Bridging the Faithfulness Gap in Prototypical Models

Andrew Koulogeorge[♠,*] akouloge@andrew.cmu.edu

Sean Xie[†,*] sean.xie.gr@dartmouth.edu

Saeed Hassanpour[†,‡] saeed.hassanpour@dartmouth.edu

Soroush Vosoughi[†] soroush.vosoughi@dartmouth.edu

♠ Department of Computer Science, Carnegie Mellon University

†Department of Computer Science, ‡Department of Biomedical Data Science, Dartmouth College

* indicates equal contribution.

Abstract

Prototypical Network-based Language Models (PNLMs) have been introduced as a novel approach for enhancing interpretability in deep learning models for Natural Language Processing (NLP). In this work, we show that, despite the transparency afforded by their case-based reasoning architecture, current PNLMs are, in fact, not faithful, i.e. their explanations do not accurately reflect the underlying model's reasoning process. By adopting an axiomatic approach grounded in the seminal works' definition of faithfulness, we identify two specific points in the architecture of PNLMs where unfaithfulness may occur. To address this, we introduce `Faithful Alignment (FA)`, a two-part framework that ensures the faithfulness of PNLMs' explanations. We then demonstrate that `FA` achieves this goal without compromising model performance across downstream tasks and ablation studies.

1 Introduction

In recent years, deep learning-based language models have drastically enhanced performance across various NLP tasks (Vaswani et al., 2017; Radford et al., 2021). Despite their high predictive accuracy, these models remain opaque, meaning their decision-making processes are not easily understandable to humans. Consequently, numerous Explainable AI (XAI) techniques have been developed to interpret model decisions for end users (Ribeiro et al., 2016b; Lundberg and Lee, 2017; Shrikumar et al., 2019). A recent advancement in the XAI domain is the use of prototypical networks for interpretability. Originally introduced for few-shot learning (Snell et al., 2017), prototypical networks offer a unique advantage in interpretability due to their case-based reasoning architecture. Although initially adapted for increased interpretability in Computer Vision (CV) tasks (Chen et al., 2019; Hase et al., 2019; Ma et al., 2023), several recent works in NLP have begun to develop similar prototypical-based models to enhance interpretability in NLP contexts (Xie et al., 2023; Das et al., 2022; Van Aken et al., 2022; Friedrich et al., 2022). Despite their application in a wide range of tasks from propaganda detection (Das et al., 2022) to ICD-9 diagnosis prediction (Van Aken et al., 2022), the *faithfulness*[1] of PNLMs remains unexamined. Faithfulness is a necessary condition for any deployed machine learning model since unfaithful model explanations can lead to dangerous outcomes such as leading a user to trust a model's *incorrect* prediction simply because its explanation looks convincing (Rudin, 2018; Bansal et al., 2021; Lyu et al., 2024).

In this work, we assess the faithfulness of PNLMs using axioms from seminal interpretability studies (Chen et al., 2019) and identify two key flaws in state-of-the-art PNLM architectures (Xie et al., 2023; Das et al., 2022; Van Aken et al., 2022) that result in explanations of current PNLMs to be **unfaithful** (see Figure 1). To address these shortcomings, we propose `Faithful Alignment`, a two-part framework designed to ensure faithful prototypical model explanations. Our contributions are as follows:

1. We define reasoning in prototypical models as comprising (1) class connections in the final linear layer and (2) the similarity between the encoded test example and the learned prototypes in prototypical space. Through this lens, we identify two areas in the existing PNLMs' workflow where their provided explanations deviate from model reasoning and thus are unfaithful.

2. We propose a solution in the form of the `Faithful Alignment` framework (hence-

[1]For additional details on PNLMs, interpretability and faithfulness, we refer the reader to §A

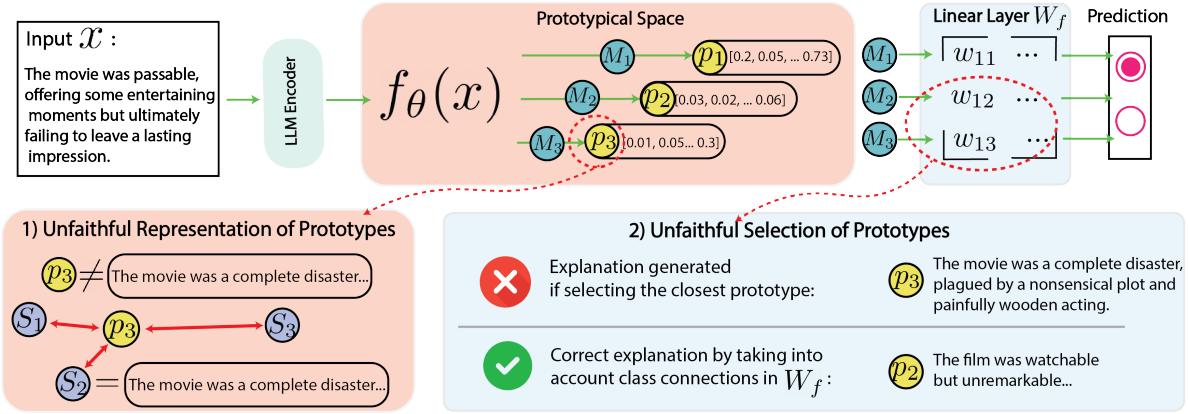

Figure 1: General work flow of Prototypical Network-based Language Models (PNLM). $f_\theta(x)$ is the compressed encoding of the input at test time. p_i is the ith learned prototype and S_i is the compressed encoding of the ith training example. Red circles indicate architectural flaws which can lead to unfaithful explanations: 1) Unfaithful explanations may result from a prototype vector being misrepresented by text from another latent encoding. 2) Unfaithful explanations may result from the wrong prototype being selected as most influential on prediction.

force abbreviated as FA), which addresses the aforementioned faithfulness issues in PNLMs.

3. We empirically validate that our solution for faithfulness in PNLMs does not significantly degrade performance across a wide range of PNLM architectures and tasks (§4). We conduct additional ablation studies to demonstrate the robustness of our framework (§E).

2 Unfaithfulness in Prototypical Networks

We briefly outline the PNLM workflow before describing two architectural shortcomings that lead to unfaithfulness. Let f_θ be a language model (e.g., BERT) such that $f_\theta : \mathbb{R}^{L\times d} \rightarrow \mathbb{R}^H$ where f_θ maps x, a sequence of L embedded tokens, to a compressed hidden representation $f_\theta(x) \in \mathbb{R}^H$. Through specialized training objectives, the PNLM learns $P = \{p_j\}_{j=1}^m$, a set of m prototypes where $p_j \in \mathbb{R}^H$. During inference, the PNLM feeds $f_\theta(x)$ into the prototypical space to obtain $M \in \mathbb{R}^m$ where M_j is the similarity between $f_\theta(x)$ and p_j. These similarities in M are then either passed through a final linear layer $W_f \in \mathbb{R}^{C\times m}$ to produce logits for prediction[2] (Das et al., 2022; Xie et al., 2023) or are used directly to obtain a prediction (Van Aken et al., 2022). This architecture enables PNLMs to generate textual explanations by (a) identifying the most influential prototypes for prediction and (b) extracting textual representation from those prototypes. In the following subsections, we analyze the unfaithfulness of existing PNLMs through the lenses of (a) and (b).

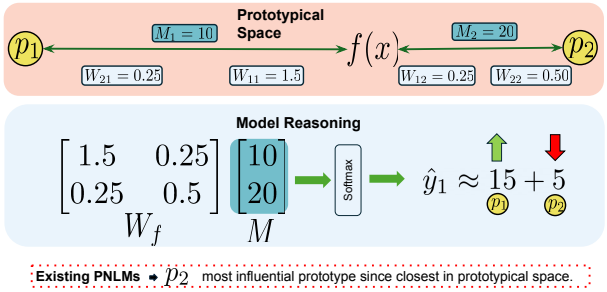

Figure 2: Example of how existing PNLMs are susceptible to incorrectly identifying the most influential prototypes. The prototype with the largest influence on prediction is p_1 due to W_{11}'s large magnitude, despite p_2 being closest to $f_\theta(x)$ in the prototypical space. Since existing PNLM's derive their explanations from prototypes chosen solely based on distance in the prototypical space, existing works would incorrectly identify p_2 as most influential on prediction.

2.1 Unfaithful Selection of Prototypes

First, we investigate how existing PNLMs can exhibit unfaithfulness when selecting the most influential prototypes. Despite Xie et al. (2023) and Das et al. (2022) claiming to use the most influential prototypes when deriving their explanations, both *only* consider proximity in the prototypical space when ranking prototype influence. **By neglecting the impact of W_f on prediction, previous works do not capture the full reasoning pro-**

[2]Similarity calculation and training objectives for existing PNLMs vary. For details, please refer to §C.2

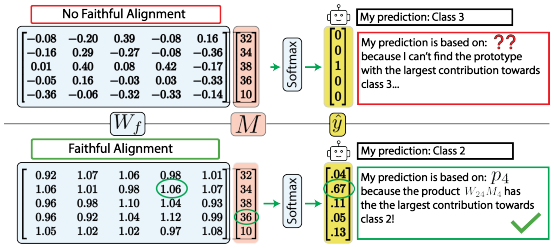

Figure 3: Empirical Demonstration of Faithful Retrieval with ProtoTex model on SST5 with 5 prototypes. (Top) without FR, negative reasoning in W_f clouds the model's ability to discern the most influential prototype. (Bottom) After FR, the PNLM is able to unambiguously determine the most influential prototype.

cess of PNLMs and thus are prone to unfaithful explanations (We show such an example in Figure 2). In this section, we eliminate the ambiguity surrounding reasoning for PNLMs by defining it *directly* based on the computations that dictate model output. The output logit of class c is calculated as:

$$\hat{y}_c = \sum_{j=1}^{m} W_{cj} M_j \tag{1}$$

where the product $W_{cj}M_j$ is the *total* contribution of the jth prototype on predicting the cth class. Because similarity measures are multiplied by W_f to obtain logits, **the reasoning process of PNLMs is the combined effect of distance calculation and weighted product in the final layer**. Since faithfulness is defined as the extent to which explanations accurately reflect a model's reasoning process (Jacovi and Goldberg, 2020), and W_f directly affects model prediction, current PNLMs that neglect the impact of W_f when generating the most influential prototypes are **unfaithful**. In this work, we contend that to accurately represent model reasoning, PNLMs (and Prototypical Networks at large) *must* generate their explanations using both the similarity between the example $f_\theta(x)$ and prototype p_j, along with the $W_{cj} \in W$ weighting their distance[3]. Moreover, current PNLM architectures (Das et al., 2022; Xie et al., 2023) permit negative parameters in W_f which leads the model to utilize negative reasoning, thus obscuring the determination of the prototype that contributed most significantly to model output. Specifically, the model may predict a particular class because it's confident the example *does not* belong to other classes. In Figure 3 (Top), we show an instance of how negative parameters in W_f create ambiguity in identifying the most contributing prototype *even when accounting for the combined effects of similarity measures M and their weights W_f.*

[3]Note that for (Van Aken et al., 2022) $W_f = I$

2.2 Unfaithful Representation of Prototypes

Second, unfaithfulness can occur in PNLMs when prototypes are represented by text that inaccurately reflects model reasoning. Let D be the training data set and $S = \{f_\theta(x_i) \mid \forall\, x_i \in D\}$ be the set of encoded training data where we denote $S_i = f_\theta(x_i)$. Since each prototype $p_j \in P$ is a learned, dense vector in $\mathbb{R}^H$, there does not exist a direct, human-interpretable textual representation of p_j. In order to render these inherently opaque prototypes understandable to humans, existing PNLMs represent their prototypes using the text of the most similar [4] encoded training example S_i. More formally, a PNLM obtains a human-readable textual representation for prototype p_j via the following assignation:

$$\text{Text of } [p_j] \leftarrow \text{Text of } \left[\arg\min_{S_i} \|S_i - p_j\|_2^2\right] \tag{2}$$

where Text of $[S_i]$ is the raw text associated with the embedded token sequence x_i. We illustrate this process in Figure 4 (Left). Since prototypes are represented by S_i, which does not partake in *any* of the model's computation, explanations utilizing S_i do not accurately reflect model reasoning and are therefore unfaithful (Jacovi and Goldberg, 2020). We empirically evaluate this imprecision in explanations as the *faithfulness gap* in Figure 4 (Right). If current PNLMs are indeed *inherently faithful* as they claim to be, our experiments should reveal the textual representations (S_i) to align with the parameters involved in computation, i.e. there should be no difference between each textual encoding S_i and p_j (green box). Nonetheless, our experiments in Figure 4 show that there exists a non-zero faithfulness gap for every prototype in SOTA PNLM architectures (Das et al., 2022; Xie et al., 2023).

3 Solution: Faithful Alignment

In this section we describe `Faithful Alignment` (FA), a two-part solution consisting of `Faithful Retrieval` (FR) and `Faithful Projection` (FP).

[4]The most similar example is defined as the minimum L_2 distance in the prototypical space

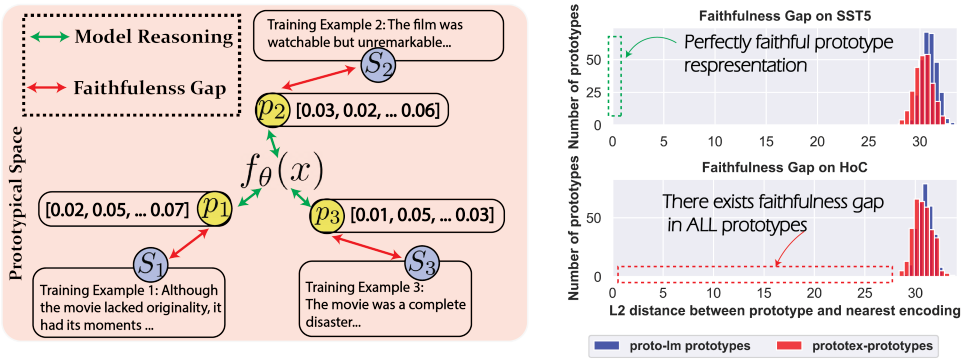

Figure 4: (Left) Faithfulness Gap in the Prototypical Space for PNLMs. The distance between a test time example $f(x)$ and the learned prototypes p_j influences the model's prediction, yet the model's explanations for p_j are derived from S_i which *does not* participate in the computation of the output. (Right) Faithfulness gap evaluated for `proto-lm` and `ProtoTex` on two datasets (SST5 and HoC). For each learned prototype p_j, we compute the L_2 distance to the encoding of its assigned textual representation. The green box indicates prototype representations that would be perfectly faithful i.e. 0 faithfulness gap. The red box highlights the fact that there exists a faithfulness gap for all prototypes in both models, empirically demonstrating the misalignment between model reasoning and model explanations.

3.1 `Faithful Retrieval`

To enable PNLMs to faithfully select prototypes for explanations, we propose the following two constraints, which, together, form `Faithful Retrieval`:

- If k textual explanations are provided by a PNLM, the k prototypes must be retrieved via:

$$\arg\max_{j}{}_k \{W_{\hat{c}j} M_j \mid \forall j \in [m]\} \quad (3)$$

  where $W_{\hat{c}l}$ represents the class connection between the output class $\hat{c}$ and prototype j. More concretely, suppose that the predicted class is $\hat{c}$. `FR` then selects top k most influential prototypes computing all m weight/distance pairs $W_{\hat{c}j} M_j$ and returning prototypes at the largest k pairs' indices.

- All class connections in W_f that weigh incoming similarity measures must be positive

$$\{W_{cj} > 0 \mid \forall j \in [m], \forall c \in C\} \quad (4)$$

Equation 3 ensures that prototypes' influence takes into account both similarity M and weights W_f. Equation 4 prevents PNLM from utilizing negative reasoning in the final layer which obstructs the identification of the most influential prototype. In Figure 3 (bottom), we show an example where the application of `FR` allows us to unambiguously determine the top k most influential prototypes by directly comparing their contributions ($W_{cj} M_j$) towards the output. Together, Equations 3 and 4 formalize the procedure for robust prototype selection, a crucial detail neglected by the works of Das et al. (2022) and Xie et al. (2023) that lead to unfaithfulness.

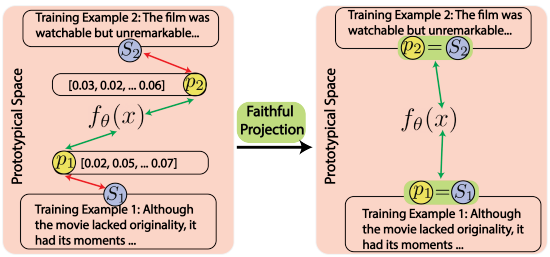

Figure 5: Illustration of Faithful Projection's effect in the prototypical space. `FP` aligns the PNLM's reasoning and explanations by forcing each prototype to become exactly equal to an encoded training example.

3.2 `Faithful Projection`

To ensure that PNLM explanations participate in the models computation, we propose Faithful Projection (FP) which endows each prototype with an encoding from the training dataset. More formally, let $S_i \in S$ be the encoding of a known example in D. For each prototype in the prototypical layer, we perform the following projection after training:

$$p_j = \arg\min_{S_i} \|S_i - p_j\|_2^2 \quad \forall\, j \in [m] \quad (5)$$

`FP` closes the faithfulness gap by forcing each

PNLM	Base LLM	SST2	SST5	QNLI	RTE	HoC	AVg Gain/Loss (per PNLM)
Proto-lm + (FA) / Proto-lm	BERT-base	**90.8/90.0**	50.3/53.2	**85.9/85.8**	67.5/68.6	**94.0/84.3**	**+0.72**
	RoBERTa-large	**92.4/92.3**	47.4/52.9	**87.9/87.8**	**62.8/62.1**	–	
	BART-large	92.2/92.4	55.8/54.8	**89.0/89.0**	74.7/75.1	–	
	ELECTRA	**93.3/90.1**	**50.1/47.3**	**91.0/90.1**	**74.8/74.7**	**54.3/49.9***	
	Llama-2-7b	**93.6/91.2**	47.7/48.3	**92.2/91.3**	**80.3/80.0**	36.7/36.9*	
ProtoTex + (FA) / ProtoTex	BERT-base	**91.1/90.8**	48.7/50.5	82.0/90.3	53.0/66.7	89.3/95.3	-2.94
	RoBERTa-large	92.3/94.0	**46.7/44.4**	83.5/92.3	52.7/60.0	–	
	BART-large	94.6/95.9	51.3/52.2	88.9/93.2	63.1/77.6	–	
	ELECTRA	**95.0/94.8**	47.6/49.2	**92.3/92.2**	72.2/73.2	52.1/51.0*	
	Llama-2-7b	**94.3/94.3**	43.0/44.3	93.5/93.6	**81.3/81.0**	32.5/36.0*	
ProtoPatient + (FA) / ProtoPatient	BERT-base	**91.7/91.7**	**52.1/51.0**	**82.8/82.7**	**67.5/64.2**	**94.5/93.0**	**+0.64**
	RoBERTa-large	**91.5/91.4**	53.7/54.0	**84.2/84.2**	**76.5/72.9**	–	
	BART-large	**90.8/90.8**	49.7/53.8	**87.4/87.4**	**77.6/73.3**	–	
Avg Gain/Loss (per task):		**+0.30**	-0.91	-1.48	-1.95	**+1.47**	

Table 1: Performance of FA on NLP tasks and across PNLM architectures. **Bolded** numbers indicate scenarios where the FA-applied PNLM achieved performance that is higher than or equivalent to the regular PNLM. Overall, we observe that FA-applied models experience only minor decreases in accuracy. Results with * are obtained from PNLMs with a base LM that was *not* finetuned on the task (HoC), hence relatively lower performance. We note that, even in these cases, applying FA did not significantly degrade the performance of the PNLM.

prototype to become an encoding from the training dataset. Figure. 5 provides an illustration of this process. After FP, textual representations now directly reflect the parameters contributing to modeling reasoning. Thus, FP not only allows us to map a continuous prototype tensor onto discrete words for human comprehension but also align PNLMs' prototype-based explanations completely with model reasoning.

4 Effect of FA on Downstream Tasks

Because FA makes PNLM architectures mechanically faithful, it is important to examine its impact on model performance. Specifically, we conduct experiments to compare the performance of PNLMs without FA against FA-applied PNLMs. In Table 1, we verify that FA does not degrade accuracy on downstream tasks on a wide range of PNLM architectures (§C.2), base LLM encoders and NLP tasks (§C.1) . Our experimental results indicate that FA-applied models performs well overall, with only minor reductions in performance across NLP tasks. To further demonstrate the robustness of FA, we also conduct ablation studies in §E. We believe the robust performance of FA shows its promise as a framework for ensuring faithfulness for prototypical networks while maintaining performance.

5 Conclusion

In this work, we build on foundational ideas in XAI, interpretability, and faithfulness to identify two key shortcomings of existing PNLMs that cause their explanations to be unfaithful: 1) the selection of incorrect prototypes for explanation, and 2) the misrepresentation of prototypes in model reasoning. To address these issues, we introduce a faithfulness-ensuring framework, FA, and validate its robustness through extensive experiments. We believe our contribution bridges a crucial gap in the current understanding of faithfulness in the context of prototypical models. We hope future prototypical networks can leverage our framework to prevent unfaithful explanations.

6 Limitations

We outline several limitations of our work below:

- Despite the improvements made to the faithfulness of PNLMs through FA, they still remain dependent on an underlying language model to convert text into a semantic space. Consequently, the interpretability of PNLMs is still constrained by the interpretability of the foundational language model.

- Additionally, while the case-based reasoning of FA-applied PNLM explanations is faithful, the cases used for explanation are restricted to examples from the training set. This limit on the *expressiveness* of the interpretability of PNLMs is an area that warrants further research and exploration.

- Moreover, a significant issue affecting the faithfulness of current PNLMs is that LLM embeddings cannot be directly translated back into discrete words. Making progress in this research area will not only resolve a key unfaithfulness issue addressed by our framework

but will also greatly benefit the field of interpretability as a whole.

7 Ethics Statement

All datasets used in our experiments are publicly available datasets. None of the experimental data directly utilized and/or collected in our experiments is related to identifiable individuals, ensuring that all data remains anonymous by nature. We conducted our experiments with a focus on transparency and reproducibility. Any potential limitations of our experiments are thoroughly analyzed and discussed.

8 Acknowledgements

This research was supported in part by grants from the US National Library of Medicine (R01LM012837 & R01LM013833), the US National Cancer Institute (R01CA249758), the US National Science Foundation (NSF Award 2242072), and the John Templeton Foundation. The authors extend their sincerest gratitude to Andrew Brennan, Yiren Jian, Alex DeJournett, Alan Sun and Jeffrey Jiang for their help and support during the research process. Additionally, the authors would also like to thank Akira Toriyama, who passed away during the preparation of this work. Toriyama's art touched countless lives, including our own, inspiring many to face life's challenges with courage and determination.

References

Abien Fred Agarap. 2018. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. 2020. A diagnostic study of explainability techniques for text classification. *arXiv preprint arXiv:2009.13295*.

Simon Baker, Ilona Silins, Yufan Guo, Imran Ali, Johan Högberg, Ulla Stenius, and Anna Korhonen. 2015. Automatic semantic classification of scientific literature according to the hallmarks of cancer. *Bioinformatics*, 32(3):432–440.

Gagan Bansal, Tongshuang Wu, Joyce Zhou, Raymond Fok, Besmira Nushi, Ece Kamar, Marco Tulio Ribeiro, and Daniel Weld. 2021. Does the whole exceed its parts? the effect of ai explanations on complementary team performance. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA. Association for Computing Machinery.

Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. 2015. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1721–1730.

Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. 2019. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8):832.

Aaron Chan, Maziar Sanjabi, Lambert Mathias, Liang Tan, Shaoliang Nie, Xiaochang Peng, Xiang Ren, and Hamed Firooz. 2022. Unirex: A unified learning framework for language model rationale extraction. In *International Conference on Machine Learning*.

Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. 2019. This looks like that: deep learning for interpretable image recognition. *Advances in neural information processing systems*, 32.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, pages 177–190. Springer.

Anubrata Das, Chitrank Gupta, Venelin Kovatchev, Matthew Lease, and Junyi Jessy Li. 2022. Prototex: Explaining model decisions with prototype tensors. *arXiv preprint arXiv:2204.05426*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. 2020. Eraser: A benchmark to evaluate rationalized nlp models.

Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Radwa ElShawi, Youssef Sherif, Mouaz Al-Mallah, and Sherif Sakr. 2021. Interpretability in healthcare: A comparative study of local machine learning interpretability techniques. *Computational Intelligence*, 37(4):1633–1650.

Felix Friedrich, Patrick Schramowski, Christopher Tauchmann, and Kristian Kersting. 2022. Interactively providing explanations for transformer language models.

Tianyu Gao, Xu Han, Zhiyuan Liu, and Maosong Sun. 2019. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6407–6414.

Peter Hase, Chaofan Chen, Oscar Li, and Cynthia Rudin. 2019. Interpretable image recognition with hierarchical prototypes. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 7, pages 32–40.

Qihan Huang, Mengqi Xue, Wenqi Huang, Haofei Zhang, Jie Song, Yongcheng Jing, and Mingli Song. 2023. Evaluation and improvement of interpretability for self-explainable part-prototype networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2011–2020.

Alon Jacovi and Yoav Goldberg. 2020. Towards faithfully interpretable nlp systems: How should we define and evaluate faithfulness? *arXiv preprint arXiv:2004.03685*.

Zhong Ji, Xingliang Chai, Yunlong Yu, Yanwei Pang, and Zhongfei Zhang. 2020. Improved prototypical networks for few-shot learning. *Pattern Recognition Letters*, 140:81–87.

José Jiménez-Luna, Francesca Grisoni, and Gisbert Schneider. 2020. Drug discovery with explainable artificial intelligence. *Nature Machine Intelligence*, 2(10):573–584.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Zachary C Lipton. 2018. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Scott Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions.

Qing Lyu, Marianna Apidianaki, and Chris Callison-Burch. 2024. Towards Faithful Model Explanation in NLP: A Survey. *Computational Linguistics*, pages 1–67.

Chiyu Ma, Brandon Zhao, Chaofan Chen, and Cynthia Rudin. 2023. This looks like those: Illuminating prototypical concepts using multiple visualizations.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Ibrahim Naji. 2012. TSATC: Twitter Sentiment Analysis Training Corpus. In *thinknook*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016a. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016b. "why should i trust you?": Explaining the predictions of any classifier.

Cynthia Rudin. 2018. Please stop explaining black box models for high stakes decisions. *Stat*, 1050(26):457.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2019. Learning important features through propagating activation differences.

Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical networks for few-shot learning.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Shengli Sun, Qingfeng Sun, Kevin Zhou, and Tengchao Lv. 2019. Hierarchical attention prototypical networks for few-shot text classification. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 476–485.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Betty Van Aken, Jens-Michalis Papaioannou, Marcel G Naik, Georgios Eleftheriadis, Wolfgang Nejdl, Felix A Gers, and Alexander Löser. 2022. This patient looks like that patient: Prototypical networks for interpretable diagnosis prediction from clinical text. *arXiv preprint arXiv:2210.08500*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Sean Xie, Soroush Vosoughi, and Saeed Hassanpour. 2023. Proto-lm: A prototypical network-based framework for built-in interpretability in large language models. *arXiv preprint arXiv:2311.01732*.

Sean Xie, Soroush Vosoughi, and Saeed Hassanpour. 2024. Ivra: A framework to enhance attention-based explanations for language models with interpretability-driven training. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 431–451.

A Additional Related Works and Background

A.1 Prototypical Networks and Interpretability

Prototypical networks (Snell et al., 2017) originated as a type of few-shot learning model designed to classify new examples by comparing them to class prototypes obtained via averaging instances of each class in an embedding space (Ji et al., 2020). The transparency of prototypical models and their intuitive case-based reasoning process led to their adoption to interpret deep neural networks (Gao et al., 2019; Sun et al., 2019; Chen et al., 2019; Hase et al., 2019). Continued development in XAI has revealed the multi-facetedness of the field of interpretability (Doshi-Velez and Kim, 2017; Lipton, 2018). There now exists various different aspects in interpretability such as "plausibility" (how convincing explanations are to humans (Jacovi and Goldberg, 2020; ElShawi et al., 2021; Chan et al., 2022)), or "consistency" (similarity between explanations for similar inputs) (Carvalho et al., 2019; Atanasova et al., 2020; Xie et al., 2024). In this work, we focus on the criterion of ***faithfulness***, which is qualitatively described as how accurately a model's reasoning process is reflected in its explanations (Ribeiro et al., 2016a; Jacovi and Goldberg, 2020; Lyu et al., 2024). Recent research has shown that ***faithfulness*** has emerged as a crucial interpretability criterion since unfaithful explanations pose risks in high-stakes areas. (Caruana et al., 2015; Rudin, 2018; Jiménez-Luna et al., 2020).

A.2 Existing Techniques for Measuring Faithfulness and Pitfalls

A plethora of techniques have emerged to evaluate the faithfulness of model explanations. These techniques include Axiomatic Evaluation (prove unfaithfulness by showing necessary faithfulness assumptions are violated), Simulatability (using model explainations to predict model outputs), Perturbation Methods (stability of explanation under input perturbation), and others. (Lyu et al., 2024). **Unfortunately, no definitive measure of faithfulness exists in the XAI community** and evaluation metrics are often not directly comparable with each other and yield inconsistent results, making it difficult to objectively assess progress. (Lyu et al., 2024). Regarding the interpretability of prototypical networks, works that quantify model faithfulness have done so on a token/pixel level on the input text/image[5]. For example, Huang et al. 2023 defines Inconsistency and Instability to understand variations in the pixel attribution map on top of the test-time example, and Van Aken et al. 2022 applies the benchmark established by Atanasova et al. 2020 to quantify the faithfulness of highlighted input tokens compared to post-hoc methods. While these metrics are significant in their own right, they do not capture the faithfulness of explanations generated on the example level from the prototype itself (training image patches in CV, training text examples in NLP). Specifically, they ignore the question of whether or not the prototype was accurately *identified* or *represented* in the first place. On the whole, existing faithfulness metrics fall short in measuring the faithfulness of prototypical networks effectively. We will show in §2 that prototypical networks, even those asserting to be faithful by design suffer from underlying faithfulness issues.

A.3 Axiomatic Evaluation of Faithfulness in Prototypical Networks

On gauging the faithfulness of prototypical networks, the seminal work of Chen et al. 2019 took an axiomatic approach in their evaluation strategy.

[5]The work of Xie et al. (2023) attempted to extend existing faithfulness metrics to the example level , but we believe these metrics were applied incorrectly. We delve into details in §F

Chen et al. 2019 emphasized that their architecture is faithful by design by establishing a direct connection between their generated explanations and the computations driving predictions. When adapting prototypical networks to NLP, similar mechanical arguments were made to assert model faithfulness, but **without** the necessary architectural design to support these claims. In this work, we aim to expose the faithfulness shortcomings in current PNLM architectures by leveraging the same axiomatic analysis employed by (Chen et al., 2019). Specifically, recent works (Das et al., 2022; Van Aken et al., 2022; Xie et al., 2023) on PNLM have failed to align prototypes with latent training examples and has neglected prototype influence on prediction when generating explanations. For further discussion on our evaluation of faithfulness, see §F.

B Proto-lm Faux Faithfulness

As noted in the related works section, Xie et al. 2023 is the only work to our knowledge that claim's PNLM's are faithful on the example level. In this section, we outline why we believe the faithfulness experiments in Xie et al. 2023 were faulty and resulted in a false sense of model faithfulness.

Xie et al. 2023 extends the perturbation based metrics introduced in DeYoung et al. 2020, Comprehensiveness and Sufficiency, to claim the faithfulness of their PNLMs explanations. In the original work, DeYoung et al. 2020 defined these metrics with respect to a model's explanations (or "rationals", as they refereed to them) which are highlighted input tokens. Let x_i be the original input text sequence, r_i be the models explanation, and $m(x_i)_j$ be the original prediction provided by a model m for class j. The Comprehensiveness and Sufficiency of the models explanation r_i are numerically defined by:

$$\text{Comprehensiveness} = m(x_i)_j - m(x_i/r_i)_j \quad (6)$$

$$\text{Sufficiency} = m(x_i)_j - m(r_i)_j \quad (7)$$

See Figure 6 for an illustration of how Comprehensiveness and Sufficiency are computed as defined in DeYoung et al. 2020. Intuitively, these metrics are measuring how the model's performance changes when the model's explanation is taken away and when only the explanation is available to the model, allowing us to quantify how much the model is leveraging its explanations for its predictions. For more detail, see the original work. (DeYoung et al., 2020)

Xie et al. 2023 extends these metrics to prototypical networks by treating PNLM prototypes as rationales, which were previously considered to be highlighted tokens from the input text r_i. Comprehensiveness and Sufficiency were then computed based on analyzing the model's prediction change when removing/retaining prototypes from the last layer during inference following the same Equations as 6 and 7.

As demonstrated in this work, without FA, prototypes in PNLMs (like Proto-lm) are not utilized for generating model explanations; instead, the nearest training encoding to the prototype is used. The explanations for existing PNLMs without FA is the text associated with the nearest training encoding S_i to each prototype, yet S_i appears nowhere in Xie et al. 2023's faithfulness experiments.

We illustrate the faithfulness gap between the tensors PNLMs reason with and the tensors PNLMs generate explanations with in Figure 4. From the Faithfulness Gap, its clear that the experiments conducted in Xie et al. 2023 did not incorporate the PNLM's explanation in the experimentation at all, leading Sufficiency and Comprehensiveness to fail to quantify the faithfulness of the PNLM's explanations and ultimately lead to a false sense of the model's faithfulness.

We note that if FA was applied to Proto-lm, aligning the prototypes with latent training examples, the efficacy of Comprehensiveness and Sufficiency measuring the faithfulness of the model's explanations would be restored since then the prototypes would be faithfully interpretable as the model's explanations. We excluded this experiment comparing Comprehensiveness and Sufficiency before and after applying FA to a PNLM since the results obtained in the former case are meaningless- without FA, the PNLM explanations are derived from S_i, which does not participate in the original experimentation from Xie et al. 2023 at all.

C Experimental Details

C.1 Datasets and Tasks

The datasets and tasks in our experiments in Table. 1 include sentiment classification (Socher et al., 2013) , natural language inference(Wang et al., 2018), entailment recognition (Dagan et al., 2005),

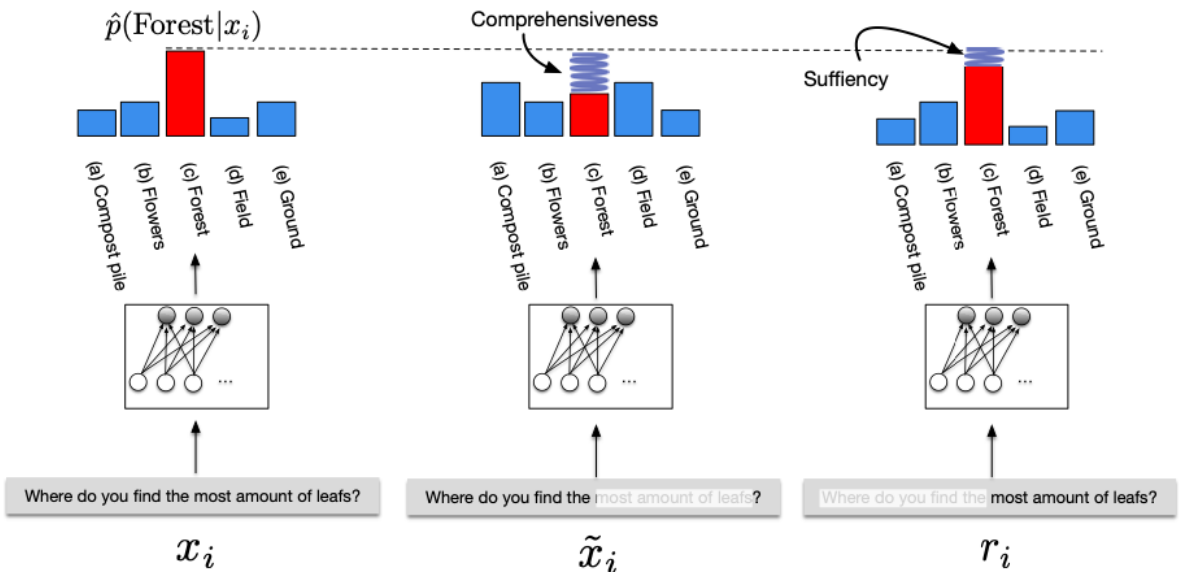

Figure 6: An illustration of how Comprehensiveness and Sufficiency are computed based on an input text sequence x_i and the models explanation, which is a highlighted sequence of input tokens, r_i. Figure from (DeYoung et al., 2020)

and cancer type classification (Baker et al., 2015). We use BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), BART (Lewis et al., 2019), ELECTRA (Clark et al., 2020) and Llama-2 (Touvron et al., 2023) as base models upon which we implement PNLM architectures described in the following section.

C.2 PNLM Architecture Description

In this section we provide brief descriptions of the following three PNLMs architectures pertinent to our experiments in §4.

C.2.1 Proto-lm

Let P denote the set of its prototypes, `proto-lm` (Xie et al., 2023) utilizes $|P| = N$ prototypes that are evenly distributed between C classes i.e. each prototype is assigned a class $c \in C$, with each class c having $\frac{N}{C}$ prototypes. To obtain $f_\theta(x_i)$, `proto-lm` leverages a token-attention layer that takes $x_i \in \mathbb{R}^{L \times d}$ to $f_\theta(x_i) \in \mathbb{R}^H$. Similarities measures M in `protolm` are obtained via taking the inverse L2 distance between the prototypical encoding $f_\theta(x_i)$ and $p \in P$. During training `proto-lm` seeks to minimize a cohesion loss term $\mathscr{L}_{coh}$ and maximize a separation loss term $\mathscr{L}_{sep}$. Let K be an integer hyperparameter, p_j be a prototype, and P_{y_i} represent all prototypes in P that belong to class y_i, the cohesion and separation loss terms in `proto-lm` are defined respectively:

$$\mathscr{L}_{coh} = \frac{1}{K} \cdot \sum_{\forall j : p_j \in P_{y_i}} \max_K \|S_i - p_j\|_2^2 \quad (8)$$

$$\mathscr{L}_{sep} = -\frac{1}{K} \cdot \sum_{\forall j : p_j \notin P_{y_i}} \min_K \|S_i - p_j\|_2^2 \quad (9)$$

During training, `proto-lm` incorporates the above two loss terms along with standard cross entropy loss into its training objective. The architecture of `proto-lm` utilizes class connections with preset directions. Specifically, let j_c indicate the class assigned to prototype j, all class connections in W_f between class c' and prototypes that have $j_c = c'$ are positive and class connections between class c' and prototypes with class $j_c \in \{C \setminus c'\}$ are negative. i.e.

$$W_{\{j,j_c\}} \geq 0 \;\; \forall j \;\; \text{s.t.} \;\; j_c = c' \quad (10)$$

$$W_{\{j,j_c\}} \leq 0 \;\; \forall j \;\; \text{s.t.} \;\; j_c \neq c' \quad (11)$$

When applying `FR` to `proto-lm`, we set all class connections between prototype j and classes $c \in \{C \setminus c'\}$ to 0. Formally, we perform the following update after training, let $jc \in C$:

$$W_{\{j,j_c\}} \leftarrow 0 \;\; \forall j \;\; \text{s.t.} \;\; j_c \neq c' \quad (12)$$

When applying `FP` on `proto-lm`, all encodings projected are restricted to encodings from the same class as each prototype.

PNLM	Base LLM	SST2	SST5	QNLI	RTE	HoC
Proto-lm + (FA) / Proto-lm	BERT-base	3e-6/64/1000	3e-6/16/200	3e-5/16/800	3e-5/16/200	3e-6/32/1100
	RoBERTa-large	3e-5/64/1000	3e-5/64/200	3e-5/16/800	3e-6/16/200	–
	BART-large	3e-5/64/1000	3e-5/64/200	3e-5/16/800	3e-5/16/200	–
	ELECTRA	3e-6/32/1000	3e-6/32/200	3e-5/16/800	3e-6/16/200	3e-6/32/200
	Llama-2-7b	3e-6/16/400	3e-6/16/200	3e-6/16/400	3e-6/16/200	3e-6/16/200
ProtoTex + (FA) / ProtoTex	BERT-base	3e-5/64/200	3e-5/64/200	3e-5/16/200	3e-5/16/200	3e-7/64/220
	RoBERTa-large	3e-5/64/200	3e-5/64/200	3e-5/16/200	3e-6/16/200	–
	BART-large	3e-5/64/200	3e-5/64/200	3e-5/16/200	3e-5/16/200	–
	ELECTRA	3e-6/32/1000	3e-6/32/200	3e-5/16/800	3e-6/16/200	3e-6/32/200
	Llama-2-7b	3e-6/16/400	3e-6/16/200	3e-6/16/400	3e-6/16/200	3e-6/16/200
ProtoPatient + (FA)	BERT-base	3e-5/64/5	3e-5/64/5	3e-5/64/2	3e-5/64/2	3e-5/32/11
	RoBERTa-large	3e-5/64/5	3e-5/64/5	3e-5/64/2	3e-5/64/2	–
	BART-large	3e-5/64/5	3e-6/64/5	3e-6/64/2	3e-5/64/2	–

Table 2: Hyperparameters of models whose accuracies are reported in Table. 1. The numbers reported in each cell correspond to **learning rate/batch size/number of prototypes** For `proto-lm`, $\lambda = 0.5, \lambda_1 = 0.25, \lambda_2 = 0.25$. For `prototex`, $\lambda_1 = \lambda_2 = \frac{1}{3}$.

C.2.2 ProtoTex

`ProtoTex` (Das et al., 2022) is a PNLM that does not utilize class-assigned prototypes. The N prototypes in `ProtoTex` and their class connections in W_f are freely learned. In addition, calculation of prototypical encodings in `ProtoTex` do not involve token attention as in the case of `proto-lm` and `ProtoPatient`. In our implementation, we chose to use prototypes of dimension $\mathbb{R}^H$ because it led to the highest performance across tasks. Since the output of base LLMs is in dimension $\mathbb{R}^{L \times H}$, we took the mean along the token dimension when calculating encodings $f_\theta(x_i)$.

Similarity measures M in `ProtoTex` are the raw L2 distances between prototypes and $f_\theta(x_i)$. The training objective of `ProtoTex` is similar to `proto-lm`. Specifically, during training, in addition to minimizing cross entropy, `ProtoTex` seeks to minimize distance between prototypes and at least one encoded input.

$$\mathscr{L}_{p1} = \frac{1}{M} \cdot \sum_{j=1}^{M} \min_{i=1,n} \|p_j - S_i\|_2^2 \qquad (13)$$

as well as minimizng distance between encoded input and at least one prototype

$$\mathscr{L}_{p2} = \frac{1}{n} \cdot \sum_{i=1}^{n} \min_{j=1,M} \|S_i - p_j\|_2^2 \qquad (14)$$

Because there are no predetermined negative class connections in W_f, when applying `FR` we remove negative reasoning from `ProtoTex` by applying the ReLU (Agarap, 2018) activation function on weights in W_f during each forward pass i.e.

$$W_{jc} = \max\{W_{jc}, 0\} \qquad \forall j \in [m], \ \ \forall c \in C \qquad (15)$$

C.2.3 ProtoTex

`ProtoTex` (Das et al., 2022) is a PNLM that does not utilize class-assigned prototypes. The N prototypes in `ProtoTex` and their class connections in W_f are freely learned. In addition, calculation of prototypical encodings in `ProtoTex` do not involve token attention as in the case of `proto-lm` and `ProtoPatient`. In our implementation, we chose to use prototypes of dimension $\mathbb{R}^H$ because it led to the highest performance across tasks. Since the output of base LLMs is in dimension $\mathbb{R}^{L \times H}$, we took the mean along the token dimension when calculating encodings $f_\theta(x_i)$.

C.3 ProtoPatient

The PNLM of `ProtoPatient` (Van Aken et al., 2022) utilizes 1 prototype per class $c \in C$. Prototypical encoding calculation utilize both a dimension reduction layer as well as a label-wise attention layer. For each input x_i, the dimension reduction layer reduces the output dimension H of the base LLM to $\frac{H}{3}$ and the label-wise attention collapses the L dimension to produce $f_\theta(x_i) \in \mathbb{R}^{\frac{H}{3}}$. Similarity measures M in `ProtoPatient` are the *negative* of raw L2 distance measures between $f_\theta(x_i)$ and prototypes. In addition, `ProtoPatient` does not utilize class connections and instead produces output logits via taking the sigmoid of M.

Similarity measures M in ProtoPatient are the *negative* of raw L2 distance measures between $f_\theta(x_i)$ and prototypes. In addition, ProtoPatient does not utilize class connections and instead produces output logits via taking the sigmoid of M. The training objective of ProtoPatient consists of a single cross entropy term defined as follows

$$\mathscr{L} = \sum_{p \in P} \sum_{c \in C} \mathsf{BCE}(\hat{y}_{pc}, y_{pc}) \tag{16}$$

where $\hat{y}_{pc}$ is the output of prototype p for class c and $y_{pc} \in 0, 1$ is the ground truth. We note here that ProtoPatient initializes its prototypes with the average encoding examples from the prototypes' respective classes before training for the BCE objective in eq. 16.

We empirically found that ProtoPatient has a smaller faithfulness gap than Proto-lm and ProtoTEx, and we hypothesize this is due to (1) the objective function not directly optimizing for prototype clustering/separation and (2) the prototype initialization to latent training encoding of examples from the same class as the prototype. While ProtoPatient is still structurally vulnerable to faithfulness issues since their explanations (encoded training examples) don't impact predictions, it is our belief that Proto-Patient is more faithful than Proto-lm and ProtoTEx since Proto-Patient has a smaller faithfulness gap, i.e, the predictions and explanations are more aligned in the latent space.

D Hyperparameters and Compute Resources

Our compute resources consist of 4× RTX 6000, 4× RTX 4500 and 4× RTX 3090. In Table. 2 below, we describe the hyperparameter setup (learning rate, batch size and the number of prototypes) we used to obtain results in Table 1. We ran all models for a maximum of 15 epochs and report performance from the best iteration of the model during training. The running time of no individual experiment in Table. 2 exceeded 5 hours.

E Additional Experiments

E.1 Size of Projection Dataset

Although we have shown in earlier sections that FA can achieve competitive results on a wide range of tasks, it is important to consider whether the amount of training data used during projection impacts performance. Specifically, what happens if we use only a *subset* of the training data to faithfully project prototypes during FA? To investigate this, we conduct experiments by applying FA with varying sizes of training datasets. Specifically, we trained 8 proto-lm models on SST2, SST5, and HoC ($C = 2$, $C = 5$ and $C = 11$, respectively) with prototype counts that are multiples of C, totaling $|\{1, 2, 5, 10, 20, 40, 80, 100\} \times \{2, 5, 11\}| = 24$ models. We then apply FA to each model with different amount of training data and calculate the change in accuracy. In Figure 7, we show the gain/loss in accuracy averaged across the three tasks. We find that, when the number of prototypes in the PNLM is large but the training data for projection is limited, performance generally drops significantly. We reason that this is because prototypical features are derived from a shared, small sample, thus reducing prototype uniqueness (Das et al., 2022). On the other hand, this issue does not arise with large amount of projection data and fewer prototypes, as the prototypes can leverage features from the most apt sample encodings. Interestingly, when both the number of prototypes and training data size are small, we observe notable performance improvements, as FA helps the few prototypes capture the most important features. Overall, our experiments show that FA-applied models maintain performance with sufficient training data, but selecting the right number of prototypes is crucial if one wishes to reduce the amount of training data used during FA.

E.2 Generalizability of FA

Given that the FP component of FA restricts prototypical features to encodings of samples within a specific dataset, it is reasonable to expect that FA-applied PNLMs is prone to overfitting. To explore the generalizability of faithfully-aligned prototypes, we apply FA to a trained PNLM on the SST2 dataset and observe how well it performs when classifying examples from two *other* sentiment classification datasets: IMDb movie reviews (Maas et al., 2011) and the Twitter Sentiment Analysis Training Corpus (Naji, 2012). In Fig. 8, we present the zero-shot accuracy (on IMDb and TSATC) of PNLMs that were trained on SST2 and were also applied FA using the SST2 dataset in red. For comparison, we also show the zero-shot accuracy of PNLMs that were trained on SST2 but *without* applying FA in yellow. We find that, in zero-shot settings, FA-applied PNLMs are able to outperform

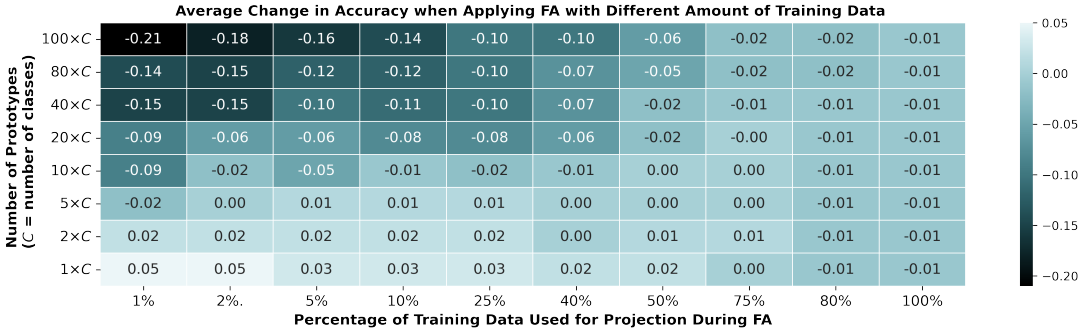

Figure 7: Change in accuracy of `Proto-lm` models after applying FA with different amount of training data. Results obtained are the average across SST2, SST5 and HoC. `Proto-lm` models have prototypes counts that are multiples of C which is the number of classes in each dataset, with $C = 2$ for SST2, $C = 5$ for SST5 and $C = 11$ for HoC.

Figure 8: Model accuracy on IMDb and TSATC. red bars represent the accuracy of a zero-shot PNLM with FA applied using examples from SST2. yellow bars represent the zero-shot accuracy of the base PNLM but without FA applied. Both models were trained on SST2.

PNLMs without FA across architectures with different number of prototypes, indicating that FA-applied PNLMs generalize well to unseen data. Furthermore, as the number of prototypes increases, the performance difference between the models with and without FA widens. We believe this is because a larger set of prototypes provides greater representational power during the FA process, resulting in a robust model that excels on various datasets.

E.3 Comparison with Case-based-reasoning Models

Another notable effect of FA projecting the learned prototypes onto training encodings is the resulting model architecture's similarity to K-Nearest-Neighbor models. We therefore conduct experiments comparing the performances of FA and KNN-based approaches under different settings. First, we build a KNN model using the encodings of a *base language model* (i.e. BERT). We then build a KNN model using the encodings (in prototypical space) of a non-FA PNLM (i.e. an instance of `proto-lm`). Finally, we build a KNN model using the encodings of a FA-applied PNLM. We compare the performance of these three case-based-reasoning models against the PNLM as well as the FA-applied PNLM on the 11-class cancer classification dataset (HoC). We present our results for PNLMs of varying prototypes in Fig. 9. We observe that with a high number of prototypes, FA allows PNLMs to outperform KNN models. Additionally, KNN-based models suffer in terms of performance when K becomes too large, unless aided by FA. This is likely because the HoC dataset has a limited number of examples per class. A large K parameter forces KNN models to use neighbors from other classes, leading to a drop in performance. FA-applied PNLMs do not have this issue because the class connections in W_f ensure that examples from the correct class receive more weight.

F Frequently Asked Questions

Where are the quantitative metrics justifying (A) that the identified PNLMs are unfaithful and (B) that FA improves these models faithfulness?

(A) We reiterate that no existing faithfulness metrics are applicable to quantify the faithfulness of

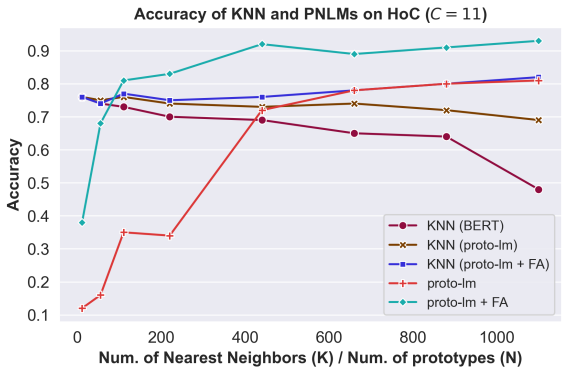

Figure 9: Accuracy of KNN-based models and FA-applied PNLMs.

Prototypical Networks on the example level. If such metrics existed, the faithfulness issues identified in this work would have been immediately identified and resolved by previous work. We believe that developing such a metric is an important area of future research and that the lack of said metric played a role in the prevalence of existing PNLM unfaithfulness.

Our claims regarding the unfaithfulness of the identified PNLMs in §2 is rooted in prototypical faithfulness principles established in (Chen et al., 2019), such as the principle that latent encoding need to be equal to prototypes in the prototypical layer, as well as the qualitative description of faithfulness, which states that for an explanation to be faithful, it must align with the reasoning process of the model. The case-based reasoning architecture of Prototypical Networks enables us to explicitly define the reasoning process of the model based on the similarity computation in P and the class connections in W_f. This framing allows us to transparently view inference as a weighted similarity (using W_f) between the encoded test-time example, $f_\theta(x)$, and prototypes $P = \{p_j\}_{j=1}^m$. Explanations in (Xie et al., 2023; Das et al., 2022) are chosen solely based on the similarity in P and do not consider class connections in W_f, resulting in an overt diversion with the reasoning process of the Prototypical Network and thus unfaithful explanations.

Faithful Alignment resolves both of these issues: prototypes are restored as latent training examples, adhering to (Chen et al., 2019), and prototypes are selected for explanation based on **all** the computation that effects prediction instead of just computation in P, adhering to (Jacovi and Goldberg, 2020).

Why does FA not work as well on ProtoTEx as it does on Proto-lm and Proto-Patient?

We hypothesize this is because ProtoTEx does not assign a specific class to the prototypes before learning, unlike how it's done in Proto-lm and ProtoPatient. Since projection is done to the nearest encoding in the training example, we hypothesize that there is greater risk of the prototype being projected onto a less representative training example when classes are not preassigned to prototypes. For example, if the prototype before projection was of class 3 but the training example nearest to it was of class 4, then faithful projection could more significantly affect the model's reasoning.

We remark that the performance of FA with ProtoTEx still remains competitive across a diverse set of tasks and the performance was stronger when applied using more recent models such as ELECTRA or Llama.

Why are there no experimentation with FA applied to CV models/datasets?

In short, we didn't experiment with any CV models/datasets because we are not aware of any unfaithful Prototypical Networks in CV. We experimented with the three PNLMs we identified as having architectural flaws that lead to unfaithful model explanations to demonstrate we can resolve the previously unknown faithfulness issues without sacrificing model performance. We reiterate that FA is a general framework that can ensure faithful explanations for Prototypical Networks in both CV and NLP. The techniques which comprise FA, namely FP and FR, act on the prototypical layer P and final linear layer W_f, which are present in Prototypical Networks independent of input modality.