Debugging Models using Explanations

Stefano Teso

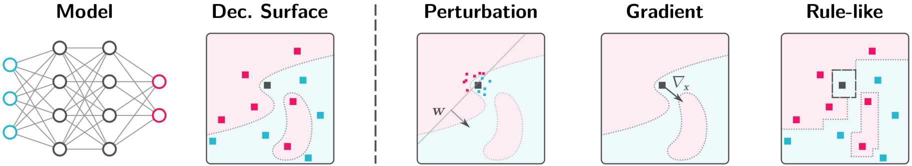
Advanced Topics in Machine Learning & Optimization — 2023-24

Outline

- Interacting via Input Attributions
  - Model-agnostic
  - End-to-end Differentiable
- Interacting via Example-based Explanations
  - Adapting the example's influence
  - Changing the example's label(s)
- Interacting via Concept-based Explanations
  - Concept-based Models
  - Neuro-symbolic Models
- Take-away

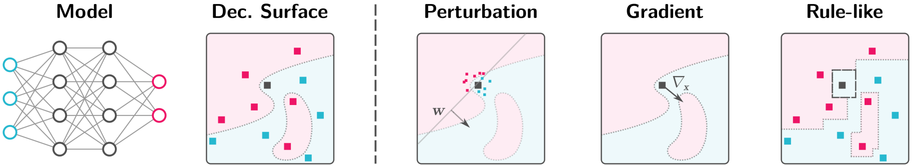
Input Attributions

- Given a predictor f and a target decision (x, y), **input attributions** identify what **input variables** are "most relevant" for the decision



Examples: LIME, SHAP, Integrated Gradients, GradCAM, ...

- Given a predictor f and a target decision (x, y), **input attributions** identify what **input variables** are "most relevant" for the decision

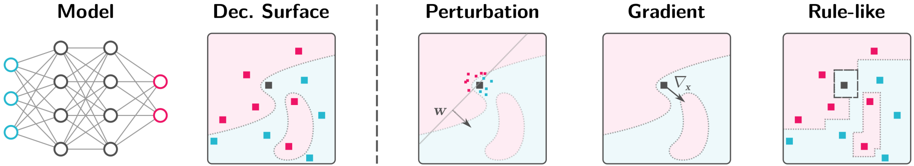


Examples: LIME, SHAP, Integrated Gradients, GradCAM, ...

- Typically:
  - **Ignore architectural details** (e.g., rely on decision surface only)

Input Attributions

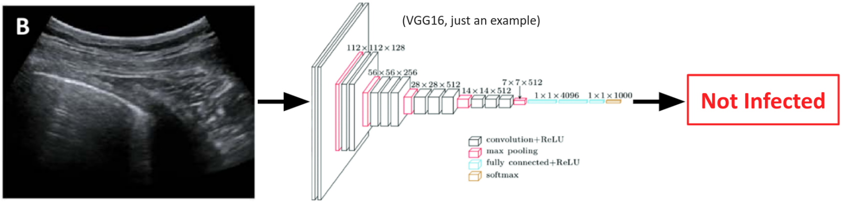
- Given a predictor f and a target decision $(\mathbf{x}, y)$, **input attributions** identify what **input variables** are "most relevant" for the decision



Examples: LIME, SHAP, Integrated Gradients, GradCAM, ...

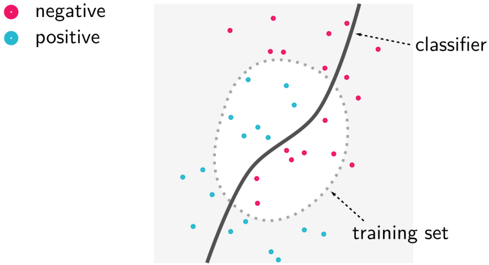
- Typically:
  - **Ignore architectural details** (e.g., rely on decision surface only)
  - **Local**: variables relevant for $(\mathbf{x}, y)$ may be irrelevant for a different, even similar, decision $(\mathbf{x}', y')$

You need to be checked for COVID-19. The doctor takes a scan of your lungs and uses a state-of-the-art deep neural network to automatically compute a diagnosis. The model thinks that you are not infected.



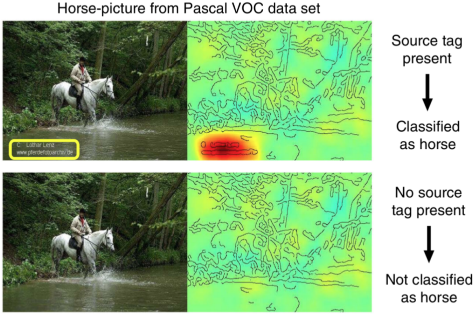
Question: Would you trust the model's prediction?

Clever Hans behavior & Explanations



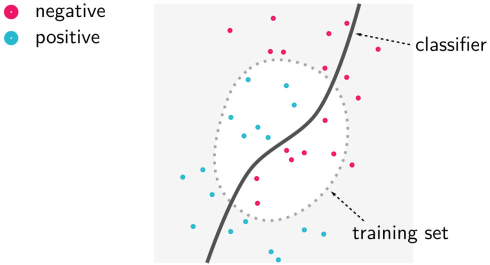
- Training data is not representative of full distribution
- **Clever Hans behavior**: model picks up shortcuts that optimize performance on training set
- Compromises test & OOD performance

Can be spotted using **model's explanations**



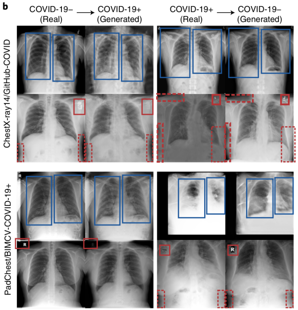
Source: (Lapuschkin et al., 2019)

Clever Hans behavior & Explanations



- Training data is not representative of full distribution
- **Clever Hans behavior**: model picks up shortcuts that optimize performance on training set
- Compromises test & OOD performance

Can be spotted using **model's explanations**



Source: (DeGrave et al., 2021)

Explanations Don't Fix Bugs

- Explanations are great for **identifying** bugs.
- They also hint at what should be done to **fix** those bugs.
- However, taken in isolation, they are insufficient to **correct** the model.

Explanations Don't Fix Bugs

- Explanations are great for **identifying** bugs.
- They also hint at what should be done to **fix** those bugs.
- However, taken in isolation, they are insufficient to **correct** the model.

Idea: show explanations to *sufficiently expert* users (e.g., domain experts, machine learning/data science practitioners) and integrate their feedback into the model.

Explanations Don't Fix Bugs

- Explanations are great for **identifying** bugs.
- They also hint at what should be done to **fix** those bugs.
- However, taken in isolation, they are insufficient to **correct** the model.

Idea: show explanations to *sufficiently expert* users (e.g., domain experts, machine learning/data science practitioners) and integrate their feedback into the model.

This hints at integrating **explanations** into **interactive machine learning**. How?

Dimensions

- **Goal**: learning vs debugging vs editing.
- **Explanations used**: local (input-, example-, concept- based) vs global.
- **Feedback received**: very many, often corrections to the explanations or to the training data.
- **Incorporation**: data augmentation vs additional loss.

A fuller description can be found in (Teso et al., 2022).

Explanatory Debugging (Kulesza et al., 2015)

- Designed for quick customization in, e.g., spam detection.

- Assumes **naive Bayes classifier**:

$$p(Y \mid \mathrm{x}) = \frac{p(y) \cdot p(\mathrm{x} \mid y)}{p(\mathrm{x})}$$

where:

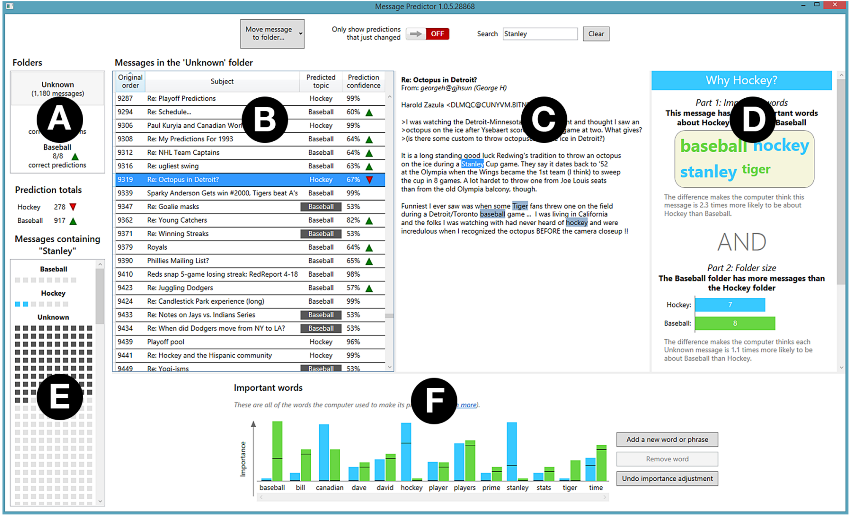
- $p(y)$ is proportion of documents in class y
- $p(\mathrm{x} \mid y)$ assumes cond. indep. between words $x_1, \ldots, x_\ell$

$$p(\mathrm{x} \mid y) = \prod_i p(x_i \mid y)^{n_i}$$

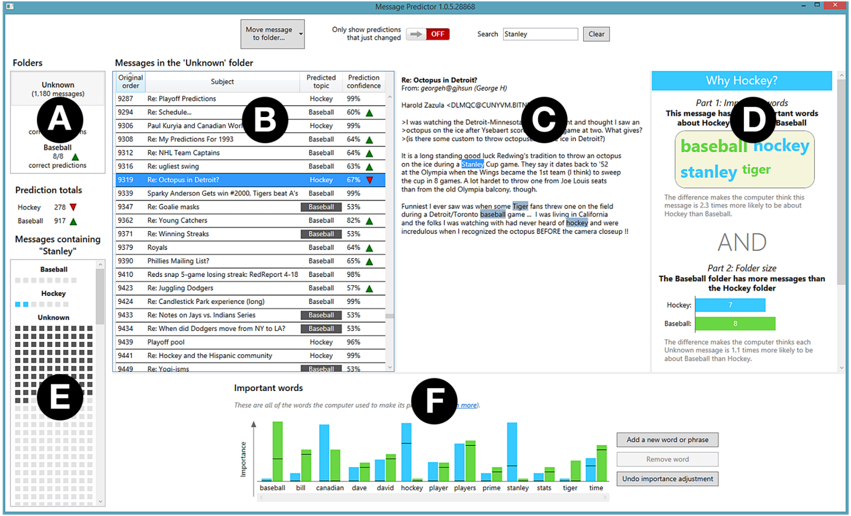
Here, n_i is the $\#$ of copies of word x_i in documents of class y, plus extra regularization (e.g., Laplace correction).

Debugging is carried out on these two terms independently.

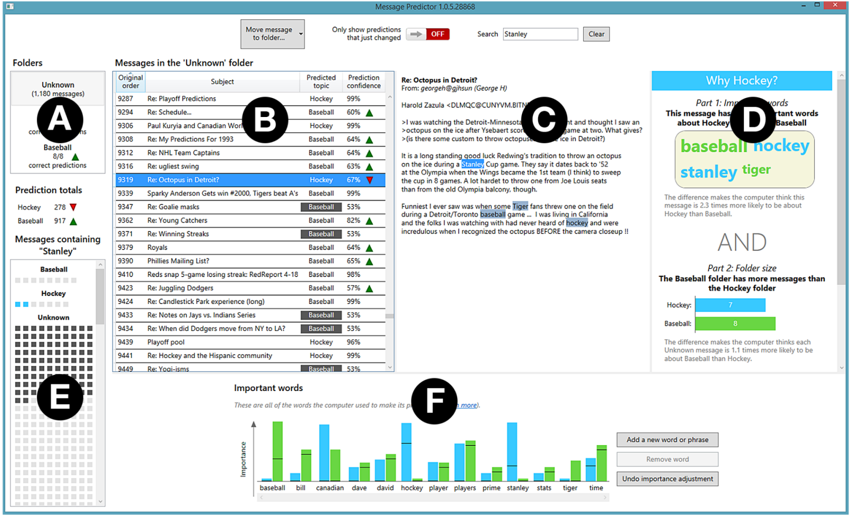
Step 1: pick an example that looks fishy or are costly.



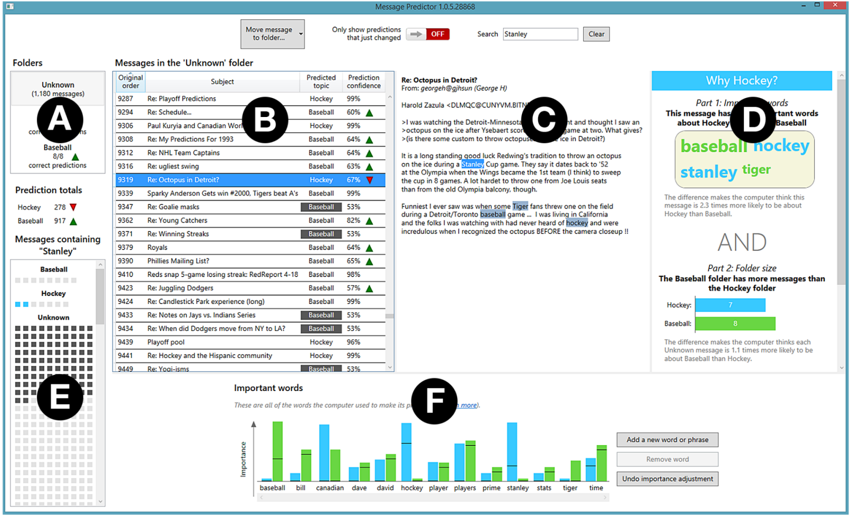
Step 2: look at the class probabilities.



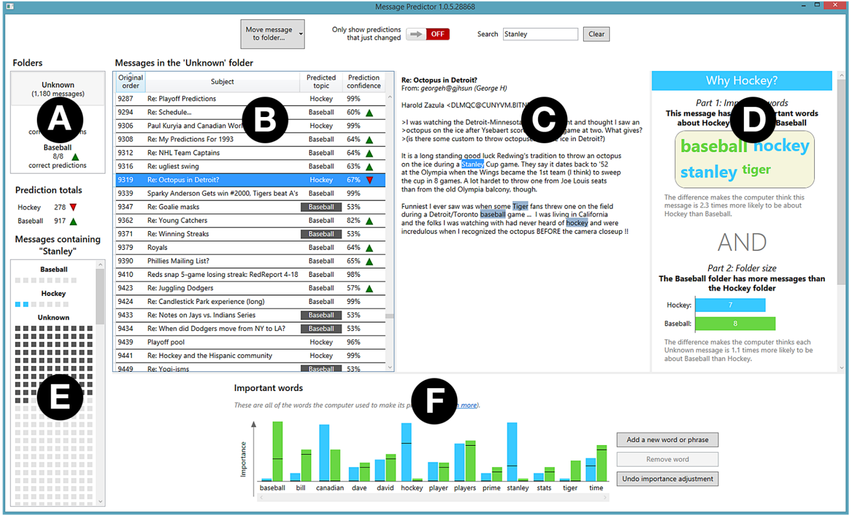
Step 3: look at the words that most impact the prediction – akin to a local explanation.



Step 4: check influence of class prior and individual words – in context.



Step 5: always have an overiview of overall word impact – akin to a global explanation.



Explanatory Debugging

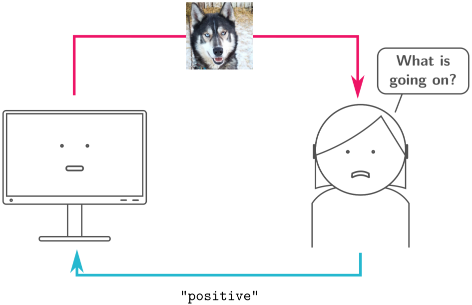
PROs:

- Designed to be user-compatible from the ground up.
- Allows users to select instances.
- Makes use of ad-hoc local and global explanations for communicating what the model learned.
- User feedback is integrated directly into parameters.
- User immediately sees impact of their feedback.

CONs:

- Limited to naive Bayes classifiers.
- User is responsible for choosing predictions to be debugged.
- Limited to simple tasks.

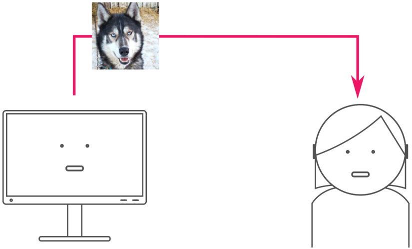
Active Learning is Opaque



- The user cannot:
  - observe the **model's beliefs**
  - **affect** them directly
  - see what her **feedback does**
- How can she:
  - prevent the model from acquiring shortcuts?
  - fix shortcuts acquired by the model?
  - justifiably build/reject trust? [a]

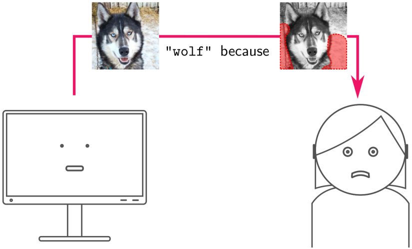
[a] But see (Honeycutt et al., 2020).

Explanatory Interactive Learning (Teso and Kersting, 2019; Schramowski et al., 2020)



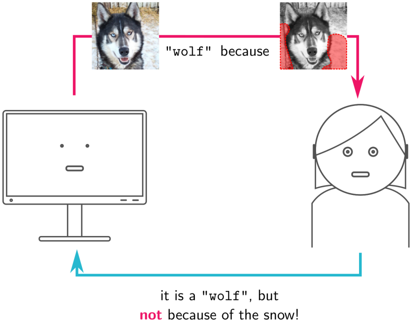
- Machine explains own predictions
  - helps with **understandability**
  - helps assessing **competence**

Explanatory Interactive Learning (Teso and Kersting, 2019; Schramowski et al., 2020)



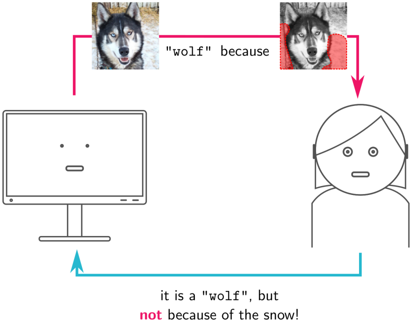
- Machine explains own predictions
  - helps with **understandability**
  - helps assessing **competence**
- User can supply corrections
  - substantially improves **directability**

Explanatory Interactive Learning (Teso and Kersting, 2019; Schramowski et al., 2020)



- Machine explains own predictions
  - helps with **understandability**
  - helps assessing **competence**
- User can supply corrections
  - substantially improves **directability**
- Compared to previous approaches:
  - Focus on **modern**, **complex** ML models
  - Builds on **general** (rather than *ad hoc*) explanatory AI tools

Explanatory Interactive Learning (Teso and Kersting, 2019; Schramowski et al., 2020)

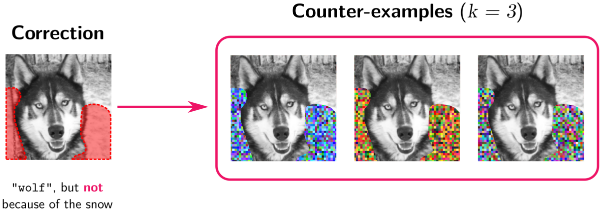


- Machine explains own predictions
  - helps with **understandability**
  - helps assessing **competence**
- User can supply corrections
  - substantially improves **directability**
- Compared to previous approaches:
  - Focus on **modern**, **complex** ML models
  - Builds on **general** (rather than *ad hoc*) explanatory AI tools

How to **align** *model to user's corrections in a general enough manner?*

- **Corrections** identify false relevant pixels: CAIPI converts them to **regular examples** & retrains

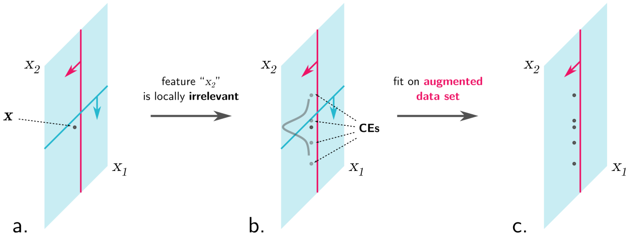
Example: if "`wolf`" predicted right for the wrong reasons:



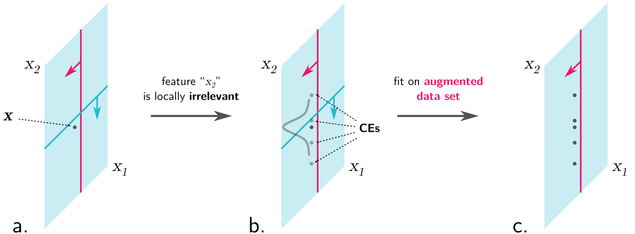
Such CEs teach the model to predict the right label **without using the random data**

CAIPI: Counter-examples

Idea: sample k perturbed copies of $(\mathbf{x}, y)$ by **randomizing** irrelevant pixels



Idea: sample k perturbed copies of $(\mathbf{x}, y)$ by **randomizing** irrelevant pixels



- CEs capture **invariances**: they show that the label does *not* change with x_2.

- In a sense, they are the *opposite* of counterfactuals/adversarial examples: counter-examples tell the model that the label should **not** change, counterfactuals seek changes that **do** impact the label.

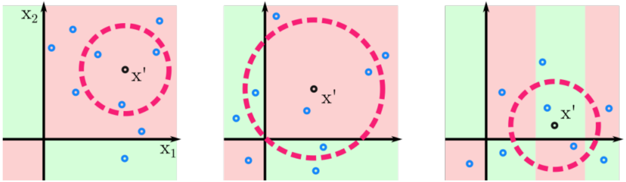
- Intuitively, **prediction (or score) at $\mathbf{x}$ should not depend on x_2**, i.e.,

$$\langle (w_1, w_2), \mathbf{x} \rangle = \langle (w_1, 0), \mathbf{x} \rangle \qquad \longrightarrow \qquad \forall \text{ local } x_2' \, . \, \langle (w_1, w_2), \mathbf{x}) \rangle = \langle (w_1, w_2), (x_1, x_2') \rangle$$

where $\mathbf{w}$ is a hyperplane that (locally) approximates f. The augmented examples $((x_1, x_2'), y)$ approximate an **orthogonality constraint**

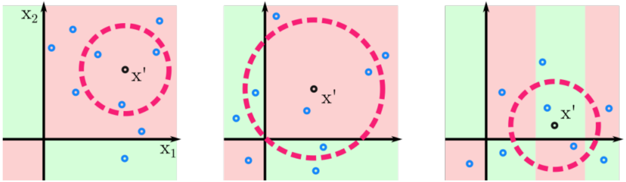
- Also works in feature space $\phi(\mathbf{x})$: randomize ϕ_i in place of x_i

CAIPI and issues with LIME (Teso, 2019)



- A bug appearing in LIME explanations may **not** reflect the model's actual reasoning.
- Asking the user to correct such "fake" issues wastes the user's effort and does not improve the model.

CAIPI and issues with LIME (Teso, 2019)



- A bug appearing in LIME explanations may **not** reflect the model's actual reasoning.
- Asking the user to correct such "fake" issues wastes the user's effort and does not improve the model.
- Can we do away with LIME?

Self-explainable Neural Networks (Alvarez-Melis and Jaakkola, 2018)

$$p_\theta(1 \mid \mathbf{x}) = \sigma\Big(\underbrace{\sum_i w_i(\mathbf{x})\phi_i(\mathbf{x})}_{\text{“score” of } \mathbf{x}}\Big)$$

- $\phi : \mathbb{R}^d \to \mathbb{R}^k$ embeds inputs into feature space
- $\mathbf{w} : \mathbb{R}^d \to \mathbb{R}^k$ computes a weight vector for each input

Self-explainable Neural Networks (Alvarez-Melis and Jaakkola, 2018)

$$p_\theta(1 \mid \mathbf{x}) = \sigma\Big(\underbrace{\sum_i w_i(\mathbf{x})\phi_i(\mathbf{x})}_{\text{“score” of } \mathbf{x}} \Big)$$

- $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^k$ embeds inputs into feature space
- $\mathbf{w} : \mathbb{R}^d \rightarrow \mathbb{R}^k$ computes a weight vector for each input
- $\mathbf{w}(\mathbf{x})$ is regularized to **vary slowly** w.r.t. $\mathbf{x}$

Self-explainable Neural Networks (Alvarez-Melis and Jaakkola, 2018)

$$p_\theta(1 \mid \mathbf{x}) = \sigma\Big(\underbrace{\sum_i w_i(\mathbf{x})\phi_i(\mathbf{x})}_{\text{"score" of } \mathbf{x}}\Big)$$

- $\phi : \mathbb{R}^d \to \mathbb{R}^k$ embeds inputs into feature space
- $\mathbf{w} : \mathbb{R}^d \to \mathbb{R}^k$ computes a weight vector for each input
- $\mathbf{w}(\mathbf{x})$ is regularized to **vary slowly** w.r.t. $\mathbf{x}$

Defines **a different linear model for every $\mathbf{x} \in \mathbb{R}^d$**

Self-explainable Neural Networks (Alvarez-Melis and Jaakkola, 2018)

$$p_\theta(1 \mid \mathbf{x}) = \sigma\Big(\underbrace{\sum_i w_i(\mathbf{x})\phi_i(\mathbf{x})}_{\text{“score” of } \mathbf{x}}\Big)$$

- $\phi : \mathbb{R}^d \to \mathbb{R}^k$ embeds inputs into feature space
- $\mathbf{w} : \mathbb{R}^d \to \mathbb{R}^k$ computes a weight vector for each input
- $\mathbf{w}(\mathbf{x})$ is regularized to **vary slowly** w.r.t. $\mathbf{x}$

■ Defines **a different linear model for every $\mathbf{x} \in \mathbb{R}^d$**

■ Linear models associated to nearby inputs $\mathbf{x}$ **encouraged to be similar**, i.e., in the neighborhood of any $\mathbf{x}_0$ there exists a constant vector $\mathbf{w}_0$ that depends only on $\mathbf{x}_0$ and a “large enough” $\alpha > 0$ such that:

$$\sum_i w_i(\mathbf{x}')\phi_i(\mathbf{x}') \approx \sum_i w_{0i}\phi_i(\mathbf{x}_0) \qquad \text{for all } \mathbf{x}' \text{ that are closer than } \alpha \text{ to } \mathbf{x}_0$$

Self-explainable Neural Networks (Alvarez-Melis and Jaakkola, 2018)

$$p_\theta(1 \mid \mathbf{x}) = \sigma\Big(\underbrace{\sum_i w_i(\mathbf{x})\phi_i(\mathbf{x})}_{\text{“score” of } \mathbf{x}} \Big)$$

- $\phi : \mathbb{R}^d \to \mathbb{R}^k$ embeds inputs into feature space
- $\mathbf{w} : \mathbb{R}^d \to \mathbb{R}^k$ computes a weight vector for each input
- $\mathbf{w}(\mathbf{x})$ is regularized to **vary slowly** w.r.t. $\mathbf{x}$

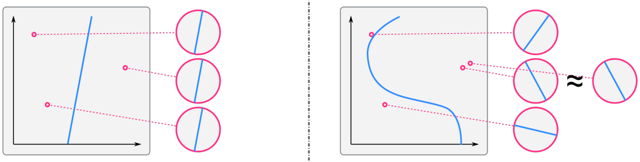
■ Defines **a different linear model for every $\mathbf{x} \in \mathbb{R}^d$**

■ Linear models associated to nearby inputs $\mathbf{x}$ **encouraged to be similar**, i.e., in the neighborhood of any $\mathbf{x}_0$ there exists a constant vector $\mathbf{w}_0$ that depends only on $\mathbf{x}_0$ and a “large enough” $\alpha > 0$ such that:

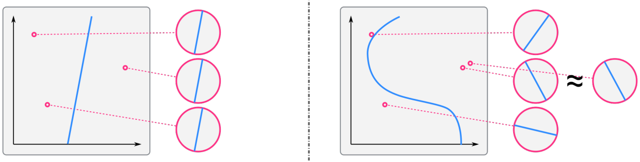
$$\sum_i w_i(\mathbf{x}')\phi_i(\mathbf{x}') \approx \sum_i w_{0i}\phi_i(\mathbf{x}_0) \qquad \text{for all } \mathbf{x}' \text{ that are closer than } \alpha \text{ to } \mathbf{x}_0$$

■ If $\mathbf{w}(\mathbf{x}) \equiv \mathbf{w}$ is **constant** w.r.t. $\mathbf{x}$, we obtain a linear model again

- **Left**: a linear model. Notice that the weights $\mathbf{w}$ are constant everywhere.
- **Right**: a SENN. Notice that **locally** the weights $\mathbf{w}(\mathbf{x})$ are almost identical!



- **Left**: a linear model. Notice that the weights $\mathbf{w}$ are constant everywhere.
- **Right**: a SENN. Notice that **locally** the weights $\mathbf{w}(\mathbf{x})$ are almost identical!



- SENNs are stable locally (**interpretability**) but flexible globally (**large capacity**)

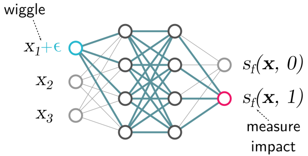
XIL with Right for the Right Reasons (Schramowski et al., 2020)

Input Gradient (Baehrens et al., 2010)

The **Input Gradient** (IG) of f at $\mathbf{x}$ is:

$$\mathrm{IG}_i(f, (\mathbf{x}, y)) := \frac{\partial}{\partial x_i} \mathrm{score}_f(\mathbf{x}, y)$$

I.e., **relevance of i-th input is proportional to how much score of class y changes when perturbing x_i**. It can be viewed from the lens of independent causal effects (ICE) (**?**).



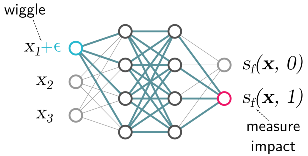
XIL with Right for the Right Reasons (Schramowski et al., 2020)

Input Gradient (Baehrens et al., 2010)

The **Input Gradient** (IG) of f at $\mathbf{x}$ is:

$$\mathrm{IG}_i(f, (\mathbf{x}, y)) := \frac{\partial}{\partial x_i} \mathrm{score}_f(\mathbf{x}, y)$$

I.e., **relevance of i-th input is proportional to how much score of class y changes when perturbing x_i**. It can be viewed from the lens of independent causal effects (ICE) (**?**).



- Instead of adding CEs, **directly penalize f for relying on irrelevant attributes** (Ross et al., 2017):

$$\ell_{\mathrm{rrr}}(f, (x, y)) := \sum_{i \text{ irrelevant for } (\mathbf{x}, y)} (\mathrm{IG}_i(f, (\mathbf{x}, y)))^2$$

IG of irrelevant inputs should be **near-zero**; IG of relevant inputs is not penalized.

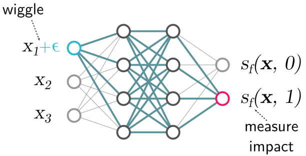
XIL with Right for the Right Reasons (Schramowski et al., 2020)

Input Gradient (Baehrens et al., 2010)

The **Input Gradient** (IG) of f at $\mathbf{x}$ is:

$$\mathrm{IG}_i(f, (\mathbf{x}, y)) := \frac{\partial}{\partial x_i} \mathsf{score}_f(\mathbf{x}, y)$$

I.e., **relevance of i-th input is proportional to how much score of class y changes when perturbing x_i**. It can be viewed from the lens of independent causal effects (ICE) (**?**).



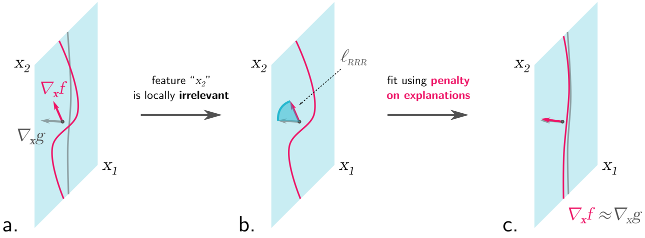
- Instead of adding CEs, **directly penalize f for relying on irrelevant attributes** (Ross et al., 2017):

$$\ell_{\mathsf{rrr}}(f, (x, y)) := \sum_{i \text{ irrelevant for } (\mathbf{x}, y)} (\mathrm{IG}_i(f, (\mathbf{x}, y)))^2$$

IG of irrelevant inputs should be **near-zero**; IG of relevant inputs is not penalized.

- Benefits:
  - Enables **end-to-end** training: backprop through IGs is straightforward using `Tensorflow`, `Pytorch`, *etc.*
  - No more **data augmentation**: no extra space & no costly sampling required
  - No need to **fine-tune** and/or **over-sample** LIME, no variability in explanations

XIL with Right for the Right Reasons (Schramowski et al., 2020)



- Ideally, **align** IG of (score of) learned model f to ground-truth g. In reality, the ground-truth gradient $\nabla_{\mathbf{x}} g$ is unknown; supervision only denotes (some of) its **null entries**.

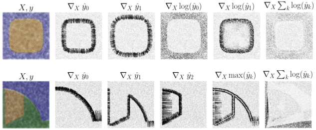
- Like CAIPI, introduces a **orthogonality constraint**, except it is explicit & no sampling required

Alternatives to RRR Loss

- There exist a variety of "RRR-like" losses:

Name	Formalization
RRR (Ross et al., 2017)	$\left(\sum_y \frac{\partial}{\partial x_i} s_f(\mathbf{x}, y)\right)^2$
GradMask (Simpson et al., 2019)	$\left\lvert \frac{\partial}{\partial x_i}(s_f(\mathbf{x}, 0) - s_f(\mathbf{x}, 1))\right\rvert$
CDEP (Rieger et al., 2020)	$\left\lVert \mathrm{attr}(f, (\mathbf{x}, y)) - \mathrm{expl}_i \right\rVert_2$
RBR (Shao et al., 2021)	$\left(\sum_y \mathrm{IF}(\frac{\partial}{\partial x_i} s_f(\mathbf{x}, y))\right)^2$
...	...

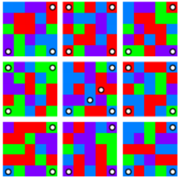
- Many variants of RRR itself!

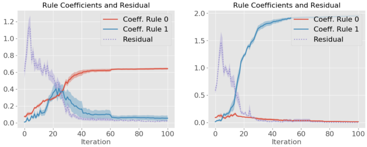


Some alternatives lead to better and more stable results

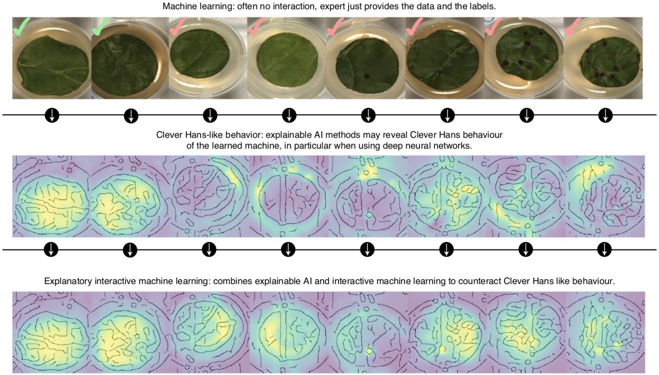
Source: (Ross et al., 2017).

Application: Hyperspectral Images





Source: (Teso and Kersting, 2019).



Source: (Schramowski et al., 2020).

CAIPI

- $\oplus$ Model-agnostic = LIME + Data Augmentation
- $\ominus$ Space hungry for larger k
- $\ominus$ Substantial overhead (sampling)
- $\ominus$ Can be unfaithful (sampling)

XIL + RRR

- $\ominus$ Requires differentiable model & attr. method
- $\oplus$ Space efficient
- $\oplus$ Limited overhead
- $\oplus$ As faithful as the attr. method

CAIPI

- ⊕ Model-agnostic = LIME + Data Augmentation
- ⊖ Space hungry for larger k
- ⊖ Substantial overhead (sampling)
- ⊖ Can be unfaithful (sampling)

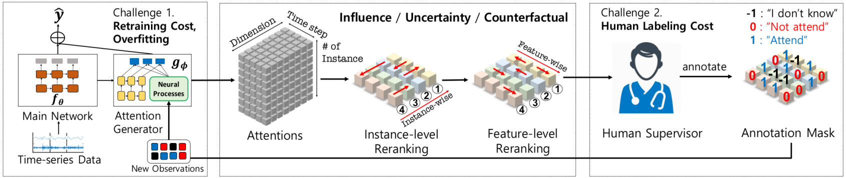
XIL + RRR

- ⊖ Requires differentiable model & attr. method
- ⊕ Space efficient
- ⊕ Limited overhead
- ⊕ As faithful as the attr. method

Remark

Both strategies can be adapted to using **partial corrections**, i.e., where the annotator only corrects *part* of the inputs (pixels). This makes interaction much more affordable.

■ Human-in-the-loop learning for models with **attention**[1]



(A) Neural Attention Process (B) Cost-Effective Re-ranking (C) Human Annotation

Source: (Heo et al., 2020).

■ Lowers **retraining cost**:
- Only updates attention module rather than full model
- Considerably speeds-up integrating human feedback
- Can help with overfitting

■ Prioritizes annotating incorrect/influential elements
- Reranks both **instances** & **features** (e.g., words)
- Can leverage uncertainty, influence; CF score for feats
- E.g., prevents annotating already-correct examples

[1] Notice that attention is not necessarily explainable, cf. (Bastings and Filippova, 2020).

Are Local Explanations Enough?

Machine-initiated Interaction → Narrative Bias

- The queries, predictions and explanations output by the model narrate the evolution of the model over time
- By witnessing that the model's narrative, the human "teacher" builds trust into the "student" model

[2] Loss on explanations is affected too.

- The queries, predictions and explanations output by the model narrate the evolution of the model over time
- By witnessing that the model's narrative, the human "teacher" builds trust into the "student" model
- What if the machine (unintentionally) lies about its own performance?

[2] Loss on explanations is affected too.

- The queries, predictions and explanations output by the model narrate the evolution of the model over time
- By witnessing that the model's narrative, the human "teacher" builds trust into the "student" model
- What if the machine (unintentionally) lies about its own performance? Nothing prevents the machine from repeatedly choosing instances on which it does well → **narrative bias**

$$\underbrace{\frac{1}{T}\sum_{t=1}^{T} L(f_t, x_t)}_{\text{loss at queries}} - \underbrace{\mathbb{E}_{\mathbf{x}\sim p(\mathbf{X})}[L(f_T, \mathbf{x})]}_{\text{actual loss at } T}$$

i.e., difference between perceived and actual performance (loss).[2]

[2] Loss on explanations is affected too.

Machine-initiated Interaction → Narrative Bias

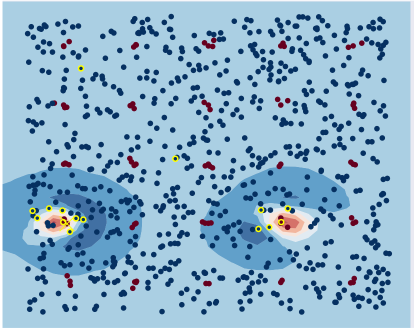
- The queries, predictions and explanations output by the model narrate the evolution of the model over time
- By witnessing that the model's narrative, the human "teacher" builds trust into the "student" model
- What if the machine (unintentionally) lies about its own performance? Nothing prevents the machine from repeatedly choosing instances on which it does well → **narrative bias**

$$\underbrace{\frac{1}{T}\sum_{t=1}^{T} L(f_t, x_t)}_{\text{loss at queries}} - \underbrace{\mathbb{E}_{\mathbf{x}\sim p(\mathbf{X})}[L(f_T, \mathbf{x})]}_{\text{actual loss at } T}$$

i.e., difference between perceived and actual performance (loss).[2]

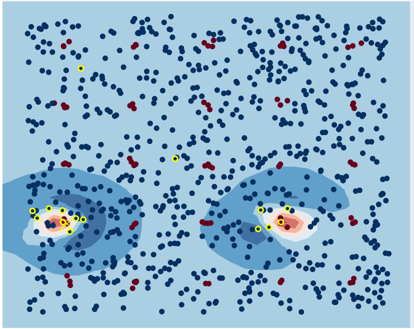
- This can occur even if the machine is not malicious!

[2] Loss on explanations is affected too.

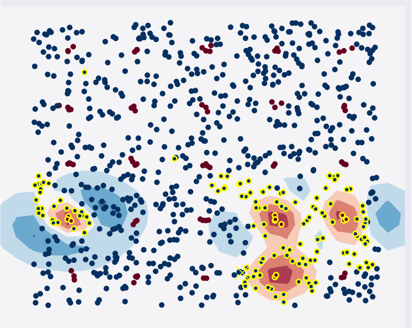


- This is active learning with uncertainty sampling running on top of an RBF SVM on a simple **red** vs. **blue** classification task. The red points are arranged on a 5×5 grid.

Narrative Bias



- The machine is affected by **unknown unknowns**: it is uncertain about regions close to the **red** clusters it has found, but completely certain that everything else is **blue**.



■ After 140 iterations, the machine is **still unaware** of most red clusters. Its predictions and explanations in the known region, however, will be very high quality → **narrative bias**.

- Summarizing, the “narrative” overestimates quality of the model because the machine doesn’t know where the hard instances are

- Summarizing, the “narrative” overestimates quality of the model because the machine doesn’t know where the hard instances are

- **Idea**: if the machine cannot know – ask the **user** instead!

- Summarizing, the “narrative” overestimates quality of the model because the machine doesn’t know where the hard instances are

- **Idea**: if the machine cannot know – ask the **user** instead!

- If the user knows, give him/her tools for choosing challenging instances. (This is what professors do when testing students!)

- Summarizing, the “narrative” overestimates quality of the model because the machine doesn’t know where the hard instances are

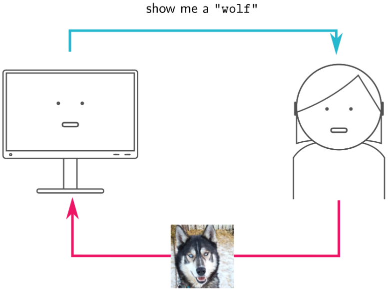
- **Idea**: if the machine cannot know – ask the **user** instead!

- If the user knows, give him/her tools for choosing challenging instances. (This is what professors do when testing students!)

- However, normally the human teacher is **blind**:
  - impossible to establish justifiable trust
  - she may provide examples that teach nothing new to the machine

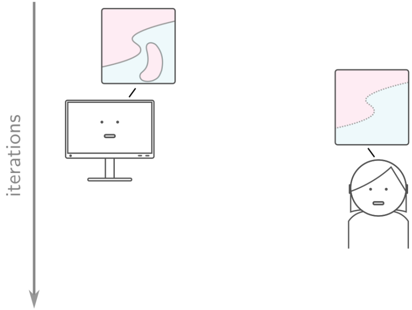
How can we expect the human to provide useful examples?

Explanatory Guided Learning (Popordanoska et al., 2020)



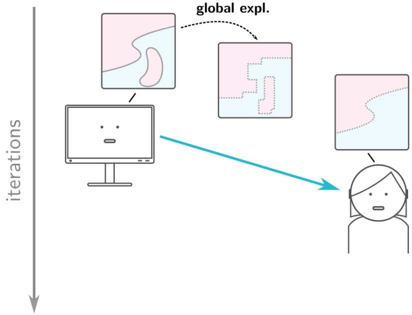
- **Idea**: build on top of **guided learning**: the machine provides a target label y, asks the user for an instance $\mathbf{x}$ of that class. Easy to do with search engines.

Explanatory Guided Learning (Popordanoska et al., 2020)



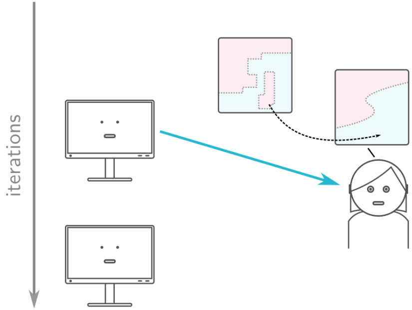
- Assume the machine and user "decision surfaces" differ as shown in the picture. In regular guided learning, interaction is **opaque**: the user has no clue it should provide a blue example!

Explanatory Guided Learning (Popordanoska et al., 2020)



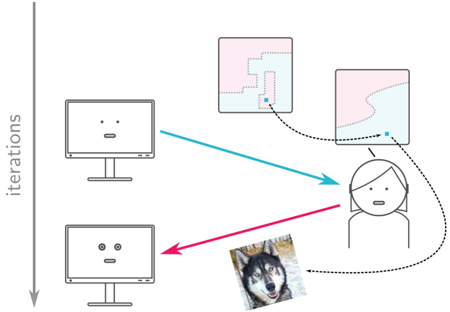
- Instead, we use **global explanations**, e.g., rules that summarize the overall decision surface of the model. They provide a simplified, human-friendly view of the model's beliefs.

Explanatory Guided Learning (Popordanoska et al., 2020)



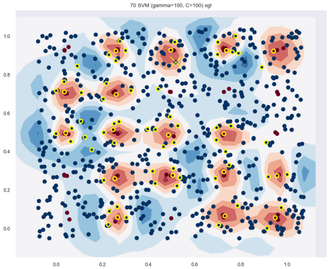
- Based on rules, the user has the chance of identifying bugs and issues in the model – in this case, a **blue** region wrongly classified as **red**.

Explanatory Guided Learning (Popordanoska et al., 2020)



- At this point, the user is free to provide a **counterexample** to the model, which is can then added to the training set and used to fine-tune the model as usual.

Explanatory Guided Learning (Popordanoska et al., 2020)



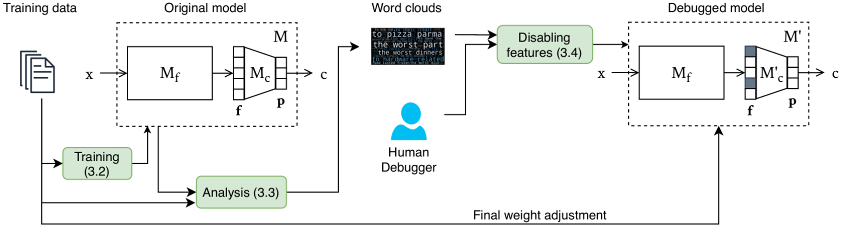
Dataset	AL (repr.) F_1	AL (repr.) NB	AL (unc.) F_1	AL (unc.) NB	GL F_1	GL NB	XGL F_1	XGL NB
synthetic	0.55 ± 0.03	0.30	0.52 ± 0.01	0.34	0.47 ± 0.04	0.06	0.70 ± 0.12 •	-0.69 •
adult	0.66 ± 0.04	-0.17	0.67 ± 0.02 •	-0.15	0.66 ± 0.05	0.08	0.64 ± 0.06	-0.64 •
australian	0.80 ± 0.06	-0.28	0.81 ± 0.06	-0.31	0.79 ± 0.06	0.01	0.83 ± 0.07 •	-0.83 •
banknote	0.99 ± 0.04 •	-0.07	0.99 ± 0.04 •	-0.08	0.97 ± 0.04	0.00	0.99 ± 0.04 •	-0.19 •
cancer	0.95 ± 0.03	-0.19	0.96 ± 0.03 •	-0.18	0.93 ± 0.03	0.01	0.95 ± 0.02	-0.46 •
credit	0.61 ± 0.02	-0.08	0.61 ± 0.02	-0.10	0.58 ± 0.02	0.06	0.64 ± 0.01 •	-0.64 •
german	0.59 ± 0.03 •	-0.07	0.55 ± 0.03	-0.04	0.59 ± 0.02 •	0.02	0.53 ± 0.02	-0.53 •
glass	0.77 ± 0.03	-0.11	0.79 ± 0.03 •	-0.06	0.77 ± 0.03	0.02	0.77 ± 0.03	-0.47 •
heart	0.69 ± 0.08	-0.23	0.70 ± 0.06	-0.18	0.69 ± 0.06	0.01	0.71 ± 0.07 •	-0.71 •
hepatitis	0.64 ± 0.05	0.09	0.66 ± 0.07	0.08	0.67 ± 0.06	0.05	0.68 ± 0.05 •	-0.22 •
iris	0.94 ± 0.02	-0.03	0.95 ± 0.01 •	-0.01	0.94 ± 0.01	-0.00	0.94 ± 0.02	-0.08 •
magic	0.66 ± 0.05 •	-0.15	0.65 ± 0.06	-0.17	0.66 ± 0.03 •	0.04	0.64 ± 0.04	-0.64 •
phoneme	0.69 ± 0.07	-0.16	0.71 ± 0.04 •	-0.17	0.68 ± 0.05	0.03	0.63 ± 0.04	-0.63 •
plate-faults	0.65 ± 0.06	-0.07	0.62 ± 0.08	0.01	0.66 ± 0.08 •	0.09	0.66 ± 0.07 •	-0.65 •
risk	0.90 ± 0.05	0.08	0.95 ± 0.08	-0.20	0.96 ± 0.07 •	-0.00	0.96 ± 0.07 •	-0.37 •
wine	0.89 ± 0.02	0.03	0.90 ± 0.02	0.02	0.91 ± 0.03	0.03	0.95 ± 0.02 •	-0.17 •

- XGL achieves comparable performance and dramatically reduces narrative bias!

FIND (Lertvittayakumjorn et al., 2020)

- Local explanations present a **partial view** of the model
- Local fixes may have **unclear global effects**

- **FIND** (Feature Investigation aNd Disabling) paints a more complete picture of the learned model by combining information about *multiple* local explanations.



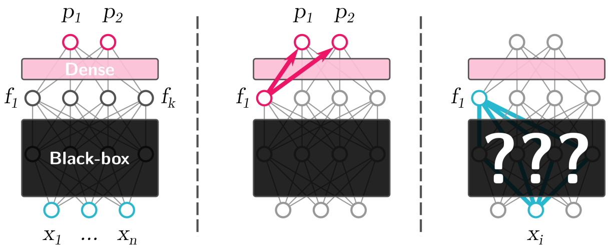
Source: (Lertvittayakumjorn et al., 2020).

FIND (Lertvittayakumjorn et al., 2020)

- NLP models prone to **bad word–class associations**, e.g., ⊕ sentiment word associated to ⊖ or neutral class
- Realistic models with (i) black-box embedding $M_f : x \mapsto \mathbf{f}$ and (ii) dense layer $M_c : \mathbf{f} \mapsto c$:

$$\mathbf{p} = (\mathrm{softmax} \circ M_c \circ M_f)(x)$$

Unclear what a latent feature f_j means, but **clear** how it maps to each class c



Makes it hard to map association between words (x) and classes probabilities (p)

FIND (Lertvittayakumjorn et al., 2020)

- For each h_j, **illustrate features** by identifying relevant words/n-grams using LRP (**?**). This gives a **word cloud** that combines relevance *across all training examples*

- User indicates **bad word–class associations** by clicking on words/n-grams in word cloud
- **Disable** bugged latent features:

$$\mathbf{p} = \mathrm{softmax}((W \odot Q)M_f(x) + \mathbf{b}))$$

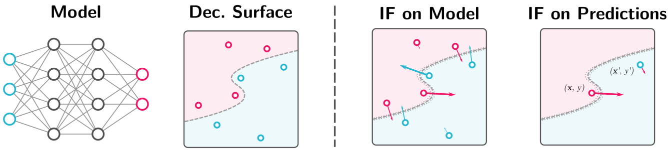
  where $\odot$ is element-wise multiplication and Q is a 0-1 “weight disabling matrix” built using user’s feed-back
- Fine-tune dense layer params W while keeping embeddings frozen

Related to (Lage and Doshi-Velez, 2020), more in Part 4.

Interacting via Example-based Explanations

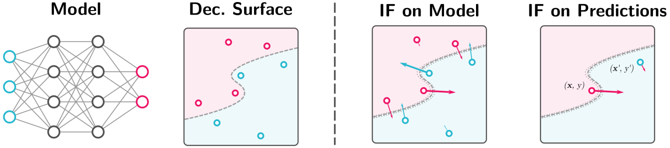
Example Attributions & Influence Functions

- Given a predictor f and a target decision $(\mathbf{x}, y)$, **example attributions** identify what **training examples** are responsible for the model f & for the decision. Ideal for **case-based** reasoning & debugging when features are not sufficient



Example Attributions & Influence Functions

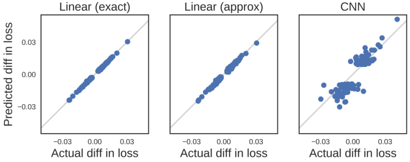
- Given a predictor f and a target decision $(\mathbf{x}, y)$, **example attributions** identify what **training examples** are responsible for the model f & for the decision. Ideal for **case-based** reasoning & debugging when features are not sufficient



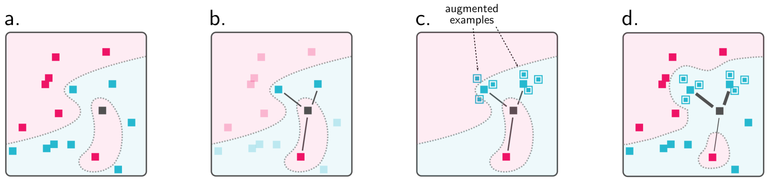
- **Influence Functions** (IFs) estimate impact of reweighting a training example **without retraining** (Koh and Liang, 2017):

$$\mathcal{I}(z) = -H(\theta_m)^{-1}\nabla_\theta \ell(z, \theta_m)$$

Use **chain rule** to eval. impact on class scores, loss, input gradients, *etc.*



- Model's predictions may rely on the wrong examples → **correct the model's IF**
- **HILDIF** for Natural Language Inference:
  - Select validation examples $\{(\mathrm{x}_i, y_i)\}$
  - For each of them, pick k **most influential** training points $\{(\tilde{\mathrm{x}}_{ij}, \tilde{y}_{ij})\}$
  - User supplies similarity $s_{ij} \in \{1, \ldots, 5\}$ between i-th and j-th examples
  - Augment data with $10 \times s_{ij}$ perturbed training points



Sequential Learning under Noise

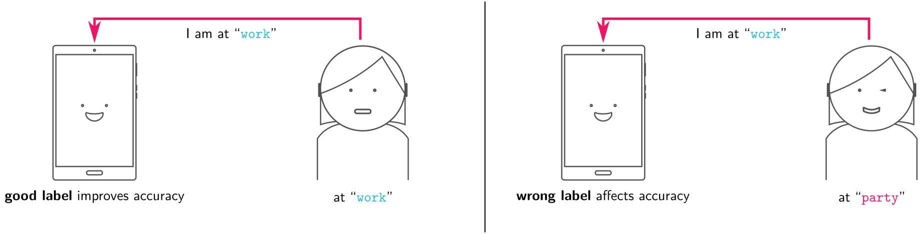
- Learning from **sequence** of examples $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots$ but **labels are noisy**!

Applications: crowd-sourcing, citizen science, interactive personal assistants learning from diary data, ...

Sequential Learning under Noise

- Learning from **sequence** of examples $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots$ but **labels are noisy**!

Applications: crowd-sourcing, citizen science, interactive personal assistants learning from diary data, ...



E.g., **inexperienced** annotators, **unwillingness** to self-report, $\rightarrow$ *poor performance*

Skeptical Learning (SKL)

- **Skeptical** machines challenge the user about suspicious examples

Intuition: suspicious examples $(\mathbf{x}, \tilde{y})$ identified by computing their **margin**:

$$\mu_\theta(\mathbf{x}) := p_\theta(\hat{y} \mid \mathbf{x}) - p_\theta(\tilde{y} \mid \mathbf{x})$$

i.e., the difference in likelihood between the model's **prediction** $\hat{y}$ and the observed **annotation** $\tilde{y}$

Skeptical Learning (SKL)

- The user is asked to **double-check** and **relabel** the suspicious examples

. . . this is often enough to correct mistakes due, e.g., to **inattention**

SKL cleans incoming examples only!

- Noise can **accumulate** in the training set, because of:
  - Pre-existing noisy data in the bootstrap set
  - Incoming noisy data that elude the skeptical check (e.g. at initialization, when the model is uncertain)

This seriously impacts both **prediction performance** and **ability to be skeptical**: incoming noisy examples falling close to corrupted regions are not detected

- SKL is also completely **black-box**:
  - The user has no clue why the model is skeptical – what if it is because of a **mistake in the training set**?

CINCER: XIL for Skeptical Learning (Teso et al., 2021)

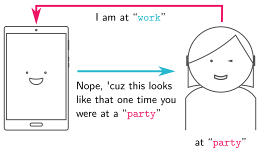
Ask (trustworthy) annotator to **double-check and relabel** incoming examples with suspiciously low likelihood:

$$\underbrace{p_\theta(\hat{y} \mid \mathbf{x})}_{\text{pred. label}} \gg \underbrace{p_\theta(\tilde{y} \mid \mathbf{x})}_{\text{annotation}}$$

Ask (trustworthy) annotator to **double-check and relabel** incoming examples with suspiciously low likelihood:

$$\underbrace{p_\theta(\hat{y} \mid \mathbf{x})}_{\text{pred. label}} \gg \underbrace{p_\theta(\tilde{y} \mid \mathbf{x})}_{\text{annotation}}$$

Skeptical for the right reasons



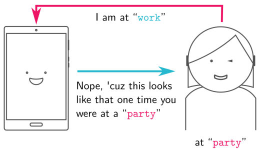
E.g., there is a past example "similar" to the current one but has a *different* label & it is annotated correctly.

CINCER: XIL for Skeptical Learning (Teso et al., 2021)

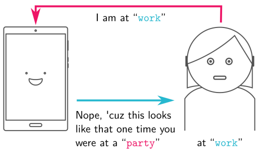
Ask (trustworthy) annotator to **double-check and relabel** incoming examples with suspiciously low likelihood:

$$\underbrace{p_\theta(\hat{y} \mid \mathbf{x})}_{\text{pred. label}} \gg \underbrace{p_\theta(\tilde{y} \mid \mathbf{x})}_{\text{annotation}}$$

Skeptical for the right reasons



Skeptical for the wrong reasons



E.g., there is a past example "similar" to the current one but has a *different* label & it is annotated correctly.

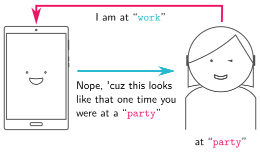
E.g., data in support of skepticism is annotated wrongly or past lies ;-)

CINCER: XIL for Skeptical Learning (Teso et al., 2021)

Ask (trustworthy) annotator to **double-check and relabel** incoming examples with suspiciously low likelihood:

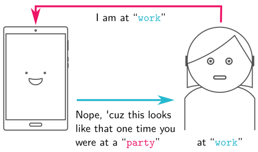
$$\underbrace{p_\theta(\hat{y} \mid \mathbf{x})}_{\text{pred. label}} \gg \underbrace{p_\theta(\tilde{y} \mid \mathbf{x})}_{\text{annotation}}$$

Skeptical for the right reasons



E.g., there is a past example "similar" to the current one but has a *different* label & it is annotated correctly.

Skeptical for the wrong reasons



E.g., data in support of skepticism is annotated wrongly or past lies ;-)

CINCER

Find tr. examples $z_k \in \mathcal{D}$ **explaining why model is skeptical** about new example $\tilde{z}_t$, user **fixes them**

Explain Skepticism using Counter-examples

- A **counter-example** is a **concrete** past example $z_k \in \mathcal{D}$ that is:

D1. **Contrastive**: it explains why $\tilde{z}_t$ is suspicious (highlighting an **inconsistency in data**)

D2. **Influential**: if mislabeled, correcting it should improve the model as much as possible.

- Are there examples that satisfy *both* desiderata? **Yes!**

Selecting a contrastive CE

Intuition: a CE is **contrastive** if *removing* it from the data set and retraining leads to a *less* suspicious model.

- Find CE $z_k \in \mathcal{D}$ for $\tilde{z}_t$ by maximizing the difference in likelihood:

$$\text{argmax}_{k \in [t-1]} \left\{ P(\tilde{y}_t \mid \mathbf{x}_t; \theta_{t-1}^{-k}) - P(\tilde{y}_t \mid \mathbf{x}_t; \theta_{t-1}) \right\}$$

with θ_{t-1} current params, θ_{t-1}^{-k} params after deleting CE z_k.

- Requires to **retrain $|t-1|$ times**; very impractical, especially in interactive settings.

Influence Functions

- **Influence Functions** (IFs) approximate the *change in parameters* due to reweighting a training example:

$$\mathcal{I}_{\theta_t}(z) := \left.\frac{d}{d\epsilon}\theta_t(z,\epsilon)\right|_{\epsilon=0} \tag{1}$$

$$\approx -H(\theta_t)^{-1}\nabla_\theta \ell(z,\theta_t) \tag{2}$$

where the Hessian is $H(\theta_t)$. IFs work even with **non-convex models** Koh and Liang (2017))

Influence Functions

- **Influence Functions** (IFs) approximate the *change in parameters* due to reweighting a training example:

$$\mathcal{I}_{\theta_t}(z) := \frac{d}{d\epsilon}\theta_t(z,\epsilon)\Big|_{\epsilon=0} \tag{1}$$

$$\approx -H(\theta_t)^{-1}\nabla_\theta \ell(z,\theta_t) \tag{2}$$

where the Hessian is $H(\theta_t)$. IFs work even with **non-convex models** Koh and Liang (2017)

Idea: use chain rule to compute the *change in likelihood* due to **removing** an example:

$$-\frac{1}{t-1}\left(\frac{d}{d\epsilon}P(\tilde{y}_t \mid \mathbf{x}_t;\theta_{t-1}(z_k,\epsilon))\Big|_{\epsilon=0}\right) = -\frac{1}{t-1}\left(\nabla_\theta P(\tilde{y}_t \mid \mathbf{x}_t;\theta_{t-1})^\top \frac{d}{d\epsilon}\theta_{t-1}(z_k,\epsilon)\Big|_{\epsilon=0}\right) \tag{3}$$

$$= -\frac{1}{t-1}\nabla_\theta P(\tilde{y}_t \mid \mathbf{x}_t;\theta_{t-1})^\top \mathcal{I}_{\theta_{t-1}}(z_k) \tag{4}$$

- **The Algorithm**: select contrastive CE $z_k \in \mathcal{D}$ by solving:

$$\operatorname{argmax}_{k\in[t-1]}\ \nabla_\theta P(\tilde{y}_t \mid \mathbf{x}_t;\theta_{t-1})^\top H(\theta_{t-1})^{-1}\nabla_\theta \ell(z_k,\theta_{t-1}) \tag{5}$$

Solve this by i) Caching the inverse Hessian-vector product (HVP), and ii) computing the inverse HVP with an efficient stochastic estimator

Contrastive CEs are Influential!

For the **cross-entropy loss** $\ell(z, \theta) = -\log P(y \mid \mathbf{x}; \theta)$:

$$\nabla_\theta P(\tilde{y}_t \mid \mathbf{x}_t; \theta_{t-1}) = P(\tilde{y}_t \mid \mathbf{x}_t; \theta_{t-1}) \nabla_\theta \log P(\tilde{y}_t \mid \mathbf{x}_t; \theta_{t-1}) = -P(\tilde{y}_t \mid \mathbf{x}_t; \theta_{t-1}) \nabla_\theta \ell(\tilde{z}_t, \theta_{t-1}) \tag{6}$$

Hence, the **objective** to be maximized can be rewritten as:

$$-\nabla_\theta \ell(\tilde{z}_t, \theta_{t-1})^\top H(\theta_{t-1})^{-1} \nabla_\theta \ell(z_k, \theta_{t-1}) \tag{7}$$

Under this assumption, counter-example selection becomes:

$$\text{argmax}_{k \in [t-1]} \; -\nabla_\theta \ell(\tilde{z}_t, \theta_{t-1})^\top H(\theta_{t-1})^{-1} \nabla_\theta \ell(z_k, \theta_{t-1}) \tag{8}$$

This recovers *exactly* the definition of influential examples: **highly influential counter-examples are highly contrastive and vice versa**!

- The same algorithm selects examples that are *both* contrastive *and* influential.

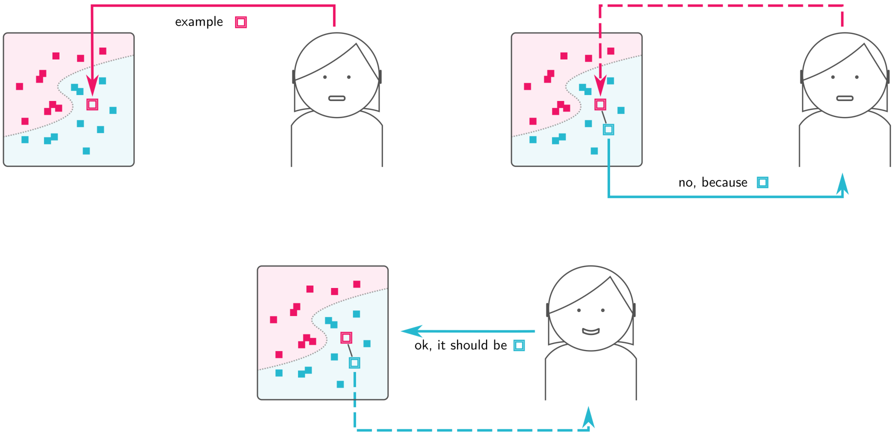
The CINCER Algorithm

Inputs: initial (noisy) training set $\mathcal{D}_0$; threshold τ.

Outputs:

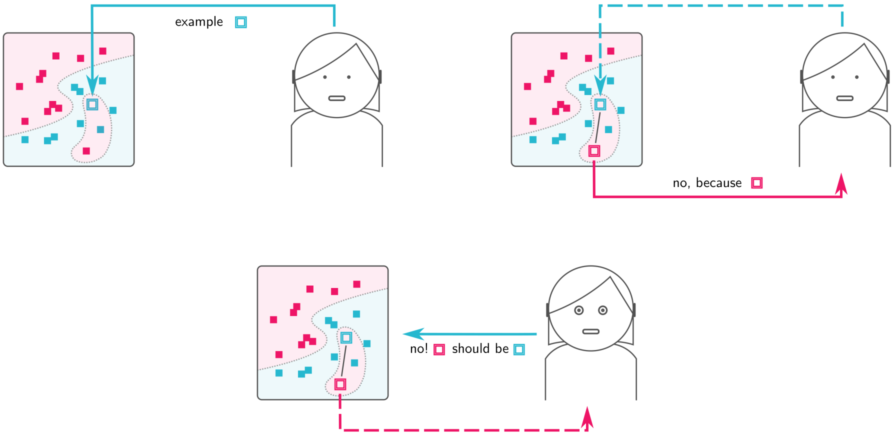
fit θ_0 on $\mathcal{D}_0$ ▷ Initialize model
for $t = 1, 2, \ldots$ **do**
  receive new example $\tilde{z}_t = (\mathbf{x}_t, \tilde{y}_t)$
  if $\mu(\tilde{z}_t, \theta_{t-1}) < \tau$ **then** ▷ Does $\tilde{z}_t$ look suspicious?
    $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{\tilde{z}_t\}$ ▷ No
  else
    find counterexample z_k using **the algorithm** ▷ Yes, find counter-example z_k
    present $\tilde{z}_t, z_k$ to the user, receive possibly cleaned labels y'_t, y'_k
    $\mathcal{D}_t \leftarrow (\mathcal{D}_{t-1} \setminus \{z_k\}) \cup \{(\mathbf{x}_t, y'_t), (\mathbf{x}_k, y'_k)\}$ ▷ Update data set
  fit θ_t on $\mathcal{D}_t$ ▷ Update model

CINCER: Skeptical for the Right Reasons



1) User supplies mislabeled example "■". 2) Machine catches the mistake because of low likelihood; CINCER identifies clean example in support of machine' skepticism. 3) Attentive user realizes and rectifies her mistake.

CINCER: Skeptical for the Wrong Reasons



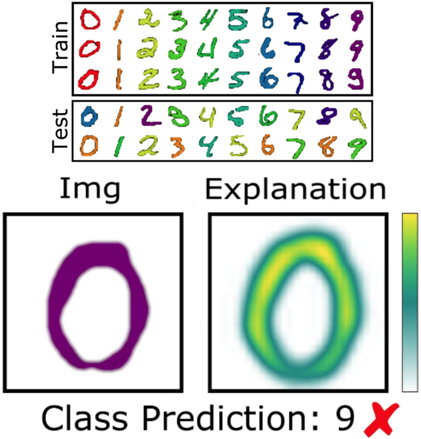
1) User supplies clean example “■”. 2) Machine is skeptical because of low likelihood; CINCER identifies mislabeled example in support of machine’ skepticism. 3) User relabels the mislabeled example.

Interacting via Concept-based Explanations

Lack of precision of input-based explanations for:

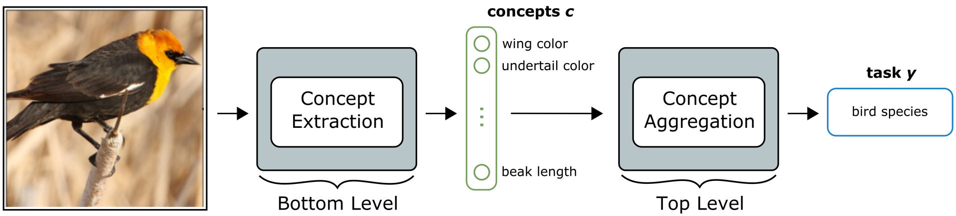
- Understandability
- Revisability

⇒ Difficult to access more abstract, concept-level reasons particularly for black-box models



Credit: (Stammer et al., 2021)

Concept-Based Models (CBMs)

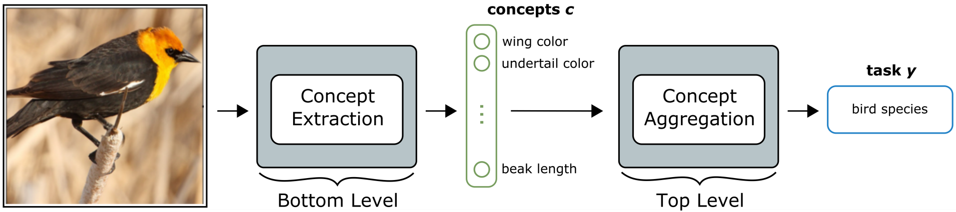


Modified figure from: (Koh et al., 2020)

Achieve partial, selective interpretability, via two step processing:

- Bottom level: $f(x) = c$, typically neural module extracts higher-level concepts from input samples
- Top level: $g(c) = y$, potentially more transparent module aggregates the concept activations for final prediction

Concept-Based Models (CBMs)



Modified figure from: (Koh et al., 2020)

- Model's decision is roughly independent of input given concept activations
- Concepts reperesentations are supervisedly or interactively human-aligned

- Auto-encoder based concepts, inspectable for human-user (Alvarez-Melis and Jaakkola, 2018)
- Prototype representations e.g. of training samples or parts there of (Chen et al., 2019; Barnett et al., 2021b)
- Concepts exlicitly learned from supervision (Koh et al., 2020; Chen et al., 2020)
- Interactively receiving feedback on feature-level dependencies (Lage and Doshi-Velez, 2020)

Part-Prototype Networks (aka ProtoPNets)

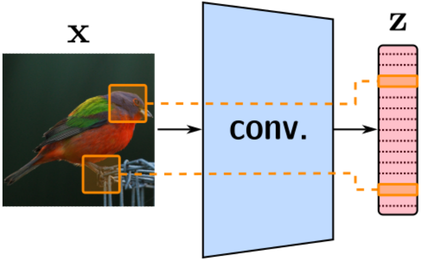
ProtoPNets Chen et al. (2019) are "gray-box" image classifiers that combine

- **transparent** reasoning
- **flexibility** of black-box neural networks

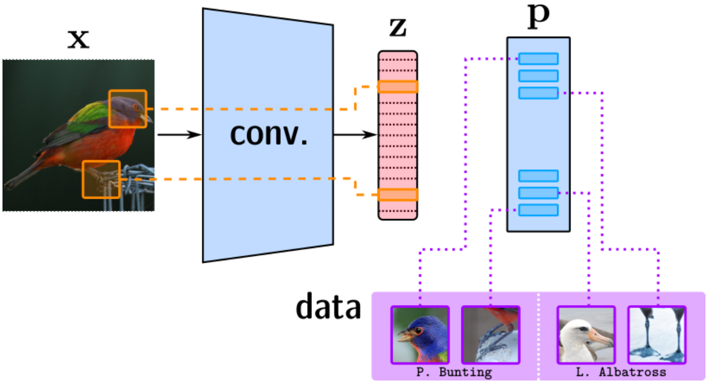
Prediction: compare the input image $\mathbf{x}$ with a set of **part-prototypes**, like objects or parts thereof.

Train: optimize the log-likelihood of the data and **two clustering** losses that encourage the part-prototypes to **strongly activate** only on examples of their associated class.

Embedding stage

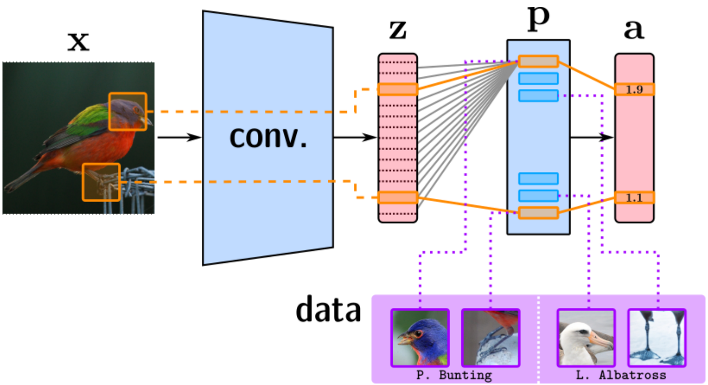


Part-Prototype stage

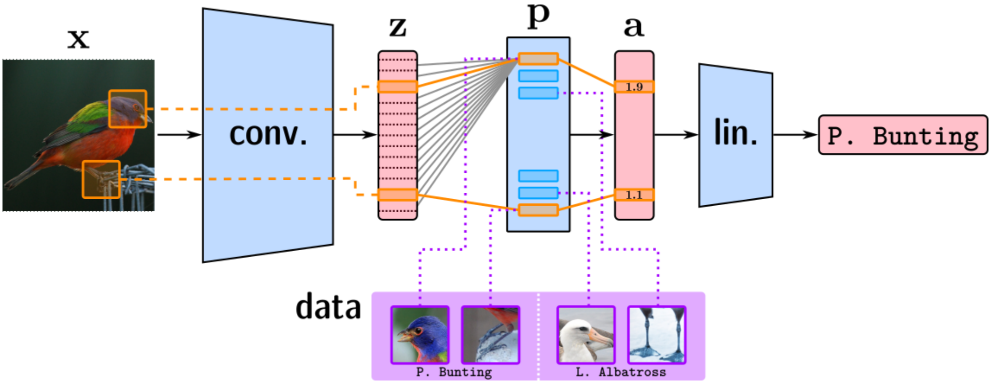


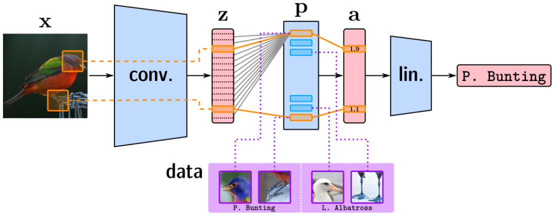
Part-Prototype Networks

Part-Prototype stage



Aggregation stage

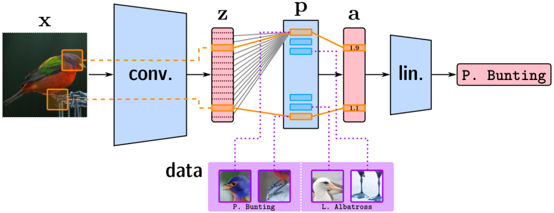




For each part-prototype $\mathbf{p}$, they compute:

1. **relevance** for the target decision $(\mathbf{x}, y)$ given by the score $w_j a_j(\mathbf{x})$

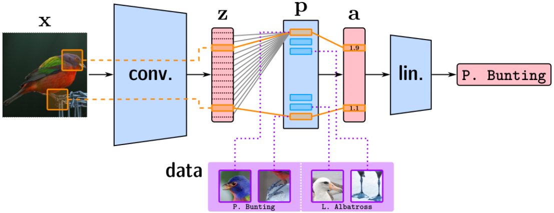
ProtoPNet explanations



For each part-prototype $\mathbf{p}$, they compute:

1. **relevance** for the target decision $(\mathbf{x}, y)$ given by the score $w_j a_j(\mathbf{x})$
2. **attribution map** $\mathrm{attr}(\mathbf{p}, \mathbf{x})$ showing where they activate on the input

ProtoPNet explanations



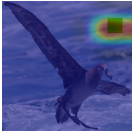
For each part-prototype $\mathbf{p}$, they compute:

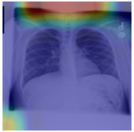
1. **relevance** for the target decision $(\mathbf{x}, y)$ given by the score $w_j a_j(\mathbf{x})$
2. **attribution map** $\mathrm{attr}(\mathbf{p}, \mathbf{x})$ showing where they activate on the input
3. **training example** that it projects onto

Confounding in ProtoPNets

Explanations expose confounds picked up from training data as part-prototypes.

Models exploit confounds to **maximize** training set performance.





- class-correlated watermarks
- irrelevant patches of background sky, sea or foliage
- borders of X-ray lung scans

Confounding in ProtoPNets

Explanations expose confounds picked up from training data as part-prototypes.

Models exploit confounds to **maximize** training set performance.

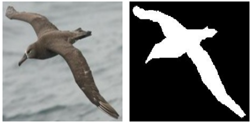
- class-correlated watermarks
- irrelevant patches of background sky, sea or foliage
- borders of X-ray lung scans

Issue: they impact **generalization** and **out-of-distribution** performance Lapuschkin et al. (2019).

How to **dissuade** the model from acquiring confounds?

Existing debugging strategies

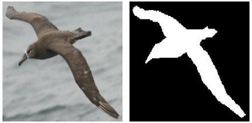
IAIA-BL **penalizes** part-prototypes activating on *pixels* annotated as irrelevant Barnett et al. (2021a)



Limitations:

- attribution masks **do not generalize** across images
- **substantial** number of examples must be annotated
- acquiring per-pixel attribution is **expensive**

Existing debugging strategies



- IAIA-BL **penalizes** part-prototypes activating on *pixels* annotated as irrelevant Barnett et al. (2021a)

Limitations:

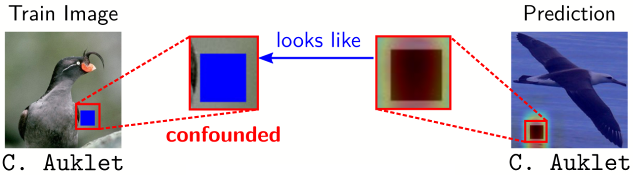
- attribution masks **do not generalize** across images
- **substantial** number of examples must be annotated
- acquiring per-pixel attribution is **expensive**

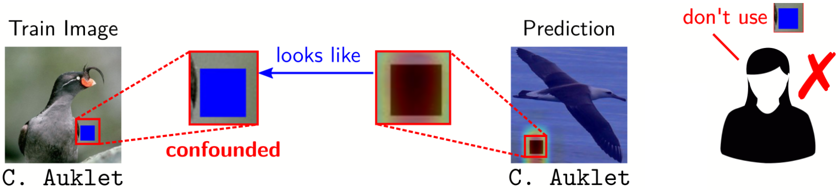
- **remove** confounded prototypes and **fine-tune** the network and/or the top layer

Limitations:

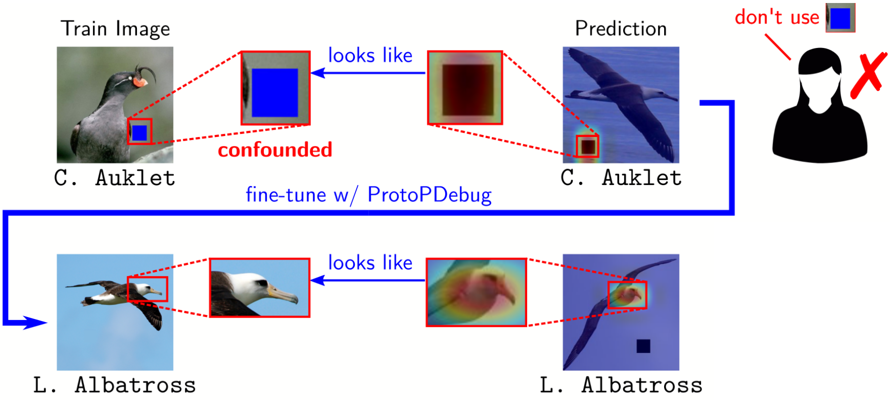
- **re-learning** the confound
- **no guarantee** that surviving prototypes capture meaningful information

Concept-level debugging with ProtoPDebug

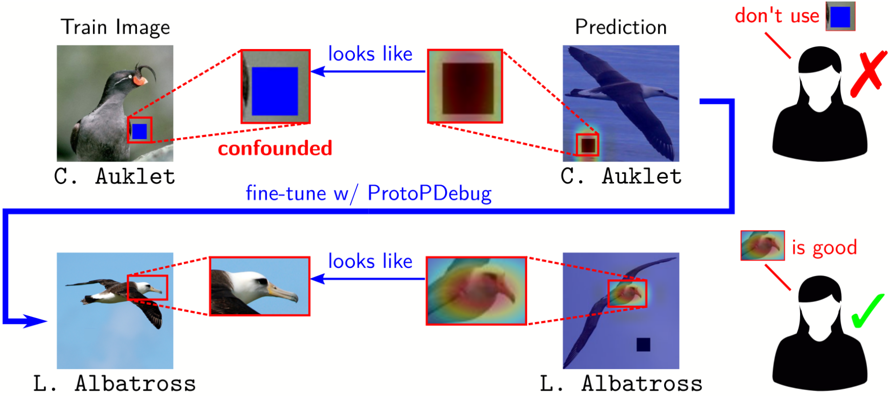



Train Image
looks like
Prediction
don't use
confounded
C. Auklet
C. Auklet

Concept-level debugging with ProtoPDebug



Concept-level debugging with ProtoPDebug



Fine-tuning the model

Forgetting loss minimizes how much the part-prototype $\mathbf{p}$ activates on the concept $\mathbf{f}$ to be forgotten

$$\ell_{\text{for}}(\theta) := \frac{1}{v} \sum_{y \in [v]} \max_{\substack{\mathbf{p} \in \mathcal{P}^y \\ \mathbf{f} \in \mathcal{F}_y}} \mathrm{act}(\mathbf{p}, \mathbf{f})$$

Fine-tuning the model

Forgetting loss minimizes how much the part-prototype $\mathbf{p}$ activates on the concept $\mathbf{f}$ to be forgotten

$$\ell_{\text{for}}(\theta) := \frac{1}{v} \sum_{y \in [v]} \max_{\substack{\mathbf{p} \in \mathcal{P}^y \\ \mathbf{f} \in \mathcal{F}_y}} \operatorname{act}(\mathbf{p}, \mathbf{f})$$

Remembering loss maximizes how much the part-prototype $\mathbf{p}$ activates on the concept $\mathbf{v}$ to be remembered

$$\ell_{\text{rem}}(\theta) := -\frac{1}{v} \sum_{y \in [v]} \min_{\substack{\mathbf{p} \in \mathcal{P}_y \\ \mathbf{v} \in \mathcal{V}_y}} \operatorname{act}(\mathbf{p}, \mathbf{v})$$

Fine-tuning the model

Forgetting loss minimizes how much the part-prototype $\mathbf{p}$ activates on the concept $\mathbf{f}$ to be forgotten

$$\ell_{\mathrm{for}}(\theta) := \frac{1}{v} \sum_{y \in [v]} \max_{\substack{\mathbf{p} \in \mathcal{P}^y \\ \mathbf{f} \in \mathcal{F}_y}} \mathrm{act}(\mathbf{p}, \mathbf{f})$$

Remembering loss maximizes how much the part-prototype $\mathbf{p}$ activates on the concept $\mathbf{v}$ to be remembered

$$\ell_{\mathrm{rem}}(\theta) := -\frac{1}{v} \sum_{y \in [v]} \min_{\substack{\mathbf{p} \in \mathcal{P}_y \\ \mathbf{v} \in \mathcal{V}_y}} \mathrm{act}(\mathbf{p}, \mathbf{v})$$

Overall loss to fine-tuning the model

$$\overbrace{\ell(\theta)}^{\text{ProtoPNets}} + \overbrace{\lambda_{\mathrm{f}} \ell_{\mathrm{for}}(\theta) + \lambda_{\mathrm{r}} \ell_{\mathrm{rem}}(\theta)}^{\text{ProtoPDebug}}$$

A debugging session with ProtoPDebug

In each round:

1. iterates over all learned part-prototypes $\mathbf{p} \in \mathcal{P}$
2. for each of them retrieves the training examples $\mathbf{x}$ that it **activates the most on**

A debugging session with ProtoPDebug

In each round:

1. iterates over all learned part-prototypes $\mathbf{p} \in \mathcal{P}$
2. for each of them retrieves the training examples $\mathbf{x}$ that it **activates the most on**
3. if $\mathbf{p}$ appears **confounded** to user $\rightarrow$ add cut-out $\mathbf{x}_R$ to **forbidden concepts**
4. if $\mathbf{p}$ appears **high-quality** to user $\rightarrow$ add cut-out $\mathbf{x}_R$ to **valid concepts**

In each round:

1. iterates over all learned part-prototypes $\mathbf{p} \in \mathcal{P}$
2. for each of them retrieves the training examples $\mathbf{x}$ that it **activates the most on**
3. if $\mathbf{p}$ appears **confounded** to user $\rightarrow$ add cut-out $\mathbf{x}_R$ to **forbidden concepts**
4. if $\mathbf{p}$ appears **high-quality** to user $\rightarrow$ add cut-out $\mathbf{x}_R$ to **valid concepts**
5. if no confounds found, **terminate**
6. **fine-tune** by minimizing $\ell(\theta) + \lambda_{\mathsf{f}} \ell_{\text{for}}(\theta) + \lambda_{\mathsf{r}} \ell_{\text{rem}}(\theta)$

Concept-based debugging

input-level	**concept-based** (ProtoPDebug)
penalizes part-prototypes that activate on **pixels annotated as irrelevant**	Penalizes part-prototype that correlate with **known-forbidden concepts**

Concept-based debugging

input-level	**concept-based** (ProtoPDebug)
penalizes part-prototypes that activate on **pixels annotated as irrelevant**	Penalizes part-prototype that correlate with **known-forbidden concepts**

Advantages of concept-based supervision:

- **generalizes** across images:

  one concept-level annotation = **several** pixel-level annotations

Concept-based debugging

input-level	**concept-based** (ProtoPDebug)
penalizes part-prototypes that activate on **pixels annotated as irrelevant**	Penalizes part-prototype that correlate with **known-forbidden concepts**

Advantages of concept-based supervision:

- **generalizes** across images:

  one concept-level annotation = **several** pixel-level annotations

- **speeds up** convergence, avoids relapse

Concept-based debugging

input-level	**concept-based** (ProtoPDebug)
penalizes part-prototypes that activate on **pixels annotated as irrelevant**	Penalizes part-prototype that correlate with **known-forbidden concepts**

Advantages of concept-based supervision:

- **generalizes** across images:

  one concept-level annotation = **several** pixel-level annotations

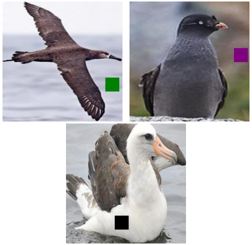
- **speeds up** convergence, avoids relapse
- **cheap** click-based feedback by showing part-prototypes activation on images

Empirical Analysis

Experiment 1	Concept-level debugging is useful ...
Experiment 2	... even for natural confounds ...
Experiment 3	... and in high-stakes applications.

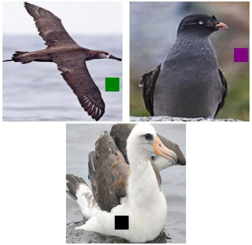
Experiment 1: Concept-level vs instance-level debugging

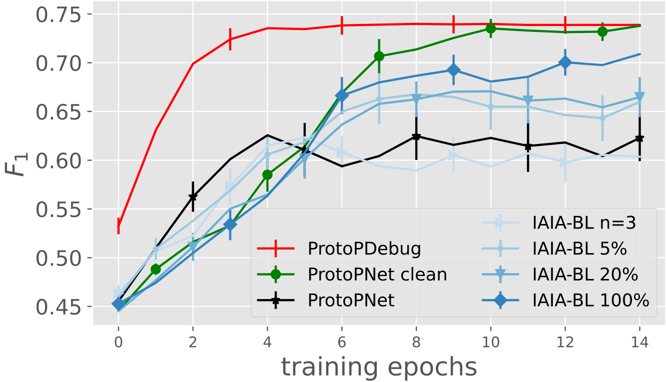
CUB5$_{box}$: synthetic colored square added to 3 classes out of 5



Experiment 1: Concept-level vs instance-level debugging

CUB5$_{box}$: synthetic colored square added to 3 classes out of 5





IAIA-BL with 100% pixel annotations **fails** to match the confound-free **ProtoPNets**$_{clean}$

ProtoPDebug with only 3 single clicks **reaches** the performance of **ProtoPNets**$_{clean}$

Experiment 2: ProtoPDebug on CUB5$_{nat}$

CUB5$_{nat}$:

- 5 *most confounded* CUB200 classes

- contains **natural confounds** (patches of sky or foliage)

- swapped backgrounds of test images to prevent confounding to affect performance

Experiment 2: ProtoPDebug on CUB5$_{nat}$

three rounds of sequential debugging

	ProtoPNets	**ProtoPDebug**
test F_1	0.56	0.62 (**+0.08**)
interpretability	0.68	0.88 (**+0.20**)

CUB5$_{nat}$:

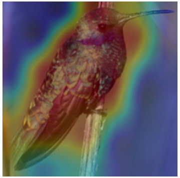
- 5 *most confounded* CUB200 classes

- contains **natural confounds** (patches of sky or foliage)

- swapped backgrounds of test images to prevent confounding to affect performance

ProtoPNets

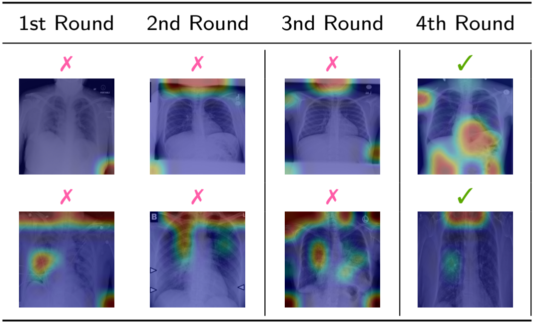
ProtoPDebug



ProtoPDebug improves over **ProtoPNets** and learn more interpretable prototypes

Experiment 3: ProtoPDebug on COVID

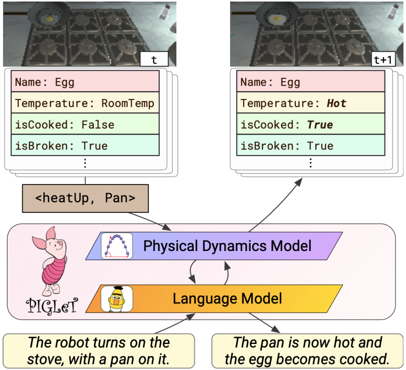
- COVID- vs. COVID+
- data sets: **????**
- four rounds of sequential debugging



test F_1: **ProtoPNets** (first column) 0.26 → 0.54 **ProtoPDebug** (last column)

Grounded symbolic explanations via transformer-based physics dynamics model

- Use a pre-trained language module (BERT)
- Model learns grounded commonsense knowledge via interactions in a simulated physics environment
- Learns to communicate action predictions in symbolic form



Credit: (Zellers et al., 2021)

Interpretability-by-design?

- What is a concept?

- Symbolic communication requires machine concepts to roughly match those understood and used by the user – in *both* directions.

- Usually machine acquires concepts from data using heuristics – e.g., autoencoders, prototypes, etc. – but this gives no guarantees.

- Even supplying concept-level supervision is **not enough** to ensure that concepts have the right semantis (e.g., concept leakage)

Interpretability-by-design?

- What is a concept?

- Symbolic communication requires machine concepts to roughly match those understood and used by the user – in *both* directions.

- Usually machine acquires concepts from data using heuristics – e.g., autoencoders, prototypes, etc. – but this gives no guarantees.

- Even supplying concept-level supervision is **not enough** to ensure that concepts have the right semantis (e.g., concept leakage)

Take-away

- Explanations are a vital part in understanding the internal reasoning of a model and identify **bugs**
- Annotators **naturally** supply explanations (**?**).
- Aligning model's explanations with user's corrections **fixes bugs**
- We covered human-in-the-loop strategies for **modern** ML models / XAI techniques
- Research indicates particular advantage of symbolic, concept-level explanations for human-machine interactions (Kambhampati et al., 2021).
- Several exciting challenges remain, maybe for you to tackle?

■ Curated literature available at: github.com/awesome-explanatory-supervision

References

Alvarez-Melis, D. and Jaakkola, T. S. (2018). Towards robust interpretability with self-explaining neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 7786–7795.

Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., and Müller, K.-R. (2010). How to explain individual classification decisions. *The Journal of Machine Learning Research*, 11:1803–1831.

Barnett, A. J., Schwartz, F. R., Tao, C., Chen, C., Ren, Y., Lo, J. Y., and Rudin, C. (2021a). A case-based interpretable deep learning model for classification of mass lesions in digital mammography. *Nat. Mach. Intell.*

Barnett, A. J., Schwartz, F. R., Tao, C., Chen, C., Ren, Y., Lo, J. Y., and Rudin, C. (2021b). Iaia-bl: A case-based interpretable deep learning model for classification of mass lesions in digital mammography. *arXiv preprint arXiv:2103.12308*.

Bastings, J. and Filippova, K. (2020). The elephant in the interpretability room: Why use attention as explanation when we have saliency methods? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 149–155.

Chen, C., Li, O., Tao, D., Barnett, A., Rudin, C., and Su, J. K. (2019). This looks like that: Deep learning for interpretable image recognition. *Advances in Neural Information Processing Systems*, 32:8930–8941.

Chen, Z., Bei, Y., and Rudin, C. (2020). Concept whitening for interpretable image recognition. *Nature Machine Intelligence*, 2(12):772–782.

DeGrave, A. J., Janizek, J. D., and Lee, S.-I. (2021). Ai for radiographic covid-19 detection selects shortcuts over signal. *Nature Machine Intelligence*, 3(7):610–619.

Heo, J., Park, J., Jeong, H., Kim, K. J., Lee, J., Yang, E., and Hwang, S. J. (2020). Cost-effective interactive attention learning with neural attention processes. In *International Conference on Machine Learning*, pages 4228–4238. PMLR.

Honeycutt, D., Nourani, M., and Ragan, E. (2020). Soliciting human-in-the-loop user feedback for interactive machine learning reduces user trust and impressions of model accuracy. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 8, pages 63–72.

Kambhampati, S., Sreedharan, S., Verma, M., Zha, Y., and Guan, L. (2021). Symbols as a lingua franca for bridging human-ai chasm for explainable and advisable ai systems. *arXiv preprint arXiv:2109.09904*.

Koh, P. W. and Liang, P. (2017). Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1885–1894.

Koh, P. W., Nguyen, T., Tang, Y. S., Mussmann, S., Pierson, E., Kim, B., and Liang, P. (2020). Concept bottleneck models. In *International Conference on Machine Learning*, pages 5338–5348. PMLR.

Kulesza, T., Burnett, M., Wong, W.-K., and Stumpf, S. (2015). Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th international conference on intelligent user interfaces*, pages 126–137.

Lage, I. and Doshi-Velez, F. (2020). Learning interpretable concept-based models with human feedback. *arXiv preprint arXiv:2012.02898*.

Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W., and Müller, K.-R. (2019). Unmasking clever hans predictors and assessing what machines really learn. *Nature communications*, 10(1):1–8.

Lertvittayakumjorn, P., Specia, L., and Toni, F. (2020). Find: human-in-the-loop debugging deep text classifiers. In *Conference on Empirical Methods in Natural Language Processing*, pages 332–348.

Popordanoska, T., Kumar, M., and Teso, S. (2020). Machine guides, human supervises: Interactive learning with global explanations. *arXiv preprint arXiv:2009.09723*.

Rieger, L., Singh, C., Murdoch, W., and Yu, B. (2020). Interpretations are useful: penalizing explanations to align neural networks with prior knowledge. In *International Conference on Machine Learning*, pages 8116–8126. PMLR.

Ross, A. S., Hughes, M. C., and Doshi-Velez, F. (2017). Right for the right reasons: training differentiable models by constraining their explanations. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2662–2670.

Schramowski, P., Stammer, W., Teso, S., Brugger, A., Herbert, F., Shao, X., Luigs, H.-G., Mahlein, A.-K., and Kersting, K. (2020). Making deep neural networks right for the right scientific reasons by interacting with their explanations. *Nature Machine Intelligence*, 2(8):476–486.

Shao, X., Skryagin, A., Schramowski, P., Stammer, W., and Kersting, K. (2021). Right for better reasons: Training differentiable models by constraining their influence function. In *Proceedings of Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*.

Simpson, B., Dutil, F., Bengio, Y., and Cohen, J. P. (2019). Gradmask: Reduce overfitting by regularizing saliency. In *International Conference on Medical Imaging with Deep Learning–Extended Abstract Track*.

Stammer, W., Schramowski, P., and Kersting, K. (2021). Right for the right concept: Revising neuro-symbolic concepts by interacting with their explanations. In *Conference on Computer Vision and Pattern Recognition*, pages 3619–3629.

Teso, S. (2019). Toward faithful explanatory active learning with self-explainable neural nets. In *Proceedings of the Workshop on Interactive Adaptive Learning (IAL 2019)*, pages 4–16.

Teso, S., Alkan, Ö., Stammer, W., and Daly, E. (2022). Leveraging explanations in interactive machine learning: An overview. *arXiv preprint arXiv:2207.14526*.

Teso, S., Bontempelli, A., Giunchiglia, F., and Passerini, A. (2021). Interactive label cleaning with example-based explanations. In *NeurIPS'21*.

Teso, S. and Kersting, K. (2019). Explanatory interactive machine learning. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 239–245.

Zellers, R., Holtzman, A., Peters, M. E., Mottaghi, R., Kembhavi, A., Farhadi, A., and Choi, Y. (2021). Piglet: Language grounding through neuro-symbolic interaction in a 3d world. In *Annual Meeting of the Association for Computational Linguistics*, pages 2040–2050.

Zylberajch, H., Lertvittayakumjorn, P., and Toni, F. (2021). Hildif: Interactive debugging of nli models using influence functions. *Workshop on Interactive Learning for Natural Language Processing*, page 1.