Supplementary Material

A. Reproducibility

All source codes, figures, models, etc., are available at https://github.com/PavlicLab/WACV2024-Hong-Concept_Centric_Transformers.git.

B. Method Details

Algorithm. Algorithm 1 shows the details of the Concept-Slot-Attention (CSA) module in our Cocenpt-Centric Transformer (CCT) in pseudo-code. Algorithm 1 is described based on the SA [57], but we simplify it by removing the last `LayerNorm` and `MLP` layers. Because of the modular characteristics in our framework, we leverage three slot-based approaches, including SA [57], I-SA [10], and BO-QSA [41], which are interchangeable. We implemented the above approaches based on their official repositories/works.

Positional Embedding for Concept Slots. Because the resulting set of concept slots is orderless [57], it is difficult for the later module to: (i) recognize which high-level concept each concept slot is representing and (ii) identify which concept slot each high-level concept belongs to. Thus, we add a positional encoding $\mathbf{p}_c$ to each concept slot to avoid these challenges; in particular, $\hat{\mathbf{s}}_c = \mathbf{s}_c + \mathbf{p}_c$. After this, the concept embedding representation $\hat{\mathbf{S}}^{\texttt{concept}} \in \mathbb{R}^{C \times d}$ with positional embedding $\mathbf{P}$ is passed as the final concept representation to the next module.

Algorithm 1 Concept-Slot-Attention (CSA) module. The module receives the set of input features $\mathbf{E} \in \mathbb{R}^{L \times D}$; the number of concepts C; and the dimension of concepts d. The model parameters include: the linear projection $q^{\texttt{CSA}}, k^{\texttt{CSA}}, v^{\texttt{CSA}}$ with output dimension d; a `GRU` network; a Gaussian distribution's mean and diagonal covariance $\mu, \sigma \in \mathbb{R}^d$.

$\mathbf{S}^{\texttt{concept}} = \texttt{Tensor}(C, d) \triangleright \texttt{concept} - \texttt{slots} \in \mathbb{R}^{C \times d}$
$\mathbf{S}^{\texttt{concept}} \sim \mathcal{N}(\mu, \sigma)$
$\mathbf{E} = \texttt{LayerNorm}(\mathbf{E})$
for $t = 0, \ldots, T$ **do**
  $\mathbf{S}^{\texttt{concept}} = \texttt{LayerNorm}(\mathbf{S}^{\texttt{concept}})$
  $\mathbf{A}^{\texttt{CSA}} = \texttt{softmax}(\frac{1}{\sqrt{d}} q^{\texttt{CSA}}(\mathbf{S}^{\texttt{concept}}) \cdot k^{\texttt{CSA}}(\mathbf{E})^\top, \texttt{axis} =' \texttt{concept} - \texttt{slots}')$
  $\mathbf{A}^{\texttt{CSA}} = \mathbf{A}^{\texttt{CSA}} \,/\, \mathbf{A}^{\texttt{CSA}}.\texttt{sum}(\texttt{axis} =' \texttt{inputs}')$
  $\mathbf{U} = \mathbf{A}^{\texttt{CSA}} \cdot v^{\texttt{CSA}}(\mathbf{E})$
  $\mathbf{S}^{\texttt{concept}} = \texttt{GRU}(\texttt{state} = \mathbf{S}^{\texttt{concept}}, \texttt{inputs} = \mathbf{U})$
end for
return $\mathbf{S}^{\texttt{concept}}$

Number of Iterations in the Concept Slot Attention module. In the original usage of slot-based approaches, the refinement could be repeated several times depending on the tasks. For setting the number of iterations T (Algorithm 1), we set the different number of iterations T for the three slot-based methods we used. For the vanilla SA, we set T to 1 because the initialization for slots in the SA has some limitations revealed by [10], and we found that a single iteration to update the concept slots benefits in which the learned concept slots possess much information is performed through the training with backpropagation. In other words, our proposed interpretable broadcast scheme (4.2 in the main text) contributes to better performances for the SA than the internally refined iterations used in the conventional slot-based methods. Furthermore, we set T to 3 for the BO-QSA and the I-SA by simply following the best values of iterations in their works.

Comparison with Concept Transformers [65]. As shown in [65, Fig. 1], Concept Transformers (CTs) are the limited usage in our formulation, which leverages learnable vectors for concept learning instead of using "a shared workspace". We highlight that the concept embedding representation in our CCT is more sophisticated and generalizable than the one in CTs. Each concept embedding in CT is represented as a simple learnable vector shared with all input batches to learn. As such, it may be difficult for the vector to capture which image features can contribute to each concept more than others. For instance, as shown in Fig. 2 in the main text, given three images belonging to the same class "Winter Wren", the image features with the same spatial positions from each image can have different importance for providing information to learn the same global concept, such as *"what is the main body color for Winter Wren?"* or *"what is the body shape for it?"*. In contrast, our proposed module naturally aggregates how much each image feature contributes to each concept using the attention $\mathbf{A}^{\texttt{CSA}}$ (See in Algorithm 1) and provides batch-specific concept embedding representations that have more semantically meaningful information. In all of the experiments, we demonstrate that our CCTs always outperform CTs.

C. Further Experimental Results and Details

In this section, we explain further experimental results and details. All experiments are conducted with three different random seeds and 95% confidence intervals.

C.1. Dataset Statistics

Table A-1 depicts the statistics of all benchmark datasets in our experiments. Because all datasets have no portion of validation, we manually pick the portion of the validation dataset for exploring the best hyperparameters for models.

For CIFAR-100 Super-class dataset, it serves as a notable example for nuanced image categorization. Originating from the original CIFAR-100 dataset, it contains 100 fine-grained image classes that are aggregated into 20 distinct super-classes for broader categorization. For instance, the super-class "vehicles 1" comprises specific, fine-grained classes like "bicycle," "bus," "motorcycle," "pickup truck," and "train." This structured hierarchy makes the dataset a widely used benchmark in evaluating deep learning models, particularly those that incorporate logical constraints.

For CUB-200-2011, we explain the pre-processing steps following [65]. Initially, the dataset has 312 binary attributes, but we filter to retain only those attributes that occur in at least 45% of all samples in a given class and occur in at least 8 classes. Thus, we get a total of 108 attributes, and based on this, we group them into two kinds of concepts: spatial and global concepts. We finally get 13 global and 95 spatial concepts by looking at each attribute. For example, `has_shape::perching-like` and `has_primary_color::black` are global concepts, and `has_eye_color::black` and `has_forehead_color::yellow` are spatial concepts.

For the ImageNet dataset, we adopted the methodology presented in [84] to validate our CCT's ability to learn latent concepts without the need for explicit concept explanations. [84] uses attention maps to identify a predefined set of concepts within the dataset, focusing on the first 200 classes of ImageNet for their evaluation. Following this approach, we also utilized the first 200 classes in ImageNet for training and conducted evaluations to determine test accuracy on the ImageNet dataset. This experiment serves to benchmark our CCT model's capabilities in learning latent concepts without explicit explanations, further showcasing its effectiveness in an unsupervised setting.

C.2. Hardware Specification of The Server

The hardware specification of the server that we used to experiment is as follows:

- CPU: Intel® Core™ i7-6950X CPU @ 3.00GHz (up to 3.50 GHz)
- RAM: 128 GB (DDR4 2400MHz)
- GPU: NVIDIA GeForce Titan Xp GP102 (Pascal architecture, 3840 CUDA Cores @ 1.6 GHz, 384-bit bus width, 12 GB GDDR G5X memory)

C.3. Model Architectures

We leverage three kinds of backbones, including Vision Transformers (ViT) [22], Swin Transformers (Swin) [55], and ConvNeXt [56]. We employ `timm` Python library supported by Hugging Face™.

Variants of ViT. In our experiments, we use three variants of Vision Transformer (ViT), `ViT-Large`, `ViT-Small`, and `ViT-Tiny` (`vit_large_patch16_224`, `vit_small_patch16_224`, and `vit_tiny_patch16_224`, respectively in `timm`). These variants are defined by their number of encoder blocks, the number of attention heads on each block, and the dimension of the hidden layer. The `ViT-Large` has 24 encoder blocks with 16 heads, and the dimension of the hidden layer is 1024. In addition, The `Vit-Small` has 12 encoder blocks with 6 heads, and the dimension of the hidden layer is 384. Finally, the `ViT-Tiny` has 12 encoder blocks with 3 heads, and the dimension of the hidden layer is 192, which is much more lightweight. A comparison between the variants of the ViT is shown in Table A-2.

Architecture of Swin-Large. In our experiment, we use the Swin Transformer (Swin)-Large (`swin_large_patch4_window7_224.ms_in22k` in `timm`). See Table A-3 for the overall architecture of Swin-Large.

Architecture of ConvNeXt-Large. In our experiment, we leverage ConvNeXt-Large (`convnext_large.fb_in22k_ft_in1k` in `timm`). See Table A-4 for the overall architecture of ConvNeXt-Large.

C.4. CIFAR100 Super-class Experiments

The Difference Between CCT and Other Deep-Learning Approaches with Logical Constraints. The logical constraint by [23] to address this task was introduced, i.e., once the model classifies an image into a class, the predicted probabilities of that super-class must first arrive at the whole mass. Referring to this, several deep-learning approaches leverage it [34, 51]. For instance, the five classes, *baby, boy, girl, man,* and *woman*, belong to the super-class *people*. Thus, $\Pr_{people}(\mathbf{x}) = 1$ if $\mathbf{x}$ is classified as *baby*. So, logical constraints can be formulated as $\wedge_{s \in superclass}(\Pr_s(\mathbf{x}) = 0 \vee \Pr_s(\mathbf{x}) = 1)$, where the probability of the super-class is the sum of its corresponding classes' probabilities, e.g., $\Pr_{people}(\mathbf{x}) = \Pr_{baby}(\mathbf{x}) + \Pr_{boy}(\mathbf{x}) + \Pr_{girl}(\mathbf{x}) + \Pr_{man}(\mathbf{x}) + \Pr_{woman}(\mathbf{x})$.

However, because our CCT (and CT) follows Proposition 1 in [65], the probability of choosing the preferred superclass is $s^c = \arg\max_s(\beta_c)_s$ of class c (Refer to Eq.(2) in the main text). This means the super-class is determined by the largest attention score of the class among all classes, which is different from the definition of the logical constraint above.

Additional Results. Figure A-1 showcases several examples of correct predictions by CT and our CCT, highlighting

Table A-1. Benchmark Dataset Statistics. † indicates the rescaled size of inputs for the ViT backbone, which is different from the original sizes of the datasets. For the ImageNet experiment setup, we follow [84].

Dataset	**CIFAR100 Super-class**	**CUB-200-2011**	**ImageNet**
Input size	$3 \times 28 \times 28$	$3 \times 224 \times 224^{\dagger}$	$3 \times 224 \times 224^{\dagger}$
# Classes	20 (super-class)	200 (bird species)	200 (objects)
# Concepts	100 (class)	13 (global), 95 (spatial)	50 (latent spatial)
# Training samples	55,000	5,994	255,224
# Validation samples	5,000	1,000	10,000
# Test samples	10,000	4,794	20,000

Model	Layers	Hidden Dim.	Heads	# Params.
ViT-Tiny	12	192	3	5M
ViT-Small	12	384	6	22M
ViT-Large	24	1024	16	304 M

Table A-2. Comparison between variants of the ViT [22]

Model	Layers (per stage)	Hidden Dim.	Attention Heads	# Params.
Swin-Large	(2, 2, 18, 2)	192	6	197M

Table A-3. Overall architecture of the Swin Transformer [55]

Model	Conv. Layers (per block)	Hidden Dim.	Attention Heads	# Params.
ConvNeXt-Large	2	512	16	50M

Table A-4. Overall architecture of the ConvNeXt [56]

that compared to CT, our CCT achieves sparser explanation attention scores, which indicates a more confident and robust degree of belief for decision-making.

Hyperparameter Settings. For the experimental results, including Table 1 and Fig. 3 in the main text, referring to [23], we first find the best experimental setups of both ViT-Tiny and CT because they have not been evaluated on CIFAR100 Super-class before. Once the hyperparameter setup for the ViT-Tiny backbone is found to achieve similar performance to the Wide ResNet (the backbone for the second baseline group), we apply the same hyperparameter setting to both CT and our CCT. Table A-5a presents all shared hyperparameters for ViT-Tiny, CT, and all configurations of CCT. Please refer to Table A-6 for hyperparameter settings used in our baseline experiments.

C.5. CUB-200-2011 Experiments

Additional Results with Comparison with CT. With concept explanations, we introduce additional experimental results to compare predictions between CT and our CCT.

Figure A-2 depicts a case where the CT's predictions were utterly different from each other despite all images having the same class. The predicted classes from the CT in Fig. A-2 were `Prothonotary_Warbler`, `Pine_Warbler`, and `Magnolia_Warbler`, respectively. The only difference between the first two predictions that the CT made is whether the spatial concept `has_under_tail_color::black` exists, indicating the CT's performance sensitivity. In addition, the third prediction from the CT contains new spatial concepts unrelated to the first two examples, which shows analytical uncertainty as to why the CT made such a decision. In contrast, the predictions made by our CCT were all correct, with consistent global explanations. This indicates that although incorporating spatial concepts provide additional in-

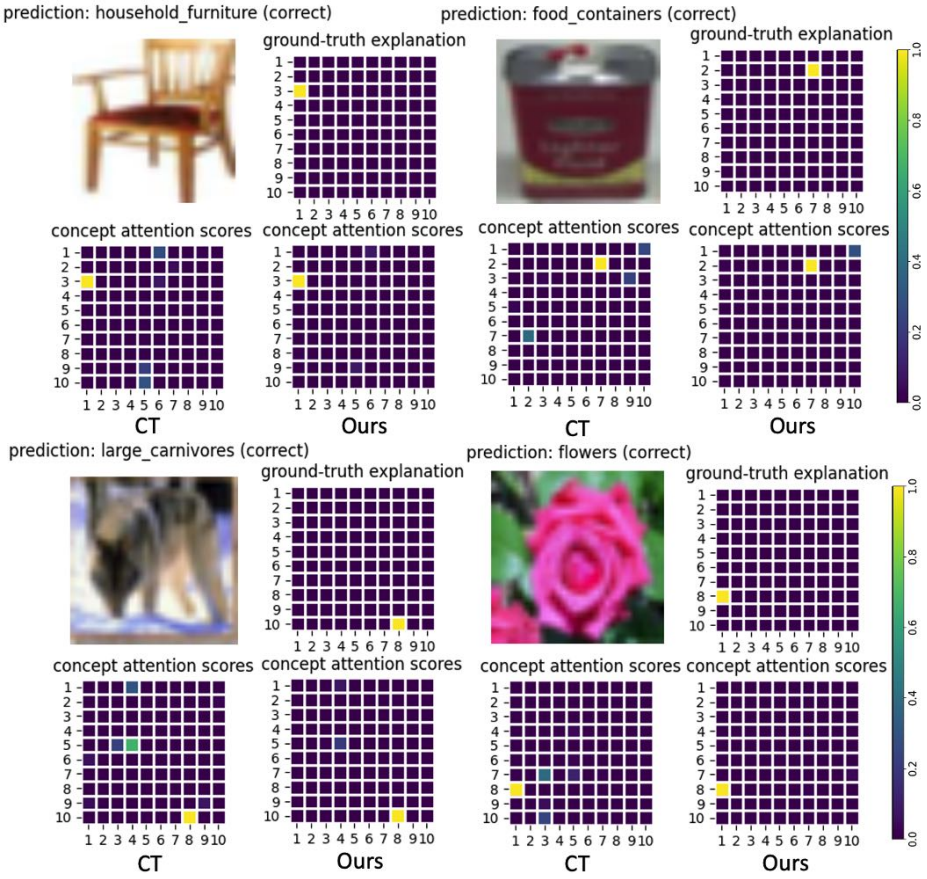

Figure A-1. Examples of correct predictions by CT and CCT on CIFAR100 super-class. The 100 classes are indexed from 1 (top left in 10x10 grid) to 100 (bottom right in 10x10 grid).

formation, determining robust global concepts is more important for the classification task.

Figure A-3 shows some examples where our CCT outperforms CT to classify `Olive_sided_Flycatcher`. All CT's predictions were `Western_Wood_Pewee`, and this is caused by their spatial explanations, including `has_eye_color::black` and `has_leg_color::black`. We discovered that these two spatial attributes are critical for `Western_Wood_Pewee`, and thus mislead the model's predictions. In contrast, the explanations achieved by CCT are more robust and consistent.

Finally, Figs. A-4 and A-5 demonstrate the failure cases where both CCT and CT predict incorrectly. In both models, noise caused by spatial explanations common in some classes tends to impair the performance of prediction results. However, compared to CT, our proposed CCT captures richer global explanations, which thus helps to achieve better classification performance in general.

Hyperparameter Settings. We follow the official implementation of CT and produce the experimental results, including Table 2, Fig. 4, in the main text, Fig. A-2, and Fig. A-3 in Appendix. The only difference in hyperparameter settings between our CCT and CT is the learning rate for the AdamW optimizer, 5e-5 for CT, 1e-5 for the configurations with the ViT-Large backbone of our CCT, 2e-5 for our CCT with SwinT-Large, and 5e-5 for our CCT with ConvNeXt-Large. Table A-5c shows the shared hyperparameters for both approaches. For those baselines not listed in Table A-5c, we refer directly to the data presented in their respective publications.

Conceptual Visualization For an in-depth look into the learnt concepts within CUB-200-2011 dataset, we adapted our CCT model to 50 distinct classes and 20 latent concepts following [84]. A comprehensive visualization of these refined latent concepts can be viewed in Figure A-6, A-7 and A-8. Please note that we present these visualization results without providing concept explanations.

In the CUB-200-2011 dataset, our model's first concept

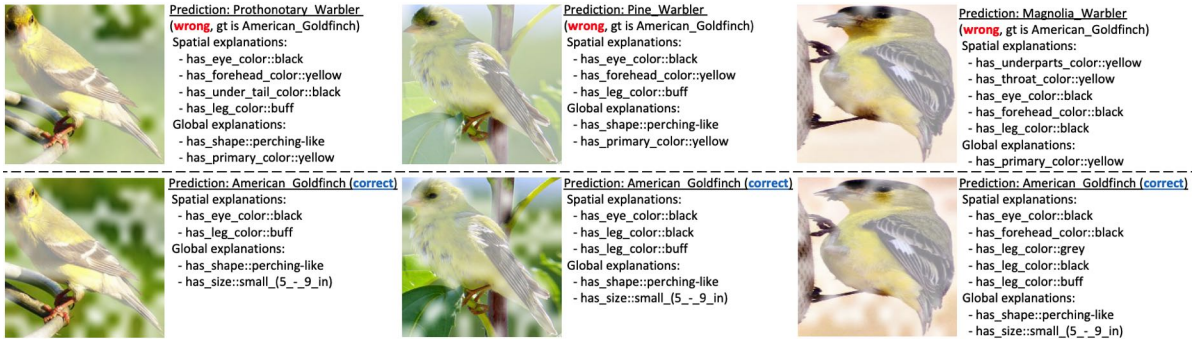

Figure A-2. Prediction comparison between CT and our CCT. (**Top**) CT's predictions are incorrect. (**Bottom**) All predictions are correct by our CCT.

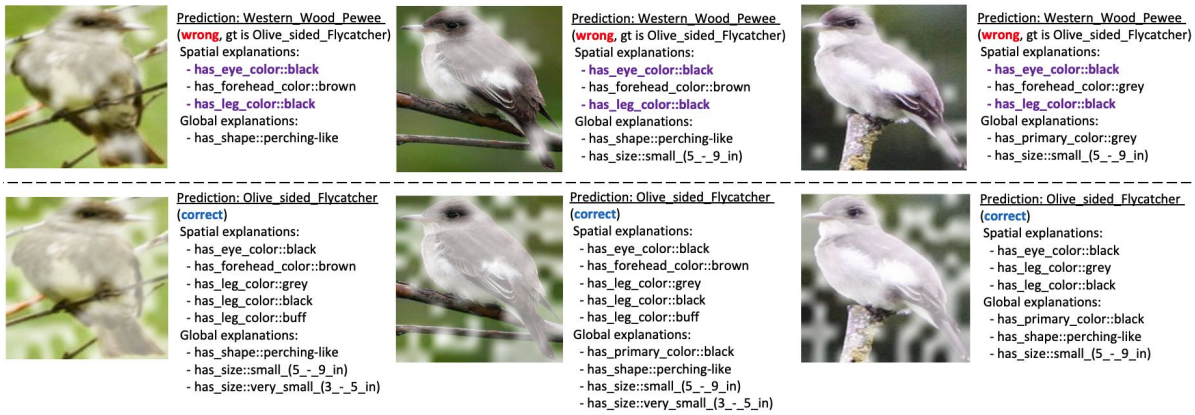

Figure A-3. Prediction comparison between CT and CCT. (**Top Row**) CT's predictions are incorrect. The purple highlighted explanations are key attributes in `Western_Wood_Pewee`. (**Bottom Row**) All predictions are correct by our CCT.

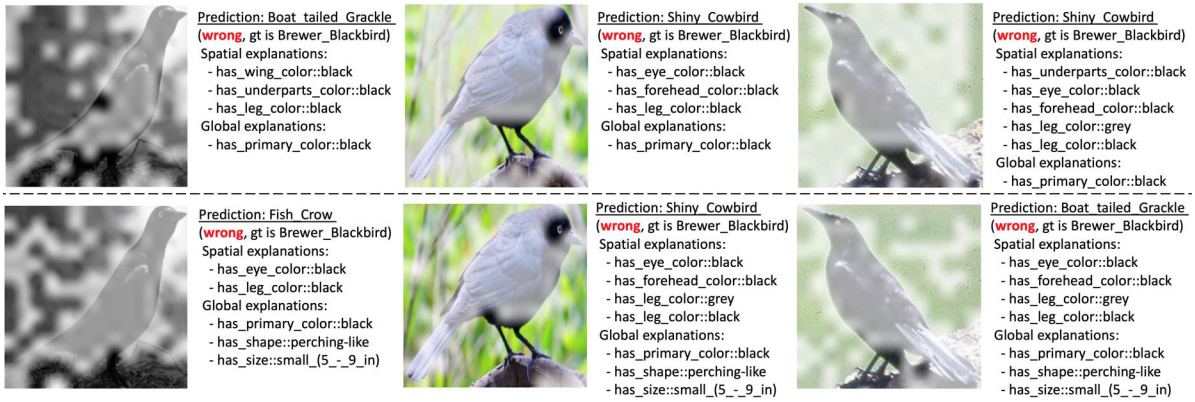

Figure A-4. The example of failure predictions of both CT (**Top Row**) and CCT (**Bottom Row**).

highlights key features of birds such as their head and beak area. Concept 2 focuses on the contours of seagulls by emphasizing the background. Concept 3 considers multiple features like the beak, eyes, and tail, while Concept 4 isolates birds from complex backgrounds. Similarly, Concept 5 outlines the bird's entire body. Concept 6 specializes in highlighting the body area of yellow-bodied birds, and Concept 7 zeroes in on the beak and upper torso. Concept 8 not only highlights contours but also focuses on the bird's eye area. Concepts 9 and 10 share similarities with Concept 8, emphasizing the eye region. Concept 11 stands out by focusing on the head area of red parrots, which is also a feature that catches human attention. Concepts 12 and 13 are dedicated to the bird's feet and lower body areas, re-

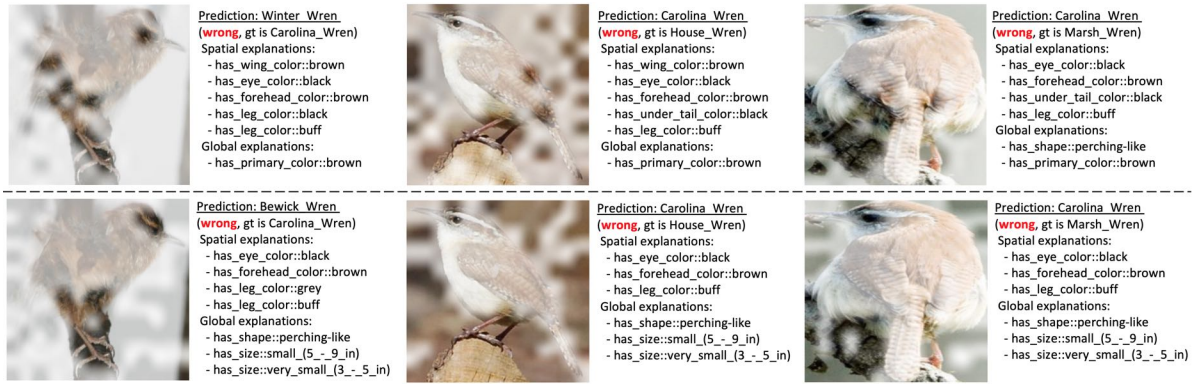

Figure A-5. The example of failure predictions of both CT (**Top Row**) and CCT (**Bottom Row**).

spectively. Concept 14 traces the entire body of a seagull, whereas Concept 15 captures the contours of birds in flying poses. Concept 16, on the other hand, focuses on the contours of birds in sitting poses. Concept 17 centers on the eye and mid-body regions, while Concept 18 partially captures key features of long-necked birds. Concept 19 captures unique head and feather structures. Finally, Concept 20 perfectly outlines the bird's contour.

These results demonstrate the model's ability to focus on a wide range of features, from beaks and eyes to tails and feet, reinforcing its classification skills. This is particularly remarkable in an environment where explicit concept explanations are unavailable, highlighting another dimension where our model excels over others.

Image Retrieval and Clustering. In addition to its robust classification capabilities, our CCT model excels in the realm of image retrieval. Remarkably, the model achieves this without requiring explicit explanations for the 20 latent concepts it identifies within the CUB-200-2011 dataset. This intuitive clustering of images based on inherent features—ranging from the contours of seagulls to the unique features of a red parrot's head—demonstrates another dimension in which our model surpasses others in the field. Essentially, it clusters semantically related images based on these latent concepts, offering coherent results even in an environment where we don't have access to concept explanation. For instance, images featuring parrots are clustered together, driven by Concept 11's focus on the head area of red parrots. Similarly, images of seagulls are grouped together, guided by Concept 2's emphasis on seagull contours.

Ablation Study. In Figure A-9, we can infer that the CCT's performance is sensitive to the explanation lambda hyperparameter λ_{expl} (4.3 in the main text). There seems to be an optimal range for explanation lambda, somewhere between 0 and 10, within which the model performs best. Values of λ_{expl} higher than 10 lead to progressively worse performance, with severe degradation observed at the highest values of lambda tested.

In Figure A-10, we can infer that a lower learning rate of 1.00e-4 provides the best model performance, especially when the explanation lambda λ_{expl} is set to 0.0. As the learning rate increases, the model's accuracy deteriorates significantly. The extremely low accuracy at higher learning rates (1.00e-01 and 1.00e+00) suggests that the model is unable to learn effectively, likely overshooting the optimal values during training due to too large updates to the model's parameters.

C.6. ImageNet Experiments

Experiment Design. To further confirm CCT's capability to derive latent concepts autonomously, we conducted an experiment on the ImageNet dataset, adhering to the methodology described in [84]. We chose ViT-S as our backbone architecture, avoiding models with more than 45M parameters, such as ResNet-101, Swin-S, and ConvNeXt-S.

Conceptual Visualization. For an in-depth look into the learnt concepts, we tailored our CCT model to 20 classes and 10 latent concepts, as in [84]. Figure A-11 provides a detailed visualization of these latent concepts. Please note that we present these visualizaton results without providing concept explanations.

The first concept our model identifies focuses on jelly shellfish, specifically isolating the contours of these creatures against their natural backgrounds. The second concept turns its attention towards chickens, particularly emphasizing the distinct red comb atop their heads. Following this, the third concept excels in separating the various components of fish species, effectively delineating between different parts such as fins, scales, and tails. Moving to marine life, the fourth concept aims to concentrate on the facial attributes of dolphins and sharks, with a particular focus on

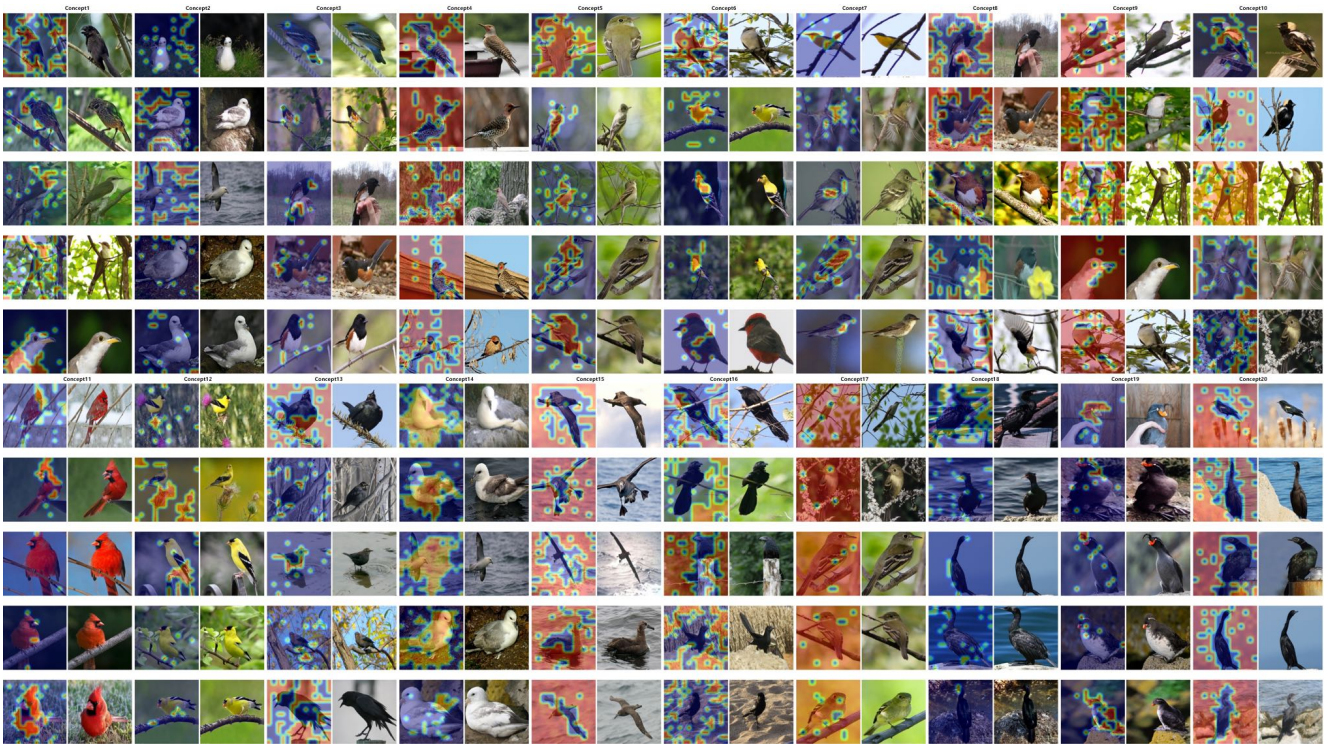

Figure A-6. From left to right, the image displays Concepts 1 through 20 as identified in the CUB-200-2011 dataset following [84]. Each pair of images consists of the masked version (with attention activation mask) on the left and the original image on the right. For better visual representation, see A-7 and A-8.

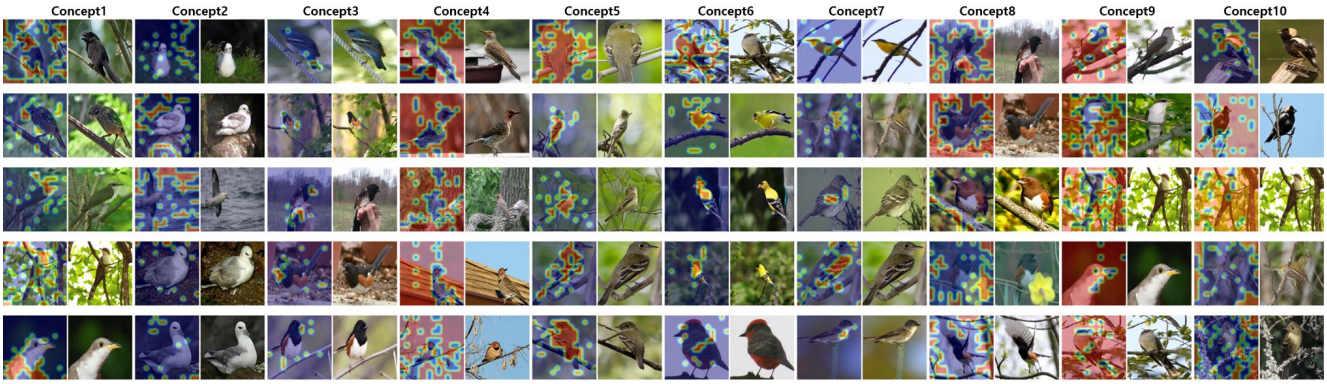

Figure A-7. First 10 concepts (1-10) in CUB-200-2011

their mouths. The fifth concept takes this a step further by specifically highlighting the regions around the oral cavities of sharks, setting them apart from other parts of the creature. In a similar aquatic vein, the sixth concept outlines the unique shapes and contours of goldfish, capturing their form effectively. The seventh concept diverges by focusing solely on the feet of birds, whether they are perched or in flight. This is complemented by the eighth concept, which takes a broader approach to birds by concentrating on their ventral regions, capturing details such as feathers and underbellies. The ninth concept specializes in ostriches, particularly focusing on the distinct features that make up their head region. Finally, the tenth concept zeroes in on hens. It particularly focuses on isolating their contours against a variety of backgrounds, thereby allowing for a clearer understanding of the hen's form and structure.

These visualizations demonstrate CCT's unparalleled ability to focus on semantically meaningful aspects of the images.

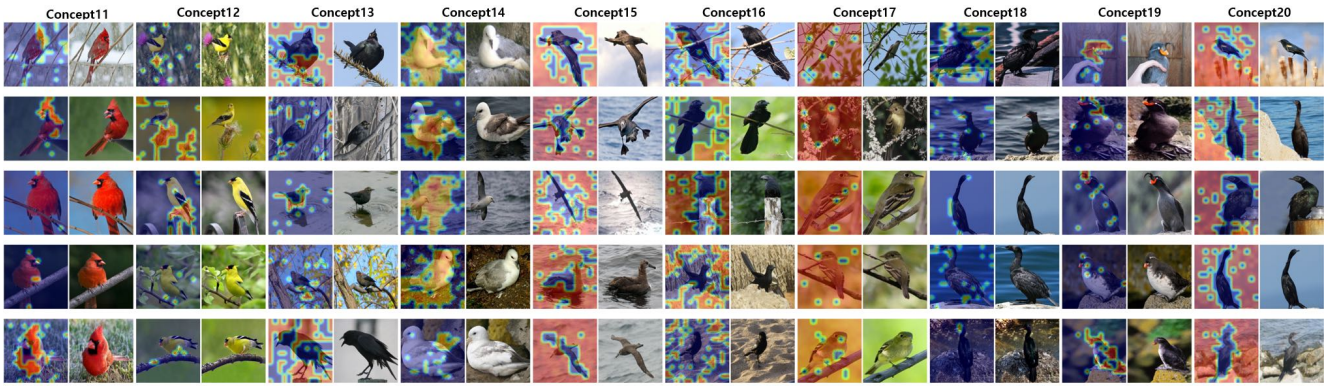

Figure A-8. Second 10 concepts (11-20) in CUB-200-2011

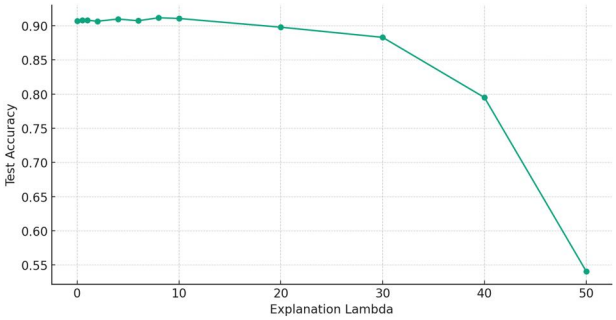

Figure A-9. Performance comparison of CCTs on CUB-200-2011 with different value of explanation lambda λ_{expl}.

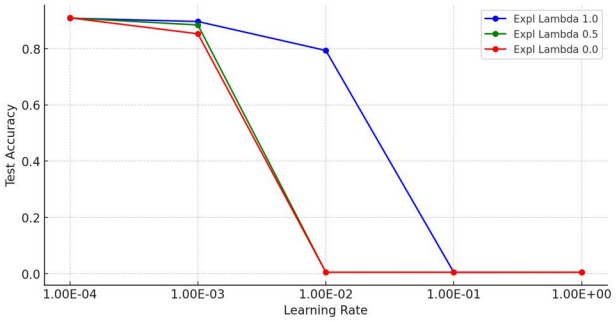

Figure A-10. Performance comparison of CCTs on CUB-200-2011 with different value of learning rates under different explanation lambda values.

Image Retrieval and Clustering. In addition to identifying intricate latent concepts, our CCT model demonstrates exceptional capabilities in image retrieval tasks. As illustrated in Fig. A-11, the model is adept at clustering images that share semantic similarities, thereby reinforcing its utility in generating semantically coherent results. Notably, our CCT achieves this level of clustering without any need for explicit concept explanations. This sets it apart from other models in the field, as it can intuitively group images based on the inherent features recognized through the latent concepts, ranging from the contours of jelly shellfish to the unique features of an ostrich's head. This ability to cluster semantically related images without detailed conceptual guidance emphasizes another aspect where our model excels over others in the domain.

Hyperparameter Settings. We follow the official implementation of CT and produce the experimental results, including Table 3 in the main text.

The only difference in hyperparameter settings between our CCT, CT, and the vanilla ViT-S is the learning rate for the AdamW optimizer. For the vanilla ViT-S, the learning rate is 0.0001. For CT, the learning rate is 5e-5, following that in their CUB-200-2011 experiments. For our CCT, the learning rate is 0.0001. Table A-5d shows the shared hyperparameters for the vanilla ViT-S, CT, and all configurations of our CCT. Please refer to the Table A-6 for hyperparameter settings used in our baseline experiments.

Limitations of Prototype-based Methods. The observation from testing prototype-based models such as ProtoPFormer, ProtoPool, ProtoPNet, and Deform-ProtoPNet on ImageNet yielded some interesting results. While we initially reported outcomes for only the first 200 labels of ImageNet in Table 3, in line with the methodology from [84], it was necessary for us to reduce the number of prototypes per instance—a critical hyperparameter for prototype based model—due to computational constraints, whereas our transformer-based model, CCT, managed the task without issue. We hypothesize that this limitation arises from the inherent design of prototype-based models, which utilize 'prototypes' to offer interpretability to the decision-making processes of complex machine learning models in areas such as image classification, object detection, or segmentation. It can be a trivial issue when the size of dataset is small, however, as the complexity of the dataset increases,

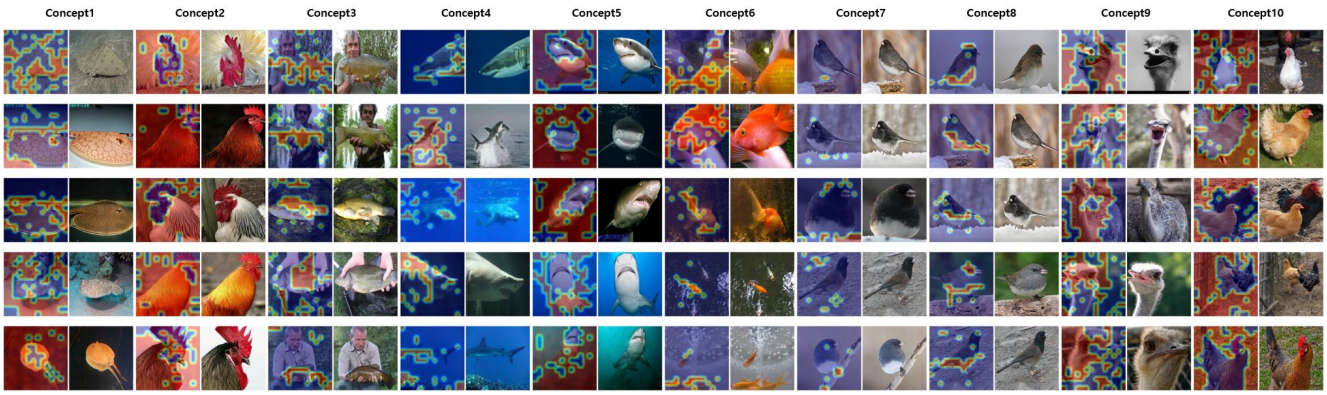

Figure A-11. All 10 concepts (1-10) in ImageNet dataset following [84]

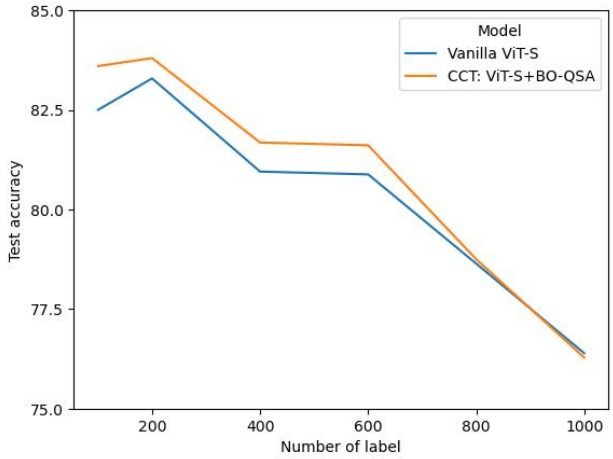

Figure A-12. Performance comparison on ImageNet with different number of labels.

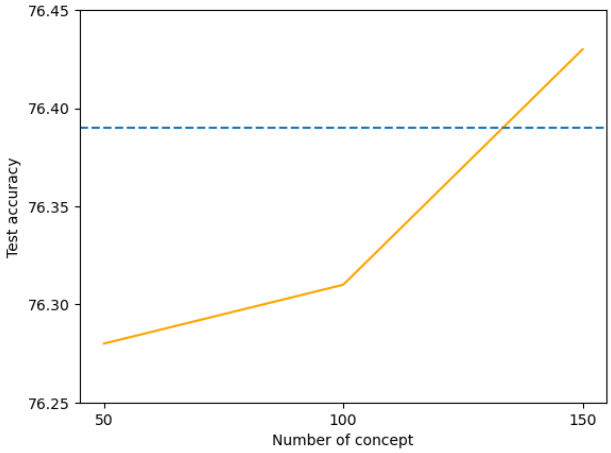

Figure A-13. Performance comparison of CCTs on ImageNet with different number of latent concepts. The dashed blue line indicates the test accuracy of vanilla ViT-S.

such as with ImageNet, which contains millions of images across thousands of categories, the number of prototypes required to effectively represent all classes grows. This growth demands significantly more memory and computational resources, particularly GPU RAM, to store and process these prototypes during both training and inference.

Additional Experimental Results. Fig. A-12 shows the experimental results on ImageNet with different number of labels to train vanilla ViT-S and CCT. We tested both models with the same hyperparameter setting in the main text, and selected the best configuration of CCT, using BO-QSA. Even though the number of latent concepts in CCT is 50 following [84], CCT mostly outperformed the pretrained backbone. When the number of labels is enormous (≥ 800), the performance of CCT is similar to or worse than that of the backbone. This is typical because the number of 50 latent concepts we set is insufficient to process all labels.

Therefore, we performed an additional experiment on ImageNet using a total of 1000 classes and different numbers of latent concepts for CCT. Fig. A-13 demonstrates the performance comparison of CCTs depending on the number of latent concepts. When the number of concepts is set to 150, CCT outperformed the vanilla ViT-S (the dashed blue line), indicating the contribution of setting the number of latent concepts to the performance of CCT.

D. Limitations

In this section, we explain the limitations of our proposed approach.

First, because of the CCT's architectural characteristics in Section 4 in the main text, the proposed approach enforces additive contributions from the user-customized concepts to the classification probabilities. In other words, we ignore the latent higher-order relations among the concepts.

Name	Value
Batch size	64
Epochs	20
Warmup Iters.	10
Learning rate	5e-5
Explanation loss λ	1.0
Weight decay	1e-3
Attention sparsity	0.5

(a) ViT-Tiny, CT and all configurations of CCT on CIFAR100 Super-class

Name	Value
Epochs	60
Learning Rate	1e-4
Number of classes	20
Number of concepts	10
Quantity Bias	0.1
Distinctiveness Bias	0.05
Consistence Bias	0.01
Weak Supervision Bias	0.1

(b) For botCL on CIFAR100 Super-class

Name	Value
Batch size	16
Epochs	50
Warmup Iters.	10
Explanation loss λ	1.0
Weight decay	1e-3
Attention sparsity	0.5

(c) For both CT and all configurations of CCT on CUB-200-2011

Name	Value
Batch size	256
Epochs	10
Warmup Iters.	10
Explanation loss λ	0.
Weight decay	1e-3
Attention sparsity	0.

(d) For both CT and all configurations of CCT on ImageNet

Table A-5. Shared hyperparameters on different datasets.

Name	CIFAR100	ImageNet
Batch size	64	16
Epochs	10	10
Freeze Epochs	10	10
Pre-train Epochs	0	0
Weight decay	0.0	0.0
LR (prototypes weights)	5e-2	5e-2
LR (backbone)	5e-4	5e-4

(a) For PIP-Net on CIFAR100 and ImageNet, LR stands for Learning Rate

Name	CIFAR100	ImageNet
Batch size	64	64
Epochs	50	10
Warmup Learning Rate	1e-4	1e-4
Feature Learning Rate	4e-4	4e-4
Prototype Learning Rate	3e-3	3e-3
Number of Prototype	2000	2000

(b) For ProtoPFormer on CIFAR100 and ImageNet

Name	CIFAR100	ImageNet
Batch size	80	80
Epochs	30	20
Learning Rate	1e-3	1e-3
Gumbel Time	30	30
Number of Classes	20	200
Number of Prototype	202	202
Prototype Depth	256	256

(c) For ProtoPool on CIFAR100 and ImageNet

Name	CIFAR100	ImageNet
Batch size	80	80
Epochs	100	100
Learning Rates	default	default
Number of Classes	20	200
Number of Prototype	100	200

(d) For Deformable-ProtoPNet on CIFAR100 and ImageNet

Name	CIFAR100	ImageNet
Batch size	80	80
Epochs	100	100
Learning Rates	default	default
Number of Classes	20	200
Number of Prototype	1000	2000

(e) For ProtoPNet on CIFAR100 and ImageNet

Table A-6. Shared hyperparameters on different datasets.

For instance, in our CUB-200-2011 experiments, we assumed that global and spatial concepts could be represented and learned parallelly in our framework and that there is no correlation between the global concept and the spatial concept, which is independent. However, this is limited because more complex relationships, such as hierarchical properties, can exist among them. It might be addressed by introducing refined architectural properties in our CCT. For example, the Bi-directional Recurrent Unit can be introduced to allow global and spatial concepts to learn each other's conceptual influences.

Secondly, although, in the experiments of CUB-200-2011 and ImageNet without using explanations, we demon-

strate that our CCT can visualize the learned semantic concepts, we acknowledge that our CCT might be integrated with more sophisticated losses to achieve better visualization of the learned concepts. For example, [90] leverages losses, including reconstruction, contrastive losses, and various regularizers, to enforce the individual consistency and mutual distinctness of concepts. In our future research, we might use those losses and regularizers to allow our model to achieve better visualization capabilities.