
CORRECTING CLASSIFICATION: A BAYESIAN FRAMEWORK USING EXPLANATION FEEDBACK TO IMPROVE CLASSIFICATION ABILITIES

Yanzhe Bekkemoen

Department of Computer Science
Norwegian University of Science and Technology
Trondheim, Norway

Helge Langseth

Department of Computer Science
Norwegian University of Science and Technology
Trondheim, Norway

ABSTRACT

Neural networks (NNs) have shown high predictive performance, however, with shortcomings. Firstly, the reasons behind the classifications are not fully understood. Several explanation methods have been developed, but they do not provide mechanisms for users to interact with the explanations. Explanations are social, meaning they are a transfer of knowledge through interactions. Nonetheless, current explanation methods contribute only to one-way communication. Secondly, NNs tend to be overconfident, providing unreasonable uncertainty estimates on out-of-distribution observations. We overcome these difficulties by training a Bayesian convolutional neural network (CNN) that uses explanation feedback. After training, the model presents explanations of training sample classifications to an annotator. Based on the provided information, the annotator can accept or reject the explanations by providing feedback. Our proposed method utilizes this feedback for fine-tuning to correct the model such that the explanations and classifications improve. We use existing CNN architectures to demonstrate the method’s effectiveness on one toy dataset (decoy MNIST) and two real-world datasets (Dogs vs. Cats and ISIC skin cancer). The experiments indicate that few annotated explanations and fine-tuning epochs are needed to improve the model and predictive performance, making the model more trustworthy and understandable.

Keywords Explainable/Interpretable Machine Learning · Bayesian Inference · Deep Learning

1 Introduction

During the 20th century, a horse named Clever Hans was claimed to have performed arithmetic and other intellectual tasks. However, it was later revealed that the horse responded to the trainer’s involuntary body language cues. The trainer was unaware that he provided the horse with hints causing it to display intelligent behavior. This behavior is today referred to as the Clever Hans effect [Pfungst, 1911].

NNs have displayed high predictive performance in many application areas in recent years, but they can focus on irrelevant features, thereby displaying the Clever Hans effect. We define irrelevant features as features not encoding the data’s true underlying relationship. To make NNs and other machine learning (ML) methods interpretable and detect such behavior, several methods for explaining classifications have been developed, specifically for NNs [Zeiler and Fergus, 2013, Simonyan et al., 2014, Selvaraju et al., 2020, Shrikumar et al., 2019] and other model-agnostic methods [Ribeiro et al., 2016]. However, these explanation methods do not offer a way to act on the explanations and remove the Clever Hans effect. As a result, new research has started to explore the possibility of leveraging explanations to remove or prevent the effect [Ross et al., 2017, Erion et al., 2020, Rieger et al., 2020, Teso and Kersting, 2019, Selvaraju et al., 2019]. We will refer to these methods as model correction methods.

Explanations are social, meaning they are a transfer of knowledge through interactions or conversations [Miller, 2019]. Many model correction methods work as one-way communication where annotations of irrelevant features need to be provided before training. However, defining such knowledge about the data before training does not satisfy the

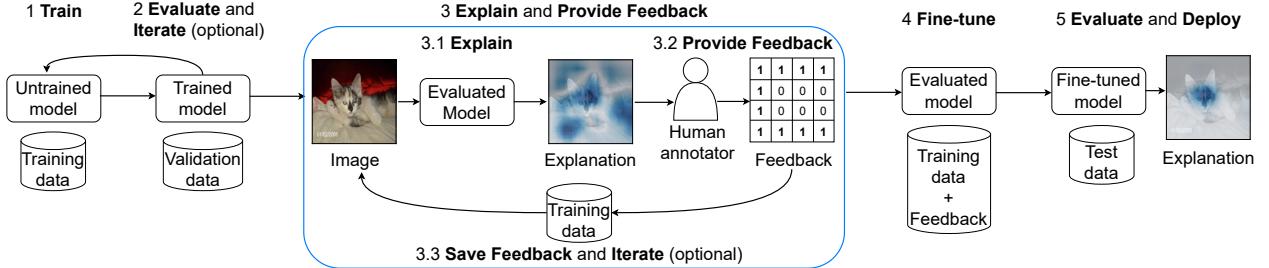


Figure 1: A “standard” ML pipeline with steps for annotating explanations and correcting a model. During Step 3, a model explains training sample classifications to a human annotator who gives feedback on those explanations. A feedback $F^{(i)}$ for a sample i is a matrix of the same width and height as the image. If a feature k, j is irrelevant $F_{k,j}^{(i)} = 1$, otherwise $F_{k,j}^{(i)} = 0$. In Step 4, a model is fine-tuned with training data and feedback. The goal is to improve the reasons behind the classifications (explanation in Step 3 vs. Step 5) and predictive performance.

explanations’ social aspect. Additionally, defining such knowledge without considering what the model will learn through training can be challenging, making model correction methods hard to use in practice.

Humans’ trust in models is affected by a model’s confidence in its predictive performance [Zhang et al., 2020]. NNs are bad at quantifying uncertainty and tend to produce overconfident classifications [Lakshminarayanan et al., 2017]. An overconfident model can be dangerous or offensive [Amodei et al., 2016]. The model can be seen as trying to gain false trust that will backlash against the model if it makes a wrong decision. Consequently, being overconfident can reduce trust in the model rather than increase it. Providing reasonable explanations induce trust [Ribeiro et al., 2016], but explanations generated by these aforementioned explanation methods only capture local behaviors, meaning an explanation applies only in the vicinity of a sample. This does not give insight into how a model will perform on out-of-distribution samples. Hence, to resolve model overconfidence, the model must capture aleatoric and epistemic uncertainty [Lakshminarayanan et al., 2017]. There are several ways to capture epistemic uncertainty; one way is to frame the NNs within a Bayesian framework that applies inference on the model’s weights [MacKay, 1992, Blundell et al., 2015].

Model correction and Bayesian inference contribute to robustness and trust in the model. However, they are difficult to use together because most model correction methods modify the objective function. The modifications do not necessarily follow probability calculus and can introduce elements with unknown distributions.

For a model to have the correct explanation, classification and model confidence, this work bridges the gap between model correction and Bayesian inference. Additionally, it takes a step towards making explanations in ML social. We present a Bayesian framework that uses explanation feedback. After training, a Bayesian CNN presents explanations of training sample classifications to a human annotator. The annotator can accept or reject the explanations by giving feedback as additional evidence. Feedback is represented as a matrix with the same width and height as a sample specifying which attribution regions to reject. This feedback is used during fine-tuning to correct the model such that the explanations and predictive performance improve.

Our main contributions are: (1) A pipeline that provides feedback to a model after training (see Figure 1). This avoids imposing restrictions on the model during the training phase and results in knowledge transfer being interactive and model specific. (2) A Bayesian framework utilizing explanation feedback to correct a model, resulting in a mathematically grounded objective function from a probabilistic view. (3) Experimental results, on one toy dataset and two real-world datasets, that demonstrate the method’s effectiveness. This indicates that few annotated explanations and fine-tuning epochs are required to improve explanations justifying the classifications and the predictive performance.

2 Related Works

Explanation Methods. Several explanation methods have been developed during the past years. These explanation methods can be divided into model-specific vs. model-agnostic and global vs. local scope [Lipton, 2017]. This work builds upon methods that create model-specific local explanations for NNs [Zeiler and Fergus, 2013, Simonyan et al., 2014, Selvaraju et al., 2020, Shrikumar et al., 2019, Sundararajan et al., 2017]. These model-specific local explanation methods produce importance scores assigned to individual features, e.g., pixels for image data or words for textual data. We will refer to the importance scores as feature attributions or attributions for short.

Model Correction Methods. The first work in this area is “Right for the Right Reasons” (RRR) [Ross et al., 2017] that regularize input gradients to suppress irrelevant features in the input space. Similarly, Erion et al. [2020] use Expected Gradients (a modified version of Integrated Gradients [Sundararajan et al., 2017]) to regularize explanations. Used in a similar fashion as RRR, contextual decomposition explanation penalization [Rieger et al., 2020] can regularize feature importance and feature interaction. To encourage NNs to focus on the same input regions as humans, Human Importance-aware Network Tuning (HINT) [Selvaraju et al., 2019] uses gradients in the last convolutional layer (forward direction). HINT encourages NNs to focus on certain regions instead of suppressing attributions. Similar to active learning, but for explanations, Teso and Kersting [2019] proposed using data augmentation to actively correct a model.

Bayesian Inference. Consider a probabilistic model $P(\mathbf{e}|\mathbf{w})$, the prior distribution $P(\mathbf{w})$ that encodes the prior knowledge about the parameter \mathbf{w} and the evidence

$$\mathbf{e} = \{(\mathbf{X}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^n \quad (1)$$

with n independent and identically distributed (i.i.d.) samples. We want to compute the posterior distribution

$$P(\mathbf{w}|\mathbf{e}) = \frac{P(\mathbf{e}|\mathbf{w})P(\mathbf{w})}{P(\mathbf{e})} \quad (2)$$

using Bayes rule. However, exact Bayesian inference is intractable, so Markov Chain Monte Carlo (MCMC) algorithms or variational inference (VI) are used to approximate the weights’ posterior distribution. Although VI is more scalable than MCMC methods, MCMC methods have an asymptotic guarantee that VI methods do not have. Bayes by Backprop (BBB) [Blundell et al., 2015] is a VI and backpropagation-compatible method for learning weight distributions. BBB uses the reparametrization trick [Kingma and Welling, 2014] to calculate a Monte Carlo (MC) estimator of the evidence lower bound (ELBO) and uses Gaussian variational distributions. To compute the MC estimator of ELBO, samples are drawn from the weights, which is computationally expensive. To reduce the number of samples needed, the local reparameterization trick (LRT) [Kingma et al., 2015] was proposed. LRT considers uncertainty at the activation level (before any nonlinear activation function) by computing the activations before drawing samples. This significantly reduces the number of samples drawn and makes it computationally cheaper.

Example 1 (LRT for a fully connected layer) Let layer l be a fully connected layer with weights $\mathbf{W}^{m \times r}$ where $q_{\lambda}(W_{i,j}) = \mathcal{N}(\mu_{W_{i,j}}, \sigma_{W_{i,j}}^2) \forall W_{i,j} \in \mathbf{W}$ and biases $\mathbf{B}^{t \times r}$ where $q_{\lambda}(B_{i,j}) = \mathcal{N}(\mu_{B_{i,j}}, \sigma_{B_{i,j}}^2) \forall B_{i,j} \in \mathbf{B}$. Given an input $\mathbf{Z}^{t \times m}$, the activation of layer l is $\mathbf{U} = \mathbf{Z}\mathbf{W} + \mathbf{B}$ of size $t \times r$ where

$$q_{\lambda}(U_{i,j}|\mathbf{Z}) = \mathcal{N}(\gamma_{i,j}, \delta_{i,j}) \forall U_{i,j} \in \mathbf{U} \quad (3)$$

$$\gamma_{i,j} = \sum_{k=1}^m Z_{i,k} \mu_{W_{k,j}} + \mu_{B_{i,j}}, \quad \delta_{i,j} = \sum_{k=1}^m Z_{i,k}^2 \sigma_{W_{k,j}}^2 + \sigma_{B_{i,j}}^2. \quad (4)$$

A new weight matrix is sampled for every training sample to reduce estimator variance, which has a computational complexity of $\mathcal{O}(tmr)$. By sampling activations rather than weights, the computational complexity is reduced to $\mathcal{O}(tr)$.

Example 2 (LRT for a convolutional layer) Let layer l be a convolutional layer with a convolutional filter $\mathbf{W}^{w' \times h' \times \bar{c}}$ where $q_{\lambda}(\text{vec}(\mathbf{W})) = \mathcal{N}(\text{vec}(\mathbf{M}), \text{diag}(\text{vec}(\mathbf{V}^2)))$ is a fully factorized Gaussian. Given an input $\mathbf{Z}^{\bar{w} \times \bar{h} \times \bar{c}}$, the activation of layer l is $\mathbf{U} = \mathbf{Z} * \mathbf{W}$ of size $\hat{w} \times \hat{h}$ where

$$q_{\lambda}(\text{vec}(\mathbf{U})|\mathbf{Z}) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{U}}, \boldsymbol{\sigma}_{\mathbf{U}}^2) \quad (5)$$

$$\boldsymbol{\mu}_{\mathbf{U}} = \text{vec}(\mathbf{Z} * \mathbf{M}), \quad \boldsymbol{\sigma}_{\mathbf{U}}^2 = \text{diag}(\text{vec}(\mathbf{Z}^2 * \mathbf{V}^2)) \quad (6)$$

is treated as a fully factorized Gaussian. We define $\text{vec}(\cdot)$ as a vectorization function, $(\cdot)^2$ as an element-wise operator, $*$ as a convolution operator and $\text{diag}(\cdot)$ as a diagonal matrix.

3 Variational Inference and Explanation Feedback

Section 3.1 outlines the pipeline to construct feedback and correct a model (Step 3 and 4 in Figure 1). Sections 3.2 and 3.3 outline the objective function used in the fine-tuning phase and how feedback is induced into a model by adding additional evidence to \mathbf{e} .

3.1 Feedback Pipeline

There might be prior knowledge available about a dataset, e.g., digits are centered in the Modified National Institute of Standards and Technology (MNIST) [LeCun et al., 1989] dataset. In this case, we can make a model focus on the center and ignore the edges during training. However, this kind of knowledge might not be needed and can unnecessarily regularize the model. Moreover, constructing such knowledge without knowing what the model will learn through training can be difficult. If irrelevant features were known before training, training data could be preprocessed instead of introducing a new technique. We propose a pipeline where, unlike most previous work, the model asks for feedback after training. By asking for feedback after training, we can define knowledge specifically targeted at the model.

The pipeline adds two additional steps to a “standard” ML pipeline (Step 3 and 4 in Figure 1). During Step 3, a human annotator provides explanation feedback on training sample classifications. For explanation-generation, any saliency-based methods can be used, e.g., Gradient-weighted Class Activation Mapping (Grad-CAM) [Selvaraju et al., 2020]. To determine samples for annotation, active learning sampling strategies can be used. However, it is not obvious how to measure explanations’ informativeness, so we leave this issue for future work. During the experiments, samples are annotated uniformly at random because the goal of our analysis is a proof of concept and not a fully specified pipeline.

As for Step 4, a model is fine-tuned using feedback and training data. The evidence and the objective function used for fine-tuning will be described in the coming sections.

3.2 Add Explanation Feedback to Evidence

We want to compute the posterior distribution $P(\mathbf{w}|e)$ of the weights of a NN. The evaluation involves computation of intractable integrals. Therefore, approximation techniques are used. VI is one of the most widely used and defines a variational distribution $q_{\lambda}(\mathbf{w})$ that is used to approximate the posterior distribution by minimizing the Kullback–Leibler (KL) divergence $D_{\text{KL}}(q_{\lambda}(\mathbf{w})\|P(\mathbf{w}|e))$. Minimizing the KL divergence is equivalent to maximizing the ELBO

$$\mathcal{L}_{\text{ELBO}} = \underbrace{\mathbb{E}_{q_{\lambda}(\mathbf{w})}[\log P(\mathbf{e}|\mathbf{w})]}_{\text{Likelihood}} - \underbrace{D_{\text{KL}}(q_{\lambda}(\mathbf{w})\|P(\mathbf{w}))}_{\text{Complexity}}. \quad (7)$$

Let $\mathbf{X}^{(i)} \in \mathbb{R}^{w \times h \times c}$ be an observation with the label $\mathbf{y}^{(i)}$, a feedback of the sample i is a matrix $\mathbf{F}^{(i)} \in \{0, 1\}^{w \times h}$ given by a human annotator. If a feature k, j is irrelevant $\mathbf{F}_{k,j}^{(i)} = 1$, otherwise $\mathbf{F}_{k,j}^{(i)} = 0$. Furthermore, let f be a Bayesian CNN that takes $\mathbf{X}^{(i)}$ as input and outputs the logit vector $f(\mathbf{X}^{(i)}) = \hat{\mathbf{y}}^{(i)}$.

Consider input gradients $\nabla_{\mathbf{X}^{(i)}} \sum_j \hat{y}_j^{(i)}$ of the observation $\mathbf{X}^{(i)}$ that can be used as an explanation that specifies feature attributions.

$$\mathbf{H}^{(i)} = \mathbf{F}^{(i)} \otimes (\nabla_{\mathbf{X}^{(i)}} \sum_j \hat{y}_j^{(i)}) \quad (8)$$

is the attributions on irrelevant features that we want to minimize. \otimes signifies broadcasting and an element-wise product. To include explanation feedback, we add additional evidence to e

$$e_{\text{grad}} = e \cup \{\mathbf{H}^{(i)} = \mathbf{0}\}_{i=1}^n \quad (9)$$

where we set $\mathbf{H}^{(i)} = \mathbf{0}$ to constrain attributions on irrelevant features, which is similar to previous work [Ross et al., 2017].

However, the distribution of the input gradients is unknown and it is hard to find analytically. Not knowing the distribution makes us unable to calculate the likelihood in Equation (7) with e_{grad} as evidence. One way to solve this issue is to look at some quantity with known distribution instead of the input gradients. Although weight distributions are known, it is not trivial to map feedback given in the input space to the parameter space. Therefore, we use the LRT to consider the activation distributions and map the feedback to activation space. Examples 1 and 2 describe how to obtain activation distributions.

In the next section, we show how considering the uncertainty at the activation level enables us to approximate the evidence e_{grad} with an alternative formulation for which the likelihood can be expressed analytically.

We use CNNs since the activations in convolutional layers retain spatial information. The spatial information is needed to map feedback defined in the input space to corresponding activations. An additional advantage of using activations rather than input gradients is reduced computational complexity since second-order partial derivatives are not needed.

3.3 Methodology

This section details how to add feedback to e using activations rather than input gradients. We do this by describing how to map the feedback from input to activation space. Moreover, we describe how to compute Equation (7) with the new evidence e_{act} (to be defined in Equation (13)); something we could not do with e_{grad} .

Consider the same setup as in Section 3.2. Let $\mathbf{A}^{(i)} \in \mathbb{R}^{u \times v \times d}$ ($u \leq w, v \leq h$) be feature maps (before any nonlinear activation function) of the last convolutional layer (forward direction) obtained by forward passing $\mathbf{X}^{(i)}$. The last convolutional layer is used since it contains the highest abstraction level of features [Zeiler and Fergus, 2013, Zhou et al., 2015, Selvaraju et al., 2019].

3.3.1 Map Feedback from Input to Activation Space

The feedback will be mapped to the activation space to include it in the evidence e . We do this in two steps: (1) downsample the feedback to the width and height of the feature maps $\mathbf{A}^{(i)}$ and (2) find activations in $\mathbf{A}^{(i)}$ spatially overlapping with the feedback and affecting the classification.

We downsample the feedback $\mathbf{F}^{(i)}$ to the width and height of $\mathbf{A}^{(i)}$ by computing

$$\mathbf{J}^{(i)} = \text{AdaptiveMaxPool}(\mathbf{F}^{(i)}, (u, v)). \quad (10)$$

The function $\text{AdaptiveMaxPool}(\mathbf{Z}, \mathbf{o})$ takes a matrix \mathbf{Z} and uses max-pooling to transform it to size $\mathbf{o} = (o_1, o_2)$. This operation is similar to how Grad-CAM interpolates attribution maps to the input space to visualize explanations. Next, we find entries in $\mathbf{J}^{(i)}$ that overlap with $\mathbf{A}^{(i)}$ by calculating

$$\mathbf{G}^{(i)} = \mathbf{A}^{(i)} \otimes \mathbf{J}^{(i)}. \quad (11)$$

CNNs often use rectified linear unit and max-pooling, implying that classifications are only affected by a subset of the activations. Therefore, activations not used in the classification will be excluded by computing

$$\mathbf{L}^{(i)} = \mathbf{G}^{(i)} \odot \mathbf{1}_{(\nabla_{\mathbf{A}^{(i)}} \sum_j \hat{y}_j^{(i)}) \neq 0} \quad (12)$$

to zero out entries in $\mathbf{G}^{(i)}$ that do not affect the classification. $\mathbf{L}^{(i)}$ is a tensor representing the activations that affect the classification and overlap with all indices k, j where $\mathbf{J}_{k,j}^{(i)} = 1$. The goal is to find the random variables where values in $\mathbf{L}^{(i)} \neq 0$ are drawn from. We will refer to those activations as $\mathbf{a}_{\mathcal{F}}^{(i)}$. The remaining activations in the network will be denoted $\mathbf{a}_{\mathcal{F}^c}^{(i)}$. We substitute input gradients with activations and add feedback to e by defining

$$e_{\text{act}} = e \cup \{\mathbf{a}_{\mathcal{F}}^{(i)} = \mathbf{0}\}_{i=1}^n \quad (13)$$

The process of finding $\mathbf{a}_{\mathcal{F}}^{(i)}$ incurs an extra forward pass.

3.3.2 Compute the ELBO with Additional Evidence

In this section, we show how to compute the ELBO with e_{act} . We follow Kingma et al. [2015] and consider uncertainty at the level of activations. This implies that the expectation in Equation (7) will be computed with respect to $q_{\lambda}(\mathbf{a}_{\mathcal{F}^c}^{(i)}, \mathbf{a}_{\mathcal{F}}^{(i)} | \mathbf{X}^{(i)})$ rather than $q_{\lambda}(\mathbf{w})$. $q_{\lambda}(\mathbf{a}_{\mathcal{F}^c}^{(i)} | \mathbf{X}^{(i)}, \mathbf{a}_{\mathcal{F}}^{(i)} = \mathbf{0})$ will be written as $q_{\lambda}(\mathbf{a}_{\mathcal{F}^c}^{(i)})$ for short. $\mathcal{L}_{\text{ELBO}}(\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(n)}) = \sum_{i=1}^n \mathcal{L}_{\text{ELBO}}^{(i)}(\mathbf{X}^{(i)})$ since the samples are i.i.d. We can reformulate Equation (7) as a loss function using activations and the evidence e_{act} for a single sample i as

$$\mathcal{L}_{\text{LRT}}^{(i)} = \frac{1}{n} D_{\text{KL}}(q_{\lambda}(\mathbf{w}) \| P(\mathbf{w})) - \mathbb{E}_{q_{\lambda}(\mathbf{a}_{\mathcal{F}^c}^{(i)})} [\log P(e_{\text{act}}^{(i)} | \mathbf{a}_{\mathcal{F}^c}^{(i)})]. \quad (14)$$

Expanding $e_{\text{act}}^{(i)}$ in the log probability in Equation (14) gives

$$\begin{aligned} \log(e_{\text{act}}^{(i)} | \mathbf{a}_{\mathcal{F}^c}^{(i)}) &= \log P(\mathbf{y}^{(i)}, \mathbf{a}_{\mathcal{F}}^{(i)} = \mathbf{0} | \mathbf{X}^{(i)}, \mathbf{a}_{\mathcal{F}^c}^{(i)}) \\ &= \log P(\mathbf{y}^{(i)} | \mathbf{a}_{\mathcal{F}}^{(i)} = \mathbf{0}, \mathbf{a}_{\mathcal{F}^c}^{(i)}) + \log P(\mathbf{a}_{\mathcal{F}}^{(i)} = \mathbf{0} | \mathbf{X}^{(i)}, \mathbf{a}_{\mathcal{F}^c}^{(i)}) \end{aligned} \quad (15)$$

where $\mathbf{a}_{\mathcal{F}^c}^{(i)}$ and $\mathbf{a}_{\mathcal{F}}^{(i)}$ are treated as fully factorized Gaussians. To compute Equation (14), we follow the result of BBB and approximate the negative log-likelihood by applying MC sampling. When both $q_{\lambda}(\mathbf{w})$ and $P(\mathbf{w})$ are fully factorized Gaussians, the complexity term can be calculated in closed form [Kingma and Welling, 2014]. Equation (14) is an objective function that uses the evidence e_{act} to incorporate feedback from an annotator and is theoretically justified from a probabilistic perspective, making it mathematically grounded. Moreover, we do not need a hyperparameter that deals with feedback penalty; instead, the objective function trades-off between complexity and data.

Dataset	Precision		Recall		F1		Accuracy	
	NF	F	NF	F	NF	F	NF	F
Decoy MNIST	0.725	0.970	0.725	0.970	0.725	0.970	0.725	0.970
Dogs vs. Cats	0.918	0.923	0.857	0.870	0.887	0.896	0.886	0.894
Skin cancer	0.280	0.320	0.904	0.798	0.427	0.457	0.815	0.799
Skin cancer NP	0.289	0.335	0.904	0.798	0.437	0.472	0.702	0.721

Table 1: Performance metrics of the model trained with no feedback (NF) and feedback (F). For decoy MNIST, all of the metrics are calculated using micro average. The skin cancer dataset is tested with and without patch data (NP) and accuracy is computed with macro average recall, also known as balanced accuracy.

Example 3 (How to approximate Equation (14)) Consider $\mathbf{w} \in \mathbb{R}^j$, where the prior $P(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ and the approximate posterior $q_{\lambda}(\mathbf{w}) = \mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2))$ are both fully factorized Gaussians. Hence, Equation (14) can be approximated as

$$\begin{aligned} \mathcal{L}_{LRT}^{(i)} \approx & \frac{1}{2n} \sum_{i=1}^j (\mu_i^2 + \sigma_i^2 - 1 - \log \sigma_i^2) \\ & - \frac{1}{m} \sum_{t=1}^m [\log P(\mathbf{y}^{(i)} | \mathbf{a}_{\mathcal{F}}^{(i,t)} = \mathbf{0}, \mathbf{a}_{\mathcal{F}^c}^{(i,t)}) + \log P(\mathbf{a}_{\mathcal{F}}^{(i,t)} = \mathbf{0} | \mathbf{X}^{(i)}, \mathbf{a}_{\mathcal{F}^c}^{(i,t)})]. \end{aligned} \quad (16)$$

m is the number of MC samples.

To summarize, feedback defined in the input space by an annotator is mapped to the activations. The feedback is included by adding it as additional evidence to e . During Step 2, the model is trained with e as the evidence and Equation (7) as the objective function. As for Step 4, to fine-tune the model, Equation (14) is used as the objective function with e_{act} as the evidence.

4 Experiments and Results

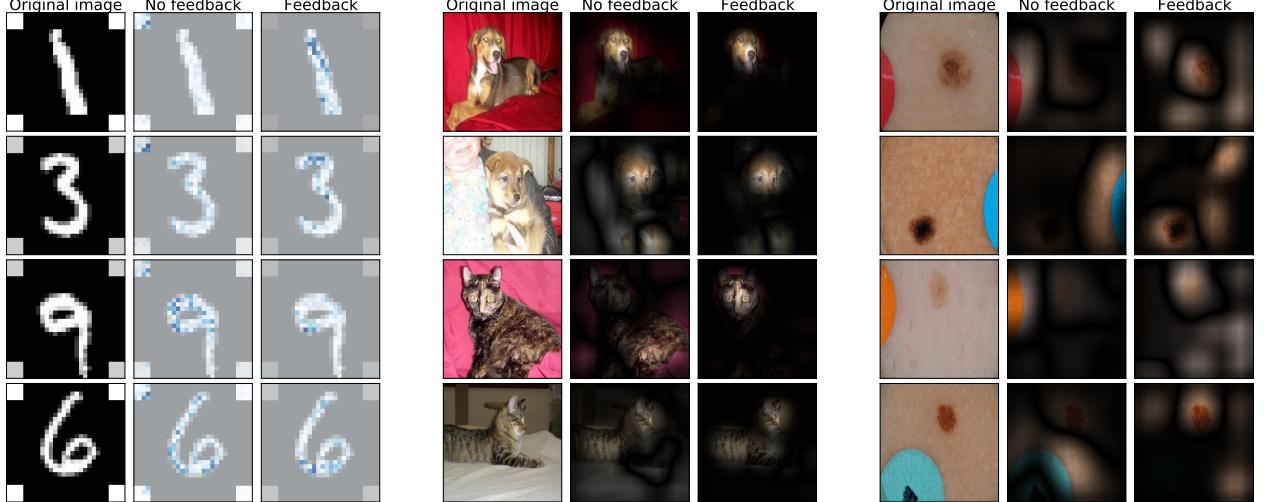
The goal of our experiments is to display the effect of our proposed pipeline and model correction method. We demonstrate our work on one toy dataset, decoy MNIST [Ross et al., 2017] and two real-world datasets, Dogs vs. Cats [Kaggle, 2014] and The International Skin Imaging Collaboration (ISIC) skin cancer [Codella et al., 2019].

The LeNet [Lecun et al., 1998] architecture was used for decoy MNIST. LeNet models were trained for 100 epochs with early stopping. The AlexNet [Krizhevsky et al., 2012] architecture with batch normalization [Ioffe and Szegedy, 2015], but without dropout was used for Dogs vs. Cats and ISIC skin cancer. AlexNet models were trained for 1000 epochs with early stopping. Both LeNet and AlexNet models were saved at the best validation F1 score. All models were trained with a batch size of 256. The Adam optimizer [Kingma and Ba, 2017] with a learning rate of 0.001 and a prior $P(\mathbf{w}) = \mathcal{N}(\mathbf{0}, 0.1^2 \mathbf{I})$ were used. Samples to annotate with feedback were chosen uniformly at random, as stated in Section 3.1.

To quantify attribution changes before and after feedback, attributions overlapping with irrelevant features were computed using four methods (see Table 2): Saliency [Simonyan et al., 2014], Deep Learning Important FeaTures (DeepLIFT) [Shrikumar et al., 2019], Grad-CAM and Occlusion [Zeiler and Fergus, 2013]. Consider an attribution mask $\mathbf{T}^{(i)} \in \mathbb{R}^{w \times h \times v}$ (v is the number of channels that depends on the method and the model) of sample i for one of the attribution methods. k is the number of samples in the test dataset with annotated explanation (not used in the fine-tuning process). The attribution overlap in Table 2 is computed as

$$\frac{1}{k} \sum_{i=1}^k \frac{\|\text{vec}(\mathbf{F}^{(i)} \otimes \mathbf{T}^{(i)})\|_1}{\|\text{vec}(\mathbf{T}^{(i)})\|_1} \quad (17)$$

where $\|\cdot\|_1$ is the ℓ_1 norm operator. The effect of fine-tuning with feedback for a few samples from the test datasets (not used in the fine-tuning process) can be seen in Figure 2.



(a) Decoy MNIST. The model focuses on both the decoys and digits before feedback. After feedback, only the digits are used for classifications.

(b) Dogs vs. Cats. After feedback, the model becomes sharper and focuses more on the animals than the background.

(c) ISIC skin cancer. Before feedback, the model uses irrelevant patches and moles to classify samples. After feedback, it uses only moles on these samples.

Figure 2: Explanations before and after fine-tuning with feedback visualized on samples from test datasets. DeepLIFT was used for decoy MNIST and Grad-CAM for both Dogs vs. Cats and ISIC skin cancer to visualize explanations.

Dataset	Saliency		DeepLIFT		Grad-CAM		Occlusion	
	NF	F	NF	F	NF	F	NF	F
Decoy MNIST	0.080	0.031	0.078	0.019	0.063	0.025	0.181	0.050
Dogs vs. Cats	0.396	0.384	0.405	0.407	0.441	0.379	0.393	0.380
Skin cancer	0.154	0.111	0.145	0.124	0.221	0.071	0.250	0.148

Table 2: Attributions overlapping with irrelevant features averaged over all samples in the test dataset with annotated explanation, as described in Section 4. The attribution overlap score is bounded [0, 1] and a **lower** score is better because it implies less attention is focused on irrelevant features. For Occlusion, a sliding window of size 3×3 was used for decoy MNIST and 23×23 for the other two datasets.

4.1 Decoy MNIST

The MNIST dataset consists of 28×28 grayscale images of digits. The training data has 60000 samples and the test data has 10000 samples. There are ten classes: 0, 1, ..., 9. Decoy MNIST is a modified version of MNIST where every sample in the dataset has a 4×4 square in each corner (see Figure 2a). In the training data, the decoys' colors correspond with the label of the digit y , $(255 - 25y)$ and in the test data, the colors are randomly drawn. This decoy rule is a modified version of the one found in Ross et al. [2017], where only one corner has a decoy and is drawn randomly. The training data was divided into 90% (54000 samples) for training and 10% (6000 samples) for validation. The model was trained for 1 epoch without feedback and fine-tuned for 17 epochs. 0.4% (189 samples) of the training samples were annotated with feedback.

To observe how much of the original predictive performance is recovered through model correction, the model was trained with MNIST (7 epochs) and decoy MNIST. The model trained without decoys has 98.7% accuracy. After adding decoys, the accuracy drops down to 72.5%. By fine-tuning the model with feedback, the difference between the original accuracy is reduced to 1.7%. The decoy rule has no randomness, which makes it more difficult to recover the original accuracy. Most previous works have feedback on all training samples. In contrast, this approach has feedback on only 0.4% of the training data, making it more appealing.

Figure 2a and Table 2 show that without feedback, the model focuses on the decoys. After fine-tuning with feedback, the attributions on the decoys are significantly reduced.

4.2 Dogs vs. Cats

The Dogs vs. Cats dataset consists of RGB images of dogs (label 0) and cats (label 1) (see Figure 2b). All images in the dataset are scaled to 227×227 . It has training data of 25000 samples and test data of 12500 samples, but the latter does not have labels. Thus, only the training data is used and divided as follows: 90% (22500 samples) training, 5% (2500 samples) validation and 5% (2500 samples) test. For the construction of feedback, a pretrained DeepLabV3 ResNet101 [Chen et al., 2017] semantic segmentation network was used. $F_{k,j}^{(i)} = 1$ when index k, j does not overlap with an animal. This tells the model to not focus on background information. The model was trained for 765 epochs without feedback and fine-tuned for 16 epochs. 4.5% (1012 samples) of the training samples were annotated with feedback.

Dogs vs. Cats is the only dataset without any apparent irrelevant features repeating in several samples. Therefore, only a slight improvement in the model’s predictive performance can be seen in Table 1. Furthermore, more feedback data is needed to affect the model. After fine-tuning with feedback, attributions overlap more with the animals (see Figure 2b and Table 2). Interestingly, explaining not to focus on background information results in the model focusing on the animals’ faces instead of the whole body.

4.3 ISIC Skin Cancer

The version of the ISIC skin cancer dataset by Rieger et al. [2020] that we used has 21654 samples where 2284 are malignant (label 1) and 19370 are benign (label 0). Of those benign samples, 48% (9209 samples) have colorful patches (see Figure 2c). Feedback for samples in this dataset are regions that contain those colorful patches because they are irrelevant for the classifications. The dataset was divided into 90% (19488 samples) training, 5% (1083 samples) validation, and 5% (1083 samples) test. The model was trained for 997 epochs without feedback and fine-tuned for 1 epoch. 1.5% (292 samples) of the training samples were annotated with feedback.

The models were tested both with and without patches to observe the patches’ effect. Table 1 shows that with patch data, the model without feedback has better accuracy than the model fine-tuned with feedback. However, without patch data, the model fine-tuned with feedback has better accuracy. Moreover, it is more trustworthy based on the explanations (Figure 2c).

Before feedback, the model uses patches to “cheat” on benign samples. When patches are removed, the model without feedback has the same number of false positives and true negatives, classifying benign samples randomly. In contrast, the model fine-tuned with feedback does not do that. Figure 2c demonstrates that the model with feedback focuses much less on the patches. Table 2 displays the same result quantitatively. Also, the positions of the patches appear not to matter.

5 Conclusion and Future Works

We propose a pipeline and a Bayesian framework using explanation feedback. The pipeline makes it possible for users to interact with a model after training to create explanation feedback (summarized in Figure 1). The users create feedback based on the model’s explanations of training sample classifications. Our Bayesian framework uses this feedback to correct the model so that explanations and predictive performance improve, making the model more trustworthy and understandable. The effectiveness of the proposed approach is shown on one toy dataset and two real-world datasets. The results indicate that focus on dataset biases are minimized, and the features representing the data’s true underlying relationship are targeted more distinctly. Thus, paying less attention to irrelevant features. Furthermore, few annotated explanations and fine-tuning epochs are needed to observe this effect.

We see several opportunities for future works, including: (1) sampling strategies for finding samples to ask for annotation, (2) extending beyond CNN architectures and (3) richer feedback semantics; currently, only rejection of attribution regions are supported.

References

- O. Pfungst. *Clever Hans:(the horse of Mr. Von Osten.) a contribution to experimental animal and human psychology.* Holt, Rinehart and Winston, 1911.
- M. D. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. *arXiv:1311.2901*, 2013. URL <http://arxiv.org/abs/1311.2901>.
- K. Simonyan, A. Vedaldi, and A. Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *arXiv:1312.6034*, 2014. URL <http://arxiv.org/abs/1312.6034>.

- R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. *International Journal of Computer Vision*, 128(2):336–359, 2020. ISSN 0920-5691, 1573-1405. doi:10.1007/s11263-019-01228-7. URL <http://arxiv.org/abs/1610.02391>. arXiv: 1610.02391.
- A. Shrikumar, P. Greenside, and A. Kundaje. Learning Important Features Through Propagating Activation Differences. *arXiv:1704.02685*, 2019. URL <http://arxiv.org/abs/1704.02685>.
- M. T. Ribeiro, S. Singh, and C. Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *arXiv:1602.04938*, 2016. URL <http://arxiv.org/abs/1602.04938>.
- A. S. Ross, M. C. Hughes, and F. Doshi-Velez. Right for the Right Reasons: Training Differentiable Models by Constraining their Explanations. *arXiv:1703.03717*, 2017. URL <http://arxiv.org/abs/1703.03717>.
- G. Erion, J. D. Janizek, P. Sturmels, S. Lundberg, and S. Lee. Improving performance of deep learning models with axiomatic attribution priors and expected gradients. *arXiv:1906.10670*, 2020. URL <http://arxiv.org/abs/1906.10670>.
- L. Rieger, C. Singh, W. J. Murdoch, and Bin Yu. Interpretations are useful: penalizing explanations to align neural networks with prior knowledge. *arXiv:1909.13584*, 2020. URL <http://arxiv.org/abs/1909.13584>.
- S. Teso and K. Kersting. Explanatory Interactive Machine Learning. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, AIES ’19, pages 239–245, New York, NY, USA, January 2019. Association for Computing Machinery. ISBN 978-1-4503-6324-2. doi:10.1145/3306618.3314293. URL <https://doi.org/10.1145/3306618.3314293>.
- R. R. Selvaraju, S. Lee, Y. Shen, H. Jin, S. Ghosh, L. Heck, D. Batra, and D. Parikh. Taking a HINT: Leveraging Explanations to Make Vision and Language Models More Grounded. *arXiv:1902.03751*, 2019. URL <http://arxiv.org/abs/1902.03751>.
- T. Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, February 2019. ISSN 0004-3702. doi:10.1016/j.artint.2018.07.007. URL <http://www.sciencedirect.com/science/article/pii/S0004370218305988>.
- Y. Zhang, Q. V. Liao, and R. K. E. Bellamy. Effect of Confidence and Explanation on Accuracy and Trust Calibration in AI-Assisted Decision Making. *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 295–305, 2020. doi:10.1145/3351095.3372852. URL <http://arxiv.org/abs/2001.02114>. arXiv: 2001.02114.
- B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. *arXiv:1612.01474*, 2017. URL <http://arxiv.org/abs/1612.01474>.
- D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. Concrete Problems in AI Safety. *arXiv:1606.06565*, 2016. URL <http://arxiv.org/abs/1606.06565>.
- D. J. C. MacKay. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4(3):448–472, May 1992. ISSN 0899-7667. doi:10.1162/neco.1992.4.3.448.
- Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, June 2015. URL <http://proceedings.mlr.press/v37/blundell11.5.html>. ISSN: 1938-7228.
- Z. C. Lipton. The Mythos of Model Interpretability. *arXiv:1606.03490*, 2017. URL <http://arxiv.org/abs/1606.03490>.
- M. Sundararajan, A. Taly, and Q. Yan. Axiomatic Attribution for Deep Networks. *arXiv:1703.01365*, 2017. URL <http://arxiv.org/abs/1703.01365>.
- D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114*, 2014. URL <http://arxiv.org/abs/1312.6114>.
- D. P. Kingma, T. Salimans, and M. Welling. Variational Dropout and the Local Reparameterization Trick. *arXiv:1506.02557*, 2015. URL <http://arxiv.org/abs/1506.02557>.
- Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, December 1989. ISSN 0899-7667. doi:10.1162/neco.1989.1.4.541. URL <https://doi.org/10.1162/neco.1989.1.4.541>.
- B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object Detectors Emerge in Deep Scene CNNs. *arXiv:1412.6856*, 2015. URL <http://arxiv.org/abs/1412.6856>.
- Kaggle. Dogs vs. Cats, January 2014. URL <https://kaggle.com/c/dogs-vs-cats>. Microsoft Research. <https://kaggle.com/c/dogs-vs-cats> (accessed 28/11/2020).

- N. Codella, V. Rotemberg, P. Tschandl, M. E. Celebi, S. Dusza, D. Gutman, B. Helba, A. Kalloo, K. Liopyris, M. Marchetti, H. Kittler, and A. Halpern. Skin Lesion Analysis Toward Melanoma Detection 2018: A Challenge Hosted by the International Skin Imaging Collaboration (ISIC). *arXiv:1902.03368*, 2019. URL <http://arxiv.org/abs/1902.03368>.
- Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. ISSN 1558-2256. doi:10.1109/5.726791.
- A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, 25:1097–1105, 2012. URL <https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html>.
- S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167*, 2015. URL <http://arxiv.org/abs/1502.03167>.
- D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980*, 2017. URL <http://arxiv.org/abs/1412.6980>.
- L. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv:1706.05587*, 2017. URL <http://arxiv.org/abs/1706.05587>.