Toward Faithful Explanatory Active Learning with Self-explainable Neural Nets*

Stefano Teso

KU Leuven, Leuven, Belgium
stefano.teso@cs.kuleuven.be

Abstract. From the user's perspective, interaction in active learning is very *opaque*: the user only sees a sequence of instances to be labeled, and has no idea what the model believes or how it behaves. Explanatory active learning (XAL) tackles this issue by making the model predict and explain its own queries using local explainers. By witnessing the model's (lack of) progress, the user can decide whether to trust it. Despite their promise, existing implementations of XAL rely on post-hoc explainers, which can produce unfaithful and fragile explanations, which misrepresent the beliefs of the predictor, confuse the user, and affect the quality of her supervision. As a remedy, we replace post-hoc explainers with self-explainable models, and show how these can be actively learned from both labels and corrected explanations. Our preliminary results showcase the dangers of post-hoc explanations and hint at the promise of our solution.

Keywords: Machine Learning · Active Learning · Explainability

1 Introduction

Explainable machine learning has so far mostly focused on black-box models learned offline [8]. It was recently observed that interactive protocols can be black-box too [24, 16]. For instance, in active learning (AL), the user receives a sequence of instances (e.g. images, documents) to be labeled, but can witness neither the behavior of the predictor nor its beliefs.

Explanatory active learning (XAL) tackles this issue by injecting explanations into the learning loop [24]: whenever asking the user to label an instance, the model also shows a prediction for that instance and an explanation for the prediction. The explanations are obtained with a local explainer [14, 8], which summarizes and visualizes the local behavior of the predictor in terms of interpretable feature relevance or other understandable artifacts. By witnessing the evolution of the beliefs and decisions of the model, the user can justifiably grant

* The author is grateful to Kristian Kersting for many useful discussions and to Giuseppe Marra for help with the implementation. This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. [694980] SYNTH: Synthesising Inductive Data Models).

or revoke trust to it. This is analogous to trust between individuals, which requires to develop appropriate expectations through interaction [23]. In XAL, the user is also free to correct the explanations by, e.g., indicating any irrelevant or sensitive features that the model is currently relying on. This extra supervision is *necessary* to correct models that are "right for the wrong reasons" [19].

Existing implementations of XAL make use of *post-hoc* local explainers for instance, CAIPI [24] uses LIME [18] which treat the predictor being learned as a black-box. These approaches extract explanations through an approximate model translation [5] step. The overall process can produce unfaithful, fragile explanations that misrepresent the model's beliefs and have high-variance [1, 2]. Unfaithful and unstable explanations may confuse the user and affect the quality of the user-provided corrections. More generally, such explanations are not trustworthy and conflict with the purpose of explanatory interaction.

As a remedy, we propose replacing post-hoc explainers with self-explainable neural networks (SENNs), a recently proposed class of models that automatically explain their own predictions. Intuitively, SENNs combine the transparency of linear models with the flexibility of neural nets [15]. The explanations produced by SENNs are exact, robust to small perturbations, and cheap to compute. In contrast with standard interpretable models (e.g. shallow decision trees), SENNs can tackle complex problems including representation learning via gradient descent. In order to integrate them into XAL, we show how to learn SENNs from labels and corrections directly by combining classification and ranking losses. Our preliminary empirical analysis shows that SENNs can substantially improve the quality of the explanations used in XAL and can be actively learned from labels and corrections.

Summarizing, we: 1) highlight the risks of unfaithful explanations in interactive learning; 2) propose a novel implementation of XAL based on self-explainable neural nets; 3) propose a joint loss to learn SENNs from labels and corrections directly; 4) report on preliminary experiments that showcase the behavior of post-hoc explainers and the promise of our solution.

2 Background

In the following, we will stick to binary classification and indicate instances as $x \in \mathcal{X}$ and labels as $y \in \mathcal{Y} = \{0, 1\}$. Our observations can be easily generalized to the multi-class case.

2.1 Post-hoc Local Explainers

Given a classifier $f : \mathcal{X} \to \mathcal{Y}$, for instance a neural network or a random forest, post-hoc local explainers explain individual predictions without looking at the exact inference steps performed by f [14]. Here we briefly detail LIME [18], which is central in current XAL implementations.

In order to explain a prediction $y^0 = f(x^0)$, LIME learns an *interpretable* local model g^0 that mimics f in the neighborhood of x^0, and then reads off an explanation from it. The local model is a sparse linear predictor or a shallow decision tree [18] built with (user-provided) interpretable features $\boldsymbol{\psi}(x)$. These may capture, e.g., individual words in document classification or objects in image tagging. The local model is learned from a synthetic dataset that describes counterfactual ("what if") information about switching on / off the interpretable features on $f(x^0)$.

More formally, the process amounts to:

1. Sampling s interpretable instances $\boldsymbol{\xi}^1, \ldots, \boldsymbol{\xi}^s$ by randomly perturbing the interpretable representation of x^0, namely $\boldsymbol{\xi}^0 = \boldsymbol{\psi}(x^0)$;
2. Labeling each instance $\boldsymbol{\xi}^i$ using the target model $y^i = f(x^i)$, where $x^i = \boldsymbol{\psi}^{-1}(\boldsymbol{\xi}^i)$ is the pre-image of $\boldsymbol{\xi}^i$;
3. Weighting each example $(\boldsymbol{\xi}^i, y^i)$ by its similarity to $\boldsymbol{\xi}^0$, i.e., $k(\boldsymbol{\xi}^i, \boldsymbol{\xi}^0)$; the kernel function k determines the size and shape of the neighborhood of $\boldsymbol{\xi}^0$;
4. Fitting a local model g^0 on the synthetic dataset via *cost-sensitive learning*, so that examples outside of the neighborhood do not have much of an impact; this is a form of model translation [5];
5. Extracting an explanation from g^0. For instance, if g^0 is linear (i.e., $g^0(x) = \sum_j w_j \psi_j(x)$) then the explanation describes the contributions of the interpretable features according to the weights w_j. In practice only the largest weights are used. If g^0 is a decision tree, then the feature contributions can be read off from the path connecting the root to the predicted leaf.

The advantage of this procedure is that it is completely model-agnostic, as it treats f as a black-box. The downside, however, is that it is not exact, and so g^0 may not approximate f well. We will discuss the consequences later on.

2.2 Explanatory Active Learning

Pool-based active learning (AL) is designed for settings where labels are scarce and expensive to obtain [22, 9]. Learning proceeds iteratively. Initially, the model has access to a small set of labeled examples $\mathcal{L} \subseteq \mathcal{X} \times \mathcal{Y}$ and a large pool of unlabeled instances $\mathcal{U} \subseteq \mathcal{X}$. In each iteration, the model asks an oracle (i.e. a human expert or a measurement device) to label any instance in $\mathcal{U}$ *for a price*. The newly labeled instance is then moved to $\mathcal{L}$ and the model is adapted accordingly. These steps are repeated until a labeling budget is exhausted or the model is deemed good enough. The key challenge in AL is to design a strategy for selecting informative and representative query instances, so to learn good predictors at a small labeling cost. A common choice is uncertainty sampling (US) [13], which picks instances where the model is most uncertain in terms of, e.g., margin or entropy.

In order to make the interaction more transparent and directable, explanatory active learning (XAL) injects explanations into the learning loop and enables the user to interact with them [24]. In XAL, when the model asks the user to label an instance x, it also presents a prediction for that instance $\hat{y} = \hat{f}(x)$ and an explanation $\hat{z}$ for the prediction. In the CAIPI implementation of XAL, the explanation is computed with LIME. In exchange, the user provides the true label of x and optionally corrects the explanation. The correction indicates, for instance, which interpretable features (e.g. pixels, words) are erroneously being used by the model. Since models cannot learn from corrections directly, CAIPI converts corrections into *counter-examples*, as follows: if the user indicated that some interpretable feature $\boldsymbol{\psi}_i(x)$ is wrongly being used by the model, then CAIPI creates c copies of x where feature $\boldsymbol{\psi}_i$ is randomized, and attaches the true label to them. Intuitively, the counter-examples teach the predictor to predict the correct label independently from the value associated to the irrelevant feature. It was shown that, for LIME, explanation corrections describe local orthogonality constraints [24], and that counter-examples approximate these constraints.

By combining interactions and explanations, XAL helps the user to build a mental model of the predictor being learned, which is necessary for justifiably according trust to it. In addition, by virtue of learning from explanation corrections, XAL makes it less likely that the model learns to predict the right labels for the wrong reasons. We will see, however, that these promises can be difficult to keep when post-hoc explainers are involved.

2.3 Self-explainable Neural Networks

Sparse linear models over interpretable features are canonically considered to be highly interpretable. These models have the form[1] $f(x) = \sigma(\boldsymbol{w}^\top \boldsymbol{\phi}(x))$, where σ is a sigmoid function, $\boldsymbol{w} \in \mathbb{R}^n$ is constant, and $\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^n$ is a fixed, interpretable feature map. The contribution of the jth feature to the output of the predictor is determined by the corresponding weight. Despite their interpretability, sparse linear models are limited to relatively simple learning tasks.

Self-explainable neural networks (SENNs) upgrade linear models by substantially increasing their capacity and flexibility while preserving their interpretability [15]. More specifically, SENNs have the same functional form, but allow the weights $\boldsymbol{w}$ to change across the space, i.e.:

$$f(x) = \sigma(\boldsymbol{w}(x)^\top \boldsymbol{\phi}(x)) \tag{1}$$

Here, both $\boldsymbol{w}(x)$ and $\boldsymbol{\phi}(x)$ are (arbitrarily deep) neural networks. The inner product can be replaced by any appropriate aggregation function [15]. In order to make $\boldsymbol{w}(x)$ act as an explanation, two additional restrictions are put in place:

1. The learned feature function $\boldsymbol{\phi}$ has to be interpretable. This can be achieved either by designing it by hand (as with linear predictors and LIME), by defining it in terms of (learned) prototypes, or by other means; see [15] for details.
2. As with linear models, the explanations should capture the local behavior of the model and not be affected by small displacements of the input instance. This is guaranteed by constraining $\boldsymbol{w}$ to vary slowly *with respect to* $\boldsymbol{\phi}$.

More formally, $\boldsymbol{w}$ is required to be locally difference bounded by $\boldsymbol{\phi}$, as per the following definition:

[1] Here and below, the bias term is left implicit.

Definition 1 ([15]). *A function* $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^a$ *is* locally difference bounded *by a function* $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^b$ *if for every* $\boldsymbol{x}^0 \in \mathbb{R}^n$ *there exists two constants* $\delta > 0$ *and* $L > 0$ *such that for every* $\boldsymbol{x} \in \mathbb{R}^n$*:*

$$\|\boldsymbol{x} - \boldsymbol{x}^0\| < \delta \implies \|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x}^0)\| \leq L\|\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{g}(\boldsymbol{x}^0)\|$$

In other words, for every point $\boldsymbol{x}^0$ there is a neighborhood within which[2] the change in $\boldsymbol{f}$ is bounded by the change in $\boldsymbol{g}$. Notice that the local Lipschitz "constant" L is allowed to vary with x^0.

This requirement is enforced during learning by penalizing the model for any deviations from linearity through the following regularization term:

$$\Omega(f) \stackrel{\text{def}}{=} \|\nabla_x f(x) - \boldsymbol{w}(x)^\top J_x\|$$

where J_x is the Jacobian matrix. The model f is learned by minimizing the empirical risk of some classification loss ℓ_Y plus the above regularizer, i.e.:

$$\min_f \ \hat{\mathbb{E}}_{(x,y)} \left[\ell_Y(f(x), y) + \alpha\Omega(f)\right] \tag{2}$$

Here $\hat{\mathbb{E}}$ indicates expectation over mini-batches and $\alpha \geq 0$ is a hyperparameter. As usual with neural networks, minimization is performed with gradient descent techniques. Of course, an additional term encouraging the output of $\boldsymbol{w}(x)$ to be sparse can be considered. We will see later on that this is automatically the case in our extension of SENNs to explanatory active learning.

It is worth pointing out that SENNs are quite different from attention models, which are yet another way of explaining (to some extent) neural networks. Indeed, the former explain exactly which features contribute to the prediction, as well as their polarity. The features themselves are arbitrary (interpretable) functions of the inputs. Attention models, in contrast, identify which *input features* (e.g. pixels) are relevant, but not how they contribute to the decision nor whether they are against/in favor.

3 Toward Faithful XAL

3.1 The Dangers of Post-hoc Explanations

Let us start by discussing the case of LIME, which is at the core of existing XAL implementations. In this case, the accuracy of the model translation process (described above) depends critically the choice of model class of g^0, interpretable features $\boldsymbol{\psi}$, kernel k, and number of samples s.

For instance:

1. If the kernel k is not chosen correctly (i.e. if it is too small, too broad, or has the wrong topology), then the synthetic dataset may fail to capture label changes around x^0, as shown in Figure 1.

[2] This can be limiting in classification scenarios, because the output should change abruptly when crossing the decision boundary. The formulation can be modified to account for this, but we keep it as is, for simplicity.

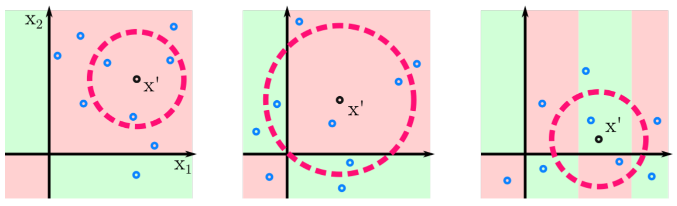

Fig. 1. Examples on which LIME produces unfaithful explanations. The decision surface of f is represented by the colored areas: light green means positive, light red negative. The pink circle represents the kernel k: points inside of it are assigned substantial weights while all others are not. Left: all synthetic examples with large weight have the same label. Middle: the synthetic examples fail to capture the non-additive interaction of the two features. Right: the kernel is too broad, and the synthetic dataset is highly complex and non-linear.

2. If the number of samples s is too small, the dataset may also not be representative.
3. ψ determines whether the pre-image of z^i is unique; if this is not the case, then the labeling step becomes unstable.

In all these cases, LIME produces high-variance explanations that substantially misrepresent the behavior of f. Increasing the number of samples may improve the situation, but also the runtime, and may not be enough to stabilize the explanations anyway.

Crucially, the same high-level argument applies to all post-hoc explainers, because all of them treat f as a black-box and thus by construction must include an inexact model translation step.

From the standpoint of XAL, unfaithfulness has two major consequences. First, high-variance explanations portray the model as behaving semi-randomly, regardless of whether it is good or not. The user may also perceive that her supervision has no effect, and feel lack of control. In rare cases, CAIPI may also accidentally persuade the user into trusting a bad model. Both cases are problematic in sensitive applications, like the ones that XAL is designed for. More generally, unfaithful explanations misrepresent the learned model, defeating the purpose of explanatory active learning. Second, unfaithful explanations compromise the usefulness of the corrections. If the user is confused by the explanations, her corrections will be not as informative. Further, the correction may specify not to use a feature that the model is not using anyway (or vice versa). Depending on the model being learned, correcting the same feature too many times may also lead to learning instabilities.

It is therefore desirable to fix the XAL pipeline to rely on faithful and trustable explanations. In the next section, we show how to do so.

Algorithm 1 Pseudocode of CALI: $\mathcal{L}$ is the set of labeled examples, $\mathcal{U}$ is the set of unlabeled instances, and T is the query budget.

```
 1: procedure CALI($\mathcal{L}$, $\mathcal{U}$, $T$)
 2:     $\leftarrow \varnothing$
 3:     $f \leftarrow \text{fit}(\mathcal{L},\ )$                                  ▷ Eq. 3
 4:     repeat
 5:         $x \leftarrow \text{select}(f, \mathcal{U})$,  $\hat{y} \leftarrow f(x)$,  $\hat{\boldsymbol{z}} \leftarrow \boldsymbol{w}(x)$
 6:         Present instance $x$, prediction $\hat{y}$, and explanation $\hat{\boldsymbol{z}}$ to the user
 7:         Receive label $y$ and explanation correction $\boldsymbol{z}$
 8:         Remove $x$ from $\mathcal{U}$, add $(x, y)$ to $\mathcal{L}$, add $\boldsymbol{z}$   $\hat{\boldsymbol{z}}$ to
 9:         $f \leftarrow \text{fit}(\mathcal{L},\ )$                              ▷ Eq. 3
10:     until budget $T$ is exhausted or $f$ is good enough
11:     return $f$
```

3.2 Explanatory Active Learning with SENNs

Our proposed algorithm dubbed Calimocho, or CALI for short follows closely the XAL learning loop; the pseudocode is listed in Algrithm 1. Notice that f here is a SENN. Explanation corrections are collected in a set $\mathcal{C}$, initially empty. In each iteration, the algorithm chooses an instance $x \in \mathcal{U}$ (using uncertainty sampling, as CAIPI, for simplicity), predicts its class $\hat{y}$, and generates an explanation $\hat{\boldsymbol{z}}$ for the prediction (line 5). The explanation $\hat{\boldsymbol{z}}$ is simply read off of $\boldsymbol{w}(x)$, i.e., $\hat{\boldsymbol{z}} = \boldsymbol{w}(x)$, without any projection or sampling step. All components are presented to the user (lines 6–7), who replies with the true label y of x and optionally with an improved explanation $\boldsymbol{z}$. Finally, the dataset is updated and the preference $\boldsymbol{z} \succeq \hat{\boldsymbol{z}}$ is added to the corrections $\mathcal{C}$.

The learning step requires to strategy to train SENNs using corrections too. One option is to follow CAIPI, an use counter-examples. However, these only approximately capture the constraint imposed by the correction, e.g., that the label should not depend on a particular interpretable feature. Depending on the application, there is also a (slim) chance that the counter-examples are actually wrong. Feature influence supervision solves this issue by asking the user [20], but this can be cognitively costly.

Instead, we opt for learning SENNs directly from corrections, as follows. Recall that, given an explanation $\hat{\boldsymbol{z}}$, a correction specifies, e.g., which features are erroneously being used by the model. Applying the correction to $\hat{\boldsymbol{z}}$ leads to a corrected explanation $\boldsymbol{z}$. It is therefore natural to impose that $\boldsymbol{w}(x)$ generates explanations that are closer to the corrected explanation rather than the predicted on. This can be accomplished by, e.g., minimizing the squared Euclidean distance $\|\boldsymbol{w}(x) - \boldsymbol{z}\|_2^2$ while maximizing $\|\boldsymbol{w}(x) - \hat{\boldsymbol{z}}\|_2^2$. It is easy to see that this is equivalent to imposing a ranking loss:

$$\|\boldsymbol{w}(x) - \boldsymbol{z}\|_2^2 - \|\boldsymbol{w}(x) - \hat{\boldsymbol{z}}\|_2^2 = 2\langle \boldsymbol{w}(x), \hat{\boldsymbol{z}} - \boldsymbol{z} \rangle + \left(\|\boldsymbol{z}\|_2^2 - \|\hat{\boldsymbol{z}}\|_2^2 \right)$$

Notice that the features that were *not* corrected by the user (i.e. $z_j - \hat{z}_j = 0$) do not contribute to the loss, as expected. Our solution also generalizes the input gradient constraints introduced in [19] from learning from full explanations to corrections, which contain information about only few interpretable features.

Letting the dataset include instances x, labels y, and preferences $\boldsymbol{z} \succeq \hat{\boldsymbol{z}}$ (represented with $\mathcal{C}$ in the pseudo-code), and denoting the ranking loss as $\ell_Z(\boldsymbol{w}(x), \boldsymbol{z} \succeq \hat{\boldsymbol{z}}) = \langle \boldsymbol{w}(x), \hat{\boldsymbol{z}} - \boldsymbol{z} \rangle$, CALI fits f by extending Eq. 2 with the above loss:

$$\min_f \ \hat{\mathbb{E}}_{(x,\bar{\boldsymbol{z}}\succeq\hat{\boldsymbol{z}},y)} \left[\lambda \ell_Y(f(x), y) + (1-\lambda)\ell_Z(\boldsymbol{w}(x), \boldsymbol{z} \succeq \hat{\boldsymbol{z}}) + \alpha \Omega(f)\right] \qquad (3)$$

Noisy corrections can be handled by tweaking the hyperparameter $\lambda \in [0, 1]$.

3.3 Discussion

A few remarks are in order. First, CALI trades off the model-agnosticism of CAIPI for exactness and efficiency. This is desirable, since faithfulness is necessary to justifiably establish trust, especially in sensitive applications. In addition, despite their apparent simplicity, SENNs are very flexible non-linear models [15], substantially more powerful than "standard" interpretable models (e.g. they natively support representation learning) and easy to learn with state-of-the-art gradient descent techniques.

It is natural to ask what kind of SENN architectures can be learned in a label-scarce framework like active learning. On the one hand, shallow SENNs can be learned efficiently from few labels alone. On the other, by supplementing label information, we expect explanation corrections do help to actively learn deep(er) models. Our preliminary results suggest that this might be the case.

4 Experiments

We address the following research questions:

Q1 Are the explanations output by LIME faithful?
Q2 Does CALI learn from corrections?
Q3 Does explanatory feedback help learn deeper models?

In order to do so, we implemented CALI[3] and ran a preliminary experiment with the synthetic color dataset used in [19, 24], where the goal is to classify small synthetic images *for the right reasons.* The images are 5×5 and have four possible colors. An image is positive if i) either the four corners have the same color, or ii) the top three middle pixels all have different colors. On all training images either both rules are satisfied or neither is. This means that labels alone are not enough to disambiguate between the two potential explanations. Both images and explanations are represented as $5 \times 5 \times 4$ one-hot arrays. Notice that the two conditions can be easily expressed as linear concepts in this space. As in [24], we consider true explanations $\boldsymbol{z}$ that highlight the k most relevant pixels. The corrections instead highlight (a subset of the) pixels that are wrongly

[3] Code at: `github.com/stefanoteso/calimocho`

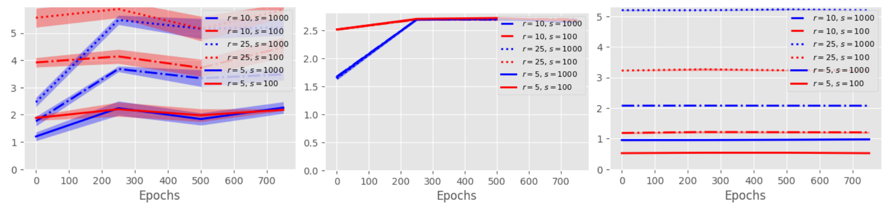

Fig. 2. Quality of the explanations produced by LIME for shallow SENNs trained passively for 1000 epochs (x axis); s is the number of samples used by LIME, while r is the number of repeats. Left: similarity between LIME explanations and the true explanations in terms of recall@k. Middle: diversity between LIME explanations across re-runs in terms of average pair-wise Euclidean distance between the r explanations. Right: runtime of LIME in seconds.

highlighted in the predicted explanations $\hat{\boldsymbol{z}}$. In all cases, we use a SENN where $\boldsymbol{\phi}(x)$ is simply the one-hot encoding of x and $\boldsymbol{w}(x)$ is a fully-connected feed-forward neural network with $L = 1, 3, 5$ hidden layers. All results are 5-fold cross-validated.

Q1: Are the explanations output by CAIPI *faithful?* We trained a SENN for 1000 epochs and every 250 iterations computed the recall k of the explanations produced by LIME on 10 random (but fixed) test examples. This was done by looking at how many of the k highest scoring features in the SENN explanation (which is exact) appear among the k highest scoring features found by LIME. In practice, CAIPI stabilizes high-variance explanation by running LIME r times and averaging the obtained explanations. To check whether this technique is effective, we repeated LIME $r = 5, 10, 25$ times and then measured the pair-wise Euclidean distance between the r explanations. The results, in Figure 2, show that the LIME explanations are never completely faithful, regardless of the number of samples s and repeats r, thus confirming our arguments about post-hoc explainers. In addition, while increasing s does have a clear beneficial effect, r is surprisingly not beneficial. Regardless, increasing either does have an effect on runtimes, as shown by the right plot. In comparison, SENN explanations are exact and robust by construction and also essentially free to compute, because $\boldsymbol{w}(x)$ must be evaluated *anyway* whenever doing inference. In practice, evaluating $\boldsymbol{w}$ amounts to forward propagation and takes a tiny fraction of a second (data not shown). This can be a substantial advantage in interactive applications to avoid making the user wait.

Q2: Does CALI *learn from corrections?* In this experiment, we select each rule in turn and use CALI to actively learn a SENN from explanation corrections. The ground-truth explanation highlight the 4 or 3 pixels that the decision actually depends on, and the corrections identify the $c = 1, \ldots, 4$ pixels whose predicted weight $w_i(x)$ is farthest away from the correct weight. Figure 3 shows the label loss ℓ_Y and the explanation loss ℓ_Z as more queries are made, up to 300. Turning

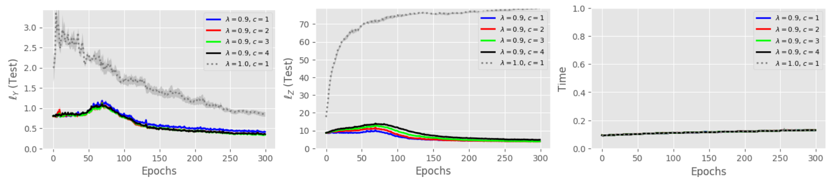

Fig. 3. Test set performance of CALI when learning from explanation corrections as more queries are asked (x axis). Left: label loss ℓ_Y. Middle: explanation loss ℓ_Z. Right: per-iteration runtime in seconds.

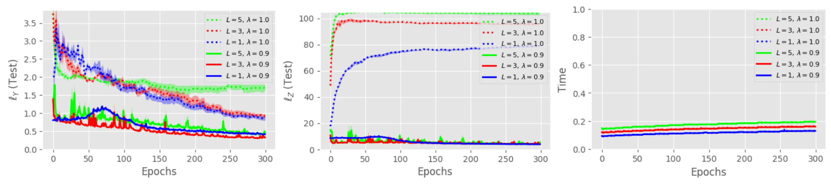

Fig. 4. Test set performance of CALI on progressively deeper SENNs. as more queries are asked (x axis). Left: label loss ℓ_Y. Middle: explanation loss ℓ_Z. Right: per-iteration runtime in seconds.

on learning from corrections brings a huge benefit, as expected. Indeed, when no corrections are provied (i.e. $\lambda = 1$), the SENN slowly learns the target concept, as shown by the leftmost plot, while corrections help the SENN to converge much faster. Even more importantly, unless corrections are enabled, the model fails to be "right for the right reasons", as the explanation loss *diverges* as more labels are obtained (middle plot). Finally, the rightmost plot shows that active learning CALI is very efficient, requiring less than 0.2 seconds per iteration on average. One surprising result is that increasing the number of corrections c does not monotonically increase performance. This needs to be validated further; however, the overall trend is clear and is consistent with the findings of [24]. The results for the two rules and different query strategies (random and margin-based uncertainty sampling) show exactly the same trend, and are not reported.

Q3: Does explanatory feedback help learn deeper models? Finally, we look at the effect of learning from corrections while increasing the number of hidden layers $L = 1, 3, 5$ of $\boldsymbol{w}(x)$. The results of increasing L are reported in Figure 4. Once again, the effect of corrections is very clear: they help the label loss to decay faster, and are necessary for the explanation loss not to diverge. Most importantly, these results hold regardless of the choice of L, with larger models behaving much worse on explanation loss when learned from labels only. This result is (preliminary but) very interesting especially in the light of the observation that CAIPI behaves best when learning sparser models [24]. In stark contrast, CALI seems to behave well even for deeper SENNs.

5 Related Work

Despite the surge of interest on explainable AI and machine learning, most research has focused on passive learning, and specifically on (1) designing interpretable predictors [25, 12] and (2) explaining black-box models such as neural networks [3, 8]. Instead, we consider explanations in an interactive learning setting.

We specifically study explanatory active learning (introduced in [24] in the wider context of explanatory *interactive* learning) which injects explanations into the active learning loop. Related approaches were proposed in [16], which uses LIME to communicate the exploration pattern of the active learner to the user, and in [20], where a model is learned explicitly from feature-level feedback. Like previous work on (interactive) feature selection and dual supervision [17, 7, 4, 21], these work either ignore the issue of trust or the advantages of learning from corrections rather than explanations. Please see [24] for a discussion. Conceptually, XAL is related to techniques like explanatory debugging [11], which however targets simple predictors only. Similar themes have been championed by [10].

The importance of explanation faithfulness has long been recognized [3]. More recently, the hidden dangers of local explainability tools have been the subject of a number of studies, e.g., [26] and [1], and the limitations of post-hoc explainers have been studied in [2]. The issue of unfaithful explanations in explanatory interactive learning, however, has not been considered before. These studies have lead to developing exact and / or robust explanatory techniques like input gradients [19] (aka instantaneous causal effects), average causal effects [6], and of course SENNs [15]. To the best of our knowledge, however, SENNs are the only method that 1) supports gradient-based optimization and interpretable representation learning, 2) generates exact explanations that are robust to perturbations, and 3) can be learned from labels and with this paper from corrections directly.

6 Conclusion and Outlook

The present paper highlights the dangers of post-hoc explainers for explanatory active learning (XAL). Post-hoc explainers are prone to generating unfaithful explanations that misrepresent the target predictor and prevent the user from appropriately allocating trust to it. In order to solve this issue, we extend existing XAL implementations by replacing post-hoc explainers with self-explainable neural networks (SENNs). SENNs automatically generate exact and robust explanations for their own predictions. In order to integrate them with XAL, we show how to learn SENNs from labels and explanation corrections by combining classification and ranking losses. Our preliminary experiments showcase the fragility of post-hoc explainers and the potential of SENNs in explanatory active learning.

Of course, our results need further validation, including a more direct comparisong with CAIPI, which we plan to carry on soon. They also hint at the promise of corrections for learning deeper networks even when labels are scarce. We plan to investigate this direction in future work.

References

1. Adebayo, J., Gilmer, J., Goodfellow, I., Kim, B.: Local explanation methods for deep neural networks lack sensitivity to parameter values. arXiv e-prints (Oct 2018), arXiv:1810.03307
2. Alvarez-Melis, D., Jaakkola, T.S.: On the robustness of interpretability methods. arXiv e-prints (Jun 2018), arXiv:1806.08049
3. Andrews, R., Diederich, J., Tickle, A.B.: Survey and critique of techniques for extracting rules from trained artificial neural networks. Knowledge-based systems **8**(6), 373–389 (1995)
4. Attenberg, J., et al.: A unified approach to active dual supervision for labeling features and examples. Machine Learning and Knowledge Discovery in Databases pp. 40–55 (2010)
5. Buciluǎ, C., et al.: Model compression. In: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 535–541. ACM (2006)
6. Chattopadhyay, A., Manupriya, P., Sarkar, A., Balasubramanian, V.N.: Neural network attributions: A causal perspective. In: International Conference on Machine Learning. pp. 981–990 (2019)
7. Druck, G., et al.: Active learning by labeling features. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1. pp. 81–90. Association for Computational Linguistics (2009)
8. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM computing surveys (CSUR) **51**(5), 93 (2018)
9. Hanneke, S.: Theory of disagreement-based active learning. Foundations and Trends◯ in Machine Learning **7**(2-3), 131–309 (2014)
10. Holzinger, A.: From machine learning to explainable ai. In: 2018 World Symposium on Digital Intelligence for Systems and Machines (DISA). pp. 55–66. IEEE (2018)
11. Kulesza, T., et al.: Principles of explanatory debugging to personalize interactive machine learning. In: Proceedings of the 20th International Conference on Intelligent User Interfaces. pp. 126–137. ACM (2015)
12. Lage, I., Ross, A., Gershman, S.J., Kim, B., Doshi-Velez, F.: Human-in-the-loop interpretability prior. In: Advances in Neural Information Processing Systems. pp. 10159–10168 (2018)
13. Lewis, D.D., Gale, W.A.: A sequential algorithm for training text classifiers. In: SIGIR'94. pp. 3–12. Springer (1994)
14. Lundberg, S., et al.: An unexpected unity among methods for interpreting model predictions. arXiv e-prints (Nov 2016), arXiv:1611.07478
15. Melis, D.A., Jaakkola, T.: Towards robust interpretability with self-explaining neural networks. In: Advances in Neural Information Processing Systems. pp. 7775–7784 (2018)
16. Phillips, R., Chang, K.H., Friedler, S.A.: Interpretable active learning. In: Conference on Fairness, Accountability, and Transparency (2018)
17. Raghavan, H., et al.: Active learning with feedback on features and instances. Journal of Machine Learning Research **7**(Aug), 1655–1686 (2006)

18. Ribeiro, M.T., Singh, S., Guestrin, C.: 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. pp. 1135–1144. ACM (2016)
19. Ross, A.S., Hughes, M.C., Doshi-Velez, F.: Right for the right reasons: training differentiable models by constraining their explanations. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence. pp. 2662–2670. AAAI Press (2017)
20. Sen, S., Mardziel, P., Datta, A., Fredrikson, M.: Supervising feature influence. arXiv e-prints (Mar 2018), arXiv:1803.10815
21. Settles, B.: Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In: Proceedings of the conference on empirical methods in natural language processing. pp. 1467–1478. Association for Computational Linguistics (2011)
22. Settles, B.: Active learning. Synthesis Lectures on Artificial Intelligence and Machine Learning **6**(1), 1–114 (2012)
23. Simpson, J.A.: Psychological foundations of trust. Current directions in psychological science **16**(5), 264–268 (2007)
24. Teso, S., Kersting, K.: Explanatory interactive machine learning. In: Proceedings of AIES'19 (2019)
25. Wu, M., Hughes, M.C., Parbhoo, S., Zazzi, M., Roth, V., Doshi-Velez, F.: Beyond sparsity: Tree regularization of deep models for interpretability. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
26. Yeh, C.K., Hsieh, C.Y., Sai Suggala, A., Inouye, D., Ravikumar, P.: On the (In)fidelity and Sensitivity for Explanations. arXiv e-prints (Jan 2019), arXiv:1901.09392