GPT AS VISUAL EXPLAINER
–Supplementary Material–

Anonymous authors
Paper under double-blind review

This document presents supplementary experiments and information regarding our proposed LVX framework. In Section 1, we provide an overview of the algorithm pipeline for LVX. Our related work is Section 2. in Additionally, Section 4 show the calibrated training further improve the OOD performance. Section 5 showcases the additional experimental results obtained from a specialized application focusing on X-Ray diagnosis. Furthermore, in Section 6, we demonstrate the explanation results achieved using self-supervised models. We also provide the raw experimental values in Section 7. Finally, we outline the experimental setup, metric definitions, and protocols followed for dataset collection.

1 ALGORITHM FOR LVX

In this section, we present the pseudocode for the LVX framework, encompassing both the construction stage and the test stage. The algorithmic pipelines are outlined in Algorithm 1 and Algorithm 2.

The category-level tree construction pipeline, as demonstrated in Algorithm 1, involves an iterative process that utilizes a large language model (LLM), like ChatGPT. This process allows us to construct an attribute tree for each visual category. It begins by generating prompts based on the category name and using them to gather attribute information. This forms the initial tree structure. Support images are collected using a text-to-image API, and their visual features are associated with the corresponding tree nodes. The process iterates until the maximum run, continuously refining the attribute tree for the visual category.

During the test stage, as outlined in Algorithm 2, the test samples undergo a traversal process within the constructed category-level trees. The goal is to locate the subtree that best aligns with the correct prediction rationales. This sample-wise visual tree parsing process enables the identification of pertinent attributes and explanations linked to each test sample, providing insights into the decision-making process of the model.

In summary, the LVX framework employs an iterative approach to construct category-level trees, leveraging the knowledge of a large language model. These trees are then utilized during the test stage to extract relevant explanations. This methodology enables us to gain understanding of the model's decision-making process by revealing the underlying visual attributes and rationales supporting its predictions.

2 RELATED WORK

Neural Tree. Neural Trees (NTs) intend to harmonize the performance of Neural Networks (NNs) and interpretability of Decision Trees (DTs) [Craven & Shavlik, 1995; Frosst & Hinton, 2017; Sato & Tsukimoto, 2001; Zilke et al., 2016; Chen et al., 2021] within a unified model. They evolved from mimicking NNs with DTs [Craven & Shavlik, 1995; Frosst & Hinton, 2017; Sato & Tsukimoto, 2001; Zilke et al., 2016; Chen et al., 2021] to becoming inherently interpretable tree-structured networks, adapting their structure via gradient descent [Stromberg et al., 1991; Zhao, 2001; Jordan & Jacobs, 1994; Yang et al., 2018; Tanno et al., 2019; Kontschieder et al., 2015]. Neural-Backed Decision Trees (NBDTs) [Wan et al., 2020] use a trained NN as a feature extractor, replacing its final layer with a decision tree. Our model builds on these advances to create a hierarchical tree from a pre-trained NN and provides *post-hoc* explanations without additional training, which increases interpretability and potentially enhances performance.

Algorithm 1 Language Model as Visual Explainer (LVX)-Construction

Input: Vision model $f = \circ h$, a large language model L, a text-to-image API `T2I`, a training set $D_{tr} = \{\mathbf{x}_i, y_i\}_{i=1}^{M}$, class names $C = \{c_i\}_{i=1}^{n}$ and a concept prompt tree input-output example $\mathcal{P}$.
Output: An explanatory tree T_i for each category c_i.

// Construct the initial Parse Tree
for $i = 1$ **to** n **do**
  In-context Prompt LLM: $d_i = L(c_i, \mathcal{P})$.
  Parse d_i into an initial tree $T_i^{(0)} = \{V_i^{(0)}, E_i^{(0)}\}$.
  Collect support images from text-to-image API: $\{\widetilde{\mathbf{x}}_i\}_{i=1}^{K} = \texttt{T2I}(v)$, where $v \in V_i^{(0)}$.
  Extract features in each tree node: $P_v = \{\mathbf{p}_i\}_{i=1}^{K} = \{g(\widetilde{\mathbf{x}}_i) | \widetilde{\mathbf{x}}_i \in \{\widetilde{\mathbf{x}}_i\}_{i=1}^{K}\}$.
end for
// Parse Tree Refinement
for $t = 0$ **to** t_{max} **do**
  for $j = 1$ **to** M **do**
    Extract feature for training data: $\mathbf{q}_j = g(\widetilde{\mathbf{x}}_j)$.
    Assign training data to a tree node: $v^* = \text{argmin}_{v \in V_{y_j}^{(t)}} D(\mathbf{q}_j, P_v)$.
  end for
  Count the number of samples for each node: $C_{v^*} = \sum_{j=1}^{M} \mathbb{1}\{v^* = \text{argmin}_{v \in V_{y_j}^{(0)}} D(\mathbf{q}_j, P_v)\}$
  Prune the least visited node: $T_i^{(t)} = \texttt{Prune}(T_i^{(t)})$.
  Grow the most visited node: $T_i^{(t+1)} = \texttt{Grow}(T_i^{(t)})$.
  Collect support images from text-to-image API: $\{\widetilde{\mathbf{x}}_i\}_{i=1}^{K} = \texttt{T2I}(v)$, where $v \in V_i^{(t+1)}$.
  Extract features in new tree node: $P_v = \{\mathbf{p}_i\}_{i=1}^{K} = \{g(\widetilde{\mathbf{x}}_i) | \widetilde{\mathbf{x}}_i \in \{\widetilde{\mathbf{x}}_i\}_{i=1}^{K}\}$.
end for
return $T_i^{(t_{\text{max}})}$ as T_i

Algorithm 2 Language Model as Visual Explainer (LVX)-Test

Input: Vision model $f = g \cdot h$, a test sample $\mathbf{x}_{ts}$ and explanatory trees T_i for each category.
Output: A explanatory tree T for test sample.

Prediction on $\mathbf{x}_{ts}$: $\mathbf{q} = g(\mathbf{x}_{ts})$ and $\hat{y} = h(\mathbf{q})$.
Find the top-matched sub-tree in $T_{\hat{y}}$

$$T^* = \underset{T^* \subseteq T_{\hat{y}}}{\text{argmin}} \sum_{i=1}^{k} D(\mathbf{q}, P_{v_i})$$
$$s.t. \quad T^* = \{V^*, E^*\}, v_i \in V^*, |V^*| = k$$

return T^* as the prediction explanation.

Prototype-based Explainable Model. Prototype models use representative training data points to symbolize classes or outcomes [Cover & Hart, 1967; Huang et al., 2002; Kohonen & Kohonen, 1995]. Revived in deep learning and few-shot learning [Snell et al., 2017; Xu et al., 2020], they justify decisions by comparing new instances to key examples [Chen et al., 2019; Yeh et al., 2018; Li et al., 2018]. Recent work has developed this approach through hierarchical and local prototypes [Nauta et al., 2021; Keswani et al., 2022; Taesiri et al., 2022]. However, the prototypes serve as an indirect explanation for model's prediction, necessitating further human justification. Our `LVX` addresses this by assigning semantic roles to prototypes through LLM, turning them from simple similarity points to data points with clear definitions, thereby enhancing explainability.

Composing Foundation Models. Model composition involves merging machine learning models to address tasks, often using modular models for sub-tasks [Andreas et al., 2016b; Hu et al., 2017; Andreas et al., 2016a; Yang et al., 2022], restructured by a symbolic executor [Yi et al., 2018]. Recently, Language Learning Models (LLMs) have been used as central controllers, guiding the logic of existing models [Shen et al., 2023; Gupta & Kembhavi, 2022; Yang et al., 2023; Liang et al., 2023] or APIs [Schick et al., 2023], with language as a universal interface [Zeng et al., 2022]. However, composing language model with non-language ones lacks a unified interface for bilateral communication. In this study, we propose the use of a text-to-image API as a medium to enable the language model to share its knowledge with the visual task. This allows the vision model to benefit from the linguistic context and knowledge hierarchy, thereby enhancing its transparency.

3 Data Collection and Analysis

3.1 Annotation Creation

The creation of test annotations for three datasets involved a semi-automated approach implemented in two distinct steps. This process was a collaborative effort between human annotators, a language model (ChatGPT), and CLIP [Radford et al., 2021], ensuring both efficiency and reliability.

1. **Concept Tree Creation**: In this first step, we utilized ChatGPT [1] to generate initial attribute trees for each class by inputting the visual category name, similar to Section 3.1. Subsequently, human annotators verified, refined, and organized the attributes tree for each class. During this stage, annotators also confirmed the correctness of existing attributes and added relevant attributes to further expand the concept tree.
2. **Attribute Verification**: To determine whether an attribute was present or absent in an image, we employed an ensemble of predictions [2] from multiple CLIP [Radford et al., 2021] models. We filtered the top-5 attributes predicted by CLIP and sought human judgments to verify their correctness. To streamline this process, we developed an annotation tool with a user interface, which is illustrated in Figure 1.

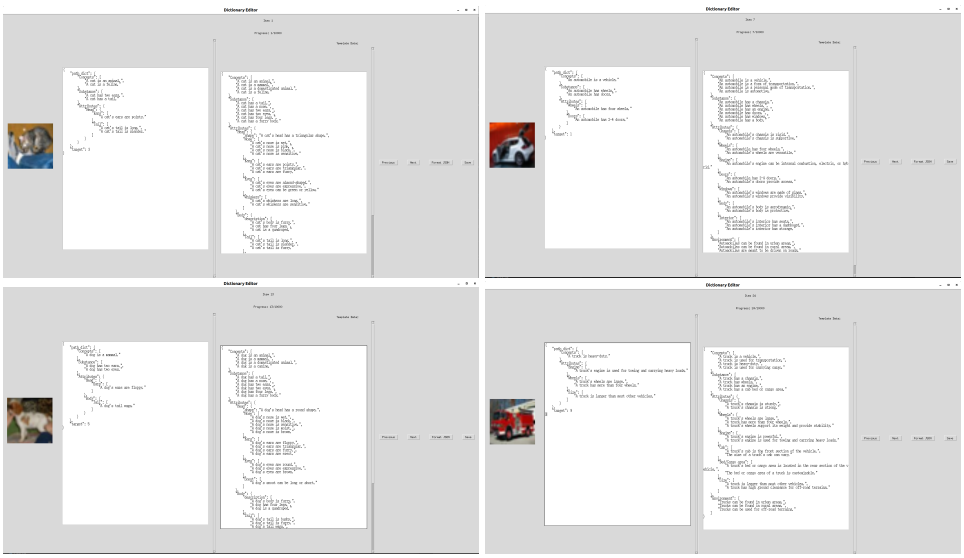


Figure 1: Software tool interface for parse tree annotation.

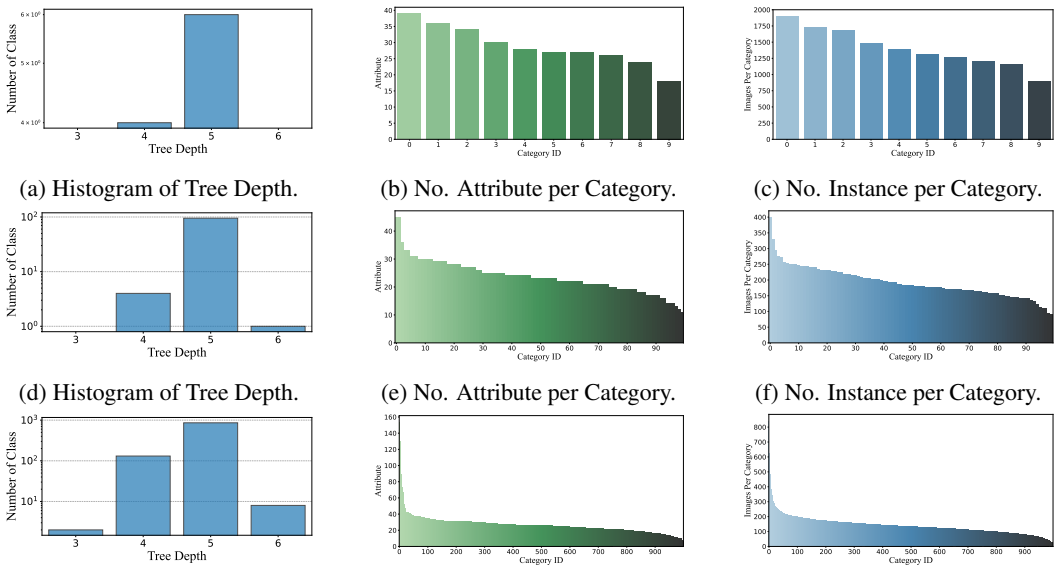
Table 1: Support set Dataset Statistics.

Dataset Name	No. Categories	No. Attributes	No. Images
CIFAR-10 Support	10	289	14,024
CIFAR-100 Support	100	2,359	19,168
ImageNet Support	1,000	26,928	142,034

3.2 Support Data

Data Collection. The practice of our `LVX` approach relies on the creation of a customized support set for each task. We use a reject sampling approach. Initially, images are generated using either Bing or the Stable Diffusion Model. Subsequently, CLIP is applied to determine if the CLIP score exceeds 0.5, based on the raw cosine similarities calculated by averaging CLIP

[1] `https://chat.openai.com/`
[2] `https://github.com/mlfoundations/open_clip`



(a) Histogram of Tree Depth. (b) No. Attribute per Category. (c) No. Instance per Category.

(d) Histogram of Tree Depth. (e) No. Attribute per Category. (f) No. Instance per Category.

(g) Histogram of Tree Depth. (h) No. Attribute per Category. (i) No. Instance per Category.

Figure 2: Statistics of the support dataset sets of (Row1) CIFAR10, (Row2) CIFAR100 and (Row3) ImageNet. We examine the (a,d,g) Tree Depth, (b,e,h) Number of Attributes for each category and (c,f,i) Number of image for each category to demonstrate the diversity of collected attributes and the completeness of hierarchical annotations.

`ViT-B/32`,`ViT-B/16`,`ViT-B/14`. If the score is above the specified threshold, the image is retained; otherwise, it is discarded

To optimize the image collection process, we merge the retrieved images from all models, leading to time and effort savings. Moreover, if an attribute generated by the LLM (Language and Vision Model) already exists in the dataset, we can conveniently retrieve the corresponding image from the existing pool. Consequently, once the initial models have completed data collection, the remaining models no longer need to collect new images and can instead retrieve local samples.

Data Statistics. We present the statistics of the support datasets collected for CIFAR10, CIFAR100, and ImageNet, highlighting the diversity and comprehensiveness of our dataset. Table 1 and Figure 2 provide an overview of these statistics. Specifically, we include the number of attributes, the number of samples for each category, the total number of samples in the dataset, as well as the distribution of tree depths. This rich collection of data showcases the diverse range of attributes and categories covered in our dataset, making it a valuable resource for training and evaluation purposes.

3.3 Limitations of Newly Collected Dataset for Explanation

While collecting a newly curated dataset can be advantageous for our tasks, it is important to acknowledge certain limitations when using such datasets for explanation purposes. Two key limitations arise: the presence of out-of-distribution samples and potential bias in the dataset.

- **False Positive Images:** We observed that both Bing and the Stable Diffusion model occasionally generate imperfect images from textual descriptions, manifesting incorrect or entangled patterns. For instance, the word "crane" could represent either a construction machine or a bird, leading to ambiguity. Furthermore, an image described as "a dog with a long tail" could potentially include not only the tail but also the head and legs, reflecting a broader scope than intended.
- **Out-of-Distribution Samples:** Newly collected datasets may include samples that are out-of-distribution, i.e., they do not align with the source data distribution of interest. These out-of-distribution samples can introduce challenges in generating accurate and reliable explanations. As a result, explanations based solely on a newly collected dataset may not generalize well to unseen instances outside the support dataset distribution.

- **Potential Bias:** Biases in the collection process or the underlying data sources can inadvertently influence the dataset, leading to biased explanations. Biases can emerge due to various factors, such as data collection source, or imbalances in attribute distributions. Consequently, relying solely on a newly collected dataset for explanations may introduce unintended biases into the interpretation process.

Solutions: To deal with mistakes in gathering images, we used two approaches. First, we used the CLIP model to sift through the images because it's good at understanding how close an image is to text concepts, helping us remove most mixed-up and incorrect images. Second, we manually sorted out words that have more than one meaning. For instance, we made it clear whether "crane" refers to the bird or the machine by labeling it as "crane (bird)" or "crane (machine)".

To mitigate the challenges posed by OOD samples and data bias, we adopt a cautious approach. Specifically, we do not directly train our models on the newly collected support dataset. Instead, we utilize this dataset solely for the purpose of providing disentangled attributes for explanations. By decoupling the training data from the support data, we aim to reduce the impact of OOD samples and potential data biases, thus promoting a more robust and unbiased analysis.

While these limitations should be taken into consideration, our focus on acquiring a dataset with disentangled attributes enhances our ability to investigate and interpret the models' behavior, enabling us to gain valuable insights in a controlled and interpretable manner.

4 Experiments on Out-of-Distribution (OOD) Evaluation

In this section, we expand upon the calibrated model's performance by examining its capabilities in Out-of-Distribution (OOD) scenarios. Our focus is on assessing the robustness and generalization of the model, building on our preliminary findings of improved in-domain performance as depicted in Figure 7 in the main paper, which are calibrated on CIFAR100. With ImageNet trained ResNet50 and ViT-S, we evaluate its OOD accuracy with and without calibration training on the ImageNet-A and ImageNet-Sketch datasets.

The results of OOD generalization, quantified by Top-1 Accuracy, are listed in Table 2. For both ResNet-50 and ViT-S 16 models, we notice significant improvements in accuracy in ImageNet-A and ImageNet-S compared to the baselines. These results imply substantial advancements in the model's OOD generalization after calibration. The refinements introduced through model calibration did not only elevate in-domain performance on CIFAR100 but also enhanced the model's proficiency in managing out-of-domain data, establishing its adaptability and robustness.

Table 2: OOD Generalization Results with and without calibration by Top-1 Accuracy

Model	ImageNet (In-Domain)	ImageNet-A	ImageNet-S
	Baseline/**Calibrated**	Baseline/**Calibrated**	Baseline/**Calibrated**
ResNet-50	76.13/**76.54(+0.41)**	18.96/**23.32(+4.36)**	24.10/**31.42(+7.32)**
ViT-S 16	77.85/**78.21(+0.36)**	13.39/**18.72(+5.33)**	32.40/**37.21(+4.81)**

5 Experiments on Chest X-Ray Diagnosis

In this section, we evaluate the performance of our `LVX` method in security-critical domains, specifically medical image analysis. We train neural networks for chest X-ray diagnosis and utilize `LVX` to interpret and calibrate the predictions.

We adopted the DenseNet-121 architecture for disease diagnosis in our study. The model was trained on the Chestx-ray14 dataset [Wang et al., 2017], which consists of chest X-ray images encompassing 14 diseases, along with an additional "`No Finding`" class. The DenseNet-121 architecture is specifically designed to generate 14 output logits corresponding to the different diseases. During training, we employed a weighted binary cross-entropy loss [Wang et al., 2017] for each disease category to optimize the model.

For optimization, we utilized the Adam optimizer [Kingma & Ba, 2014] with an initial learning rate of 1e-4, a weight decay of 1e-5, and a batch size of 32. The model underwent training for a total of 18 epochs.

Model Explanation. To enhance interpretability, we incorporated our `LVX` framework into the model. Instead of acquiring images from online sources, we gathered the support set directly from

the training data. To accomplish this, we utilized a parse tree generated by the ChatGPT language model. Leveraging this parse tree, we applied a MedCLIP [Wang et al., 2022] model to retrieve the most relevant images for each attribute from the training set. These retrieved images served as our support sets for the LVX framework.

Compared to applying the LVX framework on single-label classification, the Chestx-ray14 dataset poses a multi-label classification challenge. In this dataset, each sample can belong to multiple disease categories simultaneously. Therefore, we modified the LVX framework to accommodate and handle the multi-label nature of the classification task.

Specifically, for each input image $\mathbf{x}$, we predict its label $\hat{y} = f(\mathbf{x}) \in \{0, 1\}^{14}$. To create the visual parse tree, we begin by establishing the root node. If all elements of $\hat{y}$ are 0, the root node is set to "No Findings". Conversely, if any element of $\hat{y}$ is non-zero, the root node is labeled as "has Findings". For each positive finding, we construct a separate parse tree, with these sub-trees becoming the children nodes of the root. By combining these sub-trees, we obtain a comprehensive and coherent explanation for the image. This modification enables us to effectively handle the multi-label nature of the classification task, providing meaningful and interpretable explanations for images with multiple positive findings.

To establish the ground-truth explanation label, we adopt a MedCLIP [Wang et al., 2022] model to filter the top-5 attributes for each positive finding of the image. These attributes are then organized into a tree structure. This approach serves as an automatic explanation ground-truth, thereby eliminating the requirement for manual annotations from domain experts.

In addition to providing explanations, we aim to calibrate the model predictions with the parse tree. To achieve this, we apply a modified hierarchical contrastive loss individually on each finding. We then calculate the average of these losses, which serves as our overall loss term. We thus fine-tune the model for 3 epochs using the hieracical term and weighted cross-entropy.

Explanation Results. We conducted a comprehensive comparison of the explanation performance of our proposed LVX method against the Random and Constant baselines. The numerical results, depicted in Figure 3, highlight the superiority of our LVX approach.

Additionally, we present the parsed visual tree, showcased in Figure 4, to provide a clearer understanding of the interpretability achieved by our LVX method. Notably, our approach effectively interprets the decision-making process of black-box neural networks. For instance, in the depicted case on the right, our method accurately identifies the presence of visible fluid in the lung space and establishes its relevance to the model's prediction. Consequently, LVX enables clinical professionals to make well-informed justifications for their patients, enhancing the overall decision-making process.

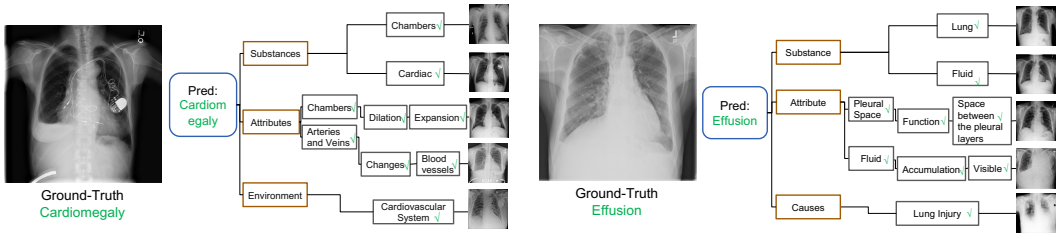


Figure 4: Explanation examples for the chest xray diagnosis task.

Calibration Results. Table 3 presents the performance comparison between the baseline model and the model with calibration, measured in terms of the Area Under the Curve (AUC) score for each disease type. The AUC score provides a measure of the model's ability to discriminate between positive and negative cases.

The calibrated model shows notable improvements in several disease types compared to the baseline. Notably, Hernia demonstrates the most significant improvement, with an AUC score of 0.936 compared to 0.914 for the baseline. This indicates that the calibration process has enhanced the model's ability to accurately detect Hernia cases.

In summary, the results clearly indicate that the LVX method significantly enhances the model's predictive capabilities, as evident from its improved calibration performance across various disease types. This suggests that the incorporation of visual attributes effectively enhances the accuracy and reliability of the model's predictions, thereby leading to improved diagnostic outcomes. The

findings highlight the potential of the LVX approach to enhance the overall performance of the model and its practical applicability in medical diagnostics.

Table 3: Model performance with and without calibration. AUC scores are reported for each disease type. *Avg.* indicates the average score.

Finding	**Baseline**	**LVX**
Atelectasis	0.767	**0.779**
Consolidation	0.747	**0.755**
Infiltration	0.683	**0.698**
Pneumothorax	0.865	**0.873**
Edema	0.845	**0.851**
Emphysema	0.919	**0.930**
Fibrosis	0.832	**0.830**
Effusion	0.826	**0.831**
Pneumonia	0.721	**0.719**
Pleural Thickening	0.784	**0.793**
Cardiomegaly	0.890	**0.894**
Nodule	0.758	**0.776**
Mass	0.814	**0.830**
Hernia	0.914	**0.936**
Avg.	0.812	**0.821**

6 Experiments on Self-supervised models

In this section, we examine self-supervised models to explore their interpretability. Unlike the supervised models discussed in the main paper, self-supervised models acquire representations without relying on labeled data. By investigating these models, our goal is to gain insights into the interpretability of the learned representations and reveal the underlying manifold discovered solely from the input data.

Model to be Explained. Our objective is to offer a comprehensive explanation for self-supervised models trained on ImageNet-1k. These models include ResNet50 trained using SimCLR [Chen et al., 2020a], BYOL [Grill et al., 2020], SwAV [Caron et al., 2020], MoCov3 [Chen et al., 2020b], DINO [Caron et al., 2021], and ViT-S trained with MoCov3 [Chen* et al., 2021] and DINO [Caron et al., 2021]. The networks are subsequently fine-tuned through linear probing while keeping the *backbone fixed*. We then utilize our LVX approach to provide explanations for their predictions. Additionally, we compare these self-supervised models with their supervised counterparts to highlight the differences in representation between the two approaches.

Numerical Results. Table 4 presents the evaluation results of self-supervised models. Our analysis reveals a strong correlation between the explanatory performance and the overall model accuracy.

However, we also noticed that self-supervised models based on transformer architecture exhibit greater attribute disentanglement compared to supervised models, despite potentially having slightly lower performance. This phenomenon is evident when comparing the DINO ViT-S/16 model and the supervised ViT-S/16 model within the context of tree parsing explanation. Although the DINO ViT-S/16 model shows slightly lower overall performance, it outperforms the supervised model in terms of providing accurate attribute explanation.

These results underscore the potential benefits of self-supervised learning in uncovering meaningful visual attributes without explicit supervision. While self-supervised models may exhibit marginally lower performance on certain tasks, their ability to capture rich visual representations and attribute disentanglement highlights their value in understanding complex visual data.

Table 4: Explanation performance analysis of self-supervised models utilizing linear probing.

Method	Top-1 Acc	TED↓	MCS↑	TK ↑
ResNet50-SimCLR	69.2	9.38	23.72	46.53
ResNet50-BYOL	71.8	9.29	24.66	48.29
ResNet50-MoCov3	74.6	9.19	25.59	50.17
ResNet50-DINO	75.3	9.14	25.77	50.71
ResNet50-SwAV	75.3	9.15	25.82	50.69
ResNet50-Sup	76.1	9.09	25.99	51.19
ViT-S/16-MoCov3	73.2	9.16	25.25	49.40
ViT-S/16-DINO	77.0	8.99	26.61	52.05
ViT-S/8-DINO	79.7	8.89	27.62	53.95
ViT-S/16-Sup	77.9	9.10	25.73	50.34

7 RAW RESULTS

This section presents the raw numerical results for Figure 5, as depicted in the main paper. Specifically, Table 5 provides the results for CIFAR-10, Table 6 for CIFAR-100, and Table 7 for ImageNet. We also observed that larger networks within the same model family deliver better results. As the models improve, so does the accuracy of the explanations, suggesting that larger networks facilitate more effective explanations. This is demonstrated by the increase in MCS and TK scores as ResNet deepens on CIFAR-100 and ImageNet, aligning with the general belief that larger neural networks offer enhanced generalization and structural representation capabilities.

Table 5: Explanation performance comparison on CIFAR-10

Model	TED↓			MCS↑			Tree Kernel↑		
	rand.	const.	LVX	rand.	const.	LVX	rand.	const.	LVX
VGG13	9.21	32.97	8.21	28.98	18.77	32.31	59.49	58.41	63.43
VGG16	9.23	32.89	8.14	29.11	19.11	32.55	59.34	59.55	63.79
VGG19	9.15	32.85	8.15	30.67	19.10	31.78	59.44	59.39	63.32
ResNet18	9.21	32.90	8.52	28.95	18.92	30.24	58.94	58.87	61.19
ResNet34	9.21	32.92	8.16	28.95	18.98	32.07	59.06	59.01	63.27
ResNet50	9.21	32.89	8.44	28.96	19.00	31.09	59.16	59.21	62.06
DenseNet121	9.19	32.89	8.20	29.09	19.11	32.13	59.53	59.48	63.44
DenseNet161	9.20	32.88	8.19	29.07	19.11	32.12	59.35	59.48	63.73
DenseNet169	9.21	32.88	8.18	29.25	19.08	32.13	59.52	59.46	63.46
MobileNet_v2	9.20	32.89	8.41	29.24	19.09	31.61	59.53	59.38	61.87
GoogLeNet	9.23	32.96	8.41	28.66	18.86	30.75	58.62	58.71	61.38
Inception_v3	9.20	32.89	8.39	29.19	19.03	31.02	59.37	59.27	61.85

Table 6: Explanation performance comparison on CIFAR-100

Model	TED↓			MCS↑			Tree Kernel↑		
	rand.	const.	LVX	rand.	const.	LVX	rand.	const.	LVX
ResNet20	9.61	28.77	8.96	22.65	17.60	24.89	45.52	46.96	47.70
ResNet32	9.57	28.67	8.86	22.84	17.92	25.39	46.45	47.87	48.58
ResNet44	9.54	28.60	8.81	23.48	18.34	25.97	47.42	48.87	49.79
ResNet56	9.52	28.54	8.83	23.87	18.60	26.47	48.04	49.58	50.17
MBNv2-x0.5	9.55	28.58	8.87	23.43	18.19	25.50	47.13	48.58	49.08
MBNv2-x0.75	9.43	28.43	8.76	24.47	18.87	26.71	49.19	50.55	51.21
MBNv2-x1.0	9.48	28.35	8.73	24.35	19.02	27.09	49.27	50.68	51.47
MBNv2-x1.4	9.43	28.16	8.65	24.87	19.47	27.41	50.52	52.08	52.91
RepVGG A0	9.44	28.28	8.74	24.65	19.21	26.84	49.78	51.37	52.01
RepVGG A1	9.42	28.21	8.72	25.27	19.59	27.43	50.63	52.17	52.81
RepVGG A2	9.40	28.08	8.70	25.46	19.82	27.99	51.26	52.87	53.04

Table 7: Explanation performance comparison on ImageNet

Model	TED↓			MCS↑			Tree Kernel↑		
	rand.	const.	LVX	rand.	const.	LVX	rand.	const.	LVX
ResNet18	9.83	34.15	9.30	22.52	16.82	23.87	45.32	44.85	46.85
ResNet34	9.75	33.78	9.17	23.74	17.71	25.09	47.66	47.16	49.24
ResNet50	9.68	33.58	9.09	24.59	18.35	25.99	49.38	48.97	51.19
ResNet101	9.64	33.48	9.04	24.94	18.66	26.51	50.25	49.77	51.99
ViT-T16	10.42	35.44	9.99	15.07	11.25	15.91	30.30	29.92	31.24
ViT-S16	9.69	33.61	9.10	24.16	18.01	25.73	48.53	48.05	50.34
ViT-B16	9.62	33.37	8.99	25.20	18.79	27.01	50.64	50.22	52.76
ViT-L16	9.45	32.84	8.79	27.27	20.35	29.29	54.83	54.36	57.14

8 EXPERIMENTAL SETUP

In this section, we provide detailed information about our experimental setup to ensure the reproducibility of our method.

8.1 EVALUATION METRICS

To evaluate the effectiveness of our proposed tree parsing task, we have developed three metrics that leverage conventional tree similarity and distance measurement techniques.

- **Tree Edit Distance (TED)** [Bille, 2005]: The Tree Edit Distance (TED) quantifies the minimum number of editing operations required to transform one hierarchical tree into another. It measures the structural dissimilarity between trees by considering node and edge modifications, insertions, and deletions. With smaller TED, the two graphs are more similar. Let's define the Tree Edit Distance formally:

  Tree Edit Distance: Given two trees T_1 and T_2, represented as rooted, labeled, and ordered trees, we define the Tree Edit Distance between them as $TED(T_1, T_2)$.
  The $TED(T_1, T_2)$ is computed recursively as follows:
  - If both T_1 and T_2 are empty trees, i.e., they do not have any nodes, then $TED(T_1, T_2) = 0$.
  - If either T_1 or T_2 is an empty tree, i.e., it does not have any nodes, then $TED(T_1, T_2)$ is the number of nodes in the non-empty tree.
  - Otherwise, let r_1 and r_2 be the roots of T_1 and T_2, respectively. Let $T_1^{'}$ and $T_2^{'}$ be the subtrees obtained by removing the roots r_1 and r_2 from T_1 and T_2, respectively.
    * If $r_1 = r_2$, then $TED(T_1, T_2)$ is the minimum among the following values:
      · $TED(T_1^{'}, T_2^{'}) + TED(\text{children of } r_1, \text{children of } r_2)$, where $TED(\text{children of } r_1, \text{children of } r_2)$ is the TED computed recursively between the children of r_1 and r_2.
      · $1 + TED(T_1^{'}, T_2^{'})$, which represents the cost of deleting the root r_1 and recursively computing TED between $T_1^{'}$ and $T_2^{'}$.
      · $1 + TED(T1, T_2^{'})$, which represents the cost of inserting the root r_2 into T_1 and recursively computing TED between T_1 and $T_2^{'}$.
      · $1 + TED(T_1^{'}, T_2)$, which represents the cost of deleting the root r_1 and inserting the root r_2 into T_2, and recursively computing TED between $T_1^{'}$ and T_2.
    * If $r_1 \neq r_2$, then $TED(T_1, T_2)$ is the minimum among the following values:
      · $TED(T_1^{'}, T_2^{'}) + TED(\text{children of } r_1, \text{children of } r_2)$, where $TED(\text{children of } r_1, \text{children of } r_2)$ is the TED computed recursively between the children of r_1 and r_2.
      · $1 + TED(T_1^{'}, T_2)$, which represents the cost of deleting the root r_1 and recursively computing TED between $T_1^{'}$ and T_2.
      · $1 + TED(T_1, T_2^{'})$, which represents the cost of inserting the root r_2 into T_1 and recursively computing TED between T_1 and $T_2^{'}$.

  The $TED(T_1, T_2)$ is the final result obtained after applying the above recursive computation.

- **Maximum Common Subgraph (MCS)**[Raymond & Willett, 2002; Kann, 1992]: The Maximum Common Subgraph (MCS) identifies the largest shared subgraph between two trees, measuring the similarity and overlap of their hierarchical structures. Here's the mathematical definition of the Maximum Common Subgraph:

Maximum Common Subgraph: Given two trees, $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$, where V_1 and V_2 are the sets of vertices and E_1 and E_2 are the sets of edges for each graph, respectively, we define the **Maximum Common Subgraph** as:

$$\begin{aligned}
&\text{MCS}(T_1, T_2) = (V_{\text{MCS}}, E_{\text{MCS}}) \\
&\text{maximize} \quad |V_{\text{MCS}}| \\
&\text{subject to} \quad V_{\text{MCS}} \subseteq V1 \quad \text{and} \quad V_{\text{MCS}} \subseteq V2 \\
&\qquad\qquad E_{\text{MCS}} \subseteq E1 \quad \text{and} \quad E_{\text{MCS}} \subseteq E2 \\
&\qquad\qquad \text{For any pair of vertices } u, v \text{ in } V_{\text{MCS}}, \text{ if } (u, v) \text{ is an edge in } E_{\text{MCS}}, \\
&\qquad\qquad\quad \text{then } (u, v) \text{ is an edge in } T_1 \text{ and } T_2, \text{ and vice versa.}
\end{aligned}$$

In our paper, We report the normalized MCS score as our measurement of tree similarity $\frac{|\text{MCS}(T_{pred}, T_{gt})| \times 100}{\sqrt{|T_{pred}||T_{gt}|}}$, where a higher score indicates greater similarity between the graphs. Here, $|\cdot|$ represents the number of nodes in a tree. We employ this normalization to address the scenario where one tree is significantly larger and encompasses all other trees as subtrees. By dividing the MCS score by the square root of the product of the numbers of nodes in the predicted tree (T_{pred}) and the ground truth tree (T_{gt}), we ensure a fair comparison across trees of varying sizes.

- **Tree Kernels (TK)**: Tree Kernels (TK) evaluate tree similarity by leveraging shared substructures, assigning higher scores to trees with common subtrees or substructures. To enhance the match, we set the decaying factor for two adjacent tree layers to 0.5, where larger values lead to better matches. Let's define the subtree kernel mathematically:

Tree Kernel: Given two trees, T_1 and T_2, represented as rooted, labeled, and ordered trees, we define the subtree kernel as follows:

$$TK(T_1, T_2) = \sum_{T} \sum_{T'} \theta(T, T') \times \theta(T, T') \times \lambda^{\max(\text{depth}(r), \text{depth}(r'))}$$

Let $TK(T_1, T_2)$ denote the similarity between subtrees T_1 and T_2 using the subtree kernel. Additionally, let $\theta(T, T')$ represent the count of shared common subtrees between trees T and T'. Furthermore, let r and r' be the roots of T and T' respectively. $\lambda < 1.0$ is the decaying factor that make sure the tree closer to the root hold greater significance.
The $\theta(T, T')$ is computed recursively as follows:

(a) If both T and T' are leaf nodes, then $\theta(T, T') = 1$ if the labels of T and T' are the same, and 0 otherwise.
(b) If either T or T' is a leaf node, then $\theta(T, T') = 0$.
(c) Otherwise, let $\{T_1, T_2, \ldots, T_n\}$ be the child subtrees of T, and $\{T'_1, T'_2, \ldots, T'_{n'}\}$ be the child subtrees of T'.
  - If the labels of r and r' are the same, then $\theta(T, T')$ is the sum of the products of $\theta(T_i, T'_j)$ for all combinations of i and j, where i ranges from 1 to n and j ranges from 1 to n'.
  - If the labels of r and r' are different, then $\theta(T, T')$ is 0.
  - Additionally, if T and T' are isomorphic (have the same structure), then $\theta(T, T')$ is incremented by 1.

In the paper, the Tree Kernel (TK) score is normalized to accommodate trees of different sizes. The normalized TK score is computed as: $\frac{TK(T_{pred}, T_{gt}) \times 100}{\sqrt{TK(T_{pred}, T_{pred}) TK(T_{gt}, T_{gt})}}$. The kernel value serves as a measure of similarity, where higher values indicate greater similarity.

8.2 Model Checkpoints

For our experiments, we utilize publicly available pre-trained models. Specifically, we employ CIFAR10 models from `https://github.com/huyvnphan/PyTorch_CIFAR10`, CIFAR100 models from `https://github.com/chenyaofo/pytorch-cifar-models`,

and ImageNet models from `torchvision` package and `timm` package. The self-supervised models are downloaded from their respective official repositories.

8.3 Explanation Baselines

In our endeavor to explain visual models using tree-structured language clues without any annotations, we are not aware of any existing methods that can be directly applied to this task. Therefore, we have developed two naive baselines, namely `Random` and `Constant`, which serve as points of comparison against our proposed method, `LVX`. In the following sections, we provide a more detailed explanation of our implementation of these two baseline methods.

Random Baseline. The Random baseline generates explanations in a random manner. For each input image, we first predict its category and then proceed to randomly sample 5 nodes from the category-specific tree. We retain the edges that connect these sampled nodes to create a tree-structured language clue, which serves as an explanation. By employing this baseline, we get a lower bound for the performance with a purely random approach. This comparison demonstrates that our method, at least, is superior to making random guesses, thereby highlighting the effectiveness of our approach.

Constant Baseline The Constant baseline generates a fixed tree-structured language clue for input images that are predicted to belong to the same class. It remains unaffected by the content or characteristics of the image. In this baseline, we utilize the initial explanatory tree $T_i^{(0)}$ generated by the LLM as the template. By employing this baseline, we can assess the effectiveness of our proposed method by comparing it to a static, non-adaptive approach.

By conducting a comparative analysis between the performance of `LVX` and these two simplistic baselines, we are able to validate the following conclusions:

(a) Our approach surpasses random guessing, which disregards the image content and relies solely on chance.

(b) Our explanations outperform a fixed template that describes the general properties of a given category, as our method adapts to individual image characteristics.

Through this comparison, we establish the superiority of `LVX` in generating explanations that are tailored to the specific image content, effectively outperforming random guessing and static template-based approaches.

8.4 Source Code and Data

For your reference, the full source code is provided in the attached zipped file named LVEXPLAINER-CODE.ZIP. The code is fully documented. Please refer to the code for detailed information and implementation details.

However, please note that we have not included the dataset in this release due to potential privacy concerns. We prioritize the protection of sensitive data. Once the paper is accepted, we will release the dataset to ensure transparency and reproducibility of our research.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Learning to compose neural networks for question answering. *arXiv preprint arXiv:1601.01705*, 2016a.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 39–48, 2016b.

Philip Bille. A survey on tree edit distance and related problems. *Theoretical computer science*, 337 (1-3):217–239, 2005.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. 2020.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

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

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 13–18 Jul 2020a. URL `https://proceedings.mlr.press/v119/chen20j.html`.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020b.

Xinlei Chen*, Saining Xie*, and Kaiming He. An empirical study of training self-supervised vision transformers. *arXiv preprint arXiv:2104.02057*, 2021.

Ying Chen, Feng Mao, Jie Song, Xinchao Wang, Huiqiong Wang, and Mingli Song. Self-born wiring for neural trees. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5047–5056, 2021.

Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

Mark Craven and Jude Shavlik. Extracting tree-structured representations of trained networks. *Advances in neural information processing systems*, 8, 1995.

Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. *arXiv preprint arXiv:2211.11559*, 2022.

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pp. 804–813, 2017.

Yea-Shuan Huang, Cheng-Chin Chiang, Jun-Wei Shieh, and Eric Grimson. Prototype optimization for nearest-neighbor classification. *Pattern Recognition*, 35(6):1237–1245, 2002.

Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

Viggo Kann. On the approximability of the maximum common subgraph problem. In *STACS*, volume 92, pp. 377–388. Citeseer, 1992.

Monish Keswani, Sriranjani Ramakrishnan, Nishant Reddy, and Vineeth N Balasubramanian. Proto2proto: Can you recognize the car, the way i do? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10233–10243, 2022.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Teuvo Kohonen and Teuvo Kohonen. Learning vector quantization. *Self-organizing maps*, pp. 175–189, 1995.

Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. Deep neural decision forests. In *Proceedings of the IEEE international conference on computer vision*, pp. 1467–1475, 2015.

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

Yaobo Liang, Chenfei Wu, Ting Song, Wenshan Wu, Yan Xia, Yu Liu, Yang Ou, Shuai Lu, Lei Ji, Shaoguang Mao, et al. Taskmatrix. ai: Completing tasks by connecting foundation models with millions of apis. *arXiv preprint arXiv:2303.16434*, 2023.

Meike Nauta, Ron Van Bree, and Christin Seifert. Neural prototype trees for interpretable fine-grained image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14933–14943, 2021.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

John W Raymond and Peter Willett. Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *Journal of computer-aided molecular design*, 16:521–533, 2002.

Makoto Sato and Hiroshi 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, pp. 1870–1875. IEEE, 2001.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580*, 2023.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

J-E Stromberg, Jalel Zrida, and Alf Isaksson. Neural trees-using neural nets in a tree classifier structure. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, pp. 137–140. IEEE Computer Society, 1991.

Mohammad Reza Taesiri, Giang Nguyen, and Anh Nguyen. Visual correspondence-based explanations improve ai robustness and human-ai team accuracy. *Advances in Neural Information Processing Systems*, 35:34287–34301, 2022.

Ryutaro Tanno, Kai Arulkumaran, Daniel Alexander, Antonio Criminisi, and Aditya Nori. Adaptive neural trees. In *International Conference on Machine Learning*, pp. 6166–6175. PMLR, 2019.

Alvin Wan, Lisa Dunlap, Daniel Ho, Jihan Yin, Scott Lee, Henry Jin, Suzanne Petryk, Sarah Adel Bargal, and Joseph E Gonzalez. Nbdt: neural-backed decision trees. *arXiv preprint arXiv:2004.00221*, 2020.

Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2097–2106, 2017.

Zifeng Wang, Zhenbang Wu, Dinesh Agarwal, and Jimeng Sun. Medclip: Contrastive learning from unpaired medical images and text. *arXiv preprint arXiv:2210.10163*, 2022.

Wenjia Xu, Yongqin Xian, Jiuniu Wang, Bernt Schiele, and Zeynep Akata. Attribute prototype network for zero-shot learning. *Advances in Neural Information Processing Systems*, 33:21969–21980, 2020.

Xingyi Yang, Jingwen Ye, and Xinchao Wang. Factorizing knowledge in neural networks. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIV*, pp. 73–91. Springer, 2022.

Yongxin Yang, Irene Garcia Morillo, and Timothy M Hospedales. Deep neural decision trees. *arXiv preprint arXiv:1806.06988*, 2018.

Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*, 2023.

Chih-Kuan Yeh, Joon Kim, Ian En-Hsu Yen, and Pradeep K Ravikumar. Representer point selection for explaining deep neural networks. *Advances in neural information processing systems*, 31, 2018.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. *Advances in neural information processing systems*, 31, 2018.

Andy Zeng, Adrian Wong, Stefan Welker, Krzysztof Choromanski, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, et al. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598*, 2022.

Qiangfu Zhao. Evolutionary design of neural network tree-integration of decision tree, neural network and ga. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546)*, volume 1, pp. 240–244. IEEE, 2001.

Jan Ruben Zilke, Eneldo Loza Mencía, and Frederik Janssen. Deepred–rule extraction from deep neural networks. In *Discovery Science: 19th International Conference, DS 2016, Bari, Italy, October 19–21, 2016, Proceedings 19*, pp. 457–473. Springer, 2016.

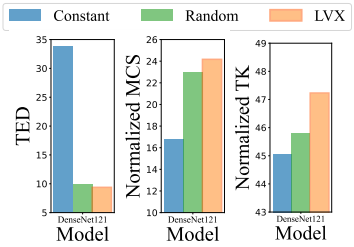


Figure 3: Explanation performance comparison on Chestx-ray14 dataset.