

000 001 002 003 004 005 006 007 008 009 010 FINDING BETTER PROTOTYPES FOR INTERPRETABLE TEXT CLASSIFIERS WITH LLM OPTIMIZATION

005 **Anonymous authors**

006 Paper under double-blind review

009 ABSTRACT

011 Prototype neural networks are the most popular form of interpretable-by-design
 012 classifiers in machine learning. Within this field, prototypes are typically learned
 013 as black-box vectors, and then projected onto the nearest example from the train-
 014 ing data for visualization and inference purposes. This improves interpretability
 015 because we can understand the logic behind predictions based on the similarity
 016 between the input instance and the nearest prototype in the network. However,
 017 because these prototypes are real training instances there are at least two major
 018 issues with this approach. Firstly, as the projected prototypes do not represent the
 019 learned “black-box” vectors which were optimized for accuracy, there is typically a
 020 performance drop off. Secondly, because the prototypes are real training instances,
 021 they are usually overly specific and full of spurious or irrelevant details, making
 022 them difficult to interpret readily. In this study, we address this problem by using
 023 large-language models (LLMs) as a tool for optimization to find better prototypes
 024 for the network. Across a series of experiments, we find that our method produces
 025 prototypes which sacrifice less performance and are more intelligible compared
 026 to baselines which project. Previously, it was not possible to visualize a learned
 027 prototype, because methods were constrained to projection using actual training
 028 data, but our approach suggests a possible path to overcome this limitation.

029 1 INTRODUCTION

031 The problem of interpretability in machine learning (ML) is perhaps the core issue preventing
 032 widespread trust of the technology globally (Amodei, 2024). If we are ever to actually appropriately
 033 trust ML in sensitive domains, more fundamental progress is urgently needed to address issues
 034 of alignment, safety, and reliability (Sharkey et al., 2025). Out of this urgency, researchers have
 035 proposed various solutions roughly divided into post-hoc justifications for predictions (Ribeiro et al.,
 036 2016), and interpretable-by-design architectures (Chen et al., 2019). In the latter category, prototype
 037 neural networks are the most popular architectural design methodology (Li et al., 2018), as they have
 038 strong justification and roots in psychological theory (Rosch, 1973), as well as great practical benefit
 039 in ML (Chen et al., 2023b; Cunningham et al., 2003; Kenny et al., 2023). However, these methods
 040 have struggled with perhaps the most fundamental problem in their design methodology, which is
 041 how to actually visualize the learned prototypes the network produces.

042 Historically, researchers have approached the problem by learning a set of black-box prototype vectors
 043 in the penultimate layer of the neural network, and then once training is complete, projecting these
 044 onto the closest real examples in the training set (Chen et al., 2019; Ming et al., 2019; Ma et al., 2024).
 045 Although this approach has proven widely successful, it poses at least two severe problems. Firstly,
 046 although not always, this approach has historically resulted in a drop in model performance, which
 047 initially hampered their adoption (Chen et al., 2019; Ma et al., 2024), because training examples
 048 cannot realistically align with the “perfect” learned black-box prototype. Secondly, and perhaps more
 049 importantly, this clashes with classical prototype theory outlined by Rosch (1973), which states that
 050 prototypes should be a more abstract representation of a class, which generalizes across instances
 051 and is not overly specific. However, especially in textual domains, as the training data are often long
 052 sequences of text with many irrelevant features, this is often not possible to achieve. To address these
 053 issues, we propose to use large-language models (LLMs) as an optimization tool to find textual inputs
 that not only map better to the learned prototypes, but also better align with prototype theory by being
 sparse featural representations of the class, as we illustrate in Figure 1.

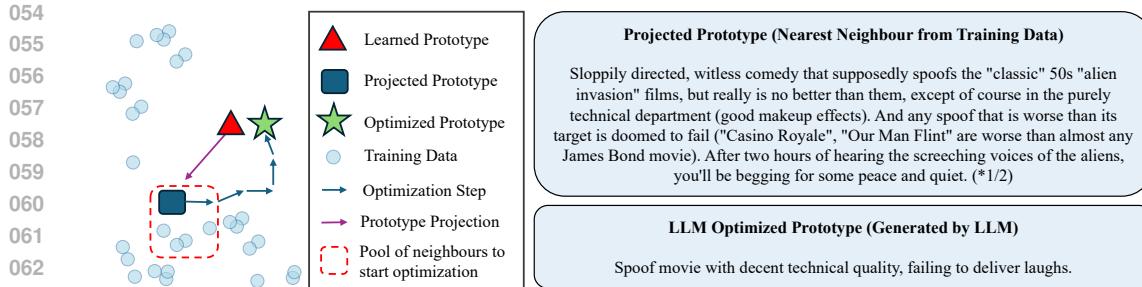


Figure 1: Our Proposal: Typically in the prototype literature researchers have projected prototypes onto the nearest training example to visualize them and run inference. However, this can result in overly long, complex, and cumbersome prototypes. In contrast, we propose to begin an optimization process from the nearest neighbours pool, which uses LLMs as optimization tools to iteratively refine a simpler, more intelligible version of the prototype, which often better aligns with the learned prototype vector. In this real example from our tests, the projected prototype discusses overly specific features such as “*Casino Royal*” and “*James Bond*”, but our optimized prototype abstracts this down to a high-level set of features involving spoof movies, technical quality, and humor, which are much more general and emblematic of prototype theory (Rosch, 1973).

In Section 2, we outline prior work in the area for context. Section 3 describes our architecture for training interpretable-by-design textual classifier neural networks, and our key innovation involving the usage of LLMs as an optimization tool to discover the learned prototypes. Section 4 runs our algorithm on six benchmark datasets across three models, demonstrating that intelligibility and accuracy are better achieved with our method compared to baseline approaches before a final discussion in Section 5.

To summarise, our three main contributions are as follows:

- We introduce a new paradigm for how to train interpretable-by-design prototype neural networks using LLMs as an optimization tool to try and discover the learned prototypes, rather than simply projecting them onto the nearest training instances.
- Our experiments show that the final prototypes produced by our optimization algorithm can radically simplify prototypes down to their core concepts and in doing so not compromise accuracy.
- We demonstrate this algorithm can also increase the accuracy of prototype neural networks compared to projection-based baselines when significant performance is lost due to prototype projection.

2 RELATED WORK

Prototype theory was first proposed by Rosch (1973) in the 1970s, the hypothesis was that humans learn abstract representations of a class for classification purposes, and indeed it was one of the first ideas from psychology imported into ML (Kim et al., 2014), largely inspired by earlier work in case-based reasoning (Smyth et al., 2001). In this section, we review initial work in the area, before a more specific discussion about text-based networks and the usage of LLMs as optimization tools.

2.1 INITIAL WORK

Prototype networks were first inspired by work in case-based reasoning (CBR) (Cunningham et al., 2003), which is based on the psychological theory that humans reason about new states based on their past history (i.e., cases), and use these past experiences to make decisions in the present. Kim et al. (2014) did foundational work in the area with their Bayesian case model, and this was followed with related CBR work in various guises across domains (Papernot & McDaniel, 2018; Kenny et al., 2023).

108 However, it was Chen et al. (2019) who inspired a flurry of work in prototype networks, and helped
 109 popularize them into the prominent interpretable-by-design architectural choice (Ma et al., 2024).
 110

111 **2.2 TEXT-BASED PROTOTYPE NETWORKS**
 112

113 Inspired by the work in images, text-based networks began to emerge. First was the work by Ming
 114 et al. (2019), who outlined a general framework for applying the work of Li et al. (2018) to text
 115 domains. The authors demonstrated widespread utility of the framework, and we adapt it here for
 116 our computational evaluation later. Van Aken et al. (2022), used prototype networks for medical
 117 classification; departing from prototype projection onto nearest training examples, the authors leave
 118 the learned “black box” prototype vectors as is to preserve classification performance, and simply
 119 show the nearest training examples without prototype projection. This is similar to Das et al. (2022)
 120 who proposed ProtoTex, and innovated with counterfactual ‘negative’ prototypes, but again opted
 121 to avoid prototype projection. This practice of not projecting prototypes has been criticized lately
 122 in the literature by Ma et al. (2024), who point out this practice does not result in interpretable
 123 networks, and is just done to avoid drops in classification performance. Having said this, there has
 124 still been a wave of work adopting the approach of prototype projection, either on the case level (Wen
 125 et al., 2025; Wen & Rezapour, 2025), the sentence level (Hong et al., 2023; Wei & Zhu, 2024),
 126 or just general evaluation of the frameworks (Sourati et al., 2024; Hanawa et al.; Davoodi et al.,
 127 2023). We offer a fresh perspective on this long standing problem, instead of projecting prototypes
 128 onto training examples (and often losing performance), or leaving them as black box vectors (and
 129 losing interpretability), we attempt to discover the actual learned “black-box” prototype vectors by
 130 optimizing towards them with LLMs.

131 **2.3 LANGUAGE MODELS AS OPTIMIZERS**
 132

133 Since the recent success of generative AI, researchers have tried to capitalize upon LLMs as optimization
 134 tools to solve previously unapproachable problems. Early work in this area mainly worked by
 135 finetuning LLMs to be optimizers (Meyerson et al., 2023; Lehman et al., 2023; Chen et al., 2023a).
 136 However, recently Yang et al. (2024) looked at just using prompting itself to solve toy problems such
 137 as linear regression and the traveling salesman problem, and had reasonable success in doing so. We
 138 are inspired by this idea, but interested in the problem of using LLMs to visualize learned prototypes
 139 for more interpretable and accurate text classifiers.

140 **3 METHOD**
 141

142 In this section we describe our backbone architecture for training these networks, which is a general-
 143 ization over prior work (Ming et al., 2019; Das et al., 2022; Ma et al., 2024). We assume some
 144 encoder $f(\cdot)$ comprising of frozen parameters $f_{enc}(\cdot)$, and a subsequent set of trainable ones $\phi(\cdot)$. In
 145 our setup, converging with the empirical observations of Sourati et al. (2024) for best performance,
 146 $f_{enc}(\cdot)$ represents all blocks of our text classification models, except for the final trainable block $\phi(\cdot)$.
 147 Thus, considering some datum $x_i \in \{X\}_{i=0}^N$, its final latent representation may be obtained via the
 148 transformation $z = \phi(f_{enc}(x_i))$. In a forward pass through the network, a datum x_i has its similarity
 149 measured against $p \in \{P\}^m$ prototypes to form an activation vector $a_i \in R^m$. Finally, a_i is fed to
 150 a linear weight matrix $W \in R^{(m,c)}$ to give the final classification logits, where c is the number of
 151 classes.

152 **Similarity Function.** Following more recent trends in prototype-based networks (Ma et al., 2024),
 153 and mirroring common practice in NLP (Mikolov et al., 2013), we use cosine similarity as our
 154 similarity function:

$$\cos(z_i, P_k) = \frac{z_i}{\|z_i\|_2} \cdot \frac{P_k}{\|P_k\|_2} \quad (1)$$

155 Although largely an insignificant hyperparameter (Sourati et al., 2024), this choice of similarity
 156 function ensures the focus on angular similarity between representations. Empirically, we also found
 157 it supported better convergence than L_2 norm-based distance functions which are another popular
 158 choice (Chen et al., 2019; Das et al., 2022; Kenny et al., 2023).

162 3.1 LOSS FUNCTION
163

164 Following the majority of work in this field (Ming et al., 2019; Sourati et al., 2024; Das et al., 2022),
165 our loss function consists of four components that are all minimized during training: a classification
166 loss, interpretability loss, clustering loss, and separation loss.

167 The classification loss L_{ce} is the standard cross-entropy loss between predicted and true labels that
168 we seek to minimize:

$$169 \quad 170 \quad 171 \quad L_{ce} = - \sum_{i=1}^N \mathbf{y}_i \cdot \log(\text{SoftMax}(W^T \cdot a_i)) \quad (2)$$

172 where $a_i = [\cos(z_i, p_1), \cos(z_i, p_2), \dots, \cos(z_i, p_m)]$ is the activation vector of cosine similarities
173 between example x_i and all prototypes, \mathbf{y}_i is the one-hot encoded true label vector for example x_i ,
174 and N the batch size.

175 The interpretability loss L_i ensures that prototypes remain interpretable by maximizing the similarity
176 between each prototype and its closest training sample:

$$177 \quad 178 \quad 179 \quad L_i = \frac{1}{M} \sum_{k=1}^M \min_{j \in \{1, \dots, N\}} (1 - \cos(p_k, z_j)) \quad (3)$$

180 This constraint keeps prototypes close to actual training samples by maximizing their similarity to
181 the nearest training examples, allowing them to be represented by their closest training examples.
182 Subtracting the cosine similarity from 1 allows us to form this as a minimization objective with a
183 lower bound of 0.

184 The clustering loss L_c ensures that training examples are close to at least one prototype by maximizing
185 the similarity between each training example and its nearest prototype:

$$186 \quad 187 \quad 188 \quad L_c = \frac{1}{N} \sum_{j=1}^N \min_{k \in \{1, \dots, M\}} (1 - \cos(z_j, p_k)) \quad (4)$$

189 where p_k represents the k -th prototype, and z_j is the encoded representation of example x_j . Minimizing
190 this loss pulls training examples closer to their nearest prototypes.

191 The separation loss L_s encourages prototype diversity by minimizing the cosine similarity between
192 different prototypes:

$$193 \quad 194 \quad 195 \quad L_s = \frac{2}{M(M-1)} \sum_{\substack{k, l \in \{1, \dots, M\} \\ k < l}} (1 + \cos(p_k, p_l)) \quad (5)$$

196 where we sum over unique pairs (k, l) with $k < l$ to avoid double counting. Since cosine similarity
197 $\cos(p_k, p_l)$ ranges from $[-1, 1]$, the term $(1 + \cos(p_k, p_l))$ ranges from $[0, 2]$. Minimizing this term
198 pushes prototypes apart in the embedding space by penalizing high similarity, with the minimum
199 value of 0 achieved when prototypes are maximally dissimilar (cosine similarity = -1).

200 The total loss is defined as:

$$201 \quad 202 \quad 203 \quad L = L_{ce} + \lambda_i L_i + \lambda_c L_c + \lambda_s L_s \quad (6)$$

204 where $\lambda_c, \lambda_i, \lambda_s \geq 0$ are regularization factors that control the contribution of each auxiliary loss
205 term. Mirroring Ming et al. (2019), we set these to $\lambda_i = 0.1$, $\lambda_c = 0.01$, and $\lambda_s = 0.01$. Due to the
206 careful formulation of each term, all components are minimization objectives with a well defined
207 lower bound, allowing more stable training.

208 3.2 LLM OPTIMIZATION
209

210 Prototype-based text classifiers learn latent prototype vectors $\mathcal{P} = \{p_1, p_2, \dots, p_m\}$ that, while
211 effective for classification, lack direct interpretability. These learned prototypes exist in high-
212 dimensional embedding spaces and can only be understood through their nearest training examples,
213 which may not capture the essential characteristics that make them effective decision boundaries (Chen
214 et al., 2019). We propose a novel approach that leverages large language models (LLMs) to discover

216 these abstract prototypes as interpretable, optimized text representations that maintain or improve
 217 classification performance.
 218

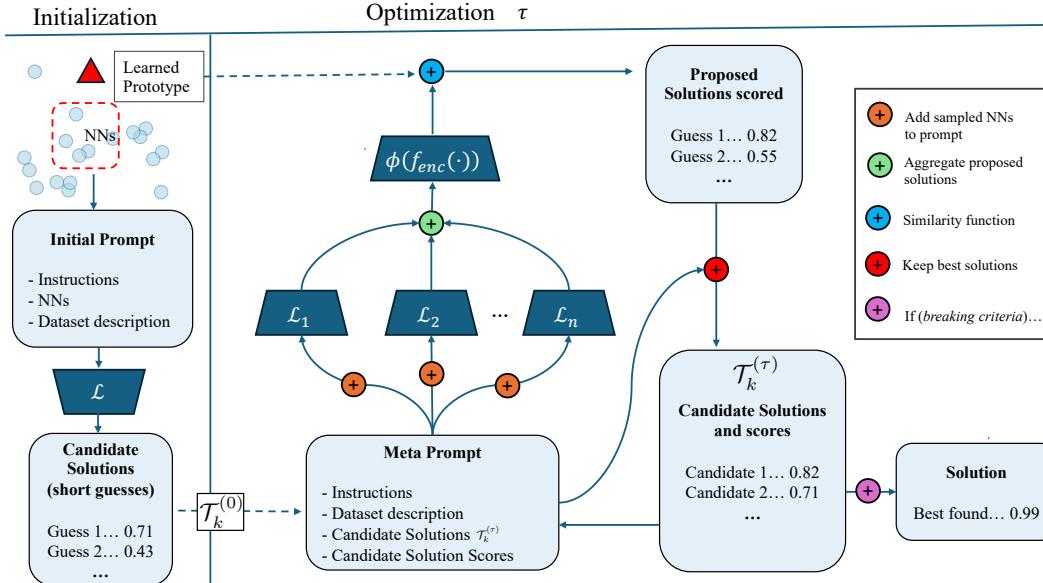
219 Formally, given a learned prototype $p_k \in \mathbb{R}^d$ and an LLM \mathcal{L} , our objective is to find an optimal
 220 textual representation t_k^* such that:

$$221 \quad t_k^* \approx \arg \max_t \cos(\phi(f_{enc}(t)), p_k) \quad (7)$$

223 where t is a textual input to the classifier generated by an LLM, and t_k^* should give an approximation
 224 of the prototype which is short, focuses on core concepts in the domain, highly general, and minimizes
 225 the presence of irrelevant features.
 226

227 3.2.1 INITIALIZATION

228 Our method begins by generating short, focused textual guesses using the nearest neighbor training
 229 examples as a guide. For each learned prototype $p_k \in \mathcal{P}$, we first identify the top- k nearest training
 230 examples $\mathcal{NN}_k = \{x_{i_1}, x_{i_2}, \dots, x_{i_k}\}$ based on cosine similarity in the embedding space. Rather
 231 than using these examples in our meta prompt and iteration loop, we first prompt $\mathcal{L}(\cdot)$ to generate
 232 concise text snippets $\mathcal{T}_k^{(0)}$ that capture the essential concepts observed in these examples, this
 233 minimizes the context window and escapes the local minima around the prototype, which often biases
 234 longer text. In practice, we often found this was necessary if similarity to p_k was already high, as
 235 it was difficult for $\mathcal{L}(\cdot)$ to guess better examples which improved upon the similarity scores. This
 236 resulted in worse initial guesses, but allowed the process to optimize towards the learned prototype
 237 using shorter textual examples, ultimately resulting in the same (or better) performance. In the end,
 238 this had the effect of speeding up the optimization loop, and finding shorter, more concise prototype
 239



261 Figure 2: Method overview: Initially we construct a prompt from the closest neighbours to the
 262 learned prototype we are trying to visualize and the dataset specific description. This prompt generates
 263 a set of n short textual guesses for the prototype which are used to begin the optimization process.
 264 Each optimization iteration constructs a meta prompt from the dataset specific instructions, the current
 265 candidate solutions, and their respective scores. The meta prompt is duplicated n times, each with
 266 randomly sampled nearest neighbors to the prototype to help diversity in the LLM’s new guesses.
 267 The guesses from each LLM are aggregated and passed into our classifier’s encoder to get their
 268 representations. These encoded guesses then have their similarity measured to the prototype, and are
 269 each scored, if any are better than the current candidate guesses, they replace them. At this point, if
 the stopped condition is met, the highest scoring guess is used for prototype projection.

270 texts. We construct an initial prompt $\pi_{init}(\mathcal{NN}_k)$ and extract candidate texts from the LLM response,
 271 selecting those with highest similarity to p_k to form our initial candidate set $\mathcal{T}_k^{(0)}$.
 272

273 **3.2.2 ITERATIONS AND STOPPING CRITERIA**
 274

275 At each iteration τ , we maintain a set of candidate prototype texts which represent the current best
 276 guesses from $\mathcal{L}(\cdot)$ across all iterations with:
 277

$$278 \quad \mathcal{T}_k^{(\tau)} = \{t_k^{(\tau,1)}, t_k^{(\tau,2)}, \dots, t_k^{(\tau,|\mathcal{T}_k^{(\tau)}|)}\} \quad (8)$$

280 for each prototype p_k , where $\mathcal{T}_k^{(\tau)}$ denotes the set of candidate prototype texts for prototype p_k at
 281 iteration τ , and $t_k^{(\tau,i)}$ represents the i -th candidate text in this set. We compute similarity scores for
 282 all current candidates using $s_i^{(\tau)} = \cos(\phi(f_{enc}(t_k^{(\tau,i)})), p_k)$ and construct an optimization prompt
 283 that includes current candidates with their scores, along with sample training examples from \mathcal{NN}_k .
 284 The LLM generates new candidate texts, which we evaluate and use to update our candidate set
 285 by replacing lower-scoring texts with better alternatives that achieve higher similarity to the target
 286 prototype p_k such that:
 287

$$288 \quad \mathcal{T}_k^{(\tau+1)} = \text{TopK} \left(\mathcal{T}_k^{(\tau)} \cup \mathcal{G}_k^{(\tau+1)}, K \right) \quad (9)$$

289 where $\mathcal{G}_k^{(\tau+1)}$ represents the new candidate texts generated by the LLMs at iteration $\tau + 1$, and
 290 $\text{TopK}(\cdot, K)$ selects the K texts with highest cosine similarity scores $\cos(\phi(f_{enc}(t)), p_k)$.
 291

292 To maximize exploration at each iteration, we employed a set of LLMs such that $\mathcal{L} =$
 293 $\{\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_n\}$. This set of LLMs process the same meta prompt customized with different
 294 sampled training examples from \mathcal{NN}_k , serving to ensure each prompt receives variation while focusing
 295 on the relevant part of the latent space containing the concept we seek to refine. The stopping
 296 criteria can either be when a near-perfect solution is found such that:
 297

$$298 \quad \max_{t \in \mathcal{T}_k^{(\tau)}} \cos(\phi(f_{enc}(t)), p_k) \approx 1.0, \quad (10)$$

300 or a set number of iterations is performed. In practice, it is unlikely to map to a perfect solution as the
 301 prototype may lay slightly off the textual data manifold, so a threshold would be needed for the prior
 302 stopping approach.
 303

304 **3.2.3 IMPLEMENTATION DETAILS**
 305

306 We employed 3 parallel LLM instances with `Meta-Llama-3-8B-Instruct`. We set the
 307 number of candidate solutions at each iteration to a maximum of 10, meaning the algorithm could
 308 replace its current 10 best candidates with potentially better ones from the pool of 30 newly generated
 309 texts from 3 LLMs (each generating 10 solutions). When showing random training examples in each
 310 prompt at each iteration, we sampled 2 from the nearest neighbor pool \mathcal{NN}_k with $|\mathcal{NN}_k| = 20$. Our
 311 stopping criteria was to terminate the optimization after 20/15 iterations on our two set of experiments,
 312 respectively.
 313

314 **Prototype Projection.** After optimization, each learned prototype p_k is projected onto its best
 315 textual approximation from the LLM in the embedding space:
 316

$$317 \quad p_k^{proj} = \phi(f_{enc}(t_k^*)) \quad \text{where} \quad t_k^* = \arg \max_{t \in \mathcal{T}_k^{(T_{max})}} \cos(\phi(f_{enc}(t)), p_k) \quad (11)$$

318 This projection ensures that each prototype can be directly interpreted through its optimized textual
 319 representation t_k^* , facilitating human understanding and analysis of the model’s decision-making
 320 process. In our baseline methods, we project each prototype onto its nearest training example:
 321 $p_k^{proj} = z_j^*$ where $j^* = \arg \max_{j \in \{1, \dots, N\}} \cos(p_k, z_j)$. As Theorem 2.1 from ProtoPNet (Chen
 322 et al., 2019) states this stage should only result in accuracy performance drop if the similarity
 323 difference is too great. We study situations when this gap is small and large in the next section to see
 how our algorithm can help in both circumstances.
 324

324

3.3 HYPERPARAMETER CHOICES

326 As LLMs are typically familiar with NLP datasets, we found adding dataset-specific descriptions
 327 improved the guesses at initialization. The number of nearest neighbors is dataset specific and a
 328 hyperparameter to be tuned, we empirically found via a grid-search that if the number was too high,
 329 the information given to the LLM(s) is too volatile and disrupts optimization, likewise we found
 330 if the number was too low, then diversity suffers and early local minima become problematic. The
 331 number of LLMs used is important as the more one can use, the more guesses that are possible each
 332 iteration, and the better convergence is (Yang et al., 2024). We found our LLM was only capable of
 333 making 10 guesses reliably each iteration, so by using three in parallel we could triple the amount of
 334 guesses, which again helps avoid early local minima (Yang et al., 2024).

335

4 EXPERIMENTS

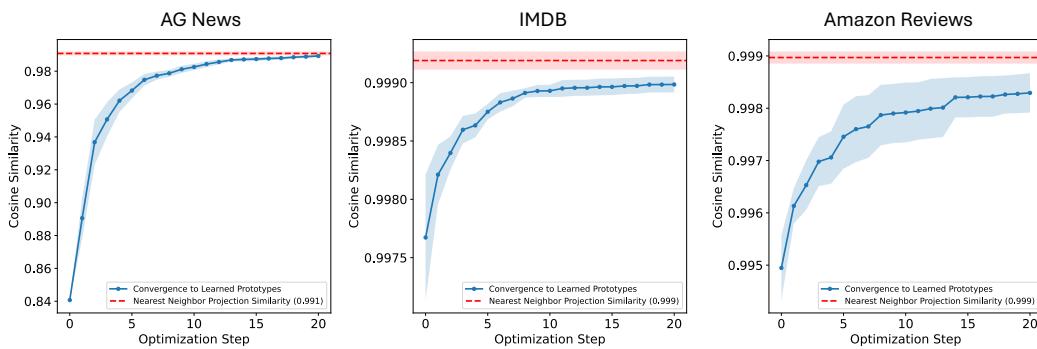
336 We were interested in testing two hypothesis. First, we anticipate that our method could be used to
 337 simplify overly long, complex prototypes to their core semantic meaning without losing accuracy.
 338 Second, it is hypothesized that in cases where prototype projection results in a significant decrease in
 339 classifier accuracy, our approach can converge to a better solution which preserves some of the delta
 340 loss. The next two sections put these hypotheses to the test.

341

4.1 OPTIMIZING PROTOTYPE INTELLIGIBILITY

342 In this section we test if our method can simplify complex prototype textual representations into more
 343 intelligible ones. To do this, we utilize three popular models in the literature for text-based prototype
 344 models (Das et al., 2022), BERT (Devlin et al., 2019), Electra (Clark et al., 2020), and RoBERTa (Liu
 345 et al., 2019).¹ All three had their parameters frozen except for the final block $\phi(\cdot)$ which was finetuned
 346 to help learn better representations (Sourati et al., 2024). For datasets we consider IMDB (Maas et al.,
 347 2011), Amazon Reviews (McAuley & Leskovec, 2013), and AG News (Zhang et al., 2015), because
 348 their typical textual datum are suitably long. We set the number of prototypes in each model equal to
 349 three times the number of classes to increase prototype variability while maintaining computational
 350 tractability, and allowed our algorithm to optimize for 20 iterations per prototype. Additional details
 351 are in Appendix A and B. All tests were run across 6 seeds, with mean and standard error reported.

352 ¹In early tests we also considered Llama and Qwen LLM models as backbone encoders, but we found they
 353 performed worse so ultimately omitted them from the results.



373 Figure 3: Optimization curves on the intelligibility experiment: All models and datasets converged
 374 gradually towards better prototypes which near equaled the original projection prototype’s similarity
 375 scores, but were on average only 13.9% of the number of characters in length, thus helping to optimize
 376 interpretability of the prototypes. AG News had the initial biggest drop in similarity when beginning
 377 optimization, but quickly converged nonetheless towards a similar result to the other datasets. All
 378 results are averaged across models, and show mean and standard error across 6 random seeds.

378

379 Table 1: Intelligibility experiment results: Overall, our LLM-based optimization approach to prototype
 380 discovery achieved notably shorter prototypes. Specifically, the average character length of our
 381 prototypes was 95.5, compared to using the training data as projections which was 682.3. There was
 382 a relatively insignificant drop in accuracy with our approach overall corresponding to 0.02%. Length
 383 is represented by the number of characters in the text. The change in accuracy is represented by Δ .

384	Dataset	Model	Learned	Optimized Prototype			Projected Prototypes		
			Acc.	Acc.	Δ	Length	Acc.	Δ	Length
386	IMDB	MPNet	92.70 \pm 0.11	92.69 \pm 0.12	-0.01	67.7	92.70 \pm 0.10	-0.01	898.1
		Modern-Bert	87.30 \pm 0.24	87.27 \pm 0.21	-0.03	115.2	87.32 \pm 0.23	+0.02	846.7
		Bert	90.51 \pm 0.16	90.51 \pm 0.17	-0.00	73.2	90.49 \pm 0.17	-0.02	885.6
		Roberta	93.17 \pm 0.07	93.14 \pm 0.05	-0.03	96.9	93.18 \pm 0.07	+0.01	886.8
		Electra	93.39 \pm 0.02	93.32 \pm 0.09	-0.07	131.5	93.39 \pm 0.01	+0.00	933.5
		MPNet	83.73 \pm 0.45	83.52 \pm 0.32	-0.20	97.3	83.63 \pm 0.46	-0.10	365.2
390	Amazon reviews	Modern-Bert	79.85 \pm 0.60	78.52 \pm 1.04	-1.33	90.8	79.85 \pm 0.65	-0.00	340.0
		Bert	81.91 \pm 0.56	81.45 \pm 0.91	-0.46	64.8	81.39 \pm 0.84	-0.52	307.4
		Roberta	84.33 \pm 0.36	84.28 \pm 0.19	-0.05	89.0	84.33 \pm 0.21	+0.01	402.3
		Electra	85.10 \pm 0.60	85.20 \pm 0.44	+0.10	60.3	85.06 \pm 0.64	-0.04	254.1
		MPNet	93.22 \pm 0.10	93.22 \pm 0.10	+0.00	139.2	93.20 \pm 0.09	-0.02	185.4
		Modern-Bert	91.61 \pm 0.06	91.60 \pm 0.08	-0.01	150.2	91.63 \pm 0.07	+0.01	194.4
393	AG News	Bert	93.03 \pm 0.09	93.03 \pm 0.13	+0.00	128.4	93.04 \pm 0.12	+0.01	199.3
		Roberta	93.35 \pm 0.07	93.33 \pm 0.03	-0.02	142.1	93.33 \pm 0.05	-0.02	193.2
		Electra	92.27 \pm 0.10	92.27 \pm 0.13	+0.00	143.0	92.26 \pm 0.11	-0.01	185.9
		MPNet	93.22 \pm 0.10	93.22 \pm 0.10	+0.00	139.2	93.20 \pm 0.09	-0.02	185.4
		Modern-Bert	91.61 \pm 0.06	91.60 \pm 0.08	-0.01	150.2	91.63 \pm 0.07	+0.01	194.4
		Bert	93.03 \pm 0.09	93.03 \pm 0.13	+0.00	128.4	93.04 \pm 0.12	+0.01	199.3
397	Mean				-0.14	106.0		-0.04	471.9

398

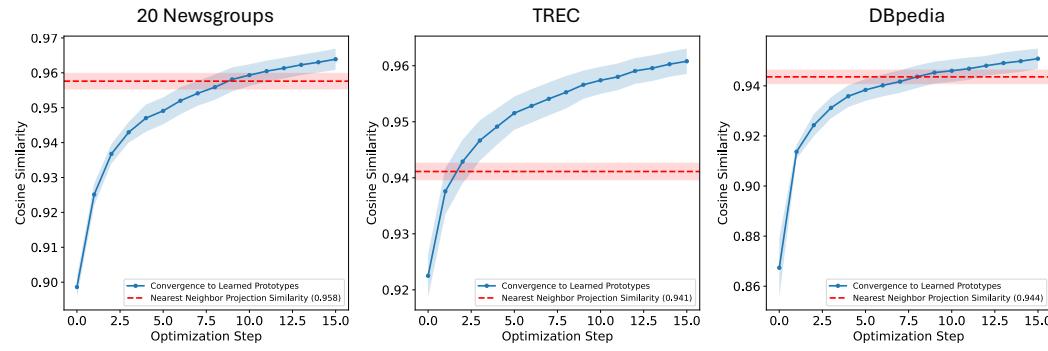
399

400

401 The results may be seen in Figure 3 and Table 1. Overall, we found that it was possible to optimize
 402 towards prototypes which had near perfect similarity scores to the learned prototypes as their projected
 403 counterparts, resulting in a relatively insignificant accuracy performance difference (Opt.=0.03% drop
 404 v. Proj.=0.01%). Notably, AG News saw similarity drop-off most at the beginning of optimization
 405 (0.84 cosine similarity v. 0.99 nearest neighbour projection), signally a harder optimization problem,
 406 but nevertheless it progressed to the same similarity as the projection variant over 20 iterations
 407 and matched its accuracy performance, but with prototypes which were on average 25% shorter.
 408 Across all three datasets the change in prototype intelligibility was highly significant, as they dropped
 409 from an average of 682.3 characters in length using nearest neighbor projection, to just 95.5 with
 410 the optimized variant, meaning our prototypes are on average just 13.9% the size compared to the
 411 baseline. For more detailed results see Appendix C.

412

413



424

425

426 Figure 4: Optimization curves on the accuracy experiment: We found across all three datasets that
 427 it was possible to discover prototypes with higher similarity to the learned prototypes after just 15
 428 iterations on average. Initially, there is a significant drop in similarity, but the optimization curves
 429 quickly achieve better similarity than the nearest neighbour prototypes which are typically used in the
 430 literature. Overall, this resulted in a significantly smaller accuracy drop for the optimized prototype
 431 networks compared to the projection-based ones (Opt=0.78% drop v. Proj=1.12% drop). All results
 432 are averaged across models, and show mean and standard error across 6 random seeds.

432
 433 Table 2: Accuracy experiment results: The learned prototype network accuracy is displayed before
 434 any projection or optimization, alongside the subsequent accuracy, delta drop in accuracy, and cosine
 435 similarity to the learned prototypes using both our optimization approach, and prototype projection.
 436 Overall, there is significantly higher cosine similarity to the learned prototypes after using our
 437 optimization approach, and a smaller Δ drop in accuracy.

438	439	Dataset	Model	Learned		Optimized		Projected									
				440	Acc.	441	Acc.	442	Δ	443	$s(\cdot, \cdot)$	444	Acc.	445	Δ	446	$s(\cdot, \cdot)$
20newsgroups	MPNet	83.60 \pm 0.15	83.40 \pm 0.07	-0.20	0.97	83.03 \pm 0.19	-0.58	0.95									
		70.51 \pm 1.72	69.65 \pm 1.83	-0.87	0.98	68.89 \pm 1.79	-1.62	0.98									
		81.27 \pm 0.40	81.05 \pm 0.35	-0.21	0.97	80.51 \pm 0.43	-0.76	0.95									
		80.42 \pm 1.50	80.14 \pm 1.52	-0.28	0.98	80.07 \pm 1.50	-0.35	0.98									
		69.18 \pm 1.76	67.95 \pm 2.50	-1.22	0.97	67.77 \pm 2.56	-1.40	0.97									
		87.77 \pm 0.56	87.83 \pm 0.47	+0.06	0.99	87.57 \pm 0.32	-0.20	0.98									
Dbpedia	Modern-Bert	84.11 \pm 0.31	84.10 \pm 0.33	-0.01	0.98	83.75 \pm 0.34	-0.36	0.97									
		88.22 \pm 0.25	88.48 \pm 0.26	+0.26	0.96	88.37 \pm 0.45	+0.15	0.94									
		87.63 \pm 0.46	87.26 \pm 0.72	-0.37	0.98	86.07 \pm 1.09	-1.56	0.96									
		86.34 \pm 0.26	86.36 \pm 0.24	+0.02	0.98	86.33 \pm 0.23	-0.01	0.98									
		91.60 \pm 0.76	91.40 \pm 0.76	-0.20	1.00	91.33 \pm 0.81	-0.27	0.99									
		73.53 \pm 0.07	72.53 \pm 0.35	-1.00	0.99	72.73 \pm 0.64	-0.80	0.99									
Trec	Bert	91.07 \pm 0.55	90.80 \pm 0.23	-0.27	0.98	90.87 \pm 0.29	-0.20	0.97									
		86.93 \pm 2.29	86.93 \pm 1.91	+0.00	0.99	86.27 \pm 2.32	-0.67	0.98									
		86.07 \pm 1.16	85.93 \pm 1.09	-0.13	0.99	85.73 \pm 1.35	-0.33	0.98									
		Mean			-0.29	0.98			-0.60		0.97						

453

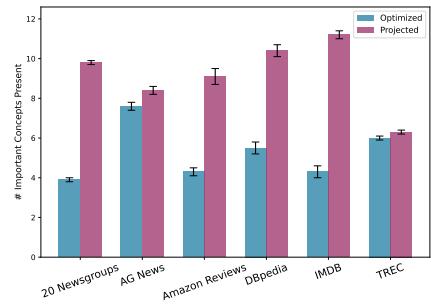
454 4.2 CLOSING THE ACCURACY GAP DUE TO PROTOTYPE PROJECTION

455
 456 In this section we test if our method can improve the classification performance when prototype
 457 projection results in a significant degradation. We again utilize the same models, but this time consider
 458 the datasets TREC (Li & Roth, 2002), DBpedia (Zhang et al., 2015), and 20 Newsgroups (Lang,
 459 1995). We selected these datasets due to their relative difficulty which is necessary to maximize
 460 the effect of accuracy loss due to prototype projection (Chen et al., 2019). We set the number of
 461 prototypes in each model equal to the number of classes, and under-sampled DBpedia to its 20 most
 462 difficult to classify classes, to maximize the number of seeds reported. We optimized for 15 iterations
 463 per prototype. Additional details may be found in Appendix A and B. All tests were run across 6
 464 seeds, with the means and standard errors reported.

465
 466 The results may be seen in Figure 4 and Table 2, where
 467 the commonly reported artifact of accuracy loss due to
 468 prototype projection was observed. We found that on aver-
 469 age it was possible to optimize towards prototypes which
 470 were more similar to the learned prototype vectors than the
 471 nearest neighbour projections, this resulted in both higher
 472 cosine similarity scores (Opt=0.96 v. Proj=0.95), and
 473 accuracy metrics (Opt.=0.78% drop v. Proj.=1.12% drop).
 474 Notably, TREC benefited the most from the optimization
 475 with the highest increase in cosine similarity and accuracy,
 showing that the prior is a reasonable proxy for the latter.

476 477 4.3 QUALITATIVE ANALYSIS

478
 479 Recent studies have shown LLMs can function as a reliable
 480 proxy for user evaluation (Cui et al., 2024), hence, we used
 481 claude-3-7-sonnet-20250219-v1:0 to analyze
 482 our systems in an “LLM as a judge” setup (Gu et al.,
 483 2024). We sampled 100 test instances from each dataset
 484 and model pairing, across seeds (i.e., 10,800). Then, we
 485 ran inference and took the closest prototypes in both model
 types (LLM-optimized and training data projection). For
 each, a prompt was constructed which included the test instance and both prototypes. We instructed



486
 487 Figure 5: Qualitative test: Our method
 488 preserved 57% relevant concepts to the
 489 test instances, despite being only 11%
 490 of the length of the projected prototypes.
 491 This highlights that our process has a
 492 positive tradeoff between the prototype
 493 length and concept preservation.

486 the LLM to list all high-level concepts in both prototypes which could be used for classification,
 487 and then decide which was most similar. The results are in Figure 5, where our method was flagged
 488 as having 57% of the concepts on average compared to projection (5.3 ± 0.2 v. 9.2 ± 0.2), despite
 489 being only 11% the length. This showed that our method generates prototypes which are nearly 10x
 490 shorter while preserving most of the important concepts. We observed no significant difference in
 491 the decisions of which prototype was most similar to the test example (Opt.=49% v. Proj.=51%),
 492 illustrating the important concepts for similarity were preserved. **Put another way, the vast majority**
 493 **of the seemingly important concepts identified by the LLM were actually not relevant for assessing**
 494 **similarity to the test instances. Hence, although our approach did lose almost half the concepts, they**
 495 **were spurious or irrelevant features which would only serve to confuse human users and increase**
 496 **cognitive load (Doshi-Velez & Kim, 2017).** For the full prompt see Appendix G.

497 5 DISCUSSION

500 We proposed a paradigm shift in how to train prototype neural networks, which works by visualizing
 501 the learned prototypes through an optimization process involving foundation models rather than
 502 simply projecting them to nearest neighbours in the training data. Our results showed that we can
 503 drastically simplify prototypes down to their core concepts without sacrificing performance, and in
 504 the cases where projection does result in performance drops, we can help mitigate this to improve the
 505 networks. **The reduction in prototype length will drastically reduce cognitive load for users (Matthews**
 506 **& Folivi, 2023), and increase the accessibility of the system (Darling-White & Polkowitz, 2023).**

507 During early testing we initially used Claude-3.7-Sonnet as our optimizer LLM, and it produced
 508 even better results as well as being 10x faster, especially when queried in parallel (up to 50 times).
 509 However, we opted to use Meta-Llama-3-8B-Instruct because of two reasons. First, the
 510 model is far more accessible for researchers, as long as the lab has one GPU, it will work with no
 511 API cost. Secondly, we found other open source models such as Qwen 7B-70B in size gave no great
 512 benefit, but just slowed down generation due to being more computationally expensive.

513 A possible criticism of our work is that we only focused on text domains, in future work it would
 514 be interesting to replicate our results in an image domain with part-prototype networks using stable
 515 diffusion (or other suitable models) in place of our LLMs.

516 517 REPRODUCIBILITY STATEMENT

518 The code to reproduce the results was given with the submission. Follow the `readme.txt` file and
 519 the results will reproduce. An API subscription to anthropic is required to reproduce the results of
 520 our qualitative test, and access to multiple GPUs is recommended.

521 523 ETHICS STATEMENT

524 One of the potential ethical considerations of using prototypes for explanations is that the examples
 525 used must preserve privacy and not be offensive or unethical in any sense. However, provided the
 526 LLM used for optimization is appropriately aligned, the probability of producing offensive content
 527 is highly mitigated. Moreover, as the number of prototypes in our models is quite small, it is more
 528 easily screened before deployment, which helps significantly compared to more traditional CBR
 529 approaches that may use the whole dataset or a large portion.

531 532 REFERENCES

533 Dario Amodei. The urgency of interpretability, 2024. URL <https://www.darioamodei.com/post/the-urgency-of-interpretability>. Accessed: 2025-09-22.

534 Angelica Chen, David Dohan, and David So. Evoprompting: Language models for code-level neural
 535 architecture search. *Advances in neural information processing systems*, 36:7787–7817, 2023a.

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

540 Valerie Chen, Q Vera Liao, Jennifer Wortman Vaughan, and Gagan Bansal. Understanding the role of
 541 human intuition on reliance in human-ai decision-making with explanations. *Proceedings of the*
 542 *ACM on Human-computer Interaction*, 7(CSCW2):1–32, 2023b.

543

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

546

547 Ziyuan Cui, Ning Li, and Huaikang Zhou. Can ai replace human subjects? a large-scale replication of
 548 psychological experiments with llms. *A Large-Scale Replication of Psychological Experiments*
 549 *with LLMs (August 25, 2024)*, 2024.

550

551 Pádraig Cunningham, Dónal Doyle, and John Loughrey. An evaluation of the usefulness of case-based
 552 explanation. In *International conference on case-based reasoning*, pp. 122–130. Springer, 2003.

553

554 Meghan Darling-White and Rachel Polkowitz. Sentence length effects on intelligibility in two groups
 555 of older children with neurodevelopmental disorders. *American Journal of Speech-Language*
 556 *Pathology*, 32(5):2297–2310, 2023.

557

558 Anubrata Das, Chitrang Gupta, Venelin Kovatchev, Matthew Lease, and Junyi Jessy Li. Prototex:
 559 Explaining model decisions with prototype tensors. In *Proceedings of the 60th Annual Meeting of*
 560 *the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2986–2997, 2022.

561

562 Omid Davoodi, Shayan Mohammadizadehsamakosh, and Majid Komeili. On the interpretability of
 563 part-prototype based classifiers: a human centric analysis. *Scientific Reports*, 13(1):23088, 2023.

564

565 Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep
 566 bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of*
 567 *the North American chapter of the association for computational linguistics: human language*
 568 *technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

569

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

572

573 Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen,
 574 Shengjie Ma, Honghao Liu, et al. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*,
 575 2024.

576

577 Kazuaki Hanawa, Sho Yokoi, Satoshi Hara, and Kentaro Inui. Evaluation of similarity-based
 578 explanations. In *International Conference on Learning Representations*.

579

Dat Hong, Tong Wang, and Stephen Baek. Prototynet-interpretable text classification via prototype
 580 trajectories. *Journal of Machine Learning Research*, 24(264):1–39, 2023.

581

Eoin M Kenny, Mycal Tucker, and Julie Shah. Towards interpretable deep reinforcement learning with
 582 human-friendly prototypes. In *The Eleventh International Conference on Learning Representations*,
 583 2023.

584

585 Been Kim, Cynthia Rudin, and Julie Shah. The bayesian case model: A generative approach for
 586 case-based reasoning and prototype classification. *Advances in neural information processing*
 587 *systems*, 27, 2014.

588

589 Ken Lang. Newsweeder: Learning to filter netnews. In *Machine learning proceedings 1995*, pp.
 590 331–339. Elsevier, 1995.

591

592 Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O Stanley.
 593 Evolution through large models. In *Handbook of evolutionary machine learning*, pp. 331–366.
 594 Springer, 2023.

595

Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep learning for case-based reasoning
 596 through prototypes: A neural network that explains its predictions. In *Proceedings of the AAAI*
 597 *conference on artificial intelligence*, volume 32, 2018.

598

Xin Li and Dan Roth. Learning question classifiers. In *COLING 2002: The 19th International*
 599 *Conference on Computational Linguistics*, 2002.

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

597

598 Chiyu Ma, Jon Donnelly, Wenjun Liu, Soroush Vosoughi, Cynthia Rudin, and Chaofan Chen.
 599 Interpretable image classification with adaptive prototype-based vision transformers. In *The
 600 Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

601 Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts.
 602 Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the
 603 association for computational linguistics: Human language technologies*, pp. 142–150, 2011.

604

605 Nestor Matthews and Folly Folivi. Omit needless words: Sentence length perception. *PLoS one*, 18
 606 (2):e0282146, 2023.

607

608 Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimen-
 609 sions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pp.
 610 165–172, 2013.

611

612 Elliot Meyerson, Mark J Nelson, Herbie Bradley, Arash Moradi, Amy K Hoover, and Joel Lehman.
 613 Language model crossover: Variation through few-shot prompting.(2023). URL <https://arxiv.org/abs/2302.12170>, 2023.

614

615 Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representa-
 616 tions in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

617

618 Yao Ming, Panpan Xu, Huamin Qu, and Liu Ren. Interpretable and steerable sequence learning via
 619 prototypes. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge
 Discovery & Data Mining*, pp. 903–913, 2019.

620

621 Nicolas Papernot and Patrick McDaniel. Deep k-nearest neighbors: Towards confident, interpretable
 622 and robust deep learning. *arXiv preprint arXiv:1803.04765*, 2018.

623

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

627

628 Eleanor H Rosch. Natural categories. *Cognitive psychology*, 4(3):328–350, 1973.

629

630 Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeff Wu, Lucius Bushnaq, Nicholas
 631 Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Bloom, et al. Open problems in
 mechanistic interpretability. *arXiv preprint arXiv:2501.16496*, 2025.

632

633 Barry Smyth, Mark T. Keane, and Pádraig Cunningham. Hierarchical case-based reasoning integrating
 634 case-based and decompositional problem-solving techniques for plant-control software design.
IEEE Transactions on Knowledge and Data Engineering, 13(5):793–812, 2001.

635

636 Zhivar Sourati, Darshan Deshpande, Filip Ilievski, Kiril Gashteovski, and Sascha Saralajew. Robust
 637 text classification: Analyzing prototype-based networks. In *Findings of the Association for
 638 Computational Linguistics: EMNLP 2024*, pp. 12736–12757, 2024.

639

640 Betty Van Aken, Jens-Michalis Papaioannou, Marcel Naik, Georgios Eleftheriadis, Wolfgang Nejdl,
 641 Felix Gers, and Alexander Loeser. This patient looks like that patient: Prototypical networks for
 642 interpretable diagnosis prediction from clinical text. In *Proceedings of the 2nd Conference of the
 643 Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International
 Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 172–184, 2022.

644

645 Bowen Wei and Ziwei Zhu. Advancing interpretability in text classification through prototype
 646 learning. *arXiv e-prints*, pp. arXiv–2410, 2024.

647

Ximing Wen and Rezvaneh Rezapour. A transformer and prototype-based interpretable model for
 contextual sarcasm detection. *arXiv preprint arXiv:2503.11838*, 2025.

648 Ximing Wen, Wenjuan Tan, and Rosina Weber. Gaprotonet: A multi-head graph attention-based
649 prototypical network for interpretable text classification. In *Proceedings of the 31st International*
650 *Conference on Computational Linguistics*, pp. 9891–9901, 2025.

651

652 Chenguang Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun
653 Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning*
654 *Representations*, 2024. URL <https://openreview.net/forum?id=Bb4VGOWELI>.

655 Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text
656 classification. *Advances in neural information processing systems*, 28, 2015.

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

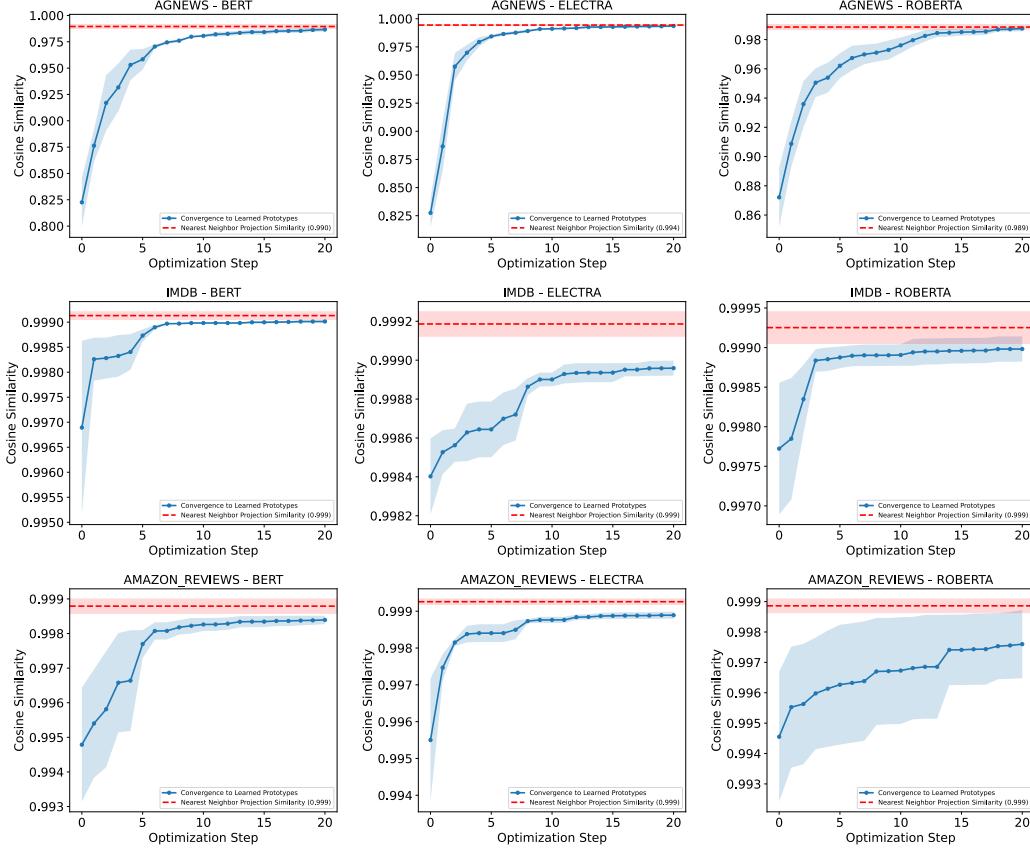
699

700

701

702 **A TRAINING HYPERPARAMETERS**
703704 The models used were BERT, RoBERTa, and ELECTRA backbones with prototype-based classifi-
705 cation heads, the final block on each was finetuned. Datasets were six text classification datasets:
706 IMDB, Amazon Reviews, AG News, TREC, DBpedia, and 20 Newsgroups.707 **Training Epochs:** Dataset-specific epochs: IMDB and Amazon Reviews (5 epochs), AG News (10
708 epochs), DBpedia (15 epochs), 20 Newsgroups (20 epochs), and TREC (100 epochs).709 **Prototypes per Class:**710
711 • 3 prototypes: IMDB, Amazon Reviews, AG News
712 • 1 prototype: TREC, DBpedia, 20 Newsgroups
713714 **Architecture:**
715716 • Input sequence length: 256 tokens
717 • Prototype dimension: 256
718 • Backbone models fine-tuned with prototype layers
719720 **Optimization:**
721722 • AdamW optimizer ($lr=3e-4$, weight decay=0.01)
723 • Batch sizes: 32 (training), 128 (validation/test)
724 • Train/validation split: 95/5 stratified
725726 **Loss Components:** Classification loss with prototype regularization terms. Loss weights: $\lambda_{p1} = 0.1$,
727 $\lambda_{p2} = 0.01$, $\lambda_{p3} = 0.01$.728 **Experimental Setup:** 6 random seeds (0-5), multi-GPU training across 4 CUDA devices, and early
729 stopping based on validation accuracy.
730731 **B DATASET PREPROCESSING**
732733 This appendix details the preprocessing steps applied to each dataset used in our experiments. Some
734 datasets were under-sampled to make training tractable, or filtered down to their most difficult classes
735 to enhance the phenomenon of prototype projection causing accuracy drops.
736737 **B.1 DBPEDIA**
738739 • Combined training, validation, and test splits from the original dataset
740 • Selected 20 target classes from the full set of categories [185, 166, 159, 57, 160, 168, 146,
741 198, 123, 38, 1, 73, 36, 56, 54, 215, 39, 128, 90, 171]. These were selected by training a
742 standard BERT classifier on the whole dataset and picking the 20 classes with the worst
743 classification performance.
744 • Applied a 90:10 train-test split to the filtered data
745746 **B.2 AMAZON REVIEWS (CELL PHONES & ACCESSORIES)**
747748 • Converted 5-point ratings to 3-class sentiment labels: ratings 4-5 mapped to Positive, rating
749 3 to Neutral, and ratings 1-2 to Negative
750 • Filtered out invalid text entries (null, empty, or whitespace-only reviews)
751 • Randomly sampled 100,000 examples from the filtered dataset
752 • Used a 90:10 train-test split
753754 **B.3 OTHER**
755

The 20 Newsgroups, TREC, IMDB, and AG News required no pre-processing.

756 C FULL OPTIMIZATION RESULTS
757758 In this section we include figures of the full optimization results from Section 4 in Figures 6 and 7.
759789
790 Figure 6: Full results for interpretability experiment.
791
792793 D OPTIMIZATION EXAMPLE
794795 Table 3 shows an example of what the textual optimizations look like in 20 Newgroups using BERT.
796 This examples started off particularly low in similarity, but quickly converged to a much higher value.
797798 E COMPUTATIONAL COST
799800 Preliminary evaluations compared the efficacy of proprietary black-box API services against locally
801 executed models on GPU infrastructure. The latter approach was selected to prioritize reproducibility
802 and mitigate operational costs for the research community. Regarding latency benchmarks, API-
803 based inference demonstrated the capacity to execute 20 optimization iterations in approximately
804 60 seconds (excluding Chain-of-Thought reasoning). In contrast, our local experimental config-
805 uration—utilizing a standard CUDA-enabled device with 24GB of VRAM—exhibited a latency
806 increase of approximately one order of magnitude ($10\times$). However, it is critical to contextualize this
807 computational overhead: the optimization of prototypes represents a non-recurring initialization cost.
808 Once optimized, the model parameters are fixed, ensuring that this initial computational expense is
809 amortized over the model’s life-cycle and does not impact subsequent inference efficiency.

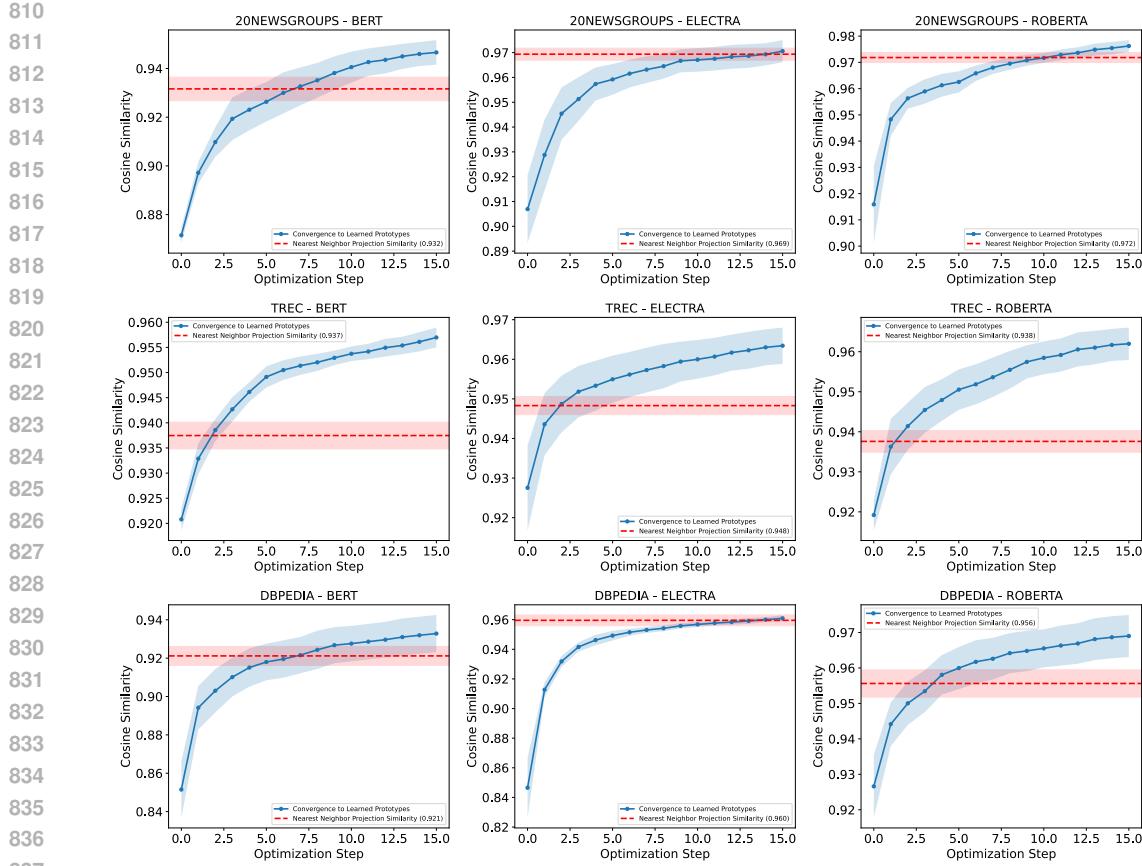


Figure 7: Full results for accuracy experiment.

F LLM USAGE

LLMs were mainly used to help edit experimental code in a collaborative setup and assist with polishing the manuscript before submission. Their usage in writing was extremely minimal. They had no role in research ideation. They had no role in retrieval or discovery of related work.

G PROMPT TEMPLATES

G.1 PROMPT FOR OPTIMIZATION INITIATION

I am trying to identify a prototypical example from the `{dataset_name}` dataset.

The prototype should represent a typical example of a `{description}`'. The following examples are very similar to the real prototype: `{examples_str}`

Based *only* on these examples, please generate a Python list containing exactly `{num_guesses_to_generate}` distinct, concise, and relevant phrases or sentences that you believe also capture the core concepts in these examples in a prototypical sentence.

Each phrase should be a potential textual description of the prototype and its core concepts.

Your output must be ONLY a single Python list of strings. For example: ["first candidate phrase", "second candidate phrase", ..., "tenth candidate phrase"]

Generated Python list:

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

Table 3: Example of prototype optimization best guesses on 20 Newsgroups using BERT.

Iteration	Current Best Guess	Cos Sim.
0	There are concerns about the rate at which two substances interact.	0.59
1	A sports highlight from a recent championship game	0.67
2	A sports news article about a recent match	0.70
3	A sports highlight from a recent tournament	0.72
4	A sports highlight from a recent tournament	0.72
5	A sports highlight from a recent tournament	0.72
6	Article about a recent sports championship win	0.73
7	Article about a recent sports championship win	0.73
8	Championship news headlines being released.	0.75
9	Championship news headlines being released.	0.75
10	Up-to-date sports news about championships.	0.76
11	Championship news from last week.	0.82
12	Championship news from last week.	0.82
13	Championship news being released recently now.	0.86
14	Recent news from championship headlines.	0.87
15	Recent news from championship headlines.	0.87

G.2 META PROMPT FOR OPTIMIZATION

You are a helpful assistant to a data scientist.

We are working together to try find a text sequence which perfectly maps to a learned black box prototype vector in the latent space of a language model. In doing so, we are querying you repeatedly in an optimization loop. This is one of those loops.

I will show you the current `{num_neighbors}` text sequences you generated previously, and their cosine similarity to the prototype. The closer the similarity is to 1, the better the guess is, because it's more similar to the prototype, the similarity ranges from -1 to 1. Our goal is to find a text sequence which perfectly maps to the prototype and gives a score of 1.

Here are the current `{num_neighbors}` text sequences you have generated previously in a query: `{population}` Their similarity scores are: `{[round(c, 2) for c in np.array(distances).flatten()]}`

Can you suggest another `{num_neighbors}` guesses which are closer to 1?

The prototype should represent a short, prototypical example of a positive '`{description}`'.

If a lot of your guesses are similar, you should try diversify them to avoid getting stuck in a local minimum, you can try vary the length, or even take random guesses. Here are some close training data neighbors of the black-box prototype to help you get some variety in your guesses: `{training_examples}`

Respond ONLY with your guesses as a Python list of strings.

For example:

`["first guess", "second guess", "...", "last guess"]`

It is extremely important you follow this format exactly.

918 G.3 PROMPT FOR QUALITATIVE ANALYSIS
919920 You are analyzing prototypes used by a neural network classifier that uses cosine similarity for
921 classification on the {dataset} dataset. The prototypes are being used to classify the test instance
922 based on their cosine similarity to it, your job is to help us analyze the prototypes.923 Test Instance to Classify: {test_text}
924925 Stage A Prototype: {stage_a_proto}
926927 Stage B Prototype: {stage_b_proto}
928929 Please analyze these prototypes and the test instance:
930931 1. First, identify ALL high-level concepts in the Stage A prototype that could be used by a classifier.
932933 2. Do the same analysis for the Stage B prototype.
934935 3. Based on cosine similarity principles, determine which prototype would be most similar to the test
936 instance.
937938 You should be comprehensive, you don't need to have the same number of concepts for both
939 prototypes, it is ok for one to have many more concepts.
940941 Provide detailed reasoning for your analysis, then output a JSON object with the following structure:
942943 "json {"
944 "stage_a_concepts_count": <integer>,
945 "stage_b_concepts_count": <integer>,
946 "most_similar_prototype": "<'stage_a' or 'stage_b'>,"
947 }" "
948949 The length and detail level of the prototypes do NOT matter for classification purposes, do not
950 consider them in your analysis, only focus on high-level concepts for the classification.
951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971