SCHEMA INFERENCE FOR INTERPRETABLE IMAGE CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we study a novel inference paradigm, termed as **schema inference**, that learns to deductively infer the explainable predictions by rebuilding the prior deep neural network (DNN) forwarding scheme, guided by the prevalent philosophical cognitive concept of *schema*. We strive to reformulate the conventional model inference pipeline into a graph matching policy that associates the extracted visual concept of an image with the pre-computed scene impression, by analogy with the human reasoning mechanism via impression matching. To this end, we devise an elaborated architecture, termed as *SchemaNet*, as a dedicated instantiation of the proposed *schema inference* concept, that models both the visual semantics of input instances and the learned abstract imaginations of target categories as topological relational graphs. Meanwhile, to capture and leverage the compositional contributions of visual semantics in a global view, we also introduce a universal *Feat2Graph* scheme in *SchemaNet* to establish the relational graphs that contain abundant interaction information. Both the theoretical analysis and the experimental results on several benchmarks demonstrate that the proposed *schema inference* achieves encouraging performance and meanwhile yields a clear picture of the deductive process leading to the predictions. Our code and model will be made publicly available.

1 INTRODUCTION

> *"Now this representation of a general procedure of the imagination for providing a concept with its image is what I call the schema for this concept* [1]*."*
>
> — *Immanuel Kant*

Deep neural networks (DNNs) have demonstrated the increasingly prevailing capabilities in visual representations as compared to conventional hand-crafted features. Take the visual recognition task as an example. The canonical deep learning (DL) form for image recognition is to yield an effective visual representation with a stack of non-linear layers along with a fully-connected (FC) classifier at the end (He et al., 2016; Dosovitskiy et al., 2021; Tolstikhin et al., 2021), where specifically the inner-product similarities are computed with each category's embedding as the prediction. Despite the success of such a DL scheme, existing deep networks are typically required to simultaneously perceive low-level patterns as well as high-level semantics to make predictions (Zeiler & Fergus, 2014; Krizhevsky et al., 2017). As such, both the procedure of computing visual representations and the learned category-specific embeddings are opaque to humans, leading to challenges in security-matter scenarios, such as autonomous driving and healthcare applications.

In this paper, unlike prior arts that merely obtain the targets in a black-box manner, here we strive to devise an innovative and generalized DNN inference paradigm by reformulating the traditional one-shot forwarding scheme into an interpretable DNN reasoning framework, resembling what occurs in deductive human reasoning. Towards this end, inspired by the **schema** in Kant's philosophy that describes human cognition as the procedure of associating an image of abstract concepts with the specific sense impression, we propose to formulate DNN inference into an interactive matching procedure between the local visual semantics of an input instance and the abstract category

[1] In *Critique of Pure Reason* (A140/B180).

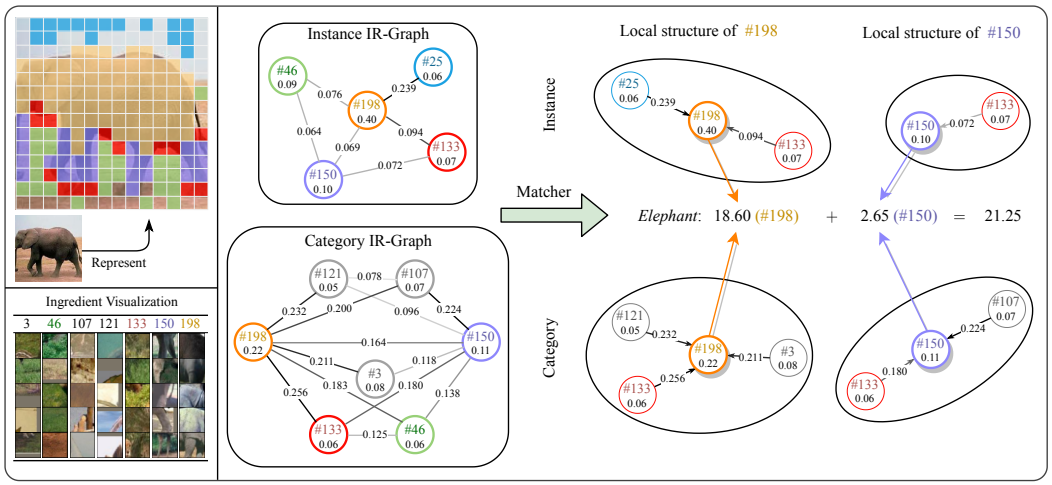


Figure 1: An example shows how an instance IR-Graph is matched to the class imagination. The vertices of IR-Graphs represent visual semantics, and the edges indicate the vertex interactions. The graph matcher capture the similarity between the local structures of joint vertices (*e.g.*, #198 and #150 vertex) by aggregating information from their neighbors as class evidences. The final prediction is defined as the sum of all evidence.

imagination, which is termed as *schema inference* in this paper, leading to the accomplishment of interpretable deductive inference based on visual semantics interactions at the micro-level.

To elaborate the achievement of the proposed concept **schema inference**, we take here image classification, the most basic task in computer vision, as an example to explain our technical details. At a *high level*, the devised *schema* scheme leverages a pre-trained DNN to extract *feature ingredients*, which are, in fact, the semantics represented by a cluster of deep feature vectors from a specific local region in the image domain. Furthermore, the obtained *feature ingredients* are organized into an *ingredient relation grpah* (IR-Graph) for the sake of modeling their interactions that are characterized by the similarity at the semantic-level as well as the adjacency relationship at the spatial-level. We then implement the category-specific imagination as an *ingredient relation atlas* (IR-Atlas) for all target categories induced from observed data samples. As a final step, the *graph similarity* between an instance-level IR-Graph and the category-level IR-Atlas is computed as the measurement for yielding the target predictions. As such, instead of relying on deep features, the desired outputs from *schema* contribute only from the relationship of visual words, as shown in Figure 1.

More specifically, our dedicated *schema*-based architecture, termed as *SchemaNet* in this paper, is based on vision Transformers (ViTs) (Touvron et al., 2021; Dosovitskiy et al., 2021), which is nowadays the most prevalent vision backbone. To effectively obtain the *feature ingredients*, we collect the intermediate features of the backbone from probe data samples clustered by the k-means algorithm. IR-Graph is then established through a customized *Feat2Graph* module that models the discretized array of *ingredients* as the graph vertices, and meanwhile builds the connections, which indicates the ingredient interaction with the self-attention mechanism (Vaswani et al., 2017) of the ViT and the spatial adjacency. Eventually, the *graph similarity* of IR-Graph and IR-Atlas is evaluated via a shallow graph convolutional network (GCN).

Our work relates to several existing methods that mine semantic-rich visual words from a DNN backbone for a self-explanatory model (Brendel & Bethge, 2019; Chen et al., 2019; Nauta et al., 2021; Xue et al., 2022b). In particular, Brendel & Bethge (2019) propose *BagNet* by using a DNN as the visual word extractor that is analogous to traditional bag-of-visual-words (BoVW) representation (Yang et al., 2007) with the SIFT features (Lowe, 2004). Moreover, Chen et al. (2019) develop an elaborated *ProtoPNet* framework that evaluates the similarity scores between image parts and learned class-specific prototypes for inference. Despite their encouraging performance, all these existing methods suffer from the lack of consideration in the compositional contribution of each visual semantics, which is, however, already proved critical for both human and DNN inference (Deng et al., 2022), and are not applicable to schema inference in consequence.

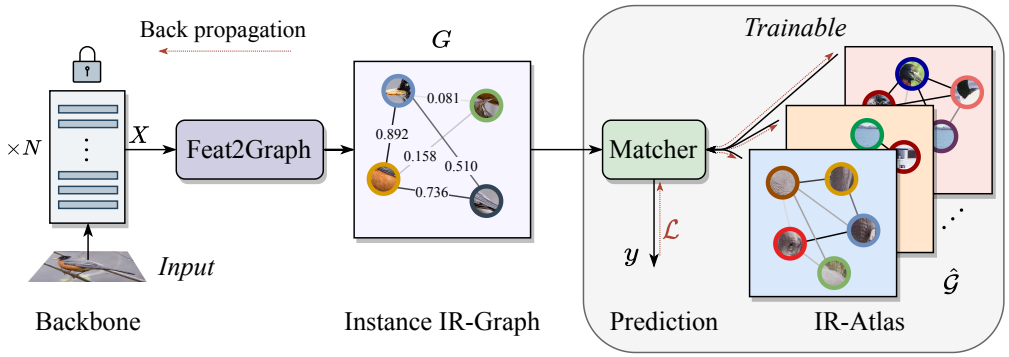


Figure 2: The overall pipeline of our proposed SchemaNet. Firstly, the intermediate feature X extracted from the backbone is fed into the *Feat2Graph* module. The converted IR-Graph is then matched to a set of imaginations, *i.e.*, category-level IR-Atlas induced from observed data, for making the prediction.

In sum, our contribution is an innovative *schema inference* paradigm that reformulates the existing black-box network forwarding into a deductive DNN reasoning procedure that allows for a clear insight into how the class evidence is gathered and deductively leads to the target predictions. This is achieved by building the dedicated IR-Graph and IR-Atlas with the extracted *feature ingredients* and then performing graph matching between IR-Graph and IR-Atlas to derive the desired predictions. We also provide a theoretical analysis to explain the interpretability of our *schema inference* results. Experimental results on CIFAR-10/100 and Caltech-101 demonstrate that the proposed *schema inference* scheme yields results superior to the state-of-the-art on general visual recognition tasks. Further, we demonstrate that by transferring to unseen tasks without fine-tuning the matcher, the learned knowledge of each class is, in fact, stored in IR-Atlas rather than the matcher, thereby exhibiting the high interpretability of the proposed method.

2 SchemaNet

In this section, we introduce our proposed SchemaNet in detail. The overall procedure is illustrated in Figure 2, including a Feat2Graph module that converts deep features to instance IR-Graphs, a learnable IR-Atlas, and a matching network for making predictions. The main idea is to model an input image as an instance-level graph in which nodes are local semantics captured by the DNN backbone and edges represent their interactions. Meanwhile, a category-level graph is maintained as the imagination for each class driven by training samples. Finally, by measuring the similarity between the instance and category graphs, we are able to interpret how predictions are made.

2.1 Preliminary

We first give a brief review of the architecture of ViTs. The simplest implementation of ViTs for image classification is proposed by Dosovitskiy et al. (2021), which treats an image as a sequence of independent 16×16 patches. The embeddings of all patches are then directly fed to a Transformer-based network (Vaswani et al., 2017), a stack of multiple encoder layers with the same structure: a multi-head self-attention mechanism (MHSA) followed by a multilayer perceptron (MLP) with residual connections. Additionally, they append a class token (CLS) to the input sequence for classification alone rather than the average pooled feature. Following this work, Touvron et al. (2021) propose DeiT appending another distillation token (DIST) to learn soft decision targets from the teacher model. This paper will mainly focus on DeiT-based backbones due to the relatively simple yet efficient network architecture.

Let $X \in \mathbb{R}^{(n+\zeta)\times d}$ denote the input feature (a sequence of n visual tokens and ζ auxillary tokens, *e.g.*, CLS and DIST, each of which is a d-dimensional embedding vector) to a MHSA module. The output can be simplified as $\mathrm{MHSA}(X) = \sum_{h=1}^{H} \tilde{\Psi}_h X W_h$, where $\tilde{\Psi}_h \in \mathbb{R}^{(n+\zeta)\times(n+\zeta)}$ denotes the self-attention matrix normalized by row-wise softmax of head $h \in \{1, \ldots, H\}$, and $W_h \in \mathbb{R}^{d\times d}$ is a

learnable projection matrix. As the MLP module processes each token individually, the visual token interactions take place only in the MHSA modules, facilitating the extraction of their relationships from the perspective of ViTs.

For convenience, we define some notations related to the self-attention matrix. Let $\bar{\Psi}$ be the average attention matrix of all heads and Ψ be the symmetric attention matrix: $\Psi = (\bar{\Psi} + \bar{\Psi}^\top)/2$. We partition Ψ into four submatrices:

$$\Psi = \begin{bmatrix} \Psi^* & \Psi^A \\ {\Psi^A}^\top & \Psi^V \end{bmatrix}, \tag{1}$$

where $\Psi^A \in \mathbb{R}^{\zeta \times n}$ is the attention to the auxiliary tokens, $\Psi^V \in \mathbb{R}^{n \times n}$ represents the relationships of visual tokens, and $\Psi^* \in \mathbb{R}^{\zeta \times \zeta}$. Finally, let $\psi^{\text{CLS}} \in \mathbb{R}^n$ represent the attention values to the CLS token extracted from Ψ^A.

2.2 FEAT2GRAPH

Shown in Figure 2, given a pre-trained backbone and an input image, the intermediate feature X is converted to an IR-Graph G for inference by Feat2Graph module, which includes three steps: (1) discretizing X into a sequence of feature ingredients with specific semantics; (2) mapping the ingredients to weighted vertices of IR-Graph; (3) assigning a weighted edge to each vertex pair indicating their interaction. To begin with, we start by defining IR-Graph and IR-Atlas mathematically.

Definition 1 (IR-Graph)**.** *An IR-Graph is an undirected fully-connected graph $G = (E, V)$, in which vertiex set V is the set of feature ingredients for an instance or a specific category, and edge set E encodes the interactions of vertex pairs.*

In particular, to indicate the vertex importance (for instance, "bird head" may contribute to the prediction of bird more than "sky" in human cognition), a non-negative weight $\lambda \in \mathbb{R}_+$ is assigned to each vertex. Let $\Lambda \in \mathbb{R}_+^{|V|} = \{\lambda_i\}_{i=1}^{|V|}$ be the collection of all vertex weights. Besides, the interaction between vertex i and j is quantified by a non-negative weight $e_{i,j} \in \mathbb{R}_+$. With a slight abuse of notation, we define $E \in \mathbb{R}_+^{|V| \times |V|}$ as the weighted adjacency matrix in the following sections.

We define IR-Atlas to represent the imagination of all categories:

Definition 2 (IR-Atlas)**.** *An IR-Atlas $\hat{\mathcal{G}}$ of C classes is a set of category-level IR-Graphs, in which element $\hat{G}_c = (\hat{E}_c, \hat{V}_c)$ for class c has learnable vertex weights $\hat{\Lambda}_c$ and edge weights $\hat{E}_c$.*

Discretization. The main purpose of feature discretization is to mine the most common patterns appearing in the dataset, each of which corresponds to a human understandable semantic, named visual word, such as "car wheel" or "bird head". However, local regions with the same semantic may show differently because of scaling, rotation, or even distortion. In our approach, we utilize the DNN backbone to extract a relatively uniform feature instead of the traditional SIFT feature utilized in (Yang et al., 2007). To be specific, a visual vocabulary $\Omega = \{\omega_i\}_{i=1}^M$ ($\omega_i \in \mathbb{R}^d$) of size M is constructed by k-means clustering running on the collection $\mathbb{X}$ of visual tokens extracted from the probe dataset[2]. Further, a deep feature X (CLS and DIST are removed) is discretized to a sequence of feature ingredients by replacing each element x with the index of the closest visual word

$$\text{Ingredient}(x) = \underset{i \in \{1,\ldots,M\}}{\arg\min} \ \|x - \omega_i\|_2. \tag{2}$$

Strictly, an *ingredient* is referred to as the *index* of visual word ω. With the entire ingredient set $\mathbb{M} = \{1, \ldots, M\}$, the discretized sequence is denoted as $\tilde{X} = (\tilde{x}_1, \ldots, \tilde{x}_n)$, where $\tilde{x}_i \in \mathbb{M}$ is computed from Equation (2).

Feat2Vertex. After we have computed the discretized feature $\tilde{X}$, the vertices of this feature are the unique ingredients: $V = \text{Unique}(\tilde{X})$. The importance of each vertex is measured from two criterions: the contribution to the DNN's final prediction and the appearance statistically. Formally, for vertex $v \in \mathbb{M}$, the importance is defined as

$$\lambda_v = \alpha_1 \lambda_v^{\text{CLS}} + \alpha_2 \lambda_v^{\text{bag}} = \alpha_1 \sum_{i \in \Xi(v|\tilde{X})} \psi_i^{\text{CLS}} + \alpha_2 |\Xi(v|\tilde{X})|, \tag{3}$$

[2]For probe dataset with D instances, the collection $\mathbb{X}$ has $n \times D$ visual tokens (ignoring CLS and DIST).

where $\Xi(v|\tilde{X})$ is the set of all the appeared position of ingredient v in $\tilde{X}$, ψ_i^{CLS} is the attention between the i-th visual token and the CLS token, and $\alpha_{1,2} \geq 0$ are learnable weights balancing the two terms. When $\alpha_1 = 0$, Λ is equivalent to BoVW representation.

Feat2Edge. With the self-attention matrix Ψ^V extracted from Equation (1), the interactions between vertices can be defined and computed with efficiency. For any two different vertices $u, v \in V$, the edge weight is the comprehensive consideration of the similarity in the view of ViT and spatial adjacency. The first term is the average attention between all repeated pairs $\Pi[(u,v)|\tilde{X}]$:

$$e_{u,v}^{\text{attn}} = \frac{1}{\Pi[(u,v)|\tilde{X}]} \sum_{(i,j)\in\Pi[(u,v)|\tilde{X}]} \Psi_{i,j}^V, \tag{4}$$

where $\Pi[(u,v)|\tilde{X}]$ is the Cartesian product of $\Xi(u|\tilde{X})$ and $\Xi(v|\tilde{X})$, and $\Psi_{i,j}^C$ is the attention between the visual tokens at i and j positions. Furthermore, the adjacency is

$$e_{u,v}^{\text{adj}} = \frac{1}{\Pi[(u,v)|\tilde{X}]} \sum_{(i,j)\in\Pi[(u,v)|\tilde{X}]} \frac{1}{\epsilon + \|\text{Pos}(i) - \text{Pos}(j)\|_2}, \tag{5}$$

where function $\text{Pos}(\cdot)$ returns the original 2D coordinates of the input visual token with regard to the patch array after the patch embedding module of ViTs. Eventually, the interaction between vertices u and v is the weighted sum of the two components with learnable $\beta_{1,2}$:

$$e_{u,v} = \beta_1 e_{u,v}^{\text{attn}} + \beta_2 e_{u,v}^{\text{adj}}. \tag{6}$$

Equations (4) and (5) are both invariance when exchanging vertex u and v, so our IR-Graph is equivalent to undirected graph.

It is worth noting that only *semantics* and *their relationships* are preserved in IR-Graphs for further inference rather than deep features.

2.3 Matcher

After converting an input image to IR-Graph, the matcher finds the most similar category-level graph in IR-Atlas. The overall procedure of is shown in Figure 3, which is composed of a GCN module and a similarity computing module (Sim) that generates the final prediction. Detailed settings of the matcher are in Appendix E.2.

For an input IR-Graph $G = (E, V)$, we assign an d_G-dimensional random vector $x \in \mathbb{R}^{d_G}$ to each vertex before feeding the graph to the GCN. All vertex embeddings are i.i.d., sampled from multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, I)$. We adopt GraphConv (Morris et al., 2019) with slight modifications for weighted edges. Let $F \in \mathbb{R}^{|V|\times d_G}$ be the input feature of all vertices to a GraphConv layer, the output is computed as

$$\text{GraphConv}(F) = \text{Norm}\left(\sigma\left((I+E)FW\right)\right), \tag{7}$$

where σ denotes a non-linear activation function, Norm denotes feature normalization, and $W \in \mathbb{R}^{d_G\times d_G}$ is a learnable projection matrix. After passing through total L_G layers of GraphConv, a weighted average pooling (WAP) layer summarizes the vertex embeddings $F^{(L_G)}$ weighted by Λ, yielding the graph representation $z = \Lambda F^{(L_G)}$. Now we have z for G and $\hat{Z} \in \mathbb{R}^{C\times d_G} = (\hat{z}_1, \ldots, \hat{z}_C)$ for all graphs in $\hat{\mathcal{G}}$. By computing the inner-product similarities, the final prediction logits is defined as $y = \hat{Z}z^\top$.

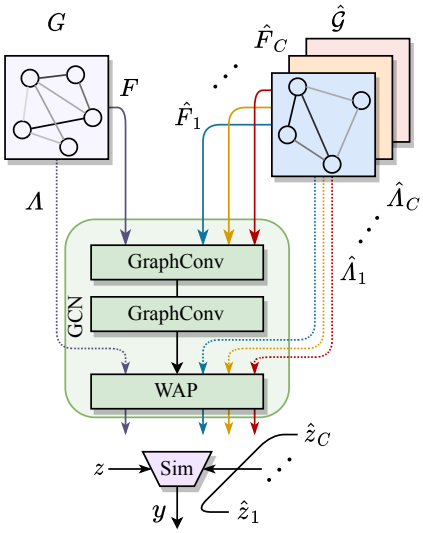


Figure 3: Illustration of the matcher.

Prediction interpretation. Given an instance graph G and a category graph $\hat{G}$, we now analyze how the prediction is made based on their vertices and edges. We start with analyzing the similarity score $s = \hat{z}z^\top$, where $z = \Lambda F^{(L_G)}$ and $\hat{z} = \hat{\Lambda}\hat{F}^{(L_G)}$ are weighted sum of the vertex embeddings

respectively. For the sake of further discussion, let $f_v^{(l)}$ denotes the embedding vector of vertex v output from the l-th GraphConv layer, and $\lambda_v^{(l)}$ denotes its weight. The original vertex embedding of v is defined as $f_v^{(0)}$, which is identical for the same ingredient in any two graphs. Besides, let $\Phi = V \cap \hat{V}$ be the set of shared vertices in G and $\hat{G}$. Now, the computation of s can be expanded as

$$s = \sum_{(u,v)\in\hat{V}\times V} \hat{\lambda}_u \lambda_v \hat{f}_u^{(L_G)} {f_v^{(L_G)}}^\top. \tag{8}$$

Since directly analyzing Equation (8) is hard, we show an approximation and proof it in Appendix B.

Theorem 1. *For a shallow GCN module, Equation (8) can be approximated by*

$$s = \sum_{\phi\in\Phi} \hat{\lambda}_\phi \lambda_\phi \hat{f}_\phi^{(L_G)} {f_\phi^{(L_G)}}^\top. \tag{9}$$

Particularly, if $L_G = 0$ and $\alpha_1 = 0$, our method is equivalent to BoVW with a linear classifier.

Such a result shows: (1) the final prediction score for a category is the summation of all class evidences (vertex ϕ is in $\hat{V}$); (2) as delineated in Corollary 1, the shared neighbors (local structure) connected to the query vertex ϕ also contribute to the final prediction.

2.4 Training SchemaNet

Except for training SchemaNet by cross-entropy loss $\mathcal{L}_{CE}$ between the predictions and ground truths, we further constrain the complexity of IR-Atlas. More formally, the complexity is defined for both edges and vertex weights of all graphs in IR-Atlas as

$$\mathcal{L}_v = \frac{1}{C}\sum_{c=1}^{C} \mathcal{H}(\hat{\Lambda}_c), \quad \mathcal{L}_e = \frac{1}{C|\hat{V}|}\sum_{c=1}^{C}\sum_{u\in\hat{V}} \mathcal{H}(\hat{E}_{c,u}), \tag{10}$$

where function $\mathcal{H}(x)$ computes the entropy of the input vector $x \in \mathbb{R}_+^k$ normalized by its sum of k components, and $\hat{E}_{c,u}$ denotes the weighted edges connected to vertex u in $\hat{G}_c$ of class c. The final optimization goal is

$$\mathcal{L} = \mathcal{L}_{CE} + \gamma_v \mathcal{L}_v + \gamma_e \mathcal{L}_e, \tag{11}$$

where γ_v and γ_e are hyperparameters.

Sparsification. Each graph in IR-Atlas is initialized as a fully-connected graph with random vertex and edge weights. However, this requires $\mathcal{O}(C|V|^2)$ memory space for storage and training the whole set of edges. To alleviate this issue, we initialize IR-Atlas by averaging the instance IR-Graphs for each class and remove the edges connected to vertices whose weights are below a given threshold $\delta_t = 0.01$ from the category-specific graph. Such a procedure will not only dramatically decrease the learnable parameters but also boost the final performance, as shown in Table 2.

The overall training procedure is shown in Appendix C.

3 Experiments

3.1 Implementation

Datasets. We evaluate our method on CIFAR-10/100 (Krizhevsky et al., 2009) and Caltech-101 (Li et al., 2022). Particularly, Caltech-101 has around 9k images, and we manually split it into a training set with 7.4k images and a test set with 1.3k images.

Selection of the hyperparameters. Several hyperparameters are involved in our method, including $\lambda_{v,e}$ in Equation (11) for adjusting the sparsity of IR-Atlas. We set $\lambda_v = 0.5$ and $\lambda_e = 0.75$ as the default value for the following evaluation and the sensitive analyses are presented in Section 3.4. The initial values of learnable $\alpha_{1,2}$ and $\beta_{1,2}$ are set to 0.5, and we show their learning curves in Appendix F.3.

Table 2: Comparison results on CIFAR-10/100 and Caltech-101 with different backbones. We report the top-1 accuracy, number of learnable parameters, and FLOPs. The results in the gray color are merely for listing the accuracy of ViTs rather than for making comparison.

Backbone	Method	CIFAR-10			CIFAR-100			Caltech-101		
		Acc	#param.	FLOPs	Acc	#param.	FLOPs	Acc	#param.	FLOPs
-	BoVW-SIFT	20.20	-	-	-	-	-	-	-	-
Deit-Tiny	Base	96.69	5.53M	1.27G	82.88	5.54M	1.27G	92.53	5.54M	1.27G
	Backbone-FC	94.58	1.93K	1.06G	76.74	19.3K	1.06G	76.71	19.5K	1.06G
	BoVW-Deep	94.95	10.3K	1.06G	75.45	103K	1.06G	60.82	104K	1.06G
	BagNet	70.27	5.73M	1.10G	42.90	5.84M	1.10G	72.54	5.85M	1.10G
	SchemaNet	95.92	396K	1.19G	77.11	105M	2.40G	83.24	106M	2.40G
	SchemaNet-Init	**95.96**	234K	1.06G	**78.45**	573K	1.06G	**87.57**	564K	1.06G
Deit-Small	Base	97.77	21.7M	4.63G	87.65	21.7M	4.63G	94.96	21.7M	4.63G
	Backbone-FC	96.62	3.85K	3.87G	82.44	38.5K	3.87G	86.94	38.9K	3.87G
	BoVW-Deep	95.61	10.3K	3.89G	77.39	103K	3.89G	65.54	104K	3.89G
	BagNet	83.90	22.1M	3.95G	50.49	22.2M	3.95G	78.13	22.2M	3.95G
	SchemaNet	**97.42**	396K	4.00G	82.21	105M	5.21G	84.66	106M	5.21G
	SchemaNet-Init	97.35	235K	3.89G	**82.46**	591K	3.89G	**90.09**	589K	3.89G
Deit-Base	Base	98.41	85.8M	17.6G	89.17	85.9M	17.6G	95.83	85.9M	17.6G
	Backbone-FC	97.04	7.69K	14.7G	**81.66**	76.9K	14.7G	88.20	77.7K	14.7G
	BoVW-Deep	95.50	10.3K	14.7G	72.23	103K	14.7G	66.17	104K	14.7G
	BagNet	90.71	86.6M	14.9G	67.84	86.8M	14.9G	86.55	86.8M	14.9G
	SchemaNet	**97.26**	396K	14.8G	79.26	105M	16.1G	81.12	106M	16.1G
	SchemaNet-Init	97.07	235K	14.7G	79.36	606K	14.7G	**90.72**	593K	14.7G

Visual vocabulary. Thanks to the relatively uniform representation of visual semantics generated by DNNs, our visual vocabulary size M can be set to around one thousand with competitive performance. Table 1 shows the default choice of M on different datasets. Results with different vocabulary sizes are presented in Section 3.3. Furthermore, all the intermediate features are extracted from the 9-th layer of the backbone ViTs, including DeiT-Tiny, DeiT-Small, and Deit-Base.

Table 1: Visual vocabulary size.

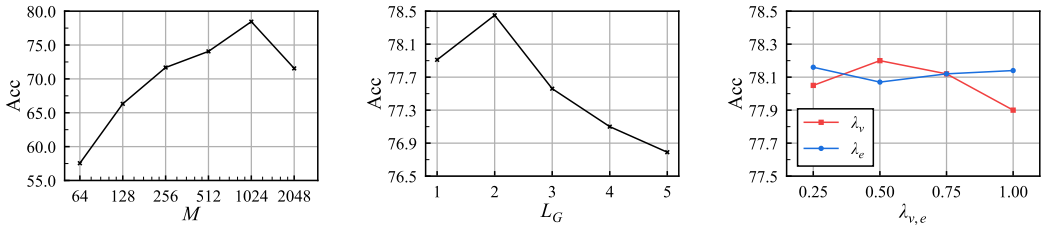
Dataset	Size M
CIFAR-10	128
CIFAR-100	1024
Caltech-101	1024

3.2 Experimental Results

We evaluate our method and comparison baselines with the following settings:

- **BoVW-SIFT:** the traditional BoVW approach that utilizing SIFT feature for constructing the visual vocabulary (Yang et al., 2007).
- **Base:** the base ViT model directly trained on the benchmark datasets with initial weights obtained from the official repository in (Touvron et al., 2021), which is our backbone.
- **Backbone-FC:** the frozen backbone with a FC layer. For the intermediate features extracted from our frozen backbone, we follow the CNN's procedure that the global average pooled embeddings are fed to an FC layer for classification.
- **BoVW-Deep:** the BoVW approach with our extracted visual vocabulary backed ViTs.
- **BagNet:** the implementation of BagNet backed with the ViT, which is constructed following the "Base" setting.
- **SchemaNet:** our proposed SchemaNet without initialization.
- **SchemaNet-Init:** our proposed SchemaNet initialized by the average instance IR-Graphs.

The comparison results are shown in Table 2, where we demonstrate the top-1 accuracy, the number of learnable parameters, as well as the FLOPs for each setting. It is noticeable that the proposed SchemaNet consistently outperforms the baseline methods. Specifically, for the backbone of Deit-Tiny, ours achieves significant improvement (*i.e.*, about 25.7% and 35.3% absolute gain on CIFAR-10 and CIFAR-100, respectively) over BagNet. With a larger backbone such as Deit-Small and Deit-Base, though the performance gap slightly shrinks, our SchemaNet with initializations still yields results superior to BagNet, by an average absolute gain of 12.7%. The reason is that, unlike BagNet that uses



(a) SchemaNet accuracy with different visual vocabulary size M.

(b) SchemaNet accuracy when using GCN with various depth L_G.

(c) Sensitivity analysis of hyperparameters involved in Equation (11).

Figure 4: Ablation study and sensitivity analysis of hyperparameters of our proposed SchemaNet on CIFAR-100 backed with Deit-Tiny.

summation to obtain the similarity map as the BoVW representation (without feature discretization), "BoVW-Deep" is implemented with the discretized visual words. As such, both of the approaches do not consider the visual word interactions, leading to the inferior performance especially when the size of the visual vocabulary expands on CIFAR-100 and Caltech-101.

Moreover, compared with "Backbone-FC" that merely relies on intermediate deep features, the proposed SchemaNet also demonstrates encouraging results, achieving about a 1.4% absolute gain. Furthermore, when initialization and redundant vertices are removed as indicated in Section 3.1, our "SchemaNet-Init", with an even less learnable number of parameters, still leads to a higher accuracy. The proposed method also delivers gratifying performance, on par with the prevalent deep ViT especially on CIFAR-10, but gives a clear insight on the reasoning procedure.

We present more experimental results in Appendix F, including the accuracy performance on mini-ImageNet (Appendix F.1), extending SchemaNet to new categories with fixed graph matcher (Appendix F.2), and visualization the learnable weights (Appendix F.3).

3.3 Ablation Study

Ablation of weight components. The effect of λ^{CLS} and λ^{bag} defined in Equation (3), and e^{attn} and e^{adj} defined in Equation (6) are shown in Table 3. By setting the corresponding weight ($\alpha_{1,2}$ and $\beta_{1,2}$) to zero once at a time, we are able to analyze the components individually. In general, removing any term will lead to varying degrees of performance degradation. More significantly, when removing the term λ^{bag}, the accuracy decreases by 6.03% on CIFAR-100, revealing that the statistics of the ingredients help filter noisy vertices. The learning curves of the parameters are presented in Appendix F.3.

Table 3: Ablation study of the components defining the vertex and edge weights with Deit-Tiny as the backbone DNN.

Settings	Accuracy	
	CIFAR-10	CIFAR-100
Learnable $\alpha_{1,2}$, $\beta_{1,2}$	**95.96**	**78.20**
Fixed $\alpha_1 = 0$, $\alpha_2 = 1$	95.88	75.91
Fixed $\alpha_1 = 1$, $\alpha_2 = 0$	95.68	72.17
Fixed $\beta_1 = 0$, $\beta_2 = 1$	95.85	77.31
Fixed $\beta_1 = 1$, $\beta_2 = 0$	95.89	77.84
Fixed $\alpha_{1,2} = \beta_{1,2} = 0.5$	95.92	75.62

Visual vocabulary size. Figure 4(a) shows SchemaNet accuracy when using different sizes of visual vocabulary. We can observe that even with a relatively small size, *e.g.*, $M = 256$ on CIFAR-100, the performance is still competitive (about 7% absolute degradation). Besides, for the case of $M = 2048$, however, the performance suffers from low generalizability of the vocabulary.

Number of GraphConv layers. The effect of using different depths of GCN is illustrated in Figure 4(b). As L_G increases to 5, the accuracy on CIFAR-100 decreases rapidly, yielding that deep GCN matcher impairs the graph matching performance, which has been explored due to the over-smoothing caused by many convolutional layers (Zhao & Akoglu, 2020; Li et al., 2018). In our method, however, the graph matcher with only two layers of GraphConv is proven to be adequate for high performance and interpretability.

3.4 Sensitivity Analysis of Hyperparameters.

Figure 4(c) shows the sensitivity analysis of λ_v and λ_e in Equation (11) that constrains the complexity of IR-Atlas. We evaluate the top-1 accuracy performance on the CIFAR-100 dataset with Deit-Tiny

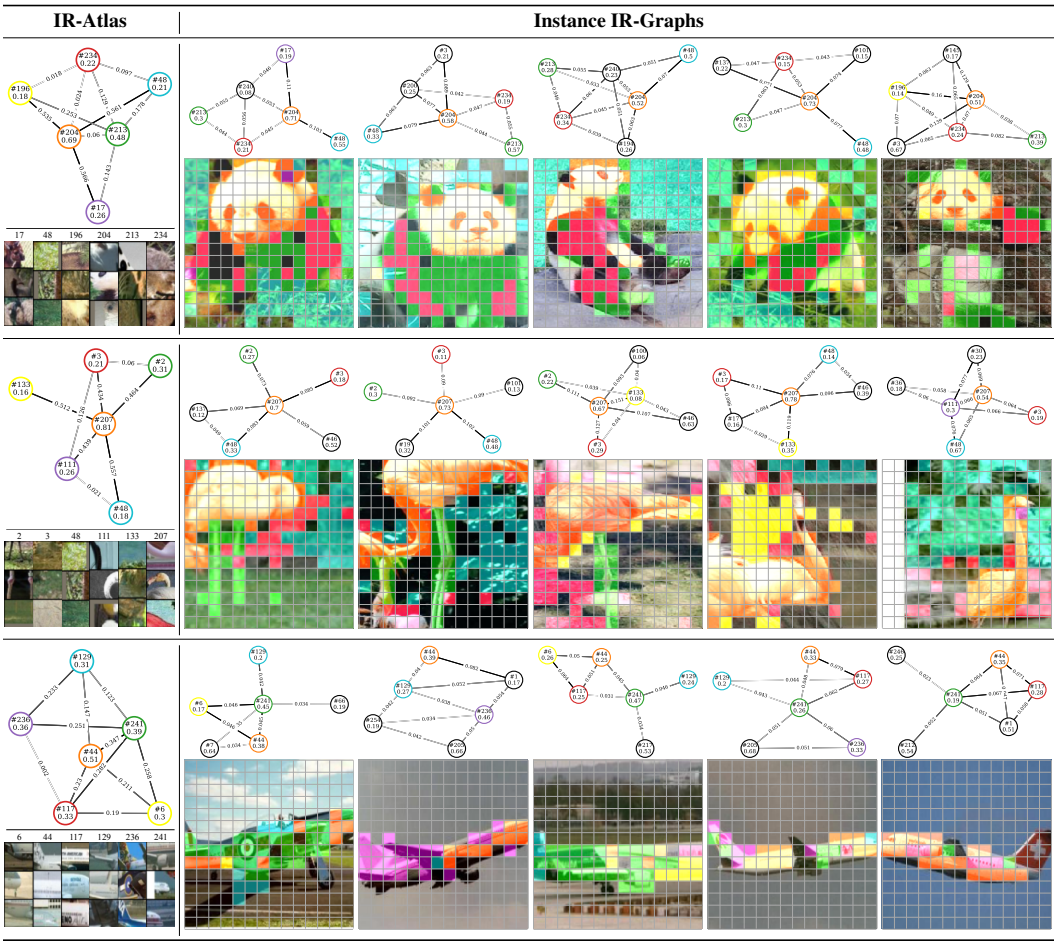


Figure 5: Examples of the learned IR-Atlas and instance IR-Graphs randomly sampled from three categories ("panda", "flamingo", and "airplanes") on Caltech-101. Please zoom in for a better view.

as the backbone. The curve of λ_v shows that neither dense nor extreme sparse IR-Atlas impairs the performance, while our method is more robust when changing the value of λ_e.

3.5 Visualization

In Figure 5, we show the examples of IR-Graph and learned IR-Atlas backed with Deit-Tiny on the Caltech-101 dataset. More examples are presented in Appendix G.

4 Conclusion and Outlook

In this paper, we propose a novel inference paradigm, named *schema inference*, guided by Kant's philosophy towards resembling human deductive reasoning of associating the abstract concept image with the specific sense impression. To this end, we reformulate the traditional DNN inference into a graph matching scheme by evaluating the similarity between instance-level IR-Graph and category-level imagination in a deductive manner. Specifically, the graph vertices are visual semantics represented by common feature vectors from DNN's intermediate layer. Besides, the edges indicate the vertex interactions characterized by semantic similarity and spatial adjacency, which facilitate capturing the compositional contributions to the predictions. Theoretical analysis and experimental results on several benchmarks demonstrate the superiority and interpretability of schema inference. In future work, we will implement schema inference to more complicated vision tasks, such as visual question answering, that enables linking the visual semantics to the phrases in human language, achieving a more powerful yet interpretable reasoning paradigm.

REFERENCES

Bag of visual words (bovw) with deep features - patch classification model for limited dataset of breast tumours, 2022. URL https://europepmc.org/article/PPR/PPR507364.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Wieland Brendel and Matthias Bethge. Approximating CNNs with bag-of-local-features models works surprisingly well on imagenet. In *International Conference on Learning Representations*, 2019.

Xiaojun Chang, Pengzhen Ren, Pengfei Xu, Zhihui Li, Xiaojiang Chen, and Alexander G Hauptmann. A comprehensive survey of scene graphs: Generation and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 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.

Huiqi Deng, Qihan Ren, Hao Zhang, and Quanshi Zhang. Discovering and explaining the representation bottleneck of dnns. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=iRCUlgmdfHJ.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Yunhao Ge, Yao Xiao, Zhi Xu, Meng Zheng, Srikrishna Karanam, Terrence Chen, Laurent Itti, and Ziyan Wu. A peek into the reasoning of neural networks: Interpreting with structural visual concepts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2195–2204, 2021.

Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Learning representations by predicting bags of visual words. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6928–6938, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Roei Herzig, Moshiko Raboh, Gal Chechik, Jonathan Berant, and Amir Globerson. Mapping images to scene graphs with permutation-invariant structured prediction. *Advances in Neural Information Processing Systems*, 31, 2018.

Adrian Hoffmann, Claudio Fanconi, Rahul Rade, and Jonas Kohler. This looks like that... does it? shortcomings of latent space prototype interpretability in deep networks. *CoRR*, abs/2105.02968, 2021. URL https://arxiv.org/abs/2105.02968.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International Conference on Machine Learning*, pp. 5338–5348. PMLR, 2020.

Yael Konforti, Alon Shpigler, Boaz Lerner, and Aharon Bar-Hillel. Inference graphs for cnn interpretation. In *European Conference on Computer Vision*, pp. 69–84. Springer, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Fei-Fei Li, Marco Andreeto, Marc'Aurelio Ranzato, and Pietro Perona. Caltech 101, 2022. URL https://data.caltech.edu/records/20086.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In Sheila A. McIlraith and Kilian Q. Weinberger (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pp. 3538–3545, 2018.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. *Advances in neural information processing systems*, 29, 2016.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 4602–4609, 2019.

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 (CVPR)*, pp. 14933–14943, June 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

MDJ Peters. Extending explainability of generative classifiers with prototypical parts. Master's thesis, University of Twente, 2022.

Dawid Rymarczyk, Aneta Kaczyńska, Jarosław Kraus, Adam Pardyl, and Bartosz Zieliński. Protomil: Multiple instance learning with prototypical parts for fine-grained interpretability. *arXiv preprint arXiv:2108.10612*, 2021.

Suprosanna Shit, Rajat Koner, Bastian Wittmann, Johannes Paetzold, Ivan Ezhov, Hongwei Li, Jiazhen Pan, Sahand Sharifzadeh, Georgios Kaissis, Volker Tresp, et al. Relationformer: A unified framework for image-to-graph generation. *arXiv preprint arXiv:2203.10202*, 2022.

Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems*, 34: 24261–24272, 2021.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pp. 10347–10357. PMLR, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

Lauren J Wong and Sean McPherson. Explainable neural network-based modulation classification via concept bottleneck models. In *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0191–0196. IEEE, 2021.

Mengqi Xue, Qihan Huang, Haofei Zhang, Lechao Cheng, Jie Song, Minghui Wu, and Mingli Song. Protopformer: Concentrating on prototypical parts in vision transformers for interpretable image recognition. *arXiv preprint arXiv:2208.10431*, 2022a.

Mengqi Xue, Haofei Zhang, Qihan Huang, Jie Song, and Mingli Song. Learn decision trees with deep visual primitives. *Available at SSRN 4210199*, 2022b.

Jun Yang, Yu-Gang Jiang, Alexander G Hauptmann, and Chong-Wah Ngo. Evaluating bag-of-visual-words representations in scene classification. In *Proceedings of the international workshop on Workshop on multimedia information retrieval*, pp. 197–206, 2007.

Alireza Zareian, Zhecan Wang, Haoxuan You, and Shih-Fu Chang. Learning visual commonsense for robust scene graph generation. In *European Conference on Computer Vision*, pp. 642–657. Springer, 2020.

Mateo Espinosa Zarlenga, Pietro Barbiero, Gabriele Ciravegna, Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, Zohreh Shams, Frederic Precioso, Stefano Melacci, Adrian Weller, et al. Concept embedding models. *arXiv preprint arXiv:2209.09056*, 2022.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Quanshi Zhang, Xin Wang, Ruiming Cao, Ying Nian Wu, Feng Shi, and Song-Chun Zhu. Extraction of an explanatory graph to interpret a cnn. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3863–3877, 2020a.

Quanshi Zhang, Xin Wang, Ying Nian Wu, Huilin Zhou, and Song-Chun Zhu. Interpretable cnns for object classification. *IEEE transactions on pattern analysis and machine intelligence*, 43(10): 3416–3431, 2020b.

Zaixi Zhang, Qi Liu, Hao Wang, Chengqiang Lu, and Cheekong Lee. Protgnn: Towards self-explaining graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 9127–9135, 2022.

Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. In *International Conference on Learning Representations, ICLR*, 2020.

Yiwu Zhong, Jing Shi, Jianwei Yang, Chenliang Xu, and Yin Li. Learning to generate scene graph from natural language supervision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1823–1834, 2021.

A ADDITIONAL RELATED WORK

In this section, we introduce literature that is related to our method.

A.1 VISUAL CONCEPT ANALYSIS

Visual concept analysis aims to interpret the behavior of pre-trained deep models by extracting and tracking intuitive visual concepts during the DNN's inference procedure. Previous works use masks (Zhang et al., 2020b), explanatory graphs (Zhang et al., 2020a) and probabilistic models (Konforti et al., 2020) to interpret the internal layers of convolutional neural networks (CNNs). Further, Deng et al. (2022) propose to capture and analyze the interaction of visual concepts contributing cooperatively to the prediction results for CNNs. However, they use Shapley values to compute the interaction, which is computationally expensive. Consequently, we adopt vision Transformers (ViTs) (Dosovitskiy et al., 2021; Touvron et al., 2021) as the DNN backbone of our method instead of conventional CNNs thanks to the multi-head self-attention (MHSA) mechanism (Vaswani et al., 2017) that explicitly encodes the interactions of visual tokens in the similarity level.

More recently, VRX (Ge et al., 2021) is proposed to use a graphical model to interpret the prediction of a pre-trained DNN, in which edges are constructed to represent the spatial relation of visual semantics. They further utilize a GCN module to make a prediction with the input structural concept graphs by training to mimic DNN's prediction. Nevertheless, their approach requires training a graph for every input sample, which is extremely expensive for applying to large-scale datasets. Besides, VRX does not construct the class imagination that carries the category-specific knowledge, meaning that the GCN predictor must learn to memorize and distinguish all the class-specific graphs, which impairs the overall interpretability. Our proposed schema inference, however, explicitly creates an interactive matching procedure between the instances and category imagination. Moreover, our GNN model only captures the local structure for gathering class evidence, which is easier to accomplish.

A.2 VISUAL CONCEPT LEARNING

Visual concept learning refers to a range of methods that mine sematic-rich visual words from a DNN, which are further utilized to make interpretable predictions toward a self-explanatory model in reality. Concept bottleneck models (CBMs) (Koh et al., 2020; Zarlenga et al., 2022; Deng et al., 2022; Wong & McPherson, 2021) link the neurons with human-interpretable semantics, *i.e.*, visual words, explicitly encouraging the trustworthiness of DNNs. Human interventions of learned bottleneck layers can fix the misclassified concepts to improve the model performance. Another line of visual concept learning, part-prototype-based methods, collectively makes predictions on target tasks with DNN and semantic-rich prototypes, which can be divided into two schools according to their prototypes. (1) Bag-of-Visual-Word (BoVW) approaches (Brendel & Bethge, 2019; Gidaris et al., 2020; PPR, 2022) obtain prototypes with semantics through passively clustering the hand-crafted features (Yang et al., 2007) or deep features into a set of discrete embeddings (visual words) that relate to specific visual semantics. The prediction procedure of these approaches is analog to the BoW model in natural language processing (NLP), in that an interpretable representation is constructed based on the statistics of the occurrence of the visual words, and then fed to an interpretable classifier, such as the linear classifier or decision trees. (2) In contrast, part-prototypical networks (ProtoPNets) (Chen et al., 2019) and the following works (Nauta et al., 2021; Xue et al., 2022a; Zhang et al., 2022; Peters, 2022; Rymarczyk et al., 2021) jointly train DNN (as the feature extractor) and parameterized prototypes. Then decisions are made based on the linear combination of similarity scores between prototypes and feature vectors at all the spatial locations. Despite the relatively high performance, as discussed in (Brendel & Bethge, 2019; Hoffmann et al., 2021), the similarity of the learned prototypes and deep features in the embedding space may be significantly different in the input space. As such, we only compare those approaches in which prototypes are generated based on clustering for the same level of interpretability. To the best of our knowledge, none of the existing works have explored a deductive inference paradigm based on the interaction of visual semantics.

A.3 SCENE GRAPH GENERATION

A scene graph can be generated from an input image to excavate the collection of objects, attributes, and relationships in a whole scene. Typical graph-based representations and learning

algorithms (Herzig et al., 2018; Chang et al., 2021; Shit et al., 2022; Zhong et al., 2021; Zareian et al., 2020) adopt graph neural networks (GNNs) (Kipf & Welling, 2016) to model the relationships between objects in a scene. Similar to scene graphs, our proposed SchemaNet also utilizes GNNs to represent the relationships between ingredients related to local semantics. However, the nodes in GNNs of SchemaNet represent more fine-grained representations, *i.e.*, local semantics of objects, compared to those relating to the whole objects in scene graphs. Moreover, in SchemaNet, the trained GNN is responsible for estimating the similarity between an instance IR-Graph and category IR-Graphs and making classification based on the similarity scores.

B Proof of Theorem 1

We first restate the theorem:

Theorem (Theorem 1 restated)**.** *For a shallow GCN module, Equation (8) can be approximated by*

$$s = \sum_{\phi \in \Phi} \hat{\lambda}_\phi \lambda_\phi \hat{f}_\phi^{(L_G)} f_\phi^{(L_G)\top}. \tag{12}$$

Particularly, if $L_G = 0$ and $\alpha_1 = 0$, our method is equivalent to BoVW with a linear classifier.

For further discussion, we first prove the following lemma:

Lemma 1. *For random vectors f, $g \in \mathbb{R}^{d_G}$ drawn **independently** from multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, I)$, we have*

$$\begin{aligned} \mathbb{E}_{f,g}\left[fW^\top W g^\top\right] &= 0 \\ \mathbb{E}_f\left[fW^\top W f^\top\right] &= \|W\|_F^2, \end{aligned} \tag{13}$$

where $\|\cdot\|_F$ is the matrix Frobenius norm, and $W \in \mathbb{R}^{d_G \times d_G}$ is a projection matrix.

Proof. To begin with, we expand the term $fW^\top W g^\top$ as

$$fW^\top W g^\top = \sum_{r=1}^{d_G}\sum_{s=1}^{d_G}\sum_{t=1}^{d_G} f_s g_t w_{r,s} w_{r,t}. \tag{14}$$

Therefore, the expectation

$$\mathbb{E}_{f,g}\left[fW^\top W g^\top\right] = \sum_{r=1}^{d_G}\sum_{s=1}^{d_G}\sum_{t=1}^{d_G} w_{r,s} w_{r,t} \mathbb{E}_{f_s}\left[f_s\right] \mathbb{E}_{g_t}\left[g_t\right] = 0, \tag{15}$$

while

$$\begin{aligned} \mathbb{E}_f\left[fW^\top W f^\top\right] &= \sum_{r=1}^{d_G} \mathbb{E}_f\left[\sum_{s=1}^{d_G}\sum_{t=1}^{d_G} w_{r,s} w_{r,t} f_s f_t\right] \\ &= \sum_{r=1}^{d_G} \mathbb{E}_f\left[\sum_{s=1}^{d_G} w_{r,s}^2 f_s^2\right] \\ &= \sum_{r=1}^{d_G}\sum_{s=1}^{d_G} w_{r,s}^2 = \|W\|_F^2. \end{aligned} \tag{16}$$

□

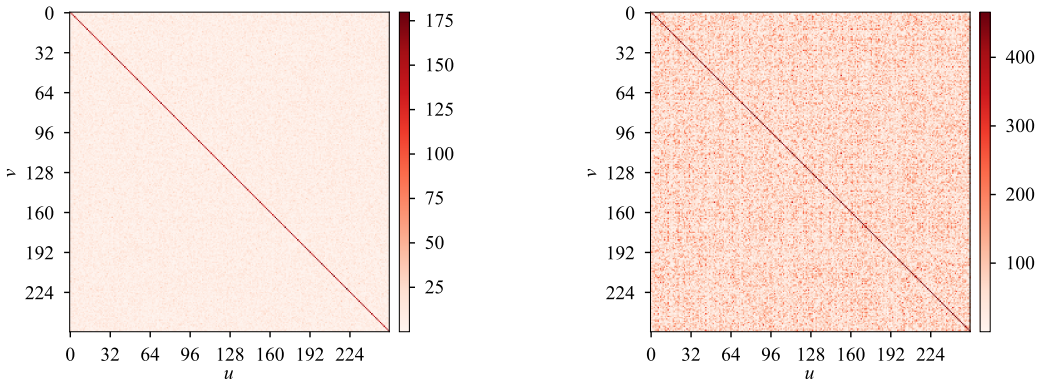
Particularly, if W is the identity matrix

$$\mathbb{E}_f\left[fW^\top W f^\top\right] = d_G. \tag{17}$$

Further, if the components of matrix W are drawn i.i.d. from Gaussian distribution $\mathcal{N}(0, 1)$ and are independent with f,

$$\mathbb{E}_{f,W}\left[fW^\top W f^\top\right] = d_G^2. \tag{18}$$

Now we start to proof Theorem 1.



(a) Raw vertex embeddings.

(b) Last layer embeddings.

Figure 6: Vertex embedding similarities between a vertex v in the instance IR-Graph and a vertex u in the category IR-Atlas, which are extracted from the original vertex embeddings and output from the last GraphConv layer. We show the average similarity over 1000 samples on the Caltech-101 dataset with a visual vocabulary size of 256.

Proof. We start with a simple case that the depth of GCN module is zero.

As G and $\hat{G}$ share the same original vertex embeddings, we have

$$s = \sum_{(u,v)\in\hat{V}\times V} \hat{\lambda_u}\lambda_v \hat{f}_u^{(0)} {f_v^{(0)}}^\top = \sum_{(u,v)\in\hat{V}\times V} \hat{\lambda_u}\lambda_v f_u^{(0)} {f_v^{(0)}}^\top. \tag{19}$$

According to Lemma 1, the expectation of the same vertex term $\mathbb{E}\left[ff^\top\right] = d_G$ is more significant than different vertices terms $\hat{\lambda_u}\lambda_v f_u^{(0)} {f_v^{(0)}}^\top$ for $u \neq v$ when G and $\hat{G}$ are constrained with sparsity. In Figure 6(a), we visualize the vertex similarities $f_u^{(0)} {f_v^{(0)}}^\top$, showing that the similarities between different vertices are significantly lower than that of the same vertex.

Consequently, Equation (8) in the case of $L_G = 0$ can be simplified as

$$s = \sum_{\phi\in\Phi} \hat{\lambda}_\phi \lambda_\phi \|f_\phi^{(0)}\|_2^2, \tag{20}$$

where $\Phi = V \cap \hat{V}$ is the set of shared vertices of G and $\hat{G}$.

With an extra assumption that $\alpha_1 = 0$, λ_ϕ therefore represents the count of visual word ϕ. Combing the learnable term $\hat{\lambda}_\phi \|f_\phi^{(0)}\|_2^2$ as w_ϕ, the prediction of the matcher is

$$y = \Lambda W_{\text{FC}}, \tag{21}$$

where $W_{\text{FC}} \in \mathbb{R}^{|V|\times C}$ is the learnable projection matrix of a linear classifier.

Now, we prove the whole theorem. Firstly, for different vertices $u \in \hat{V}$ and $v \in V$, the summation term

$$s_{u,v} = \hat{\lambda}_u \lambda_v \hat{f}_u^{(L_G)} {f_v^{(L_G)}}^\top \tag{22}$$

can be rewritten as the aggregation form with feature output from the $(L_G - 1)$-th layer:

$$s_{u,v} = \hat{\lambda}_u \lambda_v \sum_{(i,j)\in\dot{\mathcal{N}}_{\hat{G}}(u)\times\dot{\mathcal{N}}_G(v)} \hat{e}_{u,i} e_{v,j} \hat{f}_i^{(L_G-1)} W^{(L_G)} {W^{(L_G)}}^\top {f_j^{(L_G-1)}}^\top, \tag{23}$$

where $\dot{\mathcal{N}}_{\hat{G}}(u)$ represent the neighbors of vertex u in Graph $\hat{G}$ and with u itself, and $e_{u,u} = 1$ as the self loop.

With the assumption that $\hat{f}_i^{(L_G-1)}$ and $f_j^{(L_G-1)}$ are whitened random vectors (we say a random vector $x \in \mathbb{R}^d$ is whitened if all the components are zero mean and independent from each other, which can be achieved by normalization), the summation term $\hat{f}_i^{(L_G-1)} W^{(L_G)} W^{(L_G)\top} f_j^{(L_G-1)\top}$ with vertex $i = j$ will be significantly larger than those with different vertices. We further illustrate the vertex similarities output from the final layer in Figure 6(b), showing that $\hat{f}_u^{(L_G)} f_v^{(L_G)\top}$ is conspicuous for vertices $u = v$.

Corollary 1. *Consequently, $s_{u,v}$ is significant if $\dot{\mathcal{N}}_{\hat{G}}(u)$ and $\dot{\mathcal{N}}_G(v)$ have shared vertices, relative large edge weight product $\hat{e}_{u,i} e_{v,j}$, and vertex weight product $\hat{\lambda}_u \lambda_v$, which means u and v have similar **local structure**, particularly in the case that u and v are the same vertex.*

In conclusion, Equation (8) can be simplified as the summation of the joint vertices $\Phi = \hat{V} \cap V$ of the instance and category graph G and $\hat{G}$. □

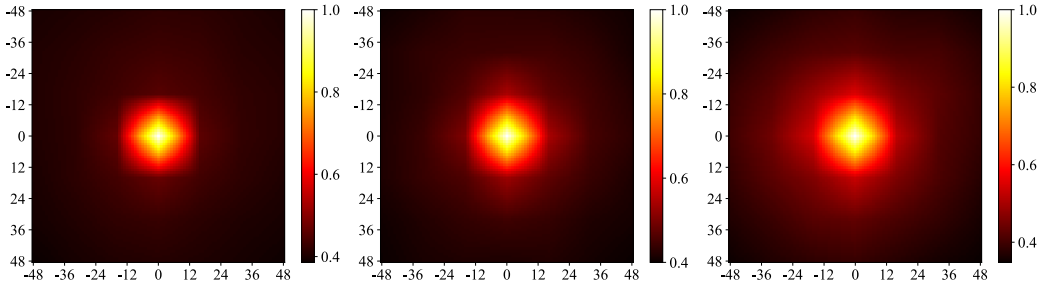
C Training Algorithm

The training algorithm of our proposed SchemaNet is shown in Algorithm 1 with initializing IR-Atlas and sparsification.

Algorithm 1 SchemaNet optimizer with initialization and sparsification.

Input: $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{D}$: the training dataset with D samples; $\hat{\mathcal{G}} = \{\hat{G}_c\}_{c=1}^{C}$: initial IR-Atlas; $\mathrm{Backbone}(\cdot)$: the ViT backbone; $\mathrm{Matcher}(\cdot,\cdot)$: the graph matcher; Ω: the visual vocabulary; Θ: the set of all trainable parameters; δ_t: the sparsification threshold.

procedure INITIALIZATION($\mathcal{D}$, Ω)
  Sample a subset $\tilde{\mathcal{D}} \subset \mathcal{D}$ as the probe dataset.
  $S_1, \ldots, S_C \leftarrow \varnothing$, $\hat{\mathcal{G}} \leftarrow \varnothing$
  for $(x, y) \in \tilde{\mathcal{D}}$ **do**
    $G \leftarrow$ FEAT2GRAPH($\mathrm{Backbone}(x)$, Ω)
    $S_y \leftarrow S_y \cup G$
  end for
  for $c = 1, \ldots, C$ **do** ▷ compute average IR-Graph from each category
    $\hat{\mathcal{G}} \leftarrow \hat{\mathcal{G}} \cup$ AVERAGE(S_c)
  end for
  return $\hat{\mathcal{G}}$
end procedure
procedure TRAINING($\hat{\mathcal{G}}$, $\mathcal{D}$, Ω)
  for $(x, y) \in \mathcal{D}$ **do**
    for $\hat{e}_{i,j} \in \hat{E}$ **do** ▷ Removing redundant edges
      if $\hat{\lambda}_i < \delta_t$ or $\hat{\lambda}_j < \delta_t$ **then**
        $\hat{e}_{i,j} \leftarrow$ NIL
      end if
    end for
    $G \leftarrow$ FEAT2GRAPH($\mathrm{Backbone}(x)$, Ω)
    $\hat{y} \leftarrow \mathrm{Matcher}(G, \hat{\mathcal{G}})$
    Compute the final loss and gradient ∇_Θ *w.r.t.* parameters Θ.
    Update parameters Θ with AdamW optimizer.
  end for
end procedure
$\hat{\mathcal{G}} \leftarrow$ INITIALIZATION($\mathcal{D}$, Ω)
TRAINING($\hat{\mathcal{G}}$, $\mathcal{D}$, Ω)



(a) DeiT-Tiny (b) DeiT-Small (c) DeiT-Base

Figure 7: The heat map of the receptive field of an anchor token over its neighborhood region. Averaged visualization results of 64 random images of three ViTs (DeiT-Ti, DeiT-S, and DeiT-B) from Caltech-101 dataset are demonstrated. The full image size is 224×224, and we crop a small region with a size of 96×96 with the anchor token as the center for better visualization.

D EFFECTIVE RECEPTIVE FIELD OF VITS

In this section, we discuss the effective receptive field (ERF) of ViTs, which is crucial as the visual token in the intermediate layers of ViTs may relate to other positions due to MHSA, affecting the interpretability of the ingredients (visual word semantics). Although Luo et al. (2016) have proposed the method to measure the ERF for CNNs, and it cannot be directly implemented to Transformer-base models. In this section, we propose a relatively simple yet effective approach to measuring the ERF of ViTs.

Definition 3 (Transformer ERF)**.** *Supposed that visual sequence $X \in \mathbb{R}^{n \times d}$ is fed into a ViT backbone with N layers, x_* is a randomly chosen anchor token in X, and $\varepsilon \in \mathbb{R}^d$ is the random vector drawn from Gaussian distribution $\mathcal{N}(0, I)$ normalizing to the unit vector. The Transformer ERF is the average Euclidean distance between the closest token y to x_* in the 2D token array and x_* so that when y is disturbed to $\hat{y} = y + \epsilon$ the output change of token x_* is less than a predefined threshold $\delta_r > 0$.*

Figure 7 shows the visualization heatmap *w.r.t.* the change of the output anchor with different ViT backbones adopted in our method. As we can observe, for an input image size of 224×224, the output tokens with relatively significant changes are mainly distributed in the circle with a radius of 25. Therefore, in our method, all the ingredients are in charge of 50×50 patches corresponding to the input image.

E EXPERIMENTAL DETAILS

E.1 IR-GRAPH

Before an instance-level graph, G and a category-level graph $\hat{G}$ are fed to the graph matcher, and their vertex and edge weights are normalized as follows. The vertex weights in Λ are divided by the sum of Λ

$$\Lambda_{\text{norm}} = \frac{\Lambda}{\sum_{i=1}^{|V|} \lambda_i}. \tag{24}$$

The edge matrix E is divided by the row-wise summation:

$$E_{\text{norm}} = \begin{bmatrix} \frac{E_{1,:}}{\sum_{i=1}^{|V|} e_{1,i}} \\ \vdots \\ \frac{E_{|V|,:}}{\sum_{i=1}^{|V|} e_{|V|,i}} \end{bmatrix}. \tag{25}$$

For making the edge matrix symmetric, the edge matrix is defined as

$$E_{\text{sym}} = \frac{1}{2}(E_{\text{norm}} + E_{\text{norm}}^{\top}) \tag{26}$$

E.2 GCN SETTINGS

Now we describe the detailed settings of the GCN module in the graph matcher. We adopt LayerNorm Ba et al. (2016) as the $\mathrm{Norm}(\cdot)$ function and rectified linear unit (ReLU) as the activation in Equation (7). Besides, the embedding dimension d_G is set to 256 for all the experiments.

E.3 TRAINING DETAILS.

The matcher and IR-Atlas in our method are optimized by AdamW (Loshchilov & Hutter, 2019) with a learning rate of 10^{-3}, weight decay of 5×10^{-4}, and cosine annealing as the learning rate decay schedule. We implement our method with Pytorch (Paszke et al., 2019) and train all the settings for 50 epochs with a batch size of 64 on one NVIDIA Tesla A100 GPU. All input images are resized to 224×224 pixels before feeding to our SchemaNet.

F ADDITIONAL RESULTS

This section contains additional experimental results, highlighting the extendability of our proposed schema inference.

F.1 RESULTS ON MINI-IMAGENET

We further implement SchemaNet on mini-ImageNet (Vinyals et al., 2016), a subset of the ImageNet (Deng et al., 2009) dataset with 50k training images and 10k testing images, evenly distributed over 100 classes. We adopt DeiT-Small as the backbone and the visual vocabulary size $M = 1024$. The results are presented in Table 4, drawing the consistent conclusion as the main results in Table 2.

Table 4: Comparison results on mini-ImageNet.

	Base	Backbone-FC	BoVW-Deep	BagNet	SchemaNet	SchemaNet-Init
Acc	95.35	90.04	<u>90.24</u>	85.64	89.40	**91.02**

F.2 EXTENDABILITY

We here evaluate the extendability of our method, which is evaluated by extending to unseen tasks while keeping the original framework frozen. As such, only the category-level IR-Graphs of the new tasks are trained and inserted into the original IR-Atlas.

Specifically, the tasks are disjointly drawn from Caltech-101 dataset. The "Base" task has 21 classes, and task 1 to 4 has 20 classes. The backbone model, *i.e.*, DeiT-Tiny, is trained on the "Base" task and then is kept unchanged for the new tasks.

The experimental results are presented in Table 5, revealing limited performance degradation for incoming new tasks. Such a result can be explained that the graph matcher is only responsible for evaluating the graph similarity merely. At the same time, the category knowledge is stored in the imagination, *i.e.*, IR-Atlas.

Table 5: Top-1 accuracy results when extending SchemaNet to unseen tasks. The accuracies are evaluated on the corresponding task, and the average accuracy over all the tasks is given as well.

	Average	**Base**				
Base	97.57	97.57	**Task 1**			
Task 1	94.68	95.95	93.69	**Task 2**		
Task 2	93.36	95.14	93.06	92.11	**Task 3**	
Task 3	93.63	95.55	93.06	92.47	93.75	**Task 4**
Task 4	92.37	95.14	92.74	91.76	90.42	91.49

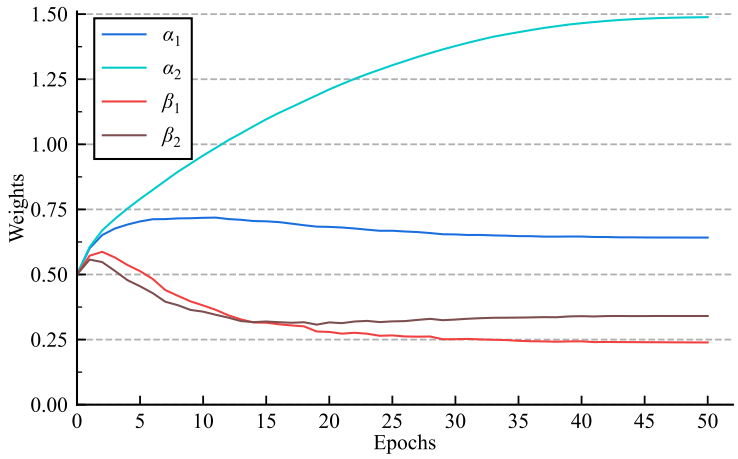


Figure 8: Learning curves of $\alpha_{1,2}$ and $\beta_{1,2}$ with DeiT-Tiny backbone on Caltech-101 dataset.

F.3 Learning Curves

The learning curves of learnable weights $\alpha_{1,2}$ in Equation (3) and $\beta_{1,2}$ in Equation (6) are shown in Figure 8, from which we can observe that the model tend to adopt a relatively larger $alpha_2$ (for visual word count) while keep the λ_v^{CLS} term for filtering noisy ingredients such as background ingredients.

F.4 Feat2Graph Evaluation

We here present a performance bottleneck of our method, *i.e.*, the *Feat2Graph* module, in which a feature ingredient array is converted to IR-Graph without benefiting from GPU parallel acceleration. Specifically, as duplicated visual words may exist after the feature discretization, which happens from time to time, we implement this module in C++ to achieve better random access performance. The experiments are conducted on one NVIDIA Tesla A100 GPU platform with AMD EPYC 7742 64-Core Processor.

In Table 6, we show the average running time of each component with an input batch size of 64. We can see that without acceleration in parallel, the running time for "Feat2Edge" is significantly longer than others (about 1.75 ms per image). However, the inference time is still acceptable for real-time applications.

Table 6: Time costing (ms) of the components in SchemaNet with input batch size of 64.

Dataset	Backbone	Feat2Vertex	Feat2Edge	Matcher
CIFAR-10	4.89	7.00	19.8	1.95
CIFAR-100	4.31	9.50	101.7	1.30
Caltech-101	4.12	8.12	111.7	1.42

G More Visualizations

We further provide more visualization examples on Caltech-101 dataset, shown in Figures 9 to 11.

Figure 9: Examples of the learned IR-Atlas and instance IR-Graphs randomly sampled from five categories (“bass”, “bonsai”, “emu”, “euphonium”, and “grand piano”) on Caltech-101.

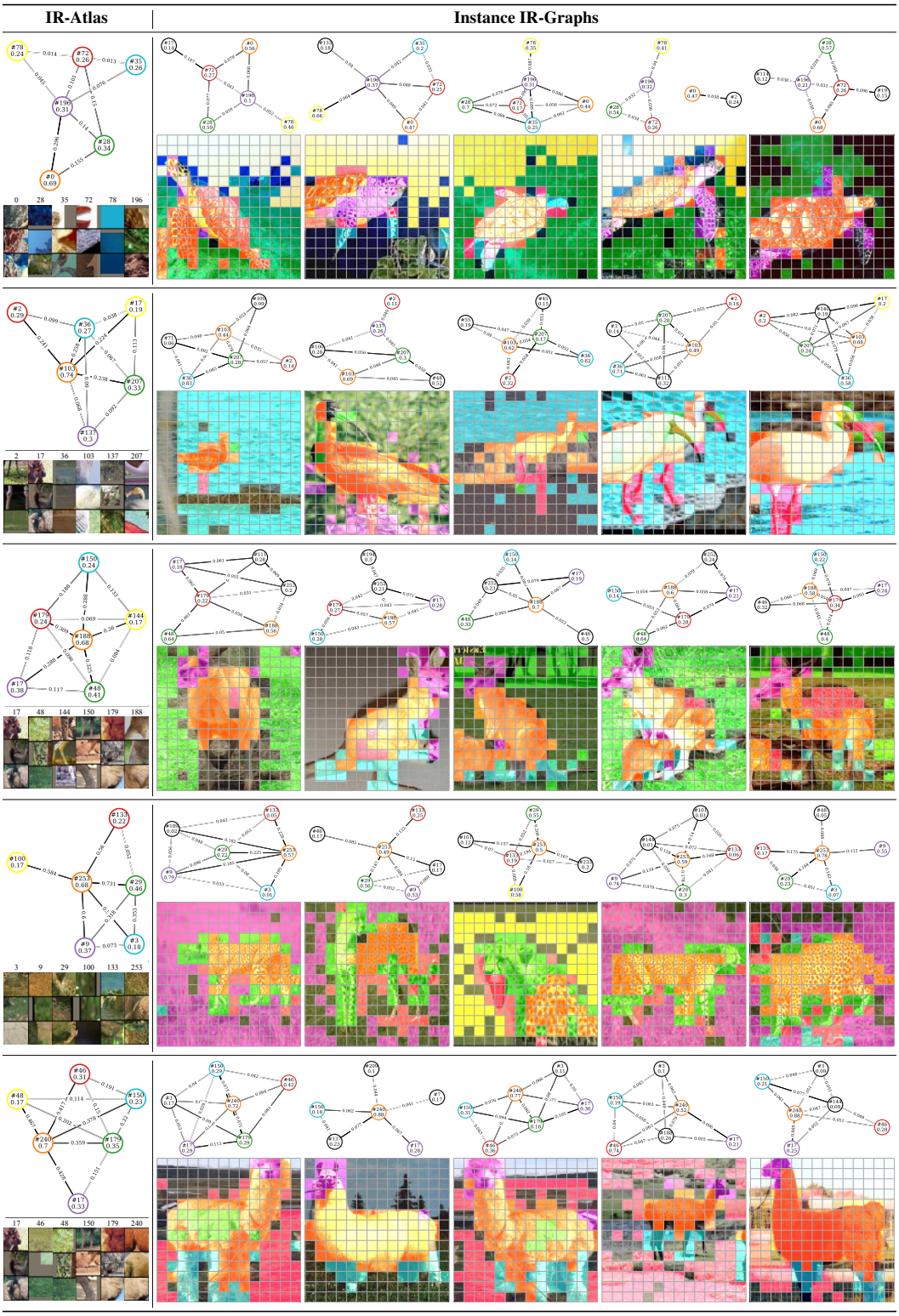


Figure 10: Examples of the learned IR-Atlas and instance IR-Graphs randomly sampled from five categories (“hawksbill”, “ibis”, “kangaroo”, “leopards”, and “llama”) on Caltech-101.

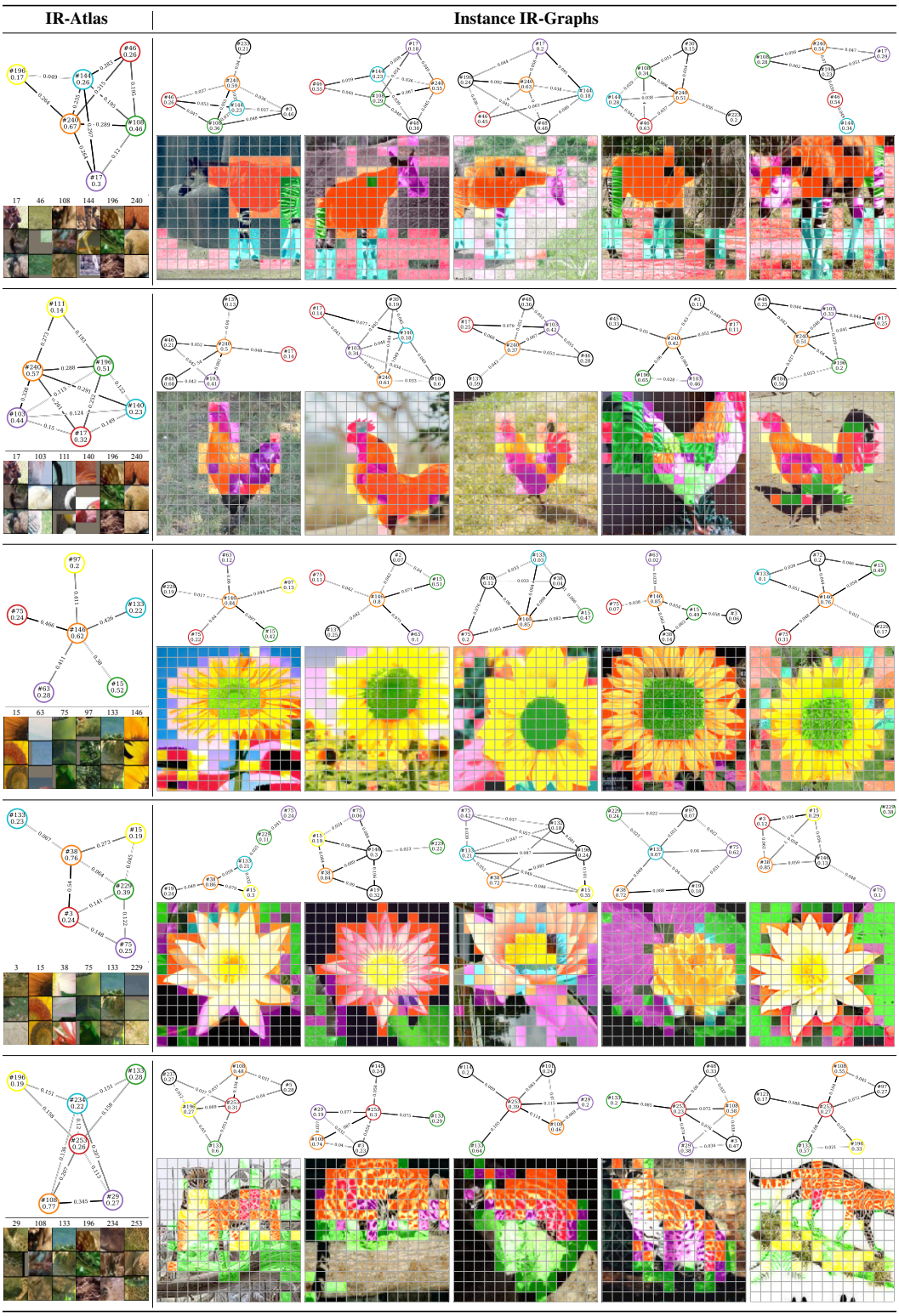


Figure 11: Examples of the learned IR-Atlas and instance IR-Graphs randomly sampled from five categories (“okapi”, “rooster”, “sunflower”, “water lilly”, and “wild cat”) on Caltech-101.