Interpretable Object Recognition by Semantic Prototype Analysis Supplementary Material

Qiyang Wan[1,2], Ruiping Wang[1,2], Xilin Chen[1,2]
[1]Key Laboratory of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing, 100190, China
[2]University of Chinese Academy of Sciences, Beijing, 100049, China
qiyang.wan@vipl.ict.ac.cn, {wangruiping, xlchen}@ict.ac.cn

1. Introduction

In this supplementary material, we elaborate on some study on the feature locality and our modification on the existing pretrained image encoder to extract feature map (Sec.3.2 in the main paper[1]), details of the datasets and preparation (Sec.4.1), implementation details of SPANet in the experiments (including hyperparameters) (Sec.4.2), and some additional clarifications on the experiment results (Sec.4.3). As mentioned in footnote 1 in the main paper, the code with instructions will be released to public upon acceptance.

2. Modification on image encoder

In the practical implementation, SPANet utilizes the pretrained models of CLIP [10] in its *Semantic Attachment Module* to label semantic tags on the learned part-wise prototypes. As mentioned in Sec.3.2, with the original image encoder of CLIP, a global feature vector $z_I \in \mathbb{R}^d$ is extracted from the input image; however, a feature map $z_L \in \mathbb{R}^{h\times w\times d}$ is required in the part-based recognition. In order to obtain local features corresponding to different spatial positions of the input image, the top layers of the CLIP encoder are required to be modified.

Many backbone choices are provided in the pretrained models of CLIP, including[2] RN50, RN101, RN50x4, RN50x16, RN50x64, ViT-B/32, ViT-B/16, ViT-L/14 and ViT-L/14@336px (in which "RN" is short for ResNet). We take ResNet-50 as an example in the following contents.

In ResNet, the 4-D tensor with shape [B,C,H,W] before the global pooling layer is usually regarded as the feature map with localization in many works, and this has also been demonstrated in XAI's implementations [1]. In the early preliminary attempts, the ResNet of CLIP also has the ability to extract grid features in a similar manner, of which the visual representational capacity has been verified in the standard image captioning task[3]. However, the grid features are not in the same feature space with embeddings from the text encoder, which implies that the alignment between the local features (in the grid feature) and text features will no longer hold.

Specifically, the ResNet as backbone in CLIP has been modified compared to the vanilla structure. According to CLIP's official code[2], it is referred to as *ModifiedResNet*, in which

- 3 "stem" convolutions are performed instead of 1, with an average pool instead of a max pool.
- anti-aliasing strided convolutions are used, where an avgpool is prepended to convolutions with stride greater than 1.
- the global pooling layer is a multi-head QKV attention instead of the original average pool.

The last change is due to the fact that grid features and text embeddings are no longer in the same space, which presents a challenge for us. The attention pooling layer takes grid features as input, and the mean feature obtained through average pooling will be used as the attention key. After applying multi-head attention (with the position embeddings), the output will be projected to obtain the final global image embeddings. We design 4 parameter-free schemes to transfer the grid features to the same-size local image features aligned with text embeddings:

1. apply the projection layer before the output directly on the input grid feature (to skip the QKV attention).
2. replace the attention key with the target feature vector to be transferred in the grid feature, while use the position embedding of the target location.

[1]For better understanding, the mentioned figures, tables, sections in the main paper are denoted in blue.

[2]https://github.com/openai/CLIP

[3]https://github.com/jianjieluo/OpenAI-CLIP-Feature

3. replace similarly as scheme 2, but use the position embedding of the mean feature.

4. expand the target feature vector $\in \mathbb{R}^d$ to a feature map $\in \mathbb{R}^{h \times w \times d}$ (in which the vector is the same for each spatial position), and then apply the entire attention pooling layer on the expanded feature map.

We conducted a simple experiment to validate the rationale behind the aforementioned designs[4]. As shown in Fig.1, a simple four-grid image was used to extract the proposed local features and compare them with embeddings of a color list [*red*, *yellow*, *green*, *blue*, *black*, *white*, *gray*, *purple*].

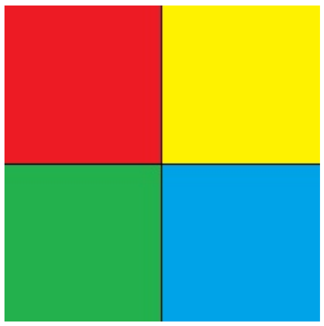


Figure 1. Test sample used to verify the alignment between the proposed local features and text embeddings.

In the experiment, the local features of the four corners extracted through different schemes will be used to calculate the similarity with the embedding of each color word. The color embeddings from the specific region are expected to have a better matching with the corresponding word.

We firstly test the original CLIP, that is using the global feature of Fig.1 to match the color list. Results are shown in Table 1, and we can find that the global feature extracted from the original image encoder is matched correctly with the corresponding color words, although some bias occurs in the results, that CLIP (ResNet ver.) may prefer *red* and *green* to *yellow* and *blue*.

Table 1. Maching scores of global feature. CLIP prefers to match Fig.1 with *red* and *green*.

global	*red*	*yellow*	*green*	*blue*
	0.4192	0.1110	**0.3757**	0.0272
	black	*white*	*gray*	*purple*
	0.0372	0.0202	0.0079	0.0016

For Scheme 1-4, results are shown in Table 2, 3, 4, and 5 respectively. Matching scores by Scheme 4 are correctly for all the colors, which indicates the effectiveness of Scheme 4. As a result, we incorporate the design of Scheme 4 when modifying ResNet backbones in our implementation of SRANet, in which we expand every local feature vector in grid feature extracted by ResNet to $h \times w$, and then apply the attention pooling layer on it to obtain the final local feature aligned with the semantic embeddings.

[4]The experiment code together with the test sample are both provided in the compressed file as "clip_test.py" and "clip_test.jpg".

Table 2. Maching scores of local feature extracted by the modified ResNet (Scheme 1). The correct matchings are in green, and the wrong matchings are in red.

top-left (**red**)	*red*	*yellow*	*green*	*blue*
	0.0096	0.0391	0.2981	0.0345
	black	*white*	*gray*	*purple*
	0.0606	0.3711	0.1809	0.0060
top-right (**yellow**)	*red*	*yellow*	*green*	*blue*
	0.0074	0.0617	0.3782	0.0320
	black	*white*	*gray*	*purple*
	0.0452	0.3135	0.1576	0.0043
bottom-left (**green**)	*red*	*yellow*	*green*	*blue*
	0.0117	0.0556	0.3008	0.0611
	black	*white*	*gray*	*purple*
	0.0447	0.3628	0.1561	0.0071
bottom-right (**blue**)	*red*	*yellow*	*green*	*blue*
	0.0053	0.0262	0.3728	0.0590
	black	*white*	*gray*	*purple*
	0.0474	0.3967	0.0885	0.0041

Table 3. Maching scores of local feature extracted by the modified ResNet (Scheme 2). The correct matchings are in green, and the wrong matchings are in red.

top-left (**red**)	*red*	*yellow*	*green*	*blue*
	0.3892	0.0932	0.3127	0.0122
	black	*white*	*gray*	*purple*
	0.1000	0.0398	0.0527	0.0001
top-right (**yellow**)	*red*	*yellow*	*green*	*blue*
	0.3149	0.0822	0.3770	0.0221
	black	*white*	*gray*	*purple*
	0.0991	0.0376	0.0671	0.0002
bottom-left (**green**)	*red*	*yellow*	*green*	*blue*
	0.3687	0.1160	0.3354	0.0171
	black	*white*	*gray*	*purple*
	0.0869	0.0356	0.0401	0.0002
bottom-right (**blue**)	*red*	*yellow*	*green*	*blue*
	0.3918	0.0766	0.3799	0.0084
	black	*white*	*gray*	*purple*
	0.0742	0.0322	0.0365	0.0003

Table 4. Maching scores of local feature extracted by the modified ResNet (Scheme 3). The correct matchings are in green, and the wrong matchings are in red.

top-left	*red*	*yellow*	*green*	*blue*
(**red**)	0.3752	0.0905	0.3040	0.0126
	black	*white*	*gray*	*purple*
	0.1050	0.0452	0.0673	0.0001
top-right	*red*	*yellow*	*green*	*blue*
(**yellow**)	0.3098	0.0783	0.3679	0.0216
	black	*white*	*gray*	*purple*
	0.1014	0.0406	0.0802	0.0002
bottom-left	*red*	*yellow*	*green*	*blue*
(**green**)	0.3596	0.1106	0.3198	0.0171
	black	*white*	*gray*	*purple*
	0.0946	0.0423	0.0556	0.0002
bottom-right	*red*	*yellow*	*green*	*blue*
(**blue**)	0.3845	0.0734	0.3726	0.0086
	black	*white*	*gray*	*purple*
	0.0787	0.0360	0.0459	0.0003

Table 5. Maching scores of local feature extracted by the modified ResNet (Scheme 4). The correct matchings are in green, and the wrong matchings are in red.

top-left	*red*	*yellow*	*green*	*blue*
(**red**)	0.9814	0.0002	0.0034	0.0023
	black	*white*	*gray*	*purple*
	0.0069	0.0035	0.0012	0.0008
top-right	*red*	*yellow*	*green*	*blue*
(**yellow**)	0.0057	0.9502	0.0133	0.0080
	black	*white*	*gray*	*purple*
	0.0089	0.0106	0.0029	0.0006
bottom-left	*red*	*yellow*	*green*	*blue*
(**green**)	0.0003	0.0001	0.9966	0.0011
	black	*white*	*gray*	*purple*
	0.0008	0.0010	0.0001	0.0001
bottom-right	*red*	*yellow*	*green*	*blue*
(**blue**)	0.0009	0.0001	0.0005	0.9971
	black	*white*	*gray*	*purple*
	0.0007	0.0004	0.0003	0.0001

3. Datasets

3.1. CUB-200-2011

CUB-200-2011 [15] is a dataset for bird species identification, including 200 bird species with 11,788 images. We adopt the official split of training set and test set, consisting of 5,994 training images and 5,794 test samples respectively. We follow the methodology of previous studies and employ the bounding boxes provided in the dataset to crop the images, which effectively eliminates potential interference from background noise. Because there are only about 30 images per category for training, an automate image augmentation with Augmentor[5] is performed. For each training image, 5 augmentation methods (rotation, skew, shear, random distortion, and random erasing) are utilized to generate a total of 40 training samples. Additionally, each augmented sample has a 0.5 probability of being horizontally flipped. We have attempted to augment a larger number of training samples, but it did not have a significant impact on the results. Therefore, we have retained a similar augmentation scale as in [1] for a fair comparison.

Regarding the concepts, we follow the processing method of CBM [4] and selected 112 relatively common attributes from the 312 attributes of CUB through majority voting as concepts. Only attributes that appear in more than half of the training samples are labeled as concepts of the class, and only concepts that are present in more than 10 classes are retained to reduce sparsity. The selected concepts are labeled on the class level, which means all the samples in the same category share the same concept labels. The adjacency matrix between selected concepts and categoires are presented in Fig.2, in which the x-axis represents 200 bird species, while the y-axis represents 112 selected concepts. The intersection of a concept and a species is marked in blue if the concept is annotated on that species, and vice versa.

In addition, to fine-tune a general vision-language model for bird species identification domain, we also utilized the captions annotated on bird images. The image-text pairs on CUB can be obtained in the previous captioning works [11, 18], in which 10 text references are provided for each image. To keep semantic consistency between text references and concepts, the concepts are randomly combined with text prompts to generate some extra descriptions for global semantic fine-tuning. In a training batch, only one caption from either text references (with the probability of 0.7) or concepts is used (with the probability of 0.3) for one training sample. Due to the context length limitation of CLIP, the actual training text that exceed a length of 77 will be truncated. Since there are much longer descriptions generated by multiple concepts on each bird species than text references, to prevent the issue of incomplete semantics caused by truncation, the generated captions for concepts are obtained by sequentially concatenating the description text of each concept after a random shuffle, stopping when approaching the context length limit of 77.

Some examples of cropped input images, text references in captioning, and generated captions by concepts are shown in Fig.3.

[5]https://github.com/mdbloice/Augmentor

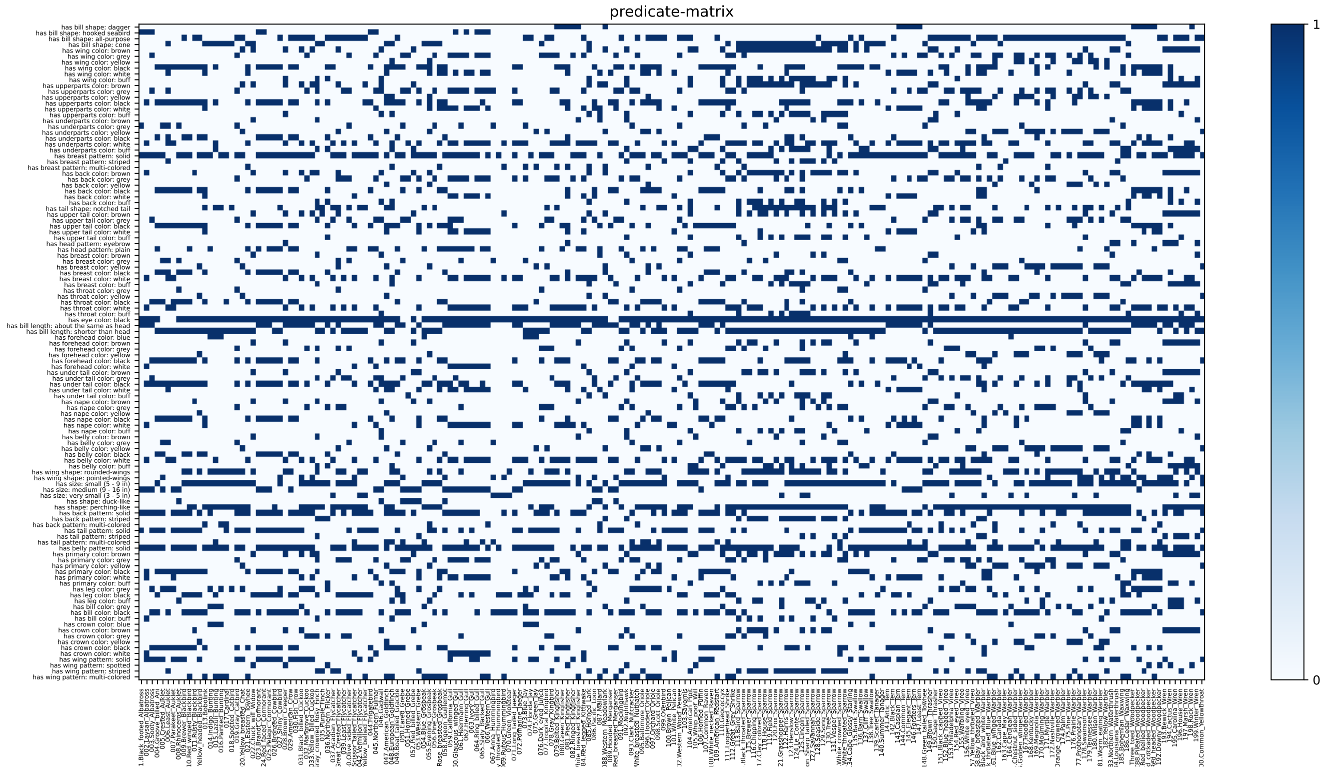


Figure 2. Visualization of relation matrix between concepts and categories.

bird species: Black-footed Albatross

text references: the medium sized bird has a dark grey color, a black downward curved beak, and long wings.
the bird is dark grey brown with a thick curved bill and a flat shaped tail.
bird has brown body feathers, white breast feathers and black beak.
...

concept caption: a photo of a bird, has bill length: about the same as head, has belly pattern: solid, ...

bird species: Rusty Blackbird

text references: a bird with a triangular bill, and pearlescent black, blue, and brown body.
this bird has metallic green sheen on its breast, and black all over besides.
a bird that has a slick black color sheen, along with a cyan-blue breast and black plumage.
...

concept caption: a photo of a bird, has shape: perching-like, has bill shape: all-purpose, has upperparts color: black, ...

bird species: Yellow-breasted Chat

text references: this small bird has a brown body with a bright yellow belly and white eyebrows and eye rings.
this bird is black with yellow on its chest and has a long, pointy beak.
bird with black eye, and gray beak, tarsus and feet, and yellow throat, breast and belly, and white abdomen.
...

concept caption: a photo of a bird, has breast color: yellow, has bill color: black, has belly pattern: solid, ...

bird species: American Crow

text references: the bird is black with short tarsals and a black bill.
a fairly tall bird that is completely black from beak to feet.
this black bird features a thick, black beak and small black eyes.
...

concept caption: a photo of a bird, has primary color: black, has back color: black, has breast pattern: solid, ...

Figure 3. Examples of training sample with text references and generated captions in the dataset.

3.2. Stanford Cars

Stanford Cars [5] is a dataset for car model recognition, including 196 car models with 16,185 images. We follow the official split of training set and test set, consisting of 8,144 training samples and 8,041 test samples of 196 car models. We use the same augmentation methods as what we do in CUB.

However, the original Stanford Cars does not contain attribute or concept annotations, so we utilize GPT-4 [9] to automatically generate a concept list for car models in the dataset. We refer to another car model dataset, CampCars [5], in which car models are annotated with five attributes, including maximum speed, displacement, number of doors, number of seats, and type of car. We consider selecting concepts that can be recognized from the visual appearance directly of car models. Then a set of concept groups including body shape (always related to maximum speed), headlight shape (often visible in the image samples), and number of doors (another main feature of vehicles) is heuristically selected as an example in the following experiments.

After determining the concept groups, we further design some prompts for GPT-4 to classify each car model in certain attribute group, such as "please categorize the following car models from their body shapes". We just provide the group names we want (such as body shape), then the potential ranges of attribute values (such as SUV, Sedan) and the annotated attribute values for car models are generated automatically without any human intervention. Afterward, we connect the selected group names and attribute values together as concepts, similar to concepts in CUB.

We conduct a random quality check on the generated concepts and annotations. We find that as long as the concept groups are selected appropriately, a significant portion of the annotations generated by GPT-4 is usable. As an example, the concept list that we use in the experiments is shown in Table 6. Additionally, the global captions required by global fine-tuning (Sec.3.3) are also generated by GPT-4, and each car model is described by a paragraph with less than 50 words.

4. Implementation details

We use specific hypermeters for different backbones in the implementation of SPANet, which are presented in Table.7. All the models are trained on 2x NVIDIA GeForce RTX 3090 or 2x NVIDIA GeForce GTX 4090.

The visual backbones are modified to generate the local features with a parameter-free design, while the text backbones keep the pretrained structure. For all the backbones, the input images are resized to 224×224, and no more augmentations (random crop or random flip) are applied after the data preparation. To simplify the problem, we set the number of the prototypes $m_c = 10$ to be the same across all the categories. The parameters ($W_h = \left\{w_h^{(k,c)}\right\}$) in the last layer h will stay the initial state during the training stage, that is $w_h^{(k,c)} = 1$ when p_k is allocated to the category c, and $w_h^{(k,c)} = -0.5$ otherwise.

The remaining hyperparameters not mentioned keep their default values in the PyTorch implementation, such as the momentum parameter of the Adam optimizer $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

Table 6. 20 generated concepts used for experiments on Stanford Cars.

ID	Concept Name
1	Body Shape: Sedan
2	Body Shape: SUV
3	Body Shape: Coupe
4	Body Shape: Convertible
5	Body Shape: Hatchback
6	Body Shape: Minivan
7	Body Shape: Wagon
8	Body Shape: Van
9	Body Shape: Crew Cab
10	Round Headlight
11	Rectangular Headlight
12	Trapezoidal Headlight
13	Angular Headlight
14	Elliptical Headlight
15	Slim Headlight
16	Two-Door Coupe
17	Four-Door Car
18	Five-Door Hatchback/Wagon
19	Three-Door Extended Cab
20	Four-Door Minivan/Van

Table 7. Hyperparameters in the implementation of SPANet.

hyperparameter	for ResNet	for ViT
image size	224	224
batch size (training)	80	80
prototype dimension	2048	768
optimizer	Adam	AdamW
weight decay	1e-2	1e-2
backbone learning rate	1e-5	5e-6
prototype learning rate	3e-4	3e-3
learning rate scheduler	step decay	cosine decay
warm-up epoches	5	10
training epoches	25	40
λ_{clst}, λ_{sep}	0.8, -0.08	0.8, -0.08
λ_{L2}	1e-4	1e-2
λ_{glb}, λ_{loc}	0.1, 0.1	0.1, 0.1

5. Clarifications on experiment results

5.1. Source of quantitative results

We compare a few of related methods in Sec.4.3, and the results are shown in Table 1. Some results in the table are reported by previous works, and some others are obtained by running the official codes by ourselves. The source of each result will be clarified in this section. For better understanding, Table 1 in the main paper is copied here as Table 8.

Vanilla CNN/ViT. On CUB, the result of ResNet-50 (84.5) is reported by [3], and results of ResNet-101 (83.5) and ViT-B/16 (89.4) is reported by [2]. On Stanford Cars, the result of ResNet-101 (91.2) is reported by [2]. The performance of ViT-B/32 (82.3) is obtained by fine-tuning pretrained model *vit_base_patch32_clip_448.laion2b_ft_in12k_in1k* in *timm* [17]. The size of input image of this baseline group is 448x448, which is different from all of the other methods. Increasing the input size benefits the improvement of fine-grained recognition performance. However, in order to ensure fair comparison with other interpretable models (and considering the limitations of the CLIP pre-training model), we still use an input size of 224x224 in SPANet.

Models with part-wise interpretability. On both of CUB and Stanford Cars, for ProtoPNet [1], the result of ResNet-34 (79.2 / 86.1) is reported by [1], and the performances of ResNet-50 (79.4 / 87.9) and ResNet-101 (77.3 / 87.6) are tested by the official code with different backbones. For ProtoPShare [13], TesNet [16], ProtoPool [12], ProtoTree [7], and PIP-Net [6], the results are all from their authors. It is worth noting that on CUB the results of some methods (marked ‡ in Table 8) on ResNet-50 are obtained by fine-tuning on a model pre-trained on the iNaturalist2017 [14], of which the data domain is closer to the target domain (bird species), which allows for relatively better results to be achieved.

Models with semantic interpretability. On CUB, for CBM [4], the result of Inception-v3 (80.1) is reported by [4], while the result on ResNet-18 (Backbone Fixed, 62.9) is obtained from [19], together with PCBM's performance (87.2). The result of Label-free CBM [8] (74.3) is reported by the authors. These methods are not evaluated on Stanford Cars usually, so we no longer list them.

It should be noted that these methods are implemented based on different backbone models, so the experimental results are not directly comparable. However, among models using the same backbone, where Inception-v3 and ResNet-50 have similar parameter sizes (24M vs 26M) are approximately comparable, comparisons can be made to demonstrate the performance advantages of different models.

Table 8. Quantitative results in the main paper, copied here for better understanding.

Method	Interpret.	Backbone	CUB	Cars
Vanilla CNN/ViT	None	RN50	84.5	86.3*
		RN101	83.5	91.2
		ViT-B/32	82.3*	89.5*
		ViT-B/16	89.4	93.7
ProtoPNet [1]	Part-wise	RN34	79.2	86.1
		RN50	79.4*	87.9*
		RN101	77.3*	87.6*
ProtoPShare [13]	Part-wise	RN34	74.7	86.4
TesNet [16]	Part-wise	RN34	82.8	90.9
ProtoPool [12]	Part-wise	RN34	80.3	89.3
ProtoPool [12]	Part-wise	RN50‡	85.5	88.9
ProtoTree [7]	Part-wise	RN50‡	82.2	86.6
PIP-Net [6]	Part-wise	RN50‡	82.0	86.5
CBM [4]	Semantic	Inc.-v3	80.1	-
		RN18	62.9	-
PCBM [19]	Semantic	RN18	58.8	-
LFCBM [8]	Semantic	RN18	74.3	-
SPANet (Ours)	Both	RN50	81.7	88.9
		RN101	77.7	85.6
		ViT-B/32	83.0	90.3
		ViT-B/16	87.2	93.7

* indicates results reproduced based on official codes.
‡ indicates models pretrained on iNaturalist2017.

5.2. Qualitative analysis on Stanford Cars

Some examples of SPANet's explanations on Stanford Cars are shown in Figure 4. Since the concept list on the Stanford Cars dataset is generated by GPT-4 [9] with minimal human intervention, the quality of concepts is significantly different from the manually annotated concepts in CUB. Additionally, due to the small size of Stanford Cars (less than 10,000 training samples), weakly supervised learning of semantic prototypes poses a great challenge.

Nevertheless, SPANet is still able to learn many representative features of car models. As shown in the upper two lines of the figure, although the annotated concept list does not include the concept of "SUV vehicles are always with *high ground clearance*," the model is still able to infer that "SUV vehicles typically have features like these patterns", and then use the patterns to recognize the car models. It just lacks the knowledge of the specific name for this feature.

From this, we can see that SPANet has a strong ability to generalize visual features, but semantic labels are limited by the quality of the concept list and the quantity of training data. When there is a high-quality concept list, the model can have a strong ability for semantic attachment.

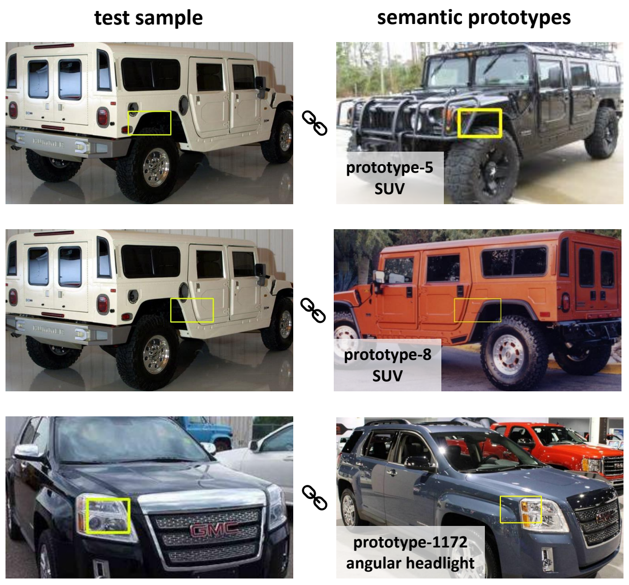


Figure 4. Examples of car model recognition on Stanford Cars. Generated concepts are used in training stage and SPRNet learns the semantic prototypes by weakly supervised learning.

However, the generated concepts still play a role in assisting the model's recognition, regardless of their quality. We further compare the recognition performance when using and not using automated annotations for concepts. The results are presented in Table 9. Despite the presence of some noise in the generated concept annotations, SPANet still achieves better performance when utilizing semantic concepts in the training stage.

Table 9. Recognition accuracy on Stanford Cars when using and not using generated concepts.

SPANet	w/o semantic intp.	**w/ semantic intp.**
RN50	85.3	**88.9**
RN101	85.6	85.6
ViT-B/32	89.9	**90.3**
ViT-B/16	92.7	**93.7**

5.3. Details of interpretability evaluation

We conduct a user study to evaluate the interpretability of SPANet. In the user study, participants are asked to indicate what they believe to be the most persuasive explanation for the classification results.

The test images are from the test split of CUB, about 1,000 out of 5,794 samples are randomly selected. The user study is conducted in three settings: the two settings ("w/o sem." and "w/ sem.") in the Table 2 and an extra setting (with complex concepts with part names may have been predicted incorrectly). In each setting, all of 1,000 images are used, and each participant will be selected to complete the user study in 2 out of 3 settings, totaling 200 images per person. We ensure that each participant is assigned completely different images in the two settings to avoid recognition biases and guarantee the randomness and fairness of the test.

The screen of user interface is shown in Fig. 5. The system presents the participants with classification results and three explanations. Participants are asked to click on the explanation they consider to be the best (including an option for "None" to indicate dissatisfaction with all explanations). Subsequently, the results are collected and the number of times each method is selected is counted for each setting. The final score is calculated by dividing this count by the total number of images in that setting, with higher scores indicating better interpretability.

References

[1] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: deep learning for interpretable image recognition. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019. 1, 3, 6

[2] Marcos V Conde and Kerem Turgutlu. Exploring vision transformers for fine-grained classification. *arXiv preprint arXiv:2106.10587*, 2021. 6

[3] Ju He, Jie-Neng Chen, Shuai Liu, Adam Kortylewski, Cheng Yang, Yutong Bai, and Changhu Wang. Transfg: A transformer architecture for fine-grained recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(1):852–860, 2022. 6

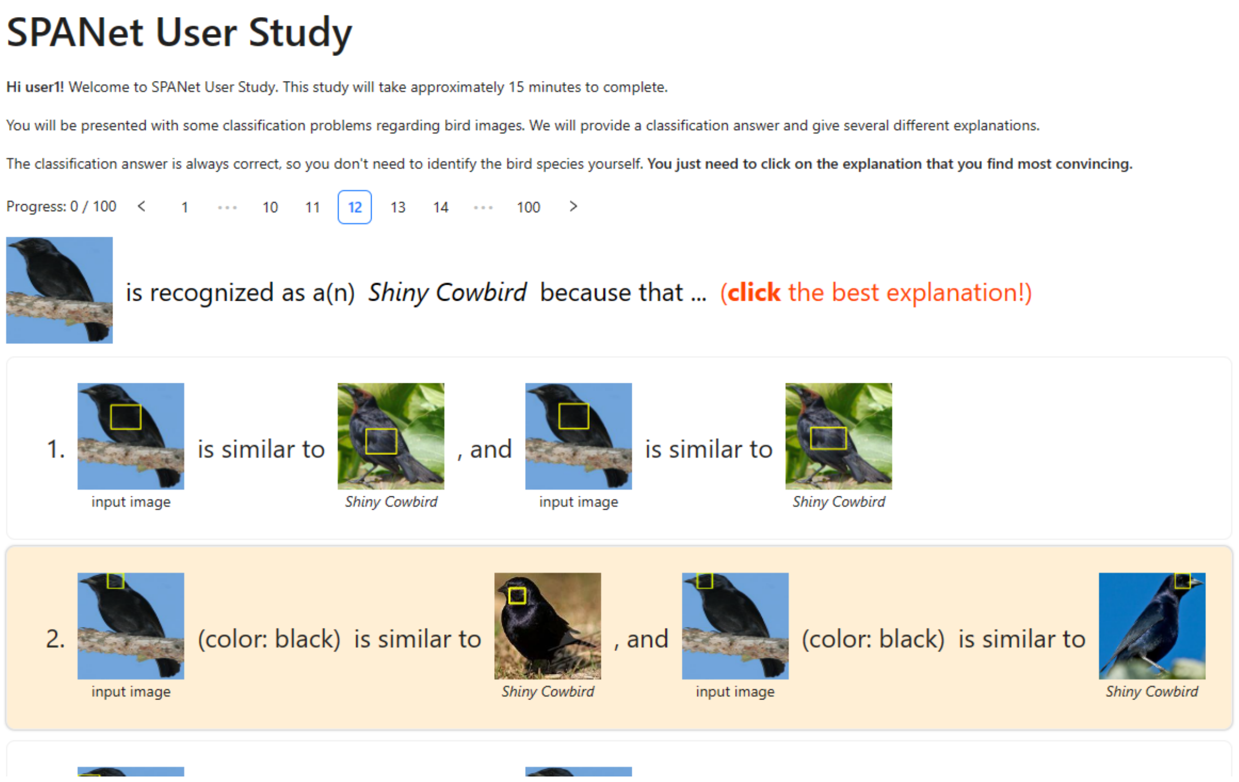
[4] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International Conference on Machine Learning (ICML)*, pages 5338–5348. PMLR, 2020. 3, 6

[5] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, June 2013. 5

[6] Meike Nauta, Jörg Schlötterer, Maurice van Keulen, and Christin Seifert. Pip-net: Patch-based intuitive prototypes for interpretable image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2744–2753, 2023. 6

[7] 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)*, pages 14933–14943, 2021. 6

[8] Tuomas Oikarinen, Subhro Das, Lam M Nguyen, and Tsui-Wei Weng. Label-free concept bottleneck models. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. 6


SPANet User Study

Hi user1! Welcome to SPANet User Study. This study will take approximately 15 minutes to complete.

You will be presented with some classification problems regarding bird images. We will provide a classification answer and give several different explanations.

The classification answer is always correct, so you don't need to identify the bird species yourself. **You just need to click on the explanation that you find most convincing.**

Progress: 0 / 100 < 1 ··· 10 11 12 13 14 ··· 100 >
is recognized as a(n) *Shiny Cowbird* because that ... (**click** the best explanation!)
1. input image is similar to Shiny Cowbird , and input image is similar to Shiny Cowbird
2. input image (color: black) is similar to Shiny Cowbird , and input image (color: black) is similar to Shiny Cowbird

Figure 5. Screenshot of user interface for user study.

[9] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 5, 6

[10] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 8748–8763. PMLR, 2021. 1

[11] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *International Conference on Machine Learning (ICML)*, pages 1060–1069. PMLR, 2016. 3

[12] Dawid Rymarczyk, Łukasz Struski, Michał Górszczak, Koryna Lewandowska, Jacek Tabor, and Bartosz Zieliński. Interpretable image classification with differentiable prototypes assignment. In *European Conference on Computer Vision (ECCV)*, pages 351–368. Springer, 2022. 6

[13] Dawid Rymarczyk, Łukasz Struski, Jacek Tabor, and Bartosz Zieliński. Protopshare: Prototypical parts sharing for similarity discovery in interpretable image classification. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1420–1430, 2021. 6

[14] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8769–8778, 2018. 6

[15] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. Caltech-ucsd birds-200-2011 (cub-200-2011). Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 3

[16] Jiaqi Wang, Huafeng Liu, Xinyue Wang, and Liping Jing. Interpretable image recognition by constructing transparent embedding space. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 895–904, 2021. 6

[17] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 6

[18] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1316–1324, 2018. 3

[19] Mert Yuksekgonul, Maggie Wang, and James Zou. Post-hoc concept bottleneck models. In *ICLR 2022 Workshop on PAIR^2Struct: Privacy, Accountability, Interpretability, Robustness, Reasoning on Structured Data*, 2022. 6