Fidelity Isn't Accuracy:
When Linearly Decodable Functions Fail to Match the Ground Truth

Jackson Eshbaugh
Lafayette College
eshbaugj@lafayette.edu

May 16, 2025

Abstract

Neural networks are powerful function approximators, but their internal complexity often obscures what function they have actually learned. In this work, we propose a simple and interpretable diagnostic for regression networks: the linearity score $\lambda(f)$, which quantifies how well a trained neural network can be mimicked by a linear model. By training a linear surrogate to approximate the network's output, we assess the degree to which the learned function is linearly decodable. We evaluate this framework across synthetic and real-world datasets and find that while $\lambda(f)$ reliably reflects how linearly structured a network's function is, high scores do not always correspond to predictive equivalence. This reveals both the interpretive utility and the inherent limitations of using linear surrogates to capture nonlinear model behavior—especially in settings where accuracy is paramount.

1 Introduction

Neural networks have revolutionized supervised learning, but their internal complexity often renders their learned functions opaque. In other words, they are black boxes. This lack of interpretability hinders trust, accountability, and transparency—especially in high-stakes domains such as healthcare, finance, and criminal justice, where understanding a model's decisions can be as important as their accuracy. While many techniques exist for interpreting classification networks—such as saliency maps, feature attribution methods like LIME [10] and SHAP [6], and probing approaches [1]—the interpretability of regression networks remains comparatively underexplored [6, 14].

One common strategy in model interpretability is to approximate a complex system with a simpler, interpretable surrogate, such as a linear model or decision tree [3, 12]. In classification settings, surrogate models are often applied to internal representations to evaluate the linear accessibility of information at each layer [1, 5]. However, in regression tasks, there has been limited work focused on assessing the linearity of a network's full input–output function.

In this paper, we introduce a simple yet powerful idea: we measure how linearly decodable a trained regression network's output function is. We define a metric, $\lambda(f)$, as the coefficient of determination R^2 between a neural network's predictions and those of a linear model trained to mimic it. This metric offers a scalar summary of how faithfully a linear surrogate can approximate the network's behavior.

To evaluate the utility and limitations of this approach, we perform experiments on synthetic and real-world regression datasets. For each, we compare a baseline linear model, a trained neural network, and a linear surrogate trained to mimic the network. Our findings show that $\lambda(f)$ reliably quantifies how linearly decodable a network's function is—but also reveal that high fidelity to the network does not always translate into predictive success on the original task.

2 Related Works

A common approach to interpreting neural networks involves training simpler surrogate models—such as linear regressors or decision trees—to approximate their behavior. Notably, Craven and Shavlik [3] introduced TREPAN, an algorithm that uses decision trees to extract disjunctive normal form (DNF) rules by approximating the overall function computed by a neural network. Sato and Tsukimoto [12] proposed CRED, which also employs decision trees but adopts a structural approach: it trains one decision tree to approximate the output layer and additional trees for each neuron in the single hidden layer. Unlike TREPAN, which operates purely on input–output behavior, CRED utilizes intermediate activations to produce layer-wise symbolic rules. In contrast to both of these algorithms, our

method uses a linear surrogate not to extract rules, but to quantify the degree to which a neural network's output function is linearly decodable.

Another strategy for interpreting neural networks is probing. In this approach, smaller linear models—called "probes"—are trained to predict task-relevant labels from internal representations, providing insight into a model's training progress or the structure of its learned features. For example, Alain and Bengio [1] introduced linear classifier probes that are inserted into each layer of a trained (or partially trained) neural network to assess how much task-relevant information is linearly accessible at that layer. This method allows researchers to analyze the flow of information during training and characterize the representational capacity of different layers. Hewitt and Manning [5] propose a structural probe to test whether syntactic information is linearly encoded in contextual word embeddings. They learn a linear transformation such that squared distances between transformed word vectors approximate syntactic distances in a dependency parse tree. This approach evaluates whether hierarchical structure is recoverable through linear geometry. While probing methods aim to characterize internal representations, our method treats the network as a black box and applies linear approximation to its overall function.

Most probing methods train predictive models on frozen representations, but Saphra and Lopez [11] take a different approach. They use Singular Vector Canonical Correlation Analysis (SVCCA) [9] to track how linguistic structure emerges over the course of language model training, comparing internal representations to those learned by supervised taggers. Though not a probe in the strict sense, their work shares the goal of understanding when and how interpretable features arise in neural representations. Our work shares their interest in emergent structure but differs in focus: we offer a concrete, output-level metric for assessing how linearly structured the network's learned function is.

Other work has examined the complexity of functions learned by neural networks using measures such as sensitivity, Lipschitz continuity, and generalization bounds [4, 7], offering theoretical insight into expressiveness and stability. Our approach is complementary: rather than analyzing complexity directly or probing internal representations, we treat the network as a black box and ask how much of its behavior is recoverable through linear approximation. By focusing on the output function itself—rather than intermediate layers—we provide a simple, scalable diagnostic that captures one interpretable dimension of the network's learned behavior.

3 Methodology

To assess whether a trained neural network's learned function is linearly structured, we propose a projection-based framework that measures how closely its output can be approximated by a linear model. Our goal is not to probe internal layers or compute gradient attributions, but to treat the network as a black-box function $f\colon \mathbb{R}^n \to \mathbb{R}$, and ask: *to what extent is the behavior of f linearly decodable from its input?*

We formalize this question by projecting f onto the space of linear functions and measuring the quality of that approximation.

Let $\mathcal{D} \subset \mathbb{R}^n$ be a domain over which the network operates, and assume $f \in L^2(\mathcal{D})$, the space of square-integrable functions. Let $\mathcal{L} \subset L^2(\mathcal{D})$ be the subspace of affine functions. We define the optimal linear surrogate as:

$$g^* = \arg\min_{g \in \mathcal{L}} \mathbb{E}_{x \sim \mathcal{D}} \left[(f(x) - g(x))^2 \right]$$

This is simply the orthogonal projection of f onto $\mathcal{L}$, and represents the best linear approximation to the network's function in mean squared error.

We then define the **linearity score** $\lambda(f)$ as the proportion of variance in the network's output that is captured by this surrogate:

$$\lambda(f) := R^2(f, g^*) = 1 - \frac{\mathbb{E}[(f(x) - g(x))^2]}{\mathrm{Var}(f(x))}$$

This score ranges from 0 to 1. When $\lambda(f) \approx 1$, the network's output function lies close to a linear subspace; when $\lambda(f) \approx 0$, it deviates substantially.

A common misconception is that $\lambda(f)$ measures the linearity of the original data (x, y). In fact, it measures the linearity of the network's *learned function* f with respect to its input x. That is, even if the underlying data is highly nonlinear, a network may learn a function that is approximately linear—making its outputs linearly decodable. Conversely, a network may learn a highly nonlinear function even on linearly structured data.

While the surrogate model g provides a closed-form linear approximation of f, it is not guaranteed to preserve the predictive accuracy of the original network. As our experiments show, even high-fidelity approximations (high

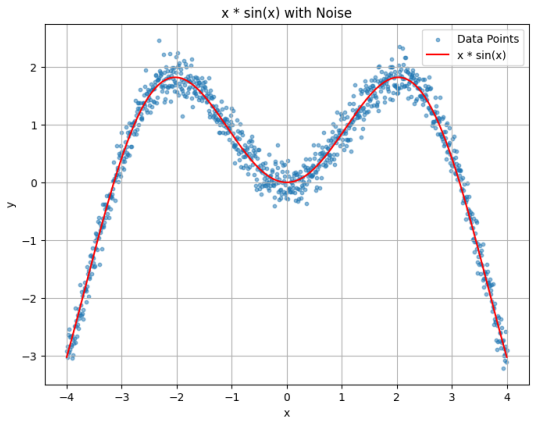

Figure 1: Synthetic dataset $y = x \cdot \sin(x) + \epsilon$ with Gaussian noise $\epsilon \sim \mathcal{N}(0, 0.2^2)$. The dataset demonstrates a smooth nonlinear relationship.

$\lambda(f)$) often fail to match the neural network's performance on the original target. This distinction is central to our analysis.

Ultimately, the primary contribution of this framework is not the surrogate model itself, but the **linearity score** $\lambda(f)$—a simple, interpretable diagnostic of how linearly structured a network's output function is.

4 Experiments

To evaluate the proposed framework, we apply it to a combination of synthetic and real-world datasets, including the Medical Insurance Cost dataset [2], the Concrete Compressive Strength dataset [13], and the California Housing dataset [8].

For each dataset, we evaluate three models: (1) a baseline linear regression model trained directly on the input–output pairs; (2) a neural network trained to predict the target; and (3) a linear surrogate model trained to approximate the output of the neural network. This setup allows us to compute and compare the linearity score $\lambda(f)$ across diverse problem domains, and to assess how closely the neural network's behavior can be captured by a linear approximation.

4.1 Synthetic

We begin with a synthetic regression task defined by the function

$$y = x \cdot \sin(x) + \epsilon, \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, 0.2^2).$$

This function exhibits a continuous, nonlinear structure, providing a testbed for analyzing the expressiveness of linear approximations to neural models. We additionally introduce Gaussian noise to simulate real-world observational error. Figure 1 illustrates a sample of the dataset.

4.2 Medical Insurance Cost

The Medical Insurance Cost dataset contains diverse profiles of individuals, including features such as age, gender, BMI, number of children, and smoking status. For each profile, the dataset records an associated insurance cost. The relationships among features are moderately nonlinear, particularly for smoking status and age, making this a useful benchmark for testing linear decodability.

4.3 Concrete Compressive Strength

The Concrete Compressive Strength dataset models a nonlinear function from eight features—including cement, water, blast furnace slag, and age of the concrete sample—to compressive strength. Each input exhibits a different degree of nonlinearity with respect to the target, making this dataset well-suited for evaluating the limitations of linear surrogates.

Dataset	Linear R^2	NN R^2	Mimic R^2	$\lambda(f)$
Synthetic	-0.0080	0.9746	-0.0140	-0.0102
Medical Costs	0.7554	0.8438	0.6897	0.9342
Concrete	0.6041	0.8501	0.5807	0.6366
California Housing	0.5758	0.7905	0.5638	0.7127

Table 1: Performance metrics across all datasets. The surrogate model's R^2 against the neural network's output defines the linearity score $\lambda(f)$, reflecting the decodability of the network function.

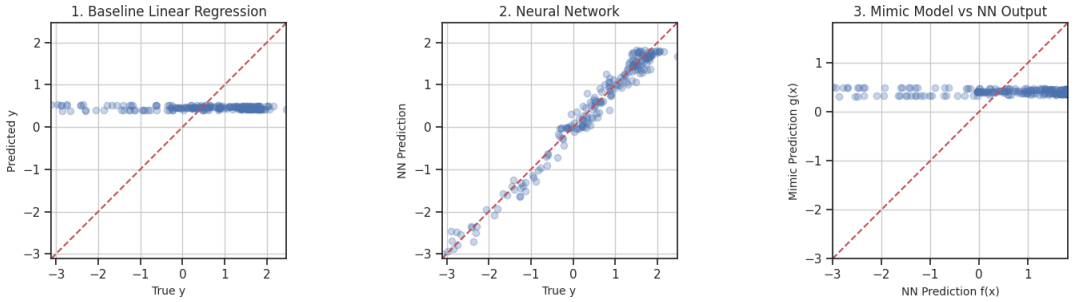

Figure 2: Predictions of the baseline linear model, neural network, and mimic model on the synthetic dataset.

4.4 California Housing

The California Housing dataset includes demographic and geographic information for housing blocks from the 1990 U.S. Census. Features include median income, median house age, average number of rooms and bedrooms, population, average household size, and geographic coordinates. The target variable is median house value (in hundreds of thousands of dollars). With its moderate nonlinearity and blend of structured and unstructured features, this dataset serves as a challenging and realistic case for testing linear decodability.

5 Results and Discussion

5.1 Results

Across all experiments, we found that the linear surrogate g fails to match the neural network's predictive accuracy on the original target y, even when it closely approximates the network's output f. In every dataset, the surrogate underperforms the neural network in predicting y, despite achieving high $\lambda(f)$ in some cases. This indicates that the portions of the function that escape linear approximation—though smaller in variance—are critical for prediction.

In the Medical dataset, for instance, $\lambda(f) \approx 0.93$, yet the surrogate performs worse than the baseline linear model. This suggests that the network's predictive gains are concentrated in its nonlinear behavior, which g cannot capture. In the Concrete and California datasets, we observe similar effects: the surrogate tracks the neural network well but fails to preserve its advantage. Only in the synthetic dataset is this mismatch more extreme, with low $\lambda(f)$ and complete surrogate failure. Together, these results highlight a key limitation: linear decodability does not guarantee predictive usefulness. Below, we discuss findings—summarized in Table 1—on each dataset in detail.

5.1.1 Synthetic

In the case of the synthetic dataset, our linear surrogate fails to approximate the neural network's behavior. The mimic model g performs nearly identically to the baseline linear regression, offering no meaningful improvement despite being trained to match the network's output. This failure is expected: the target function is globally nonlinear, with no regions of local linearity for a simple model to exploit.

The neural network itself performs very well, achieving an R^2 score of 0.9746. However, the surrogate's approximation is extremely poor, yielding $\lambda(f) \approx 0$. This indicates that the learned function f is fundamentally non-linear in input space. That is, no linear transformation $g(x)$ is able to approximate $f(x)$ over the data distribution. Figure 2 visualizes the outputs of all three models.

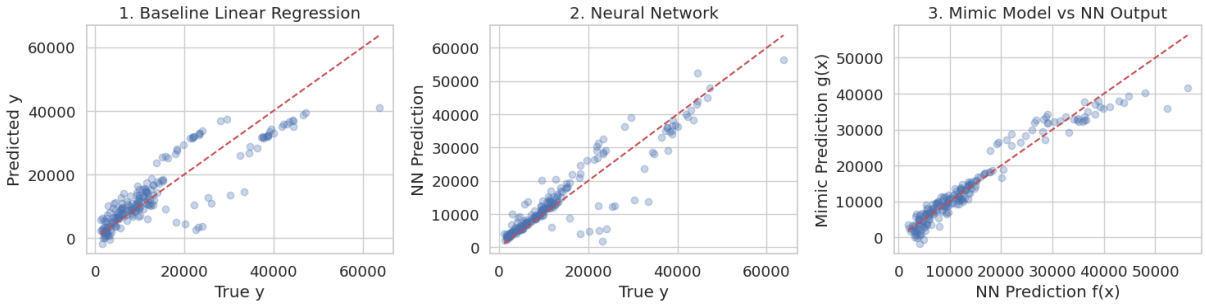

Figure 3: Predictions from the baseline linear model, neural network, and linear surrogate on the Medical Insurance Cost dataset. While the surrogate closely mimics the network's output ($\lambda(f) = 0.9342$), it fails to retain the network's predictive advantage.

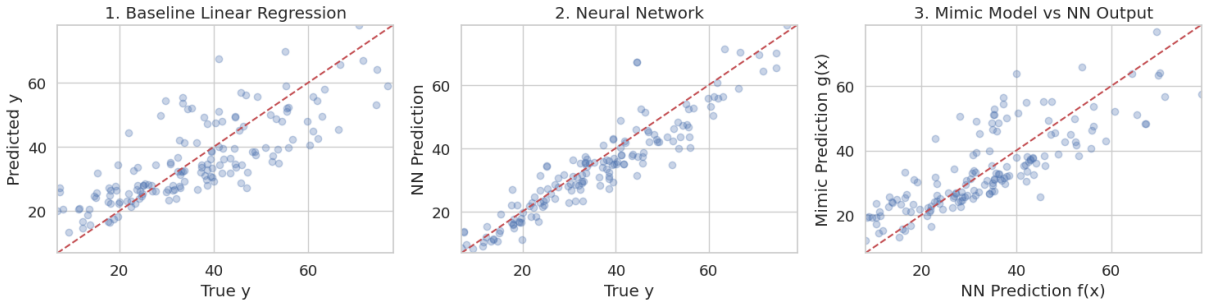

Figure 4: Predictions from the baseline linear model, neural network, and linear surrogate on the Concrete Compressive Strength dataset. The surrogate achieves moderate fidelity to the neural network ($\lambda(f) = 0.6366$), but fails to match its predictive performance.

5.1.2 Medical Insurance Cost

The Medical Insurance Cost dataset provides a moderately structured real-world task. The baseline linear regression model performs reasonably well, achieving $R^2 = 0.7554$, while the neural network improves substantially with $R^2 = 0.8438$, indicating its ability to capture nonlinear patterns. However, the linear surrogate—despite achieving a high linearity score of $\lambda(f) = 0.9342$—fails to retain the neural network's predictive advantage, yielding a lower $R^2 = 0.6897$ against the true target. This discrepancy illustrates a key insight: even when a neural network's function is highly linearly decodable, the decodable portion may not be the source of its improved predictive power. Figure 3 illustrates these results.

These results demonstrate that although the neural network has learned a function that is highly linearly decodable, the portion of its behavior recoverable by a linear surrogate does not account for its improved predictive accuracy. This highlights a subtle limitation of using linear mimics: high decodability ($\lambda(f)$) does not imply that the surrogate will match the network's effectiveness on the true target.

5.1.3 Concrete Compressive Strength

In this dataset, the baseline linear regression model achieves an R^2 score of 0.6041, reflecting a moderate ability to capture the structure in the data. The neural network substantially improves upon this with $R^2 = 0.8501$, indicating that it successfully models complex nonlinear relationships among the input features.

Figure 4 visualizes the predictions from all three models. The linear surrogate trained to mimic the neural network achieves an R^2 of 0.5807 against the true target—close to the baseline's 0.6041—but significantly lower than the network's 0.8501. Nonetheless, the surrogate yields a linearity score of $\lambda(f) = 0.6366$, indicating that it captures over 60% of the variance in the network's output.

This case again illustrates the gap between linear decodability and predictive utility: although the surrogate mimics much of the network's behavior, the nonlinear residuals it misses are critical to the network's superior performance.

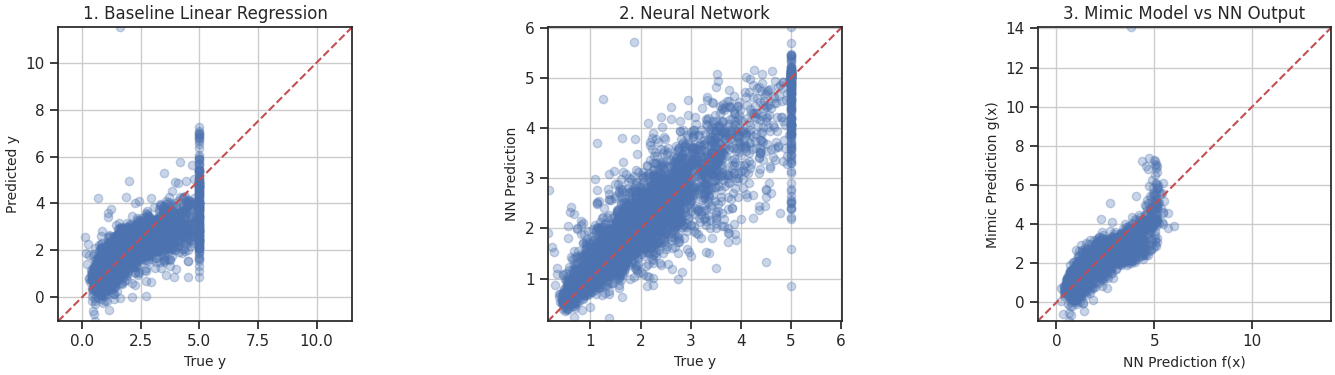

Figure 5: Predictions on the California Housing dataset. The linear surrogate tracks the neural network's output well ($\lambda(f) = 0.7127$) but does not preserve its predictive advantage over the baseline.

5.1.4 California Housing

On this dataset, the baseline linear regression model achieves an R^2 score of 0.5758, indicating moderate linear structure in the input features. The neural network improves on this with $R^2 = 0.7905$, demonstrating that it captures additional nonlinear patterns in the data. The linear surrogate trained to mimic the neural network achieves $R^2 = 0.5638$ against the true target—slightly worse than the baseline—but exhibits relatively high fidelity to the network, with a linearity score of $\lambda(f) = 0.7127$.

Figure 5 visualizes predictions from all three models. The linear surrogate closely approximates the neural network's output but falls short of matching its predictive accuracy on the true target. This gap suggests that while the learned function is largely linearly decodable, its remaining nonlinear components play an outsized role in predictive performance.

As in the Concrete dataset, this result reinforces our central claim: a high $\lambda(f)$ reflects how well a neural network's behavior can be mimicked, but it does not guarantee that the mimic will retain predictive strength. The predictive signal may be concentrated in the nonlinear residues that a linear surrogate cannot express.

5.2 Discussion

Our experiments demonstrate that the linearity score $\lambda(f)$ serves as a reliable measure of how linearly decodable a neural network's output function is. In settings where the learned function is highly nonlinear, such as our synthetic dataset, $\lambda(f)$ drops to zero, confirming that no meaningful linear approximation exists. In contrast, when $\lambda(f)$ is moderate or high—as in the Concrete and California Housing datasets—the surrogate tracks the network's output with reasonable fidelity, but fails to retain its predictive advantage.

Importantly, a high linearity score indicates fidelity to the neural network, but not necessarily to the ground truth. For example, in the case of the Concrete dataset, approximately 36% of the network's output variance is not captured by the linear surrogate. However, that residual variance appears to contain nearly all of the model's predictive improvement over the baseline. The results for the California Housing dataset share this property. These findings suggest that the portion of the neural network's function that escapes linear approximation—though smaller in variance—may be disproportionately responsible for the model's accuracy.

This distinction highlights an important nuance: a linear surrogate with high $\lambda(f)$ offers insight into the network's behavior, but may not reproduce its predictive power if that power depends on nonlinear components. In some high-λ regimes, the surrogate may still track the network's output well, but its usefulness as a predictive substitute depends on how much of the model's performance derives from linearly decodable structure.

While $\lambda(f)$ serves as a useful diagnostic of linear decodability, it is not a comprehensive interpretability method. It offers no insight into a network's internal mechanisms—such as feature salience, representational geometry, or pathway analysis—nor does it produce explanations in terms of input features, like importance scores or causal attributions. Furthermore, our approach is currently limited to regression tasks; extending it to classification would require adapting the framework to logit or softmax outputs. These limitations suggest that $\lambda(f)$ should be viewed not as a general-purpose interpretability tool, but as a focused, principled diagnostic—particularly well-suited for assessing whether a network's behavior exhibits linear structure at the output level.

6 Conclusion

Neural networks are often—and rightfully—described as black boxes. In this work, we proposed a way to measure just how illuminated or comprehensible a network is by quantifying its linear decodability. The metric, $\lambda(f)$, offers a simple and interpretable means of assessing whether a neural network's output function is well-approximated by a linear model. It is straightforward to compute and opens the door to a variety of potential applications in future research.

A natural extension of this work would be to explore how an analogous metric might be applied to classification tasks—such as by projecting logits or softmax outputs—to assess interpretability in non-regression settings. Unlike continuous-valued regression outputs, classification involves discrete decision boundaries, which complicates the notion of linear decodability. However, structure in logit space may still reveal interpretable trends—such as class separability or margin alignment—that a linear surrogate could capture.

Another promising direction is to move beyond quantifying linear decodability and begin characterizing it. Since $\lambda(f)$ captures the proportion of output variance explained by a linear model, a natural next step is to ask: what components of the network's behavior are responsible for this? Investigating which input regions, features, or latent representations contribute most to the linearly decodable structure could deepen our understanding of how and where simple functional patterns emerge within otherwise complex models.

Regardless of how it is used in the future, our experiments across synthetic and real-world datasets show that $\lambda(f)$ reliably reflects how linearly decodable a network's function is. While a high $\lambda(f)$ does not imply predictive equivalence between the surrogate and the original model, it offers a principled way to assess when linear approximation is meaningful—and when it is not. In this way, $\lambda(f)$ helps signal when a model's behavior is no longer entirely opaque, offering a step toward clearer and more interpretable systems.

Acknowledgments

I thank Professor Jorge Silveyra for the valuable discussion we had—it really sparked this project. I am also grateful for the support of the Lafayette College Computer Science Department.

References

[1] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 2018.

[2] Miri Choi. Medical cost personal dataset. `https://www.kaggle.com/datasets/mirichoi0218/insurance`, 2017. Accessed April 2025.

[3] Mark W. Craven and Jude W. Shavlik. Extracting tree-structured representations of trained networks. In *Proceedings of the 9th International Conference on Neural Information Processing Systems*, NIPS'95, page 24–30, Cambridge, MA, USA, 1995. MIT Press.

[4] Boris Hanin and David Rolnick. Complexity of linear regions in deep networks, 2019.

[5] John Hewitt and Christopher D. Manning. A structural probe for finding syntax in word representations. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*. Association for Computational Linguistics, 2019.

[6] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[7] Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study, 2018.

[8] R Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997.

[9] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. SVCCA: Singular vector canonical correlation analysis for deep learning dynamics and interpretability, 2017.

[10] Marco Tulio Ribeiro, Sameer Singh, and Carlos 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*, KDD '16, page 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery.

[11] Naomi Saphra and Adam Lopez. Understanding learning dynamics of language models with SVCCA. In *Proceedings of the 2019 Conference of the North*, pages 3257–3267. Association for Computational Linguistics, 2019.

[12] M. Sato and H. Tsukimoto. Rule extraction from neural networks via decision tree induction. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*, volume 3, pages 1870–1875 vol.3, 2001.

[13] I-Cheng Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete Research*, 28(12):1797–1808, 1998.

[14] Mitchell L. W. Zhou and Sara Hooker. On the unreliability of explanations in regression tasks. *arXiv preprint arXiv:2205.11472*, 2022.