Scalable Interpretability via Polynomials

Abhimanyu Dubey
Meta AI
dubeya@fb.com

Filip Radenovic
Meta AI
filipradenovic@fb.com

Dhruv Mahajan
Meta AI
dhruvm@fb.com

Abstract

Generalized Additive Models (GAMs) have quickly become the leading choice for interpretable machine learning. However, unlike uninterpretable methods such as DNNs, they lack expressive power and easy scalability, and are hence not a feasible alternative for real-world tasks. We present a new class of GAMs that use tensor rank decompositions of polynomials to learn powerful, *inherently-interpretable* models. Our approach, titled Scalable Polynomial Additive Models (SPAM) is effortlessly scalable and models *all* higher-order feature interactions without a combinatorial parameter explosion. SPAM outperforms all current interpretable approaches, and matches DNN/XGBoost performance on a series of real-world benchmarks with up to hundreds of thousands of features. We demonstrate by human subject evaluations that SPAMs are demonstrably more interpretable in practice, and are hence an effortless replacement for DNNs for creating interpretable and high-performance systems suitable for large-scale machine learning. Source code is available at github.com/facebookresearch/nbm-spam.

1 Introduction

Interpretable machine learning systems suffer from a tradeoff between approximation and interpretability: high-performing models used in practice have millions of parameters and are highly non-linear, making them difficult to interpret. *Post-hoc* explainability attempts to remove this tradeoff by explaining black-box predictions with interpretable instance-specific approximations (e.g., LIME [Ribeiro et al., 2016], SHAP [Lundberg and Lee, 2017]), however, they are notoriously unstable [Ghorbani et al., 2019a, Lakkaraju and Bastani, 2020], expensive to compute [Slack et al., 2021], and in many cases, inaccurate [Lipton, 2018]. It is therefore desirable to learn models that are instead *inherently-interpretable* (glass-box), i.e., do not require *post-hoc* interpretability, but provide better performance than *inherently-interpretable* classical approaches such as linear or logistic regression.

This has led to a flurry of research interest in Generalized Additive Models [Hastie and Tibshirani, 2017], i.e., methods that non-linearly transform each input feature separately, e.g., tree-based GAM and GA^{2}Ms [Lou et al., 2013], or NAMs [Agarwal et al., 2021]. While these are an improvement over linear methods, due to their highly complex training procedures, they are not a simple drop-in replacement for black-box methods. Furthermore, it is not straightforward to model feature interactions in these models: the combinatorial explosion in higher-order interactions makes this infeasible without a computationally expensive feature selection [Lou et al., 2013, Chang et al., 2021].

Contrary to these approaches, we revisit an entirely different solution: *polynomials*. It is folklore in machine learning that since polynomials are universal approximators [Stone, 1948, Weierstrass, 1885], hypothetically, modeling all possible feature interactions will suffice in creating powerful learners and eliminate the need for intricate non-linear transformations of data. Furthermore, polynomials can also provide a rubric for interpretability: lower-degree polynomials, that have interactions between a few features are interpretable but likely less accurate, whereas polynomials of higher degree have larger predictive power at the cost of interpretability. Thus, not only are polynomial models capable of capturing all possible interactions between features, they also give practitioners the ability to select a suitable model (order of interactions, i.e., degree) based precisely on their requirement in the

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

interpretability-performance tradeoff. However, such an approach has till date remained elusive, as learning interpretable feature interactions *efficiently and at scale* remains an open problem.

In this paper, we introduce a highly scalable approach to modeling feature interactions for inherently interpretable classifiers based on *rank decomposed polynomials*. Our contributions are:

1. First, we present an algorithm titled Scalable Polynomial Additive Models (**SPAM**) to learn *inherently-interpretable* classifiers that can learn all possible feature interactions by leveraging *low-rank tensor decompositions* of polynomials [Nie, 2017b]. To the best of our knowledge, **SPAM** is the among the first Generalized Additive Models with feature interactions scalable to 100K features. SPAM can be trained end-to-end with SGD (backpropagation) and GPU acceleration, but has orders of magnitude fewer parameters than comparable interpretable models.
2. We demonstrate that under a coarse regularity assumption (Assumption 1), SPAM converges to the optimal polynomial as the number of samples $n \to \infty$. Furthermore, we establish novel non-asymptotic excess risk bounds that match classic bounds for linear or full-rank polynomials.
3. SPAM outperforms all current interpretable baselines on several machine learning tasks. To the best of our knowledge, our experimental benchmark considers problems of size (number of samples, and dimensionality) orders of magnitude larger than prior work, and we show that our simple approach scales easily to all problems, whereas prior approaches fail. Moreover, we demonstrate that the need for modeling feature interactions is as important as non-linear feature transformations: on most real-world tabular datasets, *pairwise* interactions suffice to match DNN performance.
4. We conduct a detailed human subject evaluation to corroborate our performance with practical interpretability. We demonstrate that SPAM explanations are more faithful compared to *post-hoc* methods, making them a *drop-in replacement for black-box methods* such as DNNs for interpretable machine learning on tabular and concept-bottleneck [Koh et al., 2020] problems.

2 Related Work

Here we survey the literature most relevant to our work, i.e., interpretable ML with feature interactions, and polynomials for machine learning. Please see Appendix Section B for a detailed survey.

Transparent and Interpretable Machine Learning. Early work has focused on greedy or ensemble approaches to modeling interactions [Friedman, 2001, Friedman and Popescu, 2008] that enumerate pairwise interactions and learn additive interaction effects. Such approaches often pick spurious interactions when data is sparse [Lou et al., 2013] and are impossible to scale to modern-sized datasets due to enumeration of individual combinations. As an improvement, Lou et al. [2013] proposed GA^{2}M that uses a statistical test to select only "true" interactions. However GA^{2}M fails to scale to large datasets as it requires constant re-training of the model and ad-hoc operations such as discretization of features which may require tuning for datasets with a large dimensionality. Other generalized additive models require expensive training of millions of decision trees, kernel machines or splines [Hastie and Tibshirani, 2017], making them unattractive compared to black-box models.

An alternate approach is is to learn interpretable neural network transformations. Neural Additive Models (NAMs, Agarwal et al. [2021]) learn a DNN per feature. TabNet [Arık and Pfister, 2021] and NIT [Tsang et al., 2018] alternatively modify NN architectures to increase their interpretability. NODE-GAM [Chang et al., 2021] improves NAMs with oblivious decision trees for better performance while maintaining interpretability. Our approach is notably distinct from these prior works: we do not require iterative re-training; we can learn *all* pairwise interactions regardless of dimensionality; we can train SPAM via backpropagation; and we scale effortlessly to very large-scale datasets.

Learning Polynomials. The idea of decomposing polynomials was of interest prior to the deep learning era. Specifically, the work of Ivakhnenko [1971], Oh et al. [2003], Shin and Ghosh [1991] study learning neural networks with polynomial interactions, also known as *ridge polynomial networks* (RPNs). However, RPNs are typically not interpretable: they learn interactions of a very high order, and include non-linear transformation. Similar rank decompositions have been studied in the context of matrix completion [Recht, 2011], and are also a subject of interest in tensor decompositions [Nie, 2017a, Brachat et al., 2010], where, contrary to our work, the objective is to decompose existing tensors rather than directly learn decompositions from gradient descent. Recently, Chrysos et al. [2019, 2020] use tensor decompositions to learn higher-order polynomial relationships in intermediate layers of generative models. However, their work uses a recursive formulation and learns high-degree polynomials directly from uninterpretable input data (e.g., images), and hence is non-interpretable.

3 Scalable Polynomial Additive Models

Setup and Notation. Matrices are represented by uppercase boldface letters, e.g., $\mathbf{X}$ and vectors by boldface lowercase, i.e., $\boldsymbol{x}$. We assume that the covariates lie within the set $\mathcal{X} \subseteq \mathbb{R}^d$, and the labels lie within the finite set $\mathcal{Y}$. Data $(\boldsymbol{x}, y) \in \mathcal{X} \times \mathcal{Y}$ are drawn following some unknown (but fixed) distribution $\mathfrak{P}$. We assume we are provided with n i.i.d. samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ as the training set.

Motivation. Generalized Additive Models [Hastie and Tibshirani, 2017] are an excellent design choice for interpretable models, as they learn transformations of individual features, allowing us to model exactly the contribution of any feature. A typical GAM is as follows.

$$\xleftarrow[\text{Interpretability}]{}$$

$$f(\boldsymbol{x}) = \sum_{i=1}^{d} \underbrace{f_i(x_i)}_{\text{order 1 (unary)}} + \sum_{j>i}^{d} \underbrace{f_{ij}(x_i, x_j)}_{\text{order 2 (pairwise)}} + \cdots + \underbrace{f_{1\ldots d}(\boldsymbol{x})}_{\text{order } d}.$$

$$\xrightarrow[\text{Performance}]{}$$

Where f_i, etc., are possibly non-linear transformations. It is evident [Lou et al., 2013] that as the order of interaction increases, e.g., beyond pairwise interactions, these models are no longer interpretable, albeit at some benefit to performance. While some approaches (e.g., Agarwal et al. [2021], Chang et al. [2021]) learn f_i via neural networks, we want to learn the simplest GAMs, i.e., polynomials. Specifically, we want to learn a polynomial $P(\boldsymbol{x})$, of order (degree) $k \leq d$ of the form:

$$P(\boldsymbol{x}) = b + \sum_{i=1}^{d} w_i^{(1)} \cdot x_i + \sum_{i,j}^{d} w_{ij}^{(2)} \cdot x_i x_j + \ldots \sum_{i_1\ldots,i_k}^{d} \left(w_{i_1\ldots i_k}^{(k)} \cdot \prod_{j=1}^{k} x_{i_j} \right).$$

Here, the weights $\mathbf{W}^{(l)} = \{w_{i_1\ldots i_l}^{(l)}\}, 1 \leq l \leq k$ capture the $l-$ary interactions between subsets of l features. For small values of d and k, one can potentially enumerate all possible interactions and learn a linear model of $\mathcal{O}(d^k)$ dimensionality, however this approach quickly becomes infeasible for large d and k. Furthermore, the large number of parameters in $P(\boldsymbol{x})$ make regularization essential for performance, and the computation of each interaction can be expensive. Alternatively, observe that any polynomial that models $k-$ary interactions can be written as follows, for weights $\{\mathbf{W}^{(l)}\}_{l=1}^k$,

$$P(\boldsymbol{x}) = \mathbf{W}^{(1)} \odot_1 \boldsymbol{x} + \mathbf{W}^{(2)} \odot_2 \boldsymbol{x} + \ldots \mathbf{W}^{(k)} \odot_k \boldsymbol{x} + b.$$

Here, the weights $\mathbf{W}^{(l)} \in \mathbb{R}^{d^l}$ are written as a tensor of order l, and the operation $\odot_l$ refers to a repeated inner product along l dimensions, e.g., $\mathbf{W} \in \mathbb{R}^{d^2}$, $\boldsymbol{x}^\top \mathbf{W} \boldsymbol{x} = \mathbf{W} \odot_2 \boldsymbol{x} = (\mathbf{W} \odot \boldsymbol{x}) \odot \boldsymbol{x}$ where $\odot$ denotes the inner product. More generally, any *symmetric* order l tensor admits an equivalent polynomial representation having *only* degree$-l$ terms. Now, this representation is still plagued with the earlier curse of dimensionality for arbitrary weight tensors $\{\mathbf{W}^{(l)}\}_{l=1}^k$, and we will now demonstrate how to circumvent this issue by exploiting rank decompositions.

3.1 Learning Low-Rank Decompositions of Polynomials

The primary insight of our approach is to observe that any *symmetric* tensor $\mathbf{W}^{(l)}$ of order l also admits an additional *rank decomposition* [Nie, 2017a, Brachat et al., 2010] ($\otimes$ being the outer product):

$$\mathbf{W}^{(l)} = \sum_{i=1}^{r} \lambda_i \cdot \underbrace{\boldsymbol{u}_i \otimes \boldsymbol{u}_i \cdots \otimes \boldsymbol{u}_i}_{l \text{ times}}.$$

Where $\{\boldsymbol{u}_i\}_{i=1}^r$ are the (possibly orthonormal) d dimensional basis vectors, $r \in \mathbb{Z}_+$ denotes the *rank* of the tensor, and the scalars $\{\lambda_i\}_{i=1}^r, \lambda_i \in \mathbb{R}$ denote the singular values. Our objective is to directly learn the rank decomposition of an order l tensor, and therefore learn the required polynomial function. This gives us a total of $\mathcal{O}(rd)$ learnable parameters per tensor, down from a previous dependence of $\mathcal{O}(d^l)$. This *low-rank* formulation additionally enables us to compute the polynomial more efficiently. Specifically, for a degree k polynomial with ranks $\boldsymbol{r} = \{1, r_2, \ldots, r_k\}$, we have:

$$P(\boldsymbol{x}) = b + \langle \boldsymbol{u}_1, \boldsymbol{x} \rangle + \sum_{i=1}^{r_2} \lambda_{2i} \cdot \langle \boldsymbol{u}_{2i}, \boldsymbol{x} \rangle^2 + \sum_{i=1}^{r_3} \lambda_{3i} \cdot \langle \boldsymbol{u}_{3i}, \boldsymbol{x} \rangle^3 \ldots + \sum_{i=1}^{r_k} \lambda_{ki} \cdot \langle \boldsymbol{u}_{ki}, \boldsymbol{x} \rangle^k. \quad (1)$$

Where $\{\boldsymbol{u}_{li}\}_{i=1}^{r_l}, \{\lambda_{li}\}_{i=1}^{r_l}$ denote the corresponding bases and eigenvalues for $\mathbf{W}^{(l)}$. The above term can now be easily computed via a simple inner product, and has a time as well as space complexity

of $\mathcal{O}(d\|\boldsymbol{r}\|_1)$, instead of the earlier $\mathcal{O}(d^k)$. Here, $\|\boldsymbol{r}\|_1$ simply represents the sum of ranks of all the k tensors. Note that *low-rank* decompositions do not enjoy a free lunch: the rank r of each weight tensor can, in the worst case, be polynomial in d. However, it is reasonable to assume that correlations in the data $\mathcal{X}$ will ensure that a small r will suffice for real-world problems (see, e.g., Assumption 1).

Optimization. For notational simplicity, let us parameterize any order k polynomial by the weight set $\boldsymbol{\theta} = \{b, \boldsymbol{u}_1, \{\lambda_{2i}, \boldsymbol{u}_{2i}\}_{i=1}^{r_2}, ..., \{\lambda_{ki}, \boldsymbol{u}_{ki}\}_{i=1}^{r_k}\} \in \mathbb{R}^{r(d+1)}$ where from now onwards $r = \|\boldsymbol{r}\|_1 = 1 + r_2 + r_3 + ... + r_k$ denotes the cumulative rank across all the tensors. We write the polynomial parametrized by $\boldsymbol{\theta}$ as $P(\cdot; \boldsymbol{\theta})$, and the learning problem for any loss ℓ is:

$$\text{select } \boldsymbol{\theta}_\star = \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \sum_{i=1}^{n} \ell(P(\boldsymbol{x}_i; \boldsymbol{\theta}), y_i) + \beta \cdot \mathcal{R}(\boldsymbol{\theta}). \tag{2}$$

Here, $\boldsymbol{\Theta}$ denotes the feasible set for $\boldsymbol{\theta}$, and $\mathcal{R}(\cdot)$ denotes an appropriate regularization term scaled by $\beta > 0$. We can show that the above problem is well-behaved for certain data distributions $\mathcal{X}$.

Proposition 1. *If the regularization $\mathcal{R}$ and loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to [0, 1]$ are convex, $\lambda > 0$ and $\mathcal{X} \subset \mathbb{R}_+^d$ then the optimization problem in Equation 2 is convex in $\boldsymbol{\theta}$ for $\boldsymbol{\Theta} \subset \mathbb{R}_+^{r(d+1)}$.*

Proposition 1 suggests that if the training data is positive (achieved by renormalization), typical loss functions (e.g., cross-entropy or mean squared error) and $\mathcal{R}$ such as L_p norms are well-suited for optimization. To ensure convergence, one can use proximal SGD with a small learning rate [Nitanda, 2014]. In practice, we find that unconstrained SGD also provides comparable performance.

3.2 Improving Polynomials for Learning

Geometric rescaling. Learning higher-order interactions is tricky as the order of interactions increases: the product of two features is an order of magnitude different from the original, and consequently, higher-order products are progressively disproportionate. To mitigate this, we rescale features such that (a) the scale is preserved across terms, and (b) the variance in interactions is captured. We replace the input $\boldsymbol{x}$ with $\tilde{\boldsymbol{x}}_l = \text{sign}(\boldsymbol{x}) \cdot |\boldsymbol{x}|^{1/l}$ for an interaction of order l. Specifically,

$$P(\boldsymbol{x}) = \langle \boldsymbol{u}_1, \boldsymbol{x} \rangle + \sum_{i=1}^{r_2} \lambda_{2i} \cdot \langle \boldsymbol{u}_{2i}, \tilde{\boldsymbol{x}}_2 \rangle^2 + \sum_{i=1}^{r_3} \lambda_{3i} \cdot \langle \boldsymbol{u}_{3i}, \tilde{\boldsymbol{x}}_3 \rangle^3 ... + \sum_{i=1}^{r_k} \lambda_{ki} \cdot \langle \boldsymbol{u}_{ki}, \tilde{\boldsymbol{x}}_k \rangle^k + b. \tag{3}$$

We denote this model as SPAM-LINEAR. We argue that this rescaling for unit-bounded features ensures higher interpretability as well as better learning (since all features are of similar variance). Note that for order 1, $\tilde{\boldsymbol{x}}_1 = \boldsymbol{x}$. For $k \geq 2$, consider a pairwise interaction between two features $x_i = 0.5$ and $x_j = 0.6$. Naively multiplying the two will give a feature value of $x_i x_j = 0.3$ (smaller than both x_i and x_j), whereas an intuitive value of the feature would be $\sqrt{x_i x_j} = 0.54$, i.e., the geometric mean. Regarding the reduced variance, observe that unconstrained, the variance $\mathbb{V}(x_i x_j) \leq \mathbb{V}(x_i) \cdot \mathbb{V}(x_j)$ and $\mathbb{V}(\sqrt{x_i x_j}) \leq \mathbb{V}(\sqrt{x_i})\mathbb{V}(\sqrt{x_j}) \leq \sqrt{\mathbb{V}(x_i) \cdot \mathbb{V}(x_j)}$. If the features have small variance, e.g., $\mathbb{V}(x_i) = \mathbb{V}(x_j) = 10^{-2}$ and are uncorrelated, then the first case would provide a much smaller variance for the interaction, whereas the rescaling would preserve variance.

Shared bases for multi-class problems. When we are learning a multi-class classifier, it is observed that learning a unique polynomial for each class creates a large model (with $\mathcal{O}(2drC)$ weights) when the number of classes C is large, which leads to overfitting and issues with regularization. Instead, we propose *sharing* bases ($\boldsymbol{u}$) across classes for all higher order terms, and learning *class-specific* singular values (λ) for all higher order terms (≥ 2) per class. For any input $\boldsymbol{x}$, $P(\boldsymbol{x}; \boldsymbol{\theta}) = \text{SOFTMAX}\{P_c(\boldsymbol{x}; \boldsymbol{\theta})\}_{c=1}^{C}$, where, for any class $c \in \{1, ..., C\}$,

$$P_c(\boldsymbol{x}; \boldsymbol{\theta}) = b_c + \langle \boldsymbol{u}_1^c, \boldsymbol{x} \rangle + \sum_{i=1}^{r_2} \lambda_{2i}^c \cdot \langle \boldsymbol{u}_{2i}, \tilde{\boldsymbol{x}}_2 \rangle^2 + \sum_{i=1}^{r_3} \lambda_{3i}^c \cdot \langle \boldsymbol{u}_{3i}, \tilde{\boldsymbol{x}}_3 \rangle^3 ... + \sum_{i=1}^{r_k} \lambda_{ki}^c \cdot \langle \boldsymbol{u}_{ki}, \tilde{\boldsymbol{x}}_k \rangle^k.$$

The terms in *green* denote weights unique to each class, and terms in *red* denote weights shared across classes. This reduces the model size to $\mathcal{O}((d + r)C + rd)$ from $\mathcal{O}(rdC)$. We set ℓ as the cross-entropy of the softmax of $\{P_c(\cdot, \boldsymbol{\theta})\}_{c=1}^{C}$, as both these operations preserve convexity.

Exploring nonlinear input transformations. Motivated by non-linear GAMs (e.g., GA^{2}M [Lou et al., 2013] and NAM [Agarwal et al., 2021]), we modify our low-rank decomposed algorithm by replacing $\tilde{\boldsymbol{x}}_l$ for each l with feature-wise non-linearities. We learn a non-linear **SPAM** $P(\cdot; \boldsymbol{\theta})$ as,

$$P(\boldsymbol{x}; \boldsymbol{\theta}) = b + \langle \boldsymbol{u}_1, F_1(\boldsymbol{x}) \rangle + \sum_{i=1}^{r_2} \lambda_{2i} \cdot \langle \boldsymbol{u}_{2i}, F_2(\boldsymbol{x}) \rangle^2 + ... + \sum_{i=1}^{r_k} \lambda_{ki} \cdot \langle \boldsymbol{u}_{ki}, F_k(\boldsymbol{x}) \rangle^k.$$

Here, the function $F_i(\boldsymbol{x}) = [f_{i1}(x_1), f_{i2}(x_2), ..., f_{id}(x_d)]$ is a Neural Additive Model (NAM) [Agarwal et al., 2021]. We denote this model as SPAM-NEURAL. For more details on the NAM we use, please refer to the Appendix Section C. Note that the resulting model is still interpretable, as the interaction between features x_i and x_j, e.g., will be given by $(\sum_{k=1}^{r_2} \lambda_{2k} u_{2ki} u_{2kj}) \cdot f_{2i}(x_i) f_{2j}(x_j)$ instead of the previous $(\sum_{k=1}^{r_2} \lambda_{2k} u_{2ki} u_{2kj}) \cdot \sqrt{x_i \cdot x_j}$ for typical SPAM.

Dropout for bases via λ. To capture non-overlapping feature correlations, we introduce a *basis dropout* by setting the contribution from any particular basis to zero at random via the singular values λ. Specifically, observe that the contribution of any basis-singular value pair $(\boldsymbol{u}, \lambda)$ at order l is $\lambda \langle \boldsymbol{u}, \boldsymbol{x} \rangle^l$. We apply dropout to λ to ensure that the network learns robust basis directions.

3.3 Approximation and Learning-Theoretic Guarantees

We present some learning-theoretic guarantees for SPAM. Proofs are deferred to Appendix Section A.

Proposition 2. *Let f be a continuous real-valued function over a compact set $\mathcal{X}$. For any threshold $\epsilon > 0$, there exists a low-rank polynomial $P : \mathcal{X} \to \mathbb{R}$ such that* $\sup_{\boldsymbol{x} \in \mathcal{X}} |f(\boldsymbol{x}) - P(\boldsymbol{x})| < \epsilon$.

The above result demonstrates that asymptotically, decomposed polynomials are universal function approximators. Next, we present an assumption on the data-generating distribution as well as a novel excess risk bound that characterizes learning with polynomials more precisely (non-asymptotic).

Assumption 1 (Exponential Spectral Decay of Polynomial Approximators)**.** *Let $\mathcal{P}_k$ denote the family of all polynomials of degree at most k, and let $P_{\star,k}$ denote the optimal polynomial in $\mathcal{P}_k$, i.e., $P_{\star,k} = \arg\min_{P \in \mathcal{P}_k} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathfrak{P}}[\ell(P(\boldsymbol{x}), y)]$. We assume that $\forall k$, $P_{\star,k}$ admits a decomposition as described in Equation 1 such that, for all $1 \leq l \leq k$, there exist absolute constants $C_1 < 1$ and $C_2 = \mathcal{O}(1)$ such that $|\lambda_{lj}| \leq C_1 \cdot \exp(-C_2 \cdot j^\gamma)$ for each $j \geq 1$ and $l \in [1, k]$.*

This condition is essentially bounding the "smoothness" of the nearest polynomial approximator of f, i.e., implying that only a few degrees of freedom suffice to approximate f accurately. Assumption 1 provides a soft threshold for singular value decay. One can also replace the exponential with a slower "polynomial" decay with similar results, and we discuss this in the Appendix Section A. We can now present our error bound for L_2 regularized models (see Appendix for L_1-regularized models).

Theorem 1. *Let ℓ be 1-Lipschitz, $\delta \in (0, 1]$, the generalization error for the optimal degree-k polynomial $P_{\star,k}$ be $\mathcal{E}(P_{\star,k})$ and the generalization error for the empirical risk minimizer SPAM $\widehat{P}_{\boldsymbol{r},k}$, i.e., $\widehat{P}_{\boldsymbol{r},k} = \arg\min_P \sum_{i=1}^n \ell(P(\boldsymbol{x}_i), y_i)$ with ranks $\boldsymbol{r} = [1, r_2, ..., r_k]$ be $\mathcal{E}(\widehat{P}_{\boldsymbol{r},k})$. Then, if $\mathfrak{P}$ satisfies Assumption 1 with constants C_1, C_2 and γ, we have for L_2-regularized ERM, where $\|\boldsymbol{u}_{ij}\|_2 \leq B_{u,2}\ \forall i \in [k], j \in [r_i]$, and $\|\boldsymbol{\lambda}\|_2 \leq B_{\lambda,2}$ where $\boldsymbol{\lambda} = \{\{\lambda_{ij}\}_{j=1}^{r_i}\}_{i=1}^k$, with probability at least $1 - \delta$,*

$$\mathcal{E}(\widehat{P}_{\boldsymbol{r},k}) - \mathcal{E}(P_{\star,k}) \leq 2B_{\lambda,2} \left(\sum_{l=1}^{k} (B_{u,2})^l \sqrt{r_l} \right) \sqrt{\frac{d}{n}} + \frac{C_1}{C_2} \cdot \left(\sum_{i=2}^{k} \exp(-r_i^\gamma) \right) + 5\sqrt{\frac{\log(4/\delta)}{n}}.$$

The above result demonstrates an expected scaling of the generalization error, matching the bounds obtained for L_1 (see Appendix Section A) and L_2-regularized linear classifiers [Wainwright, 2019]. We that the rank $r = \|\boldsymbol{r}\|_1$ has an $\mathcal{O}(r)$ dependence on the Rademacher complexity. This highlights a trade-off between optimization and generalization: a larger r gives a lower approximation error (smaller second term), but a larger model complexity (larger first term). Observe, however, that even when $r = \Omega((\log d)^p)$ for some $p \geq 1$, the approximation error (second term) diminishes as $o(\frac{1}{d^{p\gamma}})$. This suggests that in practice, one only needs a cumulative rank (poly)logarithmic in d. The above result also provides a non-asymptotic variant of Proposition 2; as $n, r \to \infty$, $\mathcal{E}(\widehat{P}_{\boldsymbol{r},k}) \to \mathcal{E}(P_{\star,k})_+$.

4 Offline Experiments

Here we evaluate SPAM against a set of benchmark algorithms on both classification and regression problems. The baseline approaches we consider are (see Appendix Section C for precise details):

- **Deep Neural Networks (DNN)**: These are standard multi-layer fully-connected neural networks included to demonstrate an upper-bound on performance. These are explained via the perturbation-based LIME [Ribeiro et al., 2016] and SHAP [Lundberg and Lee, 2017] methods in Section 5.
- **Linear / Logistic Regression**: These serve as our baseline interpretable models. We learn both L_1 and L_2-regularized models on unary and pairwise features ($\binom{d}{2}$ features) with minibatch SGD.

Table 1: Evaluation of SPAM on benchmarks against prior work. (↑): higher is better, (↓): lower is better, **: variance across trials is < 0.001, * : variance across trials is < 0.005. **Boldface red** denotes the best **black-box** model, **boldface green** denotes the best **interpretable** model, **boldface** denotes where SPAM Order 3 is best. Runs averaged over 10 random trials with optimal hyperparameters.

Model	**Regression**	**Binary**	**Multi-Class Classification**				**Obj. Det.**
	RMSE (↓)	AUROC (↑)	Accuracy (↑)				mAP (↑)
	CH	FICO	CovType	News	CUB	iNat	CO114
Interpretable Baselines							
Linear (Order 1)	0.7354^{**}	0.7909^{**}	0.7254^{**}	0.8238^{**}	0.7451^{**}	0.3932^{**}	0.1917^{**}
Linear (Order 2)	0.7293^{**}	0.7910^{**}	0.7601^{**}	—	0.7617^{**}	0.4292^{**}	0.2190^{**}
EBMs (Order 1)	0.5586^{**}	0.7985^{**}	0.7392^{**}	—	—	—	—
EB^{2}Ms (Order 2)	0.4919^{**}	0.7998^{**}	—	—	—	—	—
NAM	0.5721^{*}	0.7993^{**}	0.7359^{**}	—	0.7632^{**}	0.4194^{**}	0.2056^{**}
Uninterpretable Black-Box Baselines							
XGBoost	$\mathbf{0.4428}^{**}$	0.7925^{**}	0.8860^{**}	0.7677^{**}	0.7186^{**}	—	—
DNN	0.5014^{**}	$\mathbf{0.7936}^{**}$	$\mathbf{0.9694}^{**}$	$\mathbf{0.8494}^{**}$	$\mathbf{0.7684}^{**}$	$\mathbf{0.4584}^{**}$	$\mathbf{0.2376}^{**}$
Our Interpretable Models							
SPAM (Linear, Order 2)	0.6474^{**}	0.7940^{**}	0.7732^{**}	$\mathbf{0.8472}^{**}$	$\mathbf{0.7786}^{**}$	0.4605^{**}	$\mathbf{0.2361}^{**}$
SPAM (Neural, Order 2)	$\mathbf{0.4914}^{**}$	$\mathbf{0.8011}^{*}$	$\mathbf{0.7770}^{**}$	—	0.7762^{**}	$\mathbf{0.4689}^{**}$	—
SPAM (Linear, Order 3)	0.6410^{**}	0.7945^{**}	0.8066^{**}	$\mathbf{0.8520}^{*}$	0.7741^{**}	0.4684^{**}	$\mathbf{0.2368}^{*}$
SPAM (Neural, Order 3)	0.4865^{*}	$\mathbf{0.8024}^{*}$	0.8857^{**}	—	0.7753^{**}	$\mathbf{0.4722}^{**}$	—

Table 2: Tabular Datasets

Name	California Housing (CH)	FICO	Cover Type (CovType)	Newsgroups
Source	Pace and Barry [1997]	FICO [2018]	Blackard and Dean [1999]	Lang [1995]
Instances	20,640	10,459	581,012	18,828
Features	8	23	54	146,016
Classes	-	2	7	20
Feature Type	Numeric	Mixed	Mixed	TF-IDF

- **Gradient Boosted Trees (XGBoost)**: We use the library `xgboost`. This baseline is mainly to compare accuracy, as the number of trees required are typically large and are hence uninterpretable.
- **Explainable Boosting Machines (EBMs)** [Lou et al., 2013]: EBMs use millions of shallow bagged trees operating on each feature at a time. Note that this approach is not scalable to datasets with many features or multi-class problems. We report scores on the datasets where we successfully trained EBMs without sophisticated engineering, using the interpretml library [Nori et al., 2019].
- **Neural Additive Models (NAMs)** [Agarwal et al., 2021]: These models are neural network extensions of prior EBMs. Note that this method also does not scale to some datasets.

Training Setup. SPAM is implemented by learning L_1/L_2-regularized variants by minibatch SGD implemented in PyTorch. For regression tasks, we measure the root mean squared error (RMSE). For binary classification, we report the area under the ROC (AUROC), for multi-class classification, we report the top-1 accuracy (Acc@1), and finally, for object detection, we report mean Average Precision (mAP). We tune hyperparameters via random sampling approach over a grid. Note that for all experiments, both NAM and SPAM-NEURAL have identical MLP structures to ensure identical approximation power. For definitions of metrics and hyperparameter ranges, see Appendix Section C.

4.1 Measuring Benchmark Performance

We select tasks to explore a variety of settings from regression to multi-class classification, and also explore different dataset scales, from a few hundred samples and tens of features to 100K-scale datasets (both in the number of samples and data dimensionality), while ensuring that the features are interpretable. Our datasets are summarized in Table 2. Please see Appendix Section C.1 for details. For all datasets with no defined train-val-test split, we use a fixed random sample of 70% of the data for training, 10% for validation and 20% for testing. For the 20 Newsgroups dataset, we split the pre-defined training split 7:1 for training and validation, respectively.

In an effort to scale interpretable approaches beyond tabular datasets, we consider benchmark problems using the "Independent Concept Bottleneck" framework of Koh et al. [2020] for image

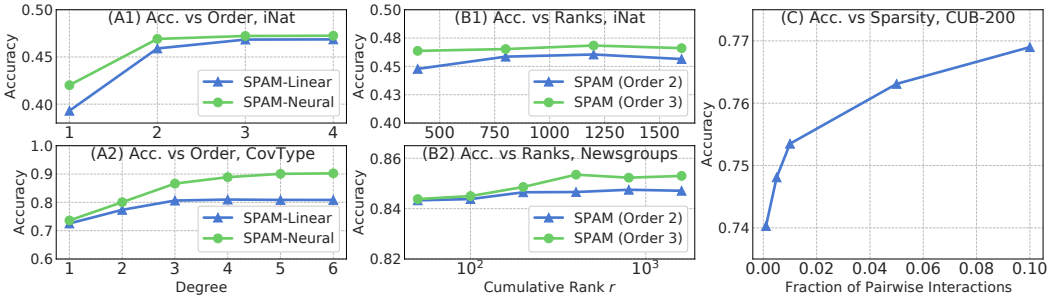

Figure 1: Ablation of Accuracy with: (A1, A2) Degree k; (B1, B2) Rank r; (C) L_1 Sparsity.

classification and object detection. We use a convolutional neural network (CNN) **backbone** ResNet-50 [He et al., 2016] that is trained to predict *interpretable concepts* (e.g., parts of object) from images. After training the **backbone**, we extract predicted concepts for all inputs, and learn an interpretable classifier **head** from these concepts. In experiments, the backbone remains identical for all comparisons, and we only compare the **head** classifiers (see Appendix Section C.3 for more details). We select three datasets for evaluation in the concept bottleneck setting:

1. **Caltech-UCSD Birds (CUB-200)** [Wah et al., 2011]: This is a fine-grained visual categorization dataset where one has to classify between 200 species of birds. The interpretable concepts are 278 *binary* bird attributes, e.g., the shape of the beak, the color of the wings, etc. The dataset has 5,994 training and 5,794 validation images. See Appendix Section C.3.1 for more details.
2. **iNaturalist "Birds"** [Van Horn et al., 2018, 2021]: iNaturalist can be thought of as a larger version of the previous dataset. We only select the "Birds" super-category of samples, which has 414,000 training and 14,860 validation instances across 1,486 classes. Since the iNaturalist dataset does not contain dense part and attribute annotations, we use the *predicted* concepts from the CUB-200 backbone model extracted on all samples.
3. **Common Objects Dataset** (CO114, Proprietary): Here we consider, for the first time, a concept bottleneck model for object detection. We construct a dataset involving common household objects (e.g., bed, saucer, tables, etc.) with their bounding box annotations. For each bounding box, we collect 2,618 interpretable annotations (e.g., parts, color, etc.). The dataset has 2,645,488 training and 58,525 validation samples across 115 classes with 2,618 interpretable concepts. We report the mean Average Precision (mAP) metric. For more details please refer to Appendix Section C.3.2.

Our results are summarized in Table 1. There are four main takeaways from the results. First, observe that both SPAM-LINEAR and SPAM-NEURAL comfortably outperform their prior interpretable counterparts on all datasets (e.g., Order 2 SPAM-LINEAR outperforms all linear methods – even the full rank pairwise model), and SPAM-NEURAL comfortably outperforms all non-linear baselines. Next, observe that in all datasets but CoverType, second degree interactions suffice to match or even outperform DNN performance. We will discuss CoverType in detail in the next paragraph. Thirdly, observe that SPAM models are as scalable as the black-box approaches, whereas prior work (e.g., NAM and EBMs) are not. In the case of NAM, we were unable to scale due to the sheer number of parameters required to model large datasets, e.g., Newsgroups ($\approx$900M parameters), and for EBMs, the training time and memory requirement increased dramatically for large datasets. Furthermore, EB^{2}Ms do not even support multi-class problems. Finally, observe that for many problems, we do not require *non-linear* feature transformations once we model feature interactions: e.g., iNat and CUB.

CoverType Dataset. XGBoost and DNNs perform substantially better on CoverType compared to linear or neural SPAM. On analysis, we found that existing SPAM models underfit on CoverType, and hence we increased the total parameters for SPAM-NEURAL via *subnets*, identical to NAMs [Agarwal et al., 2021], where each feature is mapped to s non-linear features (as opposed to 1 originally). With $s = 8$ subnets, the performance of SPAM-NEURAL (order 3) improves to 0.9405. In comparison, NAMs with 8 subnets provides a lower accuracy of 0.7551.

4.2 Ablation Studies

Rank and Degree of Interactions. By Proposition 2, it is natural to expect the approximation quality to increase with the degree k and rank r, i.e., as we introduce more higher-order interactions

Table 3: Throughput benchmarking (higher is faster).

Method	CH	FICO	CovType	News
	Throughput (images / second)			
NAM [Agarwal et al., 2021]	5×10^5	1.2×10^5	8×10^4	23
NAM (Order 2)	1.1×10^4	6×10^3	3×10^3	-
LinearSPAM (Order 2)	6.1×10^7	6.7×10^7	6.1×10^7	2.6×10^6
NeuralSPAM (Order 2)	1.7×10^5	7.9×10^3	4.1×10^3	-
LinearSPAM (Order 3)	3.2×10^7	3.7×10^7	3.9×10^7	1.8×10^5
NeuralSPAM (Order 3)	1.1×10^5	5.3×10^3	2.6×10^3	-
MLP	1.3×10^7	1.3×10^7	1.3×10^7	2.2×10^5

between features. We ablate the degree k on the tabular benchmark **CoverType** and concept bottleneck benchmark **iNaturalist (Birds)**, as summarized in Figure 1A. We observe, as expected, that increasing the degree k leads to moderate improvements beyond $k \geq 3$, but with diminishing returns. Similarly, we examine the effect of the cumulative rank r on performance on the **Newsgroups** and **iNaturalist (Birds)** datasets (Figure 1B); we observe that performance is sufficiently insensitive to r, and plateaus after a while. We imagine that as r increases, model complexity will dominate and performance will likely begin to decrease, matching the full-rank performance at $r = \mathcal{O}(d^2)$ for pairwise models.

Sparsity in Higher-Order Relationships. A requirement for interpretability is to ensure that the learned models can be explained with a few interpretable concepts. Since the number of higher-order combinations increases with k, we examine sparsity to limit the number of active feature assignments. We penalize the objective in Equation 2 with a regularization term that inhibits dense feature selection. If $\mathbf{U} = \{\{\boldsymbol{u}_{li}\}_{i=1}^{r_l}\}_{l=1}^{k}$ denotes all the basis vectors in matrix form, we add the penalty $\mathcal{R}(\boldsymbol{\theta}) \triangleq \|\mathbf{U}\|_1$. This ensures that every basis $\boldsymbol{u}$ only captures limited interactions between features, and hence the overall complexity of the model is limited. We examine accuracy as a function of the fraction of non-zero pairwise interactions for a degree 2, rank 800 SPAM on CUB-200 in Figure 1C, and find that only 6% of the possible interactions suffice to obtain competitive performance.

Examining Assumption 1 (Spectral Decay). The spectral decay assumption presented in Assumption 1 is crucial to obtain reasonable generalization bounds (Theorem 1). To test this assumption, we examine the spectra of a SPAM-LINEAR (Order 2) model for all 200 of CUB-200 classes. These are depicted in Figure 2A. Furthermore, we fit an exponential model on the decay itself, and obtain (in line with Assumption 1) $\gamma = 3$, $C_1 = 0.54$ and $C_2 = 0.006$, all in accordance with the assumption.

4.3 Additional Experiments

4.3.1 Comparisons on Common Benchmarks

In addition to the 7 datasets we considered earlier, we evaluate on 9 further benchmark datasets commonly used in the interpretability literature. We evaluate on the MIMIC2, Credit and COMPAS datasets as presented in Agarwal et al. [2021], and the Click, Epsilon, Higgs, Microsoft, Yahoo and Year datasets from Chang et al. [2021]. The result of this is summarized in Table 4. We observe that SPAM models are competitive with prior work, and outperforming prior work on a number of tasks (the best interpretable model is in **bold**).

4.3.2 Runtime Evaluation

We provide a comparison of SPAM runtimes on 4 different datasets to establish their scalability. We consider the California Housing (CH, Pace and Barry [1997], 8 features), FICO HELOC (FICO, FICO [2018], 23 features), CoverType (CovType, Blackard and Dean [1999], 54 features) and 20 Newsgroups (News, Lang [1995], 146,016 features), and evaluate the throughput of the optimal Linear and Neural SPAM models against Neural Additive Models (NAM, Agarwal et al. [2021]) and MLPs. Table 3 describes the throughput, where we observe that Linear SPAM is more efficient than MLPs, and NeuralSPAM is orders of magnitude faster than NAM, which does not scale to higher orders.

Table 4: Comparison against state-of-the-art interpretable neural networks on additional benchmarks.

Method	MIMIC2 (AUC)	Credit (AUC)	COMPAS (AUC)	Click (ERR)	Epsilon (ERR)	Higgs (ERR)	Microsoft (MSE)	Yahoo (MSE)	Year (MSE)
NAM [Agarwal et al., 2021]	0.8539	0.9766	0.7368	0.3447	0.1079	0.2972	0.5824	0.6093	85.25
NODE-GAM [Chang et al., 2021]	0.8320	0.9810	0.7420	**0.3342**	0.1040	0.2970	0.5821	0.6101	85.09
LinearSPAM (Order 2)	0.8514	0.9836	**0.7426**	0.3791	**0.1011**	0.2881	0.5710	0.5923	81.30
NeuralSPAM (Order 2)	**0.8664**	**0.9850**	0.7411	0.3348	0.1020	**0.2750**	**0.5671**	**0.5869**	**79.99**
XGBoost	0.8430	0.9780	0.7440	0.3334	0.1112	0.2328	0.5544	0.5420	78.53

Table 5: Feature importances for interpretable models used in human subject evaulations.

Model	Importances for input $\boldsymbol{x} = \{x_1, ..., x_d\}$
Linear / *post-hoc* Linear	$w_i \cdot x_i, i \in \{1, ..., d\}$
SPAM (Linear, Order 2)	$u_{1i} \cdot x_i, i \in \{1, ..., d\}$ and $\left(\sum_{l=1}^{r_2} \lambda_{2l} u_{2li} u_{2lj}\right) \cdot \sqrt{x_i \cdot x_j}, i, j \in [d]$

5 Human Subject Evaluations

Experiment Methodology. We now evaluate how well explanations from **SPAM** fare *in a practical setting* with non-interpretable benchmarks such as black-box models equipped with *post-hoc* explanations. Our objective is to mimic a practical setting, where the model user must decide between a fully-interpretable model vs. a black-box classifier with *post-hoc* explainability. Our experiment design is a Prediction Task [Hoffman et al., 2018, Muramatsu and Pratt, 2001], where the objective for the users is to guess what the model predicted given an explanation. The motivation of such a design is to ascertain both the *faithfulness* and *interpretability* of the model explanations.

Computing Explanations. We first compute the feature importances for any class c, as the *contribution* of any specific feature in the logit for class c (for binary classification or regression, there is only one class). For the Logisitc Regression model, we simply use the contribution of each feature to the prediction. For SPAM-Linear (Order 2), we still use each feature's contribution, but features now can be unary (x_i) or pairwise (x_i and x_j). As black-box *post-hoc* baselines, we consider DNNs with LIME [Ribeiro et al., 2016] and KernelSHAP [Lundberg and Lee, 2017] to generate *post-hoc* linear explanations. See Table 5 for the formulation of importance for the models we use in experiments.

Experiment Design We conduct a separate experiment for each model $\mathbf{M}$ with explanations of length E by selecting E most important features. Each such experiment runs in two phases - *training* and *testing*. In the *training* phase, participants are shown 8 images of birds and their corresponding explanations to develop their mental model. Four of these images have been predicted as class **A** by the model, and the remaining are predicted as class **B**. They then move on to the *testing* phase, where they are successively shown the explanations for 5 unseen images (which the model could have predicted as either **A** or **B**), and the users must answer *"given the explanation, which class (of* **A** *or* **B***) did the model predict?"*. If they desire, the users can move back to the training phase at any time to revise model explanations. We do not show the corresponding images in the test phase, as we want the user feedback to rely solely on the faithfulness of explanations. We measure the *mean user accuracy* on predicting model decisions. For more details on the interface, please see Appendix Section D.1. For each experiment, we follow an independent-subjects design, and gathered a total of 150 participants per experiment. Each task (corresponding to one model-explanation pair) lasted 3 minutes. The participants were compensated with \$0.75 USD (\$15/hour), and all experiments were run on Amazon Mechanical Turk (AMT, Buhrmester et al. [2016]) with a total cost of \$3000. To remove poorly performing participants, we only select those that get the first decision correct, and compute the mean user accuracy using the remaining 4 images.

Study #1: Comparing Black-Box and Transparent Explanations. We compare different interpretable models with a *fixed* explanation length of $E = 7$. Our objective is to assess the interpretability of pairwise interactions compared to black-box and linear explanations, and to re-answer whether it was even necessary to have fully-interpretable models compared to black-box models with *post-hoc* explanations. We used samples from the 5 pairs of classes, that is **A** and **B**), from CUB-200 dataset (see Section D.1 for images). The results from this experiment are summarized in Figure 2B. Observe that with SPAM, the mean user accuracy (mUA) is substantially higher (0.71) compared to both linear (0.67) and *post-hoc* LIME (0.65). We would like to remark that even for $E = 7$, SPAM has an

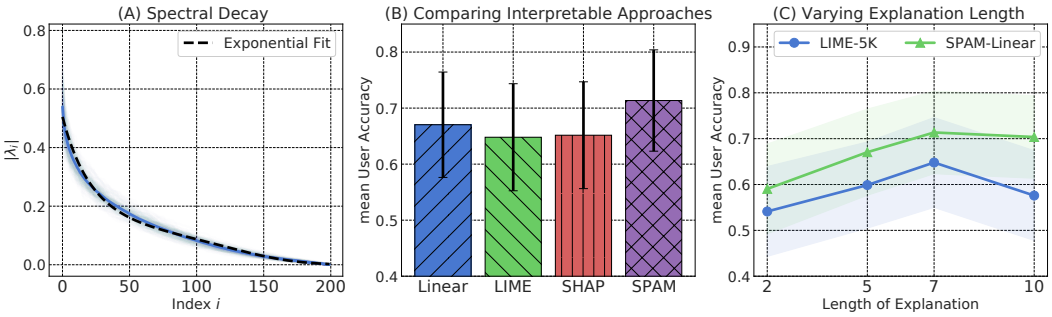

Figure 2: (A) Spectral Decay on CUB-200; Human Subject Evaluation results for (B) Comparing Black-box and glass-box explanations, and (C) LIME vs. SPAM with different explanation lengths.

average of 3.1 pairwise terms in each explanation, and therefore the higher-order terms are substantial. A one-sided t-test [Freund and Simon, 1967] provides a significance ($p-$value) of 2.29×10^{-4} and a test statistic of 3.52 for the SPAM mUA being higher than LIME, and correspondingly for Linear, we get a significance ($p-$value) of 0.03801 and a test statistic of 1.666. Hence SPAM's improvements in interpretability are statistically significant over both linear and *post-hoc* approaches.

Study #2: Varying Explanation Length. One can argue that the increased interpretability of polynomial approaches is due to the fact that each higher-order "term" involves multiple features (e.g., pairwise involves two features), and hence the increased mean user accuracy (mUA) is due to the larger number of features shown (either individually or as pairs). We examine this hypothesis by varying the explanation length E for both SPAM-Linear (Order 2) and LIME, which consequently increases the number of terms seen for both approaches. We observe in Figure 2C that regardless of explanation method, mUA is maximum at $E = 7$, and introducing more terms decreases interpretability.

6 Discussion and Conclusion

We presented a simple and scalable approach for modeling higher-order interactions in interpretable machine learning. Our approach alleviates several of the concerns with existing algorithms and focuses on presenting a viable alternative for practical, large-scale machine learning systems. In addition to offline experimental guarantees, we demonstrate by an extensive user study that these models are indeed more interpretable in practice, and therefore can readily substitute *post-hoc* interpretability without any loss in performance. Moreover, to the best of our knowledge, our work provides the first incremental analysis of interpretability and performance: we show how progressively increasing the model complexity (with higher order interactions) brings better performance while compromising interpretability in practice, and the "sweet spot" appears to rest at pairwise interactions.

We have several follow-up directions. First, given that our model provides precise guarantees on generalization, it is a feasible starting point to understanding tradeoffs between privacy and explainability from a rigorous perspective. Next, one can consider utilizing SPAM to understand failure modes and spurious correlations directly. Furthermore, SPAM-esque decompositions can also be useful in other domains beyond interpretability, e.g., language and vision, where modeling higher-order interactions is challenging due to the curse of dimensionality.

References

R. Agarwal, L. Melnick, N. Frosst, X. Zhang, B. Lengerich, R. Caruana, and G. E. Hinton. Neural additive models: Interpretable machine learning with neural nets. *Advances in Neural Information Processing Systems*, 34, 2021.

S. O. Arık and T. Pfister. Tabnet: Attentive interpretable tabular learning. In *AAAI*, volume 35, pages 6679–6687, 2021.

A. Asuncion and D. Newman. Uci machine learning repository, 2007.

S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10 (7):e0130140, 2015.

P. L. Bartlett and S. Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

B. Bercu, B. Delyon, and E. Rio. *Concentration inequalities for sums and martingales*. Springer, 2015.

J. A. Blackard and D. J. Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, 24(3):131–151, 1999.

J. Brachat, P. Comon, B. Mourrain, and E. Tsigaridas. Symmetric tensor decomposition. *Linear Algebra and its Applications*, 433(11-12):1851–1872, 2010.

M. Buhrmester, T. Kwang, and S. D. Gosling. Amazon's mechanical turk: A new source of inexpensive, yet high-quality data? 2016.

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

S. Chakraborty, R. Tomsett, R. Raghavendra, D. Harborne, M. Alzantot, F. Cerutti, M. Srivastava, A. Preece, S. Julier, R. M. Rao, et al. Interpretability of deep learning models: A survey of results. In *2017 IEEE smartworld, ubiquitous intelligence & computing, advanced & trusted computed, scalable computing & communications, cloud & big data computing, Internet of people and smart city innovation (smartworld/SCALCOM/UIC/ATC/CBDcom/IOP/SCI)*, pages 1–6. IEEE, 2017.

C.-H. Chang, R. Caruana, and A. Goldenberg. Node-gam: Neural generalized additive model for interpretable deep learning. *arXiv preprint arXiv:2106.01613*, 2021.

C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, and J. K. Su. This looks like that: deep learning for interpretable image recognition. *Advances in neural information processing systems*, 32, 2019.

G. Chrysos, S. Moschoglou, Y. Panagakis, and S. Zafeiriou. Polygan: High-order polynomial generators. *arXiv preprint arXiv:1908.06571*, 2019.

G. G. Chrysos, S. Moschoglou, G. Bouritsas, Y. Panagakis, J. Deng, and S. Zafeiriou. P-nets: Deep polynomial neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7325–7335, 2020.

C. K. Chui and X. Li. Realization of neural networks with one hidden layer. In *Multivariate approximation: From CAGD to wavelets*, pages 77–89. World Scientific, 1993.

M. Du, N. Liu, and X. Hu. Techniques for interpretable machine learning. *Communications of the ACM*, 63(1):68–77, 2019.

FICO. Fico community explainable machine learning challenge, 2018. URL https://community.fico.com/s/explainable-machine-learning-challenge?tabset-3158a=2.

J. E. Freund and G. A. Simon. *Modern elementary statistics*, volume 256. Prentice-Hall Englewood Cliffs, NJ, 1967.

J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

J. H. Friedman and B. E. Popescu. Predictive learning via rule ensembles. *The annals of applied statistics*, 2(3):916–954, 2008.

A. Ghorbani, A. Abid, and J. Zou. Interpretation of neural networks is fragile. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3681–3688, 2019a.

A. Ghorbani, J. Wexler, J. Y. Zou, and B. Kim. Towards automatic concept-based explanations. *Advances in Neural Information Processing Systems*, 32, 2019b.

T. J. Hastie and R. J. Tibshirani. *Generalized additive models*. Routledge, 2017.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

R. R. Hoffman, S. T. Mueller, G. Klein, and J. Litman. Metrics for explainable ai: Challenges and prospects. *arXiv preprint arXiv:1812.04608*, 2018.

A. G. Ivakhnenko. Polynomial theory of complex systems. *IEEE transactions on Systems, Man, and Cybernetics*, (4):364–378, 1971.

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

H. Lakkaraju and O. Bastani. " how do i fool you?" manipulating user trust via misleading black box explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 79–85, 2020.

K. Lang. Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings 1995*, pages 331–339. Elsevier, 1995.

T. Laugel, M.-J. Lesot, C. Marsala, X. Renard, and M. Detyniecki. The dangers of post-hoc interpretability: Unjustified counterfactual explanations. *arXiv preprint arXiv:1907.09294*, 2019.

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

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Y. Lou, R. Caruana, J. Gehrke, and G. Hooker. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–631, 2013.

S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

A. Madsen, S. Reddy, and S. Chandar. Post-hoc interpretability for neural nlp: A survey. *arXiv preprint arXiv:2108.04840*, 2021.

P. Massart. Some applications of concentration inequalities to statistics. In *Annales de la Faculté des sciences de Toulouse: Mathématiques*, volume 9, pages 245–303, 2000.

A. H. Miller, W. Feng, A. Fisch, J. Lu, D. Batra, A. Bordes, D. Parikh, and J. Weston. Parlai: A dialog research software platform. *arXiv preprint arXiv:1705.06476*, 2017.

J. Muramatsu and W. Pratt. Transparent queries: investigation users' mental models of search engines. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 217–224, 2001.

J. Nie. Generating polynomials and symmetric tensor decompositions. *Foundations of Computational Mathematics*, 17(2):423–465, 2017a.

J. Nie. Low rank symmetric tensor approximations. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1517–1540, 2017b.

A. Nitanda. Stochastic proximal gradient descent with acceleration techniques. *Advances in Neural Information Processing Systems*, 27, 2014.

H. Nori, S. Jenkins, P. Koch, and R. Caruana. Interpretml: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223*, 2019.

S.-K. Oh, W. Pedrycz, and B.-J. Park. Polynomial neural networks architecture: analysis and design. *Computers & Electrical Engineering*, 29(6):703–725, 2003.

R. K. Pace and R. Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3): 291–297, 1997.

B. Recht. A simpler approach to matrix completion. *Journal of Machine Learning Research*, 12(12), 2011.

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

C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

Y. Shin and J. Ghosh. The pi-sigma network: an efficient higher-order neural network for pattern classification and function approximation. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, volume i, pages 13–18 vol.1, 1991. doi: 10.1109/IJCNN.1991.155142.

A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje. Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713*, 2016.

A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR, 2017.

D. Slack, A. Hilgard, S. Singh, and H. Lakkaraju. Reliable post hoc explanations: Modeling uncertainty in explainability. *Advances in Neural Information Processing Systems*, 34, 2021.

M. H. Stone. The generalized weierstrass approximation theorem. *Mathematics Magazine*, 21(5): 237–254, 1948.

M. Tsang, H. Liu, S. Purushotham, P. Murali, and Y. Liu. Neural interaction transparency (nit): Disentangling learned interactions for improved interpretability. *Advances in Neural Information Processing Systems*, 31, 2018.

G. Van Horn, O. Mac Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778, 2018.

G. Van Horn, E. Cole, S. Beery, K. Wilber, S. Belongie, and O. Mac Aodha. Benchmarking representation learning for natural world image collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12884–12893, 2021.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

M. J. Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge University Press, 2019.

K. Weierstrass. On the analytical representability of so-called arbitrary functions of a real variable. *Meeting reports of the Royal Prussian Academy of Sciences in Berlin*, 2:633–639, 1885.

Z. Yang, C. Jin, Z. Wang, M. Wang, and M. I. Jordan. On function approximation in reinforcement learning: Optimism in the face of large state spaces. *arXiv preprint arXiv:2011.04622*, 2020.

R. Zhang, P. Madumal, T. Miller, K. A. Ehinger, and B. I. Rubinstein. Invertible concept-based explanations for cnn models with non-negative concept activation vectors. *arXiv preprint arXiv:2006.15417*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See conclusion.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Appendix Section A.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix Section A.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Please see Section 4 and supplementary material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Please see Sections 5 and 4.
   (b) Did you mention the license of the assets? [Yes] See accompanying supplementary material.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A] No new assets except code, see 3(a).
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [Yes] See supplementary material.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] No participant risks were identified in the experiment design.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [Yes] See Section 5.

A Deferred Proofs

A.1 Proof of Proposition 1

This follows from noting that the Hessian of the objective function is positive definite for all $\lambda > 0$ when the constraints in the proposition hold.

A.2 Proof of Proposition 2

This follows directly from the Stone-Weierstrass Theorem [Stone, 1948, Weierstrass, 1885] and Theorem 1 of Chui and Li [1993].

A.3 Proof of Theorem 1

We state Theorem 1 in full here.

Theorem 2. *Let ℓ be a 1-Lipschitz loss, $\delta \in (0,1]$ and the generalization error for the optimal degree k polynomial $P_{\star,k}$ be given by $\mathcal{E}(P_{\star,k})$ and the training error for the ERM SPAM $\widehat{P}_{\boldsymbol{r},k}$, i.e., $\widehat{P}_{\boldsymbol{r},k} = \arg\min_P \sum_{i=1}^n \ell(P(\boldsymbol{x}_i, y_i)$ with ranks $\boldsymbol{r} = [1, r_2, ..., r_k]$ be given by $\widehat{\mathcal{E}}_n(\widehat{P}_{\boldsymbol{r},k})$. Then, for L_2-regularized models, where $\|\boldsymbol{u}_{ij}\|_2 \leq B_{u,2}$ for all $i \in [k], j \in [r_i]$, and $\|\boldsymbol{\lambda}\|_2 \leq B_{\lambda,2}$ where $\boldsymbol{\lambda} = \{\{\lambda_{ij}\}_{j=1}^{r_i}\}_{i=1}^k$, we have that with probability at least $1-\delta$ there exists an absolute constant C such that,*

$$\mathcal{E}(\widehat{P}_{\boldsymbol{r},k}) - \mathcal{E}(P_{\star,k}) \leq 2B_{\lambda,2}\left(\sum_{l=1}^k (B_{u,2})^l \sqrt{r_l}\right)\sqrt{\frac{d}{n}} + C\cdot\left(\sum_{i=2}^k \exp(-r_i^\gamma)\right) + 5\sqrt{\frac{\log(4/\delta)}{n}}.$$

For L_1-regularization, $\|\boldsymbol{u}_{ij}\|_1 \leq B_{u,1}$ for all $i \in [k], j \in [r_i]$, and $\|\boldsymbol{\lambda}\|_1 \leq B_{\lambda,1}$ where $\boldsymbol{\lambda} = \{\{\lambda_{ij}\}_{j=1}^{r_i}\}_{i=1}^k$, we have with probability at least $1-\delta$ there exists an absolute constant C such that,

$$\mathcal{E}(\widehat{P}_{\boldsymbol{r},k}) - \mathcal{E}(P_{\star,k}) \leq 2B_{\lambda,1}\left(\sum_{l=1}^k (B_{u,1})^l\right)\sqrt{\frac{\log(d)}{n}} + C\cdot\left(\sum_{i=2}^k \exp(-r_i^\gamma)\right) + 5\sqrt{\frac{\log(4/\delta)}{n}}.$$

While we will eventually prove Theorem 1, the approach is to first establish the necessary mathematical background and outline the proof approach. We will provide a general result under arbitrary Lipschitz loss functions via a metric entropy bound for low-rank polynomial approximations. We will denote, for any $k > 1$, the space of all *low-rank* polynomials of order k with cumulative rank $\boldsymbol{r} = [1, r_1, r_2, ..., r_k] \in \mathbb{R}^k$ as $\boldsymbol{\Theta}_{\boldsymbol{r}} \subset \mathbb{R}_+^{d(r+1)}$ where $r = \|\boldsymbol{r}\|_1$. Recall from Section 3 that any $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ is composed of the components, $\{b, \{\{\lambda_{ij}\}_{j=1}^{r_i}\}_{i=1}^k, \{\{\boldsymbol{u}_{ij}\}_{j=1}^{r_i}\}_{i=1}^k\}$ and we use $\boldsymbol{\theta}$ as a shorthand.

Now, for any function $f : \mathcal{X} \to \mathcal{Y}$ and any bounded $L-$Lipschitz loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to [0,1]$ (we assume, without loss of generality, that the range of ℓ is the closed interval $[0,1]$), the *training error* over n samples as,

$$\widehat{\mathcal{E}}_n(f) = \frac{1}{n}\sum_{i=1}^n \left[\ell(f(\boldsymbol{x}_i), y_i)\right].$$

Similarly, we can define the *expected risk* over the sample distribution $\mathfrak{P} : \mathcal{X} \times \mathcal{Y} \to [0,1)$ as,

$$\mathcal{E}(f) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathfrak{P}}\left[\ell(f(\boldsymbol{x}), y)\right].$$

At a high level, our objective is to bound the *expected risk* of the ERM *low-rank decomposed polynomial* (LRDP) $\mathcal{E}(\widehat{P}_{\boldsymbol{r},k})$ of some degree k and (cumulative) rank r with risk of the *optimal* polynomial of degree k, i.e., $\mathcal{E}(P_{\star,k})$. Note that $P_{\star,k}$ is not necessarily low-rank and can have cumulative rank of order $\mathcal{O}(d^k)$. Let us denote the maximum possible rank of any polynomial of degree k as $\bar{r}$ and the corresponding set of ranks as $\bar{\boldsymbol{r}}$. Observe that any polynomial of degree k has a (non-unique) representation in $\boldsymbol{\Theta}_{\bar{\boldsymbol{r}}}$, and furthermore,

$$P_{\star,k} = \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}_{\bar{\boldsymbol{r}}}} \left(\mathcal{E}(P(\cdot;\boldsymbol{\theta}))\right). \tag{4}$$

Similarly, we have that the *empirical risk minimizer* LRDP $\widehat{P}_{\boldsymbol{r},k}$ of degree k and ranks $\boldsymbol{r}$ satisfies,

$$\widehat{P}_{\boldsymbol{r},k} = \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}_{\boldsymbol{r}}} \left(\widehat{\mathcal{E}}_n(P(\cdot;\boldsymbol{\theta}))\right). \tag{5}$$

We can now begin the proof. For the optimal polynomial $P_{\star,k}$, let us denote the corresponding singular values as $\{\{\lambda_{ij}^\star\}_{j=1}^{\bar{r}_i}\}_{i=1}^k$, bases as $\{\{\boldsymbol{u}_{ij}^\star\}_{j=1}^{\bar{r}_i}\}_{i=1}^k$ and bias as $b_{\star,k}$. We will bound the *excess risk* $\mathcal{E}(\widehat{P}_{\boldsymbol{r},k}) - \mathcal{E}(P_{\star,k})$ by introducing a third polynomial $\widetilde{P}_{\boldsymbol{r},k} \in \boldsymbol{\Theta}_{\boldsymbol{r}}$ which is defined as a *truncated* version of $P_{\star,k}$ up to the ranks $\boldsymbol{r}$. Specifically, we set the singular values $\{\{\widetilde{\lambda}_{ij}\}_{j=1}^{r_i}\}_{i=1}^k$,

bases as $\{\{\widetilde{\boldsymbol{u}}_{ij}\}_{j=1}^{r_i}\}_{i=1}^{k}$ and bias as $\widetilde{b}_k$ as,

$$\text{For } 1 \leq j \leq r_i, 1 \leq i \leq k, \;\; \widetilde{\lambda}_{ij} = \lambda^\star_{ij}, \widetilde{\boldsymbol{u}}_{ij} = \boldsymbol{u}^\star_{ij}, \text{ and } \widetilde{b}_k = b^\star_k. \tag{6}$$

The polynomial $\widetilde{P}_{\boldsymbol{r},k}$ can therefore be thought of as a "truncated" version of the optimal polynomial $P_{\star,k}$ up to the ranks $\boldsymbol{r}$. Observe that we can then write the excess risk as,

$$\mathcal{E}(\widehat{P}_{\boldsymbol{r},k}) - \mathcal{E}(P_{\star,k}) = \underbrace{\mathcal{E}(\widehat{P}_{\boldsymbol{r},k}) - \widehat{\mathcal{E}}_n(\widehat{P}_{\boldsymbol{r},k})}_{\text{Ⓐ}} + \underbrace{\widehat{\mathcal{E}}_n(\widehat{P}_{\boldsymbol{r},k}) - \widehat{\mathcal{E}}_n(\widetilde{P}_{\boldsymbol{r},k})}_{\leq 0} + \underbrace{\widehat{\mathcal{E}}_n(\widetilde{P}_{\boldsymbol{r},k}) - \mathcal{E}(P_{\star,k})}_{\text{Ⓑ}}.$$

The term $\widehat{\mathcal{E}}_n(\widehat{P}_{\boldsymbol{r},k}) - \widehat{\mathcal{E}}_n(\widetilde{P}_{\boldsymbol{r},k}) \leq 0$ since $\widehat{P}_{\boldsymbol{r},k}$ minimizes the empirical risk within $\boldsymbol{\Theta}_{\boldsymbol{r}}$ (Equation 5). Hence bounding terms Ⓐ and Ⓑ will provide us the bound. We will first bound term Ⓑ via Lemma 1. We have that with probability at least $1 - \delta/2$ for any $\delta \in (0, 1]$,

$$\left|\widehat{\mathcal{E}}_n(\widetilde{P}_{\boldsymbol{r},k}) - \mathcal{E}(P_{\star,k})\right| \leq LC \cdot \left(\sum_{i=2}^{k} \exp(-r_i^\gamma)\right) + 2\sqrt{\frac{\log(2/\delta)}{n}}.$$

To bound term Ⓐ, we proceed via bounding the Rademacher complexity, a classic approach in statistical learning [Wainwright, 2019]. We will first establish a bound on the Rademacher complexity of $\boldsymbol{\Theta}_{\boldsymbol{r}}$ under L^1 and L^2-regularization. Note that similar bounds via directly bounding the metric entropy or Dudley's entropy can also be obtained, but we present this approach for simplicity. We anticipate that unless alternative assumptions are made about the polynomials, the bounds will remain identical via these approaches. Observe first, that since the loss function is Lipschitz and bounded, we have with probability at least $1 - \delta/2, \delta \in (0, 1]$, from Theorem 4 and Theorem 8 of Bartlett and Mendelson [2002],

$$\mathcal{E}(\widehat{P}_{\boldsymbol{r},k}) - \widehat{\mathcal{E}}_n(\widehat{P}_{\boldsymbol{r},k}) \leq \mathfrak{R}_n(\ell \odot \mathcal{P}(\boldsymbol{\Theta}_{\boldsymbol{r}})) + \sqrt{\frac{8\log(4/\delta)}{n}}. \tag{7}$$

Where $\mathfrak{R}_n$ denotes the empirical Rademacher complexity at n samples [Bartlett and Mendelson, 2002], and $\mathcal{P}(\boldsymbol{\Theta}_{\boldsymbol{r}})$ denotes the set of all polynomials represented by the parameterization $\boldsymbol{\Theta}_{\boldsymbol{r}}$. Now, observe that any element in $\mathcal{P}(\boldsymbol{\Theta}_{\boldsymbol{r}})$ comprises $k + 1$ polynomials of a fixed degree (one for each degree from 0 to k). With some abuse of notation, let us denote the family of all polynomials with a degree exactly l and rank exactly r as $\mathcal{P}_{l,r}$. Observe that by Theorem 4,

$$\mathfrak{R}_n(\mathcal{P}(\boldsymbol{\Theta}_{\boldsymbol{r}})) \leq \sum_{l=0}^{k} \mathfrak{R}_n(\mathcal{P}_{l,r_l}) = \sum_{l=1}^{k} \mathfrak{R}_n(\mathcal{P}_{l,r_l}). \tag{8}$$

The last equality follows from the fact that $\mathcal{P}_{0,\cdot}$ is always a constant function with a Rademacher complexity of 0. Furthermore, since ℓ is Lipschitz, we have once again by Theorem 4 that $\mathfrak{R}_n(\ell \odot \mathcal{P}(\boldsymbol{\Theta}_{\boldsymbol{r}})) \leq 2L \cdot \mathfrak{R}_n(\mathcal{P}(\boldsymbol{\Theta}_{\boldsymbol{r}}))$. Putting this together, we have that with probability at least $1 - \delta/2, \delta \in (0, 1]$,

$$\mathcal{E}(\widehat{P}_{\boldsymbol{r},k}) - \widehat{\mathcal{E}}_n(\widehat{P}_{\boldsymbol{r},k}) \leq 2L \cdot \sum_{l=1}^{k} \mathfrak{R}_n(\mathcal{P}_{l,r_l}) + \sqrt{\frac{8\log(4/\delta)}{n}}. \tag{9}$$

Now, to bound the remaining term, we consider two regularization constraints for $\boldsymbol{\Theta}_{\boldsymbol{r}}$. The first is L_2-regularization, where we assume that the parameter $\lambda, \boldsymbol{u}$ are bounded in L_2. Let $\boldsymbol{\lambda} = \{\{\lambda_{ij}\}_{j=1}^{r_i}\}_{i=1}^{k}$ denote the singular values across all ranks. Then, we assume for all $i \in [1, ..., k]$ and $j \in [1, ..., r_i]$ that $\|\boldsymbol{u}_{ij}\|_2 \leq B_{u,2}$ and $\|\boldsymbol{\lambda}\|_2 \leq B_{\lambda,2}$. Under this constraint, we can bound the empirical Rademacher complexity of each polynomial via Lemma 2. Replacing this result, we obtain with probability at least $1 - \delta/2, \delta \in (0, 1]$,

$$\mathcal{E}(\widehat{P}_{\boldsymbol{r},k}) - \widehat{\mathcal{E}}_n(\widehat{P}_{\boldsymbol{r},k}) \leq 2LB_{\lambda,2}B_2^x \cdot \sum_{l=1}^{k} (B_{u,2})^l \sqrt{\frac{r}{n}} + \sqrt{\frac{8\log(4/\delta)}{n}}. \tag{10}$$

Where $B_2^x = \sup_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{x}\|_2$. Without loss of generality we can assume $B_2^x = \sqrt{d}$. Now, for L_1-regularized models, we assume that for all $i \in [1, ..., k]$ and $j \in [1, ..., r_i]$ that $\|\boldsymbol{u}_{ij}\|_1 \leq B_{u,1}$ and $\|\boldsymbol{\lambda}\|_1 \leq B_{\lambda,1}$. This gives us the following bound via Lemma 3:

$$\mathcal{E}(\widehat{P}_{\boldsymbol{r},k}) - \widehat{\mathcal{E}}_n(\widehat{P}_{\boldsymbol{r},k}) \leq 2LB_{\lambda,1}B_\infty^x \cdot \sum_{l=1}^{k} (B_{u,1})^l \cdot \sqrt{\frac{\log(d)}{n}} + \sqrt{\frac{8\log(4/\delta)}{n}}. \tag{11}$$

Where $B_\infty^x = \sup_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{x}\|_\infty$. Without loss of generality we can assume $B_\infty^x = 1$. Using the bound for term Ⓑ and applying a union bound provides us the final result.

A.4 Polynomial Decay of Spectrum

As discussed earlier, one can replace the "exponential decay" assumption in Assumption 1 to a "softer" decay criterion with fatter tails, e.g., like a *polynomial* decay. We present extensions of earlier results to this polynomial decay condition. First we state the polynomial decay criterion precisely.

Assumption 2. *Let $\mathcal{P}_k$ denote the family of all polynomials of degree at most k, and let $P_{\star,k}$ denote the optimal polynomial in $\mathcal{P}_k$, i.e., $P_{\star,k} = \arg\min_{P\in\mathcal{P}_k} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathfrak{P}}[\ell(P(\boldsymbol{x}), y)]$. We assume that $\forall k$, $P_{\star,k}$ admits a decomposition as described in Equation 1 such that, for all $1 \le l \le k$, there exist constants $C_1 < 1$ and $C_2 = \mathcal{O}(1)$ such that $|\lambda_{lj}| \le C_1 \cdot j^{-\gamma-1}$ for each $j \ge 1$ and $l \in [1, k]$.*

Theorem 3. *Let ℓ be a 1-Lipschitz loss, $\delta \in (0, 1]$ and the generalization error for the optimal degree k polynomial $P_{\star,k}$ be given by $\mathcal{E}(P_{\star,k})$ and the training error for the ERM SPAM $\widehat{P}_{\boldsymbol{r},k}$, i.e., $\widehat{P}_{\boldsymbol{r},k} = \arg\min_P \sum_{i=1}^n \ell(P(\boldsymbol{x}_i, y_i)$ with ranks $\boldsymbol{r} = [1, r_2, ..., r_k]$ be given by $\widehat{\mathcal{E}}_n(\widehat{P}_{\boldsymbol{r},k})$. Then, for L_2-regularized models, where $\|\boldsymbol{u}_{ij}\|_2 \le B_{u,2}$ for all $i \in [k], j \in [r_i]$, and $\|\boldsymbol{\lambda}\|_2 \le B_{\lambda,2}$ where $\boldsymbol{\lambda} = \{\{\lambda_{ij}\}_{j=1}^{r_i}\}_{i=1}^k$, we have that with probability at least $1-\delta$ there exists an absolute constant C such that,*

$$\mathcal{E}(\widehat{P}_{\boldsymbol{r},k}) - \mathcal{E}(P_{\star,k}) \le 2B_{\lambda,2}\left(\sum_{l=1}^k (B_{u,2})^l \sqrt{r_l}\right)\sqrt{\frac{d}{n}} + C\cdot\left(\sum_{i=2}^k r_i^{-\gamma}\right) + 5\sqrt{\frac{\log(4/\delta)}{n}}.$$

For L_1-regularization, $\|\boldsymbol{u}_{ij}\|_1 \le B_{u,1}$ for all $i \in [k], j \in [r_i]$, and $\|\boldsymbol{\lambda}\|_1 \le B_{\lambda,1}$ where $\boldsymbol{\lambda} = \{\{\lambda_{ij}\}_{j=1}^{r_i}\}_{i=1}^k$, we have with probability at least $1-\delta$ there exists an absolute constant C such that,

$$\mathcal{E}(\widehat{P}_{\boldsymbol{r},k}) - \mathcal{E}(P_{\star,k}) \le 2B_{\lambda,1}\left(\sum_{l=1}^k (B_{u,1})^l\right)\sqrt{\frac{\log(d)}{n}} + C\cdot\left(\sum_{i=2}^k r_i^{-\gamma}\right) + 5\sqrt{\frac{\log(4/\delta)}{n}}.$$

Proof. The proof is identical to the proof of Theorem 1, except for the application of Lemma 1. Note that following Lemma 1, the sum of the singular values can be bound using Assumption 2 as,

$$\sum_{i=1}^k \sum_{j=r_i}^{\bar{r}_i} |\lambda_{ij}^\star| \le \sum_{i=1}^k \sum_{j=r_i}^{\bar{r}_i} C_1 \cdot j^{-\gamma-1} \le \sum_{i=1}^k \int_{j=r_i}^\infty C_1 \cdot j^{-\gamma-1} d\gamma = \frac{C_1}{\gamma+1}\sum_{i=1}^k r_i^{-\gamma}.$$

Replacing this result completes the proof. □

Discussion. Observe that in contrast to the result for exponential decay (where one required r to be polylogarithmic in d), we now require a larger rank r to achieve competitive performance. Specifically, one can observe from Theorem 3 that setting $r_i = r/k = d^{\gamma/m}$ for some $m > 1$ guarantees that the error term decays at the rate $\mathcal{O}\left(k^{\frac{1-\gamma}{m}} d^{-\frac{1}{m}}\right)$ which diminishes quickly for small m and large d. However, this degrades the rate of growth of the Rademacher complexity (first term) from a $\mathcal{O}(\sqrt{d})$ dependence to $\mathcal{O}(d^{\frac{1}{p}(1+\frac{\gamma}{m})})$ for L_p regularization, $p \in \{1, 2\}$.

A.5 Omitted Results

Lemma 1. *For the polynomials $\widetilde{P}_{\boldsymbol{r},k}$ and $P_{\star,k}$ as defined in Equations 4 and 6, the following holds with probability at least $1-\delta, \delta \in (0, 1]$, for some absolute constant $C \ll 1$,*

$$\left|\widehat{\mathcal{E}}_n(\widetilde{P}_{\boldsymbol{r},k}) - \mathcal{E}(P_{\star,k})\right| \le LC\cdot\left(\sum_{i=2}^k \exp(-r_i^\gamma)\right) + 2\sqrt{\frac{\log(2/\delta)}{n}}.$$

Proof. Observe,

$$\begin{aligned}\widehat{\mathcal{E}}_n(\widetilde{P}_{\boldsymbol{r},k}) - \mathcal{E}(P_{\star,k}) &= \widehat{\mathcal{E}}_n(\widetilde{P}_{\boldsymbol{r},k}) - \mathcal{E}(\widetilde{P}_{\boldsymbol{r},k}) + \mathcal{E}(\widetilde{P}_{\boldsymbol{r},k}) - \mathcal{E}(P_{\star,k}) \\ &\le \underbrace{\left|\widehat{\mathcal{E}}_n(\widetilde{P}_{\boldsymbol{r},k}) - \mathcal{E}(\widetilde{P}_{\boldsymbol{r},k})\right|}_{(B1)} + \underbrace{\left|\mathcal{E}(\widetilde{P}_{\boldsymbol{r},k}) - \mathcal{E}(P_{\star,k})\right|}_{(B2)}.\end{aligned}$$

To bound (B1), observe that for any point $(\boldsymbol{x}, y)$ within the training set, $\mathbb{E}[\ell(\widetilde{P}_{\boldsymbol{r},k}(\boldsymbol{x}), y)] = \mathcal{E}(\widetilde{P}_{\boldsymbol{r},k})$. Furthermore, $0 \le \ell(\cdot,\cdot) \le 1$. We can therefore apply the Azuma-Hoeffding inequality [Bercu et al., 2015] and obtain with probability at least $1-\delta, \delta \in (0, 1]$,

$$\left|\widehat{\mathcal{E}}_n(\widetilde{P}_{\boldsymbol{r},k}) - \mathcal{E}(\widetilde{P}_{\boldsymbol{r},k})\right| \le 2\sqrt{\frac{\log(2/\delta)}{n}}.$$

To bound (B2), observe that since ℓ is $L-$Lipschitz, for some $x_1, x_2, y \in \mathcal{Y}$,

$$\begin{aligned}|\ell(x_1, y) - \ell(x_2, y)| &\leq |L \cdot |x_1 - y| - L \cdot |x_2 - y||\\ &\leq L \cdot |x_1 - x_2|\,.\end{aligned}$$

Using this, we have that,

$$\begin{aligned}\left|\mathcal{E}(\widetilde{P}_{\boldsymbol{r},k}) - \mathcal{E}(P_{\star,k})\right| &\leq \left|\mathbb{E}_{(\boldsymbol{x},y)\sim\mathfrak{P}}\left[\ell(\widetilde{P}_{\boldsymbol{r},k}(\boldsymbol{x}), y) - \ell(P_{\star,k}(\boldsymbol{x}), y)\right]\right|\\ &\leq \mathbb{E}_{(\boldsymbol{x},y)\sim\mathfrak{P}}\left[\left|\ell(\widetilde{P}_{\boldsymbol{r},k}(\boldsymbol{x}), y) - \ell(P_{\star,k}(\boldsymbol{x}), y)\right|\right]\\ &\leq L \cdot \mathbb{E}_{(\boldsymbol{x},y)\sim\mathfrak{P}}\left[\left|\widetilde{P}_{\boldsymbol{r},k}(\boldsymbol{x}) - P_{\star,k}(\boldsymbol{x})\right|\right]\\ &\leq L \cdot \sup_{\boldsymbol{x}\in\mathcal{X}}\left|\widetilde{P}_{\boldsymbol{r},k}(\boldsymbol{x}) - P_{\star,k}(\boldsymbol{x})\right|.\end{aligned}$$

Observe now that for any $\boldsymbol{x} \in \mathcal{X}$,

$$\left|P_{\star,k}(\boldsymbol{x}) - \widetilde{P}_{\boldsymbol{r},k}(\boldsymbol{x})\right| = \left|\sum_{i=1}^{k}\sum_{j=r_i}^{\bar{r}_i}\lambda_{ij}^{\star}\cdot\langle\boldsymbol{u}_{ij}^{\star}, \tilde{\boldsymbol{x}}_i\rangle^i\right| \leq \sum_{i=1}^{k}\sum_{j=r_i}^{\bar{r}_i}\left|\lambda_{ij}^{\star}\cdot\langle\boldsymbol{u}_{ij}^{\star}, \tilde{\boldsymbol{x}}_i\rangle^i\right| \leq \sum_{i=1}^{k}\sum_{j=r_i}^{\bar{r}_i}|\lambda_{ij}^{\star}|.$$

Recall that Assumption 1 can control $\sum_{i=1}^{k}\sum_{j=r_i}^{\bar{r}_i}|\lambda_{ij}^{\star}|$. If the spectrum obeys the $D-$finite condition and $r_i = D \forall i \in [1, k]$ we have that $\sum_{i=1}^{k}\sum_{j=r_i}^{\bar{r}_i}|\lambda_{ij}^{\star}| = 0$. Next, if the spectrum obeys the $\gamma-$exponential decay, we have that $\lambda_{ij} = C_1\exp(-C_2 j^{\gamma})$. This gives us,

$$\sum_{i=1}^{k}\sum_{j=r_i}^{\bar{r}_i}|\lambda_{ij}^{\star}| \leq \sum_{i=1}^{k}\sum_{j=r_i}^{\bar{r}_i}C_1\exp(-C_2 j^{\gamma}) \leq \sum_{i=1}^{k}\int_{j=r_i}^{\infty}C_1\exp(-C_2 j^{\gamma}).$$

By Equation E.16 from Yang et al. [2020] we can bound the R.H.S. since $\gamma \geq 1$,

$$\int_{j=r_i}^{\infty}C_1\exp(-C_2 j^{\gamma}) \leq \frac{C_1}{C_2}\exp\left(-r_i^{\gamma}\right).$$

Replacing this result above completes the proof. □

Lemma 2 (Empirical Rademacher Complexity under L_2 regularization)**.** *Let $\mathcal{P}_{l,r}$ denote the set of polynomials that have rank exactly r and degree exactly l such that they admit a rank-decomposed representation of $\sum_{j=1}^{r}\lambda_j\cdot\boldsymbol{u}_j\otimes\ldots\otimes\boldsymbol{u}_j$, where $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_r]$ s.t. $\|\boldsymbol{\lambda}\|_2 \leq B_{\lambda,2}$ and $\|\boldsymbol{u}_j\|_2 \leq B_{u,2}$ for all $j \in 1, \ldots, l$. Let the rescaled data distribution as defined in Equation 3 for degree l be given by $\widetilde{\mathcal{X}}_l = \{sign(\boldsymbol{x})\cdot|\boldsymbol{x}|^{1/l} | \boldsymbol{x} \in \mathcal{X}\}$ and we sample n points i.i.d. from $\widetilde{\mathcal{X}}_l$. Then the empirical Rademacher complexity $\mathfrak{R}_n$ of $\mathcal{P}_{l,r}$ obeys,*

$$\mathfrak{R}_n(\mathcal{P}_{l,r}) \leq B_{\lambda,2}B_2^x(B_{u,2})^l\sqrt{\frac{r}{n}}.$$

Proof.

$$\begin{aligned}\mathfrak{R}_n(\mathcal{P}_{l,r}) &= \frac{1}{n}\mathbb{E}\left[\sup_{\boldsymbol{\lambda}\in\boldsymbol{\Lambda},\mathcal{U}}\sum_{i=1}^{n}\epsilon_i\sum_{j=1}^{r}\lambda_r\cdot\langle\boldsymbol{u}_j, \tilde{\boldsymbol{x}}_{li}\rangle^l\right]\\ &\leq \frac{1}{n}\sup_{\|\boldsymbol{\lambda}\|_2\leq B_{\lambda,2}}\sum_{j=1}^{r}\lambda_r\cdot\mathbb{E}\left[\sup_{\|\boldsymbol{u}_j\|_2\leq B_{u,2}}\sum_{i=1}^{n}\epsilon_i\cdot\langle\boldsymbol{u}_j, \tilde{\boldsymbol{x}}_{li}\rangle^l\right]\\ &\leq \frac{1}{n}\sup_{\|\boldsymbol{\lambda}\|_2\leq B_{\lambda,2}}\|\boldsymbol{\lambda}\|_2\cdot\sqrt{\sum_{j=1}^{r}\mathbb{E}\left[\sup_{\|\boldsymbol{u}_j\|_2\leq B_{u,2}}\sum_{i=1}^{n}\epsilon_i\cdot\langle\boldsymbol{u}_j, \tilde{\boldsymbol{x}}_{li}\rangle^l\right]^2}\\ &\leq \frac{B_{\lambda,2}}{n}\sqrt{\sum_{j=1}^{r}\mathbb{E}\left[\sup_{\|\boldsymbol{u}_j\|_2\leq B_{u,2}}\sum_{i=1}^{n}\epsilon_n^2\cdot\langle\boldsymbol{u}_j, \tilde{\boldsymbol{x}}_{li}\rangle^{2l}\right]}\\ &\leq \frac{B_{\lambda,2}}{n}\sqrt{\sum_{j=1}^{r}\mathbb{E}\left[\sup_{\|\boldsymbol{u}_j\|_2\leq B_{u,2}}\sum_{i=1}^{n}\|\boldsymbol{u}_j\|_2^{2l}\|\tilde{\boldsymbol{x}}_{li}\|^{2l}\right]}\\ &\leq B_{\lambda,2}(B_{u,2})^l B_2^x\sqrt{\frac{r}{n}}.\end{aligned}$$

□

Lemma 3 (Empirical Rademacher Complexity under L_1 regularization)**.** *Let $\mathcal{P}_{l,r}$ denote the set of polynomials that have rank exactly r and degree exactly l such that they admit a rank-decomposed representation of $\sum_{j=1}^{r} \lambda_j \cdot \boldsymbol{u}_j \otimes ... \otimes \boldsymbol{u}_j$, where $\boldsymbol{\lambda} = [\lambda_1, ..., \lambda_r]$ s.t. $\|\boldsymbol{\lambda}\|_1 \leq B_{\lambda,1}$ and $\|\boldsymbol{u}_j\|_1 \leq B_{u,1}$ for all $j \in 1, ..., l$. Let the rescaled data distribution as defined in Equation 3 for degree l be given by $\widetilde{\mathcal{X}}_l = \{sign(\boldsymbol{x}) \cdot |\boldsymbol{x}|^{1/l} | \boldsymbol{x} \in \mathcal{X}\}$ and we sample n points i.i.d. from $\widetilde{\mathcal{X}}_l$. Then the empirical Rademacher complexity $\mathfrak{R}_n$ of $\mathcal{P}_{l,r}$ obeys,*

$$\mathfrak{R}_n(\mathcal{P}_{l,r}) \leq (B_{\lambda,1} \cdot r)(B_{u,1})^l B_1^x \sqrt{\frac{\log(d)}{n}}.$$

Proof.

$$\begin{aligned}
\mathfrak{R}_n(\mathcal{P}_{l,r}) &= \frac{1}{n} \mathbb{E}\left[\sup_{\boldsymbol{\lambda},\boldsymbol{u}} \sum_{i=1}^{n} \epsilon_i \sum_{j=1}^{r} \lambda_r \cdot \langle \boldsymbol{u}_j, \tilde{\boldsymbol{x}}_{li} \rangle^l\right] \\
&\leq \frac{1}{n} \sup_{\|\boldsymbol{\lambda}\|_1 \leq B_{\lambda,1}} \sum_{j=1}^{r} \lambda_r \cdot \mathbb{E}\left[\sup_{\|\boldsymbol{u}_j\|_1 \leq B_{u,1}} \sum_{i=1}^{n} \epsilon_i \cdot \langle \boldsymbol{u}_j, \tilde{\boldsymbol{x}}_{li} \rangle^l\right] \\
&\leq \frac{1}{n} \sup_{\|\boldsymbol{\lambda}\|_1 \leq B_{\lambda,1}} \|\boldsymbol{\lambda}\|_1 \cdot \max_{j \in [r]} \left| \mathbb{E}\left[\sup_{\|\boldsymbol{u}_j\|_1 \leq B_{u,1}} \sum_{i=1}^{n} \epsilon_i \cdot \langle \boldsymbol{u}_j, \tilde{\boldsymbol{x}}_{li} \rangle^l\right]\right| \\
&\leq \frac{B_{\lambda,1}}{n} \max_{j \in [r]} \left[\mathbb{E}\left[\left|\sup_{\|\boldsymbol{u}_j\|_1 \leq B_{u,1}} \sum_{i=1}^{n} \epsilon_i \cdot \langle \boldsymbol{u}_j, \tilde{\boldsymbol{x}}_{li} \rangle^l\right|\right]\right] \\
&\leq \frac{B_{\lambda,1}}{n} \mathbb{E}\left[\max_{j \in [r]} \left|\sup_{\|\boldsymbol{u}_j\|_1 \leq B_{u,1}} \sum_{i=1}^{n} \epsilon_i \cdot \sum_{j=1}^{r} \langle \boldsymbol{u}_j, \tilde{\boldsymbol{x}}_{li} \rangle^l\right|\right] \\
&= \frac{B_{\lambda,1}}{n} \mathbb{E}\left[\left|\sup_{\|\boldsymbol{u}_j\|_1 \leq B_{u,1}} \sum_{i=1}^{n} \max_{j \in [r]} \|\boldsymbol{u}_j\|_1^l \|\epsilon_i \tilde{\boldsymbol{x}}_{li}\|_\infty^l\right|\right] \\
&\leq \frac{(B_{u,1})^l \cdot B_{\lambda,1}}{n} \mathbb{E}\left[\sum_{i=1}^{n} \|\epsilon_i \boldsymbol{x}_i\|_\infty\right] \\
&\leq B_{\lambda,1} (B_{u,1})^l B_\infty^x \cdot \sqrt{\frac{\log(d)}{n}}.
\end{aligned}$$

The last line follows from Massart's Finite Lemma (Massart [2000], Lemma 5.2). □

Theorem 4 (Structural Results for Rademacher Complexity, Theorem 12 of Bartlett and Mendelson [2002])**.** *Let $\mathcal{F}, \mathcal{F}_1, ..., \mathcal{F}_k$ and $\mathcal{H}$ be classes of functions. Then,*

1. *If $\mathcal{F} \subseteq \mathcal{H}$, $\mathfrak{R}_n(\mathcal{F}) \leq \mathfrak{R}_n(\mathcal{H})$.*
2. *$\mathfrak{R}_n(\sum_{i=1}^{k} \mathcal{F}_i) \leq \sum_{i=1}^{k} \mathfrak{R}_n(\mathcal{F}_i)$.*
3. *For any L-Lipschitz function h, $\mathfrak{R}_n(\mathcal{F} \odot h) \leq 2L \cdot \mathfrak{R}_n(\mathcal{F})$.*

B Related Work

Post-hoc explainability. In contrast to designing fully-interpretable models such as GAMs (Generalized Additive Models), a dominant line of work in interpretable machine learning is of *post-hoc* explainability. *Post-hoc* methods refers to delivering explanations of model predictions after the prediction has been made. The most popular line of work in this domain is that of *instance-based* explanations, i.e., explaining each prediction via a local input-specific explanation. For example, the work of Ribeiro et al. [2016] introduces LIME (Local Interpretable Model Explanations) that fit a weighted linear regression model, for any input $\boldsymbol{x}$, using random samples generated from $\mathcal{B}_2(\boldsymbol{x})$, where the weights are computed via the distance of the random point from the true sample. SHAP [Lundberg and Lee, 2017] computes a similar linear explanation based on Shapley values using influence functions derived from cooperative game theory. It has been shown to unify several prior approaches, e.g., DeepLIFT [Shrikumar et al., 2017, 2016], LIME [Ribeiro et al., 2016], and Layerwise Relevance Propagation (LRP) Bach et al. [2015]. Several follow-up works have addressed shortcomings in this line of work, please see Madsen et al. [2021], Chakraborty et al. [2017], Du et al. [2019], Carvalho et al. [2019] for excellent surveys on recent developments in this area. More importantly, however, these approaches are not effective for high-stakes decision-making. They are notoriously unstable [Ghorbani et al., 2019a, Lakkaraju and Bastani, 2020], expensive to compute [Slack et al., 2021], unjustified [Laugel et al., 2019] and in many cases, inaccurate [Lipton, 2018]. Rudin [2019] outlines several of these shortcomings in incredible detail, and hence makes the case to replace *post-hoc* interpretable approaches with methods that are inherently explainable.

Transparent and Interpretable Machine Learning. Early work has focused on greedy or ensemble approaches to modeling interactions [Friedman, 2001, Friedman and Popescu, 2008] that enumerate pairwise interactions and learn additive interaction effects. Such approaches often pick up spurious interactions when data is sparse [Lou et al., 2013] and are impossible to scale to modern-sized datasets due to enumeration of individual combinations. As an improvement, Lou et al. [2013] proposed GA^2M that uses a statistical test to filter out only "true" interactions. However GA^2M fails to scale to large datasets as it requires constant re-training of the model and ad-hoc operations such as discretization of features which may require tuning for datasets with a large dimensionality. Other generalized additive models require expensive training of numerous decision trees, kernel machines or splines [Hastie and Tibshirani, 2017], which make them unattractive compared to black-box models.

An alternate approach is is to learn interpretable neural network transformations. Neural Additive Models (NAMs, Agarwal et al. [2021]) learn a DNN per feature. TabNet [Arık and Pfister, 2021] and NIT [Tsang et al., 2018] alternatively modify NN architectures to increase their interpretability. NODE-GAM [Chang et al., 2021] improves NAMs with oblivious decision trees for better performance while maintaining interpretability. Our approach is notably distinct from these prior works: we do not require iterative re-training; we can learn *all* pairwise interactions regardless of dimensionality; we can train SPAM via backpropagation; and we scale effortlessly to very large-scale datasets.

Learning Polynomials. The idea of decomposing polynomials was of interest prior to the deep learning era. Specifically, the work of Ivakhnenko [1971], Oh et al. [2003], Shin and Ghosh [1991] study learning neural networks with polynomial interactions, also known as *ridge polynomial networks* (RPNs). However, RPNs are typically not interpretable: they learn interactions of a very high order, and include non-linear transformation. Similar rank decompositions have been studied in the context of matrix completion [Recht, 2011], and are also a subject of interest in tensor decompositions [Nie, 2017a, Brachat et al., 2010], where, contrary to our work, the objective is to decompose existing tensors rather than directly learn decompositions from gradient descent. Recently, Chrysos et al. [2019, 2020] use tensor decompositions to learn higher-order polynomial relationships in intermediate layers of generative models. However, their work uses a recursive formulation and learns high-degree polynomials directly from uninterpretable input data (e.g., images), and hence is non-interpretable.

C Experimental Details

All our benchmarks are implemented in PyTorch and run on a cluster, where each machine was equipped with $8\times$V100 NVIDIA GPU machines with 32GB VRAM. We set the batch size of at most 1024 per GPU, and in case a model is too large to fit a batch size of 1024, we perform a binary search to find the largest batch size that it will accommodate. All datasets have features renormalized between the range $[0, 1]$ for uniformity and simplicity. For all experiments we use the Adam with decoupled weight decay (AdamW) optimizer [Loshchilov and Hutter, 2017].

C.1 Tabular Dataset Details

We use the following tabular datasets.

1. **California Housing** (CH, Pace and Barry [1997]): This dataset was derived from the 1990 U.S. census, and is a regression task, fitting the median house value for California districts using demographic information. The dataset contains 20,640 instances and 8 numeric features.
2. **FICO HELOC** (FICO, FICO [2018]): This is a binary task, part of the FICO Explainability challenge. The target variable denotes the "risk" of home equity line of credit (HELOC) applicants and features correspond to credit report information. It has 10,459 instances and 23 features.
3. **Cover Type** (CovType, Blackard and Dean [1999]): This multi-class dataset is part of the UCI ML repository [Asuncion and Newman, 2007]. The task is classification of forest cover type from cartographic variables. It contains 581,012 instances and 54 attributes over 6 classes.
4. **20 Newsgroups** (Newsgroups, Lang [1995]): This is a popular benchmark dataset where the problem is multi-class classification of email text into 20 subject categories. Following standard convention, we extract TF-IDF features on the training set, which gives us 18,828 instances and 146,016 features over 20 classes (see Appendix Section C.2 for details).

C.2 20 Newsgroups Feature Extraction

We use the following feature extraction pipeline from the `sklearn` repository. We first use the `TfIdfVectorizer` to compute the TF-IDF features on only the training set. Next, we transform the testing and validation sets using the learnt vectorizer, and rescale all the features within the range $[0, 1]$. This gives us 18,828 instances and 146,016 features over 20 classes.

C.3 Concept Bottleneck Implementation

In an effort to scale interpretable approaches to problems beyond tabular datasets, we consider benchmark problems using the "Sequential Concept Bottleneck" framework of Koh et al. [2020]. The general idea of concept bottleneck (CB) models are to enable feature extraction necessary for problems in domains such as computer vision by learning a black-box model that can first predict human interpretable "concepts", and then learn a simple, typically fully-transparent model on these concepts to predict the final category. For example, in the problem of bird species classification from images [Wah et al., 2011], one can consider visible parts and attributes of the birds as intermediary "concepts" on which one can learn a linear classifier in order to provide reasonable classification performance while at the same time providing interpretable decisions. Similar approaches have also been explored in the work of, e.g., Chen et al. [2019], Ghorbani et al. [2019b] and Zhang et al. [2020]. We consider concept bottleneck models for computer vision tasks.

C.3.1 CUB-200 Concept Bottleneck

The Caltech-UCSD Birds (CUB)-200 [Wah et al., 2011] dataset is an image classification dataset that has images belonging to 200 different species of birds, a problem that is an instance of the broader problem of *fine-grained visual classification*. The dataset is equipped with keypoint annotations of 15 bird parts, e.g., crown, wings, etc., and each part has ≥ 1 part-attribute labels, e.g., *brown leg*, *buff neck*, etc. For the concept bottleneck model, we train a convolutional neural network architecture that predicts these intermediary concepts from the images, and then used this trained CNN model (with weights fixed) to extract features that are then trained using one of the several benchmark approaches (including SPAM). One key difference between prior implementations and ours in this concept bottleneck setting is that we ignore laterality within the annotations, e.g., "left wing" and "right wing" are treated simply as "wing".

CNN architecture. We train a ResNet-50 [He et al., 2016] on images resized to the size 448×448 resolution until the last pooling layer of the network (`pool5`). The features are 2048-dimensional but ignoring the spatial pooling, are evaluated on each 14×14 patch within the image. We train part-attribute linear classifiers on these spatially-arranged extracted features, where the part locations are used from the training annotations. Once the network has been trained, we run max-pooling over the 14×14 grid to extract features that are used in the second stage training. Note that this setup is different from the original Concept Bottleneck [Koh et al., 2020] architecture, as the original setup predicted the keypoints from the entire image, instead of a spatially-supervised version. We use this architecture as the original setup overfit very easily.

C.3.2 Common Objects Dataset Details

We construct a dataset by collecting public images from Instagram[1], involving common household objects (e.g., bed, stove, saucer, tables, etc.) with their bounding box annotations. For each bounding box, we collect 2,618 interpretable annotations, namely, parts (e.g., leg, handle, top, etc.), attributes (e.g., colors, textures, shapes, etc.), and part-attribute compositions. The dataset has 2,645,488 training and 58,525 validation samples across 115 classes with 2,618 interpretable concepts.

C.3.3 Hyperparameters

Gradient-descent based methods are trained with the Adam with decoupled weight decay (AdamW) optimizer Loshchilov and Hutter [2017]. All methods are trained for 1000 epochs on the California Housing and FICO HELOC datasets, 100 epochs on iNaturalist Birds and Common Objects datasets, and 500 epochs on the CoverType, Newsgroups and CUB datasets. We use a cosine annealing learning rate across all gradient-descent based approaches and do a random hyperparameter search within the following specified grids.

Linear (Order 1 and Order 2). The learning rate range is $[1\text{e}{-5}, 100]$, and the weight decay interval is $[1\text{e}{-13}, 1.0)$. We use the same interval for L_1-regularized models as well.

Deep Neural Networks. We experiment with 3 neural network architectures that are all comprised of successive fully-connected layers. The first has 5 hidden layers of the shapes $[128, 128, 64, 64, 64]$ units; the second is three hidden layers deep with $[1024, 512, 512]$ units, and finally we have a shallow but wide architecture having 2 hidden layers with $[2048, 1024]$ units. We report the best accuracies achieved across all three architectures. Moreover, we observed that increasing the depth with more layers did not improve accuracy. We search the initial learning rate in the interval $[10^{-6}, 10.0)$, weight decay in $[1\text{e}{-9}, 1.0)$, dropout in the set $\{0, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$.

Neural Additive Models. We proceed with the standard proposed architectures in the original implementation of Agarwal et al. [2021]. We experiment with two architectures: the first is an MLP with 3 hidden layers and $[64, 64, 32]$ units, and the second is an MLP with 1 hidden layer with $1,024$ units and an ExU activation. We tune the learning rate in the interval $[1\text{e}{-5}, 1.0)$, weight decay in $[1\text{e}{-9}, 1.0)$, output penalty weight in $[1\text{e}{-6}, 100)$, dropout and feature dropout in the set $\{0, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$.

EBMs. We describe the hyperparameter followed by the range of values we search from.

- Maximum bins: $\{8, 16, 32, 64, 128, 256, 512\}$
- Number of interactions: $\{0, 2, 4, 8, 16, 32, 64, 128, 256, 512\}$ (0 for EBMs, ≥ 0 for EB^{2}Ms)
- learning rate: $[1\text{e}{-6}, 100)$
- maximum rounds: $\{1000, 2000, 4000, 8000, 16000\}$
- minimum samples in a leaf node: $\{1, 2, 4, 8, 10, 15, 20, 25, 50\}$
- maximum leaves: $\{1, 2, 4, 8, 10, 15, 20, 25, 50\}$
- binning: {"quantile", "uniform", "quantile_humanized"}
- inner/outer bags: $\{1, 2, 4, 8, 16, 32, 64, 128\}$.

[1] www.instagram.com

XGBoost. We describe the hyperparameter followed by the range of values we search from.

- number of estimators: $\{1, 2, 4, 8, 10, 20, 50, 100, 200, 250, 500, 1000\}$
- max-depth: $\{\infty, 2, 5, 10, 20, 25, 50, 100, 2000\}$
- η: $[0.0, 1.0)$
- `subsample`: $[0.0, 1.0)$
- `colsample_bytree`: $[0.0, 1.0)$

CART. We describe the hyperparameter followed by the range of values we search from.

- criterion: {"absolute error", "friedman mse", "squared error", "poisson"}
- splitter: {"best", "random"}
- minimum samples leaf: $\{4, 8, 16, 32, 64, 128, 256\}$
- minimum samples split: $\{4, 8, 16, 32, 64, 128, 256\}$

SPAM-Linear . For SPAM-Linear, there are 4 sets of hyperparameters: initial learning rate (tuned logarithmically from the set $[10^{-7}, 10^{2}]$), weight decay (tuned logarithmically from the set $[10^{-13}, 10^{1}]$), dropout for the singular values λ, which is tuned from the range $[0, 1]$, and the vector of ranks $\boldsymbol{r}$ which consequently specifies the degree of the polynomial. For degree 2 SPAM, we search ranks from the set $\mathcal{S} = \{[25], [50], [100], [200], [250], [400], [500], [750], [800], [1000], [1200], [1400], [1600]\}$ and we for order 3, we search from the set $\mathcal{S} \times \mathcal{S}$ (i.e., product set of original range).

SPAM-Neural . For SPAM-Neural, in addition to the 4 original hyperparameters, we search from the NAM space identical to the range employed for NAM (please see above). Regarding the original hyperparameters, we search: initial learning rate (tuned logarithmically from the set $[10^{-7}, 10^{2}]$), weight decay (tuned logarithmically from the set $[10^{-13}, 10^{1}]$), dropout for the singular values λ, which is tuned from the range $[0, 1]$, and the vector of ranks $\boldsymbol{r}$ which consequently specifies the degree of the polynomial. For degree 2 SPAM, we search ranks from the set $\mathcal{S} = \{[25], [50], [100], [200], [250], [400], [500], [750], [800], [1000], [1200], [1400], [1600]\}$ and we for order 3, we search from the set $\mathcal{S} \times \mathcal{S}$ (i.e., product set of original range).

D Human Subject Evaluation Details

D.1 CUB-200 Dataset Details

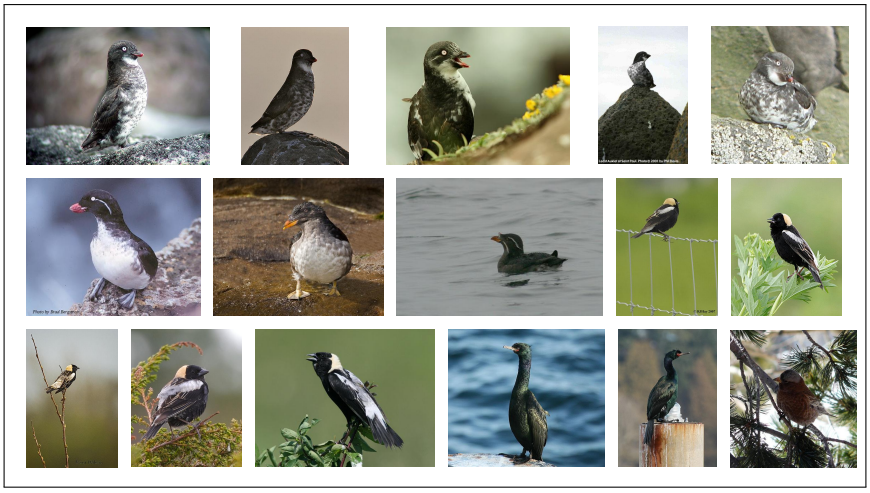

Figure 3: Random sample Images from the CUB-200 dataset used in Human Subject Evaluations

Figure 3 denotes a random sample of images that were shown to the participants. We randomly sampled images from a pre-selected set of a pair of classes. There were a total of 5 pairs, corresponding to the following classes:

1. “122.Harris_Sparrow", “060.Glaucous_winged_Gull"
2. “057.Rose_breasted_Grosbeak", “034.Gray_crowned_Rosy_Finch"
3. “066.Western_Gull", “035.Purple_Finch"
4. “064.Ring_billed_Gull", “008.Rhinoceros_Auklet"
5. “095.Baltimore_Oriole", “086.Pacific_Loon"

All images were sampled from the validation set.

D.2 Experiment Interface

The experiment was implemented using the `Mephisto` [Miller et al., 2017] library in `react.js`. Please see the subsequent pages for screenshots of the exact interface.

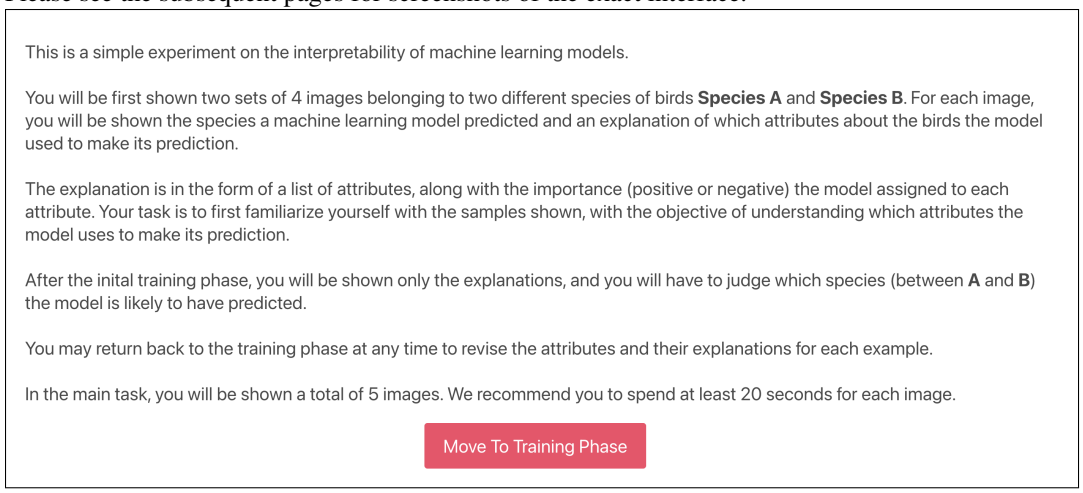

Figure 4: Human Subject Experiment: Introduction page

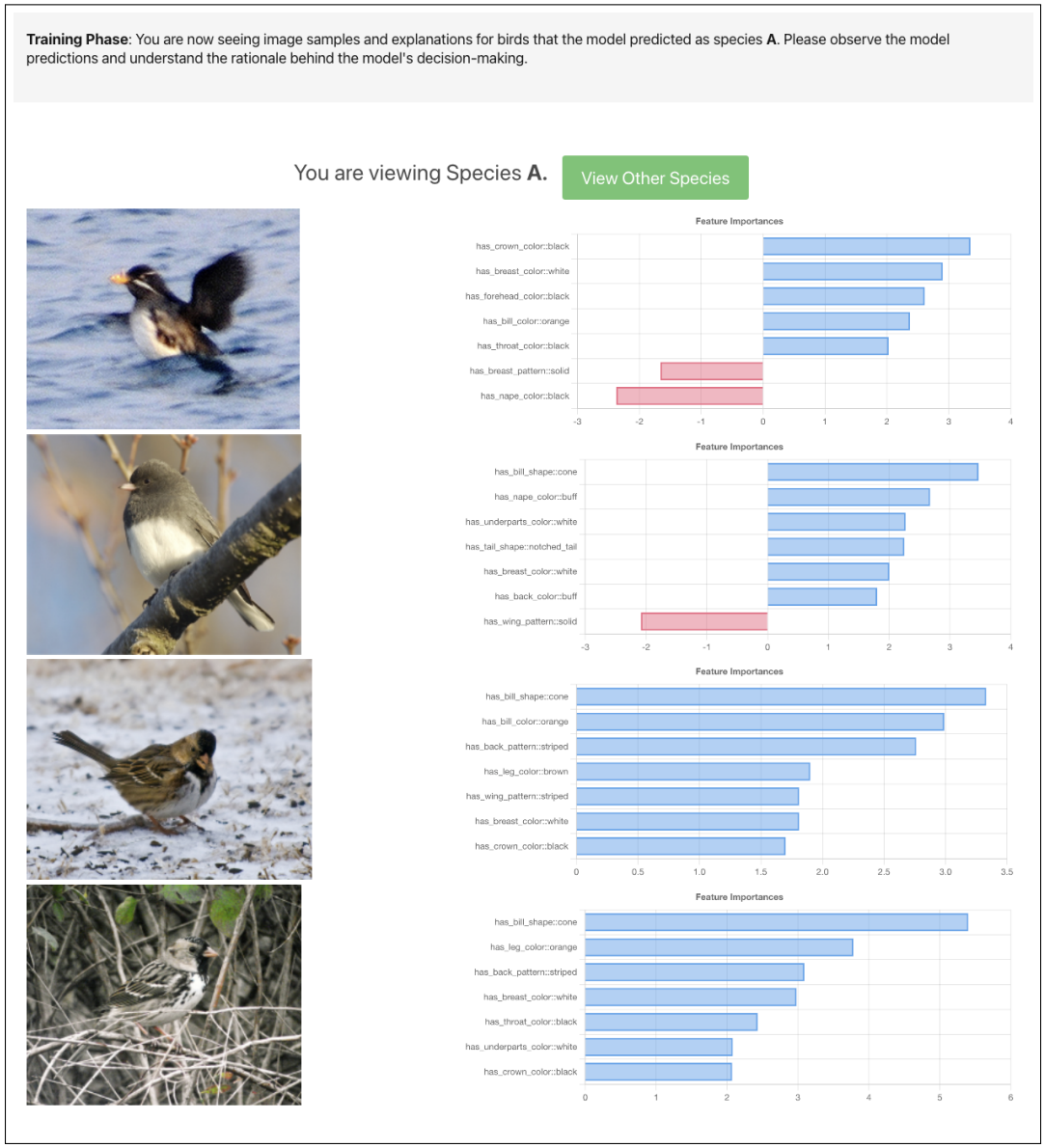

Figure 5: Human Subject Experiment: Example training phase, class **A** for linear explanations.

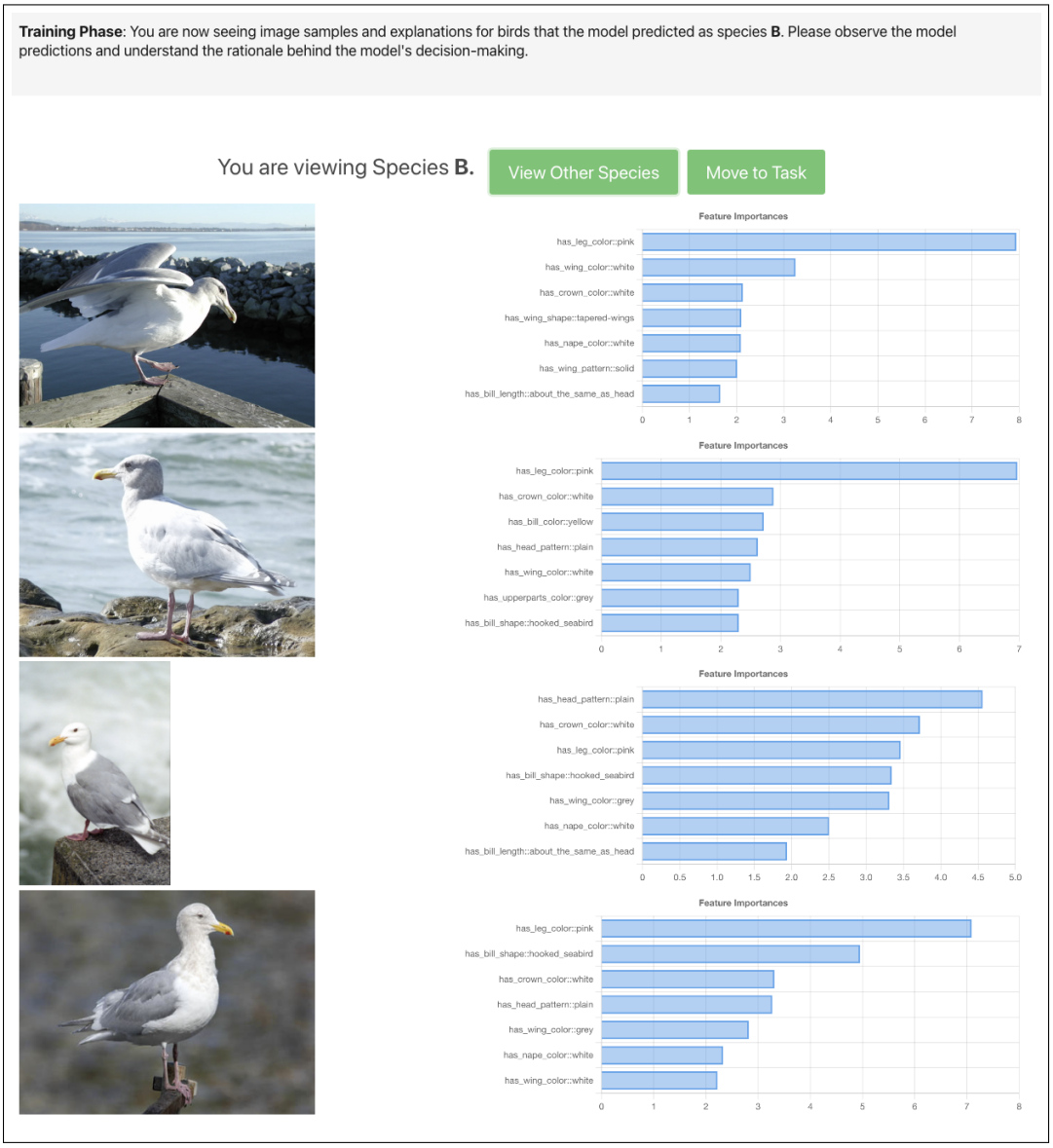

Figure 6: Human Subject Experiment: Example training phase, class **B** for linear explanations.

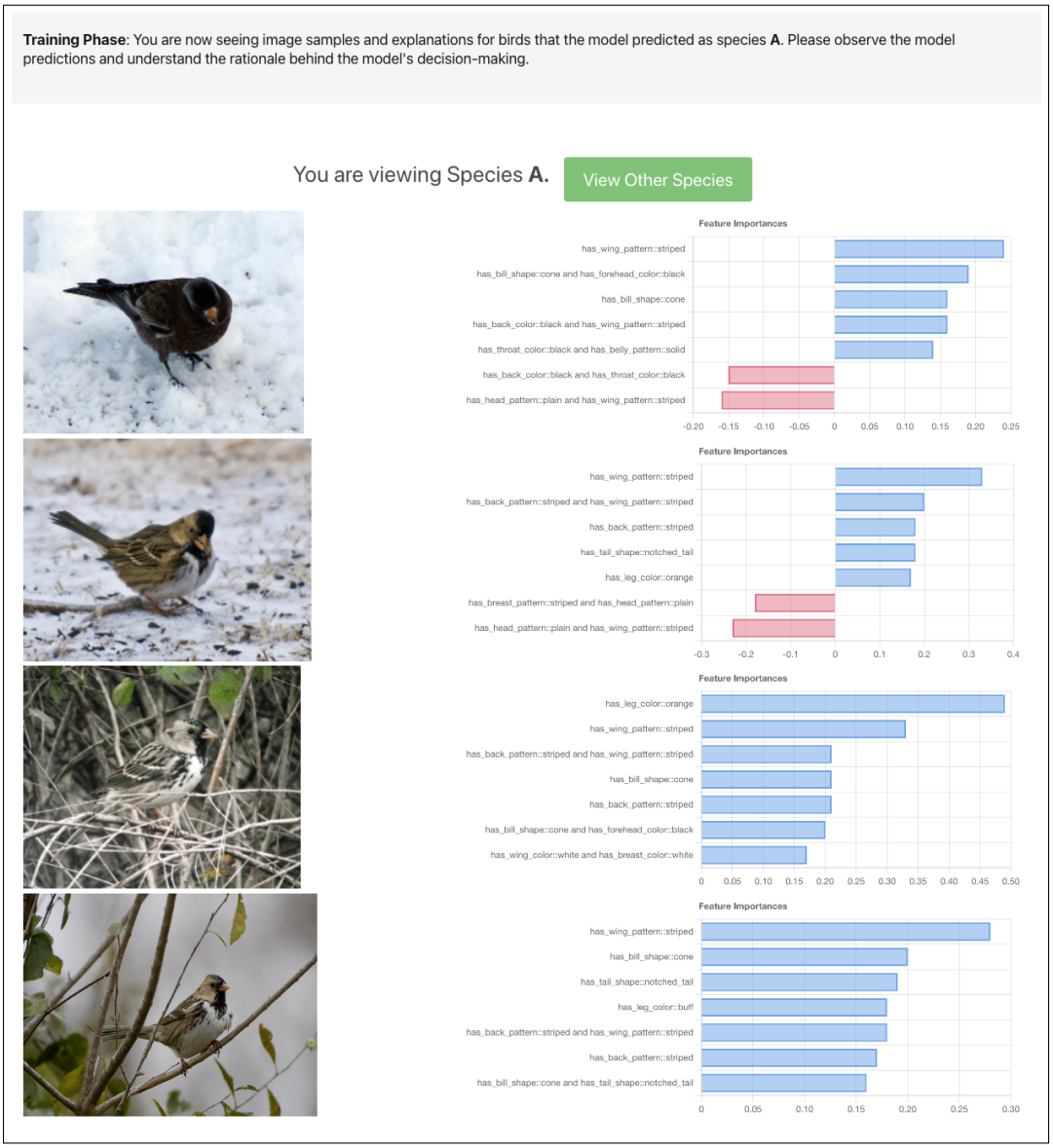

Figure 7: Human Subject Experiment: Example training phase, class **A** for SPAM (order 2) explanations.

Figure 8: Human Subject Experiment: Example testing phase, for linear explanations.

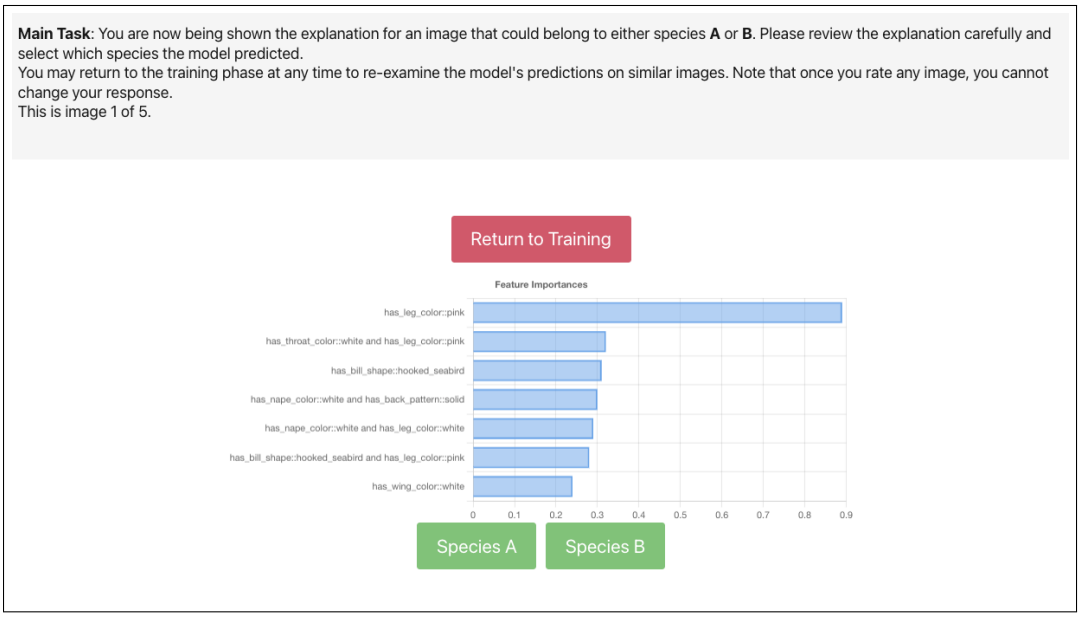

Figure 9: Human Subject Experiment: Example testing phase, for SPAM explanations.