process. Human emotions/expressions are subjective quantities and are difficult to judge. A "happy" face for one annotator might be that of "surprise" for another. This anomaly results in inconsistent labels for images belonging to the same expression class and subsequently poor network performance. Labeling using strategies like majority voting solve the problem but only to an extent.

- The available feature space to learn in FER+ is limited to the facial region. The classes of expressions are differentiated on the basis of subtle differences in this small subset of features. This could also have an impact on performance of networks on this dataset and subsequently the quality of explanations of the dataset.

- After training the networks using different configurations, we found that the networks using *flipout_dense* and *dropconnect_dense* layers take longer to converge as compared to the training the models either using *stochastic_dropout* layers or as Deep Ensembles.

- Analysing the standard deviation visualization of the explanation distribution, we observe that Flipout tends to generate low variance in the output. This observation has been made in the results of both the tasks (image classification and numerical regression) and all the three datasets (CIFAR-10, FER+ and California Housing).

- For a given input image, GBP is likely to generate a less noisier explanation as compared to IG. A possible reason for this could be that IG might introduce noise in the averaging step that could corrupt the final explanation. No such interference is introduced in the computation of explanation in GBP.

- Combinations with LIME as explanation method might be computationally expensive as this method requires sampling and training a surrogate model in the neighborhood of a given input for a single explanation instance. Generating an explanation distribution of similar instances is bound to consume more resources. This might a potential shortcoming of combinations using LIME.

- Deep Ensembles are rigid to scaling. When working with Deep Ensembles, if we need to generate an explanation distribution with more number of samples than we already have, we would need to train additional ensemble components in order to do so. This is not the case for MC Dropout, MC DropConnect and Flipout. In these three scenarios, during the training phase, we learn a posterior distribution of weights which can be used to sample any number of instances. This allows for scaling and saves the efforts for the additional task of training the model again.

7.4 Future work

The research conducted in this work can be extended in order to be more exhaustive and holistic. Following are the possible aspects that could be explored or improved upon in the future:

- As discussed in previous chapters, there are numerous explanation methods that have been developed to explain the working of neural networks. Additionally, there exist several uncertainty estimation methods as well. Similar to the approach we adopted in this thesis to combine these two types of methods to generate an explanation distribution, it is possible to generate many more combinations of uncertainty estimation and explanation methods.
- Research works such as [33], [73] discussed approaches that could be used to improve the performance of explanation methods such as IG. This could be a potential aspect to investigate.
- As an alternative to the pixel deletion/insertion evaluation, a possible method to analyse the performance of explanation method could be human-based evaluation. This could involve conducting a survey amongst people from different backgrounds and recording their responses on a pre-defined set of questions concerning an input image and its explanation. An analysis of the human responses and the algorithm generated explanations could provide potential insights for improvement of explanation methods.
- The network training on the FER+ can be improved further. In this research, the focus was to generate an explanation distribution, hence not a lot of effort was invested in producing a state of the art performance on FER+ dataset. This could be a potential task to work on.
- As the objective of this research work is to test the working of our proposed approach, using datasets having a single class per image suffices. In order to extend this work, datasets having multiple classes in a single image could be used. This could provide further insights into the decision making process of the networks under consideration.
- In this thesis, we extended the existing task of pixel deletion/insertion for the image classification. However, no such evaluation method is available to quantify the quality of explanation for a numerical regression task to the best of our knowledge. A potential task could be to design an approach to evaluate the quality of explanation representations generated from the numerical regression task.

A

Hyperparameter search for MLP

This chapter discusses the hyperparameter search conducted prior to training MLP on the California Housing Dataset for the regression task. The search is conducted using the HParams dashboard[1] that is helpful in identifying an optimal set of hyperparameters from a defined search space.

For the task of regression on housing dataset, the hyperparameter search space is defined in the form of sets as follows:

Hyperparameter	**Search Space**
# of units in a layer	{1, 2, 4, 6, 8}
# of layers	{1, 2, 3, 4, 5}
# of epochs	{10, 50, 100, 150}
optimizer	{'SGD', 'RMSProp', 'Adam'}

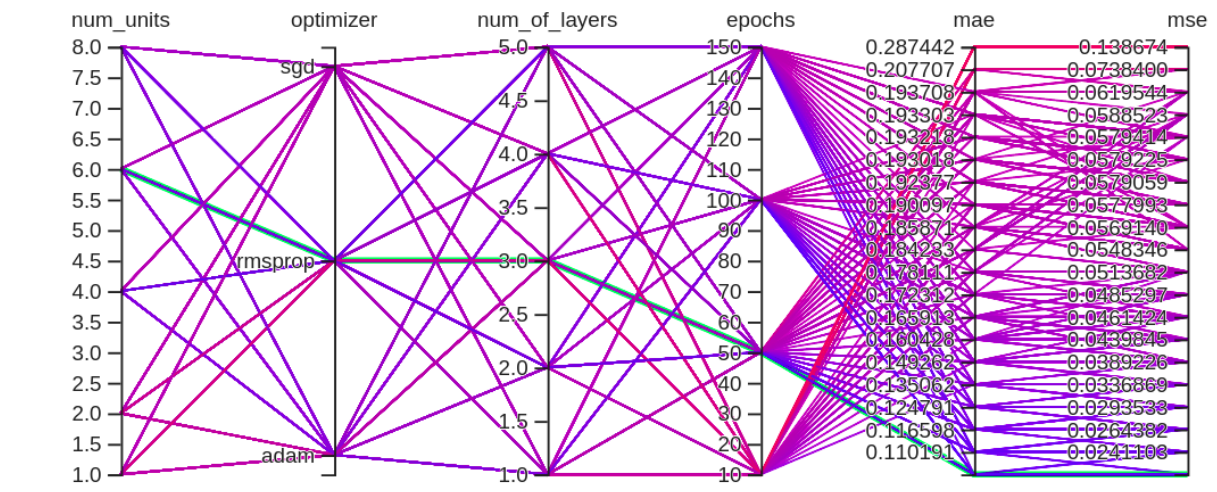
Table A.1: A summary of the search space used for the hyperparameter tuning conducted for the MLP.

From the aforementioned definition of search space, it can be computed that the number of combinations that are possible are:

$$5 \times 5 \times 4 \times 3 = 300 \tag{A.1}$$

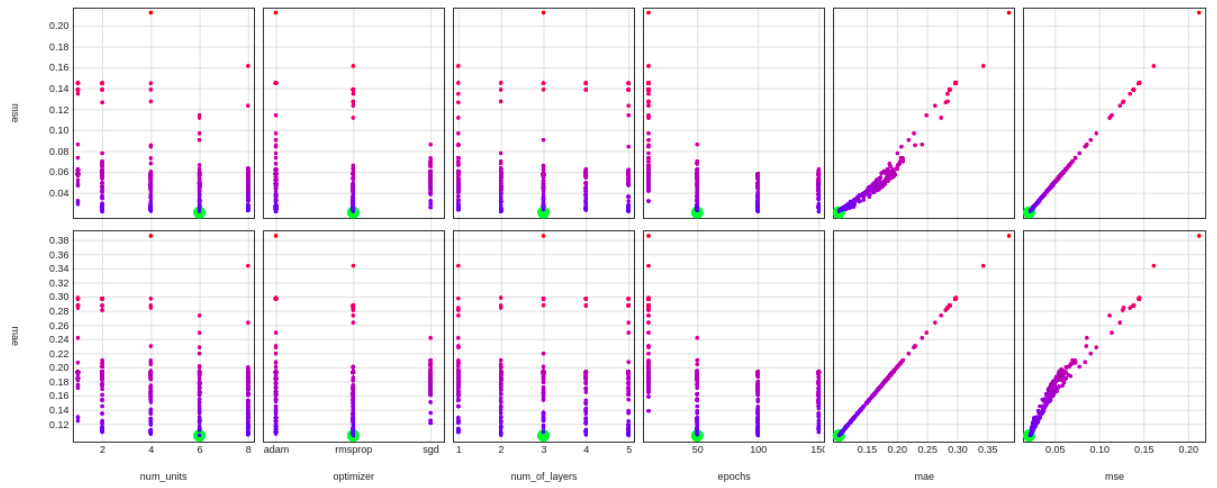
Equation A.1 provides the number of combinations and by extension, the networks that are to be trained to identify the optimum hyperparameter configuration for the defined search space. Figure A.1 shows the parallel plot wherein the best performing combination of hyperparameter in the defined search space is highlighted. This combination of hyperparameter produces the lowest MAE on the test set. Another visualization for the same results in the form of scatter plot is depicted in Figure A.2. The value of individual hyperparameter has been provided in the tabular form at the bottom of both these figures. The hyperparameters obtained from this analysis have been used as a guideline in designing the MLP architecture and its training procedure. It should be noted that it is not necessary to exactly replicate these hyperparameters during training the actual MLP.

[1] `https://www.tensorflow.org/tensorboard/hyperparameter_tuning_with_hparams`



Trial ID	num_units	optimizer	num_of_layers	epochs	mae	mse
a413133f61f16...	6.0000	rmsprop	3.0000	50.000	0.10339	0.021090

Figure A.1: Parallel plot depicting the results of hyperparameter search conducted for creating an MLP. The optimum values of the hyperparameters are provided in the table below the plot.



Trial ID	num_units	optimizer	num_of_layers	epochs	mae	mse
a413133f61f16...	6.0000	rmsprop	3.0000	50.000	0.10339	0.021090

Figure A.2: Scatter plot depicting the results of hyperparameter search conducted for creating an MLP. The optimum values of the hyperparameters are provided in the table below the plot.

B

Network training specifications

This section provides the specifications of the network trainings in this thesis. Additionally, the data augmentations applied for image datasets have also been provided. The discussions of these training configurations have been divided into sections on the basis of the dataset on which the model is trained.

B.1 Training miniVGG on CIFAR-10

In this section, the configurations used for models trained on CIFAR-10 have been covered. A total of 4 models have been trained on this dataset. These models are obtained by modifying the base miniVGG architecture and the details of these structural changes have already been discussed in Chapter 5. Table B.1 presents the hyperparameter settings used for training these 4 models.

Hyperparameter	**Value**
optimizer	adam
loss	categorical cross-entropy
metric	accuracy
learning rate	0.001
train samples	50000
validation samples	9500
test samples	500
train batch size	2048
validation batch size	2048

Table B.1: Hyperparameter settings used for training the miniVGG on CIFAR-10.

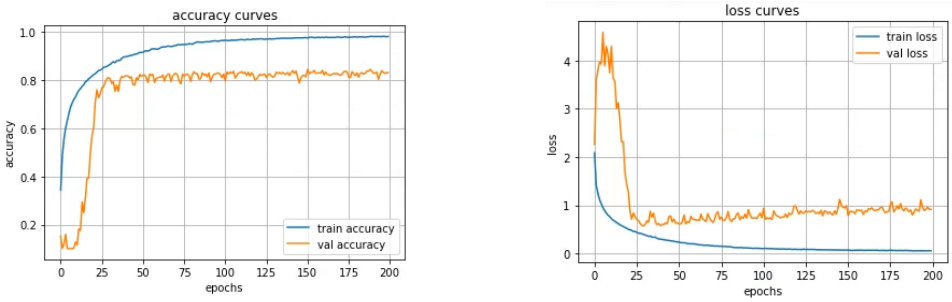
In addition to the these hyperparameter values, the data augmentation values are also provided. It should be noted that the settings for data augmentation listed in Table B.2 are uniformly applied when training the models on CIFAR-10.

Augmentation	Value
width shift range	0.1
height shift range	0.1
shear range	0.1
zoom range	0.1
horizontal flip	True
rescale	1/255

Table B.2: Specifications of data augmentation applied on CIFAR-10.

B.1.1 Deep Ensembles

A total of 10 instances of the miniVGG model are trained for 200 epochs each. All the models are instantiated randomly hence the trajectory of optimization for each will be different. Owing to space constraints in this report, the accuracy and loss curves for only a single ensemble component have been included here. Figure B.1 depicts these curves.



(a) Accuracy v/s Epochs

(b) Loss v/s Epochs

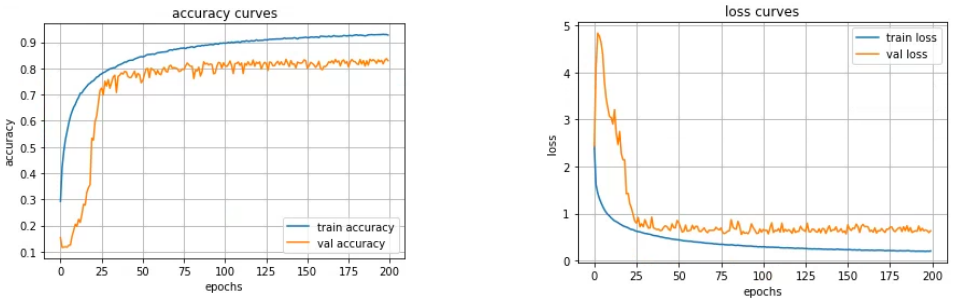
Figure B.1: Visualizing plots obtained by training miniVGG on CIFAR-10 using Deep Ensembles.

It can be seen that the loss curves gradually starts to diverge as the training progresses. This is the reason to terminate the training after 200^{th} epoch. All the 10 model instances are trained for the same number of epochs.

B.1.2 Dropout

Figure B.2 depicts the accuracy and the loss curves obtained when miniVGG is trained on CIFAR-10 using dropout. The probability of dropout is set to 0.25 and the network is trained for 200 epochs.

As the performance does not improve significantly for a number of epochs, the training is terminated after 200 epochs.



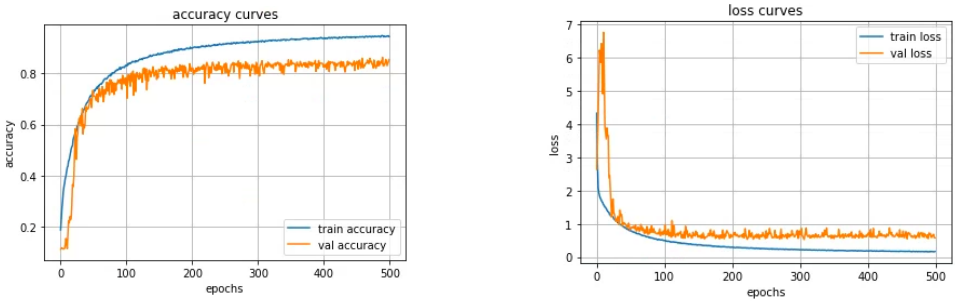
(a) Accuracy v/s Epochs

(b) Loss v/s Epochs

Figure B.2: Visualizing plots obtained by training miniVGG on CIFAR-10 using Dropout.

B.1.3 DropConnect

Figure B.3 shows the accuracy and the loss plots obtained when miniVGG is trained using dropconnect. The model is trained for 500 epochs and the probability for dropconnect is set as 0.25. The number of epochs for this case is set higher as the convergence is relatively slower than previous two cases.



(a) Accuracy v/s Epochs

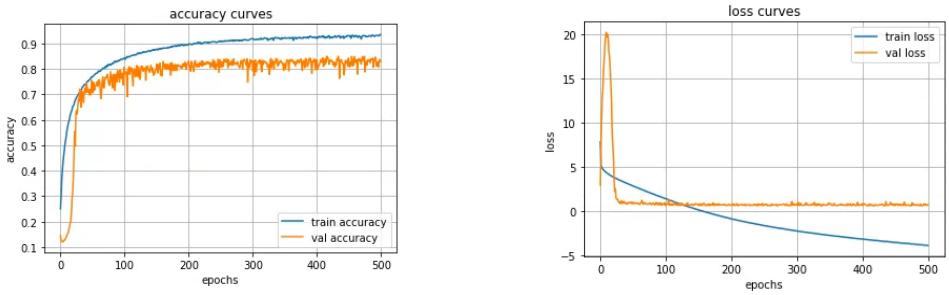
(b) Loss v/s Epochs

Figure B.3: Visualizing plots obtained by training miniVGG on CIFAR-10 using DropConnect.

B.1.4 Flipout

Figure B.4 depicts the accuracy and loss curves obtained when miniVGG is trained on CIFAR-10 with flipout layers. Similar to the previous case, the model is trained for longer as the convergence in this case is slow as well. The probability of flipout is set at 0.1 and the model is trained for 500 epochs. Interestingly, it can be seen in Figure B.4b that the loss goes below 0 after about 100 epochs. The loss metric is categorical cross-entropy and mathematically this should be greater than or equal to 0 (for the configuration used in this training). In the loss plot, it can be seen going below zero. To troubleshoot this, we attempted a number of solutions: (a) checked if the labels of the train set were correctly one hot encoded or not, (b) attempted to train the network without using data generators to test if generators

were the source of errors, (c) analysed the softmax layer output to check if these were causing the loss to behave erratically. Finally, after attempting these solutions, and observing that this phenomenon persists for this setup (but does not replicate for other model trainings), we attribute it to numerical instabilities of this particular configuration.



(a) Accuracy v/s Epochs

(b) Loss v/s Epochs

Figure B.4: Visualizing plots obtained by training miniVGG on CIFAR-10 using Flipout.

B.2 Training miniVGG on FER+

This section provides the specifications of training the models on FER+ dataset. The dataset is available as a *csv* file and the pipeline to load this data has been adopted from the works of [11]. The *csv* file contains the image pixels in vector representation along with their labels and the aforementioned pipeline helps in creating data generators to train the model. The hyperparameters and the data augmentations used on this dataset have been listed in Table B.3 and Table B.4.

Hyperparameter	**Value**
optimizer	adam
loss	categorical cross-entropy
metric	accuracy
learning rate	0.001
train samples	28559
validation samples	3579
test samples	3573
train batch size	2048
validation batch size	2048

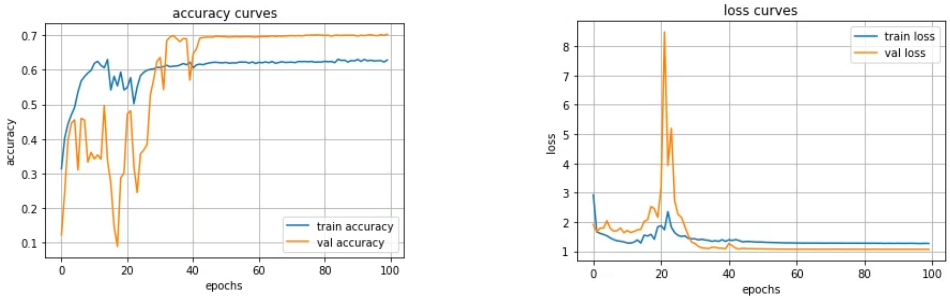
Table B.3: Hyperparameter settings used for training the miniVGG on FER+.

Augmentation	Value
rotation range	30
width shift range	0.1
height shift range	0.1
zoom range	0.1
horizontal flip	True
rescale	1/255

Table B.4: Specifications of data augmentation applied on FER+.

B.2.1 Deep Ensembles

Similar to the case of training the network on CIFAR-10, a total of 10 instances are trained on FER+ as well. As can be observed in Figure B.5, the performance plateaus after 100 epochs, therefore the training is terminated. Similar to the case of CIFAR-10, owing to the space restrictions in this report, the accuracy and the loss plots for a single instance have been depicted here for analysis.



(a) Accuracy v/s Epochs

(b) Loss v/s Epochs

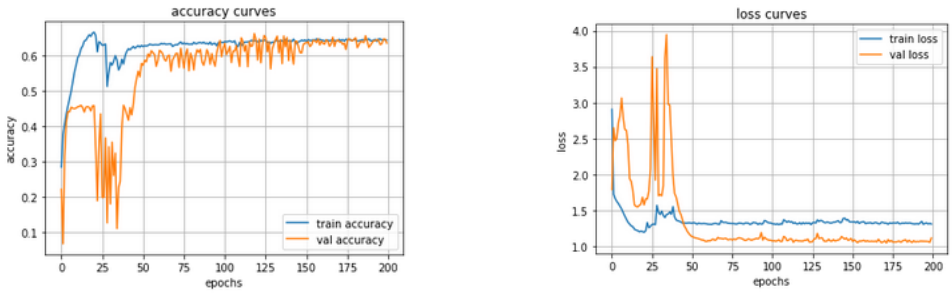
Figure B.5: Visualizing plots obtained by training miniVGG on FER+ using Deep Ensembles.

B.2.2 Dropout

In this case, the convergence is achieved relatively slower when compared to Deep Ensembles. The probability of dropout is set to 0.1 and the training is terminated after 200 epochs. The accuracy and the loss curves obtained after training miniVGG on FER+ using dropout are provided in Figure B.6.

B.2.3 DropConnect

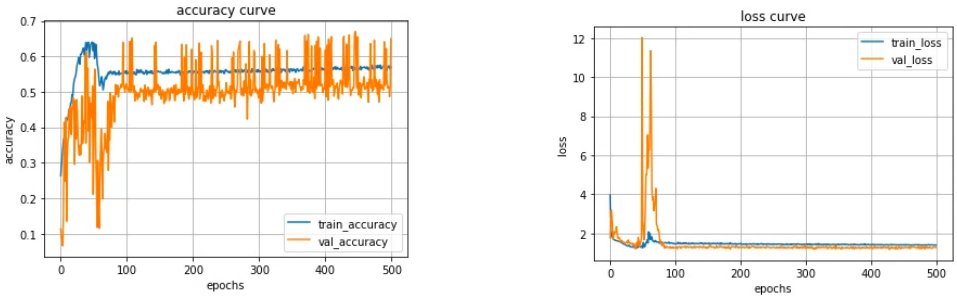
Figure B.7 depict the accuracy and loss curves generated after training the miniVGG on FER+ using dropconnect layers. The probability of dropconnect is set to 0.1 and the network is trained for 500 epochs.



(a) Accuracy v/s Epochs (b) Loss v/s Epochs

Figure B.6: Visualizing plots obtained by training miniVGG on FER+ using Dropout.

The unusually high loss obtained during the initial training process can be attributed to the difficulty of learning the features of this dataset.



(a) Accuracy v/s Epochs (b) Loss v/s Epochs

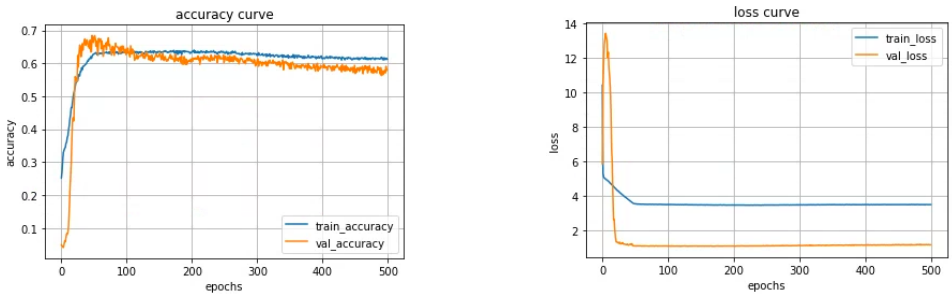
Figure B.7: Visualizing plots obtained by training miniVGG on FER+ using DropConnect.

B.2.4 Flipout

In this variation, the miniVGG network with flipout layers is trained on FER+ dataset. The probability of flipout is set to 0.1. The network performance gradually starts to deteriorate after 500 epochs as can be seen in Figure B.8a. The problem (negative loss) encountered in the CIFAR-10 equivalent of this configuration is not replicated here. The accuracy and the loss plots obtained for this training are provided for reference in Figure B.8.

B.3 Training MLP on California Housing dataset

This section provides the training specifications of the MLP for the regression task. Also, the data normalization applied prior to training has been discussed. Data normalization is important as the feature values in California Housing dataset range greatly and might pose a problem while training. The dataset



(a) Accuracy v/s Epochs

(b) Loss v/s Epochs

Figure B.8: Visualizing plots obtained by training miniVGG on FER+ using Flipout.

is subjected to the following operations:

$$X_{std} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{B.1a}$$

$$X_{scaled} = X_{std} \times (max - min) + min \tag{B.1b}$$

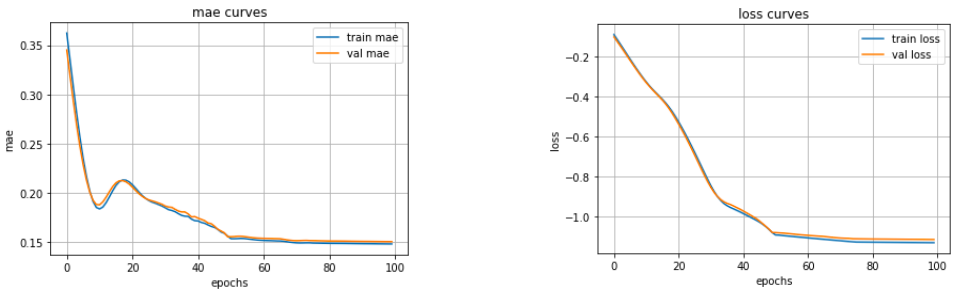
where min and max are the feature range. Equation B.1a and Equation B.1b have been adopted from the documentation of *sklearn.preprocessing.MinMaxScale* used in this thesis. Table B.5 provides the configuration used for training the variants of the MLP.

Hyperparameter	**Value**
optimizer	rmsprop
loss	MSE
metric	MAE
learning rate	0.001
train samples	12750
validation samples	4250
test samples	4250
train batch size	4096
validation batch size	4096

Table B.5: Hyperparameter settings used for training the MLP on California Housing dataset.

B.3.1 Deep Ensembles

Similar to the previous cases of training a network using this methodology, we instantiate multiple copies of the MLP from scratch and train them. Figure B.9 depicts the curves obtained after training the network. Owing to space constraint, the training curves generated by a single instance of the Deep Ensemble have been provided here.



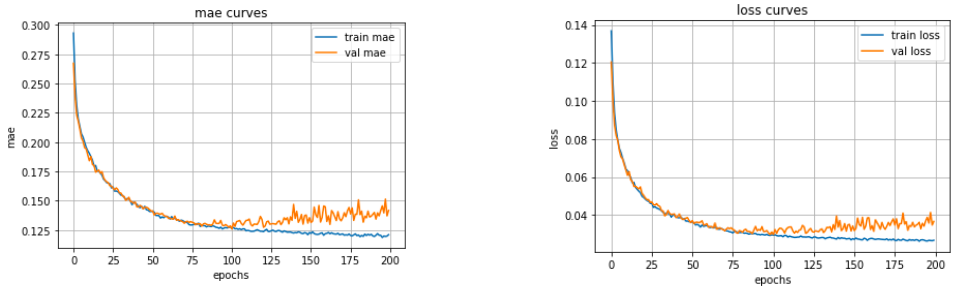
(a) MAE v/s Epochs

(b) Loss v/s Epochs

Figure B.9: Visualizing plots obtained by training MLP on California Housing dataset using Deep Ensembles.

B.3.2 Dropout

Figure B.10 depicts the training performance of the MLP with dropout layers. The network training is terminated after 200 epochs as the performance begins to deteriorate. A probability of 0.3 is selected empirically for the dropout layers.



(a) MAE v/s Epochs

(b) Loss v/s Epochs

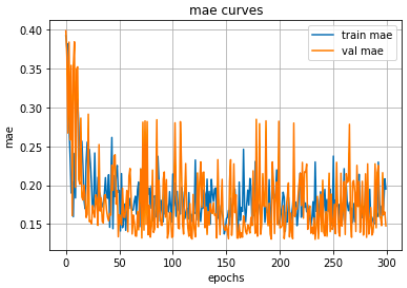
Figure B.10: Visualizing plots obtained by training MLP on California Housing dataset using Dropout.

B.3.3 DropConnect

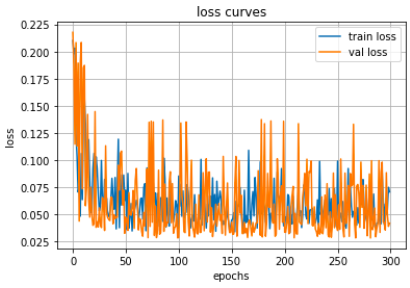
Figure B.11 depicts the curves obtained after training the MLP with dropconnect layers. A probability of 0.3 is set for the these layers and the network is trained upto 300 epochs.

B.3.4 Flipout

The probability for flipout is set at 0.1 and the network is trained for 500 epochs. The performance curves obtained after training the network are presented in Figure B.12. Though the training loss is gradually decreasing, we focus on validation loss that has already plateaued by the 500^{th} epoch.

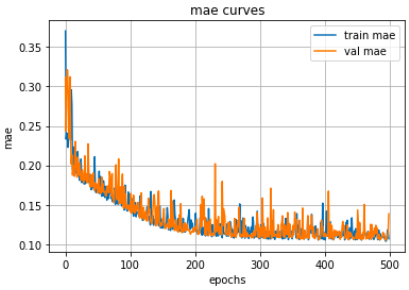


(a) MAE v/s Epochs

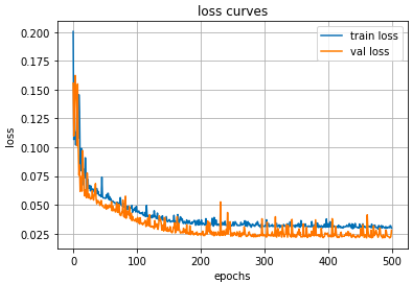


(b) Loss v/s Epochs

Figure B.11: Visualizing plots obtained by training MLP on California Housing dataset using DropConnect.



(a) MAE v/s Epochs



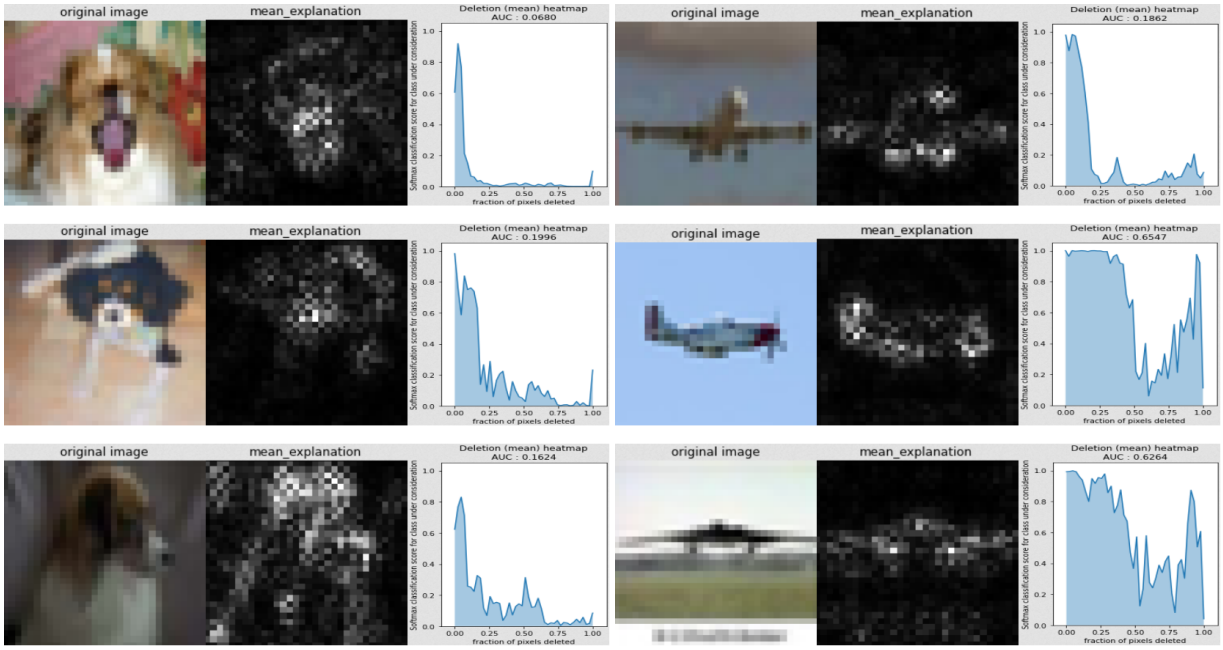
(b) Loss v/s Epochs

Figure B.12: Visualizing plots obtained by training MLP on California Housing dataset using Flipout.

C

Pixel Deletion/Insertion plots

In this section, we present the plots obtained during the pixel deletion and insertion analysis. As discussed previously, the AUC of these plots is critical in determining the quality of explanation. Approximately 27520000 inferences/forward passes are required to compute all the metrics listed in Table 6.1 - Table 6.16 and to plot the curves depicted in Figure C.2 - Figure C.25. The specifications of this amount are provided in this calculation[1].



(a) Deletion plots for multiple images of *dog* class.

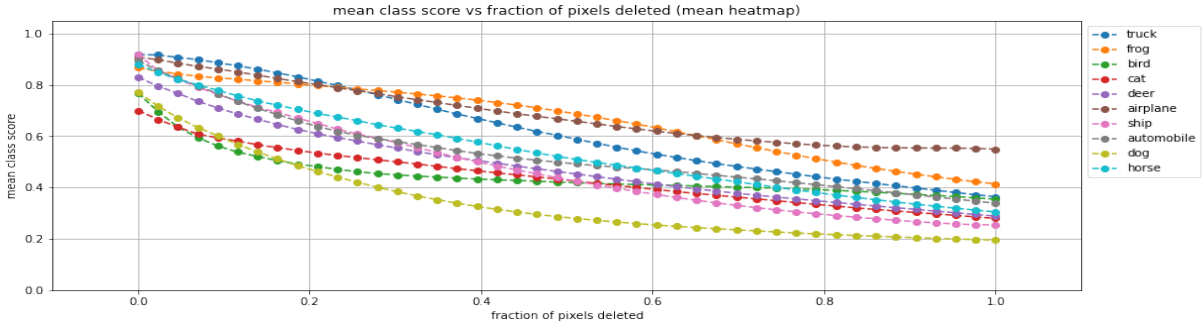
(b) Deletion plots for multiple images of *plane* class.

Figure C.1: Example of deletion curves for mean explanation representation of individual images belonging to two classes. This visualization highlights the variations in the deletion (and by extension, insertion) curves for images belonging to the same class. It is because of these variations, the behavior of the deletion/insertion curves depicted in Figure C.2 - Figure C.24 is to be expected.

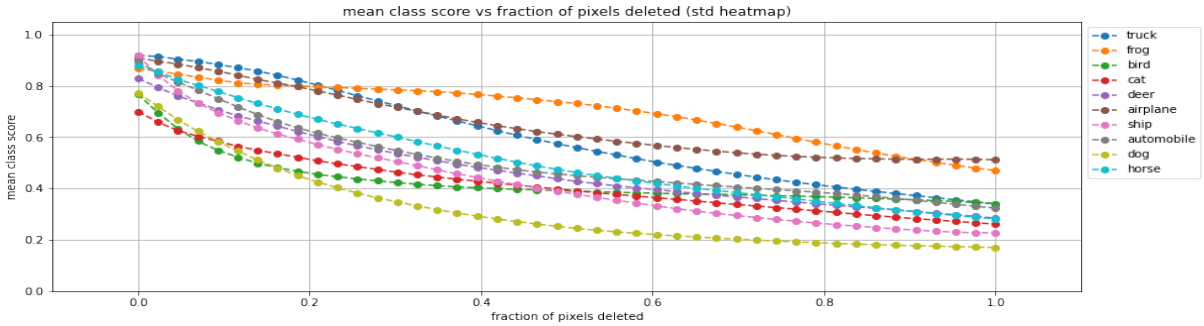
[1] [IG, GBP]×[μ, σ]×[deletion insertion]×[CIFAR-10, FER+]×[Deep Ensemble, MC Dropout, MC DropConnect, Flipout]×[#classes]×[#instances]×[#steps]×[#images]

Figure C.1 depicts the deletion AUC plots of the mean explanation representation for 3 images belonging to two classes namely dog and airplane each. An interesting insight to be obtained from this figure is the averaging effect of a large number of images. For a large batch of images, the deletion/insertion curves are pushed farther away from their expected behaviour. Ideally, a deletion curve should drop as quickly as possible and an insertion curve should rise as steeply as possible. However, due to the large variation in the class images, the deletion and the insertion curves do not exactly adhere to their ideal behaviours. This results in an "averaged" version of the curve. This observation is useful to understand the behaviour of plots obtained during the pixel deletion and insertion analysis (Figure C.2 - Figure C.25).

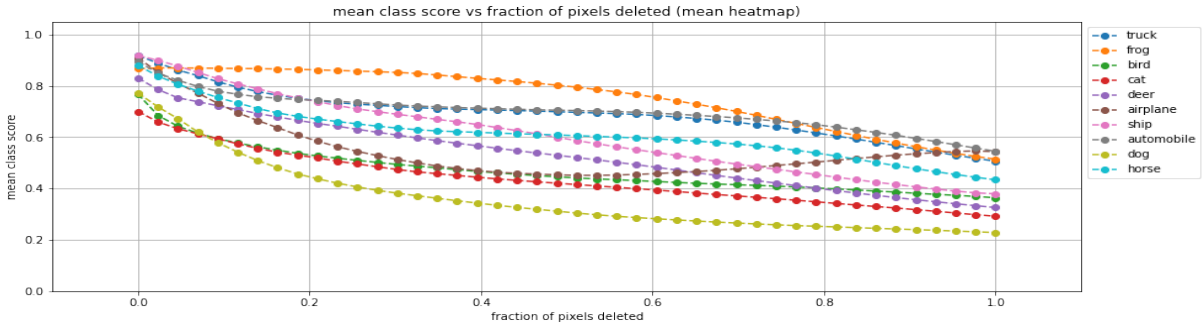
Figure C.2 and Figure C.3 depict the average deletion and insertion curves respectively for the mean and the standard deviation representation of CIFAR-10 dataset tested on: IG + Deep Ensembles and GBP + Deep Ensemble combinations. The envelope i.e. the curves with the largest and the smallest AUC in the subplots of Figure C.2 and Figure C.3 along with their ± 1 standard deviation have been depicted side-by-side in Figure C.4. For instance, Figure C.4a shows the envelope of the plot in Figure C.2a. Basically, the curves with the highest and the lowest AUC have been highlighted for clarity. This pattern is adhered to in the remaining figures of this chapter.



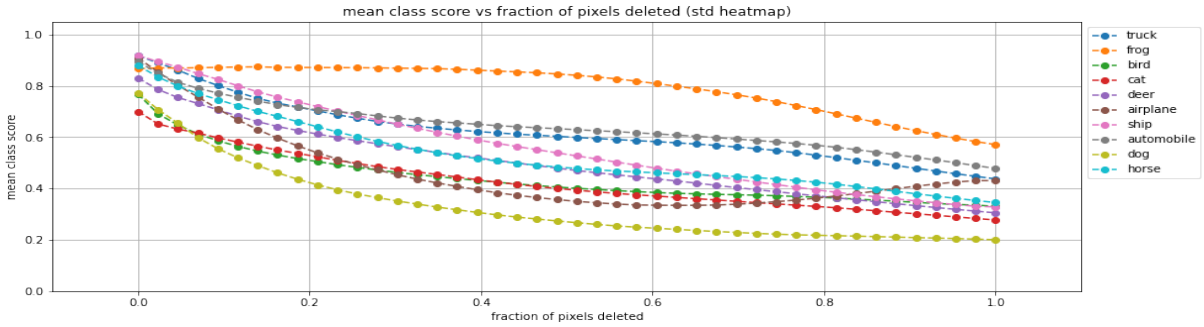
(a) Class-wise deletion curves for the mean explanation representations obtained using GBP.



(b) Class-wise deletion curves for the standard deviation explanation representations obtained using GBP.

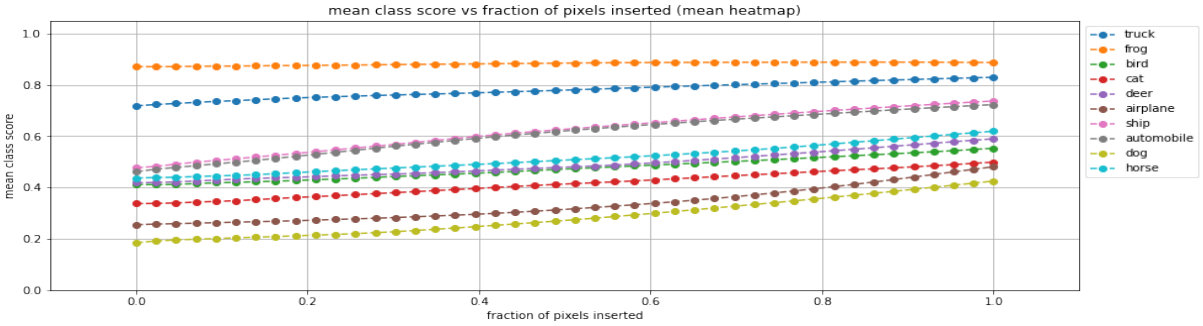


(c) Class-wise deletion curves for the mean explanation representations obtained using IG.

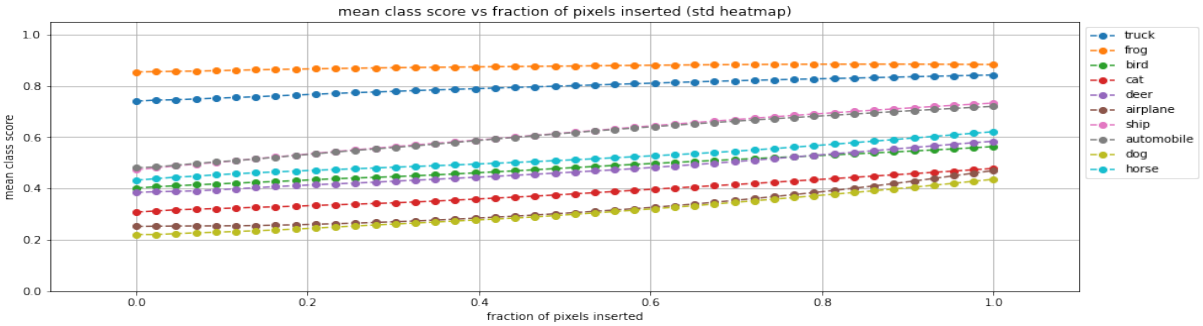


(d) Class-wise deletion curves for the standard deviation explanation representations obtained using IG.

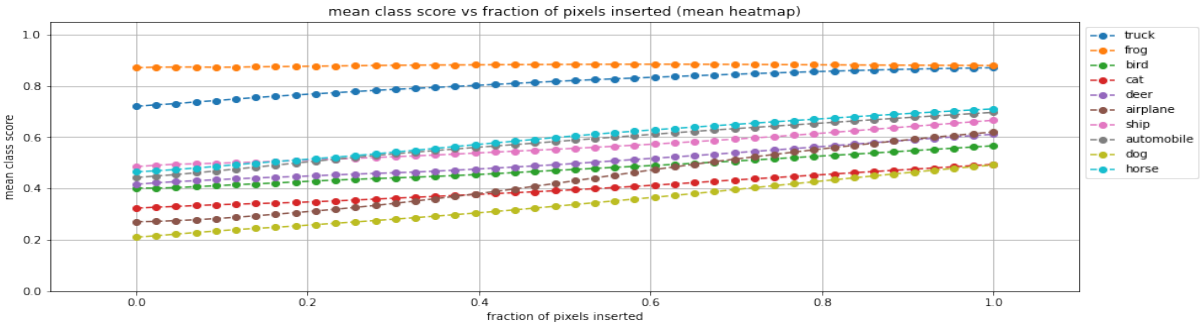
Figure C.2: Deletion curves for Deep Ensembles with CIFAR-10 images.



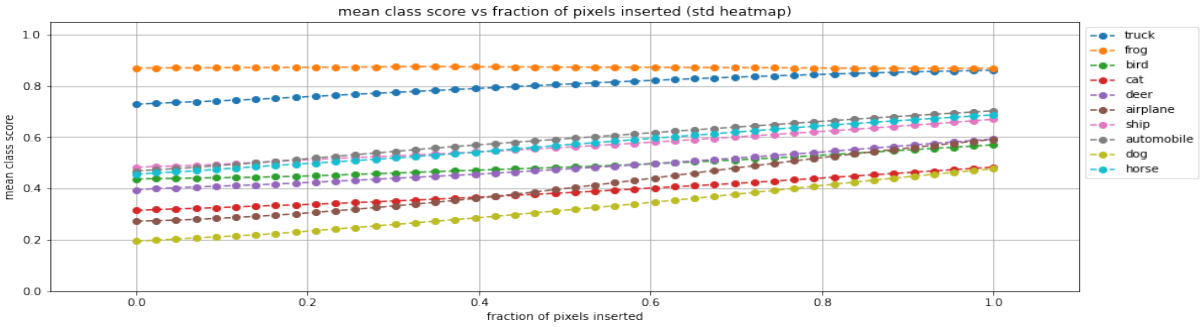
(a) Class-wise insertion curves for the mean explanation representations obtained using GBP.



(b) Class-wise insertion curves for the standard deviation explanation representations obtained using GBP.

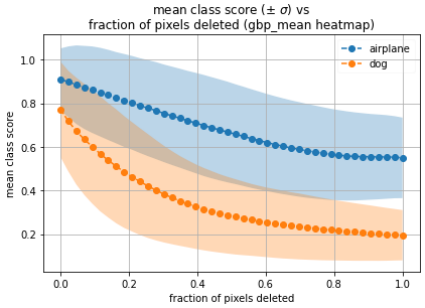


(c) Class-wise insertion curves for the mean explanation representations obtained using IG.

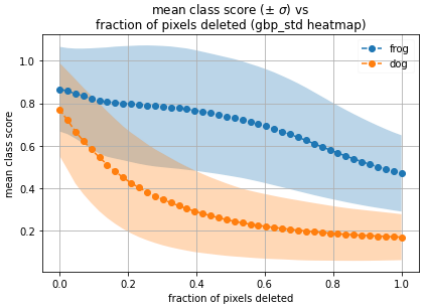


(d) Class-wise insertion curves for the standard deviation explanation representations obtained using IG.

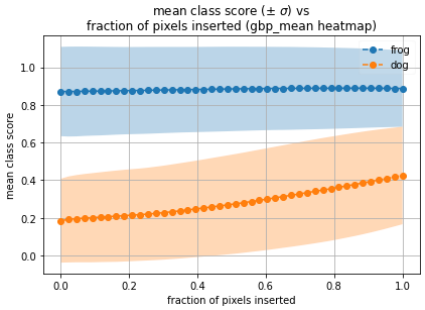
Figure C.3: Insertion curves for Deep Ensembles with CIFAR-10 images.



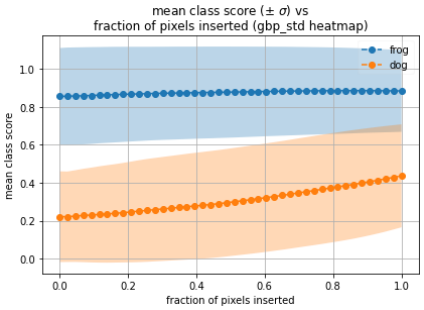
(a) Plots with max. and min. AUC in Figure C.2a.



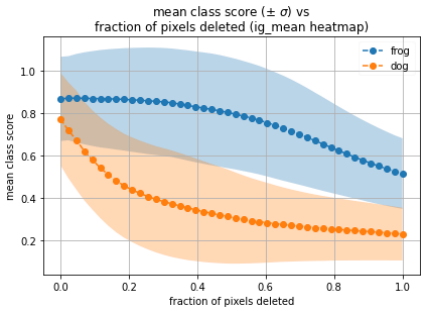
(b) Plots with max. and min. AUC in Figure C.2b.



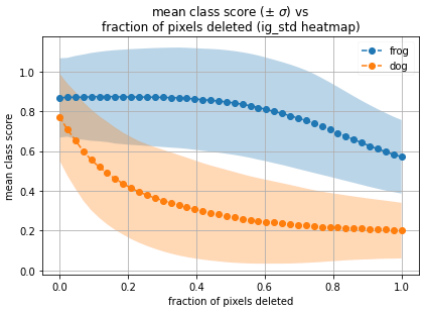
(c) Plots with max. and min. AUC in Figure C.3a.



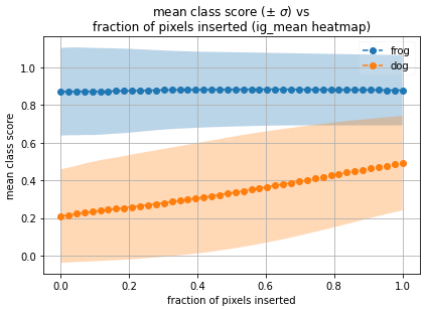
(d) Plots with max. and min. AUC in Figure C.3b.



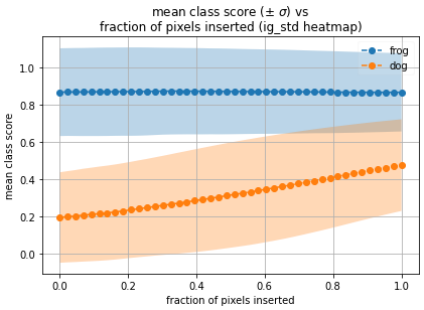
(e) Plots with max. and min. AUC in Figure C.2c.



(f) Plots with max. and min. AUC in Figure C.2d.

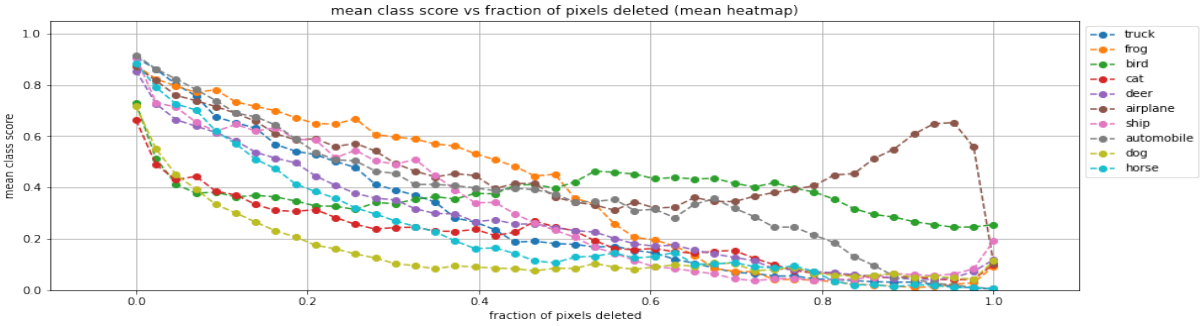


(g) Plots with max. and min. AUC in Figure C.3c.

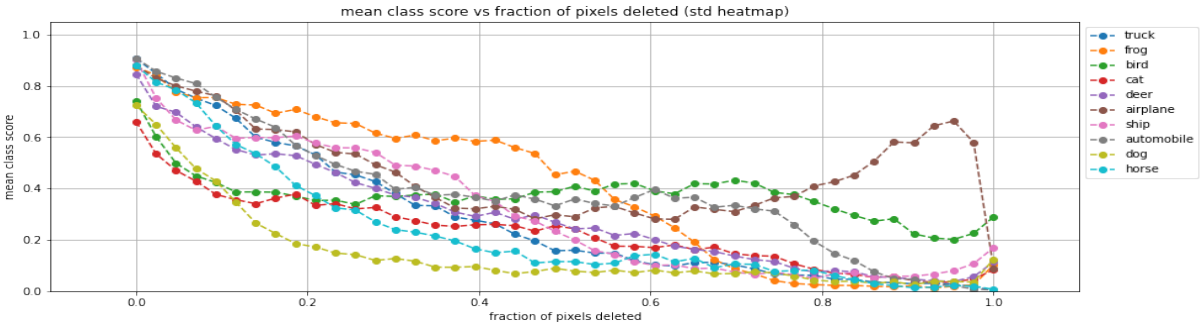


(h) Plots with max. and min. AUC in Figure C.3d.

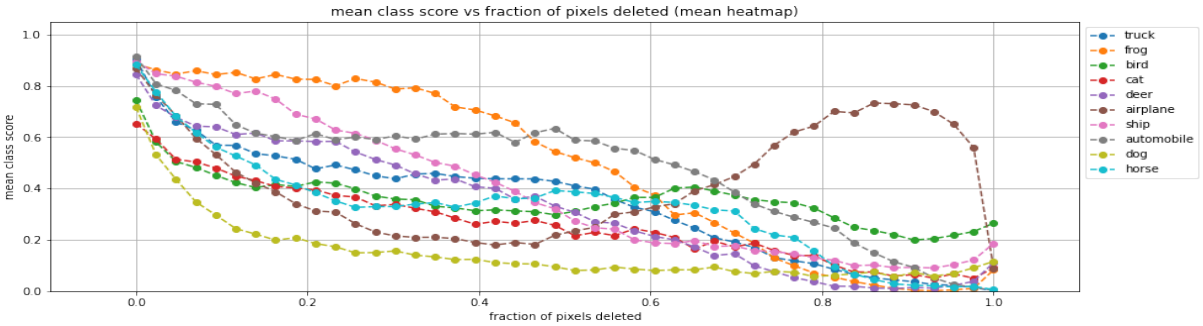
Figure C.4: Plots with maximum and minimum AUC along with their ±1 standard deviation envelopes for Deep Ensemble on CIFAR-10 images.



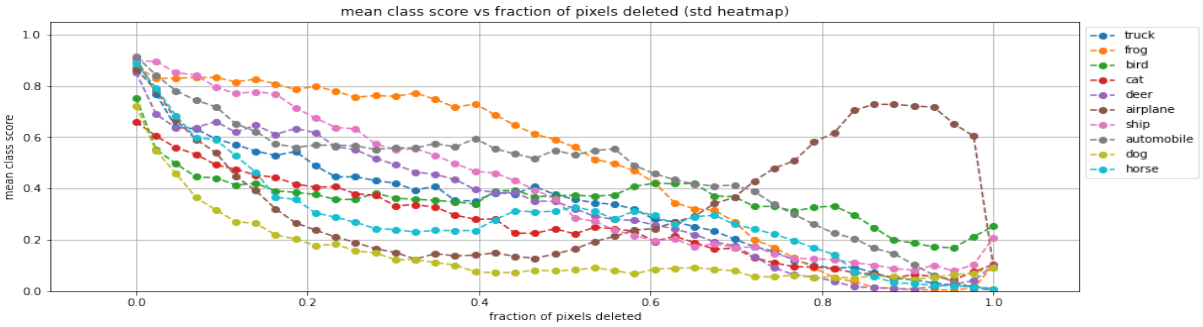
(a) Class-wise deletion curves for the mean explanation representations obtained using GBP.



(b) Class-wise deletion curves for the standard deviation explanation representations obtained using GBP.

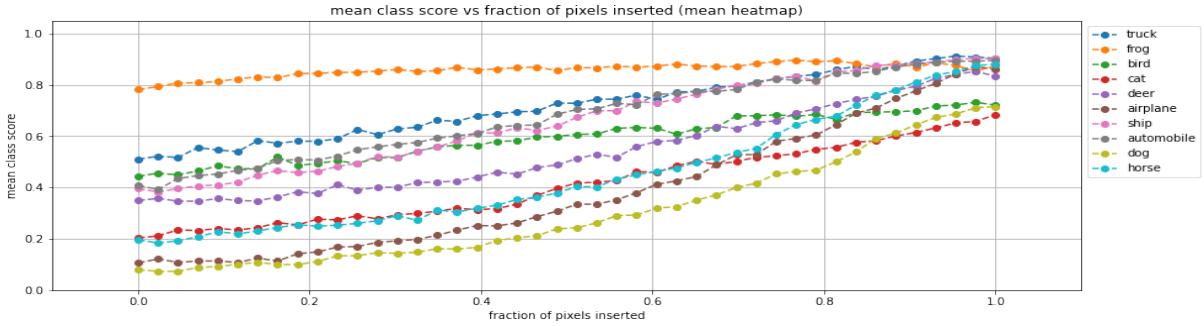


(c) Class-wise deletion curves for the mean explanation representations obtained using IG.

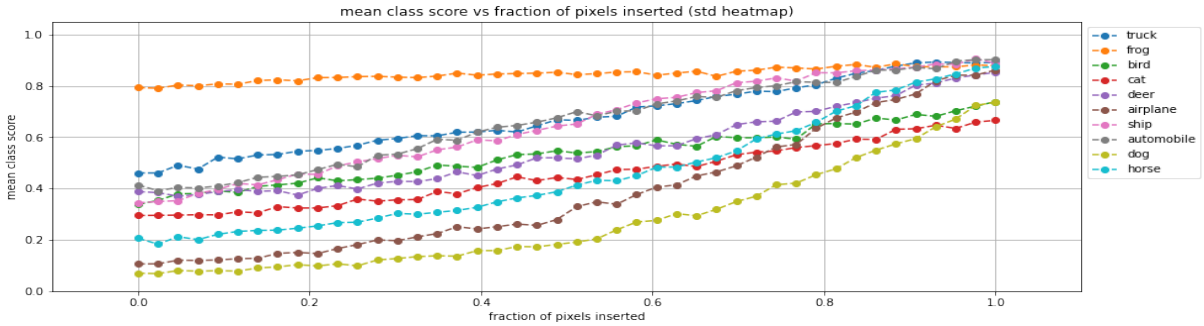


(d) Class-wise deletion curves for the standard deviation explanation representations obtained using IG.

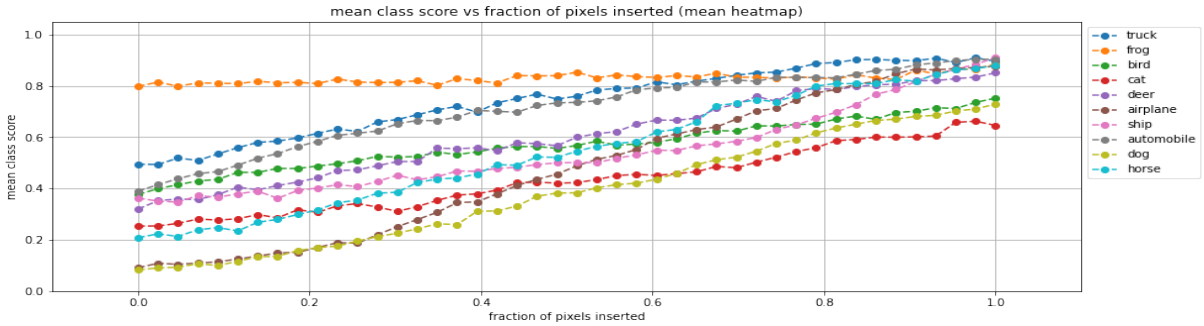
Figure C.5: Deletion curves for Dropout with CIFAR-10 images.



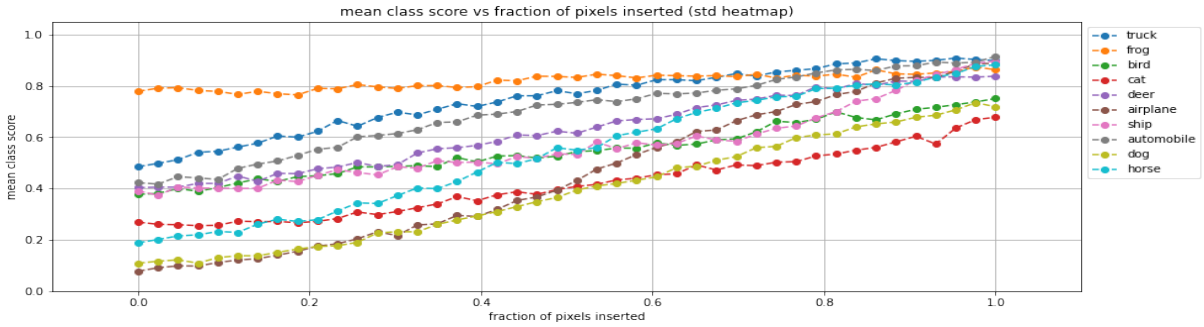
(a) Class-wise insertion curves for the mean explanation representations obtained using GBP.



(b) Class-wise insertion curves for the standard deviation explanation representations obtained using GBP.

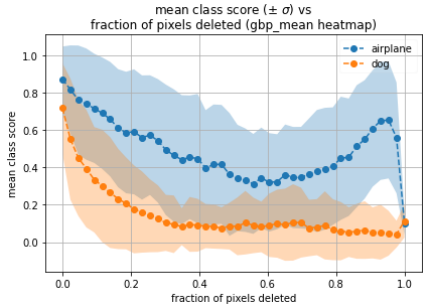


(c) Class-wise insertion curves for the mean explanation representations obtained using IG.

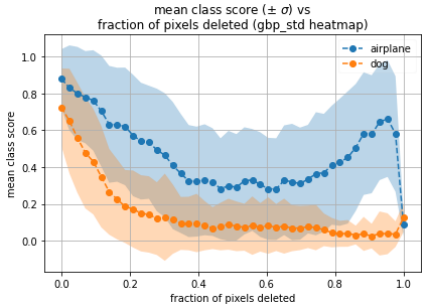


(d) Class-wise insertion curves for the standard deviation explanation representations obtained using IG.

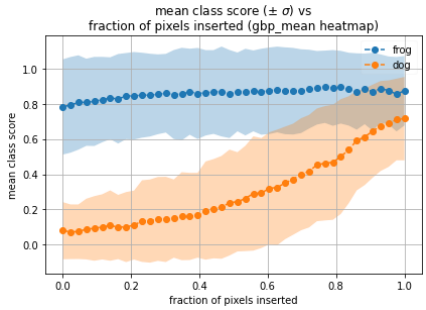
Figure C.6: Insertion curves for Dropout with CIFAR-10 images.



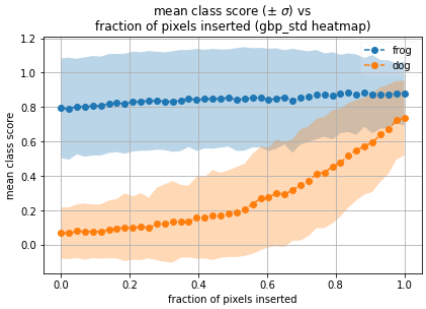
(a) Plots with max. and min. AUC in Figure C.5a.



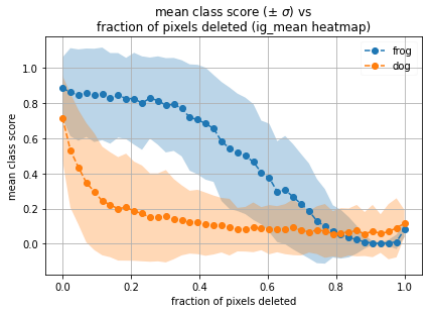
(b) Plots with max. and min. AUC in Figure C.5b.



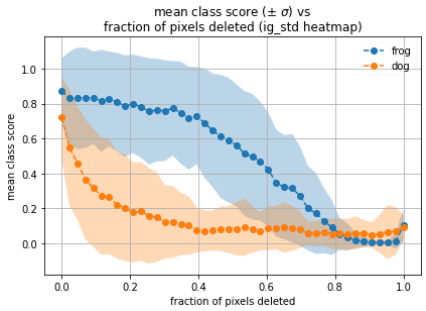
(c) Plots with max. and min. AUC in Figure C.6a.



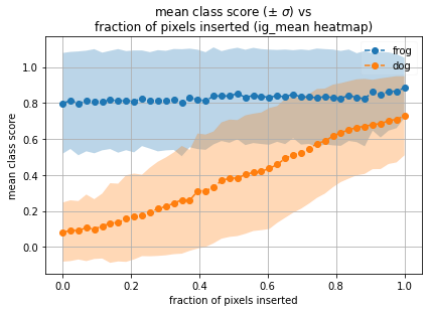
(d) Plots with max. and min. AUC in Figure C.6b.



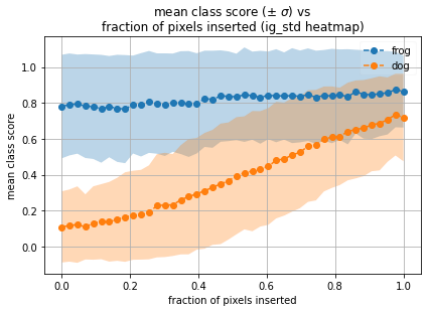
(e) Plots with max. and min. AUC in Figure C.5c.



(f) Plots with max. and min. AUC in Figure C.5d.

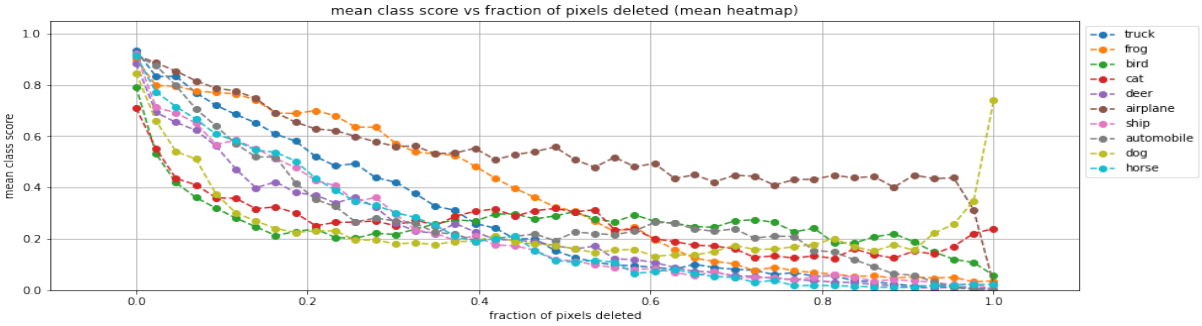


(g) Plots with max. and min. AUC in Figure C.6c.

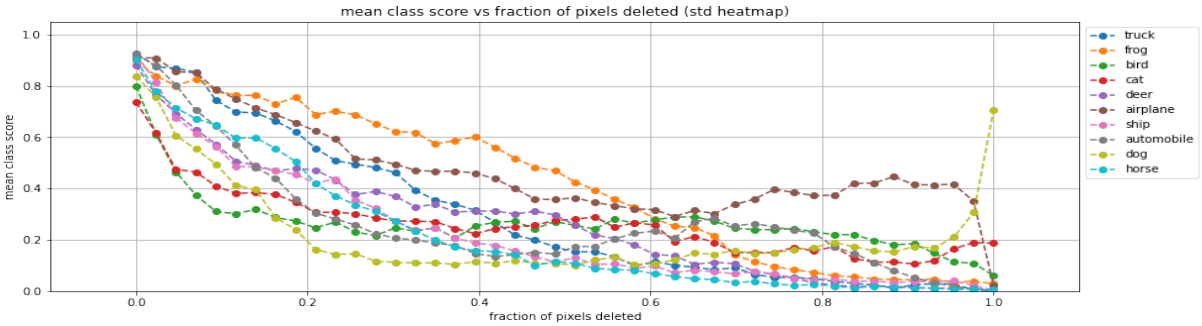


(h) Plots with max. and min. AUC in Figure C.6d.

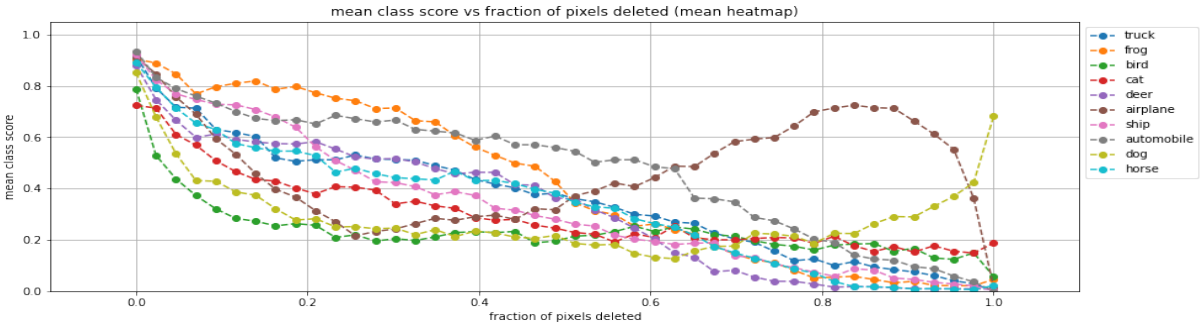
Figure C.7: Plots with maximum and minimum AUC along with their ±1 standard deviation envelopes for Dropout on CIFAR-10 images.



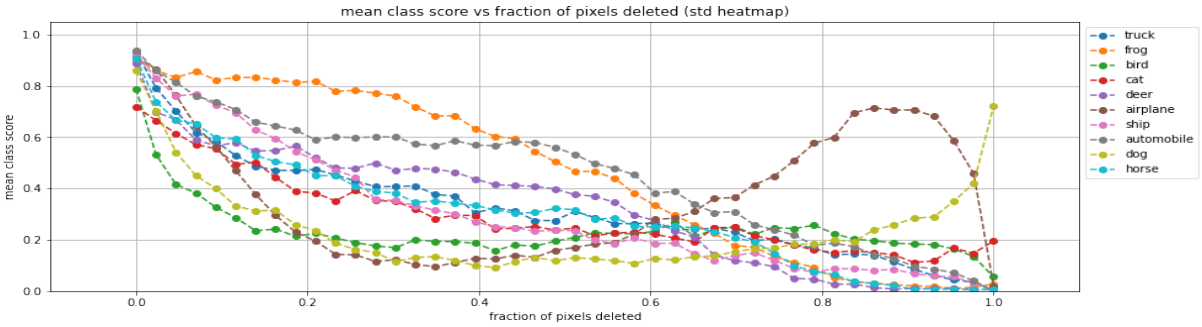
(a) Class-wise deletion curves for the mean explanation representations obtained using GBP.



(b) Class-wise deletion curves for the standard deviation explanation representations obtained using GBP.

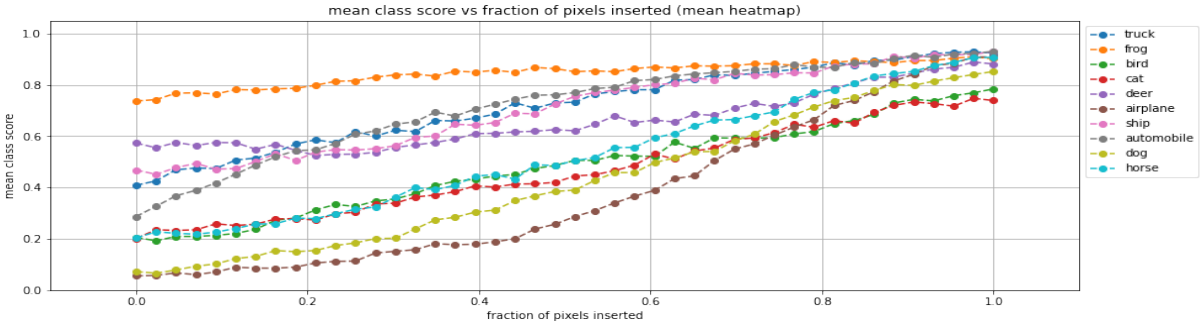


(c) Class-wise deletion curves for the mean explanation representations obtained using IG.

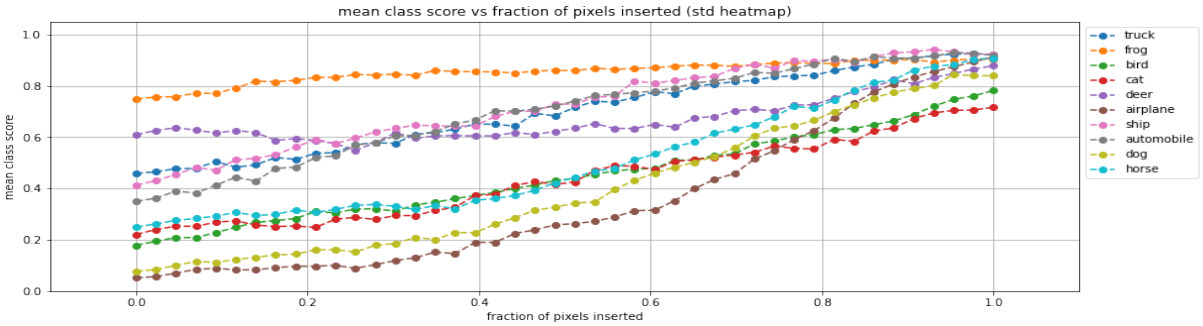


(d) Class-wise deletion curves for the standard deviation explanation representations obtained using IG.

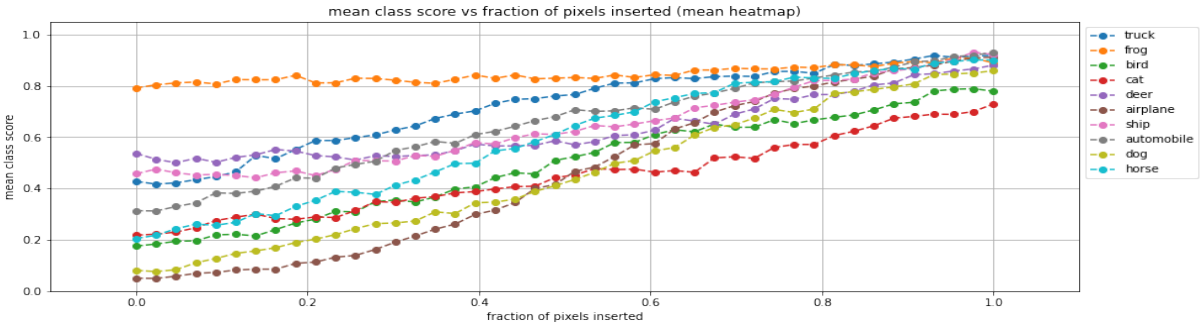
Figure C.8: Deletion curves for DropConnect with CIFAR-10 images.



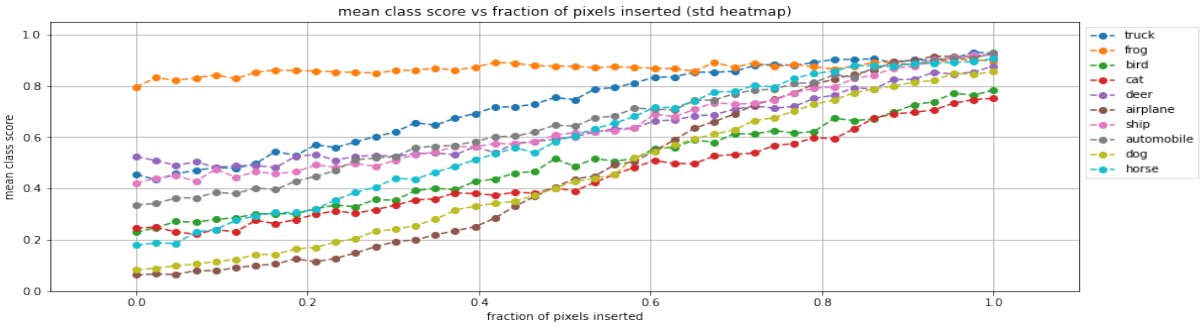
(a) Class-wise insertion curves for the mean explanation representations obtained using GBP.



(b) Class-wise insertion curves for the standard deviation explanation representations obtained using GBP.

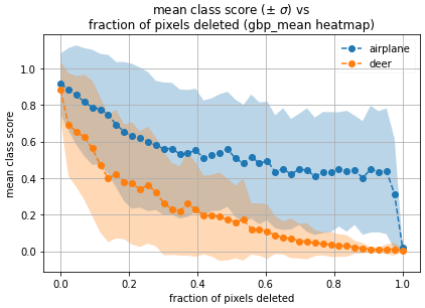


(c) Class-wise insertion curves for the mean explanation representations obtained using IG.

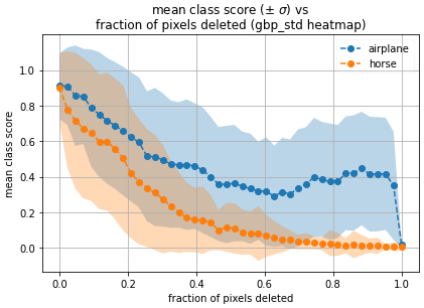


(d) Class-wise insertion curves for the standard deviation explanation representations obtained using IG.

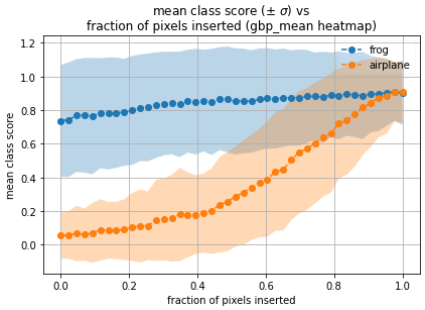
Figure C.9: Insertion curves for DropConnect with CIFAR-10 images.



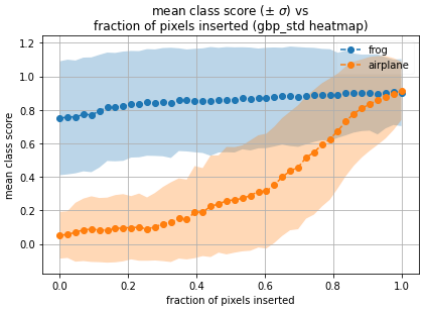
(a) Plots with max. and min. AUC in Figure C.8a.



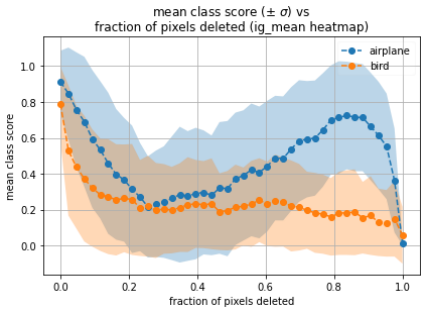
(b) Plots with max. and min. AUC in Figure C.8b.



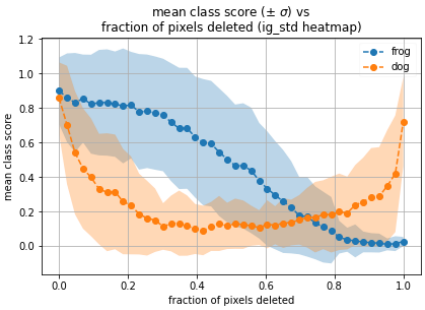
(c) Plots with max. and min. AUC in Figure C.9a.



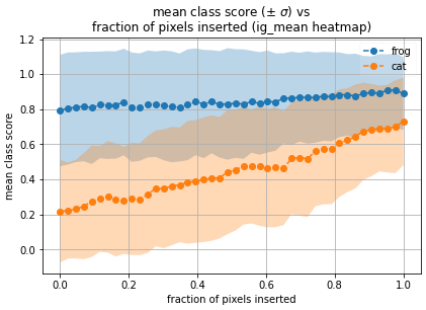
(d) Plots with max. and min. AUC in Figure C.9b.



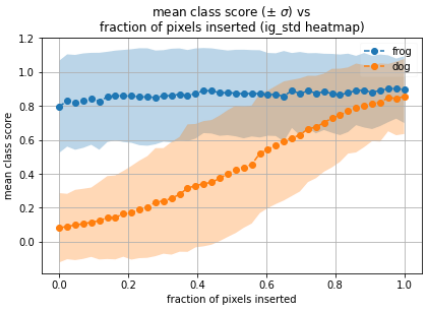
(e) Plots with max. and min. AUC in Figure C.8c.



(f) Plots with max. and min. AUC in Figure C.8d.

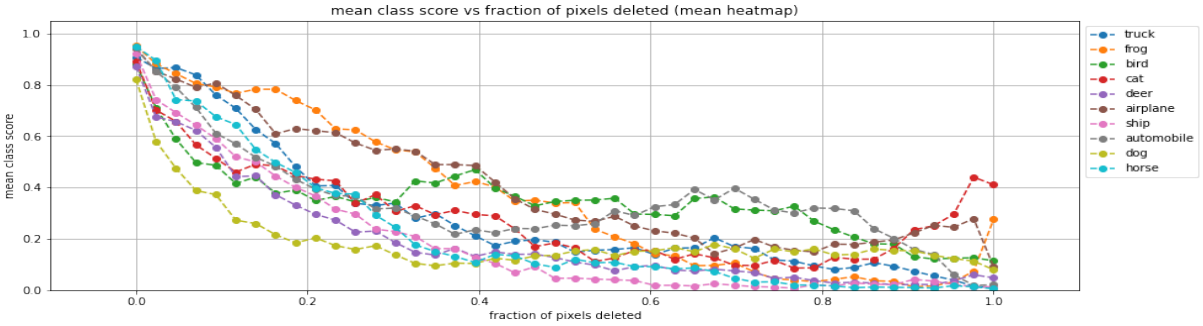


(g) Plots with max. and min. AUC in Figure C.9c.

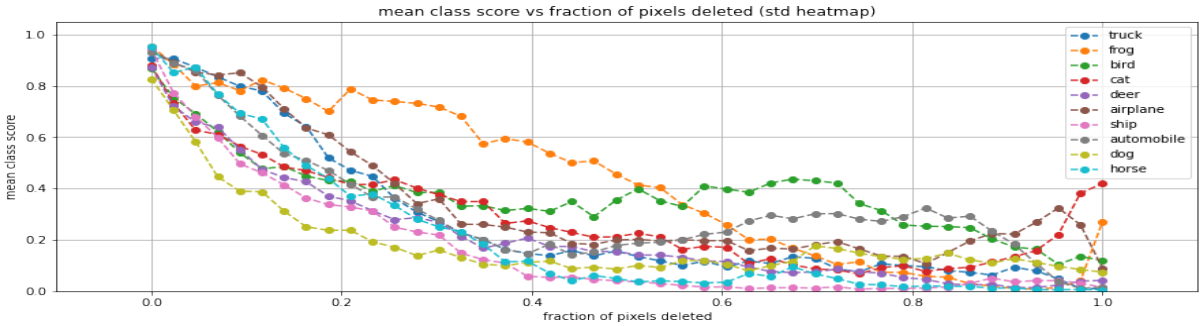


(h) Plots with max. and min. AUC in Figure C.9d.

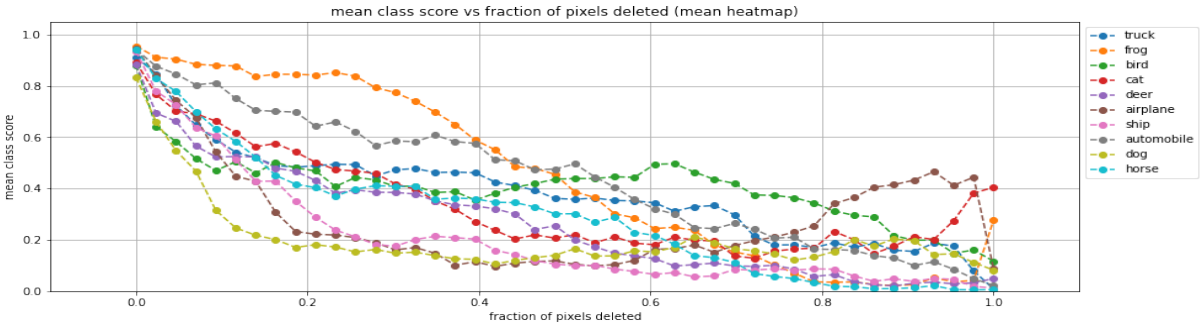
Figure C.10: Plots with maximum and minimum AUC along with their ±1 standard deviation envelopes for DropConnect on CIFAR-10 images.



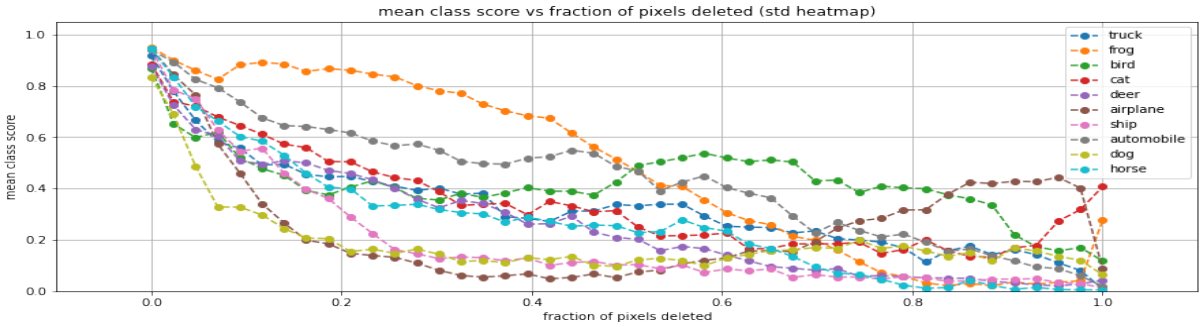
(a) Class-wise deletion curves for the mean explanation representations obtained using GBP.



(b) Class-wise deletion curves for the standard deviation explanation representations obtained using GBP.

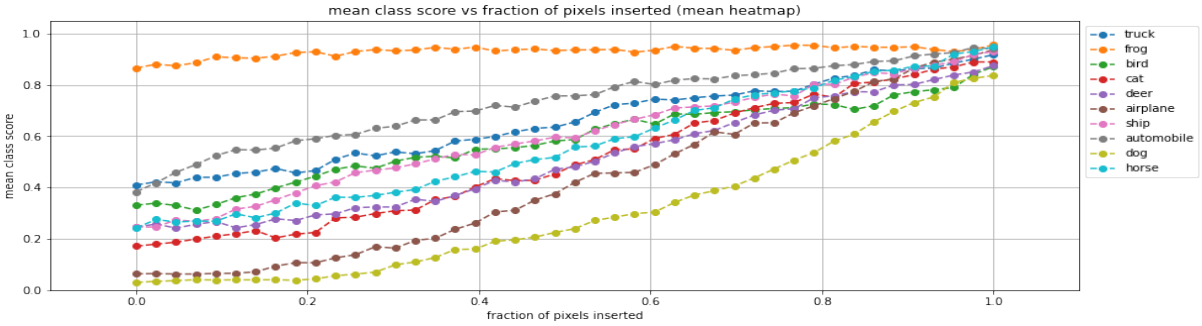


(c) Class-wise deletion curves for the mean explanation representations obtained using IG.

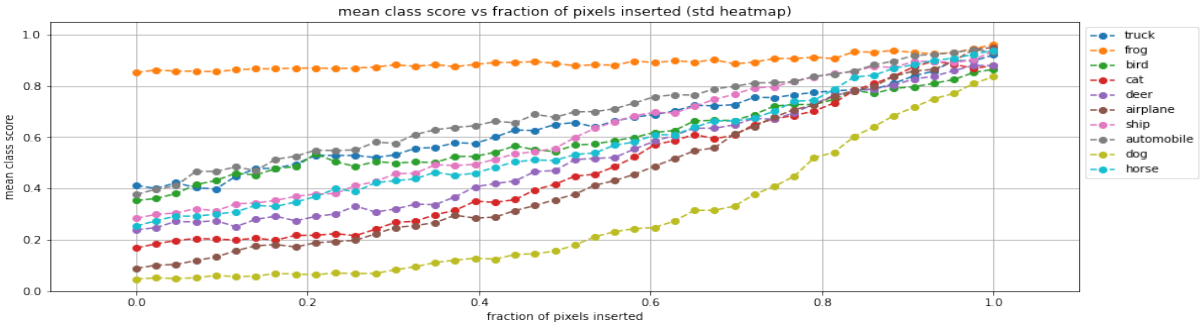


(d) Class-wise deletion curves for the standard deviation explanation representations obtained using IG.

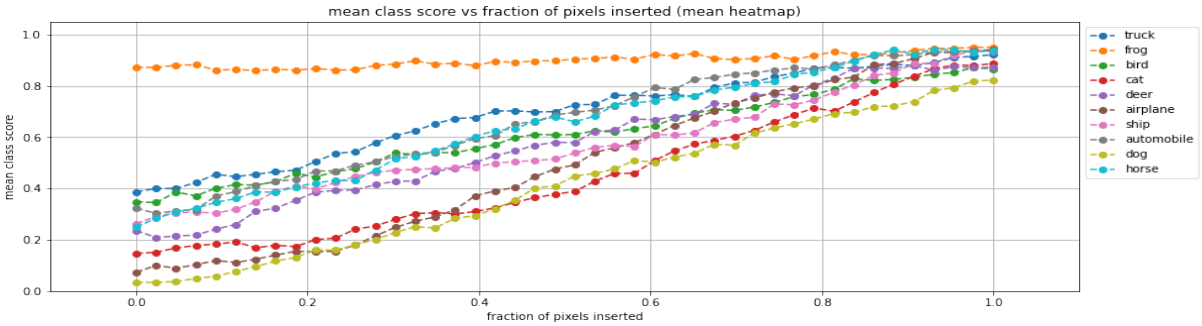
Figure C.11: Deletion curves for Flipout with CIFAR-10 images.



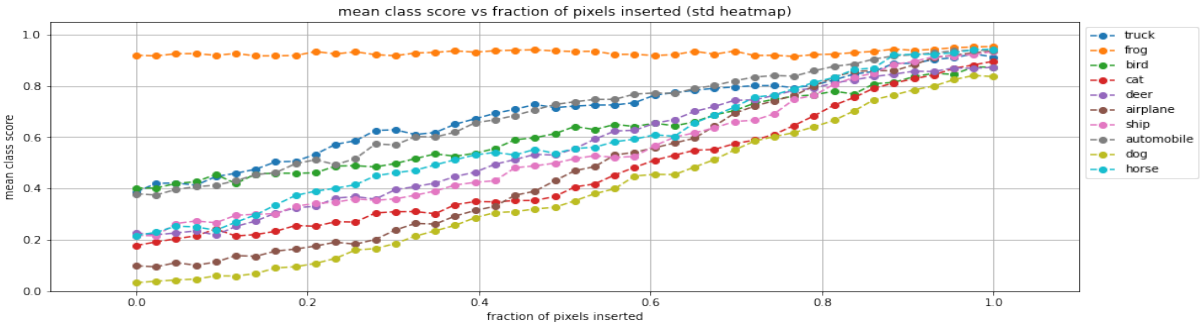
(a) Class-wise insertion curves for the mean explanation representations obtained using GBP.



(b) Class-wise insertion curves for the standard deviation explanation representations obtained using GBP.

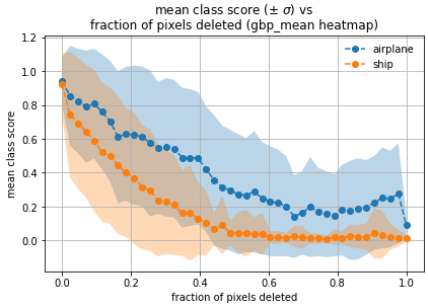


(c) Class-wise insertion curves for the mean explanation representations obtained using IG.

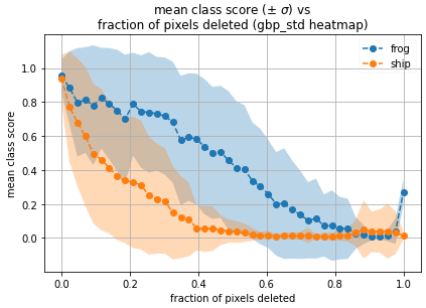


(d) Class-wise insertion curves for the standard deviation explanation representations obtained using IG.

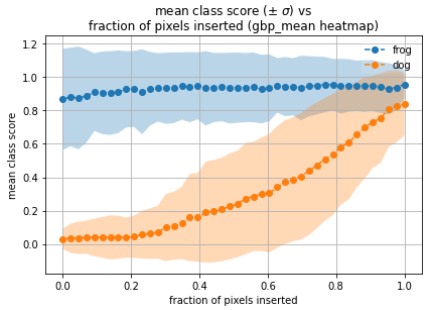
Figure C.12: Insertion curves for Flipout with CIFAR-10 images.



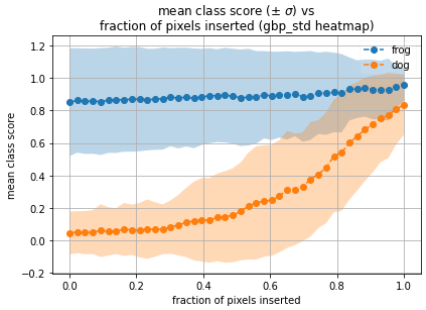
(a) Plots with max. and min. AUC in Figure C.11a.



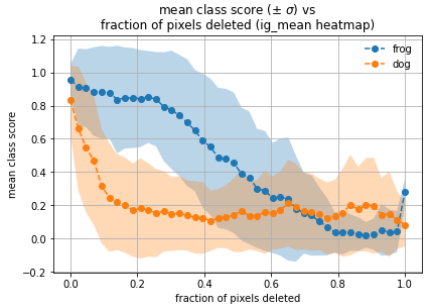
(b) Plots with max. and min. AUC in Figure C.11b.



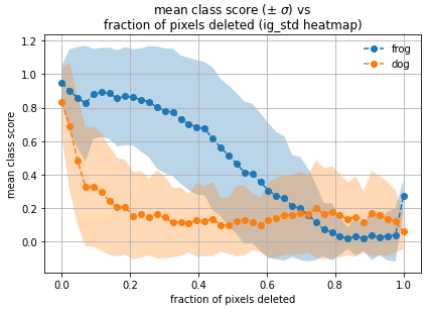
(c) Plots with max. and min. AUC in Figure C.12a.



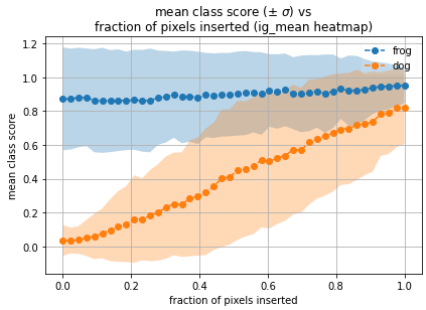
(d) Plots with max. and min. AUC in Figure C.12b.



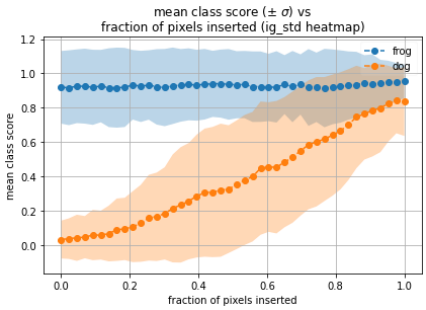
(e) Plots with max. and min. AUC in Figure C.11c.



(f) Plots with max. and min. AUC in Figure C.11d.

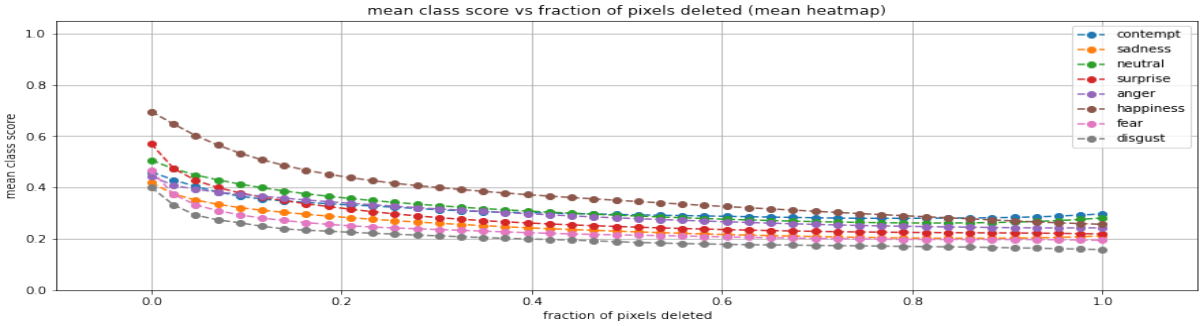


(g) Plots with max. and min. AUC in Figure C.12c.

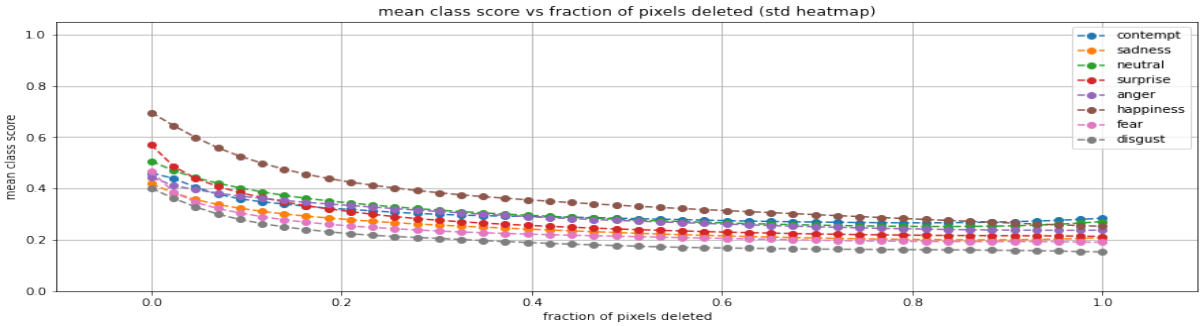


(h) Plots with max. and min. AUC in Figure C.12d.

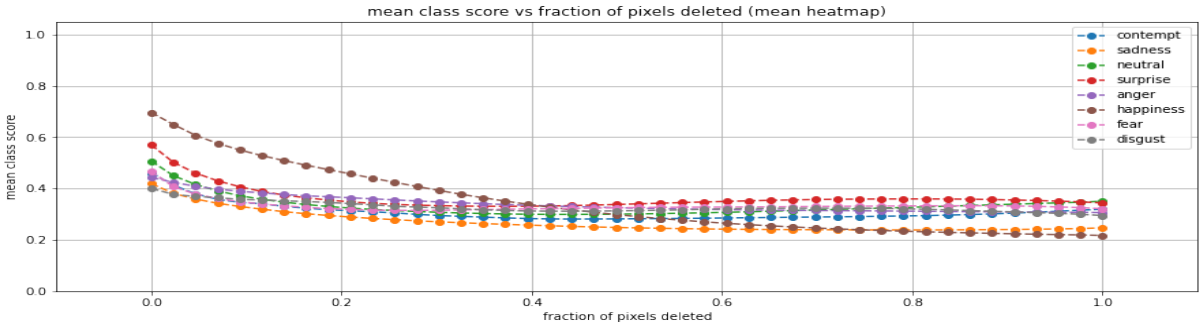
Figure C.13: Plots with maximum and minimum AUC along with their ±1 standard deviation envelopes for Flipout on CIFAR-10 images.



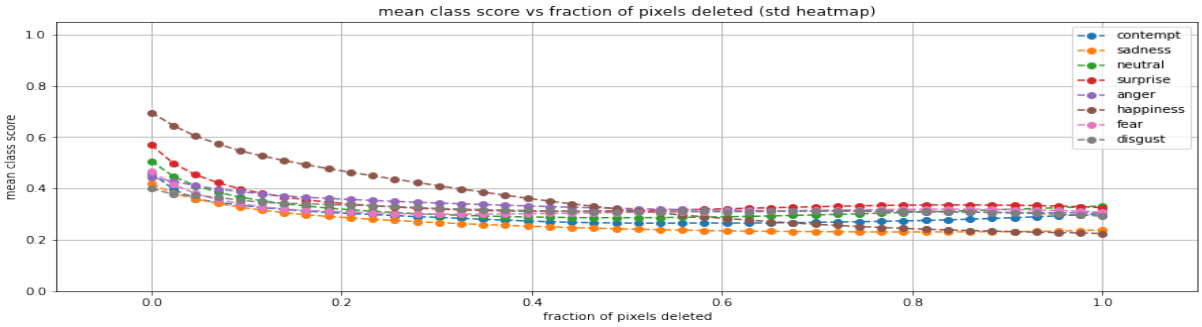
(a) Class-wise deletion curves for the mean explanation representations obtained using GBP.



(b) Class-wise deletion curves for the standard deviation explanation representations obtained using GBP.

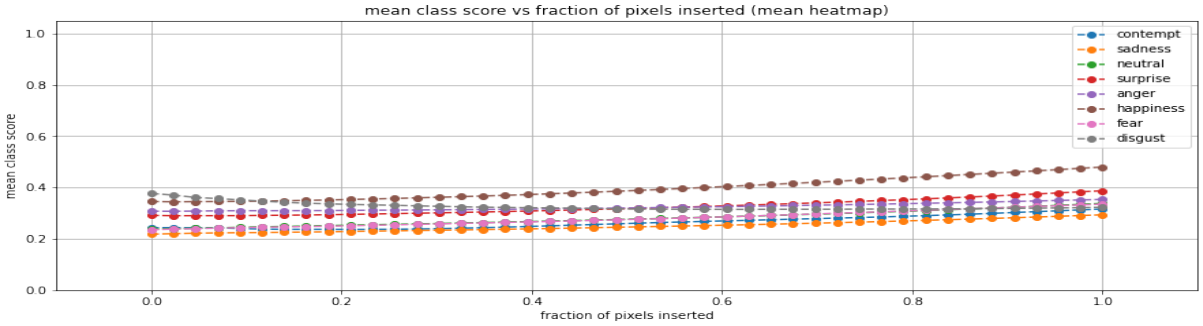


(c) Class-wise deletion curves for the mean explanation representations obtained using IG.

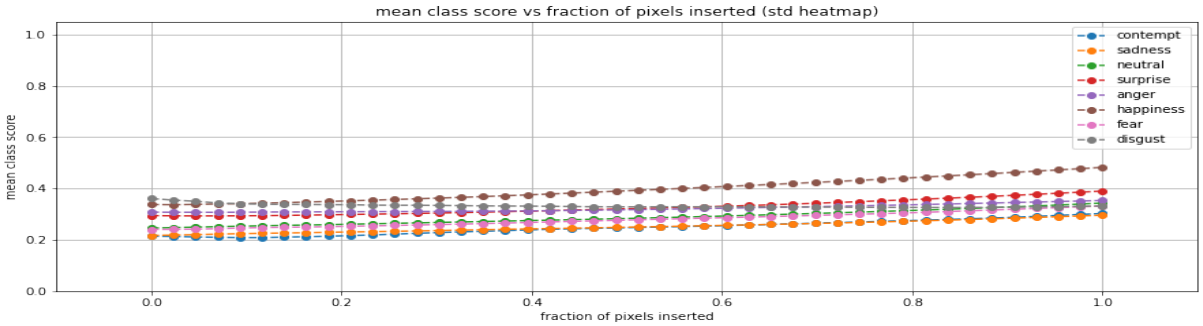


(d) Class-wise deletion curves for the standard deviation explanation representations obtained using IG.

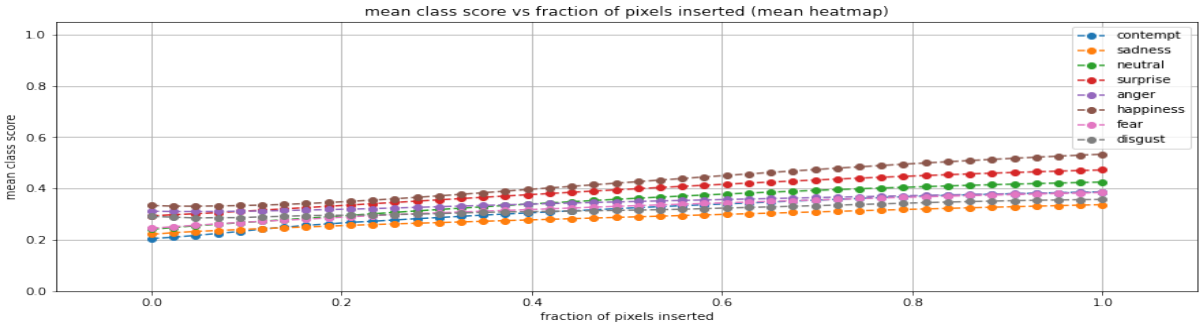
Figure C.14: Deletion curves for Deep Ensemble with FER+ images.



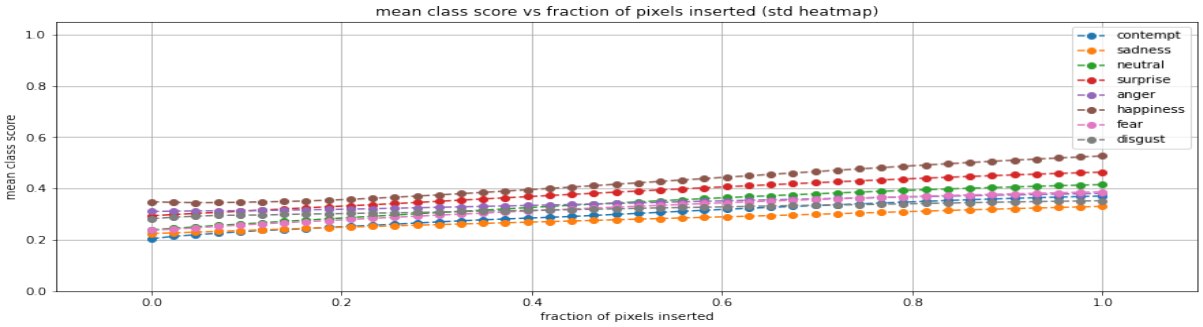
(a) Class-wise insertion curves for the mean explanation representations obtained using GBP.



(b) Class-wise insertion curves for the standard deviation explanation representations obtained using GBP.

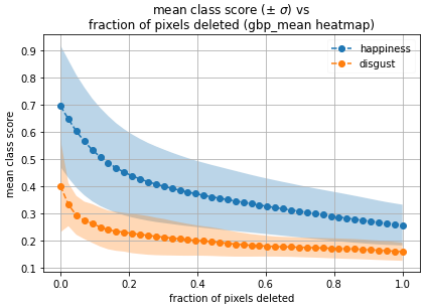


(c) Class-wise insertion curves for the mean explanation representations obtained using IG.

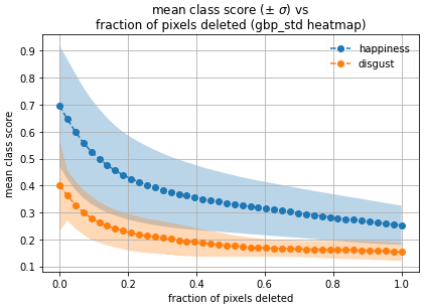


(d) Class-wise insertion curves for the standard deviation explanation representations obtained using IG.

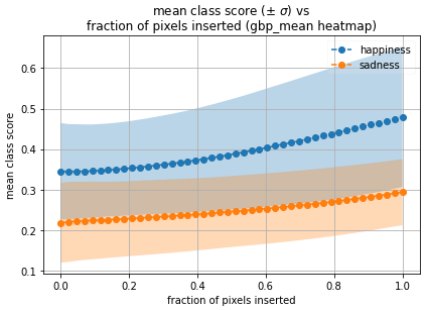
Figure C.15: Insertion curves for Deep Ensemble with FER+ images.



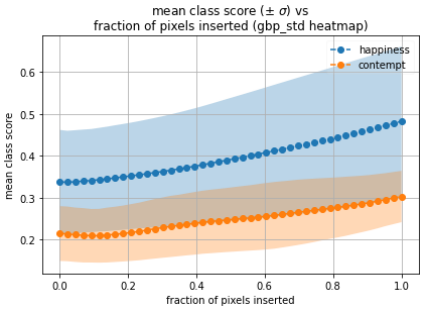
(a) Plots with max. and min. AUC in Figure C.14a.



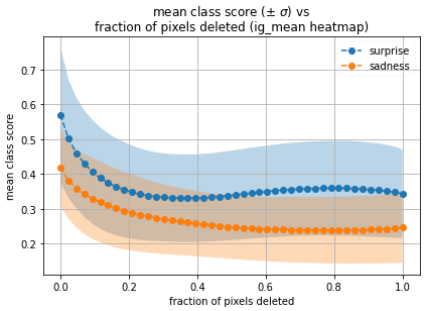
(b) Plots with max. and min. AUC in Figure C.14b.



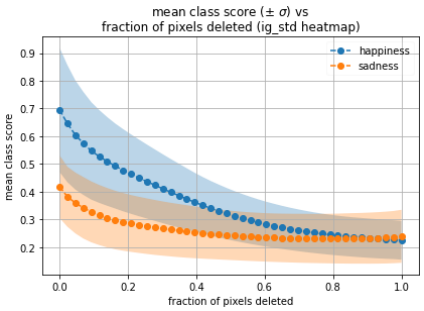
(c) Plots with max. and min. AUC in Figure C.15a.



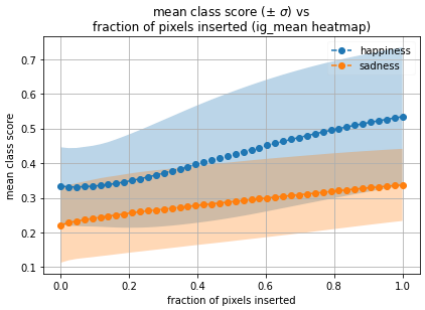
(d) Plots with max. and min. AUC in Figure C.15b.



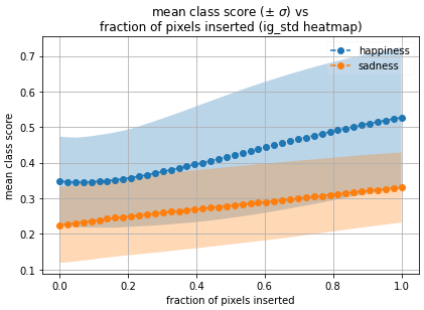
(e) Plots with max. and min. AUC in Figure C.14c.



(f) Plots with max. and min. AUC in Figure C.14d.

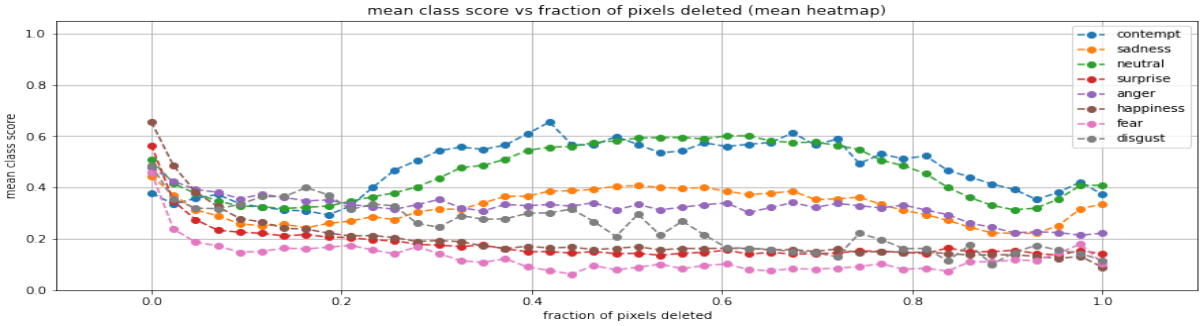


(g) Plots with max. and min. AUC in Figure C.15c.

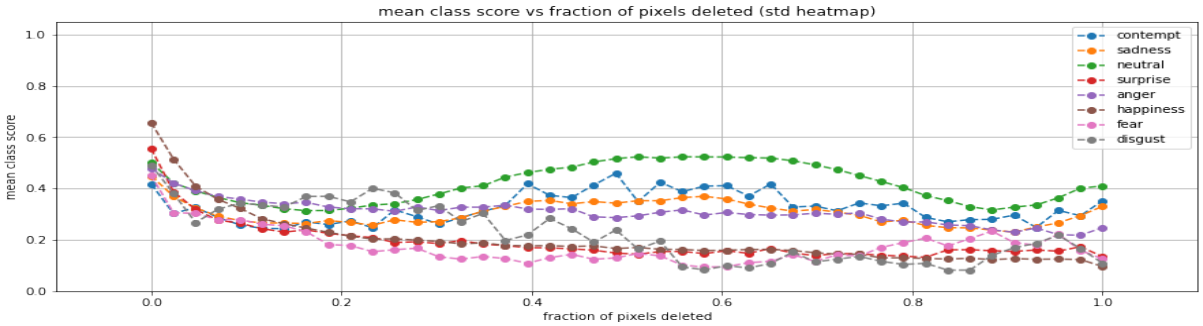


(h) Plots with max. and min. AUC in Figure C.15d.

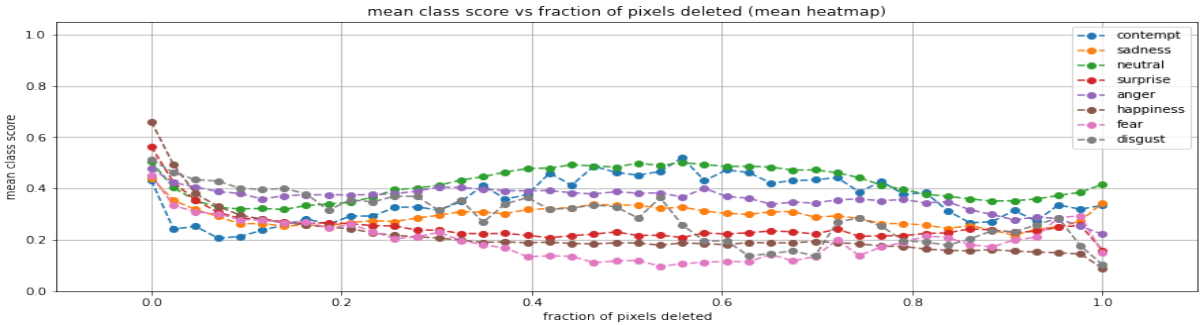
Figure C.16: Plots with maximum and minimum AUC along with their ±1 standard deviation envelopes for Deep Ensemble on FER+ images.



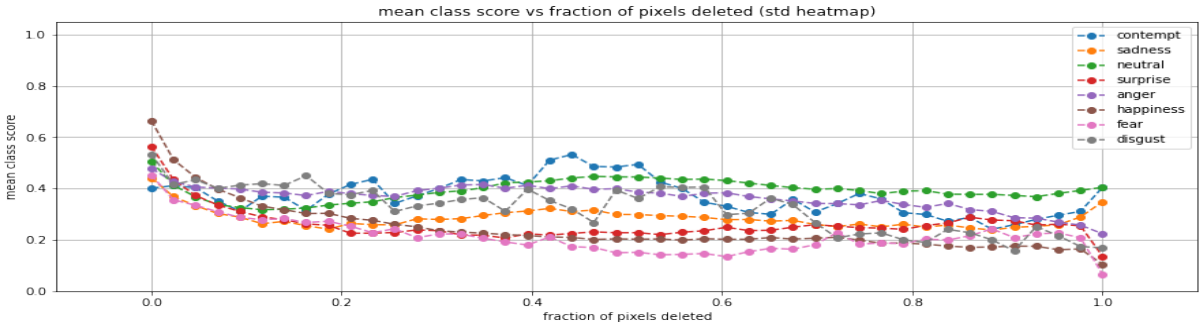
(a) Class-wise deletion curves for the mean explanation representations obtained using GBP.



(b) Class-wise deletion curves for the standard deviation explanation representations obtained using GBP.

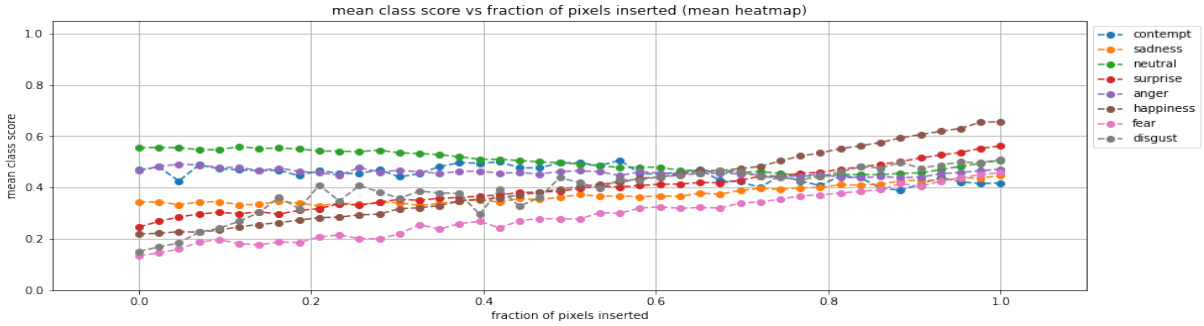


(c) Class-wise deletion curves for the mean explanation representations obtained using IG.

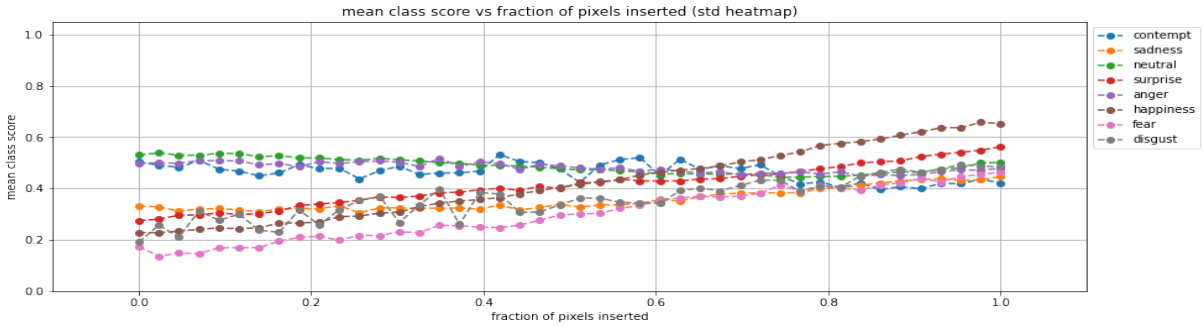


(d) Class-wise deletion curves for the standard deviation explanation representations obtained using IG.

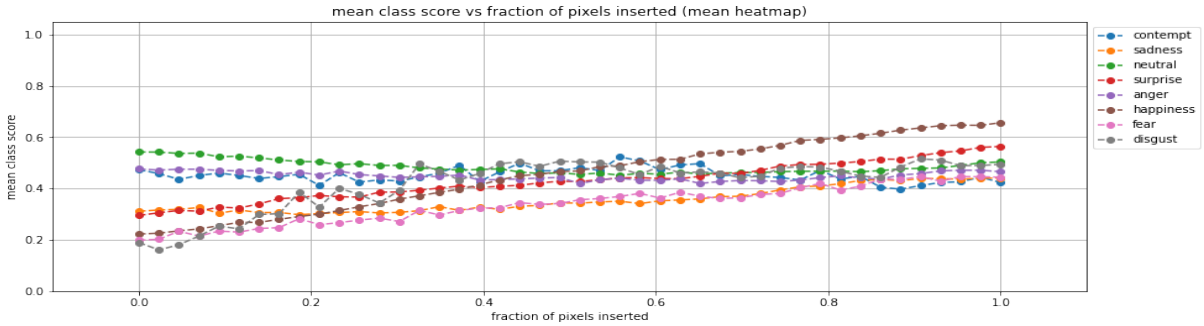
Figure C.17: Deletion curves for Dropout with FER+ images.



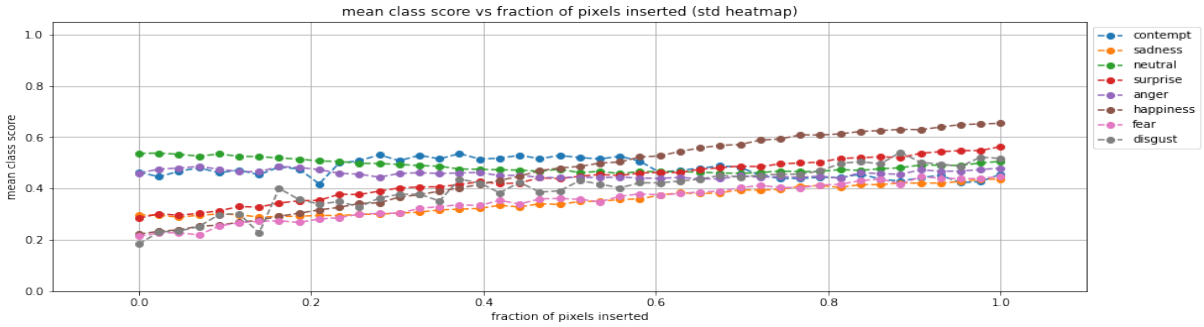
(a) Class-wise insertion curves for the mean explanation representations obtained using GBP.



(b) Class-wise insertion curves for the standard deviation explanation representations obtained using GBP.

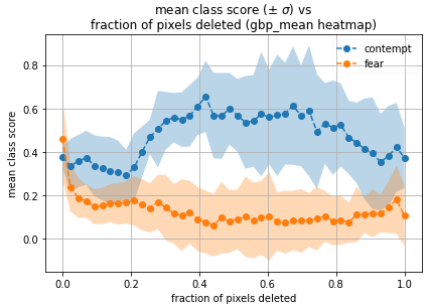


(c) Class-wise insertion curves for the mean explanation representations obtained using IG.

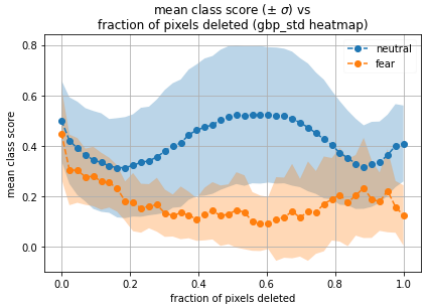


(d) Class-wise insertion curves for the standard deviation explanation representations obtained using IG.

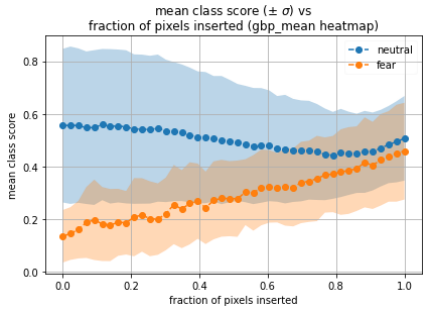
Figure C.18: Insertion curves for Dropout with FER+ images.



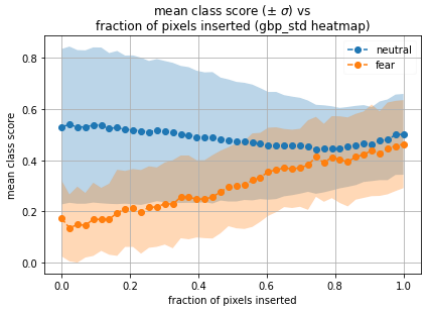
(a) Plots with max. and min. AUC in Figure C.17a.



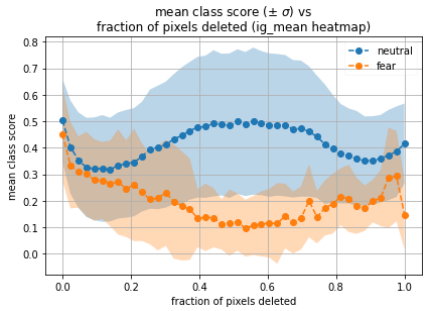
(b) Plots with max. and min. AUC in Figure C.17b.



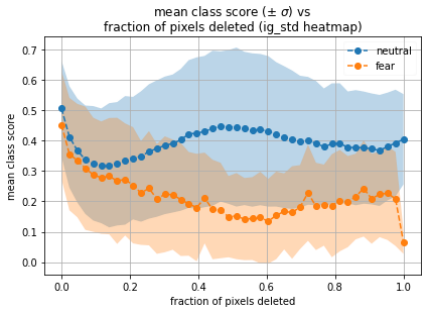
(c) Plots with max. and min. AUC in Figure C.18a.



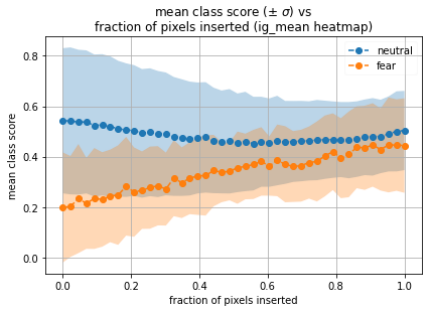
(d) Plots with max. and min. AUC in Figure C.18b.



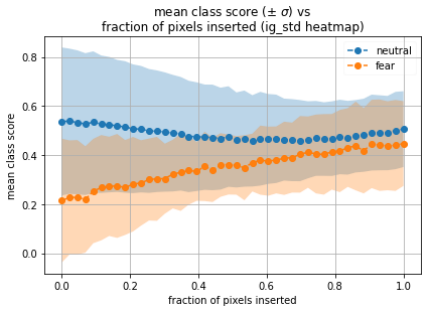
(e) Plots with max. and min. AUC in Figure C.17c.



(f) Plots with max. and min. AUC in Figure C.17d.

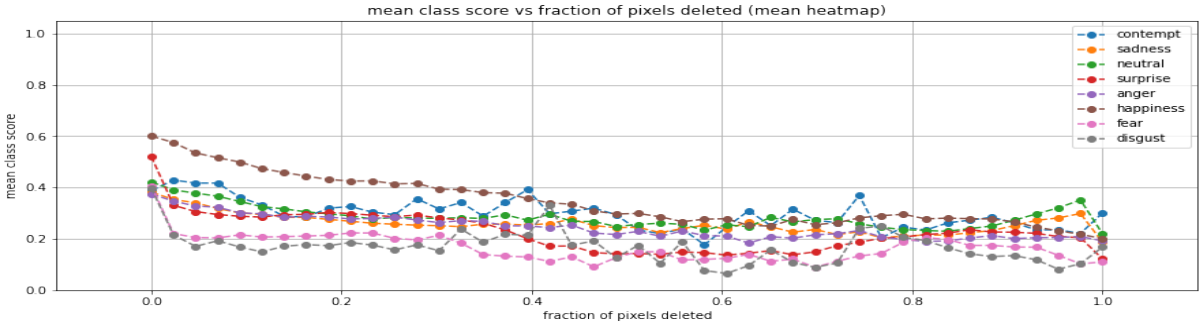


(g) Plots with max. and min. AUC in Figure C.18c.

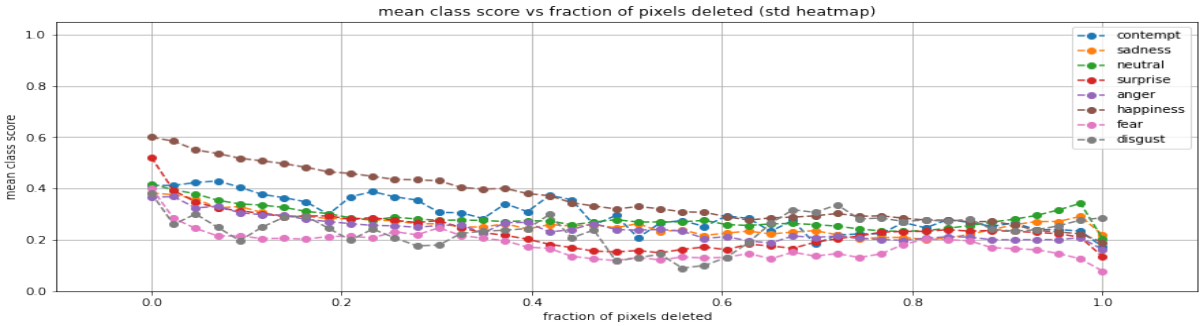


(h) Plots with max. and min. AUC in Figure C.18d.

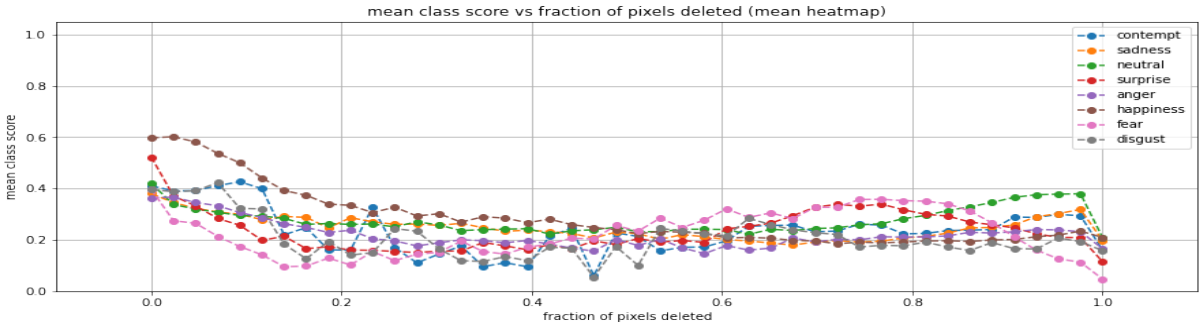
Figure C.19: Plots with maximum and minimum AUC along with their ±1 standard deviation envelopes for Dropout on FER+ images.



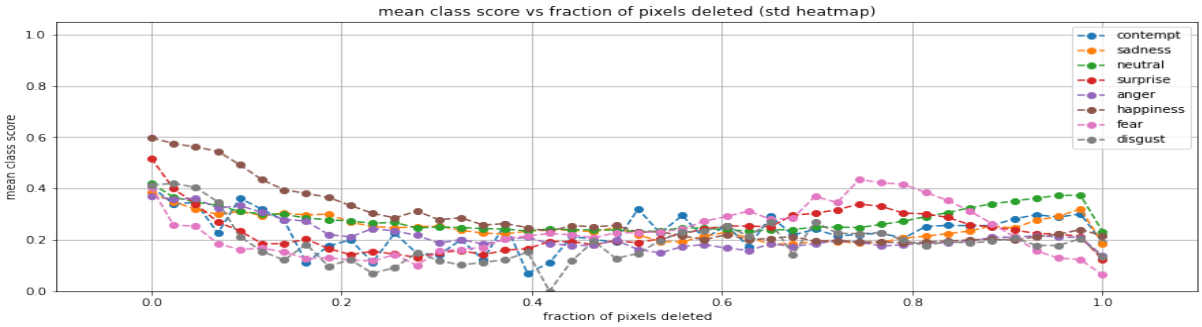
(a) Class-wise deletion curves for the mean explanation representations obtained using GBP.



(b) Class-wise deletion curves for the standard deviation explanation representations obtained using GBP.

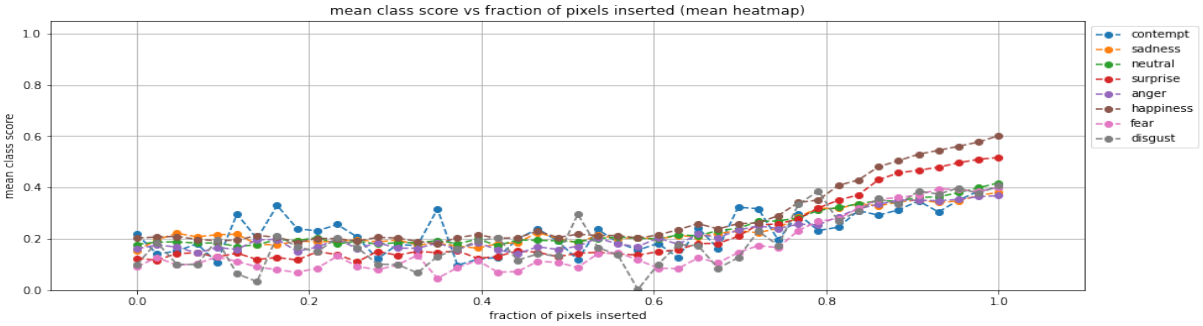


(c) Class-wise deletion curves for the mean explanation representations obtained using IG.

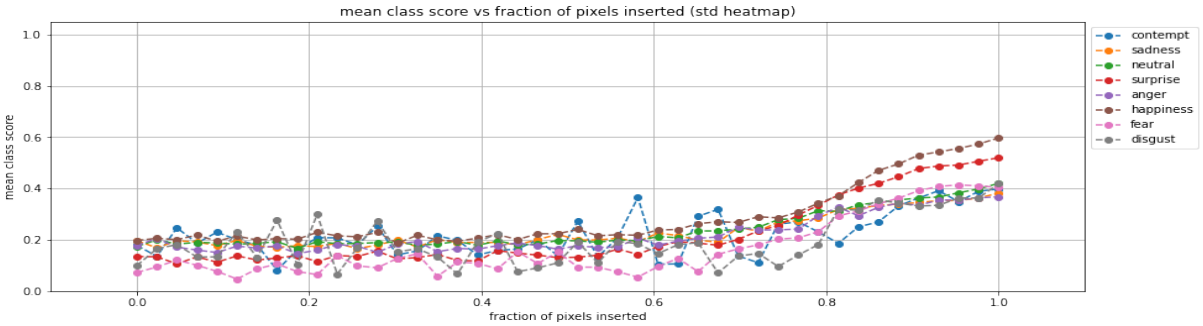


(d) Class-wise deletion curves for the standard deviation explanation representations obtained using IG.

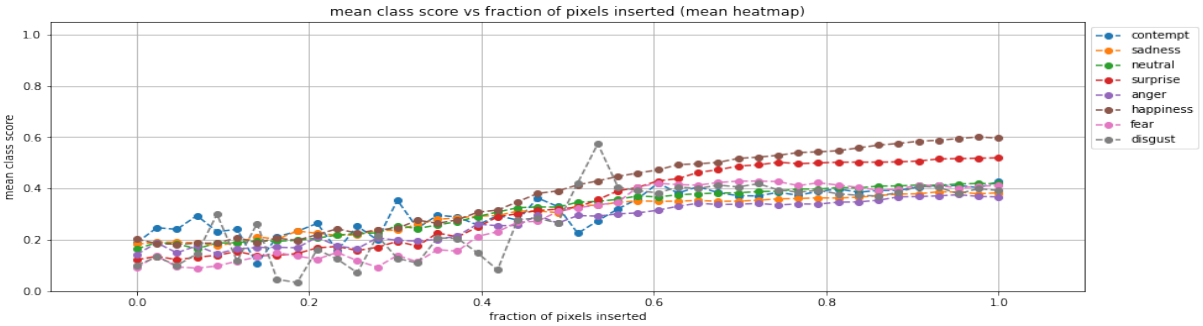
Figure C.20: Deletion curves for DropConnect with FER+ images.



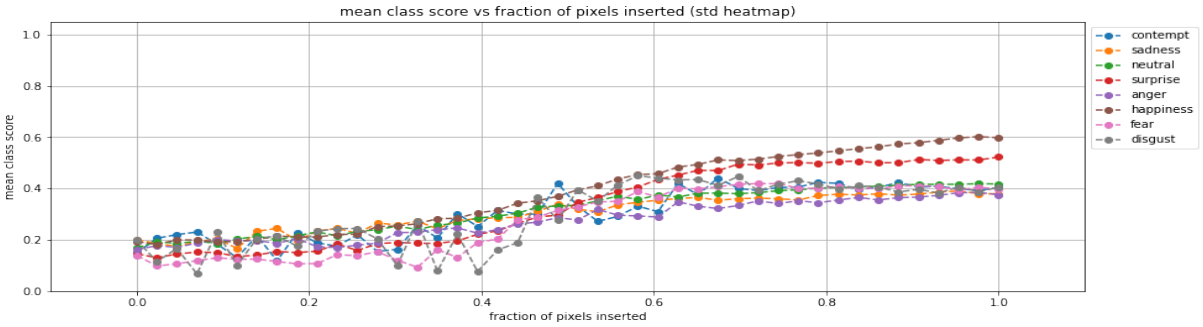
(a) Class-wise insertion curves for the mean explanation representations obtained using GBP.



(b) Class-wise insertion curves for the standard deviation explanation representations obtained using GBP.

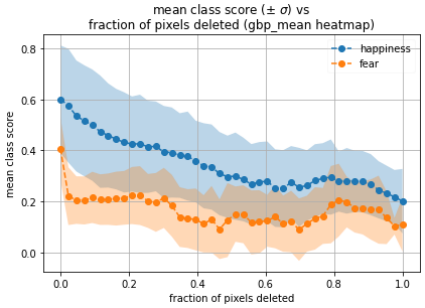


(c) Class-wise insertion curves for the mean explanation representations obtained using IG.

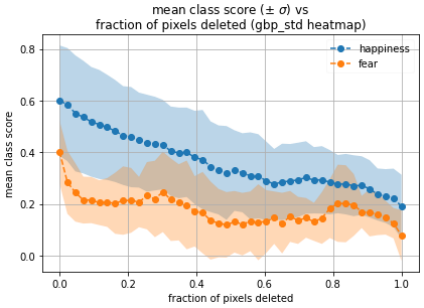


(d) Class-wise insertion curves for the standard deviation explanation representations obtained using IG.

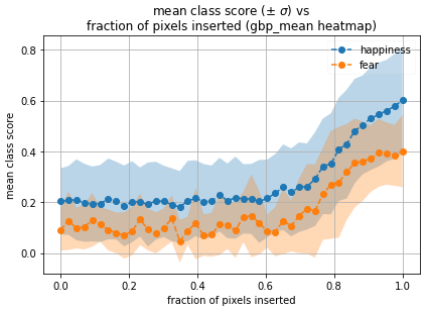
Figure C.21: Insertion curves for DropConnect with FER+ images.



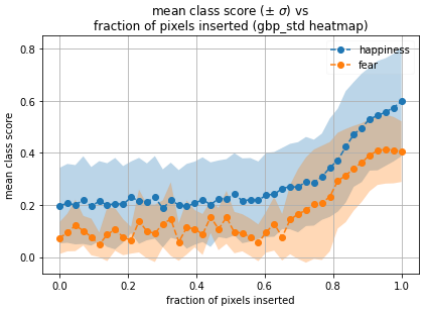
(a) Plots with max. and min. AUC in Figure C.20a.



(b) Plots with max. and min. AUC in Figure C.20b.

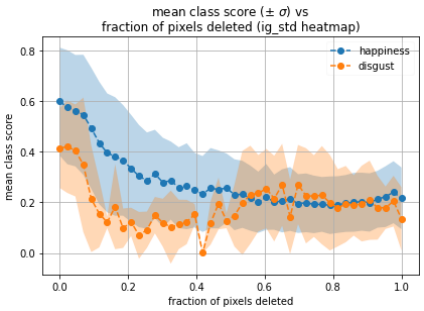


(c) Plots with max. and min. AUC in Figure C.21a.

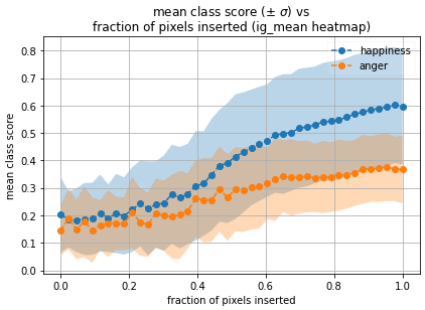


(d) Plots with max. and min. AUC in Figure C.21b.

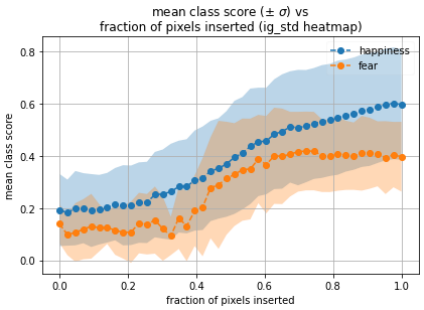
(e) Plots with max. and min. AUC in Figure C.20c.



(f) Plots with max. and min. AUC in Figure C.20d.

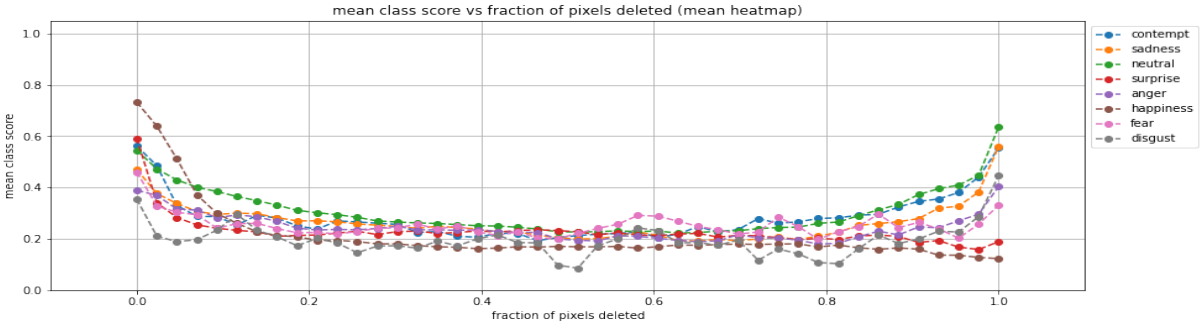


(g) Plots with max. and min. AUC in Figure C.21c.

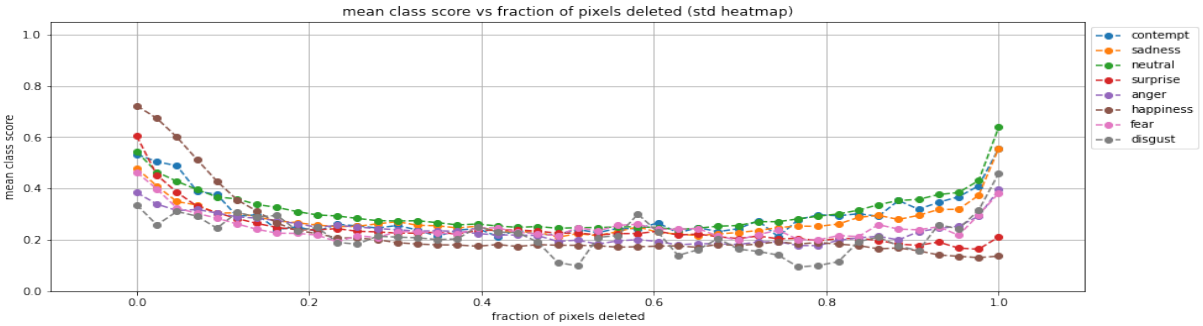


(h) Plots with max. and min. AUC in Figure C.21d.

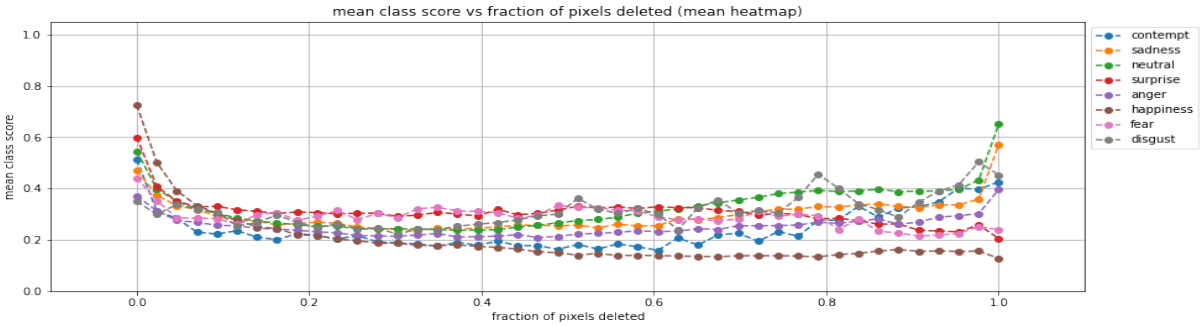
Figure C.22: Plots with maximum and minimum AUC along with their ±1 standard deviation envelopes for DropConnect on FER+ images.



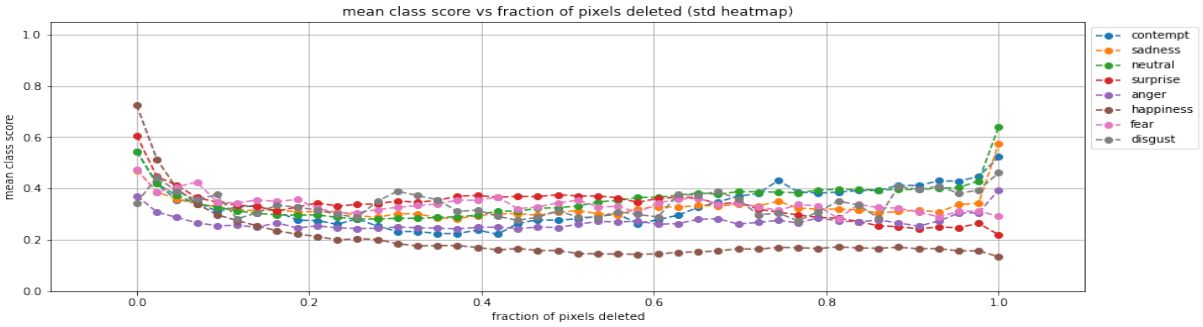
(a) Class-wise deletion curves for the mean explanation representations obtained using GBP.



(b) Class-wise deletion curves for the standard deviation explanation representations obtained using GBP.

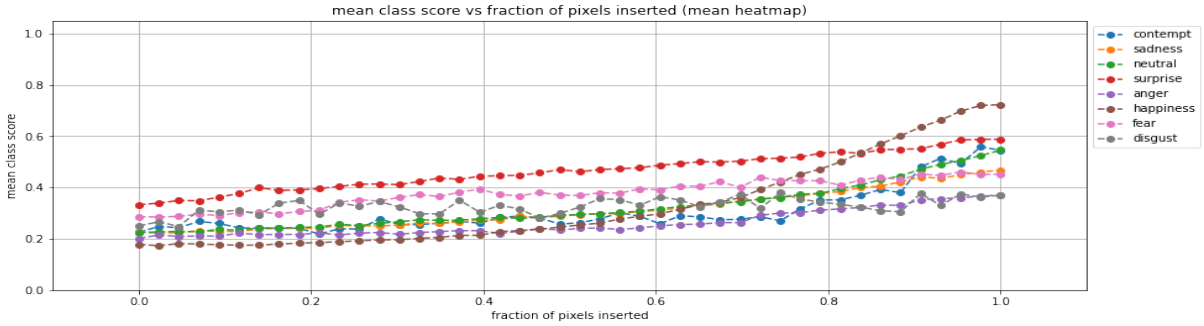


(c) Class-wise deletion curves for the mean explanation representations obtained using IG.

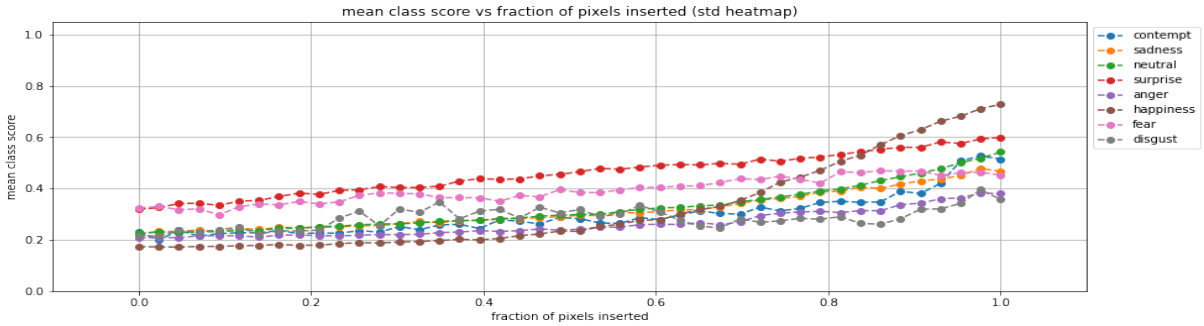


(d) Class-wise deletion curves for the standard deviation explanation representations obtained using IG.

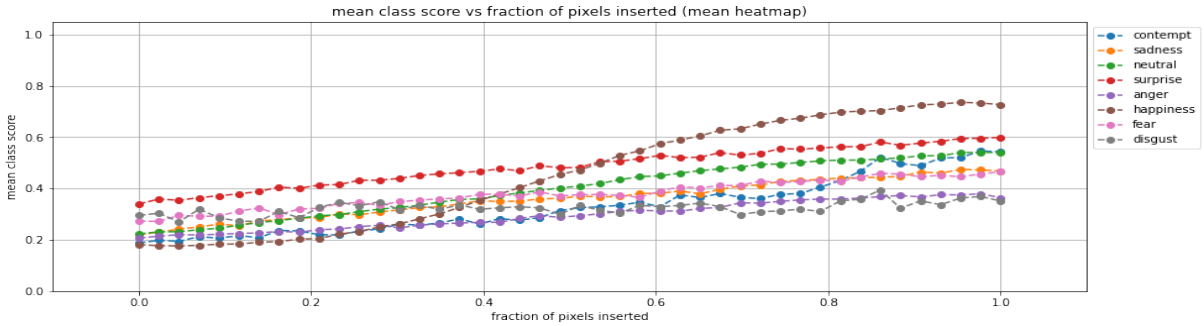
Figure C.23: Deletion curves for Flipout with FER+ images.



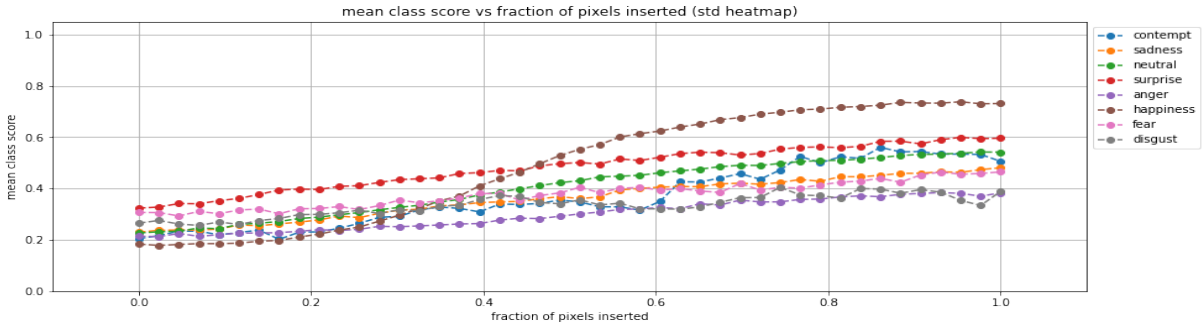
(a) Class-wise insertion curves for the mean explanation representations obtained using GBP.



(b) Class-wise insertion curves for the standard deviation explanation representations obtained using GBP.

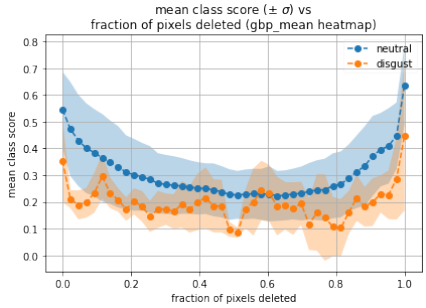


(c) Class-wise insertion curves for the mean explanation representations obtained using IG.

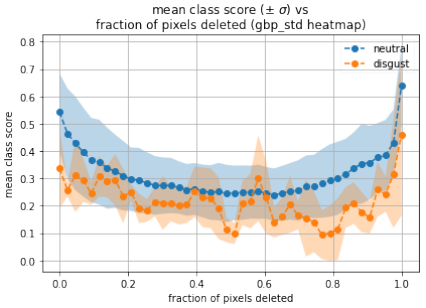


(d) Class-wise insertion curves for the standard deviation explanation representations obtained using IG.

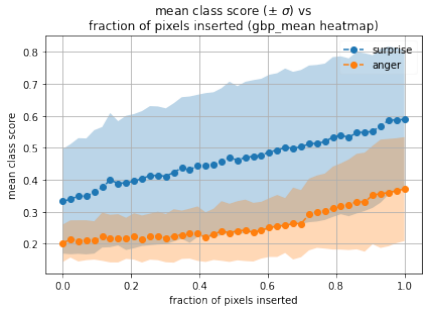
Figure C.24: Insertion curves for Flipout with FER+ images.



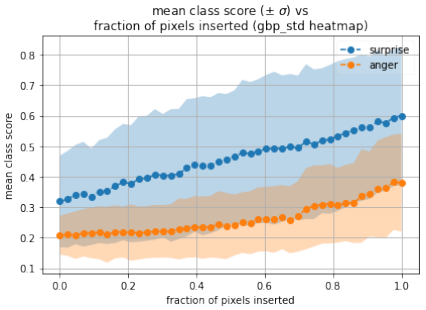
(a) Plots with max. and min. AUC in Figure C.23a.



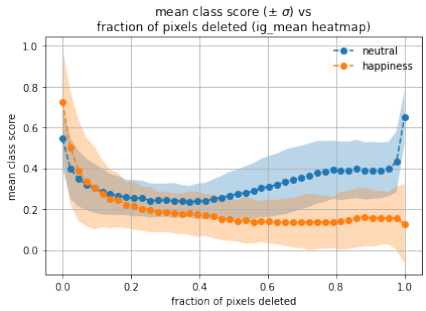
(b) Plots with max. and min. AUC in Figure C.23b.



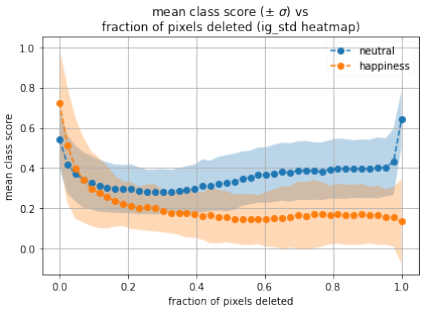
(c) Plots with max. and min. AUC in Figure C.24a.



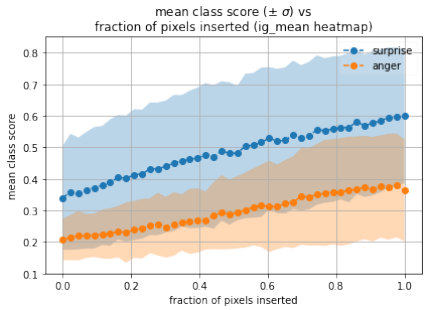
(d) Plots with max. and min. AUC in Figure C.24b.



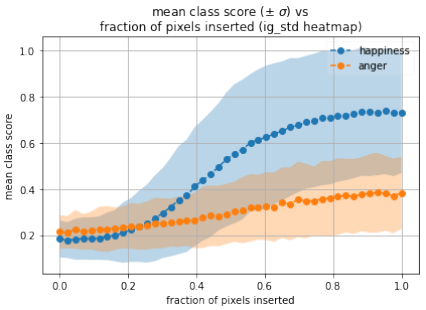
(e) Plots with max. and min. AUC in Figure C.23c.



(f) Plots with max. and min. AUC in Figure C.23d.



(g) Plots with max. and min. AUC in Figure C.24c.



(h) Plots with max. and min. AUC in Figure C.24d.

Figure C.25: Plots with maximum and minimum AUC along with their ±1 standard deviation envelopes for Flipout on FER+ images.

References

[1] California housing data set description. URL https://developers.google.com/machine-learning/crash-course/california-housing-data-description.

[2] Tensorflow tutorials : Integrated gradients. URL https://www.tensorflow.org/tutorials/interpretability/integrated_gradients.

[3] Bayesian Neural Networks, 2022. URL https://www.cs.toronto.edu/%7Eduvenaud/distill_bayes_net/public/.

[4] Shapley Values, 2022. URL https://c3.ai/glossary/data-science/shapley-values/.

[5] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 2021.

[6] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160, 2018.

[7] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/294a8ed24b1ad22ec2e7efea049b8737-Paper.pdf.

[8] Alma Y Alanis, Nancy Arana-Daniel, and Carlos Lopez-Franco. *Artificial neural networks for engineering applications*. Academic Press, 2019.

[9] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Sy21R9JAW.

[10] Teresa Araújo, Guilherme Aresta, Adrian Galdran, Pedro Costa, Ana Maria Mendonça, and Aurélio Campilho. Uolo-automatic object detection and segmentation in biomedical images. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 165–173. Springer, 2018.

[11] Octavio Arriaga, Matias Valdenegro-Toro, Mohandass Muthuraja, Sushma Devaramani, and Frank Kirchner. Perception for Autonomous Systems (PAZ). *Computing Research Repository (CoRR)*, abs/2010.14541, 2020. URL https://arxiv.org/abs/2010.14541.

[12] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

[13] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[14] Emad Barsoum, Cha Zhang, Cristian Canton Ferrer, and Zhengyou Zhang. Training deep networks for facial expression recognition with crowd-sourced label distribution. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, pages 279–283, 2016.

[15] Valérie Beaudouin, Isabelle Bloch, David Bounie, Stéphan Clémençon, Florence d'Alché Buc, James Eagan, Winston Maxwell, Pavlo Mozharovskyi, and Jayneel Parekh. Flexible and context-specific ai explainability: a multidisciplinary approach. *Available at SSRN 3559477*, 2020.

[16] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

[17] Wray L. Buntine and Andreas S. Weigend. Bayesian back-propagation. *Complex Syst.*, 5, 1991.

[18] Kirill Bykov, Marina M-C Höhne, Klaus-Robert Müller, Shinichi Nakajima, and Marius Kloft. How much can i trust you?–quantifying uncertainties in explaining neural networks. *arXiv preprint arXiv:2006.09000*, 2020.

[19] Kirill Bykov, Marina M-C Höhne, Adelaida Creosteanu, Klaus-Robert Müller, Frederick Klauschen, Shinichi Nakajima, and Marius Kloft. Explaining bayesian neural networks. *arXiv preprint arXiv:2108.10346*, 2021.

[20] Captum. Attribution Algorithm Comparison Matrix. Captum, 2020. URL `https://captum.ai/docs/algorithms_comparison_matrix`.

[21] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 839–847. IEEE, 2018.

[22] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2): 303–338, 2010.

[23] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[24] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. A survey of uncertainty in deep neural networks. *arXiv preprint arXiv:2107.03342*, 2021.

[25] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE, 2018.

[26] Ian J Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, et al. Challenges in representation learning: A report on three machine learning contests. In *International conference on neural information processing*, pages 117–124. Springer, 2013.

[27] Alex Graves. Practical variational inference for neural networks. *Advances in neural information processing systems*, 24, 2011.

[28] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.

[29] David Gunning, Mark Stefik, Jaesik Choi, Timothy Miller, Simone Stumpf, and Guang-Zhong Yang. Xai—explainable artificial intelligence. *Science Robotics*, 4(37), 2019.

[30] Miriam Hägele, Philipp Seegerer, Sebastian Lapuschkin, Michael Bockmayr, Wojciech Samek, Frederick Klauschen, Klaus-Robert Müller, and Alexander Binder. Resolving challenges in deep learning-based analyses of histopathological images using explanation methods. *Scientific reports*, 10(1):1–12, 2020.

[31] Andreas Holzinger, Georg Langs, Helmut Denk, Kurt Zatloukal, and Heimo Müller. Causability and explainability of artificial intelligence in medicine. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(4):e1312, 2019.

[32] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021.

[33] Andrei Kapishnikov, Subhashini Venugopalan, Besim Avci, Ben Wedin, Michael Terry, and Tolga Bolukbasi. Guided integrated gradients: An adaptive path method for removing noise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5050–5058, 2021.

[34] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017.

[35] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280. Springer, 2019.

[36] Frederick Klauschen, K-R Müller, Alexander Binder, Michael Bockmayr, M Hägele, P Seegerer, Stephan Wienert, Giancarlo Pruneri, S De Maria, S Badve, et al. Scoring of tumor-infiltrating lymphocytes: From visual estimation to machine learning. In *Seminars in cancer biology*, volume 52, pages 151–157. Elsevier, 2018.

[37] Maximilian Kohlbrenner, Alexander Bauer, Shinichi Nakajima, Alexander Binder, Wojciech Samek, and Sebastian Lapuschkin. Towards best practice in explaining neural network decisions with lrp. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2020.

[38] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[39] Solomon Kullback. *Information theory and statistics*. Courier Corporation, 1997.

[40] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016.

[41] Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Unmasking clever hans predictors and assessing what machines really learn. *Nature communications*, 10(1):1–8, 2019.

[42] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2014.

[43] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1):18, 2020.

[44] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, pages 4768–4777, 2017.

[45] Markus M. Geipel. Lecture-6: Uncertainty in neural networks, 2017. URL `https://www.dbs.ifi.lmu.de/Lehre/DLAI/WS18-19/script/06_uncertain.pdf`.

[46] Maryam Matin and Matias Valdenegro-Toro. Hey human, if your facial emotions are uncertain, you should use bayesian neural networks! *arXiv preprint arXiv:2008.07426*, 2020.

[47] Matthew McAteer. A quick intro to bayesian neural networks. *matthewmcateer.me*, 2019. URL `https://matthewmcateer.me/blog/a-quick-intro-to-bayesian-neural-networks/`.

[48] Aryan Mobiny, Pengyu Yuan, Supratik K Moulik, Naveen Garg, Carol C Wu, and Hien Van Nguyen. Dropconnect is effective in modeling uncertainty of bayesian deep networks. *Scientific reports*, 11 (1):1–14, 2021.

[49] Christoph Molnar. *Interpretable Machine Learning*. 2nd edition, 2022. URL `christophm.github.io/interpretable-ml-book/`.

[50] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018.

[51] Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. Layer-wise relevance propagation: an overview. *Explainable AI: interpreting, explaining and visualizing deep learning*, pages 193–209, 2019.

[52] Weili Nie, Yang Zhang, and Ankit Patel. A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. In *International Conference on Machine Learning*, pages 3809–3818. PMLR, 2018.

[53] R Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33 (3):291–297, 1997.

[54] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. *arXiv preprint arXiv:1806.07421*, 2018.

[55] 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*, pages 1135–1144, 2016.

[56] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys*, 16:1–85, 2022.

[57] Wojciech Samek and Klaus-Robert Müller. Towards explainable artificial intelligence. In *Explainable AI: interpreting, explaining and visualizing deep learning*, pages 5–22. Springer, 2019.

[58] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673, 2016.

[59] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*, 2017.

[60] Wojciech Samek, Grégoire Montavon, Sebastian Lapuschkin, Christopher J Anders, and Klaus-Robert Müller. Toward interpretable machine learning: Transparent deep neural networks and beyond. *arXiv e-prints*, pages arXiv–2003, 2020.

[61] Wojciech Samek, Grégoire Montavon, Sebastian Lapuschkin, Christopher J Anders, and Klaus-Robert Müller. Explaining deep neural networks and beyond: A review of methods and applications. *Proceedings of the IEEE*, 109(3):247–278, 2021.

[62] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[63] Lloyd S Shapley. A value for n-person games, contributions to the theory of games, 2, 307–317, 1953.

[64] Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713*, 2016.

[65] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153. PMLR, 2017.

[66] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[67] Bachstein Simon. Uncertainty quantification in deep learning - inovex gmbh. 2020. URL `https://www.inovex.de/de/blog/uncertainty-quantification-deep-learning/`.

[68] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations (ICLR) Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1409.1556`.

[69] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *In Workshop at International Conference on Learning Representations*, 2014.

[70] Amitojdeep Singh, Sourya Sengupta, and Vasudevan Lakshminarayanan. Explainable deep learning models in medical image analysis. *Journal of Imaging*, 6(6):52, 2020.

[71] Leon Sixt, Maximilian Granz, and Tim Landgraf. When explanations lie: Why many modified bp attributions fail. In *International Conference on Machine Learning*, pages 9046–9057. PMLR, 2020.

[72] Dylan Slack, Sophie Hilgard, Sameer Singh, and Himabindu Lakkaraju. Reliable post hoc explanations: Modeling uncertainty in explainability. *arXiv preprint arXiv:2008.05030*, 2020.

[73] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[74] Evgeny A Smirnov, Denis M Timoshenko, and Serge N Andrianov. Comparison of regularization methods for imagenet classification with deep convolutional neural networks. *Aasri Procedia*, 6: 89–94, 2014.

[75] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[76] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[77] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[78] Chris Tanner. Bayesian auto-encoders part two, 2022. URL `https://harvard-iacs.github.io/2021-CS109B/lectures/lecture28/presentation/VAE_part2_pp.pdf`.

[79] Ahmed Tealab. Time series forecasting using artificial neural networks methodologies: A systematic review. *Future Computing and Informatics Journal*, 3(2):334–340, 2018.

[80] Erico Tjoa and Cuntai Guan. A survey on explainable artificial intelligence (xai): Toward medical xai. *IEEE transactions on neural networks and learning systems*, 32(11):4793–4813, 2020.

[81] Matias Valdenegro-Toro. keras-uncertainty. GitHub, 2018. URL `https://github.com/mvaldenegro/keras-uncertainty`.

[82] Matias Valdenegro-Toro. Deep sub-ensembles for fast uncertainty estimation in image classification. *arXiv preprint arXiv:1910.08168*, 2019.

[83] Matias Valdenegro-Toro. I find your lack of uncertainty in computer vision disturbing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1263–1272, 2021.

[84] Matias Valdenegro Toro. Uncertainty quantification methods - part ii variational inference and gaussian processes, 2022.

[85] Matias Valdenegro-Toro and Daniel Saromo. A deeper look into aleatoric and epistemic uncertainty disentanglement, 2022. URL `https://arxiv.org/abs/2204.09308`.

[86] Matias Valdenegro-Toro, Octavio Arriaga, and Paul Plöger. Real-time convolutional neural networks for emotion and gender classification. In *ESANN*, 2019.

[87] Dmitry Vetrov and Ekaterina Lobacheva. deepbayes-2019/2. dmitry vetrov - variational inference.pdf at master · bayesgroup/deepbayes-2019, 2022. URL `https://github.com/bayesgroup/deepbayes-2019/blob/master/lectures/day1/2.%20Dmitry%20Vetrov%20-%20Variational%20inference.pdf`.

[88] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pages 1058–1066. PMLR, 2013.

[89] Haofan Wang, Zifan Wang, Mengnan Du, Fan Yang, Zijian Zhang, Sirui Ding, Piotr Mardziel, and Xia Hu. Score-cam: Score-weighted visual explanations for convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 24–25, 2020.

[90] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[91] Yeming Wen, Paul Vicol, Jimmy Ba, Dustin Tran, and Roger Grosse. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *arXiv preprint arXiv:1803.04386*, 2018.

[92] Kristoffer Wickstrøm, Michael Kampffmeyer, and Robert Jenssen. Uncertainty modeling and interpretability in convolutional neural networks for polyp segmentation. In *2018 ieee 28th international workshop on machine learning for signal processing (mlsp)*, pages 1–6. IEEE, 2018.

[93] Kristoffer Wickstrøm, Michael Kampffmeyer, and Robert Jenssen. Uncertainty and interpretability in convolutional neural networks for semantic segmentation of colorectal polyps. *Medical image analysis*, 60:101619, 2020.

[94] Gesa Wiegand, Matthias Schmidmaier, Thomas Weber, Yuanting Liu, and Heinrich Hussmann. I drive-you trust: Explaining driving behavior of autonomous cars. In *Extended abstracts of the 2019 chi conference on human factors in computing systems*, pages 1–6, 2019.

[95] Gal Yona and Daniel Greenfeld. Revisiting sanity checks for saliency maps. *arXiv preprint arXiv:2110.14297*, 2021.

[96] Jason Yosinski, Jeff Clune, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. In *In ICML Workshop on Deep Learning*.

[97] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[98] Liliang Zhang, Liang Lin, Xiaodan Liang, and Kaiming He. Is faster r-cnn doing well for pedestrian detection? In *European conference on computer vision*, pages 443–457. Springer, 2016.

[99] Yujia Zhang, Kuangyan Song, Yiming Sun, Sarah Tan, and Madeleine Udell. "Why Should You Trust My Explanation?" understanding uncertainty in lime explanations. *arXiv preprint arXiv:1904.12991*, 2019.

[100] Xingyu Zhao, Wei Huang, Xiaowei Huang, Valentin Robu, and David Flynn. Baylime: Bayesian local interpretable model-agnostic explanations. In *37th Conference on Uncertainty in Artificial Intelligence (UAI)*, May 2021. URL `https://auai.org/uai2021/`. 37th Conference on Uncertainty in Artificial Intelligence 2021, UAI 2021 ; Conference date: 27-07-2021 Through 30-07-2021.

[101] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.