Prospector Heads: Generalized Feature Attribution for Large Models & Data

Gautam Machiraju [1] **Alexander Derry** [1] **Arjun Desai** [2] **Neel Guha** [3] **Amir-Hossein Karimi** [4] **James Zou** [1] **Russ B. Altman** [1] **Christopher Ré** [3 *] **Parag Mallick** [5 *]

Abstract

Feature attribution, the ability to localize regions of the input data that are relevant for classification, is an important capability for ML models in scientific and biomedical domains. Current methods for feature attribution, which rely on "explaining" the predictions of end-to-end classifiers, suffer from imprecise feature localization and are inadequate for use with small sample sizes and high-dimensional datasets due to computational challenges. We introduce prospector heads, an efficient and interpretable alternative to explanation-based attribution methods that can be applied to any encoder and any data modality. Prospector heads generalize across modalities through experiments on sequences (text), images (pathology), and graphs (protein structures), outperforming baseline attribution methods by up to 26.3 points in mean localization AUPRC. We also demonstrate how prospector heads enable improved interpretation and discovery of class-specific patterns in input data. Through their high performance, flexibility, and generalizability, prospectors provide a framework for improving trust and transparency for ML models in complex domains.

1 Introduction

Most ML models are optimized solely for predictive performance, but many applications also necessitate models that provide insight into features of the data that are unique to a particular class. This capability is known as *feature attribution*, which in unstructured data (*e.g.*, text, images, graphs) consists of identifying subsets of the input datum most responsible for that datum's class membership (*e.g.*, pixels or patches of an image, often represented as a heatmap). Feature attribution is especially important for scientific and biomedical applications. For example, for a model to assist a pathologist in making a cancer diagnosis, it ideally should not only accurately classify which images contain tumors, but also precisely locate the tumors in each image (Song et al., 2023; Niazi et al., 2019).

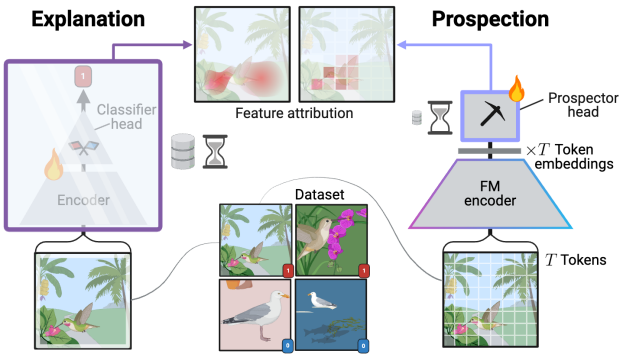

Figure 1: Explanation-based attribution can be conceptualized as a "wrapper function" for trained classifiers using internals, forward or backward passes, or input perturbations. Prospector heads are instead encoder-equippable like classifier heads and adapt token embeddings with data- and time-efficiency. Flame icon indicates trainable parameters.

Unfortunately, modern ML systems can struggle to perform feature attribution. Most existing attribution techniques attempt to provide "explanations" for trained classifiers (Figure 1) — descriptions of how model weights interact with different input features (*e.g.*, gradients (Simonyan & Zisserman, 2014), attention (Jetley et al., 2018)) or of how each feature contributes to prediction (*e.g.*, SHAP (Lundberg & Lee, 2017), LIME (Ribeiro et al., 2016)). Explanation-based attribution methods are inherently (a) data-inefficient as they require ample labeled training data to train underlying classifiers. Additionally, methods producing explanations can themselves be (b) computationally inefficient (Ancona et al., 2019; Chen et al., 2023) and thus may not actually improve tractability relative to annotation by domain experts, particularly for large inputs. Finally, (c) the attributed features are often found to be inaccurate and irrelevant to target classes (Arun et al., 2020; Zech et al., 2018; Jain & Wallace, 2019; Zhou et al., 2021b; Bilodeau et al., 2022).

*Joint senior authorship [1]Department of Biomedical Data Science, Stanford University [2]Cartesia AI [3]Department of Computer Science, Stanford University [4]Department of Electrical & Computer Engineering, University of Waterloo [5]Department of Radiology, Stanford University. Correspondence to: Gautam Machiraju <gmachi@stanford.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

We explore whether foundation models (FMs) can be used to solve challenges (a–c) without traditional explanations. Prior work demonstrates that FMs learn high quality data representations and can learn class-specific properties through a few labeled examples (Bommasani et al., 2021; Brown et al., 2020; Gondal et al., 2023). However, it is unclear whether FM representations can be used to perform feature attribution in a scalable and accurate manner. Our key insight is to build on top of FM representations, rather than explain an FM fine-tuned as an end-to-end classifier.

In this work we present *prospector heads* (a.k.a. "prospectors"), simple modules that aim to *equip* feature attribution to any encoder — including FMs — just as one would equip classification heads. Prospectors inductively reason over two layers: layer (I) categorizes learned representations into a finite set of "concepts" and layer (II) learns concepts' spatial associations and how those associations correlate with a target class. To (a) enable data efficiency, prospectors are parameter-efficient and with only hundreds of parameters. To (b) limit time complexity, prospectors operate with efficient data structures and linear-time convolutions, all without model backpropagation. To (c) improve attribution accuracy, prospector heads are explicitly trained to perform feature attribution, unlike explanation methods.

We show that prospector heads outperform attribution baselines over multiple challenging data modalities. Prospector-equipped models achieve gains in mean area under the precision-recall curve (AUPRC) of 8.5 points in sequences (text documents), 26.3 points in images (pathology slides), and 22.0 points in graphs (protein structures) over the top modality-specific baselines. Additionally, we show that prospector-equipped FMs are particularly robust to variation in the prevalence and dispersion of class-specific features. Finally, we also present visualizations of prospectors' internals and outputs to demonstrate their interpretability in complex domain applications.

2 Related Work

To adequately motivate our approach (Section 3), this section focuses on central methods ideas. We present a full version of Related Work, including baselines, in Appendix A.

Feature attribution via explanation: In the current explanation-based paradigm, feature attribution is performed by (1) training a supervised model before (2) interrogating the model's behavior (*e.g.*, via internals, forward or backward passes, or input perturbations) and inferring class-specific features. Both model-specific (*e.g.*, gradients (Simonyan & Zisserman, 2014), attention maps (Jetley et al., 2018)) and model-agnostic (*e.g.*, SHAP (Lundberg & Lee, 2017), LIME (Ribeiro et al., 2016)) methods of today are either computationally prohibitive (Ancona et al., 2019; Chen et al., 2023) or poor localizers of class-specific features (Arun et al., 2020; Zech et al., 2018; Jain & Wallace, 2019; Zhou et al., 2021b; Bilodeau et al., 2022).

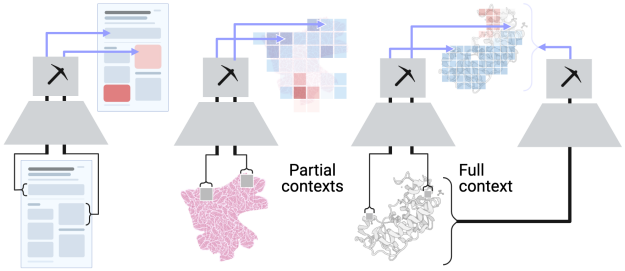

Figure 2: Prospectors are modality-generalizable, amenable to sequences (*e.g.*, text), images (*e.g.*, pathology), and graphs (*e.g.*, protein structures). They can also operate on embeddings from either partial- or full-context encoders.

Modern encoders & context sizes: Most modern encoders for unstructured data operate on **tokens**, or relatively small pieces of a datum, and their representations. Tokens can be user-prespecified and/or constructed by the encoder itself (potentially with help from a tokenizer), where these encoders are respectively referred to as *partial-context* and *full-context* (Figure 2). Due to computational constraints, high-dimensional unstructured data (*e.g.*, gigapixel images) often require user-prespecified tokens (*i.e.*, patches) and partial-context encoders that embed each token (Lu et al., 2023; Huang et al., 2023; Klemmer et al., 2023; Lanusse et al., 2023).

Gradient-based saliency and attention maps have been used to explain partial-context classifiers for high-dimensional unstructured data like gigapixel imagery (Campanella et al., 2019; Chen et al., 2022c). However, studies report low specificity and sensitivity (Machiraju et al., 2022) in part because attribution for the entire datum is built by concatenating attributions across prespecified tokens. Partial-context strategies incorrectly assume prespecified tokens are independent and identically distributed (IID).

Concept-based modeling: The use of **concepts** in ML inherently increases model interpretability by forcing models to reason over unstructured data with respect to said concepts. Concepts themselves can be human-derived, machine-derived (Ghorbani et al., 2019; Talukder et al., 2024), or co-derived with humans in the loop (Lam et al., 2024).

Early concept-based methods examine models' use of concepts in prediction (Kim et al., 2017), while recent methods can also attribute concept importance *in situ* (Ghorbani et al., 2019; Crabbé & van der Schaar, 2022; Brocki & Chung, 2019; Zhou et al., 2018). Sets of concepts can also form a hidden layer, *i.e.*, "bottleneck" (Koh et al., 2020; Kim et al., 2024; Talukder et al., 2024), offering a form of multimodal grounding when concepts are human-derived. More recently, concepts are being assigned to pre-specified tokens

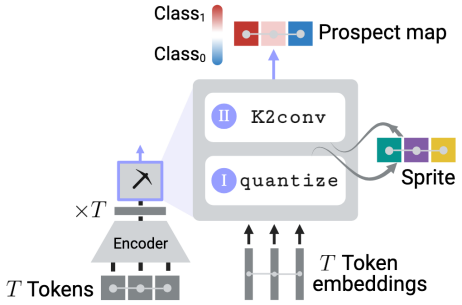

Figure 3: Prospector-equipped encoders produce attribution maps (called "prospect maps") over two layers. Details for fitting and inference are in Sections 3.3, 3.4, and C.3.

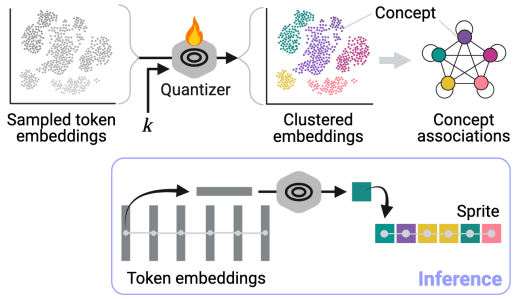

Figure 4: Layer (I) fitting and inference ($K = 5$). Quantized token embeddings define spatially resolved concepts, which together form data sprites.

in high-dimensional data, *e.g.*, subsequences (Talukder et al., 2024) and sentences (Lam et al., 2024). These "spatially resolved" concepts have allowed for hierarchical concept formation when paired with LLMs (Lam et al., 2024).

3 Methods

3.1 Prospection: Attribution sans Explanation

Prospectors are designed to perform few-shot feature attribution for high-dimensional data while meeting challenges (a-c). Instead of explaining end-to-end classifiers, prospectors interface with encoders by adapting their *token embeddings*. Crucially, prospectors foster a form of inductive reasoning over token embeddings to learn class-specific patterns. The use of tokens as the core unit of analysis depends on the key assumption that the equipped encoders have learned adequate distributional semantics in large-scale pretraining. Prospectors can then learn class-specific patterns in small labeled datasets via a simple two-layer module. In layer (I), prospectors transform token embeddings into spatially resolved concepts learned from the training set, constructing a parsimonious "vocabulary" or "codebook" that can be user-verified and/or user-defined. Layer (II) then attributes scores to each token using a novel form of graph convolution that operates on concept frequencies and co-occurrences. The following sections describe the inference and fitting procedures of each layer.

3.2 Preliminaries

To enable any encoder to perform feature attribution regardless of input modality, we first define a generalized language for unstructured data. Any unstructured datum can be represented by a *map graph* $G(\mathcal{V}, \mathcal{E})$ where each vertex $v \in \mathcal{V}$ represents a discrete *token*, or piece of that datum in Euclidean space (Definition C.2). G is composed of $T = |\mathcal{V}|$ tokens. For example, in image data, tokens can be defined as pixels or patches. An edge $e_{i \leftrightarrow j} \in \mathcal{E}$ connects vertex v_i to v_j. Both G's token resolution and token connectivity are defined based on data modality (Figure S1).

Problem setup: Suppose we have a dataset containing map graphs G and binary class labels y. We assume that a class$_y$ graph $G(\mathcal{V}, \mathcal{E})$ contains a set of class$_y$-specific vertices $\mathcal{V}_y \subseteq \mathcal{V}$, with $|\mathcal{V}_y| \geq 1$ (Zhou et al., 2016). One main goal of feature attribution is to locate $\mathcal{V}_y$ in each datum given a set of (G, y) pairs as a training dataset. This task is inherently *coarsely supervised* (Robinson et al., 2020) and is discussed further in Appendix A.

3.3 Prospector Inference

3.3.1 Receiving Token Embeddings

Prospectors receive token embeddings $\mathbf{x}_1 \ldots \mathbf{x}_T$ from an equipped encoder and update map graph G such that each vertex $v_i \in \mathcal{V}$ is featurized by an embedding $\mathbf{x}_i \in \mathbb{R}^d$. This vertex-specific "feature loading" uses the notation: $G[v_i] := \mathbf{x}_i$. Details for partial- and full-context encoders are specified in Appendix C.1.

3.3.2 Layer I: Quantizing Embeddings

Next, prospectors use an encoder's learned semantics to define K spatially resolved concepts $\mathcal{C} = \{1, \ldots, K\}$. This is achieved by quantizing each token embedding $\mathbf{x} \in \mathbb{R}^d$ as a scalar concept $c \in \mathcal{C}$: $c_i = \texttt{quantize}(\mathbf{x}_i)\ \forall i = 1 \ldots T$. When the `quantize` layer (Section 3.4.1) is applied over the full graph G, it is transformed into graph S with the same topology as G, but with categorical vertex features $S[v_i] := c_i\ \forall i$. We refer to S as a *data sprite* due to its low feature dimensionality compared to G (a data compression ratio of d). Intuitively, the heterogeneity of S is parameterized by the choice of K. This layer is depicted in Figure 4.

3.3.3 Layer II: Convolution over Concepts

Prospectors next perform feature attribution using a form of graph convolution over sprite S. This convolution requires a global kernel ω that computes an attribution score $a \in \mathbb{R}$ for each vertex v based on the concepts c_i (*i.e.*, monograms) and co-occurrences c_i, c_j (*i.e.*, skip-bigrams) present within the graph neighborhood defined by receptive field r. The

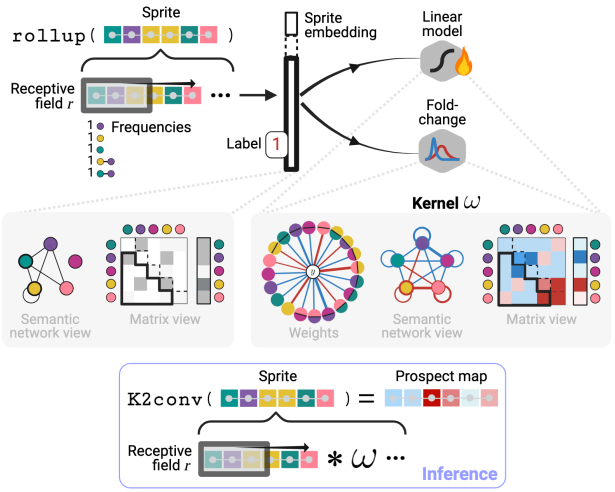

Figure 5: Layer (II) fitting and inference ($K = 5$). Concept frequencies are used to build sprite embeddings, which are used to fit a `K2conv` kernel. Flame icon indicates trainable parameters.

kernel ω serves as a form of *associative memory* and can be conceptualized as a dictionary, scoring each concept monogram or skip-bigram in the combinatorial space $\mathcal{Z} = \mathcal{C} \cup \{\mathcal{C} \otimes \mathcal{C}\}$, where $\otimes$ is the Cartesian product. The global kernel is fit over the training set (Section 3.4.2).

To perform feature attribution at inference time, we apply the fitted kernel over each vertex in a datum to produce a *prospect map* P. P is a map graph with the same topology as G and S but featurized by scalar continuous attribution scores $P[v] := a$. We call this layer `K2conv` in reference to kernel ω's implicit structure (Definition C.5). An attribution score a_i is computed for each vertex v_i in S, where $\mathcal{N}_r$ represents all vertices within the r-neighborhood of v_i (including v_i itself):

$$
\begin{aligned}
P[v_i] \quad &:= \quad a_i = \overbrace{\mathcal{N}_r * \omega}^{\texttt{K2conv}} \\
&= \sum_{\forall v_i \in \mathcal{N}_r} \omega \left\langle \overbrace{S[v_i]}^{c_i} \right\rangle + \sum_{\forall (v_j, v_k) \in \mathcal{N}_r} \omega \left\langle \left(\overbrace{S[v_j]}^{c_j}, \overbrace{S[v_k]}^{c_k} \right) \right\rangle,
\end{aligned}
$$

where $\omega\langle\cdot\rangle$ denotes dictionary lookup. The above expression resembles the energy function for 2D Markov random fields, but adjusted to allow for longer-range dependencies in the second term via skip-bigrams. The resulting prospect map P targets class-specific region $\mathcal{V}_y$ by assigning high absolute positive or negative values to each token. Intuitively, r parameterizes the level of smoothing over P by modulating the number of neighboring tokens used to compute a token's importance. This layer is depicted in Figure 5.

3.4 Prospector Fitting

In our implementation, layers (I) and (II) are fitted separately and sequentially using the procedures below. Further details for each layer are found in Appendix C.3.

3.4.1 Quantizer Fitting (Layer I)

Token embeddings sampled from across the training set are partitioned into K subspaces using an unsupervised algorithm (*e.g.*, K-means clustering). Afterward, each subspace represents a semantic concept $c \in \mathcal{C}$ discovered in the corpus. To reduce computation, clustering can be performed over a representative sample ($> 10^3$) of the token embedding space, randomly sampled without replacement. Fitting is depicted in Figure 4.

3.4.2 Kernel Fitting (Layer II)

Fitting the `K2conv` kernel involves computing the class-attribution weights for each monogram and skip-bigram in $\mathcal{Z}$ across the training set. These weights represent the only learnable parameters of a prospector head. The total number of parameters $|\mathcal{Z}|$ is thus dependent on K and is at maximum (Appendix C.3.2): $|\mathcal{Z}| = 2K + \binom{K}{2}$. The kernel is fit in two steps, as outlined below.

Step 1: Computing frequencies & co-occurrences. For each sprite S in the training set, prospectors first build a datum-level representation in order to learn dataset-wide patterns. This is performed by the `rollup` operator, which traverses S's vertices, tracks concept monogram and skip-bigrams $z_i \in \mathcal{Z}$, and counts their frequencies over all r-neighborhoods. This operation constructs a *sprite embedding* $\mathbf{z} \in \mathbb{R}^{|\mathcal{Z}|}$, which resemble "bag-of-words" vectors with longer-range "skip" interactions. Sprite embeddings are rescaled to account for differences in baseline frequencies (*e.g.*, using TF-IDF (Sparck Jones, 1972)) and thus can be viewed as probabilities: $\mathbb{P}(c_i)$ for monograms and $\mathbb{P}(c_j, c_k)$ for skip-bigrams. The `rollup` operator and this step as a whole are described in Algorithm 1 and Figure S2.

Step 2: Learning kernel weights. Prospectors next use the datum-level sprite embeddings $\mathbf{z}$ to learn a vector $\mathbf{w} \in \mathbb{R}^{|\mathcal{Z}|}$ of class-specific weights for each monogram and skip-bigram across the entire training set. After fitting $\mathbf{w}$, we construct ω as a dictionary mapping each element in $\mathcal{Z}$ to its corresponding weight in $\mathbf{w}$. We implement two approaches to learning weights, which make up the two main prospector variants: a linear classifier $h_{\mathbf{w}}$ and a parameter-free fold-change computation. These variants are discussed further in Appendix C.3.3 and depicted graphically in Figure 5.

Linear classifier: This variant trains a linear classifier $h_{\mathbf{w}}(\mathbf{z}) = \mathbf{w}^{\intercal}\mathbf{z}$ to learn a mapping from $\mathbf{z} \mapsto y$ over the training dataset. The learned coefficients $\mathbf{w}$ then represent the class-specific importance of each index in $\mathbf{z}$. We implement this as a logistic regression with elastic net regularization with the mixing hyperparameter λ.

Fold-change computation: Inspired by bioinformatics (An-

ders & Huber, 2010), this variant involves first computing mean sprite embeddings for each class over the training data. For example, for the negative class, $\overline{\mathbf{z}}_0 = \frac{1}{|\mathcal{D}_0|}\sum_{S^{(i)}\in\mathcal{D}_0}\mathbf{z}^{(i)}$, where $\mathcal{D}_0$ is the subset of the training dataset $(S^{(i)}, y^{(i)})$ for which $y^{(i)} = 0$. This mirrors the "baseline vector" commonly used by popular feature attribution methods (Sundararajan et al., 2017; Bilodeau et al., 2022; Afchar et al., 2021). Then, we compute $\mathbf{w}$ as a fold-changes $\mathbf{w} = \log_2(\overline{\mathbf{z}}_1) - \log_2(\overline{\mathbf{z}}_0)$ and select significant weights using a hypothesis test for independent means. The latter step serves as a form of regularization.

3.5 Meeting Challenges with Intentional Design

Prospectors overcome the limitations of current feature attribution methods by observing the following design principles. Firstly, for (a) data efficiency and few-shot capabilities, prospectors are parameter efficient due to the sole use of concept monograms and skip-bigrams to build its kernel — at maximum only requiring $2K + \binom{K}{2}$ parameters. Both variants for computing importance weights $\mathbf{w}$ are thus data efficient due to their parsimony. Secondly, prospectors are (b) computationally efficient: by operating as an equippable head, prospectors are "plug-in-ready" without encoder retraining (Kim et al., 2017) and or backpropagation. The combination of efficient data structures and modeling primitives such as dictionaries and convolutions allow prospectors to efficiently scale feature attribution to high-dimensional data: namely, linear-time with respect to the tunable number of tokens T. We outline runtime complexity and speed benchmarking in Sections C.4 and D.1. Finally, prospectors achieve (c) improved localization and class-relevance by explicitly training on token embeddings to learn G_y instead of using end-to-end classifiers to identify G_y *post hoc*. We detail other favorable model properties in Appendix C.6.

4 Experiments

4.1 Datasets, Encoders, & Baselines

We evaluate prospectors using three primary tasks, each representing a different data modality (sequences, images, and graphs). Each also poses unique challenges for prospector training and feature attribution: class imbalance (sequences), high input dimensionality with few examples (images), and very coarse supervision (graphs). As is common in scientific and biomedical data, all three datasets are amenable to the *multiple instance assumption* (MIA) — that class_1 data largely resemble class_0 data with the exception of tokens only found in class_1 data (Amores, 2013; Foulds & Frank, 2010). Details for each dataset's construction are shared in Appendix D.5. For each task, we select representative encoders to which we equip prospector heads and relevant baseline attribution methods. We summarize encoders in

Encoder Alias	Architecture	Learning Regime	Training Epochs	Embed Size (d)
MiniLM	MiniLM-L6-v2	KD	✗	384
DeBERTa	DeBERTa-v3-base	SSL	✗	—
tile2vec	ResNet-18	USL	20	128
ViT	ViT/16	WSL	30	1024
CLIP	ViT-B/32	SSL	✗	512
PLIP	ViT-B/32	SSL	✗	512
COLLAPSE	GVP-GNN	SSL	✗	512
ESM2	t33_650M_UR50D	SSL	✗	1028
AA	—	—	✗	21

Table 1: Prospector-equipped encoders in descending order by modality: sequences, images, and graphs. Learning regimes are knowledge distillation (KD), unsupervised learning (USL), weakly supervised learning (WSL), and self-supervised learning (SSL). Pre-training denoted by ✗. Non-applicability denoted by "—".

Table 1 (and Appendix D), baselines in Appendix D.9, and ruled-out baselines in Appendix B.

For both baselines and prospectors, we perform a grid-search over tunable hyperparameters. Due to the MIA, the best models were selected based on their ability to localize ground truth class_1 regions in the training set, since these were not seen by prospectors during training. We use a sequential ranking criteria over four token-level metrics: precision, dice coefficient, Matthews correlation coefficient, and AUPRC. Details of hyperparameter tuning and model selection are found in Appendix D.2 and D.3. The results in the remainder of this paper present the localization AUPRC and average precision (AP) over a set of thresholds, for class_1 regions in our held-out test data.

Sequences (1D): key sentence retrieval in text documents. Retrieval is an important task in language modeling that provides in-text answers to user queries. For this task, we use the WikiSection (Arnold et al., 2019) benchmark dataset created for paragraph-level classification. We repurpose WikiSection to assess the ability to retrieve target sentences specific to a queried class. We specifically use the "genetics" section label as a query, and class_1 data are defined as documents in the English-language "disease" subset that contain this section label. Our goal is to identify sentences that contain genetics-related information given only coarse supervision from document-level labels. After preprocessing the pre-split dataset, our dataset contained 2513 training examples (2177 in class_0, 336 in class_1) and 712 test examples. The relationship between sentences in each document is represented as a graph with 2-hop connectivity (Figure S1).

Encoders & baselines: We assess two pretrained language models, MiniLM (Wang et al., 2020) and DeBERTa (He et al., 2020; 2021), used at partial-context. While DeBERTA is an off-the-shelf LLM for zero-shot classification (ZSC) and natural language inference (NLI), MiniLM is a sentence and paragraph embedding model — thus requiring prospectors to perform feature attribution at the sentence-level.

For baselines attribution methods, we present a mix of (1) su-

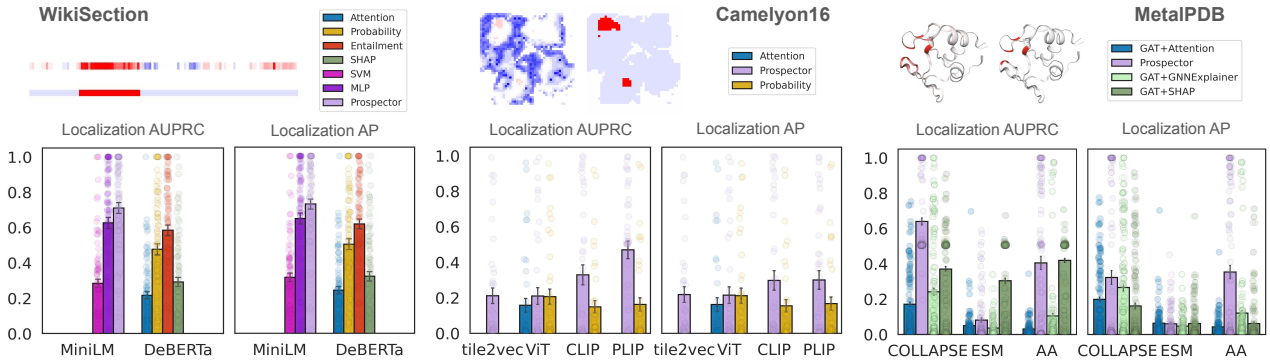

Figure 6: Prospectors vs. baselines for sequences (left), images (middle), and graphs (right). Dots represent performance on individual test-set examples, while bars represent means with whiskers as standard errors. Numerical results are found in Appendix D.10.

pervised heads and (2) off-the-shelf LLM inference. Firstly, supervised heads train on token-level class labels to identify class-specific sentences in testing. Specifically, we train a multi-layer perception (MLP) on labeled token embeddings and a one-class support vector machine (SVM) trained solely on class_0 token embeddings. In the latter case, we perform novelty detection to identify class_1 tokens. While not traditional explanation methods, the MLP and SVM heads are given a large advantage as semi- and fully supervised baselines (as opposed to prospectors, which are coarsely supervised at the datum-level). For LLM inference, we used DeBERTA to output sentence-level ZSC probabilities (*i.e.*, logits), NLI entailment scores, NLI entailment attention, and pooled Shapley values for ZSC. Implementation details are listed in Appendix D.9.

Images (2D): tumor localization in pathology slides. Identifying tumors is an important task in clinical pathology, where manual annotation is standard practice. We evaluate prospectors on Camelyon16 (Ehteshami Bejnordi et al., 2017), a benchmark of gigapixel pathology images, each presenting either healthy tissue or cancer metastases. All images are partitioned into prespecified 224×224 patch tokens and filtered for foreground tissue regions. After preprocessing the pre-split dataset, our dataset contained 218 images for training (111 for class_0 and 107 for class_1) and 123 images for testing. The relationship between patches in each image is represented as a graph using up to 8-way connectivity (Figure S1).

Encoders & baselines: We equip prospectors to four encoders: tile2vec (Jean et al., 2019), ViT (Dosovitskiy et al., 2020), CLIP (Radford et al., 2021), and PLIP (Huang et al., 2023). The first two encoders are trained with partial context, where tile2vec is unsupervised while ViT is weakly supervised with image-level label inheritance (Machiraju et al., 2022). Details on encoder training are provided in Appendix D. CLIP serves as a general-domain vision-language foundation model (VLM) and PLIP serves as a domain-specific version of CLIP for pathology images. Both VLM encoders are pretrained and used for partial-context inference on prespecified image patches. We choose two popular and computationally feasible explanation-based attribution baselines (Section 2): concatenated mean attention (Chen et al., 2022c) for ViT and concatenated prediction probability (Campanella et al., 2019; Machiraju et al., 2022; Halicek et al., 2019) for ViT, CLIP, and PLIP.

Graphs (3D): binding site identification in protein structures. Many proteins rely on binding to metal ions in order to perform their biological functions, such as reaction catalysis in enzymes, and identifying the binding-specific amino acids is important for engineering and design applications. We generated a dataset of metal binding sites in enzymes using MetalPDB (Putignano et al., 2018), a curated dataset derived from the Protein Data Bank (PDB) (Berman et al., 2002). Focusing on zinc, the most common metal in the PDB, we generate a gold standard dataset of 610 zinc-binding (class_1) enzymes and 653 non-binding (class_0) enzymes (see Appendix D.5.3). Each protein structure is defined using the positions of its atoms in 3D space and subdivided into tokens representing amino acids (a.k.a. "residues"). The relationship between residues is represented as a graph with edges defined by inter-atomic distance (Figure S1). This task is particularly challenging due to potentially overlapping class-specific features (*i.e.*, proteins of both classes are metal-binders), highly heterogeneous background data (proteins in train and test sets adopt a wide variety of structural folds), and relatively small target regions, making this an example of a "needle-in-the-haystack" task (Pawlowski et al., 2019).

Encoders & baselines: We apply prospector heads to three encoders: COLLAPSE, an FM which produces embeddings of the local 3D structure surrounding each residue (Derry & Altman, 2023); ESM2, a protein LLM which produces embeddings for each residue based on 1D sequence (Lin et al., 2023); and a simple amino acid encoder (AA), where each residue is one-hot encoded by amino acid identity. By construction, ESM2 is a full-context encoder while COLLAPSE and AA are partial-context encoders. We present three baselines built on top of a supervised GAT (Veličković et al.,

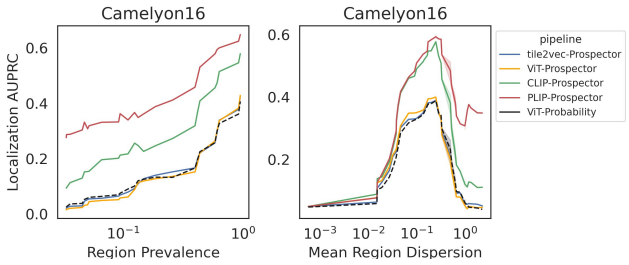

Figure 7: Robustness analysis for Camelyon16 data: prospector and top baseline performance with respect to region characteristics.

2017) classifier head (trained on protein-level labels) to identify binding residues: Attention, Shapley values (SHAP), and GNNExplainer (Ying et al., 2019). Implementation details are listed in Appendix D.9.

4.2 Results

Prospectors outperform baseline attribution methods in region localization and generalize across data modalities. In all tasks, prospectors achieve higher AUPRC and AP than baseline methods, often with large improvements (Figure 6). For text retrieval, we improve mean test-set AUPRC to 0.711 from 0.626 (*i.e.*, 8.5-point gain) with the top supervised baseline (MLP head) and from 0.584 with the top LLM inference baseline (NLI entailment) — in summary, MiniLM with an equipped prospector head is able to outperform DeBERTa's baselines by 12.7 points in AUPRC despite being 5× smaller in size and with relatively limited pretraining (Table S5). We also observe improved localization over baselines for Camelyon16 (26.3 points in AUPRC and 8.8 points in AP) and MetalPDB (22.0 points in AUPRC and 8.8 points in AP). For the MetalPDB dataset, the optimal methods tend to exhibit bimodal performance, with almost perfect predictions for a subset of the test dataset (particularly cysteine-dependent binding motifs, see Figure 9) and poor performance on other subsets, resulting in the clustering of points around 0.5 and 1.0 AUPRC. This behavior suggests that AP more clearly reflects task performance, highlighting the ability of prospectors to identify small conserved binding patterns.

The choice of encoder is key to optimal prospector performance. While prospectors overall improve localization performance over baselines regardless of the chosen encoder, the performance gain is maximized by choosing domain-specialized encoders for each dataset. For Camelyon16 and MetalPDB, the combination of prospectors with FM encoders (CLIP, PLIP, COLLAPSE) showed the strongest localization results, as shown in Figure 6. Among FMs, the best-performing encoders are those with the most task-specificity — PLIP has a domain advantage by virtue of being a CLIP-style encoder trained on pathology images, while COLLAPSE accounts for complex 3D atomic geometry rather than simply amino acid sequence (as in ESM2) or one-hot encoding (AA). Interestingly, we note that the AA encoder presents an exception to encoder generalization, supporting that prospectors themselves can identify salient patterns with rudimentary encoder semantics. This is likely due to the fact that many zinc-binding motifs rely on atomic coordination by three to four cysteine residues, which are otherwise rarely found in such arrangements. For tasks which require the detection of less amino acid-dependent structural patterns, we expect the COLLAPSE encoder to result in optimal prospector performance.

Prospectors are robust to coarse supervision. Next, we explore the relationship between the properties of class-specific regions and localization performance. To characterize class-specific regions, we compute two metrics acting as proxies for coarse supervision (Section 3.2): *region prevalence* (# class_1 tokens / # tokens) and *mean region dispersion* (# connected components / mean component size). For Camelyon16, we plot the relationship between test-set AUPRC and both metrics in Figure 7. Full results over all datasets are presented in Appendix D.11. For each plot, we also display the top baseline method.

Firstly, we observe that most encoders exhibit a positive correlation between region prevalence and localization AUPRC across all modalities. However, some encoders are particularly robust to region prevalance and achieve high AURPC despite low prevalence (MiniLM, PLIP, COLLAPSE), and prospectors are consistently more robust than top baselines over all data modalities. Secondly, mean region dispersion and localization performance (both AUPRC and AP) demonstrate a *parabolic* relationship — indicating that some level of dispersion is needed for detectable regions, while too much dispersion makes the task challenging. These results recapitulate each task's challenges: the pathology task contains a wide range of dispersion values, while the protein task contains the lowest levels of prevalence and highest levels of dispersion (Appendix C.6). Despite these task differences, prospector-equipped FMs demonstrate an high levels of robustness to coarse supervision across modalities.

Prospectors' sprite embeddings and kernels are interpretable and enable internal visualization. In addition to improved localization performance, prospectors are inherently *interpretable* because their parameters provide insights into invariant class-specific patterns. Prospect maps visualize the feature attribution outputs in the input token space — but importantly, these maps can be further contextualized by visualizing prospector internals themselves. Due to the use of learned semantic concepts, the global convolutional kernel can be represented as a semantic network or as a heatmap (Appendix C.2), along with each input example as it passes through layers of the prospector head.

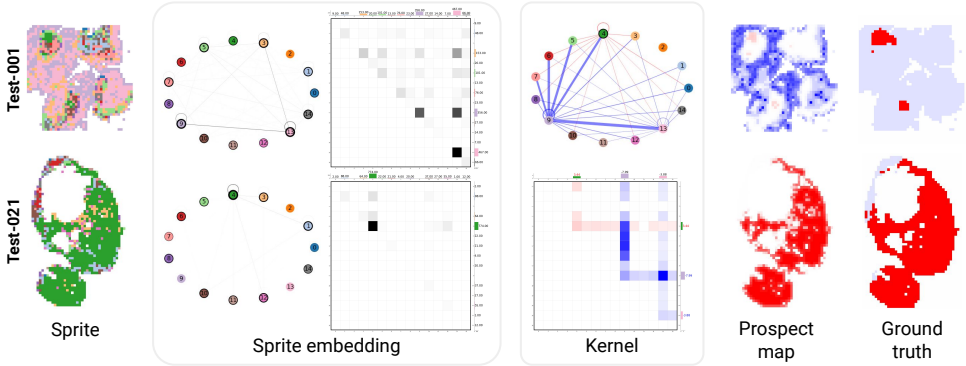

Figure 8: Prospector visualization for pathology, using the top PLIP configuration (Table S4). Visualizations are shown for two test-set examples, from left to right: data sprites; sprite embeddings viewed as semantic networks and heatmaps (Appendix C.2), where line thicknesses or cell shade reflect monogram or skip-bigram count; the kernel viewed as a semantic network and heatmap, where line thickness and cell intensity reflect learned weights; prospect map, with vertex attribution scores mapped back onto tokens in original data; and ground-truth class-specific regions in the image (in red). Sprites and sprite embeddings are colored by the $K = 15$ learned concepts. Kernel weights and prospect maps are colored red and blue to reflect class$_1$-specific and class$_0$-specific associations, respectively.

We illustrate this interpretability for pathology images (Figure 8) and protein structure (Figure 9) using two test-set examples. We first visualize data sprites, which reflect the learned concepts mapped onto data inputs (from layer (I)). By analyzing semantic concepts on the data sprite, it is possible to assign domain-specific meaning to each concept. Additionally, by visualizing concept and co-occurrence frequencies in the sprite embedding, we can identify over- or under-represented patterns within each input. By visualizing the global kernel, which captures dataset-wide concept associations and their correlations with class labels, it is possible to cross-reference between the sprite and the class-specific regions of the resulting prospect map. The ability to visualize the internals of a prospector head in terms of concepts facilitates human-in-the-loop model development and the incorporation of domain knowledge, a major advantage relative to "black box" models.

Prospector kernels allow for parsimony to find "hub" concepts. Our pathology visualization (Figure 8) demonstrates a kernel with "hub," or densely connected and highly predictive concepts: concept #4 is indicative of class$_1$ while concept #9 is indicative of class$_0$. Such kernels demonstrate how prospectors *do not detract* from the rich semantics offered by FM encoders like PLIP for pathology data.

Prospection is robust to concept distributional shifts. Visualizing kernels for protein structures outlines prevalent class$_1$-specific concepts in training data (*e.g.*, concepts #7, #17) that are rare in the test set but nonetheless are critical for classification. Despite their low frequency, top prospectors achieved performant localization for this task. The distributional shift between train and test set is a likely explanation for the bimodal localization performance on this task, and suggests that improvements to kernel design and fitting (*e.g.*, feature scaling and choice of K) along with constructing optimally representative training datasets (*e.g.*, for a more varied class$_0$) would improve prospector performance on more difficult data subsets.

Sprite embeddings also carry class signal. Further analysis of learned parameters can also help to better understand the nature of discovered patterns. For example, there may be more than one pattern which results in a particular class label, and differentiating examples that exhibit each pattern can uncover mechanistic subgroups of the data. To demonstrate this, we hierarchically cluster the sample-level sprite embeddings in the MetalPDB test set. This identified two major subgroups of zinc binding sites (Figure S7) defined by the number of cysteine residues coordinating the bound ion. One subgroup is enriched for proteins which contain four coordinating cysteines, while in the other there are one or more histidine residues involved in the binding interaction. Figure 9 shows an example from each cluster, including a visualization of the zinc-binding site on the far right. This finding recapitulates known subtypes of zinc binding motifs (Wu et al., 2010), and more broadly demonstrates the potential for prospectors to discover biological mechanisms when applied to less well-studied phenomena.

5 Discussion & Conclusion

This work presents prospector heads, encoder-equippable modules for (a) data-efficient, (b) time-efficient, and (c) performant feature attribution. We show that prospectors are both modality-generalizable and encoder-agnostic with particularly dominant performance when equipped to domain-specialized FMs. Finally, we show that prospectors are interpretable through their use of concept-based kernels.

Prospectors' improved localization performance over explanation-based baselines calls into question the underlying assumption of explanations themselves: that end-to-end classifiers implicitly "segment" data in the input token space

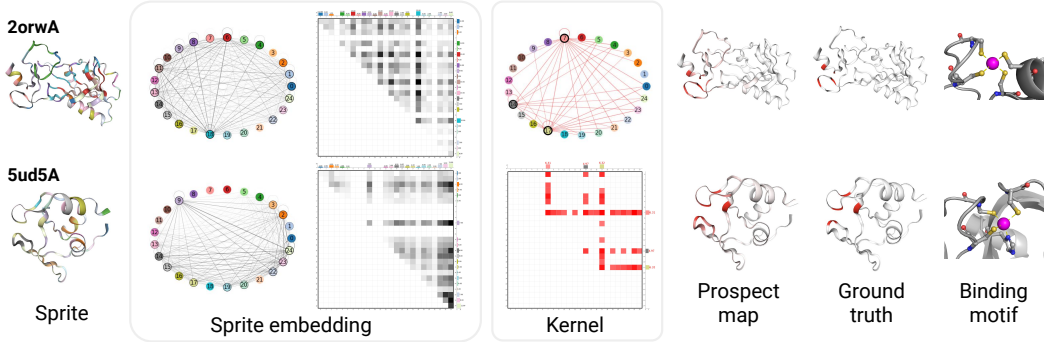

Figure 9: Prospector visualization for protein data, using the top COLLAPSE configuration (Table S4). We show the same five visualizations as before, as well as a visualization of the atomic configuration of the zinc binding site to illustrate the binding motifs discovered by sprite embedding clustering (Figure S7). Whole proteins are visualized as cartoons instead of graphs for clarity.

en route to making class predictions, and that these "segmentations" can be extracted *post hoc*. Our results suggest that using machine-derived concepts and modeling class-specific associations directly in the input token space helps to avoid such modeling assumptions.

We believe a key driver of prospectors' performance is the combination of token-level representations with the local inductive bias provided by convolution. This combination fosters a form of inductive reasoning through "token mixing" and kernel construction. Several other aspects of prospector design draw inspiration from ideas across ML research (Appendix A.3), giving insights into their performance characteristics. Our results suggest that FMs in particular contain strong distributional semantics which yield precise feature attribution even with partial-context encoders and coarse levels of supervision. In other words, FMs (in tandem with quantization) remove the burden of long-context reasoning by reducing input data to mosaics of concepts (*i.e.*, sprites). Prospectors can thus functionally operate over long-range dependencies even with a local inductive bias. This claim of capturing short- and long-range dependencies between tokens is backed by prospectors' localization robustness to region prevalence and dispersion. Additionally, because domain-specific FMs do improve performance when they are available (*e.g.*, PLIP vs. CLIP), we hypothesize that as FMs continue to improve and be adapted to new applications and data modalities, so will the utility of prospectors across diverse domains.

Prospectors are flexible and modular by design, enabling not only variable encoders but also simple changes in their fitting. Of the two variants we fitted, the non-trainable fold-change variant was superior for almost all evaluated settings (Appendix D.3). This may be because the variant explicitly learns dataset-wide concept associations and deviations from a class_0 "baseline vector" (Sundararajan et al., 2017; Bilodeau et al., 2022; Afchar et al., 2021) — which closely reflects the MIA (Section 4.1). It is possible that different kernel fitting methods may be better suited to detecting different types of class-specific patterns, but further investigation is needed to explore this question.

One limitation of this work is the lack of sensitivity analysis for all design choices and hyperparameters. For example, due to time and compute constraints, we relied on domain knowledge to select token resolution and connectivity for each task instead of testing their impacts on performance. Furthermore, we did not study the choice of clustering method nor embedding sample size in the quantization step, and we limited our experimentation to open-source encoders only. Future work involves *Pytorch* implementation for GPU acceleration, enabling kernels to learn higher-order n-grams, adding new variants for kernel fitting, deployments on varied data modalities, and exploring prospector utility with frontier non-Transformer architectures (*e.g.*, state-space models (Gu et al., 2021; Poli et al., 2023a) and their attention hybrids (Poli et al., 2023b)) and API-locked LLMs (Bommasani et al., 2023).

We anticipate many potential use cases for prospectors, particularly in tandem with vector databases and in other compound AI systems and agentic workflows (Zaharia et al.). One particular use case is to screen or classify data with FMs equipped with performant classifier heads (Swanson et al., 2022), and then swap in prospector heads when feature attribution is required. This process can enable users to investigate multiple class labels (*e.g.*, scientific phenomena) without encoder retraining. Another use case is to use prospector-generated attributions to train downstream rationale models (Jain & Wallace, 2019; Chen et al., 2022a; Yang et al., 2023; Bujel et al., 2023). In general, we believe that prospectors expands the toolkit for improving the transparency and utility of large FMs, high-dimensional data, and large-scale datasets — ultimately inspiring new few-shot inference modes for FMs. For scientific and biomedical applications, including in data-scarce settings, prospectors have the potential to provide mechanistic insights and discover phenomena in complex data (Wang et al., 2023).

Impact Statement

Trust and safety considerations are increasingly important as AI becomes an increasingly prominent part of high-impact disciplines such as science and biomedicine. This concern is particularly relevant for large "black box" foundation models. The goal of this work is to provide a new approach to feature attribution for large models and complex datasets to improve transparency of AI systems. It is important to note that that our method is specifically not designed to be an *explanation* of a model's reasoning, and any feature attributions made by prospector heads should be carefully interpreted by the user in the context of the data modality.

Code Availability

Our code is made available at: https://github.com/gmachiraju/K2.

Acknowledgements

We thank Mayee Chen, Simran Arora, Eric Nguyen, Silas Alberti, Ben Viggiano, and B. Anana for their helpful feedback. Gautam Machiraju is supported by the Stanford Data Science scholarship program. Neel Guha is supported by the Stanford Interdisciplinary Graduate Fellowship and the HAI Graduate Fellowship.

We gratefully acknowledge the support of NIH under No. U54EB020405 (Mobilize), GM102365, LM012409, 1R01CA249899, and 1R01AG078755; NSF under Nos. CCF2247015 (Hardware-Aware), CCF1763315 (Beyond Sparsity), CCF1563078 (Volume to Velocity), and 1937301 (RTML); US DEVCOM ARL under Nos. W911NF-23-2-0184 (Long-context) and W911NF-21-2-0251 (Interactive Human-AI Teaming); ONR under Nos. N000142312633 (Deep Signal Processing); Stanford HAI under No. 247183; NXP, Xilinx, LETI-CEA, Intel, IBM, Microsoft, NEC, Toshiba, TSMC, ARM, Hitachi, BASF, Accenture, Ericsson, Qualcomm, Analog Devices, Google Cloud, Salesforce, Total, the HAI-GCP Cloud Credits for Research program, the Stanford Data Science Initiative (SDSI), [STANFORD/STUDENT-SPECIFIC FELLOWSHIPS], and members of the Stanford DAWN project: Meta, Google, and VMWare. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views, policies, or endorsements, either expressed or implied, of NIH, ONR, or the U.S. Government.

Finally, figures 1, 2, 3, 4, 5, S1, and S2 were created with BioRender.com.

References

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., and Kim, B. Sanity checks for saliency maps. *arXiv [cs.CV]*, October 2018.

Adebayo, J., Muelly, M., Abelson, H., and Kim, B. Post hoc explanations may be ineffective for detecting unknown spurious correlation. *arXiv [cs.LG]*, December 2022.

Afchar, D., Hennequin, R., and Guigue, V. Towards rigorous interpretations: a formalisation of feature attribution. *arXiv [cs.LG]*, April 2021.

Akyürek, E., Wang, B., Kim, Y., and Andreas, J. In-Context language learning: Architectures and algorithms. *arXiv [cs.CL]*, January 2024.

Alain, G. and Bengio, Y. Understanding intermediate layers using linear classifier probes. *arXiv [stat.ML]*, October 2016.

Amores, J. Multiple instance classification: Review, taxonomy and comparative study. *Artif. Intell.*, 201:81–105, August 2013.

Ancona, M., Öztireli, C., and Gross, M. Explaining deep neural networks with a polynomial time algorithm for shapley values approximation. *arXiv [cs.LG]*, March 2019.

Anders, S. and Huber, W. Differential expression analysis for sequence count data. *Genome Biol.*, 11(10):R106, October 2010.

Arnold, S., Schneider, R., Cudré-Mauroux, P., Gers, F. A., and Löser, A. SECTOR: A neural model for coherent topic segmentation and classification. *arXiv [cs.CL]*, February 2019.

Arun, N., Gaw, N., Singh, P., Chang, K., Aggarwal, M., Chen, B., Hoebel, K., Gupta, S., Patel, J., Gidwani, M., Adebayo, J., Li, M. D., and Kalpathy-Cramer, J. Assessing the (un)trustworthiness of saliency maps for localizing abnormalities in medical imaging. *bioRxiv*, July 2020.

Bairoch, A. The ENZYME database in 2000. *Nucleic Acids Res.*, 28(1):304–305, January 2000.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. Automatic differentiation in machine learning: a survey. *J. Mach. Learn. Res.*, 18(1):5595–5637, January 2017.

Belinkov, Y. Probing classifiers: Promises, shortcomings, and advances. *arXiv [cs.CL]*, February 2021.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The Long-Document transformer. *arXiv [cs.CL]*, April 2020.

Berman, H. M., Battistuz, T., Bhat, T. N., Bluhm, W. F., Bourne, P. E., Burkhardt, K., Feng, Z., Gilliland, G. L., Iype, L., Jain, S., Fagan, P., Marvin, J., Padilla, D., Ravichandran, V., Schneider, B., Thanki, N., Weissig, H., Westbrook, J. D., and Zardecki, C. The protein data bank. *Acta Crystallogr. D Biol. Crystallogr.*, 58(Pt 61): 899–907, June 2002.

Bietti, A., Cabannes, V., Bouchacourt, D., Jegou, H., and Bottou, L. Birth of a transformer: A memory viewpoint. *arXiv [stat.ML]*, June 2023.

Bilodeau, B., Jaques, N., Koh, P. W., and Kim, B. Impossibility theorems for feature attribution. *arXiv [cs.LG]*, December 2022.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., Donahue, C., Doumbouya, M., Durmus, E., Ermon, S., Etchemendy, J., Ethayarajh, K., Fei-Fei, L., Finn, C., Gale, T., Gillespie, L., Goel, K., Goodman, N., Grossman, S., Guha, N., Hashimoto, T., Henderson, P., Hewitt, J., Ho, D. E., Hong, J., Hsu, K., Huang, J., Icard, T., Jain, S., Jurafsky, D., Kalluri, P., Karamcheti, S., Keeling, G., Khani, F., Khattab, O., Kohd, P. W., Krass, M., Krishna, R., Kuditipudi, R., Kumar, A., Ladhak, F., Lee, M., Lee, T., Leskovec, J., Levent, I., Li, X. L., Li, X., Ma, T., Malik, A., Manning, C. D., Mirchandani, S., Mitchell, E., Munyikwa, Z., Nair, S., Narayan, A., Narayanan, D., Newman, B., Nie, A., Niebles, J. C., Nilforoshan, H., Nyarko, J., Ogut, G., Orr, L., Papadimitriou, I., Park, J. S., Piech, C., Portelance, E., Potts, C., Raghunathan, A., Reich, R., Ren, H., Rong, F., Roohani, Y., Ruiz, C., Ryan, J., Ré, C., Sadigh, D., Sagawa, S., Santhanam, K., Shih, A., Srinivasan, K., Tamkin, A., Taori, R., Thomas, A. W., Tramèr, F., Wang, R. E., Wang, W., Wu, B., Wu, J., Wu, Y., Xie, S. M., Yasunaga, M., You, J., Zaharia, M., Zhang, M., Zhang, T., Zhang, X., Zhang, Y., Zheng, L., Zhou, K., and Liang, P. On the opportunities and risks of foundation models. *arXiv [cs.LG]*, August 2021.

Bommasani, R., Klyman, K., Longpre, S., Kapoor, S., Maslej, N., Xiong, B., Zhang, D., and Liang, P. The foundation model transparency index. *arXiv [cs.LG]*, October 2023.

Brocki, L. and Chung, N. C. Concept saliency maps to visualize relevant features in deep generative models. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pp. 1771–1778. IEEE, December 2019.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv [cs.LG]*, April 2021.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are Few-Shot learners. *arXiv [cs.CL]*, May 2020.

Bujel, K., Caines, A., Yannakoudakis, H., and Rei, M. Finding the needle in a haystack: Unsupervised rationale extraction from long text classifiers. *arXiv [cs.CL]*, March 2023.

Campanella, G., Hanna, M. G., Geneslaw, L., Miraflor, A., Silva, V. W. K., Busam, K. J., Brogi, E., Reuter, V. E., Klimstra, D. S., and Fuchs, T. J. Clinical-grade computational pathology using weakly supervised deep learning on whole slide images. *Nat. Med.*, 25(8):1301, August 2019.

Chen, H., He, J., Narasimhan, K., and Chen, D. Can rationalization improve robustness? *arXiv [cs.CL]*, April 2022a.

Chen, H., Lundberg, S. M., and Lee, S.-I. Explaining a series of models by propagating shapley values. *Nat. Commun.*, 13(1):4512, August 2022b.

Chen, P., Li, Q., Biaz, S., Bui, T., and Nguyen, A. gScoreCAM: What objects is CLIP looking at? In *Computer Vision – ACCV 2022*, pp. 588–604. Springer Nature Switzerland, 2023.

Chen, R. J., Chen, C., Li, Y., Chen, T. Y., Trister, A. D., Krishnan, R. G., and Mahmood, F. Scaling vision transformers to gigapixel images via hierarchical Self-Supervised learning. *arXiv [cs.CV]*, June 2022c.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv [cs.LG]*, April 2019.

Crabbé, J. and van der Schaar, M. Concept activation regions: A generalized framework for Concept-Based explanations. *arXiv [cs.LG]*, September 2022.

Derry, A. and Altman, R. B. COLLAPSE: A representation learning framework for identification and characterization of protein structural sites. *Protein Sci.*, 32(2):e4541, February 2023.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer,

M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv [cs.CV]*, October 2020.

Ehteshami Bejnordi, B., Veta, M., Johannes van Diest, P., van Ginneken, B., Karssemeijer, N., Litjens, G., van der Laak, J. A. W. M., the CAMELYON16 Consortium, Hermsen, M., Manson, Q. F., Balkenhol, M., Geessink, O., Stathonikos, N., van Dijk, M. C., Bult, P., Beca, F., Beck, A. H., Wang, D., Khosla, A., Gargeya, R., Irshad, H., Zhong, A., Dou, Q., Li, Q., Chen, H., Lin, H.-J., Heng, P.-A., Haß, C., Bruni, E., Wong, Q., Halici, U., Öner, M. Ü., Cetin-Atalay, R., Berseth, M., Khvatkov, V., Vylegzhanin, A., Kraus, O., Shaban, M., Rajpoot, N., Awan, R., Sirinukunwattana, K., Qaiser, T., Tsang, Y.-W., Tellez, D., Annuscheit, J., Hufnagl, P., Valkonen, M., Kartasalo, K., Latonen, L., Ruusuvuori, P., Liimatainen, K., Albarqouni, S., Mungal, B., George, A., Demirci, S., Navab, N., Watanabe, S., Seno, S., Takenaka, Y., Matsuda, H., Ahmady Phoulady, H., Kovalev, V., Kalinovsky, A., Liauchuk, V., Bueno, G., Fernandez-Carrobles, M. M., Serrano, I., Deniz, O., Racoceanu, D., and Venâncio, R. Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer. *JAMA*, 318(22):2199–2210, December 2017.

Fey, M. and Lenssen, J. E. Fast graph representation learning with PyTorch geometric. *arXiv [cs.LG]*, March 2019.

Foulds, J. and Frank, E. A review of Multi-Instance learning assumptions. *The Knowledge Engineering Review*, 0(0): 1–24, 2010.

Ghorbani, A., Wexler, J., Zou, J., and Kim, B. Towards automatic concept-based explanations. *arXiv [stat.ML]*, February 2019.

Gondal, M. W., Gast, J., Ruiz, I. A., Droste, R., Macri, T., Kumar, S., and Staudigl, L. Domain aligned CLIP for few-shot classification. *arXiv [cs.CV]*, November 2023.

Graves, A., Jaitly, N., and Mohamed, A.-R. Hybrid speech recognition with deep bidirectional LSTM. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 273–278. IEEE, December 2013.

Gu, A., Goel, K., and Ré, C. Efficiently modeling long sequences with structured state spaces. *arXiv [cs.LG]*, October 2021.

Halicek, M., Shahedi, M., Little, J. V., Chen, A. Y., Myers, L. L., Sumer, B. D., and Fei, B. Head and neck cancer detection in digitized Whole-Slide histology using convolutional neural networks. *Sci. Rep.*, 9(1):14043, October 2019.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *arXiv [cs.CV]*, December 2015.

He, P., Liu, X., Gao, J., and Chen, W. DeBERTa: Decoding-enhanced BERT with disentangled attention. *arXiv [cs.CL]*, June 2020.

He, P., Gao, J., and Chen, W. DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled embedding sharing. *arXiv [cs.CL]*, November 2021.

Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. U. S. A.*, 79(8):2554–2558, April 1982.

Hopfield, J. J. Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Natl. Acad. Sci. U. S. A.*, 81(10):3088–3092, May 1984.

Huang, Z., Bianchi, F., Yuksekgonul, M., Montine, T., and Zou, J. Leveraging medical twitter to build a visual–language foundation model for pathology AI. *bioRxiv*, pp. 2023.03.29.534834, April 2023.

Jain, S. and Wallace, B. C. Attention is not explanation. *arXiv [cs.CL]*, February 2019.

Jakubik, J., Roy, S., Phillips, C. E., Fraccaro, P., Godwin, D., Zadrozny, B., Szwarcman, D., Gomes, C., Nyirjesy, G., Edwards, B., Kimura, D., Simumba, N., Chu, L., Karthik Mukkavilli, S., Lambhate, D., Das, K., Bangalore, R., Oliveira, D., Muszynski, M., Ankur, K., Ramasubramanian, M., Gurung, I., Khallaghi, S., Hanxi, Li, Cecil, M., Ahmadi, M., Kordi, F., Alemohammad, H., Maskey, M., Ganti, R., Weldemariam, K., and Ramachandran, R. Foundation models for generalist geospatial artificial intelligence. *arXiv [cs.CV]*, October 2023.

Javed, S., Juyal, D., Padigela, H., Taylor-Weiner, A., Yu, L., and Prakash, A. a. Additive MIL: Intrinsically interpretable multiple instance learning for pathology. *Adv. Neural Inf. Process. Syst.*, 2022.

Jean, N., Wang, S., Samar, A., Azzari, G., Lobell, D., and Ermon, S. Tile2Vec: Unsupervised representation learning for spatially distributed data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3967–3974, July 2019.

Jethani, N., Sudarshan, M., Covert, I., Lee, S.-I., and Ranganath, R. FastSHAP: Real-Time shapley value estimation. *arXiv [stat.ML]*, July 2021.

Jetley, S., Lord, N. A., Lee, N., and Torr, P. H. S. Learn to pay attention. *arXiv [cs.CV]*, April 2018.

Kalibhat, N., Bhardwaj, S., Bruss, B., Firooz, H., Sanjabi, M., and Feizi, S. Identifying interpretable subspaces in image representations. *arXiv [cs.CV]*, July 2023.

Karimi, A.-H., Muandet, K., Kornblith, S., Schölkopf, B., and Kim, B. On the relationship between explanation and prediction: A causal view. *arXiv [cs.LG]*, December 2022.

Keles, F. D., Wijewardena, P. M., and Hegde, C. On the computational complexity of Self-Attention. *arXiv [cs.LG]*, September 2022.

Khosla, S., Zhu, Z., and He, Y. Survey on Memory-Augmented neural networks: Cognitive insights to AI applications. *arXiv [cs.AI]*, December 2023.

Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., and Sayres, R. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). *arXiv [stat.ML]*, November 2017.

Kim, C., Gadgil, S. U., DeGrave, A. J., Omiye, J. A., Cai, Z. R., Daneshjou, R., and Lee, S.-I. Transparent medical image AI via an image-text foundation model grounded in medical literature. *Nat. Med.*, 30(4):1154–1165, April 2024.

Klemmer, K., Rolf, E., Robinson, C., Mackey, L., and Rußwurm, M. SatCLIP: Global, General-Purpose location embeddings with satellite imagery. *arXiv [cs.CV]*, November 2023.

Koh, P. W., Nguyen, T., Tang, Y. S., Mussmann, S., Pierson, E., Kim, B., and Liang, P. Concept bottleneck models. *arXiv [cs.LG]*, July 2020.

Kohonen, T. Correlation matrix memories. *IEEE Trans. Comput.*, C-21(4):353–359, April 1972.

Kuo, W., Cui, Y., Gu, X., Piergiovanni, A. J., and Angelova, A. F-VLM: Open-Vocabulary object detection upon frozen vision and language models. *arXiv [cs.CV]*, September 2022.

Lam, M. S., Teoh, J., Landay, J., Heer, J., and Bernstein, M. S. Concept induction: Analyzing unstructured text with High-Level concepts using LLooM. *arXiv [cs.HC]*, April 2024.

Lanusse, F., Parker, L., Golkar, S., Cranmer, M., Bietti, A., Eickenberg, M., Krawezik, G., McCabe, M., Ohana, R., Pettee, M., Blancard, B. R.-S., Tesileanu, T., Cho, K., and Ho, S. AstroCLIP: Cross-Modal Pre-Training for astronomical foundation models. *arXiv [astro-ph.IM]*, October 2023.

Lin, Z., Akin, H., Rao, R., Hie, B., Zhu, Z., Lu, W., Smetanin, N., Verkuil, R., Kabeli, O., Shmueli, Y., Dos Santos Costa, A., Fazel-Zarandi, M., Sercu, T., Candido, S., and Rives, A. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, March 2023.

Liu, Q. and Mukhopadhyay, S. Unsupervised learning using pretrained CNN and associative memory bank. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 01–08. IEEE, July 2018.

Lu, M. Y., Chen, B., Williamson, D. F. K., Chen, R. J., Liang, I., Ding, T., Jaume, G., Odintsov, I., Zhang, A., Le, L. P., Gerber, G., Parwani, A. V., and Mahmood, F. Towards a Visual-Language foundation model for computational pathology. *arXiv [cs.CV]*, July 2023.

Lundberg, S. and Lee, S.-I. A unified approach to interpreting model predictions. *arXiv [cs.AI]*, May 2017.

Machiraju, G., Plevritis, S., and Mallick, P. A dataset generation framework for evaluating megapixel image classifiers and their explanations. In *Computer Vision – ECCV 2022*, pp. 422–442. Springer Nature Switzerland, 2022.

Niazi, M. K. K., Parwani, A. V., and Gurcan, M. N. Digital pathology and artificial intelligence. *Lancet Oncol.*, 20 (5):e253–e261, May 2019.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Johnston, S., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. In-context learning and induction heads. *arXiv [cs.LG]*, September 2022.

Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, Ł., Shazeer, N., Ku, A., and Tran, D. Image transformer. *arXiv [cs.CV]*, February 2018.

Pawlowski, N., Bhooshan, S., Ballas, N., Ciompi, F., Glocker, B., and Drozdzal, M. Needles in haystacks: On classifying tiny objects in large images. *arXiv [cs.CV]*, August 2019.

Poli, M., Massaroli, S., Nguyen, E., Fu, D. Y., Dao, T., Baccus, S., Bengio, Y., Ermon, S., and Ré, C. Hyena hierarchy: Towards larger convolutional language models. *arXiv [cs.LG]*, February 2023a.

Poli, M., Wang, J., Massaroli, S., Quesnelle, J., Carlow, R., Nguyen, E., and Thomas, A. StripedHyena: StripedHyena: Moving beyond transformers with hybrid signal processing models. https://github.com/togethercomputer/stripedhyena, November 2023b. Accessed: 2024-6-2.

Putignano, V., Rosato, A., Banci, L., and Andreini, C. MetalPDB in 2018: a database of metal sites in biological macromolecular structures. *Nucleic Acids Res.*, 46(D1): D459–D464, January 2018.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 2021.

Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Adler, T., Gruber, L., Holzleitner, M., Pavlović, M., Sandve, G. K., Greiff, V., Kreil, D., Kopp, M., Klambauer, G., Brandstetter, J., and Hochreiter, S. Hopfield networks is all you need. *arXiv [cs.NE]*, July 2020.

Ribeiro, M. T., Singh, S., and Guestrin, C. "why should I trust you?": Explaining the predictions of any classifier. *arXiv [cs.LG]*, February 2016.

Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J., and Fergus, R. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proc. Natl. Acad. Sci. U. S. A.*, 118(15), April 2021.

Robinson, J., Jegelka, S., and Sra, S. Strength from weakness: Fast learning using weak supervision. *arXiv [cs.LG]*, February 2020.

Saab, K., Tu, T., Weng, W.-H., Tanno, R., Stutz, D., Wulczyn, E., Zhang, F., Strother, T., Park, C., Vedadi, E., Chaves, J. Z., Hu, S.-Y., Schaekermann, M., Kamath, A., Cheng, Y., Barrett, D. G. T., Cheung, C., Mustafa, B., Palepu, A., McDuff, D., Hou, L., Golany, T., Liu, L., Alayrac, J.-B., Houlsby, N., Tomasev, N., Freyberg, J., Lau, C., Kemp, J., Lai, J., Azizi, S., Kanada, K., Man, S., Kulkarni, K., Sun, R., Shakeri, S., He, L., Caine, B., Webson, A., Latysheva, N., Johnson, M., Mansfield, P., Lu, J., Rivlin, E., Anderson, J., Green, B., Wong, R., Krause, J., Shlens, J., Dominowska, E., Ali Eslami, S. M., Cui, C., Vinyals, O., Kavukcuoglu, K., Manyika, J., Dean, J., Hassabis, D., Matias, Y., Webster, D., Barral, J., Corrado, G., Semturs, C., Sara Mahdavi, S., Gottweis, J., Karthikesalingam, A., and Natarajan, V. Capabilities of gemini models in medicine. *arXiv [cs.AI]*, April 2024.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-CAM: Visual explanations from deep networks via gradient-based localization. *arXiv [cs.CV]*, October 2016.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for Large-Scale image recognition. *arXiv [cs.CV]*, September 2014.

Singhal, K., Azizi, S., Tu, T., Mahdavi, S. S., Wei, J., Chung, H. W., Scales, N., Tanwani, A., Cole-Lewis, H., Pfohl, S., Payne, P., Seneviratne, M., Gamble, P., Kelly, C., Babiker, A., Schärli, N., Chowdhery, A., Mansfield, P., Demner-Fushman, D., Agüera Y Arcas, B., Webster, D., Corrado, G. S., Matias, Y., Chou, K., Gottweis, J., Tomasev, N., Liu, Y., Rajkomar, A., Barral, J., Semturs, C., Karthikesalingam, A., and Natarajan, V. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180, August 2023.

Song, A. H., Jaume, G., Williamson, D. F. K., Lu, M. Y., Vaidya, A., Miller, T. R., and Mahmood, F. Artificial intelligence for digital and computational pathology. *Nature Reviews Bioengineering*, 1(12):930–949, October 2023.

Sparck Jones, K. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, January 1972.

Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. *arXiv [cs.LG]*, March 2017.

Swanson, K., Chang, H., and Zou, J. Predicting immune escape with pretrained protein language model embeddings. In Knowles, D. A., Mostafavi, S., and Lee, S.-I. (eds.), *Proceedings of the 17th Machine Learning in Computational Biology meeting*, volume 200 of *Proceedings of Machine Learning Research*, pp. 110–130. PMLR, 2022.

Talukder, S., Yue, Y., and Gkioxari, G. TOTEM: TOkenized time series EMbeddings for general time series analysis. *arXiv [cs.LG]*, February 2024.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *arXiv [cs.CL]*, June 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. *arXiv [stat.ML]*, October 2017.

Wang, H., Wang, Z., Du, M., Yang, F., Zhang, Z., Ding, S., Mardziel, P., and Hu, X. Score-CAM: Score-Weighted visual explanations for convolutional neural networks. *arXiv [cs.CV]*, October 2019.

Wang, H., Fu, T., Du, Y., Gao, W., Huang, K., Liu, Z., Chandak, P., Liu, S., Van Katwyk, P., Deac, A., Anandkumar, A., Bergen, K., Gomes, C. P., Ho, S., Kohli, P., Lasenby, J., Leskovec, J., Liu, T.-Y., Manrai, A., Marks, D., Ramsundar, B., Song, L., Sun, J., Tang, J., Veličković, P., Welling, M., Zhang, L., Coley, C. W., Bengio, Y., and

Prospector Heads: Generalized Feature Attribution for Large Models & Data

Zitnik, M. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60, August 2023.

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., and Zhou, M. MiniLM: Deep Self-Attention distillation for Task-Agnostic compression of Pre-Trained transformers. *arXiv [cs.CL]*, February 2020.

Weber, M., Welling, M., and Perona, P. Unsupervised learning of models for recognition. In *Computer Vision - ECCV 2000*, Lecture notes in computer science, pp. 18–32. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

Wiegreffe, S. and Pinter, Y. Attention is not not explanation. *arXiv [cs.CL]*, August 2019.

Wu, S., Liu, T., and Altman, R. B. Identification of recurring protein structure microenvironments and discovery of novel functional sites around CYS residues. *BMC Struct. Biol.*, 10:4, February 2010.

Xu, H., Usuyama, N., Bagga, J., Zhang, S., Rao, R., Naumann, T., Wong, C., Gero, Z., González, J., Gu, Y., Xu, Y., Wei, M., Wang, W., Ma, S., Wei, F., Yang, J., Li, C., Gao, J., Rosemon, J., Bower, T., Lee, S., Weerasinghe, R., Wright, B. J., Robicsek, A., Piening, B., Bifulco, C., Wang, S., and Poon, H. A whole-slide foundation model for digital pathology from real-world data. *Nature*, May 2024.

Yang, C., Yin, F., He, H., Chang, K.-W., Ma, X., and Xiang, B. Efficient shapley values estimation by amortization for text classification. *arXiv [cs.CL]*, May 2023.

Ying, R., Bourgeois, D., You, J., Zitnik, M., and Leskovec, J. GNNExplainer: Generating explanations for graph neural networks. *Adv. Neural Inf. Process. Syst.*, 32:9240–9251, December 2019.

Zaharia, M., Khattab, O., Chen, L., Davis, J. Q., Miller, H., Potts, C., Zou, J., Carbin, M., Frankle, J., Rao, N., and Ghodsi, A. The shift from models to compound AI systems. https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/. Accessed: 2024-5-17.

Zech, J. R., Badgeley, M. A., Liu, M., Costa, A. B., Titano, J. J., and Oermann, E. K. Confounding variables can degrade generalization performance of radiological deep learning models. *arXiv [cs.CV]*, July 2018.

Zhang, K., Yu, J., Adhikarla, E., Zhou, R., Yan, Z., Liu, Y., Liu, Z., He, L., Davison, B., Li, X., Ren, H., Fu, S., Zou, J., Liu, W., Huang, J., Chen, C., Zhou, Y., Liu, T., Chen, X., Chen, Y., Li, Q., Liu, H., and Sun, L. BiomedGPT: A unified and generalist biomedical generative pre-trained transformer for vision, language, and multimodal tasks. *arXiv [cs.CL]*, May 2023a.

Zhang, S., Xu, Y., Usuyama, N., Bagga, J., Tinn, R., Preston, S., Rao, R., Wei, M., Valluri, N., Wong, C., Lungren, M. P., Naumann, T., and Poon, H. Large-Scale Domain-Specific pretraining for biomedical Vision-Language processing. *arXiv [cs.CV]*, March 2023b.

Zhang, Z., Zhao, T., Guo, Y., and Yin, J. RS5M and GeoRSCLIP: A large scale Vision-Language dataset and a large Vision-Language model for remote sensing. *arXiv [cs.CV]*, June 2023c.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. Learning deep features for discriminative localization. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2016.

Zhou, B., Sun, Y., Bau, D., and Torralba, A. Interpretable basis decomposition for visual explanation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 119–134, 2018.

Zhou, X., Zhang, W., Chen, Z., Diao, S., and Zhang, T. Efficient neural network training via forward and backward propagation sparsification. *arXiv [cs.LG]*, November 2021a.

Zhou, Y., Booth, S., Ribeiro, M. T., and Shah, J. Do feature attribution methods correctly attribute features? *arXiv [cs.LG]*, April 2021b.

15

A Related Work (Extended)

A.1 Feature Attribution via Explanation

In the current explanation-based paradigm, feature attributions are referred to as "explanations" and are performed by (1) training a supervised model before (2) interrogating the model's behavior (*e.g.*, via internals, forward or backward passes, or input perturbations) and inferring class-specific features. This framework can be described as *weak* or *coarse supervision* (Robinson et al., 2020) due to the sole use of class labels as a supervisory signal in combination with a low signal-to-noise ratio in the datum-label pairs — particularly when the prevalence of class-specific features is low (Pawlowski et al., 2019).

Explanations, and feature attribution more broadly, can be categorized as either *model-specific methods*, which aim to describe how model weights interact with different input features), or *model-agnostic methods*, which aim to describe how each feature contributes to prediction. Explanation-based attribution methods in general are inherently data-inefficient as they require ample labeled training data to train underlying classifiers. It should be noted that few methods can also be applied to all data modalities.

Model-specific methods like gradient-based saliency maps (Simonyan & Zisserman, 2014), class-activation maps (CAMs) (Zhou et al., 2016; Selvaraju et al., 2016; Wang et al., 2023), and attention maps (Jetley et al., 2018) use a classifier's internals (*e.g.*, weights, layer outputs), forward passes, and/or backpropagated gradients for attribution. Recent work has demonstrated gradients serve as poor localizers (Arun et al., 2020; Zech et al., 2018) — potentially due to their high sensitivity to inputs (Adebayo et al., 2018), unfaithfulness in reflecting classifiers' reasoning processes (Karimi et al., 2022) and propensity to identify spurious correlations despite classifier non-reliance (Adebayo et al., 2022). Furthermore, CAMs pose high computational costs with multiple forward and backward passes (Chen et al., 2023). Finally, attention maps are demonstrably poor localizers (Zhou et al., 2021b) also perhaps due to their unfaithfulness (Wiegreffe & Pinter, 2019) and difficulties in assigning class membership to input features (Jain & Wallace, 2019).

On the other hand, model-agnostic methods like SHAP (Lundberg & Lee, 2017) and LIME (Ribeiro et al., 2016) perturb input features to determine their differential contribution to classification. Recent work has shown SHAP struggles to localize class-specific regions and is provably no better than random guessing for inferring model behavior or for downstream tasks (Bilodeau et al., 2022). Furthermore, SHAP-style methods can be computationally expensive for a variety of reasons. Some methods face exponential or quadratic time complexities (Ancona et al., 2019) with respect to the number of input features (*e.g.*, pixels in an image) and are thus infeasible for high-dimensional data, while others require multiple forward and/or backward passes (Chen et al., 2022b) or require training additional comparably sized deep networks along with the original classifier (Jethani et al., 2021).

A.2 Modern Encoders & Context Sizes

Most modern encoders for unstructured data operate on **tokens**, or relatively small pieces of a datum, and their representations. Tokens can be user-prespecified and/or constructed by the encoder itself (potentially with help from a tokenizer) — where these encoders are respectively referred to as *partial-context* and *full-context* (Figure 2). Due to computational constraints, high-dimensional unstructured data (*e.g.*, gigapixel images) often require user-prespecified tokens (*i.e.*, patches) and partial-context encoders that embed each token (Lu et al., 2023; Huang et al., 2023; Klemmer et al., 2023; Lanusse et al., 2023).

We provide an illustrative example for the image modality. In this setting, determining encoder context is based on practical modeling constraints: computational complexity of an architecture's modeling primitives, input data dimensionality, and hardware. For example, an attention-based Vision Transformer (Dosovitskiy et al., 2020) experiences quadratic time complexity (Keles et al., 2022) with respect to input dimension. Standard images (*e.g.*, 224×224 pixels) easily fit in modern GPU memory, enabling us to train full-context encoders that construct token embeddings via intermediary layers. However, gigapixel images require user-prespecified tokens (*i.e.*, patches) and partial-context encoders.

Full-context encoders now include foundation models (FMs) for a variety of data modalities and domains, including natural imagery (Radford et al., 2021), radiology images (Zhang et al., 2023b;a; Singhal et al., 2023; Saab et al., 2024), pathology images (Xu et al., 2024), protein sequences (Rives et al., 2021; Lin et al., 2023), and molecular graphs like protein structures (Derry & Altman, 2023). On the other hand, partial-context encoders train on prespecified tokens like document sentences and image patches. In image domains like histopathology, remote sensing, and cosmology, numerous encoders have been proposed with varying training regimes: unsupervised encoders (Jean et al., 2019), weakly supervised classifiers (Chen

et al., 2022c; Campanella et al., 2019), and FMs (Klemmer et al., 2023; Jakubik et al., 2023; Zhang et al., 2023c; Huang et al., 2023; Lu et al., 2023; Lanusse et al., 2023) all build representations for patches.

Regarding feature attribution for partial context models, gradient-based saliency and attention maps have been used to explain class predictions for high-dimensional unstructured data like gigapixel imagery (Campanella et al., 2019; Chen et al., 2022c). However, studies report low specificity and sensitivity (Machiraju et al., 2022) in part because attribution for the entire datum is built by concatenating attributions across prespecified tokens (*e.g.*, image patches). Partial-context strategies incorrectly assume prespecified tokens are independent and identically distributed (IID).

Our work hinges on the assumption that FMs learn particularly rich embeddings and distributional semantics — and thus, sets of concepts — by virtue of their representational power. While feature attribution has not been explored by adapting FM embeddings, this work is inspired by the recent efforts to perform object detection and visual grounding via FM adaptation (Kuo et al., 2022; Kalibhat et al., 2023).

A.3 Broader Connections across ML

Prospectors bring together ideas from many classical and modern works in adaptation, interpretability, memory augmentation (Khosla et al., 2023), information retrieval, and language modeling. On the surface, prospectors resemble probing models (Alain & Bengio, 2016; Belinkov, 2021), but the fact that they learn token associations between *multiple* token embeddings is more akin to constellation models (Weber et al., 2000), self-attention layers (Vaswani et al., 2017), or multiple instance learning approaches (Javed et al., 2022). Layer (I) is inspired by concept bottlenecks (Section 2), but extends the definition of concepts to carry spatial semantics. To learn higher-order associations between concepts, *i.e.*, "token mixing" and inductive reasoning, layer (II) is inspired by both sliding window attention (Parmar et al., 2018; Child et al., 2019; Beltagy et al., 2020) and the emergent n-gram circuits seen in transformer induction heads (Akyürek et al., 2024; Olsson et al., 2022; Bietti et al., 2023). We foster the pattern-recognition capability via associative memory units (Hopfield, 1982; 1984; Kohonen, 1972; Ramsauer et al., 2020) built with an encoder's learned representations and graphical models (Liu & Mukhopadhyay, 2018; Graves et al., 2013). The result is that while prospectors are inspired by LLM reasoning, their implementation uses efficient statistical techniques, modeling primitives, and data structures.

B Out-of-scope Attribution Methods

For transparency, we also outline our choice to rule out certain baselines for our experiments. A top priority for baseline selection was modality generalizability.

LIME: we rule out LIME (Ribeiro et al., 2016) as a baseline for any of our tested data modalities. This is primarily due to the fact that LIME requires ground truth labels to explain each input. Since the inputs to our partial context encoders (and, in turn, LIME) are prespecified tokens (*e.g.*, sentences for the WikiSection task), LIME requires token-level labels to explain the importance of sub-tokens (*e.g.*, words). This requirement of token-level labels in our setting is fundamentally at odds with prospectors' goal to predict token labels, *i.e.*, learn class-specific tokens *de novo*.

FastSHAP: we do not compare prospectors to modern methods like FastSHAP (Jethani et al., 2021), which requires training additional models. FastSHAP specifically requires training two comparable models to the original encoder (*i.e.*, with a classifier head) with respect to parameter count: a "surrogate" model that typically mimics the encoder in architecture but trained with a masked-input training regime and an "explainer" model that learns to identify class-specific tokens. Such approaches are out of scope for this work, which aims to perform feature attribution with large models like FMs. Training surrogates for FMs is often practically infeasible.

C Prospector Heads

C.1 Core Definitions

All unstructured data can be represented as map graphs of tokens interacting in physical space. We introduce mathematical definitions to describe these representations. Map graphs are also depicted in Figure S1.

Definition C.1 (Map Graph)**.** A map graph $G(\mathcal{V}, \mathcal{E})$ is a collection of vertices $\mathcal{V}$ and edges $\mathcal{E}$ connecting neighboring vertices in Euclidean space. Each vertex $v^{(i)} \in \mathcal{V}$ has features $\mathbf{x}^{(i)}$ and each edge $e^{(i \leftrightarrow j)} \in \mathcal{E}$ connects vertices $v^{(i)}$ to $v^{(j)}$.

Definition C.2 (Connectivity)**.** A map graph G's connectivity δ is its maximal node degree.

Definition C.3 (Partial-context Encoder)**.** Given a map graph G, an encoder f is considered partial-context if it produces an embedding $\mathbf{x} = f(v) \in \mathbb{R}^d$.

Definition C.4 (Full-context Encoder)**.** Given a map graph G, an encoder f is considered full-context if it produces embeddings $[\mathbf{x}_1 \dots \mathbf{x}_T] = f(G)$, where $\mathbf{x}_i \in \mathbb{R}^d \quad \forall i = 1 \dots T$.

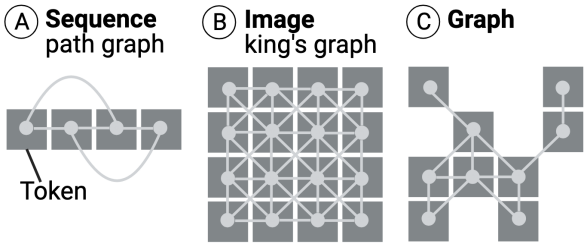

Figure S1: Unstructured data represented as map graphs. Sequences (A) and images (B) require a specified resolution (*e.g.*, words or sentences for text, pixels or patches for images) and connectivity (*e.g.*, 2-hop, 8-way) for discretization.

C.2 Visualizing Prospectors

We choose to visualize any dictionaries created by prospectors (*e.g.*, kernel ω and during `rollup` Appendix C.3.1) in two main styles throughout this work. Firstly, visualization can take the form of (1) *semantic networks*, which easily allow us to visualize either frequencies (in sprites) or importance weights (in kernels) for monogram and skip-bigram associations. These plots are sometimes referred to as "chord diagrams" or "circos plots". This data structure is defined mathematically as a self-complete graph:

Definition C.5 (Self-complete graph)**.** A self-complete graph $K_K(\mathcal{V}, \mathcal{E})$ is a fully connected graph with K vertices, where every pair of distinct vertices $v^{(i)}, v^{(j)}$ is connected by a unique edge $e^{(i \leftrightarrow j)}$. It also contains all self-edges that connect any vertex $v^{(i)}$ to itself with edge $e^{(i \leftrightarrow i)}$. Thus, self-complete graphs contain K vertices and $K + \binom{K}{2}$ edges.

This data structure is referenced in figures 4, 5, 8, and 9. Additionally, we can visualize all associations as (2) *heatmaps*, or unordered symmetric arrays, as seen in figures 5, 8, and 9.

C.3 Prospector Internals & Fitting

C.3.1 Layer II

The `rollup` operator, named after the function of the same name in relational databases, draws similarity to a sliding *bag of words* featurization scheme. Internally, a dictionary ζ is constructed to capture all monograms and skip-bigrams in each neighborhood of S. This operator is described by Algorithm 1 and depicted in Figure S2. For a full view of fitting layer (II), including both steps 1 (`rollup`) and 2, refer to Figure 5. We note that all sprite embeddings created in `rollup` were normalized using TF-IDF scaling (Sparck Jones, 1972) prior to kernel fitting.

Algorithm 1 `rollup`

Require: Sprite S, receptive field r
initialize dictionary ζ
for vertex v_i in S **do**
  for vertex v_j neighborhood $\mathcal{N}_r$ **do**
    $\zeta\langle S[v_i]\rangle \leftarrow$ occurrences of concept monogram $S[v_i]$
    $\zeta\langle (S[v_i], S[v_j])\rangle \leftarrow$ occurrences of concept skip-bigram $(S[v_i], S[v_j])$
  end for
end for
return $\mathbf{z} = \text{linearize}(\zeta)$
{where linearize returns values in sorted order}

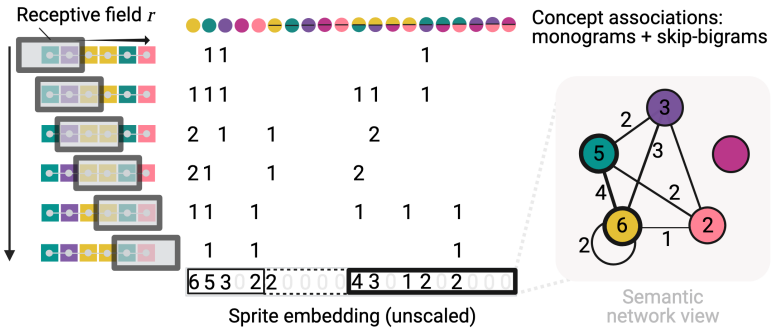

Figure S2: A depiction of the `rollup` operation as seen in layer (II).

C.3.2 Parameterization

Prospectors can have the following maximum number of importance weights, depending on r and K:

$$|\mathcal{Z}| = \begin{cases} K & r = 0 \quad \text{(monograms only)} \\ 2K + \binom{K}{2} & r > 0 \quad \text{(monograms \& bigrams)} \end{cases}$$

C.3.3 Prospector Variants: Implementation

As depicted in Figure 5, and discussed in the main body of this work, we implement two variants of prospector heads: a linear classifier variant and a fold-change variant. We provide additional details here for both variants. Details on hyperparameter selection λ, τ, α are discussed in Appendix D.2.

Linear classifier: This variant was implemented with the *sklearn* python package. The elastic net classifiers ($\lambda = 0.5$) trained for a maximum of 3000 iterations using the saga solver.

Fold-change computation: In order to supply an alternative to regularization for fold-change variants, we use two-way thresholding as inspired by differential expression analysis (Anders & Huber, 2010). These thresholds offer a form of "masking" importance weights $w_i \in \mathbf{w}$. As described in the main body of this work, the first threshold is τ, or the minimum fold-change required. The other threshold is α, which is a threshold used for a statistical hypothesis test, which is tests for independent class means. This test is conducted for each weight entry w_i in $\mathbf{w}$ and significance is assessed via a Mann-Whitney U hypothesis test. Prior to weight masking, given the number of independent tests being conducted, we adjust our chosen significance threshold using the commonly used Bonferroni correction: our original α threshold is divided by the number of entries in $\mathbf{w}$ ($|\mathcal{Z}|$), *i.e.*, $\alpha^* := \alpha/|\mathcal{Z}|$. Finally, to perform masking: we use $\pm\tau$ to mask out sufficiently small absolute fold changes (*e.g.*, ± 1, which indicates a requirement for doubling in $\log_2$-scale), and use α^* to mask out non-significant differences assessed by our hypothesis test.

C.4 Inferential Time Complexity

Here we conduct a comparative runtime analysis, where we analyze the worst-case time complexity required to explain a single input datum. We focus our analysis on the image modality due to compatibility with many baseline attribution methods. Suppose we have a trained encoder (*e.g.*, an end-to-end classifier, unsupervised learner, etc.) and our datum has $T = |\mathcal{V}|$ (tunable) tokens to analyze. Importantly, full-context encoders process all T tokens at once while partial-context encoders process T tokens in sequence. This distinction affects runtime complexity, so we analyze complexity for both partial- and full-context settings.

C.4.1 Prospectors

To analyze prospectors, we consider two main variables in computation: the number of tokens T and the number of operations for a forward pass (F) of the underlying encoder. Given these variables, prospectors themselves require only $O(T)$ computations per layer at inference time: $O(T)$ to quantize each token and $O(T)$ to traverse over all tokens during convolution. The latter operation ignores a near-constant term for the worst-case number of interactions, *i.e.*, skip-bigrams between central token and tokens in the r-neighborhood. The worst-case number of interactions is modality- and user-specific and is dependent on G's topology, r, and connectivity δ (*i.e.*, max node degree). Parameters r and δ are both

Attribution method	Full-context	Partial-context
Gradients	$O(F+B)$	$O(T(F+B))$
CAM (Zhou et al., 2016)	$O(pF)$	$O(TpF)$
GradCAM (Selvaraju et al., 2016)	$O(pF+B)$	$O(T(pF+B))$
ScoreCAM (Wang et al., 2019)	$O(pF)$	$O(TpF)$
Attention	$O(F)$	$O(TF)$
SHAP (Lundberg & Lee, 2017)	$O(F2^T)$	$O(TF2^{T^*})$
DASP (Ancona et al., 2019)	$O(FT^2)$	$O(TFT^{*2})$
G-DeepSHAP (Chen et al., 2022b)	$O(p(F+B))$	$O(Tp(F+B))$
Prospectors (ours)	$O(F+T)$	$O(TF)$

Table S1: Comparison of inference-time computational complexity between attribution methods. Parameters T, T^*, F, B, and p are respectively the number of: tokens, sub-tokens, operations in a forward pass, operations in a backward pass, number of passes.

typically set as small constants, so we can consider them negligible for time complexity.

Because prospectors are equipped to backbone encoders, inference in totality must account for the encoder's computational costs as well. Namely, an encoder requires $O(F)$ for inference with full context and $O(TF)$ for partial context (since each prespecified token requires a forward pass). Thus, total computational complexity of an encoder-prospector pipeline operating on a single input datum is $O(TF)$ for partial-context encoders and $O(T+F)$ for full-context encoders.

C.4.2 Baseline Methods

To properly characterize baseline methods, we also consider variables for backward pass operations (B), sub-tokens (T^*), and number of passes (p) if applicable. Sub-tokens are any (tunable) constituents within a token (*e.g.*, pixels in a patch) and are required by some baselines in partial-context settings. For example, given a text document (datum) and its sentence-level tokens, SHAP (Lundberg & Lee, 2017) may analyze the contribution of word sub-tokens. For comparative summary between prospectors and multiple baselines, please refer to Table S1. We discuss ruled-out baselines in Appendix B.

C.5 Floating Point Operations (Theoretical)

Because prospectors only rely on forward passes from an equipped encoder, our approach to feature attribution is approximately 3-4× more computationally efficient than gradient-based attribution methods. This analysis is based on empirical results from computing floating point operations (FLOPs) for model inference (*i.e.*, passes with frozen weights). Since prospectors' FLOPs are significantly less than a forward pass at inference time, this efficiency boost is approximated by the forward-backward pass FLOP ratio of 1:2 to 1:3 (Zhou et al., 2021a; Baydin et al., 2017). This efficiency is especially relevant for multi-billion parameter foundation models that could be used as upstream encoders.

C.6 Additional Properties

Prospectors have additional desirable properties:

- A form of "glocal" attribution: prospectors simultaneously build a global, dataset-level kernel of scored concept associations while also building local, datum-specific prospect maps
- *Interpretable*: prospector kernels can be inspected and verified by users to interpret class-specific concepts
- Arguably *generative* in nature via kernel construction: kernels can be rescaled and interpreted as joint probabilities, *i.e.*, $\mathbb{P}(c_i, y)$ for monograms and $\mathbb{P}(c_j, c_k, y)$ for skip-bigrams
- Shift- and rotation-equivariant, thus order-free (Bronstein et al., 2021): controlling for any randomness, fitting and inference can start at any origin vertex to yield consistent attributions
- Scale-invariant: input data can be of any number of tokens
- Can be *deterministic*: given the above, fold-change variants (which have no trainable parameters) can create deterministic prospect maps if hypothesis testing is forgone
- Theoretically can learn *implicit* skip-n-grams over a datum, as discussed in Appendix C.7

C.7 Theoretical Insights

C.7.1 Implicit n-grams

Prospector heads are inspired by the induction heads (Olsson et al., 2022), also referred to as n-gram heads (Akyürek et al., 2024), found in trained transformers for language modeling — even inspiring our method's name. While induction heads perform a sort of "pattern completion" (Olsson et al., 2022) using tokens, our approach achieves a form of "pattern recognition" and simplifies this computation and parameter space in multiple ways: a quantization of token to a set of K concepts and the explicit learning of monograms and skip-bigrams ($n = 1$ and $n = 2$).

We claim that this strategy to learn monograms and skip-bigrams is sufficient for implicitly learning higher order n-gram targets. Namely, we argue that skip-bigrams can be "chained" together to form implicit skip-n-grams during attribution, *i.e.*, during the creation of prospect maps in layer (II). For example, iteration i of convolution may find a skip-bigram of concepts A–B within the r-sized receptive field (i.e. A and B may be up to r hops away) and then iteration $i + r$ may find a skip-bigram of concepts B–C. Together, one can argue that both skip-bigrams form an implicit skip-trigram A–B–C. This implicit chaining of skip-n-grams can also lead to implicitly capturing longer-range dependencies. In Theorem 1 below, we show that skip-n-grams can be implicitly chained up to $(n - 1)r$ hops away in a map graph of tokens, G.

Theorem C.6 (Range of implicit n-grams)**.** Given a map graph G of cardinality T, prospectors with receptive field r and an ideal kernel can find all target 1-grams, skip-2-grams, ..., skip-n-grams spanning up to $(n - 1)r$ node hops.

Proof sketch. First, we explore the $n := 1$ case (*i.e.*, monograms). Here, all target 1-grams are found trivially via kernel look-ups. Next, we take a look at the $n := 2$ case (*i.e.*, skip-bigrams). Given the receptive field r, skip-bigrams can be found up to r hops away from the central node. Both the $n := 1$ and $n := 2$ cases can be generalized to single `k2conv` iterations over a large graph G (large T) or for small G where $T \leq 2r$ (G fully captured within r hops). Given prospectors natively find monograms and skip-bigrams, multiple convolutional iterations are needed to find $n \geq 3$. We explore these cases next.

For the $n := 3$ case, *i.e.*, skip-trigrams, two skip-bigrams must be found in sequence with a shared token between them. We call this process "bigram chaining." Given a skip-bigram can be learned over r hops, prospectors can thus learn a skip-trigram over $2r$ node hops. The desired property trivially generalizes over any choice of n (and r) via induction. ■

Through the kernel's "memorization" of salient monograms and skip-bigram "rules," prospectors offer flexibility without exorbitant parameterization (as with attention) — *i.e.*, the kernel does not need to see and learn a particular skip-n-gram in training, but at inference-time it can implicitly construct and recognize higher order skip-n-grams from its learned bigrams.

C.7.2 Prospector Failure Modes

One potential failure mode for prospection is triggered by small receptive fields (r), which can prevent prospectors from learning target skip-bigrams or skip-n-grams for any n. In the previous section, we show how prospectors can potentially "chain" skip-bigrams to implicitly learn higher-order skip-n-grams (as seen with transformer induction heads). However this expressivity is hinged on a sufficient choice of r — prospectors must ensure the r-size field captures the target bigrams at the minimum. We hope to study this potential failure mode with synthetic benchmarks in future works.

C.7.3 Impossibility Theorems for Feature Attribution

Finally, another main motivation in prospector design is recent work on impossibility theorems (Bilodeau et al., 2022), showing that (a) complete and (b) linear attribution methods can provably fail to improve on random guessing for inferring model behavior. Our approach sought to develop attribution methods outside of these traditional axioms (a) and (b) (Sundararajan et al., 2017). Prospector heads are *not* complete by nature of not constraining all token attribution scores in a datum to sum to a class prediction. The linear model variant uses its coefficients to attribute tokens, while the fold-change variant does not even output a class prediction.

D Experimental details

D.1 Speed benchmarking for Inference

We run a speed benchmarking analysis between two main encoder-attribution pipelines: (1) MiniLM with a prospector head and (2) DeBERTa with a zero-shot classification head and PartitionSHAP. Given the Huggingface implementation for

	CPU time		Wall clock time	
Attribution	token	datum	token	datum
SHAP	25.459	1428.347	19.222	1078.738
Prospectors		0.108		0.108

Table S2: Speed benchmarking between PartitionSHAP (denoted as SHAP) and prospectors, as applied to partial-context encoders and text (WikiSection) data. We report mean values, and SHAP's values reflect a random sample of 86% (621/718) of test-set examples.

Name	WikiSection	Camleyon16	MetalPDB
Token resolution	sentence	224×224 patch	atom
Token connectivity (δ)	2(-hop)	8(-way)	-
Concept count (K)	{10,15,20,25,30}	{10,15,20,25,30}	{15,20,25,30}
Receptive field (r)	{0,1,2,4,8}	{0,1,2,4,8}	{0,1,2,4}
Significance threshold (α)	{0.01,0.025,0.05,∞}	{0.01,0.025,0.05,∞}	{0.001,0.01,0.05,0.5,∞}
Fold-change threshold (τ)	{0,1,2}	{0,1,2}	{0,1,2,4}
Regularization factor (λ)	0.5	0.5	{0.0,0.5,1.0}
Edge cutoff (ϵ)	-	-	{4.0,6.0,8.0}

Table S3: Hyperparameters tuned during training grid search. Note: edge cutoff (ϵ), the distance cutoff to control graph density, only applies to MetalPDB. We use ∞ to represent any large number that acts as a non-threshold.

DeBERTa's zero-shot classification, PartitionSHAP was automatically selected by the `shap` Python package (over other SHAP methods like DeepSHAP). We present the speed benchmarking in Table S2.

D.2 Hyperparameter Tuning via Grid Search

For each task, we conduct a grid-search of hyperparameter configurations to select an optimal prospector model. The prospector kernel has two main hyperparameters—the number of concepts k and the skip-gram neighborhood radius r. We also evaluate two prospector variants based on how the kernel is trained: hypothesis testing (with additional hyperparameters for the p-value α and fold change τ cutoffs) and linear modeling (with elastic net mixing hyperparameter λ). We describe all tested hyperparameters in our training grid search in Table S3.

D.3 Hyperparameter & Model Selection via Sequential Ranking

To select a top prospector configuration after the training grid-search, we first compute four token-level evaluation metrics for training set localization and apply sequential ranking over those chosen metrics. Applied in order, our chosen metrics were: precision, Matthews correlation coefficient (MCC), Dice coefficient, and AUPRC. These metrics were chosen because they enable segmentation-style evaluation, and we preferentially select on precision because it is especially important for detecting the small-scale class-specific regions in our data. For metrics that require a threshold (precision, MCC, and Dice coefficient), we select models based on the highest value attained over 11 thresholds: $0.0, 0.1, \ldots, 1.0$. Top prospectors per encoder, selected from the grid search and sequential ranking, are listed in Table S4.

D.4 Test Set Evaluation

After prospect graphs are created by prospector heads, we map back the values of each token to its original coordinates (referred to in the main body as "prospect maps"). Prior to evaluation, we feature scale values to $[0, 1]$. Experimenting with other feature scaling schemes, *e.g.*, dataset-level scaling based on minimum and maximum values, is left for future work. For reporting results on the held-out test set we focus mainly on AUPRC to provide a threshold-agnostic evaluation of each method. To compute AP, average the precision scores over a set of predetermined thresholds to binarize the prospect maps, as described in the previous section: $0.0, 0.1, \ldots, 1.0$.

Encoder Alias	K	r	τ	α	λ	ϵ
MiniLM	25	1	1	0.05	—	—
DeBERTa	—	—	—	—	—	—
tile2vec	20	8	0	∞	—	—
ViT	20	2	0	0.05	—	—
CLIP	30	2	2	∞	—	—
PLIP	15	1	2	0.01	—	—
COLLAPSE	25	4	4	1.0	—	8.0
ESM2	30	1	0	1.0	—	8.0
AA	21*	1	1	0.1	1.0	6.0

Table S4: Top prospectors per encoder, after model selection and sequential ranking. All selected prospectors except AA are parameter-free fold-change variants. *The AA encoder does not use clustering for quantization, since amino acids are already a discrete set of 21 tokens (20 standard amino acids + 1 entry for any non-standard amino acid). Symbol "—" denotes non-applicable hyperparameter.

D.5 Construction of Task Datasets

D.5.1 Sequences (WikiSection)

Wikisection's "disease" annotated subset contains $n = 3231$ documents total with 2513 training examples and 718 test examples. We preprocess the data into classes by searching each document for the presence of "disease.genetics" section labels. If this section label is found, we assign a document-level label of class_1 and class_0 otherwise. Because our task is at the sentence-level, we then create tokens by breaking sections into sentences by the full-stop delimiter (".") . We then label sentences by their source section labels. Raw-text sentences are then fed into our chose encoder, which handles natural language tokenization.

D.5.2 Images (Camelyon16)

This benchmark contains 400 gigapixel whole slide images (270 train, 130 test) of breast cancer metastases in sentinel lymph nodes. All images were partitioned into prespecified patch tokens (size 224×224) and filtered for foreground tissue regions (as opposed to the glass background of the slide). This process resulted in more than 200K unique patches without augmentation. For ground truth annotations, binary masks were resized with inter-area interpolation and re-binarized (value of 1 is assigned if interpolated value > 0) to match the dimensionality of data sprites.

We also visualize the token embedding spaces of our encoders for the image task in Figure S3. The lack of natural clustering of class_1-specific tokens (thick $\times$ markers) from class_0 tokens ($\circ$ markers) intuitively depicts the difficulty of our task. In other words, class-specific regions are made up of tokens that are conceptually similar to non-region tokens.

D.5.3 Graphs (MetalPDB)

We constructed a binary classification dataset of zinc-binding and non-binding proteins from the MetalPDB dataset (Putignano et al., 2018). We specifically focus on proteins annotated as enzymes, since metal ions are often critical for enzymatic activity. Such enzymes are known as metallo-enzymes, and our global classification labels reflect whether a metallo-enzyme relies on zinc or a different metal ion. For the positive set, we consider only biologically-relevant zinc ions which occur within a chain (*i.e.*, are bound to residues in the main chain of the protein, rather than ligand-binding or crystallization artifacts). We sample only one protein chain from each enzymatic class, as determined by Enzyme Commission numbers (Bairoch, 2000), selecting the structure with the best crystallographic resolution. This process resulted in 756 zinc-binding sites from 610 proteins, with 653 corresponding non-zinc-binding proteins sampled from unique enzymatic classes using the same procedure. For each zinc ion in the positive set, we extract all interacting residues annotated in MetalPDB to serve as our ground truth nodes for feature attribution. This dataset was split by enzyme class to ensure that no enzyme exists in both train and test sets, reserving 20% of chains for held-out evaluation. After removing four structures which produced embedding errors, this produced a training set of 1007 unique protein chains for the train set and 252 for the test set. Each protein is featurized as a graph where each node represents a residue and edges are defined

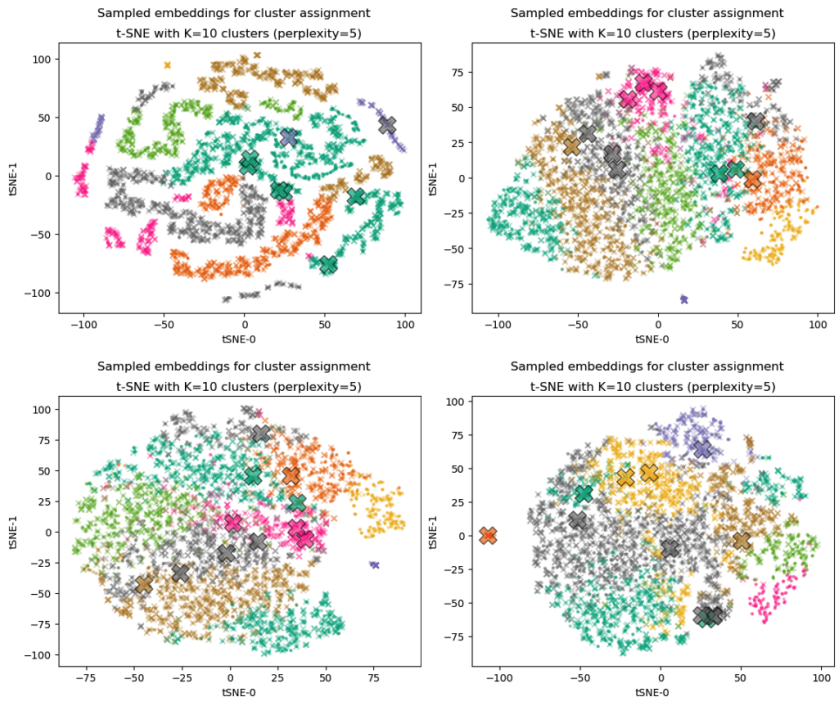

Figure S3: t-SNE plots, left-to-right then top-to-bottom: tile2vec, ViT, CLIP, PLIP. Marker color denotes concept, marker type denotes ground truth annotation for a patch: $\circ$ for class$_0$, $\times$ for patches from class$_1$ images that do originate from target regions, and the much thicker $\mathbf{\times}$ for class$_1$-specific target regions.

between residues which share any atom within a distance of ϵ angstroms, where ϵ varies the density of the graph.

D.6 Multi-class Settings

Prospectors can be easily adapted to the multi-class setting by training multiple models for each class of interest. For example, if faced with three classes a, b, c, prospectors could be applied in the following settings (class-1 and class-0, respectively):

- Prospector trained on a vs. $\{b, c\}$
- Prospector trained on b vs. $\{a, c\}$
- Prospector trained on c vs. $\{a, b\}$

In fact, both our protein (MetalPDB) and text (WikiSection) datasets are adapted from multi-class settings: MetalPDB contains data for many different metals, and WikiSection contains 27 different labels in the English disease document subset. In each case, we selected one class to evaluate for simplicity (zinc-binding proteins and genetics-related text, respectively), but one could easily construct an analogous dataset and train a model for any other class label.

D.7 Pre-trained Encoders

We outline specific models and how to access them.

D.7.1 Text (WikiSection)

MiniLM: We specifically use the `all-MiniLM-L6-v2` sentence Transformer model (via the *sentence-transformers* package), which is approximately 22M parameters in size.

DeBERTa: We specifically use the `DeBERTa-v3-base-mnli-fever-anli` model (via the *transformers* package),

which is approximately 98M parameters in size. DeBERTa is able to perform off-the-shelf ZSC and NLI.

Note: We forgo prospection with DeBERTa since it emits word embeddings without a simple and effective way to construct sentence embeddings. We reiterate that annotations are at the sentence-level (*i.e.*, our prespecified tokens).

D.7.2 IMAGES (CAMELYON16)

CLIP: We specifically use the `clip-vit-base-patch32` model via the *transformers* package.

PLIP: We specifically use the `plip` model via the *transformers* package.

D.7.3 GRAPHS (METALPDB)

COLLAPSE: We use the implementation and weights available at `https://github.com/awfderry/COLLAPSE`.

ESM: We use the ESM implementation and weights available at `https://github.com/facebookresearch/esm`, specifically the 33-layer, 650M parameter ESM2 model (`esm2_t33_650M_UR50D`).

D.8 Trained Encoders

We train two backbone encoders to equip with prospectors for the image task (Camelyon16):

tile2vec: This encoder uses a ResNet-18 architecture (He et al., 2015) trained for 20 epochs on a single NVIDIA T4 GPU. For training, the training set of 200K patches were formed into nearly 100K triplets with a sampling scheme similar to that of Jean et al. (2019). These triplets were then used to train tile2vec with the triplet loss function (Jean et al., 2019).

ViT: We trained a custom ViT for trained for 30 epochs on a single NVIDIA T4 GPU. It was trained to perform IID patch predictions under coarse supervision, which involved image-level label inheritance (Machiraju et al., 2022) — the process of propagating image-level class labels to all constituent patches.

D.9 Baseline Feature Attribution Methods

D.9.1 SEQUENCES (WIKISECTION)

MiniLM: support vector machine (SVM): Using our sampled training embeddings from clustering, we train a one-class SVM on class_0 token embeddings ($n = 4809$) to perform novelty detection on the held-out test set. The SVM was implemented with the *sklearn* package and trained with an RBF kernel and hyperparameter $\gamma = 1/d$ (where d is embedding dimension). Training ran until a stopping criterion was satisfied with 1e-3 tolerance.

MiniLM: multi-layer perceptron (MLP): Using our sampled training embeddings from clustering, we train an MLP on all ($n = 5000$) token embeddings (labeled as class_0 or class_1) to perform fully supervised token classification held-out test set — acting as a stand-in for a segmentation-like baseline. The MLP was implemented with the *sklearn* package and trained with one hidden layer (dimension 100), ReLU activations, adam optimizer, L2-regularization term of 1e-4, initial learning rate 1e-3, and minibatch size of 200. Training ran for a maximum of 1000 iterations, where inputs are shuffled.

DeBERTa: zero-shot classification (ZSC): The DeBERTa model can perform ZSC out of the box, giving us sentence-level ZSC probabilities (*i.e.*, logits). We used the ZSC binary labels of [“genetics”, “other”]. While we considered all possible labels in the WikiSection dataset’s disease subset (see below), we ultimately went with binary classification due to higher performance.

Unused multi-class labels: [“genetics”, “other”, “classification”, “treatment”, “symptom”, “screening”, “prognosis”, “tomography”, “mechanism”, “pathophysiology”, “epidemiology”, “geography”, “medication”, “fauna”, “surgery”, “prevention”, “infection”, “culture”, “research”, “history”, “risk”, “cause”, “complication”, “pathology”, “management”, “diagnosis”, “etymology”]

DeBERTa: ZSC with PartitionSHAP: We implemented a Shapley scoring pipeline for DeBERTa since it can perform ZSC end-to-end. The pipeline was implemented via the *transformers* package using the object class `zeroShotClassificationPipeline`. Shapley computation was performed via the *shap* package. The pipeline defaults to PartitionSHAP — in this setting, PartitionSHAP is applied to the partial-context DeBERTa model and considers sub-tokens (words) over all possible T tokens (sentences), ultimately pooling over sub-tokens to get a token-level score. We

run a speed benchmarking analysis for this approach in Appendix D.1.

Note: full-context shapley score computation was also considered. However, due to poor computational scaling (for both DeBERTa and PartitionSHAP), we ruled out this strategy.

DeBERTa: NLI entailment: The DeBERTa model can perform NLI entailment off the shelf, yielding sentence-level NLI entailment scores. We provide the model with an NLI hypothesis ("this sentence is about genetics") and NLI premise (*i.e.*, the input sentence). Labels extracted refer to ["entailment", "neutral", "contradiction"].

DeBERTa: NLI entailment attention: The DeBERTa model can also output attention scores. Attention scores are computed by max-pooling over the attention weights for the NLI hypothesis ("this sentence is about genetics") given the NLI premise (*i.e.*, the input sentence).

D.9.2 IMAGES (CAMELYON16)

For this task, baselines were chosen due to their popularity and efficiency.

Concatenated mean attention (ViT only): attention maps are created per input token and their values are averaged. This creates a single attention score per token, after which tokens are concatenated by their spatial coordinates. These values are scaled to values in $[0, 1]$.

Concatenated prediction probability: For ViT, each token's prediction probability for class_1 is used to score each token, after which tokens are concatenated by their spatial coordinates. For both vision-language models, CLIP and PLIP, we prompt both FMs' text encoders with zero-shot classification labels for class_0 and class_1, respectively: ["normal lymph node", "lymph node metastasis"]. These labels match the benchmark dataset's descriptions of class labels. Similarly to ViT, each token's class_1 prediction probability is used to score each token, after which tokens are concatenated by their spatial coordinates

D.9.3 GRAPHS (METALPDB)

Our baseline for zinc binding residue identification is a graph attention network (GAT) (Veličković et al., 2017) containing two GAT layers, each followed by batch normalization, followed by a global mean pooling and a fully-connected output layer. The input node features for each residue were given by the choice of encoder (COLLAPSE, ESM2, or AA). The GAT model was trained using weak supervision (*i.e.*, on graph-level labels y) using a binary cross-entropy loss and Adam optimizer with default parameters and weight decay of 1×10^{-4}. To select the best baseline model, we use a gridsearch over the edge cutoff (ϵ) for the underlying protein graph (6.0 or 8.0 Å), the learning rate (1×10^{-5}, 1×10^{-4}, 5×10^{-4}, 1×10^{-3}), and the GAT node feature dimension (100, 200, 500). Feature attribution for all explanation methods was performed using implementations provided by Pytorch Geometric (Fey & Lenssen, 2019). The best model was selected using the selection criteria in Appendix D.2. The final classification models for COLLAPSE, ESM, and AA encoders used edge cutoffs (ϵ) of 8.0 Å, 6.0 Å, and 6.0 Å, learning rates of 5×10^{-4}, 5×10^{-3}, and 5×10^{-3}, and feature dimensions of 100, 500, and 200, respectively.

GAT head + GNNExplainer: We use GNNExplainer (Ying et al., 2019) to produce explanations for nodes (*i.e.*, residues) only. We train the GNNExplainer module for 100 epochs with a default learning rate of 0.01.

GAT head + Attention: The attention baseline uses the attention scores of the trained GAT model to produce attribution scores. Attention scores across layers and heads are first max-pooled to produce aggregated attention scores for each edge. Then, we compute the attribution score for each node by averaging over the scores of all edges connected to it.

GAT head + SHAP: We deploy SHAP using a Shapley value sampling approach adapted specifically for graph data and implemented using Captum (`https://captum.ai`). SHAP is computationally feasible for this task primarily due to the use of full-context classifier heads to plug into each tested encoder. This allows SHAP to explain individual tokens (amino acids) by aggregating over edge weights using the same procedure as described for our attention baseline.

D.10 Tabular Results

We report quantitative results corresponding to Figure 6 in tables S5, S6, and S7. All reported errors reflect the standard error of the mean.

Encoder-Attribution	Mean AUPRC	Error AUPRC	Mean AP	Error AP
DeBERTa-ZSC	0.476	0.032	0.502	0.031
DeBERTa-SHAP	0.292	0.026	0.322	0.026
DeBERTa-NLI	**0.584**	0.030	**0.617**	0.023
DeBERTa-Attention	0.217	0.020	0.244	0.021
MiniLM-SVM	0.284	0.024	0.317	0.024
MiniLM-MLP	*0.626*	0.031	*0.648*	0.030
MiniLM-Prospector	**0.711**	0.030	**0.730**	0.028

Table S5: Tabular results for sequences (WikiSection). Top section contains baseline methods while bottom section contains prospector-equipped encoders. **Boldface** indicates best-in-encoder results. *Italics* indicates top non-prospector pipeline.

Encoder-Attribution	Mean AUPRC	Error AUPRC	Mean AP	Error AP
ViT-Attention	0.158	0.038	0.162	0.038
ViT-Probability	*0.207*	0.043	*0.212*	0.043
CLIP-Probability	0.149	0.035	0.155	0.034
PLIP-Probability	0.163	0.037	0.167	0.037
tile2vec-Prospector	**0.212**	0.044	**0.218**	0.044
ViT-Prospector	**0.210**	0.047	**0.215**	0.047
CLIP-Prospector	**0.330**	0.056	**0.298**	0.055
PLIP-Prospector	**0.470**	0.050	**0.300**	0.052

Table S6: Tabular results for images (Camelyon16). Top section contains baseline methods while bottom section contains prospector-equipped encoders. **Boldface** indicates best-in-encoder results. *Italics* indicates top non-prospector pipeline.

Encoder-Attribution	Mean AUPRC	Error AUPRC	Mean AP	Error AP
COLLAPSE-GNNExplainer	0.242	0.020	*0.266*	0.020
COLLAPSE-Attention	0.171	0.013	0.199	0.013
COLLAPSE-SHAP	0.370	0.015	0.162	0.016
ESM-GNNExplainer	0.036	0.003	0.047	0.004
ESM-Attention	0.050	0.005	**0.064**	0.005
ESM-SHAP	**0.304**	0.015	0.062	0.006
AA-GNNExplainer	0.105	0.015	0.121	0.015
AA-Attention	0.031	0.005	0.043	0.004
AA-SHAP	***0.420***	0.011	0.062	0.008
COLLAPSE-Prospector	**0.640**	0.020	**0.323**	0.039
ESM-Prospector	0.082	0.013	0.060	0.007
AA-Prospector	0.405	0.037	**0.354**	0.037

Table S7: Tabular results for protein graphs (MetalPDB). Top section contains baseline methods while bottom section contains prospector-equipped encoders. **Boldface** indicates best-in-encoder results. *Italics* indicates top non-prospector pipeline.

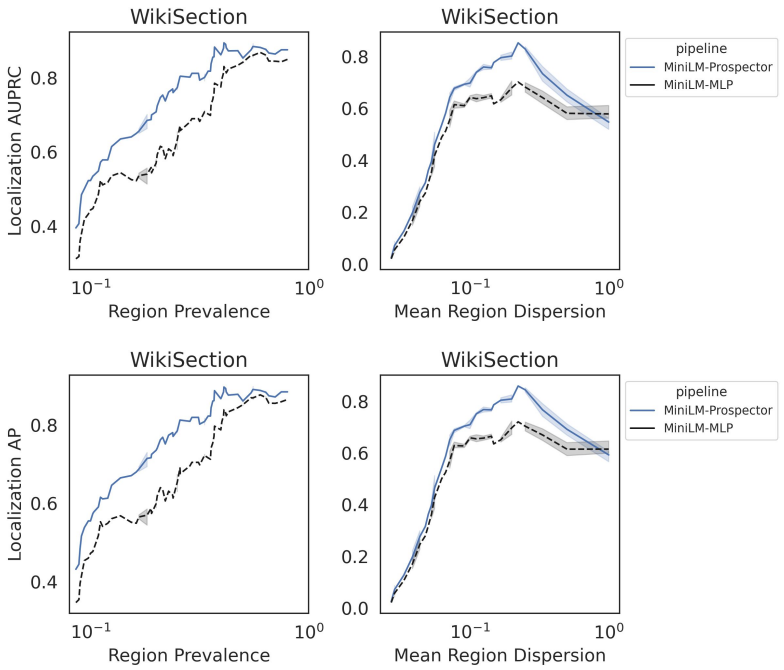

Figure S4: Robustness for WikiSection data. Top baseline, MiniLM encoder with MLP head, is denoted by a black dashed line.

D.11 Robustness to Coarse Supervision

We also briefly study the robustness of prospector (and top baseline) test-set performance with respect to salient region characteristics: region prevalence and mean region dispersion. We display these results in figures S4, S5, and S6. The more that lines gravitate to the top of each plot, the more robust an encoder-attribution pipeline is to target region characteristics. Lines are created by convolving over the test-set examples.

D.11.1 Additional analysis for MetalPDB

The metal-binding protein task is particularly challenging as the majority of its class-specific regions are below 0.1 prevalence, but prospectors were nonetheless able to achieve high performance on most test-set examples (Figure S6). Interestingly, ESM2 showed bimodal performance, with high AUPRC on one subset and a correlated, low performance on another. This suggests that a subset of data does not contain clear sequence patterns that are correlated with zinc binding, while structure-based encoders can capture local interactions between residues far apart in sequence. In addition to the prevalence of class-specific regions, mean region dispersion provides a view into their spatial organization.

D.12 Domain-Specific Analysis of Prospector Internals

Figure S7 displays the results of hierarchically clustering sprite embeddings for the zinc binding task.

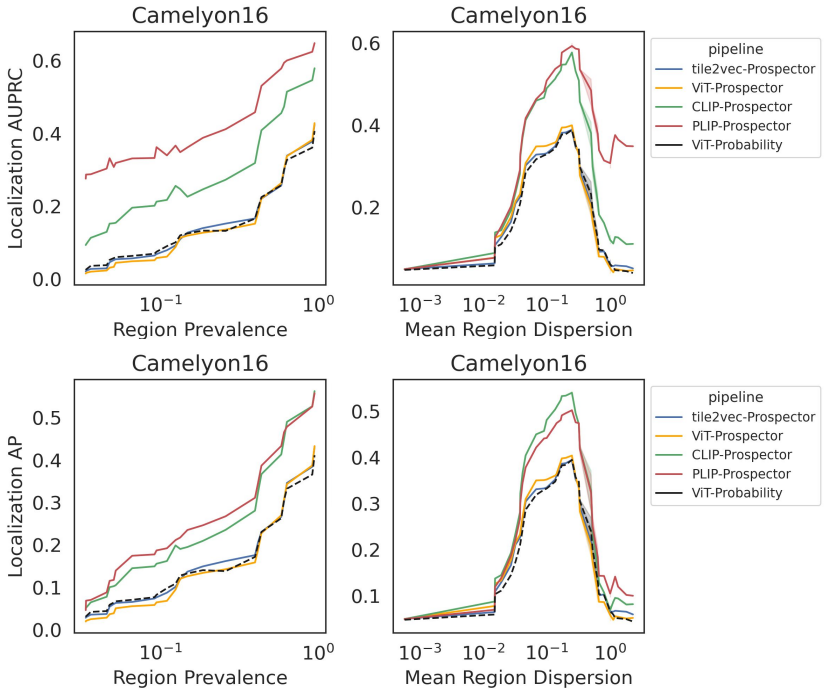

Figure S5: Robustness for Camelyon16 data. Top baseline, ViT encoder with prediction probabilities, is denoted by a black dashed line.

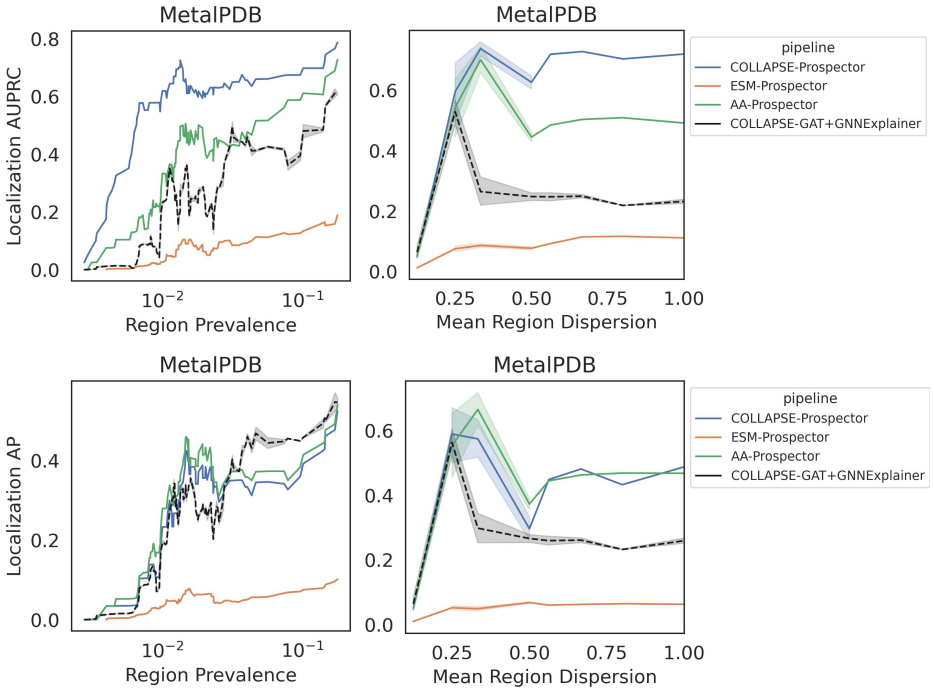

Figure S6: Robustness for MetalPDB data. Top baseline, COLLAPSE encoder with GAT head and GNNExplainer, is denoted by a black dashed line.

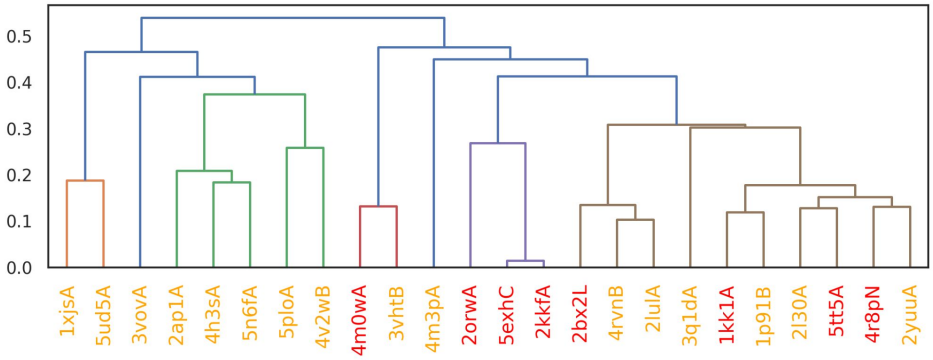

Figure S7: Hierarchical clustering results for sprite embeddings computed from high-precision test-set examples. X-axis labels are colored by the number of cysteine residues coordinating the central zinc ion, which is a key feature that is correlated with the resulting clusters: orange=3, red=4.